



# Curso de Javascript

Unidad Didáctica 13: DOM



Ayuntamiento  
de Vitoria-Gasteiz  
Vitoria-Gasteizko  
Udala

# Índice de contenidos

- Introducción
- Árbol de nodos
- Tipos de nodos
- Acceso directo a los nodos
- Creación y eliminación de nodos
- Acceso directo a los atributos XHTML
- Conclusiones



# Introducción

La creación del Document Object Model o DOM es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas



# Introducción

DOM permite a los programadores web acceder y manipular las páginas XHTML como si fueran documentos XML. De hecho, DOM se diseñó originalmente para manipular de forma sencilla los documentos XML



# Introducción

A pesar de sus orígenes, DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript) y cuyas únicas diferencias se encuentran en la forma de implementarlo



# Árbol de Nodos

Una de las tareas habituales en la programación de aplicaciones web con JavaScript consiste en la manipulación de las páginas web



# Árbol de Nodos

De esta forma, es habitual obtener el valor almacenado por algunos elementos (por ejemplo los elementos de un formulario), crear un elemento (párrafos, <div>, etc.) de forma dinámica y añadirlo a la página, aplicar una animación a un elemento (que aparezca/desaparezca, que se desplace, etc.)



# Árbol de Nodos

Todas estas tareas habituales son muy sencillas de realizar gracias a DOM. Sin embargo, para poder utilizar las utilidades de DOM, es necesario "transformar" la página original





# Árbol de Nodos

Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular

Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular



# Árbol de Nodos

Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla

El motivo por el que se muestra el funcionamiento de esta transformación interna es que condiciona el comportamiento de DOM y por tanto, la forma en la que se manipulan las páginas



# Árbol de Nodos

DOM transforma todos los documentos XHTML en un conjunto de elementos llamados nodos, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos

Por su aspecto, la unión de todos los nodos se llama "árbol de nodos"



# Árbol de Nodos

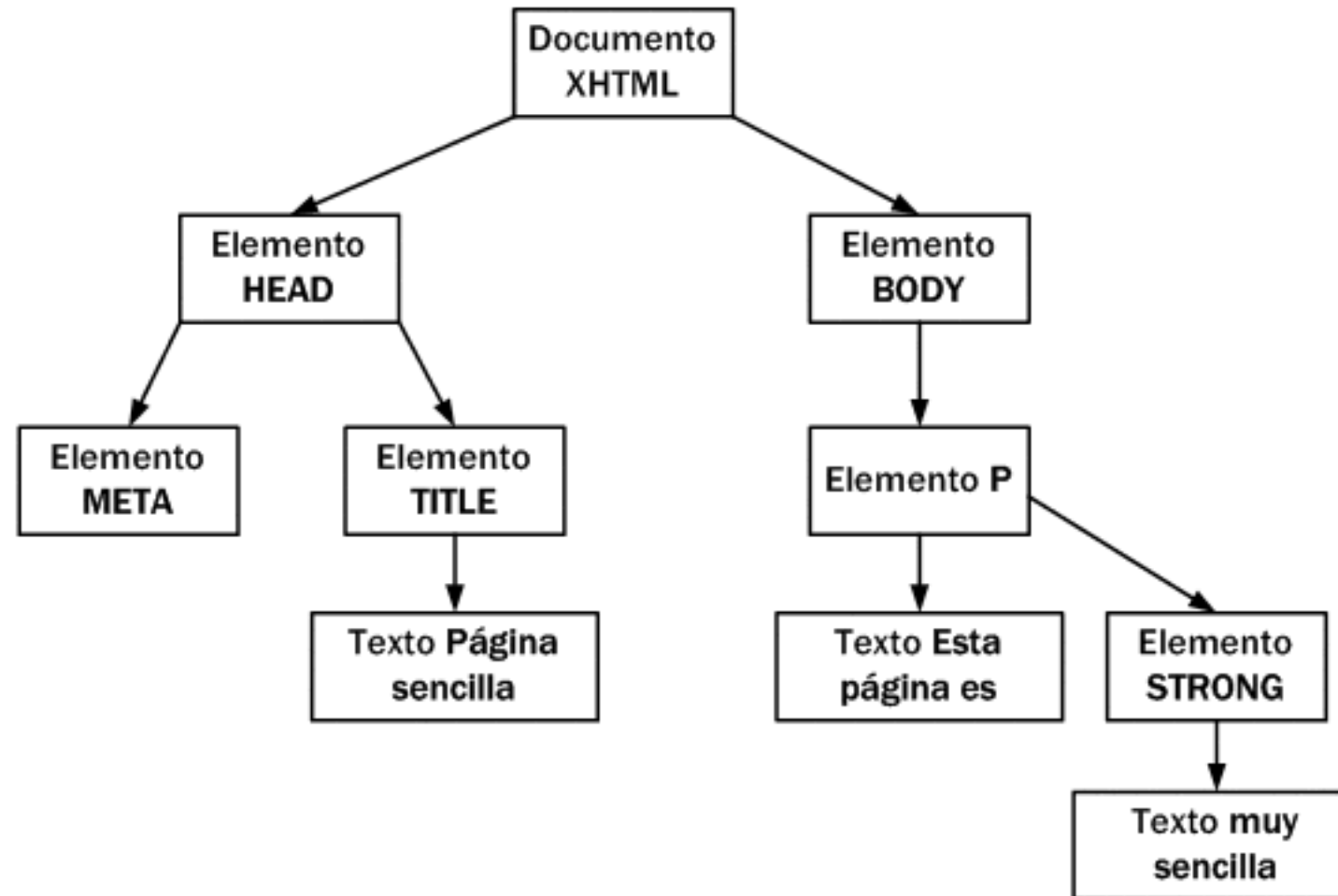
La siguiente página XHTML sencilla:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Página sencilla</title>
</head>
<body>
    <p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

<http://cursosdedesarrollo.com/>



# Árbol de Nodos



# Árbol de Nodos

En el esquema anterior, cada rectángulo representa un nodo DOM y las flechas indican las relaciones entre nodos

Dentro de cada nodo, se ha incluido su tipo (que se verá más adelante) y su contenido



# Árbol de Nodos

La raíz del árbol de nodos de cualquier página XHTML siempre es la misma: un nodo de tipo especial denominado "Documento"



# Árbol de Nodos

A partir de ese nodo raíz, cada etiqueta XHTML se transforma en un nodo de tipo “Elemento”

La conversión de etiquetas en nodos se realiza de forma jerárquica





# Árbol de Nodos

De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY

A partir de esta derivación inicial, cada etiqueta XHTML se transforma en un nodo que deriva del nodo correspondiente a su "etiqueta padre"



# Árbol de Nodos

La transformación de las etiquetas XHTML habituales genera dos nodos: el primero es el nodo de tipo "Elemento" (correspondiente a la propia etiqueta XHTML) y el segundo es un nodo de tipo "Texto" que contiene el texto encerrado por esa etiqueta XHTML



# Árbol de Nodos

Así, la siguiente etiqueta XHTML:

```
<title>Página sencilla</title>
```



# Árbol de Nodos



# Árbol de Nodos

De la misma forma, la siguiente etiqueta XHTML:

```
<p>Esta página es <strong>muy sencilla</strong></p>
```



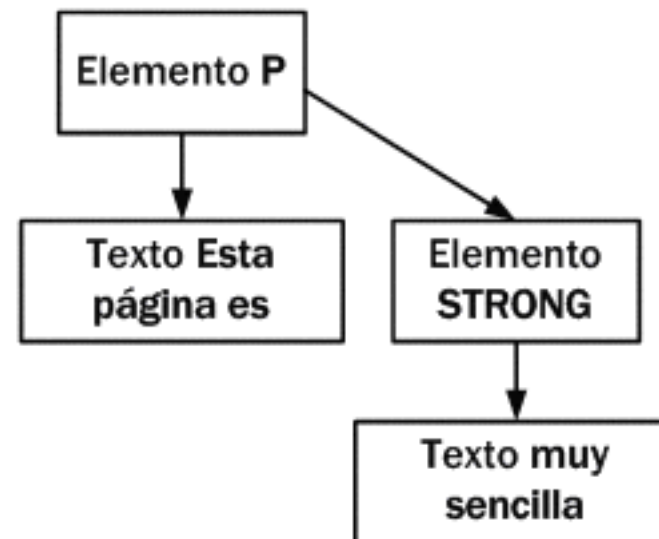
# Árbol de Nodos

Genera los siguientes nodos:

- Nodo de tipo "Elemento" correspondiente a la etiqueta <p>.
- Nodo de tipo "Texto" con el contenido textual de la etiqueta <p>.
- Como el contenido de <p> incluye en su interior otra etiqueta XHTML, la etiqueta interior se transforma en un nodo de tipo "Elemento" que representa la etiqueta <strong> y que deriva del nodo anterior.
- El contenido de la etiqueta <strong> genera a su vez otro nodo de tipo "Texto" que deriva del nodo generado por <strong>.



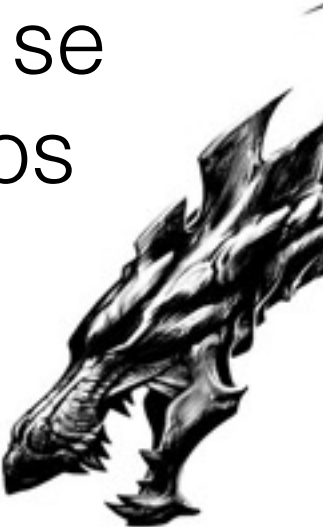
# Árbol de Nodos



# Árbol de Nodos

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

- Las etiquetas XHTML se transforman en dos nodos: el primero es la propia etiqueta y el segundo nodo es hijo del primero y consiste en el contenido textual de la etiqueta.
- Si una etiqueta XHTML se encuentra dentro de otra, se sigue el mismo procedimiento anterior, pero los nodos generados serán nodos hijo de su etiqueta padre.





# Árbol de Nodos

Como se puede suponer, las páginas XHTML habituales producen árboles con miles de nodos

Aun así, el proceso de transformación es rápido y automático, siendo las funciones proporcionadas por DOM (que se verán más adelante) las únicas que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata



# Tipos de Nodos

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas XHTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- Document, nodo raíz del que derivan todos los demás nodos del árbol.
- Element, representa cada una de las etiquetas XHTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- Attr, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas XHTML, es decir, uno por cada par atributo=valor.
- Text, nodo que contiene el texto encerrado por una etiqueta XHTML.
- Comment, representa los comentarios incluidos en la página XHTML.



# Tipos de Nodos

Los otros tipos de nodos existentes que no se van a considerar son `DocumentType`, `CDataSection`, `DocumentFragment`, `Entity`, `EntityReference`, `ProcessingInstruction` y `Notation`



# Acceso Directo a los Nodos

Una vez construido automáticamente el árbol completo de nodos DOM, ya es posible utilizar las funciones DOM para acceder de forma directa a cualquier nodo del árbol



# Acceso Directo a los Nodos

Como acceder a un nodo del árbol es equivalente a acceder a "un trozo" de la página, una vez construido el árbol, ya es posible manipular de forma sencilla la página: acceder al valor de un elemento, establecer el valor de un elemento, mover un elemento de la página, crear y añadir nuevos elementos, etc.



# Acceso Directo a los Nodos

DOM proporciona dos métodos alternativos para acceder a un nodo específico: acceso a través de sus nodos padre y acceso directo



# Acceso Directo a los Nodos

Las funciones que proporciona DOM para acceder a un nodo a través de sus nodos padre consisten en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama terminada por el nodo buscado



# Acceso Directo a los Nodos

Sin embargo, cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo y no llegar hasta él descendiendo a través de todos sus nodos padre





# Acceso Directo a los Nodos

Por ese motivo, no se van a presentar las funciones necesarias para el acceso jerárquico de nodos y se muestran solamente las que permiten acceder de forma directa a los nodos



# Acceso Directo a los Nodos

Por último, es importante recordar que el acceso a los nodos, su modificación y su eliminación solamente es posible cuando el árbol DOM ha sido construido completamente, es decir, después de que la página XHTML se cargue por completo



# Acceso Directo a los Nodos

Más adelante se verá cómo asegurar que un código JavaScript solamente se ejecute cuando el navegador ha cargado entera la página XHTML



# Acceso Directo a los Nodos

GETELEMENTSBYTAGNAME()

Como sucede con todas las funciones que proporciona DOM, la función `getElementsByTagName()` tiene un nombre muy largo, pero que lo hace autoexplicativo



# Acceso Directo a los Nodos

GETELEMENTSBYTAGNAME()

La función `getElementsByTagName(nombreEtiqueta)` obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función



# Acceso Directo a los Nodos

El siguiente ejemplo muestra cómo obtener todos los párrafos de una página XHTML:

```
var parrafos =  
document.getElementsByTagName("p");
```



# Acceso Directo a los Nodos

El valor que se indica delante del nombre de la función (en este caso, document) es el nodo a partir del cual se realiza la búsqueda de los elementos

En este caso, como se quieren obtener todos los párrafos de la página, se utiliza el valor document como punto de partida de la búsqueda

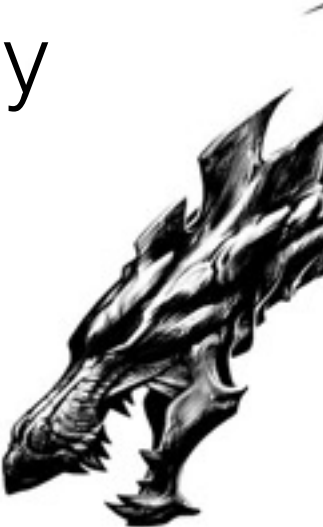


# Acceso Directo a los Nodos

El valor que devuelve la función es un array con todos los nodos que cumplen la condición de que su etiqueta coincide con el parámetro proporcionado

El valor devuelto es un array de nodos DOM, no un array de cadenas de texto o un array de objetos normales

Por lo tanto, se debe procesar cada valor del array de la forma que se muestra en las siguientes secciones





# Acceso Directo a los Nodos

De este modo, se puede obtener el primer párrafo de la página de la siguiente manera:

```
var primerParrafo = parrafos[0];
```



# Acceso Directo a los Nodos

De la misma forma, se podrían recorrer todos los párrafos de la página con el siguiente código:

```
for(var i=0; i<parrafos.length; i++) {  
  
    var parrafo = parrafos[i];  
  
}
```



# Acceso Directo a los Nodos

La función `getElementsByTagName()` se puede aplicar de forma recursiva sobre cada uno de los nodos devueltos por la función

En el siguiente ejemplo, se obtienen todos los enlaces del primer párrafo de la página



# Acceso Directo a los Nodos

```
var parrafos =  
document.getElementsByTagName("p");  
  
var primerParrafo = parrafos[0];  
  
var enlaces =  
primerParrafo.getElementsByTagName("a");
```



# Acceso Directo a los Nodos

GETELEMENTSBYNAME()

La función `getElementsByName()` es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo `name` sea igual al parámetro proporcionado



# Acceso Directo a los Nodos

En el siguiente ejemplo, se obtiene directamente el único párrafo con el nombre indicado:

```
var parrafoEspecial =  
document.getElementsByName("especial");
```

```
<p name="prueba">...</p>
```

```
<p name="especial">...</p>
```

```
<p>...</p>
```



# Acceso Directo a los Nodos

Normalmente el atributo name es único para los elementos HTML que lo definen, por lo que es un método muy práctico para acceder directamente al nodo deseado

En el caso de los elementos HTML radiobutton, el atributo name es común a todos los radiobutton que están relacionados, por lo que la función devuelve una colección de elementos



# Acceso Directo a los Nodos

## GETELEMENTBYID()

La función getElementById() es la más utilizada cuando se desarrollan aplicaciones web dinámicas.

Se trata de la función preferida para acceder directamente a un nodo y poder leer o modificar sus propiedades





# Acceso Directo a los Nodos

La función `getElementById()` devuelve el elemento XHTML cuyo atributo `id` coincide con el parámetro indicado en la función

Como el atributo `id` debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado



# Acceso Directo a los Nodos

```
var cabecera =  
document.getElementById("cabecera");
```

```
<div id="cabecera">
```

```
<a href="/" id="logo">...</a>
```

```
</div>
```



# Acceso Directo a los Nodos

La función `getElementById()` es tan importante y tan utilizada en todas las aplicaciones web, que casi todos los ejemplos y ejercicios que siguen la utilizan constantemente



# Acceso Directo a los Nodos

## QUERYSELECTOR()

La función `querySelector()` acepta como parámetro un selector que identifica el elemento (o elementos) a seleccionar



# Acceso Directo a los Nodos

En el caso de esta función, únicamente es devuelto el primer elemento que cumple la condición. Si no existe el elemento, el valor retornado es null



# Acceso Directo a los Nodos

```
var logo = document.querySelector(".enlace");
```

```
<div id="cabecera">
```

```
<a href="/" class="enlace">...</a>
```

```
</div>
```

```
<div id="cuerpo">
```

```
<p>Loren ipsum <a href="enlace">...</a></p>
```

```
</div>
```



# Acceso Directo a los Nodos

En este caso, a pesar de existir varios elementos de la clase enlace, únicamente es seleccionado el primero de ellos



# Acceso Directo a los Nodos

## QUERYSELECTORALL()

La función `querySelectorAll()` acepta como parámetro un selector que identifica el elemento (o elementos) a seleccionar. Esta función devuelve un objeto de tipo `NodeList` con los elementos que coincidan con el selector





# Acceso Directo a los Nodos

```
var enlaces = document.querySelectorAll(".enlace");
```

```
<div id="cabecera">
```

```
<a href="/" class="enlace">...</a>
```

```
</div>
```

```
<div id="cuerpo">
```

```
<p>Loren ipsum <a href="enlace">...</a></p>
```

```
</div>
```



# Acceso Directo a los Nodos

Para acceder a los elementos almacenados en NodeList, recorreremos el objeto como si de un array se tratase:

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



# Creación y Eliminación de Nodos

Acceder a los nodos y a sus propiedades (que se verá más adelante) es sólo una parte de las manipulaciones habituales en las páginas

Las otras operaciones habituales son las de crear y eliminar nodos del árbol DOM, es decir, crear y eliminar "trozos" de la página web



# Creación y Eliminación de Nodos

## CREACIÓN DE ELEMENTOS XHTML SIMPLES

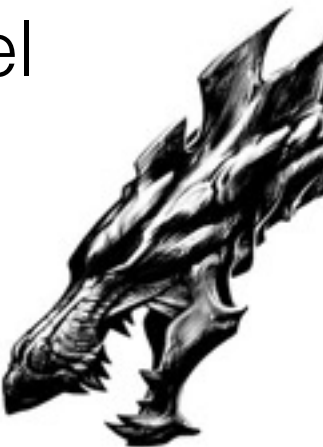
Como se ha visto, un elemento XHTML sencillo, como por ejemplo un párrafo, genera dos nodos: el primer nodo es de tipo Element y representa la etiqueta `<p>` y el segundo nodo es de tipo Text y representa el contenido textual de la etiqueta `<p>`



# Creación y Eliminación de Nodos

Por este motivo, crear y añadir a la página un nuevo elemento XHTML sencillo consta de cuatro pasos diferentes:

- Creación de un nodo de tipo Element que represente al elemento.
- Creación de un nodo de tipo Text que represente el contenido del elemento.
- Añadir el nodo Text como nodo hijo del nodo Element.
- Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.



# Creación y Eliminación de Nodos

De este modo, si se quiere añadir un párrafo simple al final de una página XHTML, es necesario incluir el siguiente código JavaScript:

```
// Crear nodo de tipo Element
```

```
var parrafo = document.createElement("p");
```

```
// Crear nodo de tipo Text
```

```
var contenido = document.createTextNode("Hola Mundo!");
```

```
// Añadir el nodo Text como hijo del nodo Element
```

```
parrafo.appendChild(contenido);
```

```
// Añadir el nodo Element como hijo de la pagina
```

```
document.body.appendChild(parrafo);
```

<http://cursosdedesarrollo.com/>



# Creación y Eliminación de Nodos

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización de tres funciones DOM:

- `createElement(etiqueta)`: crea un nodo de tipo `Element` que representa al elemento XHTML cuya etiqueta se pasa como parámetro.
- `createTextNode(contenido)`: crea un nodo de tipo `Text` que almacena el contenido textual de los elementos XHTML.
- `nodoPadre.appendChild(nodoHijo)`: añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo `Text` como hijo del nodo `Element` y a continuación se añade el nodo `Element` como hijo de algún nodo de la página.



# Creación y Eliminación de Nodos

## ELIMINACIÓN DE NODOS

Afortunadamente, eliminar un nodo del árbol DOM de la página es mucho más sencillo que añadirlo. En este caso, solamente es necesario utilizar la función `removeChild()`





# Creación y Eliminación de Nodos

```
var parrafo =  
document.getElementById("provisional");  
parrafo.parentNode.removeChild(parrafo);
```

```
<p id="provisional">...</p>
```



# Creación y Eliminación de Nodos

La función `removeChild()` requiere como parámetro el nodo que se va a eliminar

Además, esta función debe ser ejecutada desde el elemento padre de ese nodo que se quiere eliminar

La forma más segura y rápida de acceder al nodo padre de un elemento es mediante la propiedad `nodoHijo.parentNode`



# Creación y Eliminación de Nodos

Así, para eliminar un nodo de una página XHTML se llama a la función `removeChild()` desde el valor `parentNode` del nodo que se quiere eliminar

Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo



# Acceso Directo a los Atributos XHTML

Una vez que se ha accedido a un nodo, el siguiente paso natural consiste en acceder y/o modificar sus atributos y propiedades

Mediante DOM, es posible acceder de forma sencilla a todos los atributos XHTML y todas las propiedades CSS de cualquier elemento de la página



# Acceso Directo a los Atributos XHTML

Los atributos XHTML de los elementos de la página se transforman automáticamente en propiedades de los nodos

Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo



# Acceso Directo a los Atributos XHTML

El siguiente ejemplo obtiene de forma directa la dirección a la que enlaza el enlace:

```
var enlace = document.getElementById("enlace");  
console.log(enlace.href); // muestra http://www...com  
  
<a id="enlace" href="http://www...com">Enlace</a>
```



# Acceso Directo a los Atributos XHTML

En el ejemplo anterior, se obtiene el nodo DOM que representa el enlace mediante la función `document.getElementById()`

A continuación, se obtiene el atributo href del enlace mediante `enlace.href`

Para obtener por ejemplo el atributo id, se utilizaría `enlace.id`



# Acceso Directo a los Atributos XHTML

Las propiedades CSS requieren un paso extra para acceder a ellas

Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style. El siguiente ejemplo obtiene el valor de la propiedad margin de la imagen:

```
var imagen = document.getElementById("imagen");
```

```
console.log(imagen.style.margin);
```

```

```





# Acceso Directo a los Atributos XHTML

Si el nombre de una propiedad CSS es compuesto, se accede a su valor modificando ligeramente su nombre:

```
var parrafo = document.getElementById("parrafo");
```

```
console.log(parrafo.style.fontWeight); // muestra  
"bold"
```

```
<p id="parrafo" style="font-weight: bold;">...</p>
```



# Acceso Directo a los Atributos XHTML

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio (lo que se conoce como nomenclatura lowerCamelCase)



# Acceso Directo a los Atributos XHTML

A continuación se muestran algunos ejemplos:

- font-weight se transforma en fontWeight
- line-height se transforma en lineHeight
- border-top-style se transforma en borderTopStyle
- list-style-image se transforma en listStyleImage



# Acceso Directo a los Atributos XHTML

El único atributo XHTML que no tiene el mismo nombre en XHTML y en las propiedades DOM es el atributo class

Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento XHTML



# Acceso Directo a los Atributos XHTML

En su lugar, DOM utiliza el nombre className para acceder al atributo class de XHTML:

```
var parrafo = document.getElementById("parrafo");
```

```
console.log(parrafo.class); // muestra "undefined"
```

```
console.log(parrafo.className); // muestra "normal"
```

```
<p id="parrafo" class="normal">...</p>
```



# Conclusiones

Hemos visto cómo acceder,  
crear, modificar y eliminar  
ramas, elementos, atributos  
y estilos CSS en el DOM



# Datos de Contacto

<http://www.cursosdedesarrollo.com>  
[info@cursosdedesarrollo.com](mailto:info@cursosdedesarrollo.com)

<http://cursosdedesarrollo.com/>



# Licencia



David Vaquero Santiago

Esta obra está bajo una  
Licencia Creative Commons Atribución-  
NoComercial-CompartirIgual 4.0  
Internacional

Derivada de:

<http://www.arkaitzgarro.com/javascript/>

y

<http://javiereguiluz.com/>

<http://cursosdedesarrollo.com/>

