**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.

**JavaScript Unit Testing**

1. JavaScript Unit tests are usually run in the browser or on the frontend. Some JavaScript code is written for running a test for a page of the website or a single module of an application, and then this code is combined with HTML as an inline event handler
2. These unit tests are organized one by one in a Suite. Each and every suite contains number of tests separately performed for a separate module
3. Most importantly they don't conflict with any other module and runs with fewer dependencies on each other (some critical situation may cause dependencies)

**Challenges in JavaScript Unit Testing**

There are certain problems one can find while performing JavaScript Unit Testing such as;

1. Many other languages support unit testing in browsers, in the stable as well as in runtime environment but JavaScript can not
2. You can understand some system actions with other languages, but this is not the case with JavaScript
3. Some JavaScript are written for a web application may have multiple dependencies
4. JavaScript is good to use in combination with HTML and CSS rather than on the web
5. Difficulties with page rendering and DOM manipulation

To avoid such issues what you can do is;

1. Do not use global variables
2. Do not manipulate predefined objects
3. Design core functionalities based on library
4. Try to create small pieces of functionalities with lesser dependencies

**JavaScript Unit Testing Frameworks**

Sometimes you find the error message on your screen regarding such as 'Unable to load example.js' or any other JavaScript error regarding version control, these vulnerabilities comes under JavaScript Unit Testing.

We will review some tools and frameworks that are being used to perform JavaScript Unit Testing.

1. [JSUnit](http://www.jsunit.net/): JSUnit is an open source unit testing framework for client-side JavaScript running in the browser. This framework runs on different platforms and browsers. It fulfills server needs such as;
   1. Logging results in XML format
   2. Running JSUnit from JUnit and Ant and more than one or more browser and remote machines as well

2. [Unit.js](http://unitjs.com/): It is known as an open source assertion library running on browser and Node.js. It is extremely compatible with other JavaScript Unit Testing frameworks like Mocha, Karma, Jasmine, QUnit, Protractor, etc. Provides the full documented API of assertion list

3. [QUnit](https://qunitjs.com/): It is used for both client-side as well as server-side JavaScript Unit Testing. This Free framework is used for jQuery projects. It follows Common JS unit testing Specification for unit testing. It supports the Node Long-term Support Schedule.

4. [Jasmine](https://jasmine.github.io/): Jasmine is the behavior-driven development framework for JavaScript unit Testing. It is used for testing both synchronous and asynchronous JavaScript Code. It does not require DOM and comes with the easy syntax that can be Written for any test.

5. [Karma](https://karma-runner.github.io/2.0/index.html): Karma is an open source productive testing environment. Easy workflow control Running on the command line. Offers the freedom to write the tests with Jasmine, Mocha, and QUnit. You can run the test on real devices with easy debugging.

6. [Mocha](https://mochajs.org/): Mocha runs on Node.js and in the browser. Mocha performs asynchronous Testing in a simpler way. Provides accuracy and flexibility in reporting. Provides tremendous support of rich features such as test-specific timeouts, JavaScript APIs etc.

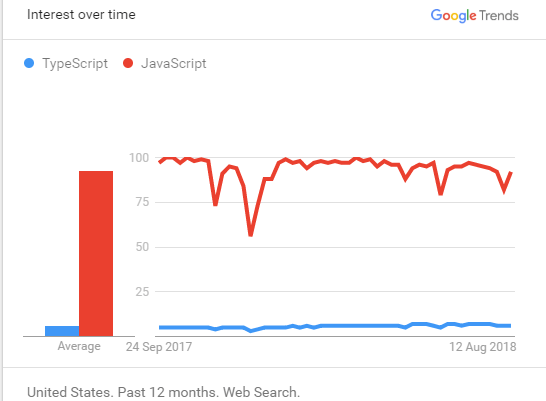
7. [Jest](https://facebook.github.io/jest/): Jest is used by Facebook so far to test all of the JavaScript code. It provides the 'zero-configuration' testing experience. Supports independent and non-interrupting running test without any conflict. Do not require any other setup configuration and libraries.

8. [AVA](https://github.com/avajs): AVA is simple JavaScript Unit Testing Framework. Tests are being run in parallel and serially. Parallel tests run without interrupting each other. AVA Supports asynchronous testing as well. AVA uses subprocesses to run the test.

**What is Typescript?**

Typescript is a modern age Javascript development language. It is a statically compiled language to write clear and simple Javascript code. It can be run on Node js or any browser which supports ECMAScript 3 or newer versions.

Typescript provides optional static typing, classes, and interface. For a large JavaScript project adopting Typescript can bring you more robust software and easily deployable with a regular JavaScript application.

[](https://www.guru99.com/images/1/101218_1344_Typescriptv1.png)

**Why JavaScript?**

* Open source project with Microsoft's patronage
* Specially designed tool for small scripts
* Supports classes, interfaces & modules.
* Compiled JavaScript runs in any browser
* Allows cross-compilation
* You can extend JavaScript for writing large apps
* Adds support for classes, interfaces, and modules

**Why Typescript?**

* TypeScript supports JS libraries & API Documentation
* It is a superset of Javascript
* It is optionally typed scripting language
* TypeScript Code can be converted into plain JavaScript Code
* Better code structuring and object-oriented programming techniques
* Allows better development time tool support
* It can extend the language beyond the standard decorators, async/await

**History of Javascript**

Netscape Communications Corporation programmer Brendan Eich created Javascript. It was meant to working Netscape navigator. However, after becoming a popular scripting tool, it is had become LiveScript. Later on, it was renamed as JavaScript to reflect Netscape's support of Java within its browser.

Let see an important landmark in the history of Javascript:

* It was launched in September 1995, and It took just ten days to develop this e scripting language which was initially called Mocha
* In November 1996, Netscape submitted JavaScript to ECMA (European Computer Manufacturers Association) International
* ECMAScript 2 was released in 1998
* ECMAScript 3 was released in 1999
* In 2005, Eich and Mozilla join ECMA to develop E4X Java script
* In January 2009, the CommonJS project was launched with the aim of defining a common standard library
* In June 2011, ECMAScript 5.1 was released
* In June 2015. ECMAScript 2016 was released
* The current version is ECMAScript 2017 which was released in June 2017

**History of Typescript**

Let see important landmarks from the History of Typescript:

* The typescript was first made public in the year 2012
* After two years of internal development at Microsoft. TypeScript 0.9, released in 2013
* Additional support for generics TypeScript 1.0 was released at Build 2014
* In July 2014, a new TypeScript compiler came which is five times faster then it's previous version
* In July 2015, support for ES6 modules, namespace keyword, for, of support, decorators
* In November 2016, an added feature like key and lookup types mapped types, and rest
* On March 27, 2018, conditional types, the improved key with intersection types supports added in the Typescript.

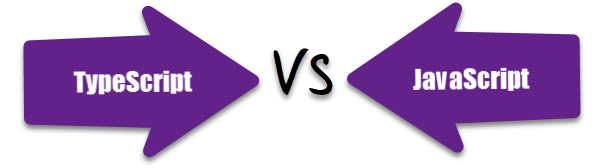
**Features of Javascript**

* It's a cross-platform language
* It's used for client side and server side
* It's easy to learn and to start with
* It's a dynamic language: flexible and powerful
* You have 'the great freedom' to do whatever you want with any object
* Strong Testing Workflow
* Added Dependencies
* Framework Unsupported

**Features of Typescript**

* Maintainability
* Offered great productivity for developers
* Code navigation and bug prevention
* Code 'discoverability' & refactoring
* Optional Static Type Annotation / Static Typing
* Additional Features for Functions
* Supports ES6
* Supports interfaces, sub-interfaces, classes, and subclasses
* Scalable HTML5 client-side development
* Rich IDE available with autocomplete and code navigation features.
* Class-based object-oriented with the inheritance of private members and interfaces.

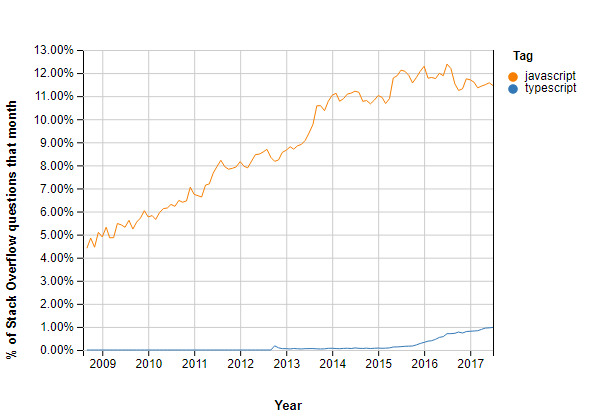
**Typescript vs. JavaScript**

[](https://www.guru99.com/images/1/101218_1344_Typescriptv2.png)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Typescript** | **JavaScript** |
| What is | Powerful type system, including generics & JS features | Lightweight, interpreted, object-oriented language with first-class functions |
| Data Binding | TypeScript uses concepts like types and interfaces to describe data being used. | No such concept is available with JavaScript. |
| Ecosystem | The Ecosystem is quite powerful and intuitive. Thus, it allows you to statically type various type of idiomatic JavaScript features like union types, intersection, discriminated union. | Javascript offers the option to explore and create code without a build step. |
| Npm package | With Typescript, many npm packages either come with static type definitions or have an external one that is easy to install. | Javascript offers the option to explore and create code without a build step. |
| Learning curve | Stiff learning curve. Requires prior scripting knowledge. | Flexible and easy to learn, scripting language. |
| Prototyping | Typescript has a feature of prototyping. | JavaScript doesn't have this feature. |
| Community | Typescript does not have a large community of developers. | The JavaScript has a huge community of developers |
| Compilation | TypeScript code needs to be compiled | No need to compile JavaScript. |
| Annotation | To get the most out of TypeScript features, developers should constantly annotate their code. | No Annotations Required is need for Javascript. |
| Famous Company using the Technology | Asana, Clever, Screen award | Airbnb, Codecademy, Instagram |
| Salary | The average salary for "Typescript developer" ranges from approximately $148,027 per year in United States | The average salary for a Javascript Developer is $110,777 per year in the United States. |

**What is better?**

In the end, we can say that if an experienced developer is working on relatively small coding projects, then JavaScript is ideal. However, if you have knowledge and expertise development team, then Typescript is a most preferred option.

[](https://www.guru99.com/images/1/101218_1344_Typescriptv3.png)

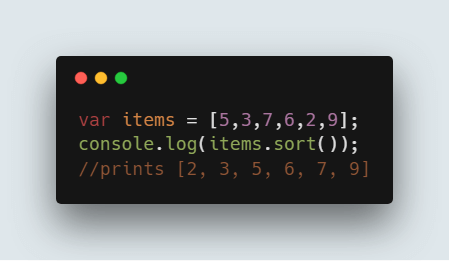
**Quick Sort** algorithm is one of the most used and popular algorithms in any programming language. But, if you are a JavaScript developer, then you might of heard of **sort()**which is already available in JavaScript. Then, you might have been thinking what the need of this Quick Sort algorithm is. To understand this, first we need what is sorting and what is the default sorting in JavaScript.

**What is Sorting?**

Sorting is nothing but, arranging elements in the order we want. You might have come across this in your school or college days. Like arranging numbers from smaller to greater (ascending) or from greater to smaller (descending) is what we saw till now and is called sorting.

**Default sorting in JavaScript**

As mentioned earlier, JavaScript has **sort()**. Let us take an example with few array of elements like [5,3,7,6,2,9] and want to sort this array elements in ascending order. Just call **sort()** on items array and it sorts array elements in ascending order.

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl1.png)

**Code:**

var items = [5,3,7,6,2,9];

console.log(items.sort());

//prints [2, 3, 5, 6, 7, 9]

**What is the reason to choose Quick sort over default sort() in JavaScript**

Though sort() gives the result we want, problem lies with the way it sorts the array elements. Default sort() in JavaScript uses **insertion sort** by **V8 Engine of Chrome** and **Merge sort** by **Mozilla Firefox** and **Safari**.

But, other this is not suitable if you need to sort large number of elements. So, the solution is to use Quick sort for large dataset.

So, to understand completely, you need to know how Quick sort works and let us see that in detail now.

**What is Quick sort?**

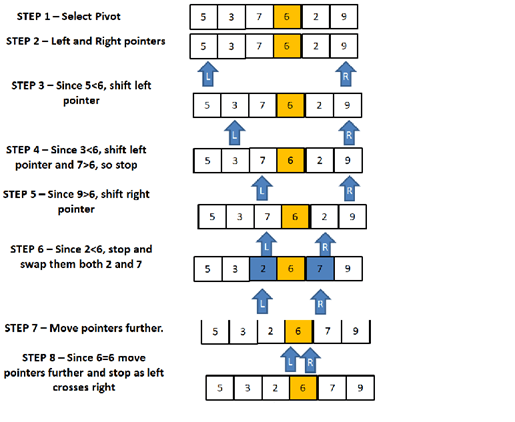
Quick sort follows **Divide and Conquer**algorithm. It is dividing elements in to smaller parts based on some condition and performing the sort operations on those divided smaller parts. Hence, it works well for large datasets. So, here are the steps how Quick sort works in simple words.

1. First select an element which is to be called as **pivot**element.
2. Next, compare all array elements with the selected pivot element and arrange them in such a way that, elements less than the pivot element are to it's left and greater than pivot is to it's right.
3. Finally, perform the same operations on left and right side elements to the pivot element.

So, that is the basic outline of Quick sort. Here are the steps which need to be followed one by one to perform Quick sort.

**How does QuickSort Work**

1. First find the **"pivot"** element in the array.
2. Start the left pointer at first element of the array.
3. Start the right pointer at last element of the array.
4. Compare the element pointing with left pointer and if it is less than the pivot element, then move the left pointer to the right (add 1 to the left index). Continue this until left side element is greater than or equal to the pivot element.
5. Compare the element pointing with right pointer and if it is greater than the pivot element, then move the right pointer to the left (subtract 1 to the right index). Continue this until right side element is less than or equal to the pivot element.
6. Check if left pointer is less than or equal to right pointer, then saw the elements in locations of these pointers.
7. Increment the left pointer and decrement the right pointer.
8. If index of left pointer is still less than the index of the right pointer, then repeat the process; else return the index of the left pointer.

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl2.png)

So, let us see these steps with an example. Let us consider array of elements which we need to sort is [5,3,7,6,2,9].

**Determine Pivot element**

But before going forward with the Quick sort, selecting the pivot element plays a major role. If you select the first element as the pivot element, then it gives worst performance in the sorted array. So, it is always advisable to pick the middle element (length of the array divided by 2) as the pivot element and we do the same.

Here are the steps to perform Quick sort that is being shown with an example [5,3,7,6,2,9].

**STEP 1:**Determine pivot as middle element. So, **7**is the pivot element.

**STEP 2:**Start left and right pointers as first and last elements of the array respectively. So, left pointer is pointing to **5** at index 0 and right pointer is pointing to **9** at index 5.

**STEP 3:**Compare element at the left pointer with the pivot element. Since, 5 < 6 shift left pointer to the right to index 1.

**STEP 4:**Now, still 3 <6 so shift left pointer to one more index to the right. So now 7 > 6 stop incrementing the left pointer and now left pointer is at index 2.

**STEP 5:**Now, compare value at the right pointer with the pivot element. Since 9 > 6 move the right pointer to the left. Now as 2 < 6 stop moving the right pointer.

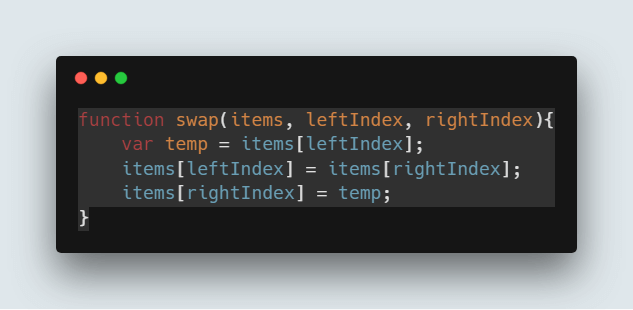
**STEP 6:**Swap both values present at left and right pointers with each other.

**STEP 7:**Move both pointers one more step.

**STEP 8:**Since 6 = 6, move pointers to one more step and stop as left pointer crosses the right pointer and return the index of the left pointer.

So, here based on the above approach, we need to write code for swapping elements and partitioning the array as mentioned in above steps.

**Code to swap two numbers in JavaScript:**

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl3.png)

function swap(items, leftIndex, rightIndex){

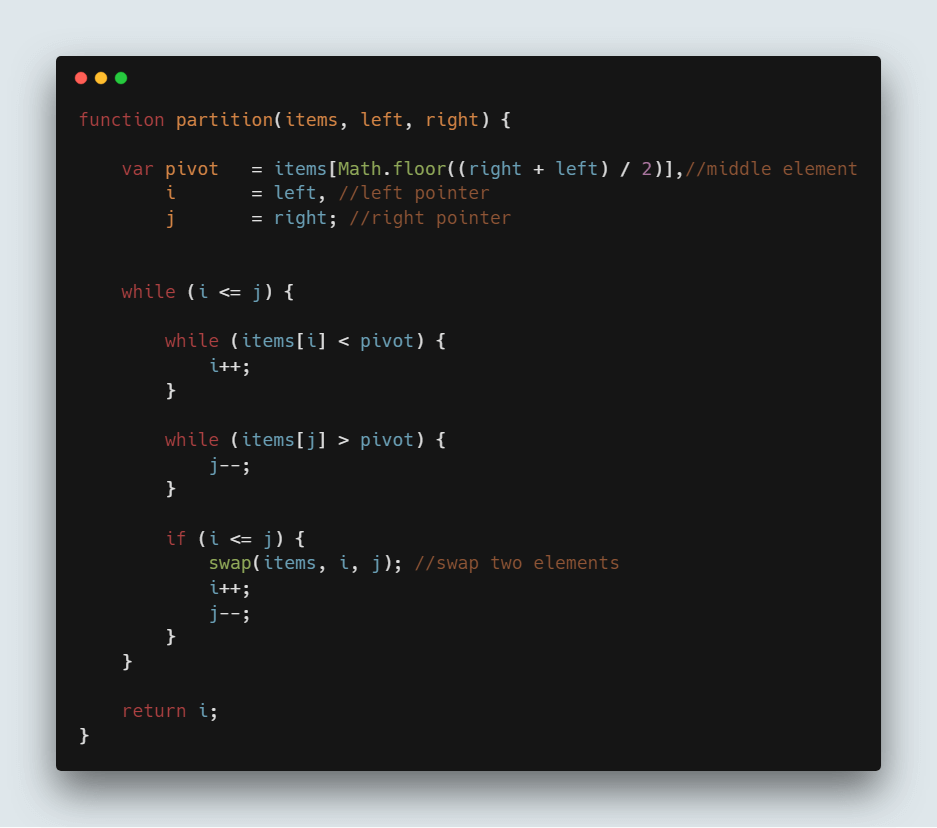
var temp = items[leftIndex];

items[leftIndex] = items[rightIndex];

items[rightIndex] = temp;

}

**Code to perform the partition as mentioned in above steps:**

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl4.png)

function partition(items, left, right) {

var pivot = items[Math.floor((right + left) / 2)], //middle element

i = left, //left pointer

j = right; //right pointer

while (i <= j) {

while (items[i] < pivot) {

i++;

}

while (items[j] > pivot) {

j--;

}

if (i <= j) {

swap(items, i, j); //swap two elements

i++;

j--;

}

}

return i;

}

**Perform the recursive operation**

Once you perform above steps, index of the left pointer will be returned and we need to use that to divide the array and perform the Quick sort on that part. Hence, it is called Divide and Conquer algorithm.

So, Quick sort is performed until all elements on the left array and right array are sorted.

**Note:** Quick sort is performed on the same array and no new arrays are created in the process.

So, we need to call this **partition()** explained above and based on that we divide the array in to parts. So here is the code where you use it,

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl5.png)

function quickSort(items, left, right) {

var index;

if (items.length > 1) {

index = partition(items, left, right); //index returned from partition

if (left < index - 1) { //more elements on the left side of the pivot

quickSort(items, left, index - 1);

}

if (index < right) { //more elements on the right side of the pivot

quickSort(items, index, right);

}

}

return items;

}

// first call to quick sort

var result = quickSort(items, 0, items.length - 1);

**Complete Quick sort code:**

var items = [5,3,7,6,2,9];

function swap(items, leftIndex, rightIndex){

var temp = items[leftIndex];

items[leftIndex] = items[rightIndex];

items[rightIndex] = temp;

}

function partition(items, left, right) {

var pivot = items[Math.floor((right + left) / 2)], //middle element

i = left, //left pointer

j = right; //right pointer

while (i <= j) {

while (items[i] < pivot) {

i++;

}

while (items[j] > pivot) {

j--;

}

if (i <= j) {

swap(items, i, j); //sawpping two elements

i++;

j--;

}

}

return i;

}

function quickSort(items, left, right) {

var index;

if (items.length > 1) {

index = partition(items, left, right); //index returned from partition

if (left < index - 1) { //more elements on the left side of the pivot

quickSort(items, left, index - 1);

}

if (index < right) { //more elements on the right side of the pivot

quickSort(items, index, right);

}

}

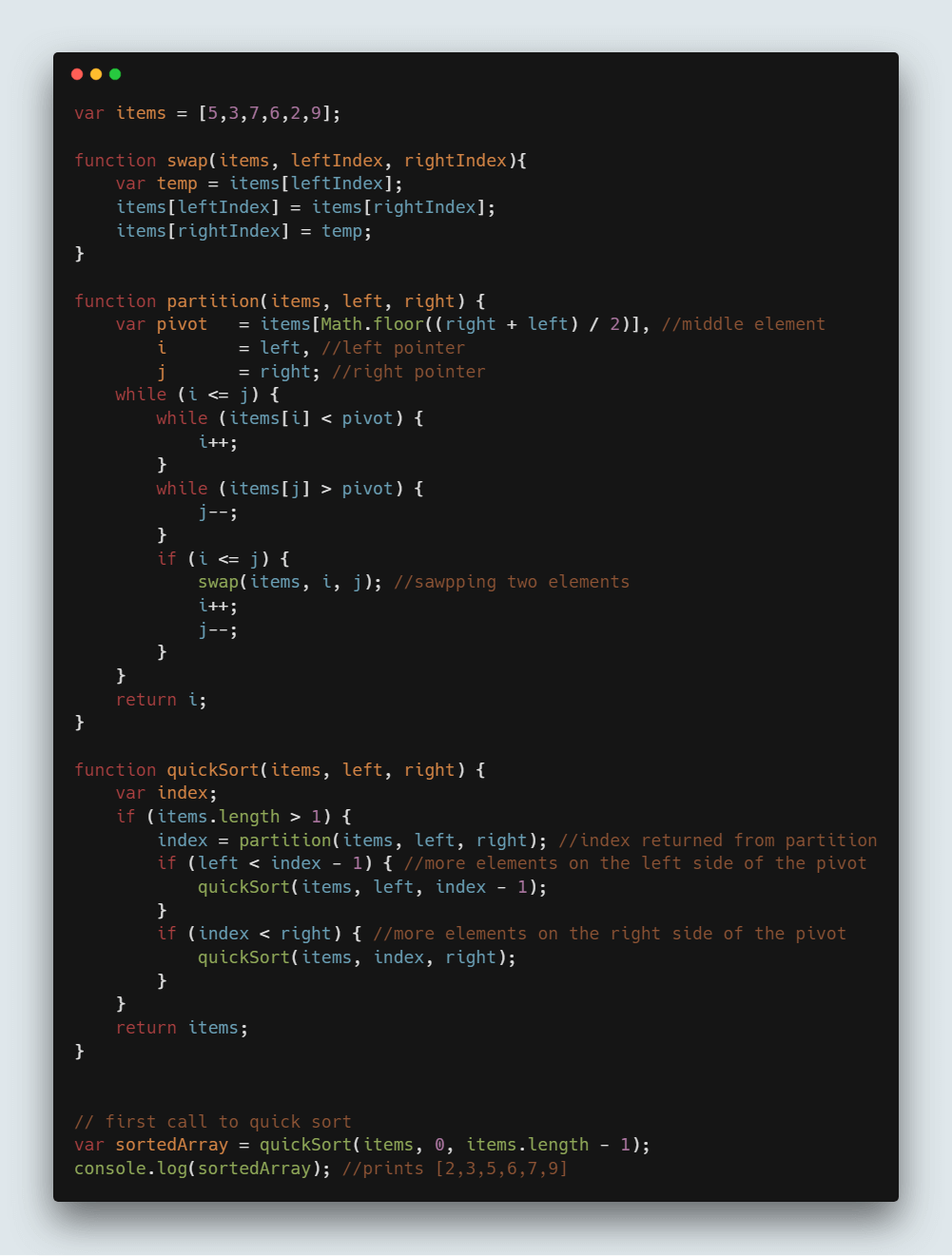
return items;

}

// first call to quick sort

var sortedArray = quickSort(items, 0, items.length - 1);

console.log(sortedArray); //prints [2,3,5,6,7,9]

[](https://www.guru99.com/images/1/011019_1052_QuickSortAl6.png)

**NOTE:**Quick sort runs with the Time Complexity of **O(nlogn).**

### 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.

## 2 Question

### What close() does in Javascript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/what-close-does-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-close-does-in-javascript/)

## #3 Question

### What is the difference between let and var?

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.

[0 Answer](https://www.onlineinterviewquestions.com/what-is-the-difference-between-let-and-var/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-the-difference-between-let-and-var/)

## #4 Question

### Explain Closures in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-closures-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-closures-in-javascript/)

## #5 Question

### Explain JavaScript Event Delegation Model?

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/)

[0 Answer](https://www.onlineinterviewquestions.com/explain-javascript-event-delegation-model/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-javascript-event-delegation-model/)

## #6 Question

### Describe negative infinity in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/describe-negative-infinity-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/describe-negative-infinity-in-javascript/)

## #7 Question

### Explain function hoisting in JavaScript?

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');

};

[0 Answer](https://www.onlineinterviewquestions.com/explain-function-hoisting-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-function-hoisting-in-javascript/)

## #8 Question

### What is the use of let & const?

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 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.

[0 Answer](https://www.onlineinterviewquestions.com/what-is-the-use-of-let-const/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-the-use-of-let-const/)

## #9 Question

### Explain Arrow functions?

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();

[0 Answer](https://www.onlineinterviewquestions.com/explain-arrow-functions/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-arrow-functions/)

## #10 Question

### What are exports and imports?

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};

[0 Answer](https://www.onlineinterviewquestions.com/what-are-exports-and-imports/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-are-exports-and-imports/)

## #11 Question

### What is difference between module.exports and export?

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

[0 Answer](https://www.onlineinterviewquestions.com/what-is-difference-between-module-export/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-difference-between-module-export/)

## #12 Question

### How to import all exports of a file as an object.

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();

[0 Answer](https://www.onlineinterviewquestions.com/how-to-import-all-exports-of-a-file-as-a/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-import-all-exports-of-a-file-as-a/)

## #13 Question

### Explain “use strict” ?

“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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-use-strict/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-use-strict/)

## #14 Question

### In Javascript are calculations with fractional numbers guaranteed to be precise?

NO, calculations with fractional numbers are not guaranteed to be precise in Javascript

[0 Answer](https://www.onlineinterviewquestions.com/javascript-calculations-fractional-numbers-guaranteed-precise/) [Suggest an edit](https://www.onlineinterviewquestions.com/javascript-calculations-fractional-numbers-guaranteed-precise/)

## #15 Question

### List the comparison operators supported by Javascript?

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

[0 Answer](https://www.onlineinterviewquestions.com/list-comparison-operators-supported-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-comparison-operators-supported-javascript/)

## #16 Question

### How do you declare variables in Javascript?

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/)

[0 Answer](https://www.onlineinterviewquestions.com/declare-variables-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/declare-variables-javascript/)

## #17 Question

### What will happen if an infinite while loop is run in Javascript?

The program will crash the browser.

[0 Answer](https://www.onlineinterviewquestions.com/will-happen-infinite-loop-ran-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/will-happen-infinite-loop-ran-javascript/)

## #18 Question

### List HTML DOM mouse events?

HTML DOM mouse events

* onclick
* ondblclick
* mousemove
* mousedown
* mouseover
* mouseout
* mouseup

[0 Answer](https://www.onlineinterviewquestions.com/list-html-dom-mouse-events/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-html-dom-mouse-events/)

## #19 Question

### How to get the last index of a string in Javascript?

**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);

[0 Answer](https://www.onlineinterviewquestions.com/get-last-index-string-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/get-last-index-string-javascript/)

## #20 Question

### How to get the primitive value of a string in Javascript?

In Javascript **valueOf()** method is used to get the primitive value of a string.

**Example Usage:**

var myVar= "Hi!"

console.log(myVar.valueOf())

[0 Answer](https://www.onlineinterviewquestions.com/get-primitive-value-string-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/get-primitive-value-string-javascript/)

## #21 Question

### What are the primitive data types in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/primitive-data-types-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/primitive-data-types-javascript/)

## #22 Question

### Explain Event bubbling and Event Capturing in JavaScript?

**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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-event-bubbling-and-event-capturing-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-event-bubbling-and-event-capturing-in-javascript/)

## #23 Question

### What does the instanceof operator do?

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

[0 Answer](https://www.onlineinterviewquestions.com/what-does-the-instanceof-operator-do/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-does-the-instanceof-operator-do/)

## #24 Question

### What is Javascript BOM?

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.

[0 Answer](https://www.onlineinterviewquestions.com/what-is-javascript-bom/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-javascript-bom/)

## #25 Question

### What are different types of Popup boxes available in Javascript?

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/)

[0 Answer](https://www.onlineinterviewquestions.com/different-types-popup-boxes-available-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/different-types-popup-boxes-available-javascript/)

## #26 Question

### How can you create an array in Javascript?

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');

[0 Answer](https://www.onlineinterviewquestions.com/can-create-array-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/can-create-array-javascript/)

## #27 Question

### 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, 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.

[0 Answer](https://www.onlineinterviewquestions.com/strict-mode-javascript-can-enabled/) [Suggest an edit](https://www.onlineinterviewquestions.com/strict-mode-javascript-can-enabled/)

## #28 Question

### How to calculate Fibonacci numbers in JavaScript?

#### 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;

}

[0 Answer](https://www.onlineinterviewquestions.com/calculate-fibonacci-numbers-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/calculate-fibonacci-numbers-javascript/)

## #29 Question

### What is the difference between the substr() and substring() functions in JavaScript?

#### 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

[0 Answer](https://www.onlineinterviewquestions.com/difference-substr-substring-functions-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/difference-substr-substring-functions-javascript/)

## #30 Question

### What are different types of Inheritence? Which Inheritance is followed in Javascript.

There are two types of Inherientence in OOPS Classic and Prototypical Inheritance. Javascript follows Prototypical Inheritance.

[0 Answer](https://www.onlineinterviewquestions.com/what-are-different-types-of-inheritence/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-are-different-types-of-inheritence/)

## #31 Question

### What is output of undefined \* 2 in Javascript?

nan is output of undefined \* 2.

[0 Answer](https://www.onlineinterviewquestions.com/what-is-output-of-undefined-2-in-javas/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-output-of-undefined-2-in-javas/)

## #32 Question

### How to add/remove properties to object dynamically in Javascript?

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);

[0 Answer](https://www.onlineinterviewquestions.com/how-to-add-remove-properties-to-object-d/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-add-remove-properties-to-object-d/)

## #33 Question

### How to convert Javascript date to ISO standard?

**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

[0 Answer](https://www.onlineinterviewquestions.com/how-to-convert-javascript-date-to-iso-standard/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-convert-javascript-date-to-iso-standard/)

## #34 Question

### How to get inner Html of an element in JavaScript?

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>

[0 Answer](https://www.onlineinterviewquestions.com/how-to-get-inner-html-of-an-element-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-get-inner-html-of-an-element-in-javascript/)

## #35 Question

### How to clone an object in Javascript?

Object.assign() method is used for cloning an object in Javascript.Here is sample usage

var x = {myProp: "value"};

var y = Object.assign({}, x);

[0 Answer](https://www.onlineinterviewquestions.com/how-to-clone-an-object-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-clone-an-object-in-javascript/)

## #36 Question

### List different ways of empty an array in Javascript?

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)

[0 Answer](https://www.onlineinterviewquestions.com/list-different-ways-of-empty-an-array-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-different-ways-of-empty-an-array-in-javascript/)

## #37 Question

### How to get an element by class in JavaScript ?

**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 |

[0 Answer](https://www.onlineinterviewquestions.com/how-to-get-an-element-by-class-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-get-an-element-by-class-in-javascript/)

## #38 Question

### Explain Typecasting in Javascript?

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

[0 Answer](https://www.onlineinterviewquestions.com/explain-typecasting-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-typecasting-in-javascript/)

## #39 Question

### How to encode and decode a URL in JavaScript?

**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);

[0 Answer](https://www.onlineinterviewquestions.com/how-to-encode-and-decode-a-url-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-encode-and-decode-a-url-in-javascript/)

## #40 Question

### How to you change the title of the page by JavaScript?

You can change the title of a webpage using setting the title property of the document object.

**Example usage**

document.title="My New Title";

[0 Answer](https://www.onlineinterviewquestions.com/how-to-you-change-the-title-of-the-page-by-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-you-change-the-title-of-the-page-by-javascript/)

## #41 Question

### What is difference between deep and shallow object coping in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/what-is-difference-between-deep-and-shallow-object-coping-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-difference-between-deep-and-shallow-object-coping-in-javascript/)

## #42 Question

### List some Unit Testing Frameworks JavaScript

Below is the list of few most Popular Javascript Unit Testing Frameworks:

* Unit.js
* Jasmine
* Karma
* Chai
* AVA
* Mocha
* JSUnit
* QUnit
* Jest

[0 Answer](https://www.onlineinterviewquestions.com/list-some-unit-testing-frameworks-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-some-unit-testing-frameworks-javascript/)

## #43 Question

### How to add a new property in existing function JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/how-to-add-a-new-property-in-existing-function-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-add-a-new-property-in-existing-function-javascript/)

## #44 Question

### Explain JavaScript Accessors ?

JavaScript Accessors

[1 Answer](https://www.onlineinterviewquestions.com/explain-javascript-accessors/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-javascript-accessors/)

## #45 Question

### List few difference between primitive and non primitive JavaScript data types?

* 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.

[0 Answer](https://www.onlineinterviewquestions.com/list-few-difference-between-primitive-and-non-primitive-javascript-data-types/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-few-difference-between-primitive-and-non-primitive-javascript-data-types/)

## #46 Question

### Explain higher-order functions in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-higher-order-functions-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-higher-order-functions-in-javascript/)

## #47 Question

### Explain few difference between null, undefined or undeclared JavaScript variable?

**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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-few-difference-between-null-undefined-or-undeclared-javascript-variable/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-few-difference-between-null-undefined-or-undeclared-javascript-variable/)

## #48 Question

### How host objects are different from native objects in JavaScript?

**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.

[0 Answer](https://www.onlineinterviewquestions.com/how-host-objects-are-different-from-native-objects-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-host-objects-are-different-from-native-objects-in-javascript/)

## #49 Question

### What is difference between var x =1; and x=1;?

[0 Answer](https://www.onlineinterviewquestions.com/what-is-difference-between-var-x-1-and-x-1/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-difference-between-var-x-1-and-x-1/)

## #50 Question

### Explain spread operator in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-spread-operator-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-spread-operator-in-javascript/)

## #51 Question

### How to remove duplicates from JavaScript Array?

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.

[0 Answer](https://www.onlineinterviewquestions.com/how-to-remove-duplicates-from-javascript-array/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-remove-duplicates-from-javascript-array/)

## #52 Question

### How to call a function in every x seconds in JavaScript?

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.

[1 Answer](https://www.onlineinterviewquestions.com/how-to-call-a-function-in-every-x-seconds-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-call-a-function-in-every-x-seconds-in-javascript/)

## #53 Question

### Explain Promise in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/explain-promise-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-promise-in-javascript/)

## #54 Question

### What is difference between Array.splice() and Array.slice() method in JavaScript?

* 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)

[0 Answer](https://www.onlineinterviewquestions.com/what-is-difference-between-array-splice-and-array-slice-method-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-difference-between-array-splice-and-array-slice-method-in-javascript/)

## #55 Question

### Is JavaScript multi-threaded or single-threaded?

JavaScript is single-threaded.

[1 Answer](https://www.onlineinterviewquestions.com/is-javascript-multi-threaded-or-single-threaded/) [Suggest an edit](https://www.onlineinterviewquestions.com/is-javascript-multi-threaded-or-single-threaded/)

## #56 Question

### Explain JavaScript Debounce Function?

[0 Answer](https://www.onlineinterviewquestions.com/explain-javascript-debounce-function/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-javascript-debounce-function/)

## #57 Question

### List some Design patterns in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/list-some-design-patterns-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-some-design-patterns-in-javascript/)

## #58 Question

### What is console.time() and console.timeEnd()? What is its syntax, and why is it used?

[0 Answer](https://www.onlineinterviewquestions.com/what-is-console-time-and-console-timeend-what-is-its-syntax-and-why-is-it-used/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-console-time-and-console-timeend-what-is-its-syntax-and-why-is-it-used/)

## #59 Question

### What are different types of Scope Chain available in JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/what-are-different-types-of-scope-chain-available-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-are-different-types-of-scope-chain-available-in-javascript/)

## #60 Question

### How to remove duplicate values from a JavaScript array?

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));

[0 Answer](https://www.onlineinterviewquestions.com/how-to-remove-duplicate-values-from-a-javascript-array/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-remove-duplicate-values-from-a-javascript-array/)

## #61 Question

### How to redirect a page to another page in Javascript?

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/;");

[0 Answer](https://www.onlineinterviewquestions.com/how-to-redirect-a-page-to-another-page-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/how-to-redirect-a-page-to-another-page-in-javascript/)

## #62 Question

### Is it possible to do 301 redirects in Javascript ?

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.

[0 Answer](https://www.onlineinterviewquestions.com/is-it-possible-to-do-301-redirects-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/is-it-possible-to-do-301-redirects-in-javascript/)

## #63 Question

### Write a program to reverse a string in pure JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/write-a-program-to-reverse-a-string-in-pure-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/write-a-program-to-reverse-a-string-in-pure-javascript/)

## #64 Question

### Write program to remove duplicate in an array ?

**#C program to remove duplicate programme:**

#include <stdio.h>

int main(){

int n, a[100], b[100], calc = 0, i, j;

printf("Enter no. of elements in array\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

for (i = 0; i < n; i++)

scanf("%d", &a[i]);

for (i = 0; i<n; i++) {

for (j = 0; j < calc; j++) {

if(a[i] == b[j])

**break**; }

if (j== calc){

b[count] = a[i];

calc++; } }

printf("Array obtained after removing duplicate elements:\n");

for (i = 0; i < calc; i++)

printf("%d\n", b[i]);

return 0;}

[0 Answer](https://www.onlineinterviewquestions.com/write-program-to-remove-duplicate-in-an-array/) [Suggest an edit](https://www.onlineinterviewquestions.com/write-program-to-remove-duplicate-in-an-array/)

## #65 Question

### List few Difference between JAVA and JavaScript?

[0 Answer](https://www.onlineinterviewquestions.com/list-few-difference-between-java-and-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-few-difference-between-java-and-javascript/)

## #66 Question

### Explain MUL function in Javascript?

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; };

};

}

[0 Answer](https://www.onlineinterviewquestions.com/explain-mul-function-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/explain-mul-function-in-javascript/)

## #67 Question

### List few advantages of using JavaScript?

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.

[0 Answer](https://www.onlineinterviewquestions.com/list-few-advantages-of-using-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/list-few-advantages-of-using-javascript/)

## #68 Question

### What is use of settimeout function in JavaScript?

[0 Answer](https://www.onlineinterviewquestions.com/what-is-use-of-settimeout-function-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-use-of-settimeout-function-in-javascript/)

## #69 Question

### What is difference between local and global scope in JavaScript ?

[0 Answer](https://www.onlineinterviewquestions.com/what-is-difference-between-local-and-global-scope-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-is-difference-between-local-and-global-scope-in-javascript/)

## #70 Question

### What are anonymous functions in JavaScript ?

[0 Answer](https://www.onlineinterviewquestions.com/what-are-anonymous-functions-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/what-are-anonymous-functions-in-javascript/)

## #71 Question

### Please explain equality operators in JavaScript?

[0 Answer](https://www.onlineinterviewquestions.com/please-explain-equality-operators-in-javascript/) [Suggest an edit](https://www.onlineinterviewquestions.com/please-explain-equality-operators-in-javascript/)

## #72 Question

### What are the different types of errors available in JavaScript?

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.

**Some Interesting facts about Javascript:**

1. JavaScript works on web users computers even when they are offline!
2. JavaScript is the world's most popular programming language.
3. World's most misunderstood programming language.
4. JavaScript will be used as long as people use the internet.
5. JavaScript is also known as Mocha, or LiveScript, or JScript, or ECMAScript.
6. Virtually every personal computer in the world has at least one JavaScript interpreter installed on it and in active use.
7. Javascript was popularly known as a Client Side language, but you can even send Server request with it.
8. With the popularity of Node.js, it can be used as both frontend and backend programming language, JavaScript has become a full-stack language.
9. In JavaScript, NAN (Not a Number) is of type Number. Unbelievable, isn't it?
10. JavaScript is the most-used programming language on Github.