DOM

LABORATORIO DE COMPUTACIÓN III

UTN-FRA

HTML Dinámico

DHTML tiene por objetivo aumentar la funcionalidad de un sitio web. Se puede utilizar para crear animaciones, menús desplegables, mostrar y ocultar partes de una página luego que la página fue cargada completamente, crear un entramado de capas que con sólo el HTML y CSS sería imposible alcanzar.
DHTML ofrece a los creadores de páginas web la posibilidad de modificar, cambiar la apariencia, ocultar, mostrar y animar el contenido dinámicamente.
Con DHTML podemos, después de cargada la página en el navegador, acceder a cada una de las marcas HTML, modificar sus atributos, ocultarlas, volverlas a mostrar y acceder al estilo definido para dicha marca.
Mediante JavaScript accedemos al DOM (Document Object Model) sin utilizar librerías específicas como puede ser JQuery.

Document Object Model

Mediante el DOM podemos acceder al contenido y estilo de cada marca del documento y modificarlo de acuerdo a algún evento.

Mediante el DOM podemos insertar, borrar, modificar marcas HTML. Podemos acceder a la hoja de estilo definida a la página y dinámicamente agregar, modificar o borrar propiedades. Todos esto sin tener que recargar la página del servidor, es decir todo se hace en el cliente (navegador) mediante JavaScript

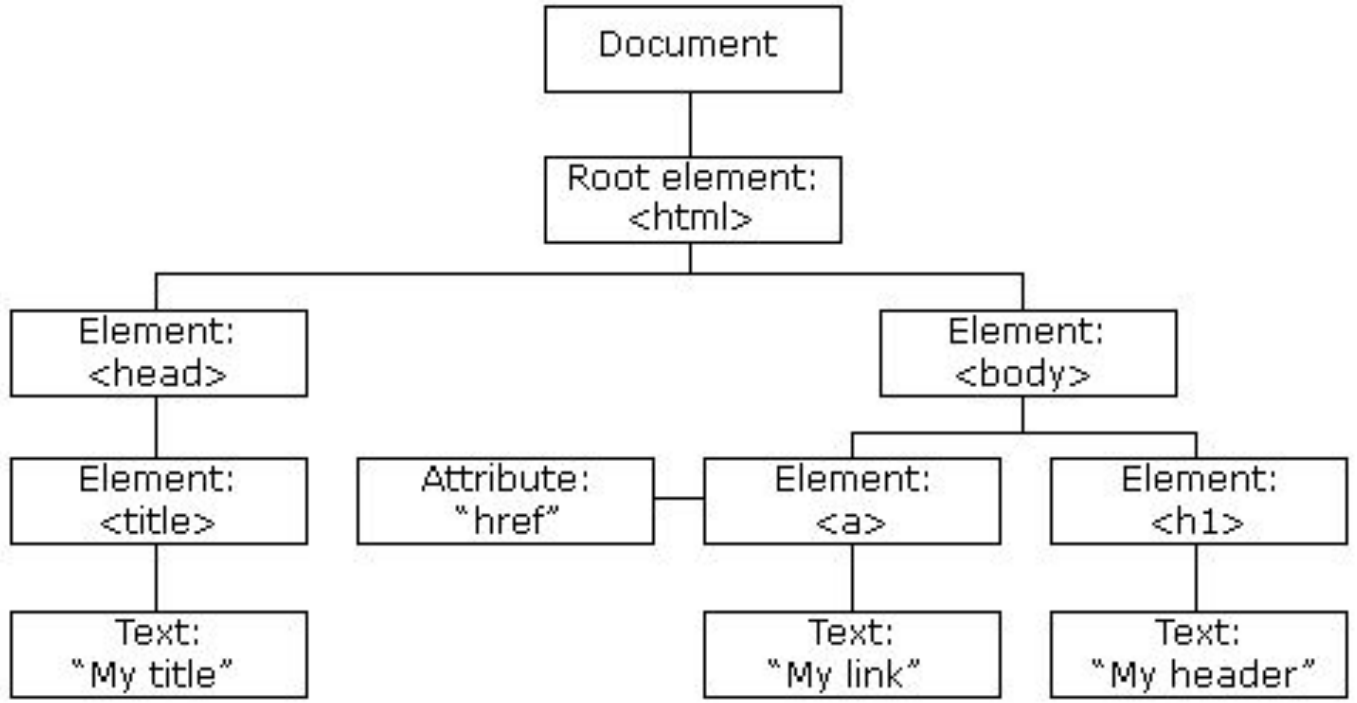
DOM - Document Object Model -

```html
<!DOCTYPE html>
<html>
<head>
    <title>My title</title>
</head>
<body>

<a href="">My link</a>

<h1>My header</h1>

</body>
</html>
```



Nodos Hijos (childNodes)

Cada nodo tiene un solo nodo padre, pero puede tener muchos hijos.
El DOM provee a cada nodo de un vector que almacena la referencia a cada nodo hijo, la propiedad se llama:

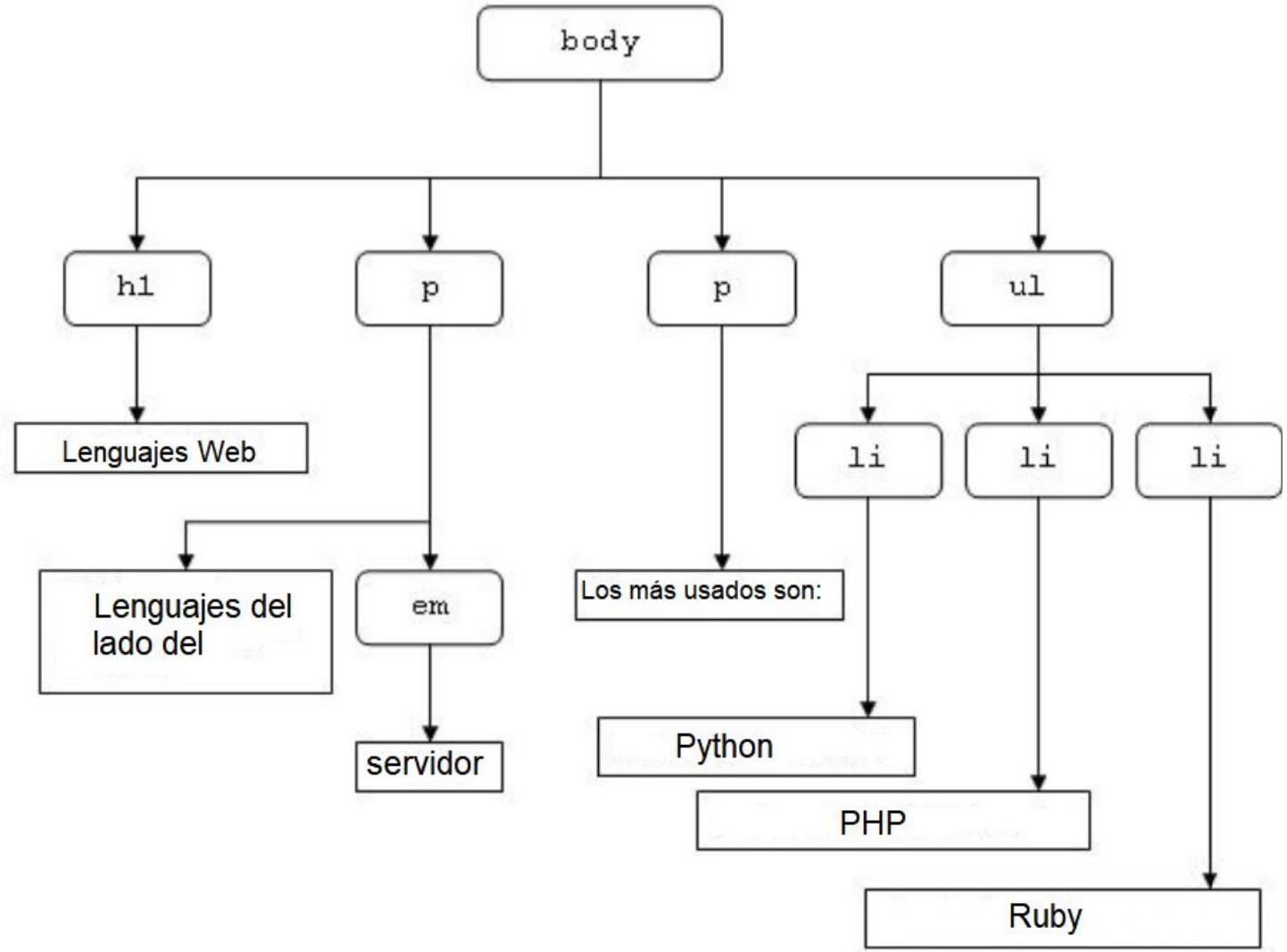
childNodes

Este vector almacena una referencia a cada nodo hijo.
Los nodos pueden ser de tipo: **ELEMENT_NODE** (nodo elemento) o **TEXT_NODE** (nodo texto).

Es importante notar que el texto contenido en un elemento de HTML no pertenece al nodo, sino que se encuentra en otro nodo especial que se llama nodo texto (TEXT_NODE)

Tipos de Nodos nodeType

1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE


body
h1
p
p
ul
Lenguajes Web
li
li
li
Lenguajes del lado del
em
Los más usados son:
servidor
Python
PHP
Ruby

Nodo padre (parentNode)

Si tenemos la referencia a un nodo podemos obtener fácilmente la referencia del nodo padre mediante la propiedad **parentNode**.

Nodos hermanos (nextSibling previousSibling)

Si tenemos la referencia de un nodo podemos fácilmente acceder a los otros nodos que se encuentran en la misma altura dentro del árbol que genera el DOM.
La propiedad **nextSibling** retorna la referencia del nodo hermano que se encuentra inmediatamente más abajo en el archivo HTML, pero a la misma altura si lo pensamos al archivo HTML como un árbol. También existe una propiedad llamado **previousSibling** que retorna la referencia del nodo hermano que se encuentra inmediatamente más arriba en el archivo HTML.
nextSibling y **previousSibling** retornan null en caso de no existir más nodos en dicho nivel.

Recordar: Algo muy importante que hay que tener en cuenta es que si hay un salto de línea entre dos elementos lo interpreta y crea un nodo de texto.

Propiedades firstChild lastChild

Habíamos visto anteriormente que podemos acceder a todos los hijos de un nodo por medio del vector childNodes.

Otra forma de acceder al primer y último nodo hijo es por medio de las propiedades obj.**firstChild** (retorna la referencia del primer hijo, es lo mismo que poner obj.childNodes[0]) y obj.**lastChild** (retorna la referencia del último hijo, es sinónimo de poner obj.childNodes[obj.childNodes.length-1])

Agregar un nodo de texto (appendChild - createTextNode)

Para la creación de un nodo de texto disponemos del siguiente método de la clase document:

```
var nt=document.createTextNode('Texto del nodo’);
```

Para añadirlo luego a un párrafo por ejemplo debemos llamar al método appendChild:

```
var nparrafo=document.getElementById(‘parrafo’);
nparrafo.appendChild(nt);
```

hasChildNodes - removeChild

Un método importante que contiene todo nodo de tipo elemento (son los nodos que apuntan a una marca HTML). Es **hasChildNodes()**, el mismo retorna true en caso que el nodo tenga nodos hijos (sean nodos de tipo texto o nodos de tipo elemento). Por ejemplo:

```
var parrafo=document.getElementById('parrafo');
    if (nparrafo.hasChildNodes()) {
    parrafo.removeChild(parrafo.lastChild);
    }
```

En caso que la condición se verifique verdadera entrará al bloque del if y se ejecutará la llamada a la función **removeChild**. La misma requiere una referencia al nodo texto que queremos borrar. Y lo último que aparece es la propiedad **lastChild** que tiene todo nodo de tipo elemento, el mismo almacena la referencia al último nodo hijo que contiene dicha marca.

Crear y agregar un nodo de tipo elemento (createElement - appendChild)

Para crear nodos de tipo elemento disponemos del método **createElement** que contiene el objeto document:

```
var elemento=document.createElement(“p");
```

Pero recordemos que los nodos elemento no contienen el texto, sino que hay que crear un nodo de texto y añadirlo al nodo de tipo elemento:

```
var texto=document.createTextNode('Hola Mundo');
elemento.appendChild(texto);
```

Por último obtenemos una referencia a un div y añadimos el párrafo que acabamos de crear:

```
var obj=document.getElementById('parrafos');
obj.appendChild(elemento);
```

Para removerlo usamos removeChild() igual que con los nodos texto.

Crear un atributo y agregárselo a un nodo de tipo elemento (setAttribute)

Hay muchas marcas HTML que pueden tener definidos atributos. Muchos de estos son casi obligatorios para su funcionamiento. Imaginemos la marca HTML <a> , si no definimos el atributo href con la dirección del sitio poco sentido tendrá incluirla en la página.

```
var enlace = document.createElement('a');

var nodotexto = document.createTextNode('google');

enlace.appendChild(nodotexto);

enlace.setAttribute('href','http://www.google.com.ar');

document.getElementById(‘div').appendChild(enlace);
```

El primer parámetro es el nombre de la propiedad (en este caso href) y el segundo es el valor que toma la propiedad.

removeAttribute – getAttribute - hasAttribute

La actividad inversa de agregar un atributo a una marca HTML se logra mediante el método **removeAttribute.**

```
enlace.removeAttribute('href');
```

Si queremos conocer el valor de un atributo de un nodo de tipo elemento lo podemos hacer llamando al método **getAttribute**.

```
console.log(enlace.getAttribute('href'));
```

Para controlar si un elemento HTML tiene un cierto atributo disponemos de un método llamado **hasAttribute** al que le pasamos como referencia el atributo a verificar su existencia:

```
enlace.getAttribute(‘href');
```

Accediendo al estilo de una marca HTML.

Las propiedades de estilo CSS de una marca HTML se pueden modificar luego que la página se cargó en el navegador.
Debemos tener en cuenta que para acceder a los estilos, los nombres de las propiedades tienen un ligero cambio, esto es debido a que el carácter '-' no está permitido en JavaScript para la definición de una variable.
A todas las propiedades debemos sacarle el guion.

Una propiedad llamada: font-size:10px;

Desde JavaScript la debemos invocar como: puntero.style.fontSize='60px';

Siendo puntero una referencia a un nodo.

Seleccionar elementos

```
// Seleccionar un elemento por su id
   document.getElementById("unId");

// Seleccionar elementos segun su etiqueta
   document.getElementsByTagName("p");

// Seleccionar elementos que pertenezcan a una clase
   document.getElementsByClassName("miClase");

// Seleccionar elementos con el mismo name (RadioButtons)
   document.getElementsByName("name");

// Seleccionar un elemento dentro de otro
var contenedor = document.getElementById("container");
var parrafo1 = contenedor.getElementsByTagName("p")[0];



// Cambiar la clase de un elemento
   document.getElementById("elemento").className = "miClase";
   document.getElementById("elemento").setAttribute("class", "miClase");
```

Creación de nodos y atributos

```
// Cambiar la clase de un elemento
   document.getElementById("elemento").className = "miClase";
   document.getElementById("elemento").setAttribute("class", "miClase");

// Crear un nuevo elemento
    var parrafo = document.createElement('p');

// Crear un elemento texto
    var texto = document.createTextNode("Esto es un texto");

// Agregar un hijo a un nodo ( en este caso insertamos el texto al parrafo)
    parrafo.appendChild(texto);

// Añadir atributo a un elemento
   parrafo.setAttribute('class', "miClase");

// Quitar un atributo a un elemento
   parrafo.removeAttribute('class', "miClase");
```

Acceso a nodos

```
// Obtener los hijos de un elemento
document.getElementById('ul').childNodes;

// Obtener la referencia del primer hijo
document.getElementById('ul').firstChild;

// Obtener la referencia del último hijo
document.getElementById('ul').lastChild;

// Valor de un nodo hijo
document.getElementById('titulo').childNodes[0].nodeValue;
document.getElementById('titulo').value;

// Sustituir nodo hijo
document.getElementById('div').replaceChild(nodoNuevo, nodoViejo);

// Acceder a un nodo padre
document.getElementById('li').parentNode;
```

Agregar classlist