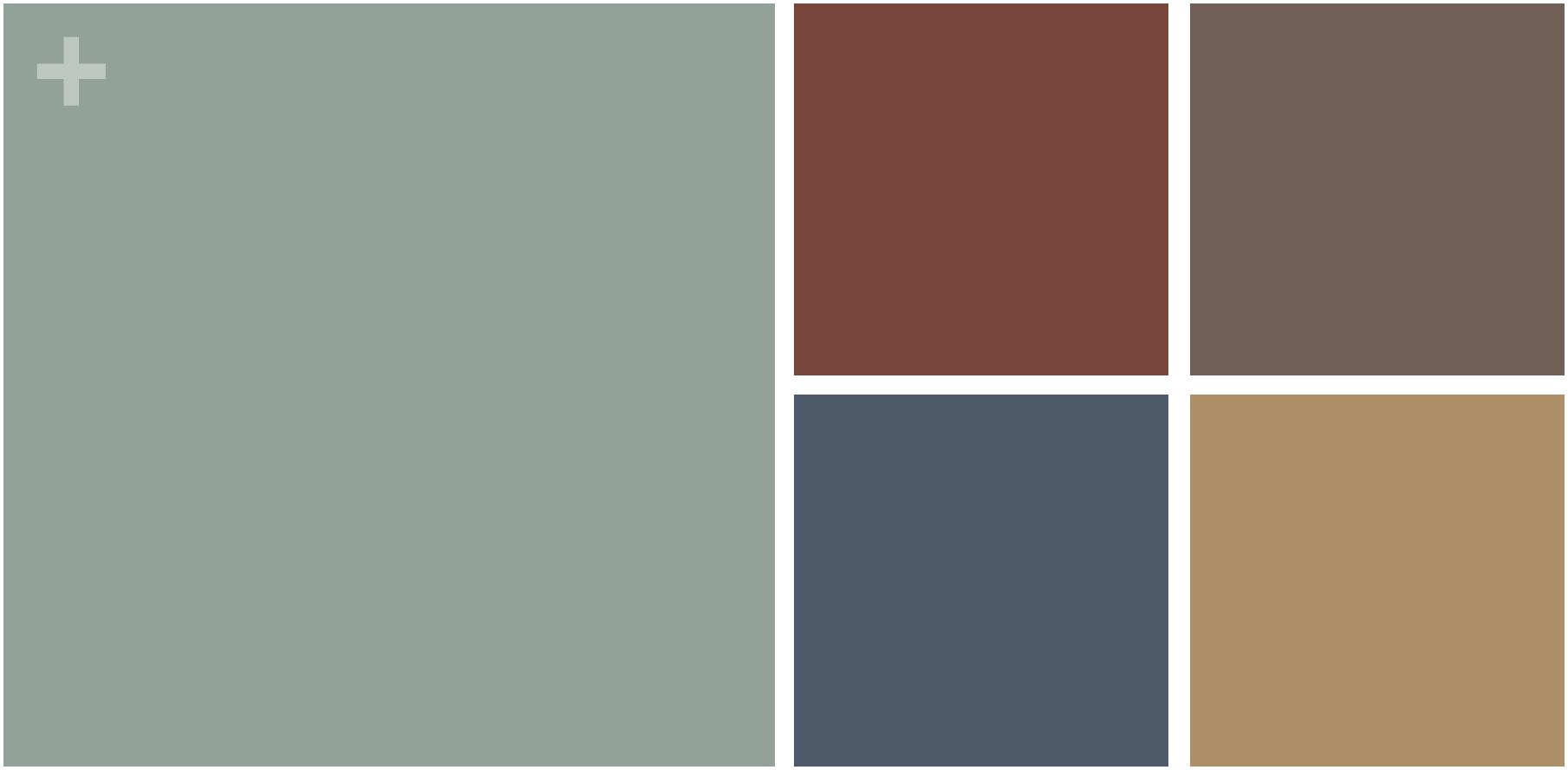


JAVASCRIPT
Capítulo 5

Lenguajes de Marcas y Sistemas de Gestión de Información

Curso 2019/2020

Capítulo 5: DOM (DOCUMENT OBJECT MODEL)

1. Introducción
2. Árbol de Nodos
3. Tipos de Nodos
4. Acceso Directo a los Nodos
5. Creación y Eliminación de Nodos
6. Acceso Directo a los Atributos XHTML

1. INTRODUCCIÓN

DOM permite **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.

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**.

2. ÁRBOL DE NODOS

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

- Obtener el **valor almacenado** por algunos elementos (formularios).
- **Crear** un elemento (párrafos, , etc.) de forma dinámica y añadirlo a la página.
- Aplicar una **animación** a un elemento (que aparezca/desaparezca, que se desplace, etc.).

Para poder utilizar las utilidades de DOM, es necesario *"transformar"* la página original. Por ello, los navegadores web transforman automáticamente todas las páginas web en una **estructura más eficiente** de manipular.

2. 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.

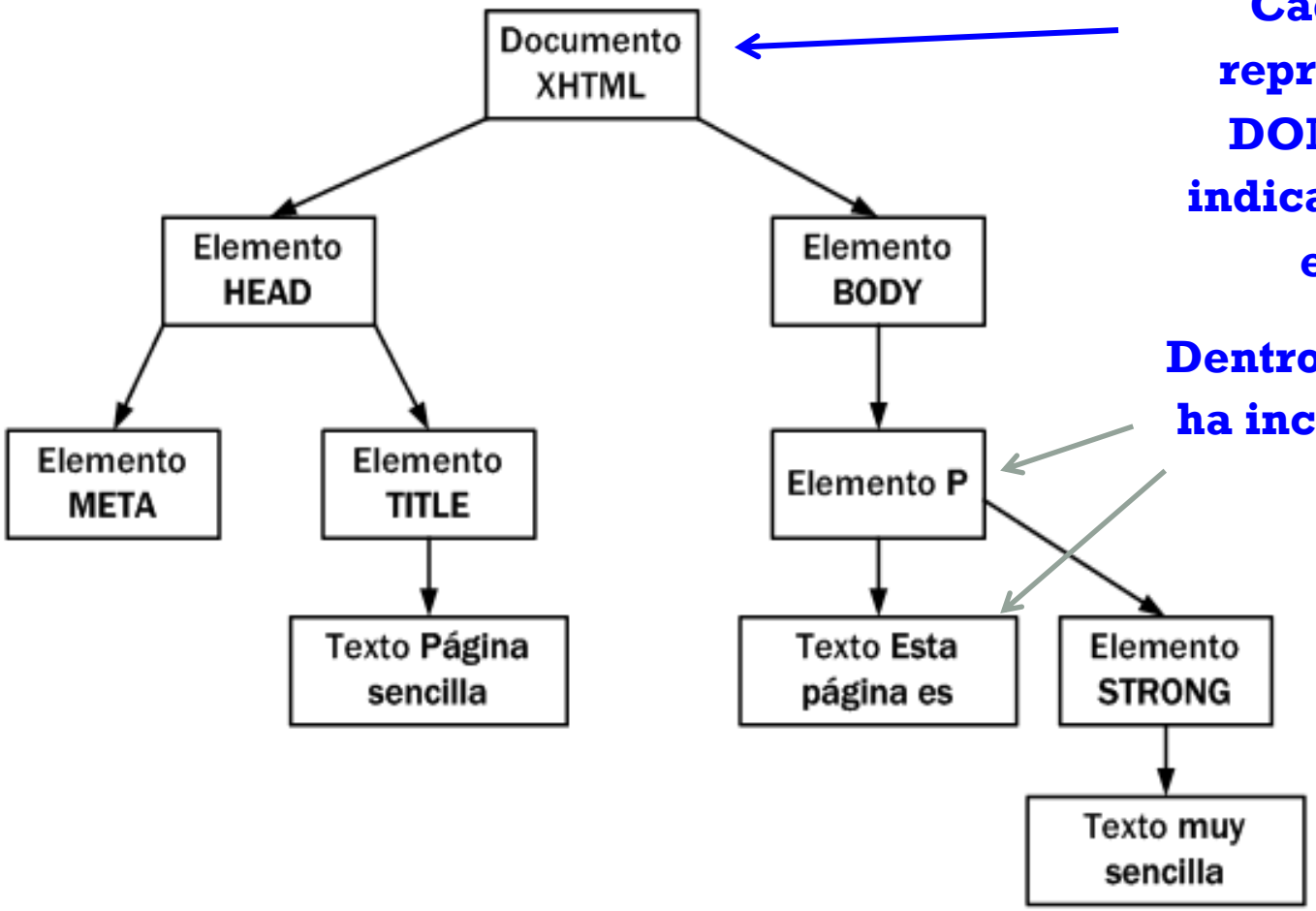
Por su aspecto, la unión de todos los nodos se llama **árbol de nodos**.

2. ÁRBOL DE NODOS

```
<!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=iso-8859-1" />
<title>Página sencilla</title>
</head>

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

2. ÁRBOL DE NODOS



Cada rectángulo representa un nodo DOM y las flechas indican las relaciones entre nodos.

Dentro de cada nodo, se ha incluido su tipo y su contenido

2. ÁRBOL DE NODOS

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

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. 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**.

La transformación de las etiquetas XHTML habituales genera **dos nodos**:

1. El **primero** es el nodo de tipo `Elemento` correspondiente a la propia etiqueta XHTML.
2. El **segundo** es un nodo de tipo `Texto` que contiene el texto encerrado por esa etiqueta XHTML.

2. ÁRBOL DE NODOS

La siguiente etiqueta XHTML genera los siguientes nodos:

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

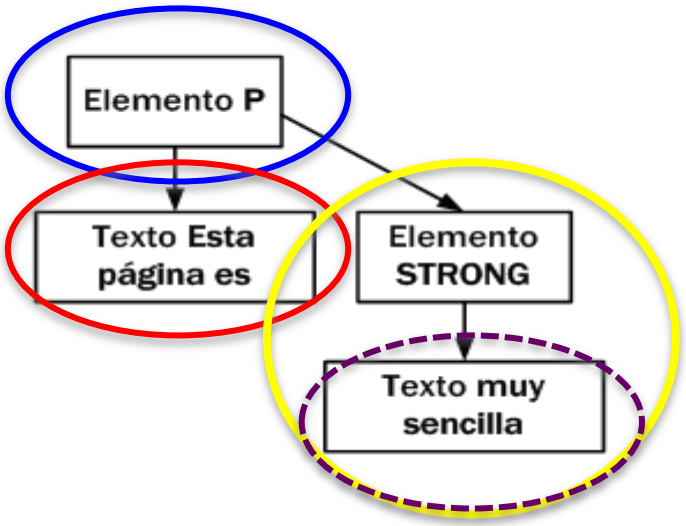
2. ÁRBOL DE NODOS

De la misma forma, la siguiente etiqueta XHTML:

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

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>`.



2. 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**.

Como se puede suponer, las páginas XHTML habituales producen árboles con **miles de nodos**. → El proceso de transformación es rápido y automático, siendo las funciones de DOM las únicas que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata.

3. 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.

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

4. 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.

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…

DOM proporciona **dos métodos alternativos** para acceder a un nodo específico:

- Acceso a través de sus nodos padre.
- Acceso directo.

4. ACCESO DIRECTO A LOS NODOS

Las funciones 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.

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.

4.1. 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:**

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

El valor que se indica delante del nombre de la función (`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.**

4.1. getElementsByTagName()

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

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. **Es un array de nodos DOM**, no un array de cadenas de texto o un array de objetos normales → Se debe procesar cada valor del array de la forma que se muestra en las siguientes secciones.

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

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

4.1. getElementsByTagName()

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:

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

var primerParrafo = parrafos[0];

var enlaces = primerParrafo.getElementsByTagName("a");
```

4.2. getElementsByName()

Es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo `name` sea igual al parámetro proporcionado. 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>
```

4.2. `getElementsByName()`

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**.

Internet Explorer no implementa de forma correcta esta función, ya que **sólo la tiene en cuenta** para los elementos de tipo `<input>` **y** `<img>`.

4.3. 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.

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.**

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

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

5. CREACIÓN Y ELIMINACIÓN DE NODOS

Acceder a los nodos y a sus propiedades es sólo una parte de las manipulaciones habituales en las páginas.

Las otras **operaciones** habituales son:

- **Crear** elementos XHTML simples.
- **Eliminar** nodos del árbol DOM.

Es decir, crear y eliminar "trozos" de la página web.

5.1. CREACIÓN ELEMENTOS SIMPLES

Un elemento XHTML sencillo 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>`.

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

1. Creación de un nodo de tipo **`Element`** que represente al elemento.
2. Creación de un nodo de tipo **`Text`** que represente el contenido del elemento.
3. Añadir el nodo **`Text`** como nodo hijo del nodo **`Element`**.
4. 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.

5.1. CREACIÓN ELEMENTOS SIMPLES

Si se quiere **añadir un párrafo simple** al final de una página XHTML, es necesario incluir el siguiente código JavaScript:

```
// 1° Crear nodo de tipo Element
var parrafo = document.createElement("p");

// 2° Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");

// 3° Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);

// 4° Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

5.1. CREACIÓN ELEMENTOS SIMPLES

El **proceso de creación de nuevos nodos** 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`.
  - Después, se añade el nodo `Element` como hijo de algún nodo de la página.

5.2. ELIMINACIÓN DE NODOS

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()`:

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

parrafo.parentNode.removeChild(parrafo);
```

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

5.2. ELIMINACIÓN DE NODOS

La función `removeChild()` requiere como parámetro el nodo que se va a eliminar. Esta función debe ser **invocada 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`.

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.

6. 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.

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.

6. 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");

alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

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`.

6. ACCESO DIRECTO A LOS ATRIBUTOS XHTML

Las propiedades CSS no son tan fáciles de obtener como los atributos XHTML. 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");

alert(imagen.style.margin);
```

```
<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

6. ACCESO DIRECTO A LOS ATRIBUTOS XHTML

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

alert(parrafo.style.fontWeight);  // muestra "bold"

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

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:

- `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`**

6. 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`.

`class` es una palabra reservada en JavaScript, no es posible utilizarla para acceder al atributo `class` del elemento XHTML.

DOM utiliza el nombre `className` para acceder al atributo `class`:

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

alert(parrafo.class); // muestra "undefined"

alert(parrafo.className); // muestra "normal”

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