Javascript

Thierry Sans

Introduction

Javascript (aka Ecmascript)

Scripting language (dynamically and weakly typed)

Object-oriented (prototype based)

Functional (first-class functions)

Reflexive (eval method) … although it is not a good thing.

- It has absolutely nothing to do with Java

"Java is to Javascript is what a car is to a carpet"

The (from-the-past) alternatives to JavaScript

Third-party plugins are were used to palliate to the lacks of Javascript:

- Java applets
- Flash
- Silverlight
- and so on

Why Javascript is ahead in the game now?

Open and standard (multi platforms)

Come for free on many browsers/platforms

Javascript engines are getting “incredibly” faster

HTML 5

Javascript is getting out of the browser (Node.js)

Elements of Syntax

Comments

```
// This is a comment
```

```
/* This is
another one */
```

Debugging

```
console.log(“Houston, there is a problem”);
```

Variables

```
var name = "Alice";
var age = 28;
```

IF statement

```
if ((age<20 && name="Alice")||(age>=20)){
   age = age + 1;
}
else{
   name = "Alice" + "Alicson";
}
```

`else` statement is optional

Look at the operator switch as well

Loops

```
var i = 0;
while (i<100){
    console.log(i++);
}

for(var i=0;i<100;i++){
  console.log(i);
}
```

First-class functions

```
function getAge(){
    return 28;
};

getAge();
```

or

```
var getAge = function(){
    return  28;
};

getAge();
```

Anonymous functions will be very useful for object methods and callback methods

Prototype-Based Object-Oriented

```
// defining a constructor
function Person(name){
    this.name = name;
}

// adding a method
Person.prototype.getName = function(){
    return(this.name);
};

// creating an object
var p = new Person('Mariam');
console.log(p.getName());
console.log(p.constructor.name);
console.log(p instanceof Person);
```

Inheritance

```
// defining a constructor calling a super class
function Employee(name,title){
    this.title = title;
    Person.call(this, name);
}

// setting up the inheritance
Employee.prototype = new Person();

// fixing the constructor
Employee.prototype.constructor = Employee;

// creating an object
var e = new Employee('Mariam','CEO');
console.log(e.getName());
console.log(e.title);
console.log(e.constructor.name);
console.log(e instanceof Employee);
console.log(e instanceof Person);
```

Data Structures

Arrays

```
var myArray = new Array();
myArray[0] = "JavaScript";
myArray[1] = "is";
myArray[2] = "fun";
```

Or

```
var myArray = new Array ("Javascript","is","fun");
```

Or

```
var myArray = ["Javascript","is","fun"];
```

Associative Arrays (aka Hashtables or Dictionaries)

```
var myDict = new Object();
myDict[“first”] = "JavaScript";
myDict[“second”] = "is";
myDict[“third”] = "fun";
```

Or

```
var myDict = {};
myDict.first = "JavaScript";
myDict.second = "is";
myDict.third = "fun";
```

Or

```
var myDict = {first:,“Javascript”,
              second:”is”,
              third:”fun”}
```

Iterate through collections

```
var person={
   fname:"Alice",
   lname:”Alicson",
   age:30
};

for (var x in person){
   console.log(person[x] + " ");
}
```