# JS: Primeros pasos

JavaScript (JS) es un lenguaje de programación, un lenguaje con su propio vocabulario, sintaxis, semántica, expresiones, errores, etc.

JavaScript nos permite darle vida a la web, hacerla más dinámica e interactiva, y por tanto mostrar algo más que información de manera estática.

¿Qué se puede hacer con JavaScript?:

- Operaciones matemáticas, lógicas, etc.
- Controlar el flujo del programa
- Validar formularios
- Cargar contenidos mediante peticiones HTTP
- Modificar el DOM
- Acceder a información como la versión del navegador, tamaño de la ventana, sistema operativo, localización, etc.
- Etc.

# Mi primer script

Este es el script más simple que podemos hacer:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mi primer script</title>
</head>

<body>
  <script>
    document.writeln('Hola Mundo!');
  </script>
</body>
</html>
```

Vamos a ver cómo interpretar este script. Lo que estamos haciendo es:

1. Llamar al **método** `writeln` [1] que escribe en el DOM lo que recibe como parámetro seguido de un salto de línea, en este caso `Hola Mundo!` seguido de un salto de línea ( `\n` ).

2. Este método está definido en el `document`[2] y que representa al DOM y que tiene otras funciones para acceder a elementos del DOM, etc.[3]

Para evitar errores que pueden pasar desapercibidos en JavaScript (por si flexibilidad) te recomiendo que introduzcas siempre la expresión `'use strict';` al principio de tus scripts. El modo estricto significa entre otras cosas que hay que declarar todas las variables y objetos[4].

Así quedaría:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mi primer script</title>
</head>

<body>
  <script>
    'use strict';
    document.writeln('Hola Mundo!');
  </script>
</body>
</html>
```

# Sintaxis

Algunas de las características de JavaScript son:

1. Es sensible a mayúsculas y minúsculas (o lo que es lo mismo, es *case-sensitive*), por tanto:

   ```
   var variable;
   ```

   no es equivalente a

   ```
   var Variable;
   ```

2. No es obligatorio (pero sí recomendado) declarar las variables
3. No se define el *tipo* de las variables
4. No es necesario (pero sí recomendado) terminar cada expresión con el carácter de punto y coma (;)
5. Se pueden incluir comentarios en una línea usando `//` y en múltiples líneas usando `/* */`.

Aclaraciones:
1. Más información sobre el método writeln
2. Más información sobre la interfaz document.
3. Y el documento está definido como parte del objeto **`window`** que representa a la ventana del navegador donde está cargado el DOM y donde se almacena mucha más información. Añadir la palabra `window` es opcional.

</small>

- Y otras tantas restricciones más.

# Variables

Las variables en los lenguajes de programación se asemejan a las variables utilizadas en matemáticas, se utilizan para almacenar y hacer referencia a valores, gracias a ellas podemos darle vida a la web.

Para declarar/definir una variable utilizaremos la palabra clave `var` seguida del nombre de la variable y un punto y coma ( `;` ), por ejemplo:

```
var counter;
```

En este caso hemos declarado una variable con el nombre `counter` pero no se le ha asignado ningún valor.

> **Consejos**:
>
> - Aunque no es obligatorio, acaba siempre las sentencias con punto y coma (por convención).
> - Escribe siempre el código en inglés (se considera más profesional).

El nombre de una variable debe cumplir las siguientes normas:

- El primer carácter **no** puede ser un número.
- Sólo puede estar formado por letras, números y los símbolos: dólar ( `$` ) y guión bajo ( `_` ).

Por tanto, las siguientes variables estarían bien definidas:

```
var $num1;
var _$name;
var $$$otherNumber;
var $_a__$4;
```

> **Consejo**: elige nombres de variables que sean representativos del valor que almacenan para facilitar la comprensión del código.

Por ejemplo:

```
var counter = 0;
var name = "Raul";

// En lugar de:
var aux = 0;
var tmp = "Raul";
```

# Palabras reservadas

Antes de continuar me gustaría comentarte que existen palabras reservadas que tienen un significado en el lenguaje y que no podremos usar como nombres de variables: `abstract, boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, volatile, void, while, with` .

# Tipos de variables

En todos los lenguajes de programación existen distintos tipos de variables, en JavaScript tendremos:

```
// Númericas (integer & floats)
// ----------------------------
var counter = 16;   // variable tipo entero
var price = 19.99;  // variable tipo decimal
```

Que nos permiten almacenar números enteros y con decimales para realizar operaciones.

```
// Cadenas de texto (strings)
// --------------------------
var msg = 'Bienvenido a nuestro sitio web';
var txt = 'Una frase con "comillas dobles" dentro';
var txt = 'Una frase con \'comillas simples\' dentro';
```

Que nos permiten trabajar con cadenas de texto. Para ello tenemos con encerrar la cadena entre comillas simples o dobles, pero normalmente se recomienda hacerlo con comillas simples. En caso de querer introducir una comilla simple dentro de una cadena podemos hacerlo incluyendo el carácter contra-barra (\) justo delante, para evitar que se cierre la cadena.

```
// Colecciones (arrays)
// ----------------------------
// Definiendo los días de la semana en cadenas de texto
var day1 = 'Lunes', day2 = 'Martes', ... , 'Domingo';

// Definición equivalente en un Array
var days = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo'];

// days[0] = 'Lunes'
// days[1] = 'Martes'
// ...
// days[6] = 'Domingo'
```

Los `Arrays` o colecciones nos permiten añadir varios valores dentro de un elemento.

```
// Booleanos (boolean)
// ----------------------------
var valid = false;
var prime = true;
```

Los booleanos se utilizan para almacenar valores lógicos: `true` o `false`.

# Funciones

Existen múltiples funciones para trabajar con números: Para los números hay una función muy útil:

```
var n = 231.8273;
n.toFixed(2); // 231.82
```

cadenas de texto, por ejemplo:

```
var hello = 'Hola ';
var world = 'Mundo!';

// Para contar el número de caracteres
console.log(hello.length); // 5

// Para concatenar cadenas
console.log(hello + ' ' + world); // Hola Mundo!
console.log(hello.concat(' ' + world));  // Hola Mundo!

// Para busacar subcadenas en una cadena
var pos = hello.indexOf('a'); // pos = 3
var pos = hello.indexOf('b'); // pos = -1
```

Y otros métodos: `lastIndexOf` , `substring` , `split` , etc.

Al igual para trabajar con Arrays:

```
var fruits = ['banana', 'melon, 'orange'];

// Para contar
var n = fruits.length; // n = 3

//Para añadir elementos
fruits.push('apple', 'peach'); // fruits = ['banana', 'melon, 'orange', 'apple', 'peac
h']
```

`contact` , `join` , `pop` , `shift` , Y otras como: `unshift` , `reverse` .

# Operadores

Los operadores nos vas a servir para modificar y comprobar el valor de las variables, vamos a ver diferentes tipos de operadores:

- Matemáticos
- Lógicos
- Relacionales

# Operadores matemáticos

Los operadores matemáticos nos van a permitir realizar operaciones matemáticas sobre las variables, veamos algunos ejemplos:

```
// Asignación (=)
var pi = 3.1416;
```

Nos permite darle un valor a una variable.

> **Consejo**: Añade siempre un espacio antes y otro después de cualquier operador ( `=` , `<` , ..).

```
// Incremento (++) y decremento (--)
var x = 1, y = 4;
x++; // x = 2
y--; // y = 3
```

Nos permite incrementar o decrementar en una unidad el valor de una variable.

```
// Suma (+) y resta (-)
var x = 2, y = 3, z;
z = x + y // z = 5;
z = x - y // z = -1;
```

```
// División (/) y multiplicación (*)
var x = 4, y = 2, z;
z = x / y // z = 2;
z = x * y // z = 8;
```

```
// Abreviaciones
var x = 5;
x += 3;  // x = x + 3 => 8
x -= 1;  // x = x - 1 => 4
x *= 2;  // x = x * 2 => 10
x /= 5;  // x = x / 5 => 1
```

// Módulo (%) numero1 %= 4; // numero1 = numero1 % 4 = 1

# Operadores lógicos

Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para tomar decisiones sobre las instrucciones que debería ejecutar el programa en función de ciertas condiciones.

El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o booleano.

```
var visible = true;
!visible;  // Devuelve "false" y no "true"
```

| x | !x |
|---|---|
| true | false |
| false | true |

**Operación AND (&&)**

La operación lógica AND obtiene su resultado combinando dos valores booleanos. El operador se indica mediante el símbolo && y su resultado solamente es `true` si los dos operandos son `true`:

| x | y | x && y |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

```
var x = true;
var y = false;
result = x && y; // result = false

x = true;
y = true;
result = x && y; // result = true
```

### Operación OR (||)

La operación lógica OR también combina dos valores booleanos. El operador se indica mediante el símbolo || y su resultado es `true` si alguno de los dos operandos es `true` :

| x | y | x \| \| y |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

```
var x = true;
var y = false;
result = x || y; // result = true

x = false;
y = false;
result = x || y; // result = false
```

# Operadores relacionales

Los operadores relacionales definidos por JavaScript son los mismos que los matemáticos:

- Mayor que: `>`
- Menor que: `<`
- Mayor o igual: `>=`
- Menor o igual: `<=`
- Igual que: `==`
- Distinto de: `!=`

Aunque también existe el operador `===` que quiere decir **exáctamente igual**, teniendo en cuenta no sólo el valor de la variable sino también el tipo, por ejemplo:

```
0 == ""         // true
0 === ""        // false

0 == false      // true
0 === false     // false

2 == '2'        // true
2 === '2'       // false
```

Vamos a ver en la siguiente lección que estos operadores son imprescindibles a la hora de controlar el flujo de un programa.

El resultado de todos estos operadores siempre es un valor booleano:

```
var even = 2;
var odd = 5;
result = even > odd; // result = false
result = even < odd; // result = true

a = 5;
b = 5;
result = a >= b; // result = true
result = a <= b; // result = true
result = a == b; // result = true
result = a != b; // result = false
```

Se debe tener especial cuidado con el operador de igualdad (==), ya que es el origen de la mayoría de errores de programación, incluso para los usuarios que ya tienen cierta experiencia desarrollando scripts. El operador == se utiliza para comparar el valor de dos variables, por lo que es muy diferente del operador =, que se utiliza para asignar un valor a una variable:

```
// El operador "=" asigna valores
var x = 5;
y = x = 3;  // y = 3 y x = 3

// El operador "==" compara variables
var x = 5;
y = x == 3; // x = 5 y y = false

/*
   Los operadores relacionales también se pueden
   utilizar con variables de tipo cadena de texto:
*/
var txt1 = "hola";
var txt2 = "hola";
var txt3 = "adios";

result = txt1 == txt3; // result = false
result = txt1 != txt2; // result = false
result = txt3 >= txt3; // result = false
```

Cuando se utilizan cadenas de texto, los operadores "mayor que" ( `>` ) y "menor que" ( `<` ) siguen un razonamiento no intuitivo: se compara letra a letra comenzando desde la izquierda hasta que se encuentre una diferencia entre las dos cadenas de texto. Para determinar si una letra es mayor o menor que otra, las mayúsculas se consideran menores que las minúsculas y las primeras letras del alfabeto son menores que las últimas (a es menor que b, b es menor que c, A es menor que a, etc.)

# Ejercicios

## 1) Instalar Sublime Text

Instalar Sublime Text y en `preferences->settings-user` comprobar que están estas tres líneas (sino añadirlas):

```
{
    "indent": 2,
    "tab_size": 2,
    "translate_tabs_to_spaces": true
}
```

## 2) Operaciones simples

Realiza un script que realice lo siguiente:

- Almacenar en una variable el resultado de sumar 1 y 2
- Almacenar en una variable el resultado de dividir 6 entre 2
- Almacenar en una variable el precio de un artículo de 20€ aplicándole el 21% de IVA .
- Definir una variable con el valor 4 y utilizar el operador ( `++` ) para incrementar en uno su valor.
- Definir una variable que almacene la concatenación de dos cadenas de texto.
- Definir una variable `price` con el valor `19.99` y aplicar la abreviación `/=` para dividirlo entre `1.21` para obtener el precio sin IVA.
- Asignar a dos variables valores booleanos y hacer al menos una operación combinando un operador lógico: **AND** ( `&&` ) o **OR** ( `||` )
- Realizar 4 expresiones que utilicen operadores relacionales ( `<` , `==` , `!=` y `===` ) y almacenen los valores en tres variables distintas.

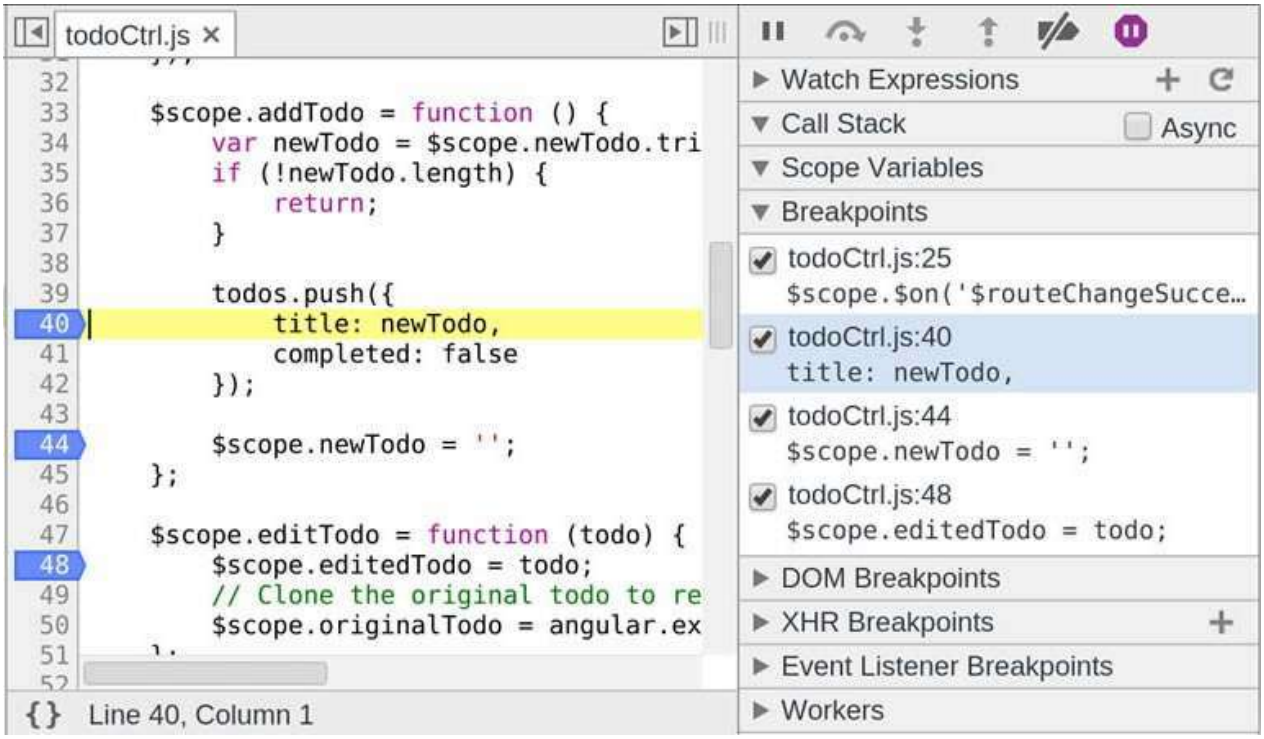
Finalmente imprime **todos valores** en la consola del navegador usando `console.log(nombre_de_la_variable)` , por ejemplo:

```
var result = 1 + 2;
console.log(result);
```

> **Nota**: Como la mayoría de los lenguajes, JavaScript se ejecuta secuencialmente (de arriba a abajo), por lo que el orden de las instrucciones importa.

# 3) Puntos de parada

Utiliza la pestaña sources y haz clic en alguna línea donde haya una expresión para establecer un punto de parada (se debe marcar en azul) y recarga la página:




Juega con el inspector.

# Opcional: Instalar W3CValidators

Recuerda que en las lecciones de HTML utilizábamos el validador online del W3C para comprobar que nuestro código era correcto. Si lo prefieres también puedes usar la extensión W3CValidators de Sublime Text para hacerlo desde el propio editor.

# Dudas

Si hay algo que no te haya quedado claro puedes preguntar cualquier duda en los issues del proyecto en Github.

Si tienes problemas o dudas con tu código súbelo a Github, abre un issue en un proyecto con la duda/problema y envíame un correo a info@cursohtml5desdecero.com.

# Recursos

- Playlist de Youtube con introducción a SublimeText

# Otras aclaraciones

## Objeto window

Otras funciones comunes definidas en el objeto `window` son:

- `alert()` que abre una ventaja con un mensaje el navegador del usuario, aquí puedes verlo en funcionamiento.
- `console` que implementa funciones para imprimir mensajes en la consola de error ( `console.error()` ), etc

## Otras características de JS

Existen más características, como que si la ejecución de un script dura demasiado tiempo (por un error, por ejemplo de programación) el navegador puede informarle al usuario de que hay un script que está consumiendo demasiados recursos y darle la posibilidad de detener su ejecución.

## Orden de definición de las variables

Definirlas en la parte superior del script

# Estructuras de control

## Estructura `if`

```
var printMsg = true;

if(printMsg) {
  console.log('Hola Mundo');
}

if(printMsg == true) {
  console.log('Hola Mundo');
}
```

Un ejemplo usando un comparador lógico:

```
var printMsg = false;

if(!printMsg) {
  console.log('Me imprimo');
}

var isFirstMsg = true;

if(!printMsg && isFirstMsg) {
  console.log('Mi primer mensaje');
}
```

Un **error típico** es introducir una asignación ( `=` ) en lugar de una comparación ( `==` )

```
// Error - Se asigna el valor 'false' a la variable
if(printMsg = false) {
  ...
}
```

## Estructura `if` ... `else`

```
var age = 18;

if(age >= 18) {
  console.log('Eres mayor de edad');
}
else {
  console.log('Eres menor de edad');
}
```

## Estructura `if` ... `else if` ... `else`

```
if(age < 18) {
  console.log('Eres menor de edad');
}
else if(age < 30) {
  console.log('Aún eres joven');
}
else {
  console.log('La sabiduría la da la experiencia');
}
```

## Estructura `for`

```
for(initialization; condition; increment) {
  ...
}
```

```
var i;
var days = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo'];

for(i = 0; i < days.length; i++) {
  alert(days[i]);
}
```

# Objetos y funciones

Iterar sobre objetos callbacks

ámbitos

```
// Objetos (objects)
// ----------------------------
var obj = {
    name: 'Raul',
    last_name: 'Jimenez Ortega',
    age: 31
};

// obj.name = 'Raul'
// obj.last_name = 'Jimenez Ortega'
// obj.age = 31
```

Los objetos nos permiten definir estructuras de datos con distintos tipos de valores, ya verás que esto te será muy útil en el futuro.

# Peticiones AJAX

Google Spreadsheets CORS http://www.html5rocks.com/en/tutorials/cors/#toc-adding-cors-support-to-the-server https://www.youtube.com/watch?v=3l13qGLTgNw

# Expresiones regulares

# Aplicaciones web offline

http://www.html5rocks.com/en/tutorials/appcache/beginner/

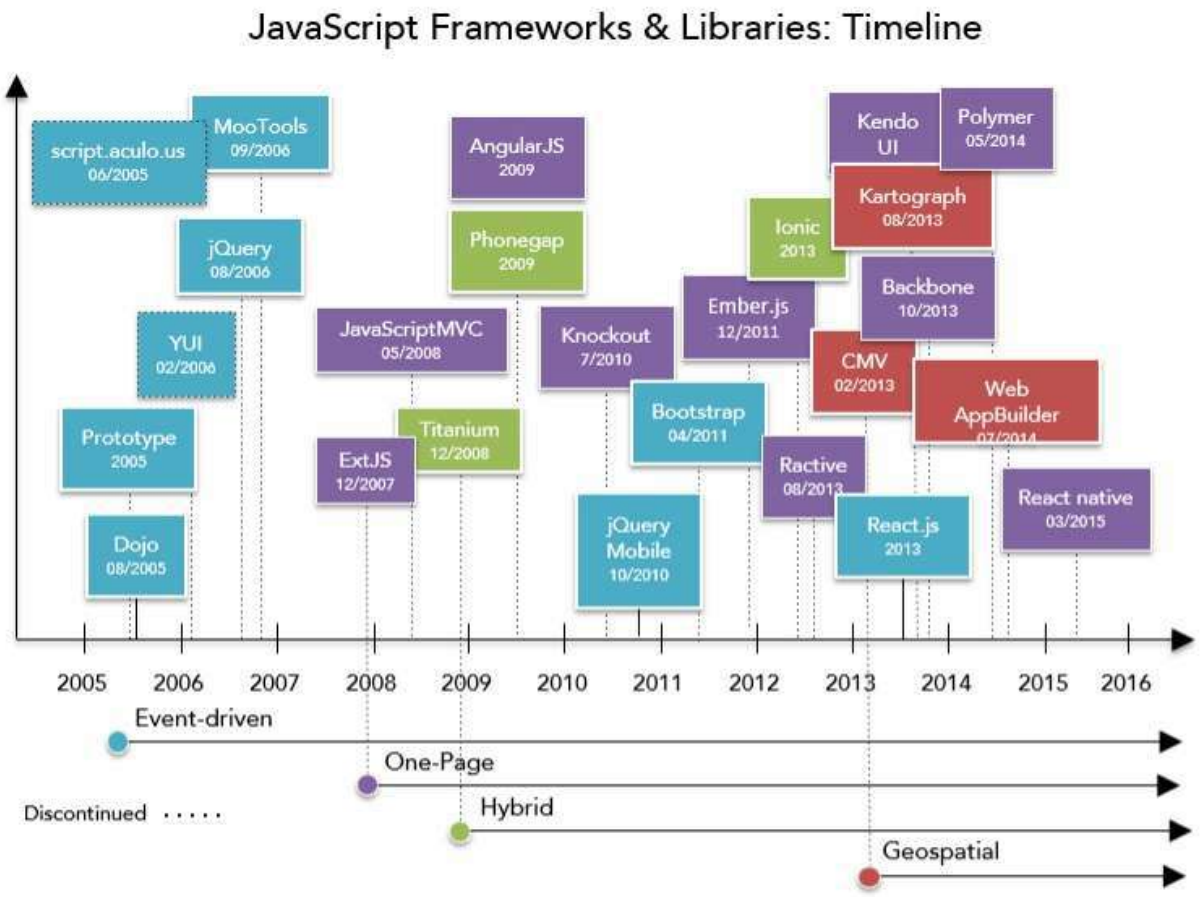
# Bibliotecas de terceros

Gráfica

jQuery

Dojo

http://download.dojotoolkit.org/release-1.6.0/cheat.html



# Ejercicios

## Instalar JSHint

https://github.com/victorporof/Sublime-JSHint