JavaScript is used to build complex, desktop-like web applications. One of the famous languages that help in building server-side software. It has gained huge popularity after the launch of Node.js It is a versatile and dynamic language. Let's understand the details of this language and have an interesting Q&A on it.

Q1. What is the difference between Java and JavaScript?

Java is an object-oriented programming language; JavaScript is an object-based scripting language. Java code can work on any platform including the web browser. However, JavaScript is written just for web browsers. Among other things, Java demands more memory. It is strongly typed and asks all variables to have a declared type. In contrast to JavaScript which is lightweight and is weakly typed.

Q2. What are the data types supported by JavaScript?

The data types supported by JavaScript are-

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

Q.3 Is JS a language conscious of case?

Yes, JavaScript is a case-sensitive language. One needs to make sure that- language keywords, function names, variables, and any other markers need to be consistent in code.

Q.4 Can you assign an anonymous function to a variable? Is it possible to pass it as an argument to another function?

Yes! An anonymous function can be assigned to a variable. Yes, it is a possibility to pass it as an argument to another function.

Q.5 What are the scopes of a variable in JavaScript?

The scope of a variable refers to the region of your program, where a variable is visible and could be accessible. JavaScript variables have two scopes.

· Global Variables – A global variable has a large scope meaning it is seen everywhere in your JavaScript code.

· Local Variables – A local variable will be visible only where it is defined inside a function. Function parameters are always local to that function.

Q.6 What is the purpose of 'This' operator in JavaScript?

The JavaScript keyword 'this' refers to the object it belongs to. It will have a different value depending on where it will be used. In a method, 'this' caters to the owner object, and in a function, 'this' mentions the global object.

Q.7 What is Callback?

A callback is a plain JavaScript function that is passed to some method as an argument or option. The name is derived as the function gets implemented after another function has

finished its implementation. In JS, functions are first-class objects. Due to this, functions can seek functions as arguments as well. It can also be returned by other functions.

Q.8 What are the variable naming conventions in JavaScript?

When naming variables in JavaScript, one needs to keep certain rules in mind-

- One cannot use JavaScript reserved keywords as a variable name. For example, debugger or default variable names are invalid.
- JavaScript variable names should not begin with a numeral (0-9). They ought, to begin with, the underscore character or a letter. For example, 789name is an invalid variable name but _789name or name789 is a valid one.
- JavaScript variable names are case-sensitive. For example, Meet and meet are two different variables.

Q.9 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.

Q.10 How to create a cookie using JavaScript?

The easiest way to create a cookie is to designate a string value to the document.cookie object, which looks like this-

Syntax :

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

Q.11 How to read a cookie using JavaScript?

Reading a cookie is like writing one. The reason is the value of the document.cookie object is the cookie. Hence, one can use this string whenever one wants 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 the strings' split() function to break the string into keys and values.

Q.12 How to delete a cookie using JavaScript?

If you want to delete a cookie so that upcoming attempts to read the cookie in JS return nothing, one needs to set the expiration date to a time in the past. One ought to 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.

Q.13 What is hoisting?

Hoisting is the default behavior of JS where all the variable and function declarations are moved on the highest echelon, irrespective of where the variables and functions are declared. The scope can be both local and global.

```
Example:
hoistedVariable = 3;
console.log(hoistedVariable);
// outputs 3 even when the variable is declared after it is initialized
var hoistedVariable;
// Variable initializations are not hoisted, only variable declarations are hoisted
Example :
var x;
console.log(x);
// Outputs "undefined" since the initialization of "x" is not hoisted
x = 23;
```

Q.14 What is a closure in JavaScript?

A closure is an inner function that admits to the variables that belong to the outer (enclosing) function's scope chain. There are three ways in which the closure can access to variables; specifically:

- variables in its own scope,
- variables in the enclosing function's scope,
- global variables.

Q.15 What is event delegation?

DOM event delegation is used for responding to user events via a single parent rather than each child node. With it, you can bind an event handler to a common parent element that will handle any event happening to one of its children.

Q.16 What are the differences in capturing and bubbling phases?

Event bubbling and capturing are two ways via which events can be handled in the HTML DOM API. When an event occurs in an element and its parent also registers a handle for the same event, the event propagation model will decide which element should receive the event first.

By default, addEventListener uses bubbling- here the event is first captured in the innermost element and propagates to outer elements.

While in the capturing phase, the opposite happens; the event is captured in the outermost element first and propagates to inner elements.

Q.17 What is the difference between Attributes and Property?

Attributes- It presents to you more info of an element like type, value, id etc.

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

Q.18 What are the different ways in which HTML elements can be accessed in JavaScript code?

These are the list of ways, through which HTML elements 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.

Q19. In how many ways a JS code can be involved in an HTML file?

There are 3 different ways, when the following can take place:

- Inline
- Internal
- External

An inline function is a JavaScript function, designated to a variable created at runtime. One can tell the difference between Inline Functions and Anonymous as an inline function is designated to a variable. It can be easily used again.
When one needs JavaScript for a function, one can either have the script embedded into the page one is working on, or have it placed in a separate file that one can use when required. This is the difference between an internal script and an external script.

Q20. What are the ways to define a variable in JavaScript?

The 3 main techniques of defining a variable in JavaScript are:

- Var – We use a JavaScript variables statement mainly to declare a variable. As an option, we can also reconfigure that variable's value. For instance: var a = 10; the Variable declarations are prepared much before that code's execution.
- Const – The prime idea of const functions is not to permit them to change the object on which they have been called. When a specific function is declared as constant, then it can be called on any sort of object.
- Let – It is considered as a sign that the variable may be reassigned, like a counter in a loop, or a swap of value in an algorithm. Besides, it also shows that the variable will be used only in the block it is defined in.

Q21. What is a Typed language?

Typed Language is that language where the values are assigned with values and not with variables. So, there are two kinds of typed language:

- Dynamically: the variable can hold several types; just like in JavaScript, a variable can consider chars or numbers.
- Statically: The variable can hold a single type. For example- in Java, a variable which is declared of string can easily take only a few set of characters and nothing else.

Q22.What type of Typed language is JS?

JavaScript is a dynamically typed language. Therefore, here, the variable type is properly checked while run-time. In contrast to statically typed language, where the specific variable type is properly checked while the compilation-time.

Q23. What is the difference between Local storage & Session storage?

Local Storage – Here, the data is not sent to the server reversely for each HTTP request (images, CSS, HTML, JavaScript, etc) – lessening the amount of traffic between the server and client. However, it stays until it is manually cleared via program or some settings.

Session Storage – It resembles the local storage, but the difference is the data which is stored in local storage has no expiration time and the data stored in session storage immediately gets cleared when the session of the page terminates. Moreover, the session storage will leave when the browser is off.

Q24. What is the difference between the operators '==' & '==='?

The prime difference between "==" and "===" operator is that "==" compares the variable by making a type correction. For instance, if you compare a number with a string with numeric literal, == permits that, however === doesn't permit that, because it not only checks the value yet also the kind of two variable if two variables are not of the same type "===" return false, while "==" return true.

Q25. What is the difference between null & undefined?

Undefined refers to a variable that has been declared yet has not been assigned a specific value. Besides, null refers to an assignment value and it can be assigned to a variable as a demonstration of no value. Moreover, undefined and null are the two distinct kinds: null is an object while undefined is itself a type.

Q26. What is the difference between undeclared & undefined?

Undeclared variables are not visible in a program and are not declared too. If a program puts an effort to read the value of an undeclared variable, then a runtime error is experienced. Certainly, undefined variables are declared in the program yet have not been provided any value. So, if the program wishes to read the value of an undefined variable, an undefined value is instantly returned.

Q.27. What is currying?

Currying is an advanced technique to transform a function of arguments n, to n functions of one or less arguments.

```
Example of a curried function:
function add (a) {
  return function(b){
    return a + b;
  }
}
add(3)(4)

For Example, if we have a function f(a,b) , then the function after currying, will be transformed to f(a)(b). With a currying technique, we do not modify the functions' functionality, we just modify the way it is called on.
Let's look at currying in action:
function multiply(a,b){
  return a*b;
}
function currying(fn){
  return function(a){
    return function(b){
      return fn(a,b);
    }
  }
}
var curriedMultiply = currying(multiply);multiply(4, 3); // Returns 12
curriedMultiply(4)(3); // Also returns 12

As mentioned in the above, we have altered the function multiply(a,b) to a function curriedMultiply , which demands one parameter at a time.
```

Q.28- What are the first-class functions?

First class function are functions that you can:

- Pass as arguments to other functions.
- Store them in a variable.
- Return as values from other functions.

Q.29- What is the difference between lexical and dynamic scoping?

Lexical scoping refers to the available variables in the location of a function's definition.

Dynamic scoping refers to the available variables in the location of a function's invocation.

Q.30- What are the different types of errors in JavaScript?

There are three types of errors:

- Load time errors: These are the errors arising when loading a web page, like improper syntax errors. Certainly, they generate the errors in a dynamic form.
- Runtime errors: These errors arise due to certain misuse of the command inside the HTML language.
- Logical Errors: These are the errors that happen due to poor logic executed on a function with a different operation.

Q.31 What is the use of the Push method in JavaScript?

We use the push method to add or affix one or more elements to an array end. However, when we use the push method, one can attach several elements by passing various arguments.

Q.32 What is the unshift method in JavaScript?

The Unshift method is similar to the Push method, which mainly works at the commencement of the array. We use this method to add one or more elements at the starting

of the array.

Q.33 What is event bubbling?

JavaScript permits DOM elements to be fixed inside each other. In this case, if one clicks the handler of the child, then the handler of the parent will also operate as if it were clicked too.

Q.34 How are JavaScript and ECMA Script related?

ECMA Script represents guidelines and rules, while JavaScript is a scripting language used mainly for web development.

Q.35 What is DOM?

DOM stands for Document Object Model and it is a programming interface for XML and HTML documents.

When the browser attempts to render a HTML document, it instantly builds an object based on the HTML document known as DOM. With DOM, we can easily manipulate or alter several elements that are inside the HTML document.

Q.36 What is a Temporal Dead Zone?

Temporal Dead Zone is a behavior seen with variables that are declared using const and let keywords. This behavior occurs, when one tries to access a variable before it is initialized. Examples of temporal dead zone:

```
x = 23; // Gives reference error
let x;

function anotherRandomFunc(){
  message = "Hello"; // Throws a reference error
  let message;
}
anotherRandomFunc();
```

In the above mentioned code, we are attempting to access variables that have not been declared yet in the functional scope and the global scope. This is called the Temporal Dead Zone.

Q.37 What is the major difference between Document and Window in JavaScript?

Document: The document lies under the windows object and can also be contemplated as its property.

Window : Window in JavaScript represents a global object that holds the structure like functions, location, variables, history, etc.

Q.38 What are web workers in JS?

Web Workers represents running scripts in background threads for the web content. The worker thread can easily execute tasks without intervening with the user interface.

Besides, they can use XMLHttpRequest to execute I/O or fetch. Once created, by posting messages to an event handler based on that code, a worker can simply send messages to the JavaScript code that created it and vice versa.

Q.39 How web workers helps in simulating multi threading in JS?

Web Workers helps you create background threads unconnected from the prime execution thread, where the logic of the user interface is usually performed. The basic benefit of this unconnectedness of workload is that inside an isolated thread and without intervening or affecting the prime thread's usability and responsiveness, one can easily execute costly operations.

Example :
Step 1: Create a project folder and add index.html in the root of it.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Club JavaScripters</title>
  </head>
  <body>
    <button>Can you click me?</button>
    <h2>Let's do the party. 💃</h2>
    <ul id="club"></ul>
    <script src="./main.js"></script>
  </body>
</html>
```

We just created a simple HTML page with a link to our javascript file which we will create in a momen
t.
Step 2: So, now we will add our JavaScript code in two individual files, worker.js and main.js in the
same folder.
main.js

```
const persons = [
  {
    name: "Hriday",
    isMember: true,
  },
  {
    name: "Hridayesh",
    isMember: false,
  },
  {
    name: "Bob",
    isMember: true,
  },
  {
    name: "Daisy",
    isMember: true,
  },
]
// Our club container
const club = document.getElementById("club")
let worker
/**
 * Function entry permits entry to those people who are coming to our club
 */
function entry(persons) {
  persons.forEach(person => {
    const { isMember, name } = person
    const listItem = document.createElement("li")
    listItem.textContent = name
    // if a person is not registered till now, register them first
    if (!isMember) {
      // create a new worker thread
      worker = new Worker("worker.js")
      // pass data to worker thread
      worker.postMessage(name)
      // listen to any data which is passed from the worker thread
      worker.addEventListener("message", event => {
        if (event.data) {
          club.appendChild(listItem)
        }
      })
    } else {
      // let them in if they are registered
      club.appendChild(listItem)
    }
  })
}
entry(persons)
```

worker.js

```
// addEventListener is directly accessible in the worker file
addEventListener("message", event => {
  // extract person passed from the main thread from event object
  let person = event.data
  registerMember(person)
})
function registerMember(member) {
```

```
  // generating membership card takes some time
  let i = 0
  while (i < Math.pow(10, 10)) {
    i++
  }
  // send the result back to the main thread
  postMessage(true)
  close()
}
Explaination :
a. Initially, we search for each person if they are a member or not. And if they are, then we permit
them in.
b. If a person is not a member of the club till now, then we develop a worker thread for it with a fr
esh Worker('path to worker code') and then pass that person to the worker thread with worker.postMess
age() and later adding that specific person as a member after registering them.
c. To imitate the entire registration process we execute some processing with the help of a long whil
e loop. In reality, this could be similar to some cryptographic code, image processing or any other t
hing that is CPU intensive and might block the main thread and make the page unresponsive.
d. Once you have registered, the worker thread let the main thread know about your registration by se
nding the data using postMessage and the main thread listens for it using on message handler and add
the person in the club i.e add to DOM.
e. Once we finally get the result from the worker thread, we instantly close it using close().
```

Q.40 How is this keyword handled in JS's ES5 regular syntax functions vs arrow functions?

Certainly, the behavior of "this" which is inside of an arrow function varies considerably from the general function's "this" behavior. However, the arrow function does not show its performance context. No matter where or how it is being performed, "this" value which is inside of an arrow function always remains the same "this" value from the outer function. In other words, the arrow function resolves "this" lexically.

```
In the following example, myMethod() is an outer function of callback() arrow function:
const myObject = {
  myMethod(items) {
    console.log(this); // logs myObject
    const callback = () => {
      console.log(this); // logs myObject
    };
    items.forEach(callback);
  }
};
myObject.myMethod([1, 2, 3]);

"this" value that is inside the arrow function callback() equals to "this" of the outer function myMe
thod().
"this" resolved verbally is one of the fine characteristics of arrow functions. When using callbacks
inside methods you have to be sure that the arrow function doesn't notify its own "this": no more con
st self = this or callback.bind(this) workarounds.
Contrary to a regular function, the indirect invocation of an arrow function using myArrowFunc.call(t
hisVal) or myArrowFunc.apply(thisVal) doesn't alter the value of "this": the context value is always
resolved lexically.
```

Q.41 What is IIFE (Immediately Invoked Function Expression)?

An IIFE (Immediately Invoked Function Expression) is a javascript function that runs just after it's defined. The '( )' at the end is responsible for calling the function.

```
(function () {
  statements
})();
```

Q.42 How is Exception handling can be done in JavaScript?

Exception handling can be done by using try catch block, any block of code which can throw an exception should be wrapped in try block. The catch block is responsible for handling the exception caught.

```
try{
   apiCall();
```

```
}catch(error){
  //error occurred in apiCall
}
```

Q.43 What and how many types of client side storages are there?

Data upto certain limit can be stored in browser instead of server.There are 3 types of client side storage present in browser 1.Cookie. 2.Local Storage. 3.Session Storage.

Q.44 How to attach event listeners in JavaScript?

You can attach an event on target by using the “addEventListener” function which accepts event type and a callback function as a parameter. Whenever this event triggers, the callback function gets executed.

```
e.g target.addEventListener( 'click' , function(){
  //event triggered
});
```

Q.45 What is implicit type coercion?

Automatic conversion of one data type to another is known as implicit type coercion.

```
 example
console.log("3" + 3). //prints 33
The integer value of 3 coerced to "3" internally, resulting 3+"3" into "33".
```

Q.46 How these array functions works (map, filter and reduce)?

Map: Map takes a callback and runs it against every element on the array. However, it is unique as it generates a new array based on your existing array. Example:

```
var sample = [1, 2, 3]
let mapped = sample.map(elem => elem * 10)
console.log(mapped);
/* output */
[10, 20, 30]
```

Filter: Filter lets you provide a callback for every element and check its return value. If the value is true element remains in the resulting array but if the return value is false the element will be removed for the resulting array.
Example:

```
var sample = [1, 2, 3]
var result = sample.filter(elem => elem !== 2)
/* output */
[1, 3]
```

Reduce: Reduce method of the array object is used to reduce the array to one single value. For example if you have to add all the elements of an array:

```
var sample = [1, 2, 3]
var sum = sample.reduce((sum, elem) => sum + elem)
console.log(sum)
/* output */
6
```

Q.47 How to check whether an object is empty or not?

We can create an empty object by using a constructor, so checking the key length of an object doesn’t work. We have to check for its constructor as well.

```
Object.keys(value).length === 0
&& value.constructor === Object;
```

Q.48 How to implement local storage in our code?

LocalStorage is a property that allows JavaScript sites and apps to save key-value pairs in a web browser with no expiration date. This means the data stored in the browser will persist even after the browser window is closed. It provides different methods to work with:-

```
setItem(): Add key and value to localStorage
getItem(): Retreive fom localStorage
removeItem(): Remove an item by key from localStorage
clear(): Clear all localStorage
key(): Passed a number to retrieve the key of a localStorage
```

Q.49 How will you target more than one html element in JS?

The document.querySelectorAll method will return you the nodeList of all element with same identifier.

```
const allParagraph = document.querySelectorAll("p");
const allElementWithFunClass= document.querySelectorAll(".fun");
```

Q.50 **What is QuickSort Algorithm in JavaScript?**

One of the most used and popular algorithms in the programming world. It is important to note there is a difference between sort() and this Quick Sort algorithm.

Quick Sort algorithm follows Divide and Conquer approach. Now, it divides elements into smaller parts. Based on those conditions, the algorithm performs operations on those divided smaller parts. It works well for large datasets. Let's see how it works in few steps

1. First, select an element that is to be called the **pivot** element.
2. Compare all array elements with the selected pivot element. Arrange them so that elements less than the pivot element are on the left and greater ones are on right.
3. Finally, perform the same operations on the left and right side elements to the pivot element.

To Conclude...

Candidates are required to clear the interview in order to pursue a career in JavaScript development. Numerous JavaScript interview questions and answers are posed to them.

We have compiled the list of JavaScript interview questions and answers that are most likely to be asked during the interview. Based on their expertise and other criteria, candidates may be asked anything from simple JavaScript interview questions to advanced JS interview questions.

All of the JavaScript questions for freshers and JavaScript interview questions for professional-level applicants are included in the list above. **Codersera** guide to JS interview questions will help you ace the interview and land your ideal JavaScript Programming job.

1. **Where is JavaScript used?**
   JavaScript is **commonly used for creating web pages**. It allows us to add dynamic behavior to the webpage and add special effects to the webpage. On websites, it is mainly used for validation purposes. JavaScript helps us to execute complex actions and also enables the interaction of websites with visitors.
2. **What is '$' in JavaScript?**
   The $ represents **the jQuery Function**, and is actually a shorthand alias for jQuery . (Unlike in most languages, the $ symbol is not reserved, and may be used as a variable name.) It is typically used as a selector (i.e. a function that returns a set of elements found in the DOM).
3. **Is JavaScript good for coding interviews**
   There are some languages that **are very safe**. Your interviewer should understand Java, C, and Python, so any of those languages are totally reasonable. Other languages,

particularly more readable ones are fine too. For example, Javascript and Ruby.

4. **What does 3 dots mean in JavaScript?**
   Spread Syntax

   (three dots in JavaScript) is called **the Spread Syntax or Spread Operator**. This allows an iterable such as an array expression or string to be expanded or an object expression to be expanded wherever placed. This is not specific to React. It is a JavaScript operator.

PREVIOUS
What is Hybrid low-code and why it'...

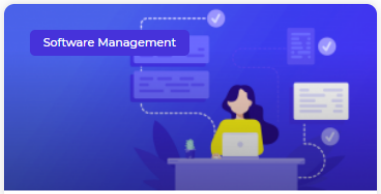
NEXT
5 APIs You Need On Your Website

More To Explore

Top 15 Ruby Project Ideas To Try Right Now

Ruby is a multifunctional programming language that is dynamic, analytical, object-oriented, and multipurpose. In the 1990s, the founder, "Matz" Matsumoto, modeled and developed this language

Lucas White · December 7, 2021



Software Development Outsourcing: A Complete Guide

Outsourcing has become a popular business practice in recent years. And for good reason! Outsourcing software development can help businesses cut costs, increase efficiency, and

Kela Casey · December 4, 2021