Day-1 : Introduction

15 June 2017
14:54

Two types of applications:

RIA (Ritch Internet Applications)/ Single Page Applications
- UI layer is created in JavaScript

Classical Web applications
- Every thing including UI layer will be in server side, on each request a separate HTML along with CSS and Javascript with dynamic data will be generated.

Classification of JavaScript ?
- Procedural
- Object oriented
- Functional

Note : JavaScript is OO and Functional

Functions are not just programming construct, but that can be treated as data. Whatever you can do with object, can be done by function in javascript.

Ref for functional programming : Execution in the kingdom of nouns : Stevey's Blog

Difference and similarity between object and function.

Object	Function
var obj={} undefined typeof obj; "object" Object is created with object expression	var fn=function(){} undefined typeof fn "function" Function is created with function expression

Object can be created as: using Object with capital O
var obj=new Object()
typeof Obj
"object"
 	
If you display the obj, you will find that it is an object.
obj
>Object
If you display obj.toString(), it will say Object is an object string as "[object Object]"
 	
There is another way of creating object.
Using object.create()
 	
Ref :
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Object/create
 	
Go through this
Assigment
	We can create the function same as object as: using Function with capital F
	var fn=new Function()
	typeof fn
	"function"
	If you display the function this time, it will show it is an anonymous function, because this function consist nothing.
	fn
	function anonymous() {
	
	}
	
	fn.toString()
	"function anonymous() {
	
	}"
	
	Why this kind of function is created in javascript?
	Because you can provide the body of the function at the time of the creation.
	
	var fn=new Function("console.log('body of the function');")
	undefined
	fn()
	body of the function
	undefined
	fn.toLocaleString()
	"function anonymous() {
	console.log('body of the function');
	}"
	
	Is it possible to pass argument in the function at the time of creation.
	var fn=new Function("x","y","return x+y;")

	undefined
	fn(10,20);
	30
	fn.toString()
	"function anonymous(x,y
	/*``*/) {
	return x+y;
	}"
	
	If you provide more than one arguments in function. Then last argument is the return statement. The first two will be the arguments.

Object can have Attributes. Exm: var obj={} undefined obj.id=100; 100 obj.id 100	Functions can have Attributes. Exm: var fn=function(){} undefined fn.id=100; 100 fn.id 100

Object can have methods.
Exm:
obj.display=function(){
console.log('My id is : '+this.id);
}
 	
obj.display();
My id is : 100
	Function can also have another function
	Exm:
	fn.display=function(){
	console.log('My id is : '+this.id);
	}
	
	fn.display();
	My id is : 100

We can pass object as an argument to a function.
function fx(y){
console.log('type of y :',typeof y);
}
 	
fx(obj);
 	
type of y : object
	We can pass function as an argument to a function.
	function fx(y){
	console.log('type of y :',typeof y);
	}
	
	fx(fn);
	
	type of y : function

We can return the object as the return value from function
function fn(){
return {}
}
undefined
var result=fn();
undefined
result
Object {}
typeof result;
"object"
	We can return the function as return value from another funtion.
	function getFunction(){
	return function(){
	console.log('I am the function returned');
	}
	}
	undefined
	var result=getFunction();
	undefined
	result
	function (){
	console.log('I am the function returned');
	}
	result()
	I am the function returned

	Practical use case of function returning as a function value
	function getAdder(){
	return function(x,y){
	return x+y;
	}
	}
	
	var adder=getAdder();
	adder(10,20);
	30
	
	You do not need to get the reference of getAdder() in adder variable. In practical situation, this can be directly invoked as:
	getAdder()(100,200)

	300
	--- 	
	function getAdder(x){
	return function(y){
	return x+y;
	}
	}
	
	var adderFor500=getAdder(500);
	adderFor500(100);
	600
	adderFor500(200);
	700
	
	Here in this code we are fixing one argument. Like in above example we are getting different adders for 500.

JavaScript is a dynamic language. It means that once the object is created, we can add, remove attributes and methods at run time.
This is not possible in java, and .net.
Example :
If we create an Employee object having some fields and methods in java language.
We can not add or remove attributes and methods at run time in and from Employee object.
But this is possible in JavaScript.
 	
var employee={};
employee.id=101;
employee.salary=10000;
employee.name="Rajesh";
employee
Object {id: 101, salary: 10000, name: "Rajesh"}
employee.display=function(){
console.log('id = ',this.id, ' name = ',this.name, ' salary =',this.salary);
}
employee.display();
id = 101 name = Rajesh salary = 10000
 	
--- 	
Remove
delete employee.id;
employee.display();
id = undefined name = Rajesh salary = 10000	

Java Script has support only of 6 types, out of that you can use only three types to represent data.

typeof 123
"number"
typeof "abc"
"string"
typeof true
"boolean"
typeof {}
"object"
typeof function(){}
"function"
typeof null
"object"
typeof []
"object"
typeof undefined
"undefined"

Out of above mentioned types, only number, string and boolean is used to represent data. Function is used to show the behaviour and object is used to encapsulate the behaviour and data. Undefined denotes the absence of something.

In java script functions are as objects.
"function objects"

Java Script is a loosly typed, dynamic and functional language.

There are some problems with loosly typed language.

```
function add(x,y){
return x+y;
}
add(10,20);
30
add(10,20,30);
30
add("10","20");
"1020"
add(10,"20");
"1020"
var ob1={};
undefined
var ob2={};
undefined
ob1.id=10;
10
ob2.id=20;
20
add(ob1,ob2);
"[object Object][object Object]"
add([10,20,30],[40,50]);
"10,20,3040,50"
```

As we do not provide the type with arguments, we can pass any value to add method. This is the problem with loosly typed language. But this if you explore it more, it will become the advantage as well. We need to exploited the loosly typed nature.
Consider the example of Jquery

```
$(document) -- dom
$(function(){}) ==> $(document.ready().function(){})
$($)
$([10,20,30])
```