**JAVASCRIPT ----**

1. What is JavaScript?

JavaScript is a client-side as well as server side scripting language that can be inserted into HTML pages and is understood by web browsers. JavaScript is also an Object based Programming language

2. Enumerate the differences between Java and JavaScript?

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for the different intent. Java is an object - oriented programming (OOPS) or structured programming language like C++ or C whereas JavaScript is a client-side scripting language.

3. What are JavaScript Data Types?

Following are the JavaScript Data types:

* Number
* String
* Boolean
* Object
* Undefined

4. What is the use of isNaN function?

isNan function returns true if the argument is not a number otherwise it is false.

5. Between JavaScript and an ASP script, which is faster?

JavaScript is faster. JavaScript is a client-side language and thus it does not need the assistance of the web server to execute. On the other hand, ASP is a server-side language and hence is always slower than JavaScript. Javascript now is also a server side language (nodejs).

6. What is negative infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

7. Is it possible to break JavaScript Code into several lines?

Breaking within a string statement can be done by the use of a backslash, '\', at the end of the first line

Example:

document.write("This is \a program");

And if you change to a new line when not within a string statement, then javaScript ignores break in line.

Example:

var x=1, y=2,

z=

x+y;

The above code is perfectly fine, though not advisable as it hampers debugging.

8. Which company developed JavaScript?

Netscape is the software company who developed JavaScript.

9. What are undeclared and undefined variables?

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

10. Write the code for adding new elements dynamically?

<html>

<head>

<title>t1</title>

<script type="text/javascript">

function addNode() { var newP = document.createElement("p");

var textNode = document.createTextNode(" This is a new text node");

newP.appendChild(textNode); document.getElementById("firstP").appendChild(newP); }

</script> </head>

<body> <p id="firstP">firstP<p> </body>

</html>

11. What are global variables? How are these variable declared and what are the problems associated with using them?

Global variables are those that are available throughout the length of the code, that is, these have no scope. The var keyword is used to declare a local variable or object. If the var keyword is omitted, a global variable is declared.

Example:

// Declare a global globalVariable = "Test";

The problems that are faced by using global variables are the clash of variable names of local and global scope. Also, it is difficult to debug and test the code that relies on global variables.

12. What is a prompt box?

A prompt box is a box which allows the user to enter input by providing a text box. Label and box will be provided to enter the text or number.

13. What is 'this' keyword in JavaScript?

'This' keyword refers to the object from where it was called.

14. Explain the working of timers in JavaScript? Also elucidate the drawbacks of using the timer, if any?

Timers are used to execute a piece of code at a set time or also to repeat the code in a given interval of time. This is done by using the functions setTimeout, setInterval and clearInterval.

The setTimeout(function, delay) function is used to start a timer that calls a particular function after the mentioned delay. The setInterval(function, delay) function is used to repeatedly execute the given function in the mentioned delay and only halts when cancelled. The clearInterval(id)function instructs the timer to stop.

Timers are operated within a single thread, and thus events might queue up, waiting to be executed.

15. Which symbol is used for comments in Javascript?

// for Single line comments and

/\* Multi

Line

Comment

\*/

16. What is the difference between ViewState and SessionState?

'ViewState' is specific to a page in a session.

'SessionState' is specific to user specific data that can be accessed across all pages in the web application.

17. What is === operator?

=== is called as strict equality operator which returns true when the two operands are having the same value without any type conversion.

18. Explain how can you submit a form using JavaScript?

To submit a form using JavaScript use document.form[0].submit();

document.form[0].submit();

19. Does JavaScript support automatic type conversion?

Yes JavaScript does support automatic type conversion, it is the common way of type conversion used by JavaScript developers

20. How can the style/class of an element be changed?

It can be done in the following way:

document.getElementById("myText").style.fontSize = "20?;

or

document.getElementById("myText").className = "anyclass";

21. Explain how to read and write a file using JavaScript?

There are two ways to read and write a file using JavaScript

* Using JavaScript extensions
* Using a web page and Active X objects

22. What are all the looping structures in JavaScript?

Following are looping structures in Javascript:

* For
* While
* do-while loops

23. What is called Variable typing in Javascript?

Variable typing is used to assign a number to a variable and the same variable can be assigned to a string.

Example

i = 10;

i = "string";

This is called variable typing.

24. How can you convert the string of any base to integer in JavaScript?

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

In order to convert 4F (of base 16) to integer, the code used will be -

parseInt ("4F", 16);

25. Explain the difference between "==" and "==="?

"==" checks only for equality in value whereas "===" is a stricter equality test and returns false if either the value or the type of the two variables are different.

26. What would be the result of 3+2+"7"?

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

27. Explain how to detect the operating system on the client machine?

In order to detect the operating system on the client machine, the navigator.platform string (property) should be used.

28. What do mean by NULL in Javascript?

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number and no array object.

29. What is the function of delete operator?

The delete keyword is used to delete the property as well as its value.

Example

var student= {age:20, batch:"ABC"};

delete student.age;

30. What is an undefined value in JavaScript?

Undefined value means the

* Variable used in the code doesn't exist
* Variable is not assigned to any value
* Property doesn't exist

31. What are all the types of Pop up boxes available in JavaScript?

* Alert
* Confirm and
* Prompt

32. What is the use of Void(0)?

Void(0) is used to prevent the page from refreshing and parameter "zero" is passed while calling.

Void(0) is used to call another method without refreshing the page.

33. How can a page be forced to load another page in JavaScript?

The following code has to be inserted to achieve the desired effect:

<script language="JavaScript" type="text/javascript" >

<!-- location.href="http://newhost/newpath/newfile.html"; //--></script>

34. What is the data type of variables of in JavaScript?

All variables in the JavaScript are object data types.

35. What is the difference between an alert box and a confirmation box?

An alert box displays only one button which is the OK button.

But a Confirmation box displays two buttons namely OK and cancel.

36. What are escape characters?

Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes and ampersands. Place backslash before the characters to make it display.

Example:

document.write "I m a "good" boy"

document.write "I m a \"good\" boy"

37. What are JavaScript Cookies?

Cookies are the small test files stored in a computer and it gets created when the user visits the websites to store information that they need. Example could be User Name details and shopping cart information from the previous visits.

38. Explain what is pop()method in JavaScript?

The pop() method is similar as the shift() method but the difference is that the Shift method works at the start of the array. Also the pop() method take the last element off of the given array and returns it. The array on which is called is then altered.

Example:

var cloths = ["Shirt", "Pant", "TShirt"];

cloths.pop();

//Now cloth becomes Shirt,Pant

39. Whether JavaScript has concept level scope?

No. JavaScript does not have concept level scope. The variable declared inside the function has scope inside the function.

40. Mention what is the disadvantage of using innerHTML in JavaScript?

If you use innerHTML in JavaScript the disadvantage is

* Content is replaced everywhere
* We cannot use like "appending to innerHTML"
* Even if you use +=like "innerHTML = innerHTML + 'html'" still the old content is replaced by html
* The entire innerHTML content is re-parsed and build into elements, therefore its much slower
* The innerHTML does not provide validation and therefore we can potentially insert valid and broken HTML in the document and break it

41. What is break and continue statements?

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

42. What are the two basic groups of dataypes in JavaScript?

They are as –

* Primitive
* Reference types.

Primitive types are number and Boolean data types. Reference types are more complex types like strings and dates.

43. How generic objects can be created?

Generic objects can be created as:

var I = new object();

44. What is the use of type of operator?

'Typeof' is an operator which is used to return a string description of the type of a variable.

45. Which keywords are used to handle exceptions?

Try… Catch---finally is used to handle exceptions in the JavaScript

Try{

Code

}

Catch(exp){

Code to throw an exception

}

Finally{

Code runs either it finishes successfully or after catch

}

46. Which keyword is used to print the text in the screen?

document.write("Welcome") is used to print the text – Welcome in the screen.

47. What is the use of blur function?

Blur function is used to remove the focus from the specified object.

48. What is variable typing?

Variable typing is used to assign a number to a variable and then assign string to the same variable. Example is as follows:

i= 8;

i="john";

49. How to find operating system in the client machine using JavaScript?

The 'Navigator.appversion' is used to find the name of the operating system in the client machine.

50. What are the different types of errors in JavaScript?

There are three types of errors:

* Load time errors: Errors which come up when loading a web page like improper syntax errors are known as Load time errors and it generates the errors dynamically.
* Run time errors: Errors that come due to misuse of the command inside the HTML language.
* Logical Errors: These are the errors that occur due to the bad logic performed on a function which is having different operation.

51. What is the use of Push method in JavaScript?

The push method is used to add or append one or more elements to the end of an Array. Using this method, we can append multiple elements by passing multiple arguments

52. What is unshift method in JavaScript?

Unshift method is like push method which works at the beginning of the array. This method is used to prepend one or more elements to the beginning of the array.

53. What is the difference between JavaScript and Jscript?

Both are almost similar. JavaScript is developed by Netscape and Jscript was developed by Microsoft .

54. How are object properties assigned?

Properties are assigned to objects in the following way -

obj["class"] = 12;

or

obj.class = 12;

55. What is the 'Strict' mode in JavaScript and how can it be enabled?

Strict Mode adds certain compulsions to JavaScript. Under the strict mode, JavaScript shows errors for a piece of codes, which did not show an error before, but might be problematic and potentially unsafe. Strict mode also solves some mistakes that hamper the JavaScript engines to work efficiently.

Strict mode can be enabled by adding the string literal "use strict" above the file. This can be illustrated by the given example:

function myfunction() {

"use strict";

var v = "This is a strict mode function";

}

56. What is the way to get the status of a CheckBox?

The status can be acquired as follows -

alert(document.getElementById('checkbox1').checked);

If the CheckBox will be checked, this alert will return TRUE.

57. How can the OS of the client machine be detected?

The navigator.appVersion string can be used to detect the operating system on the client machine.

58. Explain window.onload and onDocumentReady?

The onload function is not run until all the information on the page is loaded. This leads to a substantial delay before any code is executed.

onDocumentReady loads the code just after the DOM is loaded. This allows early manipulation of the code.

59. How will you explain closures in JavaScript? When are they used?

Closure is a locally declared variable related to a function which stays in memory when the function has returned.

For example:

function greet(message) {

console.log(message);

}

function greeter(name, age) {

return name + " says howdy!! He is " + age + " years old";

}

// Generate the message

var message = greeter("James", 23);

// Pass it explicitly to greet

greet(message);

This function can be better represented by using closures

function greeter(name, age) {

var message = name + " says howdy!! He is " + age + " years old";

return function greet() {

console.log(message);

};

}

// Generate the closure

var JamesGreeter = greeter("James", 23);

// Use the closure

JamesGreeter();

60. How can a value be appended to an array?

A value can be appended to an array in the given manner -

arr[arr.length] = value;

61. Explain the for-in loop?

The for-in loop is used to loop through the properties of an object.

The syntax for the for-in loop is -

for (variable name in object){

statement or block to execute

}

In each repetition, one property from the object is associated to the variable name, and the loop is continued till all the properties of the object are depleted.

62. Describe the properties of an anonymous function in JavaScript?

A function that is declared without any named identifier is known as an anonymous function. In general, an anonymous function is inaccessible after its declaration.

Anonymous function declaration -

var anon = function() {

alert('I am anonymous');

};

anon();

63. What is the difference between .call() and .apply()?

The function .call() and .apply() are very similar in their usage except a little difference. .call() is used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement. On the other hand, .apply() is used when the number is not known. The function .apply() expects the argument to be an array.

The basic difference between .call() and .apply() is in the way arguments are passed to the function. Their usage can be illustrated by the given example.

var someObject = {

myProperty : 'Foo',

myMethod : function(prefix, postfix) {

alert(prefix + this.myProperty + postfix);

}

};

someObject.myMethod('<', '>'); // alerts '<Foo>'

var someOtherObject = {

myProperty : 'Bar'

};

someObject.myMethod.call(someOtherObject, '<', '>'); // alerts '<Bar>'

someObject.myMethod.apply(someOtherObject, ['<', '>']); // alerts '<Bar>'

64. Define event bubbling?

JavaScript allows DOM elements to be nested inside each other. In such a case, if the handler of the child is clicked, the handler of parent will also work as if it were clicked too.

65. Is JavaScript case sensitive? Give an example?

Yes, JavaScript is case sensitive. For example, a function parseInt is not same as the function Parseint.

66. What boolean operators can be used in JavaScript?

The 'And' Operator (&&), 'Or' Operator (||) and the 'Not' Operator (!) can be used in JavaScript.

\*Operators are without the parenthesis.

67. How can a particular frame be targeted, from a hyperlink, in JavaScript?

This can be done by including the name of the required frame in the hyperlink using the 'target' attribute.

<a href="/newpage.htm" target="newframe">>New Page</a>

68. What is the role of break and continue statements?

Break statement is used to come out of the current loop while the continue statement continues the current loop with a new recurrence.

69. Write the point of difference between web-garden and a web-farm?

Both web-garden and web-farm are web hosting systems. The only difference is that web-garden is a setup that includes many processors in a single server while web-farm is a larger setup that uses more than one server.

70. How are object properties assigned?

Assigning properties to objects is done in the same way as a value is assigned to a variable. For example, a form object's action value is assigned as 'submit' in the following manner - Document.form.action="submit"

71. What is the method for reading and writing a file in JavaScript?

This can be done by Using JavaScript extensions (runs from JavaScript Editor), example for opening of a file -

fh = fopen(getScriptPath(), 0);

72. How are DOM utilized in JavaScript?

DOM stands for Document Object Model and is responsible for how various objects in a document interact with each other. DOM is required for developing web pages, which includes objects like paragraph, links, etc. These objects can be operated to include actions like add or delete. DOM is also required to add extra capabilities to a web page. On top of that, the use of API gives an advantage over other existing models.

73. How are event handlers utilized in JavaScript?

Events are the actions that result from activities, such as clicking a link or filling a form, by the user. An event handler is required to manage proper execution of all these events. Event handlers are an extra attribute of the object. This attribute includes event's name and the action taken if the event takes place.

74. Explain the role of deferred scripts in JavaScript?

By default, the parsing of the HTML code, during page loading, is paused until the script has not stopped executing. It means, if the server is slow or the script is particularly heavy, then the webpage is displayed with a delay. While using Deferred, scripts delays execution of the script till the time HTML parser is running. This reduces the loading time of web pages and they get displayed faster.

75. What are the various functional components in JavaScript?

The different functional components in JavaScript are-

First-class functions: Functions in JavaScript are utilized as first class objects. This usually means that these functions can be passed as arguments to other functions, returned as values from other functions, assigned to variables or can also be stored in data structures.

Nested functions: The functions, which are defined inside other functions, are called Nested functions. They are called 'everytime' the main function is invoked.

76. Write about the errors shown in JavaScript?

JavaScript gives a message if it encounters an error. The recognized errors are -

* Load-time errors: The errors shown at the time of the page loading are counted under Load-time errors. These errors are encountered by the use of improper syntax, and thus are detected while the page is getting loaded.
* Run-time errors: This is the error that comes up while the program is running. It is caused by illegal operations, for example, division of a number by zero, or trying to access a non-existent area of the memory.
* Logic errors: It is caused by the use of syntactically correct code, which does not fulfill the required task. For example, an infinite loop.

77. What are Screen objects?

Screen objects are used to read the information from the client's screen. The properties of screen objects are -

* AvailHeight: Gives the height of client's screen
* AvailWidth: Gives the width of client's screen.
* ColorDepth: Gives the bit depth of images on the client's screen
* Height: Gives the total height of the client's screen, including the taskbar
* Width: Gives the total width of the client's screen, including the taskbar

78. Explain the unshift() method ?

This method is functional at the starting of the array, unlike the push(). It adds the desired number of elements to the top of an array. For example -

var name = [ "john" ];

name.unshift( "charlie" );

name.unshift( "joseph", "Jane" );

console.log(name);

The output is shown below:

[" joseph "," Jane ", " charlie ", " john "]

79. Define unescape() and escape() functions?

The escape () function is responsible for coding a string so as to make the transfer of the information from one computer to the other, across a network.

For Example:

<script>

document.write(escape("Hello? How are you!"));

</script>

Output: Hello%3F%20How%20are%20you%21

The unescape() function is very important as it decodes the coded string.

It works in the following way. For example:

<script>

document.write(unescape("Hello%3F%20How%20are%20you%21"));

</script>

Output: Hello? How are you!

80. What are the decodeURI() and encodeURI()?

EncodeURl() is used to convert URL into their hex coding. And DecodeURI() is used to convert the encoded URL back to normal.

<script>

var uri="my test.asp?name=ståle&car=saab";

document.write(encodeURI(uri)+ "<br>");

document.write(decodeURI(uri));

</script>

Output -

my%20test.asp?name=st%C3%A5le&car=saab

my test.asp?name=ståle&car=saab

81. Why it is not advised to use innerHTML in JavaScript?

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML and, therefore, it is easier to insert rouge code in the document and, thus, make the web page unstable.

82. What does the following statement declares?

var myArray = [[[]]];

It declares a three dimensional array.

83. How are JavaScript and ECMA Script related?

ECMA Script are like rules and guideline while Javascript is a scripting language used for web development.

84. What is namespacing in JavaScript and how is it used?

Namespacing is used for grouping the desired functions, variables etc. under a unique name. It is a name that has been attached to the desired functions, objects and properties. This improves modularity in the coding and enables code reuse.

85. How can JavaScript codes be hidden from old browsers that don't support JavaScript?

For hiding JavaScript codes from old browsers:

Add "<!--" without the quotes in the code just after the <script> tag.

Add "//-->" without the quotes in the code just before the <script> tag.

Old browsers will now treat this JavaScript code as a long HTML comment. While, a browser that supports JavaScript, will take the "<!--" and "//-->" as one-line comments.

**Beginner Level JavaScript Interview Questions**

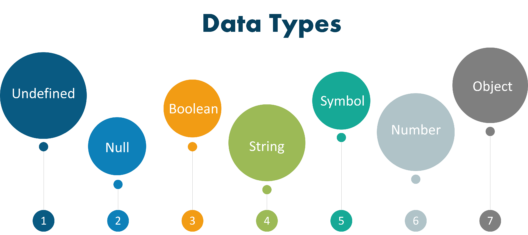
**Q1. What is the difference between Java & JavaScript?**

|  |  |
| --- | --- |
| **Java** | **JavaScript** |
| Java is an OOP programming language. | JavaScript is an OOP scripting language. |
| It creates applications that run in a virtual machine or browser. | The code is run on a browser only. |
| Java code needs to be compiled. | JavaScript code are all in the form of text. |

**Q2. What is JavaScript?**

[JavaScript](https://www.edureka.co/blog/what-is-javascript/) is a **lightweight**, **interpreted** programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

**Q3. What are the data types supported by JavaScript?**



The **data types** supported by JavaScript are:

* Undefined
* Null
* Boolean
* String
* Symbol
* Number
* Object

**Q4. What are the features of JavaScript?**



Following are the **features** of JavaScript:

* It is a **lightweight, interpreted** programming language.
* It is designed for creating **network-centric** applications.
* It is complementary to and **integrated** with Java.
* It is an **open** and **cross-platform** scripting language.

**Q5. Is JavaScript a case-sensitive language?**

Yes, JavaScript is a**case sensitive** language.  The language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

**Q6. What are the advantages of JavaScript?**

Following are the **advantages** of using JavaScript −

* **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
* **Immediate feedback to the visitors** − They don’t have to wait for a page reload to see if they have forgotten to enter something.
* **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
* **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

**Q7. How can you create an object in JavaScript?**

JavaScript supports **Object** concept very well. You can create an object using the **object literal** as follows −

|  |  |
| --- | --- |
| 1  2  3  4 | var emp = {  name: "Daniel",  age: 23  }; |

**Q8. How can you create an Array in JavaScript?**

You can define arrays using the **array literal** as follows-

|  |  |
| --- | --- |
| 1  2 | var x = [];  var y = [1, 2, 3, 4, 5]; |

**Q9. What is a name function in JavaScript & how to define it?**

A named function declares a name as soon as it is defined. It can be defined using **function**keyword as :

|  |  |
| --- | --- |
| 1  2  3 | function named(){  // write code here  } |

**Q10. Can you assign an anonymous function to a variable and pass it as an argument to another function?**

Yes! An anonymous function can be assigned to a variable. It can also be passed as an argument to another function.

**Q11. What is argument objects in JavaScript & how to get the type of arguments passed to a function?**

JavaScript variable arguments represents the **arguments** that are passed to a function. Using **typeof** operator, we can get the type of arguments passed to a function. For example −

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | function func(x){  console.log(typeof x, arguments.length);  }  func(); //==> "undefined", 0  func(7); //==> "number", 7  func("1", "2", "3"); //==> "string", 3 |

**Q12. What are the scopes of a variable in JavaScript?**

The scope of a variable is the **region** of your program in which it is **defined**. JavaScript variable will have only two scopes.  
• **Global Variables** − A global variable has global scope which means it is visible everywhere in your JavaScript code.  
•**Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

**Q13. What is the purpose of ‘This’ operator in JavaScript?**

The JavaScript **this** keyword refers to the object it belongs to. This has different values depending on where it is used. In a method, this refers to the owner object and in a function, this refers to the global object.

*[Powered by Edureka](https://www.edureka.co/mymock-interview-service" \t "_blank)*[[](https://www.edureka.co/mymock-interview-service)](https://www.edureka.co/mymock-interview-service" \t "_blank)

**[80% INTERVIEW REJECTIONS HAPPEN IN FIRST 90 SECONDS](https://www.edureka.co/mymock-interview-service" \t "_blank)**

**[Take Javascript Mock Interview](https://www.edureka.co/mymock-interview-service" \t "_blank)**

* **[Get Interviewed by Industry Experts](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Personalized interview feedback](https://www.edureka.co/mymock-interview-service" \t "_blank)**

[BOOK A SLOT](https://www.edureka.co/mymock-interview-service" \t "_blank)

**Q14. What is Callback?**

A **callback** is a plain JavaScript function passed to some method as an argument or option. It is a function that is to be **executed** after another function has finished executing, hence the name ‘**call back**‘. In JavaScript, functions are objects. Because of this, functions can take functions as arguments, and can be returned by other functions.

****

**Q15. What is Closure? Give an example.**

**Closures** are created whenever a variable that is defined outside the **current scope** is accessed from within some inner scope. It gives you access to an outer function’s scope from an inner function. In JavaScript, closures are created every time a function is created. To use a closure, simply define a function inside another function and expose it.

**Q16. Name some of the built-in methods and the values returned by them.**

|  |  |
| --- | --- |
| **Built-in Method** | **Values** |
| **CharAt()** | It returns the character at the specified index. |
| **Concat()** | It returns the character at the specified index. |
| **forEach()** | It calls a function for each element in the array. |
| **indexOf()** | It returns the index within the calling String object of the first occurrence of the specified value. |
| **length()** | It returns the length of the string. |
| **pop()** | It removes the last element from an array and returns that element. |
| **push()** | It adds one or more elements to the end of an array and returns the new length of the array. |
| **reverse()** | It reverses the order of the elements of an array. |

**Q17. What are the variable naming conventions in JavaScript?**

The following **rules** are to be followed while **naming variables** in JavaScript:

1. You should not use any of the JavaScript **reserved keyword** as variable name. For example, break or boolean variable names are not valid.
2. JavaScript variable names should not start with a **numeral** (0-9). They must begin with a letter or the underscore character. For example, 123name is an invalid variable name but \_123name or name123 is a valid one.
3. JavaScript variable names are **case sensitive**. For example, Test and test are two different variables.

**Q18. How does TypeOf Operator work?**

The **typeof** operator is used to get the data type of its operand. The operand can be either a **literal**or a **data structure** such as a variable, a function, or an object. It is a **unary** operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

**Q19. How to create a cookie using JavaScript?**

The simplest way to create a cookie is to assign a string value to the **document.cookie** object, which looks like this-

**Syntax :**

|  |  |
| --- | --- |
| 1 | document.cookie = "key1 = value1; key2 = value2; expires = date"; |

**Q20. How to read a cookie using JavaScript?**

Reading a cookie is just as simple as writing one, because the value of the document.cookie object is the cookie. So you can use this string whenever you want to access the cookie.

* The **document.cookie** string will keep a list of name = value pairs separated by semicolons, where name is the name of a cookie and value is its string value.
* You can use strings’ **split()** function to break the string into key and values.

**Q21. How to delete a cookie using JavaScript?**

If you want to delete a cookie so that subsequent attempts to read the cookie return nothing, you just need to set the expiration date to a time in the past. You should define the cookie path to ensure that you delete the right cookie. Some browsers will not let you delete a cookie if you don’t specify the path.

Now let’s move on to the next section of JavaScript interview questions.

**Intermediate Level JavaScript Interview Questions**

**Q22. What is the difference between Attributes and Property?**

**Attributes-**  provide more details on an element like id, type, value etc.

**Property-**  is the value assigned to the property like type=”text”, value=’Name’ etc.

**Q23. List out the different ways an HTML element can be accessed in a JavaScript code.**

Here are the list of ways an HTML element can be accessed in a Javascript code:  
(i) **getElementById(‘idname’):** Gets an element by its ID name  
(ii) **getElementsByClass(‘classname’):** Gets all the elements that have the given classname.  
(iii) **getElementsByTagName(‘tagname’):** Gets all the elements that have the given tag name.  
(iv) **querySelector():** This function takes css style selector and returns the first selected element.

**Q24. In how many ways a JavaScript code can be involved in an HTML file?**

There are 3 different ways in which a JavaScript code can be involved in an HTML file:

* **Inline**
* **Internal**
* **External**

An **inline** function is a JavaScript function, which is assigned to a variable created at runtime. You can differentiate between Inline Functions and Anonymous since an inline function is assigned to a variable and can be easily reused. When you need a JavaScript for a function, you can either have the script **integrated** in the page you are working on, or you can have it placed in a **separate** file that you call, when needed. This is the difference between an **internal**script and an **external**script.

**Q25. What are the ways to define a variable in JavaScript?**

The three possible ways of defining a variable in JavaScript are:

* **Var** – The JavaScript variables statement is used to declare a variable and, optionally, we can initialize the value of that variable. Example: var a =10; Variable declarations are processed before the execution of the code.
* **Const** – The idea of const functions is not allow them to modify the object on which they are called. When a function is declared as const, it can be called on any type of object.
* **Let** – It is a signal that the variable may be reassigned, such as a counter in a loop, or a value swap in an algorithm. It also signals that the variable will be used only in the block it’s defined in.

**Q26. What is a Typed language?**

Typed Language is in which the values are associated with**values** and not with **variables**. It is of two types:

* **Dynamically:** in this, the variable can hold multiple types; like in JS a variable can take number, chars.
* **Statically:** in this, the variable can hold only one type, like in Java a variable declared of string can take only set of characters and nothing else.

**Q27. What is the difference between Local storage & Session storage?**

****

**Local Storage** – The data is not sent back to the server for every HTTP request (HTML, images, JavaScript, CSS, etc) – reducing the amount of traffic between client and server. It will stay until it is manually cleared through settings or program.

**Session Storage** – It is similar to local storage; the only difference is while data stored in local storage has no expiration time, data stored in session storage gets cleared when the page session ends. Session Storage will leave when the browser is closed.

**Q28. What is the difference between the operators ‘==‘ & ‘===‘?**

The main difference between “==” and “===” operator is that formerly compares variable by making **type correction** e.g. if you compare a number with a string with numeric literal, == allows that, but === doesn’t allow that, because it not only checks the value but also type of two variable, if two variables are not of the same type “===” return false, while “==” return true.

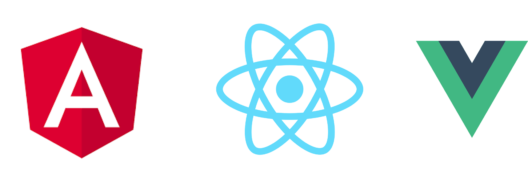
**Q29. What is the difference between null & undefined?**

Undefined means a variable has been **declared** but has not yet been **assigned** a value. On the other hand, null is an assignment value. It can be assigned to a variable as a representation of no value. Also, undefined and null are two distinct types: undefined is a type itself (undefined) while null is an object.

**Q30. What is the difference between undeclared & undefined?**

Undeclared variables are those that do not **exist** in a program and are not declared. If the program tries to read the value of an undeclared variable, then a **runtime error** is encountered. Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

**Q31. Name some of the JavaScript Frameworks**

A [JavaScript framework](https://www.edureka.co/blog/top-10-javascript-frameworks/) is an application framework written in JavaScript. It differs from a JavaScript library in its control flow. There are many JavaScript Frameworks available but some of the most commonly used frameworks are:

* [Angular](https://www.edureka.co/angular-training)
* [React](https://www.edureka.co/blog/reactjs-tutorial)
* Vue

**Q32. What is the difference between window & document in JavaScript?**

|  |  |
| --- | --- |
| **Window** | **Document** |
| JavaScript window is a global object which holds variables, functions, history, location. | The document also comes under the window and can be considered as the property of the window. |

**Q33. What is the difference between innerHTML & innerText?**

**innerHTML** – It will process an HTML tag if found in a string

**innerText** – It will not process an HTML tag if found in a string

**Q34. What is an event bubbling in JavaScript?**

Event bubbling is a way of **event propagation** in the HTML DOM API, when an event occurs in an element inside another element, and both elements have registered a handle for that event. With bubbling, the event is first captured and handled by the **innermost** element and then propagated to outer elements. The execution starts from that event and goes to its parent element. Then the execution passes to its parent element and so on till the body element.

*[Powered by Edureka](https://www.edureka.co/mymock-interview-service" \t "_blank)*[[](https://www.edureka.co/mymock-interview-service)](https://www.edureka.co/mymock-interview-service" \t "_blank)

**[Javascript Mock interviews for you](https://www.edureka.co/mymock-interview-service" \t "_blank)**

* **[Interviews by Industry Experts](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Personalized detailed interview feedback](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Access to exclusive and curated content](https://www.edureka.co/mymock-interview-service" \t "_blank)**

[SCHEDULE NOW](https://www.edureka.co/mymock-interview-service" \t "_blank)

**Q35. What is NaN in JavaScript?**

**NaN** is a short form of **Not a Number.** Since NaN always compares unequal to any number, including NaN, it is usually used to indicate an error condition for a function that should return a valid number. When a string or something else is being **converted** into a **number** and that cannot be done, then we get to see NaN.

**Q36. How do JavaScript primitive/object types passed in functions?**

One of the differences between the two is that Primitive Data Types are passed By Value and Objects are passed By Reference.

* **By Value** means creating a COPY of the original. Picture it like twins: they are born exactly the same, but the first twin doesn’t lose a leg when the second twin loses his in the war.
* **By Reference** means creating an ALIAS to the original. When your Mom calls you “Pumpkin Pie” although your name is Margaret, this doesn’t suddenly give birth to a clone of yourself: you are still one, but you can be called by these two very different names.

**Q37. How can you convert the string of any base to integer in JavaScript?**

The **parseInt()** function is used to convert numbers between different bases. It takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

For example-

|  |  |
| --- | --- |
| 1 | parseInt("4F", 16) |

**Q38. What would be the result of 2+5+”3″?**

Since 2 and 5 are integers, they will be added numerically. And since 3 is a string, its concatenation will be done. So the result would be 73. The ” ” makes all the difference here and represents 3 as a string and not a number.

**Q39. What are Exports & Imports?**

Imports and exports help us to write modular JavaScript code. Using Imports and exports we can split our code into multiple files. For example-

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | //------ lib.js ------</span>  export const sqrt = Math.sqrt;</span>  export function square(x) {</span>  return x \* x;</span>  }  export function diag(x, y) {  return sqrt(square(x) + square(y));  }    //------ main.js ------</span>   { square, diag } from 'lib';  console.log(square(5)); // 25  console.log(diag(4, 3)); // 5 |

Now with this, we have reached the final section of JavaScript Interview Questions.

**Advanced Level JavaScript Interview Questions**

**Q40. What is the ‘Strict’ mode in JavaScript and how can it be enabled?**

Strict mode is a way to introduce better error-checking into your code.

* When you use strict mode, you cannot use implicitly declared variables, or assign a value to a read-only property, or add a property to an object that is not extensible.
* You can enable strict mode by adding “use strict” at the beginning of a file, a program, or a function.

**Q41. What is a prompt box in JavaScript?**

A prompt box is a box which allows the user to enter input by providing a **text box**. The prompt() method displays a dialog box that prompts the visitor for input. A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either “OK” or “Cancel” to proceed after entering an input value.

**Q42. What will be the output of the code below?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var Y = 1;  if (function F(){})  {  y += Typeof F;</span>  }  console.log(y); |

The output would be 1undefined. The if condition statement evaluates using eval, so eval(function f(){}) returns function f(){} (which is true). Therefore, inside the if statement, executing typeof f returns undefined because the if statement code executes at run time, and the statement inside the if condition is evaluated during run time.

**Q43. What is the difference between Call & Apply?**

The **call()** method calls a function with a given this value and arguments provided individually.

**Syntax-**

|  |  |
| --- | --- |
| 1 | fun.call(thisArg[, arg1[, arg2[, ...]]]) |

The **apply()** method calls a function with a given this value, and arguments provided as an array.

**Syntax-**

|  |  |
| --- | --- |
| 1 | fun.apply(thisArg, [argsArray]) |

**Q44. How to empty an Array in JavaScript?**

There are a number of methods you can use to **empty** an **array**:

**Method 1 –**

|  |  |
| --- | --- |
| 1 | arrayList = [] |

Above code will set the variable arrayList to a new empty array. This is recommended if you don’t have references to the original array arrayList anywhere else, because it will actually create a new, empty array. You should be careful with this method of emptying the array, because if you have referenced this array from another variable, then the original reference array will remain unchanged.

**Method 2 –**

|  |  |
| --- | --- |
| 1 | arrayList.length = 0; |

The code above will clear the existing array by setting its length to 0. This way of emptying the array also updates all the reference variables that point to the original array. Therefore, this method is useful when you want to update all reference variables pointing to arrayList.

**Method 3 –**

|  |  |
| --- | --- |
| 1 | arrayList.splice(0, arrayList.length); |

The implementation above will also work perfectly. This way of emptying the array will also update all the references to the original array.

*[Powered by Edureka](https://www.edureka.co/mymock-interview-service" \t "_blank)*[[](https://www.edureka.co/mymock-interview-service)](https://www.edureka.co/mymock-interview-service" \t "_blank)

**[NEED HELP FOR YOUR UPCOMING INTERVIEW?](https://www.edureka.co/mymock-interview-service" \t "_blank)**

**[Take Javascript Mock Interview](https://www.edureka.co/mymock-interview-service" \t "_blank)**

* **[Get Interviewed by Industry Experts](https://www.edureka.co/mymock-interview-service" \t "_blank)**
* **[Personalized interview feedback](https://www.edureka.co/mymock-interview-service" \t "_blank)**

[BOOK A SLOT](https://www.edureka.co/mymock-interview-service" \t "_blank)

**Method 4 –**

|  |  |
| --- | --- |
| 1  2  3  4 | while(arrayList.length)  {  arrayList.pop();  } |

The implementation above can also empty arrays, but it is usually not recommended to use this method often.

**Q45. What will be the output of the following code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var Output = (function(x)  {  Delete X;  return X;  }  )(0);  console.log(output); |

The output would be 0. The delete operator is used to delete properties from an object. Here x is not an object but a local variable. delete operators don’t affect local variables.

**Q46. What will be the output of the following code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var X = { Foo : 1};  var Output = (function()  {  delete X.foo;  return X.foo;  }  )();  console.log(output); |

The output would be undefined. The delete operator is used to delete the property of an object. Here, x is an object which has the property foo, and as it is a self-invoking function, we will delete the foo property from object x. After doing so, when we try to reference a deleted property foo, the result is undefined.

**Q47. What will be the output of the following code?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var Employee =  {  company: 'xyz'  }  var Emp1 = Object.create(employee);  delete Emp1.company Console.log(emp1.company); |

The output would be xyz. Here, emp1 object has company as its prototype property. The delete operator doesn’t delete prototype property. emp1 object doesn’t have company as its own property. However, we can delete the company property directly from the Employee object using delete Employee.company.

**Q48. What will be the output of the code below?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | //nfe (named function expression)  var Foo = Function Bar()  {  return 7;  };  typeof Bar(); |

The output would be Reference Error. A function definition can have only one reference variable as its function name.

**Q49. What is the reason for wrapping the entire content of a JavaScript source file in a function book?**

This is an increasingly common practice, employed by many popular JavaScript libraries. This technique creates a closure around the entire contents of the file which, perhaps most importantly, creates a private namespace and thereby helps avoid potential name clashes between different JavaScript modules and libraries.  
Another feature of this technique is to allow for an easy alias for a global variable. This is often used in jQuery plugins.

**Q50. What are escape characters in JavaScript?**

JavaScript escape characters enable you to write special characters without breaking your application. Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes and ampersands. Place backslash before the characters to make it display.

**For example-**

|  |  |
| --- | --- |
| 1  2 | document.write "I am a "good" boy"  document.write "I am a "good" boy" |

1) What is JavaScript?

**JavaScript** is *a scripting language*. It is different from Java language. It is object-based, lightweight, cross-platform translated language. It is widely used for client-side validation. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.[More details.](https://www.javatpoint.com/javascript-tutorial)

2) List some features of JavaScript.

Some of the features of JavaScript are:

* Lightweight
* Interpreted programming language
* Good for the applications which are network-centric
* Complementary to Java
* Complementary to HTML
* Open source

Cross-platform

3) List some of the advantages of JavaScript.

Some of the advantages of JavaScript are:

* Server interaction is less
* Feedback to the visitors is immediate
* Interactivity is high
* Interfaces are richer

4) List some of the disadvantages of JavaScript.

Some of the disadvantages of JavaScript are:

* No support for multithreading
* No support for multiprocessing
* Reading and writing of files is not allowed
* No support for networking applications.

5) Define a named function in JavaScript.

The function which has named at the time of definition is called a named function. For example

1. function msg()
2. {
3. document.writeln("Named Function");
4. }
5. msg();

6) Name the types of functions

The types of function are:

* Named - These type of functions contains name at the time of definition. For Example:
  1. function display()
  2. {
  3. document.writeln("Named Function");
  4. }
  5. display();
* Anonymous - These type of functions doesn't contain any name. They are declared dynamically at runtime.
  1. var display=function()
  2. {
  3. document.writeln("Anonymous Function");
  4. }
  5. display();

7) Define anonymous function

It is a function that has no name. These functions are declared dynamically at runtime using the function operator instead of the function declaration. The function operator is more flexible than a function declaration. It can be easily used in the place of an expression. For example:

1. var display=function()
2. {
3. alert("Anonymous Function is invoked");
4. }
5. display();

8) Can an anonymous function be assigned to a variable?

Yes, you can assign an anonymous function to a variable.

9) In JavaScript what is an argument object?

The variables of JavaScript represent the arguments that are passed to a function.

10) Define closure.

In JavaScript, we need closures when a variable which is defined outside the scope in reference is accessed from some inner scope.

1. var num = 10;
2. function sum()
3. {
4. document.writeln(num+num);
5. }
6. sum();

11) If we want to return the character from a specific index which method is used?

The JavaScript string charAt() method is used to find out a char value present at the specified index. The index number starts from 0 and goes to n-1, where n is the length of the string. The index value can't be a negative, greater than or equal to the length of the string. For example:

1. var str="Javatpoint";
2. document.writeln(str.charAt(4));

12) What is the difference between JavaScript and JScript?

Netscape provided the JavaScript language. Microsoft changed the name and called it JScript to avoid the trademark issue. In other words, you can say JScript is the same as JavaScript, but Microsoft provides it.

13) How to write a hello world example of JavaScript?

A simple example of JavaScript hello world is given below. You need to place it inside the body tag of HTML.

1. **<script** type="text/javascript"**>**
2. document.write("JavaScript Hello World!");
3. **</script>**

[More details.](https://www.javatpoint.com/javascript-example)

14) How to use external JavaScript file?

I am assuming that js file name is message.js, place the following script tag inside the head tag.

1. **<script** type="text/javascript" src="message.js"**></script>**

[More details.](https://www.javatpoint.com/external-javascript-file)

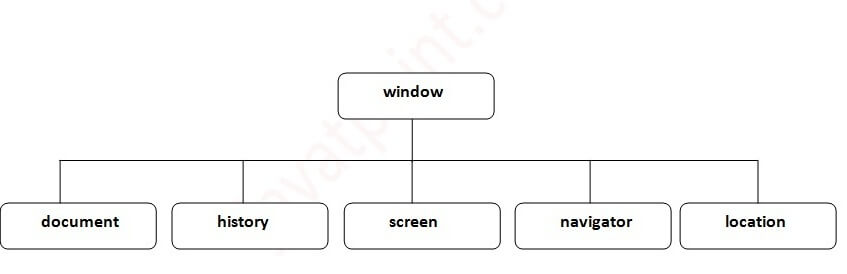
15) Is JavaScript case sensitive language?

Yes, JavaScript is a case sensitive language. For example:

1. Var msg = "JavaScript is a case-sensitive language"; //Here, var should be used to declare a variable
2. function display()
3. {
4. document.writeln(msg); // It will not display the result.
5. }
6. display();

16) What is BOM?

**BOM** stands for *Browser Object Model*. It provides interaction with the browser. The default object of a browser is a window. So, you can call all the functions of the window by specifying the window or directly. The window object provides various properties like document, history, screen, navigator, location, innerHeight, innerWidth,

[More Details: Browser Object Model](https://www.javatpoint.com/browser-object-model" \t "_blank)

17) What is DOM? What is the use of document object?

**DOM** stands for *Document Object Model*. A document object represents the HTML document. It can be used to access and change the content of HTML.

[More Details: Document Object Model](https://www.javatpoint.com/document-object-model" \t "_blank)

18) What is the use of window object?

The window object is created automatically by the browser that represents a window of a browser. It is not an object of JavaScript. It is a browser object.

The window object is used to display the popup dialog box. Let's see with description.

|  |  |
| --- | --- |
| **Method** | **Description** |
| alert() | displays the alert box containing the message with ok button. |
| confirm() | displays the confirm dialog box containing the message with ok and cancel button. |
| prompt() | displays a dialog box to get input from the user. |
| open() | opens the new window. |
| close() | closes the current window. |
| setTimeout() | performs the action after specified time like calling function, evaluating expressions. |

[More details.](https://www.javatpoint.com/window-object" \t "_blank)

19) What is the use of history object?

The history object of a browser can be used to switch to history pages such as back and forward from the current page or another page. There are three methods of history object.

1. history.back() - It loads the previous page.
2. history.forward() - It loads the next page.
3. history.go(number) - The number may be positive for forward, negative for backward. It loads the given page number.

[More details.](https://www.javatpoint.com/javascript-history-object" \t "_blank)

20) How to write a comment in JavaScript?

There are two types of comments in JavaScript.

1. Single Line Comment: It is represented by // (double forward slash)
2. Multi-Line Comment: Slash represents it with asterisk symbol as /\* write comment here \*/

[More details.](https://www.javatpoint.com/javascript-comment" \t "_blank)

21) How to create a function in JavaScript?

To create a function in JavaScript, follow the following syntax.

1. function function\_name(){
2. //function body
3. }

[More details.](https://www.javatpoint.com/javascript-function" \t "_blank)

22) What are the JavaScript data types?

There are two types of data types in JavaScript:

1. Primitive Data Types - The primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| String | represents a sequence of characters, e.g., "hello" |
| Number | represents numeric values, e.g., 100 |
| Boolean | represents boolean value either false or true |
| Undefined | represents an undefined value |
| Null | represents null, i.e., no value at all |

1. Non-primitive Data Types - The non-primitive data types are as follows:

|  |  |
| --- | --- |
| **Data Type** | **Description** |
| Object | represents an instance through which we can access members |
| Array | represents a group of similar values |
| RegExp | represents regular expression |

[More details.](https://www.javatpoint.com/javascript-data-types" \t "_blank)

23) What is the difference between == and ===?

The == operator checks equality only whereas === checks equality, and data type, i.e., a value must be of the same type.

24) How to write HTML code dynamically using JavaScript?

The innerHTML property is used to write the HTML code using JavaScript dynamically. Let's see a simple example:

1. document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";

[More details.](https://www.javatpoint.com/javascript-innerHTML" \t "_blank)

25) How to write normal text code using JavaScript dynamically?

The innerText property is used to write the simple text using JavaScript dynamically. Let's see a simple example:

1. document.getElementById('mylocation').innerText="This is text using JavaScript";

[More details.](https://www.javatpoint.com/javascript-innerText" \t "_blank)

26) How to create objects in JavaScript?

There are 3 ways to create an object in JavaScript.

1. By object literal
2. By creating an instance of Object
3. By Object Constructor

Let's see a simple code to create an object using object literal.

1. emp={id:102,name:"Rahul Kumar",salary:50000}

[More details.](https://www.javatpoint.com/javascript-objects" \t "_blank)

27) How to create an array in JavaScript?

There are 3 ways to create an array in JavaScript.

1. By array literal
2. By creating an instance of Array
3. By using an Array constructor

Let's see a simple code to create an array using object literal.

1. var emp=["Shyam","Vimal","Ratan"];

[More details.](https://www.javatpoint.com/javascript-array" \t "_blank)

28) What does the isNaN() function?

The isNan() function returns true if the variable value is not a number. For example:

1. function number(num) {
2. if (isNaN(num)) {
3. return "Not a Number";
4. }
5. return "Number";
6. }
7. console.log(number('1000F'));
8. // expected output: "Not a Number"
10. console.log(number('1000'));
11. // expected output: "Number"

29) What is the output of 10+20+"30" in JavaScript?

3030 because 10+20 will be 30. If there is numeric value before and after +, it treats as binary + (arithmetic operator).

1. function display()
2. {
3. document.writeln(10+20+"30");
4. }
5. display();

30) What is the output of "10"+20+30 in JavaScript?

102030 because after a string all the + will be treated as string concatenation operator (not binary +).

1. function display()
2. {
3. document.writeln("10"+20+30);
4. }
5. display();

31) Difference between Client side JavaScript and Server side JavaScript?

**Client-side JavaScript** comprises the basic language and predefined objects which are relevant to running JavaScript in a browser. The client-side JavaScript is embedded directly by in the HTML pages. The browser interprets this script at runtime.

**Server-side JavaScript** also resembles client-side JavaScript. It has a relevant JavaScript which is to run in a server. The server-side JavaScript are deployed only after compilation.

32) In which location cookies are stored on the hard disk?

The storage of cookies on the hard disk depends on the OS and the browser.

The Netscape Navigator on Windows uses a cookies.txt file that contains all the cookies. The path is c:\Program Files\Netscape\Users\username\cookies.txt

The Internet Explorer stores the cookies on a file username@website.txt. The path is: c:\Windows\Cookies\username@Website.txt.

33) What is the real name of JavaScript?

The original name was **Mocha**, a name chosen by Marc Andreessen, founder of Netscape. In September of 1995, the name was changed to LiveScript. In December 1995, after receiving a trademark license from Sun, the name JavaScript was adopted.

34) What is the difference between undefined value and null value?

**Undefined value:** A value that is not defined and has no keyword is known as undefined value. For example:

1. int number;//Here, a number has an undefined value.

**Null value:** A value that is explicitly specified by the keyword "null" is known as a null value. For example:

1. String str=null;//Here, str has a null value.

35) How to set the cursor to wait in JavaScript?

The cursor can be set to wait in JavaScript by using the property "cursor". The following example illustrates the usage:

1. **<script>**
2. window.document.body.style.cursor = "wait";
3. **</script>**

36) What is this [[[]]]?

This is a three-dimensional array.

1. var myArray = [[[]]];

37) Are Java and JavaScript same?

No, Java and JavaScript are the two different languages. Java is a robust, secured and object-oriented programming language whereas JavaScript is a client-side scripting language with some limitations.

38) What is negative infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing the negative number by zero. For example:

1. var num=-5;
2. function display()
3. {
4. document.writeln(num/0);
5. }
6. display();
7. //expected output: -Infinity

39) What is the difference between View state and Session state?

"View state" is specific to a page in a session whereas "Session state" is specific to a user or browser that can be accessed across all pages in the web application.

40) What are the pop-up boxes available in JavaScript?

* Alert Box
* Confirm Box
* Prompt Box

Example of alert() in JavaScript

1. **<script** type="text/javascript"**>**
2. function msg(){
3. alert("Hello Alert Box");
4. }
5. **</script>**
6. **<input** type="button" value="click" onclick="msg()"**/>**

Example of confirm() in JavaScript

1. **<script** type="text/javascript"**>**
2. function msg(){
3. var v= confirm("Are u sure?");
4. if(v==true){
5. alert("ok");
6. }
7. else{
8. alert("cancel");
9. }
11. }
12. **</script>**
14. **<input** type="button" value="delete record" onclick="msg()"**/>**

Example of prompt() in JavaScript

1. **<script** type="text/javascript"**>**
2. function msg(){
3. var v= prompt("Who are you?");
4. alert("I am "+v);
6. }
7. **</script>**
9. **<input** type="button" value="click" onclick="msg()"**/>**

41) How can we detect OS of the client machine using JavaScript?

The **navigator.appVersion** string can be used to detect the operating system on the client machine.

42) How to submit a form using JavaScript by clicking a link?

Let's see the JavaScript code to submit the form by clicking the link.

1. **<form** name="myform" action="index.php"**>**
2. Search: **<input** type='text' name='query' **/>**
3. **<a** href="javascript: submitform()"**>**Search**</a>**
4. **</form>**
5. **<script** type="text/javascript"**>**
6. function submitform()
7. {
8. document.myform.submit();
9. }
10. **</script>**

43) Is JavaScript faster than ASP script?

Yes, because it doesn't require web server's support for execution.

44) How to change the background color of HTML document using JavaScript?

1. **<script** type="text/javascript"**>**
2. document.body.bgColor="pink";
3. **</script>**

45) How to handle exceptions in JavaScript?

By the help of try/catch block, we can handle exceptions in JavaScript. JavaScript supports try, catch, finally and throw keywords for exception handling.

46) How to validate a form in JavaScript?

1. **<script>**
2. function validateform(){
3. var name=document.myform.name.value;
4. var password=document.myform.password.value;
6. if (name==null || name==""){
7. alert("Name can't be blank");
8. return false;
9. }else if(password.length**<6**){
10. alert("Password must be at least 6 characters long.");
11. return false;
12. }
13. }
14. **</script>**
15. **<body>**
16. **<form** name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" **>**
17. Name: **<input** type="text" name="name"**><br/>**
18. Password: **<input** type="password" name="password"**><br/>**
19. **<input** type="submit" value="register"**>**
20. **</form>**

**[Test it Now](http://www.javatpoint.com/oprweb/test.jsp?filename=jsvalidation1" \t "_blank)**

Visit here: [JavaScript form validation](https://www.javatpoint.com/javascript-form-validation" \t "_blank).

47) How to validate email in JavaScript?

1. **<script>**
2. function validateemail()
3. {
4. var x=document.myform.email.value;
5. var atposition=x.indexOf("@");
6. var dotposition=x.lastIndexOf(".");
7. if (atposition**<1** || dotposition**<atposition**+2 || dotposition+2**>**=x.length){
8. alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
9. return false;
10. }
11. }
12. **</script>**
13. **<body>**
14. **<form** name="myform"  method="post" action="#" onsubmit="return validateemail();"**>**
15. Email: **<input** type="text" name="email"**><br/>**
17. **<input** type="submit" value="register"**>**
18. **</form>**

**[Test it Now](http://www.javatpoint.com/oprweb/test.jsp?filename=jsvalidation3" \t "_blank)**

Visit here: [JavaScript Email validation](https://www.javatpoint.com/javascript-form-validation" \l "email" \t "_blank).

48) What is this keyword in JavaScript?

The this keyword is a reference variable that refers to the current object. For example:

1. var address=
2. {
3. company:"Javatpoint",
4. city:"Noida",
5. state:"UP",
6. fullAddress:function()
7. {
8. return this.company+" "+this.city+" "+this.state;
9. }
10. };
11. var fetch=address.fullAddress();
12. document.writeln(fetch);

49) What is the requirement of debugging in JavaScript?

JavaScript didn't show any error message in a browser. However, these mistakes can affect the output. The best practice to find out the error is to debug the code. The code can be debugged easily by using web browsers like Google Chrome, Mozilla Firebox.

To perform debugging, we can use any of the following approaches:

* Using console.log() method
* Using debugger keyword

50) What is the use of debugger keyword in JavaScript?

JavaScript debugger keyword sets the breakpoint through the code itself. The debugger stops the execution of the program at the position it is applied. Now, we can start the flow of execution manually. If an exception occurs, the execution will stop again on that particular line.. For example:

1. function display()
2. {
3. x = 10;
4. y = 15;
5. z = x + y;
6. debugger;
7. document.write(z);
8. document.write(a);
9. }
10. display();

51) What is the role of a strict mode in JavaScript?

The JavaScript strict mode is used to generates silent errors. It provides "use strict"; expression to enable the strict mode. This expression can only be placed as the first statement in a script or a function. For example:

1. "use strict";
2. x=10;
3. console.log(x);

52) What is the use of Math object in JavaScript?

The JavaScript math object provides several constants and methods to perform a mathematical operation. Unlike date object, it doesn't have constructors. For example:

1. function display()
2. {
3. document.writeln(Math.random());
4. }
5. display();

53) What is the use of a Date object in JavaScript?

The JavaScript date object can be used to get a year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

1. function display()
2. {
3. var date=new Date();
4. var day=date.getDate();
5. var month=date.getMonth()+1;
6. var year=date.getFullYear();
7. document.write("**<br>**Date is: "+day+"/"+month+"/"+year);
8. }
9. display();

54) What is the use of a Number object in JavaScript?

The JavaScript number object enables you to represent a numeric value. It may be integer or floating-point. JavaScript number object follows the IEEE standard to represent the floating-point numbers.

1. function display()
2. {
3. var x=102;//integer value
4. var y=102.7;//floating point value
5. var z=13e4;//exponent value, output: 130000
6. var n=new Number(16);//integer value by number object
7. document.write(x+" "+y+" "+z+" "+n);
8. }
9. display();

55) What is the use of a Boolean object in JavaScript?

The JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor.

1. function display()
2. {
3. document.writeln(10**<20**);//true
4. document.writeln(10**<5**);//false
5. }
6. display();

56) What is the use of a TypedArray object in JavaScript?

The JavaScript TypedArray object illustrates an array like a view of an underlying binary data buffer. There is any number of different global properties, whose values are TypedArray constructors for specific element types.

1. function display()
2. {
3. var arr1= [1,2,3,4,5,6,7,8,9,10];
4. arr1.copyWithin(2) ;
5. document.write(arr1);
6. }
7. display();

57) What is the use of a Set object in JavaScript?

The JavaScript Set object is used to store the elements with unique values. The values can be of any type i.e. whether primitive values or object references. For example:

1. function display()
2. {
3. var set = new Set();
4. set.add("jQuery");
5. set.add("AngularJS");
6. set.add("Bootstrap");
7. for (let elements of set) {
8. document.writeln(elements+"**<br>**");
9. }
10. }
11. display();

58) What is the use of a WeakSet object in JavaScript?

The JavaScript WeakSet object is the type of collection that allows us to store weakly held objects. Unlike Set, the WeakSet are the collections of objects only. It doesn't contain the arbitrary values. For example:

1. function display()
2. {
3. var ws = new WeakSet();
4. var obj1={};
5. var obj2={};
6. ws.add(obj1);
7. ws.add(obj2);
8. //Let's check whether the WeakSet object contains the added object
9. document.writeln(ws.has(obj1)+"**<br>**");
10. document.writeln(ws.has(obj2));
11. }
12. display()

59) What is the use of a Map object in JavaScript?

The JavaScript Map object is used to map keys to values. It stores each element as key-value pair. It operates the elements such as search, update and delete on the basis of specified key. For example:

1. function display()
2. {
3. var map=new Map();
4. map.set(1,"jQuery");
5. map.set(2,"AngularJS");
6. map.set(3,"Bootstrap");
7. document.writeln(map.get(1)+"**<br>**");
8. document.writeln(map.get(2)+"**<br>**");
9. document.writeln(map.get(3));
10. }
11. display();

60) What is the use of a WeakMap object in JavaScript?

The JavaScript WeakMap object is a type of collection which is almost similar to Map. It stores each element as a key-value pair where keys are weakly referenced. Here, the keys are objects and the values are arbitrary values. For example:

1. function display()
2. {
3. var wm = new WeakMap();
4. var obj1 = {};
5. var obj2 = {};
6. var obj3= {};
7. wm.set(obj1, "jQuery");
8. wm.set(obj2, "AngularJS");
9. wm.set(obj3,"Bootstrap");
10. document.writeln(wm.has(obj2));
11. }
12. display();

**Q #1) What is JavaScript?**

**Answer:** JavaScript is a scripting language developed by Netscape. It can be used to program web browser or even servers. It can dynamically update the contents of the webpage, which is the beauty of JavaScript.

**Q #2) What are the advantages of using External JavaScript?**

**Answer:** Using External JavaScript in our code has many advantages as stated below.

* Separation of Code is done.
* Code Maintainability is Easy.
* Performance is better.

**Q #3) In the following Code snippet can you please predict the output or If you get an error, please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p id="studentName"></p>

<script>

var studentName = "Sajeesh Sreeni"; // String 'Sajeesh Sreeni' stored in studentName

var studentName; // varaible is decalred again

document.getElementById("studentName").innerHTML =

"Redeclaring the varaible will not lose the value!.<br>"

+"Here the value in studentName is "+ studentName;

</script>

</body>

</html>

**Answer**: This code will not produce any errors.

Redeclaration of the variables is allowed in JavaScript. Hence, the value of the variable will not be lost after the execution of the statement here.

**Q #4) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p id="sum\_first"></p>

<p id="sum\_second"></p>

<script>

var sum\_first =50+20+' Sajeesh Sreeni ';

var sum\_second= " Sajeesh Sreeni "+50+20;

document.getElementById("sum\_first").innerHTML = "The first varaible sum is :"+sum\_first +

"<br>The second varaible sum is :"+sum\_second ;

</script>

</body>

</html>

**Answer:** This code will not show any error!

**The output of the code snippet here is:**

*The first variable sum is: 70 Sajeesh Sreeni*  
*The second variable sum is: Sajeesh Sreeni 5020*

**Q #5) What is the difference between test () and exec () methods?**

**Answer:**Both test () and exec () are RegExp expression methods.

Using test (), we will search a string for a given pattern, if it finds the matching text then it returns the Boolean value ‘true’ and else it returns ‘false’.

But in exec (), we will search a string for a given pattern, if it finds the matching text then it returns the pattern itself and else it returns ‘null’ value.

**Q #6) What are the Advantages of JavaScript?**

**Answer:**JavaScript Scripting language has many advantages as stated below.

* **Lightweight:** JavaScript is easy to implement. It has small memory footprints.
* **Interpreted:** It is an interpreted language. Instructions are executed directly.
* **Object-oriented:** JavaScript is an object-oriented language.
* **First class functions:** In JavaScript, a function can be used as a value.
* **Scripting Language:** It’s a language in which instructions are written for a run-time environment.

**Q #7) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example Const Variable </p>

<p id="display"></p>

<script>

const first\_num;

first\_num =1000;

document.getElementById("display").innerHTML = "First Number:"+ first\_num;

</script>

</body>

</html>

**Answer:**The ‘const’ variable ’first\_num’ is not initialized with a value, so the code will produce a Syntax Error.

**The output of the code snippet here is:**

*Error: Uncaught SyntaxError: Missing initializer in the const declaration*

**Q #8) Have you used any browser for debugging? If yes, how is it done?**

**Answer:**By, Pressing ‘F12’ key in the keyboard we can enable debugging in the browser. Chose the ‘Console’ tab to view the result.

In Console, we can set breakpoints and View the value in variables. All the modern browsers have a built-in debugger with them**(For Example: Chrome, Firefox, Opera, and Safari)**. This feature can be turned ON and OFF.

**Q #9) What is the use of ‘debugger’ keyword in JavaScript code?**

**Answer:** Using the ‘debugger’ keyword in the code is like using breakpoints in the debugger.

To test the code, the debugger must be enabled for the browser. If debugging is disabled for the browser, the code will not work. During debugging the code below should stop executing, before it goes to the next line.

**Q #10) What are the distinct types of Error Name Values?**

**Answer:** There are 6 types of values in ‘Error Name’ Property.

| **Error** | **Description** |
| --- | --- |
| Range Error | We will get this error if we use a number outside the range |
| Syntax Error | This error raises when we use the incorrect syntax. (Please refer Ques No: 7) |
| Reference Error | This error is thrown if used an undeclared variable Please refer Ques No: 19 |
| Eval Error | Thrown due to the error in eval(). New JavaScript version doesn’t have this error |
| Type Error | Value is outside the range of types used. Please refer Ques No :22 |
| URI Error | Due to the usage of illegal characters. |

**Q #11) What is JavaScript Hoisting?**

**Answer:** Using ‘JavaScript Hoisting’ method, when an interpreter runs the code, all the variables are hoisted to the top of the original /current scope. If you have a variable declared anywhere inside the JavaScript code then it is brought to the top.

This method is only applicable for the declaration of a variable and is not applicable for initialization of a variable. Functions are also hoisted to the top, whereas function explanations are not hoisted to the top.

Basically, where we declared the variable inside the code doesn’t matter much.

**Q #12) What is JavaScript ‘Strict Mode’?**

**Answer:**‘Strict mode’ is a restricted variant of JavaScript.

Usually, JavaScript is ‘not very strict’ in throwing errors.

But in ‘Strict mode’ it will throw all types of errors, even the silent errors. Thus, the process of debugging becomes easier. And the chances for making mistake for the developer is reduced.

**Q #13) What are the characteristics of JavaScript ‘Strict Mode’?**

**Answer: Given below are the characteristics of JavaScript ‘Strict Mode’:**

* ‘Strict Mode’ will stop developers from creating global variables.
* Developers are restricted from using duplicate parameters.
* Strict mode will restrict you from using JavaScript keyword as a variable name or function name.
* Strict mode is declared with ‘use strict’ keyword at the beginning of the script.
* All browser support strict mode.

**Q #14) What are Self Invoking Functions?**

**Answer:**They are also known as ‘Immediately Invoked Function Expressions’ or ‘Self Executing Anonymous Functions’. These functions are invoked automatically in the code, hence they are named as ‘Self Invoking Functions’.

Usually, we define a function and invoke it, but if we want to execute a function automatically where it is explained, and if we are not going to call it again, we can use anonymous functions. And these types of function have no name.

**Q #15) What is the syntax of ‘Self Invoking Function’? Give an example?**

**Answer:**

**The syntax for the Self-Invoking function:**

(function () {

return () } () ;

Here the last ‘()’ parenthesis in the syntax states that it is a function expression.

**Example of Self Invoked Functions:**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example for Self-Invoking </p>

<p id="dispaly\_num"></p>

<script>

(function (){

elem = document.getElementById("dispaly\_num");

elem.innerHTML = "This function has no name.<br>It is called automatically";

}());

</script>

</body>

</html>

Here the anonymous function is automatically invoked in the code snippet.

The function is used to set the text property of <p> tag having ‘display\_num’ as Id.

**The output of the code snippet here is:**

*This function has no name.*  
*It is called automatically*

**Q #16) In the following Code snippet, can you please predict the output or If you get an error; please explain the error?**

**Answer:**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample : Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example for JavaScript Hoisting </p>

<p id="dispaly\_num"></p>

<script>

first\_num = 100; // Assign value 100 to num

elem = document.getElementById("dispaly\_num");

elem.innerHTML = " Here the variable first\_num:<u> "+first\_num +"</u> is taken to the top <br>" +

"Since second variable is initialised the value is not taken to the top and it's value is "

+ "<u>"+second\_num +"</u> “;

var first\_num; // declaration only

var second\_num =200; // Initialised the variable

</script>

</body>

</html>

Please refer previous Ques no: 11, as explained there, the interpreter will take all the variables declared except initialization to the top.

As per this, ‘first\_num’ variable is taken to the top and ‘second\_num’ variable is initialized with a value, so it is not taken to the top. The is code will not throw an error. But the value of ‘second\_num’ is undefined.

**The output of the code snippet here is:**

*Here the variable first\_num: 100 is taken to the top*  
*Since the second variable is initialized the value is not taken to the top and its value is undefined*

**Q #17) If you need to hide the JavaScript code from the older browser versions, how will you perform it?**

**Answer:**In JavaScript Code, after the <script> tag add ‘<! –’ HTML tag.

This will make the browser to not execute the JavaScript code if it was being an older version of it. Also, after the end </script> tag add ‘//–>’ HTML tag.

This method will help in solving compatibility issues and UI issues to an extent.

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p id="display"></p>

<script> <!--

document.getElementById("display").innerHTML = "Here I am not using an older version of browser.<br>

So the code will work in my browser";

//-->

</script>

</body>

</html>

Here the code snippet after a <script> tag is executed in my browser as I am not using an older version of the browser.

**The output of the code snippet here is:**

*Here I am not using an older version of the browser.*  
*So the code will work in my browser*

**Q #18) In the following Code snippet can you please predict the output or If you get an error, please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Find the output </p>

<p id="display"></p>

<script>

var first\_num =500;

var result= function(){

document.getElementById("display").innerHTML = first\_num;

var first\_num =1000;

}

result();

</script>

</body>

</html>

**Answer:** Here in the code given above, the value of the ‘first\_num’ variable will not be 1000.

In JavaScript, there is no hoisting for variable initialization. The function ‘result ()’ will choose the local variable ‘first\_num’, as it is declared inside the function. Since the variable is declared after it is used, the value of ‘first\_num’ is undefined.

**The output of the code snippet here is:**

*Undefined*

**Q #19) What is the difference between ‘var’ and ‘let’ keyword?**

**Answer: The Differences are as follows:**

| **Var** | **let** |
| --- | --- |
| ’var’ keyword was introduced in JavaScript code from the beginning Stage itself. | ‘let’ keyword is introduced in 2015 only. |
| ’Var’ keyword has function scope. The variable defined with var is available anywhere within the function | A variable declared with ‘let’ keyword has a scope only with in that block. So, let has a Block Scope. |
| The variable declared with ‘var’ be hoisted | The variable declared with ‘let’ be hoisted |

**Q #20) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Find the output </p>

<p id="display\_first"></p>

<p id="display\_second"></p>

<script>

if(true){

var first\_num =1000;

let second\_num=500;

}

document.getElementById("display\_first").innerHTML = "First Number:" + first\_num;

document.getElementById("display\_second").innerHTML = "Second Number:" + second\_num;

</script>

</body>

</html>

**Answer:**

**The output of the code snippet here is:**

*First*Number :*1000*

We will get *‘First*Number :*1000’* as output. There is an ‘Uncaught Reference Error’ error also. In the code snippet, the scope of ‘second\_num’ is only with in the if() block. If a developer tries to access the value outside the block, he will get a ‘Uncaught Reference error’.  
Uncaught Reference Error: second\_num is not defined.

**Q #21) What is the difference between ‘==’ and ‘===’?**

**Answer:**

Both ‘==’ and ‘===’ are comparison operators.

| **‘==’ operator** | **‘===’ operator** |
| --- | --- |
| It is known as ‘Type Converting Operator’ | It is known as ‘Strict Equality Operator’ |
| It compares Value, do not compare type | It compares both value and type. |

**Q #22) What is the difference between ‘let’ and ‘const’?**

**Answer :**

| **let** | **const** |
| --- | --- |
| using ‘let’ we can change the value of variable any number of times | using ‘const’, after the first assignment of the value we cannot redefine the value again |
| Consider the code  { let first\_num =1;  first\_num=2; document. write (first\_num); } Here the code will give an output, since the change in value of first\_num is possible. | Consider the code  { const second\_num =1;  second\_num=2; document. write (second\_num); } Here the code will produce an error, since the ‘second\_num’ is assigned with a second value. |

**Q #23) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example of 'Const' Keyword </p>

<p id="display\_first"></p>

<p id="display\_second"></p>

<script>

let first\_num =500;

first\_num=501;

document.getElementById("display\_first").innerHTML = "First Number:"+ first\_num ;

const second\_num =1000;

second\_num=1001;

document.getElementById("display\_second").innerHTML = "Second Number :"+second\_num;

</script>

</body>

</html>

**Answer: Please refer Ques No: 21 before reading further**

**The output of the code snippet is:**

*First Number:501*

We will also get an Error while running the code, as we are trying to change the value of a ‘const’ variable.

**Error:** *Uncaught TypeError: Assignment to constant variable.*

**Q #24) What is the difference between ‘null’ and ‘undefined’?**

**Answer:**Both the keywords represent empty values**.**

**The differences are:**

* In ‘undefined’, we will define a variable, but we won’t assign a value to that variable. On the other hand, in ‘null’ we will define a variable and assign the ‘null’ value to the variable.
* type of (undefined) and type of (null) object.

**Q #25) What is the difference between ‘function declaration’ and ‘function expression’?**

**Answer:**

**It can be explained with an Example:**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example Function Declaration</p>

<p id="display\_add"></p>

<p id="display\_sub"></p>

<script>

function add(first\_num,second\_num){

return first\_num + second\_num;

}

var substract = function sub(first\_num,second\_num){

return first\_num - second\_num;

}

var first\_num=700;

var second\_num=300;

document.getElementById("display\_add").innerHTML = "Sum of the number is:" + add(first\_num,second\_num);

document.getElementById("display\_sub").innerHTML = "Difference of the number is:" + substract(first\_num,second\_num);

</script>

</body>

</html>

As shown in the example add() is a function declaration and subtract() is a function expression. The syntax of function declaration is like a function which is saved into a variable.

Function declarations are hoisted but function expressions are not hoisted.

**Q #26) What are ‘settimeout()’?**

**Answer:**(It better to explain this answer with an example)

Consider the code snippet

Console.log (‘First Line’);

Console.log (‘Second Line’);

Console.log (‘Third Line’);

**The output of the code snippet here is:**

*First Line*  
*Second Line*  
*Third Line*

Now you introduce settimeout() method and wrap the same set of code in it.

Settimeout(function() {

Console.log (‘First Line’);

},0);

Console.log (‘Second Line’);

Console.log (‘Third Line’);

**The output of the code snippet here is:**

*Second Line*  
*Third Line*  
*First Line*

With the introduction of settimeout(), the processes become asynchronous. The first statements to be placed in the stack is Console.log (‘Second Line’), and Console.log (‘Third Line’), and they will get executed first. You need to wait until everything in the stack is completed first.

Even though ‘0’ is the timeout period, it doesn’t mean that it will be executed right away.

**Q #27) What is a Closure and How do you use it?**

**Answer:**A closure is an inner function. It can access the outer variables of a function. In Closure, within function\_1 there is another function\_2 which returns ‘A’ value and function\_1 also returns a value; say ‘B’.

Here sum() is the outer function and add () is an inner function, it can access all the variables including ‘first\_num’ ‘second\_num’ and ‘third\_num’. The outer function is calling the inner function add().

<script>

// To find the sum of two numbers using closure method

function sum( first\_num, second\_num )

{

var sumStr= 600;

function add(first\_num , second\_num)

{

return (sumStr + (first\_num + second\_num));

}

return add();

}

document.write("Result is :"+ sum(150,350));

</script>

**The output of the code snippet here is:**

*Result is: 500*

**Q #28) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example Assignmnet Statement</p>

<p id="display"></p>

<script>

var x =500;

let y,z,p,q;

q=200;

if(true){

x=y=z=p=q;

document.getElementById("display").innerHTML = "x="+ x + "<br>y :"+ y +"<br>z :"+ z+"<br>p :"+ p+"<br>q :"+ q;

}

</script>

</body>

</html>

**Answer:** The assignment statements are considered from Right to left.

**The output of the code snippet here is:**

*x=200*  
*y:200*  
*z:200*  
*p:200*  
*q:200*

**Q #29) Can you give an example where the code snippet shows the difference between test () and exec () methods?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample : Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example for exec() methods </p>

<p>Click the button to search for a pattern "How“ in the given string "Hello. Good Morning. How do you feel today?"</p>

<p>If the "How" is found, the method will return the pattern </p>

<button onclick="searchTxt()">Search</button>

<p id="result"></p>

<script>

function searchTxt() {

var str = "Hello. Good Morning. How do you feel today?";

var search\_patt = new RegExp("How");

var res = search\_patt.exec(str);

document.getElementById("result").innerHTML ="Found the pattern :"+ res;

}

</script>

</body>

</html>

**Answer:**This is an example of the test () and exec () method, Refer Ques No: 5 for more details.

**The output of the code snippet here is:**

*Found the pattern using exec (): How*  
*Using test () the result is: true*

**Q #30) Can you give an example showing JavaScript Hoisting?**

**Answer:**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example for JavaScript Hoisting </p>

<p id="dispaly\_num"></p>

<script>

num = 100; // Assign value 100 to num

elem = document.getElementById("dispaly\_num");

elem.innerHTML = "Here the variables are used before declaring it." +

" <br>The value of the variable is " + num;

var num; // Declare the varaible </script>

</body>

</html>

Please refer Ques No:11 for more details

Here the variable ‘num’ is used before declaring it. But JavaScript Hoisting will allow it.

**The output of the code snippet here is:**

*Here the variables are used before declaring it.*  
*The value of the variable is 100*

**Q #31) Can you give an example showing the use of ‘debugger’ keyword in the JavaScript code?**

**Answer:**

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'> Example for debug keyword </p>

<p> Here to test the code, debugger must be enabled for the browser, <br>during debugging the code below should stop executing before it goes to the next line. </p>

<p id="wait\_result"></p>

<p id="show\_result"></p>

<script>

var a = 1000;

var b = 500;

var sum = a + b;

document.getElementById("wait\_result").innerHTML = "Adding numbers......<br>Select 'Resume Script execution' to continue: ";

debugger;

document.getElementById("show\_result").innerHTML = "Sum of the numbers : "+sum;

</script>

</body>

</html>

**Note:** The debugger must be enabled for the browser to test the code. Refer Ques No: 5 for more details

This is an example of debugging keyword (Browser used: Chrome)

**The output of the code snippet here is:**

*Here to test the code, the debugger must be enabled for the browser,*  
*during debugging the code below should stop executing before it goes to the next line.*  
*Adding numbers…*  
*Select ‘Resume Script execution' to continue:*

*<Click on ‘Resume Script execution’ Button>*

*Sum of the numbers: 1500*

**Q #32) In the following Code snippet can you please predict the output or If you get an error; please explain the error?**

<!DOCTYPE html>

<html>

<body>

<h2> <strong> Sample: Software Testing Help</strong> </h2>

<p style='text-decoration:underline'>Example Type Converting </p>

<p id="display"></p>

<script>

var first\_num =500;

var first\_name='500';

if(first\_num == first\_name){

document.getElementById("display").innerHTML = "Comparison will return 'true' by Type converting Operator ";

}

</script>

</body>

</html>

**Answer:**Consider the code

If (‘100’==100) {

document. write (“It’s a Type Converting Operator”);

}

Here

  typeof(‘100’) is string

   typeof(100) is number

the ‘==’ operator will convert the number type, which is on the right side of the operator to string and compare both values

**The output of the code snippet here is:**

*Comparison will return ‘true' by Type converting Operator*

**Question 1**

**1. What is the difference between undefined and not defined in JavaScript?**

In JavaScript, if you try to use a variable that doesn't exist and has not been declared, then JavaScript will throw an error var name is not defined and script will stop executing. However, if you use typeof undeclared\_variable, then it will return undefined.

Before getting further into this, let's first understand the difference between declaration and definition.

Let's say var x is a declaration because you have not defined what value it holds yet, but you have declared its existence and the need for memory allocation.

> **var** x; *// declaring x*

> console.log(x); *//output: undefined*

Here var x = 1 is both a declaration and definition (also we can say we are doing an initialisation). In the example above, the declaration and assignment of value happen inline for variable x. In JavaScript, every variable or function declaration you bring to the top of its current scope is called hoisting.

The assignment happens in order, so when we try to access a variable that is declared but not defined yet, we will get the result undefined.

**var** x; *// Declaration*

**if**(**typeof** x === 'undefined') *// Will return true*

If a variable that is neither declared nor defined, when we try to reference such a variable we'd get the result not defined.

> console.log(y); *// Output: ReferenceError: y is not defined*

**Question 2**

**What will be the output of the code below?**

**var** y = 1;

**if** (**function** **f**(){}) {

y += **typeof** f;

}

console.log(y);

The output would be 1undefined. The if condition statement evaluates using eval, so eval(function f(){}) returns function f(){} (which is true). Therefore, inside the if statement, executing typeof f returns undefined because the if statement code executes at run time, and the statement inside the if condition is evaluated during run time.

**var** k = 1;

**if** (1) {

eval(**function** **foo**(){});

k += **typeof** foo;

}

console.log(k);

The code above will also output 1undefined.

**var** k = 1;

**if** (1) {

**function** **foo**(){};

k += **typeof** foo;

}

console.log(k); *// output 1function*

**Question 3**

**What is the drawback of creating true private methods in JavaScript?**

One of the drawbacks of creating true private methods in JavaScript is that they are very memory-inefficient, as a new copy of the method would be created for each instance.

**var** Employee = **function** (name, company, salary) {

**this**.name = name || ""; *//Public attribute default value is null*

**this**.company = company || ""; *//Public attribute default value is null*

**this**.salary = salary || 5000; *//Public attribute default value is null*

*// Private method*

**var** increaseSalary = **function** () {

**this**.salary = **this**.salary + 1000;

};

*// Public method*

**this**.dispalyIncreasedSalary = **function**() {

increaseSlary();

console.log(**this**.salary);

};

};

*// Create Employee class object*

**var** emp1 = **new** Employee("John","Pluto",3000);

*// Create Employee class object*

**var** emp2 = **new** Employee("Merry","Pluto",2000);

*// Create Employee class object*

**var** emp3 = **new** Employee("Ren","Pluto",2500);

Here each instance variable emp1, emp2, emp3 has its own copy of the increaseSalary private method.

So, as a recommendation, don’t use private methods unless it’s necessary.

**Question 4**

**What is a “closure” in JavaScript? Provide an example**

A closure is a function defined inside another function (called the parent function), and has access to variables that are declared and defined in the parent function scope.

The closure has access to variables in three scopes:

* Variables declared in their own scope
* Variables declared in a parent function scope
* Variables declared in the global namespace

**var** globalVar = "abc";

*// Parent self invoking function*

(**function** **outerFunction** (outerArg) { *// begin of scope outerFunction*

*// Variable declared in outerFunction function scope*

**var** outerFuncVar = 'x';

*// Closure self-invoking function*

(**function** **innerFunction** (innerArg) { *// begin of scope innerFunction*

*// variable declared in innerFunction function scope*

**var** innerFuncVar = "y";

console.log(

"outerArg = " + outerArg + "\n" +

"outerFuncVar = " + outerFuncVar + "\n" +

"innerArg = " + innerArg + "\n" +

"innerFuncVar = " + innerFuncVar + "\n" +

"globalVar = " + globalVar);

}*// end of scope innerFunction)(5); // Pass 5 as parameter*

}*// end of scope outerFunction )(7); // Pass 7 as parameter*

innerFunction is closure that is defined inside outerFunction and has access to all variables declared and defined in the outerFunction scope. In addition, the function defined inside another function as a closure will have access to variables declared in the global namespace.

Thus, the output of the code above would be:

outerArg = 7

outerFuncVar = x

innerArg = 5

innerFuncVar = y

globalVar = abc

**Question 5**

**Write a mul function which will produce the following outputs when invoked:**

console.log(mul(2)(3)(4)); *// output : 24*

console.log(mul(4)(3)(4)); *// output : 48*

Below is the answer followed by an explanation to how it works:

**function** **mul** (x) {

**return** **function** (y) { *// anonymous function*

**return** **function** (z) { *// anonymous function*

**return** x \* y \* z;

};

};

}

Here the mul function accepts the first argument and returns an anonymous function, which takes the second parameter and returns another anonymous function that will take the third parameter and return the multiplication of the arguments that have been passed.

In JavaScript, a function defined inside another one has access to the outer function's variables. Therefore, a function is a first-class object that can be returned by other functions as well and be passed as an argument in another function.

* A function is an instance of the Object type
* A function can have properties and has a link back to its constructor method
* A function can be stored as a variable
* A function can be pass as a parameter to another function
* A function can be returned from another function

**Question 6**

**How to empty an array in JavaScript?**

For instance,

**var** arrayList = ['a','b','c','d','e','f'];

**How can we empty the array above?**

There are a couple ways we can use to empty an array, so let's discuss them all.

**Method 1**

arrayList = []

Above code will set the variable arrayList to a new empty array. This is recommended if you don't have **references to the original array** arrayListanywhere else, because it will actually create a new, empty array. You should be careful with this method of emptying the array, because if you have referenced this array from another variable, then the original reference array will remain unchanged.

For Instance,

**var** arrayList = ['a','b','c','d','e','f']; *// Created array*

**var** anotherArrayList = arrayList; *// Referenced arrayList by another variable*

arrayList = []; *// Empty the array*

console.log(anotherArrayList); *// Output ['a','b','c','d','e','f']*

**Method 2**

arrayList.length = 0;

The code above will clear the existing array by setting its length to 0. This way of emptying the array also updates all the reference variables that point to the original array. Therefore, this method is useful when you want to update all reference variables pointing to arrayList.

For Instance,

**var** arrayList = ['a','b','c','d','e','f']; *// Created array*

**var** anotherArrayList = arrayList; *// Referenced arrayList by another variable*

arrayList.length = 0; *// Empty the array by setting length to 0*

console.log(anotherArrayList); *// Output []*

**Method 3**

arrayList.splice(0, arrayList.length);

The implementation above will also work perfectly. This way of emptying the array will also update all the references to the original array.

**var** arrayList = ['a','b','c','d','e','f']; *// Created array*

**var** anotherArrayList = arrayList; *// Referenced arrayList by another variable*

arrayList.splice(0, arrayList.length); *// Empty the array by setting length to 0*

console.log(anotherArrayList); *// Output []*

**Method 4**

**while**(arrayList.length){

arrayList.pop();

}

The implementation above can also empty arrays, but it is usually not recommended to use this method often.

**Question 7**

**How do you check if an object is an array or not?**

The best way to find out whether or not an object is an instance of a particular class is to use the toString method from Object.prototype:

**var** arrayList = [1,2,3];

One of the best use cases of type-checking an object is when we do method overloading in JavaScript. For example, let's say we have a method called greet, which takes one single string and also a list of strings. To make our greet method workable in both situations, we need to know what kind of parameter is being passed. Is it a single value or a list of values?

**function** **greet**(param){

**if**(){ *// here have to check whether param is array or not*

}**else**{

}

}

However, as the implementation above might not necessarily check the type for arrays, we can check for a single value string and put some array logic code in the else block. For example:

**function** **greet**(param){

**if**(**typeof** param === 'string'){

}**else**{

*// If param is of type array then this block of code would execute*

}

}

Now it's fine we can go with either of the aforementioned two implementations, but when we have a situation where the parameter can be single value, array, and object type, we will be in trouble.

Coming back to checking the type of an object, as mentioned previously we can use  
Object.prototype.toString

**if**( Object.prototype.toString.call( arrayList ) === '[object Array]' ) {

console.log('Array!');

}

If you are using jQuery, then you can also use the jQuery isArray method:

**if**($.isArray(arrayList)){

console.log('Array');

}**else**{

console.log('Not an array');

}

FYI, jQuery uses Object.prototype.toString.call internally to check whether an object is an array or not.

In modern browsers, you can also use

Array.isArray(arrayList);

Array.isArray is supported by Chrome 5, Firefox 4.0, IE 9, Opera 10.5 and Safari 5

**Question 8**

**What will be the output of the following code?**

**var** output = (**function**(x){

**delete** x;

**return** x;

})(0);

console.log(output);

The output would be 0. The delete operator is used to delete properties from an object. Here x is not an object but a **local variable**. delete operators don't affect local variables.

**Question 9**

**What will be the output of the following code?**

**var** x = 1;

**var** output = (**function**(){

**delete** x;

**return** x;

})();

console.log(output);

The output would be 1. The delete operator is used to delete the property of an object. Here x is not an object, but rather it's the **global variable** of type number.

**Question 10**

**What will be the output of the code below?**

**var** x = { foo : 1};

**var** output = (**function**(){

**delete** x.foo;

**return** x.foo;

})();

console.log(output);

The output would be undefined. The delete operator is used to delete the property of an object. Here, x is an object which has the property foo, and as it is a self-invoking function, we will delete the foo property from object x. After doing so, when we try to reference a deleted property foo, the result isundefined.

**Question 11**

**What will be the output of the code below?**

**var** Employee = {

company: 'xyz'

}

**var** emp1 = Object.create(Employee);

**delete** emp1.company

console.log(emp1.company);

The output would be xyz. Here, emp1 object has company as its **prototype** property. The delete operator doesn't delete prototype property.

emp1 object doesn't have **company** as its own property. You can test it console.log(emp1.hasOwnProperty('company')); //output : false. However, we can delete the company property directly from theEmployee object using delete Employee.company. Or, we can also delete the emp1 object using the \_\_proto\_\_property delete emp1.\_\_proto\_\_.company.

**Question 12**

**What is undefined x 1 in JavaScript?**

**var** trees = ["redwood","bay","cedar","oak","maple"];

**delete** trees[3];

When you run the code above and type console.log(trees); into your Chrome developer console, you will get  
["redwood", "bay", "cedar", undefined × 1, "maple"]. When you run the code in Firefox's browser console, you will get ["redwood", "bay", "cedar", undefined, "maple"]. Thus, it's clear that the Chrome browser has its own way of displaying uninitialised indexes in arrays. However, when you check trees[3] === undefined in both browsers, you will get similar output as true.

**Note:** Please remember you do not need to check for the uninitialised index of array in trees[3] === 'undefined × 1', as it will give you an error. 'undefined × 1' is just way of displaying an array's uninitialised index in Chrome.

**Question 13**

**What will be the output of the code below?**

**var** trees = ["xyz","xxxx","test","ryan","apple"];

**delete** trees[3];

console.log(trees.length);

The output would be 5. When we use the delete operator to delete an array element, the array length is not affected from this. This holds even if you deleted all elements of an array using the delete operator.

In other words, when the delete operator removes an array element, that deleted element is not longer present in array. In place of value at deleted index undefined x 1 in **chrome** and undefined is placed at the index. If you do console.log(trees)output ["xyz", "xxxx", "test", undefined × 1, "apple"] in Chrome and in Firefox ["xyz", "xxxx", "test", undefined, "apple"].

**Question 14**

**What will be the output of the code below?**

**var** bar = true;

console.log(bar + 0);

console.log(bar + "xyz");

console.log(bar + true);

console.log(bar + false);

The code will output 1, "truexyz", 2, 1. Here's a general guideline for addition operators:

* Number + Number -> Addition
* Boolean + Number -> Addition
* Boolean + Number -> Addition
* Number + String -> Concatenation
* String + Boolean -> Concatenation
* String + String -> Concatenation

**Question 15**

**What will be the output of the code below?**

**var** z = 1, y = z = **typeof** y;

console.log(y);

The output would be undefined. According to the associativity rule, operators with the same precedence are processed based on the associativity property of the operator. Here, the associativity of the assignment operator is Right to Left, so typeof y will evaluate first , which is undefined. It will be assigned to z, and then ywould be assigned the value of z and then z would be assigned the value 1.

**Question 16**

**What will be the output of the code below?**

*// NFE (Named Function Expression*

**var** foo = **function** **bar**(){ **return** 12; };

**typeof** bar();

The output would be Reference Error. To make the code above work, you can re-write it as follows:

**Sample 1**

**var** bar = **function**(){ **return** 12; };

**typeof** bar();

or

**Sample 2**

**function** **bar**(){ **return** 12; };

**typeof** bar();

A function definition can have only one reference variable as its function name. In **sample 1**, bar's reference variable points to anonymous function. In **sample 2**, the function's definition is the name function.

**var** foo = **function** **bar**(){

*// foo is visible here*

*// bar is visible here*

console.log(**typeof** bar()); *// Work here :)*

};

*// foo is visible here*

*// bar is undefined here*

**Question 17**

**What is the difference between the function declarations below?**

**var** foo = **function**(){

*// Some code*

};

**function** **bar**(){

*// Some code*

};

The main difference is the function foo is defined at run-time whereas function bar is defined at parse time. To understand this in better way, let's take a look at the code below:

Run-Time **function** **declaration**

<**script**>

**foo**(); // **Calling** **foo** **function** **here** **will** **give** **an** **Error**

**var** **foo** = **function**(){

console.log("Hi I am inside Foo");

};

</script>

<script>

Parse-Time **function** **declaration**

**bar**(); // **Calling** **foo** **function** **will** **not** **give** **an** **Error**

**function** **bar**(){

console.log("Hi I am inside Foo");

};

</script>

Another advantage of this first-one way of declaration is that you can declare functions based on certain conditions. For example:

<script>

**if**(testCondition) {*// If testCondition is true then*

**var** foo = **function**(){

console.log("inside Foo with testCondition True value");

};

}**else**{

**var** foo = **function**(){

console.log("inside Foo with testCondition false value");

};

}

</script>

However, if you try to run similar code using the format below, you'd get an error:

<script>

**if**(testCondition) {*// If testCondition is true then*

**function** **foo**(){

console.log("inside Foo with testCondition True value");

};

}**else**{

**function** **foo**(){

console.log("inside Foo with testCondition false value");

};

}

</script>

**Question 18**

**What is function hoisting in JavaScript?**

**Function Expression**

**var** foo = **function** **foo**(){

**return** 12;

};

In JavaScript, variable and functions are hoisted. Let's take function hoisting first. Basically, the JavaScript interpreter looks ahead to find all variable declarations and then hoists them to the top of the function where they're declared. For example:

foo(); *// Here foo is still undefined*

**var** foo = **function** **foo**(){

**return** 12;

};

Behind the scene of the code above looks like this:

**var** foo = undefined;

foo(); *// Here foo is undefined*

foo = **function** **foo**(){

/ Some code stuff

}

**var** foo = undefined;

foo = **function** **foo**(){

/ Some code stuff

}

foo(); *// Now foo is defined here*

**Question 19**

**What will be the output of code below?**

**var** salary = "1000$";

(**function** () {

console.log("Original salary was " + salary);

**var** salary = "5000$";

console.log("My New Salary " + salary);

})();

The output would be undefined, 5000$. Newbies often get tricked by JavaScript's hoisting concept. In the code above, you might be expecting salary to retain its value from the outer scope until the point that salary gets re-declared in the inner scope. However, due to hoisting, the salary value was undefined instead. To understand this better, have a look of the code below:

**var** salary = "1000$";

(**function** () {

**var** salary = undefined;

console.log("Original salary was " + salary);

salary = "5000$";

console.log("My New Salary " + salary);

})();

salary variable is hoisted and declared at the top in the function's scope. The console.log inside returns undefined. After the console.log, salary is redeclared and assigned 5000$.

**Question 20**

**What is the instanceof operator in JavaScript? What would be the output of the code below?**

**function** **foo**(){

**return** foo;

}

**new** foo() **instanceof** foo;

Here, instanceof operator checks the current object and returns true if the object is of the specified type.

For Example:

**var** dog = **new** Animal();

dog **instanceof** Animal *// Output : true*

Here dog instanceof Animal is true since dog inherits from Animal.prototype.

**var** name = **new** String("xyz");

name **instanceof** String *// Output : true*

Here name instanceof String is true since dog inherits from String.prototype. Now let's understand the code below:

**function** **foo**(){

**return** foo;

}

**new** foo() **instanceof** foo;

Here function foo is returning foo, which again points to function foo.

**function** **foo**(){

**return** foo;

}

**var** bar = **new** foo();

*// here bar is pointer to function foo(){return foo}.*

So the new foo() instanceof foo return false;

[Ref Link](http://stackoverflow.com/questions/2449254/what-is-the-instanceof-operator-in-javascript)

**Question 21**

**If we have a JavaScript associative array**

**var** counterArray = {

A : 3,

B : 4

};

counterArray["C"] = 1;

**How can we calculate the length of the above associative array's counterArray?**

There are no in-built functions and properties available to calculate the length of associative array object here. However, there are other ways by which we can calculate the length of an associative array object. In addition to this, we can also extend an Object by adding a method or property to the prototype in order to calculate length. However, extending an object might break enumeration in various libraries or might create cross-browser issues, so it's not recommended unless it's necessary. Again, there are various ways by which we can calculate length.

Object has the keys method which can be used to calculate the length of an object:

**We** can also calculate the length **of** an **object** by iterating through an **object** **and** by counting the **object**'s own property.

```javascript

function getSize(**object**){

**var** count = 0;

**for**(key **in** **object**){

// hasOwnProperty **method** check own property **of** **object**

**if**(**object**.hasOwnProperty(key)) count++;

}

**return** count;

}

We can also add a length method directly on Object:

Object.length = **function**(){

**var** count = 0;

**for**(key **in** object){

*// hasOwnProperty method check own property of object*

**if**(object.hasOwnProperty(key)) count++;

}

**return** count;

}

*//Get the size of any object using*

console.log(Object.length(counterArray))

**Bonus**: We can also use Underscore (recommended, As it's lightweight) to calculate object length.

[1. What is the use of let & const in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled1)

In modern javascript let & const are different ways of creating variables. Earlier in javascript, we use the var keyword for creating variables. let & const keyword is introduced in version [ES6](https://www.onlineinterviewquestions.com/es6-interview-questions/)with the vision of creating two different types of variables in javascript one is immutable and other is mutable.  
**const:** It is used to create an immutable variable. Immutable variables are variables whose value is never changed in the complete life cycle of the program.  
**let:** let is used to create a mutable variable. Mutable variables are normal variables like var that can be changed any number of time.

[2. What are the different types of errors available in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled2)

There are three types of errors available in JavaScript

* **Load time errors**: Errors which come up when loading a web page like improper syntax errors are known as Load-time errors and it generates the errors dynamically.
* **Run time errors**: Errors that come due to misuse of the command inside the HTML language.
* **Logical Errors**: These are the errors that occur due to the bad logic performed on a function which is having a different operation.

[3. How to get an element by class in JavaScript ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled3)

**document.getElementsByClassName()** method is used in Javascript to get an element with a class name.

|  |  |
| --- | --- |
| getElementsByClassName() | |
| **Method Name** | getElementsByClassName |
| **Syntax** | document.getElementsByClassName('className') |
| **Parameter** | String (name of class) |
| **Output** | Array of HTMLCollection that have inputted className |

[4. What are different types of Scope Chain available in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled4)

If we check in the program, every local scope has a connection with one or more scope in their back which forms a chain. This chain goes on until it met with the global scope which is the root of this hierarchy. As global scope doesn't have a parent, so it is on the top of the chain. This is known as scope chain.

The scope chain in JavaScript is basically used to resolve the values of the variable. Without this, it is difficult for a JavaScript to choose a certain value for a variable if there are many variables defined at different scopes.

[5. Explain MUL function in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled5)

MUL means simple multiplication of numbers. It is a techique in which you pass a one value as an argument in a function and that function returns another function to which you pass the second value and the process go on. For example: x\*y\*z can be representing as:

function mul (x) {

return function (y) { // anonymous function

return function (z) { // anonymous function

return x \* y \* z; };

};

}

[6. Write a program to reverse a string in pure JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled6)

There are many ways to reverse a string in JavaScript. These are:

**Using in-built functions:** the inbuilt function reverse() reverses the string directly. Here’ how:

str="jQuery";

str = str.split(""); //convert 'jQuery' to array

str = str.reverse(); //reverse 'jQuery' order

str = str.join(""); //then combines the reverse order values.

alert(str);

First split the string to an array, then reverse an array and after that join the characters to form a string.

**Using a loop:** First, count a number of characters in a string, then apply a decrementing loop on an original string which starts from the last character and prints each character until count becomes zero.

[7. How to redirect a page to another page in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled7)

There are several ways to redirect page to another page in JavaScript. These are:

1. **Using location.href:** It is the first approach to redirect page. In this, we can go back to access the original document.**Syntax**:window.location.href =“https://www.onlineinterviewquestions.com/”
2. **Using location.replace:** Another approach to redirect page. In this, it is not possible to navigate back to the original document by clicking on the back button as it removes the URL of the original document. **Syntax:** window.location.replace(" https://www.onlineinterviewquestions.com/;");

[8. List some Design patterns in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled8)

The design pattern is a general reusable solution to a commonly occurring problem in software design. Some of the design patterns are:

1. **Creational design pattern:** These patterns dealt with the mechanism of object creation which optimize object creation with the basic approach.
2. **Structural design pattern:** these patterns deal with different classes and objects to provide new functionality.
3. **Behavioral Patterns:** These patterns are to improve communication between objects and to recognize patterns.
4. **Concurrency design patterns:** These patterns handle with multi-thread programming paradigms.
5. **Architectural design patterns:** These patterns used to deal with architectural designs.

[9. What is difference between Array.splice() and Array.slice() method in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled9)

* The array.slice() removes items from the array and then return those removed items as an array whereas array.slice() method is selected items from an array and then those elements as a new array object.
* The splice() method affects the original array whereas slice() method doesn’t affect the original array.
* Splice() method takes n number of arguments whereas slice() can take only two arguments.

Syntax of splice(): array.splice(index, howmany, item1, ....., itemX)

Syntax of slice(): array.slice(start, end)

[10. How to add/remove properties to object dynamically in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled10)

You can add a property to an object using object.property\_name =value, delete object.property\_name is used to delete a property.

**Example:**

let user = new Object();

// adding a property

user.name='Anil';

user.age =25;

console.log(user);

delete user.age;

console.log(user);

[11. Explain Promise in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled11)

A promise is an object in JavaScript which is used to produce a value that may give result in the future. The value can be resolved value or it can be a reason which tells why the value is not resolved.

A promise can be of three states:

* Fulfilled: The operation is completed and the promise has a specific value.
* Rejected: The operation is failed and promise has a reason which shows why the operation failed.
* Pending: Th operation is not fulfilled or rejected, means it has not completed yet.

[12. How to remove duplicates from JavaScript Array?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled12)

There are many ways to remove duplicates from JavaScript array. These are described below with examples:

**1. By using Set**: It is the simplest approach to remove duplicates. Set is an inbuilt object to store unique values in an array. Here's how we use set:

function uniquearray(array) {

let unique\_array= Array.from(set(array))

return unique\_array;}

As in the above code, you created a set of an array which automatically eliminates the duplicate values.

**2. By using Filter**: Another approach to remove duplicates from an array is applying filter on an array. To call filter method, it requires three arguments: array, current element, index of current element. Here’s how we use filter:

function unque\_array (arr){

let unique\_array = arr.filter(function(elem, index, self) {

return index == self.indexOf(elem); }

return unique\_array }

console.log(unique\_array(array\_with\_duplicates));

**3. By using for loop**: In this, we can use for loop to remove duplicates. In this we make an empty array in which those elements will be added from the duplicate array which are not present in this before. Thus, finally we will get an array which has unique elements. Code to implement this:

Array dups\_names = ['Ron', 'Pal', 'Fred', 'Rongo', 'Ron'];

**function** dups\_array(dups\_names) {

**let** unique = {};

names.forEach(**functio**n(i) {

**If** (!unique[i]) {

unique[i] = **true**; }

});

**return** Object.keys(unique);} // Ron, Pal, Fred, Rongo

Dups\_array(names);

These are the main three methods used in JavaScript to get a unique array.

[13. Explain few difference between null, undefined or undeclared JavaScript variable?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled13)

**Null** is a value that can be assigned to a variable or an object.

**Undefined** means a variable has been declared but no value is assigned to it. This type of variable is declared itself to be undefined.

**Undeclared** means the variable has declared without any datatype.

Null, Undefined are primitive data types whereas Undeclared is not a primitive data type.

[14. List few difference between primitive and non primitive JavaScript data types?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled14)

* The primitive data types are numbers, strings, Boolean, undefined, null and anything other than these data types are known as non-primitive such as objects and functions.
* Primitive data types are immutable while non-primitives are mutable.
* Primitives are known immutable as they can't be changed once they created but non-primitive are changeable, means once an object is created, it can be changed.
* Primitives data types are compared with their values, it means two values are strictly equal if they have the same data type and holds the same value.
* Non-primitives are not compared with values. For example, if two objects have the same properties and values, they are strictly not equal.

[15. How to add a new property in existing function JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled15)

It is easy to add a new property in existing function by just giving value to the existing function it. For example, let we have an existing object person, to give new property check the below code:

person.country= “India”;

The new property “country” has added to the object person.

[16. What is difference between deep and shallow object coping in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled16)

Some differences are:

* Deep copy means copies all values or properties recursively in the new object whereas shallow copy copies only the reference.
* In a deep copy, changes in the new object don't show in original object whereas, in shallow copy, changes in new objects will reflect in the original object.
* In a deep copy, original objects do not share the same properties with new object whereas, in shallow copy, they do.

[17. How to call a function in every x seconds in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled17)

In JavaScript, we use the function setInterval() to call any function in every x seconds.

**Syntax:** setInterval(function, milliseconds, param1, param2, ...)

**Function:** it is a required parameter which includes the function to be execute.

**Milliseconds:** required parameter which tells how often the function will execute.

Others are an additional parameter.

**For example:** setInterval(function (){ alert("Hello"); }, 3000);

In the above example, this function calls hello function in very 3 seconds.

[18. Explain spread operator in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled18)

The spread operator expands an expression in places where multiple argument/variables/elements are needed to present. It represents with three dots (…).

For example:

var mid = [3, 4];

var newarray = [1, 2, ...mid, 5, 6];

console.log(newarray);

// [1, 2, 3, 4, 5, 6]

In above example, instead of appending mid array, it rather expands in the newarray with the help of spread operator. This is how spread operator works in JavaScript.

[19. How host objects are different from native objects in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled19)

**Host objects**: These are those objects which environment gives. It means they are different for different environments. For example, browsers include objects such as windows but Node.js environments give objects such as Node List.

**Native Objects**: these are built-in objects in JavaScript. They are also known as Global Objects because they will be available to you independent of ay environment if you working in JavaScript.

[20. Explain higher-order functions in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled20)

Higher order function is the best feature of functional programming available in JavaScript. It is the function which takes a function as an argument and returns a function as a result. Some of the inbuilt higher-order functions are mapping, filtering, reduction, zipping, etc.

[21. Please explain equality operators in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled21)

[22. What are anonymous functions in JavaScript ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled22)

[23. What is difference between local and global scope in JavaScript ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled23)

[24. What is use of settimeout function in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled24)

[25. List few advantages of using JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled25)

Few advantage og Javascript

* Javascript is executed on user's computer, the meaning is that whatever you do in Javascript will not add any processing strain on the server. and that's why it is called as the client-side programming language. And this feature makes your sites responsive for the end user and less expensive for you in terms of server traffic.
* With the help of Javascript, you can create highly responsive interfaces which will improve the user experience and provide dynamic functionality, without waiting for the server to show another page.
* If you want to make online systems available offline and sync automatically once the computer goes online, then Javascript is the best technology you can use. you can do this using the right browser add-ons (Such as Google or Yahoo Browser Plus).
* Content loading and changing it dynamically. Using Ajax in Javascript you can load content into the document if and when the user needs it, without reloading the entire page.
* Using the Principles of unobtrusive JavaScript(defensive Scripting), JavaScript can test for what is possible in your browser and react accordingly.

[26. List few Difference between JAVA and JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled26)

[27. Is JavaScript multi-threaded or single-threaded?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled27)

JavaScript is single-threaded.

[28. Is it possible to do 301 redirects in Javascript ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled28)

JavaScript entirely runs on the client machine. 301 is response code that is sent by the server as a response. So it is not possible to do 301 Redirects In JavaScript.

[29. How to get inner Html of an element in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled29)

InnerHTML property of HTML DOM is used to get inner Html of an element in JavaScript.

**Example Usage:**

This is inner Element

<script type="text/javascript">

var inner= document.getElementById("inner").innerHTML ;

console.log(inner); // This is inner Element

document.getElementById("inner").innerHTML = "Html changed!";

var inner= document.getElementById("inner").innerHTML ;

console.log(inner); // Html changed!

</script>

[30. Explain Event bubbling and Event Capturing in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled30)

**Event Capture and Bubbling**: In HTML DOM API there are two ways of event propagation and determines the order in which event will be received. The two ways are Event Bubbling and Event Capturing. The first method event bubbling directs the event to its intended target, and the second is called event capture in which the event goes down to the element.

**Event Capture**

The capture procedure is rarely used but when it’s used it proves to be very helpful. This process is also called ‘trickling’. In this process, the event is captured first by the outermost element and then propagated to the innermost element. For example:

<div>

<ul>

<li></li>

</ul>

</div>

From the above example, suppose the click event did occur in the ‘li’ element, in that case capturing event it will be first handled ‘div’, then ‘ul’ and at last the target element will be hit that is ‘li’

**Event Bubbling**

Bubbling just works like the bubbles, the event gets handled by the innermost element and then propagated to the outer element.

<div>

<ul>

<li></li>

</ul>

</div>

From the above example, suppose the click event did occur in the ‘li’ element in bubbling model the event will be handled first by ‘li’ then by ‘ul’ and at last by ‘div’ element.

[31. How to import all exports of a file as an object.](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled31)

import \* as object name from ‘./file.js’ is used to import all exported members as an object. You can simply access the exported variables or methods using dot (.) operator of the object.

Example:

objectname.member1;

objectname.member2;

objectname.memberfunc();

[32. Explain Arrow functions?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled32)

An arrow function is a consise and short way to write function expressions in Es6 or above.A rrow functions cannot be used as constructors and also does not supports this, arguments, super, or new.target keywords. It is best suited for non-method functions. In general an arrow function looks like const function\_name= ()=>{}

const greet=()=>{console.log('hello');}

greet();

[33. Explain function hoisting in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled33)

JavaScript’s default behavior that allows moving declarations to the top is called Hoisting. The 2 ways of creating functions in JavaScript are **Function Declaration** and **Function Expression**. Let’s find out more about these:

***Function Declaration***

A function with the specific parameters is known as function declarations. To create a variable in JavaScript is called declarations.

**e.g:**

hoisted(); // logs "foo"

function hoisted() {

 console.log('foo');

}

**Function Expression**

When a function is created by using an expression it is called function expression.

e.g:

notHoisted(); // TypeError: notHoisted is not a function

var notHoisted = function() {

  console.log('bar');

};

[34. How to remove duplicate values from a JavaScript array?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled34)

We can use array.indexOf method to check a value exists or not. See below example to remove duplicate values.

let duplicates = ['delhi','kanpur','kanpur','goa','delhi','new york'];

function removeDuplicatesValues(arr){

let unique\_array = [];

for(let i = 0;i < arr.length; i++){

if(unique\_array.indexOf(arr[i]) == -1){

unique\_array.push(arr[i])

}

}

return unique\_array

}

console.log(removeDuplicatesValues(duplicates));

[35. What is console.time() and console.timeEnd()? What is its syntax, and why is it used?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled35)

[36. Explain JavaScript Debounce Function?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled36)

[37. What is difference between var x =1; and x=1;?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled37)

[38. Explain JavaScript Accessors ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled38)

JavaScript Accessors

[39. What is difference between module.exports and export?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled39)

The module is a plain JavaScript object with an exports property. Exports is a plain JavaScript variable that happens to be set to module.exports. At the end of your file, node.js will basically ‘return’ module.exports to the require function. A simplified way to view a JS file in Node could be this:

var module = { exports: {} };

var exports = module.exports;

// your code

return module.exports;

If you set a property on exports, like exports.a = 9;, that will set module.exports.a as well because objects are passed around as references in JavaScript, which means that if you set multiple variables to the same object, they are all the same object; so then exports and module.exports are the same objects.  
But if you set exports to something new, it will no longer be set to module.exports, so exports and module.exports are no longer the same objects.

Source : https://stackoverflow.com/questions/16383795/difference-between-module-exports-and-exports-in-the-commonjs-module-system

[40. What are exports and imports?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled40)

Imports and exports help us to write modular javascript code. Using Imports and exports we can split our code into multiple files. Imports allow taking only some specific variables or methods of a file. We can import methods or variables that are exported by a module. See the below example for more detail.

//index.js

import name,age from './person';

console.log(name);

console.log(age);

//person.js

let name ='Sharad', occupation='developer', age =26;

export { name, age};

[41. What is the difference between let and var?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled41)

Both var and let are used for variable/ method declaration in javascript but the main difference between let and var is that **var** is function scoped whereas **let** is block scoped.

[42. List the comparison operators supported by Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled42)

Javascript supports below comparison operators

* > Greater than
* < Less than
* <= Less than or equal to
* >= Greater than or equal to
* == Equal to
* != Not Equal to
* === Equal to with datatype check
* !== Not equal to with datatype check

[43. List some Unit Testing Frameworks JavaScript](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled43)

Below is the list of few most Popular Javascript Unit Testing Frameworks:

* Unit.js
* Jasmine
* Karma
* Chai
* AVA
* Mocha
* JSUnit
* QUnit
* Jest

[44. How do you declare variables in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled44)

In Javascript variable are declared using the var keyword.A variable must begin with A **letter**, **$** or \_.

**eg.** var myVar=”Online Interview Questions”;

**PS:** All variables in Javascript are Case sensitive.

Also, read **[Advanced JavaScript Interview Questions](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/)**

[45. What are different types of Popup boxes available in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled45)

In Javascript there are 3 types of Popup Boxes are available, they are

* Alert
* Confirm
* Prompt

Read **[80+ Best Angular Js Interview Questions](https://www.onlineinterviewquestions.com/angular-js-interview-questions/)**

[46. How to convert Javascript date to ISO standard?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled46)

**toISOString()** method is used to convert javascript date to ISO standard. It converts JavaScript Date object into a string, using the ISO standard.

**Usage:**

var date = new Date();

var n = date.toISOString();

console.log(n);

// YYYY-MM-DDTHH:mm:ss.sssZ

[47. How to you change the title of the page by JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled47)

You can change the title of a webpage using setting the title property of the document object.

**Example usage**

document.title="My New Title";

[48. How to clone an object in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled48)

Object.assign() method is used for cloning an object in Javascript.Here is sample usage

var x = {myProp: "value"};

var y = Object.assign({}, x);

[49. How to encode and decode a URL in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled49)

**encodeURI()** function is used to encode an URL in Javascript.It takes a url string as parameter and return encoded string. Note: encodeURI() did not encode characters like **/ ? : @ & = + $ #**, if you have to encode these characters too please use encodeURIComponent(). Usage:

var uri = "my profile.php?name=sammer&occupation=pāntiNG";

var encoded\_uri = encodeURI(uri);

**decodeURI()** function is used to decode an URL in Javascript.It takes a encoded url string as parameter and return decoded string. Usage:

var uri = "my profile.php?name=sammer&occupation=pāntiNG";

var encoded\_uri = encodeURI(uri);

decodeURI(encoded\_uri);

[50. Explain Typecasting in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled50)

In Programming whenever we need to convert a variable from one data type to another Typecasting is used. In Javascript, we can do this via library functions. There are basically 3 typecasts are available in Javascript Programming, they are:

* Boolean(value): Casts the inputted value to a Boolean
* Number(value): Casts the inputted value to an Integer or Floating point Number.
* String(value) : Casts the inputted value value a string

[51. List different ways of empty an array in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled51)

In Javascript, there are many ways to empty an array in Javascript, below we have listed 4 major

* By assigning an empty array.
* var arr1 =[1,4,5,6];
* arr1=[];
* By assigning array length to 0.
* var arr2 =[1,4,5,6];
* arr2.length=0;
* By poping the elements of the array.
* var arr2 =[1,4,5,6];
* while(arr.length > 0) {
* arr.pop();
* }
* By using .splice() .
* var arr =[1,4,5,6];
* arr.splice(0,arr.length)

[52. What is the ‘Strict’ mode in JavaScript and how can it be enabled?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled52)

**Strict mode** is a way to introduce better error-checking into your code. When you use strict mode, you cannot, for example, use implicitly declared variables, or assign a value to a read-only property, or add a property to an object that is not extensible.

You can enable strict mode by adding **“use strict”**; at the beginning of a file, a program, or a function. This kind of declaration is known as a directive prologue. The scope of a strict mode declaration depends on its context. If it is declared in a global context (outside the scope of a function), all the code in the program is in strict mode. If it is declared in a function, all the code in the function is in strict mode.

[53. How can you create an array in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled53)

There are 3 different ways to create an array in Javascript. They are

* By array literal  
  **usage:**

var myArray=[value1,value2...valueN];

* By creating instance of Array  
  **usage:**

var myArray=new Array();

* By using an Array constructor  
  **usage:**

var myArray=new Array('value1','value2',...,'valueN');

[54. What is Javascript BOM?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled54)

BOM stands for “Browser Object Modal” that allows Javascript to ‘talk’ to the browser, no standards, modern browsers implement similar BOMS – window, screen, location, history, navigator, timing, cookies.

[55. What does the instanceof operator do?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled55)

In Javascript **instanceof** operator checks whether the object is an instance of a class or not:

**Example Usage**

Square.prototype = new Square();

console.log(sq instanceof Square); // true

[56. How to get the primitive value of a string in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled56)

In Javascript **valueOf()** method is used to get the primitive value of a string.

**Example Usage:**

var myVar= "Hi!"

console.log(myVar.valueOf())

[57. How to get the last index of a string in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled57)

**string.length-1** is used to get the last index of a string in Javascript

**Example Usage:-**

var myString="JavascriptQuestions";

console.log(myString.length-1);

[58. List HTML DOM mouse events?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled58)

HTML DOM mouse events

* onclick
* ondblclick
* mousemove
* mousedown
* mouseover
* mouseout
* mouseup

[59. What will happen if an infinite while loop is run in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled59)

The program will crash the browser.

[60. In Javascript are calculations with fractional numbers guaranteed to be precise?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled60)

NO, calculations with fractional numbers are not guaranteed to be precise in Javascript

[61. What is the difference between the substr() and substring() functions in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled61)

Difference between the substr() and substring() functions in JavaScript.

The substr() function has the form substr(startIndex,length). It returns the substring from startIndex and returns ‘length’ number of characters.

var s = "hello";

( s.substr(1,4) == "ello" ) // true

The substring() function has the form substring(startIndex,endIndex). It returns the substring from startIndex up to endIndex – 1.

var s = "hello";

( s.substring(1,4) == "ell" ) // true

[62. What are the primitive data types in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled62)

A primitive is a basic data type that’s not built out of other data types. It can only represent one single value. All primitives are built-in data types by necessity, (the compiler has to know about them,) but not all built-in data types are primitives.

In JavaScript there are 5 primitive data types are available they are **undefined**, **null**, **boolean**, **string** and **number** are available.Everything else in Javascript is an object.

[63. How to calculate Fibonacci numbers in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled63)

Calculating Fibonacci series in JavaScript

Fibonacci numbers are a sequence of numbers where each value is the sum of the previous two, starting with 0 and 1. The first few values are 0, 1, 1, 2, 3, 5, 8, 13 ,…,

function fib(n) {

var a=0, b=1;

for (var i=0; i < n; i++) {

var temp = a+b;

a = b;

b = temp;

}

return a;

}

[64. What are different types of Inheritence? Which Inheritance is followed in Javascript.](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled64)

There are two types of Inherientence in OOPS Classic and Prototypical Inheritance. Javascript follows Prototypical Inheritance.

[65. What is output of undefined \* 2 in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled65)

nan is output of undefined \* 2.

[66. Explain “use strict” ?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled66)

“use strict” is a javascript directive that is introduced in Es5. The purpose of using “use strict” directive is to enforce the code is executed in strict mode. In strict mode we can’t use a variable without declaring it. “use strict” is ignored by earlier versions of Javascript.

[67. Describe negative infinity in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled67)

NEGATIVE\_INFINITY property represents negative infinity and is a number in javascript, which is derived by ‘dividing negative number by zero’. It can be better understood as a number that is lower than any other number. Its properties are as follows:  
– A number of objects need not to be created to access this static property.  
– The value of negative infinity is the same as the negative value of the infinity property of the global object.

The values behave differently than the mathematical infinity:

1. Any positive value, including POSITIVE\_INFINITY, multiplied by NEGATIVE\_INFINITY is NEGATIVE\_INFINITY.
2. Any negative value, including NEGATIVE\_INFINITY, multiplied by NEGATIVE\_INFINITY is POSITIVE\_INFINITY.
3. Zero multiplied by NEGATIVE\_INFINITY is NaN.
4. NaN multiplied by NEGATIVE\_INFINITY is NaN.
5. NEGATIVE\_INFINITY, divided by any negative value except NEGATIVE\_INFINITY, is POSITIVE\_INFINITY.
6. NEGATIVE\_INFINITY, divided by any positive value except POSITIVE\_INFINITY, is NEGATIVE\_INFINITY.
7. NEGATIVE\_INFINITY, divided by either NEGATIVE\_INFINITY or POSITIVE\_INFINITY, is NaN.
8. Any number divided by NEGATIVE\_INFINITY is zero.

[68. Explain JavaScript Event Delegation Model?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled68)

In JavaScript, there is some cool stuff that makes it the best of all. One of them is Delegation Model. When capturing and bubbling, allow functions to implement one single handler to many elements at one particular time then that is called event delegation. Event delegation allows you to add event listeners to one parent instead of specified nodes. That particular listener analyzes bubbled events to find a match on the child elements. Many people think it to be complicated but in reality, it is very simple if one starts understanding it.

Also, **Read Five [JavaScript Frameworks to learn in 2018](https://www.onlineinterviewquestions.com/blog/5-javascript-frameworks-to-learn/)**

[69. Explain Closures in JavaScript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled69)

Closures are the combination of lexical environment and function within which the function was declared. This allows JavaScript programmers to write better, more creative, concise and expressive codes. The closure will consist of all the local variables that were in-scope when the closure was created.

Sure, closures appear to be complex and beyond the scope, but after you read this article, closures will be much more easy to understand and more simple for your everyday [JavaScript](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/)programming tasks. JavaScript is  a very function-oriented language it gives the user freedom to use functions as the wish of the programmer.

[70. What close() does in Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled70)

In Javascript close() method is used to close the current window. You must write window.close() to ensure that this command is associated with a window object and not some other JavaScript object.

[71. Explain what is Javascript? List some data types supported by Javascript?](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/" \l "collapseUnfiled71)

Javascript

Javascript is an object-oriented computer programming language commonly used to create interactive effects within web browsers.It is first used by the Netscape browser, that provides access to the HTML document object model (DOM), provides access to the browser object model (BOM). Javascript syntax looks a lot like java, c or c++ syntax.

Below is the list of data types supported by Javascript:-

* Undefined
* Null
* Boolean
* String
* Symbol
* Number

Commonly asked JavaScript Interview Questions | Set 1

1. **What is**[**JavaScript(JS)**](https://www.geeksforgeeks.org/how-to-be-a-javascript-developer-without-knowing-javascript/)**?**
   * JavaScript is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages.
2. **What are the features of JavaScript?**
   * JavaScript is a lightweight, interpreted programming language.
   * JavaScript is designed for creating network-centric applications.
   * JavaScript is complementary to and integrated with Java.
   * JavaScript is complementary to and integrated with HTML.
   * JavaScript is open and cross-platform.
3. **What are the advantages of JavaScript?**
   * **Less server interaction?** You can validate user input before sending the page off to the server.
   * **Immediate feedback to the visitors?** They don’t have to wait for a page reload to see if they have forgotten to enter something.
   * **Increased interactivity?** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard
4. **Why is javascript called Richer Interface?**
   * You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.
5. **Is javascript case-sensitive?**
   * Yes, JavaScript is a case-sensitive language. This means that language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.
6. **How can we create an object in JS?**
7. var object =
8. {
9. name: "obj",
10. age: 10

};

1. **How can we read the properties of an object in js?**
   * Can write and read properties of an object using the dot(.) notation.
2. **How to create an array in js and how to read array elements?**  
   Can you define arrays using the array literal as follows?
   * var x = [];
   * var y = [1, 2, 3, 4, 5];

An array has a length property that is useful for iteration. Can we read elements of an array as follows?

for (var i = 0; i < x.length; i++)

1. **How many types of functions JS support?**
   * A function in JavaScript can be either named or anonymous.
2. **How to define an anonymous function?**
   * An anonymous function can be defined in a similar way as a normal function but it would not have any name.
3. **Which built-in method calls a function for each element in the array?**
   * forEach method calls a function for each element in the array.
4. **Which type of variable among global and local, takes precedence over other if names are same?**
   * A local variable takes precedence over a global variable with the same name.
5. **Difference between**[**“var” and “let”**](https://www.geeksforgeeks.org/difference-between-var-and-let-in-javascript/)**Keywords?**
   * **Var** was there from the beginning but the let was introduced in ES2015/ES6.  
     **Let** has block scope and “Var” has function scope
6. **Difference between**[**“==” and “===”**](https://www.geeksforgeeks.org/difference-between-and-operator-in-javascript/)**?**
   * ” ==” only compares values “===” comapre values and type both.
7. **Difference between “undefine” and “NULL” Keywords?**
   * When you define a var but not assign any value. typeof(undefine)=> undefine  
     Null- manually done. typeof(null)=> object
8. **What is prototypal Inheritance?**
   * Every object has a property called a prototype, where we can add methods to it and when you create another object from these the newly created object will automatically inherit its parent’s property.
9. **Which built-in method sorts the elements of an array?**
   * [Sort](https://www.geeksforgeeks.org/javascript-sort-method/) method sorts the elements of an array.
10. **Which built-in method reverses the order of the elements of an array?**
    * [Reverse](https://www.geeksforgeeks.org/javascript-array-prototype-reverse/) method reverses the order of the elements of an array ?? the first becomes the last, and the last becomes the first.
11. **What is**[**SetTimeout()**](https://www.geeksforgeeks.org/java-script-settimeout-setinterval-method/)**?**
    * When you setTimeout it becomes asynchronous and it has to wait on the stack to get everything got finished
12. **How to add one new element at end of an array in javascript?**
    * Push method adds one or more elements to the end of an array and returns the new length of the array.
13. **What is closure and how do you use it?**
    * When a function returns the other function the returning function will hold its environment and this is known as closure.
14. **Output of below statements**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| <script>  document.write({});  </script> |

1. **How can you move element in lowercase to uppercase from an array?**
   * [toUpperCase](https://www.geeksforgeeks.org/javascript-string-touppercase/) method returns the calling string value converted to upper case.

JavaScript Interview Questions And Answers

Q.1- What’s An Object In Javascript And How Do We Create Them?

A JavaScript object is an entity having state and behavior (properties and method). Since JavaScript is an object-based language, it treats everything as an object.

JavaScript is a template-based language. It doesn’t need to define a class for creating an object instead creates it directly

JavaScript supports the following three ways to create objects.

1. By Object Literal.

The syntax of creating an object using object literal is as follows.

object={property1:value1, property2:value2.....propertyN:valueN}

Here, property and value get separated by a colon “:” sign.

Let’s take an example.

<script>

std={id:1114, name:"Ram Bajaj", subject:"Physics"}

document.write(std.id+" "+std.name+" "+std.subject);

</script>

2. By Creating An Instance Of The Object (Using New Keyword).

The syntax of creating an object:

var objectname=new Object();

Here, the new keyword is used to create the object.

Let’s take an example.

<script>

var std=new Object();

std.id=1114;

std.name="Ram Bajaj";

std.subject="Physics";

document.write(std.id+" "+std.name+" "+std.subject);

</script>

3. By Using An Object Constructor.

In this method, we create a function with arguments. The value of each of these arguments can be assigned to the current object by using this keyword.

This keyword refers to the current object.

Let’s take an example of creating an object using the object constructor technique.

<script>

function std(id,name,subject){

this.id=id;

this.name=name;

this.subject=subject;

}

s=new std(1114,"Ram Bajaj","Physics");

document.write(s.id+" "+s.name+" "+s.subject);

</script>

Q-2. What Does A Scope Mean In JavaScript?

The scope determines the accessibility of variables, objects, and functions in particular part of your code.

In JavaScript, the scope is of two types.

1. Global Scope.

A variable defined outside a function comes under the Global scope. Variables defined inside the Global scope are accessible from any part of the code. Let’s see an example.

var name = 'TechBeamers';

console.log(name); // logs 'TechBeamers'

function logName() {

console.log(name); // 'name' is accessible here and everywhere else

}

logName(); // logs 'TechBeamers'

2. Local Scope.

Variables defined inside a function comes under the Local scope. Different functions can use a variable with the same name. It is because these variables are strictly bound to the function that defines it (each having different scopes) and is not accessible in other functions. Let’s see an example.

// Global Scope

function sampleFunction() {

// Local Scope #1

function sample2Function() {

// Local Scope #2

}

}

// Global Scope

function sample3Function() {

// Local Scope #3

}

// Global Scope

Q-3. What Is <This> In JavaScript?

All the OOPs languages use ‘this’ keyword to refer to an object that is currently instantiated by the class. However, in JavaScript, ‘this’ refers to an object which ‘owns’ the method. Though this varies, with how a function call happens.

Global Scope.

If no object is currently available, then ‘this’ represents the global object. In a web browser, ‘window’ is the top-level object which represents the document, location, history and a few other useful properties and methods. Let’s take a sample code.

window.Obj= "I represent the window object";

alert(window.Obj);

alert(this.Obj); // I'm the window object

alert(window === this); // true

The Scenario Of A Function Call.

In the case of a function call, ‘this’ refers to the global object.

window.Obj = "I represent the window object";

function TestFunction() {

alert(this.Obj); // I'm the window object

alert(window === this); // true

}

TestFunction();

Call Object Methods.

When an object constructor or any of its methods gets called, ‘this’ refers to an instance of an object. It is similar to any class-based language.

window.Obj = "I'm the window object";

function TestFunction() {

this.Obj = "I'm the Test object";

this.Verify1 = function() {

alert(this.Obj); // I'm the Test object

};

}

TestFunction.prototype.Verify2 = function() {

alert(this.Obj); // I'm the Test object

};

var tf= new TestFunction();

tf.Verify1();

tf.Verify2();

Q-4. What Is The Prototype Property In JavaScript?

Every JavaScript function has a prototype property (by default this property is null), that is mainly used for implementing inheritance. We add methods and properties to a function’s prototype so that it becomes available to instances of that function. Let’s take an example that calculates the perimeter of a rectangle.

function Rectangle(x, y) {

this.x = x;

this.y = y;

}

Rectangle.prototype.perimeter = function() {

return 2 \* (this.x + this.y);

}

var rect = new Rectangle(4, 2);

console.log(rect.perimeter()); // outputs '12'

Q-5. What Is Closure In JavaScript?

A closure is a JavaScript function defined inside another function. And that’s why it gets a special privilege to access three types of scope which are as follows.

* Internal Scope, i.e., the variables defined between its curly brackets
* Outer Function Scope, i.e., the variables of the enclosing function
* Global Scope, i.e., variables defined as globals

Please note that a closure can not only access the outer function variables but also see its parameters. But it can’t call the object of the outer function’s arguments. However, it can directly call the outer function’s parameters.

Here is a code example describing closure by adding a function inside another function.

function outerFunc(arg1, arg2) {

​var param = "I'm closure. ";

// Inner function accessing outer function variables and parameter​s

​function innerFunc() {

​ return arg1 + arg2 + " " + param;

}

​ ​return innerFunc();

}

​

outerFunc("arg1", "arg2");

Q-6. Why Is “Self” Needed Instead Of “This” In JavaScript?

Inner functions in JavaScript have access to all of the variables defined in the outer function. However, “this” variable is an exception. Since the nested function is just a regular function and not an object method, it’s “this” refers to the global namespace. To make it more clear, let’s look at the following example.

var aProperty = 'global';

var myObject = {

outerFun: function() {

this.aProperty = 'local';

setTimeout(function() {

console.log(this.aProperty); // outputs 'global'

}, 1);

}

};

Thus, we see that inside “setTimeout” function, “this” refers to the global object. We need a way to get a reference to the object, that is available inside the nested function. We assign the object from “this”, to another(non-special) variable, “self”. It is not a special variable and hence cannot be overwritten by other functions(like “this”). Thus on using “self” inside the inner function, we can refer to the local object. Following is the sample code.

var myObject = {

outerFun: function() {

var self = this;

this.aProperty = 'local';

setTimeout(function() {

console.log(self.aProperty); // outputs 'local'

}, 1);

}

};

Q-7. What Is An Anonymous Function And When Should You Use It?

Anonymous functions are functions that are dynamically declared at runtime. They’re called anonymous functions because they don’t have a name like normal functions.

We use the function operator to declare an anonymous function, instead of the function declaration. Also, the function operator can be used to create a new function, wherever it’s valid to put an expression. For example, we declare a new function to be supplied as an argument to a function call or to assign a property of another object.

Here’s a typical example of a named function.

function testFunction()

{

alert("Welcome!!");

}

testFunction();

Here’s the same example created as an anonymous function.

var testFunction= function()

{

alert("Zoom! Zoom! Zoom!");

}

flyToTheMoon();

Following are the key usage of anonymous functions.

* **Code brevity.**
  + Use them in
    - Callbacks, and
    - Event handlers.
* **Scope management.**
  + They are useful in the following scenario.
    - To create a temporary/private scope.
    - In Closures and Recursions.

Q-8. What Is The Difference Between “==” And “===”?

These are the operators provided by JavaScript – strict equality and Type converting equality.

Strict equality (===) returns true if the values which it is going to compare have the same data type. Taking an example, “2” will not be equal to 2  i.e. (“2″===2) will return false.

Secondly, Type converting equality (==), automatically converts the variable to value irrespective of the data type. Taking an example, here “2” will be equal to 2  i.e. (“2″===2) will return true.

Summarizing it, double equal (==) is an autotype converting equality operator while three equals (===) is a strict equality operator, i.e., it will not convert values automatically.

Q-9. What Are JavaScript Data Types?

JavaScript supports three Primary, two Composite and two Special data types. Next, we list down the data types in each of the categories.

Primary Data Types.

* String
* Number
* Boolean

Composite Data Types.

* Object
* Array

Special Data Types.

* Null
* Undefined

Q-10. What Is Prototypal Inheritance In JavaScript?

Most of the Object Oriented languages support classes and objects. Here, Classes inherit from other classes.

In JavaScript, the inheritance is prototype-based. It means that there are no classes. Instead, there is an object that inherits from another object.

JavaScript provides three different types of Prototypal Inheritance.

1. Delegation (I.E., The Prototype Chain).

A delegate prototype is an object that serves as a base for another object. When you inherit from a delegate prototype, the new object gets a reference to the prototype.

When we try to access any property, it first checks in the properties owned by the object. If that property does not exist there, it checks in the **‘[[Prototype]]’** and so on. If that property does not exist there, it checks in the **‘[[Prototype]]’** and so on. Gradually, it moves up the prototype chain, until it reaches the **<Object.prototype>** i.e., the root delegate for most of the objects.

2. Concatenative Inheritance (I.E. Mixins, Object.Assign()).

It is the process of inheriting the features of one object to another by copying the source objects properties. JavaScript calls these source prototypes by the name mixins. This process makes use of the JavaScript method Object.assign(). However, before ES6, the <.extend()> method was used.

3. Functional (Not To Be Confused With Functional Programming).

In JavaScript, a function can create an object. It’s not necessary to be a constructor(or a class). It is called a factory function. Functional inheritance produces an object from a factory and also extends it, by assigning properties.

Every type of Prototypal Inheritance supports a separate set of use-cases, applicable to it. All of them are equally useful in their ability to enable composition. It provides a has-a, uses-a, or can-do relationship as compared to the is-a relationship created with class inheritance.

Q-11. What Is Asynchronous Programming? Why Is It Important In JavaScript?

In Asynchronous programming, the engine runs in an event loop. On encountering a blocking operation, a request gets fired, and the code keeps running without blocking for the result. When its execution completes, and the response is ready, it fires an interrupt, which causes an event handler to be run, where the control flow continues. In this way, a single program thread can handle many concurrent operations in asynchronous programming.

The user interface is asynchronous by nature and spends most of the time, waiting for user input to interrupt the event loop and trigger event handlers. Node is asynchronous by default. It means that the server works by waiting for a network request in a loop. Thus, it accepts more incoming requests, while the first one is getting handled.

It is important in JavaScript because it’s suitable for user interface code and beneficial for maintaining the performance on the server.

Q-12. What Is Variable Typing In JavaScript?

JavaScript is very loosely typed language. It means that the variables are declared without a type. Its type is determined once a value is assigned to it. It can change as the variable appears in different contexts.

It’s in contrast to the strongly typed languages that require type declarations.

Consider the following examples.

/\* JavaScript Example (loose typing) \*/

var a = 13; // Number declaration

var b = "thirteen"; // String declaration

/\* Java Example (strong typing) \*/

int a = 13; // int declaration

String b = "thirteen"; // String declaration

Variable type in JavaScript is determined internally. In the above example, var ‘a’ will be of type ‘Number’ and var ‘b’ will be of type ‘String’.

Q-13. What Are The Different Ways To Create An Array In JavaScript?

There are two main ways to create an array in JavaScript.

1. Using An Array Initializer (Array Literal).

The array initializer (array literal) syntax is simple. It is a comma-separated list of values in square brackets.

Let’s see some examples.

var myArray1 = [1,2,3,4,5] // an array with 5 elements

var myArray2 = [5] // an array with 1 element

var myArray3 = [true,'Hi',[7]] // element types need not be the same.

2. Using The Array Constructor.

The Array constructor method has three different syntaxes. If we call the constructor with two or more arguments, it declares an array with array elements also initialized. If we provide only one argument to the Array constructor, it refers to the length of the new array with, elements not initialized. Lastly, the constructor without any argument creates an array with its length set to zero with elements not initialized.

Let’s see some examples.

var myArray4 = new Array(1,2,3,4,5) // an array with 5 elements

var myArray5 = new Array(20) // an empty array of length 20

var myArray6 = new Array() // an empty array of length 0

Q-14. What Are JavaScript Cookies?

A cookie is a piece of data which is sent from a website (that owns the requested web page) and gets stored locally by the browser at the user end. Cookies are needed because HTTP protocol which arranges for the transfer of web pages to your browser, is stateless. It means that HTTP has no way to keep track of the activities performed by the user at an earlier point in time. One way to resolve this issue is by using cookies. It contains the following data.

* A name-value pair containing the actual data
* An expiry date after which the cookie is no longer valid
* The domain and path of the server it should be sent to

When a request arrives at the server for a web page that maintains a cookie, the server appends the cookie to the HTTP header to send it across. The server-side programs can then read out the information included in it and decide that you have the right to view the page or not and other user preferences.

Thus, every time you visit the site that maintains the cookies, your information is available there.

Q-15. Which Built-In Method Adds One Or More Elements To The End Of An Array And Returns The New Length Of The Array?

The push() method adds one or more elements to the end of an array and returns the new length of the array.

Q-16. What Is The Naming Conventions For The Variables In JavaScript?

We have to follow the below rules while naming the variables in JavaScript.

* Do not use any of the JavaScript reserved keywords as a name for your variable. For example, the “break” or “boolean” is the JavaScript keywords, and if used as variable names, it’s invalid.
* JavaScript variable names should not start with a numeral (0-9). It must begin with a letter or the underscore character. For example, **123var** is an invalid variable name, but **\_123var** is a valid one.
* Also, JavaScript variable names are case sensitive. For example, test and Test are two different variables.

Q-17. How Will You Create A Cookie Using JavaScript?

The simplest way to create a cookie is to assign a string value to the <**document.cookie**> object.

Its syntax is as follows.

document.cookie = "key1 = value1; key2 = value2; expires = date";

Here, “expires” attribute is optional. We have to provide a date or time value for this attribute.

If we provide a valid value for the date or time, then the cookie will expire at the given date or time, and it will not be accessible after that.

Q-18. How To Read A Cookie Using JavaScript?

To read a Cookie, we have to access the value of the <document.cookie> object. This <document.cookie> string maintains a list of <name = value> pairs that is separated with semicolons.

Where,

"name" is the name of a cookie and

"value" is its string value.

We use String <split()> function to break the <document.cookie> Object to sub-strings. Each of these sub-strings contains a key-value pair which represents the information related to a Cookie.

Q-19. How To Delete A Cookie Using JavaScript?

To delete a Cookie, we have to set its expiry date to a time that occurred in the past. If attempts are made to read a deleted Cookie then, nothing is returned.

Q-20. How Do You Submit A Form Using JavaScript?

We use the following in the web page code to submit a form.

document.forms[0].submit();

Here <0> refers to the index of the form. If we have more than one forms on a web page, then the index for the first one will be “0”, for the second form, its value will be “1” and so on.

Q-21. Why Should You Not Prefer To Use Global Variables In JavaScript And How Can You Prevent It?

The principal issue in using a global variable is that someone else can create another variable with the same name. And you may not know it until the duplicate could overwrite the value of your variable.

To avoid using globals, follow any of the following approaches.

**1.** Create a single global variable that holds all your other variables.

var myGlobalList = {};

myGlobalList.first = "test";

**2.** Enclose all of your code in a self-executing method/function so that any variable declared inside remain in the function scope.

(function(){

var test = "myvar";

})();

Q-22. What Are The Different Objects Used In JavaScript?

JavaScript uses a hierarchical structure, applicable to all the objects created in a document. Following are the objects, used in JavaScript that shows the relationship of one object to another.

Window Object.

It is the topmost object in the hierarchy. It refers to the content area of the browser window that consists of HTML documents. Each frame is also a window that has some actions inside it.

Document Object.

A Document object represents the HTML document that the window will display. It has various properties that refer to other objects, which allow access to and modification of content in the document.

Form Object.

A form object is used to take user data as input for processing. It corresponds to an HTML input form constructed with the **<FORM>…</FORM>** tag.

Q-23. What Do We Achieve By Deferring The Loading Of JavaScript?

During the page load operation, by default, the parsing of the HTML code remains paused until the execution of the script has not stopped.

Now, if the script is heavy or the server is slow, then this will result in a delay in displaying the web page.

By deferring the loading of JavaScript, we instruct the browser to load the script, only after the loading of the web page(DOM) has finished. It reduces the loading time of the webpage, and it gets displayed faster.

In this way, it makes the page load quicker, without the average user even realizing that some JavaScript load after the DOM has loaded.

Let’s see an example.

<script>

// this piece of code runs first

</script>

<script defer="defer">

//do stuff, but defer it (runs last)

</script>

<script>

//do more stuff (runs second)

</script>

Here the middle block will execute once the page has loaded even though it appears before the last block.

Q-24. What Is The Strict Mode In JavaScript?

Strict Mode imposes a layer of constraint on JavaScript. It provides the following enhancements.

* JavaScript will throw an error if we try to use the elements of a deprecated language.
* To use a variable, it has become mandatory to declare it.
* It disallows duplicate property and parameter names.
* The **eval()** method is safer to use, but still considered evil in some cases.
* It deprecates the “with” statement.
* JavaScript will throw an error if we try to assign a value to a read-only property.
* It decreases the global namespace pollution.

To enable strict mode, we have to add, “use strict” directive to the code. The physical location of the “strict” directive determines its scope. If used at the beginning of the js file, its scope is global. However, if we declare strict mode at the first line in the function block, its scope restricts to that function only.

Q.25- What Are Event Handlers In JavaScript And How To Use Them?

JavaScript event handlers are functions that bind to a specific HTML DOM event. And events are the part of the HTML document object model (DOM). An event can take place in one of the following cases.

Due to user actions on a web page.

<onclick()>, <onmouseover()>, <onkeydown()>

Some events are callbacks triggered by the browser when a page changes its state.

<onload()>, <onunload()>, <onresize()>

Whenever the DOM receives an event, it calls the JavaScript event handler function bind to the event. To use a handler function, we can assign it to the desired event as an attribute of the target HTML element. Please follow the below example.

<HTML>

<BODY>

<FORM>

<INPUT TYPE=”button” VALUE=”Test” onClick=”window.alert ('Event Handler')”>

</FORM>

</BODY>

</HTML>

Q-26. What Does A JavaScript Function Result When It Has No Return Statement?

If a function doesn’t use the return statement, then it returns undefined as the return value.

However, the browser won’t print anything in its console as it ignores if the expression returns an undefined value. But if we explicitly call such a function inside the **console.log()** method, then it’ll print exact value.

function test(){}

console.log(test()); // will print 'undefined'

Q-27. What Is EncodeURI() Function?

The **encodeURI()** function is used to encode a URI. This function encodes all special characters, except these **< , / ? : @ & = + $ #>.**

Let’s See An Example.

var uri="//www.techbeamers.com/how to make a website using javaScript";

var encodedURI = encodeURI(uri);

console.log(encodedURI);

Output.

http://www.somedomain.com/how%20to%20make%20a%20website%20using%20javaScript

We see that JavaScript encodes the space between the words in the **<uri>** variable as **<%20>**. Thus, the encodeURI function is used to encode special reserved characters and other non-ASCII characters in the URI.

Q-28. How Does The <Array()> Differ From <[]> While Creating A JavaScript Array?

Both the **<Array()>** and **<[]>** works almost the same in JavaScript.

If we use them as is (i.e., without any argument) to create an array object, then they will result in an array object of zero length. Also, if we pass a string or a list of strings as arguments, even then the result will be similar.

However, they differ when the input argument is of integer type. In that case, the **<Array(n)>** statement will create an uninitialized array of size of n. Whereas, the **<[n]>** statement will create an array of size **<1>**and assign **<n>** as value to the first element.

Q-29. How Does Variable Hoisting Take Place In JavaScript?

In JavaScript, variable declarations are the first piece of code that executes irrespective of their location. Hence, it doesn’t matter whether we declare a variable at the top or anywhere else. This functionality which moves the declaration to the top either inside a function or in the global code is known as hoisting.

out = 2;

var out;

function myfunc() {

var in = "inside"

// some code

}

// ...

JavaScript will interpret the above in the following manner.

var out;

out = 2;

function myfunc() {

var in;

// some code

in = "inside";

}

Q-30. How Will You Replace All Occurrences Of A String In JavaScript?

We can use String’s **<replace()>** method to substitute any string. There are the following two ways to use this method.

Pass The Input String As A Regular Expression.

str = "ghktestlllltest-sdds"

str = str.replace(/test/g, '');

alert(str)

Use RegExp Class To Create A Regular Expression And Pass It.

String.prototype.replaceAll = function(find, replace) {

var target = this;

return target.replace(new RegExp(find, 'g'), replace);

};

str = "ghktestlllltest-sdds"

str = str.replaceAll('test', '');

alert(str)