PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 6: Document Object Model

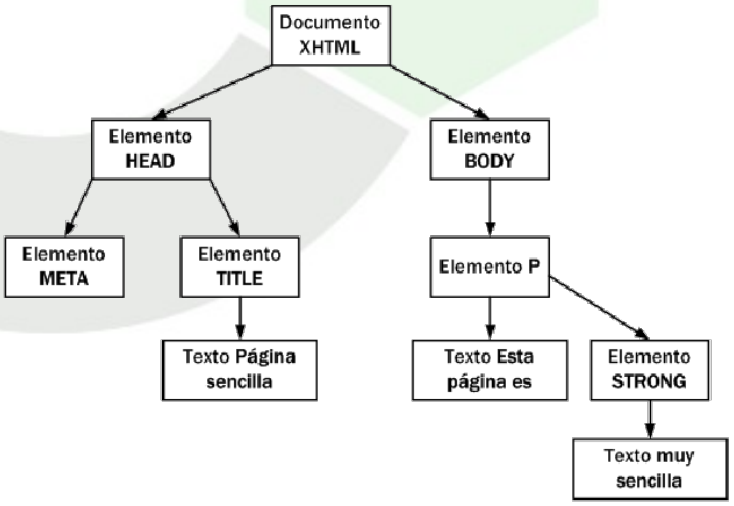
Joan Pons Tugores

Índex de continguts

Document Object Model

El **Document Object Model** és una API independent del llenguatge i de la plataforma que permet accedir i manipular el contingut, l'estructura i l'estil de documents. Així els documents poden ser processats i els resultats d'aquest processament es poden utilitzar per a modificar el document.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTM
<html>
<head>
<meta http-equiv="Content-Type" conten
<title>Página senzilla</title>
</head>
<body>
    <p>
        Esta página és
        <strong>muy senzilla</strong>
    </p>
</body>
</html>
```



Quan parlam de documents ens referim principalment a documents HTML i XML.

DOM defineix una estructura d'arbre que representa el document. Cada element del document és un node de l'arbre. Construeix una representació en memòria del document complet.

Cada node té tants nodes fills com elements conté l'element que representa.

Si un element conté text el node tendrà un node descendent amb el text. Si un element conté un altre element, el node tendrà un node amb aquest element, ...

Avantatges d'aquest model: disposam d'una representació completa a memòria del document amb la que podem treballar. Podem realitzar consultes al document i modificar-lo.

Sense saber-ho ja hem estat treballant amb el DOM. *document* és un objecte del DOM i ja hem estat utilitzant aquest objecte i alguns dels seus mètodes bàsicament

```
document.getElementById(identificador);
```

Aquest mètode ens torna un altre objecte del DOM, en aquest cas un objecte de tipus Element, del qual hem modificat algunes de les seves propietat i utilitzat els seus mètodes.

```
document.getElementById(identificador).value="Hola";

document.getElementById(identificador).setAttribute("disabled",true);
```

Qualsevol canvi efectuat als nodes d'aquest arbre es reflecteix de forma automàtica i immediata en la presentació que fa el navegador d'aquest arbre, és a dir, que en modificar qualsevol part de l'arbre DOM l'usuari veu immediatament els canvis, mantenint la resta de la pàgina inalterada.

En juntar el DOM amb la tecnologia AJAX que veurem a la unitat següent això ens permetrà demanar dades al servidor i mostrar-les a l'usuari sense haver de recarregar tota la pàgina.

Una observació: L'arbre del DOM està completament format quan s'ha acabat de carregar la pàgina, per tant si ens volem assegurar que un node determinat existeixi hem d'esperar. La forma més fàcil d'evitar problemes d'aquest tipus és posar el codi que inicialitzi la pàgina dins un *listener* de l'esdeveniment *onload*.

En aquesta unitat veurem amb una mica més de profunditat el DOM.

Node

DOM representa el document com una jerarquia de nodes, on cada node conté altres nodes, els nodes fills, que representen el seu contingut i on cada node és contingut per un únic node, el seu pare. D'aquesta manera es forma una estructura en forma d'arbre.

La classe Node és l'arrel de la jerarquia d'herència de les diferents classes que representen els distints tipus de nodes.

Els tipus més importants de nodes són:

- **Document**: el node arrel de l'arbre.
- **Element**: un element HTML, tot el que estigui inclòs entre el tag inicial i el final.
- **Attr**: Atribut d'un element.
- **Text**: contingut de text d'un element o atribut.
- **Comment**: un comentari.

Tots aquest tipus hereten totes les propietats i mètodes de la classe Node.

Cada tipus de node pot tenir les seves particularitats. Per exemple, la propietat value d'un atribut torna el seu valor, mentre que el value d'un element torna null, o el nom d'un element torna el text que hi ha dins l'etiqueta inicial (p, a, body), mentre que el nom d'un node Text torna sempre #Text.

Propietats

- .**attributes**: torna una col·lecció dels atributs del node. La propietat length ens diu quans n'hi ha.
- .**childNodes**: Ens torna una llista amb tots els fills del node.
- .**firstChild**, .**lastChild**: tornen el primer i l'ultim fill del node. *Null* si no en té.
- .**nextSibling**, .**previousSibling**: torna el següent i l'anterior germà respectivament. Els espais en blanc o salts de línia que hi ha al document es transformen en nodes de text, pot ser que aquests mètodes tornin un node de text en lloc de l'element que vos esperau trobar.
- .**nextElementSibling**, .**previousElementSibling**: torna el següent i l'anterior germà respectivament, que no sigui un node de text. És preferible utilitzar aquests abans que els anteriors.
- .**nodeName**: Torna el nom del node (li, strong, ...)
- .**nodeValue**: depenent del tipus de node torna o estableix el seu valor. Només funciona amb nodes de tipus *Comment* i *Text*.
- .**parentNode**: torna el node que conté aquest node. *Null* en cas contrari.
- .**textContent**: torna o estableix el contingut textual del node. Si assignam un text al node eliminarem tots els seus fills.
- .**ownerDocument**: torna l'objecte Document on s'ha creat el node.

Mètodes

- .**hasAttributes**: true si el node té atributs.
- .**hasChildNodes**: true si el node te fills.
- .**isEqualNode**: Comprova que dos nodes siguin iguals.
- .**isSameNode**: Comprova que dos nodes siguin el mateix.
- .**cloneNode**: torna una còpia del node.
- .**normalize**: Junta els nodes de text fills adjacents i elimina els nodes de text buits.
- .**appendChild**: afegeix el node que rep com a paràmetre com a darrer fill del node que executa el mètode.
- .**insertBefore**(nou, existent): insereix el node nou abans del node existent dins la llista de fills del node que executa el mètode.
- .**removeChild**(node): Elimina el node de la llista de fills del node que executa el mètode.
- .**remove**(): Elimina el node que l'executa de l'arbre.
- .**replaceChild**(nou, vell): Canvia el node vell per el nou dins la llista de fills.

Document

L'objecte *document* representa la pàgina al navegador. És l'arrel de l'arbre de nodes, ens dona accés a les dades del document. Els altres tipus d'objectes no tenen sentit si no estan associats a un document.

És una especialització de node.

Propietats

- .**docType**: conté la DTD associada al document
- .**documentElement**: torna l'element <html> del document

Mètodes

- .**createElement**(nom): Crea un element amb el nom(p, ul, strong, ..)
- .**createTextNode**(contingut): Crea un node de text.
- .**createAttribute**(nom): Crea un node atribut amb aquest nom. Posteriorment s'ha d'assignar el valor.
- .**getElementById**(id): Torna l'element identificat per id.
- .**getElementsByName**(nom): Torna una **llista** amb tots els elements que tenen el valor rebut com a paràmetre a l'atribut *name*.
- .**getElementsByTagName**(tag): Torna una **llista** amb tots els elements d'aquest tipus.
- .**getElementsByClassName**(classe): Torna una **llista** amb tots els elements que tenguin assignada aquesta classe css.

Creació d'elements

L'únic objecte que permet crear altres objectes és Document. Hem de tenir en compte l'estructura que té cada element i crear tots els seus components. Per exemple, per afegir una opció a una llista hem de:

- Crear un element li

```
const e=document.createElement("li");
```

- Crear un node de Text amb el contingut.

```
const t=document.createTextNode("Bola de drac");
```

- Afegir el node de tipus text a l'element li.

```
e.appendChild(t);
```

- Afegir l'element li a l'element ul

```
const llista=document.getElementById("series");
llista.appendChild(e);
```

Un node només pot tenir un pare. En fer *append* d'un node existent associat a un altre element, eliminam l'associació inicial.

Més informació

https://developer.mozilla.org/en-US/docs/Web/API/Document

http://www.w3schools.com/jsref/dom_obj_document.asp

Element

L'objecte Element és un tipus de Node, podríem dir que és una subclasse de Node. Pot tenir fills de tipus Element, Text, Comment, ... Pot tenir atributs.

Propietats

- .**tagName**: torna el nom de l'element

Mètodes

- .**getAttribute**(nom): torna el valor de l'atribut
- .**hasAttribute**(nom): true si té aquest atribut.
- .**setAttribute**(nom, nouValor): assigna nouValor com a valor de l'atribut. Si l'atribut no existia el crea, i si ja existia canvia el seu valor.
- .**removeAtribute**(nom): elimina l'atribut del node.
- .**setAttributeNode**(atr): assigna aquest node atribut a l'element.
- .**getElementById**(id): Torna l'element descendent identificat per id.
- **getElementsByTagName**(tag): Torna una llista amb tots els elements fills d'aquest tipus.
- **getElementsByName**(nom): Torna una llista amb tots els fills amb aquest *name*.

Més informació

Podeu trobar la referència a

https://developer.mozilla.org/en-US/docs/Web/API/Element

http://www.w3schools.com/jsref/dom_obj_all.asp

Attr

Representa un atribut d'un element. Sempre ha d'estar associat a un node de tipus Element. És una especialització de node.

Propietats

- .**name**: torna el nom de l'atribut.
- .**value**: torna el valor de l'atribut.
- .**ownerElement**: torna l'element al que pertany.
- .**specified**: true si és un atribut especificat per el programador.

Podeu trobar la referència de l'objecte Attr a

https://developer.mozilla.org/es/docs/Web/API/Attr

http://www.w3schools.com/jsref/dom_obj_attributes.asp

DOM i HTML

Quan el DOM accedit és HTML, disposam de mètodes més senzills per manipular i accedir a la seva estructura. Gairebé tots els atributs d'un element HTML es mapegen a propietats:

- .**text**: accedim al contingut d'un element
- .**value**: accedim al value d'un element
- .**className**: accedim al class d'un element
- .**classList**(): Torna una llista amb les distintes classes aplicades a l'element. Mètodes *add* i *remove*.
- .**id**: accedim a l'id d'un element
- .**style**: estil del node. Les distintes propietats que podem assignar a style es poden posar seguint la notació de camell:

```
paragraf.style.backgroundColor="#F00";

paragraf.style.fontWeight=bold;
```