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

**1) What is Javascript? List some data types supported by Javascript?**

Answer : Javascript is an object-oriented language commonly used to create interactive effects within web browsers. It 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++.

Below is the list of data types that Javascript supports :

- Undefined

- Null

- Boolean

- String

- Symbol

- Number

- Object

**2) What close() does in Javascript?**

Answer : 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.

**3) What is the difference between let and var?**

Answer : The main difference is in the scope. ‘var’ is function scoped whereas ‘let’ is block scoped.

**4) Explain Closures in Javascript?**

Answer : A closure is a feature in Javascript where an inner function has access to the outer (enclosing) function’s variables — a scope chain.

The closure has three scope chains:

* it has access to its own scope — variables defined between its curly brackets
* it has access to the outer function’s variables
* it has access to the global variables

(<https://medium.freecodecamp.org/javascript-closures-simplified-d0d23fa06ba4>)

**5) Explain Javascript Event Delegation Model?**

Answer : Event Delegation Model allows functions to implement one single handler to many elements at one particular time. Also it allows you to add event listeners to one parent instead of specified nodes. The particular listener analyzes bubbled events to find a match on the child elements. To understand this concept, please read about **Event Listener** and **Event Bubbling** (Question 6 and 7)

Ex : <ul class=”characters”></ul>

<script>

const characterList = document.querySelector(‘.characters’);

characterList.**addEventListener(‘click’,myFunction); // will add click event to every child element of ‘characters’ ul.**

</script>

To set event for a particular type of element (say <**li>**) we will use this statement :

**if (!event.target.matches(‘li’)) { return; }**

(<https://medium.com/@bretdoucette/part-4-what-is-event-delegation-in-javascript-f5c8c0de2983>)

**6) What is Event Listener?**

Answer : The EventListener interface represents an object that can handle an event dispatched by an EventTarget object. Basically, an event listener is something that listens for an event.

To add an event listener on an HTML element we use the addEventListener() method.

Ex : const character = document.getElementId(‘element-id’);

character.**addEventListener(‘click’,myFunction)**;

This works fine if the element with id ‘element-id’ exists on the page when the page is loaded. However what happens to the event listener when the element is added to the DOM after the page load? **Therefore, Event Delegation is needed here.**

**7) What is Event Bubbling?**

Answer : Whenever a user makes a click (an event) it will ripple all the way up to the top of the DOM and triggers **click events on all the parent elements of the elements the user clicked.** Because of it’s nature, event bubbling (**or event propagation**) basically means that anytime you trigger a child element, you are effectively triggering all of it’s parent.

**8) Describe negative infinity?**

Answer : Represents negative infinity and is a number in javascript, which is defined by dividing negative number by zero. It can be understood as a number that is lower than any other number. It’s properties are as follow :

- A number need 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.

- Any positive value, including positive infinity (PI) multiplied by negative infinity (NI) is NI.

- Any negative value, including NI multiplied by NI is PI.

- Zero multiplied by NI is NaN.

- NaN multiplied by NI is NaN.

- NI divided by any negative value except NI is PI.

- NI divided by any positive value except PI is NI.

- NI divided by either NI or PI is NaN.

- Any number divided by NI is zero.

**9) Explain function hoisting in Javascript**Answer : In Javascript, a variable can be declared after it has been used. This is because of Javascript’s default behavior of moving all declarations to the top of the current scope (top of the script or function). **Note : will not work in strict mode, and variables declared with ‘let’ and ‘const’ are not hoisted.**

**10) How many ways to create functions?**

Answer : Two ways, Function Declaration and Function Expression.

**Function Declaration (hoisting available)**

[myFunction(); // will log ‘foo’](https://medium.com/@bretdoucette/part-4-what-is-event-delegation-in-javascript-f5c8c0de2983)

function myFunction() { console.log(‘foo’); }

**Function Expression (hoisting not available)**

myFunction(); // TypeError : myFunction is not a function.

var myFunction = function() { console.log(‘bar’); }

**11) What is the use of ‘let’ and ‘const’?**

Answer : In modern Javascript let & const are different ways of creating variables. Earlier (before ES6) we use the **var** keyword for creating variables. The two new keywords were used to create two different types of variables, which is immutable and mutable.

**- const** 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** is used to create a mutable variable. Mutable variables are normal variables like var that can be changed any number of time.

**12) Explain Arrow Functions?**

Answer : An arrow function is a short way to write function expressions **in ES6 or above.** Array functions cannot be used as constructors and also **does not support this, arguments, super or new.target** keywords. It is best suited for non-method functions.

Ex : const greet=()=>{ console.log(‘Hello’); }

greet();

**13) What are exports and imports?**

Answer : 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.

Ex :

// 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};

**14) How to import all exports of a file as an object?**

Answer : Import \* as object name from ‘./file.js’ is used to import all exported members as an object. We can simply access the exported variables or methods using dot (.) operator of the object.

Ex : import \* as objectName from ‘./file’;

console.log(objectName.member1);

**14) What is Strict mode?**

Answer : The purpose of using “use strict” directive is to enforce the code is executed in strict mode (available in Es5 and above). In strict mode, the previously bad syntax will become real errors.

**15) Are calculation with fractional numbers guaranteed to be precise?**

Answer : No. Because all numbers in JavaScript are stored as **64-bits Floating point numbers.**

Ex : var x = 0.1;

var y = 0.2;

var z = x + y; // z will not be 0.3, rather 0.30000000000000004

**16) How do you declare variables in JavaScript?**

Answer : Variables are declared using the **var, let, const keyword.** A variable must begin with a **letter, $ or \_** (PS : all variables in JavaScript are Case sensitive).

**17) What will happen if an infinite while loop is run in JavaScript?**

Answer : The program will crash the browser.

**18) How to get the primitive value of a string in JavaScript?**

Answer : **valueOf()** method is used.

**19) What are the primitive data types in JavaScript?**

Answer : There are 5 (undefined, null, boolean, string, number)

**20) What is Event Capture?**

Answer : The process is also called ‘trickling’, in this process the event is captured first by the outermost element and then propagated to the innermost element. Modern browser doesn’t support event capturing by default but we can achieve that by code in JavaScript. Both **Event Capture** and **Event Bubbling** are two ways of event propagation and determines the order in which event will be received.

**21) What does the instanceof operator do?**

Answer : checks whether the object is an instance of a class or not.

**22) What is JavaScript BOM?**