Almost 70% of all the respondents of the Stackoverflow survey state that they have been using JS for their day-to-day development chores. Hence, following a career path in JavaScript development is lucrative.

**JavaScript Interview Questions**

Here we have compiled a list of important Javascript interview questions that will help you seize that upcoming JS interview.

**Question: What do you understand by JavaScript?**  
**Answer**: JavaScript is the most popular web scripting language, used for both client-side as well as server-side development. Supporting object-oriented programming abilities, the JavaScript code can be inserted into HTML pages that can be understood and executed by web browsers.

**Question: How is JavaScript different from Java?**  
**Answer**: Java is a programming language, whereas JavaScript is essentially a scripting language. Developers use Java for building applications that can run in a virtual machine, an operating system or inside a browser. Contrastingly, JS code is meant to run inside a browser only.

Nonetheless, standalone desktop applications can be built with JavaScript by using Electron and [Node.js](https://hackr.io/tutorials/learn-node-js?ref=blog-post). Another important distinction between Java and JS code is that while the former requires compilation, the latter is available in a text format.

**Question: Please explain the various JavaScript data types?**  
**Answer**: There are a total of 7 basic data types supported by JavaScript. Each one of them is briefed up as follows:

* **Boolean –** Represents true and false values
* **Null –** Represents empty, nothing, and unknown type of values
* **Number –** Represents both integer and floating point values
* **Object –** Used for storing collections of data or more complex entities
* **String –** Represents single-character, multi-character, and alphanumeric values
* **Symbol –** Used for creating unique identifiers for objects
* **Undefined –** Represents value not assigned. If a variable is only declared and not assigned in JS, then it represents the undefined data type

**Question: Could you enumerate the various features of JavaScript?**  
**Answer**: Some important features of JavaScript are:

* A lightweight interpreted a programming language with some object-oriented capabilities
* An open, cross-platform scripting language
* Complements and integrates with [the Java programming language](https://hackr.io/blog/what-is-java) as well as other backend technologies.
* Designed especially for creating network-centric applications

**Question: Please describe the most important advantages of using JavaScript?**  
**Answer**: There are several advantages to using JavaScript. Most notable amongst them are listed down as follows:

* **Enhanced interactivity –** JavaScript allows creating interfaces that react when the user activates them via the keyboard or merely hovers the cursor over the same
* **Immediate feedback –** Visitors need not to wait for a page reload to see if they had forgotten to enter some important details
* **Low server interaction –** JS allows validating user input before sending the webpage to the server. It means less server traffic and hence, less load on the server
* **Rich interfaces –** JS has items like drag-and-drop components and sliders to present a richer interface to the website visitors

**Question: Could you name some built-in methods in JavaScript?**  
**Answer**: Following are some of the inbuilt methods in JavaScript:

* **anchor() –** Creates an HTML anchor to be used as a hypertext target
* **ceil() –** returns the smallest integer that is greater than or equal to the given number
* **concat() –** Combines two strings and returns the newer string
* **constructor() –** Returns the function that created the corresponding instance of the object
* **Date() –** Returns the present date and time
* **Date.parse() –** Parses a string representation of a date and time, and then returns the internal millisecond representation for the same
* **exec() –** Searches for a match in the string parameter
* **filter() –** Creates a new array with all the elements of the array for which the filtering function returns true
* **fontcolor() –** Displays a string in the specified color
* **link() –** Creates an HTML hypertext link that requests another URL
* **localeCompare() –** Returns a number that indicates whether a reference string comes before, after, or is the same as the given string in the sort order
* **match() –** Used for matching a regular expression against a string
* **pop() –** Removes and returns the last element from an array
* **reduce() –** Applies a function simultaneously for two values of the array in order to reduce them to a single value
* **round() –** Rounds off the value of the given number to the nearest integer and returns the same
* **slice() –** Extracts a certain section of a string and returns the remaining string
* **some() –** returns true if at least one element of the array satisfies the provided testing function
* **toLocaleString() –** Return a string value of the current number in a format that depends on the browser’s locale settings
* **sup() –** Displays a string as a superscript
* **toSource() –** Returns a string containing the source of the Boolean object
* **toUpperCase() –** Converts a text to uppercase
* **valueOf() –** Returns the primitive value of the specified object

**Question: Explain the use of debugger in JavaScript?**  
**Answer**: All modern browsers (Mozilla Firefox, Safari, Google Chrome, etc.) come with an inbuilt debugger that can be summoned by pressing the F12 key. You need to choose the Console tab to view the result. Here you can set breakpoints as well as view the value of the variables.

JavaScript also features a debugger keyword that replicates the function of using breakpoints using a debugger. However, it works only when the debugging option is enabled in the web browser settings.

**Question: What are the different types of Error Name values in JavaScript?**  
**Answer**: There are 6 types of Error Name values. Each one of them is briefly explained as follows:

* **Eval Error – Thrown when coming across an error in eval() (Newer JS releases don’t have it)**
* Range Error – Generated when a number outside the specified range is used
* Reference Error – It comes into play when an undeclared variable is used
* Syntax Error – When the incorrect syntax is used, we get this error
* Type Error – This error is thrown when a value outside the range of data types is tried to be used
* URI Error – Generated due to the use of illegal characters

**Question: Please explain Self Invoking Function and its syntax.**  
**Answer**: Functions that are automatically invoked are termed as Self Invoking Functions. These are also known as Immediately Invoked Function Expressions and Self Executing Anonymous Functions. The general syntax of a Self Invoking Function is:

(some\_function () {

return () }) ();

Typically, a function is defined and then invoked. However, if there is a need to execute a function automatically at the place where it is given and it needs not to be called again then anonymous functions can be used. Such functions have no name, and thus the name.

**Question: Explain the difference between function declaration and function expression?**  
**Answer**: Following are the various differences between function declaration and function expression:

* Definition – A function that is declared as a separate statement in the main code flow is termed as the function declaration. When a function is created inside an expression or another syntax construct then it is called a function expression
* Strict Mode – When a function declaration is within a code block in the Strict mode, it is visible everywhere inside that block but not outside of it. This isn’t the case for a function expression
* Time of Use – A function expression is created when the execution reaches it. The function expression is usable only from that moment onwards. A function declaration, on the other hand, can be called before the same is defined
* When to Use – Function declaration offers better readability and offers more freedom in organizing the code. Function expressions are typically restricted to be used when there is need of a conditional declaration

**Question: Difference between attributes and property?**  
**Answer**: JS DOM objects have properties which are like instance variables for particular elements. A property can be of various data types. Properties are accessible by interacting with the object in [Vanilla JS](http://vanilla-js.com/) or using jQuery’s prop() method.

Rather than in the DOM, attributes are in the HTML. They are similar to properties but not as capable. It’s recommended to work with properties rather than attributes if the former is available. Unlike a property, an attribute is of the string data type.

**Question: What are the various ways of embedding JavaScript code in an HTML file?**  
**Answer**: There are 4 ways of embedding JS code within HTML documents:

* Adding it between <script> and </script> tags
* From an external file that is specified by the src attribute of a <script> tag
* In an HTML event handler attribute, such as onclick and onmouseover
* In a URL using the special javascript: protocol

**Question: Explain the difference between local storage & session storage?**  
**Answer**: Important differences between the local storage & session storage are listed down as follows:

* Data is stored in local storage with no expiration date. It gets cleared only via JS or when the browser cache is cleared. This isn’t the case with session storage that stores data only for a session. The data is cleared as soon as the browser or the tab is closed
* Session storage has a size limit more than a cookie but less than local storage

**Question: What are some of the best JavaScript frameworks?**  
**Answer**: Angular, React, and Meteor is 3 of the [most popular JavaScript frameworks](https://hackr.io/blog/10-best-javascript-frameworks-2019). Keep in mind that this list may get updated frequently.  
Each one of them is briefly explained as follows:

* **Angular** – Developed and maintained by Google, it extends the HTML into the JS application and interprets the attributes for performing data binding. Although Angular can be used for a variety of purposes, it is typically used for developing SPAs (Single-Page Applications)
* **React** – Developed and maintained by Facebook, it uses the virtual DOM and therefore, makes integration with any application easier. Used for developing the UI of web pages with high interactivity.
* **Meteor** – Used for a variety of web software development projects, varying from back-end development and database management to rendering the front-end of websites

**Question: Can you explain exports and imports in JavaScript?**  
**Answer**: Imports and exports in JavaScript help in writing modular code. Using imports and exports, it is possible to split the JS code into multiple files.

**Question: What do you understand by ViewState and SessionState?**  
**Answer**: The ViewState is specific to a webpage in a session while the SessionState is specific to user-specific data, accessible across all web pages in a web application.

**Question: How will you read and write files using JavaScript?**  
**Answer**: There are two ways of reading and writing files using JavaScript:

* Using the JavaScript extension
* Using a web page and ActiveX objects

**Question: Please explain how to convert the string of any base to integer in JavaScript?**  
**Answer**: JavaScript has the built-in parseInt() function for converting numbers between different bases. It has two parameters. While the first parameter is the string that is to be converted, the second parameter represents the base of the given string. The general syntax is:

parseInt (“String”, Base);

**Question: Please explain the NULL value in JavaScript?**  
**Answer**: Unlike undefined that represents that no value has been assigned, the NULL value represents assigning ‘nothing’ as a value to a variable.

**Question: Does concept level scope exist in JavaScript?**  
**Answer**: A variable declared inside a function has scope only inside the function in JS. So, the concept level scope doesn’t exist in JavaScript.

**Question: What are the various disadvantages of using innerHTML in JavaScript?**  
**Answer**: Following are the various disadvantages of using innerHTML in JS:

* Content gets replaced everywhere
* Doesn’t offer validation
* Slow execution because the entire innerHTML content is reparsed and build into elements

**Question: Can you explain exception handling in JS?**  
**Answer**: For exception handling, JavaScript offers the try…catch…finally construct and the throw operator. Although it is possible to catch programmer-generated and runtime exceptions in JS, there is no way of catching syntax errors. The general syntax of the try…catch…finally construct is:

try {

// code to be executed

[break;]

}

catch ( e ) {

// code to execute when an exception occurs

[break;]

}

[ finally {

// code that is to be executed regardless of the occurrence of the exception

}]

The try block is followed by either one catch block or one finally block or one of both. As soon as an exception occurs in the try block, it is placed in e and the catch block is executed.

The optional finally block executes the code specified inside it once the try and catch block complete execution.

**Question: How will you detect the operating system in the client machine using JavaScript?**  
**Answer**: For finding the operation system in the client machine, we can use the read-only Navigator.appversion property in JavaScript.

**Question: Please explain the role of break and continue statements?**  
**Answer**: The break statement is used for coming out of the present loop. In order to continue looping with a new recurrence, we use the continue statement.

**Question: How are the event handlers utilized in JavaScript?**  
**Answer**: Actions that result from activities like clicking a link or filling a form by the user are called events. In order to properly manage the execution of such events, event handlers are used.

In JavaScript, event handlers are an extra attribute of the object. This attribute contains the name of the event as well as the action that is to be taken when the event takes place.

**Question: How will you make the JS code hidden from old browsers that don’t support JavaScript?**  
**Answer**: In order to hide JavaScript code from old browsers, add <!– in the code just after the <script> tag and add //–> in the code just before the </script> tag. Old browsers treat this JS code as a long HTML comment.

Typically, JS code is hidden from old browsers for solving compatibility and UI issues. Interestingly, browsers that support JavaScript will take <!– and //–> as one-line comments.

**Question: What are the escape characters in JavaScript?**  
**Answer**: In JavaScript, we use escape characters, typically backslash (\ \) while working with special characters, such as ampersands (&), apostrophes (‘), double quotes (“ “), and single quotes (‘ ‘). Whatever enclosed within the escape characters gets displayed by the JavaScript.

Six additional escape characters are also available in JavaScript:

* \b – Backspace
* \f – Form feed
* \n – New line
* \r – Carriage return
* \t – Horizontal tabulator
* \v – Vertical tabulator

These aren’t in anyway executed in the HTML or JS code. These were originally designed for controlling fax machines, teletypes, and typewriters.

**Question: What do you understand by cookies? How will you create, read, and delete a cookie using JavaScript?**  
**Answer**: A cookie is simply data, usually small, sent from a website and stored on the user’s computer by the web browser used to access the website. It is a reliable way for websites to remember stateful information as well as record the browsing activity of the user.

The most basic way of creating a cookie using JS is to assign a string value to the document.cookie object. The general syntax is:

document.cookie = “key1 = value1; key2 = value2; … ; keyN= valueN; expires = date”;

**Reading a Cookie:**

Reading a cookie using JS is as simple as creating the same. As the value of the document.cookie object is the cookie, use this string whenever you wish to access the cookie.

The document.cookie string keeps a list of name = value pairs, where each pair is separated by a semicolon. The name represents the name of a cookie and the value represents the respective cookie’s string value. For breaking the string into key and value, you can use the split() method.

**Deleting a Cookie:**

For deleting a cookie using JavaScript, simply set the expiration date (expires) to a time that’s already past. Some web browsers don’t let you delete a cookie unless you don’t specify the path of the cookie. Hence, defining the cookie path is important to ensure that the right cookie is deleted.

**Question: What will be the output of the JS code below? Please explain.**

var y = 1;

if (function F(){})

{

y += typeof F;

}

console.log(y);

**Answer**: The output of the aforementioned JavaScript code will be 1undefined. The if condition statement in the code evaluates using eval. Hence, eval(function F(){}) will return function F(){}.

Inside the if statement, executing typeof F returns undefined because the if statement code executes at run time while the statement inside the if condition is being evaluated.

**Question:** **Can you differentiate between let and var?**  
**Answer**: Both let and var are used for variable and method declaration in JavaScript. However, the most important difference between the two JS keywords is that while the var keyword is function scoped, the let keyword is block scoped.

**Question:** **What do you understand by Closures in JavaScript?**  
**Answer**: Closures provide a better, concise, creative, and expressive way of writing code for JavaScript developers and programmers. Technically speaking, closures are a combination of lexical environment and function.

In other words, a closure is a locally declared variable that is related to a function and stays in the memory when the related function has returned. All local variables that were in-scope at the time of the closure creation are contained by the closure.

Following code snippet demonstrates using a normal function in JavaScript:

function greet(message) {

console.log(message);

}

function greeter(name, age) {

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

}

var message = greeter("Akhil", 26);

greet(message);

The aforementioned function can be represented in a better way by using closures. This is demonstrated in the following code snippet:

function greeter(name, age) {

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

return function greet() {

console.log(message);

};

}

// Generate the closure

var AkhilGreeter = greeter("Akhil", 26);

// Use the closure

AkhilGreeter();

**Question:** **Please explain NEGATIVE\_INFINITY in JavaScript.**  
**Answer**: The NEGATIVE\_INFINITY is a static property in JS that results when a negative number is divided by 0. Its important characteristics are:

* There is no need for creating a number of objects for accessing NEGATIVE\_INFINITY
* The value of the NEGATIVE\_INFINITY property is the same as the negative value of the infinity property of the global object

**Question:** **Is there any short and concise way of writing function expressions in JS?**  
**Answer**: JavaScript offers a short and concise way of writing function expressions known as arrow functions. This way of writing function expressions is preferred typically for non-method functions. The general syntax of an arrow function is:

const function\_name = ()=>{}

Arrow functions can’t be used as constructors. Moreover, they don’t provide support for arguments, new.target, super, and this.

**Question:** **How can you import all exports of a file as an object in JavaScript?**  
**Answer**: In order to import all exported members of an object, one can use:

import \* as object name from ‘./file.js’

The exported methods or variables can be easily accessed by using the dot (.) operator.

**Question:** **How will you empty an array in JavaScript?**  
**Answer**: There are multiple ways of emptying an array in JavaScript. Four of the most important ones are:

* By assigning an empty array:  
  var array1 = [1, 22, 24, 46];  
  array1 = [];
* By assigning array length to 0:  
  var array1 = [1, 22, 24, 46];  
  array1.length=0;
* By popping the elements of the array:  
  var array1 = [1, 22, 24, 46];  
  while(array1.length > 0) {  
  array1.pop();  
  }
* By using the splice array function:  
  var array1 = [1, 22, 24, 46];  
  array1.splice(0, array1.length)

**Question:** **What do you mean by Event Bubbling and Event Capturing?**  
**Answer**: There are two ways for accomplishing event propagation and the order in which an event is received in the HTML DOM API.

These are Event Bubbling and Event Capturing. In the former, the event is directed towards its intended target, whereas the event goes down to the element in the latter.

* **Event Capturing** – Also known as trickling, Event Capturing is rarely used. The process starts with the outermost element capturing the event and then propagating the same to the innermost element.
* **Event Bubbling** – In this process, the event gets handled by the innermost element first and then propagated to the outermost element.

**Question:** **In how many ways can you create an array in JS?**  
**Answer**: There are three different ways of creating an array in JavaScript, namely:

1. By creating instance of an array:  
   var someArray = new Array();
2. By using an array constructor:  
   var someArray = new Array(‘value1’, ‘value2’,…, ‘valueN’);
3. By using an array literal:  
   var someArray = [value1, value2,…., valueN];

**Question:** **Write a code to demonstrate how to get inner HTML of an element in JavaScript.**  
**Answer**:

<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>

**Question:** **How will you remove duplicates from a JS array?**  
**Answer**: There are several possible ways of eliminating duplicates from a JS array. Three most-used ones are described as follows:

1. By using Filter – It is possible to remove duplicates from an array in JavaScript by applying a filter to the same. In order to call the filter() method, three arguments are required. These are namely array, current element, and index of the current element.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));
2. By using the For loop – In this method of removing duplicate elements from an array, an empty array is used for storing all the repeating
3. elements.Array dups\_names = ['Akhil', 'Vijay', 'Swapnil', 'Akhil', 'Vijay'];
4. function dups\_array(dups\_names) {
5. let unique = {};
6. names.forEach(function(i) {
7. If (!unique[i]) {
8. unique[i] = true; }
9. });
10. return Object.keys(unique);}
11. Dups\_array(names);
12. By using Set – This is the simplest approach of removing duplicate elements from an array in JS. A set is an inbuilt object for storing unique values in an array. Here’s how to use it for eliminating repeating elements from an array:function uniquearray(array) {  
    let unique\_array= Array.from(set(array))  
    return unique\_array;}In the above example, a set is created out of the array having duplicate elements. As a set is an ordered collection of unique elements, the result is an array with non-repeating elements.

**Question:** **Can you draw a simple JavaScript DOM (Document Object Model)?**  
**Answer**:  


**Question:** **Write the code to force a page to load another page in JavaScript.**  
**Answer**:

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

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

**Question:** **Please explain the Strict mode in JavaScript. Also, tell how to enable it.**  
**Answer**: When in the Strict Mode, JS displays errors for some code segments that might be otherwise not available. In other words, the Strict mode adds certain compulsions to the JS. It is used for removing some code mistakes that result in dropped efficiency of JS engines.

In order to enable the Strict Mode, one needs to add the string literal “use strict” above the file that needs to be opened in the Strict Mode. For example:

function somefunction() {

"use strict";

var v = "Welcome to the Strict Mode";

**Question:** **What is the for-in loop in JavaScript? Give its syntax.**

**Answer**: The for-in loop is meant to be used for looping through the properties of a JavaScript object. Every iteration of the loop results in a property of the object getting associated with the variable name. The loop continues until all the object properties are exhausted.

The general syntax of using the for-in loop is:

for (variable name in object){

statement or block to execute

}

**Question:** **Tell us about the difference between .call() and .apply() functions. Give an example of demonstrating the difference between the two JS functions.**  
**Answer**: Both .call() and .apply() functions are almost identical in their use with a major exception in the way in which arguments are passed to the function.

Because arguments are to be passed in the .call() method, it is compulsory to know about the arguments of the function. On the other hand, .apply() method is used when the number of arguments is not known. Following example demonstrates using the two functions:

var someObject = {

myProperty : 'Foo',

myMethod : function(prefix, postfix) {

alert(prefix + this.myProperty + postfix);

}

};

someObject.myMethod('<', '>');

var someOtherObject = {

myProperty : 'Bar'

};

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

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

**Question:** **What role do deferred scripts play in JavaScript?**  
**Answer**: During page loading, the parsing of the HTML code is paused by default until the script hasn’t stopped executing. This results in a delay in the display of the webpage if the server is slow or the script that is to be loaded is bulky.

Using deferred scripts results into a delay in the execution of the script for the time the HTML parser is running. Hence, this results in a reduction in the loading time of the webpage.

**Question:** **Could you tell us about the various types of error constructors supported by JavaScript?**  
**Answer**: The Error constructor is responsible for creating an error object. Instances of the error objects are thrown when runtime errors occur. Moreover, the Error object can also be used as a base object for user-defined exceptions.

Other than the generic Error constructor, JS provides support for 7 error constructors that are:

* **EvalError –** Creates an error instance regarding the global function eval().
* **InternalError –** Creates an error instance regarding an internal error in the JS engine.
* **RangeError –** Creates an error instance regarding a numeric variable or parameter that is outside of its valid range.
* **ReferenceError –** Creates an error instance regarding de-referencing an invalid reference.
* **SyntaxError –** Creates an error instance regarding a syntax error occurring while parsing code in eval().
* **TypeError –** Creates an error instance regarding a parameter or variable not of a valid type.
* **URIError –** Creates an error instance regarding when invalid parameters are passed to the decodeURI() or encodeURI().

**Question:** **What do you understand by Screen objects? State their various properties.**  
**Answer**: In order to read the information from the client’s screen, screen objects are used in JavaScript. Properties of screen objects are:

* **AvailHeight –** Gives out the height of the client screen (Excludes taskbar)
* **AvailWidth –** Gives out the width of the client screen (Excludes taskbar)
* **ColorDepth –** Gives out the bit depth of images supported by the client screen
* **Height –** Gives out the total height of the client screen
* **Width –** Gives out the total width of the client screen

**Question:** **Could you explain escape() and unescape() functions?**

**Answer**: The escape() function allows for converting a string into a coded form in JavaScript. It is used for securely transferring information from one system to another over some network. For instance, consider the following code snippet:

<script>

document.write(escape("This string is encoded!"));

</script>

The output of the aforementioned code snipped will be something like this:

This%3F%20string%20is%20encoded%21

The unescape() function does the exact opposite of the escape() function i.e. it decodes a coded string into the original string. Therefore, the following code snippet:

<script>

document.write(unescape("This%3F%20string%20is%20encoded%21"));

</script>

Will yield the following output:

This string is encoded!

**Question:** **Please write JavaScript code for adding new elements dynamically.**  
**Answer**:

<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>

All done! That was the list of most important JavaScript interview questions. Hope these questions will help you get a step closer to your dream JavaScript job. All the best!