Lección 19

Manipulacion de DOM II

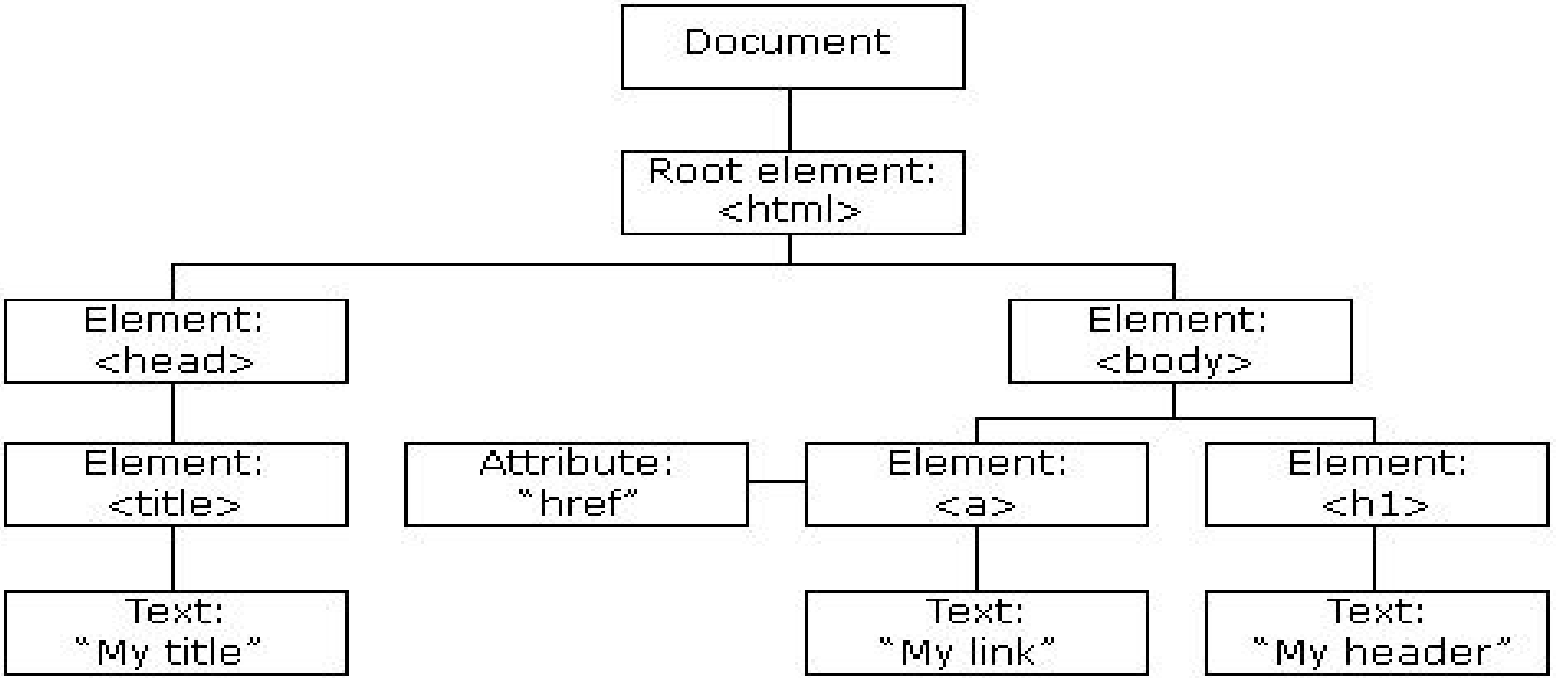
Navegación del DOM

Nodos del DOM

Con DOM se puede navegar por el árbol de nodos utilizando relaciones de nodos.

1. Todo el documento es un nodo de documento
2. Cada elemento HTML es un nodo elemento
3. El texto dentro de los elementos HTML son nodos textos
4. Cada atributo HTML es un nodo atributo
5. Todos los comentarios son nodos comentarios

Nodos del DOM



Con DOM todos los nodos pueden ser manipulados con JS (creación, lectura, edición y borrado)

Relaciones de nodos

Los nodos en el árbol tienen una relación jerárquica entre sí, para describir las relaciones se utilizan los términos padre, hijo, hermanos.

- El nodo superior se llama la raíz (o nodo raíz)
- Un nodo puede tener un número de hijos
- Hermanos son nodos con un mismo padre

Relaciones de Nodos

```
<Html>
    <head>
        <title>DOM Tutorial</title>
    </head>
    <body>
        <h1>DOM Lesson one</h1>
        <p>Hello DOM</p>
    </body>
</html>
```

<html> es el nodo raíz, es la matriz de <head> y <body>.

<head> es el primer hijo de <html>

<body> es el último hijo de <html>

Relacion de Nodos

- <head> tiene un hijo: <title>.
- <title> tiene un hijo (un nodo de texto): "DOM Tutorial".
- <body> tiene dos hijos: <h1> y <p>.
- <h1> tiene un hijo: "Lección DOM uno".
- <p> tiene un hijo: "¡Hola, mundo!".
- <h1> y <p> son hermanos.

Navegación de DOM: parentesco de elementos

Para navegar entre nodos con JavaScript se utilizan las siguientes propiedades:

- parentNode
- childNodes[nodeNumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Navegación en el DOM: Parentesco de elementos

.parentNode: Selecciona al elemento padre de otro elemento.

```
function seleccionar_padre() {
      var hijo = document.getElementById('parrafo');
      alert(hijo.parentNode);
 }
```

.firstChild y .firstElementChild

Retorna el primer nodo hijo del nodo especificado.

La diferencia entre ellas es que .**firstChild** retorna el primer nodo hijo, ya sea elemento o texto (dependiendo cual está primero), mientras que **.firstElementChild** retorna el primer nodo de tipo elemento.

```
document.getElementById('otro-parrafo').firstChild;
```

.lastChild y .lastElementChild

La propiedad **.lastChild** funciona exactamente como firstChild, pero se refiere el último de los hijos de un elemento. Se aplican, por tanto, las mismas indicaciones anteriores.

```
document.getElementById('mi-caja').lastChild;
```

.nextSibling/nextElementSibling

Gracias a **.nextSibling**, lo que podemos seleccionar es el siguiente hermano de un elemento.

Se aplican las mismas limitaciones que para las dos propiedades anteriores.

```
document.getElementById('box1').nextSibling;
```

Creando y Eliminando Nodos

Creando Nodos

Crear un nodo en el árbol DOM consta de 2 pasos:

1. Crear
2. Añadir el nodo.

Crear nodo:

- document.createElement : crea un nodo elemento
- document.createAttribute : agrega atributo al elemento
- document.createTextNode : crea un nodo texto
- document.createComment : crea un nodo comentario

Añadir nodo

Añadir nodo:

- Para añadir el nodo se utiliza **.appendChild**

  nodopadre.appendChild(nodohijo)
- Para eliminar un nodo se utiliza **.removeChild**

  nodopadre.removeChild(nodohijo)

Modificación de atributos

Modificación de atributos

.getAttribute: Devuelve el valor del atributo especificado en el elemento. Si el atributo especificado no existe, el valor retornado puede ser tanto null como " " (una cadena vacía).

```
var atributo = element.getAttribute(nombreAtributo);
```

.setAttribute: Agrega un nuevo atributo o cambia el valor de un atributo en un elemento especificado.

```
element.setAttribute(name, value);
```

Modificación de atributos

.hasAttribute: Devuelve verdadero si existe el atributo especificado, de lo contrario, devuelve false.

```
element.hasAttribute(attributename);
```

.removeAttribute: Elimina un atributo del elemento especificado.

```
element.removeAttribute(attributename);
```

Modificación de atributos

.classList: Esta propiedad retorna el nombre de la clase de un elemento. Esta propiedad es usada para agregar, remover y switchear clases de elementos CSS.

```
element.classList
```

Ejemplos

Al elemento con el id “myDiv” le queremos agregar la clase “myClass”.

```
document.getElementById("myDiv").classList.add("myClass");
```

Ahora queremos remover esa clase:

```
document.getElementById("myDiv").classList.remove("myClass");
```

Después usaremos **toggle** para switchear la clase “newClass”, esto nos permitirá agregar y quitar nuestra nueva clase.

```
document.getElementById("myDiv").classList.toggle("newClass");
```