DUMPSARENA

CIW JavaScript Specialist

CIW 1D0-735

Version Demo

Total Demo Questions: 7

Total Premium Questions: 78

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, JavaScript Essentials	6
Topic 2, JavaScript Programming Concepts	38
Topic 3, Document Object Model	7
Topic 4, JavaScript Events	14
Topic 5, HTML5 APIs	4
Topic 6, JavaScript Security	5
Topic 7, JavaScript Libraries and Frameworks	4
Total	78

QUESTION NO: 1

Consider the following code:



Which of the following is true based on the above code?

A. The function sofaPic will be called upon when sofaPic image loads

B. The function sofaPic is called when the image fails to load

C. The function sofaPic is called when the mouse pointer moves off the image

D. The function sofaPic is called when the mouse pointer moves over the image

ANSWER: C

Explanation:

The function sofaPic is called when the mouse pointer moves off the image. The image element uses the `onmouseout` inline event-handler attribute, which is associated with the mouse-out event. That event occurs when the pointing device leaves the element's active area. The handler shown is intended to invoke `sofaPic(this)`; in that call, `this` refers to the image element that raised the event. This pattern is commonly used for image rollovers or for restoring an image after a pointer leaves it. Inline event-handler attributes execute their JavaScript when their corresponding DOM event is dispatched, so the function is triggered by leaving the image rather than by the image resource loading or failing to load. The browser's mouse event model defines `mouseout` as an event fired when the cursor is moved out of an element, and JavaScript event handlers can be registered either through such attributes or with event-listener APIs. See the MDN documentation for the mouseout event and the MDN reference for HTML event-handler attributes.

QUESTION NO: 2

Derrick is building a Web form to collect user data for a retail Web site. He will be collecting the users name e-mail address phone number, physical address and credit card information. What should he do to protect his users, when collecting this data?

A. Notify users how their data is being stored and used on the Web site.

B. Use signed scripts to keep the user data secure on the server.

C. Advise users to install anti-virus software to keep their data secure.

D. Advise users to disable JavaScript in their browsers, to keep information secure.

ANSWER: A

Explanation:

"Notify users how their data is being stored and used on the Web site." is the correct choice because the form collects personally identifiable information and highly sensitive payment-card information. Before users submit that information, the site should present a clear, accessible privacy notice explaining what data is collected, the purpose for collecting it, how it will be used, whether it is shared with third parties, how long it is retained, and how users can exercise applicable privacy choices. Clear notice supports informed user decisions and is a foundational fair-information and privacy practice for Web applications.

For a retail site, this notice should be available at or near the point of collection rather than being difficult to locate elsewhere on the site. The Federal Trade Commission advises businesses to be transparent about their data practices and to provide consumers meaningful information about how their personal data is handled. The National Institute of Standards and Technology similarly identifies transparency as an important privacy-engineering objective. In a production implementation, disclosure should accompany appropriate technical safeguards, such as encrypted transport and secure payment

processing, but communicating the site's collection and use practices is the action directly addressed by this question. See the FTC privacy and security guidance and NIST Privacy Framework.

QUESTION NO: 3

Consider the following code:

Charies wants to write code to execute the changeOption function after the user chooses an option in the select menu. What change to the code should he make?

A. Change line 7 to

B. Change line 2 to function changeOption (onchange){

C. Change line 6 to

D. Change line 6 to

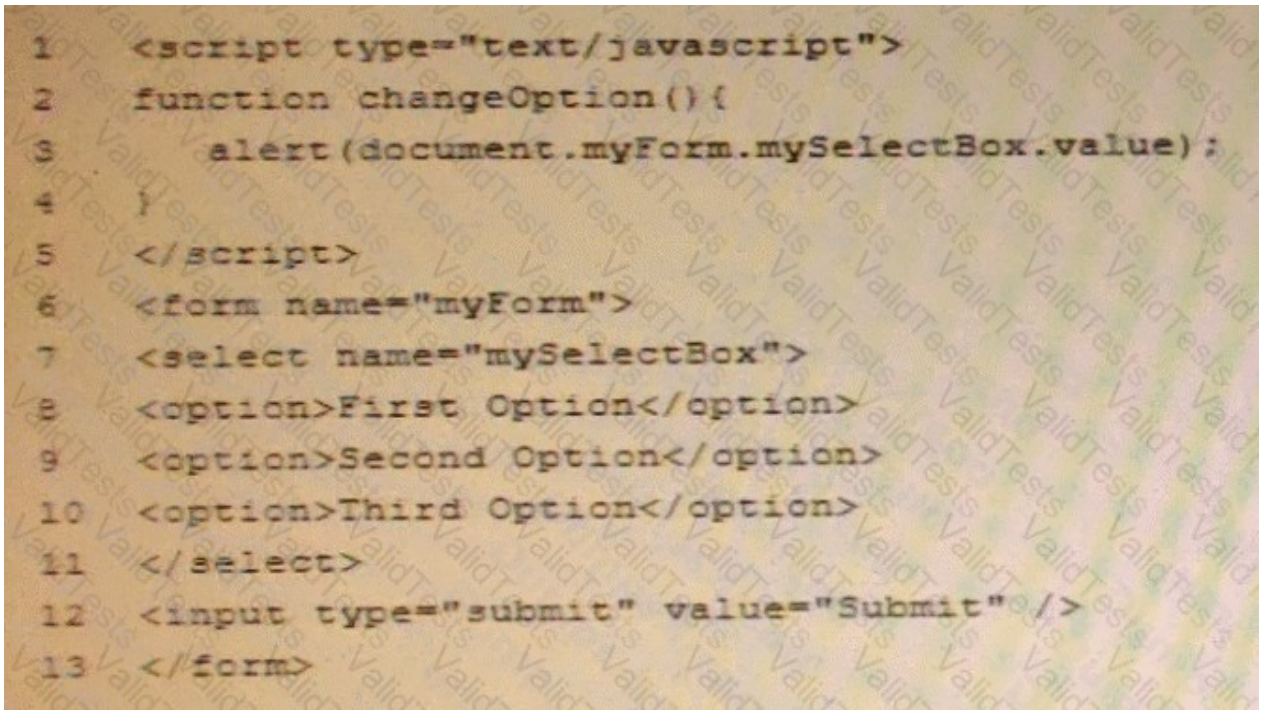
ANSWER: A

Explanation:

Change line 7 to `<select name="mySelectBox" onchange="changeOption()">`. The `change` event is the appropriate event for a `<select>` control when code must run after the user changes its selected value. Assigning `changeOption()` to the element's `onchange` event-handler attribute causes the browser to invoke the function when the selection is committed. The parentheses are important because the event-handler attribute contains JavaScript code to execute; writing `changeOption()` performs the function call when the event occurs. This places the behavior on the menu that actually receives the user's choice, matching the stated requirement. The HTML specification defines `change` as an event associated with form controls, including select elements, and MDN documents that it fires for a `<select>` when the user selects an option. See MDN's change event reference and the HTML Living Standard's select element specification.

QUESTION NO: 4

Consider the following code:



```
1  <script type="text/javascript">
2  function changeOption(){
3      alert(document.myForm.mySelectBox.value);
4  }
5  </script>
6  <form name="myForm">
7  <select name="mySelectBox">
8  <option>First Option</option>
9  <option>Second Option</option>
10 <option>Third Option</option>
11 </select>
12 <input type="submit" value="Submit" />
13 </form>
```

Charies wants to write code to execute the changeOption function after the user chooses an option in the select menu. What change to the code should he make?

A. It provides a limited set of database operations such as retrieving data and displaying it back to the originating Web page.

B. Change line 2 to function changeOption (onchange){

C. Change line 6 to

D. Change line 6 to

E. Change line 6 to add onchange="changeOption()" to the element.

ANSWER: E

Explanation:

The correct change is to attach the `onchange` event handler directly to the `<select>` element: `<select ... onchange="changeOption()">`. The change event is associated with a select control and occurs when the user makes a selection that changes the control's value. When that event fires, the browser evaluates the handler and calls `changeOption()`, allowing the function to respond immediately to the newly chosen menu item.

Inline event attributes are a valid way to connect a simple event handler in instructional examples. The handler must be written as an attribute of the element that receives the interaction, and the function invocation includes parentheses so that the function is executed when the event occurs. In production code, JavaScript commonly uses `addEventListener("change", changeOption)` instead, but adding `onchange="changeOption()"` to the select menu is the direct code correction requested here. MDN documents the `change` event and notes that it fires when a `<select>` element's value is committed by the user: HTMLElement: change event. The select element's behavior is also described in the HTML reference: <select>: The HTML Select element.

QUESTION NO: 5

Janice needs to create a pop-up window that will open a Web document in a new browser window. Which code statement should she use?

A)

```
window.open("https://www.CIWcertified.com","CIW Window");
```

B)

```
window.open("https://www.CIWcertified.com","CIWWindow");
```

C)

```
document.open("https://www.CIWcertified.com","CIW Window");
```

D)

```
document.open("https://www.CIWcertified.com","CIWWindow");
```

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: C

Explanation:

The correct statement is the one using the JavaScript `window.open()` method to open the specified Web document in another browsing context. This method is called from the `window` object and accepts a URL as its first argument. It can also accept a target window name and an optional feature string that controls characteristics such as the new window's dimensions, position, and browser chrome. A call such as `window.open("document.html", "_blank")` instructs the browser to load the named document in a newly created browsing context. This is the standard JavaScript API historically used to create pop-up windows and remains the appropriate method when a script must programmatically open another document. In modern browsers, the call should normally occur directly as part of a user action, such as a click event, because popup-blocking protections may prevent windows opened asynchronously or without user activation. The returned value is a reference to the opened window, which can be used by the originating script when same-origin rules permit access. See the MDN Window.open() reference and the HTML Standard definition of window.open().

QUESTION NO: 6

Consider the following code:

```
var name = "Jacob";
name.toUpperCase();
document.write(name);
```

What change should be made to ensure that it correctly displays the value of name in all uppercase letters?

A. Line 2 should be changed to name = name . toUpperCase () ;

B. Line 2 should be changed to name .prototype . toUpperCase () ;

C. Line 3 should be changed to document. write (NAME) :

D. Line 1 should be changed to var name = new sting("Jaccb");

ANSWER: A

Explanation:

The change `name = name.toUpperCase();` correctly ensures that the displayed value is uppercase because `String.prototype.toUpperCase()` returns a new string containing the uppercase version of the original text. JavaScript strings are immutable, so calling `name.toUpperCase()` by itself does not alter the string currently stored in `name`. Assigning the returned string back to `name` updates the variable before it is passed to `document.write()`.

For example, if `name` initially contains `"Jaccb"`, evaluating `name.toUpperCase()` produces `"JACCB"`. The assignment stores that result, so the subsequent output statement displays `JACCB`. This follows the standard behavior of JavaScript string methods: transformations such as case conversion produce a result rather than modifying the existing string value in place. See the MDN String.prototype.toUpperCase() reference and the MDN JavaScript String reference.

QUESTION NO: 7

Loni needs to create a scriptable button in her Web form that does not submit the form but instead, calls a custom function What HTML code should she use?

A)

```
<form method="function" action="generateCAPTCHA()">
```

B)

```
<input type="submit" value="Refresh" onclick="generateCAPTCHA()" />
```

C)

```
<input type="hidden" value="Refresh" onclick="generateCAPTCHA()" />
```

D)

```
<button value="Refresh" onclick="generateCAPTCHA()"></button>
```

A. Option A

B. Option B

C. [Click Me]

D. Option D

ANSWER: C

Explanation:

`<input type="button" value="Click Me" onclick="customFunction()">` is the appropriate pattern because an `input` element with `type="button"` creates a general-purpose clickable button rather than a form-submission control. Its default behavior is not to submit the containing form, so it is suitable when JavaScript must perform a custom action instead. The `onclick` event-handler attribute associates the click event with the specified JavaScript function; when the user activates the button, the browser evaluates `customFunction()`. The function can then perform any required client-side task, such as validating fields, changing page content, calculating values, or sending data through a custom asynchronous request.

The HTML standard defines the Button state of an `input` element as a button that has no default behavior, which is the key distinction needed for this requirement. Inline event-handler attributes such as `onclick` are also supported for associating an event with JavaScript code, although unobtrusive event listeners are often preferred in production applications. See the HTML specification's Button state documentation and MDN's click event reference.