JavaScript Interview Questions with Answers

→ **For more Step by Step and Interview Q&A Course visit -** www.questpond.com

→ **For offline training visit -** https://www.stepbystepschools.net/

→ **Stay tuned with Questpond for more updates -** t.me/questpond

Contents

Why do we call JavaScript as dynamic language?

JavaScript is a dynamic language means data types of the variables can change during the runtime.

How does JavaScript determine data types?

JavaScript determines data types depending on the value assigned. So, if you assign 10 value it becomes a number, if you assign “Hello” value it makes it a string.

What is “typeof” functions?

How to check data type in JavaScript?

To check data type, we will can use the “typeof” function in JavaScript.

```
var x=100;
var watisdatatype = typeof(x);
```

What are the different datatypes in JavaScript?

There are total 8 data types in JavaScript and these 8 data types are divided in to 2 categories, Primitive and Objects

Below are the Primitive data types plus objects.

String type
Null type
Number type
Undefined type
Boolean type
BigInt type
Symbol type

SNNUB + Objects

Explain Undefined Data types ?

Undefined means the variable has been declared but no value is assigned to it. In the below code you can see “x” variable is declared but not assigned any value. In this scenario “x” will have value “undefined”.

```
        var x;
```

What is Null?

Null indicates intentional absence of data. Null indicates its not ZERO , Its not empty its just absence of data.

Differentiate between Null and Undefined?

Undefined	Null
Variable has been created but value is not assigned.	We assign value NULL , it indicates absence of data.

Explain Hoisting?

It's a mechanism where variables and function declaration to the top of the scope.

Are JavaScript initialization hoisted?

No. It has value undefined.

What are global variables?

Global variables are accessible through out the webpage or the document.

What are the issues with Global variables?

Global variables makes debugging hard as its accessible everywhere.

What happens when you declare variable without VAR ?

It makes the variable global.

What is Use Strict ?

How to force developers to use Var keyword?

"Use Strict" says that variable should be defined using "var" keyword , if not done exception should be thrown. "Use Strict" strictly checks if the variable is defined using "var" keyword.

How can we handle Global Variables?

How can we avoid Global variables?

Its difficult to avoid global variables. But we can organize it properly by doing two things: -

- Put global variables in a proper Namespace.
- Module pattern using Closures and IIFE.

What are Closures?

Closures are functions inside function and it makes a normal function stateful. Below is a simple closure code.

```
function ClosureFunction(){
            var x;
            function Increment(){
                x++;
            }
            function GetXValue(){
                return x;
            }
            function Init(){
                x=0;
            }
            Init();
            return {
                Increment,
                GetXValue
            }
         }
```

Why do we need Closures?

Closures have two big advantages: -

- It helps to create self-contained modules, if we have self-contained modules then their state is also self-contained which leads to less global variables and thus better code.
- It helps to implement encapsulation and abstraction.

Explain IIFE?

IIFE Stands for Immediately Invoked Function Expression. It's an anonymous function which gets immediately invoked.

```
(function(){
   // Gets invoked immediately
})();
```

What is the use of IIFE ?

What is name collision in global scope ?

Because IIFE does not have name we can never get name collision. Name collision happens when same name function names and variable names are declared.

IIFE vs Normal Function

A normal function has a name while IIFE does not have name. So with a normal function you can have a name collision but with IIFE you will not have name , you will not name collision.

What are design patterns?

Design patterns are time tested architecture solutions. So, to create a single instance we can use singleton pattern, if the object creational process is complex, we can use factory pattern and so.

Which is the most used design pattern in JavaScript?

Module pattern / Module Revealing pattern is the most used design pattern

What is module Pattern and revealing module pattern?

Module pattern or revealing module pattern has 2 big advantages: -

- Self-contained independent components.
- Provides Encapsulation and Abstraction.

How many ways are there to create JavaScript objects ?

There are four ways of creating JavaScript objects:-

- By using Literal.
- By using Object.create()
- Constructor way of creating objects.
- By using ES6 Classes.

How can we do inheritance in JavaScript?

JavaScript uses object inheritance or prototypical inheritance. Inheritance is done using Prototype object.

What is prototype in JavaScript?

Every JavaScript object has a Prototype object. It's an inbuilt object provided by JavaScript. When you set the prototype object, the prototype object becomes the parent object of the current object.

Explain Prototype chaining?

Prototype chaining is a process where the property/methods are first checked in the current object , if not found then it checks in the prototype object , if does not find in that it try checking the prototypes prototype object , until he get the prototype object null.

What is Let Keyword?

"Let" helps to create immediate block level local scope.

Are Let variables hoisted?

Yes, they are hoisted but not initialized. So, if you try to access the variables you will get an uninitialized error.

Explain Temporal Dead Zone?

TDZ it's a period or it's a state of a variable where variables are named in memory but they are not initialized with any value. During this time if you try access the variable you will get initialized error.

Let vs Var

	Var	**Let**
Scope	Scoped to the Immediate function body.	Scoped to the immediate enclosing block.
Initialized value	Value initialized with Undefined	Value initialized with nothing.

Tricky Question around Concatenation

What will be the output of the console codes below. Output will be

1010
20
24

```
var d = "10";
var d1 = "10";
console.log(d+d1);
var i = 10;
var i1 = 10;
console.log(i+i1);
console.log(1+1+"4");
```

Explain class keyword?

So with Class Keyword can we say Javascript is OO ?

No it Is not.

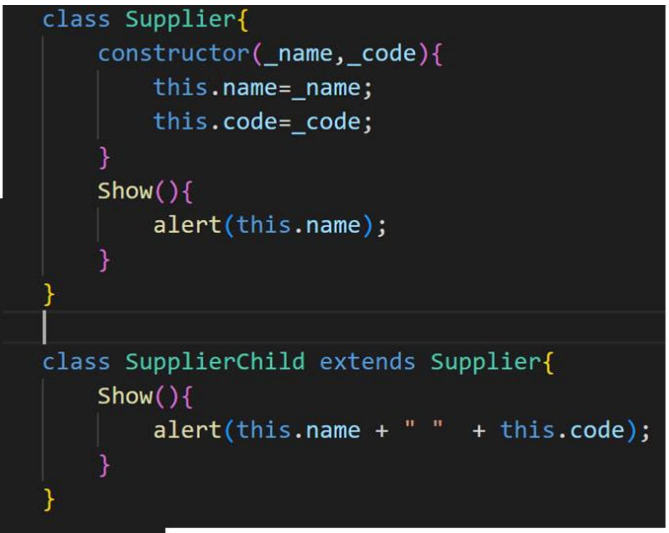
Class keyword which are in introduced in ES6 are syntactic sugar over constructor functions and prototype-based inheritance. In other words, the Class keyword internally is a constructor function.

When this question comes up many developers try to answer this question from the perspective of OOP like class is a template, it's a blue-print and to use class we create an object or an instance.

This answer is perfectly right from the point of object-oriented programming perspective. And if this is an accepted answer by the interviewer its perfectly ok.

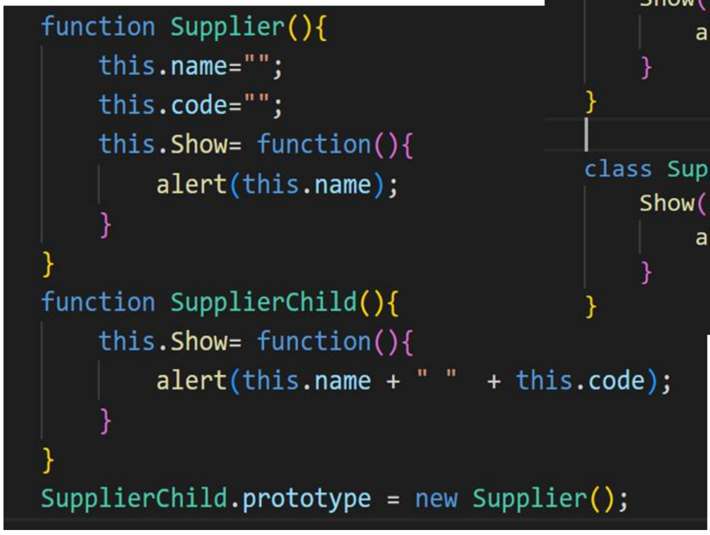
But surely they will cross question that so with this class key word can we say Javascript is a OOP language and the answer is NO. Javascript started as a functional programming language and that's how it will be ahead.

Now ,Majority of the developers who comes from java , C# have good knowledge of OOP concepts and to get in to the mindset of functional programming and added to that prototypical based inheritance makes things more tough and decreases developers productivity.



```
class Supplier{
    constructor(_name,_code){
        this.name=_name;
        this.code=_code;
    }
    Show(){
        alert(this.name);
    }
}

class SupplierChild extends Supplier{
    Show(){
        alert(this.name + " "  + this.code);
    }
}
```



```
function Supplier(){
    this.name="";
    this.code="";
    this.Show= function(){
        alert(this.name);
    }
}
function SupplierChild(){
    this.Show= function(){
        alert(this.name + " "  + this.code);
    }
}
SupplierChild.prototype = new Supplier();
```

What is difference between Class and Normal function?

Class Intent: - Represents an Entity.
Function Intent: - They just get executed; they are stateless.

Class keyword guarantees that you can call it only by using new keyword which avoids global pollution of "this" keyword. While for functions you can call it with different ways and there can be global spilling in some instances.

Classes does not work with legacy browser, so if you want to have backward compatibility with old browsers then you need to use functions.

Explain Arrow function?

Arrow function is a shorter way of writing a function.

When will you use an Arrow function?

Now many developers would say that arrow function are easy to read or write. I feel that answer is a myth. So answering that arrow function is easy to read , understand can lead more cross questions and can probably put you in a uncomfortable position.

Arrow function are used to create short hand functions which you create and fire it on predicate inputs of functions like Sort, filter, Map function etc. Its like use and throw kind of logic / expressions.

Arrow functions vs Normal functions?

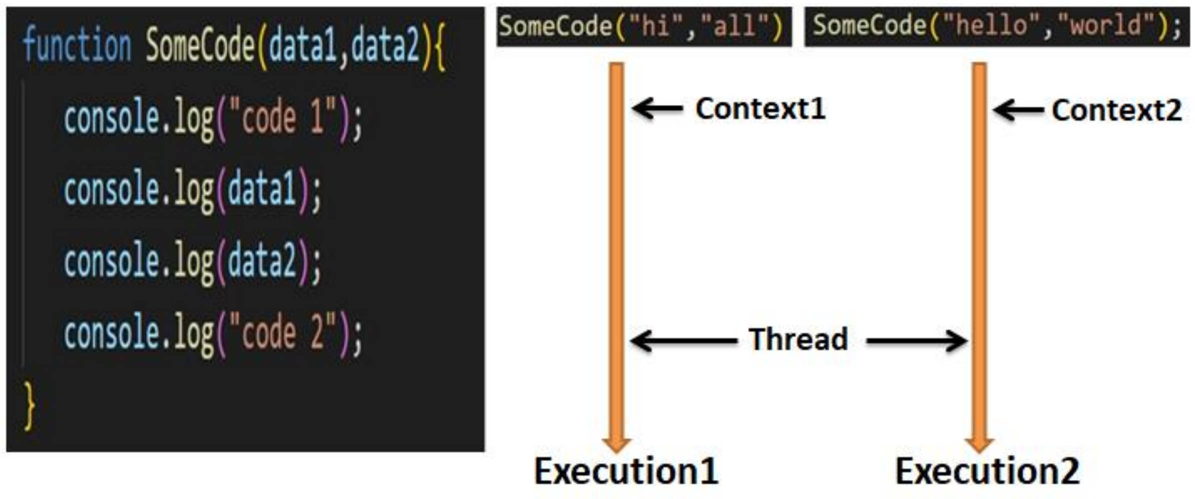
	Normal function	**Arrow function**
Intent	Reusability. Create a function and call the reusable code where needed.	Fire and forget. Create short hand functions and fire it over a predicate. Create a short hand function fire it on Sort, Filter, Map and so on.
Scope (this)	Normal function can have their own "this". "this" can change depending on how a function is called.	In Arrow functions they do not have their own "this". "this" in arrow function is borrowed lexically.

Does Arrow function have its own this ?

No.

A thread is an execution context which CPU (processor) can execute. Other way to answer is Thread is a sequence of program instruction which can be run independently by the processor.

Thread = Execution + Context (Data)



```
function SomeCode(data1,data2){
    console.log("code 1");
    console.log(data1);
    console.log(data2);
    console.log("code 2");
}
```

```
SomeCode("hi","all")
SomeCode("hello","world");
```

Explain Synchronous execution?

Synchronous execution is an execution when code is executed sequentially one after another. One statement has to wait for other statement to finish.

What is a call Stack in JavaScript ?

Call stack keeps track of functions to be executed. It follows LIFO way of execution.

What is a blocking call?

Blocking call is a call which further blocks code execution.

How to avoid blocking calls ?

Explain Asynchronous execution ?

Async code is that code which runs consecutively and avoids blocking the main execution.

Asych means not code can execute concurrently rather than sequentially. So if there is a blocking code that code can run in the back ground and code next after it can run paralley.

Async operations are needed when you have blocking code.

Synch vs Asynch ?

Synch works one after another.
In case of Asynch tasks execute simultaneously.

How can we do Asynch calls in Javascript ?

There are three ways of doing Async calls :-

- By making WebAPI Browser calls.
- Promises
- Worker threads.

Explain threads ?

Threads are code execution context which CPU can run independently.

Explain Multi-threading ?

Multi-threading means executing code parallelly.

Is JavaScript Multi-threaded ?

No , JavaScript is single threaded.

Then how does Settimeout run ?

“Settimeout” runs inside the browser context and not in the JavaScript call stack.

What is a WebAPI/Browser API ?

Web API stands for Application programming interface which helps to access browser functionalities like http,timer, bluetooth , geolocation and so on.

What is an Event loop and callback queue?

Callback queue is a queue where we have those functions which is pushed after an asynchronous operation is finished like promises, settimeout etc.

Eventloop is a loop which runs continuously and does two things:-

- Checks if Callstack is Empty.
- If Callstack is empty , it will take functions from callback queue and puts it in the call stack.

What is the output of the below code (Testing Eventloop and CallBack Queue) ?

```
<script>

        function DisplayMessage(){
            console.log("Message is displayed");
        }
        function fun1(){
            console.log("Fun1");
            setTimeout(() => DisplayMessage(),0)
            fun2();
        }
        function fun2(){
            console.log("Fun2");
        }
        fun1();

      </script>
```

The output will be :-

Fun1
Fun2
Message is Displayed.

CallBack Queue codes have less priority than Call stack code so “DisplayMessage” function will be executed after “Fun2”.