POW

Tema:
DOM

www.comp.unanleon.edu.ni/u/ocastillo
oton.castillo@ct.unanleon.edu.ni

El DOM

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.

DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.

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.

El DOM

Acorde al W3C el Modelo de Objetos del Documento es una interfaz de programación de aplicaciones (API) para documentos validos HTML y bien construidos XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula.

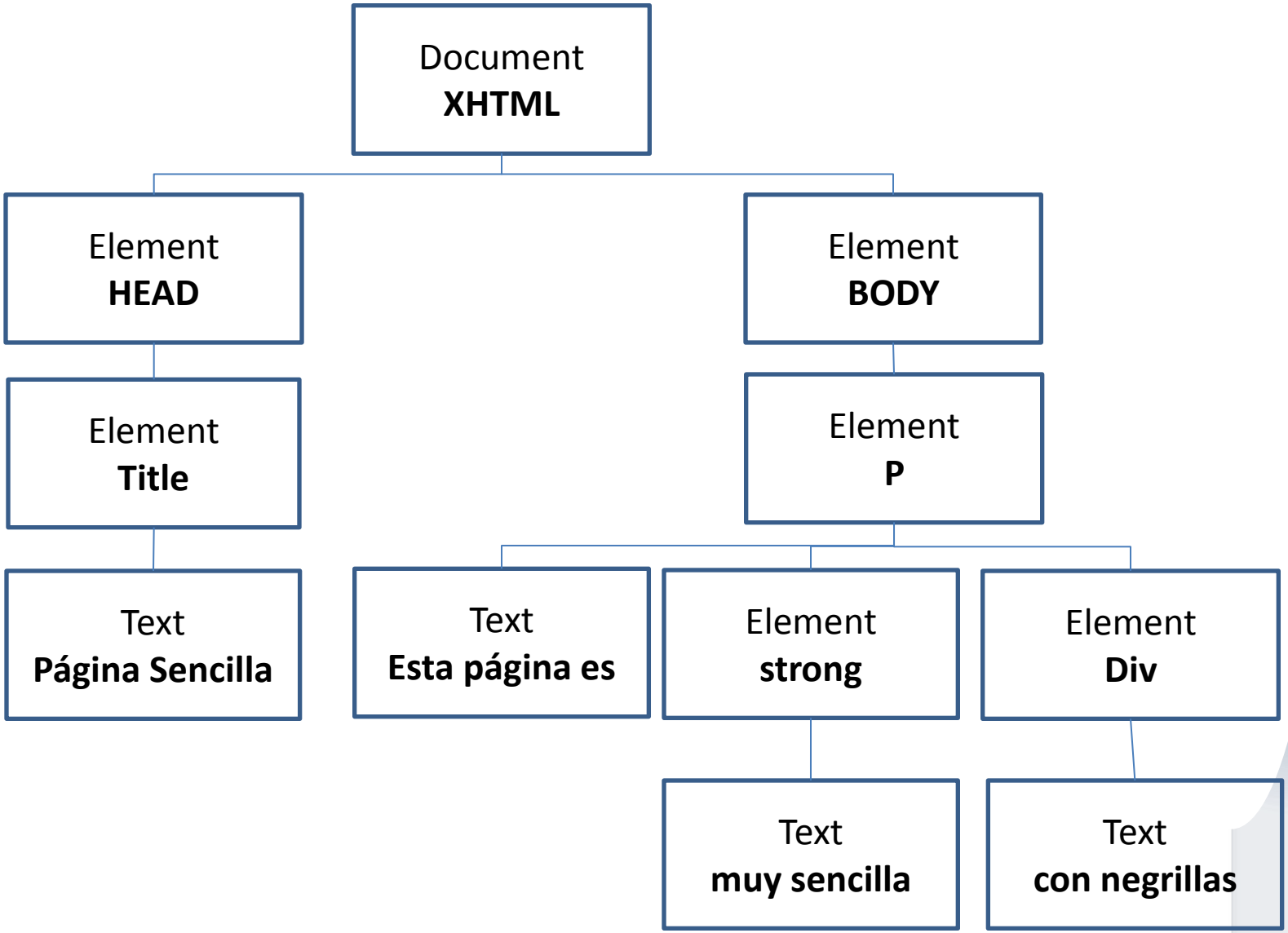
El DOM permite un acceso a la estructura de una página HTML mediante el mapeo de los elementos de esta página en un árbol de nodos. Cada elemento se convierte en un nodo y cada porción de texto en un nodo de texto.

```
<!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>
	<title>Página sencilla</title>
</head>
<body>
	<p>Esta página es
		<strong>muy sencilla</strong>
		<div>con negritas</div>
	</p>
</body>
</html>
```

Como puede verse un elemento **strong** se encuentra localizado dentro de un elemento **p** del HTML, convirtiéndose en un nodo hijo, o simplemente hijo del nodo **p**, de manera similar **p** es el nodo padre. Los nodos **div** y **strong** son hijos del mismo padre (hijos del nodo **p**), llamándose nodos hermanos o simplemente hermanos.

Es importante comprender la diferencia entre elementos y nodos de textos. Los elementos comúnmente son asociados a las etiquetas. En HTML todas las etiquetas son elementos, tales como **p**, **img** y **div** por lo que tienen atributos y contienes nodos hijos. Sin embargo, los nodos de textos no poseen atributos e hijos.

Estructura


Document
XHTML
Element
HEAD
Element
BODY
Element
Title
Element
P
Text
Página Sencilla
Text
Esta página es
Element
strong
Element
Div
Text
muy sencilla
Text
con negrillas

La raíz del árbol de nodos de cualquier página XHTML siempre es un nodo especial denominado "Documento".

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**.

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

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**.

Acceso a 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.

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 a Nodos

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

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. 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.

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.

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

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

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.

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

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

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.

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.

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

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

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];
}
```

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");
```

La función **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>
```

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 6.0 no implementa de forma correcta esta función, ya que sólo la tiene en cuenta para los elementos de tipo **<input>** y **<img>**. Además, también tiene en consideración los elementos cuyo atributo id sea igual al parámetro de la función.

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.

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>
```

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.

Internet Explorer 6.0 también interpreta incorrectamente esta función, ya que devuelve también aquellos elementos cuyo atributo **name** coincida con el parámetro proporcionado a la función.

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>**.

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.

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);
```

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la manipulación del DOM:

- **createElement**(etiqueta): Esta ***función*** crea un nodo de tipo Element que representa al elemento XHTML cuya etiqueta se pasa como parámetro.
- **createTextNode**(contenido): Esta ***función*** crea un nodo de tipo Text que almacena el contenido textual de los elementos XHTML.
- **nodoTexto.nodeValue**: Esta ***propiedad*** nos permiten acceder o asignar un texto al nodo que fue creado con la función **createTextNode.**
- **NodoPadre.appendChild**(nodoHijo): Esta ***función*** añade un nodo como hijo de otro nodo.

- **Nodo.insertBefore**(NodoInsertar, nodoactual): Esta ***función*** añade un nodo (NodoInsertar) en la posición anterior, al nodo que solicita la inserción.
- **NodoPadre.replaceChild**(nuevoNodo,nodoViejo): Esta ***función*** reemplaza un nodo hijo (nodoViejo) por un nuevo nodo (nuevoNodo).
- **NodoPadre.removeChild**(nodoremover): Esta ***función*** elimina un nodo hijo (nodoremover).
- **Nodo.cloneNode**(booleano): Esta ***función*** crear un clon de un nodo. El booleano que se pasa como parámetro define si se quiere clonar el elemento (con el valor false), o bien si se quiere clonar con su contenido (con el valor true), es decir, el elemento y todos sus descendientes.

- **NodoPadre.firstChild**: Esta ***propiedad*** nos permite acceder al primer nodo hijo, del nodo que lo solicita.
- **NodoPadre.lastChild**: Esta ***propiedad*** nos permite acceder al ultimo nodo hijo, del nodo que lo solicita.
- **NodoHijo.nextSibling**: Esta ***propiedad*** nos permite acceder al siguiente nodo hermano. Considerando que un nodo es hermano de otro, cuando son nodos hijos del mismo nodo padre.
- **NodoHijo. previousSibling**: Esta ***propiedad*** nos permite acceder al anterior nodo hermano. Considerando que un nodo es hermano de otro, cuando son nodos hijos del mismo nodo padre.

Sobre las funciones de desplazamiento los navegadores como Opera o Firefox interpretan también como hijos de un elemento los posibles espacios en blanco y saltos de línea que el elemento pueda contener.

- **Nodo.addEventListener**(‘evento’,función,booleano): Esta ***función*** nos permite agregar un manejador de evento a la interacción del usuario con el nodo. El parámetro ***‘evento’*** corresponde al evento que manejaría. El parámetro ***función*** corresponde a la función que se ejecutara si el ***‘evento’*** ocurre sobre el nodo. El parámetro ***booleano*** en la mejoría de los casos se utiliza el valor false, y corresponde el flujo de eventos.
- **Nodo.removeEventListener**(‘evento’,función,booleano): Esta ***función*** nos permite remover un manejador de evento a la interacción del usuario con el nodo.

En algunas versiones de Internet Explorer (en especial las anteriores a IE7) no soportan **addEventListener**. Lo que sí soportan son dos métodos propietarios similares: **attachEvent** y **detachEvent**:

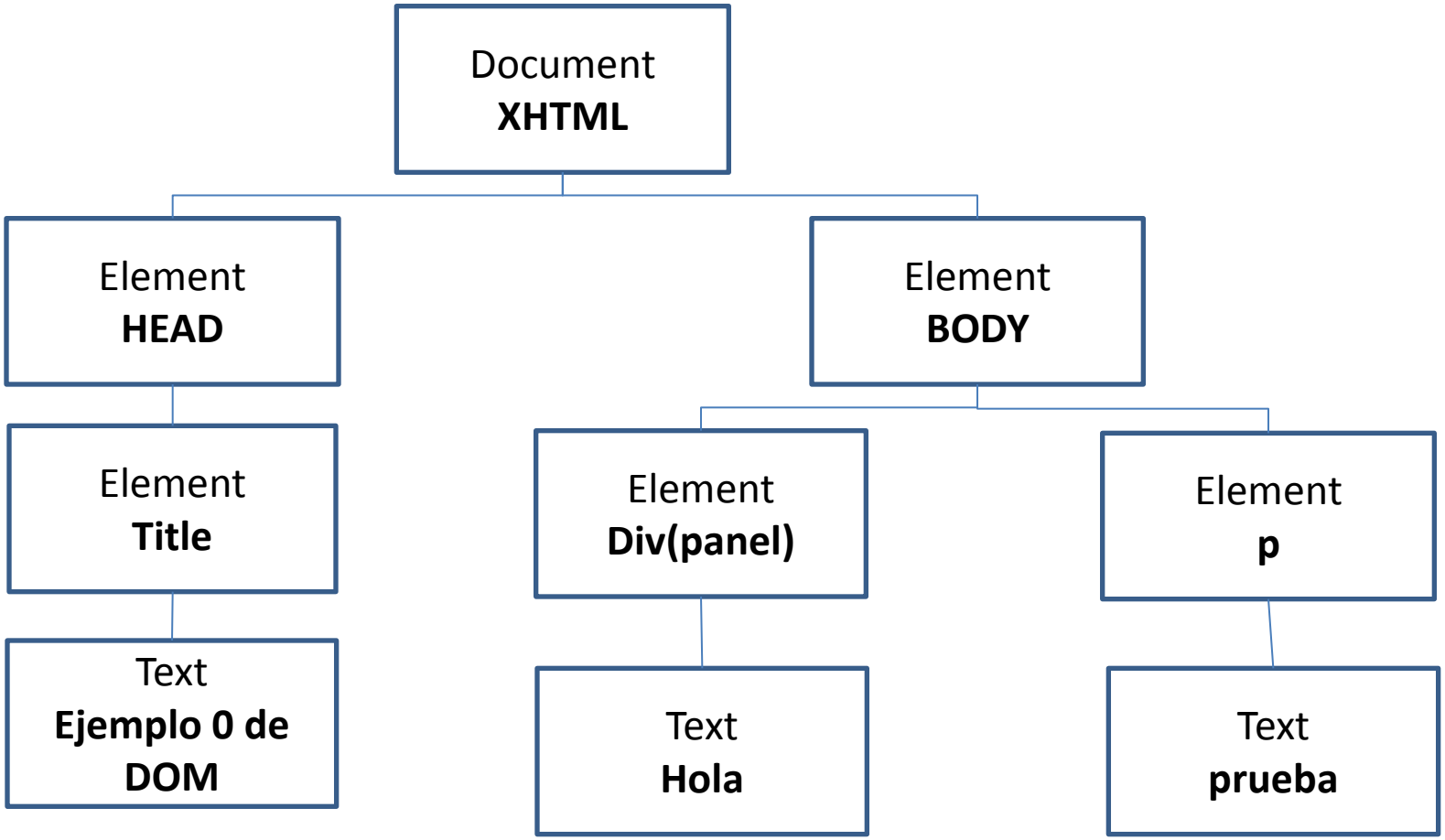
nodo.attachEvent(‘evento',función);
nodo.detachEvent(‘evento',función);

Ambos metodos son exclusivos de IE7 o inferiores.

- **NodoHijo.parentNode**: Esta ***propiedad*** nos permite acceder el objeto nodo padre, del nodo que lo solicita.
- **NodoPadre.childNodes**: Esta ***propiedad*** nos permite acceder una matriz (array) de nodos hijos, del nodo que lo solicita.

```
<html>
   <head>
      <title>Ejemplo 0 de DOM</title>
   </head>
   <body>
   <div id=‘panel’>Hola</div>
   <p style="font-weight:bold“>
   prueba</p></body>
</html>
```

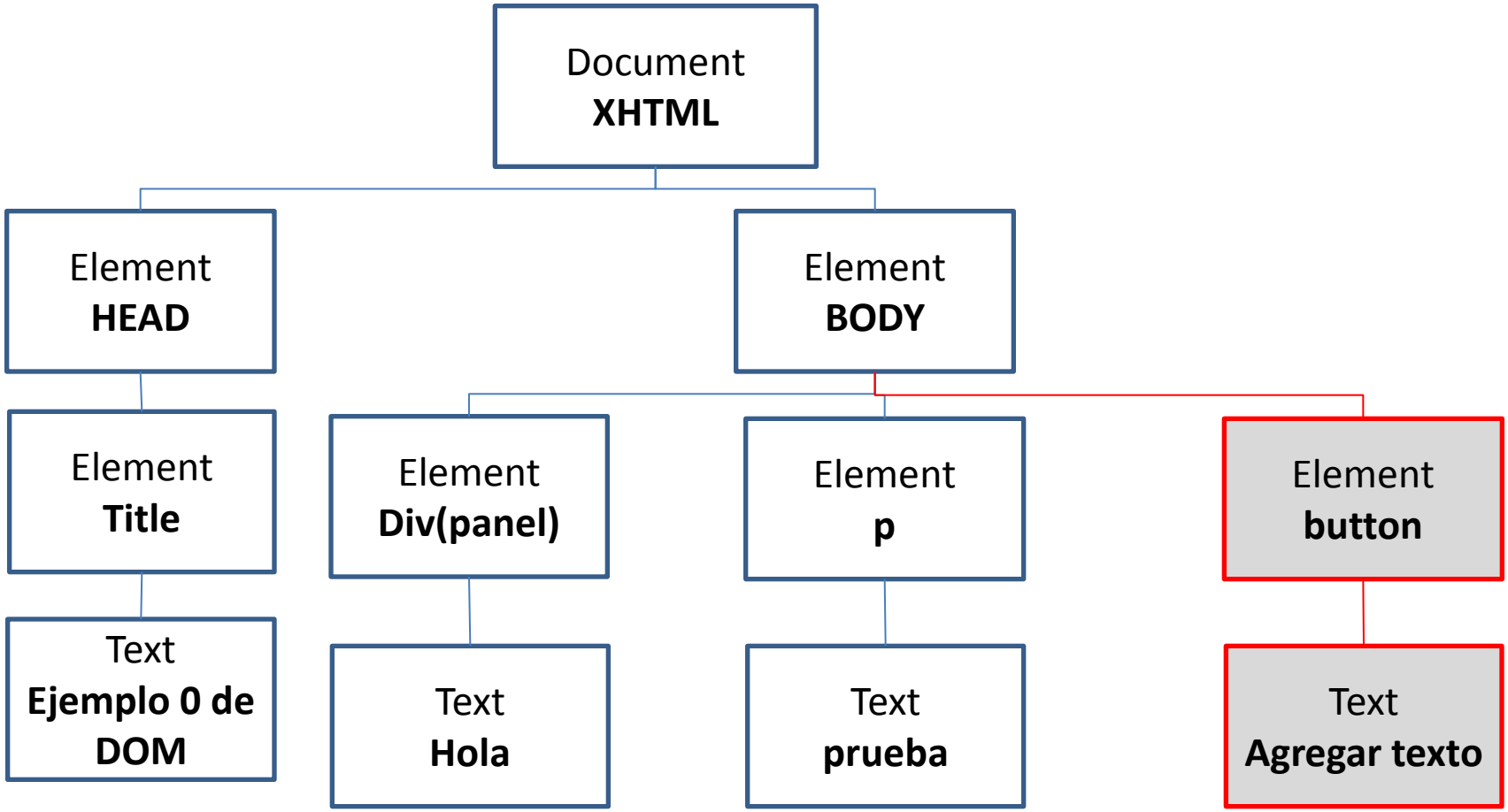
Ejemplo


Document
XHTML
Element
HEAD
Element
BODY
Element
Title
Element
Div(panel)
Element
p
Text
Ejemplo 0 de DOM
Text
Hola
Text
prueba

```
var boton=
document.createElement('button');
boton.innerHTML="Agregar texto";
boton.addEventListener('click',agregar,f
alse);
document.body.appendChild(boton);
```

```
function agregar()
{
	var texto =
document.createTextNode("agregando"
);
	var parrafo=
document.getElementsByTagName('p');
	parrafo[0].appendChild(texto);
}
```

Ejemplo


Document
XHTML
Element
HEAD
Element
BODY
Element
Title
Element
Div(panel)
Element
p
Element
button
Text
Ejemplo 0 de DOM
Text
Hola
Text
prueba
Text
Agregar texto

Al oprimir el botón agregar cambiaria así la estructura DOM.