The Effect of Visual Design and Information Content on Readers' Assessments of API Reference Topics

Robert Bennett Watson

A dissertation submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Jan H. Spyridakis, Chair

David K. Farkas

Mark P. Haselkorn

Jennifer A. Turns

Program Authorized to Offer Degree:

Human Centered Design & Engineering

University of Washington

Abstract

The Effect of Visual Design and Information Content on

Readers' Assessments of API Reference Topics

Robert Bennett Watson

Chair of the Supervisory Committee:

Dr. Jan H. Spyridakis, PhD

Human Centered Design & Engineering

Software developers must learn and use an increasing number of application-programming interfaces (APIs) to create applications and web sites. To apply these APIs in the increasingly short development time that modern markets require, software developers must learn to use APIs quickly. In the process of learning and using APIs, software developers must find and evaluate many API documentation topics, which can vary greatly in their visual design and the information they contain.

This study applied a task-based, experimental methodology to measure the effects that variations in the visual design and information concepts used in API reference topics have on software developers' speed and accuracy when they assess the topics' relevance, and their perceptions of the topics. In an Internet-based, remote study, participants performed four information-seeking tasks in which they decided whether an API reference topic was relevant to a question presented in a typical programming scenario. The study analyzed 698 individual information-seeking tasks from 201 software developers who lived in 30 different countries and who were proficient in English.

Variations in the visual design elements of API reference topics did not significantly affect the time required to assess the topics' relevance to the information-seeking task yet the variations significantly influenced participants' assessments of the topics' credibility and professional appearance. Variations in the information concepts presented in the API reference topics, on the other hand, significantly influenced both the time participants took to evaluate the topics' relevance and the participants' assessments of the topics' credibility and professional appearance.

This study contributes critically needed empirical data in an under-studied area concerning how document design elements influence readers' performance and perception of API reference topics. This study also updates best-practice recommendations for practitioners who write API reference topics to help them prioritize their documentation efforts. Finally, the study provides information about tools and methods that could provide guidance for testing and improving API documentation and other types of documents and in other contexts.

ACKNOWLEDGEMENTS

This work would not have been possible without the ongoing and understanding support of my family. My wife, Francia, who encouraged my application to the Doctoral program after standing by me through two years of graduate school, has continued her dedication through the five additional years of post-graduate studies and research that resulted from that application. My children who, while watching their father go to school for a sizable part of their lives, managed to graduate from high school and start their own academic journeys. Of course, the journey to this dissertation would have never begun were it not for my father, who encouraged my curiosity from an early age, and my mother, from whom I received the tenacity necessary to see this through to completion.

Dr. Jan Spyridakis and her Internet-Based User-Experience Lab provided the opportunities and resources that contributed to the foundation of this study. As my committee chair, Dr. Spryidakis provided the ongoing and unwavering support and guidance throughout the years that it took to bring this research to a successful conclusion. My committee members, Dr. David Farkas, Dr. Mark Haselkorn, and Dr. Jennifer Turns provided valuable guidance and diverse perspectives throughout the process to help shape and improve the research. The students, faculty, and staff of the Department of Human Centered Design and Engineering who, in addition to being some of the smartest and nicest people you will ever meet, made this possible in more ways that I can list. I cannot thank them enough for all they have done. Finally, I could not have imagined this research, let alone, accomplished it, without the support from my employers, coworkers, and professional acquaintances. I am looking forward to being able to apply this and future research to our mutual and ongoing benefit. Thank you all for your encouragement, support, and guidance throughout this amazing adventure.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Application-programming interfaces (APIs) make it possible for a program or web site to access the data and services provided by another program or web site. While their use was once confined to comparatively few software developers on comparatively few computer systems, their use has exploded in recent years—allowing more systems to be connected by more software developers and to benefit more users than ever before (Abrams, 2008; Cwalina & Abrams, 2008; ProgrammableWeb Research Center, 2014). The design and documentation of these programming interfaces currently exhibit a considerable variety of methods and styles, which is to be expected during the early stages of rapid growth (Watson, Stamnes, Jeannot-Schroeder, & Spyridakis, 2013). While the visual interactions of new mobile and tablet interfaces are being actively researched, the interfaces and interactions of the software components that make them possible are receiving comparatively little attention—academically or professionally.

As the name implies, "user interfaces" are interfaces that "users" use. "Application-programming interfaces" would imply that they are interfaces used in application programming, which they are. However, before they can be used by an application in a programming context, software developers must use them in the process of constructing (coding) the program. Clarke (2005) has shown that the principles of human-computer interfaces can therefore be applied to APIs in much the same way that those principles are applied to other types of human-computer interaction. As with visual or graphical interfaces, APIs can be easy or difficult to use, and their misuse can have negative consequences that range from disappointing to disastrous. Software developers rely on the documentation of these interfaces to use them correctly; however, except for only a handful of papers, API documentation has received very little academic study, and those papers that have been written provide very little guidance that practitioners can apply.

This research extends the current body of knowledge concerning how document design features influence the reader's ability to find information in API reference topics and the reader's perceptions of credibility and professional appearance. This study provides critically needed empirical data in an under-studied area and information about methods that apply to the academic and professional study of API documentation. The methods might also find application in other types of documents and contexts. The research in this study also provides recommendations based on empirically derived data for practitioners who write API reference topics to help them prioritize their efforts and focus on the aspects of documentation that most influence readers. For the end users—the software developers who use the API documentation—the suggestions to improve documentation characteristics, resulting from the study's results, will give them a more effective and productive software development experience. Ultimately, the consumers of the resulting products created by these developers will benefit from better and more reliable software.

APIs make it easier for programmers to develop new software because they make it possible for software developers to use software that has already been written and tested (Watson et al., 2013). While APIs have been used for almost as long as software has been written for computers, recent years have witnessed an explosion of APIs as more information and services use them to provide access to their services over the Internet (ProgrammableWeb Research Center, 2014). The widespread adoption of service-oriented architecture design patterns and standardized protocols such as the Simple Object Access Protocol (SOAP), Representational

State Transfer (REST), Extensible Markup Language (XML), and JavaScript Object Notation (JSON) has facilitated much of this explosion. Businesses are taking advantage of the additional channels that these technologies offer to provide additional value to their customers. APIs enable companies to extend their reach by making their valuable, and often unique, data available to other apps. Software developers are then able to use APIs to combine information from previously separate and disconnected sources to provide additional value to their customers. Today an app or web site can use APIs from many different sources, which allows them, for example, to access shopping information from an online store, shipping information from a delivery company, and ratings and recommendation information from other sources. Apps can also use APIs to enable people who use the app to share their experiences on a social-media site—all from within a single app or web site.

Software developers today work in the intersection of these technologies and business requirements under the pressure of short development times, which make it vital for them to stay current and to access relevant and changing information as efficiently as possible. API documentation is one tool that software developers use to accomplish this. The development demands of an innovative software application can include several, if not many, different APIs. As software developers move from one project to the next and as the technology, they employ on a single project changes over time, they must constantly learn how to use new APIs by consulting a variety of different information sources. These sources vary and can include co-workers, formal and informal online references, and hard-copy publications (Nykaza et al., 2002). In cases where the technology is very new or the documentation is poor, software developers might even need to create their own documentation through a process known as *reverse engineering* (Samuelson & Scotchmer, 2002). Reverse engineering can be a time-consuming process that involves a series of tests and experiments, which are designed to deduce or infer the functionality of an API. Reverse engineering is also a practice that the APIs' terms of use frequently prohibit (Samuelson & Scotchmer, 2002). However, the pressure to finish the job and deliver the app, even in the absence of the necessary documentation, can compel software developers to risk violating the terms of use in order to complete their task (Samuelson & Scotchmer, 2002).

Research Overview

While APIs have been used in computer programming for decades (Henning, 2007), the importance of APIs and API documentation has received comparatively little research attention in the fields of computer science and technical communication. As such, the literature on the subject is scarce and predominantly qualitative, and many best practices are anecdotal and frequently untested. Recent studies (Parnin & Treude, 2011; Watson et al., 2013; Maalej & Robillard, 2013) have attempted more quantitative approaches to the subject; however, the dearth of literature and research in this field presents both challenges and opportunities for the current research. The literature in this review shows how this study draws from related fields such as reading, learning, and cognitive load theory in order to provide a theoretical framework for the experiment conducted in the study. At the same time, this study brings together aspects of document design and information seeking in the context of API reference documentation in a novel approach that will benefit future researchers and practitioners.

This dissertation focuses on the fundamental elements of an information resource that software developers use briefly, yet frequently, in the course of their work: the API reference topic. Software developers are required to constantly learn and apply new technologies throughout their career (Robillard, 2009; Robillard & DeLine, 2011). The learning tasks they must accomplish span a spectrum from learning precise details about how to apply a programming element correctly to learning new programming languages and design patterns. This spectrum of ongoing learning presents a challenge to software developers who must continue to learn new technologies and techniques and to the researchers who wish to study this learning process (Robillard & DeLine, 2011).

The broad spectrum of learning tasks and the diversity of learning methods and artifacts require any study in this area to first identify and specify what, exactly, is and is not the object of study. The task that this study focuses on is at the more detailed end of the spectrum of learning scenarios that software developers encounter—making decisions about the relevance of an API reference topic to answer a question they might have about an element of an API. Watson et al. (2013) observed a wide variety of styles and formats in the API reference topics they studied. Because software developers learn from many sources and types of documentation, it is important to specify and clarify the documentation being studied for this research. For this study, an API reference topic is *an online document that provides information about a single, specific programming function that another program can access*.

Software developers frequently access API reference topics to locate a fact about a programming element to answer a specific, often task-oriented question (Stylos & Myers, 2005). In the course of these fact-finding tasks, software developers can encounter many possible documents. To search efficiently, software developers must decide, as quickly as possible, whether a document contains the information they seek. While the information-seeking scenario that this study examines is very narrow and specific, its significance to the academic and professional communities derives from its frequency and its cumulative impact on the overall software development process that results from this frequency. The information-seeking scenario examined in this study can occur with great frequency during the software developer's primary task of writing software. If these information-seeking tasks require too much attention, they can seriously impede the productivity of a software developer's primary task. Analyzing what influences these interruptions can provide insight into how they might be shortened.

The scenario of this study can be summarized simply: *while performing a software development task, a software developer requires a specific fact about a programming element in an API to continue and looks for that fact in an API reference topic about the programming element*. The literature review and method sections describe the scenario that this study examines in detail. While searching in the context of this scenario, software developers might encounter many documents that might provide the information they seek. In any given document, software developers want to assess whether the topic contains the information they seek as quickly as possible so they can read that document further or abandon it and continue their search (Stylos & Myers, 2005).

This task scenario and context for this study include three aspects:

- The software developers
- The information-seeking task and context
- The API reference documentation used in the information seeking tasks

The next sections introduce and discuss each of these aspects.

Software Developers

Clarke (2007) described three developer personas, each having a different learning style.

- *Systematic* developers, who "develop a deep understanding of the technology before using it"
- *Pragmatic* developers, who "develop a sufficient understanding of a technology to enable them to use it"
- *Opportunistic* developers, who "develop a sufficient understanding of a technology to understand how it can solve a business problem"

In practice, these developer personas do not describe a person as much as they describe three different ways a person (even the same person) might approach a programming task, depending on the person and the context of the programming task (Clarke, 2007). These personas also describe three different ways that a software developer can learn about an API. The following sections describe how a software developer learns when acting as each persona.

- As a **Systematic Developer**, a software developer will review concepts and architecture documentation to understand the system as a whole and review the individual programming features (e.g., object, classes, methods, and interfaces) to understand how pieces of the system work individually and together before they start using them.
- As a **Pragmatic Developer**, a software developer will learn enough to start a task and then refer to the documentation and other information resources to solve problems as they encounter them (Clarke, 2003; Clarke, 2007).

- As an **Opportunistic Developer**, a software developer will look for ways to solve their business problem quickly, ideally reusing the code from an example or tutorial as much as possible (Brandt, Guo, Lewenstein, & Klemmer, 2008; Brandt, Dontcheva, Weskamp, & Klemmer, 2010).

Each type of software developer (that is, *every* software developer) refers to an API reference topic to research a fact or detail at some point in their software development experience (Brandt, Guo, Lewenstein, Dontcheva, & Klemmer, 2009). In the case of the Opportunistic Developer persona, information gathering might be so frequent that it is integral to and indistinguishable from the primary task of developing software and writing the program code. On the other hand, Systematic Developers research the API thoroughly before starting the programming task so that researching additional API details while programming can be seen as interrupting their primary task. The distraction caused by a research task that interrupts their focus for more than a short period can cause them to lose several minutes of productive work due to the loss of flow (DeMarco & Lister, 2013). DeMarco and Lister suggest that a state of increased productivity (*flow*) can take up to 15 minutes to achieve, yet be broken with a very short interruption (they describe "an announcement over a public address system" as being sufficient to break a flow state). The influence this interruption effect has on a software developer's flow makes the importance of the contribution of improving the ease of locating information in an API reference topic much greater than just that of the incremental time an improvement might save. In every case, the time required to research information in an API reference topic contributes to slowing down or interrupting, to varying degrees, the primary task of software development and programming.

Information-Seeking Task and Context

In the context of the different information sources that software developers might consult and all the reasons they might be motivated to consult them, this research focuses only on formal reference documentation that describes a single function or method of an API in the context of a search with a single, specific, task-oriented question in mind. This scenario is a common form of information gathering employed by software developers (Brandt et al., 2008). It is important to recognize that software developers refer to API documentation to accomplish many tasks and learning goals. Further, the documentation to which software developers refer to accomplish these different tasks is quite diverse (Watson et al., 2013). From informal sources such as question-and-answer forums to formal documentation that is provided with an API, software developers can search and interact with many different sources and types of information to answer a single question, hence, the need to clarify this specific task context and documentation type.

API Reference Documentation

The variety of page design and information content and detail found in APIs was described in a recent study of API documentation (Watson et al., 2013). In the context of a multi-document search, diverse documentation styles could complicate the learning process. Literature on information seeking suggests that documentation design and the information content influence search and evaluation tasks in an information-seeking context (Redish, 2012; McGovern, 2006). Because there is no current literature on how documentation design and information content affect users of API reference topics, the findings of Watson et al. (2013) prompt the research question of this investigation.

The documentation factors evaluated in this study are the number (count) of visual design elements used in an API reference topic and the number (count) of unique information concepts found in an API reference topic. Basic web design principles suggest that extremes of either factor are likely to have a detrimental effect on the reader’s experience with a topic (Redish, 2012). However, this study does not set out to prove the effect of unrealistically extreme cases—this study does not intend to assess the effect of design patterns that are known to frustrate or confuse readers. Instead, this study seeks to assess the effects within a range that, while quite diverse, is drawn from samples of API documentation that a software developer could encounter in actual practice, as observed in the open-source software documentation that Watson et al. (2013) encountered.

Research Question

What is the effect of visual-design element count (VDEC), unique information-concept element count (ICEC), and topic relevance to the information-seeking task versus ICEC on the speed and accuracy of software developers’ assessment of an API reference topic’s suitability to answer a task-oriented question, and on their perceptions of the topic?

BACKGROUND AND LITERATURE REVIEW

The research question asks about API reference topic assessment during the information-seeking tasks that software developers perform in the specific context of their work. Qualitative research that has studied software developers in situ describes the context of the information-seeking task and the nature of the information-seeker (the software developer). The theoretical frameworks of adult-learning theory, multiple-document reading theory, and cognitive-load theory describe the information seeking process that software developers use in this context. For these aspects of the research question, that is, at a high level, there is adequate literature and very little disagreement. Going into the question further, the literature and agreement are scarcer. What literature is available on documentation design and information content in the context of API reference topics guided the experiment that was conducted as a remote, unmoderated, experimental study. A summary of the literature that guided the design and analysis of the study conducted for this dissertation follows.

Information-Seeking Tasks of Software Developers

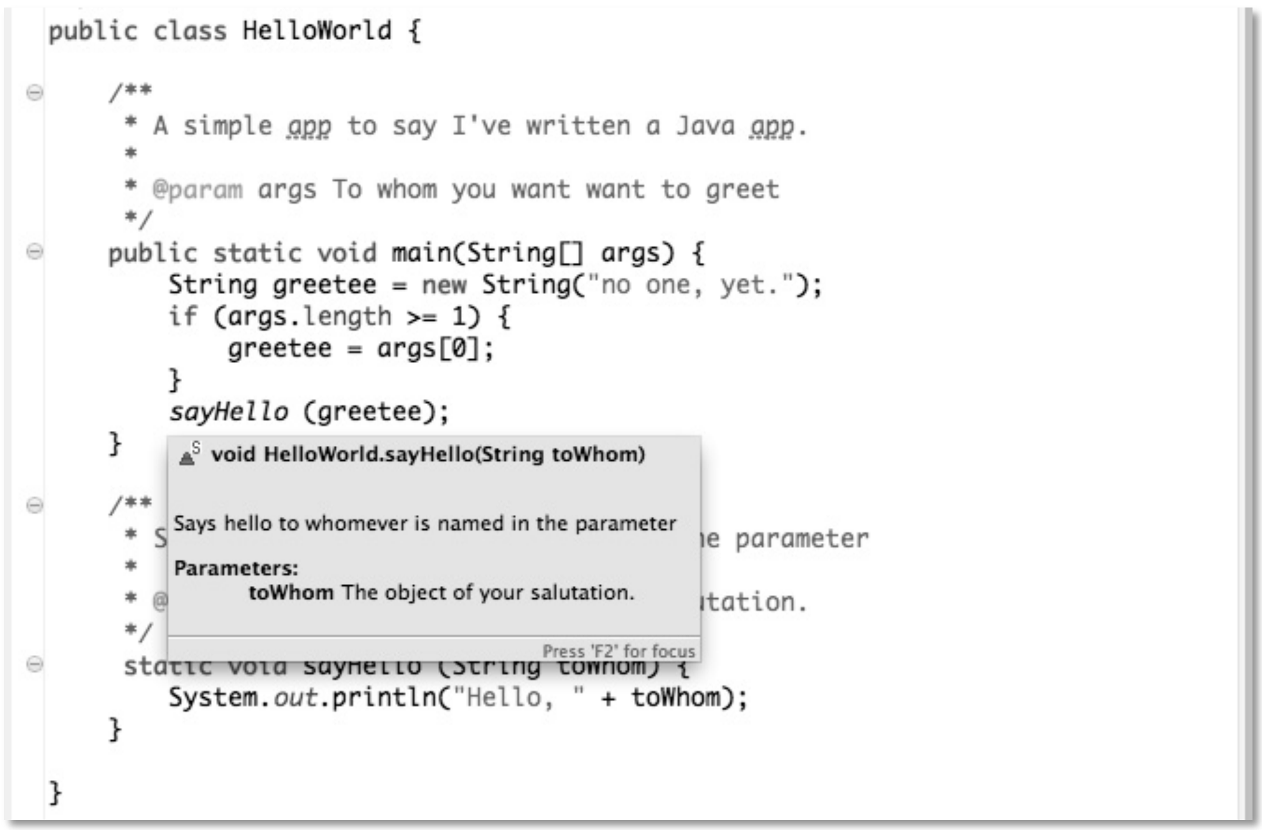
This section describes the target population of software developers and the context in which they perform the information-seeking tasks used in this study.

Software Developer Personas

Clarke (2007) identified three personas for software developers: *systematic*, *pragmatic*, and *opportunistic*—each of which embodies a different approach to programming. An individual software developer, however, can embody any one of these personas, depending on the person and the immediate situation. This study concentrates on the “just-in-time” or *incremental* type of learning style that is most commonly demonstrated by the opportunistic software developer persona, and points out that the pragmatic and systematic software developer personas apply when necessary (Clarke, 2003). Software developers who develop software opportunistically typically proceed through a task until they encounter a problem for which they lack the knowledge to continue (Clarke, 2007). At that point, they seek the specific information that will unblock them so they can continue, a process hereafter called *do-question-learn-continue*. The do-question-learn-continue process differs from other learning tasks in that it assumes that the software developers feel confident enough to start a project or task even though they might need to look up some specific details to complete it. In this way, the learning context differs from other learning scenarios, such as those in which the reader has a goal, but lacks sufficient knowledge to start a task.

Software Developer Task Context

The do-question-learn-continue process described in the previous section can occur at many levels of complexity and is similar to the "seek, relate, collect, [and apply]" process that Ko, Myers, Coblenz, and Aung (2006) observed when they watched software developers learn about software while debugging and modifying existing software. At the simplest level, this process could last only a few seconds and be satisfied by a pop-up text that provides a short, in-context help reference. The example in Figure 1 shows a source-code editor displaying a pop-up window that provides a supplemental description of the *sayHello* method. On many occasions, this in-context help is sufficient to answer software developers' basic questions about how to use the method in the immediate context. If such a pop-up cannot answer software developers' questions, the next, more detailed unit of documentation available is an API reference topic, such as this study evaluated.



```
public class HelloWorld {

    /**
     * A simple app to say I've written a Java app.
     *
     * @param args To whom you want want to greet
     */
    public static void main(String[] args) {
        String greetee = new String("no one, yet.");
        if (args.length >= 1) {
            greetee = args[0];
        }
        sayHello (greetee);
    }

    /**
     * S                                      e parameter
     *
     * @                                      tation.
     */
     static void sayHello (String toWhom) {
        System.out.println("Hello, " + toWhom);
    }

}
```

```
void HelloWorld.sayHello(String toWhom)

Says hello to whomever is named in the parameter
Parameters:
        toWhom The object of your salutation.

                                    Press 'F2' for focus
```

Figure 1. Image of pop-up help in Eclipse, a source code editor

Software developers often ask complex questions that require longer interruptions to resolve—long enough to create a new learning task that precedes or supersedes the programming task. At the extreme, for example, a software developer might need to stop or delay starting a project to attend a class or training session on a new technology before continuing. API documentation contains many different types of content that can be used at many points along this spectrum of information-seeking cases. Unfortunately, the spectrum of possible tasks has not been formally mapped, nor has the range of document types. In spite of the absence of a formal taxonomy, a range of information-seeking tasks has been observed by Stylos and Myers (2005), Ko et al. (2006), and Robillard and DeLine (2011) and examples of API documentation that range in size from the one-line pop-up described earlier to multiple-volume documentation sets are common and can be found easily. Of all the possible software development tasks and scenarios in which a software developer could consult the documentation, this study focuses on only a specific scenario. This study examines tasks that involve questions that can be answered by consulting an API reference topic, thus requiring more information than simple, pop-up reminders provide. Further, this study focuses on only the decision process that software developers' apply in order to determine if a topic is relevant to their immediate information-seeking task.

Adult Learning Theory

Adult-learning theory (andragogy) describes the context in which an adult is motivated to learn and the conditions under which that learning takes place. Included in this theory are the teaching aspects that the documentation must contribute to facilitate the reader's learning.

Learning

Knowles, Swanson, and Holton (2011) describe adult learning with their andragogical model, which they differentiate from a pedagogical model. The key elements they list in their andragogical model comprise:

1. The need to know
2. The learners' self-concept
3. The role of the learners' experiences
4. Readiness to learn
5. Orientation to learning
6. Motivation

These points reflect a strongly constructivist learning style and summarize key assumptions about adult learners described by Lindeman (1989).

Knowles et al. (2011) describe the adult learner's self-concept as being ready to learn in the sense that the learners are mature enough to learn on their own and have a sense of self-direction. Relating this to the specific scenario of reading API reference documentation, the reading scenario assumes that readers have a need to learn the information they seek.

Andragogy requires that readers are ready to learn and that their experiences are sufficient for them to learn what they need to know from the documents they encounter. In the context of just-in-time information seeking, the context of this study, the documentation serves as a proxy for a human teacher. As such, learners should be able to self-assess their level of experience vis-à-vis the document they are reading. The learners should be able to identify any experiences that might conflict with what they are reading, as well. This study establishes task scenarios to provide the context and assumes that the participants' readiness to learn, their orientation to learning, and their motivation to learn are sufficient to provide a constructive learning experience within this framework.

Teaching

The description of andragogy focuses on the learner's readiness and motivation to learn. Knowles et al. (2011) describe these factors as being key differences between andragogy and pedagogy. Because the points in the preceding section on learning focus on the learner, they are, for the most part, independent of the document and document design this study evaluates. However, even self-directed learners require some guidance and external instruction, which, in this scenario, come from the documentation. In actual practice, this type of learning rarely occurs in such a vacuum of resources. Nykaza et al. (2002), Robillard and DeLine (2011), and Watson et al. (2013), for example, describe many social aspects of software development and learning in this context. While the information-seeking process of software developers in this context has a social element, the specific interaction of software developers reading and assessing an individual API reference topic takes place with little, if any, social interaction, even if the actions that precede and succeed the event have a social component.

The learning theory elements mentioned in the previous section describe a constructivist view of learning for the reader/learner; however, the task-oriented nature of the problem they are trying to solve describes a behaviorist view of assessment. The constructivist view of learning (Piaget, 1970) takes into account the learner's prior experience, while the behaviorist's view of assessment (Skinner, 1954) is concerned with how what they learned manifests itself as a change in the learner's behavior. Adult learners want to build on their existing knowledge and see a change in behavior—demonstrated in this scenario by successfully applying their learning to the software problem that motivated the desire to learn in the first place.

These theories suggest that when documentation is used as a proxy for an instructor or facilitator in learning in this context, it must exhibit the following traits:

- **Build on the learner's knowledge**—the documentation must provide the means by which readers can assess whether they have the requisite background to understand the topic and provide the readers with the means to obtain the requisite knowledge to solve the problem at hand.
- **Provide learners with the ability to obtain the feedback they need to assess their learning**—for example, the documentation should provide a clear indication of the

behavior to expect (in the program, generally) when the readers understand the API enough to answer their question or solve their problem.

Learning Theory and API Documentation

In the context of this study, and within the context of a software developer using API documentation while programming, the software developer is a self-directed learner (Knowles et al., 2011) and the documentation is the only teaching resource being considered. The study assumes that the software developers' desire to complete a programming task provides their motivation and readiness to learn from the documentation. The study assumes that the software developers have the requisite experience and knowledge to be able to identify what they need to know in order to understand the information that they find. In that sense, the knowledge they seek in the case of an API reference topic builds, if only incrementally, on their existing knowledge. The nature of the programming task provides feedback that the software developers learned or did not learn what they needed to know in the form of a functioning or non-functioning program or program module.

Multiple-Document Reading Theory

Multiple-document reading theory describes the process that people use to identify the need for information, search for information, determine the relevance of the information they find, understand the information, apply that information to the task, and, in many cases, commit the information to long-term memory. While this study reviews the scenario of individual readers who assess the relevance of a single document in a task-oriented search context, that scenario is repeated in the larger scenario that Multiple-Document Reading Theory describes.

Multiple-Document Reading Theory Principles

Rouet (2006) describes the process of assimilating information from multiple complex documents in his Task-based Relevance Assessment and Content Extraction (TRACE) model of document processing, shown in Figure 2, which reviews the steps used to find and apply information resources and memory resources to complete a task.

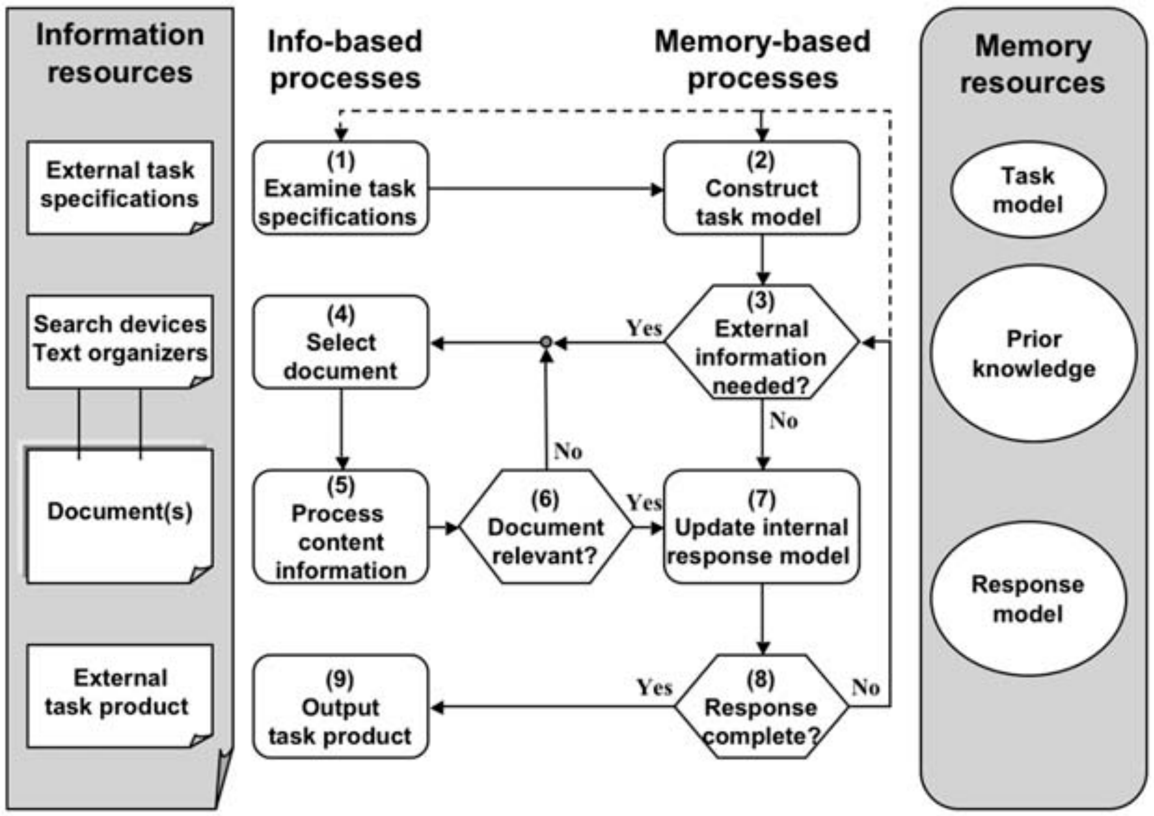


Figure 2. The Task-based Relevance Assessment and Content Extraction (TRACE Model) of document processing (Rouet, 2006)

In applying the TRACE Model to API document use, the *information resources* include the API reference documents evaluated in this study and the *memory resources* include the reader's prior and recently acquired knowledge. The TRACE Model describes the iterative process that the readers apply to identify, locate, and assimilate information from multiple documents.

When evaluating the steps of the TRACE model, the document's visual design and information content influence some steps more than others. For example, steps 1, 2, and 3 of the TRACE model are cognitive processes that readers perform independent of any information resources. Once readers decide that they need external information in step 3, the document's visual design and information content come into play. The document's visual design and information content can have a material influence in how the reader performs steps 4, 5, and 6 of the TRACE Model, discussed in the next section. Finally, steps 7, 8, and 9 of the TRACE Model are cognitive processes performed by readers and are independent of any information resources.

Relevance Theory

The element of the TRACE theory being most closely examined in this study is the relevance decision, indicated as step (6) in Figure 2. Sperber and Wilson (1986) describe relevance as a perception that is dependent upon the context and affected by both context and cognitive effort. Contextually, Sperber and Wilson (1986) describe relevance as a continuous property wherein an assumption "is relevant in a context to the extent that its contextual effects in this context are large." Therefore, if an assumption or, more generally, the object or statement being considered, has many aspects in common with the context in which it is being considered, it is highly relevant to the context. Cognitively, Sperber and Wilson (1986) continue, describing relevance as "the effort required to process it [the aspects of the assumption] in this context." An assumption, or the object, being considered is highly relevant to a context when it requires little cognitive effort to determine the aspects that it has in common with the context.

In a later study, Barry and Schamber (1998) evaluated how users made relevance decisions in information-seeking experiments. They also described relevance judgement as a "cognitive and dynamic process that involves all of the knowledge and perceptions that the user brings to the information problem situation"—a view and context reflected in in Rouet's (2006) TRACE model. Consistent with Sperber and Wilson's more general properties above, Barry and Schamber (1998) describe relevance as being:

1. "Cognitive and subjective, depending on the users' knowledge and perceptions;"
2. "Situational, relating to users' information problems;"
3. "Complex and multidimensional, influenced by many factors;"
4. "Dynamic, constantly changing over time; and yet,"
5. "Systematic, observable and measurable at a single point in time."

Multiple-Document Reading Theory and API Reference Documentation

Stepping out of the relevance evaluation and back into the broader context, Rouet's TRACE Model (2006) illustrates how readers find and assimilate information from multiple complex documents—each document containing a component of the information that the reader seeks—and integrate those documents to construct the knowledge they seek. Rouet studied how people read newspapers and other documents to accomplish their information seeking; however, his TRACE model is very similar to the model that Ko, Myers, Coblenz, and Aung (2006) used in their study. It is also very similar to the model that Stylos and Myers (2006) used as they looked more closely at the cognitive processes applied by software developers when they sought information about APIs.

Steps 4, 5, and 6 of the TRACE Model describe the aspects of information seeking that provide the context for this study—where the reader must find, assess, and decide if a document is relevant to their information-seeking task. Many elements of a document can influence the assessment process. Navigation elements and hyperlinks, for example, can aid or complicate the decision-making depending on the context of the elements vis-à-vis the reader (DeStefano & LeFevre, 2007). Text elements such as headers and introductory text can also influence decision-making in these steps (Redish, 2012). Visual-design elements or the nature of the information content in the document could prevent the software developer from finding the relevant content on a page (Stylos & Myers, 2005). Ultimately, however, it is the document's information for which the reader is searching and the reader must determine if a document is relevant to this information-building process. This study examines step 6 of Rouet's TRACE model through the lens of Relevance Theory to observe whether document design and information content affect the reader's assessment of an API reference topic.

Cognitive Load Theory

Pollock, Chandler, and Sweller (2002) define cognitive load theory as using "some aspects of human cognitive architecture and of the structure of information to provide instructional designs that facilitate understanding, learning, and problem solving." The multiple-document reading theory presented in the preceding section describes a largely cognitive process by this definition.

Assumptions of Cognitive Load Theory

In their theory, Pollock et al. (2002) assume the following:

1. *A limited working memory that can process only a few elements of current information at any given time*

2. *An effectively unlimited long-term memory holding knowledge that can be used to overcome the limitations of working memory*
3. *Schemas held in long-term memory and used to structure knowledge by organising* [sic] *elements of information comprising lower order schemas into higher order schemas that require less working memory capacity; and*
4. *Automation that allows schemas to be processed automatically rather than consciously in working memory, thus reducing working memory load.*

Within those assumptions, Pollock et al. identify these two sources of cognitive load.

1. **Intrinsic cognitive load** – The cognitive load the task-at-hand requires
2. **Extraneous cognitive load** – The cognitive load of tasks that require attention but are not related to the task-at-hand

Bannert (2002) talks about the **germane cognitive load**, which "occurs when free working memory capacity is used for deeper construction and automation of schemata." Valcke (2002) divides that further by separating the germane cognitive load into the **meta-cognitive workload** from the **schema-processing workload**.

Cognitive Load Theory and API Reference Documentation

From the perspective of document design, the design goals of API reference topics with regard to facilitating access to the topic's information are to:

- Minimize the reader's *extraneous* cognitive load.
- Make the most of the *intrinsic* cognitive load.
- Facilitate the *germane* processing load.

The design of API reference documentation can support these goals in the context of the preceding assumptions made by Pollock et al. (2002) in the following ways.

- **Limited working memory**

  In the context of API documentation, a software developer's limited working memory can come from several sources. The most likely reason for a limited working memory in the context of this scenario is that the software developer's primary task is the programming task in which they formed the question that brought them to the documentation (Brandt, Guo, Lewenstein, Dontcheva, & Klemmer, 2009). Consequently, this primary task occupied most of a software developer's working memory before they came to the documentation. API documentation, therefore, should require as little of the reader's working memory as possible. If one assumes that working memory is a limited resource, any working memory required to interpret the documentation will reduce the working memory that was being applied to the problem that brought them to the documentation in the first place. At some point, the information search could require enough of a software developer's working memory that the software developer experiences what DeMarco and Lister (2013) describe as a loss of flow. Once the flow is lost, it can take much longer than the interruption to restore the working memory the software developer had before the interruption.

- **Unlimited long-term memory**

  An unlimited long-term memory assumes that the reader can remember what they learned in the past and use that information in their current analysis. While the theory assumes the reader can remember anything they learned in the past, the reader is still limited as to how much of that knowledge they can bring into their working memory at any single point in time. Brandt et al. (2009), however, say that the increasing volume of information to retain and the ubiquity of Internet-based access to that information have caused many software developers to delegate portions of their long-term memory to Internet-based resources.

- **Schemas**

  Schemas help the reader organize the information they learned in the past so they can efficiently recall it (Mobrand, Cuddihy, Galore, & Spyridakis, 2007). They also help readers assimilate and understand new information. API reference documentation that matches an existing schema held by the reader can be understood more quickly than documentation that requires the reader to alter an existing schema or construct a new one (see also Automation, below). It is important to note that a document could require the reader to modify an existing schema or construct a new schema for several reasons—for example, if it introduces a new topic to the reader and causes him or her to modify an existing schema or construct a new one.

- **Automation**

  Fortunately, existing schemas can be processed automatically so schema updating can take place with very little additional cognitive load. However, when the reader applies a schema inappropriately, this automatic processing will misinterpret the information without the reader's realization (Rouet, 2006). In this case, the germane cognitive load increases as the reader constructs the correct schema.

Primary Task and Secondary Tasks

To explain how people deal with complex problems—problems that are too large to be processed within the available cognitive capacity—Sweller (1988) describes a "goal stack," detailing how people organize sub-goals and process them sequentially. The idea of a goal stack describes the nature of API reference topic access—where a software developer is performing the primary task of writing software and the need to search for information, the software developer creates a new task (the information search) that supersedes the primary task of writing software. The software developer must manage a goal stack onto which the temporary goal of the information search replaces the primary goal of writing software as the goal to which they must attend. These elements of cognitive theory describe the individual cognitive processes that occur during the evaluation, assessment, and knowledge-construction elements described in the section on multiple-document reading theory.

The increasing complexity of software development makes it difficult to keep all the requisite information in the software developer's working memory, which makes temporary interruptions to research detailed information inevitable while programming. The complexity of software development and the ease that Internet-based resources can be accessed makes information-seeking tasks a necessary part of modern software development (Brandt et al., 2010). That being the case, there is evidence that secondary tasks can increase the time to complete the primary task as well as increase the errors, anxiety, and annoyance felt (Bailey, Konstan, & Carlis, 2000). Bailey, Adamczyk, Chang, and Chilson (2006) found "that the disruptive effect of interruptions on completion time tends to increase with the difficulty of the primary task." In the context of the difficult tasks that comprise software development, reducing the effect of the secondary task of information seeking should improve the software developer's experience.

Document Design Literature

The design literature applied to this study includes generally accepted design principles (aesthetic), web-based documentation-design principles, and the effect of a document's visual design and the document's information content on the reader's assessment of a document. The literature about this aspect of the study is weak, sparse, and somewhat conflicting; however, this section describes some of the studies that have measured the effect of these factors on reader performance and document assessment and describes where the study is based on the best interpretation of the available literature.

Documentation Design Aesthetic

The design aesthetic of a document describes how well the visual design of a document applies the generally accepted principles of visual design. At a fundamental level, Lidwell, Holden, and Butler (2010) describe a set of basic design patterns. Tidwell (2011) describes design patterns in the context of user interfaces, for example, those that support rapid access to the information on the page. Reynolds (2008) summarizes the principles of visual design as contrast, repetition, alignment, and proximity and describes how they help present information more clearly. These are just a few of the generally accepted best practices that professional designers use.

Robins and Holmes (2008) tested the effect of design aesthetic on assessment speed and web site credibility judgments. They found that the pages with a high aesthetic were not judged any more quickly than those with a low aesthetic treatment; however, they were judged by participants to be more credible. They operationalized the aesthetic treatment as:

- **Low**: “the content is simply placed on a web site without professional graphic design”
- **High**: “presents a professional look-and-feel appropriate to the organization it represents”

These findings agree with the findings of Tractinsky, Katz, and Ikar (2000); however, Tractinsky (2004) and van Schaik and Ling (2009) note the complexity of studying the effect of a document’s aesthetic on usability. For example, van Schaik and Ling describe how the context in which an information-seeking task takes place influences the participants’ aesthetic judgments of a page’s attractiveness. Tractinsky (2004) explains that beauty alone is not sufficient, as the title of Tractinsky, Katz, and Ikar (2000), *What is Beautiful is Usable,* would seem to suggest.

In their study, van Schaik and Ling (2009) observed that the context affected the participants’ time required to assess a document’s aesthetic. When software developers use API reference topics, one contextual aspect that they encounter when reviewing an individual topic is whether the topic is relevant to their search. To test for a context-related effect on task performance, the current study varied the relevance of the question (that sets the search context for the participant) to the topic.

Web-Based Documentation Aesthetic

API reference topics have existed for as long as there have been API elements to document. Their content has changed very little in the past 40 years (if not longer). In some cases, UNIX *man pages*[1], for example, have not changed at all. In spite of all that history, there is very little literature on the format. While web content has been around for a much shorter period, it has accumulated a much larger body of knowledge and literature. Given the shortage of specific documentation, technical writers who write the API reference topics have adapted general web content guidelines to the de-facto standard format that API reference topics have followed over the years. The topics reviewed in Watson et al. (2013) illustrate the diversity of these interpretations and adaptations.

To study API reference topics in a proper context, one must understand the foundations on which they have been produced. This section describes these foundations: general web-content design best practices, current industry-accepted best practices, API documentation elements.

General Web-Design Guidelines and Best Practices

Two of the most common web-design guidelines align with elements of the research question: visual design and information content (Redish, 2012; McGovern, 2006). The visual design of a page must make the content easy to access and the information content must be useful to the reader. The guidelines also suggest that the reader's context is important to their interaction with the content and an important consideration when designing the topics.

Redish (2012) lists these specific guidelines for web content:

1. Remember that readers skim and scan the content.

[1] A *man page* (an abbreviation of manual page) is a reference topic, originally created for the UNIX operating system. A man page is generally a simple text file, displayed in only a single font and minimal visual design— usually only paragraphs and indented text. Figure A-15 shows a topic variation from this study that is very similar in style and content to a UNIX man page.

2. Focus on what the site visitor (reader) wants to do.
3. Answer content and design questions together.
4. Consider the visual design's color, space, and typography.
5. Layer information and present the most important information first.

McGovern (2006) focuses on the content in the context of the reader's task. A recurring theme in McGovern (2006) with regard to content is where "less is more"—focusing on the readers' goal: the topic should be easy to find in the content set and the desired information should be easy to find within the topic. Carroll (1998) is well known for his work on content minimalism, which is frequently (and incompletely) summarized, as "less is more." To apply the guidelines recommended by Redish, McGovern, and Carroll most effectively, the writer must know and understand the readers' goals—something that McGovern, Redish, and Carroll (1998) and many others encourage. A caveat to the "less is more" approach to content is that when the readers' goal is not clearly understood, less can simply be less when the content does not contain the information that the reader seeks. McGovern and Redish both mention that one approach to have both less content and more content is through progressive exposure or discovery.

Technical Content Best Practices

While McGovern and Redish (and many others) describe general guidelines and best practices for general web content and impress the importance of context, literature on best practices for the specific context of API documentation is comparatively scarce. The *Microsoft Manual of Style for Technical Publications (MSTP), 3rd ed.* (2004), and the later *Microsoft Manual of Style, 4th ed.* (2014), provide guidance specific to technical content and provide standards for language, typographic conventions, and common terminology in the context of technical context—

considerations mentioned by Redish (2012). These references, however, do not go into much detail about API reference topics.

Watson et al. (2013) summarized these documentation elements as being helpful or critical to learning an API.

- Overview documentation
- Short code “snippets” that demonstrate usage of an API in context
- Code examples that show best practices with an API
- Scenario and task-based documentation
- Limitations and error handling
- Meaningful documentation (as opposed to “filler” or “boilerplate” content that adds little or no value to what is obvious)

They also listed these aspects and expectations of technical documentation that reflect the content goals and expectations described by Redish (2012) and McGovern (2006).

- Accuracy, completeness, and correctness
- Scenario and task-focused examples
- Content that does not repeat the obvious, such as what can be learned from the user interface

The context studied by Watson et al. (2013) was an entire API documentation set, and as such, not all elements apply, or apply equally, to API reference topics—a specific subset of API documentation. For example, scenario and task-based documentation is critical to learning an API, but is rarely found in an API reference topic, while short code snippets are more common.

API Documentation Elements

The documentation elements listed in the previous section describe types of information more than specific topic or document types. API documentation consists of many different types of individual documents or topics. Mihaly (2011) groups API documentation into five types, which contain the elements listed in the previous section to varying degrees:

- **Reference Manual**

  Content that explains the details of API elements. API Reference Topics are an example of the content found in the API's Reference Manual.

- **Cookbook**

  A collection of recipes that use the API in common user scenarios, which can be applied directly or be easily adapted for use in a user's context or application.

- **Programmer's Guide**

  Examples of the features and benefits the API provides. Similar to a cookbook, but more conceptual in nature.

- **Code Example or Tutorial**

  Demonstrative examples of using the API in specific use cases. The use cases can vary from specific programming tasks to more general applications and broader use cases.

- **FAQ or Knowledge Base**

  A collection of questions and answers produced by the API developer, the user community, or both. This type of API documentation is generally produced on demand and after the API has been distributed to software developers to use.

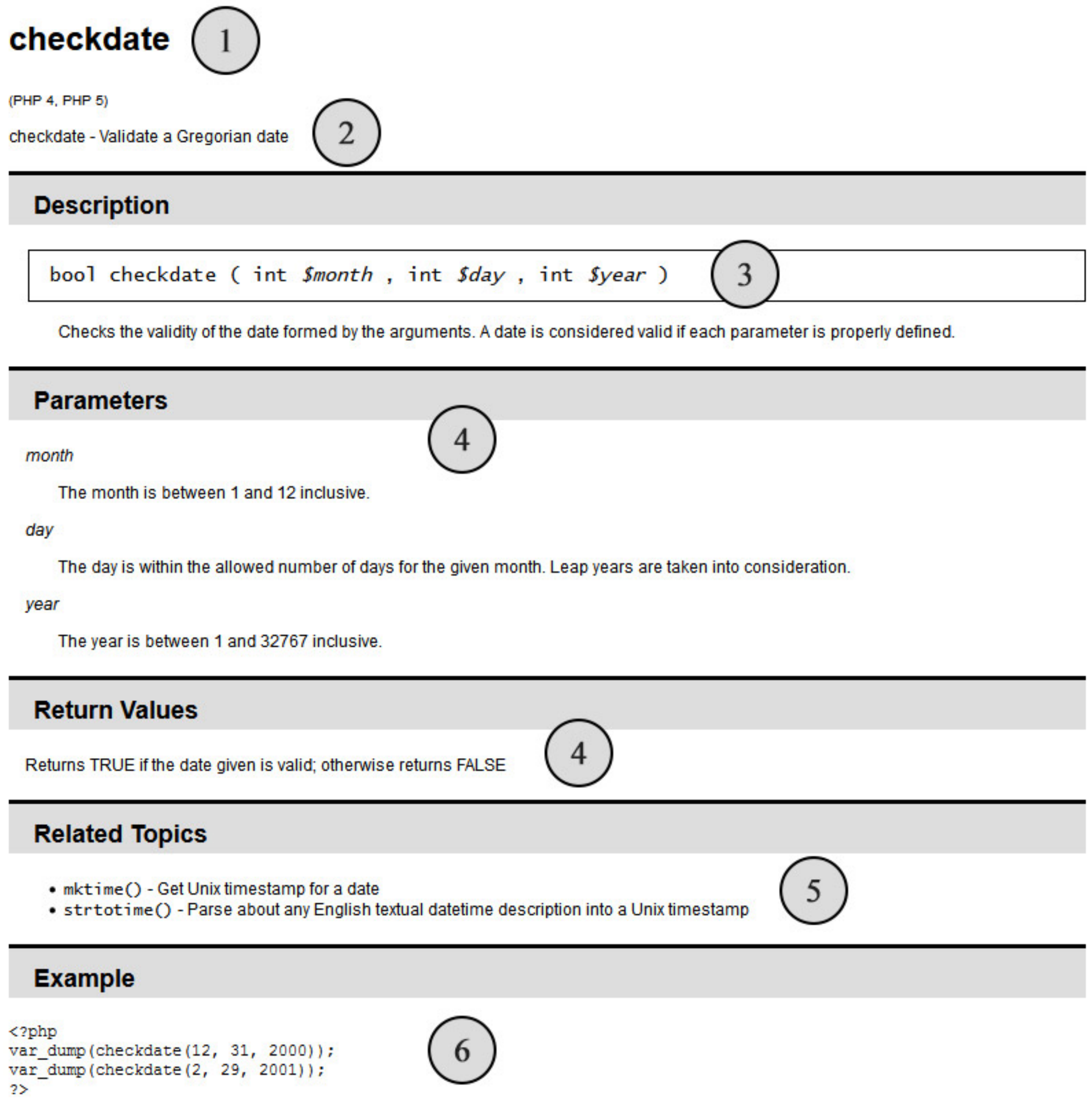
These API documentation types are general categories and not all API documentation includes every type in Mihaly's list. This study, however, concentrates on API reference topics—the topics found in the API's Reference Manual.

API reference topics consist of many common elements (Watson et al., 2013). Figure 3 shows an example of an API reference topic with some of the key sections identified. As Watson et al. (2013) observed, most of the API reference topics they studied had all of these elements. For this study, however, the elements in the documentation topic variations were as consistent as possible.

The key elements of an API reference topic listed in Figure 3 are:

1. Page title, topic title, or API name
2. Summary description
3. Syntax
4. Parameter definitions and return values
5. Remarks and related topics
6. Code example

As with API documentation in general, not every API reference topic that Watson et al. (2013) studied included every element in the preceding list. However, according to the literature, to be a useful and recognizable API reference topic, a topic must include at least elements 1-4 from the preceding list or it provides little to no value to its intended audience of software developers (Nykaza et al., 2002; Robillard & DeLine, 2011; Watson et al., 2013). Elements 5 and 6 from the preceding list can provide additional information to make an API element's function clearer and easier to use; however, for simple API elements, they can be redundant and provide no information that is not already provided by the other API reference topic elements.


checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure 3. API reference topic example with annotated sections

API Documentation Designs Observed in Practice

Recent studies of API documentation found a wide variety of API documentation design and content styles in use (Watson, 2012; Watson et al., 2013). Other recent studies have reported how API documentation is lacking and needs improvement (Robillard, 2009; Robillard & DeLine, 2011; Parnin, 2013).

A frequent complaint identified in these studies is how software developers have trouble finding the content they seek—either because the information exists, but cannot be found, or because it does not exist. In the API reference documentation sets they studied, Watson et al. (2013) found the key API reference topic elements were present in most of the documentation they studied. However, they also found that the documentation had a wide range of design and writing quality, which made it difficult to determine when the documentation contained the information and when it did not. In Watson et al. (2013), the researchers categorized the variations of the API documentation from 33 of the most popular open-source software libraries according to design and writing quality dimensions, which correspond to the visual design aesthetic and information content utility dimensions used in this study.

Sampling differences can account for some of the differences between the studies that have reported how API documentation is lacking and needs improvement (Robillard, 2009; Robillard & DeLine, 2011; Parnin, 2013) and Watson et al. (2013) who found that most API documentation had most of the elements sought in API documentation. The studies that were based on interviews and qualitative research focused on finding problems and so problems were reported. One of the questions from the survey given to participants in Robillard and DeLine (2011), for example, asked, *"What obstacles made it difficult for you to learn the API?"* Parnin

(2013) reported that social media and other informal API documentation were used for reference more than official documentation. The data from these studies were very task-specific and highlight the cases in which API documentation is deficient without identifying where and how often the API documentation satisfied the task. To be complete, it is important to understand when the API documentation works and when it does not. At the same time, it is important to remember that these two situations are different, and potentially independent, aspects of a large and complex environment.

The variety of visual design and information concepts observed in the API documentation reviewed in Watson (2012) and Watson et al. (2013) would create additional cognitive load in software developers who had to integrate the information from any combination of those API documentation sets. This additional cognitive load could affect the time it takes a reader to find and assess a document to decide if it contains the information that they seek. In today's software development environment, using multiple APIs in an application or web site is much more common than in the past, which makes it more important to understand the impact of these variations on software developers' cognitive load during information-seeking tasks.

Document Testing Methodologies

This section describes the literature on which the document testing methods used in this study are based.

Content Analysis Methods

Krippendorff (2012) defines content analysis as "*a research technique for making replicable and valid inferences from texts…to the contexts of their use.*" Krippendorff (2012) refines this by identifying three variations of content analysis. Analysis that takes content:

1. “to be contained in a text.”
2. “to be a property of the source of the text.”
3. “to emerge in the process of a research analyzing a text relative to a particular control.”

Krippendorff (2012) reiterates the importance of studying texts in context saying, “*texts acquire significance (meanings, contents, symbolic qualities, and interpretations) in the contexts of their use.”* This research looks at the content that is contained in a text, following the first variation, and in a specific context. While this study is not one of content analysis, per se, it applies some content analysis concepts and methods. The key elements of content analysis applied in this study are: (1) coding of textual content and (2) analysis of texts in a specific context.

In this study, the texts are API reference topics, which varied the number of information concepts they contained, so it was necessary to quantify the variation. The field of content analysis also requires the identification and coding of textual elements and so its principles were applied to this requirement. In this study, the coding unit of “unique information concept” was used to quantify the information contained in a text. The “information concept” used in this study was one complete fact about the documentation topic. This study operationalizes an information concept as:

- A complete concept that describes an aspect of the object of the documentation.
- A statement that can be understood on its own.

In most cases, a single information concept is no longer than a sentence. In the case of a program or software example in the documentation, however, an information concept can consist of the software necessary to demonstrate a complete function in the context of the topic; as such, a software example might require several lines to illustrate a complete concept that can stand on its own, as the second point requires. In this study, the quality of the information concept is not considered because the samples have been edited to ensure that the quality of the information is consistent and sufficient for the purpose of the document. In traditional content analysis, variations in the nature of the content are the object of study. In this study, variations in the nature and quality of the content have been minimized, while the quantity of information concepts is an independent variable in the experiment. To that end, the critical element of this aspect is that the information concepts can be operationalized and counted reliably.

These are some coding examples taken from the sample API reference topic are shown in Figure 3. Section 1 of Figure 3, contains the title, which is not an information concept. Although it provides valuable information to the reader, it is not a complete concept, but just the name of the object and the title of the topic. Section 2 contains two elements: the first one is not an information concept for the same reason the title is not. The second element, however, is a complete information concept—it describes what the feature does. Section 3 contains three information concepts: the syntax block (within the box) and the two sentences below it are each information concepts. Section 4 contains five information concepts: the descriptions for the *month* and *year* parameters have one each, the description of the *day* parameter has two, and the description of the return value is the fifth one. Section 5 contains two, and the code example in section 6 makes up the last information concept of the topic. Taken together, the topic in Figure 3 contains 12 unique information concepts.

Information Content Concept Testing

Several studies have measured the effect of information content on reader performance. Redish, Felker, and Rose (1981) compared the readers' speed and accuracy between two different versions of a Federal Communications Commission (FCC) regulation. In their study, Redish et al. (1981) took a generically formatted regulation, rewrote it, and formatted it for the target audience. They then tested how fast and accurately readers could answer questions using each version. Their results showed that applying the best practices of document design and information content—principles that provide a clear presentation of information—improved the readers' speed and accuracy. In the course of their study, Redish et al. (1981) attempted to use objective readability scores as a predictor of performance, but they noted that these scores did not correlate with the reader performance they observed. Instead, they used a participant rating to assess the topic's difficulty.

While not a study of API documentation, Redish et al. (1981) show that the visual design aesthetic and information content can influence the efficiency of a reader's access to information. They also describe methods that could be adapted to this study to measure reader performance when varying design aesthetic and information content such as using scenario-based questions and timing readers as they try to find the answers in the documentation.

With regard to information content, it is reasonable to imagine that there could be a point where changing the content of a document would bring about no corresponding improvement in reader performance. Duffy, Curran, and Sass (1983) observed such an effect in their study. They took an existing topic from a military technical manual and hired three experienced technical-writing contractors to improve it. Each contractor had many years of experience writing technical manuals but each one approached the task from a different perspective. The resulting versions varied in visual design aesthetic and presentation of the content, while keeping the information presented in the topic as consistent as possible to accomplish to goal of the manual. In their study, Duffy et al. observed no statistically significant difference in the reader's information access speed or accuracy between the different versions of the technical manual. They concluded that after a point, it might not be possible to improve these aspects of reader performance.

Although Duffy et al. did not see any significant difference in reader performance between their document versions, Watson et al. (2013) observed a wider variety of visual design aesthetic and information content utility in the API documentation they reviewed than the document variations Duffy et al. had in their study. It is therefore reasonable to expect a wider, potentially significant difference in reader performance when testing documentation variations similar to those Watson et al. (2013) observed than those Duffy et al. (1983) observed.

For measurement methods, both Redish et al. (1981) and Duffy et al. (1983) had participants test different versions of documentation to answer task-oriented questions while they timed the participants' responses. Duffy et al. (1983) and van Schaik and Link (2009) had participants rate qualities of the documents they were reviewing—a method that was adapted for use in the current study. Because van Schaik and Ling found web-site assessments to vary in response to the search context, this study varied the relevance of the topic to assess whether the context would also influence the speed and accuracy of software developers evaluating API reference topics.

Experimental Methods

The experiment designed to collect data for this study is a quantitative, remote, unmoderated, Internet-based study that has within-subject and between-subject variations. This section reviews the literature that influences the study's experimental design and analysis plan.

To collect a data set from software developers in their own environment, this study used an Internet-based remote user-assessment tool. Bartell and Spyridakis (2012) summarize best practices for Internet-based user research and data collection that guide the experiment design for this study. Some of the practices from Bartell and Spyridakis applied to this study include:

1. Mitigating measurement error
    a. Designing the study with simple and straightforward navigation
    b. Applying the principles of good question design described in Dillman (2008)
2. Mitigating drop-out error
    a. Applying the "high-hurdle" questioning technique to encourage early dropouts so that those participants who continue are more likely to continue through to the end
    b. Designing the study so that it can be completed quickly
3. Mitigating reliability error
    a. Using survey techniques that minimize the potential for repeat or duplicate submissions such as a low-value gratuity—enough to encourage participation but not so much that it encourages multiple participation
    b. Tracking the Internet protocol (IP) address of the participant to identify potential duplicate submissions

4. Ensuring participant rights
    a. Providing a clear statement of confidentiality and consent
    b. Maintaining gratuity information separate from the study data
5. Pilot testing questions before launching the actual study

Spyridakis et al. (2005) and Dillman (2008) describe additional best practices for web-based questionnaire design and formatting, specifically noting that the questions in the questionnaires should be written, reviewed, pilot-tested, and revised before their use in the study.

Literature Review Summary

Software developers frequently need to locate and apply information that they often find in API reference documentation as a secondary task in the course of their primary task of software development. Adult-learning theory frames the macro setting of the information-seeking task and the software developers' learning model in this scenario. Multiple-document-reading theory frames the specific processes used when software developers search for information among multiple documents in this scenario. Cognitive-load theory describes the software developers' cognitive processes involved in the context of both the primary task of software development and the task of information seeking when it is a part of the primary task and when it is a secondary task. Relevance theory describes the nature of the specific decision made in this context. Whether the secondary task of information seeking interrupts or is integral to a software developer's primary cognitive task of software development, it should require as little time and cognitive load as possible. Information seeking performance in general, and determining document or topic relevance in particular, are critical to the software developer's primary task performance and success.

The literature shows that the visual design and information concepts in a document can influence a reader’s performance and perceptions of a document. The design variations seen in Watson et al. (2013) and the mixed results described in the literature motivate the research question of whether these variations influence the time it takes to determine a document’s relevance.

RESEARCH QUESTION AND HYPOTHESES

This study seeks to answer the following research question and related hypotheses.

Research Question

> *What is the effect of visual-design element count (VDEC), unique information-concept element count (ICEC), and topic relevance to the information-seeking task versus ICEC on the speed and accuracy of software developers' assessment of an API reference topic's suitability to answer a task-oriented question, and on their perceptions of the topic?*

Hypotheses

Because the scenarios presented in each of the tasks are different and the topics are different, the mean relevance-decision response time for each task could be different. If the mean relevance-decision response time for each task is significantly different, the following hypotheses will be evaluated separately by task. However, if there is no significant difference between the mean relevance-decision response times of each task, the tasks will be evaluated together to provide a larger sample size per variation.

- H1. Software developers will have a shorter mean relevance-decision response time in topic variations with high vs. low VDEC levels.
- H2. Software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low VDEC levels.
- H3. Software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels.

- H4. Software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels.
- H5. When API reference topic variations have the same relevance to the task question, software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels.
- H6. When API reference topic variations have the same relevance to the task question, software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels.

The following hypothesis describes the expected effects of VDEC and ICEC on perception and credibility.

- H7. Software developers will give higher credibility ratings to API reference topic variations with high vs. low VDEC levels.

H1, H3, and H5 related to *the speed* that a software developer makes a selection and speak to the different influences on the cognitive load of evaluating the relevance of an API reference topic. A software developer would take more time to evaluate the relevance of a topic that requires a higher cognitive load than one that required a lower cognitive load. H2, H4, and H6 related to *the accuracy* of a software developer's assessment. In the context of this study, *accuracy* was operationalized as whether participants' relevance-assessments matched the scenario's designed relevance to the API reference topic used in the task. Hypothesis H7 tested the findings of van Schaik and Link (2009) to evaluate the impact that visual design had on the credibility of API documentation.

While not a part of the study's research question, the study offers an opportunity to collect additional data about related, yet still exploratory, questions. Participants interacted with the survey tool using a spot interaction after each task's topic to specify the most influential location in the topic and provided the opportunity to explore a novel method for collecting a participant's response to a document-related question in a remote, unmoderated study. Questions were asked in the questionnaires that followed each task to collect perceptual data required to test Hypothesis H7, to validate the experimental variations, and to collect additional information about the API reference topics.

METHODS

Task-based, experimental methods were used to assess the objective aspects of the study and confirm or refute the hypotheses. Perceptual data were collected to evaluate participants' decisions and perceptions. This section describes the experiment steps used to test the hypotheses. The **Experiment Design Overview** section describes the test protocol in detail and the literature that supports the method applied. The **Independent Variables** section describes the manipulated variables, their levels, and the literature that supports the variations. The **Participant Perception Measures** section describes the perceptual measures assessed by the participants. The **Participant Selection** section describes how participants were recruited and selected for the study.

Experiment Design Overview

A quantitative study that consisted of an experiment that had within- and between-subjects variables was conducted to collect objective performance data and gather perceptual assessments from participants.

Study Terms

The study method involves understanding four terms: the *study*, the *survey tool*, the *task*, and the *topic*.

Study

The *study* included the planning, design, execution, and data collection and analysis discussed in this dissertation.

Survey Tool

The *survey tool* was the web-based application that collected data from participants in the study. The survey tool implemented the study protocol as an online survey that consisted of four tasks. The survey tool also randomized and delivered the tasks and the topic variations displayed to participants.

Topic

A *topic* in this study is an API reference topic that participants assessed during a participant sessions presented by the survey tool. To measure the effects of the different independent variable levels, the combinations of the independent variable levels manipulated by the survey produced four variations of each topic.

Task

A *task* was an element of the online study protocol as implemented by the survey tool and consisted of the following steps, which are described in more detail in the **Online Study Protocol** section.

1. **Scenario**

   The scenario described a programming context and presented a question within that context for participants to research using the API reference topic for the task.

2. **API reference topic**

   After participants read the scenario, the survey tool showed them an API reference topic to use in the scenario. The topic shown to participants was one of the four possible variations of the topic.

3. **Variations (topic)**

   The topic variation was a complete API reference topic that used the VDEC level and the ICEC level selected by the survey tool.

4. **Questionnaire**

   The questionnaire was a page shown to participants after they assessed the relevance of API reference topic that the survey tool presented to them and asked participants several questions about the topic.

Study Protocol

Using an Internet-based, remote user-assessment tool, the experiment presented participants with four tasks. After reading the task's scenario, participants reviewed an API reference topic to assess its relevance by deciding whether it had the information required by the scenario. The survey tool timed and recorded their responses. After participants assessed the relevance of the API reference topic, they were asked several questions about it in a one-page questionnaire. The objective performance measures were evaluated for a significant effect using within- and between-subjects ANOVAs. The perceptual data were evaluated for significant effects by using Chi-Square and Mann-Whitney tests.

The independent variables that the survey tool manipulated for the study were derived from the API reference topic variations observed in Watson et al. (2013) and refined by expert review. Watson et al. (2013) studied 33 sets of API reference documentation and found the variations in design quality and writing quality shown in Table 1.

Table 1. Number of document design and writing quality variations observed in Watson et al. (2013)

		Writing quality		
		Low (n=6)	High (n=27)	**Pearson Chi-Square**
Design quality	Low (n=13)	5	8	χ^2 (1, N = 33) = 5.93, *p* = .015
	High (n=20)	1	19	

The researchers determined the categories and their levels that Watson et al. (2013) used and, while those categories adequately characterized the documentation reviewed by the raters, the categories lacked sufficient operationalization for use in this experiment. Students and faculty in the Department of Human Centered Design & Engineering at the University of Washington reviewed several variations of these categories to improve their suitability for experimental variation and the categories described by Watson et al. (2013) were operationalized for this study as described next.

The design quality category from Watson et al. (2013) was operationalized for this study as the count of unique visual-design elements, or the VDEC (visual design element count). To maintain ecological validity, the topic variations were designed to be consistent with industry and professional best practice as observed in the documentation reviewed by Watson et al. (2013). In the context of API reference topic visual design, however, Watson et al. (2013) found that a wide variety of visual design is considered consistent with industry and professional best practice. The VDEC levels are operationalized further in the next section.

The writing quality category from Watson et al. (2013) was operationalized for this study as the count of information-concept elements, or ICEC (information concept element count). To maintain ecological validity, the topic variations were designed to be consistent with industry and professional best practice as observed in the documentation reviewed by Watson et al. (2013). The ICEC levels are operationalized further in the next section.

Independent Variables

The independent variables manipulated in the experiment varied the API reference topics' visual design and information concepts to produce four different variations of each API reference topic studied. Each topic used in the study was derived from the API documentation sets reviewed in Watson et al. (2013). Original API reference topics were edited to reflect the different levels of the experiment's independent variables; however, the resulting variations conformed to the current industry-standard best practices that apply to API reference documentation and had a consistent level of information quality. The independent variables and the levels that were manipulated in the API reference topics used in the study were:

- **VDEC** (Visual-Design Element Count)
    - High VDEC
    - Low VDEC
- **ICEC** (Information-Concept Element Count)
    - High ICEC
    - Low ICEC
- **Topic Relevance** to the scenario
    - Relevant
    - Non-Relevant

The following sections describe these variables and levels.

VDEC (Visual-Design Element Count)

The VDEC is the number of unique visual design elements used in a topic. A unique visual design element consists of a set of visual-design characteristics, which were defined as styles in a Cascading Style Sheet in a .css file. The appearance of each text element in the HTML pages that made up the API reference topics used in the study was defined by the CSS style assigned to it, along with the styles the element inherited. A single visual-design element included such properties as the font family, font size, font weight, font color, background color, and border style. The complete listing of styles used by each level of this independent variable is found in the **Study Style Sheets** section in the Appendix. For the purpose of this study, duplicate styles—CSS styles whose attributes are shared by another CSS style on the page—are counted only once. Table 2 lists the number of uniqe CSS styles in each level of this variable.

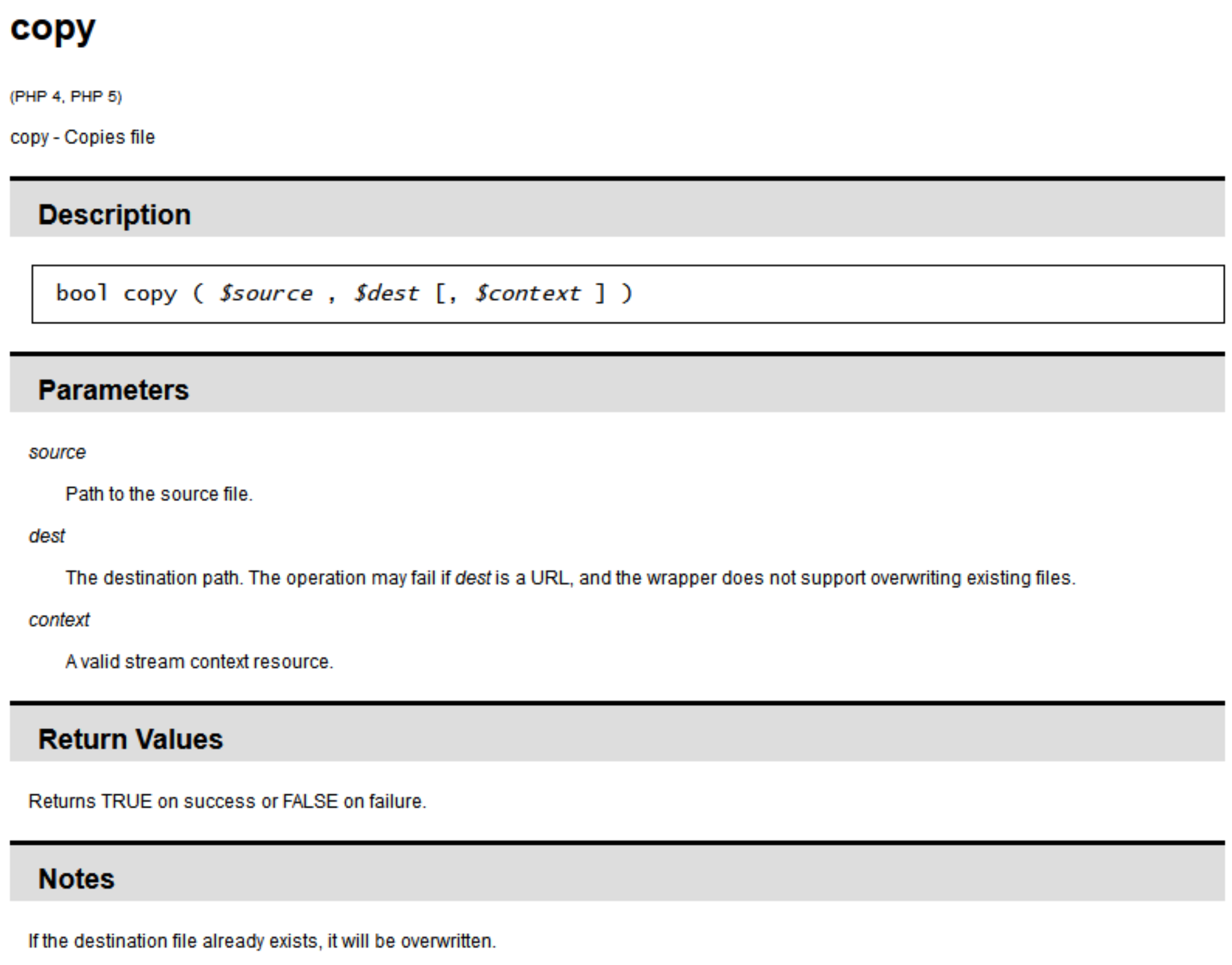
Table 2. Visual-Design Element Count levels

Variable level	**Unique Visual-Design Element Count (VDEC)**
High	12
Low	5

Two .css files were developed to produce the two levels of this variable for the survey tool. The following sections describe the .css files used in this study and the contents of the files are found in the **Study Style Sheets** section in the Appendix.

High VDEC (Visual-Design Element Count)

The .css file used by API reference topics with a high VDEC included 12 unique CSS styles. Figure A-33 in the Appendix lists the .css file used in the experiment. Because the CSS style definitions can inherit properties from other styles, the definitions in a .css file do not necessarily list all the attributes of the style. Table A-1 and Table A-2 in the Appendix, however, list the characteristics of each design element described in the hd.css file (Figure A-33 in the Appendix), even though the style definitions in Figure A-33 might not specify every design element. Figure 4 is an example of an API reference topic with a High VDEC. Topics with a High VDEC level have headers with shading and borders and variations in typography and spacing to help readability and ease of skimming.


copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( $source , $dest [, $context ] )
```

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if *dest* is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

Returns TRUE on success or FALSE on failure.

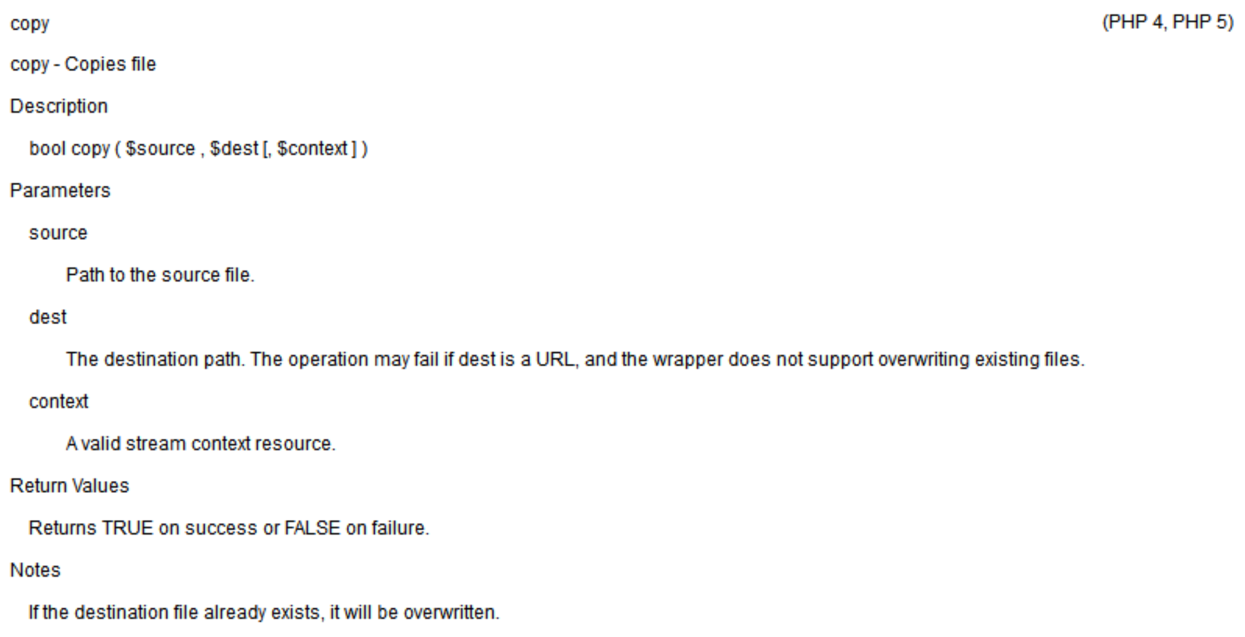
Notes

If the destination file already exists, it will be overwritten.

Figure 4. Example of API reference topic with a high VDEC

Low Visual-Design Element Count

The .css file used by API reference topics with a low VDEC included five unique CSS styles. Figure A-34 in the Appendix lists the .css file used in the experiment. Because the CSS style definitions can inherit properties from other styles, the definitions in a .css file do not necessarily list all the attributes of the style. Table A-3 and Table A-4 in the Appendix list the characteristics of each design element described in the ld.css file. Figure 5 shows an example of an API reference topic with a Low Visual-Design Element Count (VDEC). Topics with this level of VDEC have do not use graphical elements and do not vary the typography, but use only spacing to help readability and ease of skimming.



copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure 5. Example of API reference topic with a low VDEC

ICEC (Information-Concept Element Count)

An Information-Concept Element is a complete unit of conceptual or factual information. To count as a concept element, the text must describe a complete behavior, property, or action. For example, “Copy – Copies a file” would be a concept element. “Return values,” however, would not, because it does not describe a concept. The ICEC of a topic variation is the number of unique concept elements that a topic has—repeated concept elements are counted only once.

Each API reference topic in this study had two sets of information—representing each level of this variable such that one set had more information-concept elements than the other did. Both levels of ICEC reflected examples of API reference topics found in Watson et al. (2013).

ICEC had these two levels:

- **High** – the topic has a relatively high number of information-concept elements.
- **Low** – the topic has a relatively low number if information-concept elements.

The actual number of information-concept elements varied between the topics to ensure that each topic’s content conformed to current best practices for API reference topics. Table 3 lists the ICEC for each task and ICEC level.

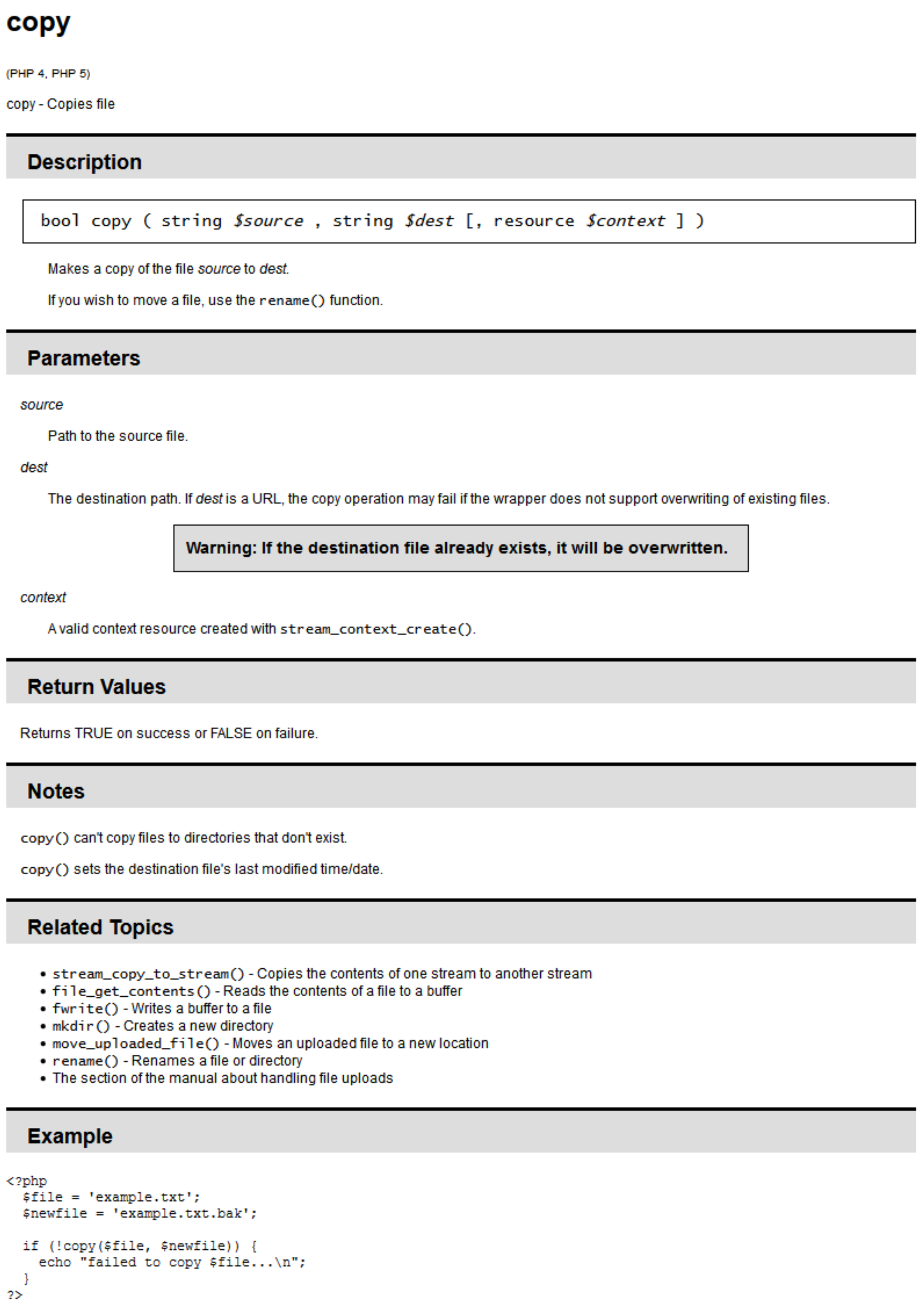
Table 3. Information-Concept Element Count (ICEC) of topics used in the study

Task	Topic	ICEC of High level variation	ICEC of Low level variation	Ratio
1	copy	20	8	2.50:1
2	print	19	7	2.71:1
3	select	26	12	2.17:1
4	join	15	7	2.14:1

Figure 6 and Figure 7 show examples of API reference topics with high and low ICEC levels. The topic in Figure 6, for example, contains more document sections and more information in each section than Figure 7. Both levels, however, are consistent with examples of documentation identified in Watson (2013). Table 4 lists the word count of each topic by ICEC level.

Table 4. Word count of topics by ICEC level

Task	**Topic**	**High ICEC**	**Low ICEC**	**Ratio**
1	copy	206	76	2.71:1
2	print	247	54	4.57:1
3	select	386	170	2.27:1
4	join	188	61	3.08:1
Average		**256.8**	**90.3**	



copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( string $source , string $dest [, resource $context ] )
```

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the `rename()` function.

Parameters

source

Path to the source file.

dest

The destination path. If *dest* is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with `stream_context_create()`.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

`copy()` can't copy files to directories that don't exist.

`copy()` sets the destination file's last modified time/date.

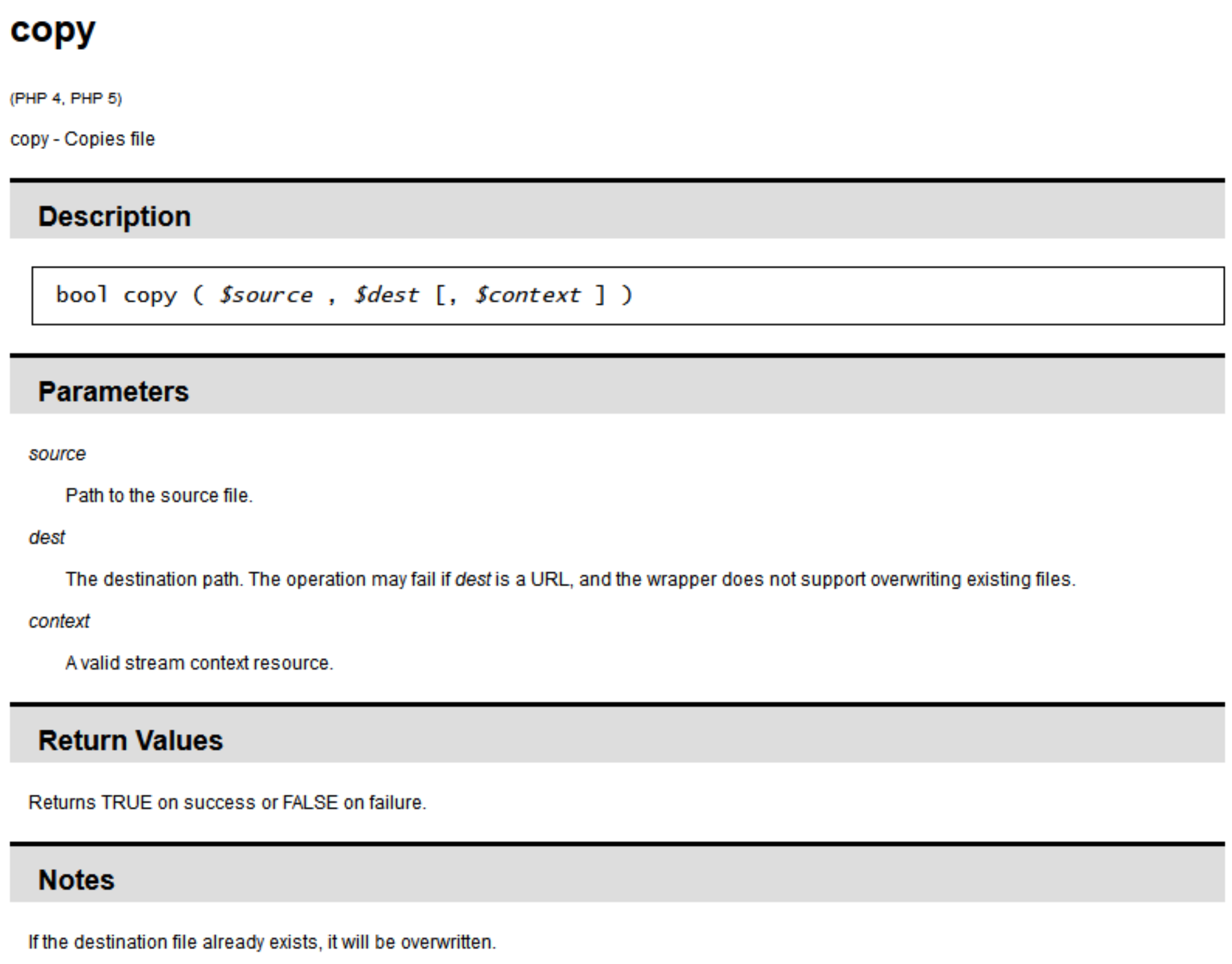
Related Topics

- `stream_copy_to_stream()` - Copies the contents of one stream to another stream
- `file_get_contents()` - Reads the contents of a file to a buffer
- `fwrite()` - Writes a buffer to a file
- `mkdir()` - Creates a new directory
- `move_uploaded_file()` - Moves an uploaded file to a new location
- `rename()` - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
  $file = 'example.txt';
  $newfile = 'example.txt.bak';

  if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
  }
?>
```

Figure 6. Example of API reference topic with a high ICEC


copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( $source , $dest [, $context ] )
```

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if *dest* is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure 7. Example of API reference topic with a low ICEC

Topic Relevance

Topic Relevance indicates whether the API reference topic shown to participants is relevant to the scenario presented to the participant at the start of a task. Topic relevance was manipulated in the study by varying the scenario presented to participants. For each task, before participants viewed the API reference topic, participants saw one of two possible scenarios for that task.

Topic relevance had two levels:

- **Relevant** – The topic was relevant to the scenario presented to the reader. Relevance is operationalized as the topic having many contextual effects in the context described by the task scenario and was manipulated by creating a scenario that describes a context in which the topic will have many contextual effects.
- **Non-Relevant** – The topic was not relevant to the scenario presented to the reader. Non-relevance is operationalized as the topic having few contextual effects in the context described by the task scenario and was manipulated by creating a scenario that describes a context in which the topic will have few contextual effects.

Both scenarios for a task were written such that the resulting API reference topic would be a reasonable or plausible topic to review in response to the scenario. The relevant scenario, however, was the only scenario that contained the contextual effects that matched the scenario. Participants assessed the topic's relevance to the scenario as a binary "yes/no," value by answering the question, "Does this topic contain the information you need?" While Sperber and Wilson (1986) describe relevance as a continuous property and Barry and Schamber (1998) describe relevance as a complex property, the binary value used here provides a sufficiently precise measure for this study.

Table 5 contains the text of each scenario level used for each task in the study.

Table 5. Task scenarios

Study Task	Relevant scenario description	Non-relevant scenario description
0	You want to know if the month parameter of the **checkdate** function can be the actual name of the month. You did a search on **checkdate** and the help topic on the next page is one of the results.	You want to know if the month parameter of the **checkdate** function can be the actual name of the month. You did a search on **checkdate** and the help topic on the next page is one of the results.
1	You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs. You searched for the **copy** function to find out and the help topic on the next page is one of the results.	You want to make a deep copy of an object using the copy method but you can't recall how to call the **copy** method to perform such an operation. You searched for the **copy** method to find out and the help topic on the next page is one of the results.
2	You have a module that calls the **print** function with an argument that includes parentheses and it's not working as you expect. You want to see if the documentation says anything about this—specifically, if parentheses are allowed in the function's argument. You searched for the **print** function to find out and the help topic on the next page is one of the results.	You want to call the **print** function to format a numeric value as a character string. You know that you need to pass a format string to do this, but you can't recall how to configure it. You searched for the **print** function to find the format string and the help topic on the next page is one of the results.
3	You want to know if the **select** function is able to return immediately or if it must wait for a file descriptor to become ready before it returns. You searched for select to find out and the help topic on the next page is one of the results.	You want to query specific fields from a SQL database and you can't remember the SELECT command syntax to do that. You searched for SELECT to find out and the help topic on the next page is one of the results.

Study Task	Relevant scenario description	Non-relevant scenario description
4	You want to insert a character between each string in an array to separate them in the returned string and you want to know if the **join** function can do that. You searched for **join** and the help topic on the next page is one of the results.	You want to create a query to join two tables in a SQL database on a specific field and you can't remember how to format the command to do that. You searched for JOIN to find out and the help topic on the next page is one of the results.

Dependent Measures

This section describes the data collected by the experiment.

Relevance-Decision Response Time

Relevance-Decision Response Time was how long the participant took to determine if the topic was relevant to the question in the task scenario, measured a continuous variable in milliseconds. The relevance-decision response time was measured as the time that elapsed between when the API reference topic was shown to the participant and when the participant indicated his or her assessment of the topic's relevance to the task's scenario. It was operationalized in the survey tool by instrumenting these events and recording the time between them.

The timer used in this study has a nominal measurement error, as measured by Resig (2008) and Zakaz (2011) of 15.6 milliseconds and a worst-case measurement error of 55 milliseconds. This error affects the measurement's precision. Because the participant must find and click the correct response button, any problems with the user-interface could also influence the measurement's accuracy. For example, such problems could add a delay between the instant the decision was made and the instant the response was recorded. Performing a practice task before the timed tasks should have helped reduce some of these participant and interface-related sources of error. Because the measurements for this study were detected and computed on the participant's computer in an unmoderated session, the actual measurement error that may have resulted from factors that influence its precision and accuracy cannot be known. The worst-case timer error of 55 milliseconds was expected to be about 1 percent of the average response times, so even in the worst-case, the measurements were expected to be sufficiently accurate for this study to detect meaningful effects.

The timer that measured the relevance-decision response time was implemented to use the JavaScript software timer that the browser on the participant's computer provides. The response time used in this study was measured as the difference between the system timestamp recorded by the measurement software when the browser indicated the web page had loaded completely and the system timestamp recorded when the participant selected a response button. The system timestamp was recorded in units of milliseconds, but Resig (2008) and Zakaz (2011) agree that time measured this way is not necessarily accurate to the millisecond. The actual accuracy depends on many factors, such as the browser, operating system, and type of computer on which the browser was running at the time of the test. Because the measurement is determined on the participant's computer before it is recorded on the study server, external factors such as network performance and latency did not influence the measurement accuracy.

Assessment Correctness

Assessment Correctness is a categorical value that indicated whether the participant correctly assessed whether or not the topic matched the scenario presented to the participant in the task.

Assessment Correctness had two values:

- **Correct**

  The participant's assessment agreed with the topic relevance level of the experiment condition. The participant should select "Yes" after reading a relevant scenario or "No" after reading a non-relevant scenario.

- **Incorrect**

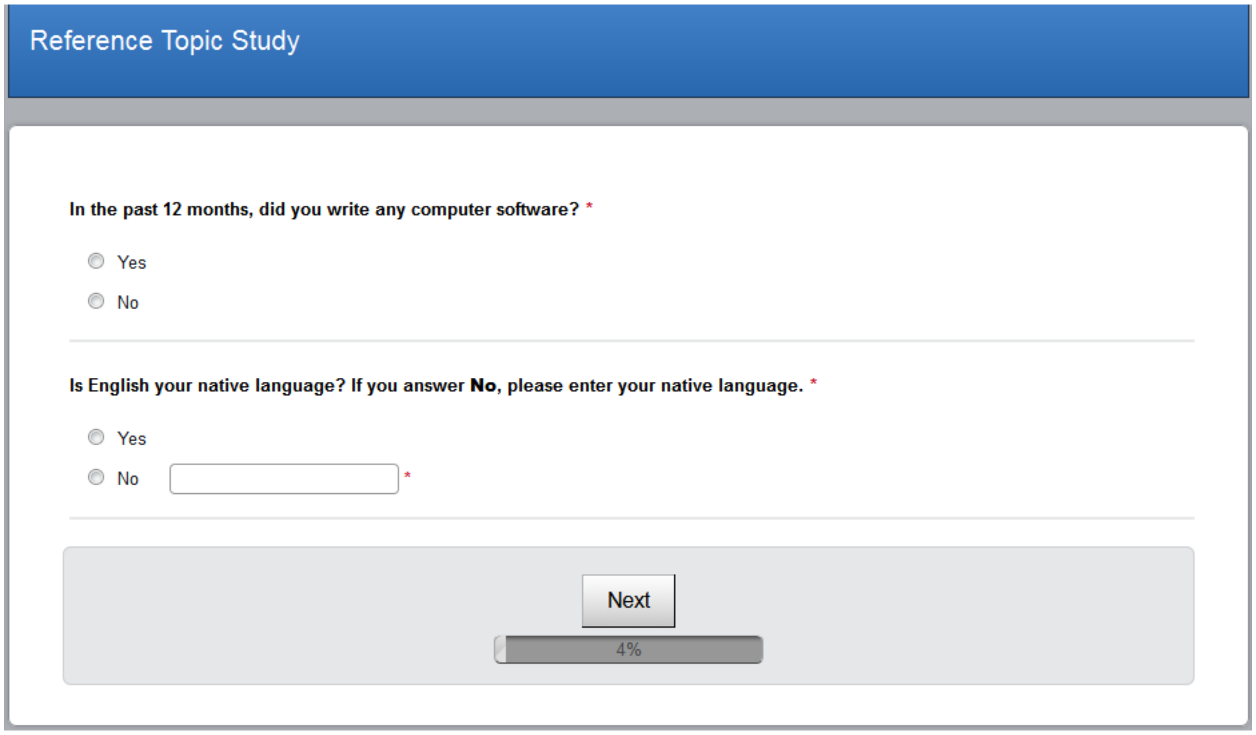
  The participant's assessment disagreed with the topic relevance level of the experiment condition. The participant selected "Yes" after reading a non-relevant scenario or "No" after reading a relevant scenario.

Participant's Programming Experience

Participants self-reported the number of years they have been programming as an integer in the initial demographic information questionnaire. These data were collected to determine if variations in programming experience influenced assessment time. Figure 8 shows the initial questionnaire format in which participants were asked if they had any experience writing software. If they answered *yes,* the form expanded, as Figure 9 shows, to ask how many years the participant had been writing software.

Participant's Proficiency in English

Participants reported their English proficiency by indicating whether English was their native language, as shown in Figure 8. If English was not the participant's native language, the questionnaire expanded, as shown in Figure 9, to show additional questions that were used to test their English proficiency. Because the task scenarios and API reference topics used in the study were written in English, participants' English-language proficiency could influence their assessment time.


Reference Topic Study

In the past 12 months, did you write any computer software? *

Yes

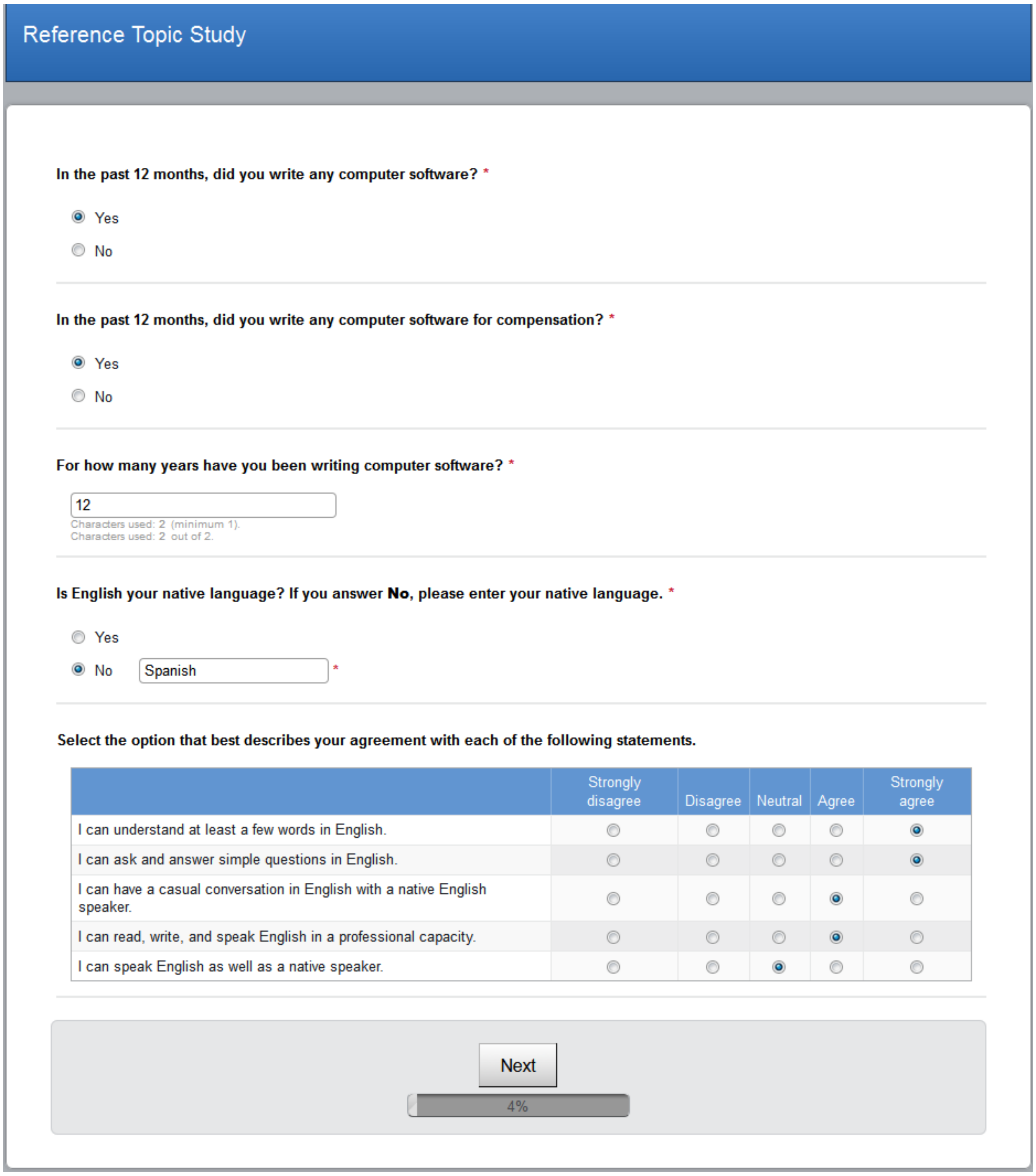
No

Is English your native language? If you answer **No**, please enter your native language. *

Yes

No *

Figure 8. Demographic questionnaire – collapsed



Reference Topic Study

In the past 12 months, did you write any computer software? *

[✓ Yes]
No

In the past 12 months, did you write any computer software for compensation? *

[✓ Yes]
No

For how many years have you been writing computer software? *

12
Characters used: 2 (minimum 1).
Characters used: 2 out of 2.

Is English your native language? If you answer No, please enter your native language. *

Yes
[✓ No] Spanish *

Select the option that best describes your agreement with each of the following statements.

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I can understand at least a few words in English.					✓
I can ask and answer simple questions in English.					✓
I can have a casual conversation in English with a native English speaker.				✓	
I can read, write, and speak English in a professional capacity.				✓	
I can speak English as well as a native speaker.			✓		

Figure 9. Demographic questionnaire – fully expanded

Participant Perception Measures

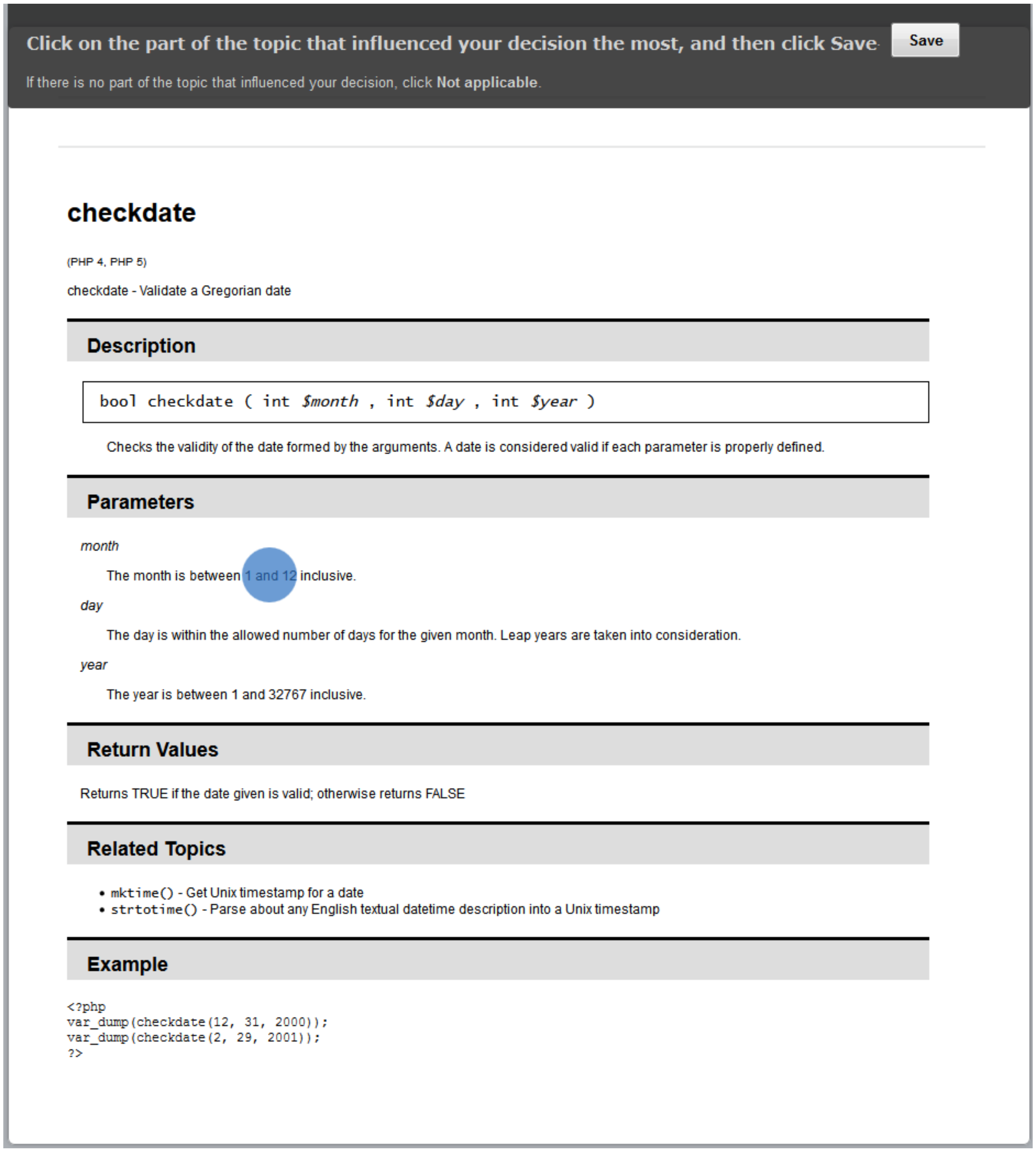
This section describes the measures used to record the participants' perceptions of the API reference topics.

Most Influential Topic Element

While looking at the API reference topic, after indicating the topic's relevance, participants were asked to "*Click on the part of the topic that influenced your decision the most, and then click **Save**"* as shown in Figure 10. Participants clicked on the topic and saw a colored circle on the topic to illustrate where they clicked. The location of the click was recorded by the survey tool as x,y coordinates when participants clicked the **Save** button. The coordinates were later mapped to the topics' sections when the data were analyzed.

Questionnaire

The following perceptual measures were used to collect information from participants about their perceptions of the API reference topic they saw after they had noted the most influential part of the topic. This statement at the top of the questionnaire prefaced all of the questions, *"The questions on this page refer to the topic you just reviewed,"* and a reduced-size image of the topic was shown alongside the questions.



Click on the part of the topic that influenced your decision the most, and then click Save

If there is no part of the topic that influenced your decision, click **Not applicable**.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

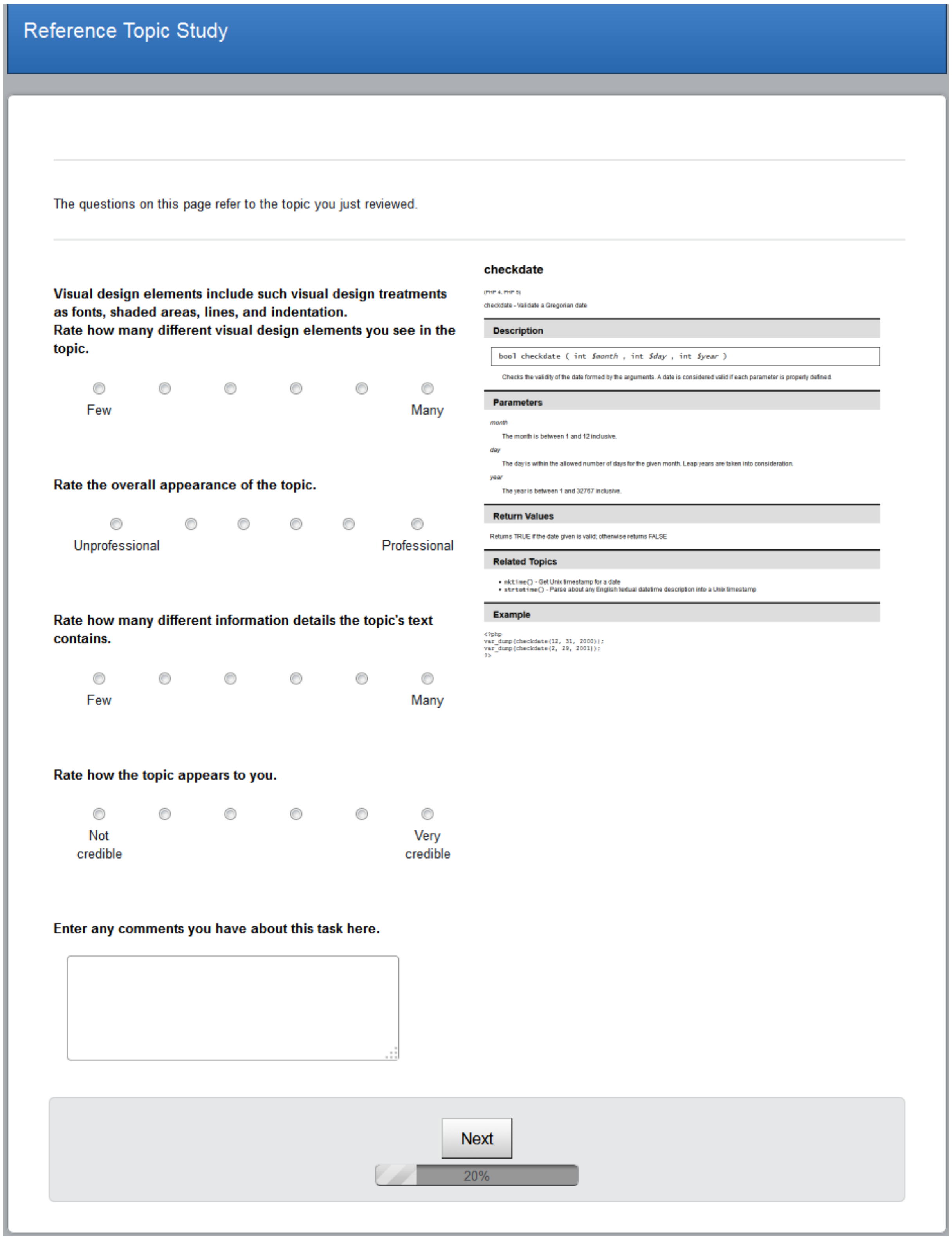
Figure 10. API reference topic annotation page – after participant notes the location

Figure 11 shows an example of the questionnaire used in the study, which included the following perception questions.

- **Visual Design**
    - Participants were prompted, *"Visual-design elements include such visual design treatments as fonts, shaded areas, lines, and indentation. Rate how many different visual-design elements you see in the topic."*
    - Their response was measured by a 6-pt Likert scale anchored by *Few* and *Many* and scored from 1 to 6, with 0 indicating a missing value.
- **Topic Appearance**
    - Participants were prompted, *"Rate the overall appearance of the topic."*
    - Their response was measured by a 6-pt Likert scale anchored by *Unprofessional* and *Professional* and scored from 1 to 6, with 0 indicating a missing value.
- **Textual information content**
    - Participants were prompted, *"Rate how many different information details the topic's text contains."*
    - Their response was measured by a 6-pt Likert scale anchored by *Few* and *Many* and scored from 1 to 6, with 0 indicating a missing value.
- **Topic credibility**
    - Participants were prompted, *"Rate how the topic appears to you."*
    - Their response was measured by a 6-pt Likert scale, anchored by *Not credible* and *Very credible*) and scored from 1 to 6, with 0 indicating a missing value.

- **Optional comments**
    - Participants were prompted, *"Enter any comments you have about the task here."*
    - Their response was recorded in a Comment box into which the participant could enter free-form text comments.

Responses to the credibility rating were used to test Hypothesis H7. Responses to the other questions were evaluated in exploratory investigations and the comments were reviewed to identify any patterns of interest.



Reference Topic Study

The questions on this page refer to the topic you just reviewed.

Visual design elements include such visual design treatments as fonts, shaded areas, lines, and indentation.
Rate how many different visual design elements you see in the topic.

Few ○ ○ ○ ○ ○ ○ Many

Rate the overall appearance of the topic.

Unprofessional ○ ○ ○ ○ ○ ○ Professional

Rate how many different information details the topic's text contains.

Few ○ ○ ○ ○ ○ ○ Many

Rate how the topic appears to you.

Not credible ○ ○ ○ ○ ○ ○ Very credible

Enter any comments you have about this task here.

Next

20%

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure 11. Perception questionnaire example

Participant Selection

Participants were recruited by promoting and advertising the study in online groups related to software development. The groups from the initial set are listed in Table 6. Other groups were added as they were encountered or referred. Table 9 in **Results** contains the complete list of the groups to which a study invitation was sent.

Table 6. Table of online groups invited to participate in the study

Invitation Group	**Population**	**URL of group in Linked In**
Java Developer	258,238	https://www.linkedin.com/groups?gid=70526
Open Source	116,133	https://www.linkedin.com/groups?gid=43875
PHP developers	100,000	https://www.linkedin.com/groups?gid=42140
Android Developer Group	98,906	https://www.linkedin.com/groups?gid=86481
Developers	80,228	https://www.linkedin.com/groups?gid=54723
iOS Developers Group	57,687	https://www.linkedin.com/groups?gid=121874
Web Designer and HTML/CSS	44,622	https://www.linkedin.com/groups?gid=3242849
iPhone Developers	36,168	https://www.linkedin.com/groups?gid=72283
Programmers and Developers	28,990	https://www.linkedin.com/groups?gid=1787637
PHP Developer Network	20,581	https://www.linkedin.com/groups?gid=2195403
Software Developer	20,000	https://www.linkedin.com/groups?gid=1074487
iOS Developer Network	10,551	https://www.linkedin.com/groups?gid=2627917
C++ Software Developers	9,290	https://www.linkedin.com/groups?gid=2771729
Open Source Developers Community	4,143	https://www.linkedin.com/groups?gid=2272881

The study was authorized by the Institutional Review Board of the University of Washington to collect data from no more than 500 participants.

Online Study Protocol

The study protocol for each participant was implemented in the survey tool and consisted of the following steps. These steps are summarized here and described in the sections that follow.

In each participant session:

1. Participants read the welcome and consent to participate message. When a participant started a study session, the survey tool selected one of four possible API reference topic variations and one of six possible relevance sequences to use in the study session.
2. Participants entered their demographic data.
3. Participants reviewed and performed a practice task.
4. Participants performed the study tasks. The study tasks included these tasks, which were displayed to each participant in a randomized order.
    a. Copy task
    b. Print task
    c. Select task
    d. Join task

    The API reference topic that a participant saw in each task was the same variation—the variation selected by the survey tool when the participant started a study session. This method accommodated the between-subjects variations of VDEC and ICEC and the randomized task sequence minimized order effects. The within-subjects variable was manipulated and randomized by showing each participant two relevant scenarios and two non-relevant scenarios—the sequence of which was randomized between participants. In the practice task and each of the four study tasks, participants performed these steps.

    a. Read the scenario for the task.

b. Reviewed the API reference topic and:

i. Assessed the topic's relevance to the scenario.

ii. Identified the location in the topic that influenced their relevance decisions the most.

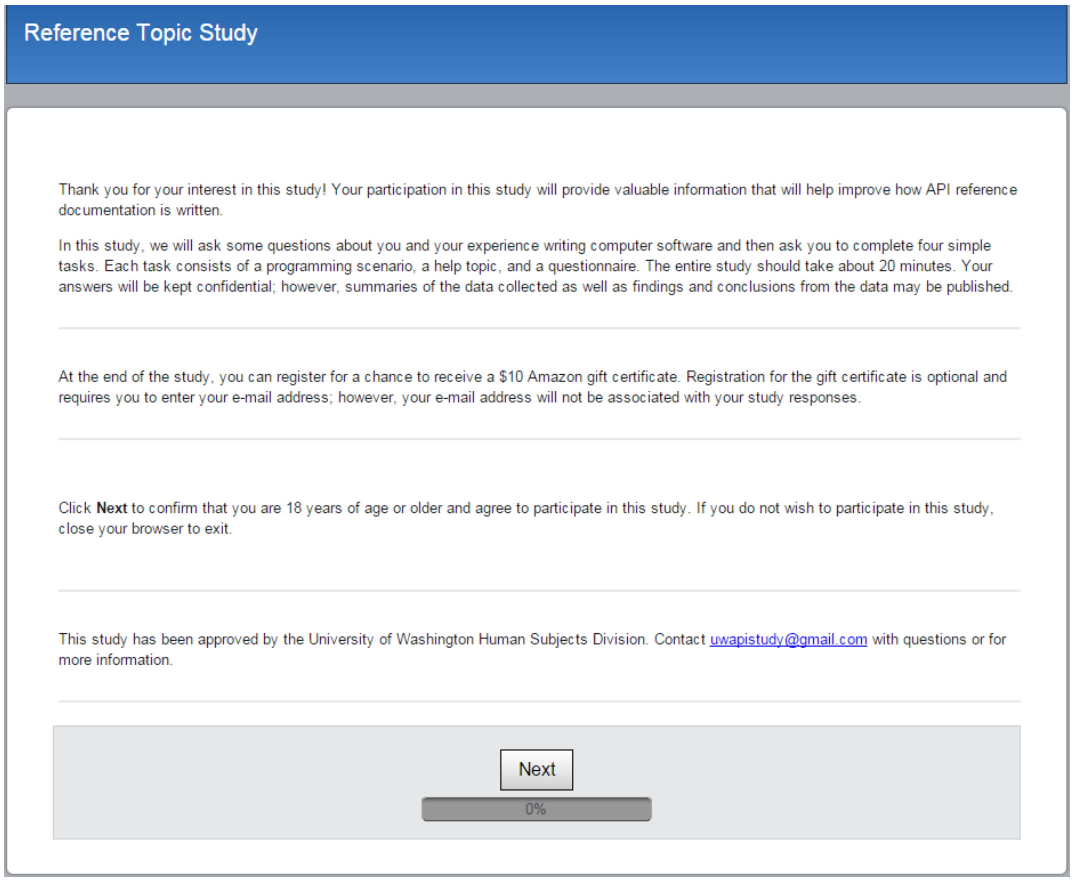
c. Answered the questions in the post-task questionnaire.

5. Participants entered their information if they wanted a chance to win the gratuity.

6. Participants were thanked for their participation in the study.

Welcome and Consent Page

Participants saw the welcome page shown in Figure 12 when they began the study.



Reference Topic Study

Thank you for your interest in this study! Your participation in this study will provide valuable information that will help improve how API reference documentation is written.

In this study, we will ask some questions about you and your experience writing computer software and then ask you to complete four simple tasks. Each task consists of a programming scenario, a help topic, and a questionnaire. The entire study should take about 20 minutes. Your answers will be kept confidential; however, summaries of the data collected as well as findings and conclusions from the data may be published.

At the end of the study, you can register for a chance to receive a $10 Amazon gift certificate. Registration for the gift certificate is optional and requires you to enter your e-mail address; however, your e-mail address will not be associated with your study responses.

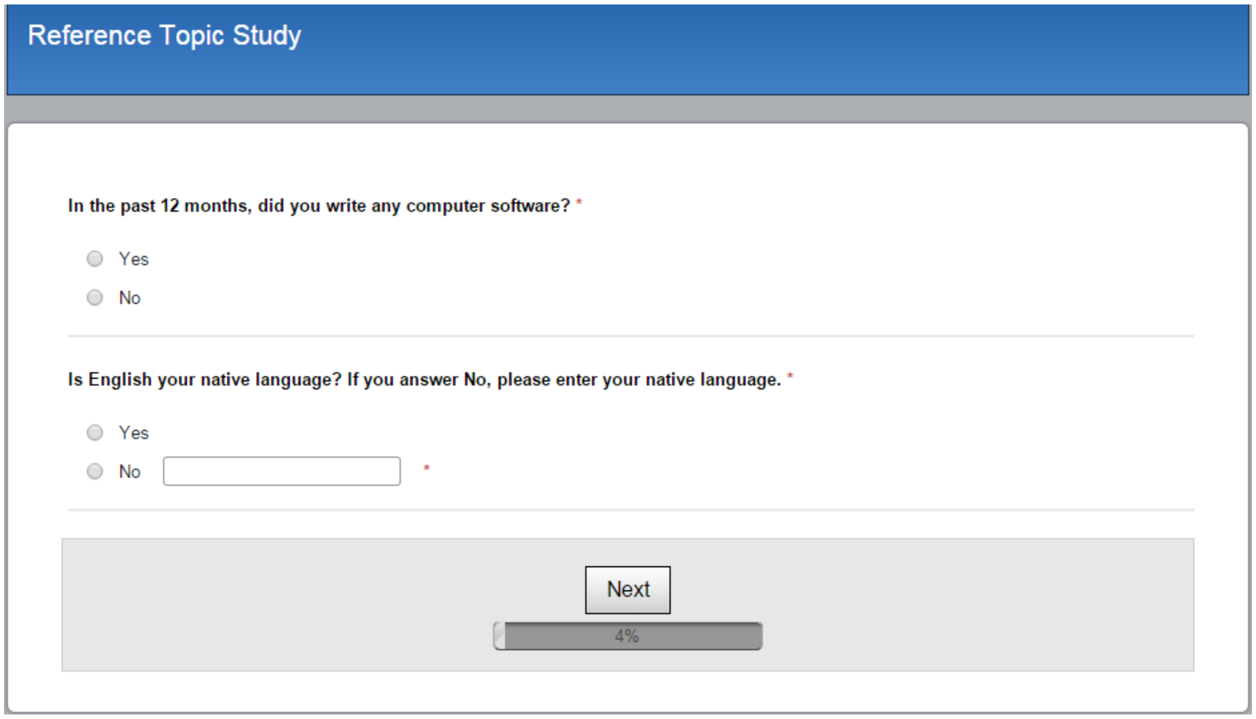
Click **Next** to confirm that you are 18 years of age or older and agree to participate in this study. If you do not wish to participate in this study, close your browser to exit.

This study has been approved by the University of Washington Human Subjects Division. Contact uwapistudy@gmail.com with questions or for more information.

Figure 12. Welcome and consent page of the online study

Participant's Demographic Data

After consenting to the study by clicking **Next** in the Welcome and Consent page (Figure 12), participants were presented with the demographic questionnaire. The demographic questionnaire first appeared in the collapsed form shown in Figure 13. If participants answered **Yes** to the first question, *"In the past 12 months, did you write any computer software?"* then two more questions about writing software appeared. If participants answered **No** to the question about their native language being English, additional questions about language proficiency appeared. Figure 14 shows the questionnaire expanded to show all of its questions.


Reference Topic Study

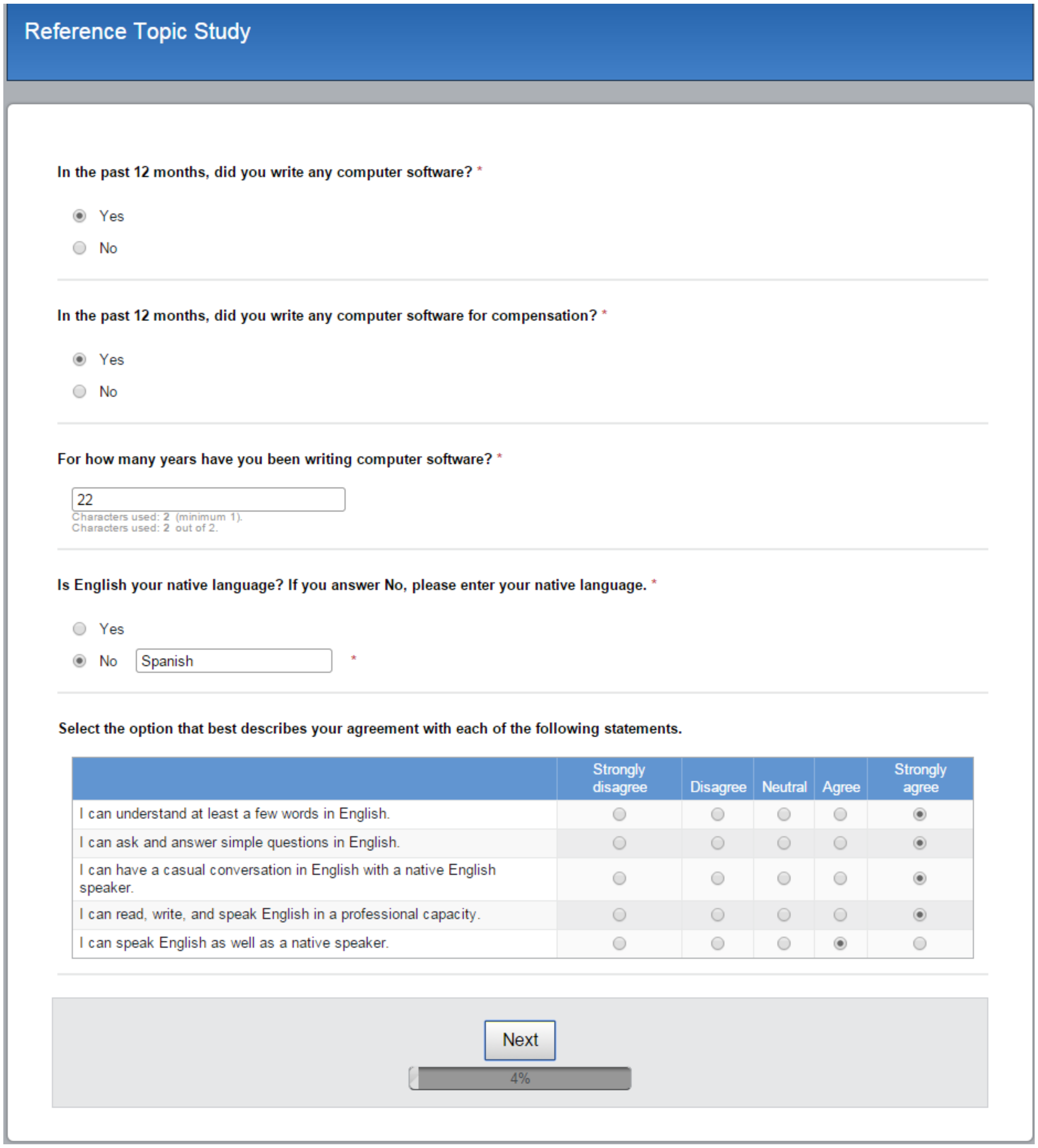
In the past 12 months, did you write any computer software? *

Yes
No

Is English your native language? If you answer No, please enter your native language. *

Yes
No *

Figure 13. Participant demographic questionnaire – collapsed


Reference Topic Study

In the past 12 months, did you write any computer software? *

[✓ Yes] No

In the past 12 months, did you write any computer software for compensation? *

[✓ Yes] No

For how many years have you been writing computer software? *

22
Characters used: 2 (minimum 1).
Characters used: 2 out of 2.

Is English your native language? If you answer No, please enter your native language. *

Yes [✓ No] Spanish *

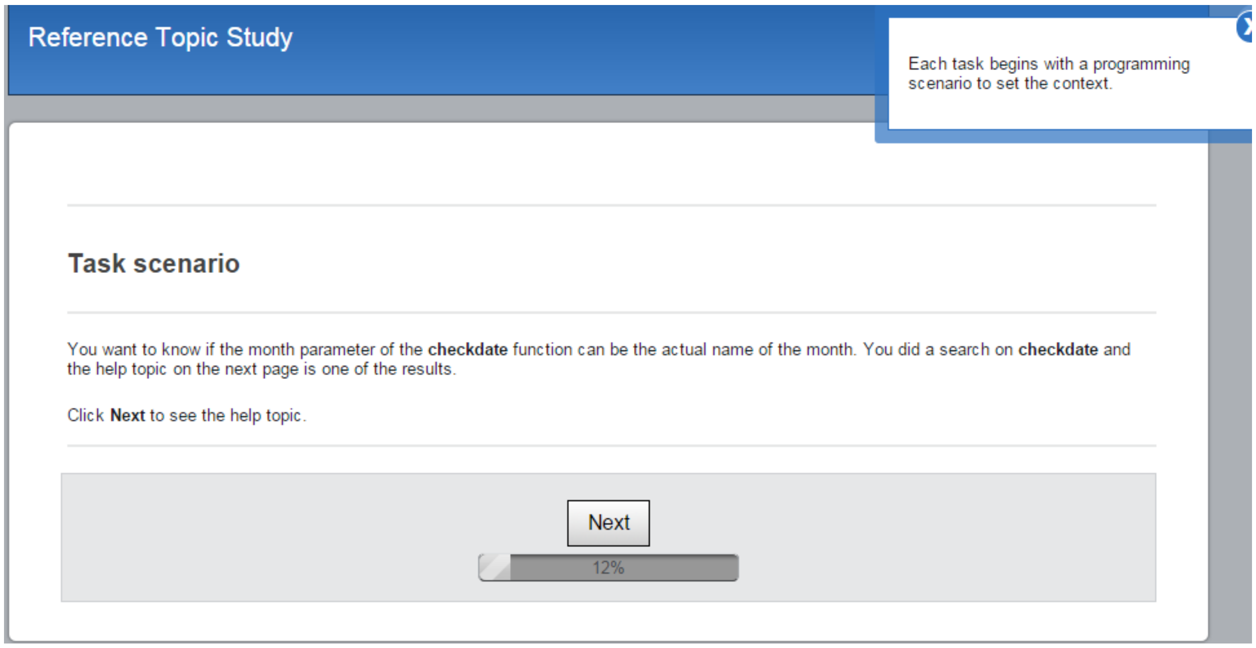
Select the option that best describes your agreement with each of the following statements.

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I can understand at least a few words in English.	○	○	○	○	●
I can ask and answer simple questions in English.	○	○	○	○	●
I can have a casual conversation in English with a native English speaker.	○	○	○	○	●
I can read, write, and speak English in a professional capacity.	○	○	○	○	●
I can speak English as well as a native speaker.	○	○	○	●	○

Figure 14. Participant demographic questionnaire – fully expanded

Study Task Practice

A sample task was shown to participants in the context of a practice scenario. Each step in the task included a help window in the upper-left corner of the screen that described the step in the task. Because the study task was timed, the practice task was designed to prepare participants for the mechanics of the study tasks to reduce measurement error that could be attributed to problems a participant might have interacting with the study. Figure 15 shows an example of a practice step with the instructions overlaid and Figure 16 shows the same page after the participant has dismissed the instructions.



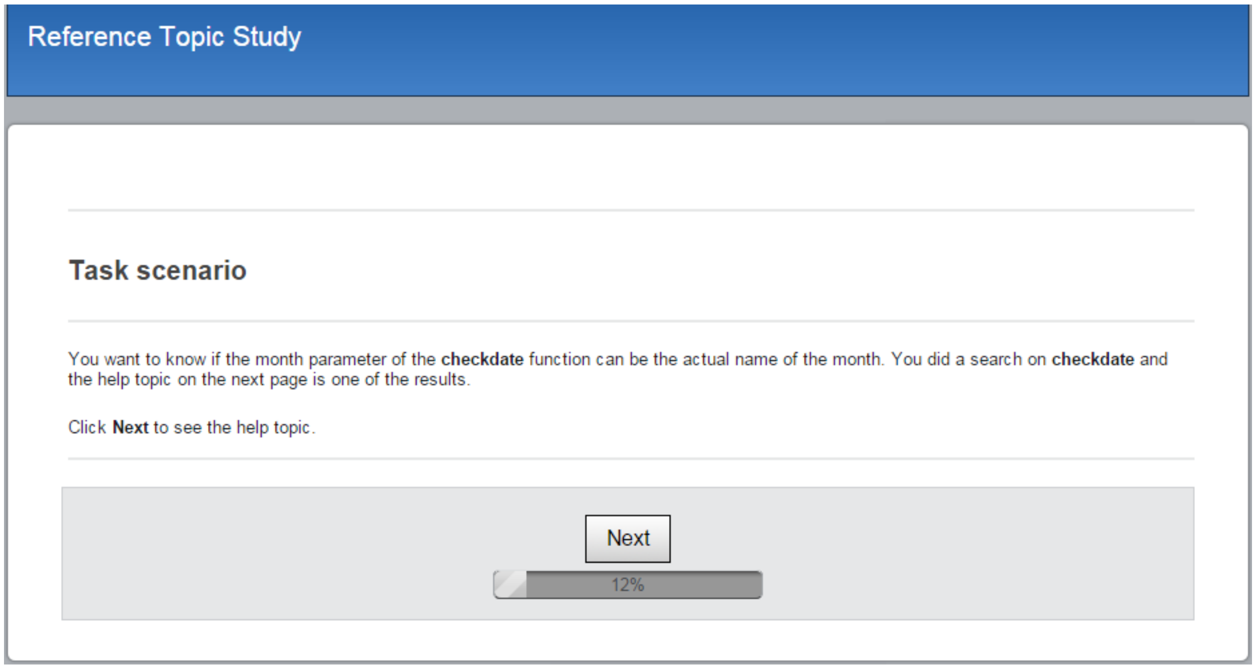
Each task begins with a programming scenario to set the context.

Task scenario

You want to know if the month parameter of the **checkdate** function can be the actual name of the month. You did a search on **checkdate** and the help topic on the next page is one of the results.

Click **Next** to see the help topic.

Figure 15. Practice study task page with instructions overlaid


Reference Topic Study

Task scenario

You want to know if the month parameter of the **checkdate** function can be the actual name of the month. You did a search on **checkdate** and the help topic on the next page is one of the results.

Click **Next** to see the help topic.

Figure 16. Practice study task after reading the instructions

Study Tasks

For each study task, participants performed the three steps listed here and described in the sections that follow.

1. Read the scenario for the task.
2. Reviewed the API reference topic and:
   a. Assessed the topic's relevance to the scenario.
   b. Identified the location in the topic that influenced their relevance decisions the most.
3. Answered the questions in the post-task questionnaire.

Read the Scenario for the Task

Each task started with a scenario. The Topic Relevance variable of the API reference topic was manipulated by changing the relevance of the scenario that framed the task. For each task, two equally plausible scenarios were developed: one for which the help topic would be relevant and one for which it would not. Table 5 in the **Independent Variables** section shows the different descriptions of the two scenarios for each task in the study. Task 0 was the practice task and so both scenario descriptions are the same and relevant. The formatting used in the scenario descriptions was designed to be as consistent as possible with the documentation and programming language referenced. For example, function names are commonly shown in bold text and Structured Query Language (SQL) commands are commonly shown as all upper-case letters.

Of the four tasks presented to the participants, two had scenarios that were relevant to the topic they would see and two had scenarios that were not. The order of the Topic Relevance level was randomized between participants. At the beginning of a participant's session, the order of topic relevance was chosen by randomly selecting one of the six possible patterns listed in Table 7. The task number in Table 7 refers to the task order of the study design, because the tasks are also randomized when presented to the participant, the order of topic relevance seen by the participant might not match the order listed in the table.

Table 7. Table of topic relevance sequences

	Scenario presented to participant				
Pattern	**Practice**	**Task 1**	**Task 2**	**Task 3**	**Task 4**
1	Relevant	Non-relevant	Non-relevant	Relevant	Relevant
2	Relevant	Non-relevant	Relevant	Non-relevant	Relevant
3	Relevant	Relevant	Non-relevant	Non-relevant	Relevant
4	Relevant	Non-relevant	Relevant	Relevant	Non-relevant
5	Relevant	Relevant	Non-relevant	Relevant	Non-relevant
6	Relevant	Relevant	Relevant	Non-relevant	Non-relevant

Assessed the API reference topic

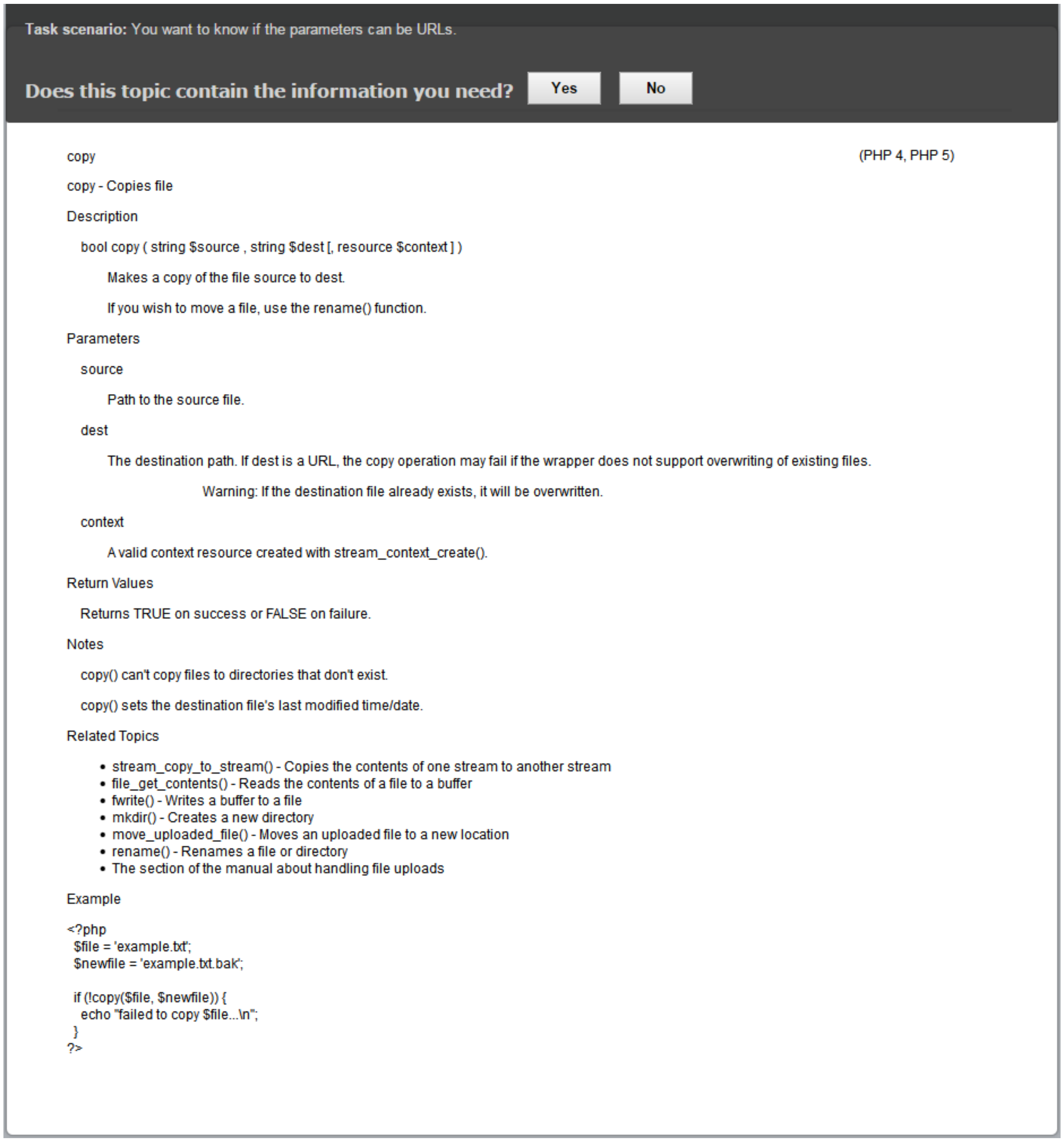
Each API reference topic used in a study task had four possible variations. Each topic had one of two visual designs, corresponding to the VDEC levels, and included one of two possible texts for that topic, corresponding to the ICEC levels. Table 8 lists the variations that result from the different combinations of these levels. Each participant saw only one of a topic's variations—the same VDEC and ICEC levels were used in each task shown to a single participant.

Table 8. Table of API reference topic variations

	High VDEC	**Low VDEC**
High ICEC	API reference Topic Variation 1	API reference Topic Variation 3
Low ICEC	API reference Topic Variation 2	API reference Topic Variation 4

In this step, participants evaluated two aspects of each API reference topic they saw: the topic's relevance to the scenario and the most influential part of the topic. First, they assessed the topic's relevance to the scenario by selecting **Yes** or **No** at the top of the screen to indicate whether the API reference topic contained the information required by the task's scenario. The buttons used to capture this are shown at the top of the screenshot in Figure 17.

After participants registered their assessment of the API reference topics' relevance, they were prompted by the spot interaction to click on the part of the topic that most influenced their decision. Figure 18 shows a screenshot of the screen used to capture this response. When participants clicked on the topic, a spot appeared on the topic to indicate the selected location, which was saved when the participant clicked the **Save** button at the top of the page. Participants who did not want to indicate a section could click the **not applicable** link at the top of the page. The link was not visually prominent to encourage the participant to leave a spot on the topic. Selecting the influential spot on the topic was not timed as precisely as the relevance decision, but the time spent by the participant was computed by survey tool.


Task scenario: You want to know if the parameters can be URLs.
Does this topic contain the information you need?
Yes
No

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy (string $source , string $dest [, resource $context])

Makes a copy of the file source to dest.

If you wish to move a file, use the rename() function.

Parameters

source

Path to the source file.

dest

The destination path. If dest is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with stream_context_create().

Return Values

Returns TRUE on success or FALSE on failure.

Notes

copy() can't copy files to directories that don't exist.

copy() sets the destination file's last modified time/date.

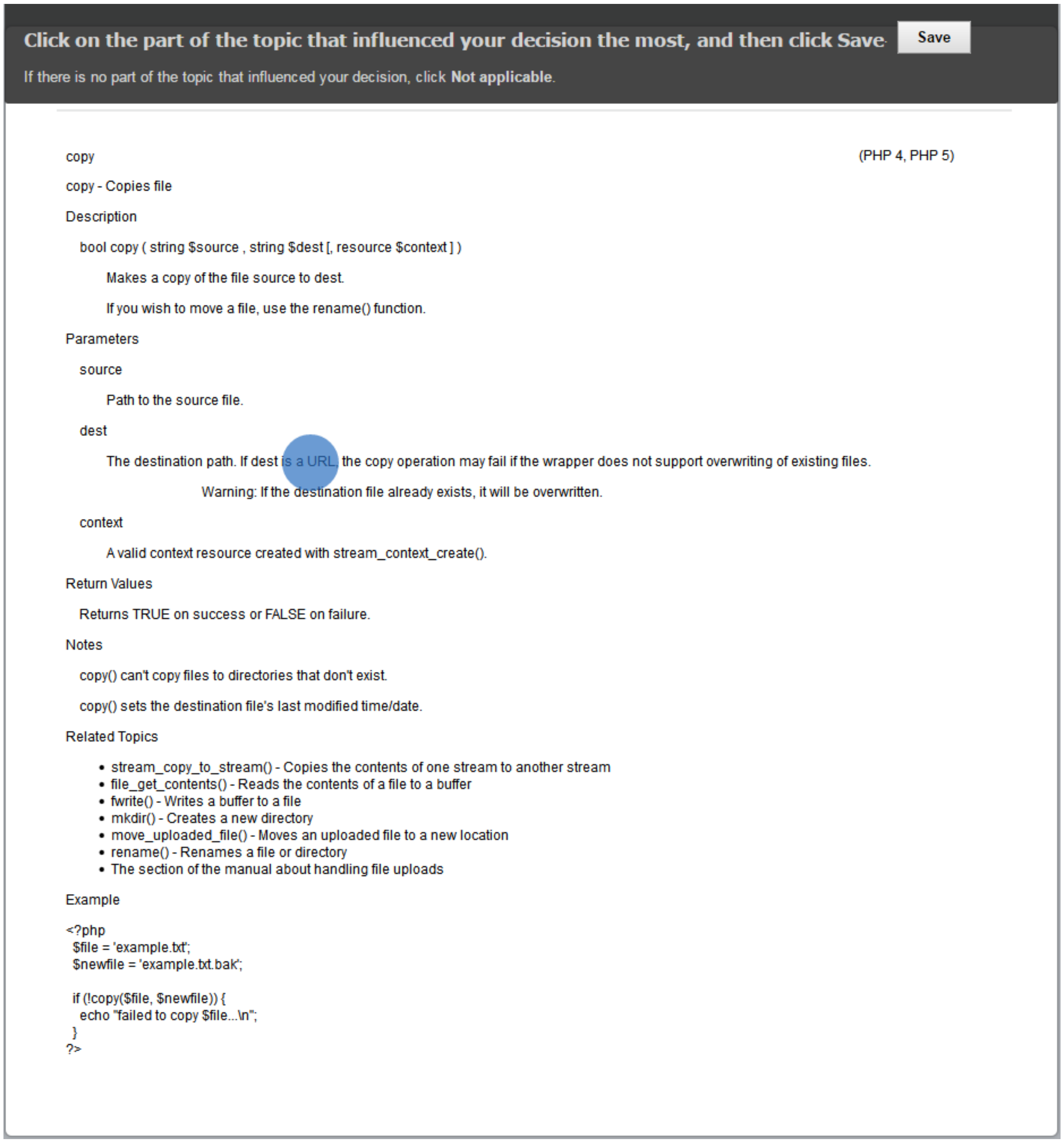
Related Topics

- stream_copy_to_stream() - Copies the contents of one stream to another stream
- file_get_contents() - Reads the contents of a file to a buffer
- fwrite() - Writes a buffer to a file
- mkdir() - Creates a new directory
- move_uploaded_file() - Moves an uploaded file to a new location
- rename() - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
 $file = 'example.txt';
 $newfile = 'example.txt.bak';

 if (!copy($file, $newfile)) {
   echo "failed to copy $file...\n";
 }
?>
```

Figure 17. Example of topic relevance buttons in an API reference topic page


Click on the part of the topic that influenced your decision the most, and then click Save
Save

If there is no part of the topic that influenced your decision, click **Not applicable**.

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy (string $source , string $dest [, resource $context])

Makes a copy of the file source to dest.

If you wish to move a file, use the rename() function.

Parameters

source

Path to the source file.

dest

The destination path. If dest is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with stream_context_create().

Return Values

Returns TRUE on success or FALSE on failure.

Notes

copy() can't copy files to directories that don't exist.

copy() sets the destination file's last modified time/date.

Related Topics

- stream_copy_to_stream() - Copies the contents of one stream to another stream
- file_get_contents() - Reads the contents of a file to a buffer
- fwrite() - Writes a buffer to a file
- mkdir() - Creates a new directory
- move_uploaded_file() - Moves an uploaded file to a new location
- rename() - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
 $file = 'example.txt';
 $newfile = 'example.txt.bak';

 if (!copy($file, $newfile)) {
   echo "failed to copy $file...\n";
 }
?>
```

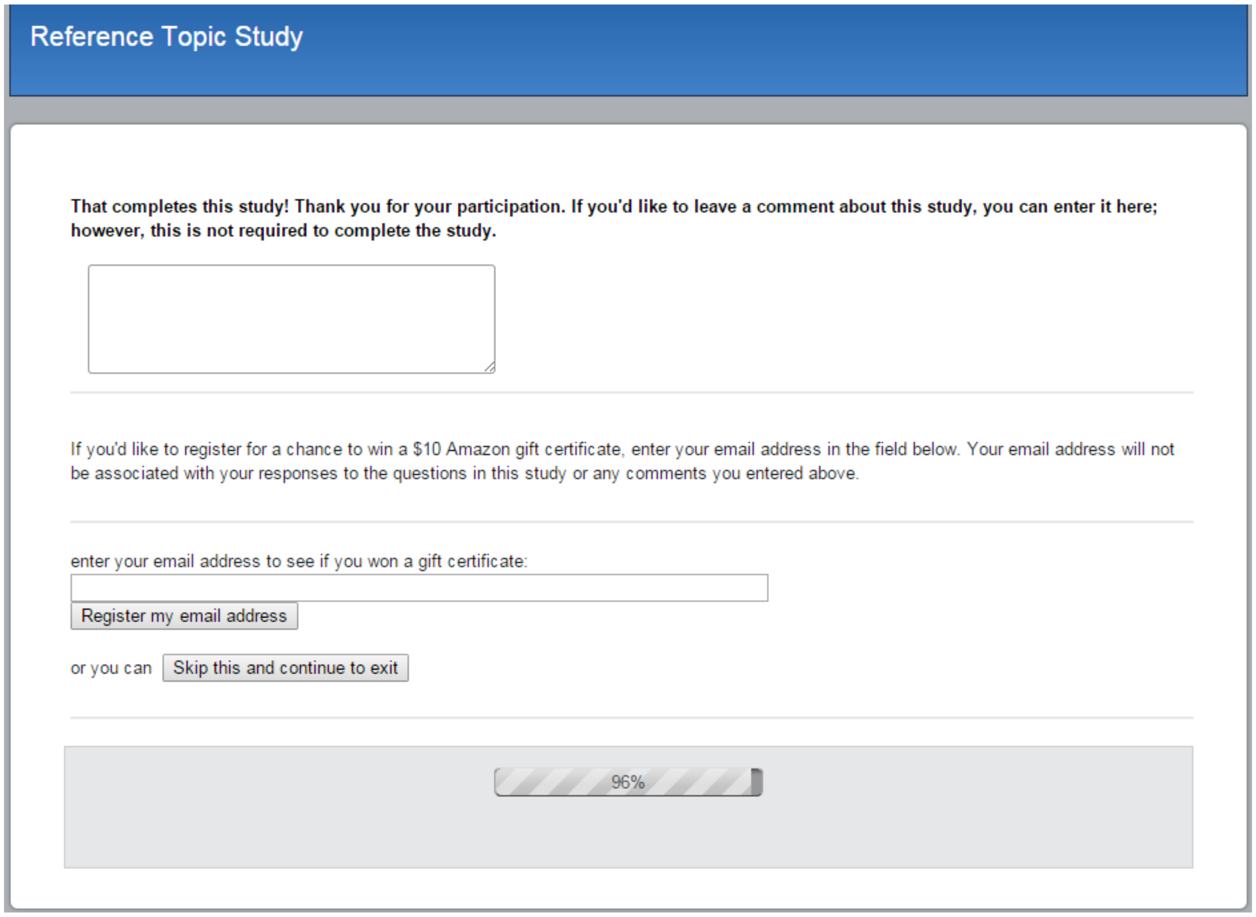
Figure 18. Example of most influential section indication in an API reference topic page

Answered a Post-Task Questionnaire that Asks about the Topic

After assessing the API reference topic, participants answered the questionnaire shown earlier in Figure 11 to record their perceptual assessments.

Gratuity Registration and Thank You

After participants completed the last task of the study, the gratuity registration page shown in Figure 19 was displayed. Participants had the option to register an email address for the chance to receive an Amazon gift certificate, or skipping the gratuity and leaving the study. If participants registered for a gratuity by entering an email address and selecting **Register my email address**, they could see one of two screens, depending on whether they won the gratuity. If they won, they would see their gift certificate code on the page as Figure 20 shows. If they did not win, they would see the thank you shown in Figure 21. If participants elected not to register for a gratuity, they could click **Skip this and continue to exit**, as shown in Figure 19.

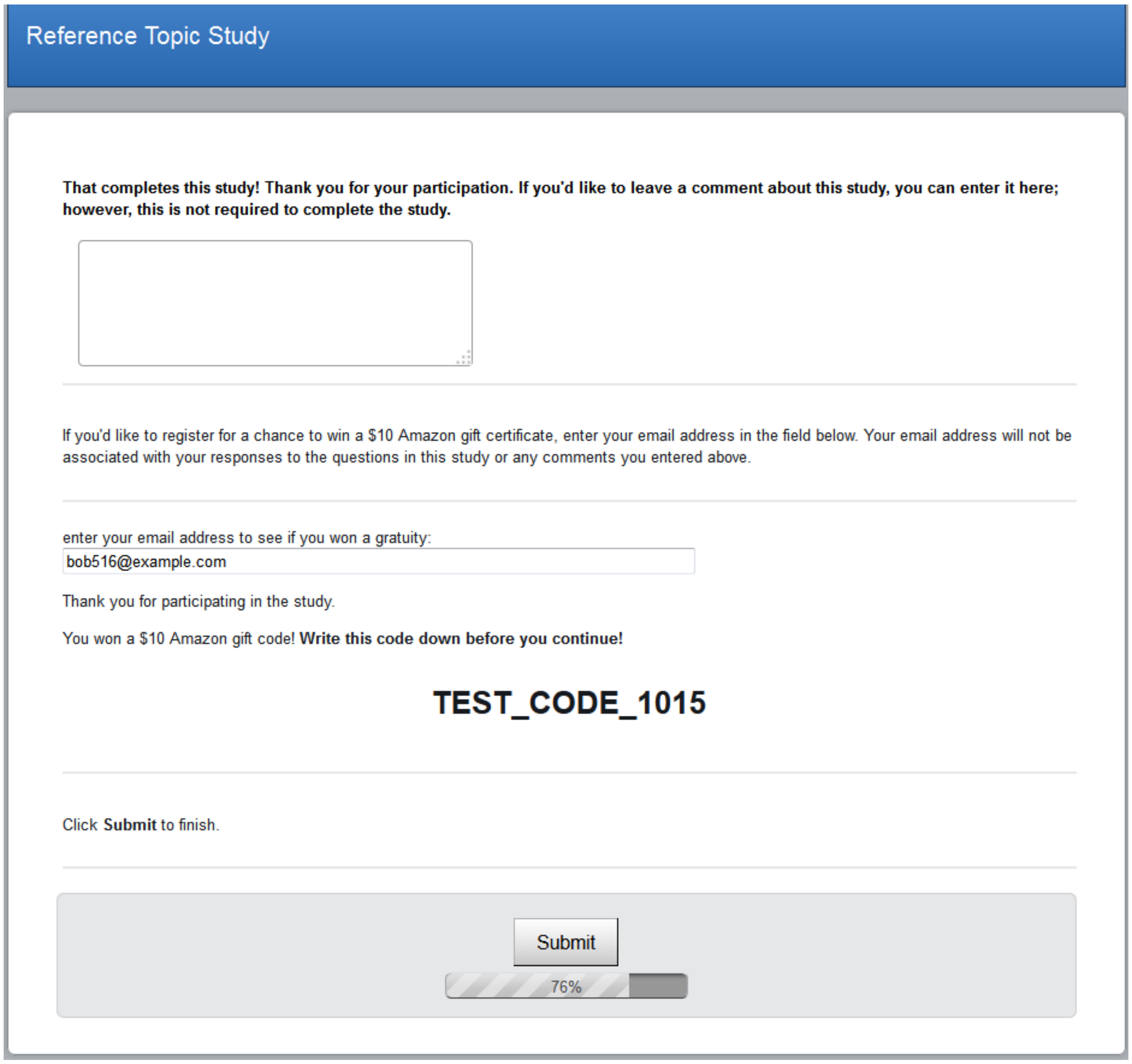

Reference Topic Study

That completes this study! Thank you for your participation. If you'd like to leave a comment about this study, you can enter it here; however, this is not required to complete the study.

If you'd like to register for a chance to win a $10 Amazon gift certificate, enter your email address in the field below. Your email address will not be associated with your responses to the questions in this study or any comments you entered above.

enter your email address to see if you won a gift certificate:

Figure 19. Gratuity registration page - initial view



That completes this study! Thank you for your participation. If you'd like to leave a comment about this study, you can enter it here; however, this is not required to complete the study.

If you'd like to register for a chance to win a $10 Amazon gift certificate, enter your email address in the field below. Your email address will not be associated with your responses to the questions in this study or any comments you entered above.

enter your email address to see if you won a gratuity:
bob516@example.com

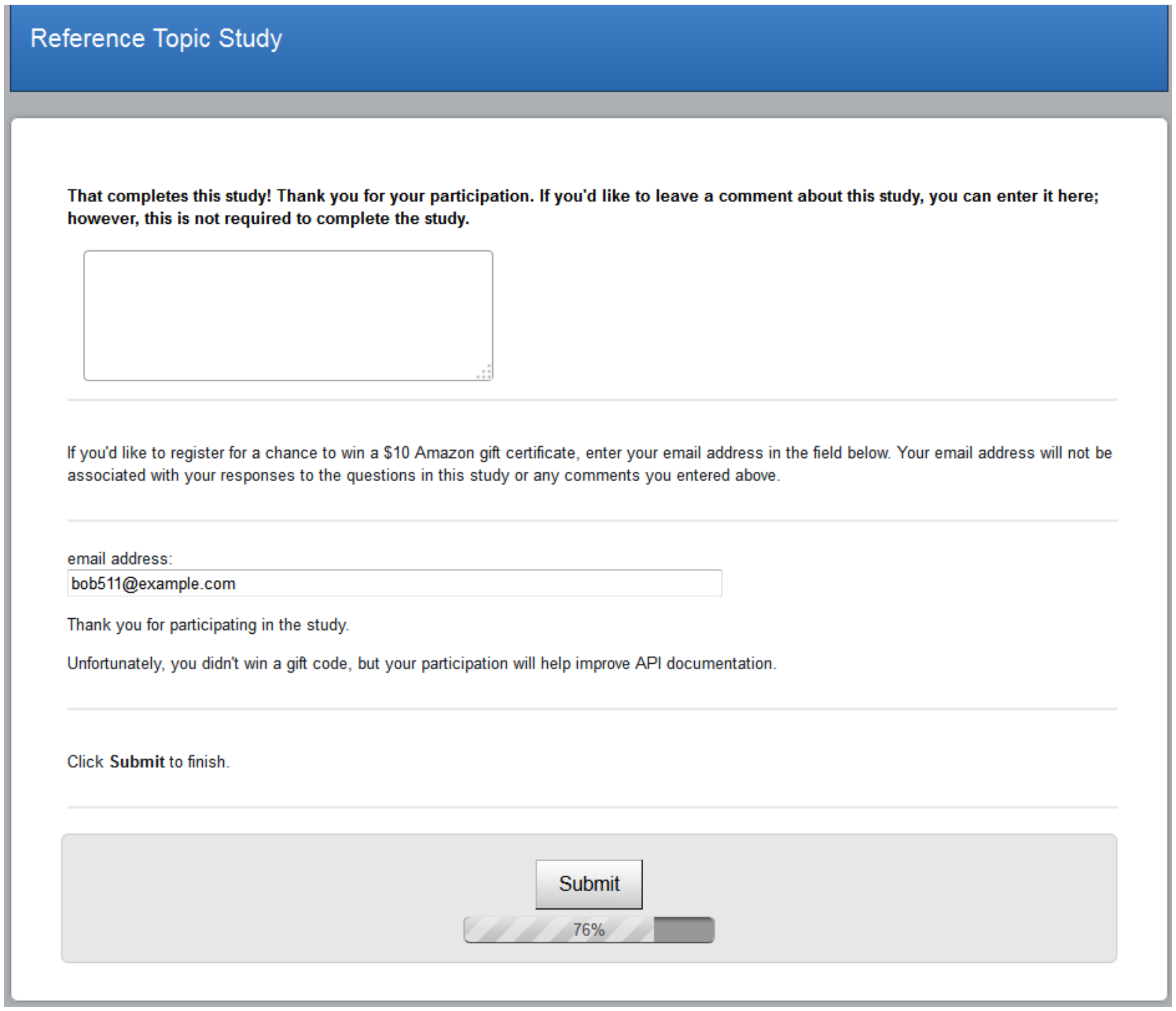
Thank you for participating in the study.

You won a $10 Amazon gift code! **Write this code down before you continue!**

TEST_CODE_1015

Click **Submit** to finish.

Figure 20. Gratuity registration page with gift certificate code


Reference Topic Study

That completes this study! Thank you for your participation. If you'd like to leave a comment about this study, you can enter it here; however, this is not required to complete the study.

If you'd like to register for a chance to win a $10 Amazon gift certificate, enter your email address in the field below. Your email address will not be associated with your responses to the questions in this study or any comments you entered above.

email address:
bob511@example.com

Thank you for participating in the study.

Unfortunately, you didn't win a gift code, but your participation will help improve API documentation.

Click **Submit** to finish.

Figure 21. Gratuity registration page without gift certificate code

Data Collection Plan

Before the data collected by the survey tool could be analyzed, they had to be merged with the response time data and the location data from the spot interaction, and then formatted into records that could be analyzed by SPSS (v.19). An automated script performed the merging and formatting process. The following sections describe these processes.

Study Data from Survey Tool

The study protocol was programmed into a commercial, online survey tool hosted by SurveyGizmo.com, which collected data in its own proprietary format. The design of the study in SurveyGizmo determined the data-record format provided by the survey tool. The SurveyGizmo tool created one record per participant wherein each record included the demographic data and the questionnaire data from all four scenarios. The SurveyGizmo survey tool did not include measurement tools of sufficient precision to measure dependent measures such as relevance-decision response time, so additional, customized measurement instrumentation was developed and added to the survey tool. The additional tools are described in the **Survey Tool Modifications** section in the Appendix. These customizations made it possible to collect measurement and interaction data that were sufficiently precise to test the hypotheses and record the area in the API reference topic that influenced the participant's decision.

Study Data from Tool Modifications

To collect data of sufficient precision, the survey tool was modified by adding additional instrumentation modules to measure the participant's relevance determination and the speed of that decision, as well as the location in the API reference topic the participant felt was most influential in their decision. The participants' relevance decisions were measured by adding to the survey tool a JavaScript code module that measured the participant's relevance decision and the time in milliseconds between the time that the topic became visible and the time that participants made their decisions. These values were sent to a web-service on a web server hosted by the researcher where they were recorded in a database.

The most influential part of the topic was measured by adding to the survey tool a JavaScript code module that prompted participants to click on the most influential part of the API reference topic. The module sent the location of the click to another web service that recorded it in the same database as the relevance decision data.

Study Data Preparation for Analysis

After the study ended, the data from the survey tool and the additional instrumentation modules were merged to form a single, normalized, composite data set that was compatible for analysis by SPSS.

RESULTS

Study Summary

Data were collected by the online survey tool from October 20 through December 20, 2014. This section describes the data collected and reports the results for each hypothesis. The results of exploratory investigations from the data follow ending with analyses of the spot interaction's location data and participant comments.

Raw Study Data

The Human-Subjects Division of the University of Washington approved the study for no more than 500 participants. Of the 436 participants who responded to the invitations, 221 completed the online survey. Of the 215 participants who did not complete the survey, 93 completed the demographic survey, practice task, and at least one study task, but not all four. A complete task is one in which the participant read the topic question, reviewed the topic, and answered the questionnaire after the topic. In total, 710 tasks were completed.

Table 9 lists each group invited to participate in the online survey. The total population of each group is listed in the column titled "Group size." Each invitation had a unique link to the online survey and each unique link was referenced by a Bitly link. Bitly provides a service that creates a short link that can be used in place of a complete Uniform Resource Locator (URL) or *link*. It also enables tracking the *click traffic* (people who click on the Bitly link), making it possible to observe the dropout rate. In this table, the column titled "Bitly clicks" lists the number of people who responded to the invitation and clicked on the link. Note that some participants bypassed the Bitly link and not every group had a Bitly link (indicated by N/A in the table). The column titled "Responses" lists the number of people who started the online survey and the column titled “Complete sessions” lists how many finished the online survey. The column in Table 9 titled “Valid tasks” lists the number of valid tasks that resulted from the corresponding invitation.

Table 9. Survey invitations and responses

Invitation group	**Group size**	**Bitly clicks**	**Re-sponses**	**Complete sessions**	**Valid tasks**
LinkedIn - Java Developer	258,238	121	56	25	59
LinkedIn - Open Source	118,374	75	29	15	54
LinkedIn - PHP Developers	100,000	86	33	13	35
LinkedIn - Android Developers	98,906	13	1	0	0
LinkedIn - Web Designer and HTML/CSS	44,622	4	1	1	4
LinkedIn - iPhone Developers	36,168	16	8	0	1
LinkedIn - Programmers and Developers	30,364	19	6	3	4
LinkedIn - Software Development Professionals Group	27,815	17	4	2	4
LinkedIn - PHP Developer Network	20,581	N/A	5	1	5
LinkedIn - iOS Developer Network	10,551	N/A	0	0	0
LinkedIn - Agile Technical Writers	6,521	0	0	0	0
LinkedIn - Technical Writers and SDK Doc	2,815	46	17	4	8
UW-CSE Facebook Group	1,643	26	18	7	24
LinkedIn - APIDocs Group	1,613	69	30	10	29
UW-HCDE	610	2	1	0	0
UTEP-CSE Facebook Group	536	39	21	11	27
Dub Contacts	235	35	25	15	53
North Carolina State - CSE	200	52	167	106	370
thePlatform devs	88	19	13	8	33
Facebook	36	0	0	0	0
LinkedIn - Twitter	22	19	0	0	0
Quora	Unknown	2	0	0	0
UW Tech Support	Unknown	0	0	0	0
Totals	**759,938**	**660**	**435**	**221**	**710**

At the end of the survey, participants who completed the online survey tool could register for a chance to receive a gift certificate. Registration for the gift certificate was not required and of the 221 participants who were given the opportunity to register, 181 registered and 41 gift certificates were awarded. The gift certificates were awarded randomly, with each registrant having a 1-in-5 chance to win. The target award rate was 20 percent, while the actual award rate was 22.7 percent.

Before beginning the analysis, the researcher cleaned and filtered the data to eliminate data that did not appear to be valid. In the first pass, 183 suspicious participant sessions were eliminated for the reasons listed in Table 10. After removing the suspicious participant sessions from the data set, the task responses were evaluated. Table 11 lists the number of individual tasks that were removed from a participant session that had at least one other valid task and the reasons for their removal.

At this point in the cleaning, the data set contained data from only participants with valid tasks. Participants were from 30 different countries and represented a diverse population. Table 12 lists the most common countries from which participants took the survey. Note that the table lists only those participant sessions with at least one valid task.

Table 10. Sessions removed to clean data set

Reason for removal	**Sessions discarded**
Did not evaluate a topic	177
Response time too short (< 2 seconds)	3
Spurious response	2
Task response time too long (>1,800 seconds)	1

Table 11. Tasks removed to clean data set

Reason for removal	**Tasks discarded**
Response time too short (< 2 seconds)	19
Did not evaluate a topic	17
Task response time too long (>1,800 seconds)	11
Repeated task	8

Table 12. Participant locations reported by survey tool

Participant's country	**Participant sessions**
United States	171
United Kingdom	13
India	11
Not reported	6
Canada	4
Spain	4
Other	44

The participants' native languages were equally diverse. Table 13 lists the native languages most frequently reported by participants. Of the 253 participants that completed at least one valid task, 109 were native English speakers, 117 were non-native English speakers with high English language proficiency, and the remaining 27 were participants who were not native English speakers and who did not have sufficiently English language proficiency. For this study, participants were determined to be proficient English speakers if they reported that they could speak English as a native as four or a five on a scale of one to five, or they rated their ability to speak English in a professional capacity as a five on a scale of one to five.

Table 13. Native languages reported by participants

Native language	Participants
English	109
Hindi	32
Tamil	16
Telegu	14
Spanish	8
Chinese	6
Italian	5
Kannada	5
Malayalam	5
Portuguese	5
Bengali	4
Other	44

Of the 253 participants in the sessions with valid tasks, 226 reported that they had programmed within the last year and they had an average 8.7 years of experience. The least experienced participant reported one year and the most experienced reported 41 years of experience writing computer software.

Final Data Set

The final data set consisted of the valid responses to tasks from the sessions of participants who had programmed within the last 12 months and were proficient in English. From the raw data set, 201 participants had valid sessions with 698 individual study tasks completed. Of the 201 qualified participants, 92 were native English speakers. The average programming experience of this group was slightly higher at 8.8 years of experience compared to the overall average of 8.7 years. Figure 22 illustrates the distribution of programming experience and Table 14 lists the completed tasks for each of the four study tasks in the final data set.

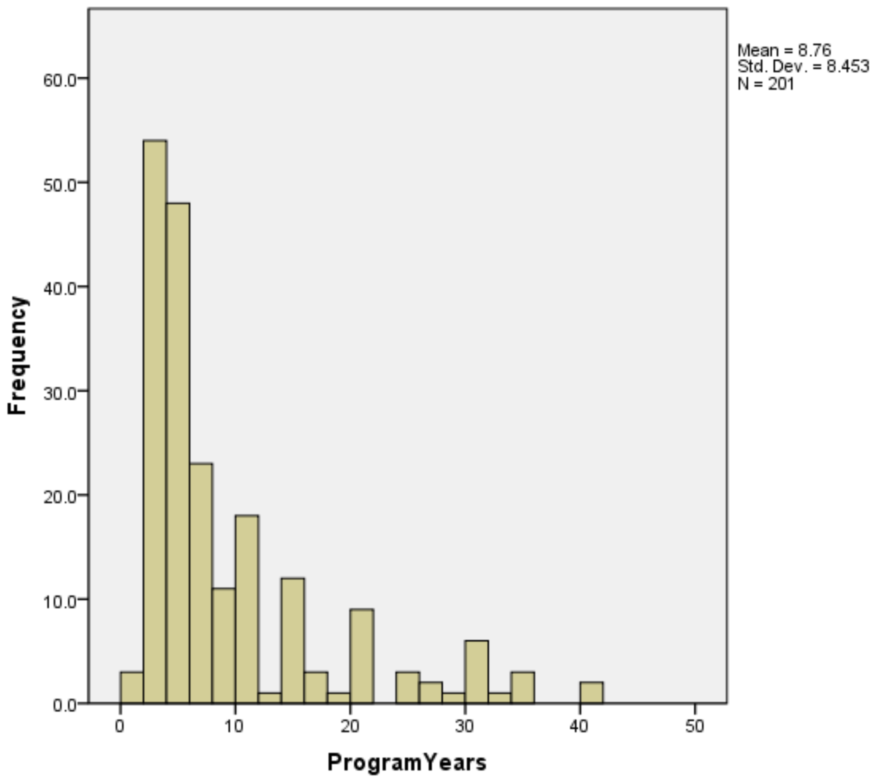


Figure 22. Histogram of programming experience (n = 201 participants)

Table 14. Summary of valid responses by task (n = 698 valid tasks)

Task	**Responses**
1	175
2	173
3	175
4	175

The study design included task randomization so that participants would not see the study tasks in the same sequence to minimize order effects. Table 15 lists the distribution of study tasks and where they appeared in the study protocol sequence—Task Sequence 1 describes the first task shown to a participant, Task Sequence 2, the second, and so on. A Pearson Chi-Square test determined that there was no statistically significant difference in where a task appeared in the sequence of tasks presented to participants, $X^2(9, N = 698) = 4.190, p = 0.898$. The lack of a significant difference in the frequency of when a task appeared in the task sequence confirms that the tasks were randomized such that no task appeared in a step of the task sequence any more often than any other task. In Table 15, the row labeled “Count” describes how often the task appeared in each step of the study protocol and the row labeled “Expected Count” describes how often the task should appear if the sequence was completely random.

Table 15. Distribution of tasks in the study protocol sequence

Study Task ID (API reference topic for that task)		**Task sequence**				
		1	**2**	**3**	**4**	**Total**
1 (Copy)	Count	43.0	46.0	45.0	41.0	175.0
	Expected Count	47.4	43.4	43.6	40.6	175.0
	% within Task ID	24.6%	26.3%	25.7%	23.4%	100.0%
	% within Task Sequence	22.8%	26.6%	25.9%	25.3%	25.1%
2 (Print)	Count	46.0	47.0	45.0	35.0	173.0
	Expected Count	46.8	42.9	43.1	40.2	173.0
	% within Task ID	26.6%	27.2%	26.0%	20.2%	100.0%
	% within Task Sequence	24.3%	27.2%	25.9%	21.6%	24.8%
3 (Select)	Count	48.0	43.0	44.0	40.0	175.0
	Expected Count	47.4	43.4	43.6	40.6	175.0
	% within Task ID	27.4%	24.6%	25.1%	22.9%	100.0%
	% within Task Sequence	25.4%	24.9%	25.3%	24.7%	25.1%
4 (Join)	Count	52.0	37.0	40.0	46.0	175.0
	Expected Count	47.4	43.4	43.6	40.6	175.0
	% within Task ID	29.7%	21.1%	22.9%	26.3%	100.0%
	% within Task Sequence	27.5%	21.4%	23.0%	28.4%	25.1%

Relevance-Decision Response

These next sections review the primary components of relevance as it was measured by the speed of and accuracy of participants’ relevance decisions in this study. First, the reliability of the time measurement used to measure the relevance-decision response time is reviewed and second, the accuracy of the participants' evaluations of the scenarios in the context of the study's tasks is reviewed.

Time Measurement Suitability

The measurement error of the JavaScript timer used to measure the relevance-decision response time is reviewed in the context of the experiment's results to determine if it is sufficiently reliable to use in this study. Resig (2008) and Zakaz (2011) describe the JavaScript timer used by the Survey Tool as having a nominal precision of 15.6 milliseconds and a worst-case measurement error of 55 milliseconds. The mean relevance-decision response time of the valid samples used in this study was 43.372 seconds (N = 698), with the longest time of 578.9 seconds and a shortest time of 1.143 seconds. With these measurements, the 15-millisecond nominal and 55-millisecond worst-case precision of the timer used to measure the decision response time represents a nominal precision of at least 1.3 percent and a worst-case accuracy of at least 5.2 percent, which seems acceptable for this study. The variation that human and human-computer interface factors introduces to influence the timer's accuracy is unknown; however, any such environmental bias or error represents the influence of participants' environment when they performed the tasks being studied. As such, in the context of this study, the variations that result from these environmental factors are not really errors but factors that contribute to the ecological validity of the study's results.

Relevance Decision Suitability

The accuracy of participants' relevance decisions is reviewed in the context of the study to evaluate whether these decisions are reasonable to use in evaluating the hypotheses. Each task included two scenarios: (1) a relevant scenario for which the topic had a many contextual effects and (2) a non-relevant scenario for which the topic had very few contextual effects. A correct response occurred when a participant's evaluation matched the scenario option selected by the survey tool. Overall, participants evaluated 520 tasks correctly and 178 incorrectly for an overall correct-response rate of 74.5 percent. No specific relevance rate threshold was indicated in the literature; however, in the context of an ecologically-valid population, the rate observed in this study seems sufficient and reasonable for this study.

Document Variation Validation and Effect of Task on Relevance-Decision Response Time

The previous sections described how the participants and tasks were validated for inclusion in the data analysis, how the randomization of the tasks in the study protocol was confirmed, and how the suitability of the measured values was evaluated. The next three subsections validate assumptions concerning the design of the study materials. The first two subsections evaluate and validate the assumption that API reference topic variations used in the study would be identifiably different to participants, and the third subsection assesses the assumption that the different tasks would not affect relevance-decision response time.

For each task in the study, the survey tool showed one of four different variations of the API reference topic that corresponded to the task. The API reference topic displayed to the participant could have one of two ICEC levels and one of two VDEC levels. The levels of each independent variable (IV) were designed to represent the API reference topic variations identified by Watson et al. (2013) and provided a set of variations that provided the study with ecological validity. To determine whether these variations appeared to be significantly different to the participants, the participants were asked to rate how many information concepts and visual design elements they observed in the topic that the study presented to them. Note that each participant only saw one variation so their assessments were not relative to any other topic variation; rather, they were based on the participant's own experience and judgment. This section evaluates their responses.

VDEC Level Validation

A Mann-Whitney U test was performed to compare how participants rated the number of visual elements they perceived in the two VDEC levels used in the API reference topics of the study. The Mann-Whitney U test found that the ratings of the topic variations with different levels of VDEC (*high* and *low*) were significantly different (U = 102,386.5, p = 0.000, N = 696), with the high VDEC variations having a higher mean rank of 479.47 (N = 322) than the low VDEC variations, which had a mean rank of 235.74 (N = 374). Figure 23 shows the rating distributions for the two levels of VDEC.

The significant difference ($p < 0.05$) between the mean ranks of the ratings for the VDEC levels confirms that participants found the VDEC levels to be different and consistent with the intended design.

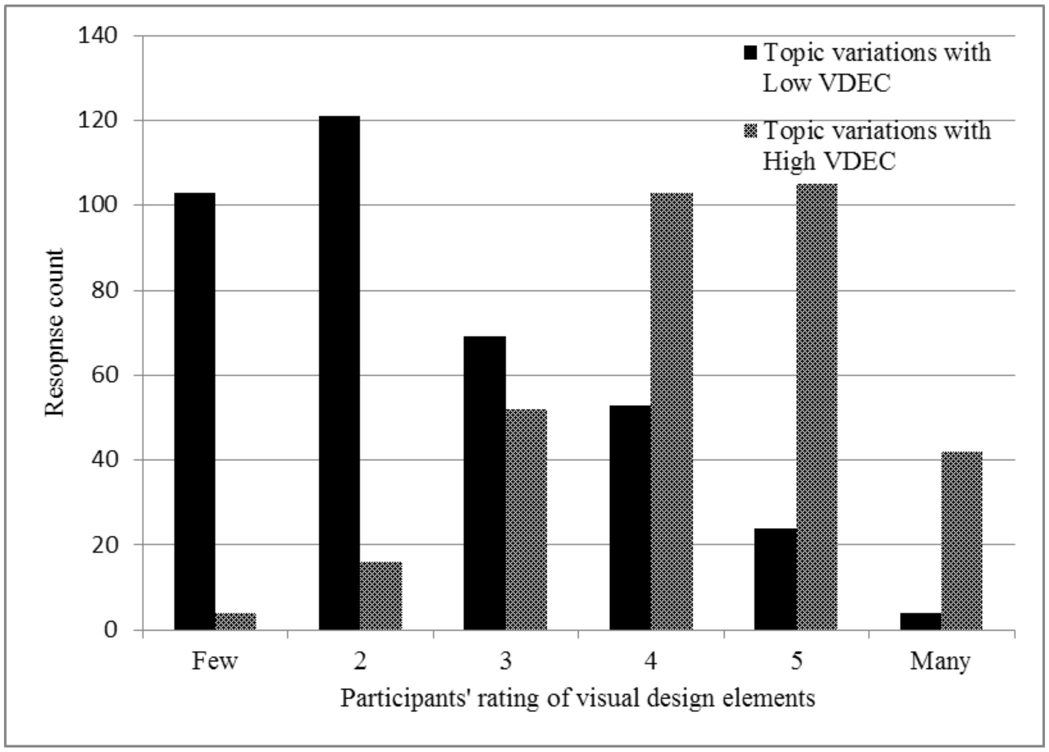


Figure 23. Histogram of reported visual design elements

ICEC Level Validation

A Mann-Whitney U test was performed to compare how participants rated the number of information concepts they perceived in the two ICEC levels. The Mann-Whitney U test found that the ratings of the topic variations with different levels of ICEC (*high* and *low*) were significantly different ($U = 82{,}767$, $p = 0.000$, $N = 697$), with the high ICEC variations having a higher mean rank of 412.15 ($N = 349$) than the low ICEC variations, which had a mean rank of 285.66 ($N = 348$). Figure 24 shows the distributions of the ratings for the two levels of ICEC.

The significant difference ($p < 0.05$) between the mean ranks of the ratings for the ICEC levels confirms that participants found the ICEC levels to be different and consistent with the intended design.

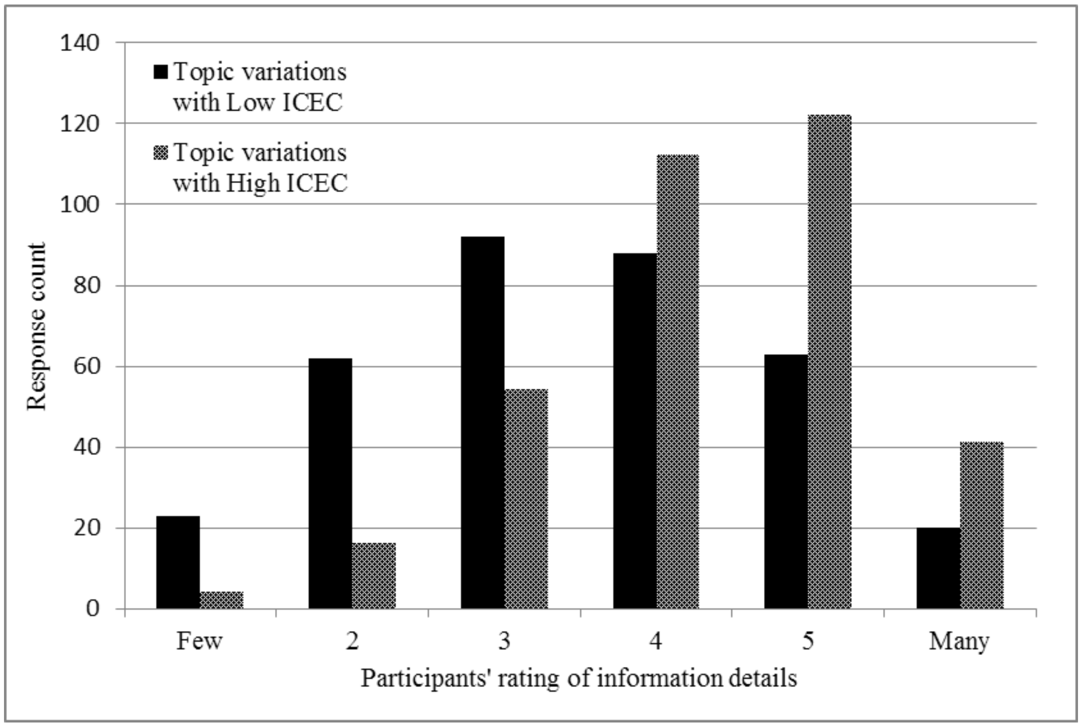


Figure 24. Histogram of reported information details

Influence of Task ID on Mean Relevance-Decision Response Time

Finally, a one-way ANOVA evaluated the effect of Task ID on relevance-decision response time to test whether the mean relevance-decision response time varied by task. No significant difference ($p < 0.05$) was observed, $F(3,697) = 1.035$, $p = 0.376$. The observed power of 0.282 ($\beta = 0.718$) and small effect size ($\eta^2 = 0.004$) suggest the probability of making a type-II error (where the null hypothesis is actually false and the mean relevance-decision response time actually varies by task) is almost as low as desired: 28.2 percent compared to 20 percent (Cohen, 1988). Table 16 lists the descriptive statistics for each task.

The lack of a significant difference in the relevance-decision response times between tasks allowed for the exclusion of task from further analyses that involved mean relevance-decision response time.

Table 16. Mean relevance-decision response time by task

Task ID (Topic)	Mean relevance-decision response time (seconds)	Std. Dev.	N
Task 1 (Copy)	37.428	41.279	175
Task 2 (Print)	47.375	52.395	173
Task 3 (Select)	44.438	67.345	175
Task 4 (Join)	44.296	54.759	175
All tasks	43.373	54.744	698

These results confirmed that the API reference topic variations were suitable for use in the study as planned and that the task did not significantly influence the mean relevance-decision response time. At this point, the study materials and experiment design used in the study had been validated and the data set contained data from only participants who were software developers with sufficient English language proficiency.

Results by Hypothesis

These results review the hypotheses that answer the research question: *What is the effect of visual-design element count (VDEC), unique information-concept element count (ICEC), and topic relevance to the information-seeking task versus ICEC on the speed and accuracy of software developers' assessment of an API reference topic's suitability to answer a task-oriented* question, and on their perceptions of the topic?

Hypothesis H1

H1: *Software developers will have a shorter mean relevance-decision response time in* topic variations with high vs. low VDEC *levels.*

A three-way ANOVA evaluated the effect of VDEC (between subjects), ICEC (between subjects), and topic relevance (within subjects) on the mean decision-response time and found no significant difference in the main effect of VDEC, $F(1,697) = 0.262$, $p = 0.609$. The results do not allow the null hypothesis to be rejected. The very small effect size ($\eta^2 = 0.000$) and low power (0.080) for this main effect suggest that differences in mean relevance-decision response time between VDEC levels would be difficult to detect. The three-way ANOVA used to evaluate this hypothesis was also used to evaluate hypotheses H3 and H5. Table 17 lists the main effects of this three-way ANOVA and the interaction between topic relevance and ICEC.

Table 17. Summary of 3-way ANOVA results: Effect of VDEC, ICEC, and Topic Relevance on Mean Relevance Decision-Response Time

Effects	**ANOVA**	**Power (1.0 - β)**	**Effect size**	**Level**	**Mean Relevance Decision-Response Time (sec.)**	**Std. Dev.**	**N**
VDEC	$F(1,697) = 0.262$, $p = 0.609$	0.080	Extremely small $\eta^2 = 0.000$	Low	42.638	57.267	374
				High	44.221	51.755	324
ICEC	$F(1,697) = 5.512$ $p = 0.019$	0.650	Moderate $\eta^2 = 0.008$	Low	38.790	44.515	349
				High	47.956	63.077	349
Topic Relevance	$F(1,697) = 2.186$ $p = 0.140$	0.315	Small $\eta^2 = 0.003$	Non-relevant	46.207	55.485	351
				Relevant	40.506	53.913	347
Topic Relevance and ICEC	$F(1,697) = 0.108$ $p = 0.743$	0.062	Extremely Small $\eta^2 = 0.000$	See Table 18			

Hypothesis H2

H2: *Software developers will determine the relevance of a topic correctly more often in* topic variations with high vs. low VDEC levels.

A Chi-Square test of independence was performed to examine the relation between the percentages of correct relevance assessments by VDEC level. No significant difference in the frequency of correct assessments was found between the VDEC levels, $X^2(1, N = 698) = 2.127$, $p = 0.145$. The results do not allow the null hypothesis to be rejected.

Hypothesis H3

H3: *Software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels.*

A three-way ANOVA evaluated the effect of VDEC (between subjects), ICEC (between subjects), and topic relevance (within subjects) on the mean decision-response time and found a significant difference in the main effect of ICEC, $F(1,697) = 5.512$ $p = 0.019$. The results allow the null hypothesis to be rejected. The mean relevance-decision response time of 38.790 seconds in the topic variations with a low ICEC was shorter than the 47.956 seconds in the topic variations with a high ICEC. While the power of 0.650 is relatively high (even though a power value of 0.80 is commonly used to reject the null hypothesis with confidence), the moderate effect size ($\eta^2 = 0.008$) suggests that it is reasonable to accept hypothesis H3. Table 17 lists the main effect of ICEC from the three-way ANOVA used for this hypothesis and mentioned above in the results for Hypothesis H1.

Hypothesis H4

H4: *Software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels.*

A Chi-Square test of independence was performed to examine the frequency of correct relevance evaluations by ICEC level and found no significant difference ($p < 0.05$) between the levels, $X^2(1, N = 698) = 2.443, p = 0.118$. The results do not allow the null hypothesis to be rejected.

Hypothesis H5

H5: *When API reference topic variations have the same relevance to the task question, software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels.*

A three-way ANOVA evaluated the effect of VDEC (between subjects), ICEC (between subjects), and topic relevance (within subjects) on the mean decision-response time and found no interaction between ICEC and topic relevance, $F(1,697) = 0.108, p = 0.743$. The power observed in the interaction was only 0.062 and the effect size was extremely small ($\eta^2 = 0.000$), suggesting that a significant interaction would be difficult to detect. Table 18 shows the ANOVA values and the descriptive statistics of the mean decision-response time for the two ICEC levels and the topic relevance levels.

Table 18. Hypothesis H5 ANOVA results of interaction of ICEC and topic relevance

ANOVA	**Power**	**Effect size**	**Topic relevance**	**ICEC level**	**Mean Relevance Decision-Response Time (sec.)**	**Std. Dev.**	**N**
$F(1,697) = 0.108$ $p = 0.743$	0.062	Extremely Small $\eta^2 = 0.000$	Non-Relevant	Low	41.209	46.789	177
				High	51.292	62.845	174
			Relevant	Low	36.301	42.039	172
				High	44.639	63.312	175

Hypothesis H6

H6: *When API reference topic variations have the same relevance to the task question, software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels.*

A Chi-Square test of independence was performed to evaluate the relation between the percentages of correct evaluations by ICEC for each level of topic relevance. No significant difference in the frequency of correct evaluations was found between levels of ICEC in the tasks with topics that were not relevant to the scenario, $X^2(1, N = 351) = 0.821$, $p = 0.365$. However, significant differences were found in the frequency of correct evaluations between levels of ICEC in the tasks with topics that were relevant to the scenario, $X^2(1, N = 347) = 12.962$, $p = 0.000$. When topics were relevant to the scenario, participants correctly evaluated the relevance of the topics with the low ICEC in 75 percent of the cases ($N = 172$), which is less than the 89.7 percent of the topics ($N = 175$) with a high ICEC in which participants correctly evaluated the topic's relevance. The results do not allow the null hypothesis to be rejected when the topic is not relevant to the question, but they do allow it to be rejected when the topic is relevant to the question.

Table 19 lists the Chi-Square values and the percent of correct evaluations in each level of ICEC and topic relevance examined for Hypothesis H6

Table 19. Hypothesis H6—Summary of Chi-Square results

Topic relevance level	**Pearson Chi-Square (X^2) value**	**ICEC level**	**Correct responses**	**N**
Non-relevant	$X^2(1, N\ 351) = 0.821$, $p = 0.365$	Low	68.9%	177
		High	64.4%	174
Relevant	$X^2(1, N = 347) = 12.962$, $p = 0.000$	Low	75.0%	172
		High	89.7%	175

Hypothesis H7

H7: Software developers will give higher credibility ratings to API reference topic variations with high vs. low VDEC levels.

A Mann-Whitney U test was performed to compare the distributions of the participants' credibility ratings for each level of VDEC in the API reference topics. The results reveal that the two levels of VDEC (*high* and *low*) were rated significantly differently ($U = 82{,}419.0$, $p = 0.000$, $N = 697$). The mean rank of the variations with a high VDEC was 417.17 ($N = 323$)—higher than the mean rank of 290.13 ($N = 374$) for those with a low VDEC. Figure 25 shows the response distributions of the two levels of VDEC. The results allow the null hypothesis to be rejected.

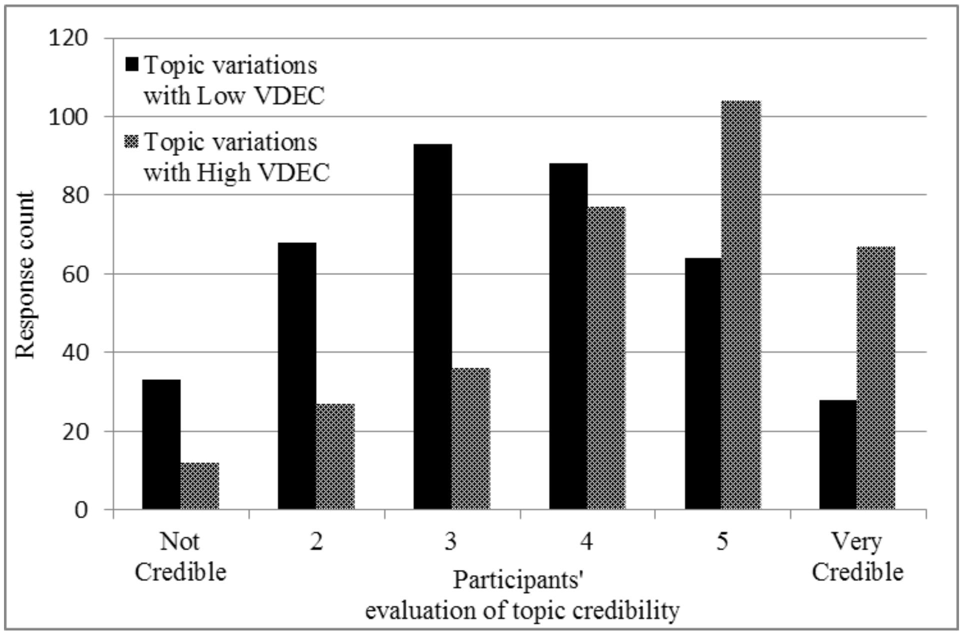


Figure 25. Participants' evaluation of topic credibility by VDEC (*N* = 697)

Results of Exploratory Investigations

The next section discusses the results of exploratory investigations related to some of the perceptual measures.

Exploratory Investigation 1

The first exploratory investigation evaluated whether software developers rated the topic variations with a high ICEC as more credible than the topic variations with a low ICEC. The literature cited in this study indicated a link between VDEC and credibility (described in H7), but not specifically between ICEC and credibility. Because this study varied both VDEC and ICEC, the variations of ICEC were evaluated to assess their effect on the participants’ assessments of credibility.

A Mann-Whitney U test was performed to compare the distributions of the participants' credibility ratings by ICEC level and revealed that participants rated the API reference topic variations with different levels of ICEC (*high* and *low*) significantly different, ($U = 92{,}910.0$, $p = 0.000$, $N = 697$). The topic variations with a high ICEC level had higher mean rank of 389.87 ($N = 349$) than the topic variations with a low ICEC level, which had a mean rank of 308.01 ($N = 348$). Figure 26 shows the participants' responses by the ICEC of the topic variations they assessed. The software developers who participated in this study rated the topics variations with a high ICEC as being more credible than the topic variations with a low ICEC.

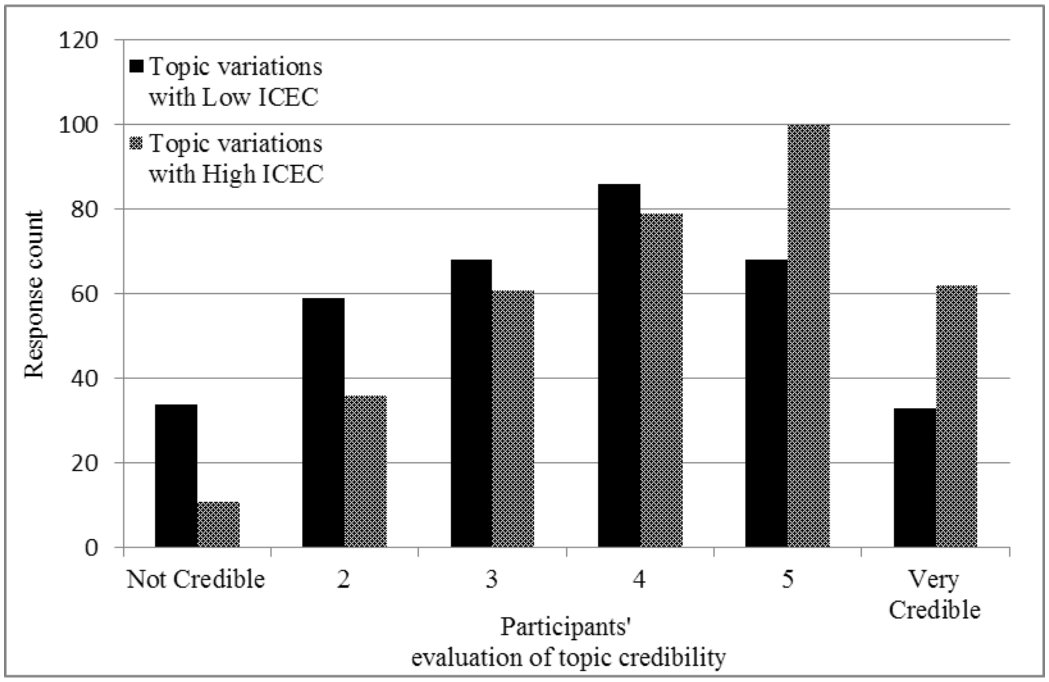


Figure 26. Participants' evaluation of topic credibility by ICEC ($N = 697$)

Exploratory Investigation 2

The second exploratory investigation evaluated whether software developers rated the topic variations with a high VDEC as having a more professional appearance than the topic variations with a low VDEC. The literature cited in this study did not relate VDEC variations to professionalism, per se, so the levels of VDEC were evaluated for their effect on the participants' rating of the topic's professional appearance.

A Mann-Whitney U test was performed to compare the distributions of the participants' appearance ratings by VDEC level and revealed that participants rated the API reference topic variations with different levels of VDEC (*high* and *low*) significantly different, ($U = 92{,}910.0$, $p = 0.000$, $N = 697$). The topic variations with a high VDEC level had a higher mean rank of 449.65 ($N = 323$) than the topic variations with a low VDEC level, which had a mean rank of 262.08 ($N = 374$). Figure 27 shows the frequency of participant ratings by the topic's VDEC. Participants rated the topic variations with a high VDEC as having a more professional appearance than the topic variations with a low VDEC.

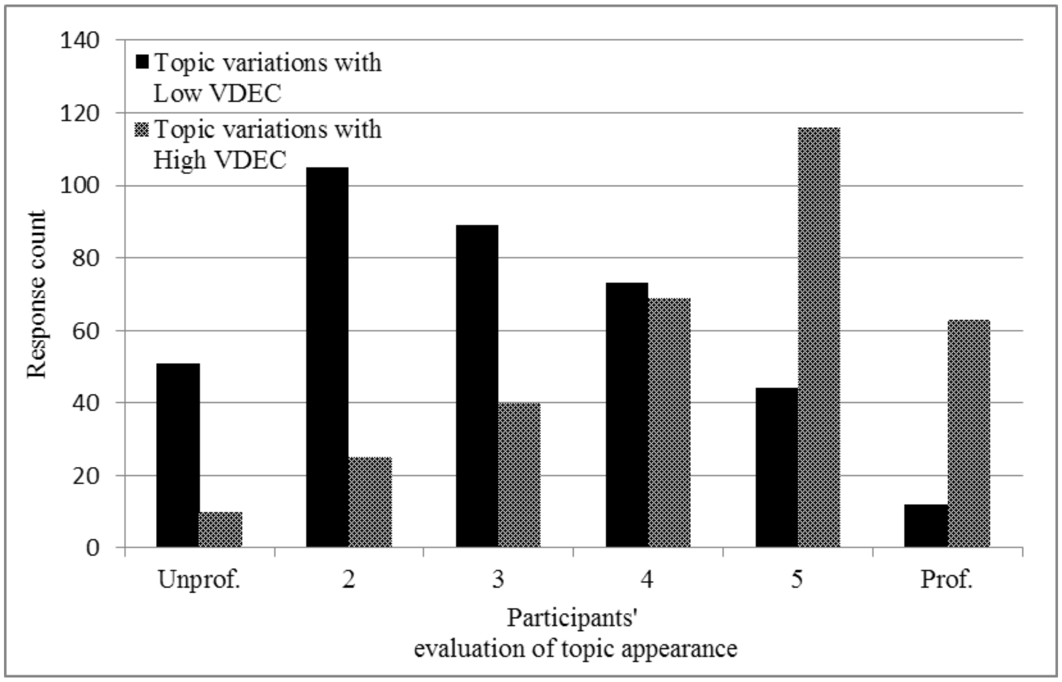


Figure 27. Participants' evaluation of topic professional appearance by VDEC ($N = 697$)

Exploratory Investigation 3

The third exploratory investigation evaluated whether software developers would find the topic variations with a high ICEC to have a more professional appearance than the topic variations with a low ICEC.

A Mann-Whitney U test was performed to compare the distributions of the participants' appearance ratings by ICEC level and revealed that participants rated the API reference topic variations with different levels of ICEC (high and low) significantly different, (U = 69,769.0, p = 0.001, N = 697). The topic variations with a high ICEC level had a higher mean rank of 374.91 (N = 349) than the topic variations with a low ICEC level, which had a mean rank of 323.01 (N = 348). Figure 28 shows the participants' responses by the ICEC of the topic variation they evaluated.

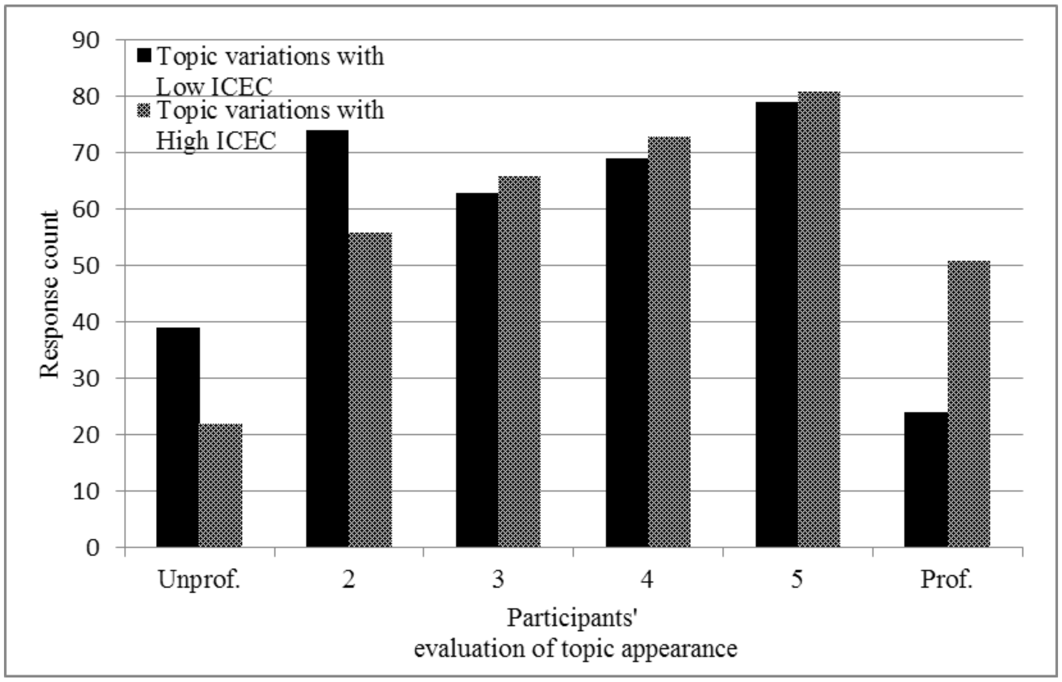


Figure 28. Participants' evaluation of topic professional appearance by ICEC

The software developers who participated in this study rated the topics variations with a high ICEC as having a more professional appearance than the topic variations with a low ICEC.

Exploratory Investigation 4

The fourth exploratory investigation evaluated whether software developers who rated topics as credible also rated those topics as having a professional appearance. A significant correlation was observed between the participants' evaluations of credibility and professional appearance (Pearson's R = .775, *p* = 0.000, *N* = 697). This study, however, does not identify a reason for this correlation.

Exploratory Investigation 5

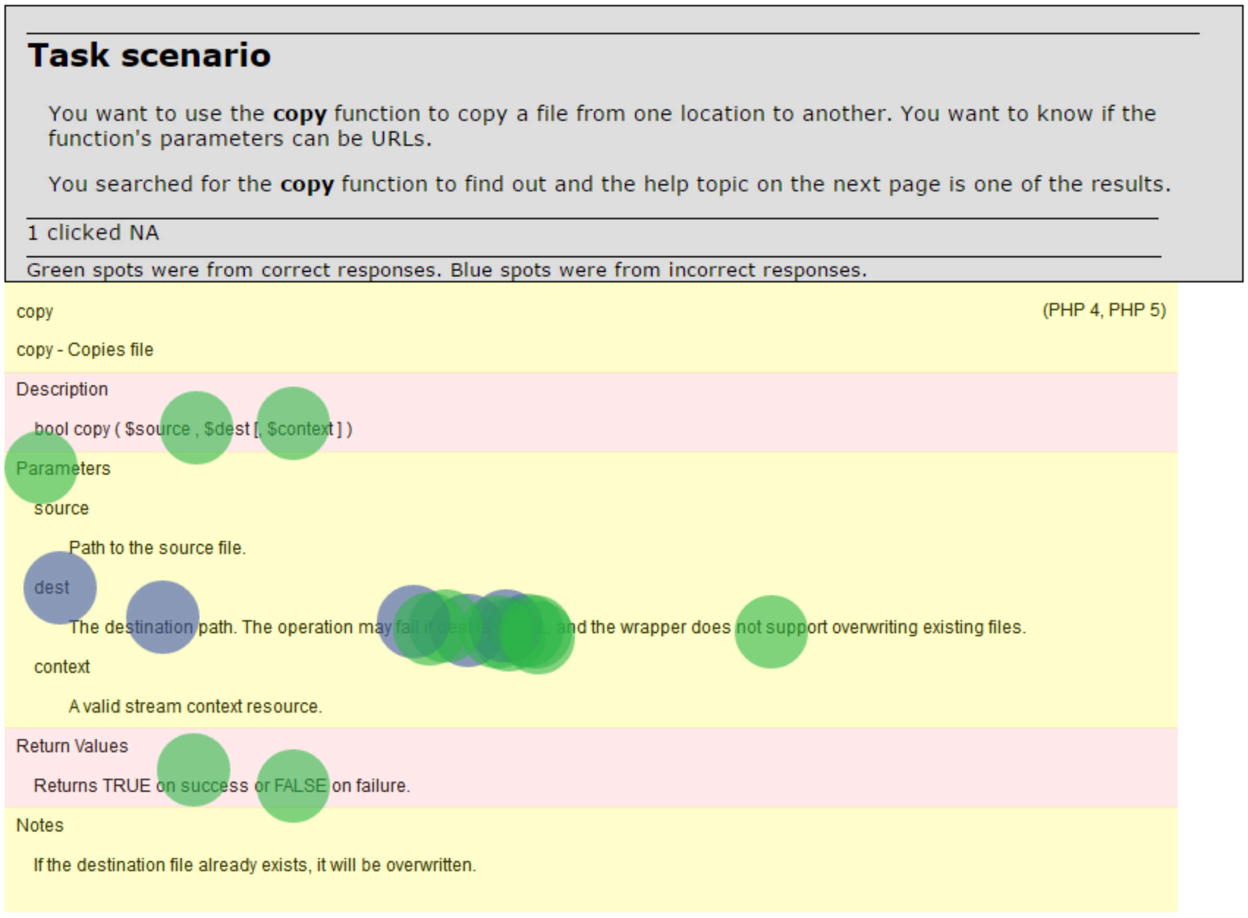
The fifth exploratory investigation evaluated whether a programmer's experience related to their relevance-decision response time and found that software developers reporting more experience writing software made their relevance decisions slightly faster than those reporting less did. A significant, slightly inverse correlation was observed between the programmer's experience and their relevance-decision response time, (Pearson's R = -0.085, *p* = 0.024, *N* = 698).

Influential Topic Location Data Analysis

The spot interaction and questions asked in the questionnaires that followed each task were used to explore additional perceptions of the API reference topic variations. Influential topic location data were collected to understand where, or whether, participants found the information requested by the task's scenario. It took participants an average of 15.8 seconds to respond to this question (Std. Dev. = 10.2, *N* = 589).The results are reported in this section as descriptive statistics and frequencies and the **Influential Spot Location Maps** section in the Appendix contains a complete set of the influential location responses from all topic variations.

Influential Location Data Analysis Overview

Table 20 summarizes the mean vertical locations and the standard deviations of the influential locations in each topic reported as image pixels counted from the top of the topic image. The specific mean value of the influential location is useful only within the specific topic variation because the content placement can vary between topic variations. The standard deviation, however, provides a numerical measure of how concentrated the responses were. The smaller the standard deviation, the tighter the responses cluster around the mean. For example, the first row in Table 20 summarizes the responses from the low VDEC, low ICEC topic variation that were reviewed after being shown a relevant scenario from Task 1. This row shows a mean Y location of 224.28 (*N* = 19) with a standard deviation of 62.19. The small standard deviation indicates that the responses were close to the mean Y position. Figure 29 shows the locations plotted on the corresponding API reference topic. This result is in contrast to the responses from the high VDEC, high ICEC topic variation of the relevant scenario from Task 1, which show a mean Y position of 535.62 (*N* = 23) and a much larger standard deviation of 230.98. In the latter case, the larger standard deviation indicates that the responses were dispersed more widely than in the first example. Figure 30 show the locations plotted on the corresponding API reference topic.



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest ... and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

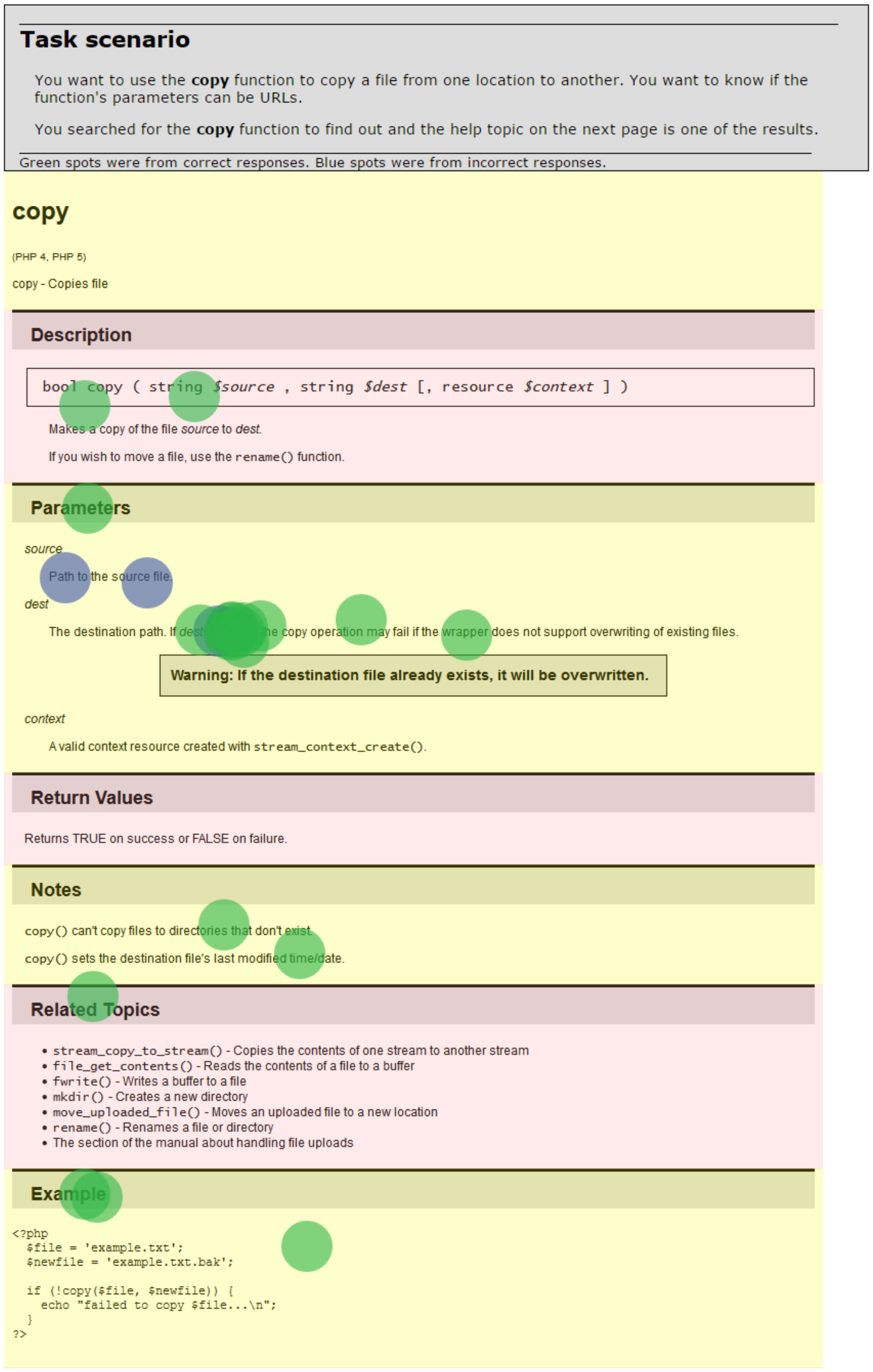
If the destination file already exists, it will be overwritten.

Figure 29. Influential location data for topic with small mean location standard deviation

Table 20. Summary of mean vertical coordinate of influential location by topic variation

Task ID (topic)	**Topic relevance**	**ICEC level**	**VDEC level**	**Mean Y position (pixels from top)**	**Std. Dev.**	**N**
1 (Copy)	Relevant	Low	Low	224.28	62.19	18
			High	343.44	73.03	27
		High	Low	288.85	67.51	24
			High	535.62	230.98	23
	Non-relevant	Low	Low	132.52	101.07	18
			High	286.60	195.46	10
		High	Low	311.38	255.64	22
			High	249.51	183.66	15
2 (Print)	Relevant	Low	Low	183.85	75.92	21
			High	347.36	139.39	15
		High	Low	321.36	224.86	24
			High	545.30	205.24	10
	Non-relevant	Low	Low	174.82	96.77	18
			High	320.87	105.73	20
		High	Low	311.58	213.67	20
			High	600.30	264.64	23
3 (Select)	Relevant	Low	Low	208.22	194.01	21
			High	472.39	226.57	18
		High	Low	263.05	241.93	13
			High	405.02	204.28	21
	Non-relevant	Low	Low	159.56	168.38	21
			High	243.43	150.19	25
		High	Low	105.02	102.10	31
			High	245.34	196.73	15
4 (Join)	Relevant	Low	Low	203.02	98.10	23
			High	315.24	127.37	19
		High	Low	327.40	189.80	35
			High	607.66	304.86	22
	Non-relevant	Low	Low	90.92	55.56	17
			High	263.09	116.03	16
		High	Low	365.31	207.09	12
			High	498.53	351.52	15

To compare variations in ICEC and VDEC, the influential locations were interpreted as the topic sections into which they fell. In this way, the participants’ responses related to the topic sections, which were consistent across variations—insofar as the variations used the same topic sections. With the influential-location data described as a specific location in the topic, it was only meaningful within the individual topic variation, although the standard deviation could be compared across topics and topic variations. By normalizing the locations to their corresponding topic sections, the response locations could be compared across topics and topic variations.



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( string $source , string $dest [, resource $context ] )
```

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the `rename()` function.

Parameters

source

Path to the source file.

dest

The destination path. If *dest* [illegible] the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with `stream_context_create()`.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

`copy()` can't copy files to directories that don't exist.

`copy()` sets the destination file's last modified time/date.

Related Topics

- `stream_copy_to_stream()` - Copies the contents of one stream to another stream
- `file_get_contents()` - Reads the contents of a file to a buffer
- `fwrite()` - Writes a buffer to a file
- `mkdir()` - Creates a new directory
- `move_uploaded_file()` - Moves an uploaded file to a new location
- `rename()` - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
  $file = 'example.txt';
  $newfile = 'example.txt.bak';

  if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
  }
?>
```

Figure 30. Influential location data for topic with large mean location standard deviation

Table 21 shows the response location data interpreted as the topic sections in which the spots were located and then how often a response was made in each topic section of the document. In this table, the response data are described by the number of responses in each topic section. The NA column represents the cases where the participant selected “Not applicable” and did not indicate a location on the topic. The NA values are not included in the calculations of the mean Y position, standard deviation, or the value of N listed in Table 20.

The distribution of response frequencies in Table 21 corresponds to the standard deviation values listed in Table 20. For example, the first row in Table 20, which shows the frequency of responses for the low VDEC, low ICEC topic variation of the relevant questions from Task 1, the responses clustered around the Parameters section of the topic. There were 14 responses placed in the Parameters section and 2 responses in each of the adjacent sections, the “Description” and “Return Values” sections, of the topic. This tight grouping corresponds to the small standard deviation of mean Y position for that same topic variation listed in Table 20. Contrast this with the responses from the high VDEC, high ICEC topic variation of the relevant questions from Task 1. As Table 20 shows, responses were made in every topic section except the title. This distribution corresponds to the much larger standard deviation of 230.98 listed in Table 20 for the same topic variation.

Table 21. Summary of most influential API reference topic section selections

				Spots in topic section							
Task	**Topic relevance**	**ICEC level**	**VDEC level**	**NA**	**Title**	**Description**	**Parameters**	**Return Values**	**Notes**	**Related Topics**	**Examples**
1 (Copy)	Relevant	Low	Low	1	0	2	14	2	0	0	0
			High	0	0	5	22	0	0	0	0
		High	Low	0	0	1	22	0	0	1	0
			High	0	0	2	15	0	2	1	3
	Non-relevant	Low	Low	5	8	4	5	1	0	0	0
			High	7	3	3	2	0	2	0	0
		High	Low	3	1	11	3	0	3	0	3
			High	2	6	5	3	0	1	0	0
2 (Print)	Relevant	Low	Low	5	0	12	5	0	2	0	0
			High	3	0	10	1	1	2	0	0
		High	Low	0	0	13	0	0	7	0	3
			High	1	0	3	0	0	6	0	1
	Non-relevant	Low	Low	3	0	11	2	2	3	0	0
			High	4	1	7	11	0	1	0	0
		High	Low	4	0	8	4	0	2	3	3
			High	4	1	4	3	0	7	3	5
3 (Select)	Relevant	Low	Low	2	12	0	6	1	1	0	0
			High	1	4	1	7	3	2	0	0
		High	Low	0	3	5	2	0	3	0	0
			High	1	1	12	6	1	1	0	0
	Non-relevant	Low	Low	3	9	5	3	1	3	0	0
			High	0	9	7	7	2	0	0	0
		High	Low	4	15	11	4	0	1	0	0
			High	1	6	6	2	0	1	0	0
4 (Join)	Relevant	Low	Low	1	0	12	2	8	1	0	0
			High	1	0	12	2	5	0	0	0
		High	Low	1	1	10	14	4	0	1	5
			High	0	0	6	7	4	0	0	5
	Non-relevant	Low	Low	6	7	7	2	0	0	0	0
			High	6	3	6	5	1	0	0	0
		High	Low	2	3	0	2	5	0	0	2
			High	1	4	3	2	3	0	0	3

Statistical Analysis of Spot Data

Grouping the spot responses by the topic section in which they were located, as Table 21 shows, enabled them to be analyzed for statistically significant differences. Chi-Square tests of independence were performed to examine the relation between the frequency of topic-section selection and VDEC levels and ICEC levels. No significant difference in the frequency of topic-section selection was found between the levels of VDEC, $X^2(7, N = 626) = 4.704, p = 0.696$. However, a significant difference in the frequency of topic-section selection was found between the levels of ICEC, $X^2(7, N = 626) = 52.092, p = 0.000$.

The different levels of ICEC appeared to influence where people got their answer more so than the different levels of VDEC did. The frequency with which participants selected different topic sections as most influential in making their relevance decisions did not significantly differ between VDEC levels, yet it did significantly differ between ICEC levels. Some of the differences found between the levels of ICEC can be attributed to the fact that the API topic variations with a high ICEC included more topic sections than those with a low ICEC—specifically the “Related Topics” and “Examples” API reference topic sections, which were only available in the API reference topics with a high ICEC. However, significant differences between ICEC levels were still found after removing those sections from the analysis, $X^2(4, N = 581) = 9.992, p = 0.041$.

Figure 31 shows the frequency that each topic section was selected as being the most influential by VDEC level and Figure 32 shows the frequency that each topic section was selected as being the most influential by ICEC level.

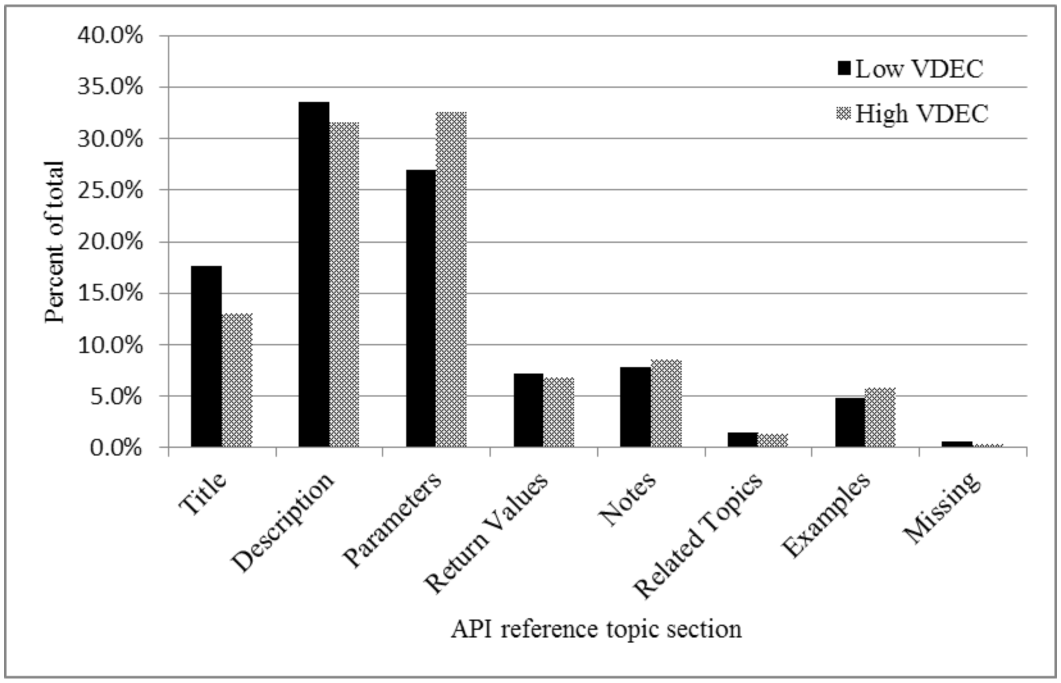


Figure 31. Influential topic section frequency by VDEC

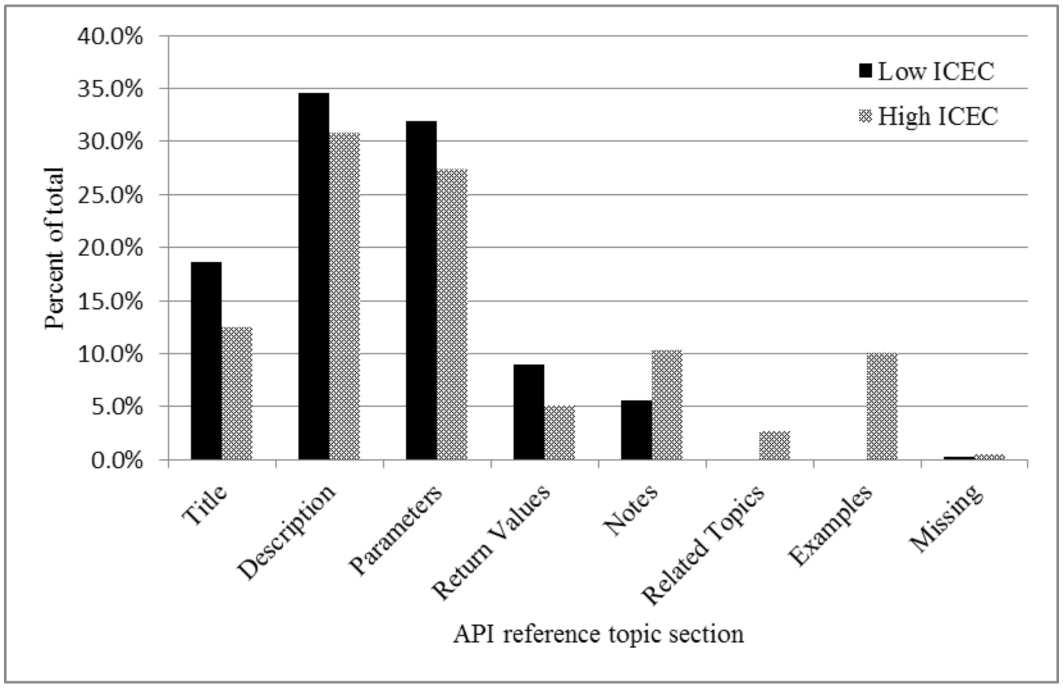


Figure 32. Influential topic section frequency by ICEC

While these results are interesting and indicate an effect of ICEC on the participants' perceptions of the API reference topics, the sample size of this study does not allow further statistical analyses into the specific differences. A more detailed analysis requires a larger sample size to evaluate specific topic variations.

Additional Spot Data Analysis Methods

The response spot data can be analyzed in these two different ways to provide a sense of where participants found the information they used to make their relevance decisions in an unmoderated, Internet-based study. Additionally, Figure 33 through Figure 36 provide graphical representations of the data tabulated in Table 21. The response clusters are represented by the length of the bar for the section. The bar charts show the responses for relevant topics next to the responses for non-relevant topics for the same topic variation.

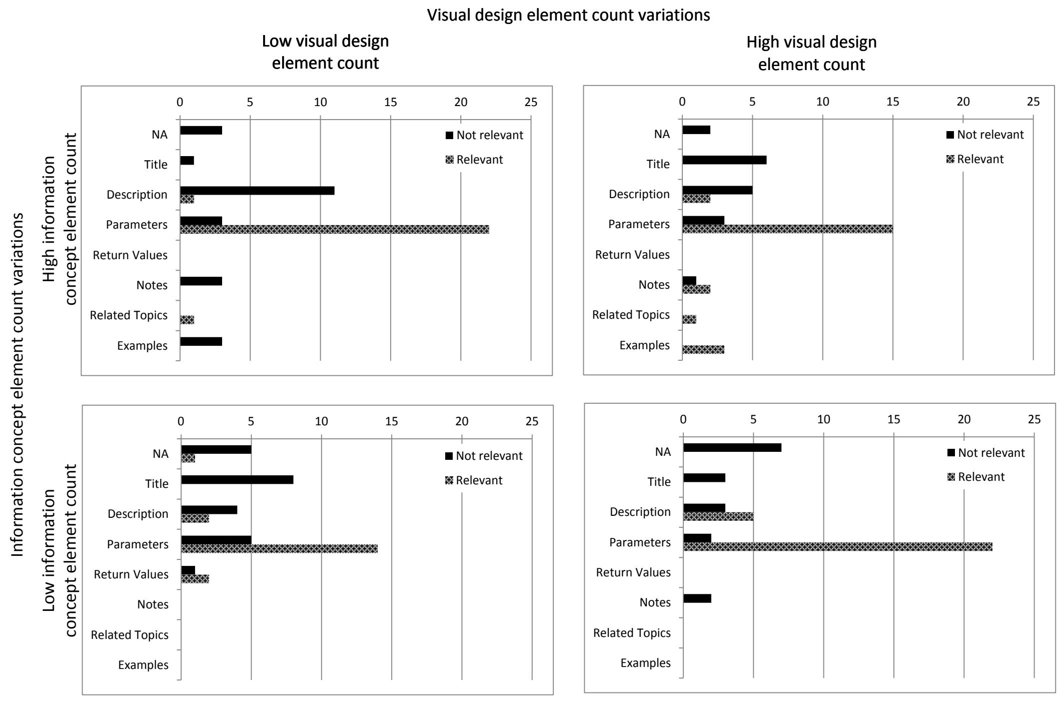


Figure 33. Task 1 Influential spot locations

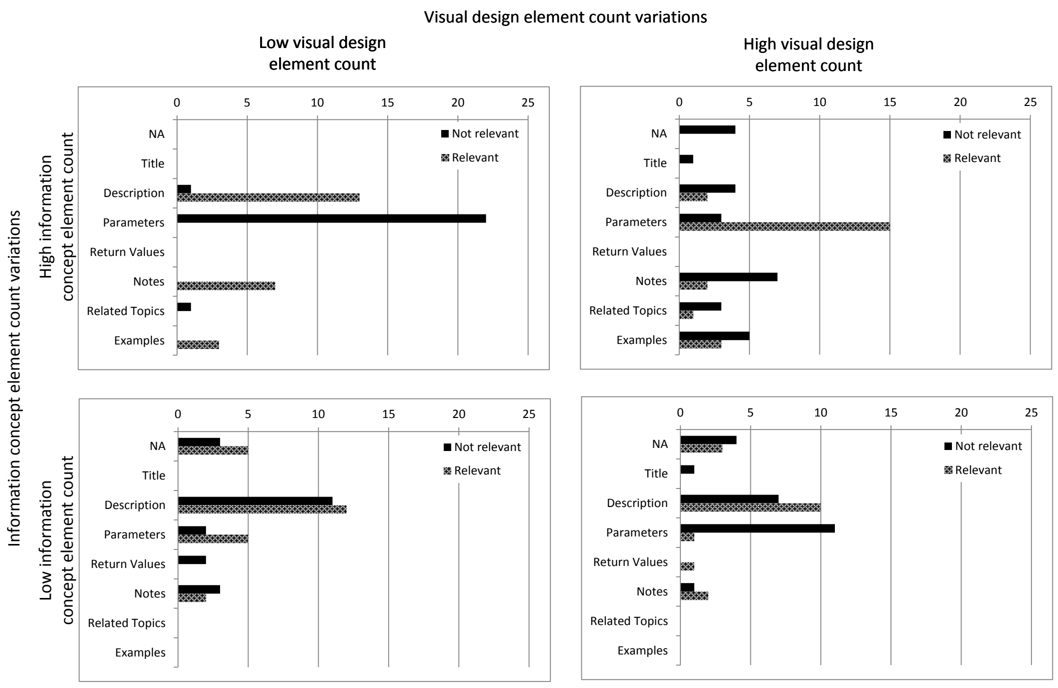


Figure 34. Task 2 Influential spot locations

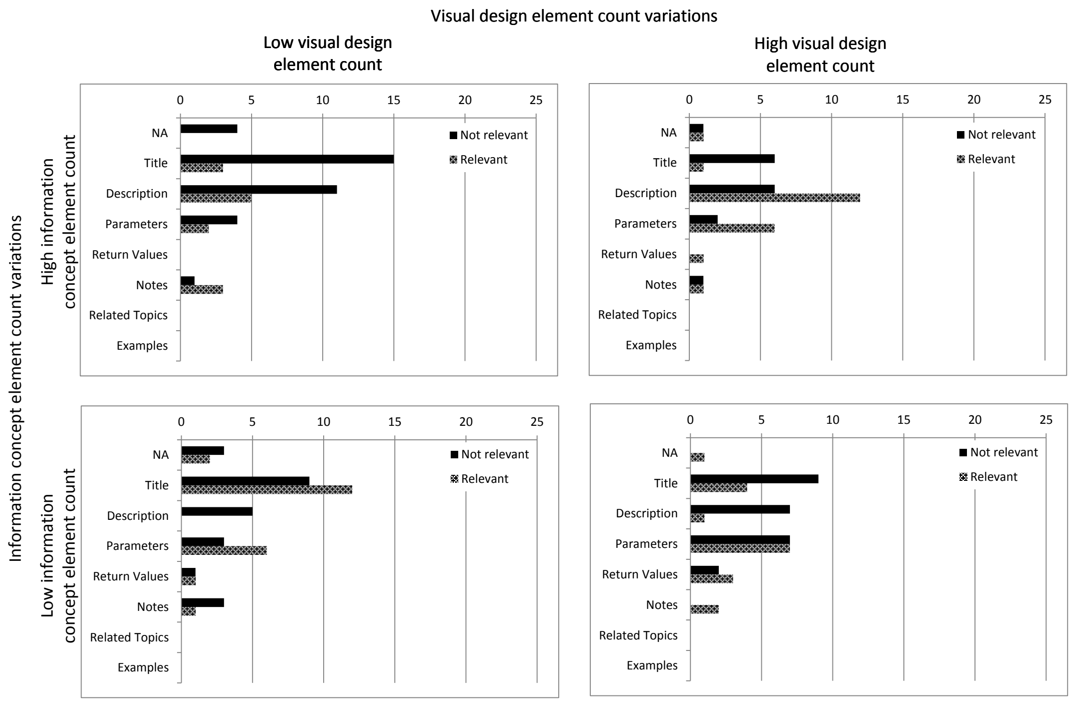


Figure 35. Task 3 Influential spot locations

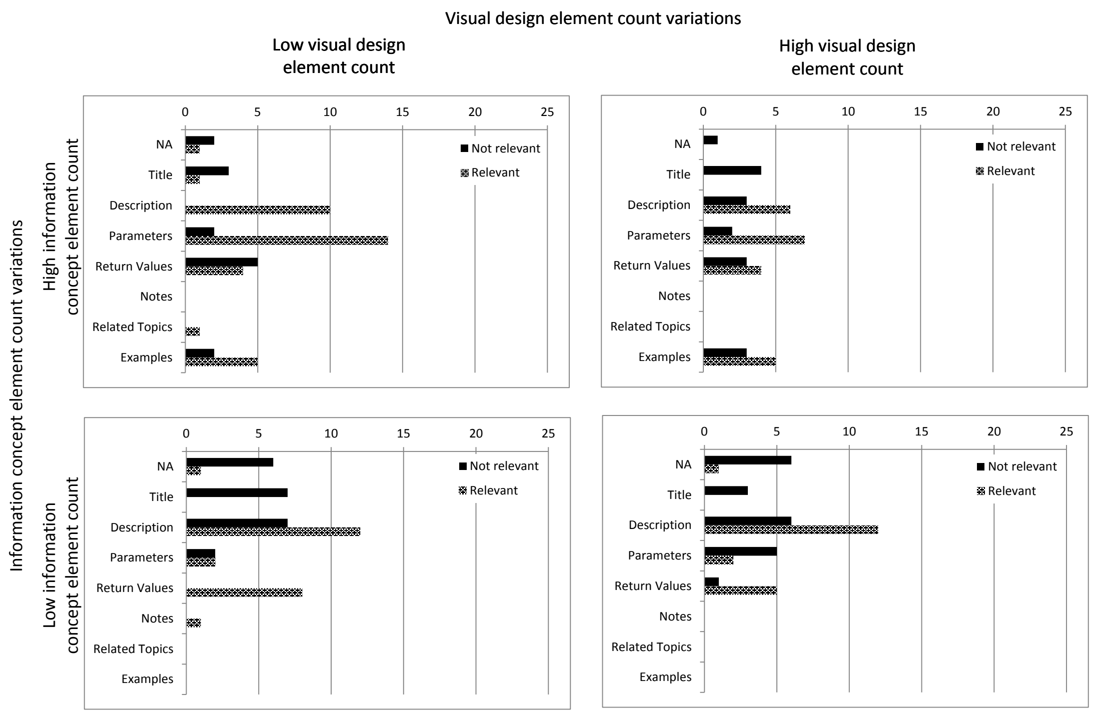


Figure 36. Task 4 Influential spot locations

Comment Analysis

A free-form comment field was included as a part of each post-task questionnaire. In addition to the comment text provided by the participants, comments were reviewed by the researcher to examine any patterns regarding three concepts:

- **Sentiment**: What type of sentiment did the comment express?
- **Suggestion**: Did the comment offer a suggestion to improve the topic or the study?
- **Problem**: Did the comment reflect a problem the participant had with the topic or the study?

A quick review of the sentiment expressed in the comments revealed that most were negative or neutral, and only a small number expressed a positive sentiment. No other relationship between the comments and other factors was observed.

Some examples of the comments submitted to the study are listed here.

In response to Task 1 with a topic that had a high ICEC, a low VDEC, and a relevant topic, participant #172 commented, "*If this study is about the format of the API documentation rather than the contents, this is silly. APIs seem unprofessional when the documentation is patchy, inconsistent, or wrong, not because the moron writing the CSS sucks.*" This comment was considered to express a negative sentiment.

In response to Task 2 with a topic that had a low ICEC, a high VDEC, and a relevant topic, participant #279 commented, "*Again, the text formatting detracts from communicating the information clearly. The topic is formatting heavy, but content light.*" This comment was considered to express a negative sentiment.

In response to Task 3 with a topic that had a high ICEC, a low VDEC, and a relevant topic, participant #299 commented, "*The topic is well written and easy to understand. The formatting of the code and text makes it a little hard to read since each element uses the same font.*" This comment was considered to express a positive sentiment.

DISCUSSION

The study advanced the state of the art by measuring the effect of important aspects of API reference topic design on readers and demonstrating the effectiveness of several Internet-based, remote user-assessment tools. This section discusses how the results and other aspects of the study inform future researchers and authors of API reference topics. It starts by reviewing the experimental design and the threats to validity to ensure that the study provided valid data and results for further discussion. It follows with a review of how the results address the research question and then discusses the implications of the results on aspects of API reference topic design and authoring. Finally, it explores how the data collected by the influential spot interaction can be used.

Review of the Experimental Design

Before considering the implications of any of the study's results and assessing the relationship of the results to the initial research question, the design of the experiment is examined and validated. The key aspects of this study on which the subsequent implications rest are:

1. The use of an unmoderated, remote study to evaluate how document variations influenced performance
2. The variations of the API reference topics evaluated by the participants
3. The study population and sample size
4. The hypotheses tested in the study

The sections that follow discuss each of these aspects of the study.

Use of an Unmoderated, Remote Study to Evaluate Document Performance

Unmoderated, Internet-based, remote user-experience studies provide low-cost access to a large, diverse, and, in this case, ecologically valid population. Further, remote studies have been used frequently to test and collect user-experience data. The study protocol implemented in the survey is consistent with earlier task-oriented Internet-based studies (Bartell and Spyridakis, 2012; Spyridakis et al., 2005; Dillman, 2008). Collecting timed response data is also common in Internet-based studies.

Environmental variables can influence unmoderated, remote studies. While these influences can degrade a survey instrument's precision, they contribute to the ecological validity of the study. At the same time, the degradation in precision can require a larger population to identify small differences in performance; however, this is easier to obtain in an Internet-based study than in a local, moderated study.

The influence of environmental variables is an important aspect of the study's ecological validity and of interest in the study. Because the API reference topics studied by this research would be used in a variety of environments, using a remote study makes it possible to test the variations in as many of those environments as possible. This approach, however, runs the risk of having the environmental aspects obscuring the effects of the independent variables manipulated by the experiment. In this context, however, it is important for practitioners to know the effect of topic variations in the customers' contexts to assess the value of these variations to the customer, even if the sources of the environmental influences cannot be identified.

Because the relevance-decision response time is the measured variable used in hypotheses H1, H3, and H5, the measurement error of the timer should be sufficiently small for this application. If the worst-case measurement error of the decision-response time measurement was 55 milliseconds, a difference of 0.55 seconds would be 10-times the worst-case measurement error and still be barely perceptible to a reader. Because, in the worst case, the timer is sufficiently accurate and precise to detect differences that are barely perceptible to the reader, it is sufficiently precise for this study.

After considering the effects of these factors, the researcher believed that the unmoderated, remote user-experience study was a suitable compromise between the type and precision of the data collected, study cost, and participant access and convenience.

Variations of the API Reference Topics Evaluated by the Participants

The API reference topic variations that resulted from the two levels of VDEC and two levels of ICEC were designed to reflect API reference topic variations encountered in real-world API reference documentation as observed in earlier studies. Using API reference topic variations that occur in real-world documentation was a way to maintain ecological validity in the study. The lack of many significant differences in the results could suggest that the VDEC and ICEC levels did not vary sufficiently to produce a measurable difference. However, participants clearly detected the VDEC and ICEC variations in their assessments of the topics' visual design and information content. Further, the topic variations are consistent with sample API reference topics found in Watson et al. (2013). That the variations of VDEC did not produce a significant difference in the mean relevance-decision response time or the influential location in the topic would, therefore, suggest that varying the VDEC levels, within an ecologically-valid range, does not have a significant effect on the mean relevance-decision response time or the most influential location to the question. If including data from more participants resulted in a significant difference between these values, the difference would likely not be very meaningful.

Study Population and Sample Size

The participants in this study were a convenience sample of people who belonged to special-interest groups related to software development. As such, this population was not a random sample of any specific population; however, the demographics of participants who provided valid data indicate that a diverse set of software developers participated in the study. Table 9 in the **Results** section shows that study invitations were sent to 23 groups representing 759,938 people with interest or experience in software development. Participants from 30 countries around the world represented software developers reporting up to 41 years of experience writing software.

For the performance tests (Hypotheses H1, H3, and H5), it is possible that the effect size was too small to be detected by the sample size used in this study. While a significant difference was found in Hypothesis H3 (*Software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels*), the power observed suggests that a larger sample could improve the results. At the same time, if only small effect sizes are observed (as was the case in this study), even statistically significant results might not be meaningful to or usable by practitioners. Future studies might also encounter small effect sizes; however, they will benefit from the results of this study by being able to use the variance observed in this study to estimate the sample size required of a future study that uses similar methods.

Hypotheses Tested by the Study

The hypotheses tested by the study are based on the literature in the **Background and Literature Review** section and test the influence of differences found in API topic documentation (Watson et al., 2013). The hypotheses tested, specifically, whether the variations observed in current API reference topics had a significant impact on the reader's performance and perceptions in a task-oriented scenario, as those variations were modeled in this study. Testing the performance of readers as they read documentation has been done in the past, such as in Redish et al. (1981) and Duffy et al. (1983), and the hypotheses tested relevant aspects of API reference topic variations observed in earlier studies (Watson et al., 2013). The hypotheses, therefore, were reasonable, meaningful, and grounded in earlier literature and professional practice.

Threats to Validity

This section discusses the different threats to validity of this study.

Threats to Internal Validity

This section reviews the threats to the study's internal validity: whether the "outcome is a function of the program or approach being tested rather than the result of other causes" (Tuckman and Harper, 2012).

English Language Proficiency: The study assumed a sufficient and uniform proficiency in reading and comprehending English by filtering the data analyzed based on the participants' self-reporting of proficiency. The assumption was based on the fact that the only people used in the final study data were at least 18 years old (based in the introduction), and were either a native English speaker or had a self-reported level of proficiency equal to a professional or native-speaker. However, none of this information was tested or verified by any other test, such as a question whose response would provide additional validation of the participants' self-reported proficiency. Consequently, the validity of this assumption depends on the honesty and accuracy of the participants' self-reports, which was presumed to be sufficient for this study.

Spurious (insincere) test takers: A concern about insincere test takers could arise because the invitation to participate in the study offered participants the chance to obtain a $10 gift certificate upon completion. Such an invitation could have attracted study participants who might race through the study simply to gain the gift certificate. To prevent such responses from skewing the study, data that revealed impossibly short response times (2 seconds or less) were removed from the data set before analysis. The perceptual data were also reviewed for patterns that indicated a racing participant and those identified as such were removed before data analysis. The **Data Cleaning** section of the **Data Analysis Plan** describes the process used to minimize the influence of such participants on the analysis. The **Raw Study** section of the **Results** describes how many tasks and participants these cleaning processes removed from the analysis. As noted earlier, using the influential-location data might also provide a way to identify insincere test takers in future studies, but the influential-location data were not used to do that in this study.

Topic themes and topic bias: It is possible that the topics used in this study and the related scenarios were biased in a way that could have influenced the results. To minimize this effect, the topics were selected from a variety of programming languages; however, some of the participants' comments indicated that this variety might have been confusing. In contrast to how API reference topics are used by software developers, the study provided the scenario contexts, which varied across tasks. Such unnatural conditions could have influenced the accuracy and consistency of the performance and accuracy measurements; however, no data were collected to know if this was a factor, and if so, to what extent it could have influenced the results.

Artificial nature of task context: Several critical factors that are often present in real-world document use were not present in the artificial nature of the task scenarios used in the study. Specifically, the study provided no external pressure to complete the tasks quickly in the test scenarios—a pressure that is likely to be present in a real-world programming-task context. Likewise, the study provided no peer pressure to complete the task presented in the scenario, which might also exist in a workplace scenario. The tasks in this study might be performed more slowly without type of pressure. Participants might also be more distracted during the study tasks than they might be in a workplace setting, which could influence the relevance-decision response time.

Unmoderated Remote Study: The data for this study were collected by using a remote, unmoderated study, which could experience many environmental influences that are not controlled for by the study. The experiment was designed to minimize the influence of factors such as network latency and computer performance differences, but as an unmoderated study, it is impossible to know if any environmental factors influenced the performance measurements.

Measurement error of test software: The measurement error of the response time measurement is described in the **Assessment Time** section of the **Method** section. The measurement software's worst-case accuracy of 1 percent could influence the accuracy of the measurements taken by the study.

Threats to External Validity

This section reviews the threats to the study's external validity: whether "the results obtained would apply in the real world to other programs or approaches" (Tuckman and Harper, 2012).

Sample size: Significant differences were not found in the tests conducted for Hypothesis H1 and H5. This could be the result of a small effect size where differences might be found to be significant with a larger sample size. The Chi-Square tests that evaluated Hypotheses H2, H4, and H6 would also benefit from larger sample sizes in each cell. Based on the effect size and variance observed in this study, a sample size of 1,000 to 1,500 would have been needed to have sufficient power to reject the null hypotheses. However, if the differences that result from the VDEC and ICEC levels are too small to be detected with the sample size used in this study, it is questionable that such differences would be meaningful in a practical, "real-world" application. Therefore, while a larger sample size would provide more statistical power for the effect sizes observed in some cases, meaningful differences should be detectable with the sample size used in the study.

Non-random sample of population: While participants were not part of a randomly selected sample of the entire population of software developers, they were a diverse representation. The study invitations were sent to 23 groups listed in Table 9, which represented at least 759,938 people with interest or experience in software development. The United States Department of Labor (2014) lists the number of United States employment level of "Computer Occupations" at 3,692,980. The number of invitations sent amounts to about 20 percent of the U.S. employment in the field. However, responses came back from 30 different countries so the percentage of all people in the world who write programs contacted for this study is smaller than

20 percent. Nevertheless, a reasonable cross-section of software developers, who had up to 41 years of experience writing software, responded. Because a diverse population responded to this study, the study's results can be applied in a general sense to software developers; however, because the context of the documentation and task influence the results, care should be taken when considering these results in a specific context.

Specific task context: The study tested API reference topic usage in four specific task scenarios. While these scenarios were taken from real-world examples, it is impossible to know how representative they are of the tasks in which any real-world documentation might be used. The method of study, however, should be applicable in most documentation scenarios in which an API reference topic is used to answer a specific question and so if a scenario is not applicable, a suitable test scenario could likely be developed.

English-language bias: The study was conducted in English for English-speaking developers. English is the language that is used in the vast majority of API reference documentation and so this is a reasonable bias; however, it is difficult to say if the results from this study would apply to documentation in other languages or if the scenarios would apply in different cultural settings.

Threats to Ecological Validity

Bracht and Glass (1968) describe *Ecological Validity* as the ability to generalize the results of an experiment to other environmental conditions. The levels of VDEC and ICEC that comprise the variations of the API reference topics used in this study were designed from the examples identified by Watson et al. (2013). Because Watson et al. (2013) studied API documentation from only open-source software, it is unknown whether the API topics used are representative of API reference topics from other types of software, such as proprietary and commercial software.

Summary Review of Method and Results

The key aspects of the study are useful in examining the implications of the results. The hypotheses were grounded in current literature, the API reference topic variations are distinctly different and ecologically valid, and the study method and population were appropriate. All of these factors and the threats to validity, when taken together, provide a solid foundation on which to evaluate the implications of the study's results.

Review of Research Question

The first aspect of the study to consider is the research question that this study was designed to answer: *What is the effect of visual-design element count (VDEC), unique information-concept element count (ICEC), and topic relevance to the information-seeking task versus ICEC on the speed and accuracy of software developers' assessment of an API reference topic's suitability to answer a task-oriented question, and on their perceptions of the topic?* The following subsections review the dependent measures of assessment speed and accuracy.

Assessment Speed

To review, the lack of significant differences found in the results for Hypothesis H1 indicate that the differences in VDEC do not produce much, if any, difference in mean relevance-decision response time. The results for Hypothesis H3, however, indicate that differences in ICEC do produce a significant difference in mean relevance-decision response time—participants evaluated the relevance of topic variations with a low ICEC 20 percent faster than topic variations with a high ICEC. The results for Hypothesis H5 indicate that the mean relevance-decision response time between levels of ICEC is not affected by topic relevance, while Exploratory Finding 5 shows that the programmers with more experience writing computer software tended to evaluate the topic relevance slightly faster than those with less experience.

The literature cited in this study and many other related studies encourage the use of visual design elements to provide navigation affordances that improve the reader's access and support skimming and scanning (Redish, 2012). The API reference topics used in this study were about a page in length and are included in the **Topic Pages** section in the Appendix. The mean decision-response time in this study of 43.4 seconds provides enough time for an average reader to read over 200 words, or more than the average word count of the API reference topic variations used in this study. Because participants spent enough time to read the content in order to determine its relevance, they might have read more than they skimmed.

The emphasis given to visual design, headings, typography, and the ability to skim the topic in literature about document design for the web would lead one to expect these factors to have a considerable (or at least a measurable) effect on relevance-decision time—a difference that should be easy to detect, even with a small sample. The results of this study, specifically the results for Hypothesis H1, are similar to those from the study conducted by Robins and Holmes (2008) in that the two levels of the VDEC used in the API reference topic variations did not influence relevance-decision time. This result is also consistent with Tractinsky (2004) in his commentary on his earlier work, Tractinsky et al. (2000), suggesting that in this context, the levels of VDEC used in this study did not significantly influence document usability—insofar as relevance-decision time is concerned.

The significant differences found in the results for Hypothesis H3 (*Software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels*) support the notion that the topic was read, more than simply skimmed before a relevance decision was made. The results for H3, where the difference in mean relevance-decision time between high and low ICEC levels was 20 percent, are on the order of the difference expected, but was not found, in H1.

The lack of a significant interaction found in the results for Hypothesis H5 (*When API reference topic variations have the same relevance to the task question, software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels*) suggest that participants took about the same amount of time to evaluate the relevance of an API reference topic with high and low ICEC levels regardless of whether the topic was relevant to the task scenario. It is as though participants have a time budget in mind and either find the answer in that time, or not. To examine this further, an ANOVA compared the mean relevance-decision response times between correct and the incorrect evaluations of all topics and found no significant difference in response time between correctly assessed topics ($M = 42.4$ seconds, $SD = 54.3$, $N = 520$) and incorrectly assessed topics ($M = 46.1$ seconds, $SD = 56.1$, $N = 178$), $F(1,697) = 0.610$, $p = 0.435$). The influential-location data collected in this experiment might also provide insight into the topic areas found influential by participants who correctly versus incorrectly evaluated the topic's relevance.

Assessment Accuracy

The lack of significant differences found in the results of Hypotheses H2 (*Software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low*

VDEC levels) and H4 (*Software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels*) indicate that variations in neither VDEC nor ICEC produce a change in relevance-decision accuracy. However, the significant difference found in the relevance decision accuracy in the topics that were relevant to the task scenarios (Hypothesis H6: *When API reference topic variations have the same relevance to the task question, software developers will determine the relevance of a topic correctly more often in topic variations with high vs. low ICEC levels*) suggests that relevance decision accuracy could be sensitive to ICEC levels. Because the relevance decision accuracy was not influenced by the ICEC levels in the non-relevant topics, future studies could be simplified by omitting the non-relevant scenarios.

Review of API Reference Topic Perceptions

Participants revealed their perceptions of the API reference topics by rating their credibility and professional appearance. Participants rated topic variations with a high VDEC as more credible than those with a low VDEC (H7: *Software developers will give higher credibility ratings to API reference topic variations with high vs. low VDEC levels*). The exploratory investigations revealed that participants rated topic variations with a high ICEC as more credible than those with a low ICEC. Similar results were seen in the ratings of the topics' professional appearance. The exploratory investigations showed that participants rated topic variations with a high VDEC and a high ICEC as having a more professional appearance than those with a low VDEC and a low ICEC, respectively.

In this study, the API reference topics were presented to participants as a document they had selected in response to a search for information about the question presented in the task scenario. In this specific context, the very small effect sizes observed in H1 demonstrated that the visual design did not significantly influence the mean relevance-decision time. In this study, the results for H1 had very low power (0.08), which a much larger set of participants would likely improve. Although, even with more statistical power, the difference (effect size) is still likely to be very small—perhaps statistically significant, but not very meaningful. However, in a broader context—such as considering the reader's decision to select the page from a list of search results—the visual design, along with other factors, could play a role in a reader's evaluation. The credibility and professional appearance ratings suggest that this might be the case; however, testing such an effect was not part of this study.

Effects of Manipulated Variables on API Reference Topic Design

The study's results were reviewed in the context of the literature referenced in the preceding sections. The following sections review the results in the context of professional practice. This study produced a set of findings that need to be considered and balanced against each other in the specific context of the API reference topics to which they are applied.

Effects of Manipulated Variables on Relevance Decision Performance

Changing the manipulated variables in this study had the following effects on reading performance, as measured by the speed and accuracy of the readers' relevance decisions.

1. Varying the topics' VDEC did not significantly affect the decision-response time.
2. Varying the topic's ICEC did significantly affect the decision-response time such that participants could determine the relevance of the shorter topics (lower ICEC) faster than longer ones.

3. Varying the topics' VDEC did not significantly affect the accuracy of relevance assessments.
4. Varying the topics' ICEC did significantly affect the accuracy of relevance decisions in the topic variations that were relevant to the task scenario.

Detailed Review of the Effects of VDEC Levels

That the different levels of VDEC did not produce significant differences in relevance decision performance could suggest several possibilities. It is possible that the levels did not vary enough to produce a difference; however, the participants detected significant differences between the topic variations. That the topic variations were within an ecologically valid range and the levels chosen for this study were determined by variations found in practice and that conform to best practices, it is possible that the best practices these topic variations followed are robust enough to produce consistent results across a wide variety of interpretations. Testing with more extreme variations, such as using much more elaborately designed (very high VDEC) topics, could be used to identify the limits of these practices.

It is also possible that relevance-decision performance is not significantly influenced by topics' VDEC. This could be the result of how the software developer (the reader) reads an API reference topic to find information or how the design and organization of the topic affords information searches. The sequence of the sections in the topics used in this study reflects the sequence of sections found in many of the API reference topics studied in Watson et al. (2013), which is likely to be familiar to software developers as the negative correlation between programming experience and relevance-decision speed observed in this study supports.

Familiarity with the topic's information organization could minimize the effect of visual-design affordances when the topics' organization matches the reader's schema.

The weight to apply to any finding listed above depends on the goals of the API reference topics in their specific contexts with the understanding that these results could be specific to the context and measured variables of this study. The methods used in this study to measure performance can be applied in other contexts to evaluate the net effect of design decisions made in a specific context and for a specific audience.

Detailed Review of the Effects of ICEC Levels

Examining the mean relevance-decision response time in the context of ICEC—the only independent variable that produced a significant difference in mean relevance-decision response time—reveals more information about how the different ICEC levels affect the decision process. An ANOVA evaluated the effect of ICEC (between subjects) and evaluation correctness (between subjects) on the mean decision-response time and found a significant difference in the main effect of ICEC, $F(1,697) = 8.384$, $p = 0.004$. The ANOVA showed a trend towards significance with the interaction of ICEC and evaluation correctness on mean decision-response time, $F(1,697) = 3.471$, $p = 0.063$.

Table 22 shows the mean relevance-decision response time for each level of ICEC and whether the participant evaluated the topic's relevance correctly. Participants who correctly evaluated the topic's relevance had a smaller difference in mean decision-response time than those who incorrectly evaluated the topic's relevance.

Table 22. Response time by evaluation correctness and ICEC level

Evaluation correctness	ICEC level	Mean Relevance Decision Response Time (sec.)	Time difference between ICEC levels (sec,)	Std. Dev.	N
Correct	Low	39.888	4.905	48.908	251
	High	44.793		58.866	269
Incorrect	Low	35.978	22.611	30.552	98
	High	58.589		74.983	80

Separating out the performance of participants who correctly evaluated the topic's relevance suggests that for those who correctly evaluated the topic's relevance, the effect of ICEC level is not as strong as the results for Hypothesis H3 (*Software developers will have a shorter mean relevance-decision response time in topic variations with low vs. high ICEC levels*) suggests. That is, it took almost the same amount of time (within 11 percent) for participants to assess correctly the relevance of topics with a low ICEC as it did for them to assess topics with a high ICEC.

The influential-location data might provide information about participants who did not assess the topic's relevance correctly, which could provide additional insights about the participants. For example, the influential-location data described above might provide insight into why they took longer to decide whether the topic was relevant or not. From a document design perspective, however, the value from the results for Hypothesis H3 would indicate whether there was a difference to investigate. Data about the correctness of the relevance decisions made would provide insight about where to look further.

Effects of Manipulated Variables on Readers' Perceptions

This study demonstrated how variations in VDEC and ICEC influenced participants' perceptions of the topics they viewed. These results suggest that API reference topic design might need to serve more than information-seeking goals. This study demonstrates that the VDEC and ICEC of an API reference topic influence these aspects of readers' perceptions of credibility and professional appearance.

1. Participants rated topic variations with a high VDEC as more credible than topics with a low VDEC.
2. Participants rated topic variations with a high ICEC as more credible than topics with a low ICEC.
3. Participants rated topic variations with a high VDEC as having a more professional appearance than topics with a low VDEC.
4. Participants rated topic variations with a high ICEC as having a more professional appearance than topics with a low ICEC.

If these perceptions also influence readers' assessments of the credibility and professionalism of the company or service that produces the documentation, those factors might be more significant to the business than the information-seeking factors. Given that variations in VDEC did not have a significant effect on relevance-decision performance, variations in VDEC might be only motivated by these perceptions. This study did not test how far beyond the API reference topic the participants' perceptions extended, so that would need to be measured and understood before the influence of API reference topic perceptions could be weighed in a larger context.

Improving Flow and Reducing Interruption Time

One of the motivations to study readers' performance with API reference topics was to study how variations in VDEC and ICEC affect the relevance-decision time as a step towards making it easier and faster for software developers to access information in API reference topics. In the development scenarios of Systematic Developers, researching information in an API reference topic while developing software can break the software developers' "flow" by introducing a secondary task. A secondary task that consists of a short interruption with a minimal cognitive load does not disturb the flow of the primary task (Sweller, 1988). However, when a task extends beyond a short time, the interruption and additional cognitive load of the interruption has a more profound effect on the primary task—to the point where the interruption can become the primary task, replacing the original task of software development. In the development scenarios of Opportunistic and Pragmatic Developers, information-seeking tasks are integral to the primary task of software development. Reducing the impact of these information-seeking tasks benefits all software development personas to varying degrees.

One premise of this study was that if the interruption of researching a question in an API reference topic could be kept below the threshold of disrupting the primary task in terms of time and cognitive load, the research task (the interruption) could be performed almost transparently to the software developer. The mean relevance-decision response times listed in Table 16, however, suggest that this might be a challenging goal for information-seeking tasks such as those studied in this research. For short reminders, a pop-up prompt as shown in Figure 1 might prevent the flow state from being disturbed. However, research tasks, such as the scenarios used in this study, required much more time than was necessary to break a flow state. DeMarco and

Lister (2013) say that the flow state can be disturbed by as little as an announcement over a public-address system and it can take up to 15 minutes to recover the flow state. Although DeMarco and Lister (2013) do not describe a specific time, the 43.373-second mean decision-response time observed in this study is much longer than the time they describe as being sufficient to disturb the flow state. Even if the mean relevance-decision response time could be reduced by 50 percent to 21.7 seconds, that is still long enough to disturb the flow state by itself, not including the other information-gathering tasks that Rouet (2006) describes in the TRACE model. While such an improvement might not prevent a systematic developer persona from losing their flow state, it would still benefit all software developer personas to some degree.

Whether the current state of the art in API reference topic design is a local or a global maximum is another factor to consider. It is possible that, as Duffy et al. (1983) found, after over 50 years of software documentation, the current designs represent the state of the art and that redesigned API reference topics will only be different, not faster. On the other hand, additional research into how API reference topics are designed and used might lead to insights that produce the disruptive change necessary to reveal a new approach to the design and authoring of API reference topics. Currently, there is not enough information to know the limits of reading and relevance decision-making performance are and their influences.

Use of Influential-Location Data

Including the direct user feedback about the influential-location in a topic, which the spot interaction provided, was a novel analysis technique for technical documentation. This study used an exploratory Chi-Square analysis of the topic sections that participants found most influential in their relevance decision and found results similar to those found by evaluating the decision-response time data: changes in the VDEC levels did not influence the topic sections that participants found influential while changes in the ICEC levels did.

The influential-location data, however, could provide much more information. They provide, at a minimum, a way to solicit feedback from the user about a specific item in or interaction with a web document. In this study, the question concerned which part of the topic was most influential in the participants' relevance decision. In other contexts, this interaction could be used to answer other questions. For example, data about the most helpful or most confusing part of the topic could be collected by changing the prompt—perhaps in response to other feedback such as the more common, "Did you find this topic helpful?" request.

The influential-location data collected by the spot interaction can provide insight into the performance measures such as the mean relevance-decision response time. For example, in the tasks where the task's scenarios are relevant, the spots that identify the influential-locations can be seen to fall on key words from the scenario. In Task 1, the scenario asks if the copy function's parameters can be URLs and many of the spots are concentrated on the word "URL" in the text. In Task 3, the scenario asks if the function can return immediately or if it must wait for a specific condition and the spots are concentrated on the word "wait." The **Influential Spot Location Maps** section in the Appendix contains the influential-location data for the topic variations used in each task with the spots that show the influential-location responses overlaid on the topics.

The influential-location responses could also be used to validate whether participants who decided that the topic was relevant, did so using a reasonable reference—possibly as a more accurate way to filter out spurious responses. Using the influential-location data to identify valid responses—that is selecting for analysis only participant responses that indicate a reasonable response to the prompt—could identify participants who responded to the question more accurately than by looking at only their response time. Using only response data from participants who indicated that they found the data in a reasonable location of the topic might improve the analysis of performance data.

CONCLUSIONS

This study contributes critically needed empirical data to the under-studied area of how document design elements and the information in a topic influence readers' assessment of API reference topics. The findings of this study illustrate that an API reference topic's VDEC does not influence readers' relevance-decision speed, but does influence their perceptions of the topic's credibility and professional appearance. Likewise, it illustrates that changing an API reference topic's ICEC to make a small improvement in readers' relevance-decision speed can adversely influence readers' perceptions of the topic's credibility and professional appearance. The study applied and refined methods that researchers and practitioners can use in other documents and contexts to expand upon this set of empirical data and validate other design and content decisions. Collecting a larger data set from diverse contexts and applications can help improve our understanding of the effects of document design, which can then lead to improved best practices for practitioners.

Implications for Research

This section describes the implications that the study results have on research into API documentation and similar types of informational documentation.

Unmoderated, Internet-Based Methods to Study API Reference Topics

The methods used in this study demonstrate and reaffirm the viability, as well as the complexity, of using an unmoderated, Internet-based user-experience research method to collect data from readers. The survey tool used in this study collected objective data about how long participants took to decide a topic's relevance, and it collected perceptual data about document characteristics by questioning participants immediately after evaluating the document's relevance. These different interactions provided a robust view of the participants' experience with the document while being minimally intrusive and without the need to observe participants directly. There are questions that this method of study cannot answer well, such as the question the study raised about whether participants skimmed the topics, read the topics, or applied some type of hybrid reading strategy. However, for the questions that it can answer, it can do it efficiently and economically.

The ecological validity offered by the methods used in this study, however, needs to be weighed against the environmental influences that are present in the methods used in this study. In this study, the environmental influences manifest themselves as variation in measured variables such as relevance-decision time. At the same time, other measured variables, such as the perception measures showed less variation with this method. Finally, the literature states that the relevance decisions on which the study's findings rest depend on the context in which they are made, adding more value to having the participants take the studies in their environment.

CSS-Based Document Variations

This study demonstrated that using multiple CSS style-sheets is an effective way to generate topic variations in web-based documentation to observe their effects in a remote, online study. This study used an adaptation of style-sheet-based topic variations. While the API reference topics displayed to the study's participants were images, they were generated from HTML documents that were styled using the .css files included in the **Study Style Sheets** section in the Appendix. Images, instead of HTML documents, were used in the study to disable browser-based tools, such as in-page search affordances like ctrl-F search, which would distort the data.

However, the method for creating the document text and visual presentation styles of the API reference topics used could be applied to an actual online API documentation set.

Conditional content that is identified by CSS styles could be used to control how the content is presented to the reader to accomplish various goals. Goals such as limiting the content displayed to that which is focused on the search terms or query context as well as to test experimental conditions would be supported by CSS-styled content tagging.

Targeted Feedback from Unmoderated Studies

The spot interaction used to collect participants' assessments of the topic location that most influenced their relevance decisions provided valuable insights into readers' experiences with very little intrusion to their experience. The interactive method used to collect feedback from participants about the API reference topic used in the task asked participants to, "*Click on the part of the topic that influenced your decision the most.*" Participants responded to this prompt in less than 16 seconds, on average, indicating that they found this interaction clear and easy to use.

Future research could use this interaction in other types of documents to collect answers to other questions about how readers interact with a web page. By changing the prompt to ask another question about the document, researchers could collect many types of user-experience data in an unmoderated, remote study. The data provided by this type of interaction could be analyzed as the specific graphical coordinates of a location in the topic being studied, as topic sections, or both to support a variety of statistical and graphical analyses and presentations.

Implications for Professional Practice

This study has implications for the practice of authoring API reference topics and other types of API documentation. Included here is a critical review of current best-practices described in the literature, a discussion of how API reference topics seem to be read more than skimmed (contrary to current guidance regarding web-content), and a summary of how the information content of API reference topics influences readers’ performance and perceptions of the topics.

Updates to Best Practices

The findings from this research underscore the importance of context that is mentioned in the best practices for authoring content for the web. Best practices for visual design of web content that are commonly discussed in the literature should describe scenarios concerning when they do and do not apply. The best practices should also provide information about the aspects of the readers’ experiences that they influence. This additional information and context would help API documentation writers understand and focus on documentation elements that are critical to meeting their readers’ goals. This study showed that some of the best practices described in the literature and that guided the design of this experiment had no significant influence in relevance decision-making performance. Variations in the visual design features commonly found in API reference topics did not have a significant effect on relevance-decision performance in this study. It appears that in this context, design features are not critical to decision-response time. From the results of this study and the variations in API reference topics that Watson et al. (2013) observed, it is possible that software developers are accustomed to the layout and organization of the topics and do not need the navigational cues offered by design features such as headings, typography, and other visual design elements. While changes in the visual design of the API reference topics in this study did not influence relevance-decision time, they did influence participants’ perceptions of the topics. Because changes to document design can be complex and costly, these findings can help API documentation authors weigh the relative costs and benefits of the visual design versus the information content when tradeoffs are needed.

None of this is to say that the best practices identified in the literature should be ignored—the topic variations used in this study applied many of them. To maintain ecological validity, the API reference topic variations intentionally used as many best practices as each topic variation would allow (following the variations identified in Watson et al., 2013). However, this study demonstrated that different best practices have different influences on how readers perceive and interact with a topic. API documentation authors and software developers would benefit greatly from a clarification of which best practices work in which contexts to influence which performance measures.

Assumptions about Reading versus Skimming

The findings of this study do not indicate a clear usage model for how participants evaluated the relevance of the topics and suggest additional study. On average, participants spent enough time deciding on the relevance of the API reference topics in this study to read most, if not, all of the content in topics they reviewed. The mean relevance-decision time of 43.4 seconds suggests that participants did more than simply skim the content or read only the headers. The assertion that "web content is skimmed," as the advice from Redish (2012) is often paraphrased, is incomplete in the context of referring to API reference topics for information.

This study showed that readers spend about the same amount of time on the page deciding that the topic is not relevant as they do when they decide that it is. While it is reasonable to expect readers to read until they believe they have the information they seek (Redish, 2012), with the API reference topics used in this study, they spent the same amount of time on the topic whether it had the information they sought or not. The length of time that participants in this study spent on a topic was about 45 seconds, whether they found the topic to be relevant to the task scenario or not.

Readers of this genre might use their familiarity of the topics' information schemas to guide their decisions. They might also skim the topic to become oriented with the specific organization of the topic, and then read many, if not all, of the details to find the answer they seek. Their approach could be the result of the technical nature of the question that brought them to the topic and of the answer such a question requires—skimming might not be a practical strategy to apply in this context. At the same time, the variance in these measurements suggests that participants might apply a range of reading and evaluation methods. In any event, readers appear to interact with this genre of document in something more than the superficial approach that the "readers always skim" mantra suggests.

Influence of Information in an API Reference Topic on Relevance Decisions and Speed

The amount of information in an API reference topic does not seem to influence the likelihood of a reader correctly assessing the topic's relevance. The results for Hypothesis H3 show that relevance decisions were made more quickly in shorter topics and demonstrated that reducing the information concept count in an API reference topic improved the relevance-decision time without degrading the accuracy of the relevance decision. The results for Hypothesis H5 show that the actual relevance of the topic to the task scenario, however, did not influence the relevance-decision response time. The relevance-decision response time results are consistent with the advice provided by web-content best practices advocated by Redish (2012) and

McGovern (2006) that “less is more.” However, the relevant case of Hypothesis H6 showed that a relevance decisions were less accurate in the topic variations with a low ICEC and participants rated the topic variations with a low ICEC as being less credible and having a less professional experience than the topic variations with a high ICEC. Whether less content is better, depends on how *better* is measured.

Summary of Implications

The results of this study reinforce some best practices, suggest that others be adapted for the specific context, and, in general, suggest that the best practices for API reference topics, and perhaps web content in general, are more nuanced than can be summed up in a catch-all slogan. However, the most valuable contribution to future research and the practice of writing API reference topics comes from applying and validating methods that can be used to test the assumptions that must be made in the process of writing API reference topics, specifically, and other types of documentation in general.

Assumptions must often be made when authoring content for the web because reliable data about the audience are not always available to writers before they must write and publish content. Even when audience data are available, they are often dynamic and change over time. While this study examined how software developers interacted with API reference topics to perform an information-seeking task, the methods described in this study could be adapted to provide data about how the audience is interacting with many other types of content. The spot interaction could also be used to collect answers to specific questions about the content in any website.

Researchers of API reference topics will be able to apply the variance data from this study to estimate more accurately the number of participants required to obtain sufficient power in similar studies in the future. The formula to estimate the minimum number of participants requires an estimate of the expected variance. Being the first study of its kind, this information did not exist and so the estimate of the participation required rose from an initial 200 participants, to the 500 participants approved for the final study—mostly as the result of trial and error. Future studies of a similar nature will be able to use the results from this study to estimate the required number of participants early in the design.

Researchers and practitioners alike will be able to apply the tools and methods used in this study to begin collecting data on API and other documentation. While the market of tools that collect data from commerce-oriented sites is quite robust, there are very few tools, methods, or best practices for collecting data from information-oriented sites. This study examined API reference documentation, but the tools and methods could be applied to other information-oriented sites with very little modification.

FUTURE WORK

This study answered the research question and identified new questions and research opportunities, which are described here in three subsections. The first subsection considers additional analyses that the study's data set enables. The second subsection summarizes the questions that arise from this study and the research opportunities they offer. Finally, opportunities to apply the study's results to professional practice by API documentation authors and authors of informational content are reviewed.

Additional Analysis of the Study Data

The study collected more data than was needed to answer the research question to take advantage of the study to test different data-collection methods in the context of an online study of API reference topics and to provide insight into the findings. This section reviews the additional research questions and data-analysis opportunities that the data set supports.

Analyze the Influential-Location -Data Further

The influential location data represent a novel form of participant response data that were only briefly analyzed for this study; however, this analysis identified many additional ways these data could be studied and applied. The influential-location data that describe the locations influential to participants' relevance decisions were collected as raw data in the form of Cartesian coordinates referenced to an image of the API reference topic variation. For this dissertation, these data were also interpreted as topic sections to compare how the influential topic section varied with the API reference topic variations of the experiment. The discussion of the influential-location data in the **Results** section describes an analysis of the data as a continuous variable (measured in pixels) and as a categorical variable of topic section—each having a different application and answering a different question. Ultimately, this study used the categorical variables to find that the different ICEC levels that influenced the topic section that participants found most influential to their relevance decision, but this data might answer other questions as well. The data collected in this study could be used to assess the following ideas in future analyses.

- Whether variations in ICEC and VDEC influence the standard deviation of the location of the most influential part of the topic. This investigation would reveal how variations in the API reference topics affect the location in the topic that participants found to be most influential.
- Whether the location that participants found to be most influential contains information that pertained to the scenario. The study data could be recoded to identify the sections with information that pertained to the task scenario. With that information, the responses could be analyzed to determine whether selecting a location that did or did not have information that pertained to the task scenario was significant.
- Whether the influential-location data could be used to filter participants and tasks for analysis. As a follow-up to the previous analysis, for example, does selecting only the responses in which the participant selected a location with information that pertains to the task scenario alter the results and conclusions? Does the influential-location data identify insincere participants more accurately than time on page or response patterns?

- Whether the influential-location data could also be used more qualitatively by relating the influential locations to the perceptual data. For example, one could examine whether the perception of credibility and professional appearance relate more to whether the participant indicated a reasonable location in the document—suggesting that credibility ratings could relate more to whether participants found the answer in the topic than the VDEC or ICEC of the API reference topic variations they saw.

Analyze Performance and Accuracy by Response Correctness

Evaluating the data by excluding incorrect relevance decisions or by including only responses with correct influential-location information could provide results that more accurately reflect the target population. As discussed earlier in **Detailed Review of the Effects of ICEC Levels**, mean relevance-decision response time results varied by the answer's correctness (see also Table 22). It is possible that other results could also be affected by whether participants correctly evaluated the topic relevance. This would be a special or an additional finding and not necessarily invalidate the results presented earlier. In an ecologically-valid group of readers, there will be some who understand the document and some who do not. This study included both in the analysis of the results, but studying those who correctly evaluated the relevance separately from those who did not might reveal new information. Likewise, studying the responses of those participants who did not correctly evaluate the relevance of the topic could provide additional insights. For example, such an analysis could identify errors in a topic's content.

Future Research Opportunities

This section reviews research opportunities that arose from this study.

Collect Eye-Tracking Data

This study found that participants spent enough time evaluating the topics to read the entire topics; however, it could not collect information about how participants were reading the topic. The locations of influential-location responses indicate also indicate that some participants read the entire topic. All of this suggests that there is more to understand about how software developers read an API reference topic and an eye-tracking study would provide valuable information towards understanding this.

Investigate Additional Applications of the Study Method

The methods applied to this study could be used to collect data about other types of documents with an interaction pattern that is similar to the one in this study—that of looking for information in a topic on the web. The spot interaction used in this study to collect data about the most influential part of the topic could collect other types of data about readers' experiences with a very short interaction, providing a way to learn more about the audience and their experience with the topic. For example, the spot interaction could be applied to collect feedback about the design and content of a particular page—the specific information collected depending on the prompt. In this study, readers were asked to indicate the part of the topic that was most influential in their relevance decision. In other types of informational sites, for example, the prompt used by the spot interaction could ask the reader to identify the part of the topic that was most helpful or most confusing and the reader could indicate it directly.

Evaluate whether the VDEC and ICEC Levels Influence Topic Selection

The context that surrounds the relevance-decision scenario examined in this study needs more research. This study evaluated the effect of VDEC and ICEC on API reference topic relevance decisions, and its findings suggested that their effects might influence other aspects of the multiple-document knowledge-building process. This study evaluated the relevance-decision time after a topic had been selected, but it did not evaluate the process of how a software developer selects a topic from a list of search results to evaluate and the documentation variables that influence that decision. This study found that the levels of VDEC and ICEC influenced perceptions of credibility and professional appearance—factors that could influence the document-selection process that occurs before a selected document is evaluated. The search context that exists around the scenario examined in this study is quite complex and interacts with the documentation being found by the search.

The tools and methods used in this study to manipulate variations and collect data could be used in a production environment to incorporate contextual information, such as search terms, in the analysis. Employing this and other opportunities to understand the context in which API reference topics are used will help evaluate the document variations examined in this study in a more representative context.

Explore More Variations of API Reference Topic Visual Design

This study found that variations in VDEC affected neither relevance-decision speed nor accuracy; however, greater deviations from the current best practices modeled for this experiment could produce different results. It is possible that current API reference topic design is a *global maximum* in that after 50 years of producing software documentation, the design cannot be improved further. However, developments such as the pop-up help illustrated in Figure 1 demonstrate that innovations outside of the traditional manual-page format can improve access to information. Perhaps by understanding the subject more deeply, a new design pattern can be created to produce a new global maximum of API reference topic design.

The literature and best practices encourage a meaningful and obvious document structure that is clearly indicated by headings and typographical affordances. While these affordances have a long history in the printed word, varying such design features in this study made very little difference in the mean relevance-decision time. The disconnect between document design history and what was observed in this study should be explored because it is possible that the variations studied here are only minor changes around a local maximum. Now that this study method has been tested with conventional API reference topics, it can used to test the effects of more radical approaches to API reference topics.

Understand the Audience Better

The instrumentation and observation methods used in this study could be applied in a variety of contexts and target audiences to help identify similarities and differences between software developers and other audiences. Software developers are the subject of frequent study because there is a lot to gain by improving their productivity, yet how they interact with documentation has received comparatively limited attention. There are many possible reasons for this lack of attention such as the difficulty associated with observing the infrequent event of software developers using documentation and that many documentation interactions are treated as secondary tasks. It is possible that the lack of differences seen in the VDEC levels is the result of usage characteristics that are specific to software developers.

Part of the motivation for this study was to identify the combination of design and information concepts to use in an API reference topic that provides the necessary information to the software developer quickly enough to prevent interrupting the primary task. However, that time limit is only vaguely specified. Knowing the maximum duration of an interruption before the flow state is lost and the environmental considerations that influence that duration are important aspects to know about the software development experience and necessary components to accomplish this goal. Studying Opportunistic Developer task in which documentation access is an integral part of their development process would help understand how to support this interaction with documentation and identify the aspects of documentation that support all development personas.

Application to Professional Writing

Practitioners can apply the results of this research immediately to improve the effectiveness of API reference topics for the software developers who use them. Some of the aspects of this research that can be applied immediately include:

1. **Identification of the model of multiple-document theory to building knowledge during software development**

   Having such a model can help practitioners identify and understand the different processes a software developer applies when researching answers in formal and informal API documentation.

2. **Reevaluation of the best practices**

   This research underscores the need to have a clear understanding of readers' tasks and attitudes, such as the readers' sensitivity to designs that could influence a topic's credibility before deciding how to apply the current best practices to a topic.

Looking forward, the findings from this study can inform testing and evaluation practices. An important contribution of this study to the practice of technical writing is that it demonstrates different ways to collect feedback and other data about the documentation in the readers' contexts in order to test and refine assumptions made about the audience and the documentation. Because the context in which the documentation is used—whether it is API reference topics or any other documentation—influences the specific results, any recommendations made here or elsewhere must be tested and verified in specific contexts. For informational web sites, the specific questions to ask about the content are difficult and often vague and the tools available to answer these questions, once asked, are scarce and difficult to apply. In the future, the methods and tools used in this study could be integrated into authoring tools and documentation systems to make such data collection easier and more accessible. Further, knowing and understanding the parameters and limitations of the recommendations is critical to finding and applying the best practices in a specific situation. Having a robust set of user-experience data collection tools that provide information specific to the goals of the documentation will make it possible to explore and measure the effectiveness of new document designs.

REFERENCES

Abrams, B. (2008, March 17). *Number of Types in the .NET Framework*. Retrieved from http://blogs.msdn.com/b/brada/archive/2008/03/17/number-of-types-in-the-net-framework.aspx

Bailey, B. P., Adamczyk, P. D., Chang, T. Y., & Chilson, N. A. (2006). A framework for specifying and monitoring user tasks. *Computers in Human Behavior*, *22*(4), 709–732.

Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2000). Measuring the effects of interruptions on task performance in the user interface (Vol. 2, pp. 757–762). In the *Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE.

Bannert, M. (2002). Managing cognitive load-recent trends in cognitive load theory. *Learning and Instruction*, *12*(1), 139–146.

Barry, C. L., & Schamber, L. (1998). Users' criteria for relevance evaluation: a cross-situational comparison. *Information Processing & Management*, *34*(2), 219–236.

Bartell, A. L., & Spyridakis, J. H. (2012). Managing risk in internet-based survey research (pp. 1–6). In the *Proceedings of the 2012 IEEE International Professional Communication Conference (IPCC)*. IEEE.

Bracht, G. H., & Glass, G. V. (1968). The external validity of experiments. *American Educational Research Journal*, *5*(4), 437–474. http://doi.org/10.2307/1161993

Brandt, J., Dontcheva, M., Weskamp, M., & Klemmer, S. R. (2010). Example-centric programming: Integrating web search into the development environment (pp. 513–522). In the *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., & Klemmer, S. R. (2009). Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code (pp. 1589–1598). In the *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

Brandt, J., Guo, P. J., Lewenstein, J., & Klemmer, S. R. (2008). Opportunistic programming: How rapid ideation and prototyping occur in practice (pp. 1–5). In the *Proceedings of the 4th International Workshop on End-User Software Engineering*. ACM.

Carroll, J. M. (Ed.). (1998). *Minimalism Beyond the Nurnberg Funnel*. Cambridge, MA. MIT Press.

Clarke, S. (2003, November 24). *Using the Cognitive Dimensions, Continued - Learning Style*. Retrieved from http://blogs.msdn.com/stevencl/archive/2003/11/24/57079.aspx

Clarke, S. (2005). Describing and measuring api usability with the cognitive dimensions. In the *Proceedings of the Cognitive Dimensions of Notations 10th Anniversary Workshop*. Citeseer.

Clarke, S. (2007, February 7). *What is an End User Software Engineer?* Retrieved October 26, 2014, from http://drops.dagstuhl.de/opus/volltexte/2007/1080/

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. L. Erlbaum Associates.

Cwalina, K., & Abrams, B. (2008). *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries*. Addison-Wesley Professional.

DeMarco, T., & Lister, T. (2013). *Peopleware: Productive Projects and Teams* (3rd ed.). Upper Saddle River, NJ: Addison-Wesley Professional.

DeStefano, D., & LeFevre, J.-A. (2007). Cognitive load in hypertext reading: A review. *Computers in Human Behavior*, *23*(3), 1616–1641.

Dillman, D. A., Smyth, J. D., & Christian, L. M. (2008). *Internet, Mail, and Mixed-Mode Surveys: The Tailored Design Method* (3rd ed.). Wiley.

Duffy, T. M., Curran, T. E., & Sass, D. (1983). Document design for technical job tasks: An evaluation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *25*(2), 143–160.

Henning, M. (2007). API design matters. *Queue*, *5*(4), 24–36.

Knowles, M. S., Swanson, R. A., & Holton, E. F. I. (2011). *The Adult Learner: The Definitive Classic in Adult Education and Human Resource Development* (7th ed.). Taylor & Francis.

Ko, A. J., Myers, B. A., Coblenz, M. J., & Aung, H. H. (2006). An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *Software Engineering, IEEE Transactions on*, *32*(12), 971–987.

Krippendorff, K. H. (2012). *Content Analysis: An Introduction to its Methodology* (3rd ed.). Los Angeles ; London: SAGE Publications, Inc.

Lidwell, W., Holden, K., & Butler, J. (2010). *Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design* (2nd ed.). Rockport Publishers.

Lindeman, E. C. (1989). *The Meaning of Adult Education (4th printing).* Norman, OK: Printing Services, The University of Oklahoma.

Maalej, W., & Robillard, M. P. (2013). Patterns of knowledge in api reference documentation. *IEEE Transactions on Software Engineering*, *39*(9), 1264–1282. doi:10.1109/TSE.2013.12

McGovern, G. (2006). *Killer Web Content: Make the Sale, Deliver the Service, Build the Brand* (1st ed.). London: A&C Black Trade.

Microsoft Corporation. (2014). *Microsoft Manual of Style* (4th ed., 3rd printing). Redmond, WA: Microsoft Press.

Microsoft Corporation Editorial Style Board. (2004). *Microsoft Manual of Style for Technical Publications* (3rd ed.). Redmond, WA. Microsoft Press.

Mihaly, F. (2011, November 1). *Writing Helpful API Documentation*. Retrieved from http://theamiableapi.com/2011/11/01/api-design-best-practice-write-helpful-documentation/

Mobrand, K. A., Cuddihy, E., Galore, E., & Spyridakis, J. H. (2007). The effect of structural cues on user comprehension, navigational behavior, and perceptions (pp. 1–7). In the *Proceedings of the IEEE International Professional Communication Conference, IPCC 2007.* IEEE.

Nykaza, J., Messinger, R., Boehme, F., Norman, C. L., Mace, M., & Gordon, M. (2002). What programmers really want: results of a needs assessment for SDK documentation. In the *Proceedings of the 20th Annual International Conference on Computer Documentation* (pp. 133–141). ACM.

Parnin, C. (2013, March 4). *API Documentation - Why it Sucks* [blog]. Retrieved November 2, 2014, from http://blog.ninlabs.com/2013/03/api-documentation/

Parnin, C., & Treude, C. (2011). Measuring api documentation on the web. In *Measuring API Documentation on the Web* (pp. 25–30). Waikiki, Honolulu, HI, USA: ACM.

Piaget, J. (1970). *Science of education and the psychology of the child* (First printing.). New York: Orion Press.

Pollock, E., Chandler, P., & Sweller, J. (2002). Assimilating complex information. *Learning and Instruction*, *12*(1), 61–86.

ProgrammableWeb Research Center. (2014). Retrieved October 26, 2014, from http://www.programmableweb.com/api-research

Redish, J. (2012). *Letting Go of the Words: Writing Web Content that Works* (2nd ed.). Amsterdam ; Boston: Morgan Kaufmann.

Redish, J. C., Felker, D. B., & Rose, A. M. (1981). Evaluating the effects of document design principles. *Information Design Journal*, *2*(3-4), 3–4.

Resig, J., 2008. *Accuracy of JavaScript Time*. John Resig. URL http://ejohn.org/blog/accuracy-of-javascript-time/ (accessed 3.7.15).

Reynolds, G. (2008). *Presentation Zen: Simple Ideas on Presentation Design and Delivery* (1st ed.). Berkeley, CA: New Riders.

Robillard, M. P. (2009). What makes apis hard to learn? Answers from developers. *IEEE Software, 26*(6), 27–34. IEEE.

Robillard, M. P., & DeLine, R. (2011). A field study of api learning obstacles. *Empirical Software Engineering*, *16*(6), 703–732.

Robins, D., & Holmes, J. (2008). Aesthetics and credibility in web site design. *Information Processing & Management*, *44*(1), 386–399.

Rouet, J.-F. (2006). *The Skills of Document Use: From Text Comprehension to Web-Based Learning* (1st ed.). Lawrence Erlbaum Associates.

Samuelson, P., & Scotchmer, S. (2002). The law and economics of reverse engineering. *The Yale Law Journal*, *111*(7), 1575–1663. doi:10.2307/797533

Skinner, B. (1954). The science of learning and the art of teaching. *Harvard Educational Review*.

Sperber, D., & Wilson, D. (1986). *Relevance: Communication and cognition*. Oxford, U.K.: Basil Blackwell Ltd.

Spyridakis, J. H., Wei, C., Barrick, J., Cuddihy, E., & Maust, B. (2005). Internet-based research: Providing a foundation for web-design guidelines. *IEEE Transactions on Professional Communication, 48*(3), 242–260. IEEE.

Stylos, J., & Myers, B. A. (2005). How programmers use internet resources to aid programming. *Submitted for Publication*.

Stylos, J., & Myers, B. A. (2006). Mica: A web-search tool for finding api components and examples. In the *Proceedings of the Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on* (pp. 195–202). IEEE.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285.

Tidwell, J. (2011). *Designing Interfaces* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Tractinsky, N. (2004). A few notes on the study of beauty in HCI. *Human–Computer Interaction*, *19*(4), 351–357.

Tractinsky, N., Cokhavi, A., Kirschenbaum, M., & Sharfi, T. (2006). Evaluating the consistency of immediate aesthetic perceptions of web pages. *International Journal of Human-Computer Studies*, *64*(11), 1071–1083.

Tractinsky, N., Katz, A., & Ikar, D. (2000). What is beautiful is usable. *Interacting with Computers*, *13*(2), 127–145.

Tuckman, B. W., & Harper, B. E. (2012). *Conducting Educational Research* (6th ed.). Rowman & Littlefield Publishers.

United States Department of Labor. (2014). *May 2014 National Occupational Employment and Wage Estimates*. Retrieved April 27, 2015, from http://www.bls.gov/oes/current/oes_nat.htm#15-0000

Valcke, M. (2002). Cognitive load: Updating the theory? *Learning and Instruction*, *12*(1), 147–154.

Van Schaik, P., & Ling, J. (2008). Modelling user experience with web sites: Usability, hedonic value, beauty and goodness. *Interacting with Computers*, *20*(3), 419–432.

Van Schaik, P., & Ling, J. (2009). The role of context in perceptions of the aesthetics of web pages over time. *International Journal of Human-Computer Studies*, *67*(1), 79–89.

Watson, R. B. (2012). Development and application of a heuristic to assess trends in api documentation. In the *Proceedings of the 30th ACM international conference on Design of communication* (pp. 295–302). Seattle, WA: ACM.

Watson, R., Stamnes, M., Jeannot-Schroeder, J., & Spyridakis, J. H. (2013). API documentation and software community values: A survey of open-source api documentation. In the *Proceedings of the 31st ACM International Conference on Design of Communication* (pp. 165–174). New York, NY, USA: ACM. doi:10.1145/2507065.2507076

Zakas, N.C., 2011. *Timer Resolution in Browsers - NCZOnline*. NCZOnline. URL http://www.nczonline.net/blog/2011/12/14/timer-resolution-in-browsers/ (accessed 3.7.15).

APPENDICES

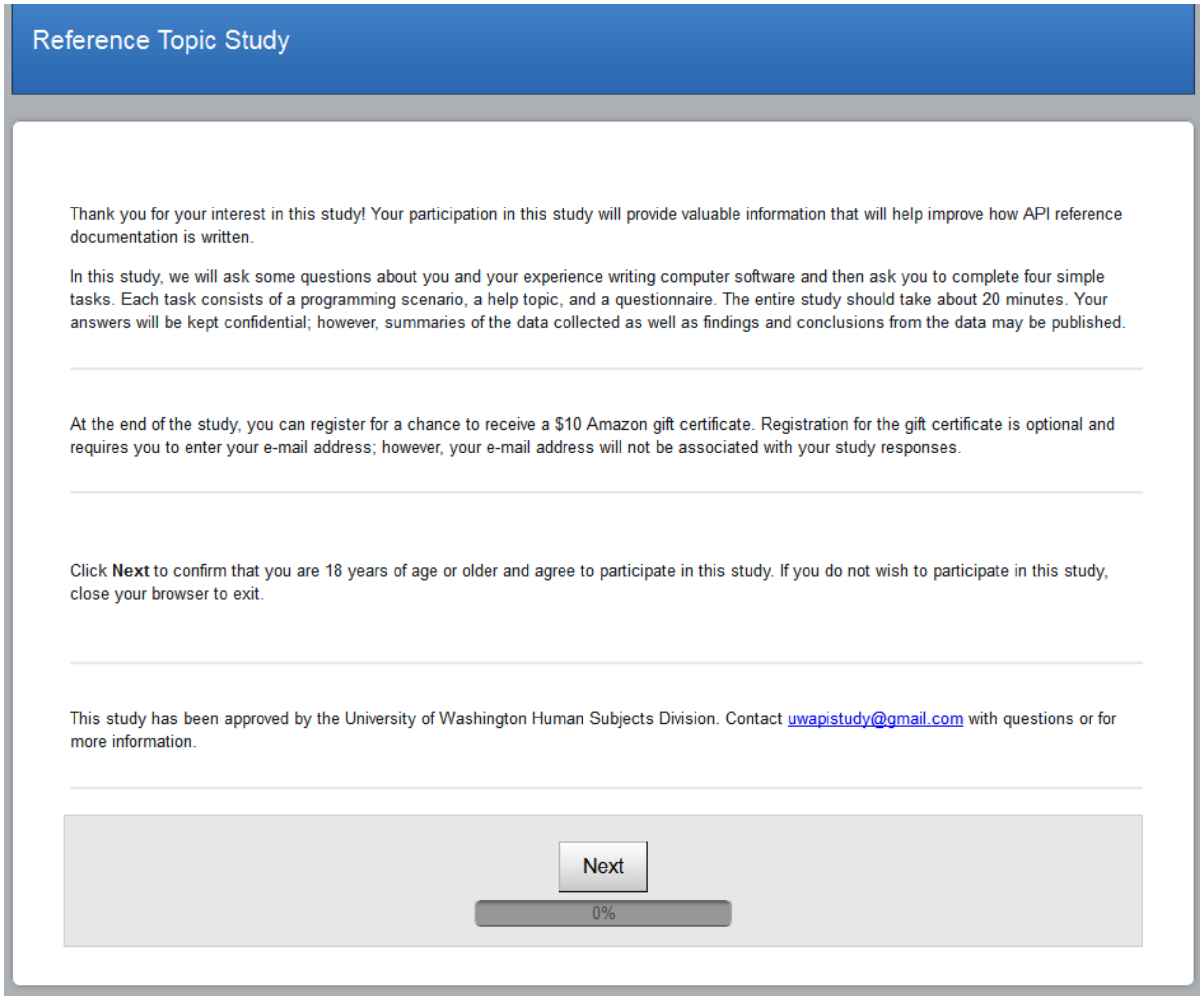
A. STUDY MATERIALS

This section contains samples of the study materials used to collect data for this research.

Online Survey Tool

This section contains screenshots from the online survey tool used to collect data for this experiment.

Welcome and Consent Page



Thank you for your interest in this study! Your participation in this study will provide valuable information that will help improve how API reference documentation is written.

In this study, we will ask some questions about you and your experience writing computer software and then ask you to complete four simple tasks. Each task consists of a programming scenario, a help topic, and a questionnaire. The entire study should take about 20 minutes. Your answers will be kept confidential; however, summaries of the data collected as well as findings and conclusions from the data may be published.

At the end of the study, you can register for a chance to receive a $10 Amazon gift certificate. Registration for the gift certificate is optional and requires you to enter your e-mail address; however, your e-mail address will not be associated with your study responses.

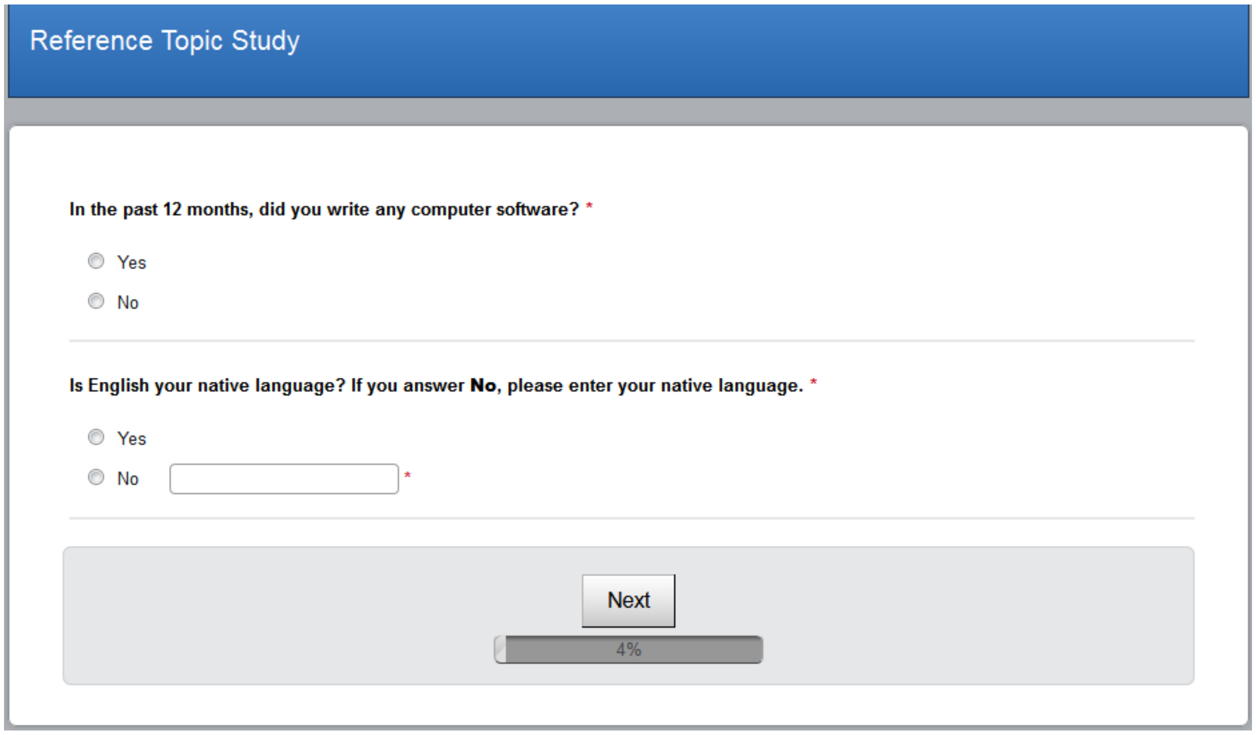
Click **Next** to confirm that you are 18 years of age or older and agree to participate in this study. If you do not wish to participate in this study, close your browser to exit.

This study has been approved by the University of Washington Human Subjects Division. Contact uwapistudy@gmail.com with questions or for more information.

Figure A-1. Welcome and consent page

Participant Demographic Page

Figure A-2 shows the demographic questionnaire in its initial, collapsed state. The sections about programming experience and English-language proficiency were collapsed initially and appear if participants indicated they had had experience writing programs. The section with the multiple choice questions about English language proficiency appeared if participants indicated they did not speak English as a native language. Figure A-3 shows the demographic questionnaire with all sections expanded.


Reference Topic Study

In the past 12 months, did you write any computer software? *

Yes

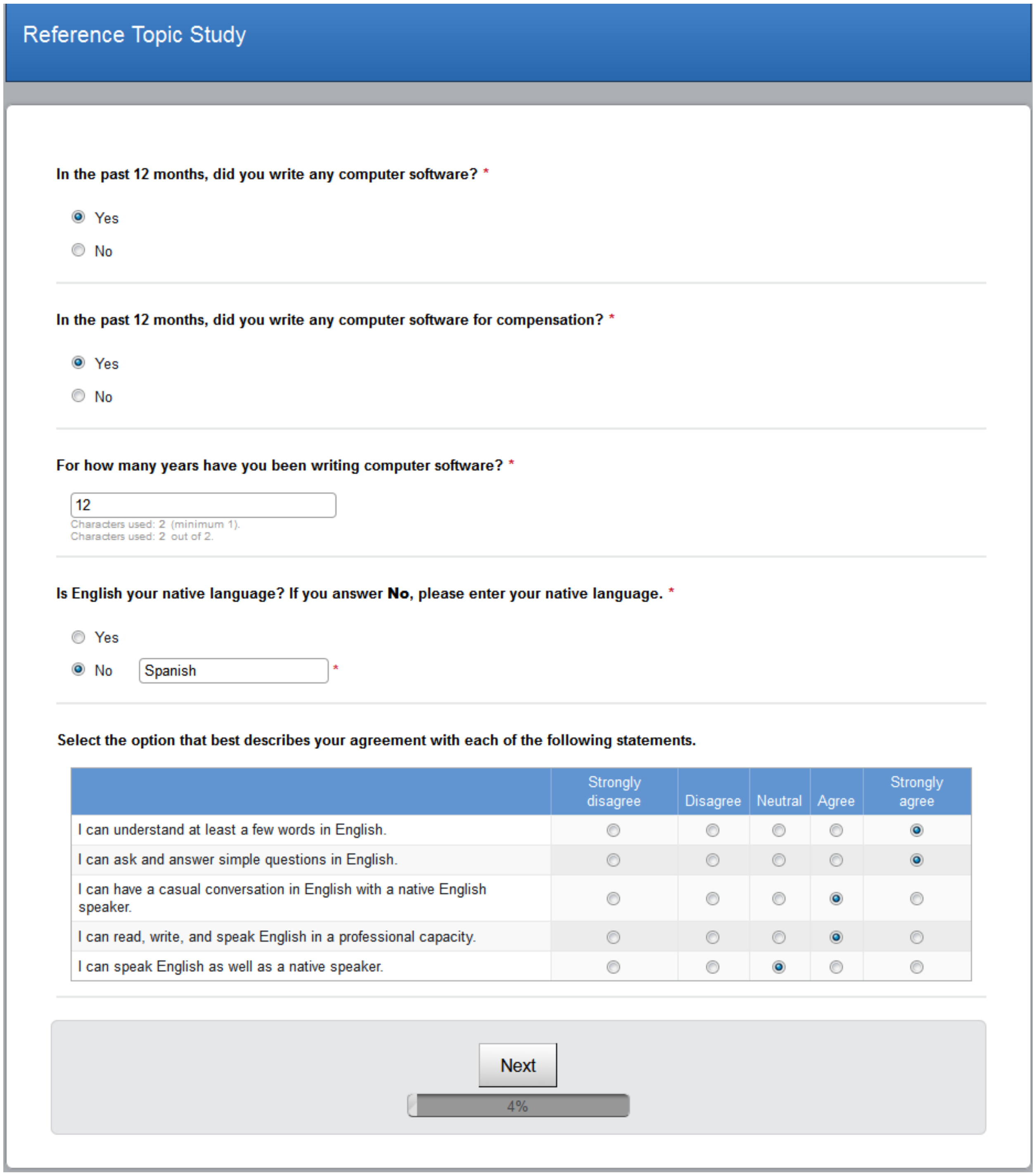
No

Is English your native language? If you answer **No**, please enter your native language. *

Yes

No *

Figure A-2. Study materials - Demographic questionnaire, collapsed



Reference Topic Study

In the past 12 months, did you write any computer software? *

[✓ Yes]
No

In the past 12 months, did you write any computer software for compensation? *

[✓ Yes]
No

For how many years have you been writing computer software? *

12
Characters used: 2 (minimum 1).
Characters used: 2 out of 2.

Is English your native language? If you answer No, please enter your native language. *

Yes
[✓ No] Spanish *

Select the option that best describes your agreement with each of the following statements.

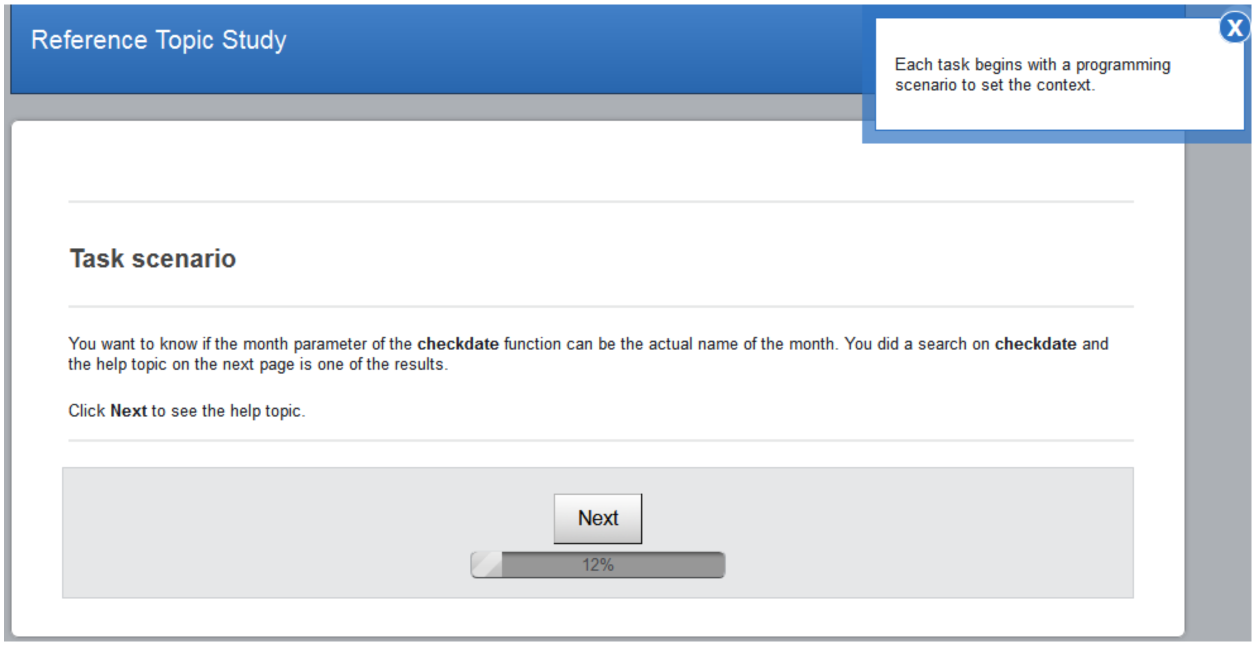
	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I can understand at least a few words in English.					✓
I can ask and answer simple questions in English.					✓
I can have a casual conversation in English with a native English speaker.				✓	
I can read, write, and speak English in a professional capacity.				✓	
I can speak English as well as a native speaker.			✓		

Figure A-3. Study materials - demographic questionnaire, fully expanded

Practice Task

Before participants started the timed study tasks, they were shown a practice task in which each step included an instruction page that overlaid the task step. After participants read the instructions, they were dismissed to reveal the underlying study page with which they could interact.

The following illustrations show the three task steps of the practice task with and without their corresponding instruction overlay.


Reference Topic Study

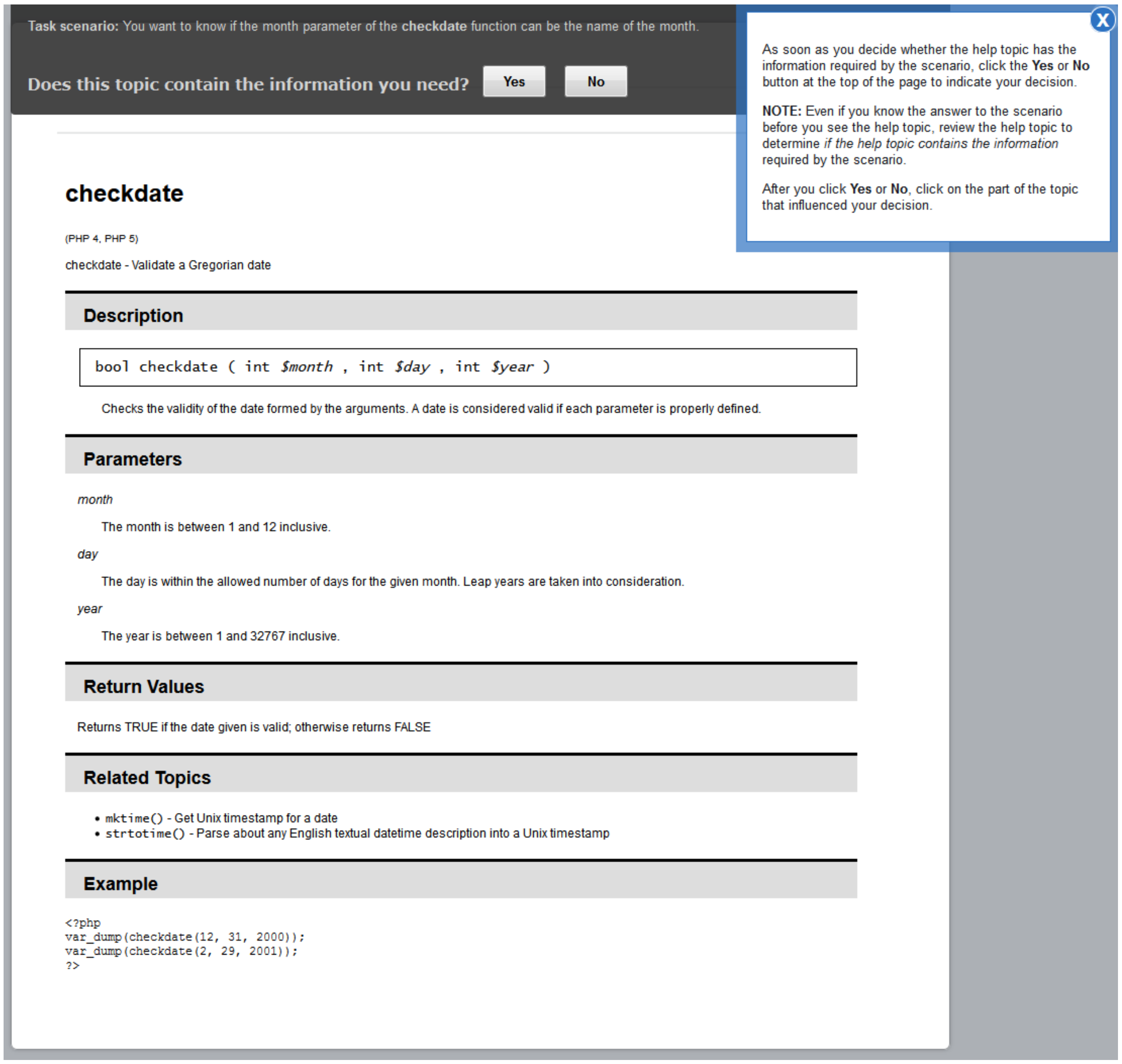
Each task begins with a programming scenario to set the context.

Task scenario

You want to know if the month parameter of the **checkdate** function can be the actual name of the month. You did a search on **checkdate** and the help topic on the next page is one of the results.

Click **Next** to see the help topic.

Figure A-4. Practice task – Scenario description with instruction overlay



Task scenario: You want to know if the month parameter of the **checkdate** function can be the name of the month.

Does this topic contain the information you need?

As soon as you decide whether the help topic has the information required by the scenario, click the **Yes** or **No** button at the top of the page to indicate your decision.

NOTE: Even if you know the answer to the scenario before you see the help topic, review the help topic to determine *if the help topic contains the information* required by the scenario.

After you click **Yes** or **No**, click on the part of the topic that influenced your decision.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

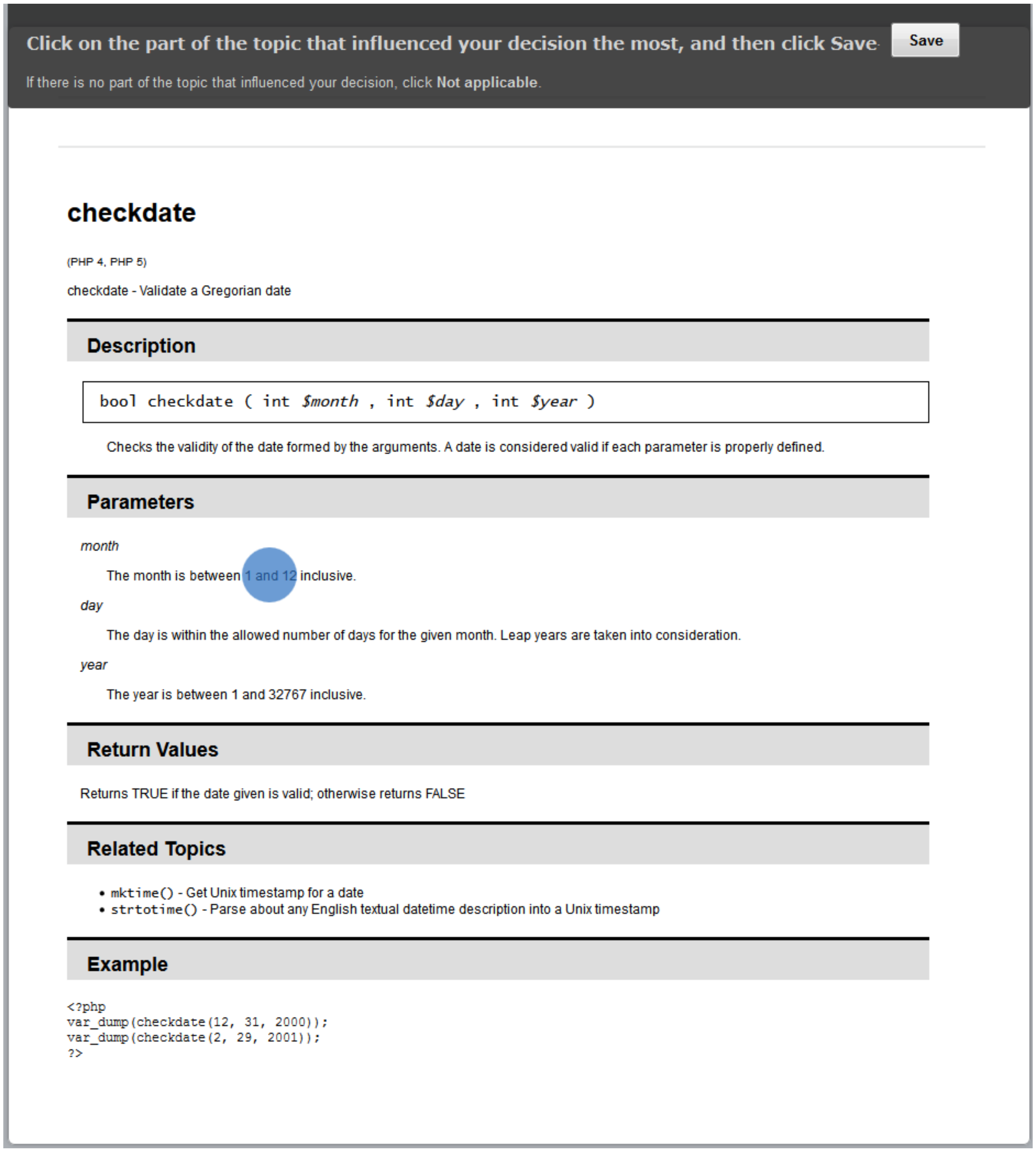
Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-5. Practice task – Topic display with instruction overlay



Click on the part of the topic that influenced your decision the most, and then click Save

If there is no part of the topic that influenced your decision, click **Not applicable**.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

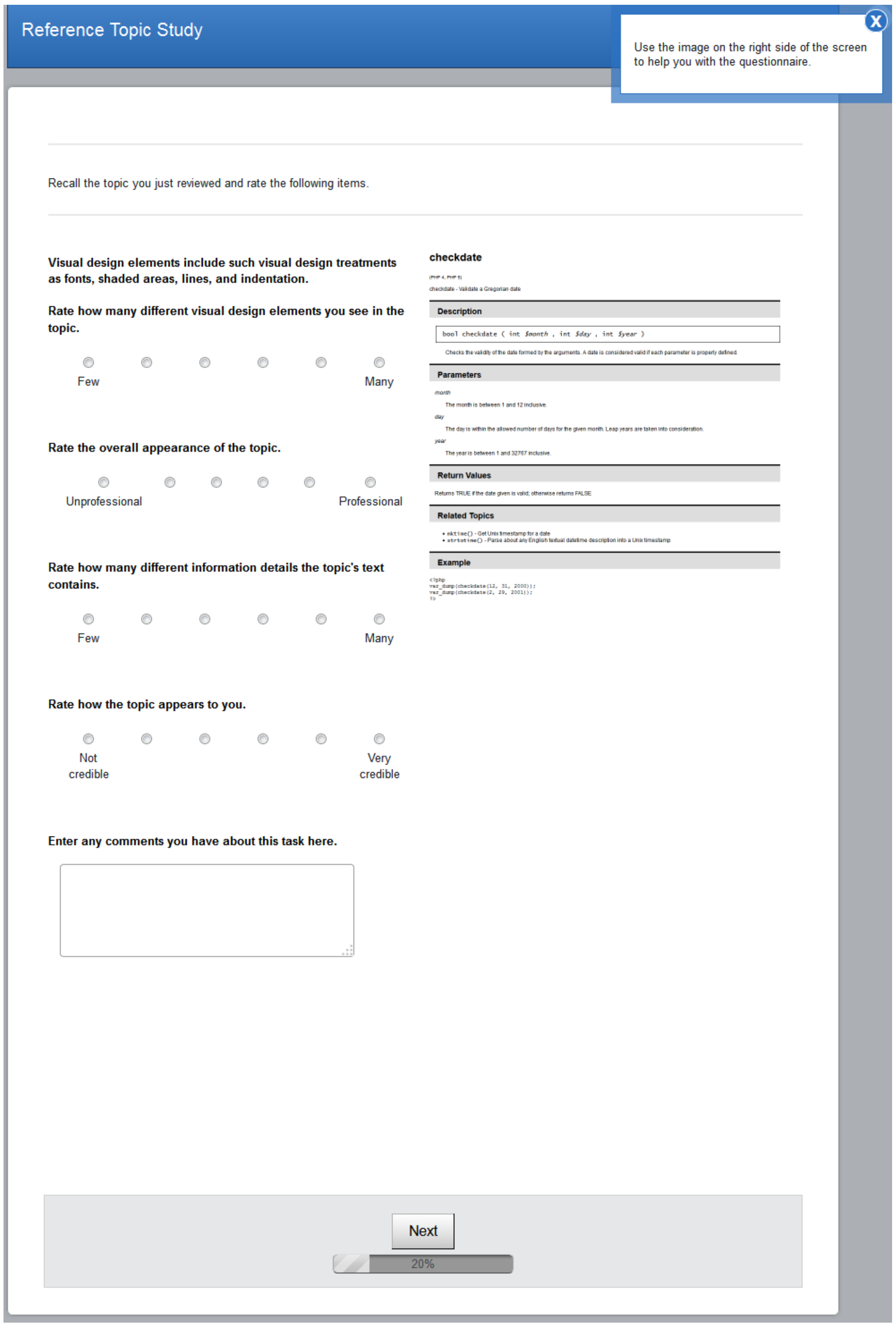
Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-6. Practice task – Topic display with most influential location annotated



Reference Topic Study

Use the image on the right side of the screen to help you with the questionnaire.

Recall the topic you just reviewed and rate the following items.

Visual design elements include such visual design treatments as fonts, shaded areas, lines, and indentation.

Rate how many different visual design elements you see in the topic.

Few ○ ○ ○ ○ ○ ○ Many

Rate the overall appearance of the topic.

Unprofessional ○ ○ ○ ○ ○ ○ Professional

Rate how many different information details the topic's text contains.

Few ○ ○ ○ ○ ○ ○ Many

Rate how the topic appears to you.

Not credible ○ ○ ○ ○ ○ ○ Very credible

Enter any comments you have about this task here.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

Related Topics

- mktime() - Get Unix timestamp for a date
- strtotime() - Parse about any English textual datetime description into a Unix timestamp

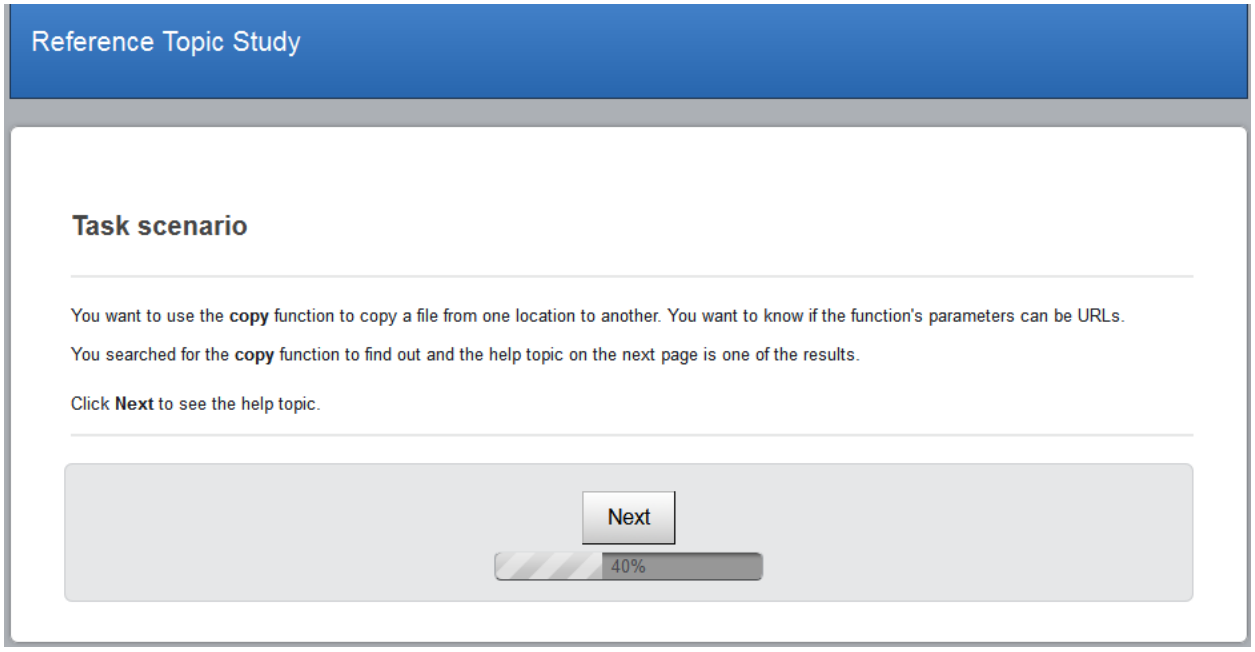
Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-7. Practice task – Task questionnaire with instruction overlay

Task Scenario Description

Each task began with a brief description of the scenario in which participants were shown a screen similar to Figure A-8. Participants were asked to imagine themselves in this scenario when they evaluated the API reference topic. For each task, there were two possible scenarios—one for each of the two levels of the Topic Relevance variable. The scenarios for each study task are listed in Table 5.



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

Click **Next** to see the help topic.

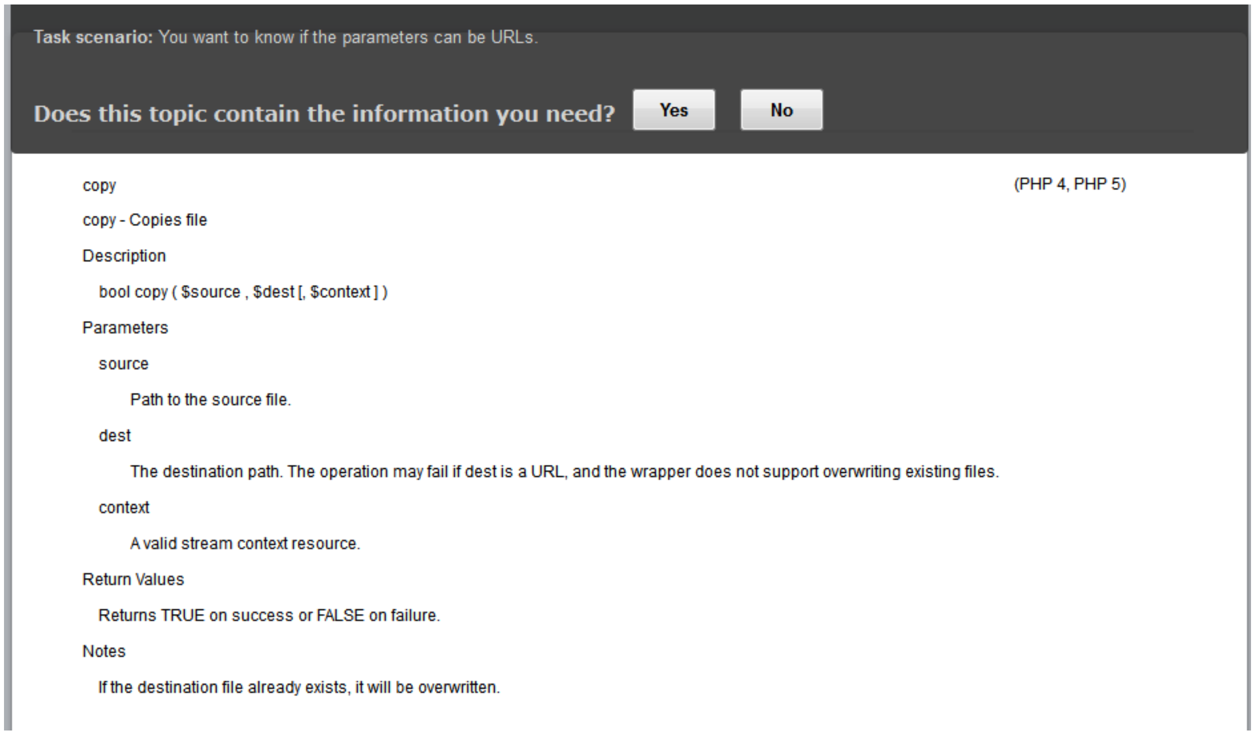
Figure A-8. Example of task scenario page in study

Topic Visual-Design Element Variations

The visual-design element count of each topic variation was controlled by the application of a Cascading Style Sheet (CSS), which is described in a .css file. Figure A-33 lists the styles defined for the High VDEC topic variations and Figure A-34 lists the styles defined for the Low VDEC topic variations.

Topic Pages

Figure A-12 through Figure A-27 illustrate the different topic page variations used in the study. A participant would see only one topic per task and only one variation in their study session tasks; however, the study included four different API reference topic variations (see Table 8 for the topic variations matrix). To collect the study data, participants were shown the API reference topic in the context of two overlays. Figure A-9 shows the relevance-assessment decision overlay that appeared above the API reference topic and asked participants if the topic contained the information required by the task scenario. After they assessed the topic's relevance, they were shown the overlay of the spot interaction shown in Figure A-10 and asked to indicate the part of the topic that influenced their [preceding] decision the most. Participants did this by clicking on the topic, which left a blue circle on the topic where they clicked. After leaving a blue spot, they clicked the **Save** button at the top of the page, which took them to the post-task questionnaire.


Task scenario: You want to know if the parameters can be URLs.
Does this topic contain the information you need?
Yes
No

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

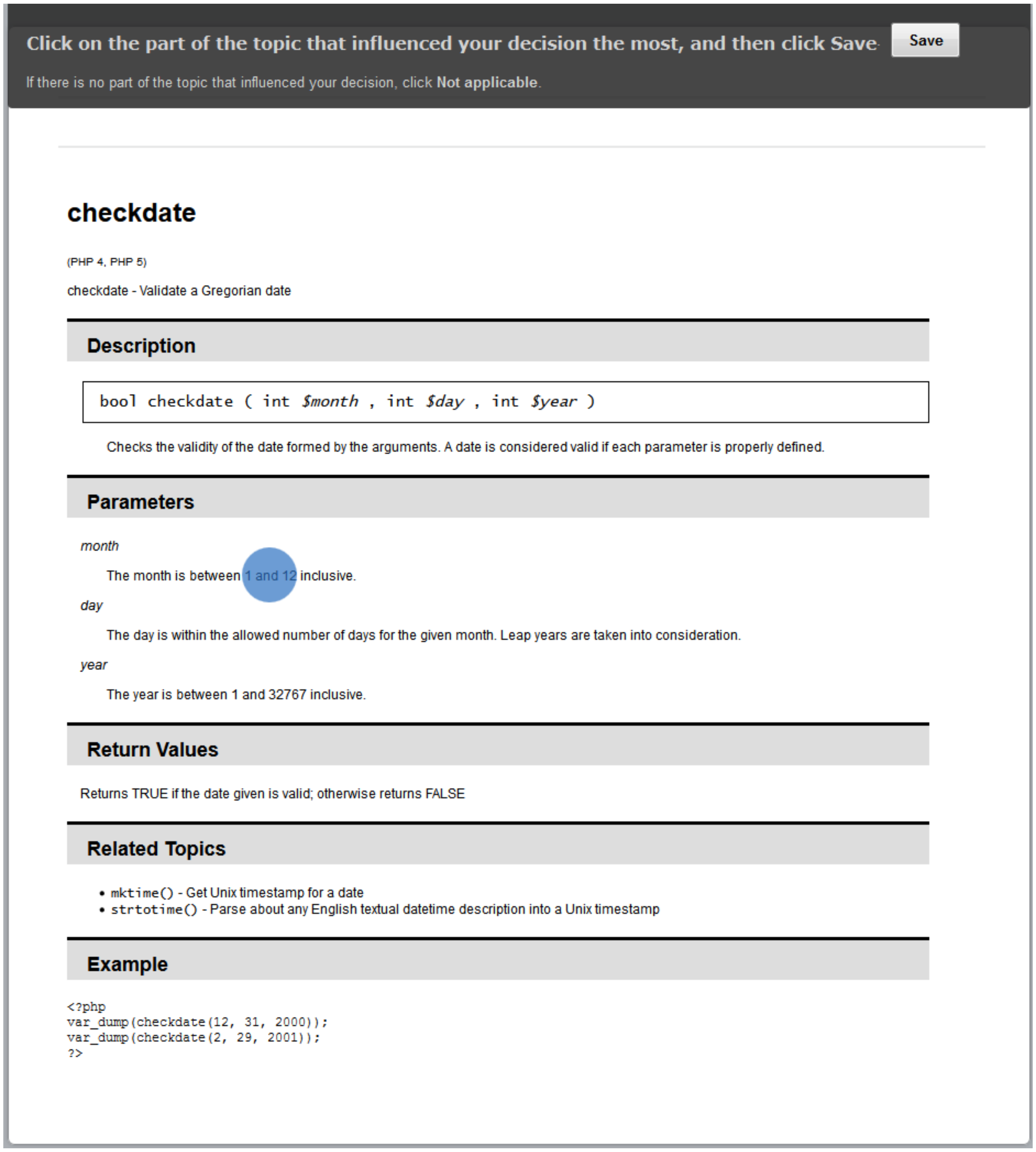
Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure A-9. Example of study-task decision overlay


Click on the part of the topic that influenced your decision the most, and then click Save

If there is no part of the topic that influenced your decision, click **Not applicable**.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

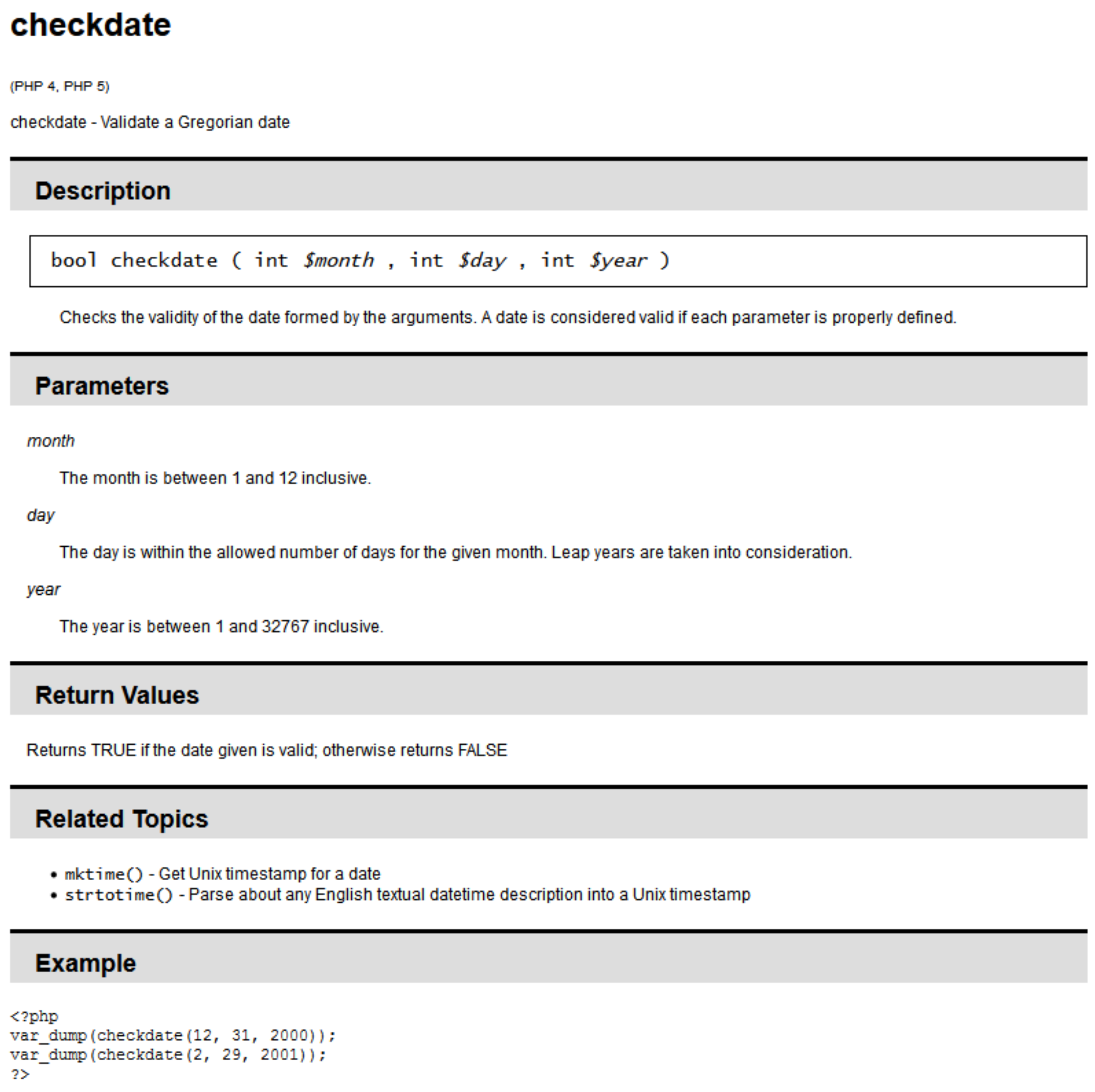
Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-10. Example of study-task topic-location marker overlay



checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

```
bool checkdate ( int $month , int $day , int $year )
```

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

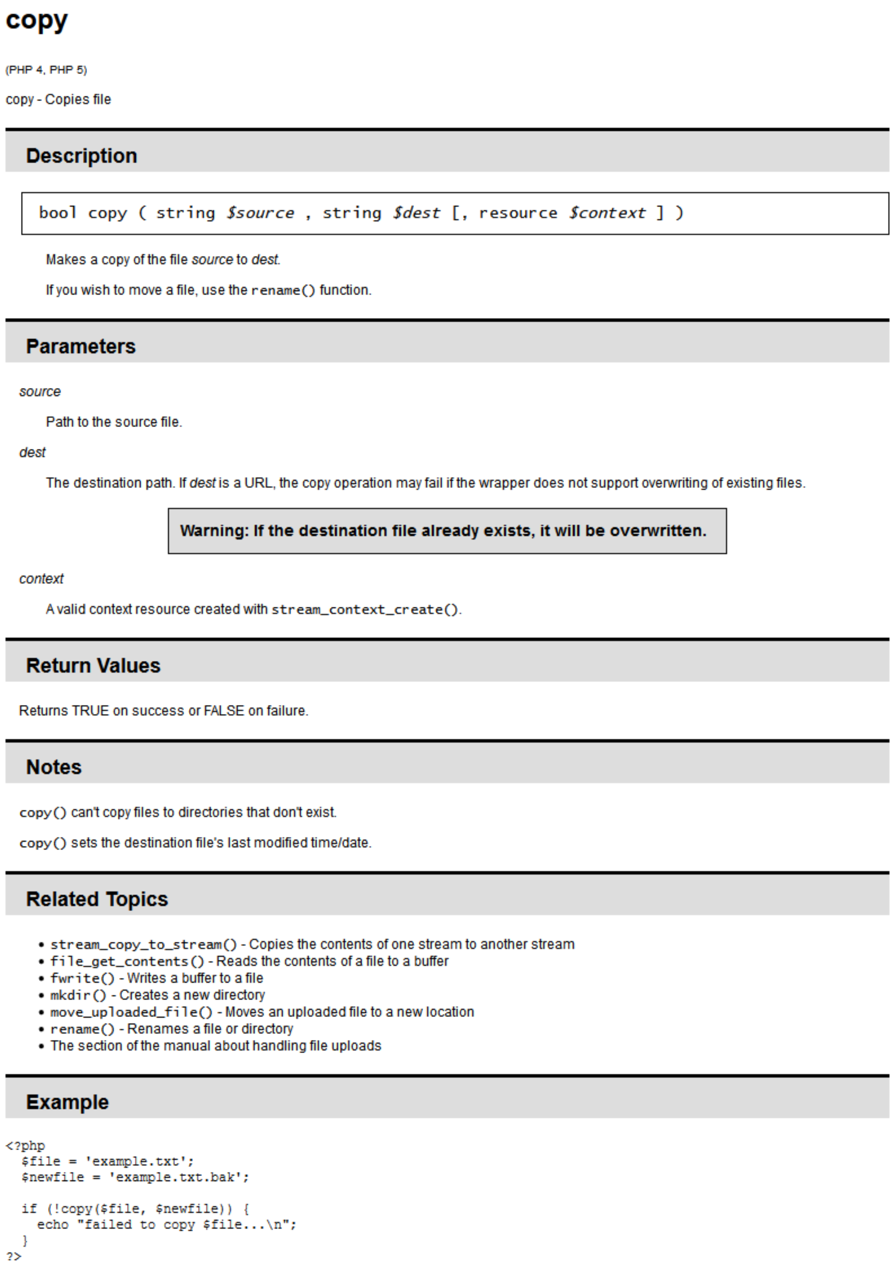
Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-11. API reference topic for practice task



copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( string $source , string $dest [, resource $context ] )
```

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the `rename()` function.

Parameters

source

Path to the source file.

dest

The destination path. If *dest* is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with `stream_context_create()`.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

`copy()` can't copy files to directories that don't exist.

`copy()` sets the destination file's last modified time/date.

Related Topics

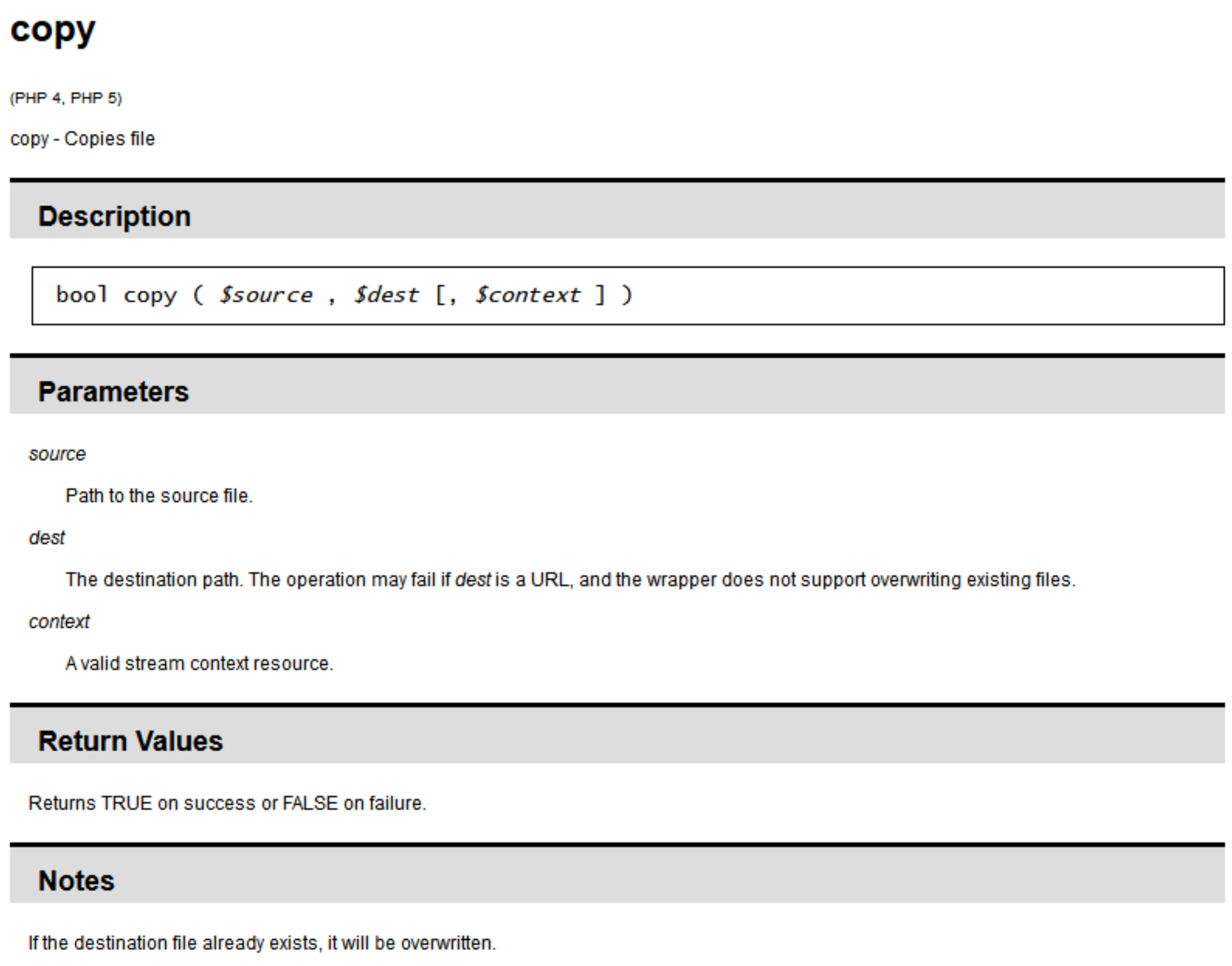
- `stream_copy_to_stream()` - Copies the contents of one stream to another stream
- `file_get_contents()` - Reads the contents of a file to a buffer
- `fwrite()` - Writes a buffer to a file
- `mkdir()` - Creates a new directory
- `move_uploaded_file()` - Moves an uploaded file to a new location
- `rename()` - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
  $file = 'example.txt';
  $newfile = 'example.txt.bak';

  if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
  }
?>
```

Figure A-12. API reference topic for Task 1 – Variation 1

Task 1, Variation 1	**VDEC**	High (12)
	ICEC	High (20)



copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( $source , $dest [, $context ] )
```

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if *dest* is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

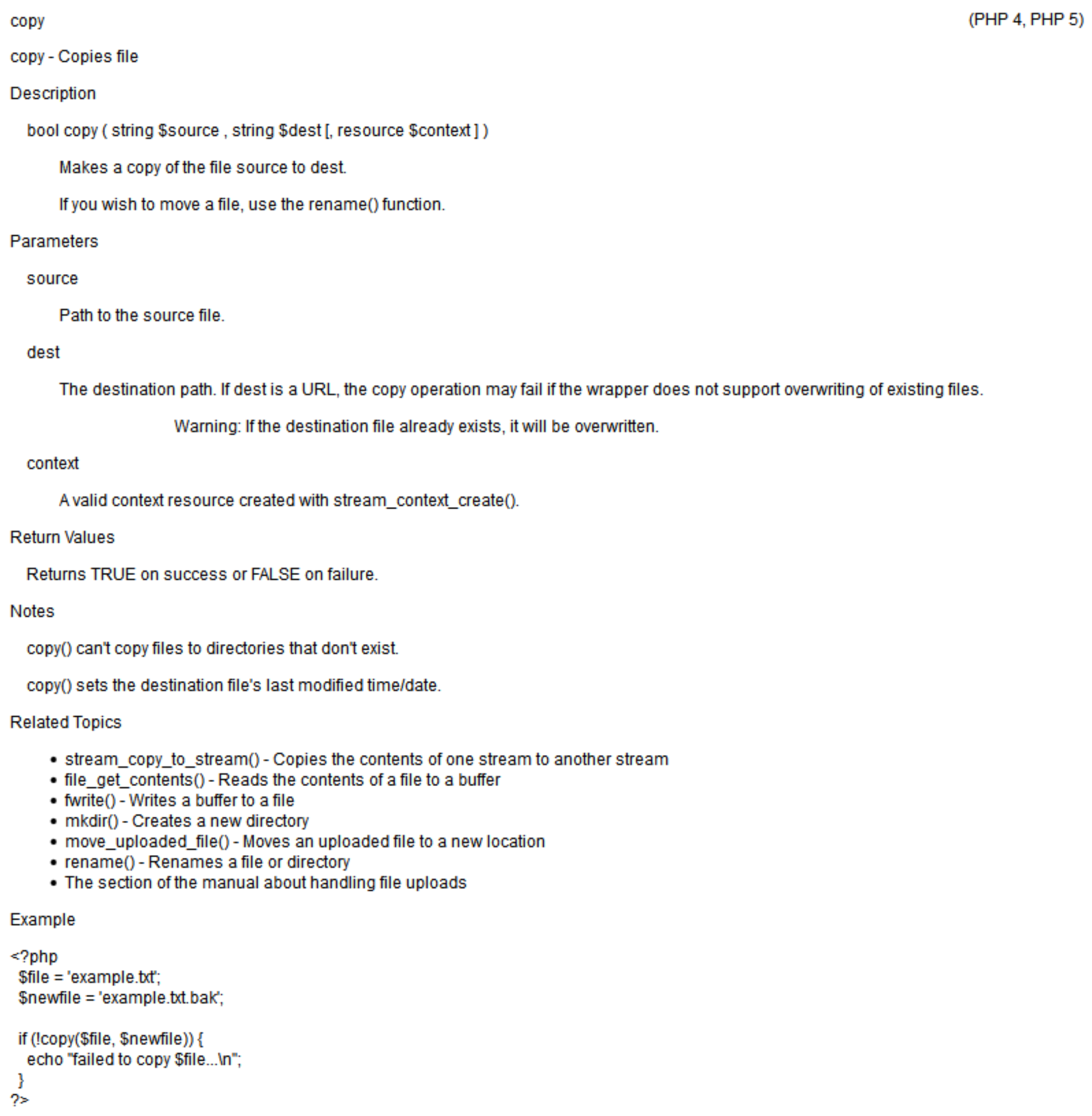
Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure A-13. API reference topic for Task 1 – Variation 2

Task 1, Variation 2	**VDEC**	High (12)
	ICEC	Low (8)



copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy (string $source , string $dest [, resource $context])

Makes a copy of the file source to dest.

If you wish to move a file, use the rename() function.

Parameters

source

Path to the source file.

dest

The destination path. If dest is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with stream_context_create().

Return Values

Returns TRUE on success or FALSE on failure.

Notes

copy() can't copy files to directories that don't exist.

copy() sets the destination file's last modified time/date.

Related Topics

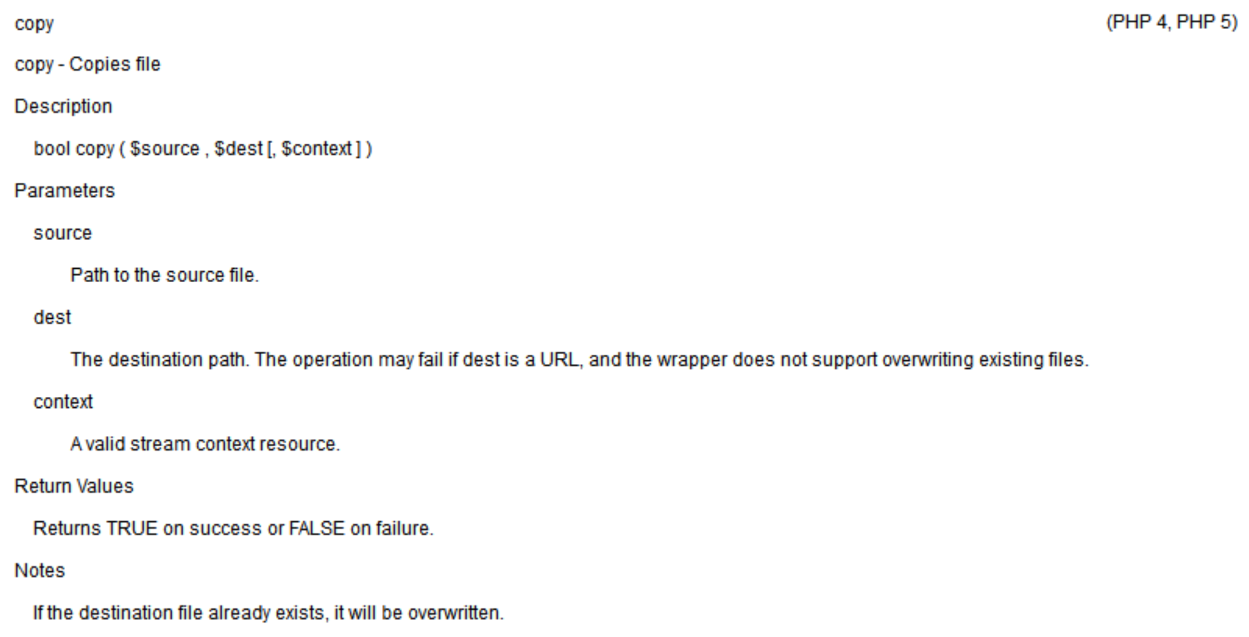
- stream_copy_to_stream() - Copies the contents of one stream to another stream
- file_get_contents() - Reads the contents of a file to a buffer
- fwrite() - Writes a buffer to a file
- mkdir() - Creates a new directory
- move_uploaded_file() - Moves an uploaded file to a new location
- rename() - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
 $file = 'example.txt';
 $newfile = 'example.txt.bak';

 if (!copy($file, $newfile)) {
   echo "failed to copy $file...\n";
 }
?>
```

Figure A-14. API reference topic for Task 1 – Variation 3

Task 1, Variation 3	**VDEC**	Low (5)
	ICEC	High (20)



copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

Return Values

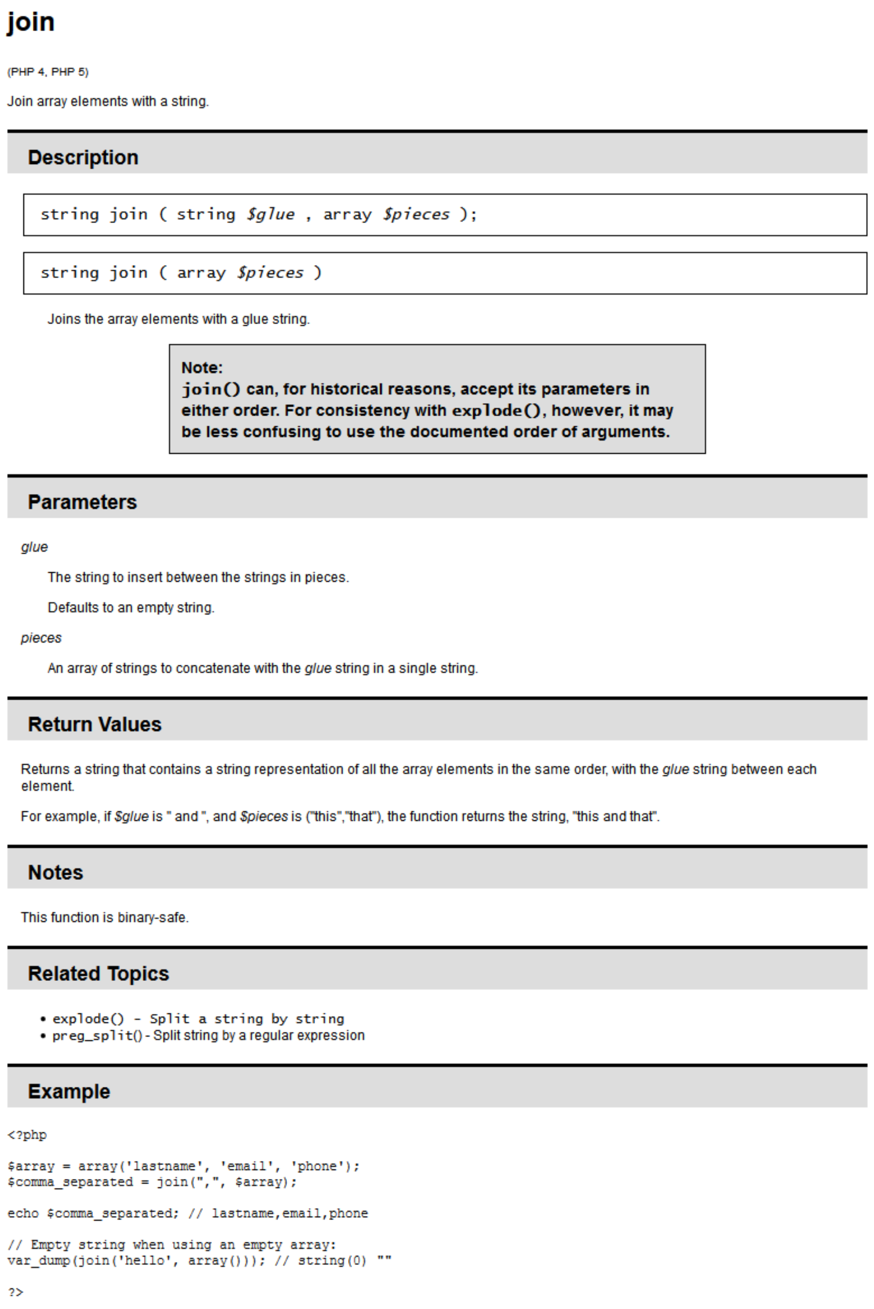
Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure A-15. API reference topic for Task 1 – Variation 4

Task 1, Variation 4	**VDEC**	Low (5)
	ICEC	Low (8)



join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

```
string join ( array $pieces )
```

Joins the array elements with a glue string.

> **Note:**
> **join() can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.**

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the *glue* string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the *glue* string between each element.

For example, if *$glue* is " and ", and *$pieces* is ("this","that"), the function returns the string, "this and that".

Notes

This function is binary-safe.

Related Topics

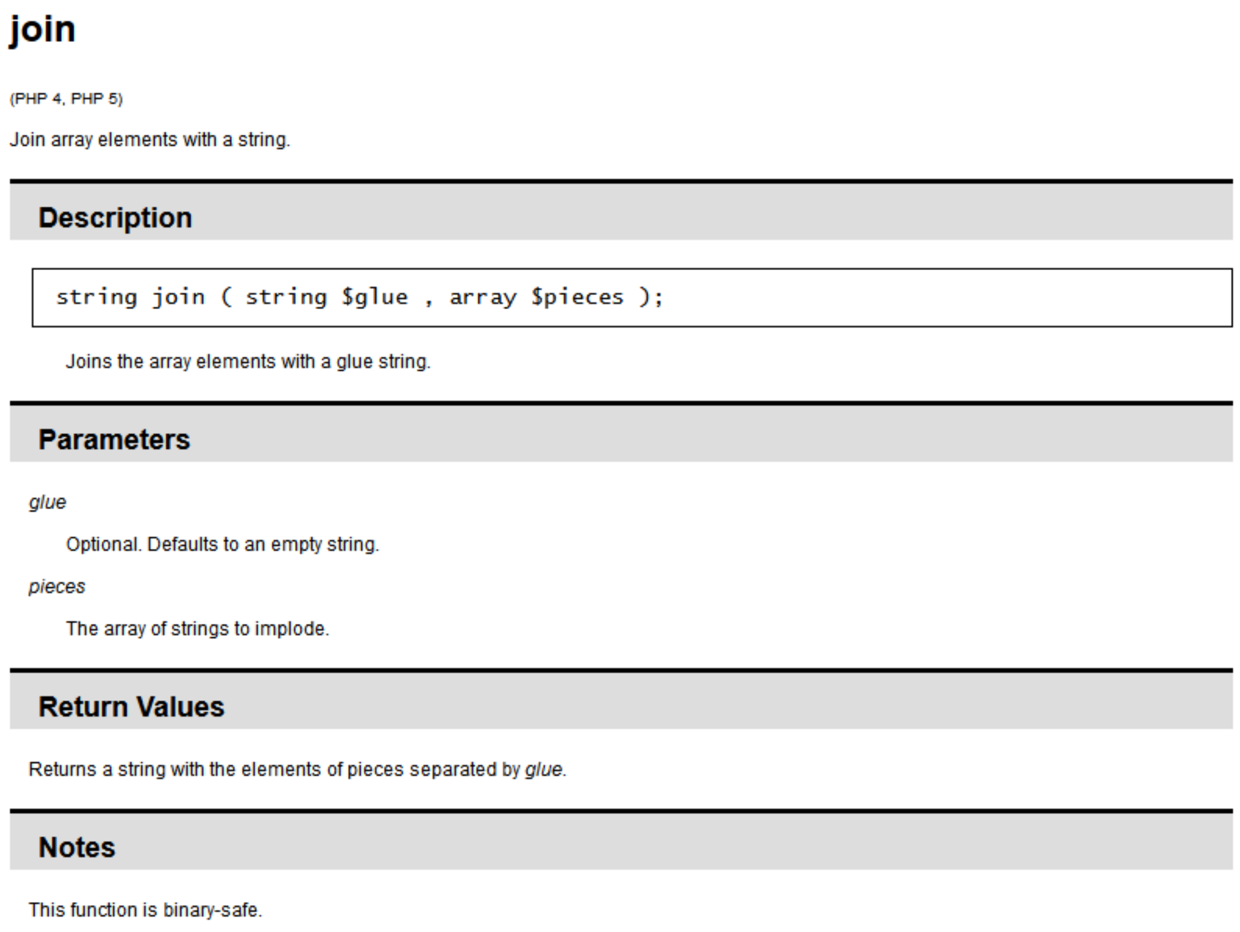
- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure A-16. API reference topic for Task 2 – Variation 1

Task 2, Variation 1	**VDEC**	High (12)
	ICEC	High (15)



join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

Return Values

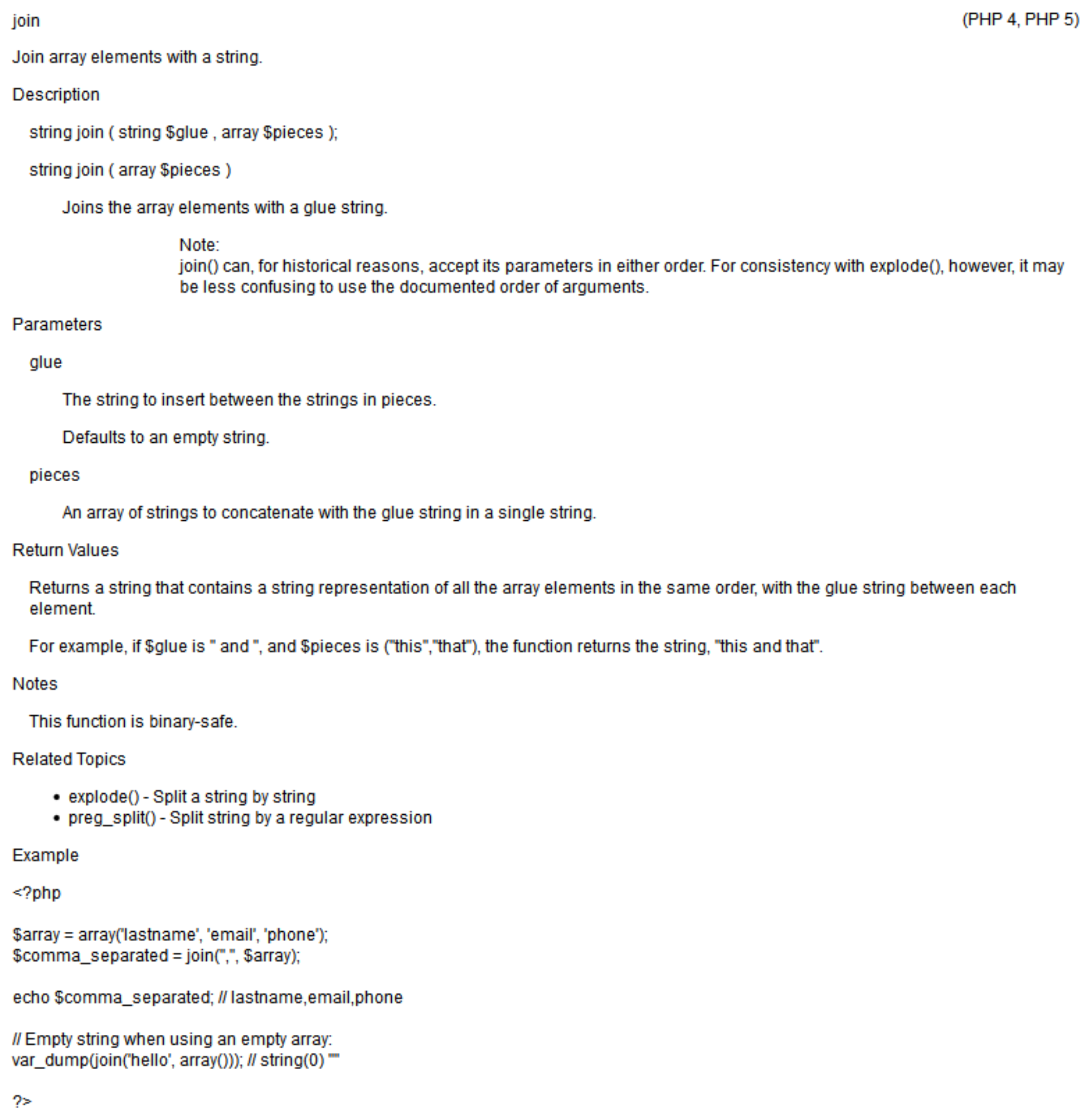
Returns a string with the elements of pieces separated by *glue*.

Notes

This function is binary-safe.

Figure A-17. API reference topic for Task 2 – Variation 2

Task 2, Variation 2	**VDEC**	High (12)
	ICEC	Low (8)



join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

string join (array $pieces)

Joins the array elements with a glue string.

Note:
join() can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the glue string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the glue string between each element.

For example, if $glue is " and ", and $pieces is ("this","that"), the function returns the string, "this and that".

Notes

This function is binary-safe.

Related Topics

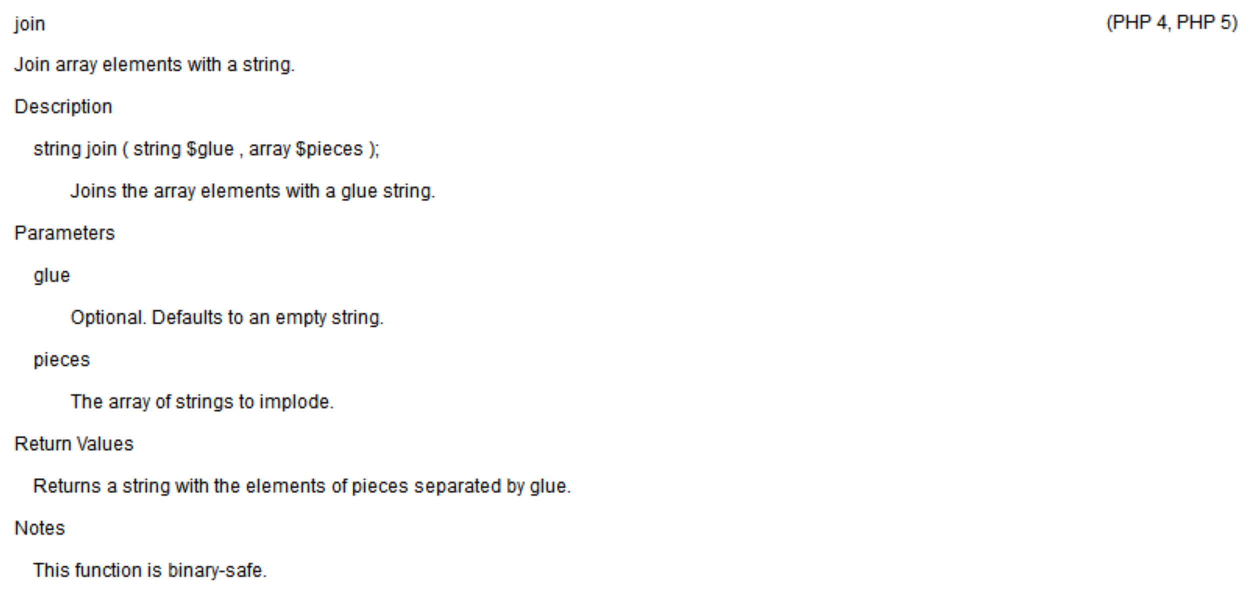
- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure A-18. API reference topic for Task 2 – Variation 3

Task 2, Variation 3	**VDEC**	Low (5)
	ICEC	High (15)



join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

Return Values

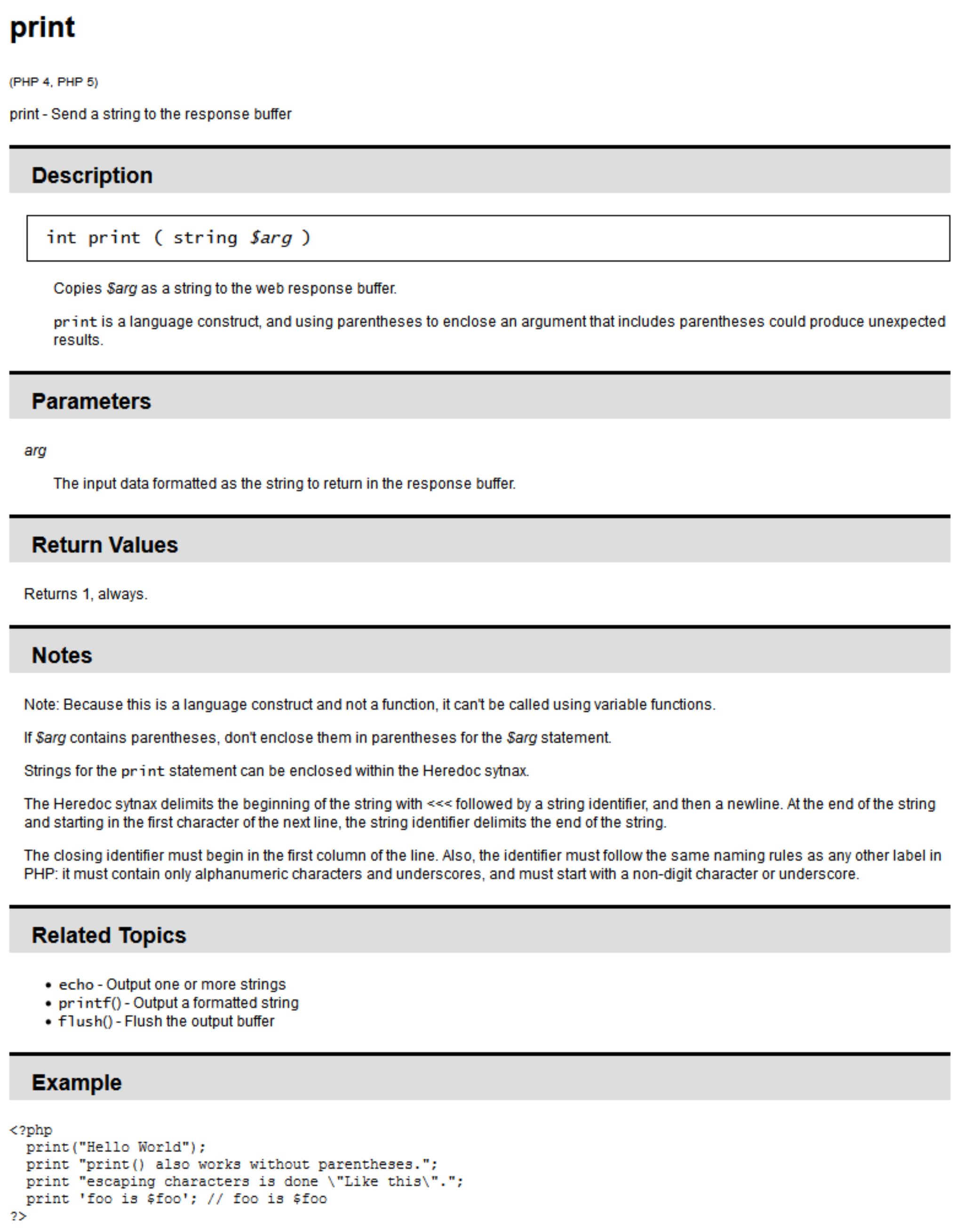
Returns a string with the elements of pieces separated by glue.

Notes

This function is binary-safe.

Figure A-19. API reference topic for Task 2 – Variation 4

Task 2, Variation 4	**VDEC**	Low (5)
	ICEC	Low (8)



print

(PHP 4, PHP 5)

print - Send a string to the response buffer

Description

```
int print ( string $arg )
```

Copies *$arg* as a string to the web response buffer.

`print` is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If *$arg* contains parentheses, don't enclose them in parentheses for the *$arg* statement.

Strings for the `print` statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Related Topics

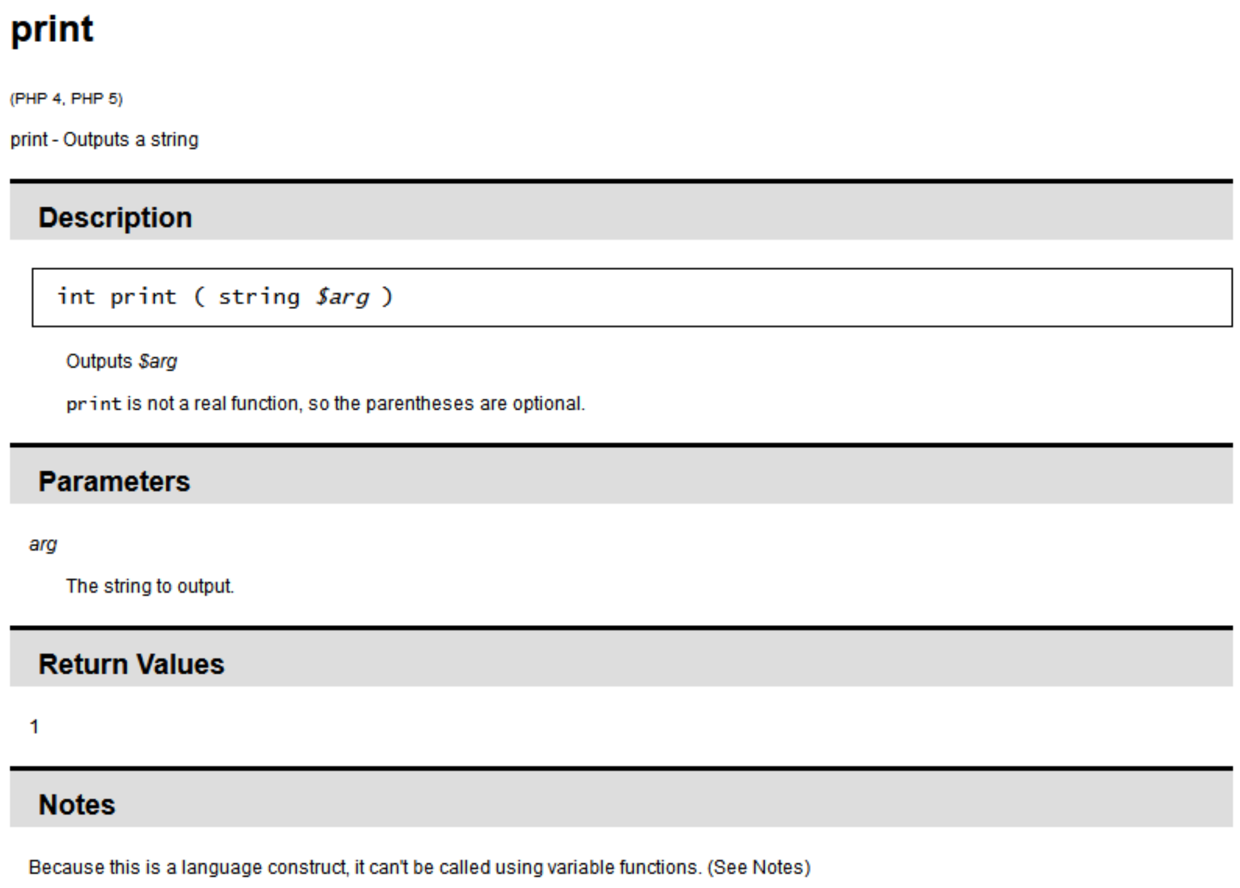
- `echo` - Output one or more strings
- `printf`() - Output a formatted string
- `flush`() - Flush the output buffer

Example

```
<?php
  print("Hello World");
  print "print() also works without parentheses.";
  print "escaping characters is done \"Like this\".";
  print 'foo is $foo'; // foo is $foo
?>
```

Figure A-20. API reference topic for Task 3 – Variation 1

Task 3, Variation 1	**VDEC**	High (12)
	ICEC	High (18)



print

(PHP 4, PHP 5)

print - Outputs a string

Description

```
int print ( string $arg )
```

Outputs *$arg*

`print` is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

Return Values

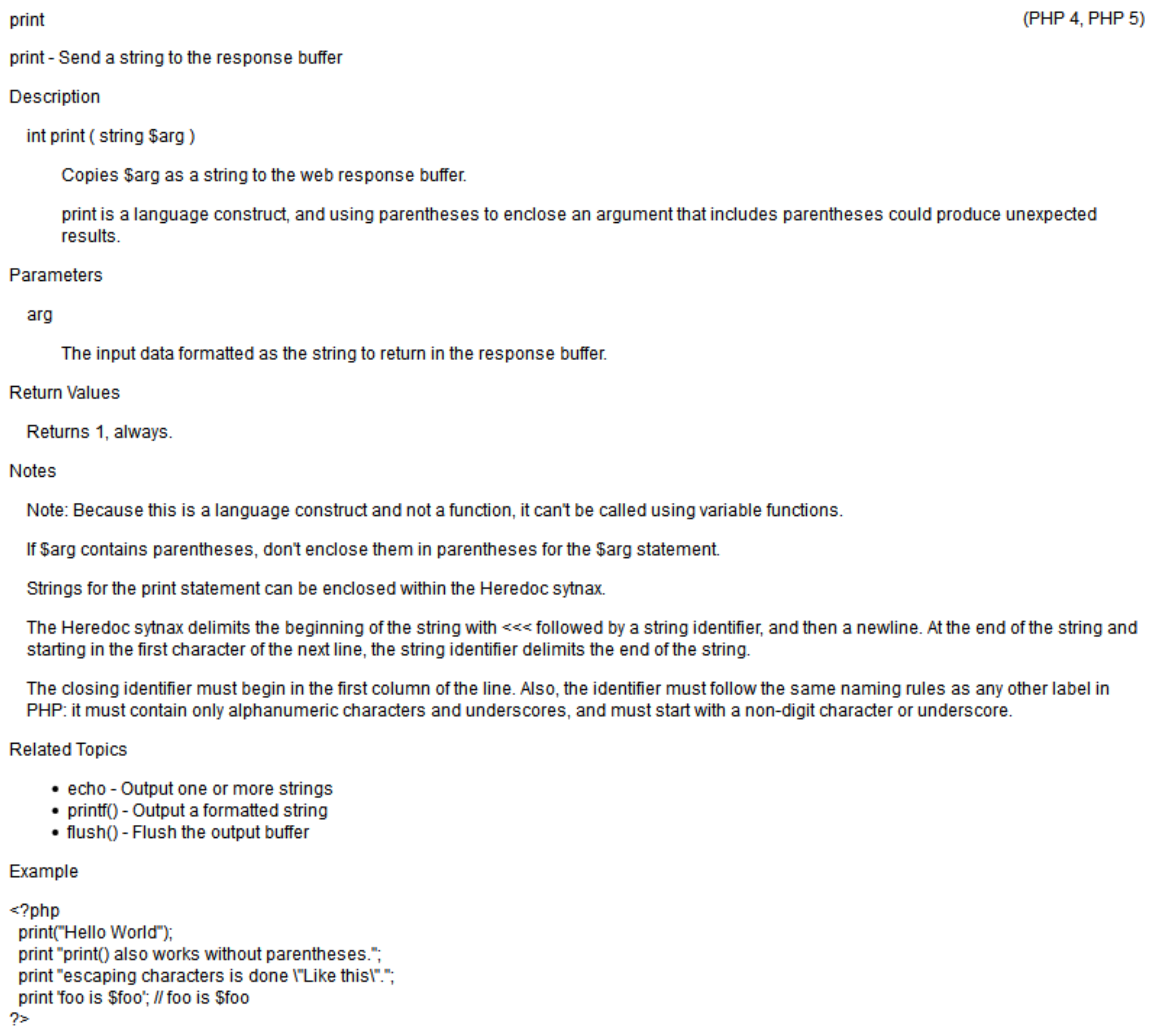
1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure A-21. API reference topic for task 3 – Variation 2

Task 3, Variation 2	**VDEC**	High (12)
	ICEC	Low (8)



print (PHP 4, PHP 5)

print - Send a string to the response buffer

Description

int print (string $arg)

Copies $arg as a string to the web response buffer.

print is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If $arg contains parentheses, don't enclose them in parentheses for the $arg statement.

Strings for the print statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Related Topics

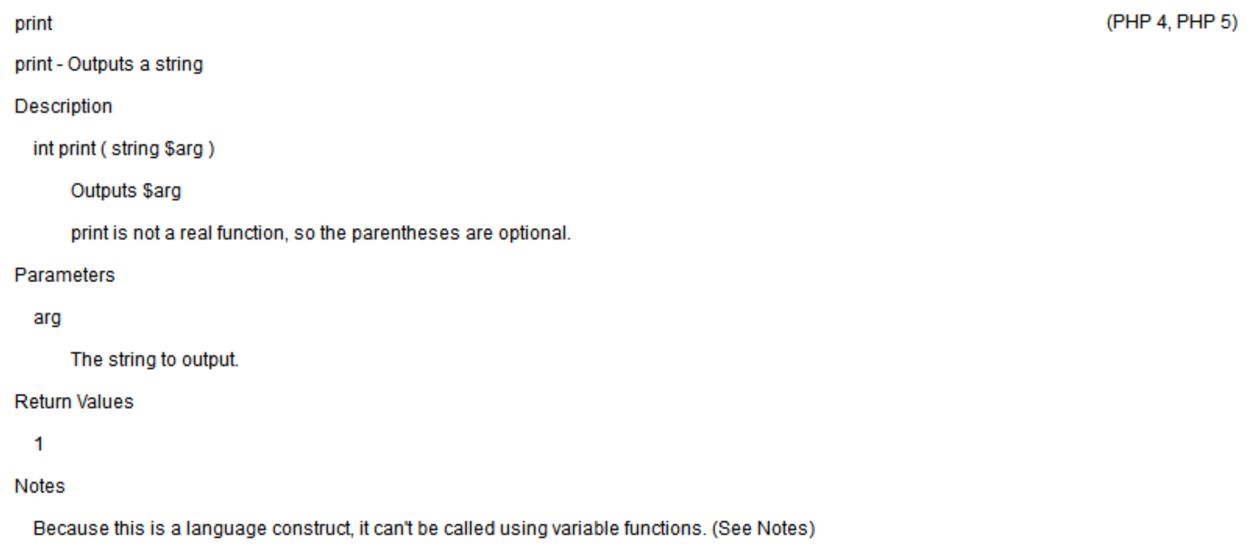
- echo - Output one or more strings
- printf() - Output a formatted string
- flush() - Flush the output buffer

Example

```
<?php
  print("Hello World");
  print "print() also works without parentheses.";
  print "escaping characters is done \"Like this\".";
  print 'foo is $foo'; // foo is $foo
?>
```

Figure A-22. API reference topic for Task 3 – Variation 3

Task 3, Variation 3	**VDEC**	Low (5)
	ICEC	High (18)



print (PHP 4, PHP 5)

print - Outputs a string

Description

int print (string $arg)

Outputs $arg

print is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

Return Values

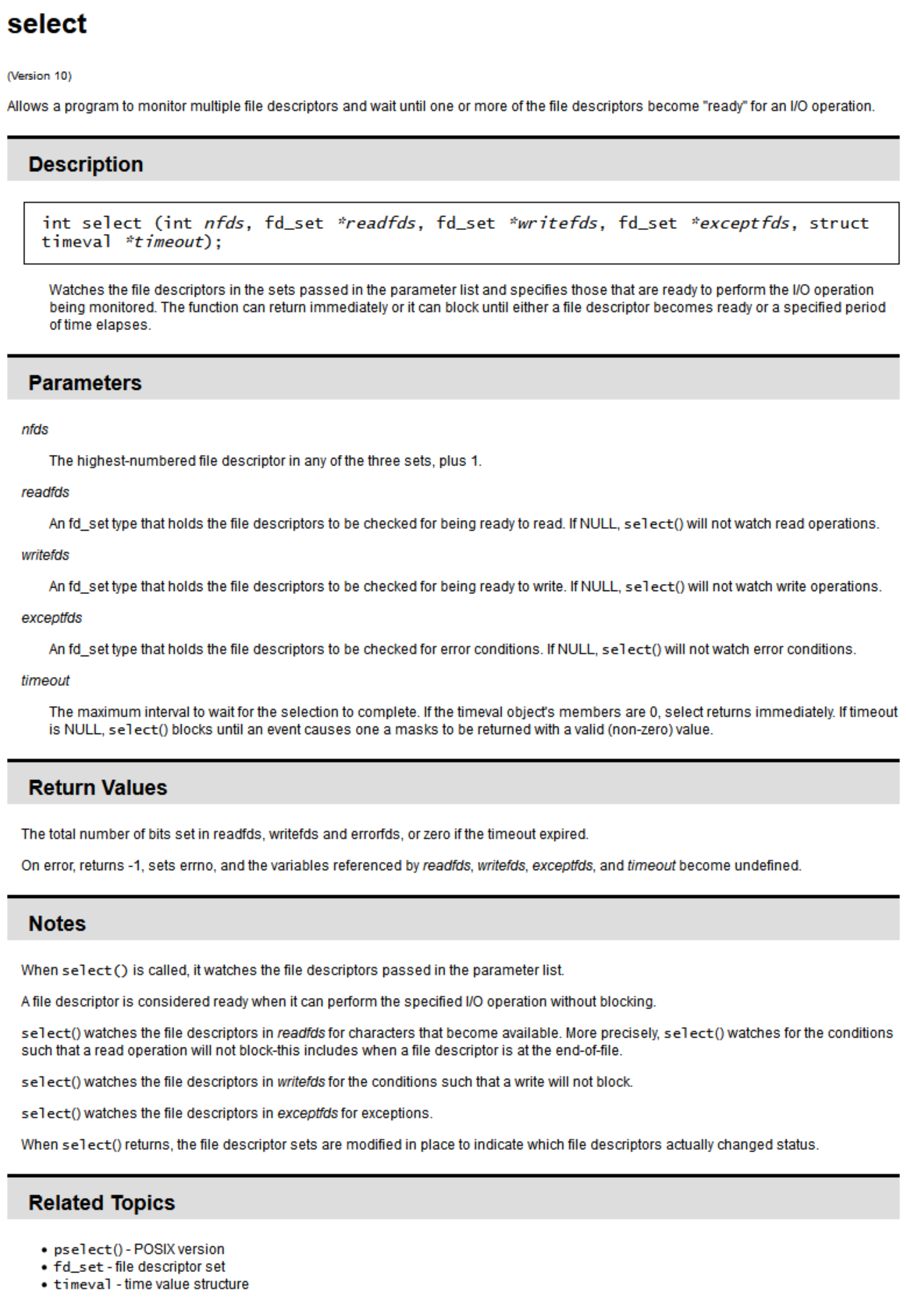
1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure A-23. API reference topic for Task 3 – Variation 4

Task 3, Variation 4	**VDEC**	Low (5)
	ICEC	Low (8)



select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, `select()` will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, `select()` will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, `select()` will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, `select()` blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by *readfds*, *writefds*, *exceptfds*, and *timeout* become undefined.

Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

`select()` watches the file descriptors in *readfds* for characters that become available. More precisely, `select()` watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

`select()` watches the file descriptors in *writefds* for the conditions such that a write will not block.

`select()` watches the file descriptors in *exceptfds* for exceptions.

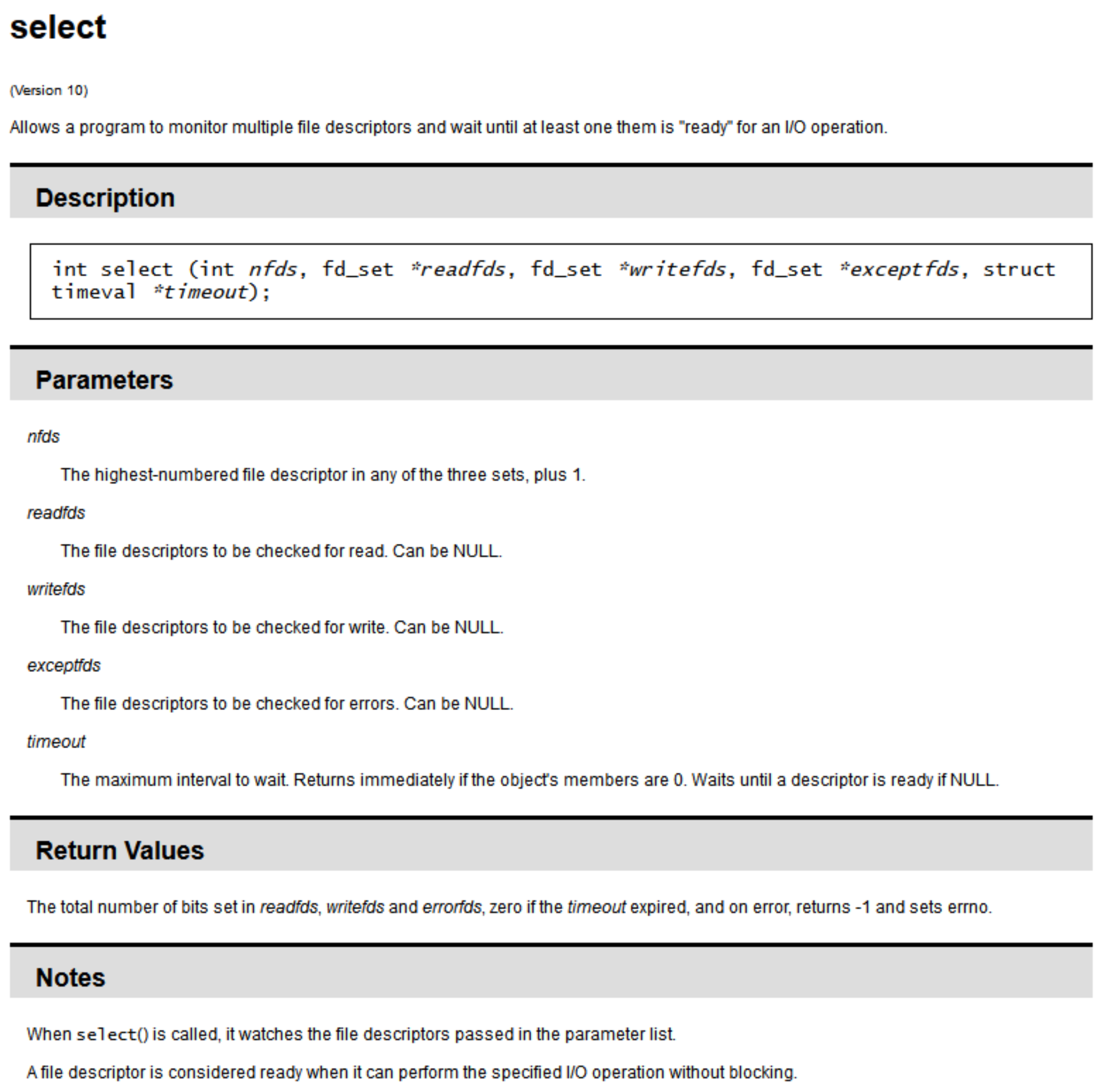
When `select()` returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- `pselect()` - POSIX version
- `fd_set` - file descriptor set
- `timeval` - time value structure

Figure A-24. API reference topic for Task 4 – Variation 1

Task 4, Variation 1	**VDEC**	High (12)
	ICEC	High (21)



select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

The total number of bits set in *readfds*, *writefds* and *errorfds*, zero if the *timeout* expired, and on error, returns -1 and sets errno.

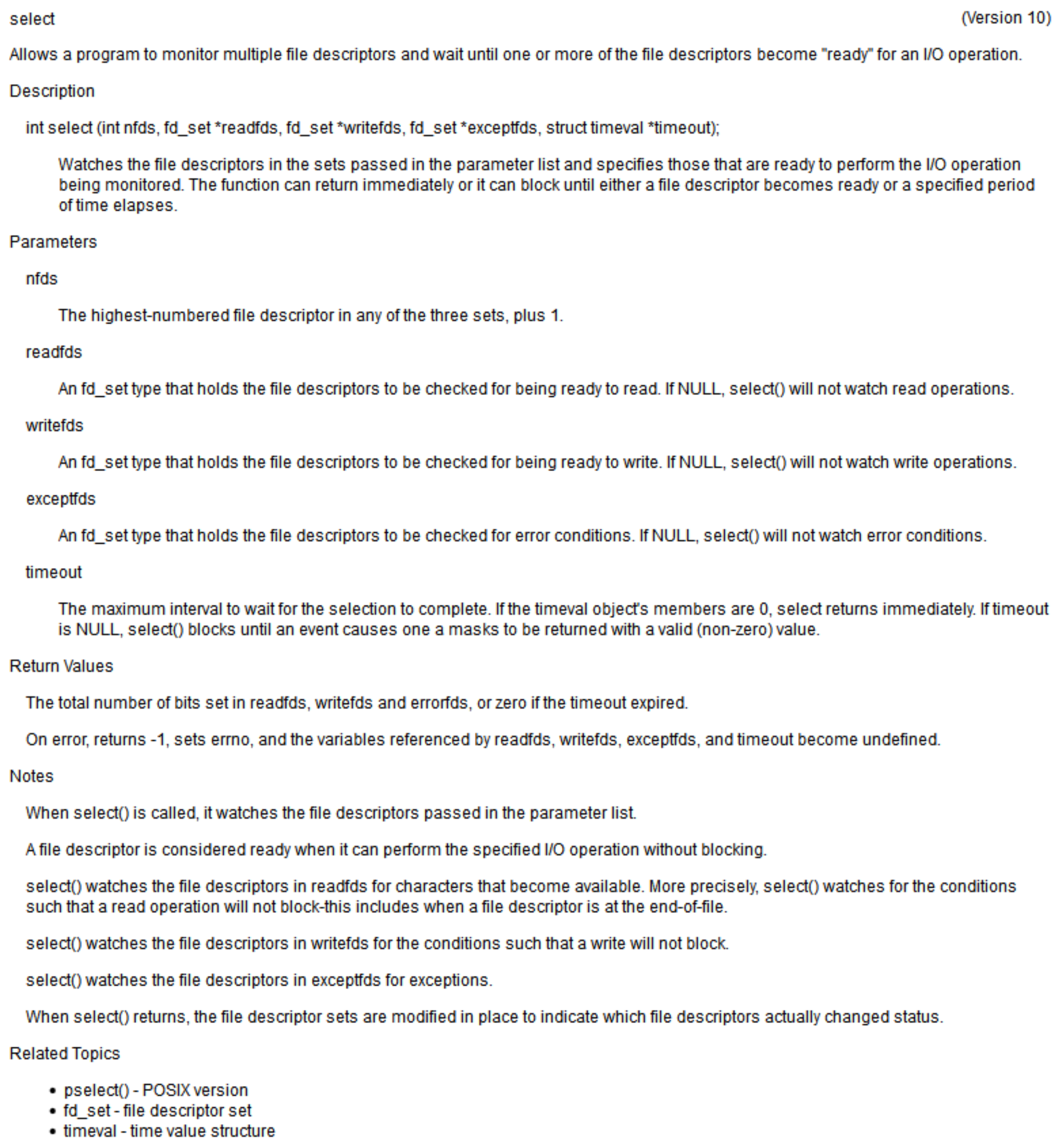
Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure A-25. API reference topic for Task 4 – Variation 2

Task 4, Variation 2	**VDEC**	High (12)
	ICEC	Low (11)



select (Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, select() will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, select() will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, select() will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, select() blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by readfds, writefds, exceptfds, and timeout become undefined.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

select() watches the file descriptors in readfds for characters that become available. More precisely, select() watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

select() watches the file descriptors in writefds for the conditions such that a write will not block.

select() watches the file descriptors in exceptfds for exceptions.

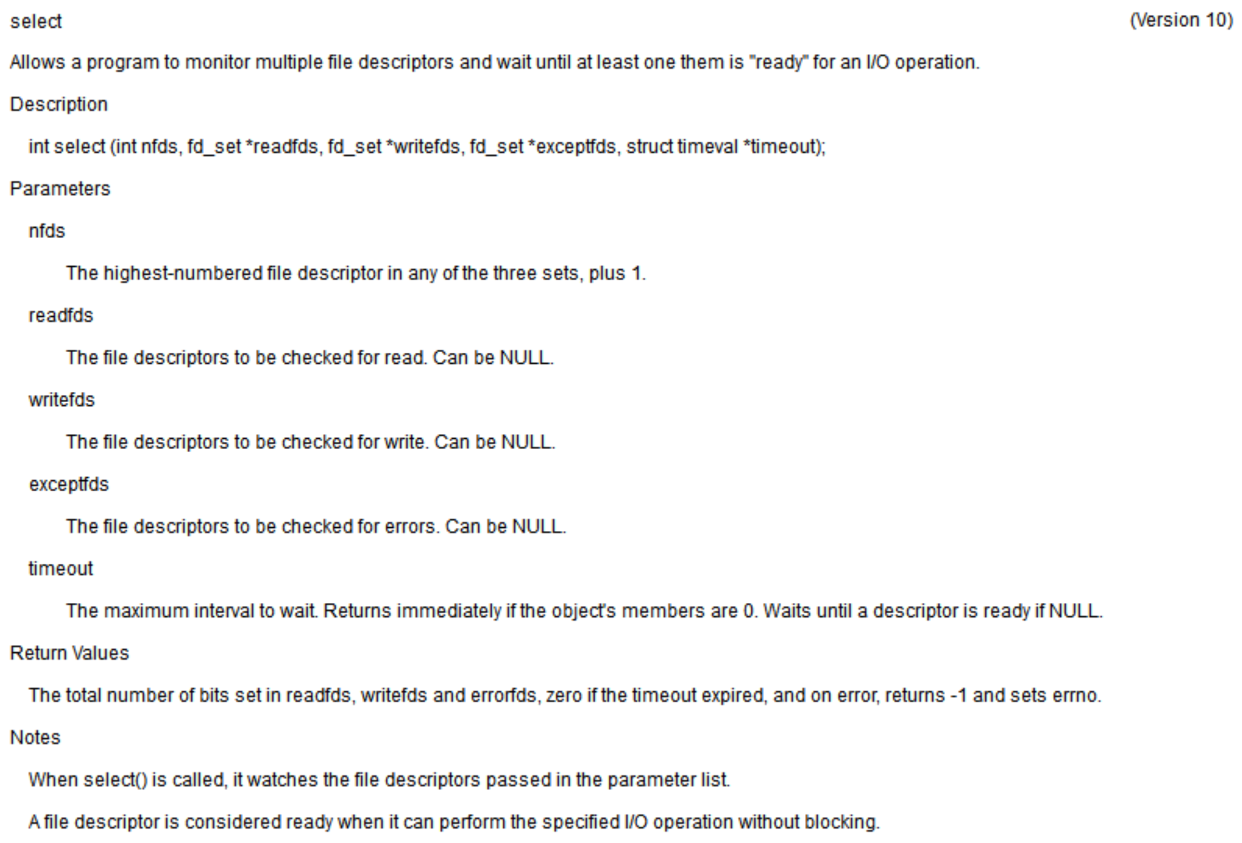
When select() returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- pselect() - POSIX version
- fd_set - file descriptor set
- timeval - time value structure

Figure A-26. API reference topic for Task 4 – Variation 3

Task 4, Variation 3	**VDEC**	Low (5)
	ICEC	High (21)



select (Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
```

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

The total number of bits set in readfds, writefds and errorfds, zero if the timeout expired, and on error, returns -1 and sets errno.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

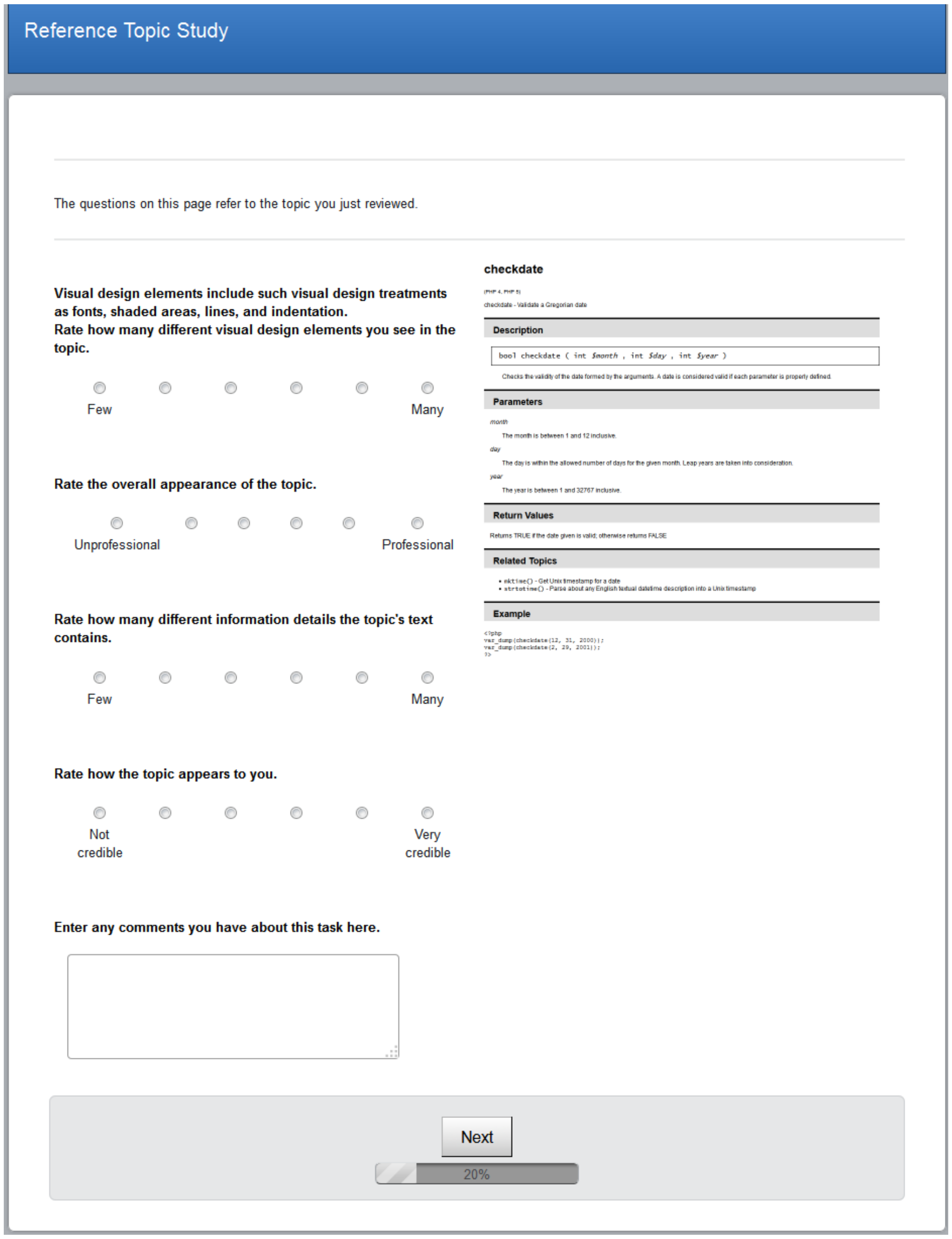
A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure A-27. API reference topic for Task 4 – Variation 4

Task 4, Variation 4	**VDEC**	Low (5)
	ICEC	Low (11)

Post-Task Questionnaire

Figure A-28 shows an example of the perception questionnaire that was presented to participants after they reviewed the API reference topic for the scenario.



Reference Topic Study

The questions on this page refer to the topic you just reviewed.

Visual design elements include such visual design treatments as fonts, shaded areas, lines, and indentation.
Rate how many different visual design elements you see in the topic.

Few ○ ○ ○ ○ ○ ○ Many

Rate the overall appearance of the topic.

Unprofessional ○ ○ ○ ○ ○ ○ Professional

Rate how many different information details the topic's text contains.

Few ○ ○ ○ ○ ○ ○ Many

Rate how the topic appears to you.

Not credible ○ ○ ○ ○ ○ ○ Very credible

Enter any comments you have about this task here.

checkdate

(PHP 4, PHP 5)

checkdate - Validate a Gregorian date

Description

`bool checkdate ( int $month , int $day , int $year )`

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

month

The month is between 1 and 12 inclusive.

day

The day is within the allowed number of days for the given month. Leap years are taken into consideration.

year

The year is between 1 and 32767 inclusive.

Return Values

Returns TRUE if the date given is valid; otherwise returns FALSE

Related Topics

- `mktime()` - Get Unix timestamp for a date
- `strtotime()` - Parse about any English textual datetime description into a Unix timestamp

Example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Figure A-28. Example of a post-task questionnaire

Survey Tool Modifications

Introduction Help Text Overlay

```
   $(document).ready(function(e) {
        $('.sg-content').prepend($('.demoOverlay'))
      $('.demoOverlay').show(500);
      $('#helpIcon').click(function(){
            $('.demoOverlay').show(300);
      });
      $('#closeIcon').click(function(){
            $('.demoOverlay').hide(300);
      });
   });
```

Figure A-29. Introduction help text overlay code

Browser Window Size Test

```
   /*
   * sgScreenSize.js
   *
   *  Used to test the browser window size to make sure
   *        it's big enough to run the test.
   *
   */
   // initialize default value
   var sgMinWidth = 900;
   var sgMinHeight = 600;
   var continueMsgInitText = '<p>Click <b>Next</b> to confirm that you '+
      'are 18 years of age or older and agree to participate in '+
      'this study.</p>';

   function checkSize() {
      // test the size and format the corresponding message
      var tooNarrow = (($(window).width() < sgMinWidth) ? 'too narrow' :'');
      var tooShort = (($(window).height() < sgMinHeight) ? 'too short' : '')
      // if not big enough, hide the next button and
      //   display the error message
```

Continued on next page

```
      if ((tooNarrow.length >0) || (tooShort.length > 0)) {
            $('.sg-footer').hide();
            var headerMsg = '<p>Your browser window is ';
            headerMsg = headerMsg + ((tooNarrow.length > 0) ?
                  tooNarrow : '');
            headerMsg = headerMsg + ((tooShort.length > 0) ?
                  ((tooNarrow.length > 0) ? ' and ' : ' ') + tooShort : '');
            headerMsg = headerMsg + ' for this study. Make the window '+
                  'larger or make the text smaller to continue; '+
                  'otherwise close the browser and try again on a '+
                  'computer with a larger screen.</p>';
            $('#continueMsg').html(headerMsg);
            $('#continueMsg').attr('class','sizeWarning');
      } else {
            // the browser is big enough so continue
            $('.sg-footer').show();
            $('#continueMsg').attr('class','');
            $('#continueMsg').html(continueMsgInitText);
      }
   }

   // check the window size and add the detection logic to
   //  check the size whenever the size changes.
   $(document).ready(function(e) {
      continueMsgInitText = $('#continueMsg').html();
      checkSize();
       $(window).resize(function(e) {
            checkSize();
      });
   });
```

Figure A-30. Browser window size test code

Decision Timing and Influential Topic Location

```
   /*
   *  sgLogAndSave.js
   *
   *  saves the location on the page image where
   *        the participant clicked
   */
   // initialize the variables used
   var clickX = -1;
   var clickY = -1;
   var clickErr = 0;
   var dataSent = false;
   var testMode = false;
```

Continued on next page

```
/*
*  record the last location the user clicked and left a spot
*        and then disable the ability to leave a spot
*        and display the next button so the participant can continue
*/
function logButton (btnName,jumpLink) {
   // save the data only if a spot has been placed or they clicked N/A
   if (((clickX != -1) && (clickY != -1)) || (btnName =='NA')) {
         // if the participant has clicked on the page, send a message to
         //   the server to record it

         // build the request buffer
         var sessionId = $('#sg_sessionid').val();
         var pageId = $('#sg_currentpageid').val();
         var logUrl =
   'https://docsbydesigncom.secure.myhosting.net/phd/data/writeClick.php'+
               '?sessionId='+sessionId+
               '&pageId='+pageId+
               '&responseButton='+btnName+
               '&responseTime='+clickErr+
               '&imageX='+clickX+
               '&imageY='+clickY;

         // send it to the server
         // the JQuery call seems to like the GET method better than POST,
         //     but the server-side doesn't care
         $.ajax({
               url: logUrl,
               type: 'GET',
               dataType: 'html'
         });
         // show the footer and jump to it so the participant can continue
         dataSent = true;
         $('.sg-footer').show();
         location.hash = '#'+jumpLink;
         // update the text in the prompt div and hide the save button
         //  so they can only save one time
         $('#spotPrompt').hide();
         $('#exitPrompt').show();
         $('#promptDiv').css('backgroundColor','green');
   } else {
         // display a message to the user when they try to save without
         //   clicking on the image
         clickErr = 1;
         alert ('Click on the image first, please!');
   }
}
```

Continued on next page

```
function imageClickSave(data){
   // define the "click-on-the-image" action
   if (!dataSent) {
         // only perform the action if the data hasn't been sent
         //  to the server, after that, do nothing with the click
         // compute the location of the click relative to the image
         var imageOffset = $('#dbd_topic').offset();
         var clickSpotX = data.clientX + $(document).scrollLeft();
         var clickSpotY = data.clientY + $(document).scrollTop();
         clickX = clickSpotX - imageOffset.left;
         clickY = clickSpotY - imageOffset.top;
         //  for debugging only
         // alert ("You clicked on coordinates: "+
         //  (clickX – imageOffset.left).toString()+
         //  ","+(clickY - imageOffset.top).toString());
         // define the location to place the spot
         var spotLeft = (testMode ? clickX : clickSpotX) –
          ($('#clickSpot').width()/2);
         var spotTop = (testMode ? (clickY  +
          ($('#clickSpot').height()/2)) :
          (clickSpotY  - ($('#clickSpot').height()/2)));
         // set the location and show it
         $("#clickSpot").css({
               left: spotLeft,
               top: spotTop
         });
         $("#clickSpot").show();
   }
}
```

Continued on next page

```
/*
*  Records response time in milliseconds
*/
var startTime = 0;

/*
*  Sends response time to server and updates the display
*/

function logTimeButton (btnName) {
  var endTime = new Date().getTime();
  var timeOnPage = endTime - startTime;
  // get sessionId from survey page
  var sessionId = $('#sg_sessionid').val();

  // get pageId from survey page --
  //  this will be resolved to the task in post-processing
  var pageId = $('#sg_currentpageid').val();
  var logUrl =
'https://docsbydesigncom.secure.myhosting.net/phd/data/writeData.php'+
      '?sessionId='+sessionId+
      '&pageId='+pageId+
      '&responseButton='+btnName+
      '&responseTime='+timeOnPage;

  $.ajax({
    url: logUrl,
    type: 'GET',
    dataType: 'html'
    });

  // hide help box if one is defined in the page
  if ($('.demoOverlay') !== undefined) {
     $('.demoOverlay').hide(300);
  }
  // switch prompts and enable image click
  $('#timedPrompt').hide();
  $('#spotPrompt').show();
  // reset page to catch spot
  $('#dbd_topic').click (imageClickSave
}
```

Continued on next page

```
$(document).ready(function() {
    var submitBtnLink = 'sbl';
   // format the prompt div at the top of the display
    $('#promptDiv').css({left: "auto"});
    $('#promptDiv').css({right: "auto"});
    $('#promptDiv').css({margin: "auto auto auto -46px"});
   $('#promptDiv').width($('.sg-wrapper').width());
   // record the start time
    startTime = new Date().getTime();
   // add an anchor to jump to after the time has been recorded and then
   //   hide the footer and header so it doesn't distract the participant
   //   until it's time to use them
   $('.sg-footer-hook-1').prepend('<a id="' + submitBtnLink +
         '" name="' + submitBtnLink + '"></a>');
    $('.sg-footer').hide();
    $('.sg-header').hide();
   // define the click actions for the buttons in the prompt div.
    $('#btnYes').click(function(e){
      logTimeButton ('yes');
    });
    $('#btnNo').click(function(e){
      logTimeButton ('no');
   });

   // enable the save button
    $('#btnSave').click(function(e){
      logButton ("save",submitBtnLink);
    });

    $('#btnNA').click(function(e){
      logButton ("NA",submitBtnLink);
    });

   // detect when the page is being run in test mode so that
   //  the offsets can be adjusted to match.
   if (window.parent !== undefined) {
         if (window.parent.document.getElementsByClassName(
               'modal-title').length > 0) {
               testMode = true;
         }
   }
})
```

Figure A-31. Decision timing and influential topic location code

Gratuity Registration

```
   /*
   *  sgGratuity.js
   */

   var gratuityUrl =
'https://docsbydesigncom.secure.myhosting.net/phd/data/gratuity.php';
   /*
   *  This function formats the data field to send in the command
   */

   function formObject (sessionId, email, comment) {
      this.studyId = 'PhD_Doc_Study';
      this.sessionId = sessionId;
      this.email = email;
      this.comment = comment;
      return this;
   }

   /*
   *  This formats the data object as a command to the service
   */
   function postGratuity (sessionId, email, comment) {
      this.gratuity = new formObject (sessionId, email, comment);
      return this;
   }

   $(document).ready(function() {
      // hide exit buttons
      $('#sgE-1777078-13-70-box').hide();
      $('#sg_SubmitButton').hide();

      // load skip and continue code
      $('#formCancel').click(function() {
            $('#sgE-1777078-13-70-box').show();
            $('#sg_SubmitButton').show();
      });

      // load registration code
      $('#formSubmit').click(function() {
            // clear out the response area in the UI
            $('#responseBody').html('<p>Sending...</p>');

            // format the post command with the fields from the form
            var buffer = new postGratuity(
                        $('#sg_sessionid').val(),
                        $('#formEmail').val(),
                        $('#formComments').val()
                  );
```

Continued on next page

```
            // send the request
            var postResponse = $.post(gratuityUrl, buffer);

            $('#sgE-1777078-13-70-box').show();
            $('#sg_SubmitButton').show();

            // if the response comes back let the user know.
            postResponse.done(function(response) {
                  var msgHtml = '';
                  $('#gratuityButtons').hide();
                  if ((response.data !== undefined) &&
                         (response.data != null)) {
                        if (response.data.gratuityGiven !== undefined) {
                              msgHtml = '<p>Thank you for participating in
the study.</p><p>You won a $10 Amazon gift certificate code! <b>Save this
code before you click <b>Submit</b>!</b></p><p style="font-size:200%; font-
weight:bold; text-align:center;">' + response.data.gratuityGiven + '</p>';

                        } else {
                              msgHtml = '<p>Thank you for participating in
the study.</p><p>Unfortunately, you didn\'t win a gift certificate, but your
participation will help improve API documentation.</p>';
                        }
                  }
                  if (response.error !== undefined) {
                        // an error was returned so just thank them
                        msgHtml = '<p>Thank you for participating in the
study. Your responses will help improve API documentation.</p>';
                  }
                  $('#responseBody').html(msgHtml);
            });
      });
   });
```

Figure A-32. Gratuity registration code

Study Style Sheets

This section contains the summarized list of the styles used in the different levels of VDEC and the .css file that defined them for this study.

High Visual Design Element Count (hd.css)

Table A-1. Description of text styles used in hd.css

Style	Font family and weight	Font size	Font color	Align	Indent	Bullet list
body (page default)	Arial normal	12 px	#000	left	0	no
pageDescription	Arial normal	12 px	#000	left	0	no
pageTitle	Arial bold	24 px	#000	left	0	no
pageVersion	Arial normal	9.6 px	#000	left	0	no
sectionHeading	Arial bold	18 px	#000	left	10 px	no
syntax	Lucida Console	14.4 px	#000	left	10 px	no
descriptionText	Arial normal	12 px	#000	left	3.0 em	no
paramName	Arial italic	12 px	#000	left	1.0 em	no
paramDescription	Arial normal	12 px	#000	left	3.0 em	no
paramWarning	Arial bold	14.4 px	#000	left	10.0 em	no
bodyText	Arial normal	12 px	#000	left	1.0 em	no
related topic	Arial normal	12 px	#000	left	0	yes
functionName	Lucida Console	12 px	#000	left	0	no
code (<pre>)	Monospace	12 px	#000	left	0	no

Table A-2. Description of graphical elements used in hd.css

Style	**Background color**	**Border (not box)**	**Box**	**Duplicate of style**
body (page default)	#FFF	no	no	
pageDescription	#FFF	no	no	body (page default)
pageTitle	#FFF	no	no	
pageVersion	#FFF	no	no	
sectionHeading	#DDD	top	no	
syntax	#FFF	no	yes	
descriptionText	#FFF	no	no	
paramName	#FFF	no	no	
paramDescription	#FFF	no	no	descriptionText
paramWarning	#DDD	no	yes	
bodyText	#FFF	no	no	
related topic	#FFF	no	no	
functionName	#FFF	no	no	
code (<pre>)	#FFF	no	no	

Unique visual design element count: 12

```
@charset "utf-8";
/* CSS Document */
body {
   font-family:Arial, Helvetica, sans-serif;
   font-size:12px;
}

p#pageTitle { font-weight:bold; font-size: 200%; }

p#pageVersion{ font-size: 80% }
p#pageDescription{ }

p.sectionHeading{
   font-size: 150%;
   font-weight:bold;
   background-color:#ddd;
   border-top: 3px #000 solid;
   padding: 10px 0 4px 1em;
}
p#syntax{
   font-size: 120%;
   font-family:"Lucida Console", Monaco, monospace;
   border-style: solid;
   border-width: thin;
   margin-left: 1em;
   padding: 10px 5px 10px 1.0em;
}
p.descriptionText{ margin-left: 3.0em;  }
p.paramName{ margin-left: 1em; font-style:italic; }
p.paramDescription{ margin-left: 3.0em; }
span.paramRef{ font-style:italic; }
p.paramWarning{
   margin-left: 10em;
   margin-right: 10em;
   font-size: 120%;
   font-weight:bold;
   padding: 10px 10px 10px 10px;
   background-color:#ddd;
   border-style: solid;
   border-width: thin;
   border-color: black;
}
p.bodyText{ margin-left: 1em;  }
li.relatedTopic{ }
span.fnName{ font-family: "Lucida Console", Monaco, monospace; }
pre {font-family:monospace; font-size:12px;}
```

Figure A-33. The .css file used by High Visual-Design Element Count topics (hd.css)

Low Visual Design Element Count (ld.css)

Table A-3. Description of text styles used in ld.css

Style	Font family and weight	Font size	Font color	Align	Indent	Bullet list
body (page default)	Arial normal	12 px	#000	left	0	no
pageDescription	Arial normal	12 px	#000	left	0	no
pageTitle	Arial normal	12 px	#000	left	0	no
pageVersion	Arial normal	12 px	#000	right	0	no
sectionHeading	Arial normal	12 px	#000	left	0	no
syntax	Arial normal	12 px	#000	left	1.0 em	no
descriptionText	Arial normal	12 px	#000	left	3.0 em	no
paramName	Arial normal	12 px	#000	left	1.0 em	no
paramDescription	Arial normal	12 px	#000	left	0	no
paramWarning	Arial normal	12 px	#000	left	10.0 em	no
bodyText	Arial normal	12 px	#000	left	1.0 em	no
related topic	Arial normal	12 px	#000	left	0	no
functionName	Arial normal	12 px	#000	left	0	no
code (<pre>)	Arial normal	12 px	#000	left	0	no

Table A-4. Description of graphical elements used in ld.css

Style	Background color	Border (not box)	Box	Duplicate of style
body (page default)	#FFF	no	no	
pageDescription	#FFF	no	no	body (page default)
pageTitle	#FFF	no	no	body (page default)
pageVersion	#FFF	no	no	
sectionHeading	#FFF	no	no	body (page default)
syntax	#FFF	no	no	
descriptionText	#FFF	no	no	
paramName	#FFF	no	no	syntax
paramDescription	#FFF	no	no	body (page default)
paramWarning	#FFF	no	no	
bodyText	#FFF	no	no	syntax
related topic	#FFF	no	no	body (page default)
functionName	#FFF	no	no	body (page default)
code (<pre>)	#FFF	no	no	body (page default)

Unique visual design element count: 5

```
@charset "utf-8";
/* CSS Document */
body {
   font-family:Arial, Helvetica, sans-serif;
   font-size:12px;
}

p#pageTitle {  }

p#pageVersion{ margin-top:-4.5ex; text-align:right }
p#pageDescription{ }

p.sectionHeading{ }
p#syntax{ margin-left: 1em; }
p.descriptionText{ margin-left: 3.0em;  }
p.paramName{ margin-left: 1em; }
p.paramDescription{ margin-left: 3.0em; }
p.paramWarning{ margin-left: 10em;  }
p.bodyText{ margin-left: 1em;  }
li.relatedTopic{}
span.fnName{ }
pre {font-family:Arial, Helvetica, sans-serif; font-size:12px;}
```

Figure A-34.   The .css file used by Low Visual-Design Element Count topics (ld.css)

Survey Tool Global Style Sheet

```
/* sgGlobal.css */
.study_page .sg-question-set {}
.study_inst_noborder {border-bottom: none;}
.sg-cc-hook {clear: left;}
.survey-image {position: relative; top: -640px; border: #777 solid 1px}
.sizeWarning {font-size:150%; font-weight: bold; background-color:yellow;}
div.prompt {
   position:fixed;
   float:none;
         border-radius: 0 0 .3em .3em;
   font-family:Verdana, Geneva, sans-serif;
   font-size: 1.0em;
   top: 0px;
        left: 0px;
        background-color: rgba(38, 38, 38, 0.85);
   color:lightGray;
        margin: 0 auto;
}

.promptTable {
   width:100%;
}

.promptTableScenario,.promptTableButtonCell,.promptTablePrompt {
   font-family:Verdana, Geneva, sans-serif;
   font-size: 1.0em;
   color:lightGray;
   padding-left:15px;
   vertical-align:text-top;
}

.promptTablePrompt {
  padding-left:0;
  font-size: 120%;
  font-weight: bold;
}

.promptTableButtons {
   text-align:right;
}

.promptButton {
   font-size:150%;
    font-weight: bold;
    width: 5em;
   padding: 5px .2em 5px .2em;
}
```

Continued on next page

```
body {
   padding-top: 50px;
}

#clickSpot {
    position:absolute;
    width: 50px;
    height: 50px;
    background: rgb(45,114,194);
    opacity: 0.7;
    -moz-border-radius: 25px;
    -webkit-border-radius: 25px;
    border-radius: 25px;
}

.closePrompt {
   color:#FFF;
   background-color:rgba(45,114,194,1.0);
   position: absolute;
   top: 5px;
   right: 5px;
   width: 24px;
   height: 24px;
    border: 1px white solid;
   text-align:center;
   text-decoration: none;
   font-weight: bold;
   -webkit-border-radius: 12px;
   -moz-border-radius: 12px;
   border-radius: 12px;
   font-family:Tahoma, Geneva, sans-serif;
   font-size:18px;
   padding-top: -5px;
   vertical-align: m;
   z-index: 100;
}
```

Continued on next page

```
.helpPrompt {
   color:#FFF;
   background-color: rgb(230,231,232);
   position: absolute;
   top: 0px;
   right: 0px;
   width: 30px;
   height: 30px;
   text-align:center;
   text-decoration: none;
   font-weight: bold;
   -webkit-border-radius: 15px;
   -moz-border-radius: 15px;
   border-radius: 15px;
   font-family:Tahoma, Geneva, sans-serif;
   font-size:23px;
   margin-left: 3px;
   vertical-align:middle;
   z-index: 100;
}

.demoOverlay {
     display: none;
     position: absolute;
     left: 70%;
     top: 0%;
    width: auto;
     text-align:center;
     z-index: 1000;
    background-color:rgba(45,114,194,0.7);
   -webkit-border-radius: 3px;
   -moz-border-radius: 3px;
   border-radius: 3px;
}

.demoOverlay div {
     width:auto;
     margin: 10px 10px 10px 10px;
     background-color: #fff;
     border:1px solid rgb(45,114,194);
    text-align: center;
}

.demoOverlay h1 {
  text-align: center;
}
```

Continued on next page

```
div.demoText {
  border:0 solid #FFF;
  margin: 0 auto;
  padding:15px;
}

p.demoPara {
  text-align: left;
}
.emailInput { width: 40em;}

.linkNA {
   font-weight: bold;
   cursor: pointer;
}
.linkNA:hover {
   text-decoration:underline;
}
```

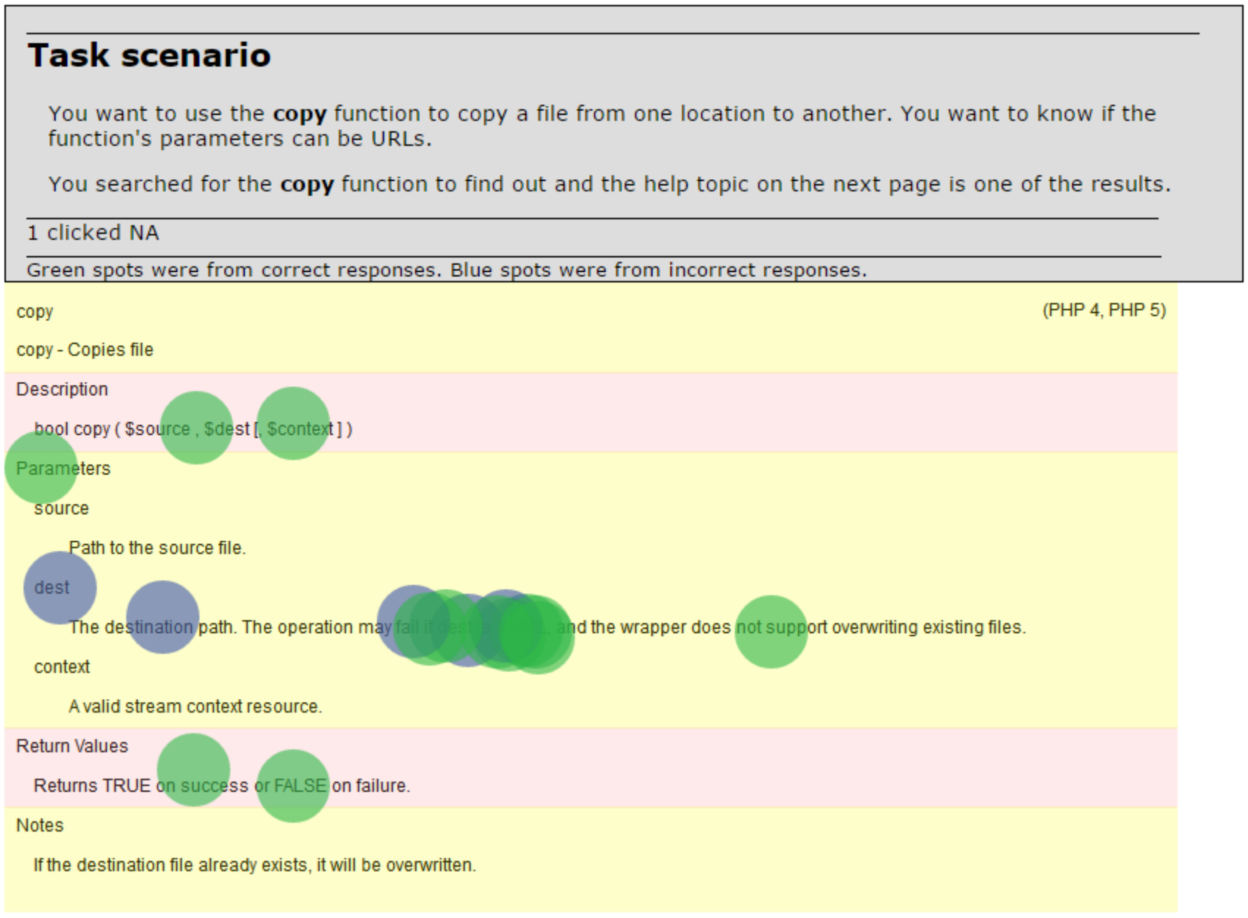
Figure A-35. Survey tool global style sheet

B. INFLUENTIAL SPOT LOCATION MAPS

The images in this section show the places in the topics where participants identified the location that was most influential in their relevance assessment. The shaded areas in the images identify the different sections of the topic as divided for analysis. These divisions were not displayed to participants during the study, but were applied after the study to facilitate analysis. The topic sections are not labeled in the illustrations, but they are named, from top to bottom:

1. Title
2. Description
3. Parameters
4. Return Values
5. Notes
6. Related Topics (Not present in all topics)
7. Examples (Not present in all topics)

The *Task Scenario* box at the top was added to these illustrations to facilitate analysis. This information did not appear to the participant as it is shown in these illustrations. Instead, it was displayed in the page that preceded the topic and the information was summarized at the top of the topic review screen. See the **Online Study Protocol** section for detailed descriptions and examples of the screens used by the study protocol.



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest is [illegible], and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

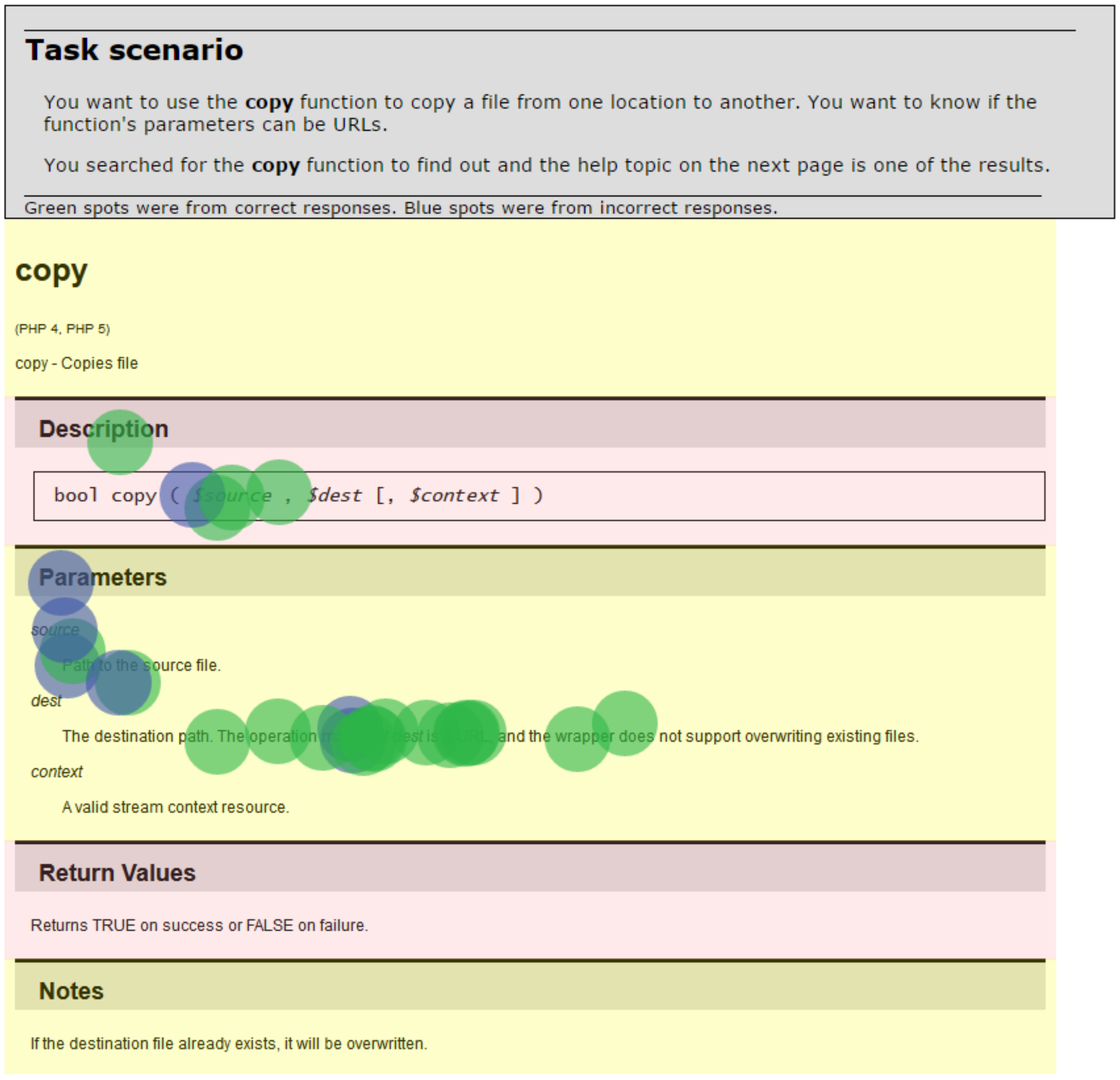
Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure B-1. Task 1 spot map for relevant question, low VDEC, low ICEC



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( $source , $dest [, $context ] )
```

Parameters

source

Path to the source file.

dest

The destination path. The operation [illegible] *dest* is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

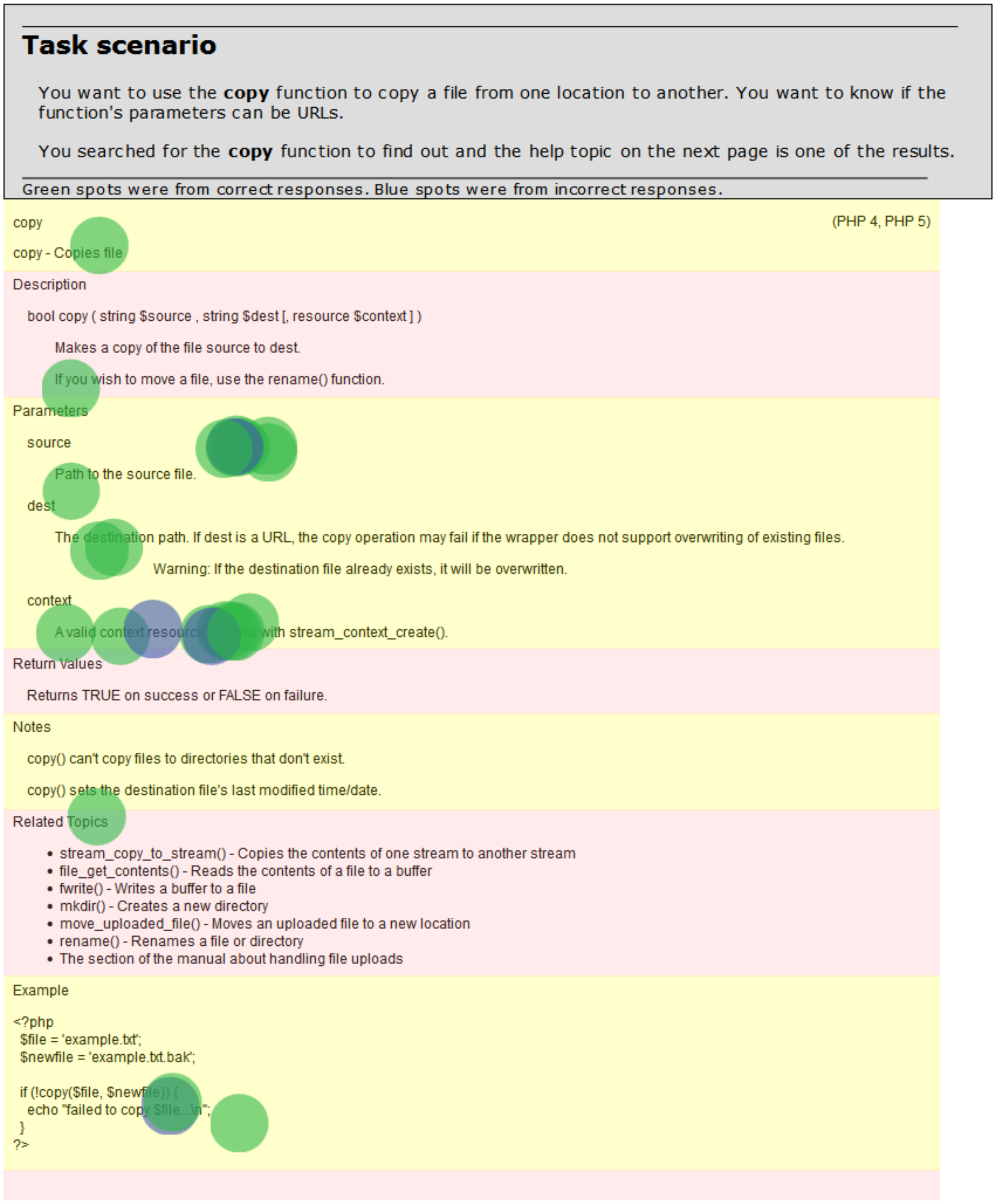
Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure B-2. Task 1 spot map for relevant question, high VDEC, low ICEC



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy (string $source , string $dest [, resource $context])

Makes a copy of the file source to dest.

If you wish to move a file, use the rename() function.

Parameters

source

Path to the source file.

dest

The destination path. If dest is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with stream_context_create().

Return Values

Returns TRUE on success or FALSE on failure.

Notes

copy() can't copy files to directories that don't exist.

copy() sets the destination file's last modified time/date.

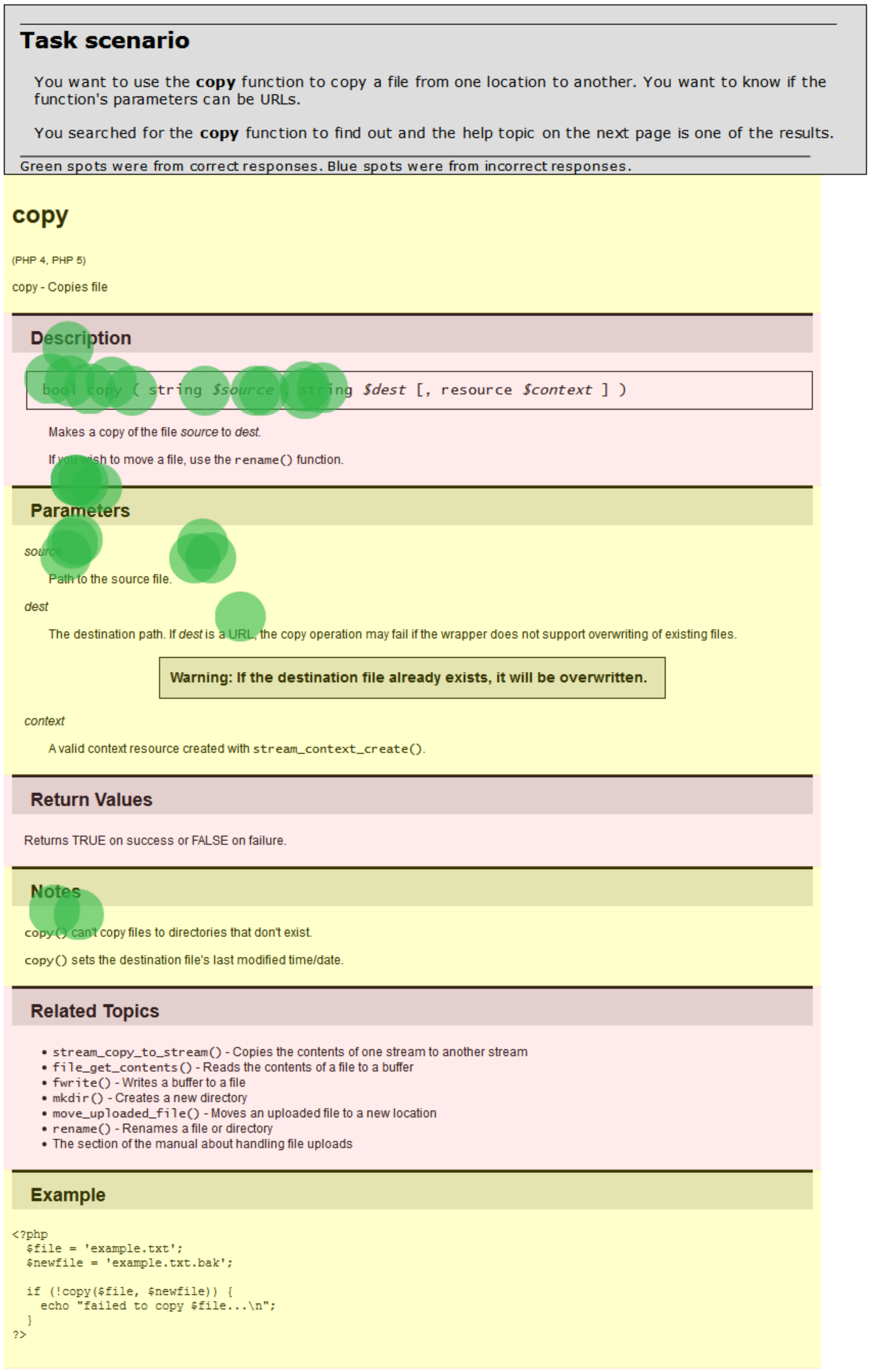
Related Topics

- stream_copy_to_stream() - Copies the contents of one stream to another stream
- file_get_contents() - Reads the contents of a file to a buffer
- fwrite() - Writes a buffer to a file
- mkdir() - Creates a new directory
- move_uploaded_file() - Moves an uploaded file to a new location
- rename() - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
 $file = 'example.txt';
 $newfile = 'example.txt.bak';

 if (!copy($file, $newfile)) {
   echo "failed to copy $file...\n";
 }
?>
```

Figure B-3. Task 1 spot map for relevant question, low VDEC, high ICEC



Task scenario

You want to use the **copy** function to copy a file from one location to another. You want to know if the function's parameters can be URLs.

You searched for the **copy** function to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( string $source , string $dest [, resource $context ] )
```

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the `rename()` function.

Parameters

source

Path to the source file.

dest

The destination path. If *dest* is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with `stream_context_create()`.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

`copy()` can't copy files to directories that don't exist.

`copy()` sets the destination file's last modified time/date.

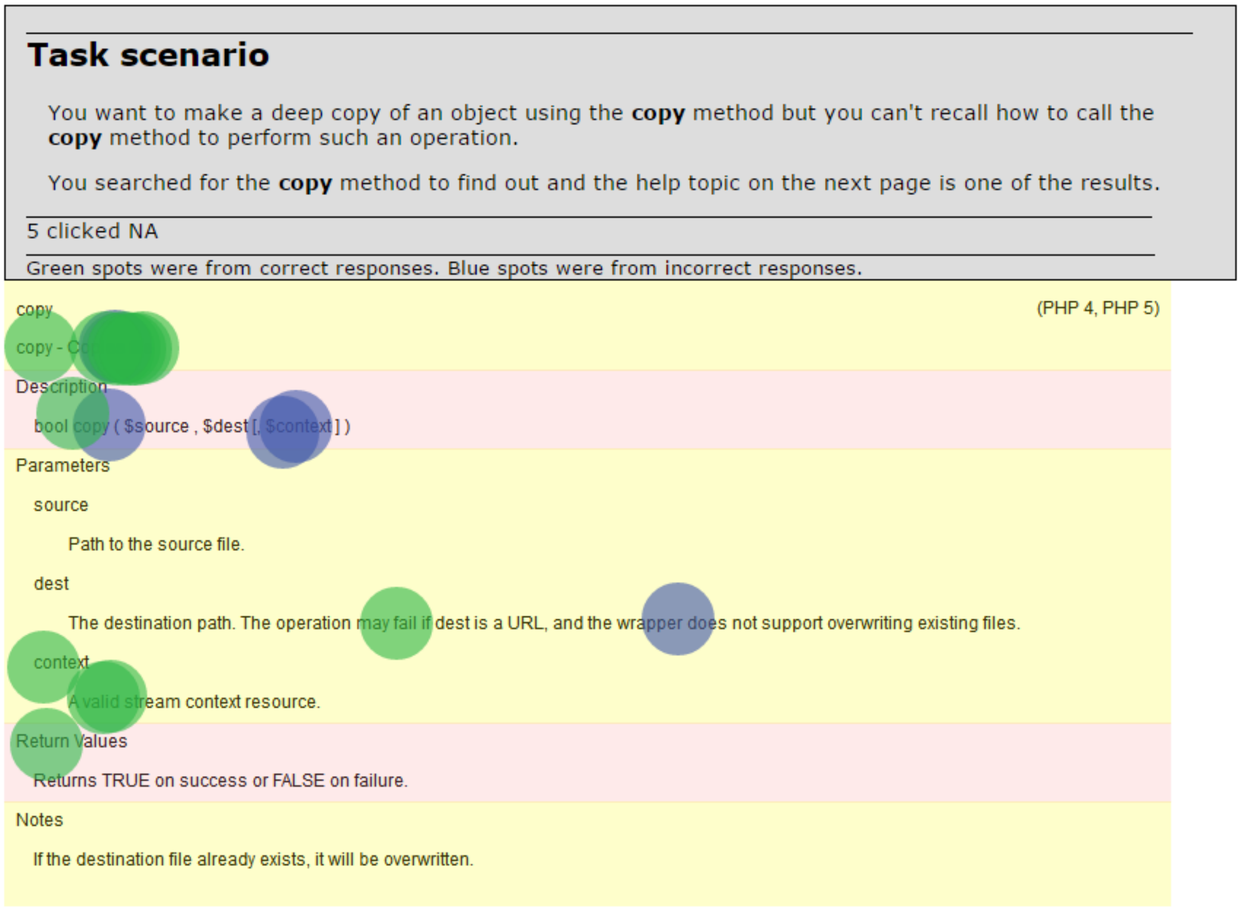
Related Topics

- `stream_copy_to_stream()` - Copies the contents of one stream to another stream
- `file_get_contents()` - Reads the contents of a file to a buffer
- `fwrite()` - Writes a buffer to a file
- `mkdir()` - Creates a new directory
- `move_uploaded_file()` - Moves an uploaded file to a new location
- `rename()` - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
  $file = 'example.txt';
  $newfile = 'example.txt.bak';

  if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
  }
?>
```

Figure B-4. Task 1 spot map for relevant question, high VDEC, high ICEC


Task scenario

You want to make a deep copy of an object using the **copy** method but you can't recall how to call the **copy** method to perform such an operation.

You searched for the **copy** method to find out and the help topic on the next page is one of the results.

5 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy (PHP 4, PHP 5)

copy - Co

Description

bool copy ($source , $dest [, $context])

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if dest is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

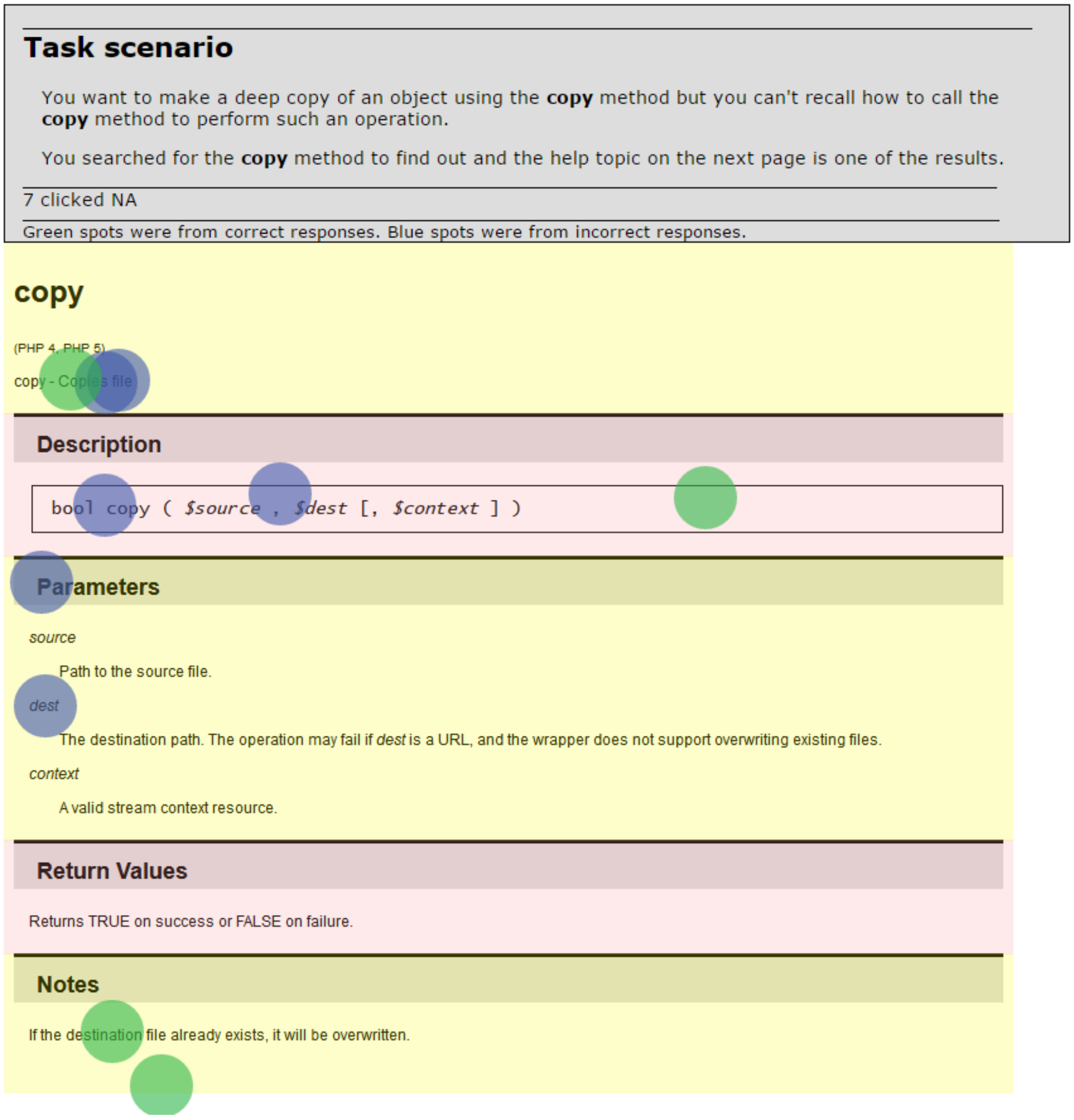
Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure B-5. Task 1 spot map for non-relevant question, low VDEC, low ICEC



Task scenario

You want to make a deep copy of an object using the **copy** method but you can't recall how to call the **copy** method to perform such an operation.

You searched for the **copy** method to find out and the help topic on the next page is one of the results.

7 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( $source , $dest [, $context ] )
```

Parameters

source

Path to the source file.

dest

The destination path. The operation may fail if *dest* is a URL, and the wrapper does not support overwriting existing files.

context

A valid stream context resource.

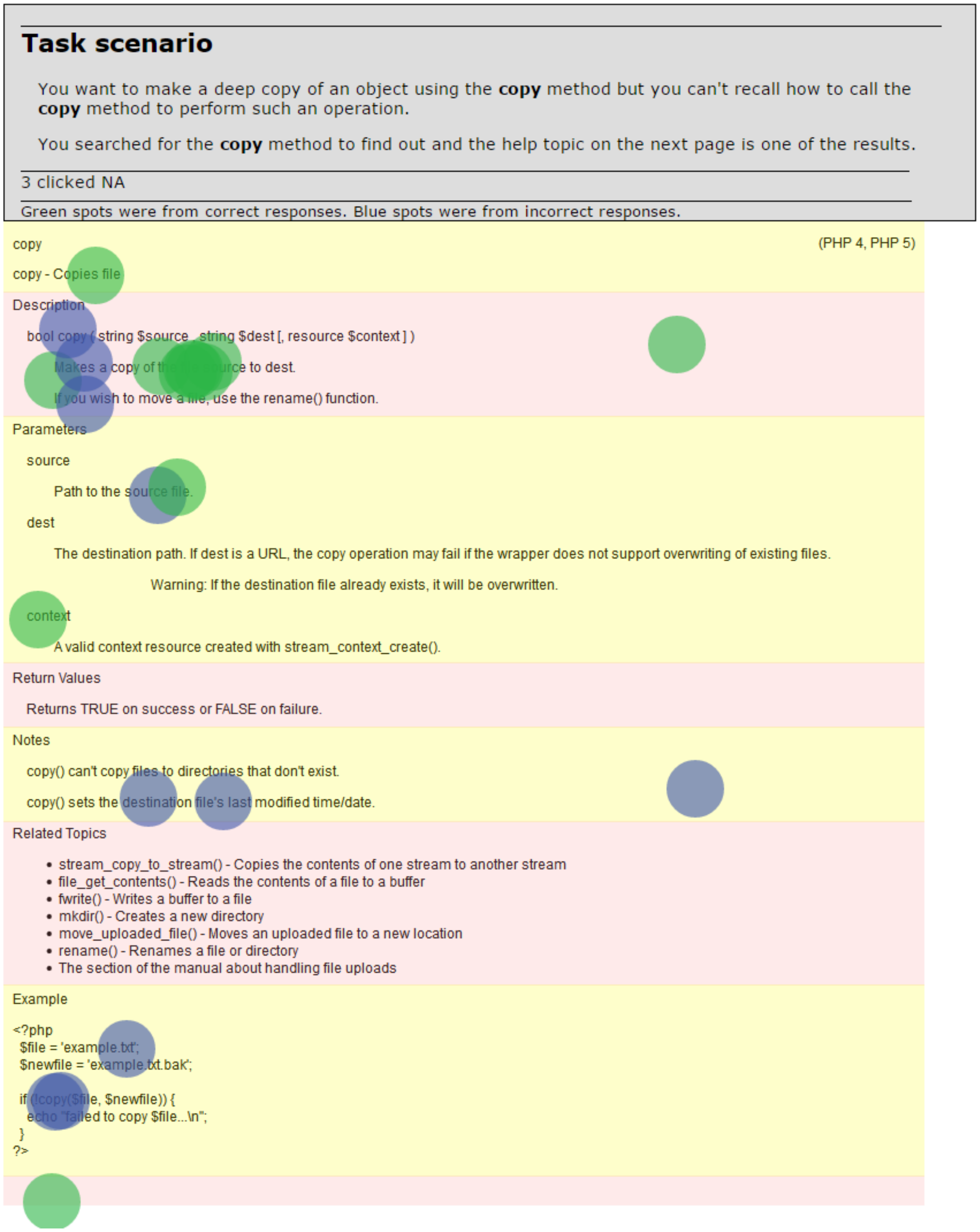
Return Values

Returns TRUE on success or FALSE on failure.

Notes

If the destination file already exists, it will be overwritten.

Figure B-6. Task 1 spot map for non-relevant question, high VDEC, low ICEC



Task scenario

You want to make a deep copy of an object using the **copy** method but you can't recall how to call the **copy** method to perform such an operation.

You searched for the **copy** method to find out and the help topic on the next page is one of the results.

3 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy (PHP 4, PHP 5)

copy - Copies file

Description

bool copy (string $source , string $dest [, resource $context])

Makes a copy of the file source to dest.

If you wish to move a file, use the rename() function.

Parameters

source

Path to the source file.

dest

The destination path. If dest is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with stream_context_create().

Return Values

Returns TRUE on success or FALSE on failure.

Notes

copy() can't copy files to directories that don't exist.

copy() sets the destination file's last modified time/date.

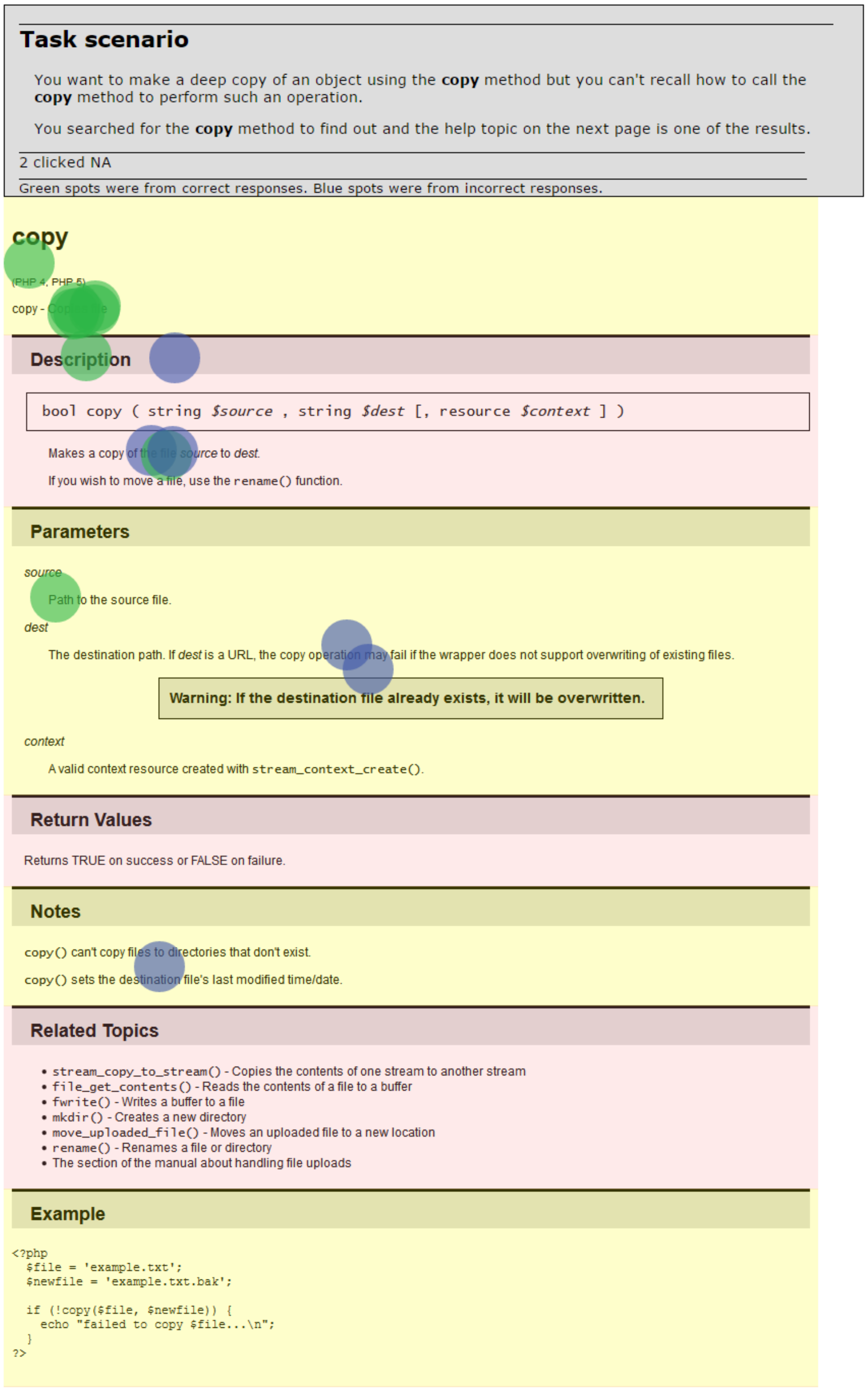
Related Topics

- stream_copy_to_stream() - Copies the contents of one stream to another stream
- file_get_contents() - Reads the contents of a file to a buffer
- fwrite() - Writes a buffer to a file
- mkdir() - Creates a new directory
- move_uploaded_file() - Moves an uploaded file to a new location
- rename() - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
 $file = 'example.txt';
 $newfile = 'example.txt.bak';

 if (!copy($file, $newfile)) {
  echo "failed to copy $file...\n";
 }
?>
```

Figure B-7. Task 1 spot map for non-relevant question, low VDEC, high ICEC



Task scenario

You want to make a deep copy of an object using the **copy** method but you can't recall how to call the **copy** method to perform such an operation.

You searched for the **copy** method to find out and the help topic on the next page is one of the results.

2 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

copy

(PHP 4, PHP 5)

copy - Copies file

Description

```
bool copy ( string $source , string $dest [, resource $context ] )
```

Makes a copy of the file *source* to *dest*.

If you wish to move a file, use the `rename()` function.

Parameters

source

Path to the source file.

dest

The destination path. If *dest* is a URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning: If the destination file already exists, it will be overwritten.

context

A valid context resource created with `stream_context_create()`.

Return Values

Returns TRUE on success or FALSE on failure.

Notes

`copy()` can't copy files to directories that don't exist.

`copy()` sets the destination file's last modified time/date.

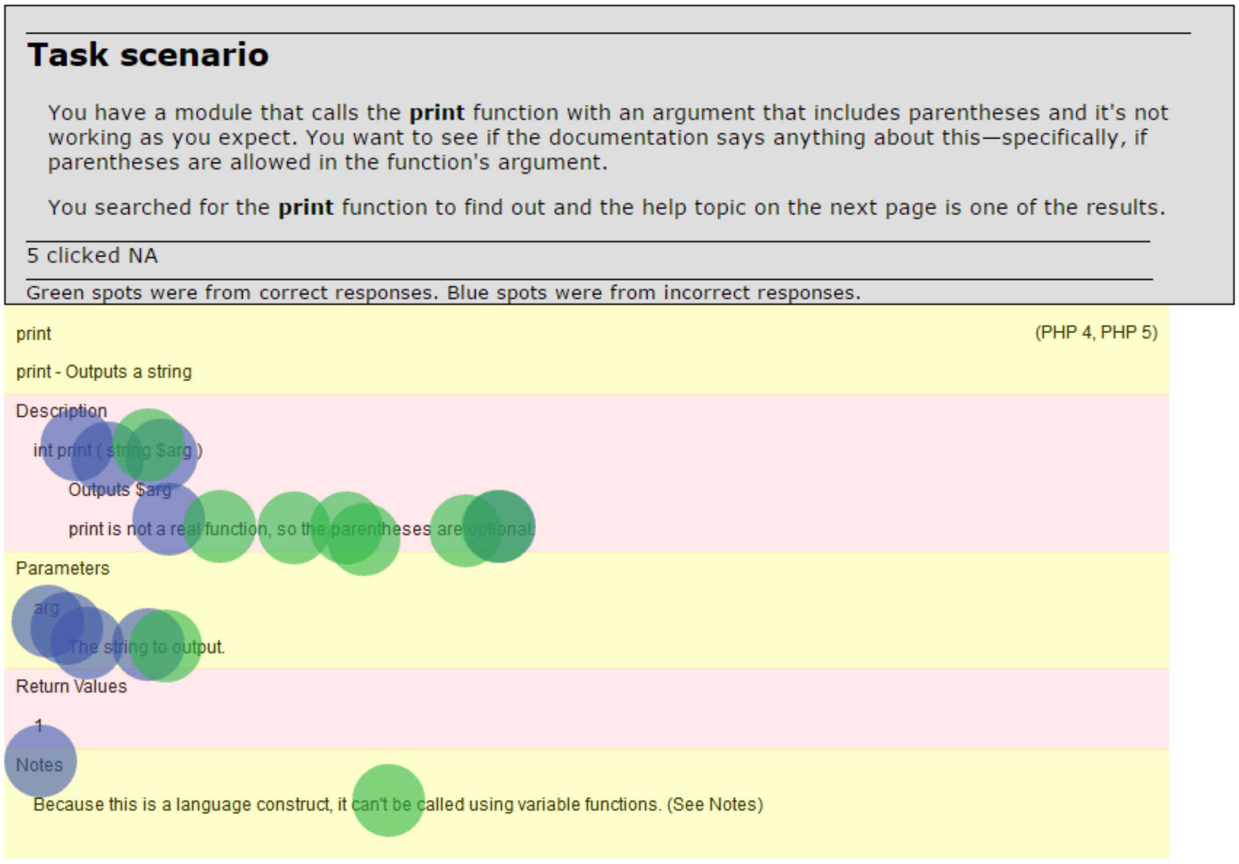
Related Topics

- `stream_copy_to_stream()` - Copies the contents of one stream to another stream
- `file_get_contents()` - Reads the contents of a file to a buffer
- `fwrite()` - Writes a buffer to a file
- `mkdir()` - Creates a new directory
- `move_uploaded_file()` - Moves an uploaded file to a new location
- `rename()` - Renames a file or directory
- The section of the manual about handling file uploads

Example

```
<?php
  $file = 'example.txt';
  $newfile = 'example.txt.bak';

  if (!copy($file, $newfile)) {
    echo "failed to copy $file...\n";
  }
?>
```

Figure B-8. Task 1 spot map for non-relevant question, high VDEC, high ICEC



Task scenario

You have a module that calls the **print** function with an argument that includes parentheses and it's not working as you expect. You want to see if the documentation says anything about this—specifically, if parentheses are allowed in the function's argument.

You searched for the **print** function to find out and the help topic on the next page is one of the results.

5 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print (PHP 4, PHP 5)

print - Outputs a string

Description

int print (string $arg)

Outputs $arg

print is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

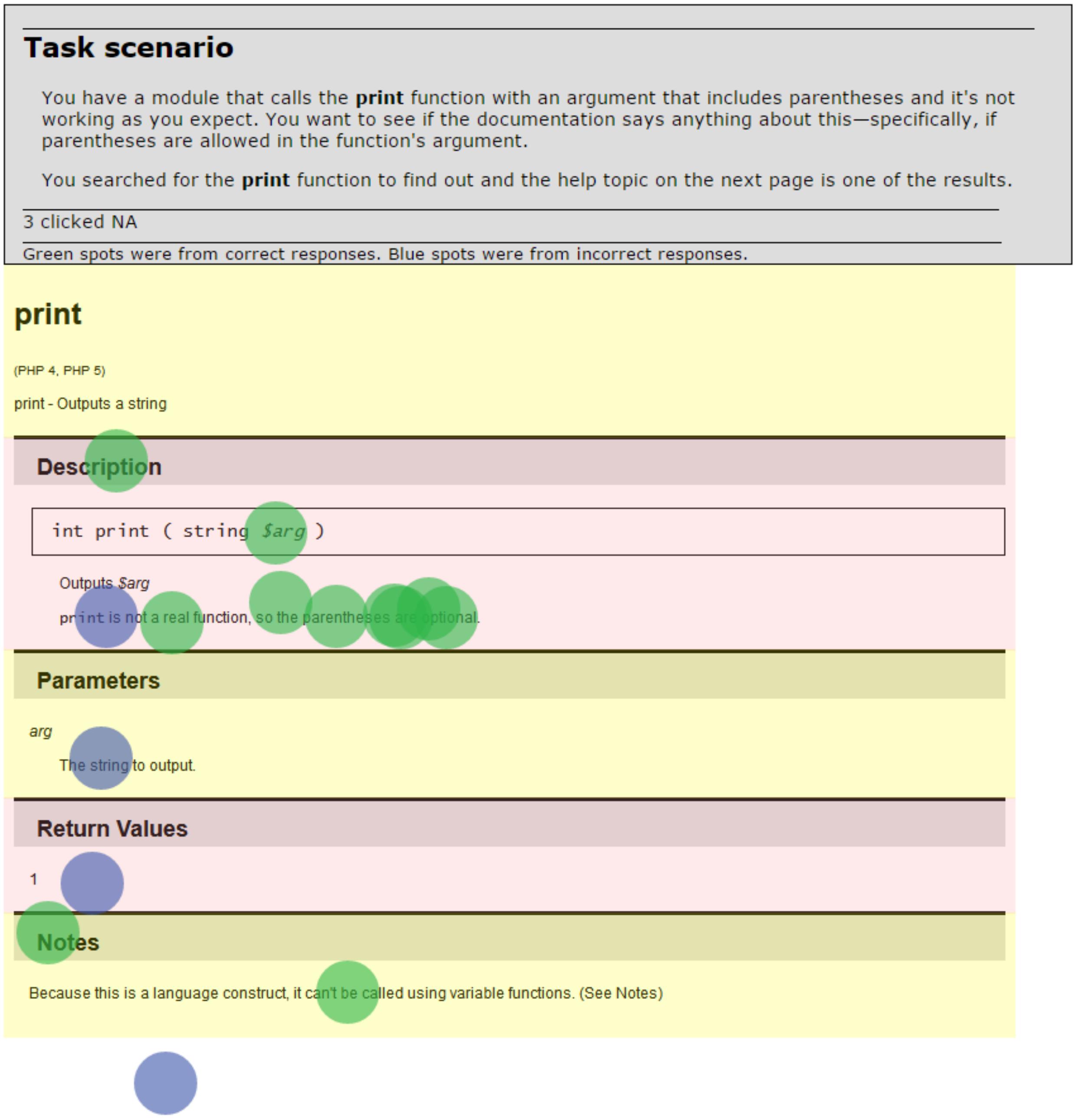
Return Values

1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure B-9. Task 2 spot map for relevant question, low VDEC, low ICEC



Task scenario

You have a module that calls the **print** function with an argument that includes parentheses and it's not working as you expect. You want to see if the documentation says anything about this—specifically, if parentheses are allowed in the function's argument.

You searched for the **print** function to find out and the help topic on the next page is one of the results.

3 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print

(PHP 4, PHP 5)

print - Outputs a string

Description

```
int print ( string $arg )
```

Outputs *$arg*

`print` is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

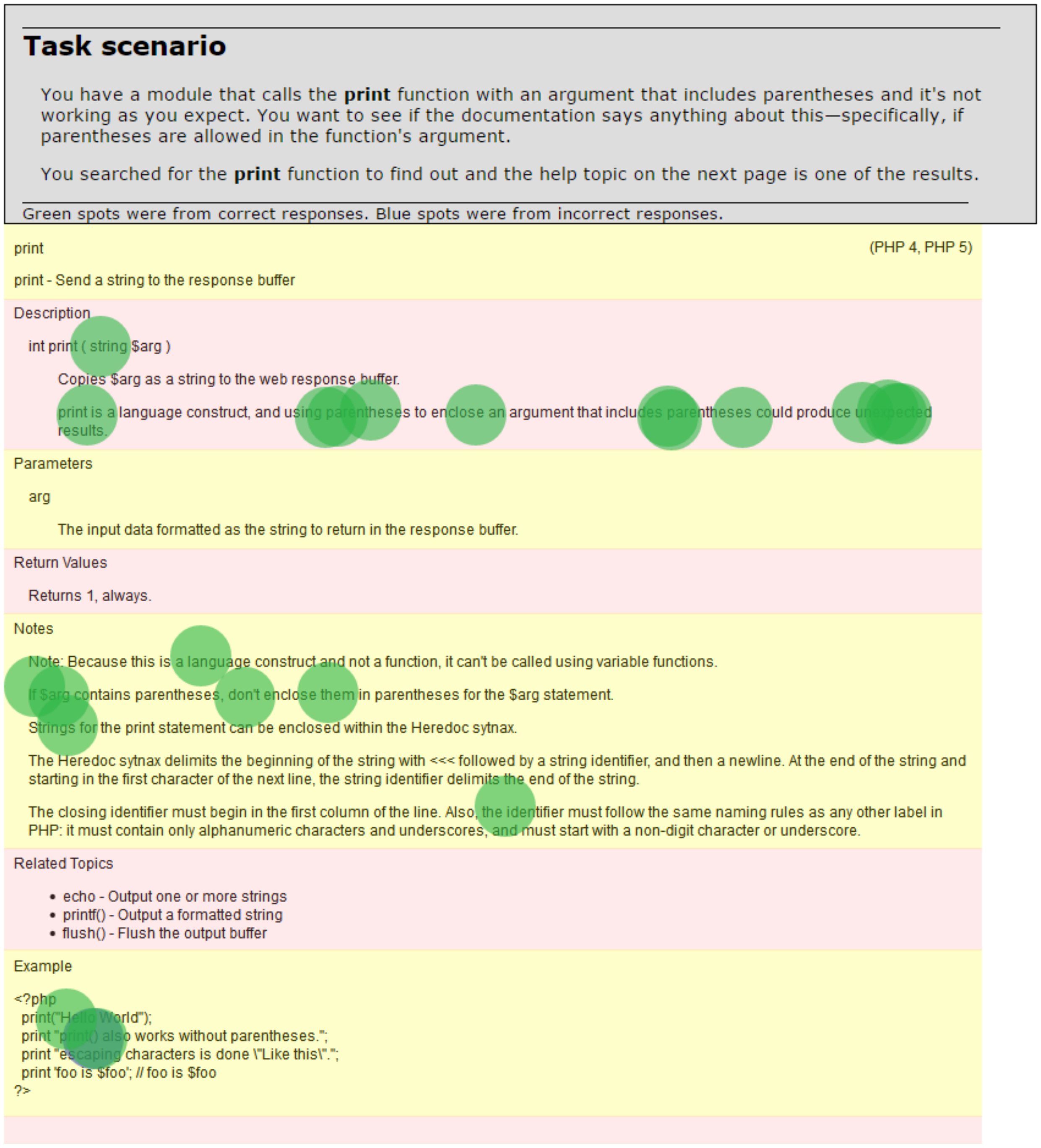
Return Values

1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure B-10. Task 2 spot map for relevant question, high VDEC, low ICEC



Task scenario

You have a module that calls the **print** function with an argument that includes parentheses and it's not working as you expect. You want to see if the documentation says anything about this—specifically, if parentheses are allowed in the function's argument.

You searched for the **print** function to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

print (PHP 4, PHP 5)

print - Send a string to the response buffer

Description

int print (string $arg)

Copies $arg as a string to the web response buffer.

print is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If $arg contains parentheses, don't enclose them in parentheses for the $arg statement.

Strings for the print statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

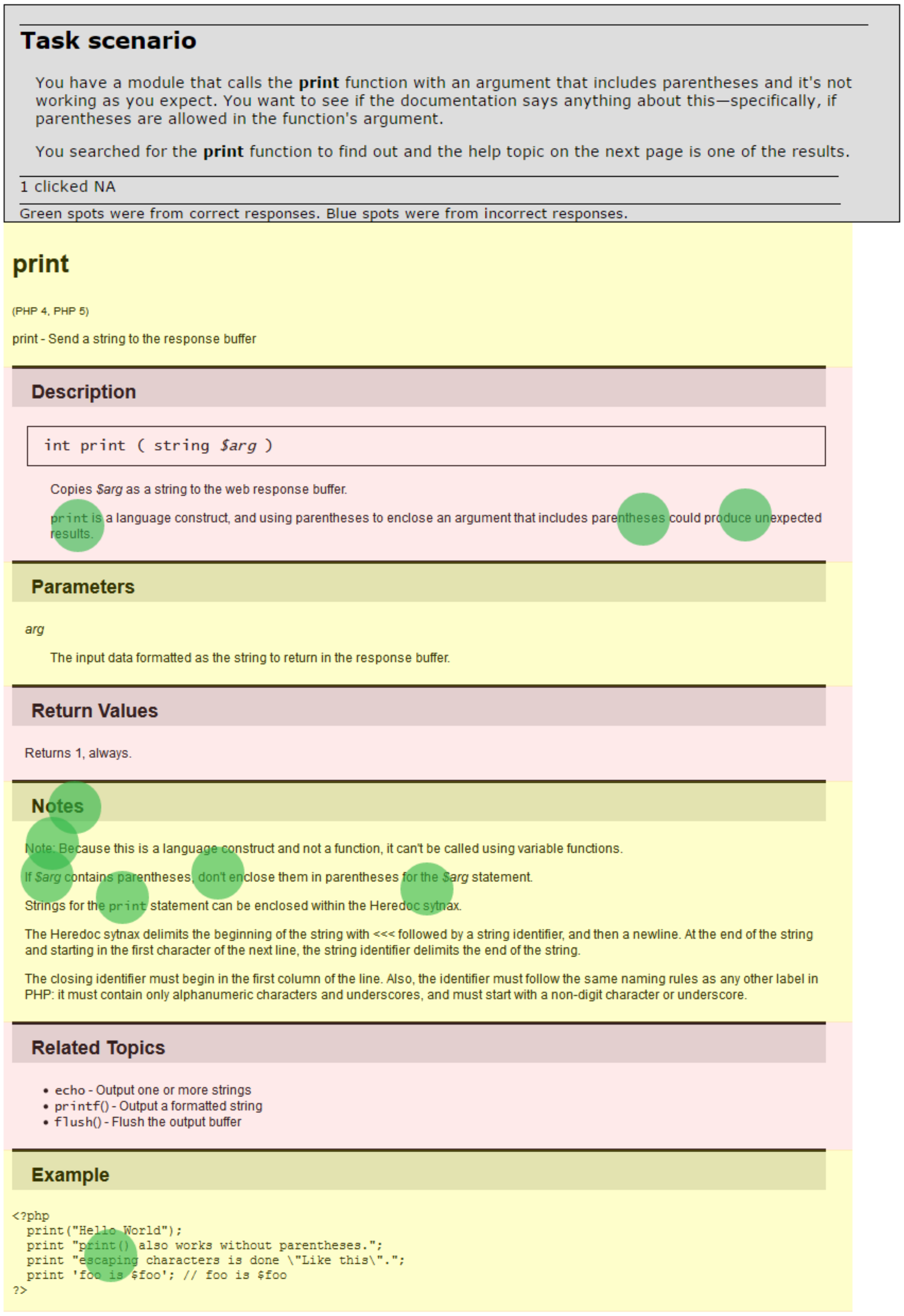
Related Topics

- echo - Output one or more strings
- printf() - Output a formatted string
- flush() - Flush the output buffer

Example

```
<?php
 print("Hello World");
 print "print() also works without parentheses.";
 print "escaping characters is done \"Like this\".";
 print 'foo is $foo'; // foo is $foo
?>
```

Figure B-11. Task 2 spot map for relevant question, low VDEC, high ICEC



Task scenario

You have a module that calls the **print** function with an argument that includes parentheses and it's not working as you expect. You want to see if the documentation says anything about this—specifically, if parentheses are allowed in the function's argument.

You searched for the **print** function to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print

(PHP 4, PHP 5)

print - Send a string to the response buffer

Description

```
int print ( string $arg )
```

Copies *$arg* as a string to the web response buffer.

`print` is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If *$arg* contains parentheses, don't enclose them in parentheses for the *$arg* statement.

Strings for the `print` statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

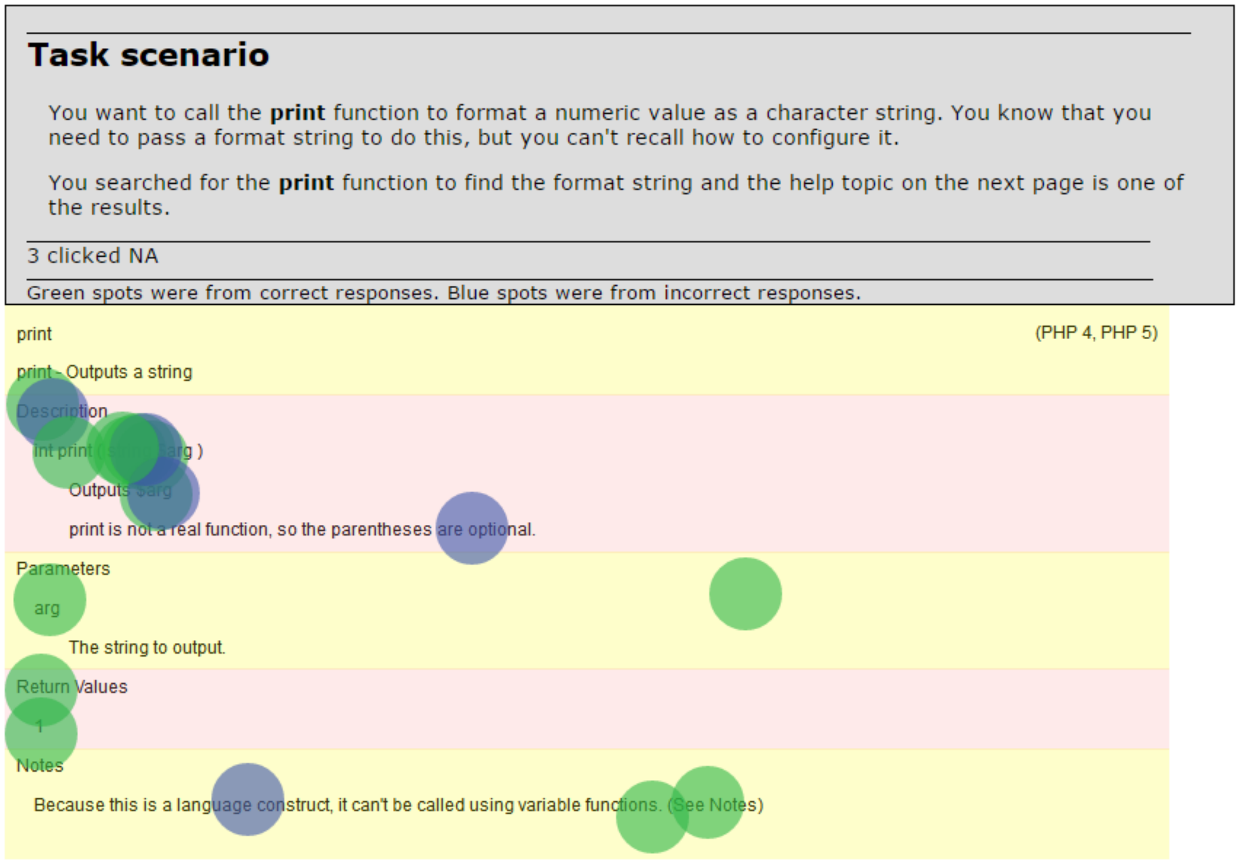
Related Topics

- `echo` - Output one or more strings
- `printf()` - Output a formatted string
- `flush()` - Flush the output buffer

Example

```
<?php
  print("Hello World");
  print "print() also works without parentheses.";
  print "escaping characters is done \"Like this\".";
  print 'foo is $foo'; // foo is $foo
?>
```

Figure B-12. Task 2 spot map for relevant question, high VDEC, high ICEC


Task scenario

You want to call the **print** function to format a numeric value as a character string. You know that you need to pass a format string to do this, but you can't recall how to configure it.

You searched for the **print** function to find the format string and the help topic on the next page is one of the results.

3 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print (PHP 4, PHP 5)

print - Outputs a string

Description

int print (string $arg)

Outputs $arg

print is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

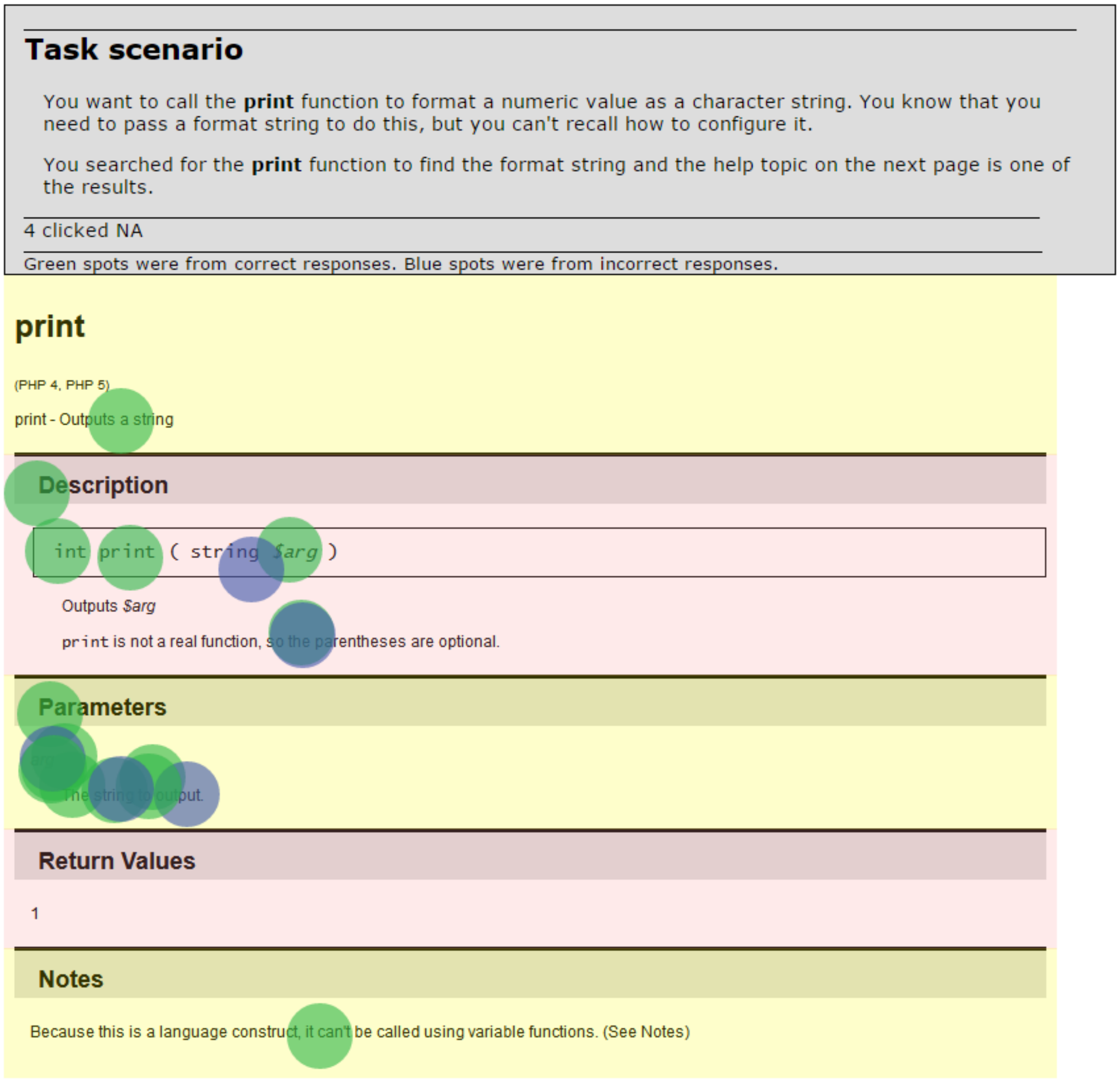
Return Values

1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure B-13. Task 2 spot map for non-relevant question, low VDEC, low ICEC



Task scenario

You want to call the **print** function to format a numeric value as a character string. You know that you need to pass a format string to do this, but you can't recall how to configure it.

You searched for the **print** function to find the format string and the help topic on the next page is one of the results.

4 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print

(PHP 4, PHP 5)

print - Outputs a string

Description

```
int print ( string $arg )
```

Outputs *$arg*

`print` is not a real function, so the parentheses are optional.

Parameters

arg

The string to output.

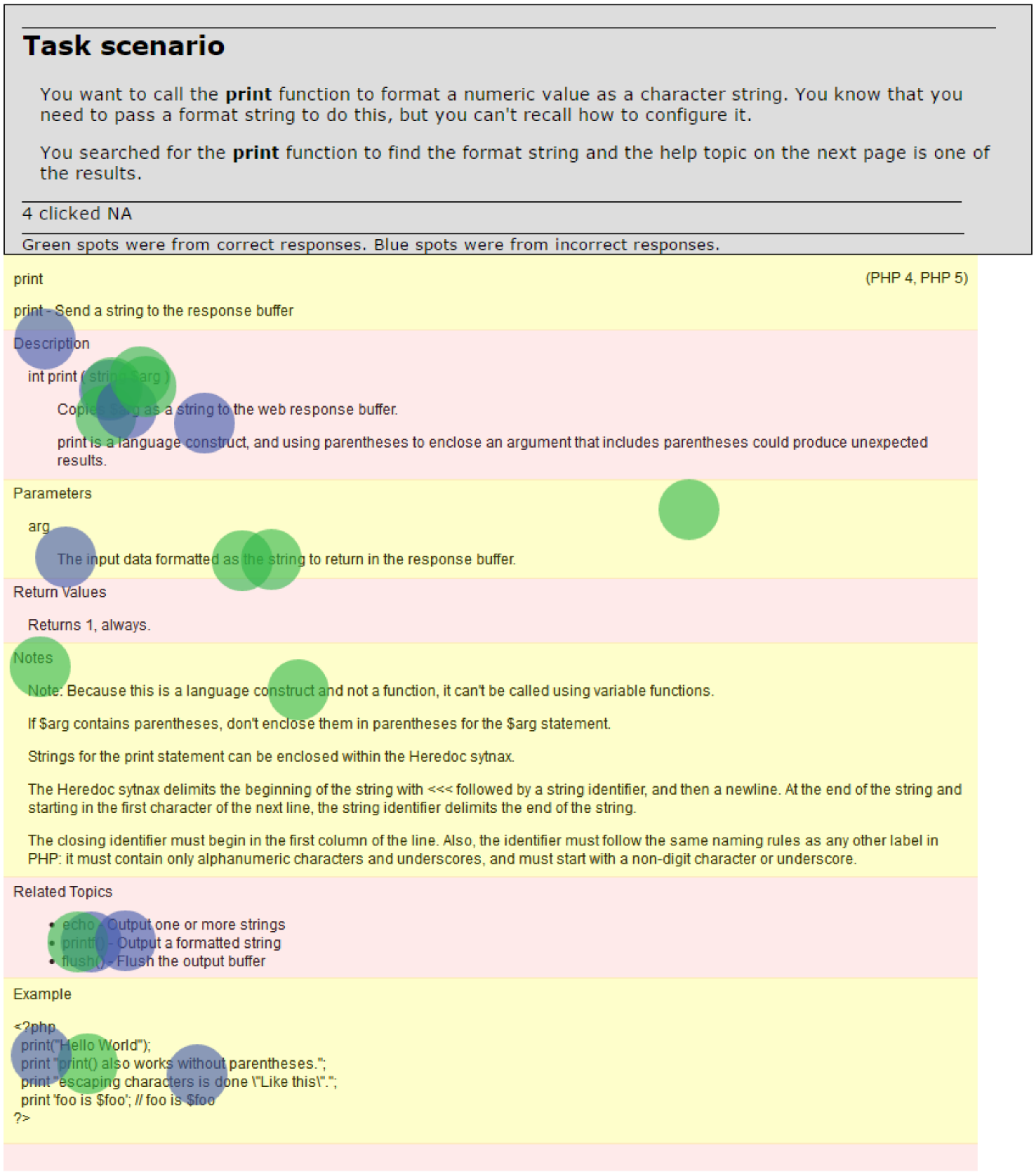
Return Values

1

Notes

Because this is a language construct, it can't be called using variable functions. (See Notes)

Figure B-14. Task 2 spot map for non-relevant question, high VDEC, low ICEC



Task scenario

You want to call the **print** function to format a numeric value as a character string. You know that you need to pass a format string to do this, but you can't recall how to configure it.

You searched for the **print** function to find the format string and the help topic on the next page is one of the results.

4 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print (PHP 4, PHP 5)

print - Send a string to the response buffer

Description

int print (string $arg)

Copies $arg as a string to the web response buffer.

print is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If $arg contains parentheses, don't enclose them in parentheses for the $arg statement.

Strings for the print statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

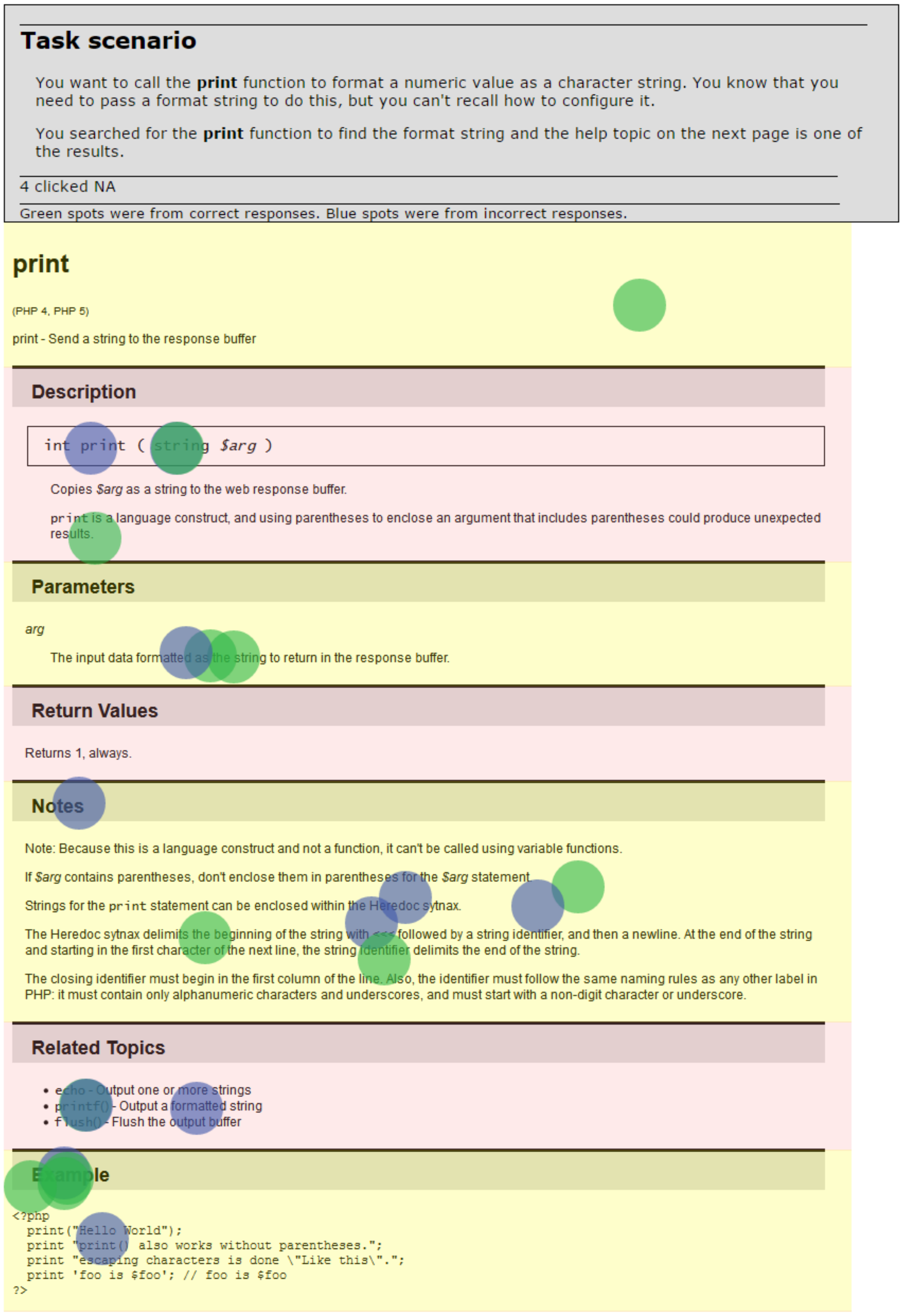
Related Topics

- echo - Output one or more strings
- printf() - Output a formatted string
- flush() - Flush the output buffer

Example

```
<?php
 print("Hello World");
 print "print() also works without parentheses.";
 print "escaping characters is done \"Like this\".";
 print 'foo is $foo'; // foo is $foo
?>
```

Figure B-15. Task 2 spot map for non-relevant question, low VDEC, high ICEC



Task scenario

You want to call the **print** function to format a numeric value as a character string. You know that you need to pass a format string to do this, but you can't recall how to configure it.

You searched for the **print** function to find the format string and the help topic on the next page is one of the results.

4 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

print

(PHP 4, PHP 5)

print - Send a string to the response buffer

Description

```
int print ( string $arg )
```

Copies *$arg* as a string to the web response buffer.

`print` is a language construct, and using parentheses to enclose an argument that includes parentheses could produce unexpected results.

Parameters

arg

The input data formatted as the string to return in the response buffer.

Return Values

Returns 1, always.

Notes

Note: Because this is a language construct and not a function, it can't be called using variable functions.

If *$arg* contains parentheses, don't enclose them in parentheses for the *$arg* statement.

Strings for the `print` statement can be enclosed within the Heredoc sytnax.

The Heredoc sytnax delimits the beginning of the string with <<< followed by a string identifier, and then a newline. At the end of the string and starting in the first character of the next line, the string identifier delimits the end of the string.

The closing identifier must begin in the first column of the line. Also, the identifier must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

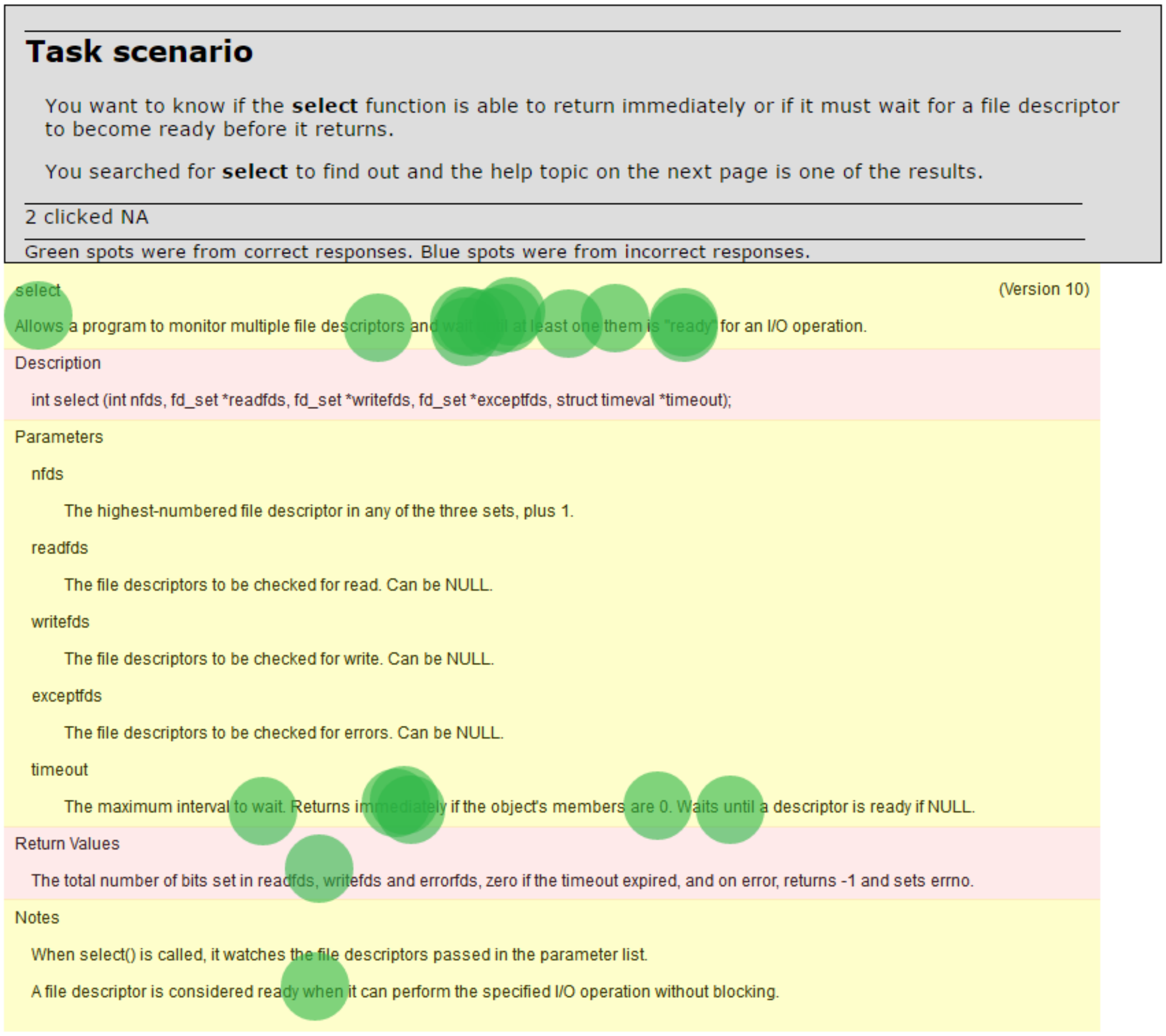
Related Topics

- `echo` - Output one or more strings
- `printf()` - Output a formatted string
- `flush()` - Flush the output buffer

Example

```
<?php
  print("Hello World");
  print "print() also works without parentheses.";
  print "escaping characters is done \"Like this\".";
  print 'foo is $foo'; // foo is $foo
?>
```

Figure B-16. Task 2 spot map for non-relevant question, high VDEC, high ICEC



Task scenario

You want to know if the **select** function is able to return immediately or if it must wait for a file descriptor to become ready before it returns.

You searched for **select** to find out and the help topic on the next page is one of the results.

2 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select (Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

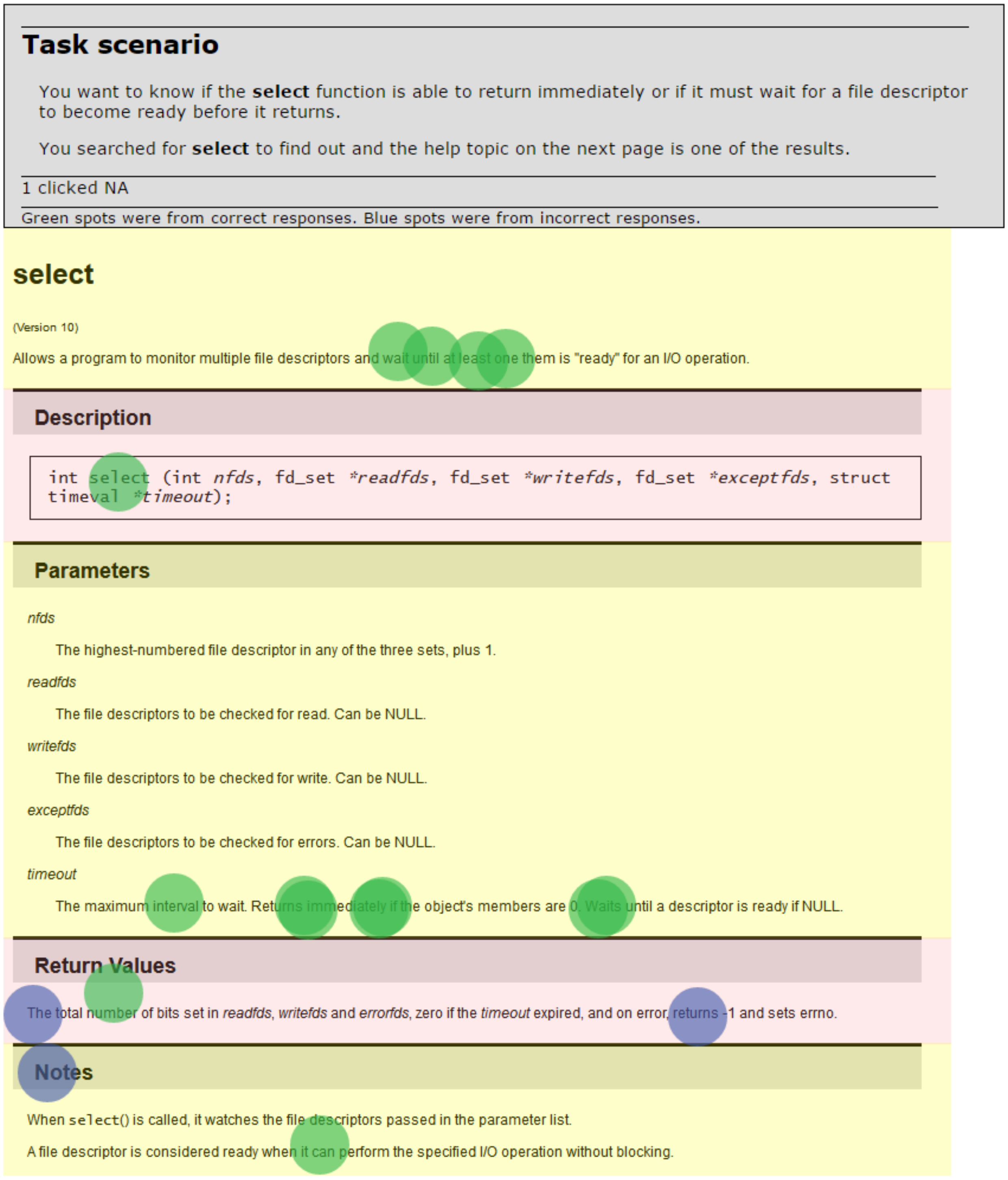
The total number of bits set in readfds, writefds and errorfds, zero if the timeout expired, and on error, returns -1 and sets errno.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure B-17. Task 3 spot map for relevant question, low VDEC, low ICEC


Task scenario

You want to know if the **select** function is able to return immediately or if it must wait for a file descriptor to become ready before it returns.

You searched for **select** to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

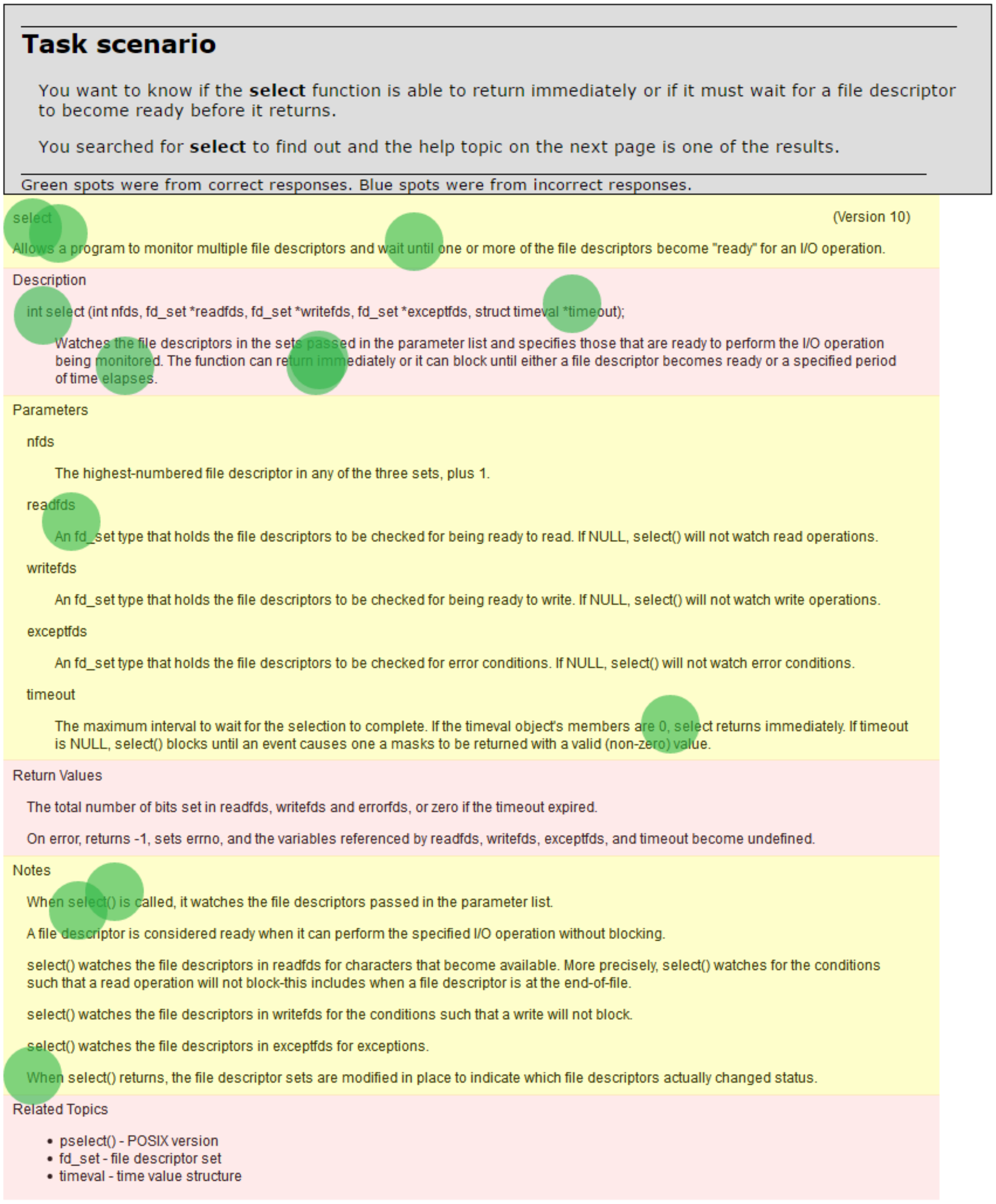
The total number of bits set in *readfds*, *writefds* and *errorfds*, zero if the *timeout* expired, and on error, returns -1 and sets errno.

Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure B-18. Task 3 spot map for relevant question, high VDEC, low ICEC



Task scenario

You want to know if the **select** function is able to return immediately or if it must wait for a file descriptor to become ready before it returns.

You searched for **select** to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

select (Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, select() will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, select() will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, select() will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, select() blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by readfds, writefds, exceptfds, and timeout become undefined.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

select() watches the file descriptors in readfds for characters that become available. More precisely, select() watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

select() watches the file descriptors in writefds for the conditions such that a write will not block.

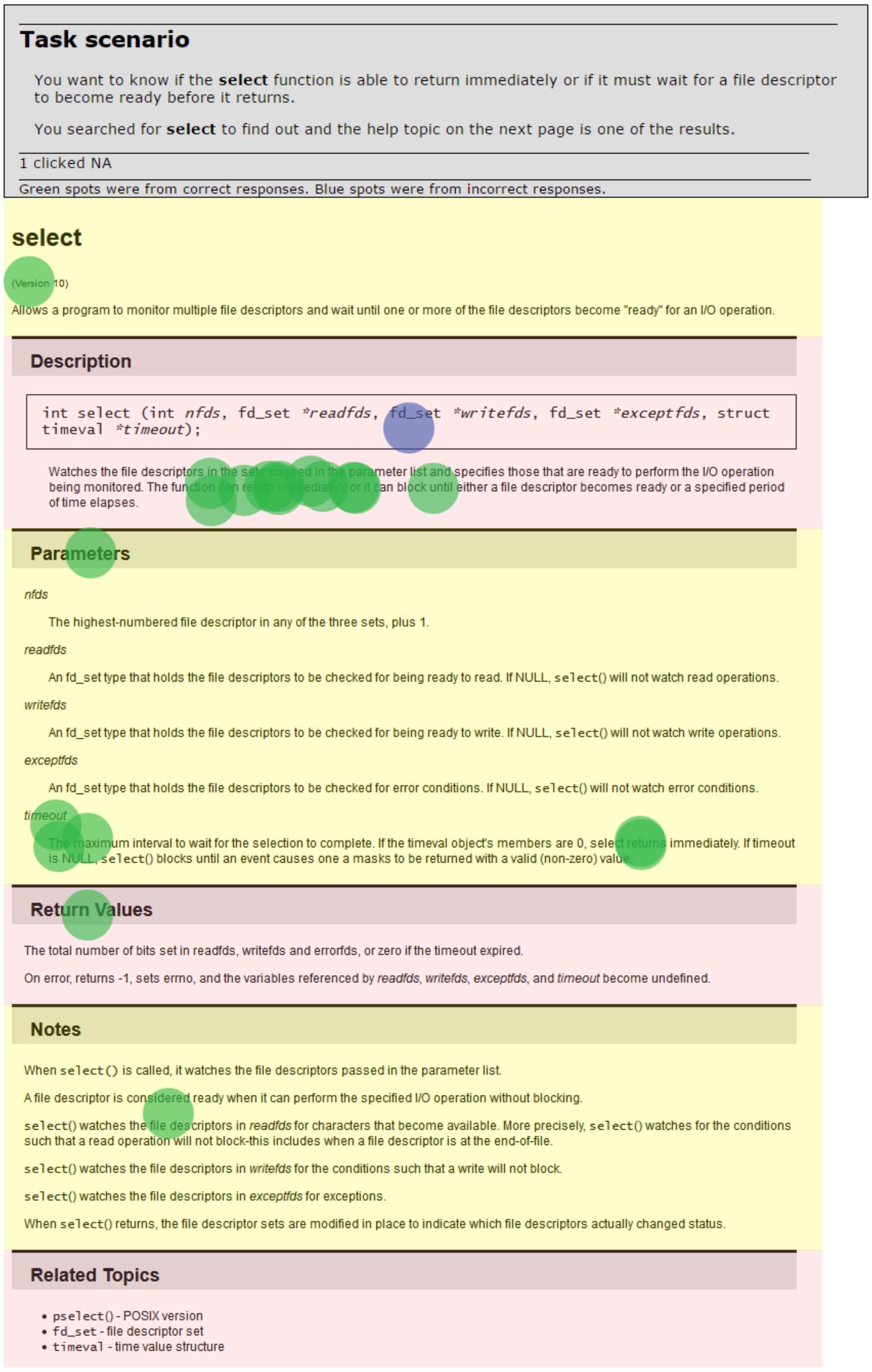
select() watches the file descriptors in exceptfds for exceptions.

When select() returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- pselect() - POSIX version
- fd_set - file descriptor set
- timeval - time value structure

Figure B-19. Task 3 spot map for relevant question, low VDEC, high ICEC



Task scenario

You want to know if the **select** function is able to return immediately or if it must wait for a file descriptor to become ready before it returns.

You searched for **select** to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, `select()` will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, `select()` will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, `select()` will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, `select()` blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by *readfds*, *writefds*, *exceptfds*, and *timeout* become undefined.

Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

`select()` watches the file descriptors in *readfds* for characters that become available. More precisely, `select()` watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

`select()` watches the file descriptors in *writefds* for the conditions such that a write will not block.

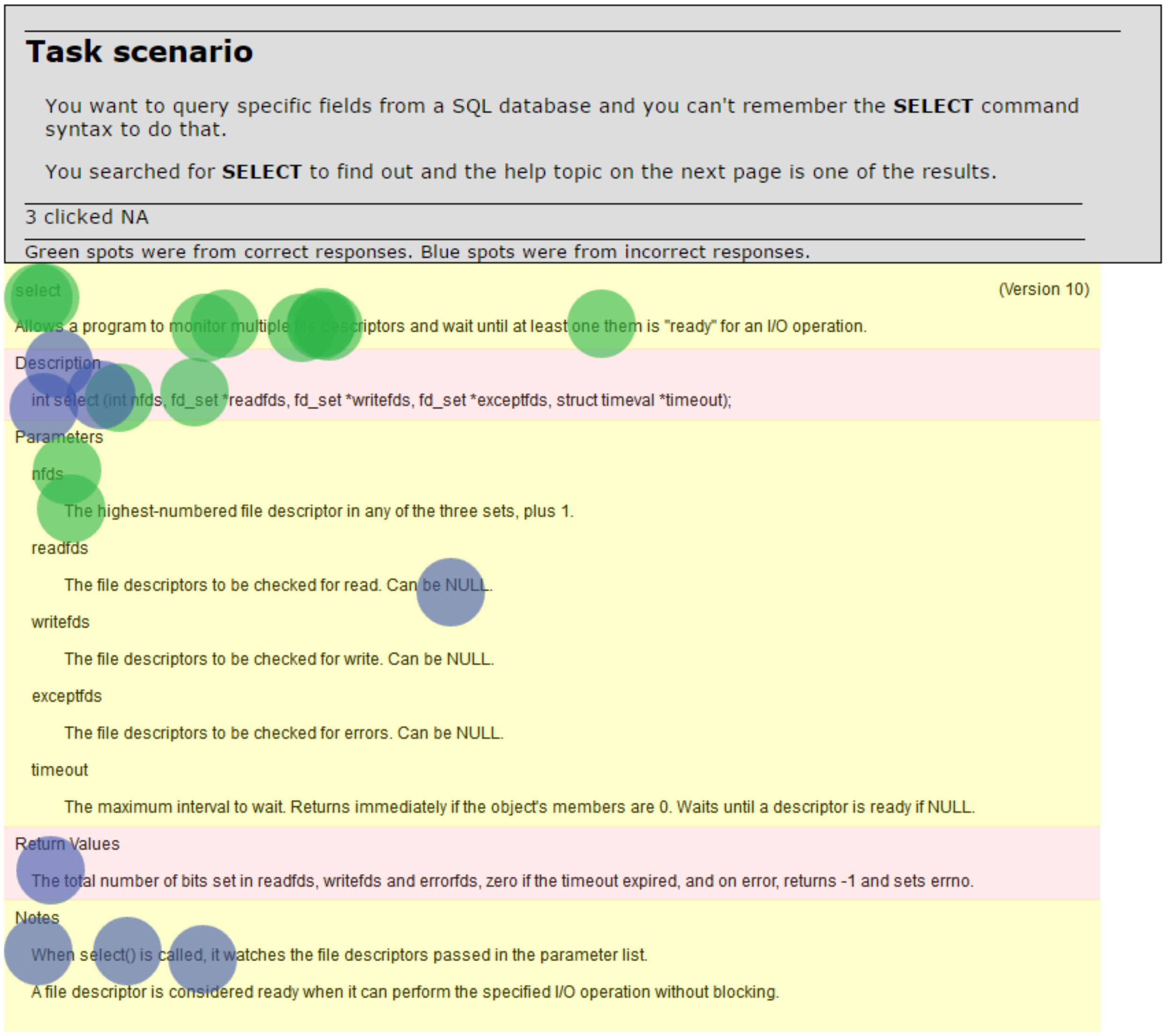
`select()` watches the file descriptors in *exceptfds* for exceptions.

When `select()` returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- `pselect()` - POSIX version
- `fd_set` - file descriptor set
- `timeval` - time value structure

Figure B-20. Task 3 spot map for relevant question, high VDEC, high ICEC



Task scenario

You want to query specific fields from a SQL database and you can't remember the **SELECT** command syntax to do that.

You searched for **SELECT** to find out and the help topic on the next page is one of the results.

3 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select (Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

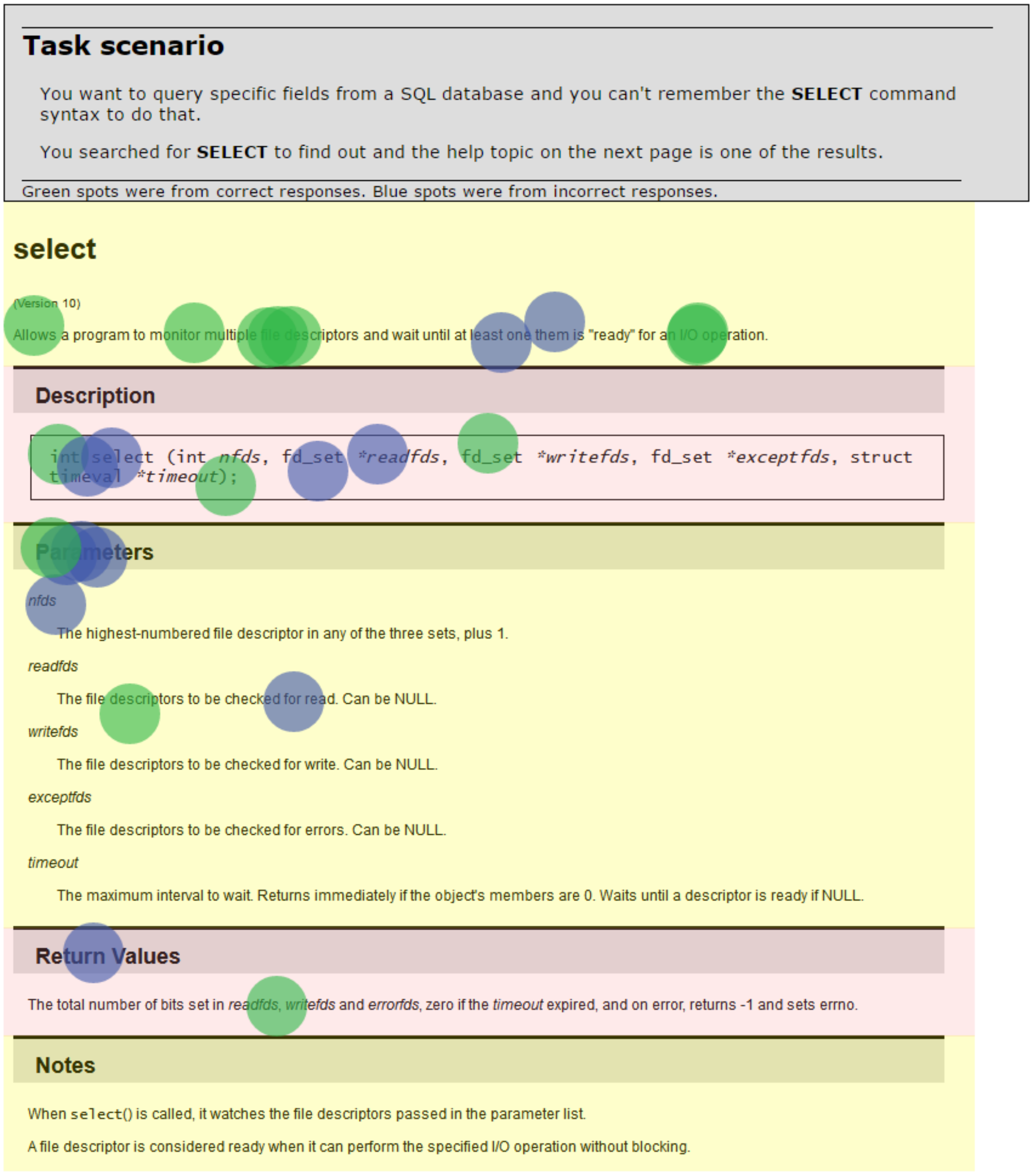
The total number of bits set in readfds, writefds and errorfds, zero if the timeout expired, and on error, returns -1 and sets errno.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure B-21. Task 3 spot map for non-relevant question, low VDEC, low ICEC



Task scenario

You want to query specific fields from a SQL database and you can't remember the **SELECT** command syntax to do that.

You searched for **SELECT** to find out and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until at least one them is "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

The file descriptors to be checked for read. Can be NULL.

writefds

The file descriptors to be checked for write. Can be NULL.

exceptfds

The file descriptors to be checked for errors. Can be NULL.

timeout

The maximum interval to wait. Returns immediately if the object's members are 0. Waits until a descriptor is ready if NULL.

Return Values

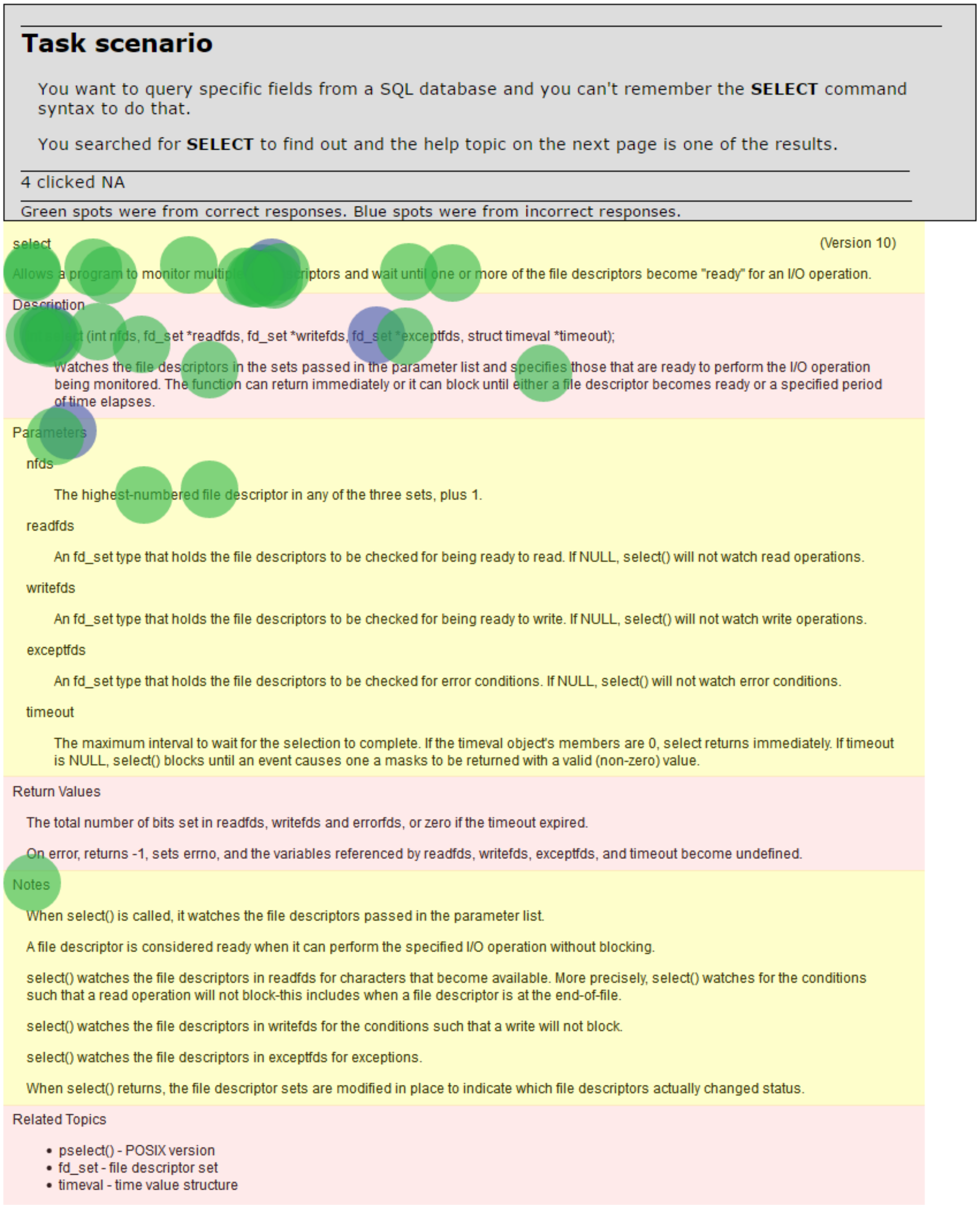
The total number of bits set in *readfds, writefds* and *errorfds*, zero if the *timeout* expired, and on error, returns -1 and sets errno.

Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

Figure B-22. Task 3 spot map for non-relevant question, high VDEC, low ICEC



Task scenario

You want to query specific fields from a SQL database and you can't remember the **SELECT** command syntax to do that.

You searched for **SELECT** to find out and the help topic on the next page is one of the results.

4 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select (Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, select() will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, select() will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, select() will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, select() blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by readfds, writefds, exceptfds, and timeout become undefined.

Notes

When select() is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

select() watches the file descriptors in readfds for characters that become available. More precisely, select() watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

select() watches the file descriptors in writefds for the conditions such that a write will not block.

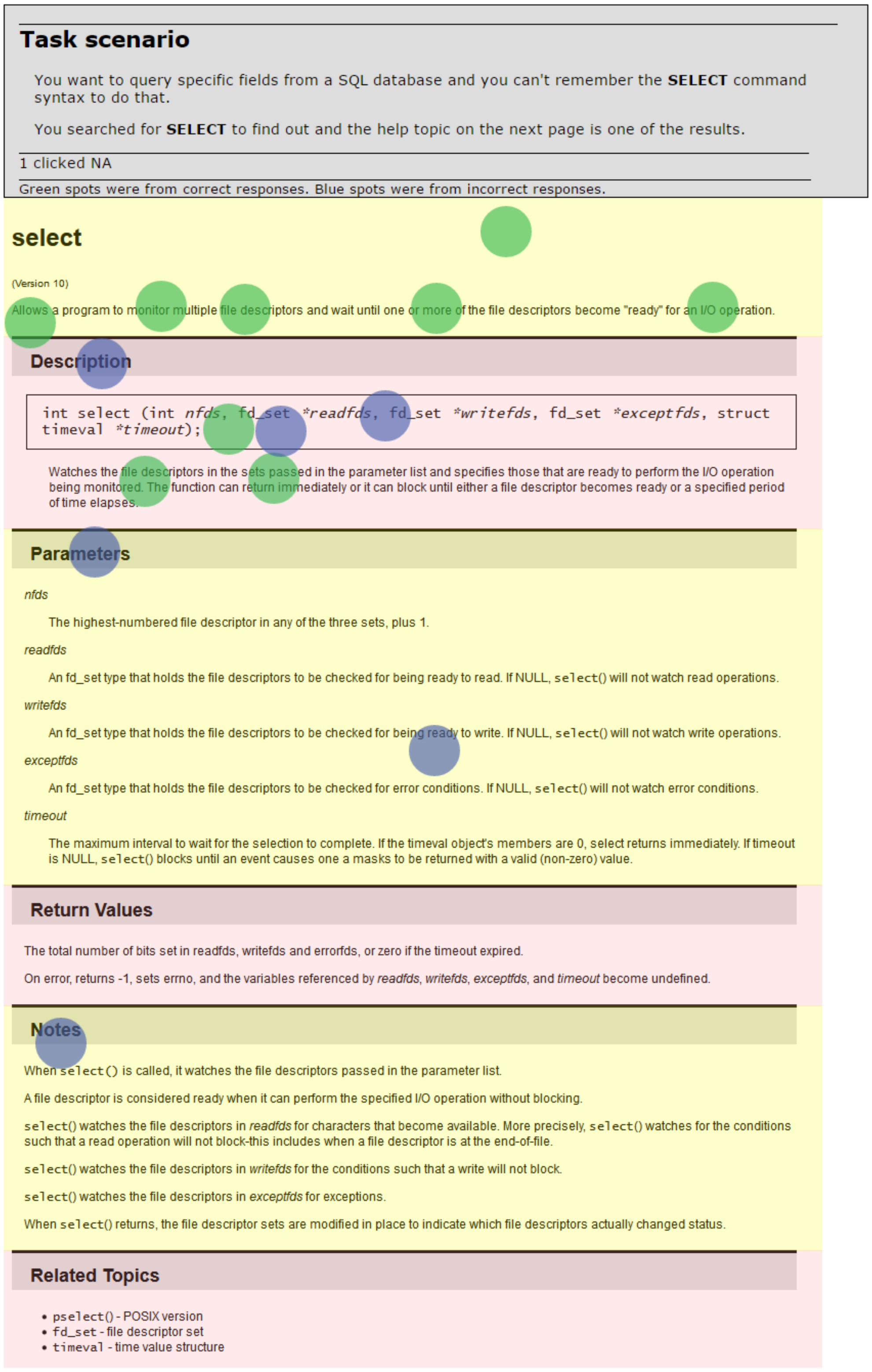
select() watches the file descriptors in exceptfds for exceptions.

When select() returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- pselect() - POSIX version
- fd_set - file descriptor set
- timeval - time value structure

Figure B-23. Task 3 spot map for non-relevant question, low VDEC, high ICEC


Task scenario

You want to query specific fields from a SQL database and you can't remember the **SELECT** command syntax to do that.

You searched for **SELECT** to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

select

(Version 10)

Allows a program to monitor multiple file descriptors and wait until one or more of the file descriptors become "ready" for an I/O operation.

Description

```
int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

Watches the file descriptors in the sets passed in the parameter list and specifies those that are ready to perform the I/O operation being monitored. The function can return immediately or it can block until either a file descriptor becomes ready or a specified period of time elapses.

Parameters

nfds

The highest-numbered file descriptor in any of the three sets, plus 1.

readfds

An fd_set type that holds the file descriptors to be checked for being ready to read. If NULL, `select`() will not watch read operations.

writefds

An fd_set type that holds the file descriptors to be checked for being ready to write. If NULL, `select`() will not watch write operations.

exceptfds

An fd_set type that holds the file descriptors to be checked for error conditions. If NULL, `select`() will not watch error conditions.

timeout

The maximum interval to wait for the selection to complete. If the timeval object's members are 0, select returns immediately. If timeout is NULL, `select`() blocks until an event causes one a masks to be returned with a valid (non-zero) value.

Return Values

The total number of bits set in readfds, writefds and errorfds, or zero if the timeout expired.

On error, returns -1, sets errno, and the variables referenced by *readfds*, *writefds*, *exceptfds*, and *timeout* become undefined.

Notes

When `select()` is called, it watches the file descriptors passed in the parameter list.

A file descriptor is considered ready when it can perform the specified I/O operation without blocking.

`select`() watches the file descriptors in *readfds* for characters that become available. More precisely, `select`() watches for the conditions such that a read operation will not block-this includes when a file descriptor is at the end-of-file.

`select`() watches the file descriptors in *writefds* for the conditions such that a write will not block.

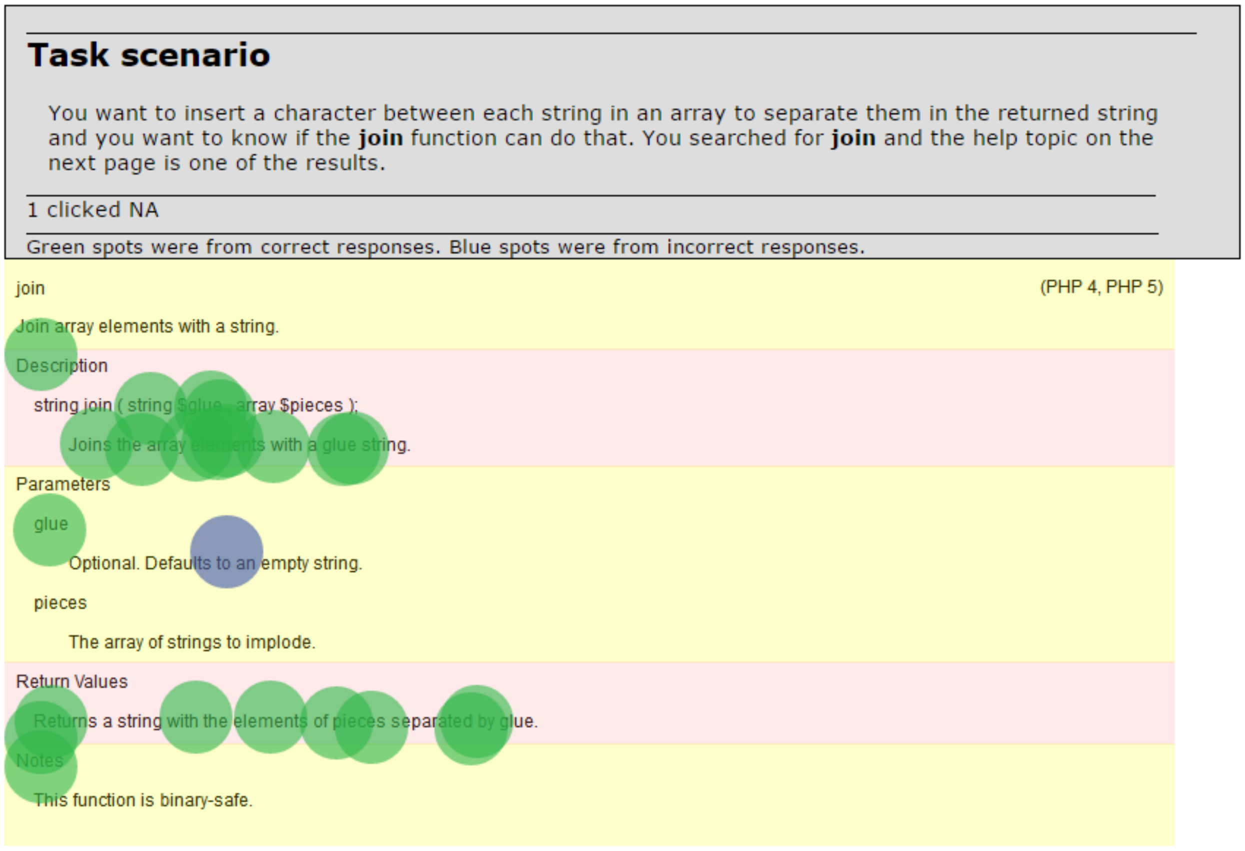
`select`() watches the file descriptors in *exceptfds* for exceptions.

When `select`() returns, the file descriptor sets are modified in place to indicate which file descriptors actually changed status.

Related Topics

- `pselect`() - POSIX version
- `fd_set` - file descriptor set
- `timeval` - time value structure

Figure B-24. Task 3 spot map for non-relevant question, high VDEC, high ICEC



Task scenario

You want to insert a character between each string in an array to separate them in the returned string and you want to know if the **join** function can do that. You searched for **join** and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

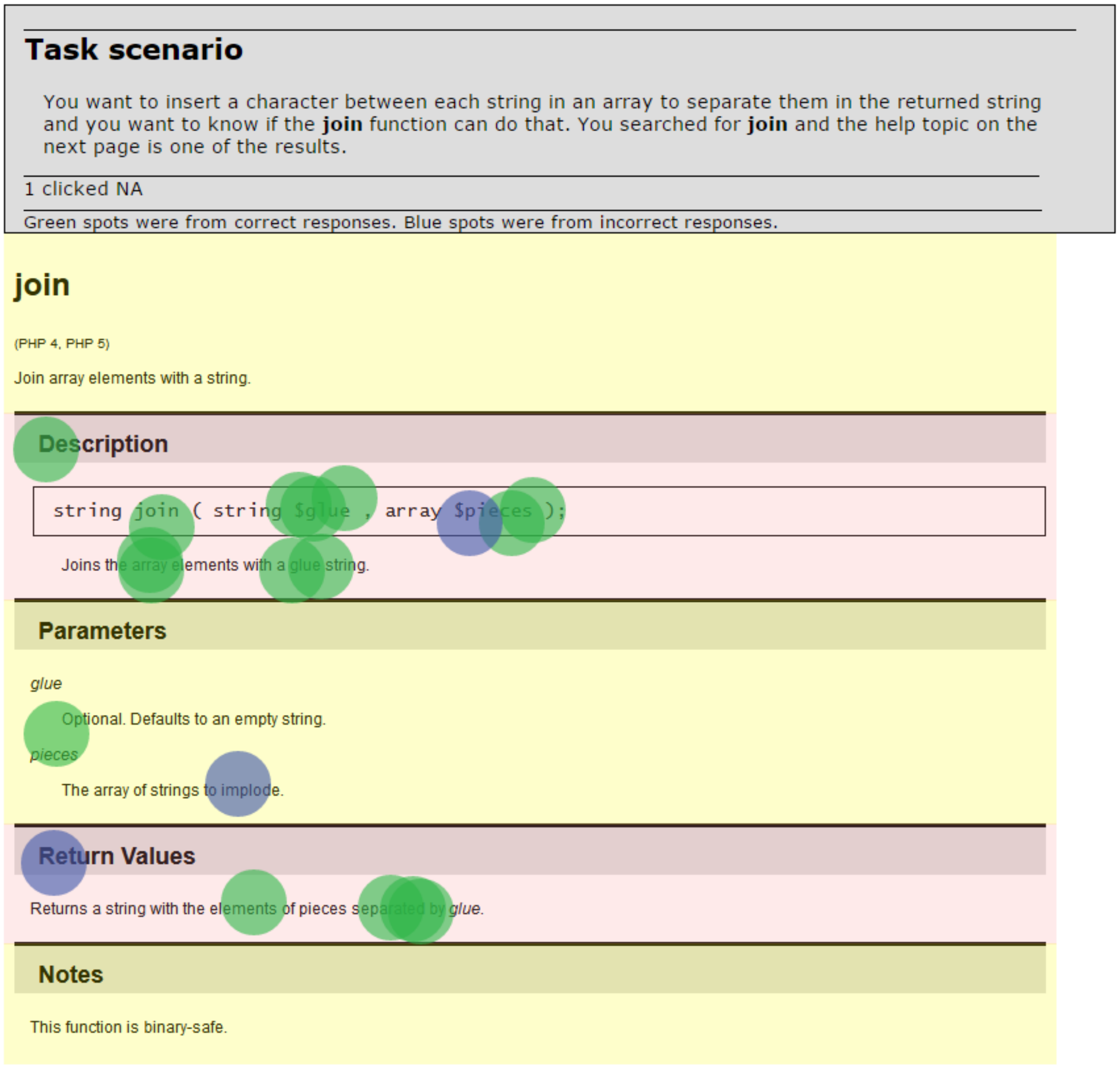
Return Values

Returns a string with the elements of pieces separated by glue.

Notes

This function is binary-safe.

Figure B-25. Task 4 spot map for relevant question, low VDEC, low ICEC



Task scenario

You want to insert a character between each string in an array to separate them in the returned string and you want to know if the **join** function can do that. You searched for **join** and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

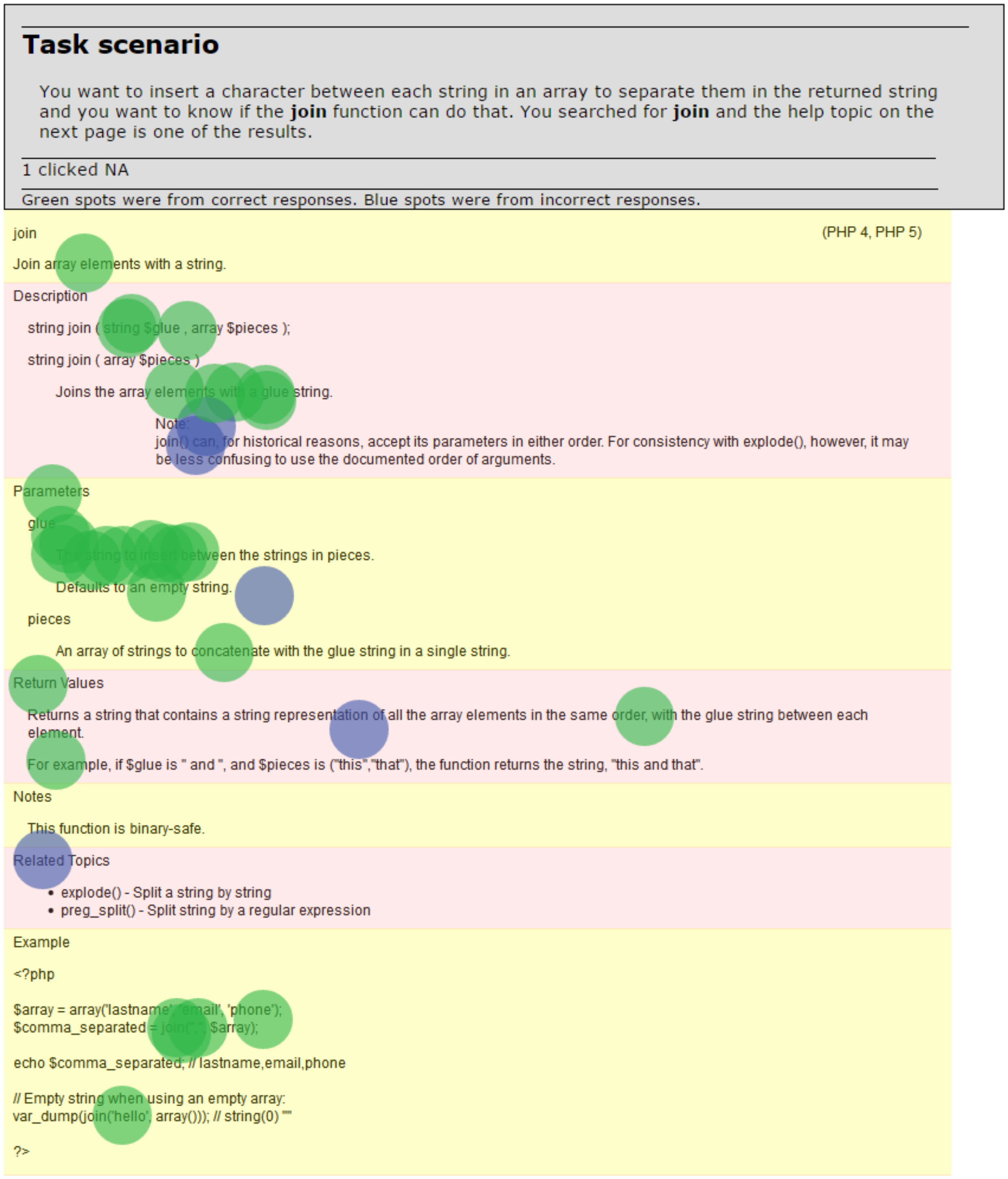
Return Values

Returns a string with the elements of pieces separated by *glue*.

Notes

This function is binary-safe.

Figure B-26. Task 4 spot map for relevant question, high VDEC, low ICEC



Task scenario

You want to insert a character between each string in an array to separate them in the returned string and you want to know if the **join** function can do that. You searched for **join** and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

string join (array $pieces)

Joins the array elements with a glue string.

Note:
join() can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the glue string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the glue string between each element.

For example, if $glue is " and ", and $pieces is ("this","that"), the function returns the string, "this and that".

Notes

This function is binary-safe.

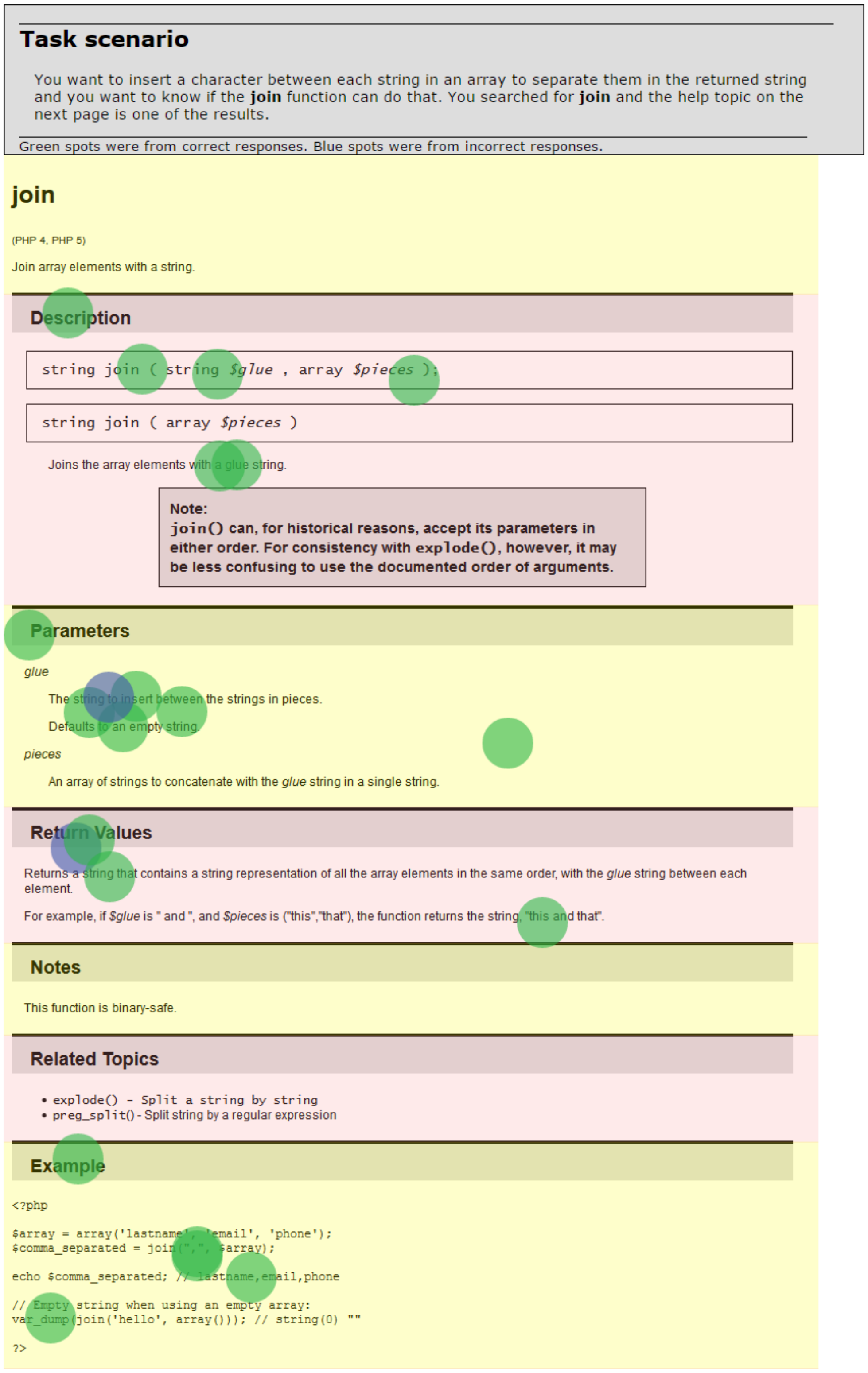
Related Topics

- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure B-27. Task 4 spot map for relevant question, low VDEC, high ICEC



Task scenario

You want to insert a character between each string in an array to separate them in the returned string and you want to know if the **join** function can do that. You searched for **join** and the help topic on the next page is one of the results.

Green spots were from correct responses. Blue spots were from incorrect responses.

join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

```
string join ( array $pieces )
```

Joins the array elements with a glue string.

Note:
`join()` can, for historical reasons, accept its parameters in either order. For consistency with `explode()`, however, it may be less confusing to use the documented order of arguments.

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the *glue* string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the *glue* string between each element.

For example, if *$glue* is " and ", and *$pieces* is ("this","that"), the function returns the string "this and that".

Notes

This function is binary-safe.

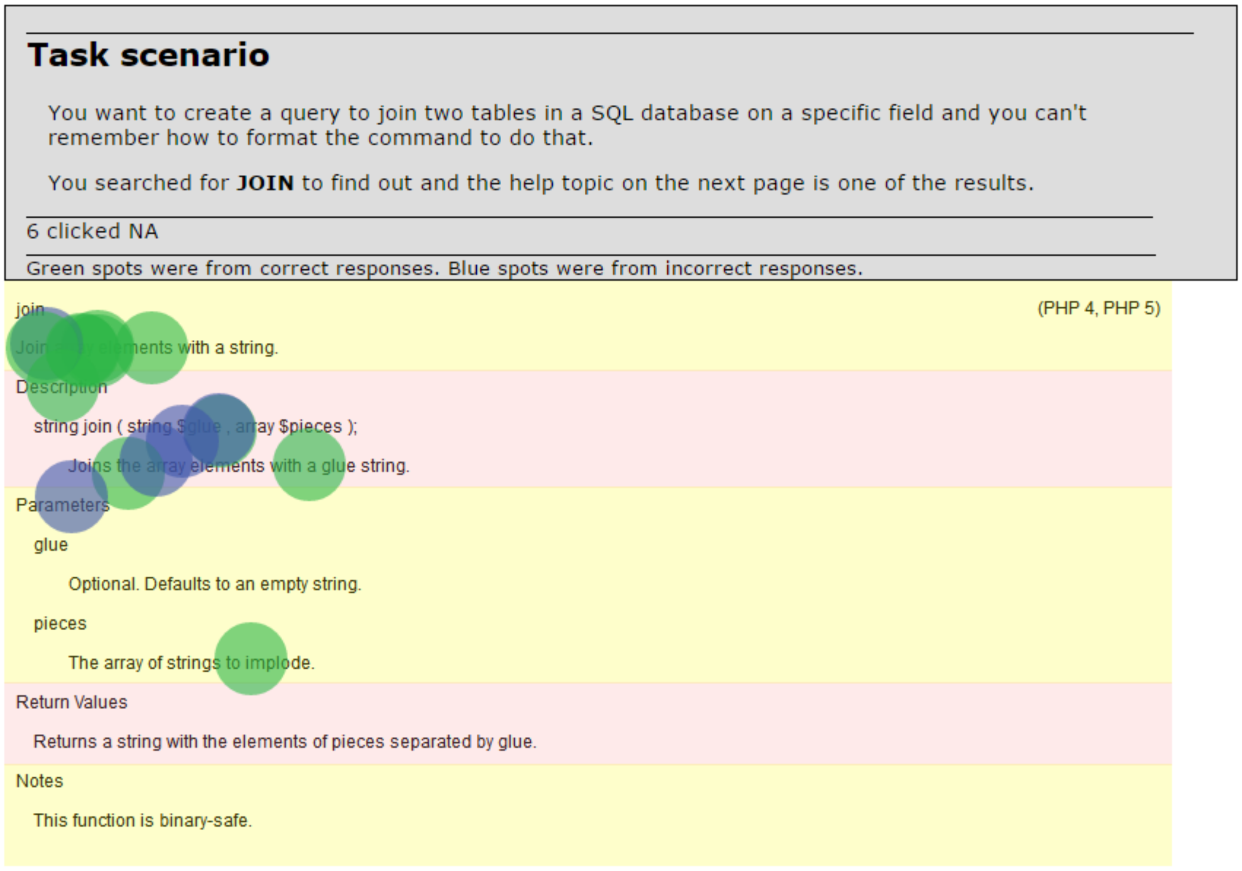
Related Topics

- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure B-28. Task 4 spot map for relevant question, high VDEC, high ICEC



Task scenario

You want to create a query to join two tables in a SQL database on a specific field and you can't remember how to format the command to do that.

You searched for **JOIN** to find out and the help topic on the next page is one of the results.

6 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

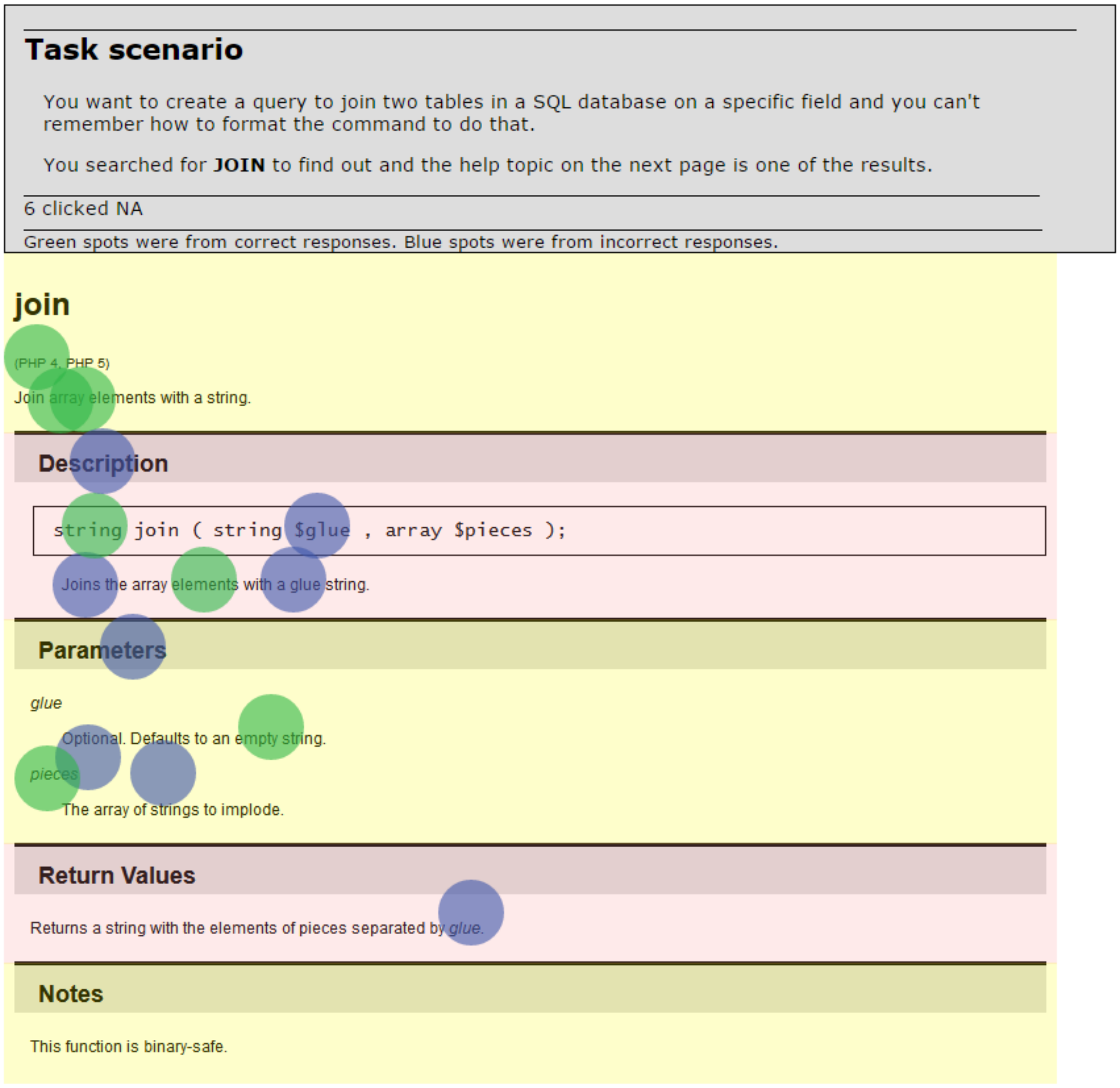
Return Values

Returns a string with the elements of pieces separated by glue.

Notes

This function is binary-safe.

Figure B-29. Task 4 spot map for non-relevant question, low VDEC, low ICEC



Task scenario

You want to create a query to join two tables in a SQL database on a specific field and you can't remember how to format the command to do that.

You searched for **JOIN** to find out and the help topic on the next page is one of the results.

6 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

Joins the array elements with a glue string.

Parameters

glue

Optional. Defaults to an empty string.

pieces

The array of strings to implode.

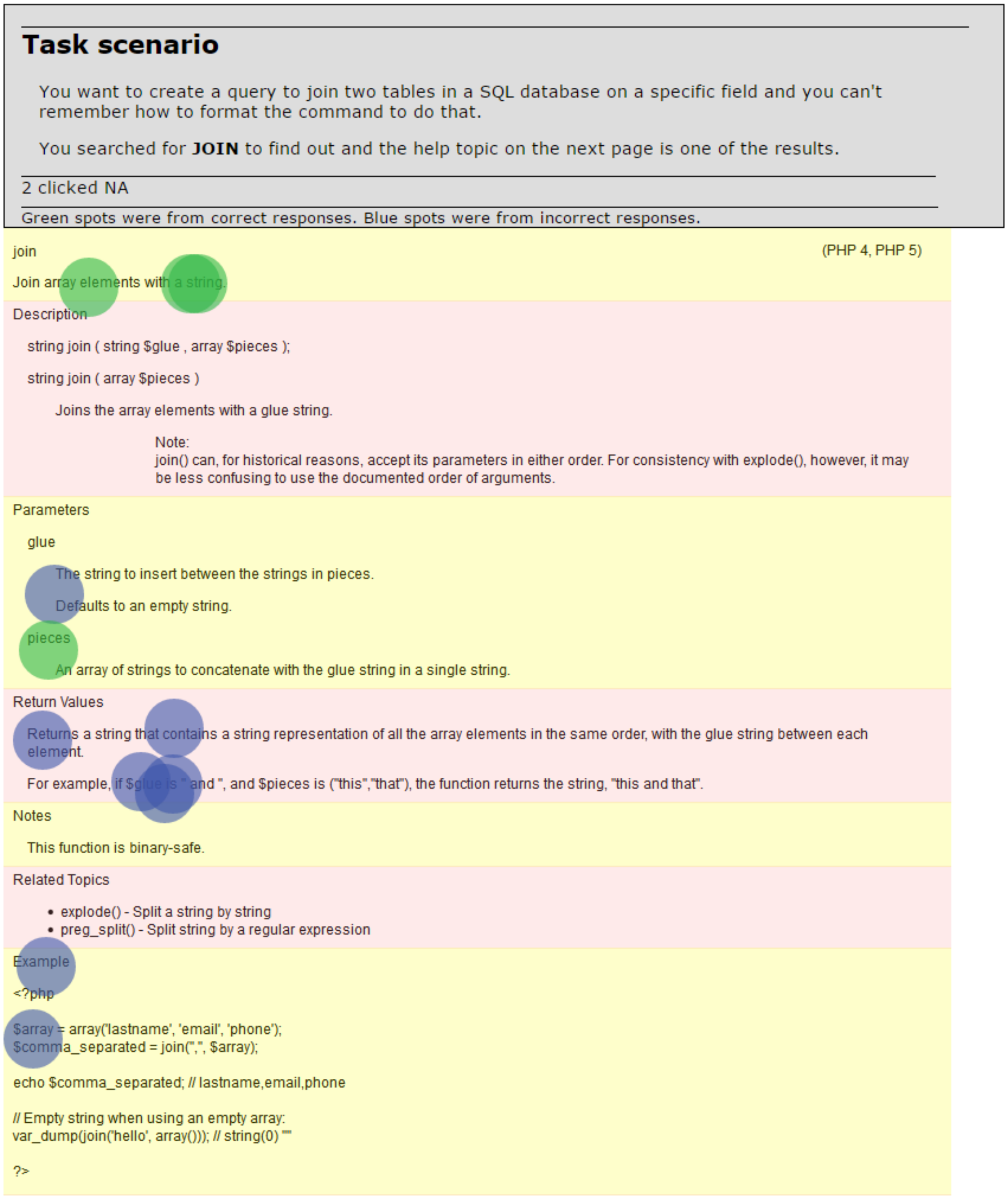
Return Values

Returns a string with the elements of pieces separated by *glue*.

Notes

This function is binary-safe.

Figure B-30. Task 4 spot map for non-relevant question, high VDEC, low ICEC



Task scenario

You want to create a query to join two tables in a SQL database on a specific field and you can't remember how to format the command to do that.

You searched for **JOIN** to find out and the help topic on the next page is one of the results.

2 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join (PHP 4, PHP 5)

Join array elements with a string.

Description

string join (string $glue , array $pieces);

string join (array $pieces)

Joins the array elements with a glue string.

Note:
join() can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the glue string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the glue string between each element.

For example, if $glue is " and ", and $pieces is ("this","that"), the function returns the string, "this and that".

Notes

This function is binary-safe.

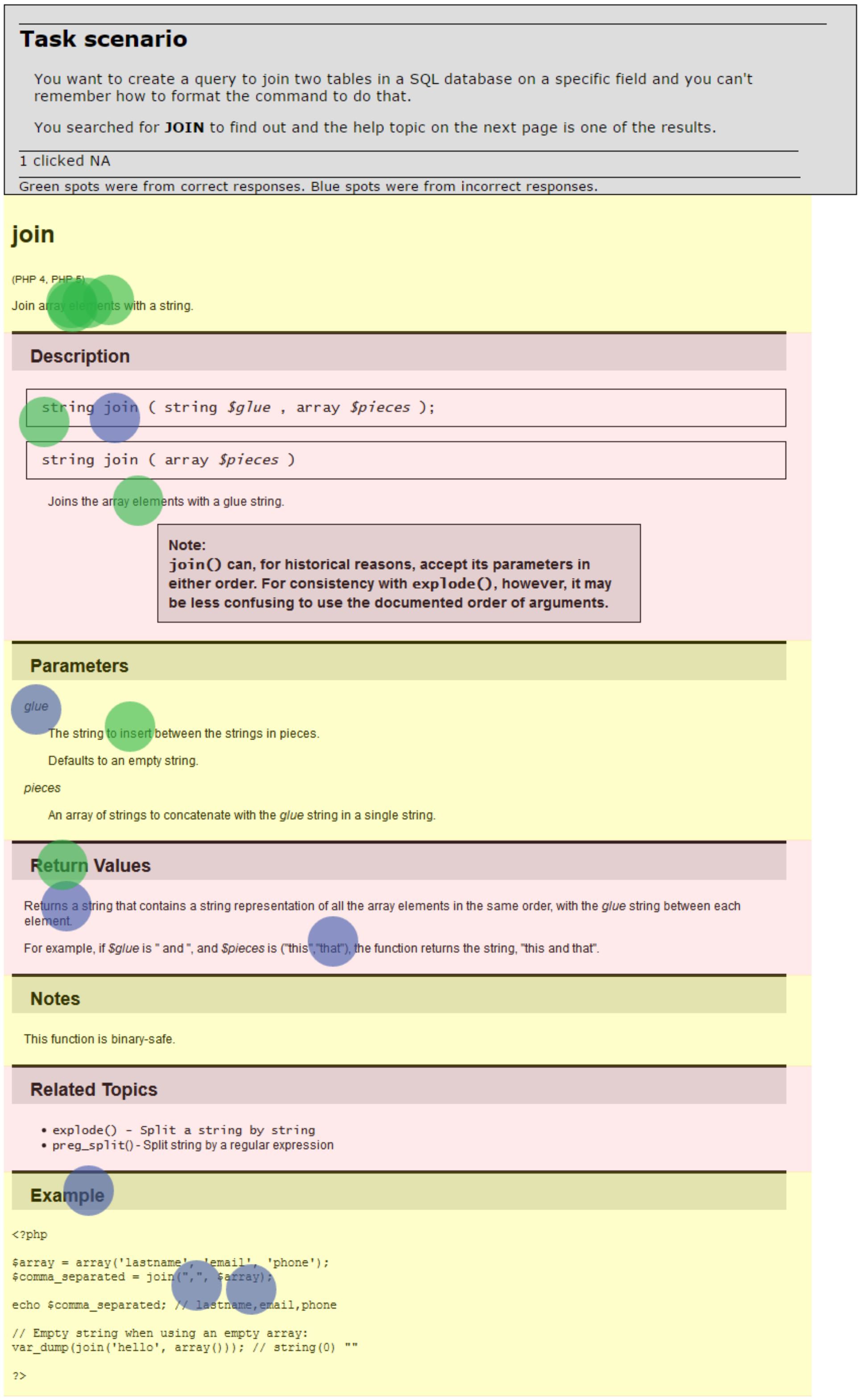
Related Topics

- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure B-31. Task 4 spot map for non-relevant question, low VDEC, high ICEC



Task scenario

You want to create a query to join two tables in a SQL database on a specific field and you can't remember how to format the command to do that.

You searched for **JOIN** to find out and the help topic on the next page is one of the results.

1 clicked NA

Green spots were from correct responses. Blue spots were from incorrect responses.

join

(PHP 4, PHP 5)

Join array elements with a string.

Description

```
string join ( string $glue , array $pieces );
```

```
string join ( array $pieces )
```

Joins the array elements with a glue string.

Note:
join() can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

Parameters

glue

The string to insert between the strings in pieces.

Defaults to an empty string.

pieces

An array of strings to concatenate with the *glue* string in a single string.

Return Values

Returns a string that contains a string representation of all the array elements in the same order, with the *glue* string between each element.

For example, if *$glue* is " and ", and *$pieces* is ("this", "that"), the function returns the string, "this and that".

Notes

This function is binary-safe.

Related Topics

- explode() - Split a string by string
- preg_split() - Split string by a regular expression

Example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = join(",", $array);

echo $comma_separated; // lastname,email,phone

// Empty string when using an empty array:
var_dump(join('hello', array())); // string(0) ""

?>
```

Figure B-32. Task 4 spot map for non-relevant question, high VDEC, high ICEC