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;
```

old way
(should not be used)

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

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

new since es6

https://hackernoon.com/why-you-shouldnt-use-var-anymore-f109a58b9b70

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

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

getAge();
```

or

```
const getAge = () => 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
const p = new Person('Mariam');
console.log(p.getName());
console.log(p.constructor.name);
console.log(p instanceof Person);
```

Prototype-based 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
const 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);
```

Class-based Object-Oriented

```
// Defining a class
class Person(){

  constructor(name){
    this.name = name;
  }

  get name(){
    return this.name;
  }

}

// creating an object
const p = new Person('Mariam');
console.log(p.name());
```

Class-based Inheritance

```
// Defining a class that inherits
class Employee extend Person{

  constructor(){
    Person.call(this, name);
    this.title = title;
  }

}

// creating an object
const e = new Employee('Mariam','CEO');
console.log(e.name());
console.log(e.title);
```

Data Structures

Arrays

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

Or

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

Or

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

Iterate through arrays

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


for (let i=0; i<myArray.length; i++){
   console.log(myArray[i]);
}

for (let x of myArray){
   console.log(x);
}

myArray.forEach(function(x){
   console.log(x);
})
```

Associative Arrays (aka Hashtables or Dictionaries)

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

Or

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

Or

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

Iterate through associative arrays

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

for (let k in person){...}

for (let k of Object.keys(person)){...}

for (let v of Object.values(person)){...}

for (const [k, v] of Object.entries(person)) {...}
```