PeopleCert
Software Development Skills

JavaScript Stream

Lesson 14

Study Guide

Copyright Details

The contents of this workshop are protected by copyright and can be reproduced under the Terms of Use agreed between PeopleCert and the ATO using this material only.

Material in this presentation has been sourced from the bibliography listed in the certification's Syllabus. All software-related images are used for educational purposes only and may differ across time.

No part of this document may be reproduced in any form without the written permission of PeopleCert International Ltd. Permission can be requested at www.peoplecert.org.

e-mail: info@peoplecert.org, www.peoplecert.org

Copyright © 2017-2019 PeopleCert International Ltd.
All rights reserved. No part of this publication may be reproduced or transmitted in any form and by any means (electronic, photocopying, recording or otherwise) except as permitted in writing by PeopleCert International Ltd. Enquiries for permission to reproduce, transmit or use for any purpose this material should be directed to the publisher.

DISCLAIMER
This publication is designed to provide helpful information to the reader. Although every care has been taken by PeopleCert International Ltd in the preparation of this publication, no representation or warranty (express or implied) is given by PeopleCert International Ltd. as publisher with respect as to the completeness, accuracy, reliability, suitability or availability of the information contained within it and neither shall PeopleCert International Ltd be responsible or liable for any loss or damage whatsoever (indicatively but not limited to, special, indirect, consequential) arising or resulting of virtue of information, instructions or advice contained within this publication.)

- Document Version: PC-SDS_SG 1.0 | February 2021

PeopleCert
All talents, certified.

PeopleCert: A Global Leader in Certification

- ✓ **Web** & **Paper based** exams in **25** languages
- ✓ **2,500 Accredited Training Organizations** worldwide
- ✓ Delivering exams across **200 countries every year**
- ✓ Comprehensive Portfolio of **500+ Exams** and Growing

AXELOS GLOBAL BEST PRACTICE · PeopleCert DevOps · digital marketing institute · IASSC · ISACA · Language Cert · LEAN IT ASSOCIATION · SIQ Quality Software Development · ECDL

PeopleCert
All talents, certified.

How to Use This Document

This document is your **PeopleCert Software Developer Skills Study Guide** to help you prepare for the **PeopleCert Software Developer Skills Foundation & Advanced examination**.

It is meant to provide you with a clear outline of everything covered in the course presentation by your instructor that will be on the PeopleCert Software Developer Skills Foundation & Advanced exams.

Your exams will be closed book. You will be given 120 minutes to complete it. It contains 100 multiple choice questions and to pass the exam you must achieve a grade of 65% or higher, or a minimum of 65/100 correct responses. For further details on your exam, including more information on question types and learning objectives, please refer to your course syllabus.

As you follow along, you may see that some material here is not replicated in the trainer presentation. This study guide includes questions, activities, knowledge checks, or other material in the presentation that are facilitated verbally by the instructor. It also does not contain content that is not examinable, but instead is designed to reinforce learning or add value to your course experience. It also provides valuable links and references, throughout the slides, which you can explore further to enhance your learning and understanding of the material provided in the study guide.

PeopleCert
All talents, certified.

Coding Bootcamp

LESSON 14

Web Design and Development Fundamentals (Front End)

Objectives:

- JavaScript Functions
- Object Prototypes
- Exception Handling
- Document Object Model
- JavaScript Events
- JSON
- XML Syntax and Validation
- Data Validation

Syllabus Items:

- 2.1 Developer Tools
- 2.2 Programming Languages and Paradigms
- 2.3 Programming Basics
- 4.6 JavaScript/ jQuery
- 4.9 Web Publishing and Hosting

Syllabus

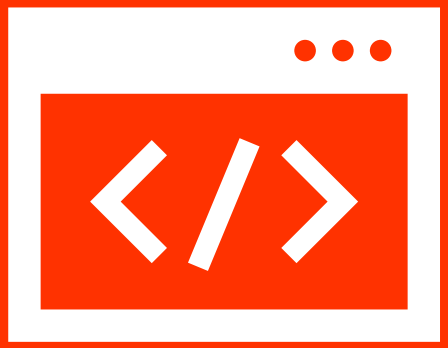
Category	Topic	Task
FSD_2 Introduction to Programming	**2.1 Developer Tools**	2.1.1 Understand and use editors and compilers
		2.1.2 Perform static analysis (e.g. FindBugs)
		2.1.3 Understand and use the Unix command line and its tools
		2.1.4 Use build tools (e.g. make, Maven)
		2.1.5 Understand and use IDE
		2.1.6 Understand and use GitHub and GitHub issues
		2.1.8 Understand and perform debugging
	2.2 Programming Languages and Paradigms	2.2.1 List the different types of programming languages like: procedural, object-oriented, data oriented, scripting. Distinguish between compiled and interpreted languages
		2.2.2 Distinguish between back-end and front-end software development

Syllabus

Category	Topic	Task
FSD_2 Introduction to Programming	**2.3 Programming Basics**	2.3.1 Define variables, write and execute a simple program
		2.3.2 List and use Command-line basics, like: editing, execution, stopping, paging, redirecting, piping
		2.3.3 Use Strings and Console Output, including creating string literals, calling a variety of string methods
		2.3.4 Use conditionals, loops and control flow
		2.3.5 Define functions/methods, use procedural programming
		2.3.6 Define and use lists, list comprehensions, dictionaries
		2.3.7 Handle input-output
FSD_4 Web Design and Development Fundamentals (Front-End)	**4.6 JavaScript/ jQuery**	4.6.1 Code in JavaScript, variables, functions
		4.6.2 Code using advance java script: if conditions, loops
		4.6.3 Improve usability of forms, validate data from the user
	4.9 Web Publishing and Hosting	4.9.1 Outline and use Web Publishing / Hosting and Continuous Integration techniques
		4.9.2 Understand the role of the web client and the web server - the HTTP protocol

Contents | Learning Objectives

- ✓ Introduction to JavaScript
- ✓ Client-Side Scripting
- ✓ Language Syntax
- ✓ Variables
- ✓ Strings
- ✓ Numbers
- ✓ Booleans
- ✓ Operators
- ✓ Conditionals
- ✓ Loops
- ✓ Arrays
- ✓ Objects
- ✓ Understand the Document Object Model and the selection process
- ✓ Familiarize with the JavaScript events
- ✓ Understand the JSON and XML model
- ✓ Learn how to validate the form data



Coding Bootcamp

LESSON 14

The Document Object Model (DOM) – Part II

Do you remember what DOM is?

Definition:

*The **Document Object Model (DOM)** is a programming interface for HTML.*

- It provides a structured representation of the document (e.g., a web page) as a tree.
- It defines methods that allow access to the tree, so that they can change the document structure, style and content.
- JavaScript is almost always used to interact with the HTML document in which it is contained
- This is accomplished through a programming interface (API) called the Document Object Model
- According to the W3C, the **DOM** is a:

  Platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

Source: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

style

Definition: *The style property is used to get or set a specific style of an element using different CSS properties.*

- The style property returns a CSSStyleDeclaration object, which represents an element's style attribute.
- It is not possible to set styles by assigning a string to the style property, e.g., element.style = "color: red;". To set the style of an element, append a "CSS" property to style and specify a value
- **Examples**

```
document.getElementById("myH1").style.color = "red";
// set the background color of an element to red
element.style.backgroundColor = "red";
// get the first <style> element
var x = document.getElementsByTagName("STYLE")[0];
```

write Method

Definition: *The write() method writes HTML expressions or JavaScript code to a document.*

- *The write() method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML.*
- Note: When this method is not used for testing, it is often used to write some text to an output stream opened by the document.open() method.
- The document.writeln() method is similar to write(), only it adds a newline character after each statement.
- **Example**

```
document.write("<h1>Hello World!</h1><p>Have a nice day!</p>");
```

Source: https://www.w3schools.com/jsref/prop_html_style.asp

Screen width and height Property

- The width property returns the total width of the user's screen, in pixels.

  var x = "Total Width: " + screen.width;

- The height property to get the total height of the user's screen.

  var x = "Total Width: " + screen.height;

- Similar Properties:
  - **availHeight.** The availHeight property returns the height of the user's screen, in pixels, minus interface features like the Windows Taskbar.
  - **availWidth.** The availWidth property returns the width of the user's screen, in pixels, minus interface features like the Windows Taskbar.

Source: https://www.w3schools.com/jsref/prop_screen_width.asp

childNodes Property

- The childNodes property returns a collection of a node's child nodes, as a NodeList object.
- The nodes in the collection are sorted as they appear in the source code and can be accessed by index numbers. The index starts at 0.
- Whitespace inside elements is considered as text, and text is considered as nodes. Comments are also considered as nodes.
- We can use the length property of the NodeList object to determine the number of child nodes, then we can loop through all child nodes and extract the info you want.
- This property is read-only.
- To return a collection of a node's element nodes (excluding text and comment nodes), use the children property.
- element.childNodes[0] will produce the same result as the firstChild property.

children Property

- The children property returns a collection of an element's child elements, as an HTMLCollection object.
- The elements in the collection are sorted as they appear in the source code and can be accessed by index numbers. The index starts at 0.
- The difference between this property and childNodes, is that childNodes contain all nodes, including text nodes and comment nodes, while children only contain element nodes.

Source: https://www.w3schools.com/jsref/prop_node_childnodes.asp

childNodes vs children Property

- Understand that .children is a property of an Element. Only Elements have .children, and these children are all of type Element.
- However, .childNodes is a property of Node. .childNodes can contain any node.
- A concrete **example**:

```
let el = document.createElement("div");
el.textContent = "foo";

el.childNodes.length === 1; // Contains a Text node child.
el.children.length === 0;   // No Element children.
```

- Most of the time, we want to use children because generally we don't want to loop over Text or Comment nodes in our DOM manipulation.

Source: https://stackoverflow.com/questions/132564/whats-the-difference-between-an-element-an

Class Exercises | Discussion

#1

- Consider the following HTML code

```
<body>
<ul>
      <li>AAA</li>
      <li>BBB</li>
      <li>CCC</li>
      <li>DDD</li>
      <li>EEE</li>
  </ul>
</body>
```

- Access the second list element
- Change second's element color
- Add FFF element in list before EEE

#2

- Create HTML table dynamically

Handling Attributes

- **getAttribute()** method returns the value of the attribute with the specified name, of an element.

var x = document.getElementsByTagName("H1")[0].getAttribute("class");

- **getAttributeNode()** method returns the attribute node with the specified name of an element, as an Attr object.

```
var elmnt = document.getElementsByTagName("H1")[0];
var attr = elmnt.getAttributeNode("class").value;
```

- **hasAttribute()** method returns true if the specified attribute exists, otherwise it returns false

```
var x = document.getElementById("myBtn").hasAttribute("onclick");
```

- **hasAttributes()** method returns true if the specified node has any attributes, otherwise false. If the specified node is not an Element node, the return value is always false.

```
var x = document.body.hasAttributes()
```

- **setAttribute()** method adds the specified attribute to an element and gives it the specified value. If the specified attribute already exists, only the value is set/changed. Although it is possible to add the style attribute with a value to an element with this method, it is recommended that you use properties of the Style object instead for inline styling, because this will not overwrite other CSS properties that may be specified in the style attribute.

document.getElementsByTagName("H1")[0].setAttribute("class", "democlass");

- **setAttributeNode()** method adds the specified attribute node to an element. If the specified attribute already exists, this method replaces it. The return value of this method is an Attr object

```
// Get the first <h1> element in the document
var h1 = document.getElementsByTagName("H1")[0];
// Create a "class" attribute
var att = document.createAttribute("class");
// Set the value of the class attribute
att.value = "democlass";
// Add the class attribute to <h1>
h1.setAttributeNode(att);
```

Source: https://www.w3schools.com/jsref/prop_node_attributes.asp

Custom Attributes

- Custom data attributes are intended to store custom data private to the page or application, for which there are no more appropriate attributes or elements.
- These attributes are not intended for use by software that is independent of the site that uses the attributes.
- Every HTML element may have any number of custom data attributes specified, with any value.
- Prefixing the custom attributes with data- ensures that they will be completely ignored by the user agent. As far as the browser and indeed the website's end user are concerned, this data does not exist.

Usage

- To store the initial height or opacity of an element which might be required in later JavaScript animation calculations
- To store parameters for external data loaded via JavaScript
- To store custom web analytics tagging data
- To store data about the health, ammo, or lives of an element in a JavaScript game
- To power accessible JavaScript <video> subtitles

Example

```
<div id="product1" data-product-category="clothing">
    Cotton Shirt
</div>
```

Source: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

Custom Attributes | Example

html

```
<div id="product1" data-product-category="clothing">
    Cotton Shirt
</div>
```

Script.js

```
function getElementAttribute(elemID) {
  var theElement = document.getElementById(elemID);
  var theAttribute = theElement.getAttribute('data-product-category');
  alert(theAttribute);
}


getElementAttribute("product1"); // Outputs clothing
```

Class Exercise | Discussion

- Consider the following supermarket's product list

```
<ul>
    <li>Tomatoes</li>
    <li> Bananas</li>
    <li>Peppers</li>
    <li>Pork</li>
    <li>Onions</li>
    <li>Chicken</li>
    <li>Apples</li>
 </ul>
```

- Use different color for each category(vegetables, fruits, meat)
- On click get element value

Popups and window Methods

- A popup window is one of the oldest methods to show additional document to user.

  window.open('https://www.peoplecert.org')

- …And it will open a new window with given URL. Most modern browsers are configured to open new tabs instead of separate windows.
- Popups exist from really ancient times. The initial idea was to show another content without closing the main window. As of now, there are other ways to do that: we can load content dynamically with fetch and show it in a dynamically generated . So, popups is not something we use everyday.
- Also, popups are tricky on mobile devices, that don't show multiple windows simultaneously.
- Still, there are tasks where popups are still used because:
  - A popup is a separate window with its own independent JavaScript environment. So opening a popup with a third-party non-trusted site is safe.
  - It's very easy to open a popup.
  - A popup can navigate (change URL) and send messages to the opener window.

Source: https://javascript.info/popup-windows

window.open

- The syntax to open a popup is: window.open(url, name, params):
- **url** : An URL to load into the new window.
- **Name**: A name of the new window. Each window has a window.name, and here we can specify which window to use for the popup. If there's already a window with such name – the given URL opens in it, otherwise a new window is opened.
- **Params** :The configuration string for the new window. It contains settings, delimited by a comma. There must be no spaces in params, for instance: width:200,height=100.
- Settings for params:
  - Position:
    - **left/top** (numeric) – coordinates of the window top-left corner on the screen. There is a limitation: a new window cannot be positioned offscreen.
    - **width/height** (numeric) – width and height of a new window. There is a limit on minimal width/height, so it's impossible to create an invisible window.
  - Window features:
    - **menubar** (yes/no) – shows or hides the browser menu on the new window.
    - **toolbar** (yes/no) – shows or hides the browser navigation bar (back, forward, reload etc) on the new window.
    - **location** (yes/no) – shows or hides the URL field in the new window. FF and IE don't allow to hide it by default.
    - **status** (yes/no) – shows or hides the status bar. Again, most browsers force it to show.
    - **resizable** (yes/no) – allows to disable the resize for the new window. Not recommended.
    - **scrollbars** (yes/no) – allows to disable the scrollbars for the new window. Not recommended.

Source: https://javascript.info/popup-windows

Window | Example

- open a new window and add some content

```
<button onclick="myFunction()">Open</button>

<script>
function myFunction() {
  var w = window.open();
  w.document.open();
  w.document.write("<h1>Hello World!</h1>");
  w.document.close();
}
</script>
```

Popup Example

- open a new popup window and add some content

```
<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  let myWindow = window.open("", "", "width=200,height=100");
  myWindow.document.write("Hello, world!")
}
</script>
```

Enable/Disable | Show/Hide | Example

- Disable and enable, using buttons, a dropdown list and show-hide the buttons

```
function disable() {
  document.getElementById("mySelect").disabled=true;
  document.getElementById("disbtn").style.display = "none";
  document.getElementById("enbtn").style.display = "block";
}
function enable() {
  document.getElementById("mySelect").disabled=false;
  document.getElementById("disbtn").style.display = "block";
  document.getElementById("enbtn").style.display = "none";
}
<form>
<select id="mySelect">
  <option>Apple</option>
  <option>Pear</option>
  <option>Banana</option>
  <option>Orange</option>
</select>
<br><br>
<input type="button" onclick="disable()" id="disbtn" value="Disable list">
<input type="button" onclick="enable()" id="enbtn" style="display:none;" value="Enable list">
</form>
```

Change img Source | Example

- Add 2 buttons and with javascript change image source

```
<img id="myImg" src="https://i.picsum.photos/id/238/200/300.jpg" width="107" height="98">
  <br><br>
  <button onclick="document.getElementById('myImg').src='https://i.picsum.photos/id/238/200/300.jpg'">img 1</button>
  <button onclick="document.getElementById('myImg').src='https://i.picsum.photos/id/237/200/300.jpg'">img 2</button>
```

Differences between textContent/innerText/innerHTML

- **textContent**
  - If the node is a document or a Doctype, textContent returns null.
  - If the node is a CDATA section, comment, processing instruction, or text node, textContent returns the text inside the node, i.e., the Node.nodeValue.
  - For other node types, textContent returns the concatenation of the textContent of every child node, excluding comments and processing instructions. (This is an empty string if the node has no children.)
  - textContent uses straight text, does not parse HTML, and is faster.
- **innerText**
  - textContent gets the content of all elements, including <script> and <style> elements. In contrast, innerText only shows "human-readable" elements.
  - textContent returns every element in the node. In contrast, innerText is aware of styling and won't return the text of "hidden" elements.
  - Moreover, since innerText takes CSS styles into account, reading the value of innerText triggers a reflow to ensure up-to-date computed styles. (Reflows can be computationally expensive, and thus should be avoided when possible.)
  - Unlike textContent, altering innerText in Internet Explorer (version 11 and below) removes child nodes from the element and permanently destroys all descendant text nodes. It is impossible to insert the nodes again into any other element or into the same element after doing so.
- **innerHTML**
  - Element.innerHTML returns HTML, as its name indicates. Sometimes people use innerHTML to retrieve or write text inside an element, but textContent has better performance because its value is not parsed as HTML.
  - Moreover, using textContent can prevent XSS attacks.
  - innerHTML parses content as HTML, so it takes longer.

outerHTML

- The outerHTML attribute of the element DOM interface gets the serialized HTML fragment describing the element including its descendants. It can be set to replace the element with nodes parsed from the given string.
- innerHTML vs. outerHTML :
  - outerHTML targets self, so it also destroy/replaces the container (self) in the process when being replaced. innerHTML replaces all the children and does not destroy the container (self). If we have an event handler added to an element which we use innerHTML on, the event handler will not be destroyed. It will continue to work, since the element is still there
- **Example**

```
<h1>outerHTML</h1>
<button onclick="myFunction()">Change Header</button>

<script>
function myFunction() {
var x = document.getElementsByTagName("h1")[0];
x.outerHTML = "<h3>Change the entire h1 element and it's content!</h3>";
}
</script>
```

Source: https://www.w3schools.com/jsref/prop_html_outerhtml.asp

Focus and Blur Methods

- The blur() method is used to remove focus from an element.
- Use the focus() method to give focus to an element.
- **Example**

```
<input type="text" id="myText" value="A text field">
<button type="button" onclick="getFocus()">Get focus</button>
<button type="button" onclick="loseFocus()">Lose focus</button>

<script>
function getFocus() {
  document.getElementById("myText").focus();
}

function loseFocus() {
  document.getElementById("myText").blur();
}
</script>
```

Time Methods

- **setInterval**
  - setInterval() method calls a function or evaluates an expression at specified intervals (in milliseconds).
  - setInterval() method will continue calling the function until clearInterval() is called, or the window is closed.
  - The ID value returned by setInterval() is used as the parameter for the clearInterval() method.

  Example: setInterval(function(){ alert("Hello"); }, 3000); // *Alerts Hello every 3 seconds*
- **clearInterval**
  - The clearInterval() method clears a timer set with the setInterval() method.
  - The ID value returned by setInterval() is used as the parameter for the clearInterval() method.
- **setTimeout**
  - setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds

Class Exercise | Discussion

- Create a clock
- Raise an alert with message after 5 seconds

Cookies

- What are **Cookies**?
  - Cookies are data, stored in small text files, on your computer.
  - When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.
  - Cookies were invented to solve the problem "how to remember information about the user":
  - When a user visits a web page, his/her name can be stored in a cookie.
  - Next time the user visits the page, the cookie "remembers" his/her name.

Cookie with JavaScript

- JavaScript can create, read, and delete cookies with the document.cookie property.
- With JavaScript, a cookie can be created like this:

  document.cookie = "username=John Doe";

- We can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:

  document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

- With a path parameter, we can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

  document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

Source: https://www.w3schools.com/js/js_cookies.asp

JavaScript Cookie | Examples

- Function to Set a Cookie

```
function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

- Function to Get a Cookie

```
function getCookie(cname) {
  var name = cname + "=";
  var decodedCookie = decodeURIComponent(document.cookie);
  var ca = decodedCookie.split(';');
  for(var i = 0; i <ca.length; i++) {
    var c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

Source: https://www.w3schools.com/js/js_cookies.asp

Read and Write File

- Files can be read and written by using java script functions: fopen(),fread() and fwrite().
- The function fopen() takes two parameters – 1. Path and 2. Mode (0 for reading and 3 for writing). The fopen() function returns -1, if the file is successfully opened.

  file=fopen(getScriptPath(),0);
- The function fread() is used for reading the file content.

  str = fread(file,flength(file) ;
- The function fwrite() is used to write the contents to the file.

  file = fopen("c:\MyFile.txt", 3);// opens the file for writing

  fwrite(file, str);// str is the content that is to be written into the file.

JavaScript Animations

- JavaScript animations can handle things that CSS can't.
- For instance, moving along a complex path, with a timing function different from Bezier curves, or an animation on a canvas.
- An animation can be implemented as a sequence of frames – usually small changes to HTML/CSS properties.
- For instance, changing style.left from 0px to 100px moves the element. And if we increase it in setInterval, changing by 2px with a tiny delay, like 50 times per second, then it looks smooth. That's the same principle as in the cinema: 24 frames per second is enough to make it look smooth.

Class Exercises | Discussion

#1

- Create an Animation Using JavaScript

#2

- Create a loader in the middle of the page and show "page content" after 5 seconds

It is time for a Knowledge Check!

Sample Questions

#1 innerHTML content is refreshed every time and thus is_____.

A. faster
B. slower
C. better
D. Heavy

#2 Consider the Javascript code to the right.
How many elements adds?

A. 1
B. 2
C. none
D. Has a syntax error

```
<script type="text/javascript">
    function addNode() { var newP = document.createElement("p");
    var textNode = document.createTextNode("some Text");
    newP.appendChild(textNode);
    document.getElementById("firstP").appendChild(newP); }
</script>
```

#3 What is the correct JavaScript syntax to change the content of the HTML element below?

This is a demonstration.

A. document.getElementById("demo").innerHTML = "Hello World!";

B. document.getElementByName("p").innerHTML = "Hello World!";

C. #demo.innerHTML = "Hello World!";

D. document.getElement("p").innerHTML = "Hello World!";

Sample Questions — Answers

#1 innerHTML content is refreshed every time and thus is_____.

A. faster
B. slower
C. better
D. Heavy

#2 Consider the Javascript code to the right.
How many elements adds?

A. 1
B. 2
C. none
D. Has a syntax error

```
<script type="text/javascript">
    function addNode() { var newP = document.createElement("p");
    var textNode = document.createTextNode("some Text");
    newP.appendChild(textNode);
    document.getElementById("firstP").appendChild(newP); }
</script>
```

#3 What is the correct JavaScript syntax to change the content of the HTML element below?

This is a demonstration.

A. document.getElementById("demo").innerHTML = "Hello World!";

B. document.getElementByName("p").innerHTML = "Hello World!";

C. #demo.innerHTML = "Hello World!";

D. document.getElement("p").innerHTML = "Hello World!";

Class Exercise | Discussion

- Consider the following object

  employee={first:"John", last:"Doe", department:"Accounts"};

- Use a for/in to loop through the properties of an object

Any Questions?

References | Further Reading

- https://www.w3schools.com/js/
- https://www.w3schools.com/xml/default.asp
- https://www.w3schools.com/xml/xsl_intro.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling
- https://www.tutorialspoint.com/javascript/javascript_events.htm
- https://www.w3schools.com/js/js_events.asp
- https://www.w3schools.com/js/js_json_syntax.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects
- https://docs.oracle.com/cd/E19957-01/816-6409-10/obj2.htm
- https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing/JavaScript
- Randy Connolly and Ricardo Hoar: Fundamentals of Web Development
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch#Nested_try-blocks
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch
- https://developer.mozilla.org/en-US/docs/Web/Events
- http://www.javascriptkit.com/jsref/events.shtml
- https://www.w3schools.com/js/js_htmldom_events.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

Self Study

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch#Nested_try-blocks
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch
- https://developer.mozilla.org/en-US/docs/Web/Events
- http://www.javascriptkit.com/jsref/events.shtml
- https://www.w3schools.com/js/js_htmldom_events.asp
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

PeopleCert Values Your Feedback

Like the course?
Have something to say?

Send us your comments at:
academic@peoplecert.org

Thank You!

For latest news and updates follow us.

linkedin.com/company/peoplecert-group/
twitter.com/peoplecert
youtube.com/channel/UCLBidKZS9Xk08f_PM5Edtfg
facebook.com/peoplecert.org