Parte 6: Manipulación del DOM

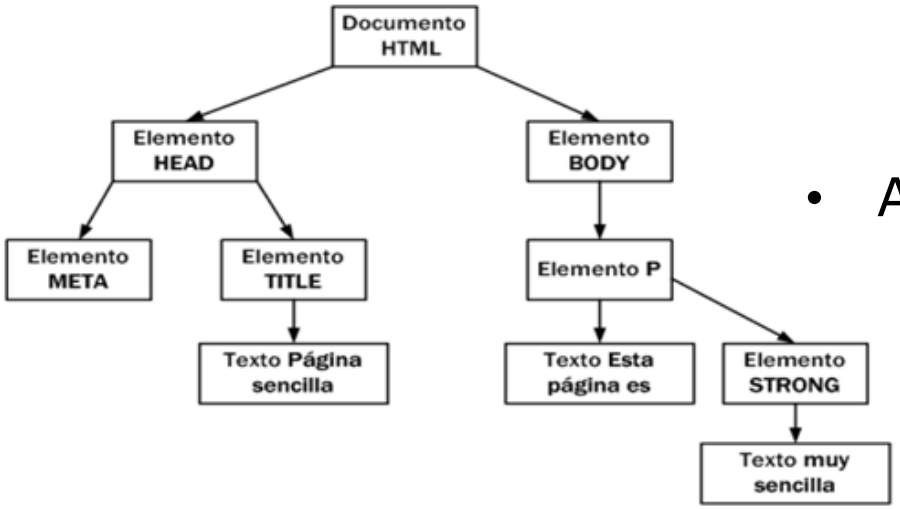
UD3: Interacción con el usuario y Modelo de objetos del documento

Después de esta clase…

- Conocerás más cosas del DOM aparte de **`querySelector`** e **`innerHTML`**
- Sabrás cómo añadir nuevos elementos al DOM de forma ordenada
- Sabrás cómo modificar y eliminar elementos sin riesgo de "cargarte" parte de la estructura

DOM (*Document Object Model*)

- **Estructura de datos** que representa todo el contenido de una página Web:




```
<!DOCTYPE html>
<head>
   <meta charset="utf-8" />
   <title>Página sencilla</title>
</head>
<body>
   <p>Esta página es
      <strong>muy sencilla</strong>
   </p>
</body>
</html>
```

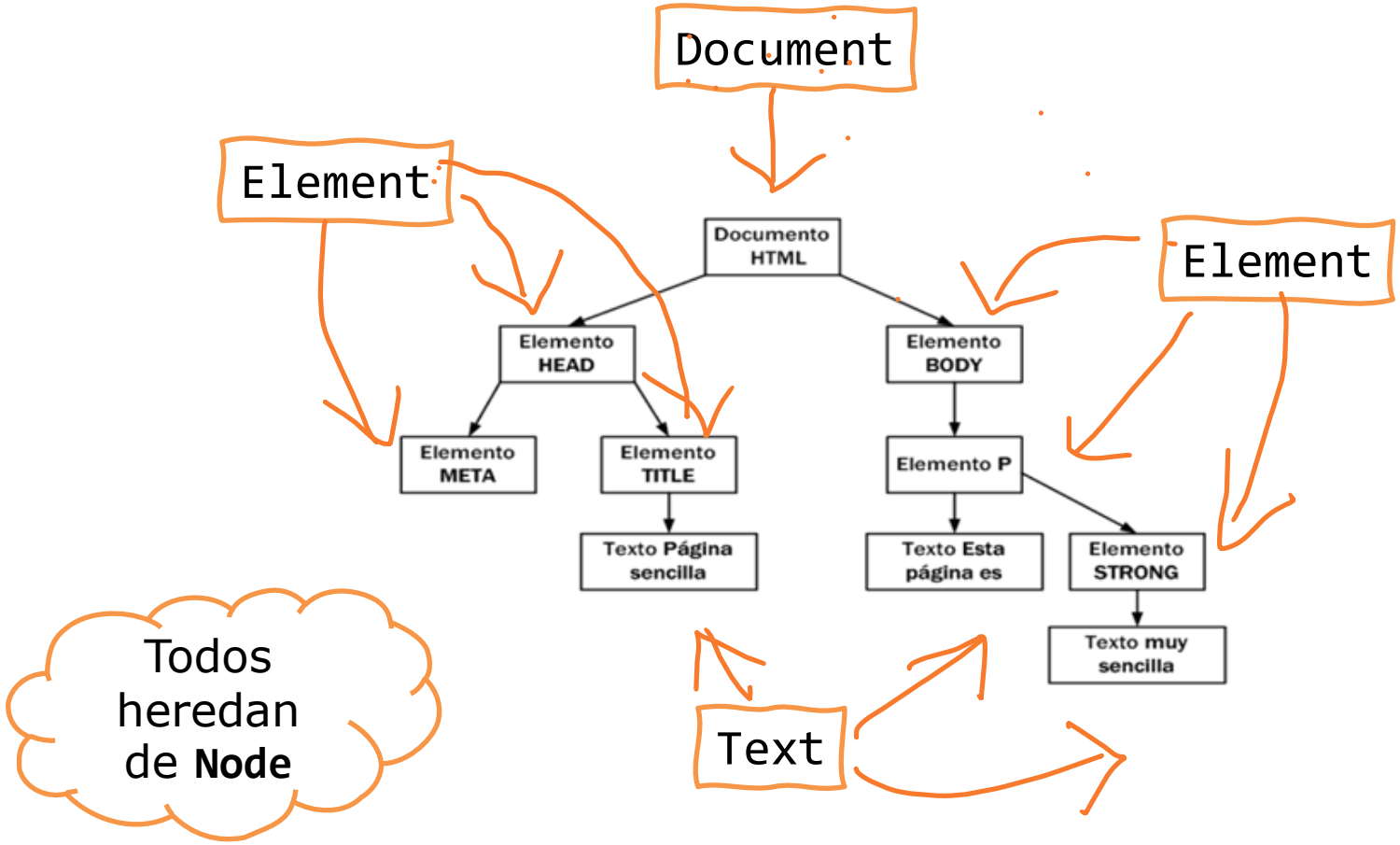
- Accediendo al DOM es posible:
  - **Leer** datos de la estructura
  - **Modificar** la estructura (y por tanto el contenido de la página Web).
    - **Modificar** elementos.
    - **Añadir** elementos.
    - **Eliminar** elementos

Nodos del DOM (Node)

- Cada nodo del DOM es un objeto de un tipo derivado de la interfaz **Node**
- Existen 12 tipos de nodos, pero normalmente manejaremos los siguientes:

Document	Raíz del que derivan el resto. Se instancia un objeto de esta clase llamado **document** mediante el cual puedo acceder al contenido del documento.
Element	Etiqueta HTML. Puede contener atributos y otros nodos.
Attr	Atributos de las etiquetas
Text	Texto encerrado por una etiqueta HTML
Comment	Comentarios

Ejemplo

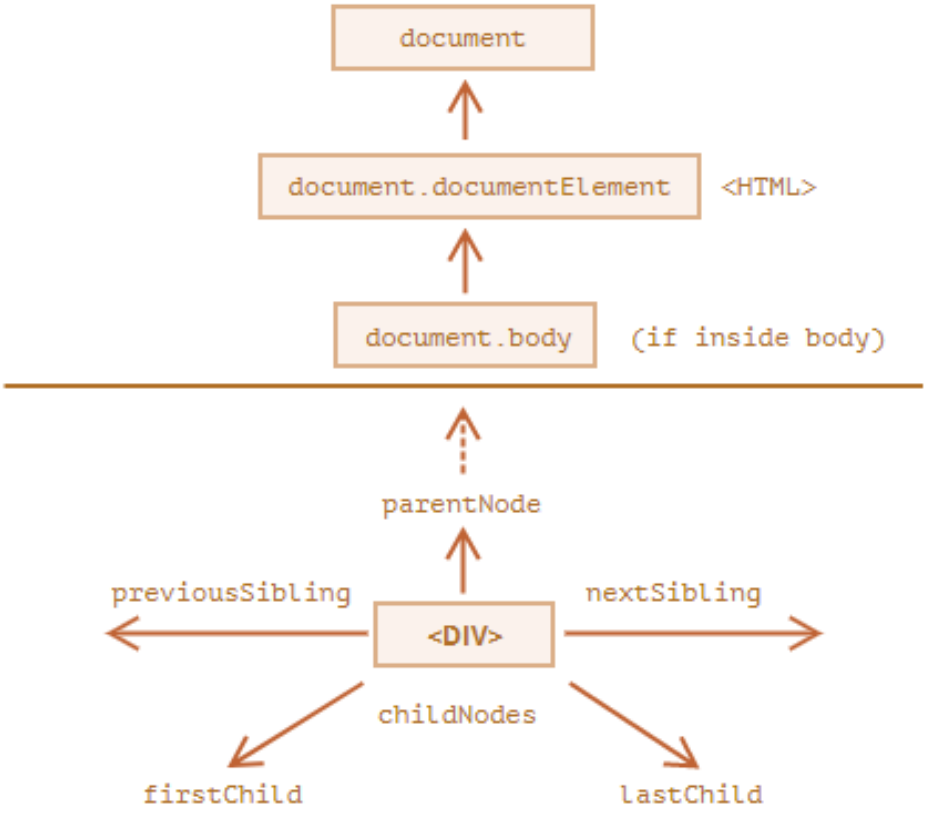

Document
Element
Element
Documento HTML
Elemento HEAD
Elemento BODY
Elemento META
Elemento TITLE
Elemento P
Texto Página sencilla
Texto Esta página es
Elemento STRONG
Texto muy sencilla
Text
Todos heredan de Node

Recordemos...

- **`document`** es un objeto de la clase **`Document`** que representa el documento cargado en una ventana del navegador
  - Permite el **acceso a todas las etiquetas HTML** dentro de una página
    - **`querySelector(selector)`** → Un elemento
    - **`querySelectorAll(selector)`** → Varios elementos
- Forma parte del objeto **`window`**, luego puede ser accedido mediante **`window.document`** o directamente **`document`**

Accediendo al DOM: Parentescos

- Desde un nodo puedo acceder a los nodos relacionados:
  - A los hijos con la propiedad **`childNodes`**.
    - **`firstChild:`** el primero
    - **`lastChild:`** el último
  - A su padre con la propiedad **`parentNode`**
  - A sus hermanos con las propiedades **`previousSibling`** y **`nextSibling`**.



Accediendo al DOM: Ejemplo

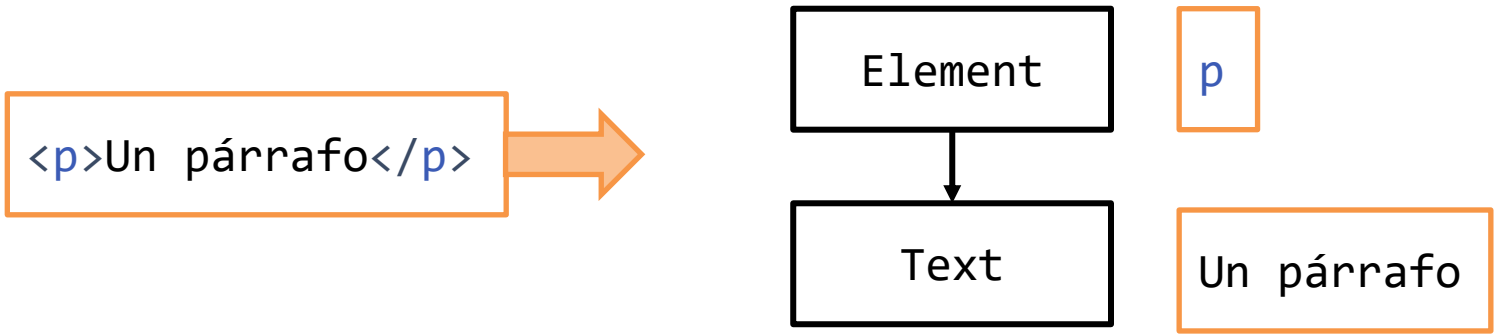
```
var listado=document.querySelector("ul");
listado.childNodes.forEach((hijo)=>
                console.log(hijo.innerHTML));

/* Manera alternativa de recorrer los hijos */
 var hijo=listado.firstChild /* listado.childNodes[0] */

while(hijo!=null){
        console.log(hijo.innerHTML);
        hijo=hijo.nextSibling();
}
```

Creación de elementos

- Un elemento HTML genera dos nodos:
  - Un nodo de tipo **`Element`** que representa la propia etiqueta
  - Un nodo de tipo **`Text`** que representa el texto de la etiqueta (lo que hay entre la apertura y el cierre de la misma.
    - No se genera si la etiqueta es de tipo sencillo (como **`<img />`**)



Métodos de `document` para crear nuevos elementos

`createElement(tagname)`	Crea el elemento HTML cuya etiqueta indica **`tagname`**
`createTextNode(texto)`	Crea el nodo de texto con el contenido **`texto`**.
`createAttribute(name)`	Crea el atributo llamado **name** Posteriormente hay que agregarlo al elemento
`createComment(comentario)`	Crea el comentario con el contenido `comentario`.

Pasos para crear un nuevo elemento

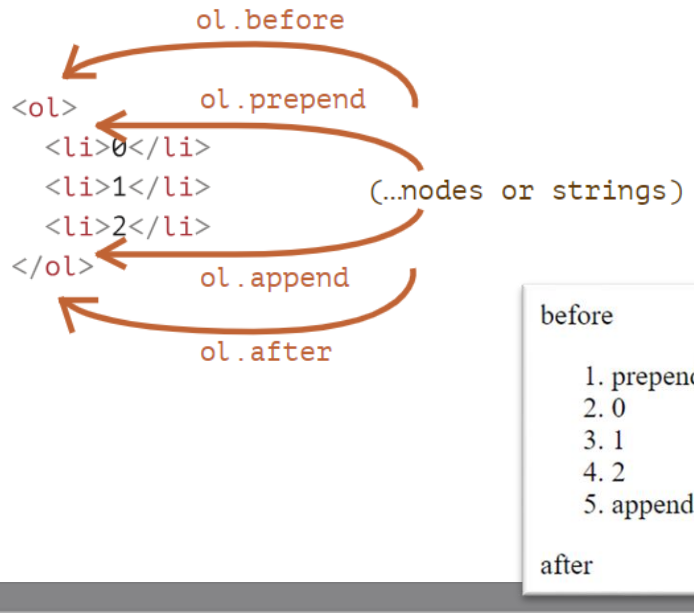
1. Creación del nodo de tipo **Element.** Para su contenido hay dos opciones:
   1. Creación del nodo de tipo **Text.** Añadir el nodo **Text** como nodo hijo del nodo **Element**
   2. Modificar la propiedad **innerHTML** del nuevo nodo.
2. Añadir el nodo **Element** a la página, usando los métodos de la siguiente diapositiva

Métodos de Node para añadir elementos

append/prepend(nodos o strings)	Añade un nodo (se le pasa) al principio/final de la lista de hijos.
before/after(nodos o strings)	Inserta el nodo nuevo (se le pasa) antes/después del nodo actual.

```
<script>
  ol.before('before');
  ol.after('after');

  let liFirst = document.createElement('li');
  liFirst.innerHTML = 'prepend';
  ol.prepend(liFirst);
  let liLast = document.createElement('li');
  liLast.innerHTML = 'append';
  ol.append(liLast);
</script>
```



Métodos de `Node` para el reemplazo, borrado y clonado

`replaceWith(nodos o strings)`	Reemplaza el nodo por el nuevo.
`remove()`	Elimina el nodo.
`cloneNode(bool)`	Si se le pasa “true” crea una clonación “profunda” del elemento, con los atributos y subelementos. Si le pasamos “false”, la clonación se hace sin sus elementos hijos

Ejemplo

```
//Clonamos el primer elemento
var primerHijo=listado.firstChild;
var nuevoHijo=primerHijo.cloneNode(true);

//Alteramos contenido del clon
nuevoHijo.innerHTML="Nuevo mensaje"

//Reemplazamos el clonado por el clon
primerHijo.replaceWith(nuevoHijo);

//Borramos el último elemento
var ultimoHijo=listado.lastChild;
ultimoHijo.remove();
```

Nodos Attribute

- Los nodos tienen **atributos** que representan los atributos HTML

createAttribute (nombre)	Crea un nodo tipo atributo nuevo con el nombre especificado en el parámetro.
hasAttribute (nombre)	Devuelve si el elemento actual tiene un atributo con el nombre especificado o no.
removeAttribute (nombre)	Elimina el atributo con el nombre especificado del elemento actual.
setAttribute (nombre, valor)	Añade un atributo a un elemento con el nombre y valor especificados.
getAttribute (nombre)	Devuelve el valor del atributo con el nombre especificado para el elemento actual.

Acceso a los atributos

```
<a id="enlace" href="http://www.iesnaranco.es">IES Naranco</a>
```

- Poniendo el nombre del atributo en minúsculas
  - A excepción del atributo **class**, al que se accede con **className**

```
var enlace=document.getElementById("enlace");
console.log(enlace.href); //muestra http://www.iesnaranco.es
```

- Usando los métodos de la transparencia anterior:

```
enlace.setAttribute("href","http://www.iesnaranco.es");
console.log(enlace.getAttribute("href"));
```

- A través de su propiedad **attributes**

```
var atributos=elemento.attributes; //Acceso a todos sus atributos
```

Acceso al estilo

- También es posible acceder al estilo de los elementos (atributo **style**)
  - El nombre de las propiedades CSS compuestas se forma eliminando todos los guiones intermedios (-) y escribiendo en mayúscula la letra siguiente.
    - **`font-weight`** se transforma en **`fontWeight`**
    - **`border-top-style`** se transforma en **`borderTopStyle`**
    - Etc.

Acceso al estilo: Ejemplo de uso

```
elemento.style.backgroundColor="red";
elemento.style.fontSize="1.5em";
elemento.style.backgroundImage=="url('fondo.png')";
```

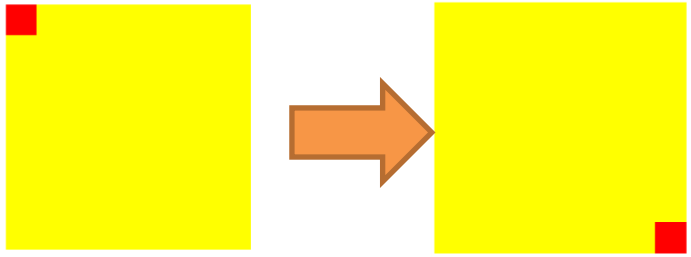
- En lugar de asignar todas las propiedades CSS individualmente, puede ser útil:
  - Agrupar las propiedades CSS que queremos asignar en una clase definida en el archivo .css
  - Asignar la clase al elemento cuyo estilo queremos cambiar (o quitarla si queremos quitar el estilo)

```
estilo-chulo{
  background-color:"red";
  font-size:1.5em;
  background-image:url('fondo.png');
}
```

```
elemento.className="estilo-chulo";
```

Animaciones

- Poder acceder al estilo nos va a permitir hacer animaciones:



```
<div id ="container">
  <div id ="animate"></div>
</div>
```

```
function myMove() {
    let id = null;
    const elem =
     document.getElementById("animate");
    let pos = 0;
    clearInterval(id);
    id = setInterval(frame, 5);
    function frame() {
      if (pos == 350) {
        clearInterval(id);
      } else {
        pos++;
        elem.style.top = pos + "px";
        elem.style.left = pos + "px";
      }
    }
  }
```

EJERCICIO PROPUESTO: Lista de la compra

- Realiza una lista de la compra formada por:
  - Una caja de texto para introducir elementos.
  - Un botón *Añadir.* Al hacer click en el mismo (o al hacer *Enter* en el teclado), se añadirá a la página HTML un elemento de lista (<li>) cuyo contenido será el valor de la caja de texto
    - Si es la primera vez que se pulsa, deberá crearse la lista
    - Los elementos de la lista responderán al evento click, pasando a estar tachados al pulsar sobre ellos.
  - Un botón *Limpiar.* Al pulsar el mismo, se eliminará del listado **todos** aquellos elementos que se hayan tachado

Ejemplo

Añadir o “enter”

Limpiar

Manejo de tablas desde el DOM

- DOM proporciona métodos específicos para trabajar con tablas:
  - Elemento `table`

`rows`	Array con las filas de la tabla
`insertRow(posicion)`	Inserta una nueva fila en la posición indicada
`deleteRow(posicion)`	Elimina una fila de la posición indicada

  - Elemento `celda`

`cells`	Devuelve un array con las columnas de la fila.
`insertCell(posicion)`	Inserta una nueva columna en la posición indicada
`deleteCell(posicion)`	Elimina la columna de la posición indicada

RETO

- Crea un código html con un div.
- Con JS, añade una tabla de 5 filas con 3 columnas siendo el contenido de cada celda Celda + nºfila+nºcol