**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?**

Answer : BOM stands for “Browser Object Modal” that allows Javascript to “talk” to browser. BOM consists of the objects **navigator, history, screen, location and document which are children of window.** In the **document** node is the DOM (Document Object Model), which represents the contents of the page.

**23) What are different types of Popup boxes available in Javascript?**

Answer : Alert, Confirm, Prompt

**24) How many ways to create an array in Javascript?**

Answer : There are 3 ways

- By array literal :

var myArray = [val1,val2,...valN];

- By creating instance of Array :

var myArray = new Array();

- By using an Array constructor :

var myArray = new Array(‘val1’,’val2’,…,’valN’);

**25) What is ‘Strict’ mode in Javascript and how can it be enabled?**

Answer : Strict mode is a way to introduce better-checking into our code. In strict mode, we cannot, for example, use implicitly (khai báo ngầm) declared variables, or assign a value to a read-only property, or add a property to an object that is not extensible.

We can enable strict mode in a selected context only (whole file or just within a function). If strict mode is declared globally, all the code in the program will be affected.

**26) What is the difference between the substr() and substring() functions?**

Answer :

- The substr() function has the form substr(startIndex,length). It returns the substring from startIndex and returns ‘length’ number of characters.

- The substring() function has the form substring(startIndex,endIndex). It returns the substring from startIndex up to endIndex-1.

**27) What are different types of Inheritence? Which Inheritance is followed in Javascript?**

Answer : There are two types of Inheritance in Object Oriented Programming System (OOPS), which is Classic and Prototypical. Javascript follows Prototypical Inheritance.

**28) What is the output of undefined \* 2 in Javascript?**

Answer : NaN

**29) How to add/remove properties to/off object dynamically?**

Answer : We can add a property to an object using **object.property\_name = value**; to delete a property we use **delete object.property\_name.**

Ex : let user = new Object();

// adding property

user.name = “Anthony”;

user.age = 25;

console.log(user);

delete user.age;

console.log(user);

**30) How to convert Javascript date to ISO standard?**

Answer : **toISOString()** method is used to convert javascript date to ISO standard.

Ex : var date = new Date();

var n = date.toISOString();

console.log(n);

// YYYY-MM-DDTHH:mm:ss.sssZ

**31) How to get inner HTML of an element in Javascript?**

Answer : InnerHTML property of HTML DOM is used.

Ex : var inner = document.getElementById(‘íd’).innerHTML;

**32) How to clone an object in Javascript?**

Answer : Object.assign();

Ex : var x = {myProp : “value” };

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

**33) List different ways of empty an array in Javascript?**

Answer : There are 4 major ways

- Assigning an empty array

Ex : var arr = [1,2,3,4,5];

arr = [];

- Assigning array length to zero.

Ex : var arr = [1,2,3,4,5];

arr.length = 0;

- Poping elements out of the array

Ex : var arr = [1,2,3,4,5];

while (arr.length > 0) {

arr.pop();

}

- Using splice()

Ex : var arr = [1,2,3,4,5];

arr.splice(0,arr.length);

**34) How to get an element by class in Javascript?**

Answer : **document.getElementsByClassName()**

**35) Explain Typecasting in Javascript**

Answer : We can do this via library functions. There are basically 3 typecasts are available in Javascript.

- Boolean(value) : casts the inputted value to Boolean.

- Number(value) : casts the inputted value to Integer or Floating point Number.

- String(value) : casts the inputted value as a string.

**36) How to encode and decode URL?**

Answer :

- **encodeURI()** function is used to encode an URL.It takes a url string as parameter and return encoded string. Note : this method does not encode special characters (**/ ? : @ & = + $ #), please use encodeURIComponent() in this case.**

**Ex : var uri = “href”;**

**var encoded\_uri = encodeURI(uri);**

**- decodeURI() is used to decode an URL**

**Ex : var uri = “href”;**

**var encoded\_uri = encodeURI(uri);**

**decodeURI(encoded\_uri);**

**37) How to change the title of the page?**

**Answer : document.title = “new title”;**

**38) What is the difference between deep and shallow object coping?**

**Answer :**

**- Deep copy means copies all values or properties recursively in the new object whereas shallow copy only copies the reference.**

**- In deep copy, changes in the new object will not affect old object whereas in shallow copy changes in new object will reflect in the original object.**

**- In deep copy, original objects do not share the same properties with new object whereas in shallow copy they do.**

**39) What’s the difference between primitive and non-primitive data types?**

**Answer :**

**- Primitive data types are number, string, boolean, null, undefined and anything other than these types are non-primitive such as objects and functions.**

**- Primitive data types are immutable while non-primitive are not.**

**- Primitive data types are compared using their value, it means two values are strictly equal if they have the same data type and hold the same value.**

**- Non-primitive are not compared with values. Even if two objects have the same properties and values, they are strictly not equal.**

**40) Explain higher-order functions?**

**Answer : Higher order function is the function that takes a function as an argument and return a function as a result. Some of the built-in functions are mapping, filtering, reductions,zipping...**

**Ex :**

**var attitude = function(original, replacement, source){**

**return function(source) {**

**return source.replace(original,replacement);**

**}**

**}**

**var test = attitude(/millenials/ig,”Snake Prople”);**

**console.log(test(“The Millenials are always up to something”);**

**// will log “The Snake People are always up to something/”**

**41) Explain the deifferences between null, undefined or undeclared?**

**Answer :**

**- null is a value that can be assigned to a variable or object.**

**- undefined means the variable was declared but no values were assigned to it.**

**- undeclared means the variable was declared without any datat ype;**

**42) How host objects are different from native objects?**

**Answer :**

**- Host objects are those objects which environment gives. It means they are different per environment. For example, browsers include objects such as windows but Node.js give objects such as Node List.**

**- Native objects are built-in objects in Javascript. They are also known as global objects because they will be available to us independent of environment we are working in.**

**43) What’s the different between var x = 1 and x = 1?**

**Answer :**

**- var x = 1 will create a variable in the current scope; x will not be available outside the scope.**

**- x = 1 will create a variable within the global scope. Thus, any other code can access and alter it’s value. It’s a very bad practice to use variables in global scope.**

**44) Explain spread operator?**

**Answer : The spread operator expression in places where multiple argument/variables/elements are needed to present. It is represented with three dots (...)**

**Ex :**

**var mid = [3,4];**

**var newArr = [1,2,...mid,5,6];**

**console.log(newArr); // 1,2,3,4,5,6**

In above example, instead of appending mid array, it rather expands in the new array with the help of spread operator. This is how spread operator works in JavaScript.