JavaScript Interview Questions for both Experienced Programmers and Freshers

**1) What is JavaScript?**   
Ans: JavaScript is a scripting language most often used for client-side web development.

**2) What is the difference between JavaScript and Jscript?**   
Ans:Both JavaScript and Jscript are almost similar. JavaScript was developed by Netscape. Microsoft reverse engineered Javascript and called it JScript.

**3) How do we add JavaScript onto a web page?**   
Ans:There are several way for adding JavaScript on a web page, but there are two ways which are commonly used by developers  
If your script code is very short and only for single page, then following ways are the best:  
a) You can place <script type="text/javascript"> tag inside the <head> element.

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

<head>

<title>Page Title</title>

<script language="JavaScript" type="text/javascript">

var name = "Vikas Ahlawta"

alert(name);

</script>

</head>

b) If your script code is very large, then you can make a JavaScript file and add its path in the following way:

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

<head>

<title>Page Title</title>

<script type="text/javascript" src="myjavascript.js"></script>

</head>

**4) Is JavaScript case sensitive?**   
Ans:Yes!  
A function getElementById is not the same as getElementbyID.

**5) What are the types used in JavaScript?**   
Ans:String, Number, Boolean, Function, Object, Null, Undefined.

**6) What are the boolean operators supported by JavaScript? And Operator: &&**  
Or Operator: ||  
Not Operator: !

**7) What is the difference between “==” and “===”?**   
Ans:  
“==” checks equality only,   
“===” checks for equality as well as the type.

**8) How to access the value of a textbox using JavaScript?**   
Ans: ex:-

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

<!DOCTYPE html>

<html>

<body>

Full name: <input type="text" id="txtFullName"

name="FirstName" value="Vikas Ahlawat">

</body>

</html>

There are following ways to access the value of the above textbox:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var name = document.getElementById('txtFullName').value;

alert(name);

or:

we can use the old way:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

document.forms[0].mybutton.

var name = document.forms[0].FirstName.value;

alert(name);

**Note**: This uses the "name" attribute of the element to locate it.

**9) What are the ways of making comments in JavaScript?**   
Ans:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

*// is used for line comments*

ex:- var x=10; *//comment text*

*/\**

*\*/* is used for block comments

ex:-

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var x= 10; */\* this is*

*block comment example.\*/*

**10) How will you get the**Checkbox **status whether it is checked or not?**   
Ans:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var status = document.getElementById('checkbox1').checked;

alert(status);

will return true or false.

**11) How to create arrays in JavaScript?**   
Ans:There are two ways to create array in JavaScript like other languages:

a) The first way to create array  
Declare Array:

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var names = new Array();

Add Elements in Array:-

names[0] = "Vikas";

names[1] = "Ashish";

names[2] = "Nikhil";

b) This is the second way:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var names = new Array("Vikas", "Ashish", "Nikhil");

**12) If an array with name as "names" contain three elements, then how will you print the third element of this array?**   
Ans: Print third array element document.write(names[2]);   
Note:- Array index starts with 0.

**13) How do you submit a form using JavaScript?**   
Ans:Use document.forms[0].submit();

**14) What does isNaN function do?**   
Ans: It returns true if the argument is not a number.  
Example:

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

document.write(isNaN("Hello")+ "<br>");

document.write(isNaN("2013/06/23")+ "<br>");

document.write(isNaN(123)+ "<br>");

The output will be:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

true

true

false

**15) What is the use of Math Object in JavaScript?**   
Ans: The math object provides you properties and methods for mathematical constants and functions.  
ex:-

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var x = Math.PI; *// Returns PI*

var y = Math.sqrt(16); *// Returns the square root of 16*

var z = Math.sin(90); Returns the sine of 90

**16) What do you understand by this keyword in JavaScript?**   
Ans: In JavaScript the this is a context-pointer and not an object pointer. It gives you the top-most context that is placed on the stack. The following gives two different results (in the browser, where by-default the window object is the 0-level context):

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var obj = { outerWidth : 20 };

function say() {

alert(this.outerWidth);

}

say();*//will alert window.outerWidth*

say.apply(obj);*//will alert obj.outerWidth*

**17) What does "1"+2+4 evaluate to?**   
Ans: Since 1 is a string, everything is a string, so the result is 124.

**18) What does 3+4+"7" evaluate to?**   
Ans: Since 3 and 4 are integers, this is number arithmetic, since 7 is a string, it is concatenation, so 77 is the result.

**19) How do you change the style/class on any element using JavaScript?**   
Ans:

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

document.getElementById(“myText”).style.fontSize = “10";

-or-

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

document.getElementById(“myText”).className = “anyclass”;

**20) Does JavaScript support foreach loop?**   
Ans: JavaScript 1.6(ECMAScript 5th Edition) support foreach loop,

See example here <http://jsfiddle.net/gpDWk/>

**21) What looping structures are there in JavaScript?**   
Ans: for, while, do-while loops

**22) What is an object in JavaScript, give an example?**   
Ans: An object is just a container for a collection of named values:  
  
// Create the man object

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var man = new Object();

man.name = 'Vikas Ahlawat';

man.living = true;

man.age = 27;

**23) How you will add function as a property in a JavaScript object? Give an example.**   
Ans:

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var man = new Object();

man.name = 'Vikas Ahlawat';

man.living = true;

man.age = 27;

man.getName = function() { return man.name;}

console.log(man.getName()); *// Logs 'Vikas Ahlawat'.*

**24) What is the similarity between the 1st and 2nd statement?**   
1st:- var myString = new String('male'); // An object.  
2nd:- var myStringLiteral = 'male'; // Primitive string value, not an object.  
Ans: Both will call String() constructor function  
You can confirm it by running the following statement:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

console.log(myString.constructor, myStringLiteral.constructor);

**25) What will be the output of the following statements?**

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var myString = 'Vikas' *// Create a primitive string object.*

var myStringCopy = myString; *// Copy its value into a new variable.*

var myString = null; *// Manipulate the value*

console.log(myString, myStringCopy);

Ans: *// Logs 'null Vikas'*

**26) Consider the following statements and tell what would be the output of the logs statements?**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var price1 = 10;

var price2 = 10;

var price3 = new Number('10'); *// A complex numeric object because new was used.*

console.log(price1 === price2);

console.log(price1 === price3);

Ans:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

console.log(price1 === price2); *// Logs true.*

console.log(price1 === price3); */\* Logs false because price3*

*contains a complex number object and price 1*

*is a primitive value. \*/*

**27) What would be the output of the following statements?**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var object1 = { same: 'same' };

var object2 = { same: 'same' };

console.log(object1 === object2);

Ans: // Logs false, JavaScript does not care that they are identical and of the same object type.  
When comparing complex objects, they are equal only when they reference the same object (i.e., have the same address). Two variables containing identical objects are not equal to each other since they do not actually point at the same object.

**28) What would be the output of the following statements?**

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var object1 = { same: 'same' };

var object2 = object1;

console.log(object1 === object2);

Ans: // Logs true

**29) What is this?**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var myArray = [[[]]];

Ans: Three dimensional array

**30) Name any two JavaScript functions which are used to convert nonnumeric values into numbers?**   
Ans:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

Number()

parseInt()

parseFloat()

**Code**

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var n1 = Number(“Hello world!”); *//NaN*

var n2 = Number(“”); *//0*

var n3 = Number(“000010”); *//10*

var n4 = Number(true); *//1*

var n5 = Number(NaN); *//NaN*

**31) Does JavaScript Support automatic type conversion, If yes give example.**

Ans: Yes! Javascript support automatic type conversion. You should take advantage of it, It is most common way of type conversion used by Javascript developers.

Ex.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/620811/Latest-JavaScript-Interview-Questions-and-Answers)

var s = '5';

var a = s\*1;

var b = +s;

typeof(s); *//"string"*

typeof(a); *//"number"*

typeof(b); *//"number"*

**32) In JavaScript, How to write read-only property that cannot be changed.**

**First Way**

var testObj = Object.defineProperties({}, {

prop1: {

value: 10,

writable: false // by default

},

prop2: {

get: function () {

}

}

});

testObj.prop1 = 20;

testObj.prop2 = 30;

A *data property* is a property that can get and set a value. Data properties contain the **value** and **writable** properties in their descriptors. The following table lists the attributes for a data property descriptor.

|  |  |  |
| --- | --- | --- |
| **Data descriptor attribute** | **Description** | **Default** |
| **value** | The current value of the property. | **undefined** |
| **writable** | **true** or **false**. If **writable** is set to **true**, the property value can be modified. | **false** |
| **enumerable** | **true** or **false**. If **enumerable** is set to **true**, the property can be enumerated by a **for…in** statement. | **false** |
| **configurable** | **true** or **false**. If **configurable** is set to **true**, property attributes can be changed, and the property can be deleted. | **false** |

**Second Way**

// Freeze the object. 🡺 It prevent user can not update and add the property and not delete the property

Object.freeze(obj);

// Seal the object. 🡺 It prevent user can not add new property and not delete the property.

Object.seal(obj);

**33) The following table lists the most important restrictions that apply in strict mode.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Language element** | **Restriction** | **Error** | **Example** |
| Variable | Using a variable without declaring it. | SCRIPT5042: Variable undefined in strict mode | JavaScript  testvar = 4; |
| Read-only property | Writing to a read-only property. | SCRIPT5045: Assignment to read-only properties is not allowed in strict mode | JavaScript  var testObj = Object.defineProperties({}, {  prop1: {  value: 10,  writable: false // by default  },  prop2: {  get: function () {  }  }  });  testObj.prop1 = 20;  testObj.prop2 = 30; |
| Non-extensible property | Adding a property to an object whose**extensible**attribute is set to **false**. | SCRIPT5046: Cannot create property for a non-extensible object | JavaScript  var testObj = new Object();  Object.preventExtensions(testObj);  testObj.name = "Bob"; |
| **delete** | Deleting a variable, a function, or an argument.  Deleting a property whose**configurable**attribute is set to **false**. | SCRIPT1045: Calling delete on <expression>is not allowed in strict mode | JavaScript  var testvar = 15;  function testFunc() {};  delete testvar;  delete testFunc;  Object.defineProperty(testObj, "testvar", {  value: 10,  configurable: false  });  delete testObj.testvar; |
| Duplicating a property | Defining a property more than once in an object literal. | SCRIPT1046: Multiple definitions of a property not allowed in strict mode | JavaScript  var testObj = {  prop1: 10,  prop2: 15,  prop1: 20  }; |
| Duplicating a parameter name | Using a parameter name more than once in a function. | SCRIPT1038: Duplicate formal parameter names not allowed in strict mode | JavaScript  function testFunc(param1, param1) {  return 1;  }; |
| Future reserved keywords | Using a future reserved keyword as a variable or function name. | SCRIPT1050: The use of a future reserved word for an identifier is invalid. The identifier name is reserved in strict mode. | * **implements** * **interface** * **package** * **private** * **protected** * **public** * **static** * **yield** |
| Octals | Assigning an octal value to a numeric literal, or attempting to use an escape on an octal value. | SCRIPT1039: Octal numeric literals and escape characters not allowed in strict mode | JavaScript  var testoctal = 010;  var testescape = \010; |
| **this** | The value of**this** is not converted to the global object when it is **null** or**undefined**. |  | JavaScript  function testFunc() {  return this;  }  var testvar = testFunc();  In non-strict mode, the value of testvar is the global object, but in strict mode the value is **undefined**. |
| **eval** as an identifier | The string "eval" cannot be used as an identifier (variable or function name, parameter name, and so on). |  | JavaScript  var eval = 10; |
| Function declared inside a statement or a block | You cannot declare a function inside a statement or a block. | SCRIPT1047: In strict mode, function declarations cannot be nested inside a statement or block. They may only appear at the top level or directly inside a function body. | JScript  var arr = [1, 2, 3, 4, 5];  var index = null;  for (index in arr) {  function myFunc() {};  } |
| Variable declared inside an **eval**function | If a variable is declared inside an**eval**function, it cannot be used outside that function. | SCRIPT1041: Invalid usage of 'eval' in strict mode | JavaScript  eval("var testvar = 10");  testvar = 15;  Indirect evaluation is possible, but you still cannot use a variable declared outside the **eval** function.  JavaScript  var indirectEval = eval;  indirectEval("var testvar = 10;");  document.write(testVar);  This code causes an error SCRIPT5009: 'testVar' is undefined. |
| **Arguments** as an identifier | The string "arguments" cannot be used as an identifier (variable or function name, parameter name, and so on). | SCRIPT1042: Invalid usage of 'arguments' in strict mode | JavaScript  var arguments = 10; |
| **arguments** inside a function | You cannot change the values of members of the local**arguments**object. |  | JavaScript  function testArgs(oneArg) {  arguments[0] = 20;  }  In non-strict mode, you can change the value of the oneArgparameter by changing the value of arguments[0], so that the value of both oneArg and arguments[0] is 20. In strict mode, changing the value of arguments[0] does not affect the value of oneArg, because the **arguments** object is merely a local copy. |
| **arguments.callee** | Not allowed. |  | JavaScript  function (testInt) {  if (testInt-- == 0)  return;  arguments.callee(testInt--);  } |
| **with** | Not allowed. | SCRIPT1037: 'with' statements are not allowed in strict mode | JavaScript  with (Math){  x = cos(3);  y = tan(7);  } |

**34) Public/Private Variables and Methods**

**private variables** are declared with the 'var' keyword inside the object, and can only be accessed by private functions and privileged methods.

**private functions** are declared inline inside the object's constructor (or alternatively may be defined via var functionName=function**(){**...**}**) and may only be called by privileged methods (including the object's constructor).

**privileged methods** are declared with this.methodName=function**(){**...**}** and may invoked by code external to the object.

**public properties** are declared with this.variableName and may be read/written from outside the object.

**public methods** are defined by Classname.prototype.methodName = function**(){**...**}** and may be called from outside the object.

**prototype properties** are defined by Classname.prototype.propertyName = someValue

**static properties** are defined by Classname.propertyName = someValue

**35) “This” keyword uses in Javascript**

By default, this refers to the global object.

When a function is called as a property on a parent object, this refers to the parent object inside that function.

When a function is called with the new operator, this refers to the newly created object inside that function.

When a function is called using call or apply, this refers to the first argument passed to call or apply. If the first argument is null or not an object, this refers to the global object.

**How to use “apply” method in javascript**

<html>

<head>

<title></title>

<script type="text/javascript">

//DEMO: using 'Apply' method with parameters

//define a function

var Emp = function (vId, vName) {

this.Id = vId;

this.Name = vName;

//method

this.ShowDetails = function (IdCaption, NameCaption) {

alert(IdCaption + this.Id.toString() + ", " + NameCaption + this.Name);

}

}

//define another function,

//which contains a couple of properties similar to previous function

var Customer = function (vId, vName) {

this.Id = vId;

this.Name = vName;

}

//create instances

var oCustomer = new Customer(1001, "Jag");

var oEmp = new Emp(2001, "Chat");

//call ShowDetails on behalf of oCustomer using "apply" (similar to oCustomer.ShowDetails)

oEmp.ShowDetails.apply(oCustomer, ["Id : ", "Name : "]); //'apply' needs parameters as array

//the above is equivalent to "call" as shown below

//oEmp.ShowDetails.call(oCustomer, "Id : ", "Name : ");

</script>

</head>

<body>

</body>

</html>

**How to use “Call” method in javascript**

<script type="text/javascript">

//DEMO: using 'Call' method

//define a function

var Emp = function (vId, vName) {

this.Id = vId;

this.Name = vName;

//method

this.ShowDetails = function () {

alert("Id = " + this.Id.toString() + ", Name = " + this.Name);

}

}

//define another function,

//which contains a couple of properties similar to previous function

var Customer = function (vId, vName) {

this.Id = vId;

this.Name = vName;

}

//create instances

var oCustomer = new Customer(1001, "Jag");

var oEmp = new Emp(2001, "Chat");

//call ShowDetails on behalf of oCustomer (temporarily, "ShowDetails" becomes a method of oCustomer)

//similar to oCustomer.ShowDetails (only for the statement)

oEmp.ShowDetails.call(oCustomer);

//oEmp.ShowDetails().call(oCustomer); //this line is different from the above, calls oEmp.ShowDetails()

</script>

# How to override methods in JavaScript OOPS (without using "prototype")

<script type="text/javascript">

//DEMO: Overriding methods (without using Prototype)

//create parent class

var Person = function (vId, vName) {

this.Id = vId;

this.Name = vName;

this.Display = function () {

alert("Id=" + this.Id.toString() +

", Name=" + this.Name);

}

};

//create new class

var Emp = function (vId, vName, vOfficeMail) {

Person.call(this, vId, vName)

this.OfficeEmail = vOfficeMail;

//overriding method

this.Display = function () {

alert("Id=" + this.Id.toString() +

", Name=" + this.Name +

", OfficeMail=" + this.OfficeEmail);

}

};

//create instance of parent class

var oPerson = new Person(1002, "Chat"); //using Parent's constructor

//call display method in parent class

oPerson.Display();

//create instance of child class

var oEmp = new Emp(1001, "Jag", "a@a.com"); //using Child's constructor

//call display method in child class

oEmp.Display();

</script>

# Advantage of adding members to a class using "prototype" (in JavaScript)

<script type="text/javascript">

//DEMO: advantage of adding methods to prototype

//create function

var Emp = function (vEmpno, vEname, vSal) {

this.Empno = vEmpno;

this.Ename = vEname;

this.Sal = vSal;

//method added normally (a copy will be created for every instance)

this.ShowAnnSal = function () {

alert("Annual Salary: " + (this.Sal \* 12).toString());

}

};

//method added through prototype (same copy will be shared by all instances)

Emp.prototype.ShowQuarterSal = function () {

alert("Quarter Salary: " + (this.Sal \* 3).toString());

}

//create objects

var oEmp1 = new Emp(1001, "Jag", 3400);

var oEmp2 = new Emp(1002, "Chat", 4400);

//demonstrating the difference

alert(oEmp1.ShowAnnSal == oEmp2.ShowAnnSal); //false - each object maintains a copy of method (more memory)

alert(oEmp1.ShowQuarterSal == oEmp2.ShowQuarterSal); //true - all objects share same method (less memory)

</script>

# Anonymous constructor functions to work as classes in JavaScript

<script type="text/javascript">

//DEMO: using anonymous constructors functions to define objects

// using "with" keyword with an object

//defining a (class/anonymous function) constructor accepting 4 parameters

var Emp = function (vEmpno, vEname, vSal, vDeptno) {

this.Empno = vEmpno;

this.Ename = vEname;

this.Sal = vSal;

this.Deptno = vDeptno;

};

//function to display objects

function ShowEmp(o) {

//accessing members of an object

//alert(o.Empno.toString() + ", " +

// o.Ename + ", " +

// o.Sal.toString() + ", " +

// o.Deptno.toString()

//);

//another way of accessing object members using "with" construct

with (o) {

alert(Empno.toString() + ", " +

Ename + ", " +

Sal.toString() + ", " +

Deptno.toString()

);

}

}

//creating (multiple) objects

var oEmp1 = new Emp(1001, "Jag", 5400, 10);

var oEmp2 = new Emp(1002, "Chat", 6400, 20);

var oEmp3 = new Emp(1003, "Win", 7800, 10);

var oEmp4 = new Emp(1004, "Dhan", 9000, 10);

//display all objects

ShowEmp(oEmp1);

ShowEmp(oEmp2);

ShowEmp(oEmp3);

ShowEmp(oEmp4);

</script>

# How to add or remove properties from a JavaScript object dynamically

<script type="text/javascript">

//DEMO: Deleting/adding properties dynamically

var oEmp = {

Empno:1001,

Ename:"Jag"

};

//display properties using a function

function dispEmp(o){

if (o.Sal) { //check if "Sal" property exists

alert(o.Ename + ' earns ' + o.Sal.toString());

}

else {

alert("Sal property does not exist");

}

}

//add property "Sal" dynamically

oEmp.Sal = 5400;

dispEmp(oEmp); //call unction to display object details

//remove "Sal" property from the object (dynamically)

delete oEmp.Sal;

dispEmp(oEmp);//call function to display object details

//same is possible using constructor functions

var Emp = function () {

this.Empno = 1001;

this.Ename = "Jag";

};

var oEmp2 = new Emp();

oEmp2.Sal = 5400;

dispEmp(oEmp2); //call function to display object details

delete oEmp2.Ename;

if(!oEmp2.Ename) alert("Property Ename got removed");

</script>

# Introducing Callback functions in JavaScript

<html>

<head>

<title></title>

<script>

//DEMO: callback functions (functions as parameters)

//function accepting another function (doProcessAfter) as parameter

function DoProcess(a, b, doProcessAfter) {

alert("Sum = " + (a + b).toString());

//calling/executing the function passed as parameter

doProcessAfter(a, b); //callback

}

//function which (itself) is passed as parameter (to DoProcess)

function ShowDifference(p, q) {

alert("Difference = " + (p - q).toString());

}

//function which (itself) is passed as parameter (to DoProcess)

function ShowMultiplied(m, n) {

alert("Multiplied = " + (m \* n).toString());

}

</script>

</head>

<body>

<!--Calling a function and passing a function as parameter (for callback)-->

<input type="button" value="Show" onclick="DoProcess(10, 20, ShowDifference);"/><br>

<!--passing a different function as parameter -->

<input type="button" value="Another Show" onclick="DoProcess(10, 20, ShowMultiplied);"/><br>

<!--Calling a function and passing an anonymous function as parameter -->

<input type="button" value="Just Another Show" onclick="DoProcess(10, 20, function(p, q){alert('Difference = ' + (p-q).toString());} );"/><br>

</body>

</html>

# Array methods in JavaScript

<script>

//DEMO: using Callback function to work with Array.forEach method

//"Array.forEach" method executes callback function for every element present in the array

//The callback function need to take three arguments (value, index and array)

function doProcess(v, i, ary) {

ary[i] = v + 1; //increment the current element value

}

//define array

var a = [2, 45, 26];

//execute "forEach" method by passing callback function

a.forEach(doProcess);

alert(a); //displays "3, 46, 27"

</script>

<script>

//DEMO: using Callback function to work with Array.filter method

//"Array.filter" method returns a new array that meet the condition specified in a callback function

//The callback function need to take three arguments (value, index and array)

function isEven(v, i, ary) {

//return true if division by two is zero, else return false

return (v % 2 == 0) ? true : false;

}

//define array

var a = [2, 45, 26, 39, 12, 7, 9];

//execute "filter" method by passing callback function

var b = a.filter(isEven); //"b" is new array with filtered elements from array "a"

alert(b); //displays "2,6,12"

</script>

<script>

//DEMO: using Callback function to work with Array.some method

//"Array.some" method returns true if any of the elements

// in the array are evaluated to true through a Callback function

//The callback function need to take three arguments (value, index and array)

function isEven(v, i, ary) {

//return true if division by two is zero, else return false

return (v % 2 == 0) ? true : false;

}

//define array

var a = [9, 37, 15, 4, 3, 11]; //all odds except one even

//execute "some" method by passing callback function

alert(a.some(isEven)); //displays "true"

</script>

<script>

//DEMO: using Callback function to work with Array.map method

//"Array.map" method executes callback function for every element present in the array and

//gives a new array with processed element (returned value) for the corresponding source array element

//The callback function need to take three arguments (value, index and array)

function doProcess(v, i, ary) {

return v \* v; //new element in 'i'th location of new array

}

//define array

var a = [2, 25, 10];

//execute "map" method by passing callback function

var b = a.map(doProcess);

alert(b); //displays "4, 625, 100"

</script>

<script>

//DEMO: using Callback/anonymous function to work with Array.every method

//"Array.every" method returns true if all the elements

// in the array are evaluated to true through a Callback function

//The "Array.every" method calls Callback function one time for every array element

//The callback function gets executed for every element, as long as it does not evaluate to false

//The "every" method returns false once the callback function returns false

//The "every" method returns true once all elements are evaluated to true by callback function

//The callback function need to take three arguments (value, index and array)

function isEven(v, i, ary) {

//return true if division by two is zero, else return false

return (v % 2 == 0) ? true : false;

}

//define array

var a = [2, 26, 38, 12, 6, 8];

//execute "every" method by passing callback function

alert(a.every(isEven)); //displays "true"

//can also do like this

alert([12, 226, 738, 312, 7, 8].every(isEven)); //displays "false"

//can also do like this (callback using anonymous function)

alert([12, 226, 738, 312, 8].every(function (v, i, ary) {

return (v % 2 == 0);

})); //displays "true"

</script>

**JSON versus JSONP**

[JSON (Javascript Object Notation)](http://en.wikipedia.org/wiki/JSON) is a convenient way to transport data between applications, especially when the destination is a Javascript application.

But this simple approach fails if the page making the ajax call is in a different domain from the server. The [Same Origin Policy](http://en.wikipedia.org/wiki/Same_origin_policy) prohibits these cross-domain calls in some browsers as a security measure.

At the time of writing, Google Chrome version 24 and Mozilla Firefox version 17 do not appear to apply this restriction but Internet Explorer version 9 does.

The security implications of allowing cross domain requests should be considered carefully in your application but if you do want to allow them then you need a way to overcome the browser restrictions.

[JSONP (JSON with Padding)](http://en.wikipedia.org/wiki/JSONP) makes this possible in all browsers.

JSONP wraps up a JSON response into a JavaScript function and sends that back as a Script to the browser. A script is not subject to the Same Origin Policy and when loaded into the client, the function acts just like the JSON object that it contains.

JSONP is a simple way to overcome browser restrictions when sending JSON responses from different domains from the client.

**Basic JS programmming**

* Scope of variable
* What is Associative Array? How do we use it?

**OOPS JS**

* Difference between Classic Inheritance and Prototypical Inheritance
* What is difference between private variable, public variable and static variable? How we achieve this in JS?
* How to add/remove properties to object in run time?
* How to achieve inheritance?
* How to extend built-in objects?
* Why extending array is bad idea?

**DOM and JS**

* Difference between browser detection and feature detection
* DOM Event Propagation
* Event Delegation
* Event bubbling V/s Event Capturing

**Misc**

* Graceful Degradation V/s Progressive Enhancement

### Let's start with the JavaScript questions

#### 1. How can you declare a class in JavaScript?

In javascript there's no classes like in Java, what we actually call a class is in reality a function simulating a class behaviour. For being so flexible, there are many ways to create a class in javascript, below you'll find 3 ways of doing that.

* Class using function as a constructor:

function Person(name) {

this.name = name;

}

// Creating an object

var person = new Person("Rafael");

person.name; // "Rafael"

It's very important to notice that you have to use the keyword new when creating anew instance of that class otherwise you will have logical problems regarding the this will reference window object.

* Class Literal notation:

var person = {

name: "",

setName: function(name) {

this.name = name;

}

}

person.setName("Rafael");

person.name; // "Rafael"

In this example we don't use a function to define our class, we are creating a singleton object person with one attribute and one method. You can use that object straightaway, no instantiation in this case.   
That notation is useful when you don't need to create instances of that class or you'll use it just once in your application.

* Singleton through a function:

var person = new function() {

this.setName = function(name) {

this.name = name;

}

this.sayHi = function() {

return "Hi, my name is " + this.name;

}

}

person.setName("Rafael");

alert(person.sayHi()); // Hi, my name is Rafael

As you can see in the code snippet above, we have a function like the first example and besides we also have the new keyword before the function declaration. It means that we are creating one instance of that class at the same time we are declaring it.

#### 2. How would you organize your Javascript code?

The following pattern is the one that I personally prefer and is called 'module pattern', where we separate our javascript into logical modules, or namespaces. What you'll see below is an example of how I would separate my user module.

// Declaring my main namespace

var myapplication = myapplication || {};

// Declaring modules usermodule

myapplication.usermodule = (function() {

// createMessage: only accessible inside this module

var createMessage = function(message) {

return "Hello! " + message;

}

return {

// sayHello is a public method

sayHello: function(message) {

return createMessage(message);

}

}

})();

// Declaring another module

myapplication.adminmodule = (function(){

// your code here

})()

// This is how we call sayHello

myapplication.usermodule.sayHello("This is my module");

#### Some explanation on the code above

Take a look at the previous code and notice how I create my module using the notation below. It makes the function to be executed immediately because of the parenthesis at the end of the command. The result of the execution will be an object which will be set to my variable myapplication.usermodule.

...

myapplication.usermodule = (function() {

// code to be executed immediately

})();

So applying this pattern to your code you may have multiple modules and you have the control over what you want to make public and what to keep private. Besides your code will be more organized therefore easy to maintain.

#### 3. Difference between == and ===.

This is pretty simple but at the same time some people never came across a triple equals or never wondered what's the difference.   
Double equals == is used to compare the value of two operands:

"2" == 2; // true

2 == 2; // true

Triple equals === is used to compare the value AND type of two operands:

"2" === 2; // false

2 === 2; // true

#### 4. Difference between null and undefined

This can be tricky and the best way to keep in your head is to memorise because if you try to relate javascript null to other languages, it will get more confusing.   
In javascript, null is an object with no value and undefined is a type.

typeof null; // "object"

typeof undefined; // "undefined"

var a;

var b = null;

a == b; // "true" because their values are the same

a === b; // "false". they have different types

#### 5. Have you already used MVC before? What you like/dislike about it?

As the UI gets more and more complex we need some good ways to keep it more and more maintainable and reusable, and  Some MVC frameworks for javascript have been widely adopted lately and it's a good plus if you have already used before and knows what's the benefits of them. The most famous MVC frameworks are[backbone.js](http://documentcloud.github.io/backbone/) and [angular.js](http://docs.angularjs.org/), it's hard to not hear about them.

There are many advantages in using these frameworks, I can point out some of them:

* **Organization:** Forces your webapp to follow a well structured pattern;
* **Maintainable:** With organization comes an easy to maintain code;
* **UI Binding:** Some frameworks allow you to do that. So everytime your model changes, the view reflects it and vice-versa;
* **Decoupled client:** MVC frameworks like backbone.js incentivise you to use REST API's though their urlRoot attribute in their Models;
* **Reusable components:** Create reusable visual components;
* **Single-page apps:** Build single-page apps with Ajax requests;
* **Friendly URL's:** Native support for client-side url mapping;

#### 6. How can you add a method to a class already defined?

You can add a new method to a javascript class using prototype:

function Person(name) {

this.name = name;

}

Person.prototype.walk = function() {

console.debug(this.name + " is walking.");

}

// Calling the new method

var person = new Person("Rafael");

person.walk(); // "Rafael is walking."

It's worth mentioning that adding methods via prototype is the most inexpensive way in terms of performance since the method is tied to the prototype of the class. It means, for every new instance of class Person, you will have access to the prototype's walk() method. Now, if you declare walk() method inside the Person class, you will end up recreating the method for every new instance of Person.

### CSS Questions

#### 1. When would you use CSS float?

Float is used when you want to make an element of your page (usually an image) be pushed to the right or left and make other elements wrap around it.

#### 2. When would you use CSS clear?

When you want an element on the left or right of the floating element not to wrap around it, you can use clear.

#### 3. Have you used Sass? What's good about it?

Every web project starts with everything neat, all CSS is organized in blocks or different CSS files and you know where everything is, right?   
Right, until your project gets bigger, deadlines get tight, more developers come on board and someday you notice a strange behaviour in some elements of the page. When you inspect their styles you spot lots of css overrides coming from everywhere. This is the moment you realise how messy CSS can be.

Sass is the modern way of doing CSS and can save many lines of code in your stylesheets. This is possible because Sass works with variables, nested syntax and mathematical operations.   
In my opinion one of the nicest features of sass is the possibility to write a selector just once and put all styles for that inside it. Do you need a more specific selector under an existing one? Just nest the specifics into the generic one.

Check out the example below taken from their official website. It's awesome how it can "neatify" your code.

/\* .sass \*/

table.hl

margin: 2em 0

td.ln

text-align: right

li

font:

family: serif

weight: bold

size: 1.2em

/\* .css \*/

table.hl {

margin: 2em 0;

}

table.hl td.ln {

text-align: right;

}

li {

font-family: serif;

font-weight: bold;

font-size: 1.2em;

}

### Other questions

#### 1. What can you do to improve page performance?

In a nutshell page performance is widely understood as the page load time from the users' perspective, so below are some steps that might improve a page's performance.

* **Use sprite images** whenever possible, try to group small images commonly used in a single file to be requested just once. See how Google uses sprites in Google Maps to make one request instead of one for each small image.
* **Javascripts should be at the bottom of the page**, instead of in the head as we use to see out there;
* **Ensure parallel requests** of your JS and CSS files. In order to force the browser to do that, you can optimize the order you include resources in your page. This item can generate its own blog post or even a book so I prefer to suggest you a really good reading about it. [Check this out](https://developers.google.com/speed/docs/best-practices/rtt), it's the google's best practices on page speed load.
* **Compress images** whenever possible, it makes a difference;
* **Browser Caching** is also very import to be set for static resources like JS and CSS files, images, PDFs and HTML. Caching is set in the HTTP header by informing browsers the expiry date or maximum age. Then browsers can load the last downloaded resource from the cache instead of request it again.

#### 2. What's the difference between HTML and XHTML?

XHTML is an HTML that follows the XML rules, which means a XHTML document must have well-formed markups in order to be rendered properly in all web browsers. Differently from XHTML, the HTML document can be rendered in most of the browsers even with markup errors such as no closing tags or wrong nested tags.

And how do I create a XHTML document?

XHTML is basically a HTML document with some rules that need to be applied. Have a look at these examples below and spot the differences.

<head>

<title>This is head</title>

</head>

<BODY>

This is the body of the document with body tag in capital letters

Notice that there's no close body tag and no tag as well.

This HTML document above can be opened with no problems in Chrome, even containing many markup errors because most browsers can fix them for you automatically.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

</head>

<body>

</body>

</html>

The code above is a well-formed XHTML document and that's the minimum you must have in order to render it. Notice the declaration of the doctype at the top of the document and the namespace (xmlns) attribute in html open tag. These elements are mandatory as well as all the tags in lowercase.

#### 3. When would you use GET and POST requests?

There are several technical differences between these two types of requests, regarding length limitation, security, caching and a few others. But if someone asks you WHEN would you use it, I'd say one of the most important points that any front-end developer should take into account is that we should only use GET for**idempotent** requests, it means requests that don't make significant changes in the backend system or database but if you do need to make inserts, updates or deletes in a database, trigger emails or any other major action, POST is recommended.

That's pretty much it that I wanted to share according to my recent interviews and I believe you'll come across some of these questions or a variation of them in your next technical interview.