DUMPSBOSS.

CIW JavaScript Fundamentals exam

CIW 1D0-435

Version Demo

Total Demo Questions: 26

Total Premium Questions: 283

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Introduction to JavaScript	32
Topic 2, JavaScript Fundamentals	61
Topic 3, JavaScript Objects	41
Topic 4, JavaScript Language Objects	34
Topic 5, JavaScript with HTML	11
Topic 6, JavaScript Event Handlers	12
Topic 7, JavaScript Functions	11
Topic 8, Custom JavaScript Objects	30
Topic 9, JavaScript Security	32
Topic 10, JavaScript and HTML Forms	18
Topic 11, Dynamic HTML	1
Total	283

QUESTION NO: 1

Which of the following statements about cookie security attributes are true?

A. The Secure attribute restricts a cookie to HTTPS requests.

B. The HttpOnly attribute prevents JavaScript from reading the cookie through document.cookie.

C. The SameSite attribute helps control cross-site cookie sending.

D. The HttpOnly attribute makes a cookie available only to JavaScript.

E. The Secure attribute prevents users from deleting a cookie.

ANSWER: A B C

Explanation:

The `Secure` attribute directs the browser to send a cookie only with HTTPS requests, helping protect the cookie from exposure over unencrypted HTTP connections. The `HttpOnly` attribute prevents JavaScript running in the page from reading the cookie through `document.cookie`. The `SameSite` attribute controls whether a browser sends the cookie with cross-site requests.

None of these attributes makes a cookie permanently undeletable. Users can generally clear browser cookies, and an application should handle the possibility that a cookie is missing. Also, `HttpOnly` cookies are intentionally unavailable to client-side JavaScript. See the MDN Set-Cookie reference.

QUESTION NO: 2

The Microsoft implementation of the Netscape JavaScript language is called

_____.

A. VBScript

B. Java

C. ECMAScript

D. JScript

ANSWER: D

Explanation:

JScript is correct because it was Microsoft's implementation of the scripting language originally developed by Netscape and known as JavaScript. Microsoft introduced JScript for Internet Explorer in the mid-1990s, using a distinct product name while providing a language designed to be compatible with JavaScript as it was standardized and evolved. In the browser era relevant to this terminology, web developers could encounter JScript in Microsoft Internet Explorer and JavaScript in Netscape Navigator, even though the languages shared core syntax, objects, and scripting purposes.

The naming distinction is historical but important for CIW terminology: JScript identifies Microsoft's implementation, whereas JavaScript is the broader language name associated with Netscape's original implementation and the subsequently standardized language family. Microsoft documentation describes JScript as an interpreted, object-based scripting language, and its legacy language reference records its use in Microsoft web technologies. ECMAScript is the standards specification on which modern JavaScript implementations are based, rather than the product name Microsoft used for its implementation. See Microsoft's JScript language reference and the ECMA International ECMAScript standard.

QUESTION NO: 3

JavaScript is most similar to which of the following languages?

A. ECMAScript

B. VBScript

C. NetScript

D. MicroScript

ANSWER: A

Explanation:

ECMAScript is correct because JavaScript is the best-known implementation of the ECMAScript language standard. ECMAScript defines the core language features JavaScript developers use, including its syntax, variables, data types, objects, functions, control flow, classes, modules, and built-in behaviors. In practical terms, modern JavaScript engines implement ECMAScript specifications, so the two share the same fundamental language model and are closely associated in both browser and server-side development.

The relationship is important because “JavaScript” is commonly used as the everyday name for the programming language, whereas “ECMAScript” is the standardized specification that guides interoperable implementations. Features identified by edition names such as ECMAScript 2015 or later ECMAScript releases describe capabilities that JavaScript environments may support. The ECMA International standard identifies ECMAScript as the specification for a general-purpose programming language, and MDN describes JavaScript as a language based on the ECMAScript standard. Therefore, among the listed language names, ECMAScript is the accurate match for JavaScript’s language foundation and behavior.

References: ECMA-262 ECMAScript Language Specification; MDN JavaScript Guide: Introduction.

QUESTION NO: 4

Which of the following are appropriate uses of the `return` statement in a JavaScript function?

A. To send a computed value back to the calling code.

B. To end execution of the current function.

C. To display a value automatically in the browser window.

D. To declare a global variable.

ANSWER: A B

Explanation:

A `return` statement can provide a value to the code that called the function. It also ends execution of the current function immediately, so statements following the executed `return` in that function body are not run.

A `return` statement does not display a value automatically, and it is not required in every function. A function that completes without an explicit `return` has the result `undefined`. See the MDN return statement reference.

QUESTION NO: 5

JavaScript is an object _____ programming language.

A. oriented

B. targeted

C. related

D. based

ANSWER: D

Explanation:

The correct completion is **"based"**, producing the phrase "object-based programming language." In the terminology commonly used by CIW JavaScript training materials, JavaScript is characterized as object-based because it provides and uses objects, including built-in objects such as `Array`, `Date`, `Math`, and `String`, and because developers can create their own objects. Objects group related data, stored as properties, with behavior, stored as methods. This model lets a script represent and manipulate meaningful entities such as a customer, product, page element, or form submission as a single value with associated information and operations.

JavaScript's object system is prototype-based: objects can inherit behavior through prototype links rather than relying exclusively on class definitions. Modern JavaScript also provides class syntax, but that syntax is built on the same prototype mechanism. Therefore, "object-based" accurately matches the wording expected by the question while recognizing that objects are a foundational part of JavaScript programming. MDN explains JavaScript objects and their properties and methods in its Working with objects guide, and documents the prototype inheritance model in its Inheritance and the prototype chain reference.

QUESTION NO: 6

Which property of the location object refers to the hostname:port solution of the URL?

A. host

B. pathname

C. hash

D. protocol

ANSWER: A

Explanation:

The `host` property is correct because the browser's `Location` object uses it to expose the host portion of the current URL: the hostname together with the port number, when a port is present. For example, if the address is `https://www.example.com:8080/products?id=4`, `location.host` returns `www.example.com:8080`. If the URL does not explicitly include a port, it returns just the hostname, such as `www.example.com`. This makes `host` useful when JavaScript needs the network location serving the current document without the scheme, path, query string, or fragment identifier. It is a read-only string property of the standard browser `Location` interface, available through `window.location` (or simply `location` in a browser context). The distinction matters when constructing URLs, checking the origin-related parts of an address, or displaying the server endpoint to a user. MDN documents `Location.host` as the hostname followed by the port, if one exists: MDN Location.host. The broader URL component model is also defined by the WHATWG URL Standard: WHATWG URL Standard.

QUESTION NO: 7

A _____ is an organized block of code that handles actions generated by user events.

A. function

B. method

C. statement

D. object

ANSWER: A B

Explanation:

A **function** is an organized, reusable block of JavaScript code that can be invoked when a user-generated event occurs. For example, a function can run after a visitor selects a button, changes a form control, moves the pointer, or submits a form. In event-driven JavaScript, the function registered to respond to an event is commonly called an event-handler function. It receives event information when necessary and contains the instructions that implement the application's response. The MDN JavaScript guide describes functions as reusable code blocks that can be called to perform a task, while its event documentation explains that event handlers are functions executed in response to events. See MDN: Functions and MDN: addEventListener().

A **method** is also appropriate when the event-handling function belongs to an object or class. In JavaScript, a method is a function stored as an object property; therefore, an object-oriented program can implement an event response through a method, such as an object's click-handling method. Whether described as a standalone function or as a method of an object, the relevant code is an organized callable block used to handle user actions.

QUESTION NO: 8

To call the alert() method, you need only place a simple line of code, called a

_____, in a script block somewhere in your document.

A. statement

B. method

C. object

D. command

ANSWER: A

Explanation:

The correct term is **statement**. In JavaScript, a statement is a complete instruction that tells the JavaScript engine to perform an action. Calling `alert()` in a script block, such as `alert("Welcome");`, is a complete executable instruction and is commonly described in introductory JavaScript materials as a statement. When the browser reaches that statement while processing the script, it invokes the `alert()` function and displays a dialog containing the supplied message.

JavaScript programs consist of statements, which can include declarations, conditional instructions, loops, and expression statements. A function call written on its own line is an expression statement: the call is evaluated for its effect rather than to produce a value that is subsequently used. The semicolon is conventionally included to terminate the statement, although JavaScript can insert semicolons automatically in many situations. This terminology accurately distinguishes the complete line of executable code from the callable feature, `alert()`, itself. See the MDN JavaScript statements reference and the MDN alert() reference.

QUESTION NO: 9

Which of the following statements about the JavaScript `Math` object are true?

A. `Math.random()` returns a number from 0 up to, but not including, 1.

B. `Math.round()` rounds a number to the nearest integer.

C. `new Math()` is required before using Math methods.

D. `Math.random()` returns only whole numbers.

ANSWER: A B

Explanation:

`Math.random()` returns a pseudo-random number greater than or equal to `0` and less than `1`. `Math.round()` returns a number rounded to the nearest integer. These are methods supplied by the built-in `Math` object.

The `Math` object is not a constructor used with `new`, and `Math.random()` does not return only whole numbers. See the MDN Math.random() reference and the MDN Math.round() reference.

QUESTION NO: 10

Which one of the following choices describes a type of information that a cookie can readily provide?

A. The current licensing status of the software on the user's hard drive

B. The addresses of all e-mail recipients with which the user has corresponded

C. The current buying habits of the user

D. A history of the sites that the user has visited

ANSWER: C

Explanation:

The current buying habits of the user is the correct choice. A cookie is a small piece of data that a website asks the browser to store and return with later requests to that same site. The cookie commonly contains an identifier rather than a complete personal profile. When the browser sends that identifier back, the site can associate the request with information held in its own server-side records, such as products viewed, items placed in a shopping cart, prior purchases, and preferences inferred from browsing activity on that site. This allows an e-commerce site to recognize a returning visitor and tailor recommendations, offers, cart contents, or advertising based on current purchasing interests and behavior.

Cookies therefore support continuity and personalization across visits: the browser preserves the cookie value, and the website uses that value to retrieve or update the related customer/session record. This is a common use of persistent and first-party cookies in online retail. The cookie does not need to contain every detail of the buying record itself; its stored identifier can readily enable the website to access that information. MDN describes cookies as data stored by the browser and sent back to the server with subsequent requests: MDN Web Docs: HTTP cookies. The Federal Trade Commission also discusses how online tracking can be used to collect information about consumers' browsing and purchasing-related interests: FTC Advertising and Marketing.

QUESTION NO: 11

document.cookie = "name = value"; or document.cookie = "name = value;expi

es = date"; are both proper syntax for assigning cookies in JavaScript.

A. TRUE

B. FALSE

ANSWER: A

Explanation:

`TRUE` is correct because JavaScript assigns browser cookies by setting the `document.cookie` property to a cookie string. The basic assignment contains a cookie name and value, such as `document.cookie = "name=value"`. A cookie assignment can also include attributes after the name/value pair, separated with semicolons. In particular, the `expires` attribute supplies an expiration date, allowing the cookie to persist beyond the current browser session. Thus, the intended forms represent the two common cases: creating a cookie with the browser's default lifetime and creating one with an explicit expiration date. In practical code, write the name/value pair and attribute in their conventional compact form, for example `document.cookie = "name=value; expires=Wed, 21 Oct 2026 07:28:00 GMT"`. The date must be a valid HTTP-date value, conventionally expressed in GMT/UTC, so the browser can determine when to remove the cookie.

Assigning `document.cookie` updates a single cookie rather than replacing every cookie available to the document. See the MDN Document.cookie reference and the cookie lifetime guidance in MDN's HTTP cookies guide.

QUESTION NO: 12

Which of the following statements about HTML form controls and JavaScript are true?

A. A text input control uses its value property for its current text.

B. A checkbox uses its checked property to indicate selection.

C. A form can receive a submit event.

D. A checkbox uses its value property to indicate whether it is checked.

E. A reset control submits the form to the server.

ANSWER: A B C

Explanation:

A text input control exposes its current text through its `value` property, so JavaScript can read or assign that value. A checkbox exposes a Boolean `checked` property that indicates whether it is selected. The `submit` event can be handled on a form to validate or process user input before the browser submits the form.

The `value` property does not indicate whether a checkbox is selected; that role belongs to `checked`. Also, a reset control restores form controls to their initial values and does not submit the form. See the MDN HTMLInputElement reference and the MDN submit event reference.

QUESTION NO: 13

Which of the following statements about JavaScript object properties are true?

A. Dot notation can access a property whose name is a valid identifier.

B. Bracket notation can access a property by using a string property name.

C. An object property can store a function.

D. Object properties can contain only string values.

E. Properties can be added only when an object is first created.

ANSWER: A B C

Explanation:

An object property associates a name with a value. A property can be accessed with dot notation when its name is a valid identifier, such as `customer.name`. Bracket notation can also access a property, such as `customer["name"]`, and is especially useful when the property name is stored in a variable or contains characters not suitable for dot notation.

Properties are not limited to string values; they can hold numbers, arrays, objects, functions, and other JavaScript values. An object can also have properties added after it has been created. See the MDN Working with objects guide.

QUESTION NO: 14

The form element _____ is used for a single-line text field used for data entry.

A. text

B. data box

C. data

D. textarea

ANSWER: A

Explanation:

The correct answer is `text`. In HTML forms, a single-line text-entry control is created with an `<input>` element whose `type` attribute is set to `text`, as in `<input type="text" name="username">`. This control accepts ordinary textual user input on one line, making it appropriate for values such as names, cities, search terms, or short identifiers. The browser renders the control as a standard text box, and the value entered by the user is available for form submission under the input control's `name`. If a user enters more text than can be displayed within the control's visible width, the field scrolls horizontally rather than becoming a multi-line editing area. The `type="text"` value is also the default type for an `<input>` element when no `type` is specified, though explicitly declaring it improves clarity. The HTML specification defines the Text state as a single-line plain-text edit control, and MDN documents its expected behavior and relevant attributes, including `maxlength`, `placeholder`, and `required`. See the WHATWG HTML specification and MDN's input type="text" reference.

QUESTION NO: 15

_____ are the actions that the object property can be made to perform, such as a calculation, an onscreen move, or the writing of text in a window.

A. Behaviors

B. Values

C. Methods

D. Properties

ANSWER: C

Explanation:

Methods are the actions or operations associated with an object. In JavaScript's object-oriented model, an object combines data with behavior: its properties hold or describe data, while its methods provide functionality that can be invoked to make the object do something. A method is typically a function stored as an object property and is called using dot notation followed by parentheses, such as `window.alert("Hello")` or `document.write("Text")`. Calling a method can perform a calculation, change an object's state, cause an element to move or otherwise update on screen, or write output to a document. The examples in the question describe exactly this operational role, so Methods is the correct completion. JavaScript's language reference describes methods as function-valued properties that are called as members of an object, and browser APIs expose many object methods for interacting with a web page and browser window. See the MDN JavaScript method definitions reference and the MDN Document.write() reference.

QUESTION NO: 16

_____ and _____ are interchangeable in JavaScript.

A. Methods and functions

B. Ojects and functions

C. Properties and attributes

D. Methods and properties

ANSWER: A

Explanation:

Methods and functions are interchangeable in the practical sense intended here because a JavaScript method is a function associated with an object. A function is a reusable block of code that can be declared independently, assigned to a variable, passed as a value, or stored in an object property. When the function is accessed through an object property and invoked in that object's context, it is called a method. For example, an object can contain `greet: function() { return "Hello"; }`; calling `object.greet()` invokes that function as the object's method. JavaScript treats functions as first-class values, so the same function can be assigned to an object property, removed, or assigned elsewhere. The distinction is therefore based chiefly on how the function is associated with and called from an object, rather than on a separate underlying JavaScript construct. This terminology is foundational when working with built-in objects, DOM APIs, and custom object behavior. See the MDN documentation on JavaScript functions and method definitions.

QUESTION NO: 17

Consider the following script:

< SCRIPT >

< !--

var name;

name=prompt(" What is your name? " , " ");

alert(" Hello, " + name + " . ");

// -- >

< /SCRIPT >

What does this script do in terms of the use of the variable " name " ?

A. The variable value is assigned via the prompt method, and then displayed via the alert method.

B. The variable value is assigned via the prompt method.

C. Nothing. This script would result in an error.

D. The variable value is assigned via the alert method.

ANSWER: A

Explanation:

The variable value is assigned via the prompt method, and then displayed via the alert method. The declaration `var name;` creates the variable before it is used. Next, `prompt("What is your name?", "")` displays a dialog box asking the user for a name. The `prompt()` method returns the text entered by the user as a string, and the assignment operator stores that returned value in `name`. If the user enters `Jordan`, for example, `name` subsequently contains the string `"Jordan"`. The following `alert()` call constructs a message by concatenating the literal text `"Hello, "`, the current value of `name`, and a period. It then displays the completed message in an alert dialog, such as `Hello, Jordan.`. Thus, the variable serves as the link between user input and the displayed greeting: it receives the value returned from the prompt and supplies that value when the alert message is evaluated. MDN documents that `window.prompt()` returns a string or `null`, while `window.alert()` displays a dialog: MDN prompt() reference and MDN alert() reference.

QUESTION NO: 18

Which of the following are valid reasons to use an external JavaScript file?

A. It allows the same script to be used by more than one Web page.

B. It can be cached by the browser.

C. It prevents JavaScript from modifying a document.

D. It eliminates the need for a browser to execute the script.

ANSWER: A B

Explanation:

An external script can be reused by multiple HTML documents, which avoids duplicating the same JavaScript source in every page. Browsers can also cache an external script resource, reducing the need to download it again on later page loads when the cached version remains valid.

An external script file does not prevent JavaScript from changing page content, and it does not remove the need for the browser to execute the script. External files are loaded with the `src` attribute of a `<script>` element. See the MDN script element reference.

QUESTION NO: 19

Which property of the document object represents the color in which text is to be displayed?

A. txColor

B. bgColor

C. fgColor

D. txtColor

ANSWER: C

Explanation:

`fgColor` is the legacy `document` property that represents the document foreground color—the color used for normal text displayed in the page. The “fg” portion denotes “foreground,” distinguishing the text color from page-background settings. In classic HTML and early browser DOM implementations, this property corresponded to the `TEXT` attribute of the `<body>` element, such as `<body text="blue">`. Reading `document.fgColor` returns the configured foreground color, and assigning a valid color value changes that legacy document-level text-color setting.

This is primarily a historical DOM feature, which is why it can appear in JavaScript fundamentals material that covers traditional `document` object properties. Modern web development normally applies text color through CSS, for example with the `color` property, because CSS provides clearer separation of structure, presentation, and behavior. Nevertheless, for the specific document-object property requested, `fgColor` is the correct name. MDN documents `Document.fgColor` as the foreground color of the current document and notes its deprecated status: MDN Document.fgColor. The related HTML body text-color attribute is also described by MDN’s body element reference.

QUESTION NO: 20

Chiyo wants to open a new window with JavaScript, but she repeatedly receives an error message. She reviews her code, as shown: open ("http://www.prolific.com", "Pro Window" , "toolbar = 1 ,location = 1 ,menubar = 1 , scrollbars = 1 ,status = 1 ,resizable = 1"); Which one of the following choices shows how Chiyo should rewrite this line of code so that no errors result when it is executed?

A. open("http://www.prolific.com","Pro Window","toolbar=1
,location=1,menubar=1,scrollbars=1,status=1,resizable=1 ");

B. open("http://www.prolific.com","Pro Window ","toolbar = 1,location =1,menubar = 1
,scrollbars = 1,status = 1 ,resizable = 1");

C. open("http://www.prolific.com","ProWindow ","toolbar=1,location=1,menubars=1,scrollbars=1,status=1,resizable=1 ");

D. open ("http://www.prolific.com", "ProWindow","toolbar = 1,location = 1,menubar = 1,scrollbars = 1,status = 1,resizable = 1 ");

ANSWER: D

Explanation:

`open ("http://www.prolific.com", "ProWindow","toolbar = 1,location = 1,menubar = 1,scrollbars = 1,status = 1,resizable = 1 ");` correctly follows the traditional `window.open()` calling pattern: a URL is supplied first, a window name is supplied second, and a single quoted feature string is supplied third. The window name `ProWindow` is a single, contiguous identifier, which is appropriate when naming a window for later reference. The final argument is one string containing comma-separated feature declarations, including `toolbar`, `location`, `menubar`, `scrollbars`, `status`, and `resizable`. Each feature is assigned a value of `1`, the conventional legacy syntax for requesting that the feature be enabled. Whitespace around JavaScript tokens and within the feature string does not prevent the argument from remaining a valid string; the essential requirements are the quoted arguments, comma-separated function arguments, and correctly formed comma-separated feature settings. Modern browsers can restrict or ignore window features, and commonly require `window.open()` to be invoked from a user action, but those browser policies are separate from the statement's JavaScript syntax. See MDN Web Docs: Window.open() and HTML Standard: window.open().

QUESTION NO: 21

Which of the following statements about JavaScript arrays are true?

A. `push()` adds an element to the end of an array.

B. The first element in an array has an index of 0.

C. The `length` property always contains the last array index.

D. An array can contain only values of one data type.

ANSWER: A B

Explanation:

`push()` is correct because it adds one or more elements to the end of an array and returns the new array length. Array indexes also begin with `0`, so the first element of an array named `colors` is accessed as `colors[0]`.

The `length` property does not identify the last index; it reports the number of elements in the array. Also, an array can contain values of different types, including strings, numbers, objects, and functions. See the MDN Array reference and the MDN push() reference.

QUESTION NO: 22

Which of the following are valid JavaScript variable declarations?

A. `let customerName = "Lee";`

B. `const MAX_ITEMS = 10;`

C. `var 2ndPlace = "Silver";`

D. `var _total = 0;`

E. `let function = 1;`

ANSWER: A B D

Explanation:

`let customerName = "Lee";`, `const MAX_ITEMS = 10;`, and `var _total = 0;` are valid declarations. JavaScript identifiers can begin with a letter, an underscore, or a dollar sign. The `let`, `const`, and `var` keywords each declare variables, although they differ in scope and reassignment rules.

`var 2ndPlace = "Silver";` is invalid because an identifier cannot begin with a numeric digit. `let function = 1;` is invalid because `function` is a JavaScript keyword and cannot be used as an ordinary variable identifier. See the MDN JavaScript identifier reference and the MDN let statement reference.

QUESTION NO: 23

Ann inadvertently deleted her cookies.txt file.

How will this affect her browsing experience?

A. She must re-enter her personal information wherever she has user accounts that rely on cookies.

B. She will enjoy a quicker browsing experience because the servers she visits need not confirm her user identity.

C. It depends on the type of cookies.txt file that was generated within her browser user profile; if the cache files and cookies.txt were deleted, all links will appear unvisited.

D. Her Web browser will no longer function because the cookies.txt file is necessary for browser operation.

ANSWER: A C

Explanation:

Cookies are small pieces of state that a website asks the browser to retain and return with later requests. In browsers that used a `cookies.txt` file, deleting that file removes the locally stored cookie records. Consequently, a site can no longer recognize the prior browser session or retrieve values it had stored in cookies, such as a remembered sign-in state, preferences, shopping-cart state, or previously supplied form details. Therefore, “She must re-enter her personal information wherever she has user accounts that rely on cookies.” correctly identifies the practical effect: Ann may need to sign in again and provide information that a site had previously remembered through cookies. Cookie behavior is described by MDN’s HTTP cookies guide.

“It depends on the type of cookies.txt file that was generated within her browser user profile; if the cache files and cookies.txt were deleted, all links will appear unvisited.” is also correct in the legacy browser context described. A browser’s profile data can include stored site data and browsing-related records used to present prior navigation state. When the relevant locally retained browsing data is removed along with the cookie file, the browser may no longer display prior links as visited. The browser’s visited-link presentation is based on its retained history state, as documented in MDN’s :visited reference.

QUESTION NO: 24

Which of the following statements about JavaScript function parameters are true?

A. A parameter receives a value supplied when a function is called.

B. A function can define more than one parameter.

C. A parameter must have the same name as the variable used by the caller.

D. A function call always requires the exact number of declared arguments.

ANSWER: A B

Explanation:

A function parameter is a named variable in a function definition that receives a supplied argument when the function is called. A function can define more than one parameter, allowing it to receive multiple values. For example, `function multiply(x, y) { return x * y; }` declares two parameters.

Parameters are not required to have the same names as the caller's variables. In addition, JavaScript functions may be called with fewer arguments than declared parameters; an omitted parameter normally has the value `undefined` unless a default parameter value is provided. See the MDN function parameters reference.

QUESTION NO: 25

What is the primary difference between methods and functions when working with custom

JavaScript objects?

A. There is no difference between methods and functions in JavaScript.

B. Methods work with single instances of objects, whereas functions can work on all instances of an object.

C. Functions are declared in the constructor, whereas methods are never declared in the constructor.

D. Methods are single entities, whereas functions can have more than one method.

ANSWER: B

Explanation:

“Methods work with single instances of objects, whereas functions can work on all instances of an object.” is the intended distinction in the context of custom JavaScript objects. A method is a function made available through an object and is invoked in connection with that object, such as `customer.displayName()`. When the method runs, its `this` value normally refers to the particular object instance used to make the call. This lets the method access and update that instance’s properties, so behavior is naturally tied to an individual object’s current state.

A function can instead be defined separately and used more generally. It can receive an object as an argument, or it can be assigned for reuse across objects, allowing the same logic to operate on multiple instances. In custom-object design, this distinction helps developers decide whether behavior belongs with an object’s own data as a method or should be reusable independently as a function. JavaScript documentation describes methods as object properties whose values are functions and shows their invocation using object-member syntax. See MDN: Method definitions and MDN: Working with objects.

QUESTION NO: 26

Which of the following statements about browser dialog methods are true?

A. alert() displays a message dialog.

B. confirm() returns a Boolean value.

C. prompt() can return null when the user cancels.

D. alert() writes its message as HTML into the document.

E. confirm() always returns the text entered by the user.

ANSWER: A B C

Explanation:

The `alert()` method displays a message dialog. The `confirm()` method displays a dialog with choices such as OK and Cancel, and it returns a Boolean value. The `prompt()` method can ask the user to enter text and returns the entered string or `null` if the user cancels the dialog.

These methods are methods of the browser's `window` object. They do not write HTML directly into the document, and `alert()` does not return the text displayed to the user. See the MDN Window.alert() reference, MDN Window.confirm() reference, and MDN Window.prompt() reference.