**<https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/>**

**JavaScript**

[**Explain what is Javascript? List some data types supported by Javascript?**](https://www.onlineinterviewquestions.com/list-some-basic-datatypes-supported-by-javascript/)

JavaScript is an object-oriented computer programming language commonly used to create interactive effects within web browsers.It was first used by the Netscape browser, that provides access to the HTML document object model (DOM), provides access to the browser object model (BOM). Javascript syntax looks a lot like java, c or c++ syntax.

Below is the list of data types supported by JavaScript:-

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

### [What close() does in Javascript?](https://www.onlineinterviewquestions.com/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.

### [What is the difference between let and var?](https://www.onlineinterviewquestions.com/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.

### [Explain function hoisting in JavaScript?](https://www.onlineinterviewquestions.com/explain-function-hoisting-in-javascript/)

JavaScript’s default behaviour that allows moving declarations to the top is called Hoisting. The 2 ways of creating functions in JavaScript are **Function Declaration** and **Function Expression**.

#### Function Declaration

A function with the specific parameters is known as function declarations. To create a variable in JavaScript is called declarations.

hoisted(); // logs "foo"

function hoisted() {

 console.log('foo');

}

***Function Expression***

When a function is created by using an expression it is called function expression.

notHoisted(); // TypeError: notHoisted is not a function

var notHoisted = function() {

  console.log('bar');

};

### [What is the use of let & const?](https://www.onlineinterviewquestions.com/what-is-the-use-of-let-const/)

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.

### [Explain Arrow functions?](https://www.onlineinterviewquestions.com/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();

### [What are exports and imports?](https://www.onlineinterviewquestions.com/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};

### [How to import all exports of a file as an object.](https://www.onlineinterviewquestions.com/how-to-import-all-exports-of-a-file-as-a/)

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.

objectname.member1;

objectname.member2;

objectname.memberfunc();

### [Explain “use strict” ?](https://www.onlineinterviewquestions.com/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.

### [What is the ‘Strict’ mode in JavaScript and how can it be enabled?](https://www.onlineinterviewquestions.com/strict-mode-javascript-can-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.

### [In Javascript are calculations with fractional numbers guaranteed to be precise?](https://www.onlineinterviewquestions.com/javascript-calculations-fractional-numbers-guaranteed-precise/)

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

### [List the comparison operators supported by Javascript?](https://www.onlineinterviewquestions.com/list-comparison-operators-supported-javascript/)

* > 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

### [How do you declare variables in Javascript?](https://www.onlineinterviewquestions.com/declare-variables-javascript/)

In Javascript variable are declared using the var keyword.A variable must begin with A **letter**, **$** or \_.

### [List HTML DOM mouse events?](https://www.onlineinterviewquestions.com/list-html-dom-mouse-events/)

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

### [How to get the last index of a string in Javascript?](https://www.onlineinterviewquestions.com/get-last-index-string-javascript/)

**string.length-1** is used to get the last index of a string in Javascript

var myString="JavascriptQuestions";

console.log(myString.length-1);

### [What are the primitive data types in JavaScript?](https://www.onlineinterviewquestions.com/primitive-data-types-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.

### [How to get the primitive value of a string in Javascript?](https://www.onlineinterviewquestions.com/get-primitive-value-string-javascript/)

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

var myVar= "Hi!"

console.log(myVar.valueOf())

### [Explain Event bubbling and Event Capturing in JavaScript?](https://www.onlineinterviewquestions.com/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:

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.

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.

### [What does the instanceof operator do?](https://www.onlineinterviewquestions.com/what-does-the-instanceof-operator-do/)

In Javascript **instanceof** operator checks whether the object is an instance of a class or not:

Square.prototype = new Square();

console.log(sq instanceof Square); // true

### [What is Javascript BOM?](https://www.onlineinterviewquestions.com/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.

### [What are different types of Popup boxes available in Javascript?](https://www.onlineinterviewquestions.com/different-types-popup-boxes-available-javascript/)

* Alert
* Confirm
* Prompt

### [How can you create an array in Javascript?](https://www.onlineinterviewquestions.com/can-create-array-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');

### [What is the difference between the substr() and substring() functions in JavaScript?](https://www.onlineinterviewquestions.com/difference-substr-substring-functions-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

### [What are different types of Inheritence? Which Inheritance is followed in Javascript.](https://www.onlineinterviewquestions.com/what-are-different-types-of-inheritence/)

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

### [What is output of undefined \* 2 in Javascript?](https://www.onlineinterviewquestions.com/what-is-output-of-undefined-2-in-javas/)

nan is output of undefined \* 2.

### [How to add/remove properties to object dynamically in Javascript?](https://www.onlineinterviewquestions.com/how-to-add-remove-properties-to-object-d/)

You can add a property to an object using object.property\_name =value , delete object.property\_name is used to delete a property.

let user = new Object();

// adding a property

user.name='Anil';

user.age =25;

console.log(user);

delete user.age;

console.log(user);

### [How to get inner Html of an element in JavaScript?](https://www.onlineinterviewquestions.com/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>

### [How to clone an object in Javascript?](https://www.onlineinterviewquestions.com/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);

### [How to get an element by class in JavaScript ?](https://www.onlineinterviewquestions.com/how-to-get-an-element-by-class-in-javascript/)

**document.getElementsByClassName()** method is used in Javascript to get an element with a class name.

### [How to you change the title of the page by JavaScript?](https://www.onlineinterviewquestions.com/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.

### document.title="My New Title";

### [What are the primitive data types in JavaScript?](https://www.onlineinterviewquestions.com/primitive-data-types-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.

## <https://www.fullstack.cafe/blog/top-26-javascript-interview-questions-and-answers-in-2019>

## What is Coercion in JavaScript?

In JavaScript conversion between different two build-in types called coercion. Coercion comes in two forms in JavaScript: explicit and implicit.

## What is Scope in JavaScript?

In JavaScript, each function gets its own *scope*. Scope is basically a collection of variables as well as the rules for how those variables are accessed by name. Only code inside that function can access that function's scoped variables.

A variable name has to be unique within the same scope. A scope can be nested inside another scope. If one scope is nested inside another, code inside the innermost scope can access variables from either scope.

## Explain equality in JavaScript

JavaScript has both strict and type–converting comparisons:

* **Strict comparison (e.g., ===)** checks for value equality without allowing *coercion*
* **Abstract comparison (e.g. ==)** checks for value equality with *coercion* allowed

## Explain Null and Undefined in JavaScript

JavaScript (and by extension TypeScript) has two bottom types: null and undefined. They are *intended* to mean different things:

* Something hasn't been initialized : undefined.
* Something is currently unavailable: null.

## Explain what a callback function is and provide a simple example.

A callback function is a function that is passed to another function as an argument and is executed after some operation has been completed. Below is an example of a simple callback function that logs to the console after some operations have been completed.

## What does "use strict" do?

The use strict literal is entered at the top of a JavaScript program or at the top of a function and it helps you write safer JavaScript code by throwing an error if a global variable is created by mistake. For example, the following program will throw an error

## How would you check if a number is an integer?

A very simply way to check if a number is a decimal or integer is to see if there is a remainder left when you divide by 1.

## Write a function that would allow you to do this

var addSix = createBase(6);

addSix(10); // returns 16

addSix(21); // returns 27

**Answer:**  
You can create a closure to keep the value passed to the function createBase even after the inner function is returned. The inner function that is being returned is created within an outer function, making it a closure, and it has access to the variables within the outer function, in this case the variable baseNumber.

function createBase(baseNumber) {

return function(N) {

// we are referencing baseNumber here even though it was declared

// outside of this function. Closures allow us to do this in JavaScript

return baseNumber + N;

}

}

var addSix = createBase(6);

addSix(10);

addSix(21);

## Explain Values and Types in JavaScript

JavaScript has typed values, not typed variables. The following built-in types are available:

* string
* number
* boolean
* null and undefined
* object
* symbol (new to ES6)

## Explain event bubbling and how one may prevent it

**Event bubbling** is the concept in which an event triggers at the deepest possible element, and triggers on parent elements in nesting order. As a result, when clicking on a child element one may exhibit the handler of the parent activating.

One way to prevent event bubbling is using event.stopPropagation() or event.cancelBubble on IE < 9.

## What is let keyword in JavaScript?

In addition to creating declarations for variables at the function level, ES6 lets you declare variables to belong to individual blocks (pairs of { .. }), using the let keyword.

## What is IIFEs (Immediately Invoked Function Expressions)?

It’s an Immediately-Invoked Function Expression, or IIFE for short. It executes immediately after it’s created:

(function IIFE(){

console.log( "Hello!" );

})();

// "Hello!"

This pattern is often used when trying to avoid polluting the global namespace, because all the variables used inside the IIFE (like in any other normal function) are not visible outside its scope.

## Could you explain the difference between ES5 and ES6

* **ECMAScript 5 (ES5)**: The 5th edition of ECMAScript, standardized in 2009. This standard has been implemented fairly completely in all modern browsers
* **ECMAScript 6 (ES6)/ ECMAScript 2015 (ES2015)**: The 6th edition of ECMAScript, standardized in 2015. This standard has been partially implemented in most modern browsers.
* **Const**.   
  Const works like a constant in other languages in many ways but there are some caveats. Const stands for ‘constant reference’ to a value. So with const, you can actually mutate the properties of an object being referenced by the variable. You just can’t change the reference itself.

const NAMES = [];

NAMES.push("Jim");

console.log(NAMES.length === 1); // true

NAMES = ["Steve", "John"]; // error

* **Block-scoped variables**.  
  The new ES6 keyword let allows developers to scope variables at the block level. Let doesn’t hoist in the same way vardoes.
* **Default parameter values** Default parameters allow us to initialize functions with default values. A default is used when an argument is either omitted or undefined — meaning null is a valid value.

// Basic syntax

function multiply (a, b = 2) {

return a \* b;

}

multiply(5); // 10

* **Class Definition and Inheritance**  
  ES6 introduces language support for classes (class keyword), constructors (constructor keyword), and the extendkeyword for inheritance.
* **for-of operator**  
  The for...of statement creates a loop iterating over iterable objects.
* **Spread Operator** For objects merging

const obj1 = { a: 1, b: 2 }

const obj2 = { a: 2, c: 3, d: 4}

const obj3 = {...obj1, ...obj2}

* **Promises**  
  Promises provide a mechanism to handle the results and errors from asynchronous operations. You can accomplish the same thing with callbacks, but promises provide improved readability via method chaining and succinct error handling.

const isGreater = (a, b) => {

return new Promise ((resolve, reject) => {

if(a > b) {

resolve(true)

} else {

reject(false)

}

})

}

isGreater(1, 2)

.then(result => {

console.log('greater')

})

.catch(result => {

console.log('smaller')

})

* **Modules exporting & importing** Consider module exporting:

const myModule = { x: 1, y: () => { console.log('This is ES5') }}

export default myModule;

and importing:

import myModule from './myModule';