**##JavaScript interview questions**

**1) What is called Variable typing in JavaScript?**

Variable typing is used to assign a number to a variable and the same variable can be assigned to a string.

Example i =10; //number

i = ”JavaScript”; //string

**2) What is asynchronous programming, and why is it important in JavaScript?**

Synchronous programming means that, barring conditionals and function calls, code is executed sequentially from top-to-bottom, blocking on long-running tasks such as network requests and disk I/O.

Asynchronous programming means that the engine runs in an event loop. When a blocking operation is needed, the request is started, and the code keeps running without blocking for the result. When the response is ready, an interrupt is fired, which causes an event handler to be run, where the control flow continues. In this way, a single program thread can handle many concurrent operations.

User interfaces are asynchronous by nature, and spend most of their time waiting for user input to interrupt the event loop and trigger event handlers.

Node is asynchronous by default, meaning that the server works in much the same way, waiting in a loop for a network request, and accepting more incoming requests while the first one is being handled.

This is important in JavaScript, because it is a very natural fit for user interface code, and very beneficial to performance on the server.

**3)** [**What is Prototype in JavaScript?**](http://www.code-sample.com/2015/06/prototype-in-javascript.html)[**How to create prototype in JavaScript?**](http://www.code-sample.com/2015/06/prototype-in-javascript.html)

The prototype is a fundamental concept of JavaScript and its must to known JavaScript developers and all the [JavaScript objects](http://www.code-sample.com/2015/06/prototype-in-javascript.html) have an object and its property called prototype and its used to add and the custom functions and property.

**var employee = function () {**

**//This is a constructor function.**

**}**

**//Crate the instance of above constructor function and assign in a variable**

**var empInstance = new employee();**

**empInstance.deportment = "IT";**

**console.log(empInstance.deportment);//The output of above is IT.**

**The example with prototype as given below.**

**var employee = function () { //This is a constructor function.}**

**employee.prototype.deportment = "IT";//Now, for every instance employee will have a deportment.**

**//Crate the instance of above constructor function and assign in a variable**

**var empInstance = new employee();**

**empInstance.deportment = "HR";**

**console.log(empInstance.deportment)**

**4) What Is a Public, Private and Static Variable in JavaScript?**

**function myEmpConsepts() { // This myEmpConsepts is a constructor  function.**

**var empId = "00201"; //This is a private variable.**

**this.empName = "Anil Singh"; //This is a public variable.**

**this.getEmpSalary = function () {  //This is a public method**

**console.log("The getEmpSalary method is a public method")**

**}**

**}**

**//This is an instance method and its call at the only one time when the call is instantiate.**

**myEmpConsepts.prototype.empPublicDetail = function () {**

**console.log("I am calling public vaiable in the istance method :" + this.empName);**

**}**

**//This is a static variable and it’s shared by all instances.**

**myEmpConsepts.empStaticVaiable = "Department";**

**var instanciateToClass = new myEmpConsepts();**

**5) What Is Closure in JavaScript?**

While you create the JavaScript function within another function and the inner function freely access all the variable of outer function i.e.

**function ourterFun(i) {**

**var var1 = 3;**

**function innerFun(j) {**

**console.log(i + j + (++var1)); // It will return the 16.**

**}**

**innerFun(10);**

**}**

**ourterFun(2); // Pass an argument 2**

The output will get 16 because innerFun() function can access to the argument "i" & variable "var1" but both are define in the outerFun() function that is closure.

That means simply accessing variable outside of your scope creates a closure.

**6) "1"+2+3 = ? And 1+2+"3" = ? OutPut Will Be?**

As you know in JavaScript, when the “first character” is a “string” the remaining characters will be converted into a “single string” that means the output of “1”+2+3 will be 123.

If the “string” is at the end the initial characters will perform normal mathematical functions before converting into a “string”, that means the output of 1+2+“3” will be 33.

**7) How to empty an array in JavaScript?**

var userList =  ['anil','kumar','singh','kushinagar','up','india'];

Method 1: - userList =[];

Method 2: - userList.length=0;

Method 3: - userList.splice(0, userList.length);

**8) difference between call and apply in JavaScript**

[CALL](http://www.code-sample.com/2015/06/difference-between-call-and-apply.html)  -    
  
Call a function with the specified arguments. You can use call, if you know how many argument are going to pass to the functions.

[APPLY](http://www.code-sample.com/2015/06/difference-between-call-and-apply.html) -    
  
Call a function with argument provided as an array. You can use apply if you don't know how many argument are going to pass to the functions.

Both (call and apply) are using to call a functions.

Here is an advantage over apply and call. The .[call()](http://www.code-sample.com/2015/06/difference-between-call-and-apply.html) method is little bit faster than .[apply()](http://www.code-sample.com/2015/06/difference-between-call-and-apply.html) method.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8" />**

**<title>**[**The Difference Between Call and Apply in JavaScript**](http://www.code-sample.com/2015/06/difference-between-call-and-apply.html)**</title>**

**<link rel="stylesheet" href="style.css" />**

**<script>**

**var var1 = {my: "Obj1" };**

**var var2 = {my: "Obj2"};**

**function preview(par1, par2) {**

**alert(this.my + ' ' + par1 + ' ' + par2);**

**console.log(this.my, par1, par2);**

**}**

**//using call as given below**

**preview.call(var1, "First", "Second"); //output: var1 First Second**

**//using apply as given below**

**preview.apply(var2, ["First", "Second"]); //output: var2 First Second**

**//using basic as given below**

**preview("First", "Second");//output: undefined "First", "Second"**

**</script>**

**</head>**

**<body>**

**<div>**

**<h3>Refresh the page and see the alert result or go to console window and see the result.</h3>**

**</div>**

**</body>**

**</html>**

**9) What are the common Errors in JavaScript?**

he common errors in JavaScript programming are the following:

1)Spelling and typing errors.

2)Missing brackets or quotation marks.

3)Mismatching quotation marks.

4)Using single equal sign instead of double equal sign in comparison.

5)Referencing objects that does not exist.

6)Using reserved keywords for the variable naming.

7)Using the wrong type of brackets.

These are the main causes of these errors.  
  
In JavaScript, there are the following three types of errors:

a)Syntax Error

b)Runtime Error

c)Logic Error

**10)  Explain Exception Handling in JavaScript**.

his is a new feature of JavaScript for handling the exceptions like in other programming languages, using try catch finally statements and the throw operator for handling the exceptions. Now, you can catch the runtime exception.  
  
The try block is used always before a catch block, because in a try block, you provide your code to be executed. If there is no error in the code, the catch block is not executed but the final block is always executed. The following example shows an exception handling:

<!DOCTYPE html>

<html>

<title>Article By Jeetendra</title>

<head> </head>

<body>

<script type="text/javascript">

document.write("Exception Handling in JavaScript</br>");

function ExceptHand() {

try {

alert("This is try block");

alert("Not present");

} catch (error) {

document.write(error);

}

}

ExceptHand();

</script>

</body>

</html>

**11) Explain the performance of JavaScript code.**

* Reduction of activities in loops  
    
  In our programming, we often use the loops for iteration.  
    
  For each iteration of the loop, every statement inside a loop is executed.  
    
  The statements or assignments which are to be searched can be placed outside the loop.
* Reduction of DOM Access  
    
  As compared to other JavaScript statements, accessing the HTML DOM is very slow.  
    
  If you want to access the DOM element several times, access it once and use it as a local variable.  
    
  This is how you can access your DOM element easily as many times as you want.

Example:

<html>

<body>

<p id="dom"></p>

<script>

var obj;

obj = document.getElementById("dom");

obj.innerHTML = "Hello JavaScript..!!";

</script>

</body>

</html>

**Avoid Unnecessary variables**  
  
Avoid creating the new variables that are not of use to save any value.  
  
This will unnecessarily create a loading problem.  
  
You can replace your code by optimizing the code.

**12) Describe JavaScript Anonymous Functions.**

A function without a name is an anonymous function. We store these inside a variable name. Thus, the invocation happens , using the variable name.

Example:

var sum = function(a,b){return a+b;};

sum();

**13) What are the Event Handlers in JavaScript?**

 Event Handlers are considered as the triggers, which execute JavaScript when something happens, such as a click or move of your mouse over a link.  
  
Here, I’ll try to provide the proper syntax of some event handlers.  
  
Some are as follows:

* onClick
* onFocus
* onLoad
* onMouseOver
* onMouseOut
* onSelect
* onUnload