

Javascript

Thierry Sans

Introduction

Javascript (aka Ecmascript)

- Scripting language (dynamically and weakly typed)
- Object-oriented (first prototype-based and now class-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 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 */
```

Debug message on the console

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


Constants and Variables - var vs let

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

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

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

new since es6

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

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

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

First-class functions

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

```
getAge();
```

or

```
let 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  
let 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
let 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

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

Or

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

Or

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

Associative Arrays (aka Hashtables or Dictionaries)

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

Or

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

Or

```
let myDict = {first: "Javascript",  
               second: "is",  
               third: "fun"}
```

Iterate through collections

```
let person={  
  fname: "Alice",  
  lname: "Alicson",  
  age:30  
};  
  
for (let x in person) {  
  console.log(person[x] + " ");  
}
```