

# Fundamentos de Aplicaciones y Servicios Telemáticos

2º Grado en Ingeniería de Tecnologías de Telecomunicación

Departamento de Ingeniería Telemática



## Tema 02

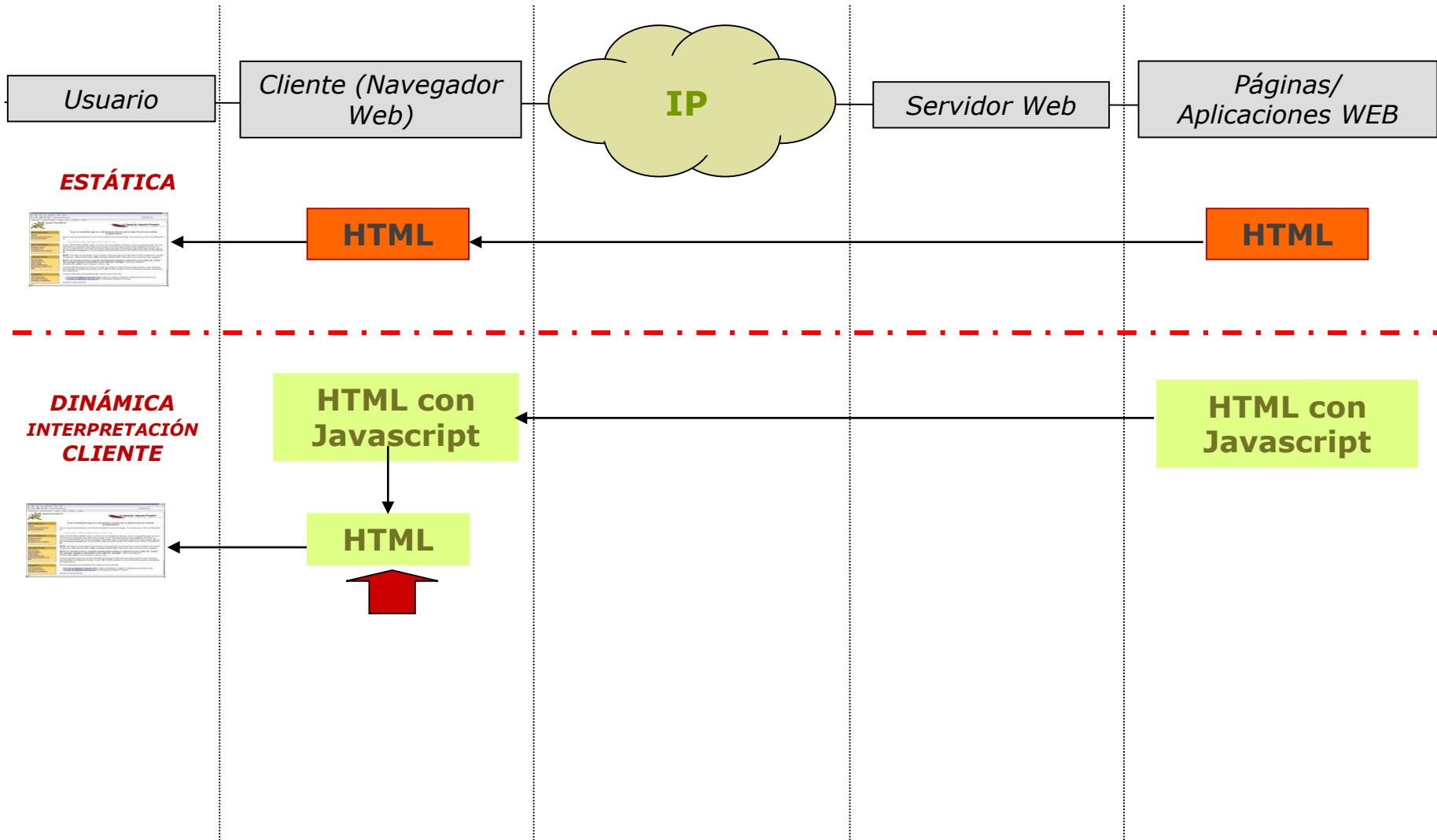
Programación Web Dinámica  
con interpretación en el Cliente

# Contenido del Tema (Teoría + Práctica)

---

- Introducción:  
Programación web  
dinámica en cliente
  - Funcionalidad
  - Clasificación
- ECMAScript
  - Introducción
  - Inserción en HTML
  - Sintaxis
    - Comentarios
    - Variables
      - Tipos de datos
    - Operadores
    - Funciones
    - Objetos
    - Estructuras de control
  - Objetos predefinidos
    - Del lenguaje ECMAScript
    - Del modelo DOM
    - Del modelo BOM
  - AJAX
  - Librerías externas

# Introducción



# Introducción (2)

---

- ❑ Programación web estática:
  - Contenidos con formato: HTML + CSS
  - Últimas versiones (HTML5, CSSlevel3) incluyen alguna interactividad
- ❑ Programación web dinámica:
  - Interactividad mas avanzada
  - Creación de contenidos en el momento de la interpretación: consultas a bases de datos, etc.
- ❑ Tipos de programación web dinámica:
  - Del lado del cliente (client-side):
    - ❑ El dinamismo se proporciona en el navegador
  - Del lado del servidor (server-side).
    - ❑ El dinamismo se proporciona en el servidor

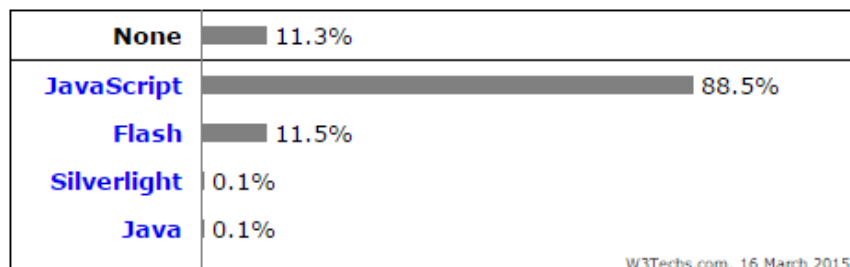
## Introducción (3): Clasificación

- Clasificación lenguajes de Programación dinámica en cliente:

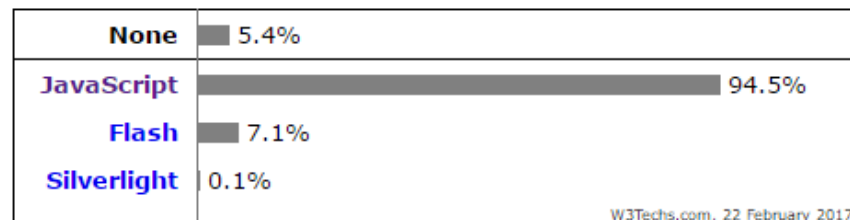
Tipos		Lenguajes Web
Mediante Plugins	Lenguajes de script (SIN plugins)	<ul style="list-style-type: none"> <li>❑ <b>ECMAScript</b> (estándar ISO/IEC, sintaxis similar C/Java)               <ul style="list-style-type: none"> <li>➤ <b>Javascript</b> (Netscape y Sun Microsystems)</li> <li>➤ <b>JScript</b> (Microsoft)</li> </ul> </li> </ul>
	Java  Otros	<ul style="list-style-type: none"> <li>❑ <b>Applets Java</b> (Sun Microsystems/Oracle)</li> <li>❑ <b>Flash: Lenguaje ActionScript</b>, deriva de ECMAScript</li> <li>❑ <b>Lenguajes C#, ...</b> (Microsoft Silverlight/Linux Moonlight)</li> <li>❑ <b>JavaFX Script</b> (Sun/Oracle, JavaFX)</li> </ul>

# Introducción (4): Porcentaje de uso

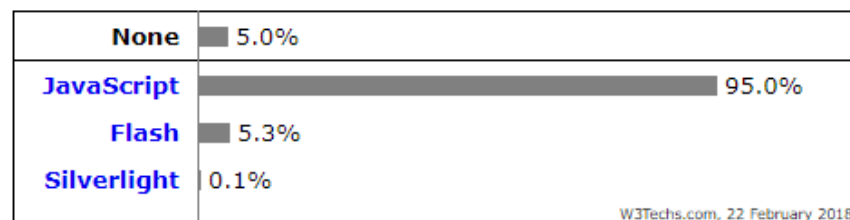
- Javascript es el más usado con diferencia



Percentages of websites using various client-side programming languages  
Note: a website may use more than one client-side programming language



Percentages of websites using various client-side programming languages  
Note: a website may use more than one client-side programming language



Percentages of websites using various client-side programming languages  
Note: a website may use more than one client-side programming language

*Estadística referida al número de "aplicaciones web" escritas en estos lenguajes, SIN tener en cuenta el tamaño ni tráfico de la página*

# ECMAScript: Índice

---

- ECMAScript
  - Introducción
  - Inserción en HTML
  - Sintaxis
    - Comentarios
    - Variables
      - Tipos de datos
    - Operadores
    - Funciones
    - Objetos
    - Estructuras de control
  - Objetos predefinidos
    - Del lenguaje ECMAScript
    - Del modelo DOM
    - Del modelo BOM
  - AJAX
  - Librerías externas

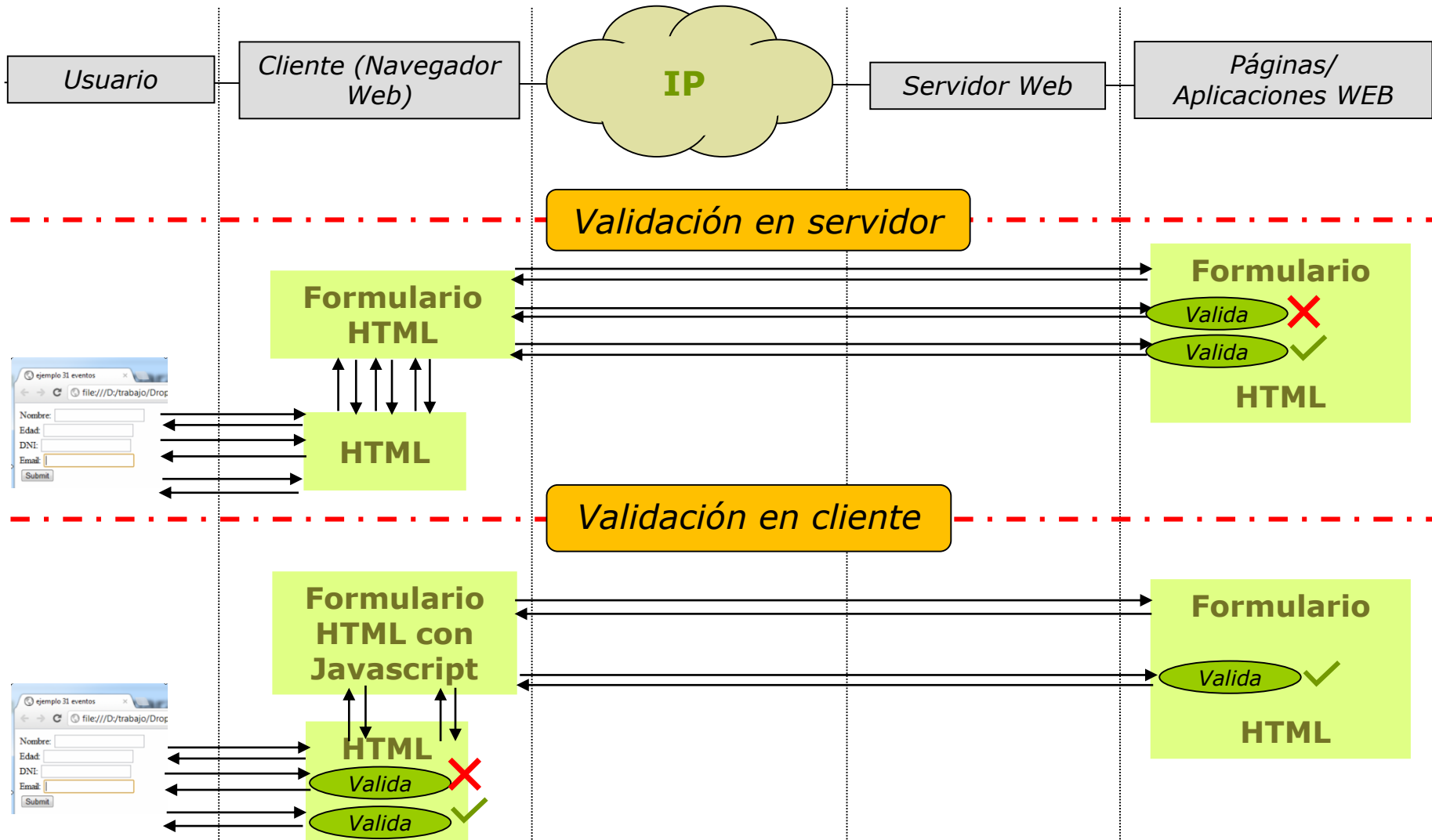
# ECMAScript: Introducción

---

- ❑ Lenguaje de script usado:
  - Principalmente: crear páginas web dinámicas interpretadas en el lado del cliente.
  - Otros: en el servidor, SSJS (Server-Side JavaScript node.js), aplicaciones móviles, etc.
- ❑ Situación que motivó su origen:
  - Años 90: conexiones lentas.
  - Páginas web de formularios: campos validados en servidor. Muchas esperas.
  - Solución: validar los campos en el cliente, enviando al servidor cuando todo esté bien; más rápido y descarga al servidor



# ECMAScript: Introducción. Ventaja de uso



# ECMAScript: Introducción. Origen

---

- LiveScript: por Netscape.
- Javascript: entre Netscape y Sun Microsystems, alcanzando un gran éxito (el nombre fue elegido para que se pareciese a "Java", en auge en el momento)
- JScript: Microsoft, para competir con Javascript
- ECMAScript: a partir de Javascript y JScript
  - Especificación de la fundación empresarial ECMA (ECMA-262), aceptado como estándar ISO/IEC 16262.
  - ECMA-262 edición 5.1 - ISO/IEC 16262:2011
  - ECMA-262 edición 6 (junio 2015: ES2015 ES6)
  - ECMA-262 edición 7 (junio 2016: ES2016)
  - ECMA-262 edición 8 (junio 2017: ES2017)
- Javacript y Jscript adoptaron el estándar ECMAScript, soportándolo y añadiendo funcionalidades propias adicionales.
- Cada navegador tiene su implementación.
- La ECMA-262 edición 5.1 (conocida como ECMA5) es la que se supone que soportan la mayoría de los navegadores actuales.

# Limitaciones ECMAScript

---

- ❑ Diseñado para ejecutarse en entorno limitado, por ejemplo en un navegador, en el cual por defecto:
  - No pueden:
    - ❑ acceder a los archivos del ordenador del usuario
    - ❑ comunicarse con recursos que no pertenezcan al mismo dominio de Internet desde el que se descargó el script.
    - ❑ cerrar ventanas que no haya abierto el script.
    - ❑ leer o modificar las preferencias del navegador.
  - Tamaño y posición de las ventanas limitados.
  - Si la ejecución dura demasiado tiempo el navegador informa al usuario de que un script está consumiendo demasiados recursos y le da la posibilidad de detener su ejecución.
- ❑ Hay alternativas para saltar limitaciones según el navegador (solicitar al usuario permiso para realizar esas acciones).

# ECMAScript: Inserción en HTML (1)

## 4 Métodos

file.html

```
<html><head>
```

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

```
</script>
```

```
<script>
```

```
function display() {
```

```
    alert("Ventana, modo 4");
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<script>
```

```
    alert("Ventana, modo 2");
```

```
</script>
```

```
<p onclick="alert('modo 3')"> Texto html.</p>
```

```
<button type="button" onclick="display()">Display</button>
```

```
</body> </html>
```

1 Referencia a *fichero externo* → eje.js

alert("Ventana, método 1");

2 Código en *<head>*

Texto html.

Display

Texto html.

Display

3 Código en *<body>*

4 atributo de Evento de etiqueta HTML

# ECMAScript: Inserción en HTML (2), Eventos HTML

---

## □ Eventos:

- Acciones detectadas por el Navegador útiles para invocar (disparar/trigger) en ese momento código Javascript.
- Se usan mediante atributos de los elementos HTML

## □ Ejemplos de eventos:

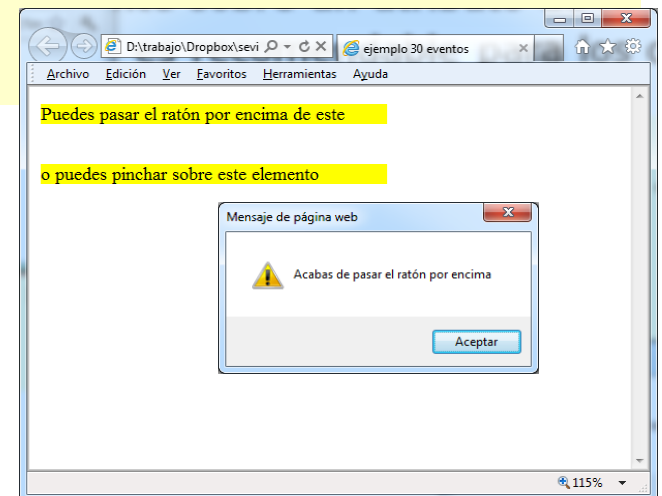
- onclick
  - El clic de un ratón
- onload
  - La carga de una página web o una imagen
- onsubmit
  - Enviar un formulario
- ...
- onfocus
- keypress
- onmouseover
- ...

# ECMAScript : Inserción en HTML (3), Eventos HTML

- Lo normal es asociar funciones Javascript a los eventos.

```
<html><head>
</head><body>
<div onmouseover="alert('Acabas de pasar el ratón por encima');">
Puedes pasar el ratón por encima de este
</div><br/><br/>
<div onclick="x='Acabas de pinchar con el ratón';alert(x); ">
o puedes pinchar sobre este elemento
</div>
</body>
</html>
```

*El valor del atributo, lo que está entre comillas, se convierte en el contenido del cuerpo de una función (sin nombre) que se ejecuta cuando sucede el evento.*



# ECMAScript : Inserción en HTML (4), en XHTML

---

- ❑ Recomendación adicional en XHTML: incluir el código JavaScript en sección CDATA, evitando problemas con operadores "<", ">", ...
- ❑ Aunque NO es obligatorio

```
<head>
  <script type="text/javascript">
    //  <![CDATA[
        alert("Pulse Intro para continuar");
    //  ]]>
  </script>
</head>
```

# ECMAScript: Sintaxis

---

- ❑ La sintaxis del código Javascript es similar a la del lenguaje "C", "C++" o "Java"
- ❑ Case sensitive
- ❑ Por defecto, estructura secuencial
- ❑ Separar instrucciones, mediante:
  - Carácter punto y coma (;)
  - O un salto de línea (FAST: usar SIEMPRE el punto y coma).
- ❑ Bloques con {}:

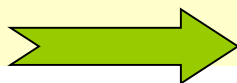
```
function avisa() {  
    alert("Primer aviso")  
    alert("Y segundo aviso");  
}
```



# ECMAScript: Sintaxis, Comentarios

- `// ...`
  - Hasta final de línea
- `/* ... */`
  - Comentario de bloque: desde `/*` hasta `*/`

```
<script>
/* The code below will write
one heading and one paragraph*/
document.write("<h1>A heading</h1>");
document.write("<p> A paragraph.</p>");
// document.write("<p>Another.. </p>");
</script>
```



Si se llama a `write` una vez leída toda la página por el navegador (usando, por ejemplo, eventos), la función "write" genera un nuevo cuerpo de página web => borra todo el contenido previo de la página (del "body")

# ECMAScript: Sintaxis, Variables

- Definición (e inicialización) de variables:

```
resultado = 0;
//declarada e inicializada
var texto="Hola mundo";
var suma;
//suma sin valor => undefined
```

- Si dentro de una función se define una variable (con **var**), es local. Si se inicializa sin estar previamente definida se define automáticamente como global (fuera de función todas son globales)

```
function f(){
  valor = 3;    //variable global
  var valor = 3; //variable local
}
```

- Antes de acceder al valor de una variable, es necesario que esté definida.

- En FAST usar SIEMPRE la directiva "use strict";

```
"use strict";
resultado = 0;
//error falta declaración
var texto="Hola mundo";
testo="Adios mundo";
//error falta declaración
```

- Caracteres para los nombres de variables:

- Letras, números (no primer carácter), '\$' y '\_'.

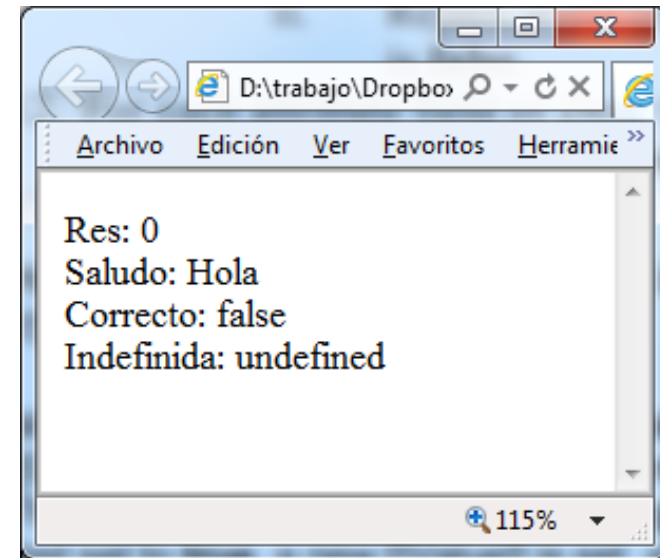
# ECMAScript: Sintaxis, Tipos de datos

- ❑ Al declarar la variable no se define el tipo.
  - Se define en el momento de asignarle un valor.
- ❑ El tipo se puede modificar con un nuevo valor.
- ❑ Tipos de **datos primarios o nativos**:

Tipo	Valores	Comentario
undefined	Undefined	Variables sin valor asignado
boolean	true o false	
string	Texto	Indexado a partir de cero
number	Enteros o decimales	
object	Objetos	Base para nuevos tipos de objetos

## ECMAScript: Sintaxis, Tipos de datos (2)

```
<script>
var res=0;
var saludo="Hola"
var correcto=false;
var indefinida;
document.write("Res: ", res,
               "<br/>Saludo: ", saludo,
               "<br/>Correcto: ", correcto,
               "<br/>Indefinida: ", indefinida);
</script>
```



# ECMAScript: Sintaxis, Operadores

---

- ❑ Similar a C, Java
- ❑ Asignación: `=`, `+=`, `*=`, etc.
- ❑ Matemáticos: `+`, `-`, `*`, `/`, `%`, `++`, `--`
- ❑ Lógicos: `&&`, `||`, `!`
- ❑ Relacionales: `<`, `>`, `<=`, `==`, `!=`,  
`===` (strict equal), `!==`
- ❑ Condicional: `variable=(condición)?valor1:valor2;`

*Sólo valor (posible  
conversión)*

*Valor y tipo (sin  
conversión)*

## ECMAScript: Sintaxis, Operadores (2)

- "+" con string: Concatenación
  - Si suma string y número: *conversión* automática a string

```
var x=7;   var y="4";  
var z=x+y ; // z vale "74"
```

- Igualdad estricta

```
x=5; y="5"; // se remarca lo que se imprime  
if ( x == y )
```

```
    document.write('<br/> 5 es igual(==) a "5"');
```

```
else
```

```
    document.write('<br/>5 NO es igual(==) a "5"');
```

```
if ( x === y )
```

```
    document.write('<br/>5 es igual(===) a "5"');
```

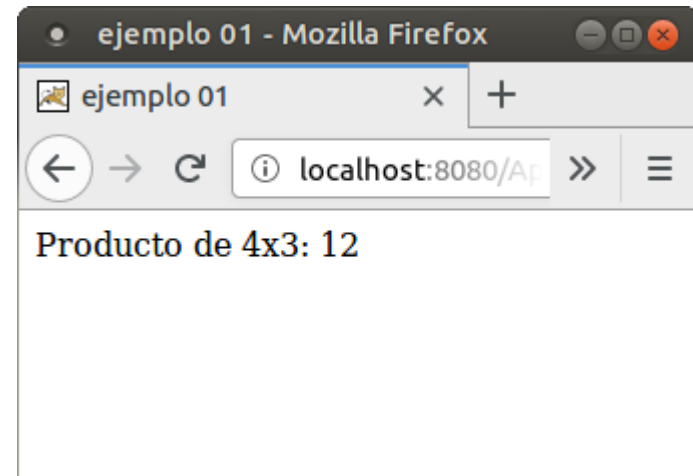
```
else
```

```
    document.write('<br/>5 NO es igual(===) a "5"');
```

# ECMAScript: Sintaxis, Funciones

- ❑ Mismos criterios de nombrado que en las variables

```
<html>
<head>
<title> ejemplo 01</title>
<script>
function producto(a,b) {
    return a*b;
}
</script>
</head>
<body>
<script>
document.write("Producto de
                4x3: ");
document.write(producto(4,3));
</script>
</body>
</html>
```



# ECMAScript: Sintaxis, Objetos

## □ Objetos:

■ Todos derivan del tipo base "object", constructor Object()

■ Pueden ser:

□ Definidos por el usuario. Sintaxis:

Notación JSON  
(JavaScript Object Notation)

**Definición e  
Inicialización  
(3 sintaxis)**

```
var objeto= { campo1:"cadena", campo2:2 };
```

```
var objeto= { "campo1":"cadena", "campo2":2 };
```

```
var objeto= new Object();
```

```
objeto.campo1="cadena";
```

```
objeto.campo2=2;
```

**Uso (2  
sintaxis)**

```
objeto["campo1"]
```

```
objeto.campo2
```

Sintaxis equivalente a la de una  
"tabla asociativa" (índices textuales)  
de lenguajes como Java

□ Predefinidos (con sus variables y funciones miembro): "Date", "Math", ...

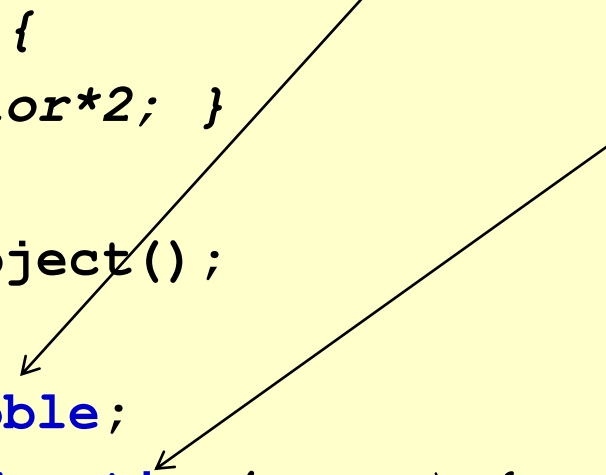
```
alert( Math.random() );
```



# Ecmascript: Métodos en objetos

- ❑ Método: función miembro de un objeto.
- ❑ Puede usarse una función existente o definir nueva.

```
function doble () {  
    return this.valor*2; }  
  
var ancho1=new Object();  
ancho1.valor=5.5;  
ancho1.duplica=doble;  
ancho1.multiplo=function (veces) {  
    return this.valor*veces; }  
  
//Uso  
ancho1.duplica(); ancho1.multiplo(3)
```



# Ecmascript: Constructor de objetos

---

- ❑ Constructor: Función que sirve para crear un nuevo objeto (y darle valores)

```
function Ancho(v) {  
    function doble() {  
        return this.valor*2; }  
    this.valor=v;  
    this.duplica=doble;  
    this.multiplo=function(veces) {  
        return this.valor*veces; }  
}
```

```
//Uso  
var anchoMesa= new Ancho(10);  
anchoMesa.duplica();
```

## Ecmascript: Sintaxis, Estructuras de control

---

- ❑ Similares a C, java, etc.
- ❑ if, if.. else, if .. else if ...

- ❑ switch

- ❑ for

- ❑ for in

- ❑ while

- ❑ do while

- ❑ try-catch-throw

```
var error=false;
if (error)
    document.write("Error <br/>");
else {
    limite=5;
    document.write("No error <br/>");
}
for (i=0; i<limite;i++)
    document.write(i, " ");
</script>
```

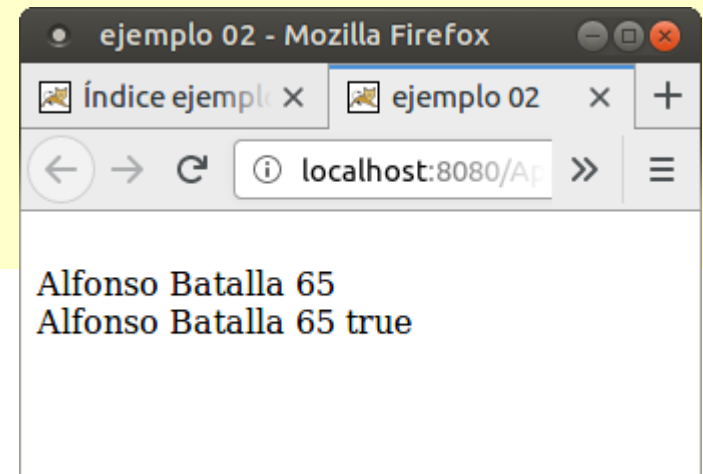
- ❑ Falso: 0, -0, null, "", false, undefined, NaN

## Ecmascript: Sintaxis, Estructuras de control (2)

### □ "for ... in": uso con objetos.

```
<script>
var alumno={nombre:"Alfonso ",apellido:"Batalla",edad: 65, casado:
true };
document.write(    "<br/>",
    alumno.nombre,    " ",
    alumno.apellido,  " ",
    alumno.edad,      "<br/>");

var x;
for (x in alumno) {
    document.write( alumno[x] + " ");
}
</script>
```



Equivalentes

alumno.nombre

alumno["nombre"]

# EcmaScript: Objetos predefinidos

---

- ❑ Clasificados en 3 grupos:
  - Objetos del lenguaje ECMAScript
  - Objetos del modelo DOM (*Document Object Model*)
  - Objetos del navegador, modelo BOM (*Browser Object Model*)
- ❑ Cada objeto predefinido posee sus funciones (métodos) y variables (propiedades) miembro predefinidas

# EcmaScript: Objetos del Lenguaje

Objeto	Función
Boolean	Envoltorio del tipo "boolean" (envoltorio de un objeto primitivo: tiene su valor y añade funciones miembro para operar con él)
String	Envoltorio del tipo "string"
Number	Envoltorio del tipo "number"
Array	Tablas
Math	Operaciones matemáticas
...	

- Objeto String: conversión automática del tipo "string"

```
saludo="cadena";  
alert(saludo.length);
```

# Ecmascript: Objetos del Lenguaje.

## Array (Tabla)

---

### □ Objeto Array:

#### ■ Definición e Inicialización: 2 sintaxis.

```
var tabla = new Array(2, "hola", true);
```

```
var tabla = new Array();  
tabla[0]=2;  
tabla[1]="hola";  
tabla[2]=true;
```

```
var tabla = [2, "hola", true];
```

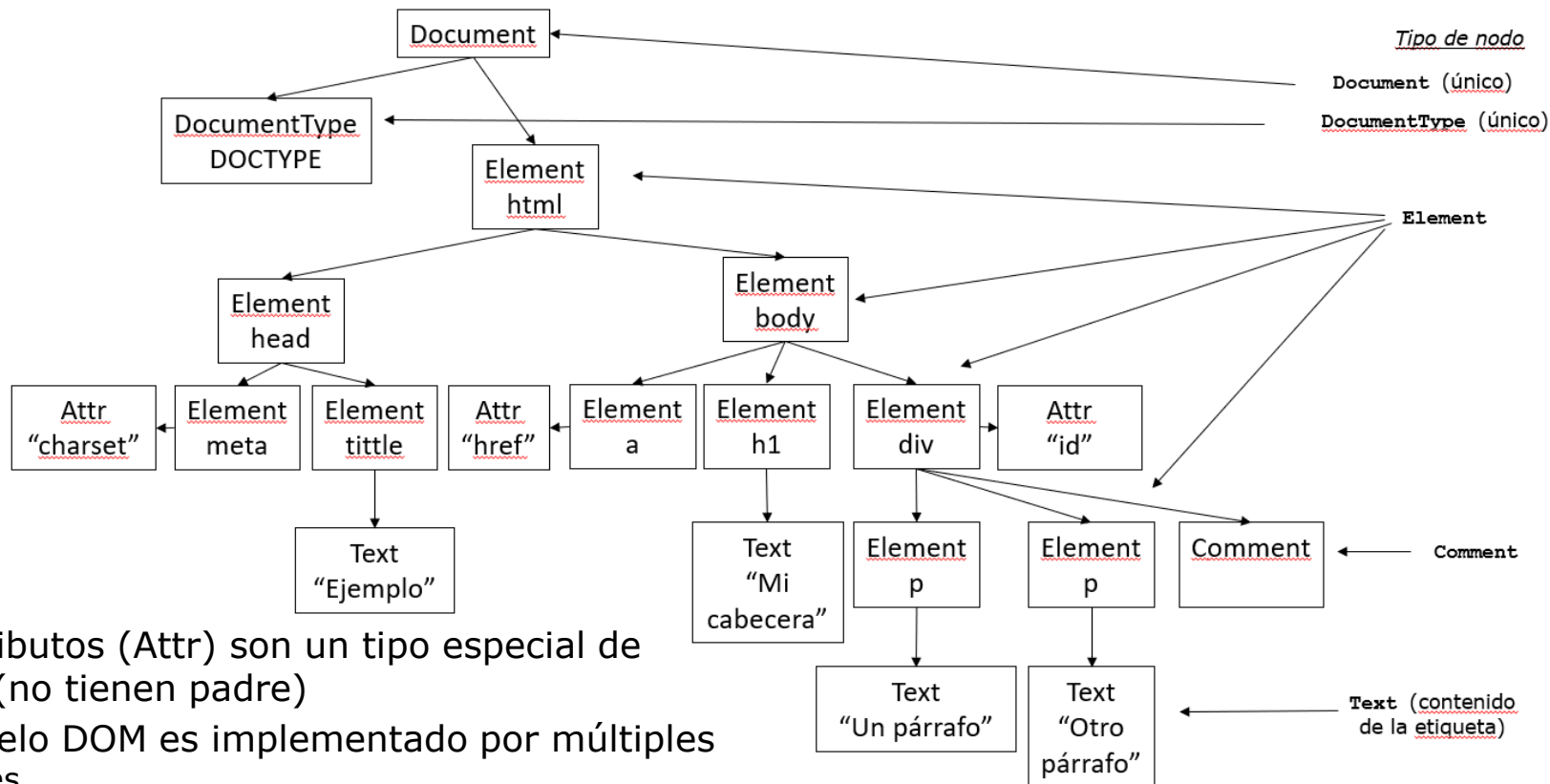
```
var tabla = [];  
tabla[0]=2;  
tabla[1]="hola";  
tabla[2]=true;
```

#### ■ Acceso:

```
alert (tabla[1]);
```

# Ecmascript: DOM

- **DOM (Document Object Model):** modelo estándar W3C para acceder (leer/modificar) a documentos HTML y XML (Core DOM + HTML DOM)
- Modela el documento como un **árbol de nodos** (cada nodo es un objeto de tipo "Node"; un nodo hijo es un objeto miembro del nodo padre):



- Los atributos (Attr) son un tipo especial de nodos (no tienen padre)
- El modelo DOM es implementado por múltiples lenguajes



# Ecmascript: DOM tipos de nodos e hijos

- Normalizado por W3C
  - <https://www.w3.org/TR/dom/>
- Tipos de nodos más conocidos (usados en FAST):

Tipo	descripción	hijos posibles
Document	Todo el documento es un nodo documento	Element (máximo uno), Comment, DocumentType, ...
Element	Cada elemento HTML es un nodo elemento	Element, Text, Comment, ...
Text	El texto dentro de los elementos HTML son nodos de texto	Sin hijos
Comment	Todos los comentarios son nodos comentario	Sin hijos
DocumentType	Asociado a DOCTYPE	Sin hijos
...	...	...

# Ecmascript: visor de nodos de DOM

□ <http://software.hixie.ch/utilities/js/live-dom-viewer/>

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" id="x">
<title>Ejemplo DOM </title>
</head>
<body>Texto en body
<div id="cabecera" class="brillante" >
<h1>Prueba <span class="destacado">de
DOM</span>, para probar</h1>
</div>
<pre id="relleno"></pre>
<div id="com">
<!-- Esto es un comentario
al Final del código-->
</div>
<script type = "text/javascript"
src="ejemplo41.js">
</script>
</body>
</html>
```

```
DOCTYPE: html
HTML
  HEAD
    #text:
    META charset="UTF-8" id="x"
    #text:
    TITLE
      #text: Ejemplo DOM
    #text:
  #text:
  BODY
    #text: Texto en body
    DIV id="cabecera" class="brillante"
      #text:
      H1
        #text: Prueba
        SPAN class="destacado"
          #text: de DOM
        #text: , para probar
      #text:
    #text:
    PRE id="relleno"
    #text:
    DIV id="com"
      #text:
      #comment: Esto es un comentario al Final del código
      #text:
    #text:
    SCRIPT type="text/javascript" src="ejemplo41.js"
      #text:
```

# Ecmascript: métodos/propiedades de DOM

## ❑ Métodos de DOM

DOM	Tipo nodo de "x"	Método	Valor
CORE	document	document.get <b>Element</b> By <b>Id</b> ( <i>id</i> )	Nodo con ese valor del atributo <i>id</i>
CORE	element	x. <b>parentNode</b>	Nodo padre de x
CORE	document y element	x. <b>nodeType</b>	Tipo del nodo de x
CORE	document y element	x. <b>nodeName</b>	Nombre del nodo de x
CORE	document y element	x. <b>nodeValue</b>	Valor del nodo de x (sólo útil en Text y Comment)
HTML	element	x. <b>innerHTML</b>	Valor (contenido) de la etiqueta HTML del nodo

- ❑ Los métodos de HTML DOM sólo tienen sentido en HTML
- ❑ <https://www.w3.org/TR/dom/#dom-nonelementparentnode-getelementbyid>
  - Pertenece a la interfaz NonElementParentNode (implementada por Document pero NO por Element)
  - Devuelve el primero o null si no lo encuentra
- ❑ <https://www.w3.org/TR/dom/#dom-node-parentnode>
  - Pertenece a la interfaz Node

# Ecmascript: ejemplo parentNode

```
<!DOCTYPE html>
<html>
<body>
<div>
<h1 id="et1">Prueba</h1>
<p id="et2"></p>
</div>
<script>
var x = document.getElementById("et1");
var y = x.parentNode;
var z = document.getElementById("et2");
z.innerHTML = "Tipo: "+x.nodeType+"<br\>Nombre:
"+x.nodeName+"<br\>Valor:
"+x.nodeValue+"<br\>Tipo padre:
"+y.nodeType+"<br\>Nombre padre:
"+y.nodeName+"<br\>Valor padre: "+y.nodeValue;
</script>
</body>
</html>
```

## Prueba

Tipo: 1

Nombre: H1

Valor: null

Tipo padre: 1

Nombre padre: DIV

Valor padre: null

## Ecmascript: propiedades de Core DOM

- Propiedades: Core DOM define propiedades para recorrer el árbol como "nodos"

Tipo nodo de "x"	Propiedad	Valor
document o element	x. <b>childNodes</b>	Nodos <b>hijos</b> de x
document o element	x. <b>children</b>	Nodos <b>hijos</b> de x que sean de tipo Element
element	x. <b>parentNode</b>	Nodo <b>padre</b> de x
...	...	...

- childNodes devuelve más de un nodo, ya que un nodo puede tener más de un nodo hijo (colección de nodos, que es como una tabla)
  - Devuelve los nodos que sean hijos (los atributos NO son hijos)
- children devuelve sólo nodos de tipo Element (no tipo Text, ...)
- parentNode devuelve un nodo, ya que sólo puede haber un nodo padre para cada nodo.

# Ecmascript: ejemplo childNodes/children

```
<!DOCTYPE html>
<html>
<body>
<div>
<h1 id="et1">Hijos <span> y otros hijos</span></h1>
<p id="et2"></p>
</div>
<script>
var x = document.getElementById("et1");
var primNodHijo = x.childNodes[0];
var primEltoHijo = x.children[0];
var z = document.getElementById("et2");
z.innerHTML =
"Tipo: "+x.nodeType+"<br\>Nombre: "+x.nodeName+"<br\>Valor: "+x.nodeValue+
"<br\>Tipo nodo hijo: "+ primNodHijo.nodeType+"<br\>Nombre nodo hijo: "+
primNodHijo.nodeName+"<br\>Valor nodo hijo: "+ primNodHijo.nodeValue+
"<br\>Tipo elemento hijo: "+ primEltoHijo.nodeType+"<br\>Nombre elemento hijo: "+
primEltoHijo.nodeName+"<br\>Valor elemento hijo: "+ primEltoHijo.nodeValue;
</script>
</body>
</html>
```

## Hijos y otros hijos

Tipo: 1

Nombre: H1

Valor: null

Tipo nodo hijo: 3

Nombre nodo hijo: #text

Valor nodo hijo: Hijos

Tipo elemento hijo: 1

Nombre elemento hijo: SPAN

Valor elemento hijo: null

# Ecmascript: métodos de Core DOM

## ❑ Métodos de Core DOM

Tipo nodo de "x"	Método	Valor
document	document.get <b>ElementById</b> ( <i>id</i> )	Nodo con ese valor del atributo "id"
	document.create <b>Element</b> ( <i>nodename</i> )	Crea nodo tipo Element
	document.create <b>TextNode</b> ( <i>text</i> )	Crea nodo tipo Text
document o element	x.get <b>ElementsByClassName</b> ( <i>classname</i> )	Nodos con ese valor del atributo "class"
	x.get <b>ElementsByTagName</b> ( <i>tagname</i> )	Nodos con ese nombre (etiqueta).
	x.appendChild( <i>n</i> )	Añade un nodo hijo "n" a x
	x.removeChild( <i>n</i> )	Elimina un nodo hijo "n" de x
	x.replaceChild( <i>m,n</i> )	Reemplaza un nodo hijo "n" de x por "m"
...	...	...

- ❑ Diferencia entre:
  - getElementBy...
  - getElementsBy...
- ❑ Añadir un nodo:
  - Crear
  - Añadir

```
var h = document.createElement("h1");
var t = document.createTextNode("Hola FAST");
h.appendChild(t);
document.body.appendChild(h);
```

## EcmaScript: HTML DOM

- HTML DOM: define objetos a partir de las etiquetas HTML existentes en la página web leída por el navegador
  - Facilitan el acceso a los nodos al poder acceder a ellos usando el nombre de la etiqueta HTML a que corresponden.

Objetos HTML DOM	Referencia a todos los (de la página)
<b>anchors</b>	Enlaces "<a>" (con atributo name)
<b>forms</b>	Formularios "<form>"
<b>images</b>	Imágenes "<img>"
...	...

- <https://www.w3.org/TR/html/>
- <https://www.w3.org/TR/html/dom>



# Ecmascript: métodos/propiedades

## HTML DOM

---

- Heredan a "document": heredan métodos/propiedades Core DOM
- HTML DOM añade nuevos métodos/propiedades para recorrer el árbol usando los nombres de los elementos HTML.

Tipo nodo	<i>Propiedad/Método</i>	Valor
element	x. <b>innerHTML</b>	Valor (contenido) de la etiqueta HTML del nodo
document	x.get <b>ElementsByName</b> (name)	Elementos HTML cuyo atributo "name" tenga ese valor (devuelve un Array)
...	...	...

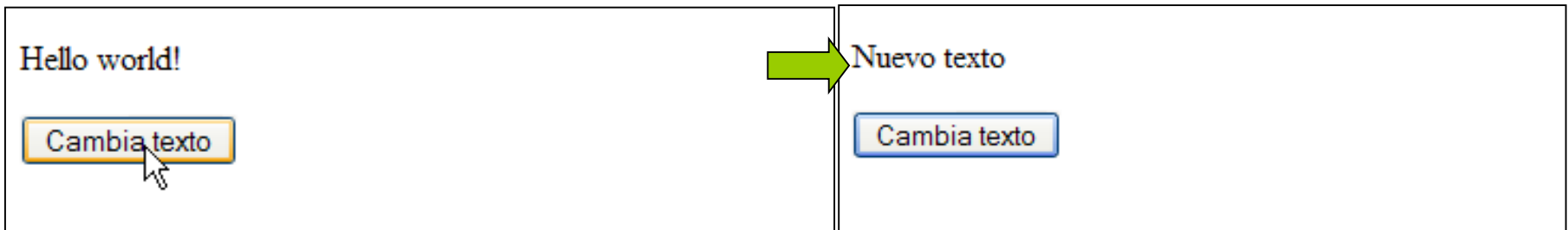
# Ecmascript: ejemplo 1 DOM

## Objetos Core DOM + HTML DOM : Ejemplo 1

```
<html><head>
/* Modifica el texto del párrafo al pulsar el botón */
<script>
function CambiaTexto() {
    document.getElementById("p1").innerHTML="Nuevo texto";
}
</script>
</head><body>
<p id="p1">Hello world!</p>
<input type="button" onclick="CambiaTexto()"
      value="Cambia texto" />
</body></html>
```

Diagram illustrating the DOM structure and the function call:

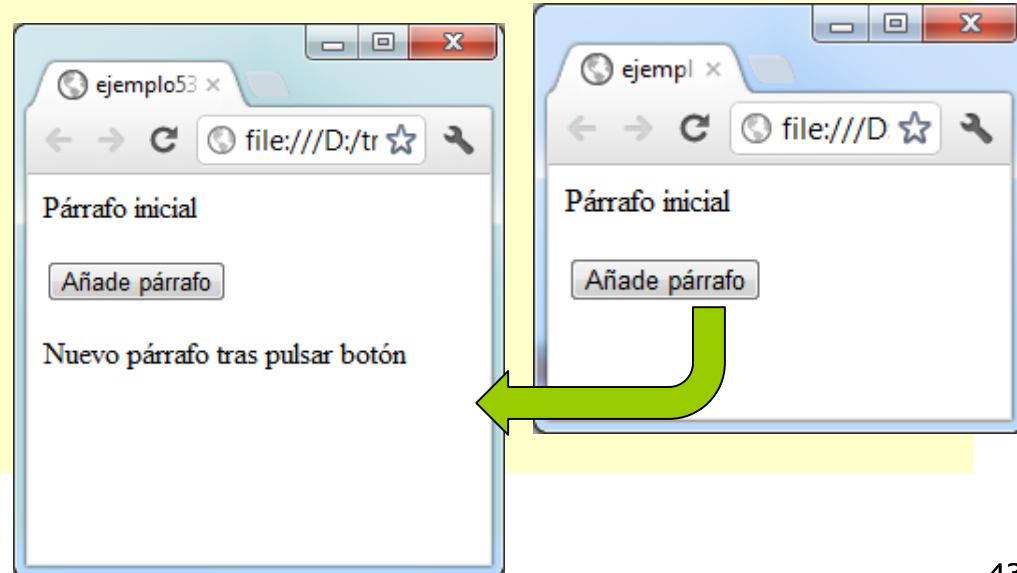
- Objeto Core**: Points to `document` in the code.
- Método Core**: Points to `getElementById` in the code.
- Propiedad HTML DOM**: Points to `innerHTML` in the code.



# EcmaScript: ejemplo 2 DOM

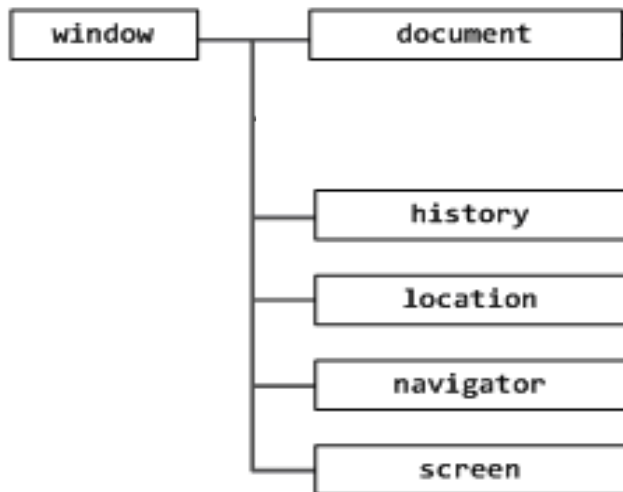
## Objetos Core DOM + HTML DOM : Ejemplo 2

```
<html><head>
<script>
  function AnadeElemento () {
    // Crear nodo de tipo Element
    var parrafo = document.createElement("p");
    // Añadir el nodo Element como hijo de la pagina
    document.body.appendChild(parrafo);
    parrafo.innerHTML="Nuevo Párrafo tras pulsar botón";
  }
</script>
</head>
<body>
<p id="p1">Párrafo inicial</p>
<input type="button"
onclick="AnadeElemento () "
value="Añade párrafo"/>
</body>
</html>
```



## Ecmascript: BOM

- BOM (Browser Object Model): objetos que permiten acceder/modificar propiedades del navegador (navigator, location, ...)
  - Crear, mover, ventanas de navegador.
  - Obtener información sobre el propio navegador.
  - Propiedades de la página actual ...
- Inconveniente: no estandarizado (depende del navegador).



□ El objeto "document" (DOM) descende del objeto "window" (BOM):

□ *"window.document.xxx" equivale "document.xxx"*

# Ecmascript: BOM ejemplo

## ▣ Objetos BOM: Ejemplo de uso del objeto "navigator":

```
<script>
"use strict";
txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
document.write(txt);
</script>
```

The image displays three browser windows side-by-side, each showing the output of the JavaScript code. The first window is Mozilla Firefox, the second is Netscape, and the third is a generic browser showing detailed user agent information.

Browser	Browser CodeName	Browser Name	Browser Version	Cookies Enabled	Platform	User-agent header
Mozilla Firefox	Mozilla	Netscape	5.0 (X11)	true	Linux x86_64	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:58.0) Gecko/20100101 Firefox/58.0
Netscape	Mozilla	Netscape	5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299	true	Win32	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36 Edge/16.16299
Generic Browser	Mozilla	Netscape	5.0 (X11; Linux x86_64) AppleWebKit/605.1 (KHTML, like Gecko) Version/11.0 Safari/605.1	true	Linux x86_64	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/605.1 (KHTML, like Gecko) Version/11.0 Safari/605.1

## Ecmascript: Sintaxis, Funciones Predefinidas

---

- ❑ Existen en el lenguaje muchas funciones predefinidas.
- ❑ **Ejemplo:** funciones de "popup boxes" (ventanas emergentes)
  - alert: "OK" para continuar
  - confirm: "OK" (true) o "Cancel" (false)
  - prompt: "OK" (valor introducido) o "Cancel" (null)

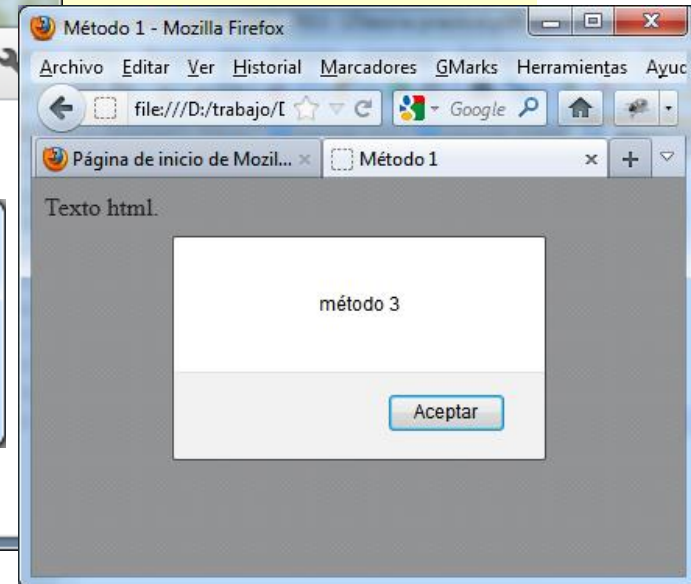
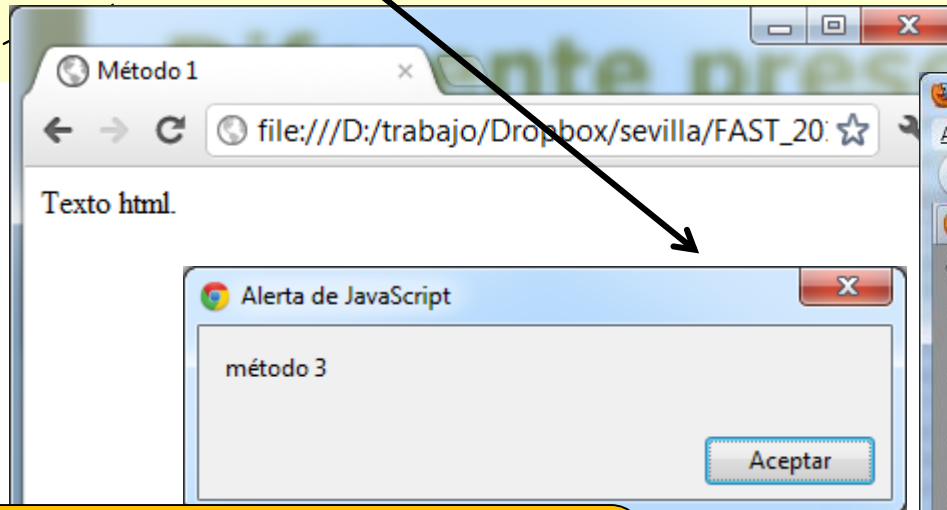
*Estas funciones son métodos de objetos definidos en el modelo BOM. Aquellas que no usan un objeto en su invocación están referidas al objeto "window" de dicho modelo.*

*Por lo tanto, **son equivalentes:***

***window.alert("Hola")***  
***alert("Hola").***

## EcmaScript: Sintaxis, Funciones Predefinidas

```
<html> <head>
<title> Ejemplo 25</title></head>
<body><script>
    alert("método 3");
</script>
</body>
```



*Diferente representación  
según el navegador*

# AJAX: Introducción (1)

---

- ❑ AJAX: **A**synchronous **J**avaScript and **X**ML.
- ❑ No es una tecnología o un lenguaje sino el uso conjunto de varias tecnologías de desarrollo web
- ❑ Se usan en el lado del cliente, y su objetivo es crear aplicaciones web asíncronas
  - ❑ Es posible actualizar partes de la página:
    - Sin recargar la página completa.
  - ❑ Permite:
    - Mientras se actualiza, interactuar con la página
  - ❑ El intercambio se realiza en:
    - segundo plano



# AJAX: Introducción (2)

---

- ❑ **Orígenes**
- ❑ **1998**: Primeros trabajos de Microsoft
  - 1999, crea control ActiveX XMLHttpRequest
- ❑ **2002**: más generalizado, otros navegadores empiezan a soportarlo como objeto JavaScript
- ❑ **2004-2005**: Google lo utiliza en Gmail y Google maps.
- ❑ **2005**: el termino Ajax aparece en un artículo de James Garrett titulado "Ajax: A New Approach to Web Applications"
- ❑ **2006**: W3C publica primera especificación del objeto ***XMLHttpRequest***.
- ❑ **2006**: Microsoft da soporte al objeto *XMLHttpRequest*.
- ❑ Aplicaciones lo que usan: Google Maps, Gmail, Youtube, Facebook, Amazon, Flickr, Panoramio, Kayak, Yahoo,...

# AJAX: tecnologías que usa

---

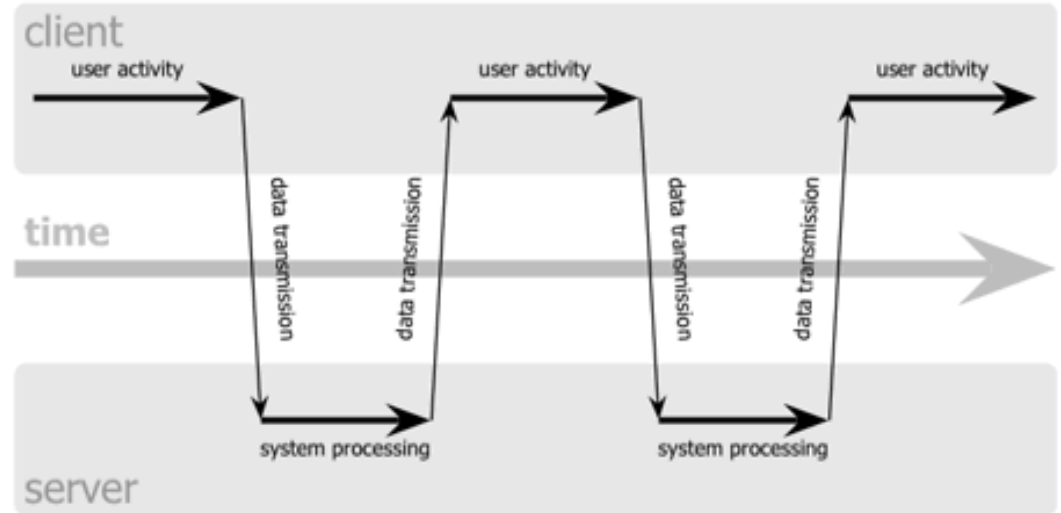
- ❑ **XHTML y CSS.**
- ❑ **DOM.**
- ❑ **XML** (*o JSON*).
- ❑ **API (objeto) XMLHttpRequest** para intercambio asíncrono de información.
- ❑ **ECMAScript:** unir todas las demás tecnologías.

La especificación (definida por el W3C) de **XMLHttpRequest** define una API que proporciona facilidades a un script en el cliente para transferir datos entre cliente y servidor.

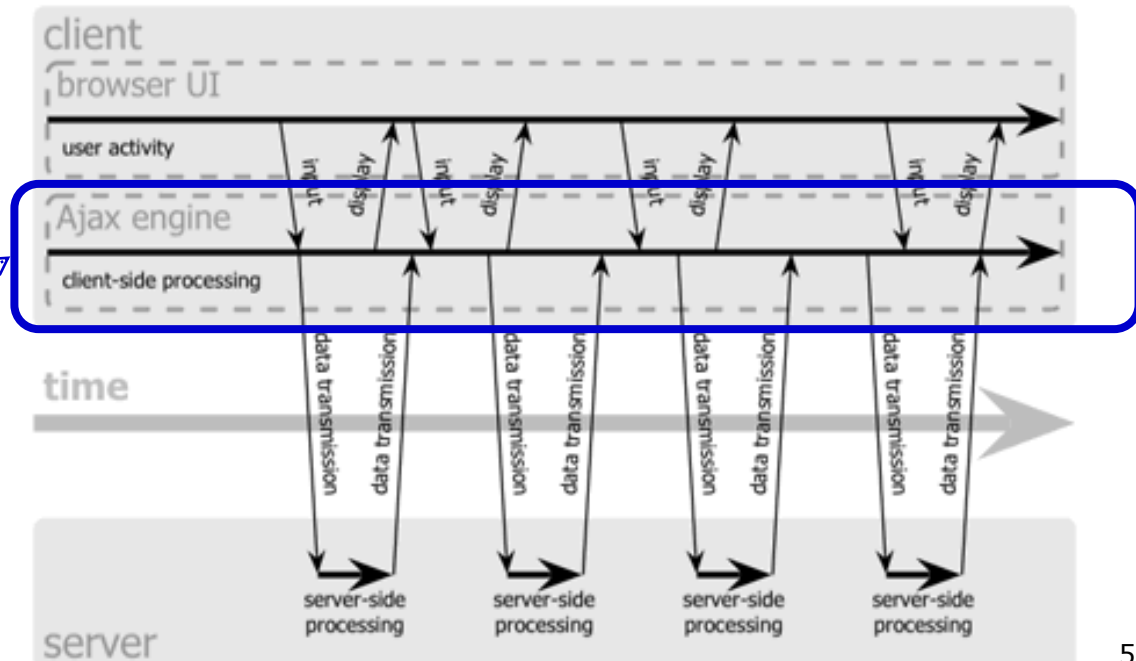
# AJAX: Modelo

- Mejora interacción usuario, evitando recargas de la página: intercambio de información en segundo plano (e.g. Google)
- ¿Cómo? Creando un elemento intermedio entre el usuario y el servidor (Motor AJAX).

classic web application model (synchronous)



Ajax web application model (asynchronous)



# AJAX: Uso habitual

---

## 1. Crear objeto XMLHttpRequest

```
var xmlhttp = new XMLHttpRequest();
```

## 2. Preparar la petición

*open(Método, url, async)*

```
xmlhttp.open("GET", "ajax_info.txt", true);
```

## 3. Asociar función a cambio en el estado de la petición

```
xmlhttp.onreadystatechange = function() {  
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {  
    //Respuesta recibida completamente (4) y sin  
    //errores del servidor (codigo HTTP 200)  
  }  
}
```

## 4. Enviar la petición

```
xmlhttp.send();
```

*Prepara función de callback*

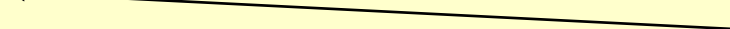
*Envía el mensaje HTTP*

## AJAX: ejemplo

### ▣ Objetos Core DOM + HTML DOM : Ejemplo 2

```
<script>
"use strict";
function loadTextDoc(url) {
    var xmlhttp;
    var txt;
    xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            txt = xmlhttp.responseText; // alert(txt);
            document.getElementById('Info').innerHTML = txt;
        }
    }
    xmlhttp.open("GET", url, true);
    xmlhttp.send();
}
```

El envío el mensaje  
HTTP debe ser lo  
último



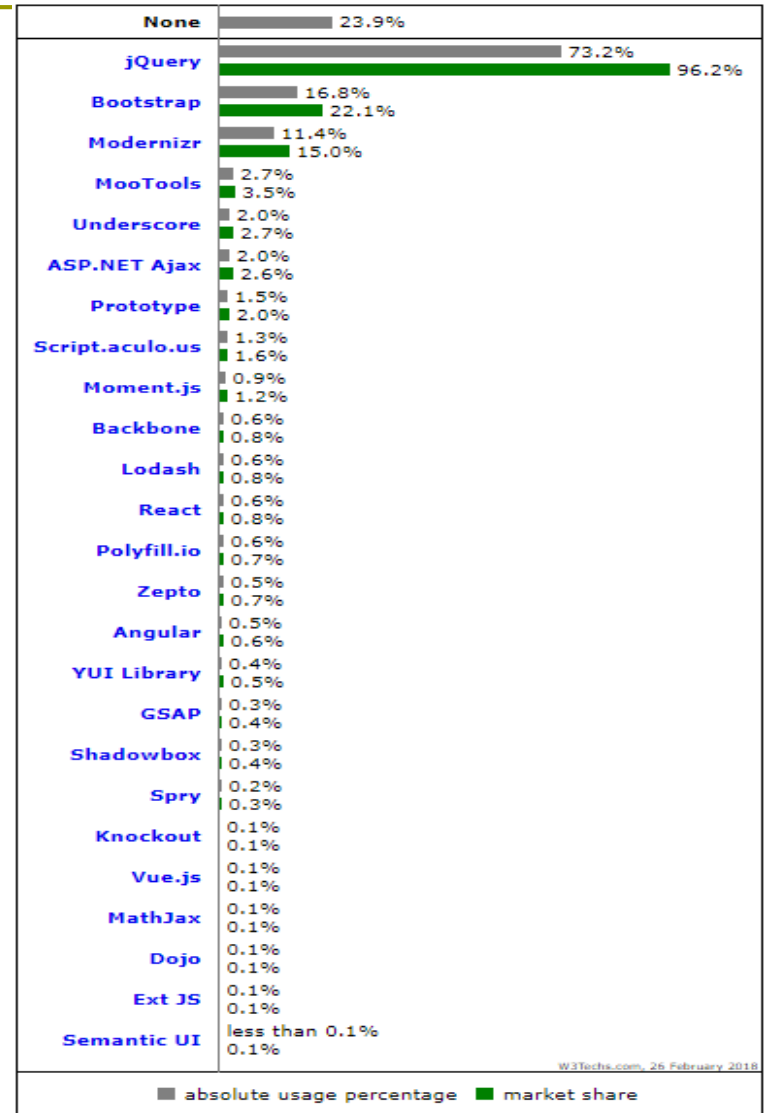
# Ecmascript: Librerías ECMAScript externas

- Permiten ampliar la funcionalidad del Core ECMAScript
  - Se suministran en ficheros ".js"
- jQuery es la mas usada con diferencia:

*Estadística referida al número de "aplicaciones web" escritas en estos lenguajes, SIN tener en cuenta el tamaño ni tráfico de la página*

```
<script src="jquery.js"></script>
```

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery
/1/jquery.min.js"></script>
```



# Resumen/Conclusiones/Referencias

---

- ❑ Tener una visión global de los lenguajes de programación en el lado del cliente
- ❑ Aplicación práctica con uno de ellos: ECMAScript
  - ❑ Uso y características del lenguaje
  - ❑ Fundamentos del modelo DOM y BOM
  - ❑ Ajax
- ❑ Referencias:
  - Estándar ECMA-262:
    - ❑ <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
  - Uso práctico:
    - ❑ <http://www.w3schools.com/js/>
    - ❑ <https://www.w3schools.com/jsref/default.asp>
  - Estándar DOM:
    - ❑ <https://www.w3.org/TR/dom>
  - Uso práctico
    - ❑ [https://www.w3schools.com/jsref/dom\\_obj\\_document.asp](https://www.w3schools.com/jsref/dom_obj_document.asp)