39 Best Object Oriented JavaScript Interview Questions and Answers

Anil Singh 1:25 AM

Most Popular 37 Key Questions for JavaScript Interviews.

What is Object in JavaScript?	What is the Prototype object in JavaScript and how it is used?
What is "this"? What is its value?	Explain why "self" is needed instead of "this".
What is a Closure and why are they so useful to us?	Explain how to write class methods vs. instance methods.
Can you explain the difference between == and ===?	Can you explain the difference between call and apply?
Explain why Asynchronous code is important in JavaScript?	Can you please tell me a story about a JavaScript performance problems?
Tell me your JavaScript Naming Convention?	How do you define a class and its constructor?
What is Hoisted in JavaScript?	What is function overloading in JavaScript?
What is scope variable in JavaScript?	What is Associative Arrays in JavaScript?
How to Achieve Inheritance in JavaScript?	What are two-way data binding and one-way data flow, and how are they different?
What will be the output of the following code?	What will be the output of the JavaScript code?
How to empty an array in JavaScript?	What is the difference between the function declarations below?
"1"+2+3 = ? And 1+2+"3" = ? OutPut Will Be?	How to show progress bar while loading using Ajax call?
How To Create the Namespace in JavaScript?	What Is Difference Between Undefined and Object?
How To convert JSON Object to String?	How To convert JSON String to Object?
How To Convert a string to Lowercase?	How To modify the URL of page without reloading the page?
How To open URL in new tab in JavaScript?	TypeScript Interview Questions and Answers
JavaScript let vs var - When should let be used over var?	

"Need to work on your software development environment from anywhere from multiple devices? Take a free trial from a **Desktop-as-a-Service** provider such as CloudDesktopOnline.com. Add Office 365 applications to the desktop with full support from **O365CloudExperts"**

What is JavaScript?
The **JavaScript** is most popular scripting languages and it developed by **Netscape** and used to develop the client side **web** applications.

Is JavaScript Case Sensitive?
Yes! JavaScript is a case sensitive because a function "**str**" is not equal to "**Str**".

What Is the Type of JavaScript?
There are different of Type as given below.

1. String,
2. Number,
3. Boolean,
4. Function,
5. Object,
6. Null,
7. Undefined etc.

What Types of Boolean Operators in JavaScript?
There are three types of Boolean operators as given below.

1. AND (&&) operator,
2. OR (||) operator and
3. NOT (!) Operator

What Is the difference Between "==" and "==="?

The double equal "==" is an auto-type conversion and it checks only value not type.
The three equal "===" is not auto-type conversion and it check value and type both.

The example looks like–

```
if(1 == "1")
//Its returns true because it's an auto-type conversion and it checks only value not type.

if(1 === "1")
//Its returns false because it's not auto-type conversion and it check value and type both.

if(1=== parseInt("1"))
// its returns true.
// alert(0 == false); // return true, because both are same type.
// alert(0 === false); // return false, because both are of a different type.
// alert(1 == "1"); // return true, automatic type conversion for value only.
// alert(1 === "1"); // return false, because both are of a different type.
// alert(null == undefined); // return true.
// alert(null === undefined); // return false.
// alert('0' == false); // return true.
// alert('0' === false); // return false.
// alert(1=== parseInt("1")); // return true.
```

What Is an Object?

The object is a collection of properties and the each property associated with the **name-value pairs**. The object can contain any data types (numbers, arrays, object etc.)

The example as given below –

```
var myObject= {empId : "001", empCode :"X0091"};
```

In the above example, here are two properties one is empId and other is empCode and its values are "**001**" and "**X0091**".

The properties name can me string or number. If a property name is number i.e.

```
var numObject= {1 : "001", 2 :"X0091"};
Console.log(numObject.1); //This line throw an error.

Console.log(numObject["1"]); // will access to this line not get any error.
```

As per my thought, the number property name should be avoided.

Types of creating an object

1. Object literals
2. Object constructor

Object Literals: This is the most common way to create the object with object literal and the example as given below.

The empty object initialized using object literal i.e.

```
var emptyObj = {};
```

This is an object with 4 items using object literal i.e.

```
var emptyObj ={
empId:"Red",
empCode: "X0091",
empDetail : function(){
   alert("Hi");
  };
};
```

Object Constructor: The second way to create object using object constructor and the constructor is a function used to initialize new object.

The example as given below -

```
var obj = new Object();
Obj.empId="001";
Obj.empCode="X0091";
Obj.empAddressDetai = function(){
console.log("Hi, I Anil");
};
```

What Is Scope Variable in JavaScript?

The scope is set of objects, variables and function and the JavaScript can have global scope variable and local scope variable.

The global scope is a window object and it's used out of function and within the functions.
The local scope is a function object and it's used within the functions.

The example for global scope variable -

```
var gblVar = "Anil Singh";

function getDetail() {
    console.log(gblVar);
}
```

And other example for global -

```
function demoTest() {
     x = 15;
};
console.log(x); //output is 15
```

The example for local scope variable -

```
function getDetail() {
    var gblVar = "Anil Singh";
    console.log(gblVar);
}
```

And other example for local -

```
function demoText() {
    var x = 15;
};

console.log(x); //undefined
```

What Is an Associative Array in JavaScript?

What is array?

Array is a collection of index items and it is a number indexes.

Some of programming language support array as named indexes and the JavaScript not support the array as name indexes and its provide only number indexes but provide this feature using the associative array.

The array with name indexes are called associative array and the associative array is provide a ways to store the information.

The number index array example as given below -

```
var users = new Object();
users["Name1"] = "Anil 1";
users["Name2"] = "Anil 2";
users["Age"] = 33;

alert(Object.keys(users).length); //output is 3.
var length = Object.keys(users).length;  // 3
```

The name index array example as given below -

```
var users = [];
users["Name1"] = "Anil 1";
users["Name2"] = "Anil 2";
users["Age"] = 33;

var length = users.length;         // users.length will return 0
var detail = users[0];             // users[0] will return undefined
```

Where To Use the Associate Array?

I am going to explain the associate array over the database table columns. A table have some columns and its type i.e.

The **empName** as text type, **empAge** as number type and **enpDOB** as date type.

If we need to find the type of a column that time we can create the associate array i.e.

```
var empDetailType = new Array();

empDetailType["empName"] = "ANil";
empDetailType["empAge"] = 30;
empDetailType["empDOB"] = "10/03/1984";

console.log("Find the emp age type :" + empDetailType["empAge"]);
```

How To Achieve Inheritance in JavaScript?

In the JavaScript, we can implement the inheritance using the some alternative ways and we cannot define a class keyword but we create a constructor function and using new keyword achieves it.

The some alternative ways as given below –

1. Pseudo classical inheritance
2. Prototype inheritance

Pseudo classical inheritance is the most popular way. In this way, create a constructor function using the new operator and add the members function with the help for constructor function prototype.

The **prototype based** programming is a technique of object oriented programming. In this mechanism we can reuse the exiting objects as prototypes. The prototype inheritance also known as prototypal, Classless or instance based inheritances.

The Inheritance example for **prototype based** as given below –

```
//create a helper function.
if (typeof Object.create !== 'function') {

Object.create = function (obj) {
function fun() { };

fun.prototype = obj;
return new fun();
};
}

//This is a parent class.
var parent = {
sayHi: function () {
alert('Hi, I am parent!');
},

sayHiToWalk: function () {
alert('Hi, I am parent! and going to walk!');
}
};

//This is child class and the parent class is inherited in the child class.
var child = Object.create(parent);
child.sayHi = function () {
alert('Hi, I am a child!');
};
```

HTML –

```
<button type="submit" onclick="child.sayHi()"> click to oops</button>
//The output is: Hi, I am a child!
```

What Is typeof Operator?

The **typeof** operator is used to find the type of variables.

The example as given below -

```
typeof "Anil Singh"    // Returns string
typeof 3.33            // Returns number
typeof true           // Returns Boolean
typeof { name: 'Anil', age: 30 } // Returns object
typeof [10, 20, 30, 40] // Returns object
```

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

I am going to explain like strongly type object oriented language (OOPs) like(C#, C++ and java etc.).

Fist I am creating a conductor class and trying to achieve to declare the public, private and static variables and detail as given below –

```
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();
```

How To Create the Namespace in JavaScript?

Please see the below example for how to create the namespace in JavaScript.

```
//Create the namespace.
var nameSpace = nameSpace || {};

nameSpace.myEmpConsepts = function () {
var empId = "00201"; //This is a private variable.
this.empName = "Anil Singh"; //This is a public variable.

//This is public function
this.getEmp = function () {
return "Anil Singh"
}

//This is private function
var getEmp = function () {
return "Anil Singh"
}

return {

getEmp: getEmp,// work as public
getEmp: getEmp // work as public
}
}();
```

How to Add/Remove Properties to Object in run-time in JavaScript?

I am going to explain by example for add and remove properties from JavaScript objects as give below.

This example for delete property -

```
//This is the JSON object.
var objectJSON = {
id: 1,
name: "Anil Singh",
dept: "IT"
};

//This is the process to delete
delete objectJSON.dept;


//Delete property by the array collection
MyArrayColection.prototype.remove = function (index) {
this.splice(index, 3);
}
```

This example for add property -

```
//This is used to add the property.
objectJSON.age = 30;
console.log(objectJSON.age); //The result is 30;

//This is the JSON object.
```

```
var objectJSON = {
id: 1,
name: "Anil Singh",
dept: "IT",
age: 30
};
```

How To Extending built-in Objects in JavaScript?

JavaScript support **built-in objects** which use to develop the flexibility of JavaScript. The built in object are date, string, math, array and object.

It's very similar to other languages and it's available in the window content and wok independently when brewers are loaded.

Example as give below -

```
var date = new Date(); //This is date built-in object.
var math = Math.abs(10); // this is math built-in object.

var string = "my string" // this is string built-in object.
```

Why Never Use New Array in JavaScript?

We have some fundamental issues with **new Array ()** the example in detail for array constructor function as given below.

When Array Have more the one Integer?

```
var newArray = new Array(10, 20, 30, 40, 50);
console.log(newArray[0]); //returns 10.
console.log(newArray.length); //returns 5.
```

When Array Have Only One Integer?

```
var newArray = new Array(10);
console.log(newArray[0]); //returns undefined
console.log(newArray.length); //returns 10 because it has an error "array index out of bound";

//This is the fundamental deference to need to avoid the new array ();
```

What is eval() and floor() functions in JavaScript?

The **eval()** function used in execute an argument as expression or we can say that evaluate a string as expression and it used to parse the JSON.

The example over **eval()** function as given below -

```
var x = 14;
eval('x + 10'); //The output is 24.
```

Another over **eval()** function example -

```
eval('var myEval = 10');
console.log(myEval); // The output is 10.
```

The **floor ()** function is a static method of Math and we can write as **Math.floor()** and used to round the number of downwards i.e.

```
Math.floor(1.6);//The output is 1.
```

What is join() and isNaN() functions in JavaScript?

The is **join()** function used to join the separator in the array.

Syntax -

```
myArray.join(mySeparator);
```

The example as given below -

```
var alfabets = ["A", "B", "C", "D"];

//Join without seperator
var result1 = alfabets.join();//The output is A B C D.

//Join with seperator.
var result2 = alfabets.join(','); //The output is A, B, C, D.
```

The **isNaN()** function is used to check the value is not-a-number.

The example as given below -

```
var var1 = isNaN(-1.23);//The output is false.
var var2 = isNaN(3);//The output is false.
var var3 = isNaN(0);//The output is false.
var var3 = isNaN("10/03/1984"); //The output is true.
```

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.

```
//OR Other WAYS
function ourterFun(i) {
var var1 = 3;

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

var innerFun = ourterFun(2); // innerFun() function is now a closure.

innerFun(10);
```

What is JavaScript Hoisted?

In the JavaScript, the variables can be used before declared, this kinds of mechanism is called Hoisted. It's default behaviour of JavaScript.

You can easily understanding in the below example in detail -

```
//The variable declaration looks like.
var emp;

//The variable initialization looks like.
emp = "Anil Singh";

var emp; //The declaration of emp is hoisted but the value of emp is  undefined.
emp = 10; //The Assignment still occurs where we intended (The value of emp is 10)

function getEmp() {
var emp; //The declaration of a different variable name emp is hoisted but the value of emp is  undefined.
console.log(emp); //The output is undefined
emp = 20; //The assignment values is 20.
console.log(emp); //The output is 20.
}

getEmp();

console.log(emp); //The variable named emp in the outer scope still contains 10.
```

What Is Function Overloading in JavaScript?

There is no real function overloading in JavaScript and it allows passing any number of parameters of any type.

You have to check inside the function how many arguments have been passed and what is the type arguments using ***typeof***.

The example for function overloading not supporting in JavaScript as gives below -

```
function sum(a, b) {
   alert(a + b);
}
function sum(c) {
   alert(c);
}
sum(3);//The output is 3.
sum(2, 4);//The output is 2.
```

In the JavaScript, when we write more than one functions with same name that time JavaScript consider the last define function and override the previous functions. You can see the above example output for the same.

We can achieve using the several different techniques as give below -

1. You can check the declared argument name value is undefined.
2. We can check the total arguments with ***arguments.length***.
3. Checking the type of passing arguments.
4. Using number of arguments
5. Using optional arguments like x=x || 'default'
6. Using different name in the first place
7. We can use the arguments array to access any given argument by using ***arguments[i]***

What Is Prototype in JavaScript?

The prototype is a fundamental concept of JavaScript and its must to known JavaScript developers.

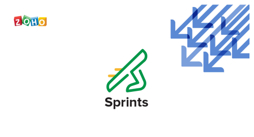
All the JavaScript objects have an object and its property called prototype and its used to add and the custom functions and property.

The example without prototype as given below -

```
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 functions and assign in a variable
var empInstance = new employee();
empInstance.deportment = "HR";

console.log(empInstance.deportment);//The output of above is IT not HR.
```

SHARES Share Pin it Tweet Share

Anil Singh is an author, tech blogger, and software programmer. Book writing, tech blogging is something do extra and Anil love doing it. For more detail, kindly refer to this link..
My Blogs - https://code-sample.com and https://code-sample.xyz
My Book1 - BEST SELLING ANGULAR BOOK (INCLUDING ALL VERSIONS 2, 4, 5, 6, 7)
My Book2 - ANGULAR 2 INTERVIEW QUESTIONS BOOK - Both Books are Available on WorldWide.

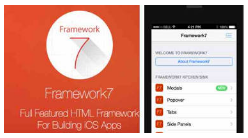


Agile complicated? Switch now.

Ad Zoho Sprints

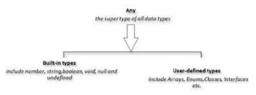
55 Best TypeScript Interview Questions and Answers -...

code-sample.com



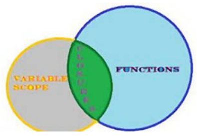
Object in JavaScript - Object Creation by Constructor, Function...

code-sample.com



What are Types in TypeScript?

code-sample.com



Closure in JavaScript

code-sample.com

What Is Angular 4 or 5 or 6?
Predictable, Transparent & Incremental Evolution
Version 4 March 2017
Version 5 September/October 2017
Version 6 March 2018
Version 7 September/October 2018

Angular 6 Interview Questions and Answers | A Complete Guide...

code-sample.com

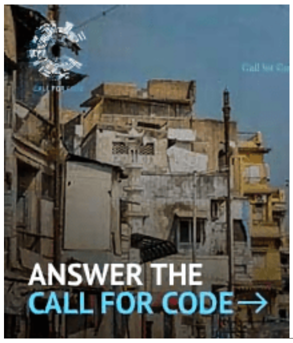
How To open URL in new tab in JavaScript?

code-sample.com

15 Best Grun Interview Qu and Answers

code-sample.com

Newer Post Home Older Post

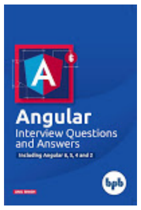


Agile complicated? Switch

Don't let the complex tools get your agility down.

Zoho Sprints

BEST SELLING ANGULAR BOOK (ALL-IN-ONE)


Angular
Interview Questions and Answers
Including Angular 6, 5, 4 and 2
bpb

All in One
JavaScript
TypeScript
Angular 1
Angular2
Angular 4
Angular 5, 6
Vue.js
Angular 4 Charts
Ionic 3
ASP.Net
KnockoutJs
HTML5
Database
Calculators
FAQ Interviews
Home

POPULAR POSTS

Angular 4 Interview Questions and Answers | A Complete Guide Book

69 Best Angular 2 Interview Questions and Answers [Angular 2]

C# OOPs
ASP.NET MVC
Design Pattern
Repository Pattern
WCF Security
Database
Software Testing
Apache Kafka

Ember.Js
Gulp.Js
Grunt.Js
Meteor.Js
VVVV.Js
DOJO
Ajax
Perl Programming

Donate By Using PayPal (safer easier way to pay)
AngularJs
Angular 2, 4, 5, and 6+
JavaScript
TypeScript
NodeJs
Go Programming

^

Angular 6 Interview Questions and Answers | A Complete Guide Book

Angular 6 Search Filter Pipe - Table by Columns

39 Best Object Oriented JavaScript Interview Questions and Answers

Angular 5 and 4 - Open and Close Modal Popup Using Typescript and Bootstrap

Java Play Framework
OpenXava
Groovy
Facebook Marketing
BackboneJs
Python
Vue.Js
LimeJS framework
Swift
D3.Js
Ext.Js
ReactJs
Vaadin
RequireJs

Spring Boot
Scala Programming
Maven
Google AdWords
AngularJs
Angular 2, 4, 5, and 6+
JavaScript
TypeScript
NodeJs
Go Programming
R Programming
Smalltalk Programming
Ruby Programming
KnockoutJs

R Programming
Smalltalk Programming
Ruby Programming
KnockoutJs
RequireJs
BackboneJs
Python
Vue.Js
LimeJS framework
Swift
D3.Js
Ext.Js
ReactJs
Vaadin

ARTICLES

Articles ▼

Anil Singh

Home | About Me | Contact Me | Books | Ng-eBook

^