

**Módulo Segundo**  
*Conceptos básicos de  
Javascript*

# Técnicas Avanzadas Web

## Modulo 2: Javascript

---

- Javascript != Java
- Creado por Bredan Eich para Navigator 2.0
- Lenguaje ligero **interpretado** en navegadores web (aunque no exclusivamente).
- Javascript es una implementación de ECMAScript (lenguaje de scripting basado en prototipos)

- INCLUSIÓN en (X)HTML:

- Inline en el documento:

```
<script type="text/javascript">  
//<br/>// código....<br/>//]]&gt;<br/>&lt;/script&gt;</pre></div><div data-bbox="225 653 850 740" data-label="List-Group"><ul><li>- Separando estructura(xhtml) / diseño(css) y ahora programación(js)</li></ul></div><div data-bbox="219 795 632 858" data-label="Text"><pre>&lt;script type="text/javascript"<br/>src="scripts.js"&gt;&lt;/script&gt;</pre></div><div data-bbox="62 562 117 972" data-label="Page-Footer"><p><b>IRONTEC</b><br/>INTERNET Y SISTEMAS SOBRE GNU/LINUX</p></div><div data-bbox="911 931 928 957" data-label="Page-Footer"><p>3</p></div><div data-bbox="747 962 917 986" data-label="Page-Footer"><p>Javier Infante Porro</p></div>
```

- Tipos básicos de datos

```
var intValue = 1;
```

```
var floatValue = 3.0;
```

```
var stringValue = "This is a string\n";
```

```
var sqString = 'This is also a string';
```

- Javascript es un lenguaje con tipo de datos dinámicos
- Las variables son declaradas con la palabra clave “var”
- Todos los tipos simples de datos están soportados

- Arrays: Un array es un objeto interno de Javascript.

```
var cadena = new Array();  
var cadena = [2,'33',123];
```

- Mapas de Propiedades

```
var MAPA = {propiedad : "valor", otra : "valor2"};  
alert(MAPA.propiedad); // valor  
alert(MAPA["otra"]);
```

```
delete MAPA.prop4;
```

```
for (var key in MAPA) alert(MAPA[key]);
```

- Funciones

- Una función en javascript es un objeto: puede ser creado dinámicamente, almacenado, pasado y devuelto como cualquier otro valor.

```
function miMiniFuncion(arg1,arg2) {  
    alert("Hola "+arg1+" de "+arg2);  
}  
miMiniFuncion("Jose", "Valencia");
```

```
var miFuncion = function(arg1,arg2) {  
    alert("Hola "+arg1+" de "+arg2);  
}  
miFuncion("Pedro", "Cuenca");
```

- Objetos
  - Javascript Orientación a Objetos basada en prototipos (no en clases).
  - Es necesario un objeto constructor, que no es otra cosa que una función normal.
  - palabra reservada “**this**”, hace referencia a la instancia en sí.

```
function Calculadora(o1,o2) {  
    this.op1 = o1;  
    this.op2 = o2;  
}  
var cal = new Calculadora(6,4);  
alert(cal.op1);
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- Es necesario que un objeto tenga métodos, y no sólo propiedades: PROTOTYPE:

```
Calculadora.prototype.suma() {  
    return this.op1+this.op2;  
}
```

```
var cal = new Calculadora(6,4);
```

```
alert(cal.op1+" "+cal.op2+"="cal.suma());
```

- DOM (Document Object Model)
  - Interfaz independiente de la plataforma que accede y puede modificar dinámicamente la estructura y el estilo de documentos.
  - Existe un “estandar”: W3C DOM
    - implementado “parcialmente” por casi todos los navegadores comerciales.
  - Los documentos HTML poseen una estructura hereditaria representada como una estructura de **árbol**.
  - Conjunto de propiedades y métodos para recorrer árboles XML (HTML), y examinar o modificar sus nodos.

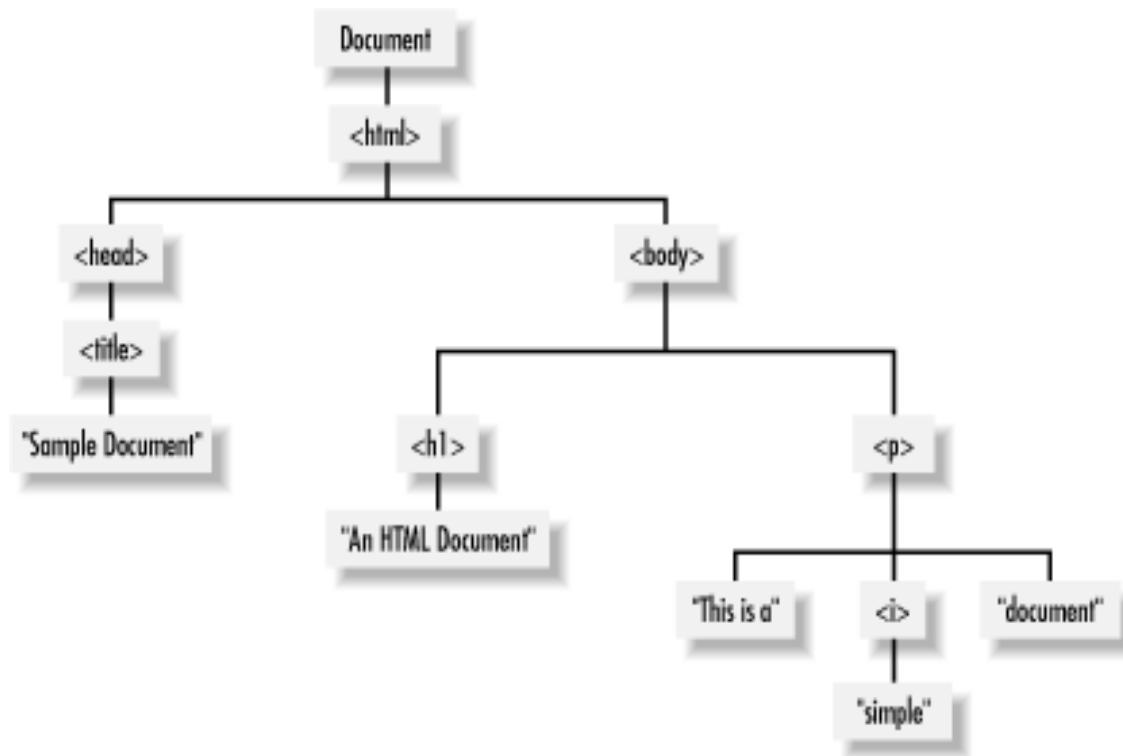
- Partiendo de este código HTML:

```
<html>
  <head>
    <title>Sample Document</title>
  </head>
  <body>
    <h1>An HTML Document</h2>
    <p>This is a <i>simple</i> document.
  </body>
</html>
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

- Ésta sería su representación en forma de árbol:



- **Nodo document**
  - Nodo del que cuelgan todos los elementos de un documento HTML. (éstos, heredarán sus métodos).
  - **Métodos** (más útiles)
    - **document.getElementById(iden)**
      - Devuelve 1 único elemento identificado por el atributo único “id”;
    - **document.getElementsByTagName(“tag”)**
      - Devuelve un array con todos los nodos cuyo tag coincida con “tag”.

- **Tipos de Nodos (NodeType)**

- Elemento (1)

```
<p>Hola</p>
```

- Texto (3)

```
<p>Hola</p>
```

- Atributo (2)

```
<p class="estilo">Hola</p>
```

- **Propiedades**

- **attributes**

- Devuelve array con los atributos de un nodo.

HTML:

```
<p id="par1" class="est_par">ejemplo</p>
```

JS:

```
alert(document.getElementById("par1").attributes.length); // 2
```

- **childNodes**

- Devuelve array con los nodos hijos.

HTML:

```
<p id="p1"><strong>Hola</strong><em>¿que tal?</em>
```

JS:

```
alert(document.getElementById("p1").childNodes.length); //
```

### – data

- Devuelve el contenido dentro de un nodo. (tener en cuenta que un texto, está a su vez dentro de un nodo)

HTML:

```
<p id="p1">hola que tal?</p>
```

JS:

```
alert(document.getElementById("p1").firstChild.data); // alert->hola que tal?
```

### – firstChild

- Devuelve el primer nodo hijo.

HTML:

```
<p id="p1"><strong>hola</strong> que tal?</p>
```

JS:

```
document.getElementById("p1").firstChild.firstChild.data="adios";
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### – lastChild

- Devuelve el último nodo hijo.

HTML:

```
<p id="p1"><strong>hola</strong> ¿que tal?</p>
```

JS:

```
document.getElementById("p1").lastChild.data="  
¿como estás?";
```

### – nextSibling

- Devuelve el siguiente nodo hermano.

HTML:

```
<p id="p1">hola</p><p id="p2">que tal?</p>
```

JS:

```
alert(document.getElementById("p1").nextSibling.fi  
rstChild.data); // alert->que tal?
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### – nodeName

- Devuelve el nombre de un nodo.

HTML:

```
<p id="p1"><strong>hola</strong> ¿que tal?</p>
```

JS:

```
p = document.getElementById("p1");  
alert(p.firstChild.nodeName); // strong  
alert(p.lastChild.nodeName); // #text
```

### – nodeType

- Devuelve el tipo de un nodo.

HTML:

```
<p id="p1"><strong>hola</strong> ¿que tal?</p>
```

JS:

```
p = document.getElementById("p1");  
alert(p.firstChild.nodeType); // 1  
alert(p.lastChild.nodeType); // 3
```

### – **nodeValue**

- Devuelve para nodos de texto y atributos, el valor del nodo. Para nodos elementos, NULL.

HTML:

```
<p id="p1"><strong>hola</strong> ¿que tal?</p>
```

JS:

```
p = document.getElementById("p1");  
alert(p.firstChild.nodeValue); // NULL  
alert(p.lastChild.nodeValue); // ¿que tal?
```

### – **parentNode**

- Devuelve el Padre de un nodo.

HTML:

```
<p><strong id="s1">hola</strong> ¿que tal?</p>
```

JS:

```
alert(document.getElementById("s1").parentNode.lastChild.nodeValue); // ¿que tal?
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### – previousSibling

- Devuelve el nodo hermano anterior.

HTML:

```
<p id="p1">hola</p><p id="p2">que tal?</p>
```

JS:

```
alert(document.getElementById("p2").previousSibling.firstChild.nodeValue); // hola
```

- Métodos
  - **appendChild()**
    - Inserta un nodo hijo en la última posición.

HTML:

```
<p id="p1"><strong>hola</strong></p>
```

JS:

```
var cursiva = document.createElement("em");  
var texto = document.createTextNode("adios");  
cursiva.appendChild(texto);  
document.appendChild(cursiva);
```

### – **appendData()**

- Inserta un “data” (texto) al final del data de una nodo.

HTML:

```
<p id="p1">hola</p>
```

JS:

```
document.getElementById("p1").firstChild.appendData(" ¿que tal?");
```

### – **cloneNode()**

- Clona un nodo a otro nodo.

HTML:

```
<p id="p1"><strong>hola</strong></p>
```

JS:

```
p = document.getElementById("p1");  
var st2 = p.firstChild.cloneNode(true);  
p.appendChild(st2);
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### – **getAttribute()**

- Devuelve un atributo determinado.

HTML:

```
<p id="p1" class="estilo_parrado">hola</p>
```

JS:

```
document.getElementById("p1").getAttribute("class"); // estilo_parrafo
```

### – **setAttribute()**

- introduce / modifica un atributo de un nodo.

HTML:

```
<p id="p1" class="estilo_parrafo">hola</p>
```

JS:

```
document.getElementById("p1").setAttribute("class","estilo_parrafo2");
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### – insertBefore()

- Inserta un nodo, bajo un nodo padre, antes de un nodo hermano.

HTML:

```
<p id="p1"><strong id="s1">hola</strong></p>
```

JS:

```
var o = document.createElement("em");  
o.appendChild(document.createTextNode("adios"));  
document.getElementById("p1").insertBefore(o, document.  
getElementById("s1"));
```

### – removeChild()

- Elimina un nodo

HTML:

```
<p id="p"><strong id="s">hola</strong>que tal</p>
```

JS:

```
document.getElementById("p1").removeChild(document.  
getElementById("s1"));
```

### – **hasChildNodes()**

- Devuelve valor booleano (si un nodo tiene hijos o no)

**HTML:**

```
<p id="p1"><strong>hola</strong> que tal</p>
```

**JS:**

```
p = document.getElementById("p1");  
p.firstChild.hasChildNodes(); // true  
p.lastChild.hasChildNodes(); // false
```

- Manejadores de Eventos
  - EVENTO: método que se activa tras la **interactuación** del usuario o del navegador.
  - Son la esencia principal del Javascript; sin eventos, no sirve de nada.
  - Existen diferentes implementaciones del sistema de eventos entre todas los navegadores del mercado.
- TIPOS DE EVENTOS
  - **TECLADO**
    - keydown / keypress /keydown

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- **RATÓN**
  - click
  - dblclick
  - mouseover
  - mousemove
- **INTERFAZ**
  - load / unload
  - resize / scroll
  - focus / blur
- **FORMULARIO**
- submit / reset
- change / select

- Registrando Eventos
  - Método tradicional inline
    - Se trata de añadir un atributo a los nodos HTML.
    - El nombre de ese atributo será el evento que se quiere disparar en ese nodo concreto, con el prefijo “on”.

```
<input type="text" onclick="javascript:go();" />
```

- **DESACONSEJADO.**
  - difícil mantenimiento.
  - rompe la regla de separar diseño / funcionalidad.

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- Tradicional

- Asignando funciones a objetos:

```
elemento.onclick = nombre_funcion;
```

```
elemento.onclick = funcion () { //codigo }
```

- W3C (no funciona en IE)

- addEventListener

```
elemento.addEventListener('click',funcion,true);
```

- Microsoft

- attachEvent

```
elemento.attachEvent ('onclick',funcion);
```

- Propiedades de un Evento
  - Al asignar una función a un evento determinado, tendremos acceso a un objeto de tipo evento con información muy valiosa del mismo (sobre que objeto ha saltado el evento, que botón del ratón se ha pulsado, que letra has pulsado...)
  - Dependiendo del tipo de evento, el objeto evento tendrá unas propiedades u otras.
  - El acceso al objeto evento, así como sus propiedades son dependientes del navegador.

- Accediendo al evento:
  - W3C:
    - Depende del tercer parámetro en el método *addEventListener*
    - *El método recibe como argumento el evento en sí:*

```
function haz_click(e) {  
    alert(e.type);  
}
```

```
document.getElementById("boton1").addEventListener  
("click", haz_click, false);
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- Microsoft IE
  - El evento esta disponible en una propiedad del objeto window:

```
function haz_click() {  
    alert(window.event.type);  
}
```

```
document.getElementById("boton1").attachEvent("onclick",haz_click);
```

- Propiedades más comunes:
  - e.target (W3C) / e.srcElement (IE)
    - Sin duda, la propiedad más importante que nos trae el evento: El objeto origen sobre el que se ha disparado.
    - W3C:

```
function haz_click(e) {  
    alert(e.target.value); // alerta-> GO!  
}  
--  
document.getElementById("boton1").addEventListener  
("click", haz_click, false);  
--  
<input type="button" id="boton1" value="GO!" />
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

– Microsoft IE

```
function haz_click() {  
    alert(window.event.srcElement.value); //  
    alerta-> GO!  
}  
--  
document.getElementById("boton1").attachEvent("onclick", haz_click);  
--  
<input type="button" id="boton1" value="GO!" />
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- e.keyCode
  - Devuelve el código ASCII de la tecla pulsada.
- e.pageX / e.pageY (No IE)
  - Devuelve las coordenadas x e y del puntero del ratón relativas al documento.
- e.screenX / e.screenY
  - Devuelve las coordenadas x e y del puntero del ratón relativas a la pantalla.
- e.button
  - Identifica qué botón del ratón se ha pulsado:
    - IE: 1(izda) / 2(dcha) / 4(centro)
    - W3C: 0(izda) / 1(centro) / 2(dcha)

- XMLHttpRequest
  - Objeto interno Javascript, encargado de hacer peticiones al servidor web y procesar la salida de éste.
  - Creado como control ActiveX por MS, esta implementado nativamente en otros navegadores (y realiza el mismo trabajo).
  - No es un objeto nuevo, simplemente ha sido puesto de moda por Gmail y la moda WEB2.0.

- Instanciando el objeto

- Nativamente:

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

- ActiveX

```
var oXML = new ActiveXObject("Microsoft.XMLHTTP");
```

- CrossBrowser

```
if (window.XMLHttpRequest)
    var oXML = new XMLHttpRequest();
else if (window.ActiveXObject)
    var oXML = new
    ActiveXObject("Microsoft.XMLHTTP");
```

- Métodos:
  - **abort()** - Detiene la petición en curso.
  - **getAllResponseHeaders()** - Devuelve todas las cabeceras de la respuesta (etiquetas y valores) como una cadena.
  - **getResponseHeader(etiqueta)** - Devuelve el valor de la etiqueta en las cabeceras de la respuesta.
  - **open(método,URL,asíncrona,nombre,password)**
    - Abre una conexión con esa URL mediante ese método (GET o POST), aplicando los valores opcionales de la derecha.
  - **send(contenido)** - Envía la petición.
  - **setRequestHeader(etiqueta,valor)** - Establece el valor de las cabeceras de petición.

- Propiedades
  - **onreadystatechange** - Contiene el nombre de la función a ejecutar cuando el estado de la conexión cambie.
  - **readyState** - Estado de la conexión:
    - 0 : No iniciado
    - 1: Cargando
    - 2: Cargado pero sin incorporar el contenido a los objetos correspondientes
    - 3: Incorporando contenido a los objetos correspondientes
    - 4: Carga completada

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- **responseText** - Datos devueltos por el servidor en formato cadena.
- **responseXML** - Datos devueltos por el servidor en forma de documento XML que puede ser recorrido mediante las funciones del DOM.
- **status** - Código enviado por el servidor, del tipo 404 (documento no encontrado) o 200 (OK).
- **statusText** - Mensaje de texto enviado por el servidor junto al código (status), para el caso de código 200 contendrá "OK"

### - Ejemplo GET

```
if (window.XMLHttpRequest)
    var oXML = new XMLHttpRequest();
else if (window.ActiveXObject)
    var oXML = new
ActiveXObject("Microsoft.XMLHTTP");

oXML.open("GET", "llamando.php?op1=3");
oXML.onreadystatechange = cambiaestado;
oXML.send(null);

function cambiaestado() {
    if (oXML.readyState==4)
        alert(oXML.responseText);
}
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

### - Ejemplo POST

```
// creación oXML...
var str = "op1=valor1&op2=valor2";

oXML.open("POST", "llamando.php");
oXML.onreadystatechange = cambiaestado;
oXML.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
oXML.setRequestHeader("Content-Length",
str.length);
oXML.send(str);
function cambiaestado() {
    if (oXML.readyState==4)
        alert(oXML.responseText);
}
```

# Técnicas Avanzadas Web

## Módulo 2: Javascript

---

- Enlaces interesantes
  - <http://www.google.com>
  - <http://alistapart.com/>
  - <http://openrico.org/>
  - <http://www.twinhelix.com/>
  - <http://www.alvit.de/web-dev/index.html>
  - <http://www.onlinetools.org/articles/unobtrusivejavascript/>
  - [http://www.irontec.com/~jabi/mod2\\_javascript.pdf](http://www.irontec.com/~jabi/mod2_javascript.pdf)