




Introducción JavaScript

Extraído de Apuntes de la Universidad Politécnica
de Valencia

Profesor: Javier Esparza Peidro



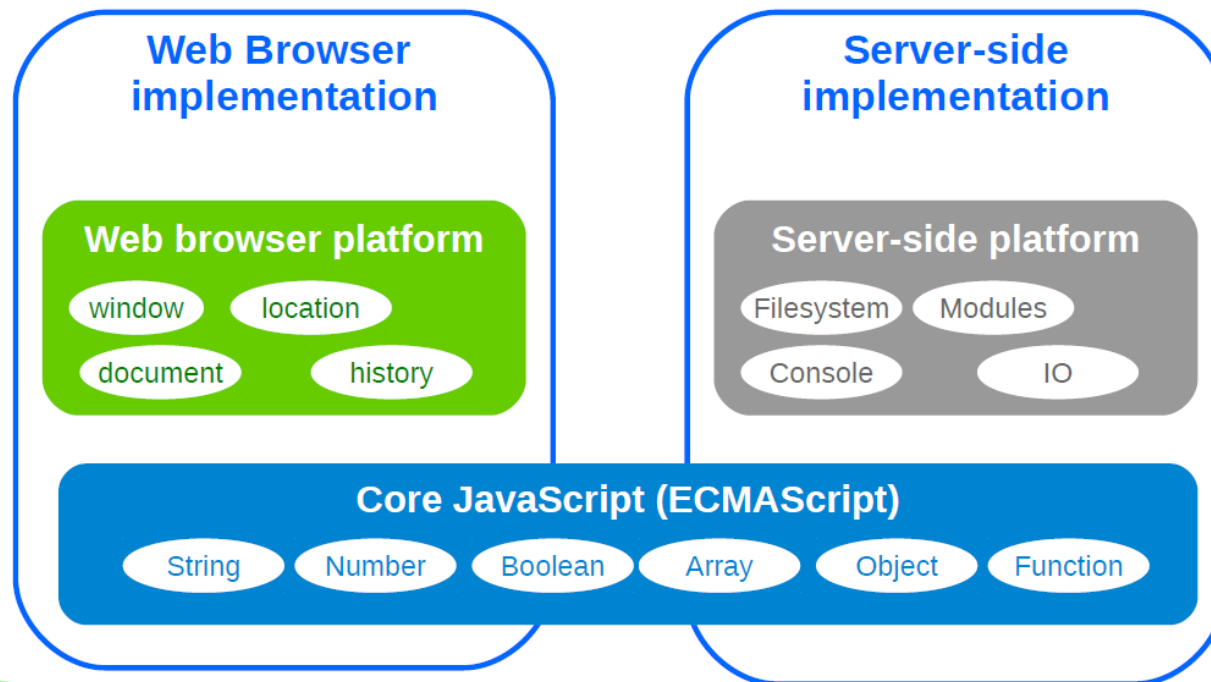
JavaScript

- ▶ EcmaScript 
 - Es el estándar de JavaScript (actual versión 9)
 - JavaScript sólo es el nombre comercial de ese estándar
- ▶ Mozilla Developer Network 
- ▶ W3Schools 

JavaScript

- CORE → **la parte estandarizada** de JAVASCRIPT (constructores básicos, tipos datos, ...)
- PLATAFORMA → **peculiaridades** de cada plataforma (**librerías** u objetos.
 - Ej: window en los navegadores, o document,...
 - Ej: FileSystem en un servidor

JavaScript = lenguaje (core) + plataforma






JavaScript

- ▶ PRIMITIVOS
- ▶ OBJETOS

JavaScript: tipos PRIMITIVOS

Tipos de datos

Primitivos

- ▶ Números (+, -, *, /, %): 0, 3.14, Infinity, NaN 
- ▶ Strings (.length, +): "ho", 'la', str.length, 'ho' + 'la' 
- ▶ Lógicos (&&, ||, !): true/false 
- ▶ null, undefined: ausencia de valor (voluntaria vs profunda)

Variables

- ▶ Se declaran antes de usarse: con var (sin tipo)
- ▶ Si no se inicializan, su valor es undefined

JavaScript: tipos OBJETOS

Objetos son:

- **Diccionarios:** **colección** no ordenada de pares nombre–valor
 - Nombre es un **string**, el valor es **cualquier cosa**.
- **Creación:**
 - `var obj = {x:0, y:'a'};`
 - `var obj = new Object();`
 - `var obj = {};` //objeto vacío
- **Acceder a propiedades:**
 - Se **pueden crear propiedades dinámicas a un objeto al intentar acceder a ellas si no existen.**
 - Ej: `var obj = {};`
`obj.x = 10;` //**NO DA ERROR**, crea la propiedad x con valor 10
 - `obj.x = 10; obj['x'] = 10;`

JavaScript: tipos OBJETOS

- **Comprobar si un objeto tiene una propiedad:**
 - 1) intentar acceder a la propiedad: `obj.x`
 - 2) usar el operador “in”. Ejemplo: `‘x’ in obj`
 - 3) usar el método “hasOwnProperty(propiedad)” que tienen todos los objetos. Ej: `obj.hasOwnProperty(‘x’)`
- **Todos los objetos javascript tienen ya un conjunto de operaciones predefinidas:**
 - `.hasOwnProperty()`
 - `.toString()`
 - `.valueOf()`
 - ...

JavaScript: tipos OBJETOS más potentes

ARRAYS O VECTORES son:

- **Colección heterogénea ordenada** de valores
- Son objetos “especializados” (heredan todo lo dicho)
 - Por tanto, también son diccionarios.

➤ Creación:

```
var v= [1, 'a', {a:1,b:2}];
```

```
var v= new Array();
```

- Acceder a los elementos: `array [<indice>]`
- El tamaño del array es dinámico: `array.length`
- Métodos disponibles: `.push()`, `.pop()`, `.shift()`, `.unshift()`,
`.reverse()`, `.sort()`, `.splice()`, `.concat()`, `.join()`, `.slice()`

JavaScript: OBJETO GLOBAL

El objeto global

- ▶ Siempre existe un objeto global, accesible desde cualquier sitio (en el navegador web se llama ***window***)
- ▶ Se puede acceder a cualquiera de sus propiedades sin referenciar el objeto (crea un contexto global)
- ▶ El objeto global contiene:
 - ▶ Propiedades: *undefined*, *Infinity*, *NaN*, ...
 - ▶ Funciones: *isNaN()*, *parseInt()*, *eval()*, ...
 - ▶ Constructores: *Date()*, *String()*, *Object()*, *Array()*, ...
 - ▶ Otros objetos globales: *Math*, *JSON*, ...

JavaScript: Sentencias

Sentencias

- ▶ Comentarios: // para una línea, /* ... */ para varias líneas
- ▶ Separador de línea: ; ó fin de línea. Bloques con { ... }
- ▶ Operadores aritmético-lógicos, relacionales, otros:

+ - * / % ++ -- && || ! == != === !== > < >=
<= ?: delete in typeof instanceof void



- ▶ Asignaciones: = += -= *= /= %=

```
console.log('hola')  
console.log('hola'); console.log('adios');  
{  
  console.log('hola');  
  console.log('adios');  
}
```



JavaScript: Sentencias

Sentencias

► Condicionales

```
if (<expr>) <instr> else <instr>   
switch(<expr>) { case <vall>: break; ... default: } 
```

► Bucles

```
while(<expr>) <instr>   
do <instr> while (<expr>);  
for (var=ini; var < fin; var++) <instr>   
for (var in objeto) <instr>
```

```
if (x > 0) console.log('x > 0');  
else if (x < 0) console.log('x < 0');  
else console.log('x == 0');  
  
while(true) console.log('hola');  
for (var i = 0; i < 100; i++) console.log('mensaje ' + i);  
var x = {a: 1, b: 2};  
for (var att in x) console.log(att + '->' + x[att]);
```

JavaScript: Funciones

FUNCIONES son:

- Son objetos “especializados” (heredan todo lo dicho)

- Por tanto, también son diccionarios.

- Definición :

`function f(va1, va2, ...) { ... return [va1];}` → **función con nombre**

`var f=function (va1, var, ...) {... return [va1];}` → **función anónima**

- Parámetros:

- No se comprueba el número de parámetros que se le pasan a las funciones. Así puedo hacer:

```
function f(a,b) { ... };  
f(1); // => a=1, b=undefined
```

- No se define su tipo

JavaScript: Funciones

- Se pueden definir funciones dentro de otras (anidadas):

```
function hipotenusa(a,b) {  
    function cuadrado(x) { return x*x; }  
    return Math.sqrt(cuadrado(a) + cuadrado(b));  
}
```

JavaScript: para plataforma WEB


En el navegador


► Insertar JS en un documento HTML


```
<script type="text/javascript"> /* inline */ </script>
```


```
<script type="text/javascript" src="..."></script>
```


► Objetos predefinidos en el navegador


► window: el objeto global 

► window.navigator 

► window.screen 

► window.history 

► window.location 

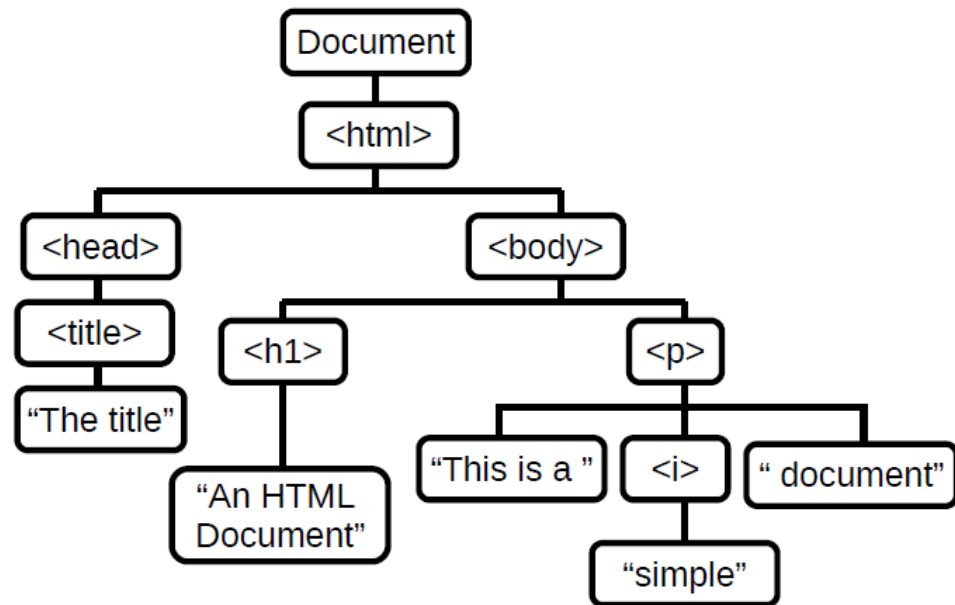
► window.document 

JavaScript: para plataforma WEB

El DOM – window.document

- Modelo en forma de árbol que representa el contenido del documento HTML

```
<html>
  <head>
    <title>The title</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i>
document</p>
  </body>
</html>
```



JavaScript: para plataforma WEB

El DOM – `window.document`

- ▶ Una aplicación interactiva accede y manipula el DOM en reacción a ciertos eventos
- ▶ Distintos tipos de nodos: Document, Element, Attr, Text, Comment
- ▶ Recuperación de nodos: `document.getElementById()`
`sByName()``sByTagName()``sByClassName()`
- ▶ Manipulación de contenido: `.innerHTML`
- ▶ Manipulación del árbol: `.append()``.remove()``.replaceChild()`,
`.insertBefore()`
- ▶ Manejadores de eventos: `.addEventListener()`

JavaScript: para plataforma WEB

El DOM – window.document

Ejemplo

Script al final: el DOM
debe construirse antes

Seleccionar el
botón

Añadir manejador
evento 'click':
cambia estilo del
elemento <p>

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<p id="p1">Hola mundo!</p>
<input id="elBoton" type="button" value="Cambiar estilo" />

<script>
var btn = document.getElementById('elBoton');
btn.addEventListener('click', function() {
  var p = document.getElementById('p1');
  p.style.color = 'blue';
  p.style.fontFamily = 'Arial';
});
</script>
</body>
</html>
```