JavaScript Vs Java interview Q&A

Posted on September 13, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

The client-side JavaScript based MVW (Model View Whatever) frameworks like **AngularJS**, **Backbone**, **CanJS**, **Ember**, etc have become mainstream and replacing server side Java based frameworks like JSF, Struts, Spring MVC. So, time to get a better handle on JavaScript and market yourself as a full stack Java developer.

Q1. What is the difference between Java and JavaScript?
A1. Don't be fooled by the term Java in both. Both are quite different technologies. The key differences can be summarized as follows:

1) JavaScript variables are dynamically typed, whereas the Java variables are statically typed.

```
1 var myVar1 = "Hello";          //string type
2 var myVar2 = 5;                //number type
3 var myVar3 = new Object( );    //empty object type
4 var myVar4 = {};               //empty object type
5
```

2) In JavaScript **properties and methods are dynamically added**, whereas Java uses a template called a class. The ***myVar3*** empty object dynamically adds properties and a method.

```
1 var myVar3 = new Object( );
2 myVar3.firstName = "John";    // add a property t
3 myVar3.lastName = "Samuel";   // add a property t
4 // add a method
5 myVar3.someFunction = function( ) {
6    document.write(this.firstName + "   " +  this.l
7 }
```

3) JavaScript **function can take variable arguments**. You can call the function shown below as myFunction( ), myFunction(20), or myFunction(20,5).

```
1 function myFunction( value ) {
2    //…. do something here
3 }
```

JavaScript has an implicit keyword known as the "**arguments**", which holds all the passed arguments. It also has a "length" property as in arguments.length to display the number of arguments. Technically an "arguments" is not an array as it does not have the methods like push, pop, or split that an array has. Here is an example.

```
1 myFunction(5,10,15,20);
```

```
1 function myFunction(value) {
2        //value is 5;
3        //arguments[0] is 5
4        //arguments[1] is 10
5        //arguments[2] is 15
6        //arguments[3] is 20
7        //arguments.length is 4
8 }
```

4) JavaScript objects are basically like **name/value pairs stored in a HashMap** with string key and object values. For example, a JavaScript object is represented in **JSON** style as shown below.

16 Technical Key Areas

```
var personObj = {
          firstName: "John",
          lastName: "Smith",
          age: 25,
          printFullName: function() {  //function
             document.write(this.firstName + "  "
          } ,

          printAge: function () {  //function
             document.write("My age is: " +  this
        }
    }

```

You can invoke the methods as shown below

```
personObj.printFullName();

personObj.printAge();

```

5) JavaScript **functions are objects as well**. Like objects, the functions can be **stored to a variable**, **passed as arguments**, **nested within each other**, etc. In the above example, nameless functions are attached to variables “printFullName” and “printAge” and invoked via these variables. A function that is attached to an object via a variable is known as a “**method**“. So, printFullName and printAge are methods.

In the example shown below, technically, what is done with the “add” and “sum” functions is that we have created a new function object and attached them to the variables “add” and sum. As you can see in the example below, the “add” variable is assigned to variable “demo”, and the function is invoked via demo(2,5) within the “sum” function.

```
function add(val1, val2) {
    var result = val1 + val2;
    alert("The result is:" + result);
    return result;
}

var demo = add;

function sum() {
   var output = demo(5, 2);
}
```

⊞ Scalability (4)
⊞ SDLC (6)
⊞ Security (13)
⊞ Transaction Managen

80+ step by step Java Tutorials

open all | close all
⊞ Setting up Tutorial (6)
⊞ Tutorial - Diagnosis (2
⊞ Akka Tutorial (9)
⊞ Core Java Tutorials (2
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all
⊞ Can you write code? (
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

Now the above **temp.js** under **tutorial/js** folder can be invoked from an HTML file under tutorial/html as shown below.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Tra
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/ht
5
6  <script language="javascript" type="text/javascr
7  </script>
8
9  <title>Insert title here</title>
10 </head>
11 <body>
12     <form id="evaluate1">
13        <input type="button" value="evaluate"  on
14        <input type="button" value="evaluate2" on
15        <input type="button" value="evaluate3" on
16     </form>
17 </body>
18 </html>
19
```

6) Now we know that functions in JavaScript are objects, and can be passed around. Every function in JavaScript also has a number of attached (or implicit) methods including **toString( )**, **call( )**, and **apply( )**.

a) **toString()** implicit method example

```
1  function add(val1, val2) {
2      var result = val1 + val2;
3      alert("Result is:" + result);
4      return result;
5  }
6
7  var printAdd = add.toString(); //converts the "a
8
9  function demo() {
10     alert(printAdd); //alerts the whole source c
11 }
12
13
```

b) In JavaScript, functions can be invoked 5 different ways.

1) function_name(param1, param2, etc); //"this" refers to global object like window.
2) obj1.function_name(param1,param2,etc); //"this" refers to obj1.

3) new Object(); // The constructor.
4) function_name.call(objRef, param1); // function object implicit method
5) function_name.apply(objRef, params[parama1,param2, etc]); // function object implicit method

So, why use function_name.**call**(…) or function_name.**apply**(…) as opposed to just function_name(…)? Let's look at this with some examples.

```
1  var x = 1;            //global variable x;
2
3  var obj1 = {x:3};    //obj1 variable x
4  var obj2 = {x:9};    //obj2 variable x
5
6  function function_name(message) {
7     alert(message + this.x) ;
8  }
9
10 function_name("The number is ");              //
11
12 //the first argument is the obj reference on whi
13 //the second argument is the argument to the fun
14
15 function_name.call(obj1, "The number is ");   //
16 function_name.call(obj2, "The number is ");   //
17
18 //the first argument is the obj reference on whi
19 //the second argument is the argument to the fun
20
21 function_name.apply(obj1, ["The number is "]);
22 function_name.apply(obj2, ["The number is "]);
23
```

The purpose is of call and apply methods are to invoke the function for any object without being bound to an instance of the this object. In the above example, the **this** object is the global object with the x value of 1. In a function called directly without an explicit owner object, like function_name(), causes the value of this to be the default object (**window** in the browser). The call and apply methods allow you to pass your own object to be used as the "**this**" reference. In the above example, the **obj1** and **obj2** were used as "**this**" reference.

7) JavaScript **variables need to be treated like records stored in a HasMap and referenced by name**, and not by memory address or pass-by-reference as in Java. The following code snippet demonstrates this.

```
var x = function () { alert("X"); }
var y = x;
x = function () { alert("Y"); };
y();           // alerts "X" and NOT "Y"
x();

```

8) Java does not support closure till version 8. A **closure** is a function plus a binding environment. closures can be passed downwards (as parameters) or returned upwards (as return values). This allows the function to refer to variables of its environment, even if the surrounding code is no longer active. JavaScript supports closure.

In JavaScript a closure is created every time you create a function within a function. When using a closure, you will have access to all the variables in the enclosing (i.e. the parent) function.

```
var calculate = function(x) {
     var myconst = 2;
     return function(y) {
         return x + y + myconst;    // has visib
     };
}

var plus5 = calculate(5);          //plus5 is now
alert(plus5(3));                   //returns 10
alert(plus5(7));                   //returns 14
alert(plus5(10));                  //returns 17

```

Q2. Does JSE provide support for JavaScript?
A2. Yes. Until Java SE 7, JDKs shipped with a JavaScript scripting engine based on Mozilla Rhino. Java SE 8 will instead ship with a new engine called **Oracle Nashorn**, which has a **bin/jjs** command-line tool to get started with JavaScript. For example, learnjs.js

```
var hello = function() {
  print("Start learning JavaScript!");
};

hello();


```

```
$ jjs learnjs.js
```

Q3. Java has packages to organize your code. How would you organize your code in JavaScript?
A3. The concept of namespaces does not exist in JavaScript. To add insult to injury, everything you create in JavaScript is by default global. Now obviously, this is a recipe for disaster. In JavaScript, you can use a number of techniques to modularize your code.

1) Nested objects acting as name spaces

```
var MYAPPLICATION = {
    MODEL: {
        product: function (cost) {
                    this.cost = cost;
                    this.getCost = function(){
                         return this.cost;
                     };
                }
    },
    LOGIC: {
        calculateGST: function (baseCost) {
            return baseCost * 1.10;
        },
        performCalc: function () {
            var p = new MYAPPLICATION.MODEL.prod
            alert(this.calculateGST(p.getCost())
        }
    }
}
```

The above pattern is fairly simple to avoid name collisions, but useful only for smaller projects.

2) Creating a general purpose namespace method that allow us to create namespaces.

3) Using a JavaScript library like AMD (Asynchronous Model Definition) API and **RequireJS** to modularize your JavaScript files.

Option 3 is recommended. Beware that JavaScript is very powerful, but not properly applying best practices and

patterns can lead to maintenance nightmare. Use of proper name spacing pattern or API like AMD is very important.

You may also like

1) Top 30 FAQ JavaScript Interview Questions & Answers.

2. More AngularJS Interview Questions & Answers.

Popular Member Posts

01: ♦ 15+ Hibernate basics Q1 – Q7 interview questions & answers

1,825 views

♦ 11 Spring boot interview questions & answers

976 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

866 views

01: ♦ 15 Beginner level Java multi-threading interview Q&A

551 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

445 views

18 Java scenarios based interview Questions and Answers

422 views

♦ 7 Java debugging interview questions & answers

347 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

296 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

294 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

294 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. Published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. join my LinkedIn Group to get notified of special offers. User & expert **reviews**.

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. Published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. join my LinkedIn Group to get notified of special offers. User & expert **reviews**.

‹ XML basics interview Q&A JavaScript mistakes interview Q&A ›

Posted in FAQ JEE Job Interview Q&A Essentials, JavaScript Vs Java Q&A

Tags: Java/JEE FAQs

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

E-mail *

Website

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?

☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

 ↑