Desarrollo de Aplicaciones Web

JavaScript. HTML DOM.

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Pablo García Tahoces

HTML DOM

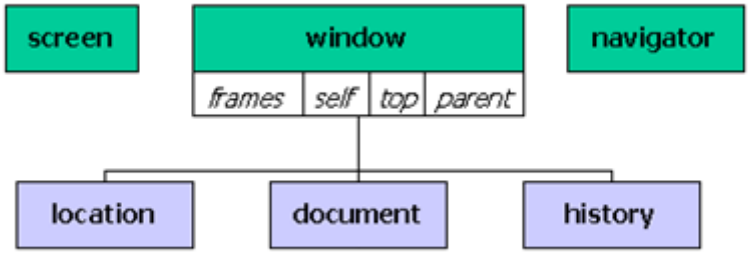
W3C, a través del "*Document Object Model*" (DOM), define un mecanismo estándar de acceso a documentos HTML y XML, permitiendo de esta manera que programas y scripts puedan modificar su contenido, estructura y/o estilo.

W3C define tres partes separadas:

- **Core DOM** → modelo estándar para cualquier documento estructurado
- **XML DOM** → modelo estándar para cualquier documento XML
- **HTML DOM** → modelo estándar para cualquier documento HTML

HTML DOM

Mediante JavaScript, es posible tener acceso a los diferentes objetos relacionados con un documento HTML, permitiendo la realización de todo tipo de modificaciones.



HTML DOM

Objeto Screen →Permite obtener información sobre la pantalla en que se está ejecutando el código JavaScript.

Propiedades:

- availHeight
- availWidth
- colorDepth
- height
- pixelDepth
- width

Métodos:

HTML DOM

Objeto Navigator → Permite obtener información sobre el navegador en que se está ejecutando el código JavaScript.

Propiedades:

- appCodeName
- appName
- appVersion
- userAgent
- plugins
- mimeTypes
- …

Métodos:

- javaEnabled()
- …

HTML DOM

Objeto Window → Representa cualquier ventana abierta por el navegador.

Propiedades:

- name
- closed
- length
- self
- parent
- opener
- top
- status
- defaultStatus
- location
- …

Métodos:

- alert()
- setTimeout()
- setInterval()
- clearTimeout()
- prompt()
- confirm()
- blur()
- focus()
- close()
- scroll()
- open()
- …

HTML DOM

Objeto Location → Contiene información referente a la localización del documento que se muestra.

Propiedades:

- href
- hash
- port
- hostname
- host
- protocol
- pathname
- search

Métodos:

- assign()
- reload()
- replace()

HTML DOM

Objeto History → Permite navegar por el histórico de páginas visitadas.

Propiedades:
- length

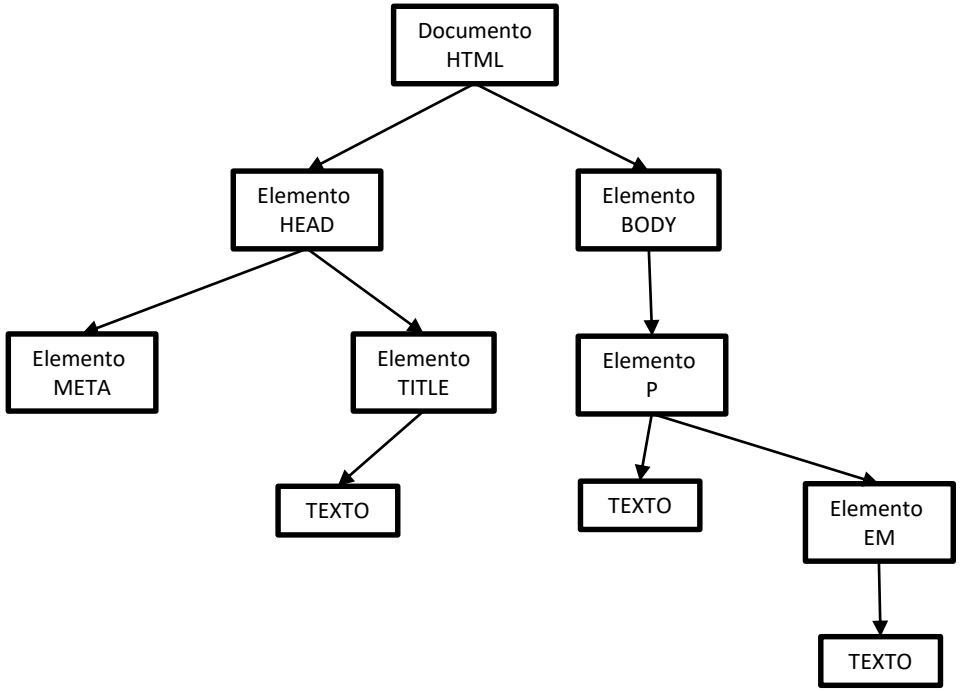
Métodos:
- back()
- forward()
- go()

HTML DOM

Objeto Document → Representa al propio documento que se está mostrando en la ventana del navegador.

Basados en HTML DOM, los navegadores transforman automáticamente las etiquetas de un documento HTML en una estructura arborescente.

```
<html>
  <head>
     <meta  ----------------- />
     <title> ------------------- </title>
  </head>
  <body>
      <p> ------ <em> ----- </em></p>
  </body>
</html>
```



Pablo García Tahoces

HTML DOM

Mediante DOM, JavaScript obtiene capacidad completa para crear todo tipo de efectos DHTML y así poder:

- Cambiar todos las etiquetas HTML de la página.
- Cambiar todos los atributos HTML de la página.
- Cambiar todos los estilos CSS de la página.
- Eliminar etiquetas y atributos HTML.
- Añadir nuevas etiquetas y atributos HTML.
- Reaccionar a todos los eventos HTML de la página.
- Crear nuevos eventos HTML en la página.

HTML DOM

La transformación automática de la página en un árbol sigue las reglas:

- Las etiquetas HTML se transforman en dos nodos:
  - La propia etiqueta.
  - Texto, que aparece como hijo de nodo que representa la etiqueta en cuestión.
- Si una etiqueta HTML se encuentra dentro de otra, se sigue el procedimiento anterior, siendo los nodos generados hijos del que representa a la etiqueta en cuestión.

HTML DOM

HTML DOM define métodos y propiedades para acceder/modificar cada uno de los nodos del árbol generado. Algunos de estos son:

- *getElementByTagName(nombreEtiqueta)* → devuelve un array con todos los elementos de la página cuya etiqueta sea igual al parámetro pasado.
- *getElementByName(nombre)* → devuelve el elemento HTML cuyo atributo name coincide con el parámetro pasado.
- *getElementById(nombre)* → devuelve el elemento HTML cuyo atributo id coincide con el parámetro pasado.
- write(txt) → escribe el texto “txt” en la posición del documento en que ha sido invocado
- innerHTML → propiedad que almacena el contenido de una etiqueta HTML.
- style → permite modificar el estilo del elemento asociado.

HTML DOM

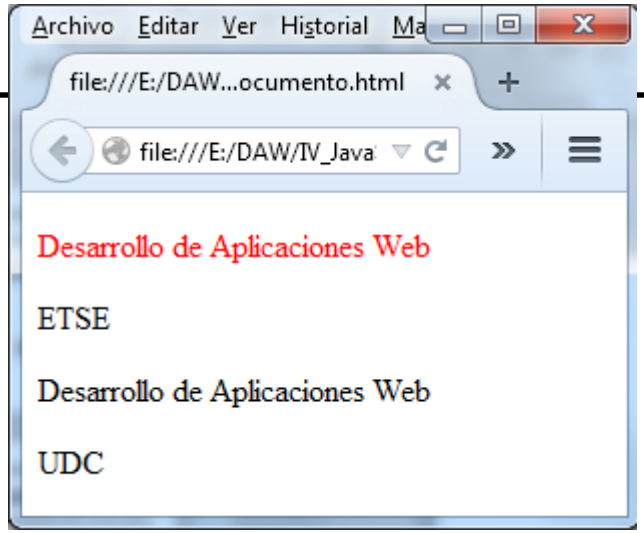
HTML DOM define métodos y propiedades para acceder/modificar cada uno de los nodos del árbol generado. Algunos de estos son:

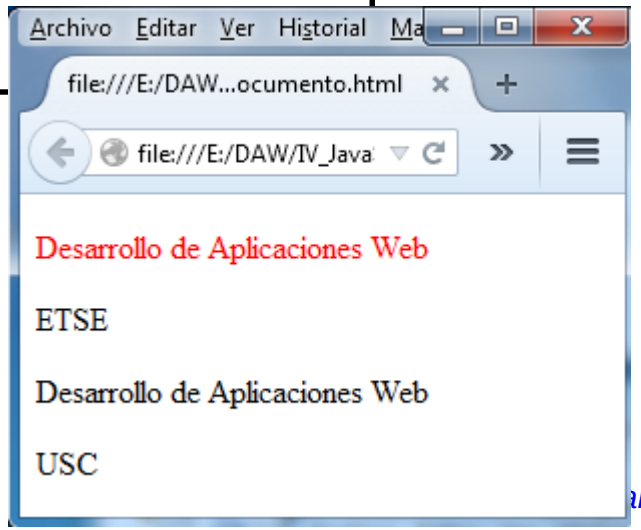
- *getAttribute(atrributo)* → Recupera el valor de un atributo
- *setAttribute(atributo, valor)* → Añade nuevos atributos a un elemento seleccionado o modifica su valor.
- *removeAttribute(atributo)* → Elimina un atributo de un elemento

```
<script>
        var enlace = document.getElementById("miEnlace");
        enlace.setAttribute("href", "https://www.usc.es");
</script>
```

HTML DOM

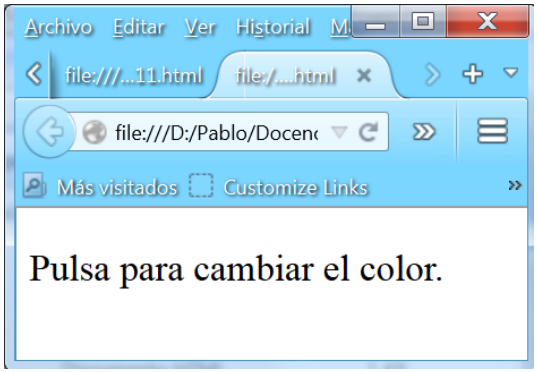
```
<!DOCTYPE html>
<html>
    <body>

        <p id="daw">Desarrollo de Aplicaciones Web</p>
        <p id="usc"></p>

        <script>
                var txt=document.getElementById("daw").innerHTML;
                document.write(txt);
                document.getElementById("daw").style.color= "red";
                document.getElementById("usc").innerHTML= "ETSE";
        </script>

        <p onclick="this.innerHTML='USC'">UDC</p>

    </body>
</html>
```

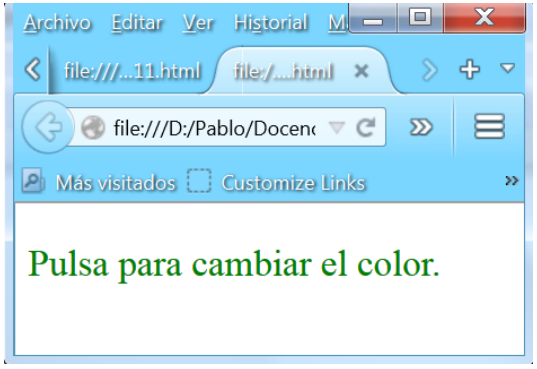




HTML DOM

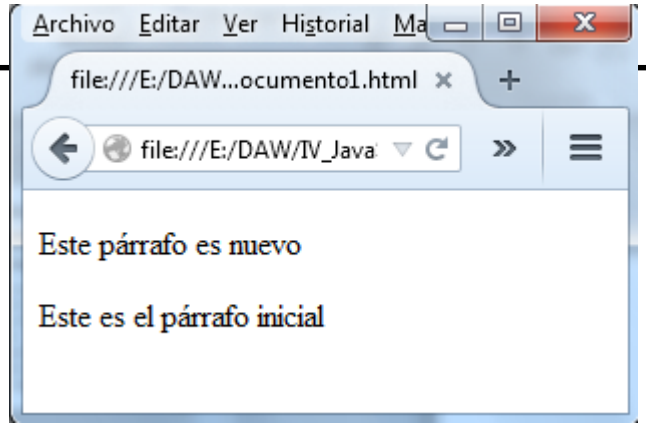
```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function miFuncion() {
                document.getElementById("daw").style.color= "green";
            }
        </script>
    </head>
    <body>
        <p id="daw" onclick="miFuncion()">Pulsa para cambiar el color.</p>
    </body>
</html>
```





HTML DOM

```
<!DOCTYPE html>
<html>
    <body>
        <div id="div1">
            <p id="p1">Este es el párrafo inicial</p>
        </div>
        <script>
            var et1= document.createElement("p");
            var txt1= document.createTextNode("Este párrafo es nuevo");
            et1.appendChild(txt1);

            var et2= document.getElementById("div1");
            var et3= document.getElementById("p1");
            et2.insertBefore(et1, et3);
        </script>
    </body>
</html>
```



javaScript vs HTML5

HTML5 incorpora nuevos mecanismo de acceso a los nodos DOM, basado en nuevos métodos que incorporan selectores de las reglas CSS:

- *querySelector(CSS_Selector)* → devuelve el primer elemento de la página coincidente con el parámetro pasado.
- *querySelectorAll(CSS_Selector)* → devuelve todos los elementos de la página coincidententes con el parámetro pasado.

Ej.-

```
var x= document.querySelector(“.usoClass1”);

var y= document.querySelector(“a:visited”);
```

javaScript vs HTML5

```
<!DOCTYPE html>
<html>
    <body>
        <p class="daw">ETSE</p>
        <p class="daw">Desarrollo de Aplicaciones Web</p>

        <script>
            document.querySelector(".daw").style.color= "red";
        </script>
    </body>
</html>
```

javaScript vs HTML5

Objetos para almacenamiento local → permiten el almacenamiento de datos compartidos por las diferentes páginas del sitio web, en el lado cliente.

Los datos son almacenados en el navegador.

Es un mecanismo alternativo a las cookies.

Existen dos objetos diferentes:

- localStorage → los datos se almacenan sin fecha de expiración.
- sessionStorage → los datos se almacenan para la sesión → Se pierden al cerrar la ventana del navegador.

javaScript vs HTML5

Objetos para almacenamiento local → permiten el almacenamiento de datos compartidos por las diferentes páginas del sitio web, en el lado cliente.

Existen dos formas de almacenar los datos:

- Mediante el método setItem().

```
localStorage.setItem(nombreDato, valorDato);
```

- Usando el dato como atributo y dándole un valor

```
localStorage.nombreDato= valorDato;
```

javaScript vs HTML5

Objetos para almacenamiento local → permiten el almacenamiento de datos compartidos por las diferentes páginas del sitio web, en el lado cliente.

Existen dos formas de recuperar los datos:

- Mediante el método getItem().

```
valorDatos= localStorage.getItem(nombreDato);
```

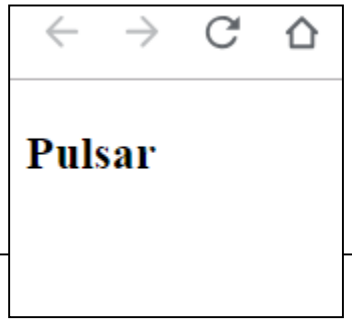
- Usando el dato como atributo y dándole un valor

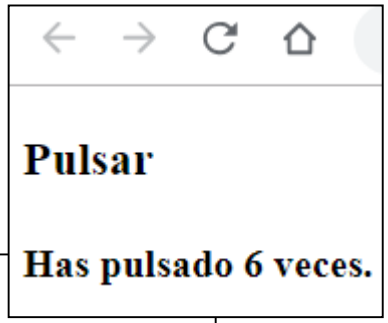
```
valorDato= localStorage.nombreDato;
```

javaScript vs HTML5

Objetos para almacenamiento local → permiten el almacenamiento de datos compartidos por las diferentes páginas del sitio web, en el lado cliente.

```
<body>
    <h3 onclick="nClicks()">Pulsar</h3>
    <h4 id="resultado"></h4>
</body>
```






```
<script>
function nClicks() {
    if(typeof(Storage) !== "undefined") {
        if (localStorage.n)
            localStorage.n= Number(localStorage.n) + 1;
        else
            localStorage.n= 1;
        document.getElementById("resultado").innerHTML = "Has pulsaado " + localStorage.n + " veces.";
    }
}
</script>
```