Universidad Nacional de Quilmes

Mundo Javascript

CONSTRUCCIÓN DE INTERFACES DE USUARIO

Consideraciones

- Esta clase es una **presentación** de JavaScript, **no es** una guía exhaustiva.
- Todo nuevo lenguaje se aprende codeando.
- Al final de la presentación hay links útiles.
- Y tienen prácticas y los TPs para codear.

Acerca de Javascript

- Lenguaje Interpretado de propósito general
- Multi Plataforma
- Basado en prototipos
- Multi-paradigma
  - Orientado a Objetos
  - Imperativo
  - Funcional
- Tipado débil y dinámico

Historia

- Fue desarrollado originalmente por Brendan Eich (Netscape) en 1995 en plena “guerra de los navegadores”
- Internet Explorer y Netscape Navigator se disputaban el floreciente mercado de internet.
- Se llamó originalmente *Mocha* (luego *LiveScript*)
- **JavasScript** fue una elección comercial dado que Netscape acababa de implementar *snippets* de Java dentro de su navegador (un gran avance para internet).
- Explorer tenía su contraparte llamada **JScript** (elección comercial para generar confusión con JavaScript)

ECMAScript

- Ante la puja entre *JScript* y *JavaScript*, en 1996 se definió un estándar para lenguajes de scripting en browsers llamado **ECMAScript**.
- ***E**uropean **C**omputer **M**anufacturers **A**ssociation*
- Todos los navegadores debían interpretar el estándar
- Implementaciones de ECMAScript
  - JavaScript
  - JScript
  - ActionScript
- Actualmente JavaScript ganó la puja a tal nivel que mucha gente cree que ECMAScript y JavaScript son lo mismo

Por qué es importante JS

- Porque funciona en todos lados
  - Web ⇒ Manipular HTML/CSS vía DOM
  - Browsers ⇒ Plugins de Chrome, Firefox, Opera, ...
  - Server side ⇒ NodeJS (ej: APIs con Express)
  - Apps Desktop ⇒ Atom
  - Apps Mobile ⇒ React Native, Ionic, PhoneGap, ...
  - Otros ⇒
    - Google “office” Scripts
    - Amazon Lambdas
    - Postman

ECMAScript++

- Existen varias versiones de ES
  - ES4 y anteriores (Deprecadas)
  - ES5
    - Actualmente soportada por todos los browsers
    - La que realmente se ejecuta
  - ES6 / ES2015
    - Salto cualitativo con respecto a ES5
    - La que nos gusta escribir
  - ES2016, ES2017, ES2018, ES2019, ...
    - Mejoras sobre ES2015

Sintaxis ES5 - Variables

```
// Soy un comentario de línea
var str = "Soy un string";
var num = 42;

/* Comentario Multilínea:
JS tiene tipado débil y dinámico */
var a, b;
a = 1;
b = "foo" // El ; es opcional pero recomendado
a = "ahora soy un string";

// Caracteres Unicode en vars
var Numero_Visitas, temp99, _nombre, $nombre;
```

ES5 - Tipos de Datos

```
// Boolean
var t = true; var f = false;

// Number
var int = 42; var float = 3.14159;
var nan = NaN; var inf = Infinity;

// String
var str = "soy un string"; var str2 = 'yo también';

// Special values
var nul = null; var not_defined = undefined;

// Object
var o = { key: "value" };

// Array
var arr = [1, "dos", 3.14, null, undefined, NaN];
```

ES5 - Igualdades

```
// Asignación
var foo = "un solo igual asigna valor a variable";


// Igualdad débil (compara valor)
42 == 42   // true
42 == "42" // true


// Igualdad fuerte (compara valor y tipo)
42 === 42   // true
42 === "42" // false
```

ES5 - Valores de verdad

```
Boolean(true)   // true
Boolean(false)  // false

Boolean(0)      // false
Boolean(-3.14)  // true
Boolean(NaN)    // false

Boolean("")     // false
Boolean("coso") // true

Boolean(null)      // false
Boolean(undefined) // false

Boolean({})                // true
Boolean({ key: "value" }) // true

Boolean([])        // true
Boolean([1, 2, 3]) // true
```

ES5 - Operadores Booleanos

```
// And
true && true  // true
true && false // false

// Or
true  || false  // true
false || false  // false

// Not
!true   // false
!0      // true
!1      // false
```

```
// "Short" Boolean(val)
!!0       // false
!!1       // true
!!""      // false
!!"coso" // true

// "Problemas":
1 && true   // true
0 && true   // 0
false || 42 // 42
undefined || false // false
undefined && true  // undefined
```

ES5 - IF

```
var num = 0;

if(num) { num = 1; } else { num = 0; }

if(num > 0) {
  num = 3;
} else if(num == 0) {
  num = -2;
} else {
  num = 0;
}

// Ternary operator (short if)
var val = (num) ? num : 2;

// "elvis operator" (en kotlin: num ?: 0)
var newNumber = num || 2;
```

ES5 - Switch/For

```
var days;
var month = "Febrero";
switch (month) {
  case "Febrero":
    days = 28;
    break;
  case "Enero":
  case "Marzo":
  case "Mayo":
  case "Julio":
  case "Octubre":
  case "Diciembre":
   days = 31;
   break;
  default = 30;
}
```

```
var pairs = [];
var n = [1, 2, 3];
var l = n.length;
for(i = 0; i < l; i++) {
  pairs[i] = n[i] * 2;
}
// pairs = [2, 4, 6]

for(var item of pairs) {
  n[i] = item * 3;
}
// n = [6, 12, 18]

n.forEach(function(item){
  console.log(item);
});
```

ES5 - Funciones (I)

```
function sum(a, b) {
  return a + b;
}
var result = sum(1, 2) // result == 3

// En ES las funciones son valores.
// O sea, es posible guardar una función en una variable
var sum2 = function(a, b) {
  return a + b;
}

var result2 = sum2(3, 4)
// result == 7
```

ES5 - Funciones (II)

```
// Y como las funciones son valores
// se pueden pasar por parámetro
var mulRes = function(sum, a, b, c) {
  return sum(a, b) * c;
}
var sum3 = function(a, b) { return a + b + 1; }
var result3 = mulRes(sum3, 5, 10, 2);
// result3 == 32


// Pero también se pueden definir Funciones Anónimas
// Son funciones que no se guardan, sino que se
// utilizan solo en el momento
var result4 = mulRes(function(a, b) {
  return 2*a + b;
}, 5, 10, 2);
// result4 == 40
```

ES5 - Objects

```
var obj = {
  num: 42,
  name: "el sentido de la vida, el universo ...",
  description: function() {
    return "soy " + this.num + ", " + this.name
  }
};
obj.num  // 42
obj.name // "el sentido de la vida, el universo ..."
obj.description   // function
obj.description() // "soy 42, el sentido de la vida, el universo ..."

// Es posible agregar propiedades "desde afuera"
obj.foo = 123
obj.foo // 123
```

ES5 - “Classes”

```
function Person(name, house) {
  this.name = name;
  this.house = house;
  this.description = function() {
    return name + " de la casa " + house;
  };
}

var jon = new Person("Jon", "Stark");
var tyrion = new Person("Tyrion", "Lannister");

jon.description()    // "Jon de la casa Stark"
tyrion.description() // "Tyrion de la casa Lannister"
```

ES5 - Funciones de Arrays

```
var list = [1, 2, 3];

list.filter(function(e) { return e > 2 }); // [3]
list.map(function(e) { return e * 2 });    // [2, 4, 6]
list.every(function(e) { return e > 2 });  // false
list.find(function(e) { return e > 3});    // undefined

list[0] // 1
list[list.length - 1] // 3

[3, 2, 1].includes(2)           // true
[3, 2, 1].sort()                // [1, 2, 3]
[3, 2, 1].concat([4, 5, 6])     // [3, 2, 1, 4, 5, 6]
[1, 2, 3, 4, 5, 6].slice(2, 4) // [3, 4]

[1, 2, [3, 4, [5, 6]]].flat()   // [1, 2, 3, 4, [5, 6]]
[1, 2, [3, 4, [5, 6]]].flat(10) // [1, 2, 3, 4, 5, 6]
```

ES6 / ES2015

- ES2015 trae infinidad de mejoras
  - Scope de variables, constantes
  - Arrow functions
  - Deconstrucción
  - Declaración de clases
  - Etc …
- Pero por supuesto, los browsers aún no soportan nativamente ES2015
- Y para cuando la soporten, ya habrá versiones nuevas
- Solución: convertir ES5 en ES6

ES6 - var to let & const

```
// var queda "deprecada", se puede usar pero no se recomienda
// se utiliza let para variables
let foo = "bar"

// y const para constantes
const PI = 3.14159

// a diferencia de var, let pertenece al scope donde se definió
for(var i = 0; i < 5; i++) { /* hago cosas */ }
for(let j = 0; j < 5; j++) { /* hago cosas */ }
console.log(i); // 5
console.log(j); // Reference Error: j is not defined

var x = 0;
let y = 0;
if(true) { var x = 1; let y = 1; }
console.log("x =", x) // x = 1
console.log("y =", y) // y = 0
```

ES6 - Arrow Functions

```
// ---------- ES5 ----------
function num5() { return 5; }
function obj5(name) { return { foo: name }; }
function sum5(a, b) {
    return a + b;
}
function large5() {
    var x = 5;
    return x;
}

// ---------- ES6 ----------
const num6 = () => 6;
const obj6 = name => ({ foo: name });
const sum6 = (a, b) => a + b;
const large6 = () => {
    let x = 6;
    return x;
}
```

ES6 - Deconstrucción ✊

```es5
// ----- ES5 -----
var obj1 = {
  foo: 'foo',
  bar: 'bar'
};
var foo = obj1.foo;
var bar = obj1.bar;

var list = [1, 2, 3, 4, 5];
var l1 = list[0];
var l2 = list[1];

var a = 'A';
var b = 'B';
var obj5 = { a: a, b: b };
```

```es6
// ----- ES6 -----
let obj1 = {
  foo: 'foo',
  bar: 'bar'
};
let { foo, bar } = obj1;

let list = [1, 2, 3, 4, 5];
let [l1, l2] = list;

let a = 'A';
let b = 'B';
let obj6 = { a, b };
```

ES6 - Spread operator ... (I)

```
// ----- Objects -----
let tony = {
  name: 'tony',
  age: 50,
  avenger: true,
  planet: 'earth'
};
let steve = { ...tony, name: 'steve', age: 87 };
let thor = {
  ...tony,
  name: 'thor',
  age: 1200,
  planet: 'asgard'
};
// {avenger: true, name: 'thor', age: 1200, planet: 'asgard'}

let { planet, ...rest } = thor;
// planet = 'earth, rest = {name:'thor', age:1200, ...}
```

ES6 - Spread operator ... (II)

```
// ----- Arrays -----
let avengers = ['tony', 'steve', 'thor'];

let assemble = [...avengers, 'strange', 'peter'];
// ['tony', 'steve', 'thor', 'strange', 'peter']

let more = ['bruce', 'natasha'];

let all  = [...assemble, ...more];
// ['tony', 'steve', 'thor', 'strange', 'peter', 'bruce',
'natasha']

let [first, second, ...rest] = all;
// first = 'tony', second = 'steve'
// rest = ['thor', 'strange', 'peter', 'bruce', 'natasha']
```

ES6 - Spread operator ... (III)

```
// ----- Functions List -----
function spreadList(...xs) {
  console.log(xs);
  return xs;
}
let [x1, ...rl] = spreadList(1, 2, 3, 4, 5); // [1, 2, 3, 4, 5]
// x1 = 1, r = [2, 3, 4, 5]

// ----- Functions Object -----
function spreadObj({ name, age, ...obj}) {
    return {...obj, name: name.toUpperCase(), age: age * 2}
}

let thor = { name: 'thor', age: 1200, planet: 'asgard' }
let { age, ...ro } = spreadObj(thor);
// age = 2400, ro = { name: 'THOR', planet: 'asgard' }
```

ES6 - Módulos

```
// ---------- file: export.js ----------
var toExport = { a: 1, b: 2 };

// ***** ES5 *****
module.exports = toExport;

// ***** ES6 *****
export default toExport;
export const extraExport1 = 1;
export const extraExport2 = 2;


// ---------- file: import.js ----------
// ***** ES5 *****
var toExport = require('./export');

// ***** ES6 *****
import toExport from './export';
import { extraExport1, extraExport2 } from './export';
```

ES6 - Classes

```
class User {
  constructor(name, age) {
  this.name = name;
  this.age = age;
}
  sayMyName() { return this.name; }
  static sayHi() {
    return "Hello, there!";
  }
}
class Supervisor extends User {
  constructor(name, age, job) {
    super(name, age);
    this.job = job
  }
  sayMyName() {
    return `${super.sayMyName()} i'm ${this.job}`;
  }
}
```

```
const jessie = new User('Jessie', 20);
const walter = new Supervisor(
  'Heisenberg', 50, 'the danger!'
);

jessie.sayMyName()
// Jessie

walter.sayMyName()
// Heisenberg i'm the danger!

Supervisor.sayHi()
// Hello, there!
```

ES5 ⇒ ES6

- Los Browsers interpretan ES5
- Pero ES5 es un asco
- Preferimos escribir en ES6
- ¿Entonces?
  - Vamos a usar Babel
  - Es un transpilador que transforma ES6 en ES5
  - Ejemplo: https://babeljs.io/repl
  - Eventualmente para nosotros va a ser transparente

Links Útiles

- https://developer.mozilla.org/es/docs/Web/JavaScript/Guide
- https://devhints.io/es6
- https://www.w3schools.com/js/
- https://babeljs.io
- http://sporto.github.io/blog/2013/02/22/a-plain-english-guide-to-javascript-prototypes/