Beginner Level JavaScript Interview Questions and Answers

Q1. What is the difference between Java & JavaScript?

Java	JavaScript
Java is an OOP programming language.	JavaScript is an OOP scripting language.
It creates applications that run in a virtual machine or browser.	The code is run on a browser only.
Java code needs to be compiled.	JavaScript code are all in the form of text.

Q2. What is JavaScript?

JavaScript is a **lightweight**, **interpreted** programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Q3. What are the data types supported by JavaScript?

The **data types** supported by JavaScript are:

- Undefined
- Null
- Boolean
- String
- Symbol
- Number
- Object

Q4. What are the features of JavaScript?

Following are the **features** of JavaScript:

- It is a **lightweight, interpreted** programming language.
- It is designed for creating **network-centric** applications.
- It is complementary to and **integrated** with Java.
- It is an **open** and **cross-platform** scripting language.

Q5. Is JavaScript a case-sensitive language?

Yes, JavaScript is a **case sensitive** language. The language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

Q6. What are the advantages of JavaScript?

Following are the **advantages** of using JavaScript −

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Q7. How can you create an object in JavaScript?

JavaScript supports **Object** concept very well. You can create an object using the **object literal** as follows –

```
var emp = {
name: "Daniel",
age: 23
};
```

Q8. How can you create an Array in JavaScript?

You can define arrays using the **array literal** as follows-

```
var x = [];
var y = [1, 2, 3, 4, 5];
```

Q9. What is a name function in JavaScript & how to define it?

A named function declares a name as soon as it is defined. It can be defined using **function** keyword as:

```
function named(){
        //write code here
}
```

Q10. Can you assign an anonymous function to a variable and pass it as an argument to another function?

Yes! An anonymous function can be assigned to a variable. It can also be passed as an argument to another function.

In case you are facing any challenges with these JavaScript Interview Questions, please comment on your problems in the section below.

Q11. What is argument objects in JavaScript & how to get the type of arguments passed to a function?

JavaScript variable arguments represents the **arguments** that are passed to a function. Using **typeof** operator, we can get the type of arguments passed to a function. For example –

```
function func(x){
console.log(typeof x, arguments.length);
```

```
}
func(); //==> "undefined", 0
func(7); //==> "number", 1
func("1", "2", "3"); //==> "string", 3
```

Q12. What are the scopes of a variable in JavaScript?

The scope of a variable is the **region** of your program in which it is **defined**. JavaScript variable will have only two scopes.
• **Global Variables** − A global variable has global scope which means it is visible everywhere in your JavaScript code.
• **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Q13. What is the purpose of 'This' operator in JavaScript?

The JavaScript **this** keyword refers to the object it belongs to. This has different values depending on where it is used. In a method, this refers to the owner object and, in a function, this refers to the global object.

Q14. What is Callback?

A **callback** is a plain JavaScript function passed to some method as an argument or option. It is a function that is to be **executed** after another function has finished executing, hence the name '**call back**'. In JavaScript, functions are objects. Because of this, functions can take functions as arguments, and can be returned by other functions.

Q15. What is Closure? Give an example.

Closures are created whenever a variable that is defined outside the **current scope** is accessed from within some inner scope. It gives you access to an outer function's scope from an inner function. In JavaScript, closures are created every time a function is created. To use a closure, simply define a function inside another function and expose it.

Q16. Name some of the built-in methods and the values returned by them.

Built-in Method	Values
CharAt()	It returns the character at the specified index.
Concat()	It joins two or more strings.
forEach()	It calls a function for each element in the array.
indexOf()	It returns the index within the calling String object of the first occurrence of the specified value.
length()	It returns the length of the string.
pop()	It removes the last element from an array and returns that element.
push()	It adds one or more elements to the end of an array and returns the new length of the array.
reverse()	It reverses the order of the elements of an array.

Q17. What are the variable naming conventions in JavaScript?

The following **rules** are to be followed while **naming variables** in JavaScript:

1. You should not use any of the JavaScript **reserved keyword** as variable name. For example, break or Boolean variable names are not valid.
2. JavaScript variable names should not start with a **numeral** (0-9). They must begin with a letter or the underscore character. For example, 123name is an invalid variable name but _123name or name123 is a valid one.
3. JavaScript variable names are **case sensitive**. For example, Test and test are two different variables.

Q18. How does TypeOf Operator work?

The **typeof** operator is used to get the data type of its operand. The operand can be either a **literal** or a **data structure** such as a variable, a function, or an object. It is a **unary** operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

Q19. How to create a cookie using JavaScript?

The simplest way to create a cookie is to assign a string value to the **document.cookie** object, which looks like this-

Syntax :

```
document.cookie = "key1 = value1; key2 = value2; expires = date";
```

Q20. How to read a cookie using JavaScript?

Reading a cookie is just as simple as writing one, because the value of the document.cookie object is the cookie. So you can use this string whenever you want to access the cookie.

- The **document.cookie** string will keep a list of name = value pairs separated by semicolons, where name is the name of a cookie and value is its string value.
- You can use strings' **split()** function to break the string into key and values.

Q21. How to delete a cookie using JavaScript?

If you want to delete a cookie so that subsequent attempts to read the cookie in JavaScript return nothing, you just need to set the expiration date to a time in the past. You should define the cookie path to ensure that you delete the right cookie. Some browsers will not let you delete a cookie if you don't specify the path.

Intermediate Level JS Interview Questions and Answers

Q22. What is the difference between Attributes and Property?

Attributes- provide more details on an element like id, type, value etc.

Property- is the value assigned to the property like type="text", value='Name' etc.

Q23. List out the different ways an HTML element can be accessed in a JavaScript code.

Here are the list of ways an HTML element can be accessed in a Javascript code:
(i) **getElementById('idname'):** Gets an element by its ID name
(ii) **getElementsByClass('classname'):** Gets all the elements that have the given classname.
(iii) **getElementsByTagName('tagname'):** Gets all the elements that have the given tag name.
(iv) **querySelector():** This function takes css style selector and returns the first selected element.

Q24. In how many ways a JavaScript code can be involved in an HTML file?

There are 3 different ways in which a JavaScript code can be involved in an HTML file:

- **Inline**
- **Internal**
- **External**

An **inline** function is a JavaScript function, which is assigned to a variable created at runtime. You can differentiate between Inline Functions and Anonymous since an inline function is assigned to a variable and can be easily reused. When you need a JavaScript for a function, you can either have the script **integrated** in the page you are working on, or you can have it placed in a **separate** file that you call, when needed. This is the difference between an **internal** script and an **external** script.

Q25. What are the ways to define a variable in JavaScript?

The three possible ways of defining a variable in JavaScript are:

- **Var** – The JavaScript variables statement is used to declare a variable and, optionally, we can initialize the value of that variable. Example: var a =10; Variable declarations are processed before the execution of the code.
- **Const** – The idea of const functions is not allow them to modify the object on which they are called. When a function is declared as const, it can be called on any type of object.

- **Let** – It is a signal that the variable may be reassigned, such as a counter in a loop, or a value swap in an algorithm. It also signals that the variable will be used only in the block it's defined in.

Q26. What is a Typed language?

Typed Language is in which the values are associated with **values** and not with **variables**. It is of two types:

- **Dynamically:** in this, the variable can hold multiple types; like in JS a variable can take number, chars.
- **Statically:** in this, the variable can hold only one type, like in Java a variable declared of string can take only set of characters and nothing else.
    -
        -
        -

Q27. What is the difference between Local storage & Session storage?

Local Storage – The data is not sent back to the server for every HTTP request (HTML, images, JavaScript, CSS, etc) – reducing the amount of traffic between client and server. It will stay until it is manually cleared through settings or program.

Session Storage – It is like local storage; the only difference is while data stored in local storage has no expiration time, data stored in session storage gets cleared when the page session ends. Session Storage will leave when the browser is closed.

In case you are facing any challenges with these JavaScript Interview Questions, please comment on your problems in the section below.

Q28. What is the difference between the operators '== '& '=== '?

The main difference between "==" and "===" operator is that formerly compares variable by making **type correction** e.g. if you compare a number with a string with numeric literal, == allows that, but === doesn't allow that, because it not only checks the value but also type of two variable, if two variables are not of the same type "===" return false, while "==" return true.

Q29. What is the difference between null & undefined?

Undefined means a variable has been **declared** but has not yet been **assigned** a value. On the other hand, null is an assignment value. It can be assigned to a variable as a representation of no value. Also, undefined and null are two distinct types: undefined is a type itself (undefined) while null is an object.

Q30. What is the difference between undeclared & undefined?

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.

Q31. Name some of the JavaScript Frameworks

A JavaScript framework is an application framework written in JavaScript. It differs from a JavaScript library in its control flow. There are many JavaScript Frameworks available but some of the most used frameworks are:

- Angular
- React
- Vue

Q32. What is the difference between window & document in JavaScript?

Window	Document
JavaScript window is a global object which holds variables, functions, history, location.	The document also comes under the window and can be considered as the property of the window.

Q33. What is the difference between innerHTML & innerText?

innerHTML – It will process an HTML tag if found in a string

innerText – It will not process an HTML tag if found in a string

Q34. What is an event bubbling in JavaScript?

Event bubbling is a way of **event propagation** in the HTML DOM API, when an event occurs in an element inside another element, and both elements have registered a handle for that event. With bubbling, the event is first captured and handled by the **innermost** element and then propagated to outer elements. The execution starts from that event and goes to its parent element. Then the execution passes to its parent element and so on till the body element.

Q35. What is NaN in JavaScript?

NaN is a short form of **Not a Number.** Since NaN always compares unequal to any number, including NaN, it is usually used to indicate an error condition for a function that should return a valid number. When a string or something else is being **converted** into a **number** and that cannot be done, then we get to see NaN.

In case you are facing any challenges with these JavaScript Interview Questions, please comment on your problems in the section below.

Q36. How do JavaScript primitive/object types passed in functions?

One of the differences between the two is that Primitive Data Types are passed By Value and Objects are passed By Reference.

- **By Value** means creating a COPY of the original. Picture it like twins: they are born exactly the same, but the first twin doesn't lose a leg when the second twin loses his in the war.
- **By Reference** means creating an ALIAS to the original. When your Mom calls you "Pumpkin Pie" although your name is Margaret, this doesn't suddenly give birth to a clone of yourself: you are still one, but you can be called by these two very different names.

Q37. How can you convert the string of any base to integer in JavaScript?

The **parseInt()** function is used to convert numbers between different bases. It takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

For example-

```
parseInt("4F", 16)
```

Q38. What would be the result of 2+5+"3"?

Since 2 and 5 are integers, they will be added numerically. And since 3 is a string, its concatenation will be done. So the result would be 73. The " " makes all the difference here and represents 3 as a string and not a number.

Q39. What are Exports & Imports?

Imports and exports help us to write modular JavaScript code. Using Imports and exports we can split our code into multiple files. For example-

```
//------ lib.js ------</span>
export const sqrt = Math.sqrt;</span>
export function square(x) {</span>
return x * x;</span>
}
export function diag(x, y) {
return sqrt(square(x) + square(y));
}

//------ main.js ------</span>
 { square, diag } from 'lib';
console.log(square(5)); // 25
console.log(diag(4, 3)); // 5
```

Advanced Level JavaScript Interview Questions and Answers for Experienced Professionals

Q40. 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 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.

Q41. What is a prompt box in JavaScript?

A prompt box is a box which allows the user to enter input by providing a **text box**. The prompt() method displays a dialog box that prompts the visitor for input. A prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either “OK” or “Cancel” to proceed after entering an input value.

Q42. What will be the output of the code below?

```
var Y = 1;
if (function F(){})
{
y += Typeof F;</span>
}
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.

Q43. What is the difference between Call & Apply?

The **call()** method calls a function with a given this value and arguments provided individually.

Syntax-

```
fun.call(thisArg[, arg1[, arg2[, ...]]])
```

The **apply()** method calls a function with a given this value, and arguments provided as an array.

Syntax-

```
fun.apply(thisArg, [argsArray])
```

Q44. How to empty an Array in JavaScript?

There are a number of methods you can use to **empty** an **array**:

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 arrayList anywhere 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.

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.

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.

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.

Q45. 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.

Q46. What will be the output of the following code?

```
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 is undefined.

Q47. What will be the output of the following code?

```
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. However, we can delete the company property directly from the Employee object using delete Employee.company.

Q48. What will be the output of the code below?

```
//nfe (named function expression)
var Foo = Function Bar()
{
return 7;
};
typeof Bar();
```

The output would be Reference Error. A function definition can have only one reference variable as its function name.

Q49. What is the reason for wrapping the entire content of a JavaScript source file in a function book?

This is an increasingly common practice, employed by many popular JavaScript libraries. This technique creates a closure around the entire contents of the file which, perhaps most importantly, creates a private namespace and thereby helps avoid potential name clashes between different JavaScript modules and libraries. Another feature of this technique is to allow for an easy alias for a global variable. This is often used in jQuery plugins.

Q50. What are escape characters in JavaScript?

JavaScript escape characters enable you to write special characters without breaking your application. 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.

For example-

```
document.write "I am a "good" boy"
document.write "I am a "good" boy"
```