Apunts UD4 (JavaScript II i DOM)

lloc: CIFP Francesc de Borja Moll
Curs: Llenguatges de marques i sistemes de gestió d'informació
Llibre: Apunts UD4 (JavaScript II i DOM)
Imprès per: Alejo Morell Bethencourt
Data: dilluns, 21 desembre 2020, 15:45

Taula de continguts

1. DOM

Quan comencem en el món del desenvolupament web, normalment comencem per aprendre a escriure etiquetatge o marcat HTML i a més, afegir estils CSS per a donar-li color, forma i una mica d'interacció. No obstant això, a mesura que avancem, ens adonem que en una certa forma podem estar bastant limitats.

Si únicament utilitzem HTML/CSS, només podrem crear pàgines «estàtiques» (sense massa personalització per part de l'usuari), però si afegim Javascript, podrem crear pàgines «dinàmiques». Quan parlem de pàgines dinàmiques, ens referim al fet que podem dotar de la potència i flexibilitat que ens dóna un llenguatge de programació per a crear documents i pàgines molt més riques, que brindin una experiència més completa i amb el qual es puguin automatitzar un gran ventall de tasques i accions.

Quan un document HTML es carrega en el navegador, en realitat el navegador construeix un arbre de contingut (*DOM tree*) amb tots els seus elements.

```
El DOM és una interface que permet a programes i scripts, accedir dinàmicament i actualitzar el contingut, estructura i estil d'un document. (definició del W3C)
```

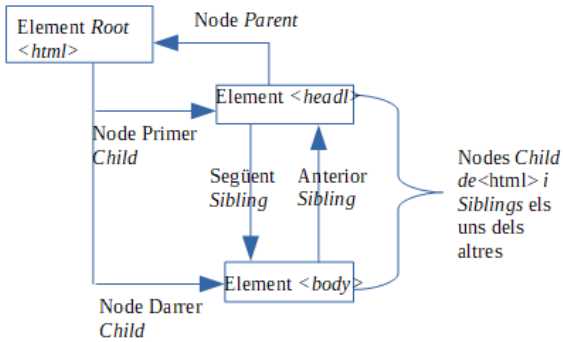
En aquest arbre tots els elements tenen una relació de parentesc amb altres elements. D'aquesta manera podem trobar les següents relacions:

- Pare (*Parent*)
- Fill (*Child*)
- Germà (*Sibling*)
- Germà adjacent (*Adjacent Sibling*)
- Net o descendent (*Descendent*)

Exemple:

```
<html>
 <head>
   <title>DOM Tutorial</title>
 </head>

 <body>
   <h1>Exemple nodes DOM </h1>
   <p>Jerarquies DOM</p>
 </body>
</html>
```



IMPORTANT:

- **El DOM és l'estructura d'objectes que genera el navegador quan es carrega un document i es pot alterar mitjançant Javascript per canviar dinàmicament els continguts i aspectes de la pàgina.**
- **El Model d'Objectes de Document (DOM) permet veure el mateix document d'una altra manera, mostrant el contingut del document com un conjunt d'objectes, podríem dir que defineix l'estructura lògica dels documents i la manera en que s'accedeix i manipula un document.**

2. Programació dirigida per esdeveniments

Javascript permet realitzar scripts amb dos mètodes de programació: seqüencial i basada en esdeveniments. Els esdeveniments de Javascript permeten la interacció entre les aplicacions Javascript i els usuaris. Per exemple: cada vegada que es prem un botó, es produeix un esdeveniment, Cada vegada que es prem una tecla també es produeix un esdeveniment. També hem de saber que, perquè es produeixi un esdeveniment no és obligatori que intervingui l'usuari, ja que per exemple, cada vegada que es carrega una pàgina, també es produeix un esdeveniment.

La versió 3 del DOM inclou l'especificació completa dels esdeveniments JavaScript.

- **Classes d'esdeveniments**

  - Cada element HTML, té definida la seva pròpia llista de possibles esdeveniments.
  - El nom dels esdeveniments es construeix mitjançant el prefix "on", seguit del nom amb anglès de l'acció associada a l'esdeveniment. Per exemple, a l'esdeveniment de clicar damunt un element amb el ratolí, s'anomena "onclick", o l'esdeveniment associat a l'acció de moure el ratolí s'anomena "onmousemove".

La següent llista resumeix els esdeveniments més importants definits per JavaScript:

Tipus d'esdeveniment	**Nom**	**Descripció**
Relacionats amb el ratolí	onclick	Click sobre un element
	ondblclick	Doble click sobre un element
	onmousedown	Se prem un botó del ratolí sobre un element
	onmouseenter	El punter del ratolí entra a l'àrea d'un element
	onmouseleave	El punter del ratolí surt de l'àrea d'un element
	onmousemove	El punter del ratolí s'està movent sobre l'àrea d'un element
	onmouseover	El punter del ratolí se situa damunt de l'àrea d'un element
	onmouseout	El punter del ratolí surt fora de l'àrea de l'element o fora d'un dels seus fills
	onmouseup	Un botó del ratolí s'allibera estant sobre un element
	contextmenu	Se prem el botó dret del ratolí (abans que aparegui el menú contextual)
Relacionats amb el teclat	onkeydown	L'usuari té polsada una tecla (per elements de formulari i body)
	onkeypress	L'usuari prem una tecla (moment just en que es prem) (per elements de formulari i body)
	onkeyup	L'usuari allibera una tecla que tenia polsada (per elements de formulari i body)
Relacionats amb formularis	onfocus	Un element del formulari pren el focus
	onblur	Un element del formulari perd el focus
	onchange	Un element del formulari canvia
	onselect	L'usuari selecciona el text d'un element input o textarea
	onsubmit	Se prem el botó d'enviament del formulari (abans de l'enviament)
	onreset	Se prem el botó reset del formulari
Relacionats amb finestres o frames	onload	S'ha completat la càrrega de la finestra
	onunload	L'usuari ha tancat la finestra
	onresize	L'usuari ha canviat la grandària de la finestra
Relacionats amb animacions i transicions	animationend, animationiteration, animationstart, beginEvent, endEvent, repeatEvent, transitionend	

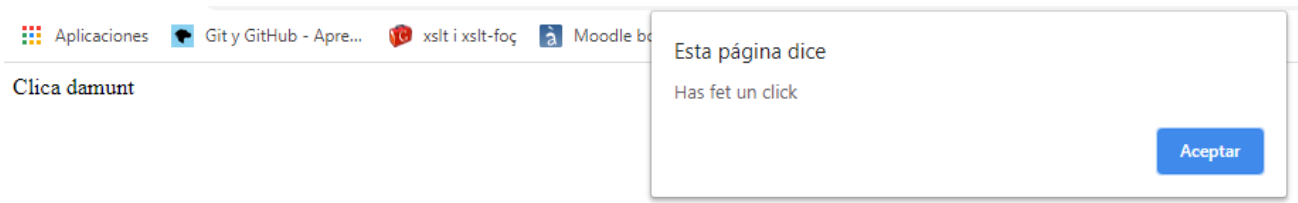
Relacionats amb la bateria	chargingchange, chargingtimechange, dischargingtimechange, levelchange
Relacionats amb cridades de telefonia	alerting, busy, callschanged, connected, connecting, dialing, disconnected, disconnecting, error, held, holding, incoming, resuming, statechange
Relacionats amb canvis al DOM	DOMAttrModified, DOMCharacterDataModified, DOMContentLoaded, DOMElementNameChanged, DOMNodeInserted, DOMNodeInsertedIntoDocument, DOMNodeRemoved, DOMNodeRemovedFromDocument, DOMSubtreeModified
Relacionats amb l'arrossegament d'elements (drag and drop)	drag, dragend, dragenter, dragleave, dragover, dragstart, drop
Relacionats amb video i audio	audioprocess, canplay, canplaythrough, durationchange, emptied, ended, ended, loadeddata, loadedmetadata, pause, play, playing, ratechange, seeked, seeking, stalled, suspend, timeupdate, volumechange, waiting, complete
Relacionats amb la connexió a internet	disabled, enabled, offline, online, statuschange, connectionInfoUpdate

Exemples:

Exemple 1: Si fem click damunt un text, mostrarà una finestra emergent

```
<html>
  <head>
    ...
  </head>
 <body>
  <span onclick="alert('Has fet un click');"> Clica damunt </ span>
 </body>
</html>
```

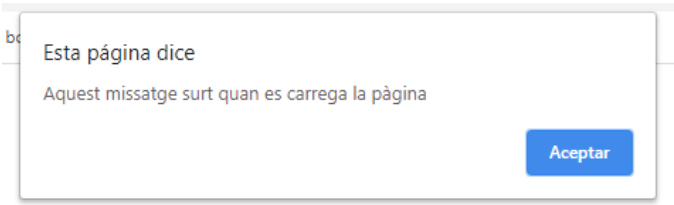
Resultat:



Exemple 2: Carregam al body un funció que genera una missatge amb finestra emergent

```
<html>
  <head>
    <meta charset = "UTF-8"/>
  </head>
  <body onload='missatge()'>
    <script>
       function missatge(){
          alert('Aquest missatge surt quan es carrega la pàgina');
        }
    </script>
  </body>
</html>
```

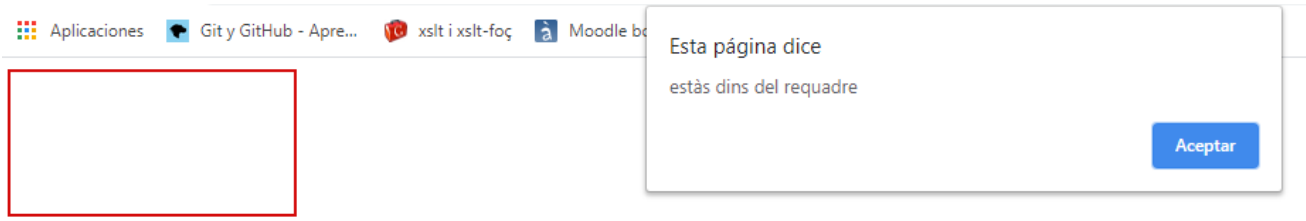
Resultat:



Exemple 3: Quan passam el ratolí sobre el rectangle es veu un missatge

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         div {
           width: 200px;
           height: 100px;
           border: 2px solid rgb(212, 13, 13);
            }
      </style>
   </head>
   <body>
      <div onmousemove="ratoli()"></div>
      <p>Passa el ratolí sobre el rectangle de dalt i veuràs un missatge.</p>
      <script>
          function ratoli() {
             alert('estàs dins del requadre');
          }
      </script>
   </body>
</html>
```

Resultat:



Passi el ratolí sobre el rectangle de dalt i obtingui les coordenades de l'punter de l'ratolí.

3. Aprendre a manejar el DOM

El model d'objecte de document (DOM) és una interfície de programació per als documents HTML i XML.

- És una representació completament orientada a l'objecte de la pàgina web i pot ser modificat amb un llenguatge de script com JavaScript.
- Facilita una representació estructurada del document i defineix de quina manera els programes poden accedir, a fi de modificar, tant la seva estructura, estil i contingut.
- Dóna una representació del document com un grup de nodes i objectes estructurats que tenen propietats i mètodes. Essencialment, connecta les pàgines web a scripts o llenguatges de programació.
- Va ser dissenyat per ser independent de qualsevol llenguatge de programació particular.
- Totes les propietats, mètodes i esdeveniments disponibles per a la manipulació i la creació de pàgines web està organitzat dins d'objectes.

```
Mitjançant el DOM podem accedir, modificar, crear, afegir o eliminar qualsevol objecte ja sigui del propi document, etiquetes, atrib
uts, esdeveniments o estil de qualsevol etiqueta html, el css o inclús el propi javascript.
```

DOM i JavaScript

El DOM no és un llenguatge de programació, però sense ell el llenguatge JavaScript no té cap model o noció de les pàgines web, de les pàgines XML ni dels elements amb els quals és usualment relacionat. Cada element (per exemple, tot el document, el títol, les taules, els títols de les taules, etc.), és part del model d'objecte del document, d'aquesta manera, es pot accedir i manipular-los (mitjançant el DOM) i amb un llenguatge d'escriptura com és JavaScript.

Al començament, JavaScript i DOM estaven hermèticament enllaçats, però després es van desenvolupar com a entitats separades.

- **Com s'accedeix a DOM?**

  No s'ha de fer res especial per començar a utilitzar el DOM. Els diferents navegadors tenen directrius DOM diferents, i aquestes directrius tenen diversos graus de conformitat a l'actual estàndard DOM, però tots els navegadors web empren el model d'objecte de document per fer accessibles les pàgines web a l'script.

  En Javascript, la forma d'accedir al DOM, és a través dels seus nodes, anem a veure els tipus de nodes que hi podem trobar.

- **Tipus de nodes del DOM**

  L'especificació completa de DOM defineix 12 tipus de nodes, encara que les pàgines HTML, es poden manejar només en 4 o 5 tipus de nodes:
  - **Document:** El node document és el node arrel, a partir del qual deriven la resta de nodes.
  - **Element**: Són els nodes definits per etiquetes html. Per exemple, una etiqueta "div" genera un node. Si dins del "div" tenim 3 etiquetes "p", aquestes etiquetes defineixen nodes fills de l'etiqueta "div".
  - **Text**: El text dins d'un node element es considera un nou node fill de tipus text.
  - **Attr**: Els atributs de les etiquetes defineixen nodes, encara que amb JavaScript no les veurem com a nodes, si no que es consideren informació associada al node de tipus element.
  - **Comment:**: Els comentaris i altres elements com són les declaracions "doctype" a la capçalera dels documents HTML generen també nodes.

- **Cóm s'accedeix als nodes?**

  DOM proporciona dos mètodes alternatius per accedir a un node específic: a través del node pare (l'arrel) o amb accés directe. És més ràpid accedir de forma directa, per tant veurem les funcions per accedir directament.
  És important saber que per accedir, modificar o eliminar algun node, només es pot fer quan l'arbre DOM s'ha construït per complet, és a dir, quan la pàgina s'ha carregat per complet.

- **Els mètodes d'accés directe al DOM són:**

  Mètodes del node «document» que permeten cridar als nodes segons les seves característiques:

  - **document.getElementById()**: És el mètode o funció més emprada quan es fan aplicacions web dinàmiques. Permet accedir directament a un node per llegir i/o modificar les seves propietats. Retorna un element HTML que té un atribut que coincideix amb el que s'ha indicat al paràmetre de la funció.

Exemple: Canviar el color d'un encapçalament de la pàgina

```
<html>
   <head>
      <title>Exemple getElementById</title>
   </head>
   <body>
      <h2 id="clic">Text que ha de canviar de color</h2>
      <button onclick="canviarcolor('orange');">Taronja</button>
      <button onclick="canviarcolor('red');">Vermell</button> <!-- a través de "onclick", cridam la funció JavaScript "canviarcolor-->
      <script>
         function canviarcolor(pcolor) {
            var x = document.getElementById('clic'); // Posa dins la variable "x" l'element HTML gràcies al id="clic"
            x.style.color = pcolor; //posam aquest element en el color que indica el paràmetre "pcolor"
          }
       </script>
    </body>
 </html>
```

Nota: Els estils des de Javascript es poden posar emprant la propietat style com veim a l'exemple.
Podem consultar les equivalències: https://developer.mozilla.org/es/docs/Web/CSS/CSS_Properties_Reference

- **document.getElementsByTagName()***:* Obté en forma d'array tots els elements de la pàgina HTML on l'etiqueta sigui igual que el paràmetre que se li passa a la funció.

  Sintaxi:
  ```
  var variable = document.getElementsByTagName(element);
  ```

  El valor que retorna és un objecte de tipus "nodelist" (similar a l'array) amb tots els nodes que tenen la etiqueta igual al paràmetre que passa el mètode. Per obtenir el valors d'aquest objecte, es farà igual que amb els arrays:

  Exemple: Obtenir els encapçalaments de la pàgina

  ```
  var capçalera = document.getElementsByTagName("h1");
  ```

  Per recorre l'objecte i obtenir totes les etiquetes d'encapçalament de la pàgina podriem fer amb un for:

  Exemple: Canviar el color del fons dels h1
  ```
  <html>
     <head>
        <meta charset="UTF-8">
        <title>h1 en vermell</title>
     </head>
     <body>
        <h1>Primer encapçalament</h1>
        <h1>Segon encapçalament</h1>
        <script>
           var encap = document.getElementsByTagName("h1");
           for (var i=0; i< encap.length; i++) {
             encap[i].style.backgroundColor = 'red'
           }
        </script>
     </body>
  </html>
  ```

- **document.getElementsByClassName():** Retorna tots els elements amb el nom de la classe especificada.

  Sintaxi:
  ```
  var x = document.getElementsByClassName(classe); //on x és de tipus nodelist (array).
  ```

Exemple: Canviar l'estil d'una capçalera amb classe i d'un paràgraf amb classe

```
<html>
   <head>
      <meta charset="UTF-8">
      <title>Exemple</title>
   </head>
   <body>
      <h3 class="canvi">Capçalera amb classe</h3>
      <p class="canvi">Paràgraf amb classe</p>
      <button type=button onclick="classes()"> clica aquí</button>
      <script>
         function classes() {
             var classe=document.getElementsByClassName('canvi');
             for (i = 0; i < classe.length; i++){
                classe[i].style.color = "orange";
                classe[i].style.fontSize = "25px";
                classe[i].style.backgroundColor = "yellow";
                classe[i].style.borderStyle = "solid";
              }
          }
        </script>
    </body>
  </html>
```

Nota: les propietats del DOM style se poden consultar per exemple: https://www.w3schools.com/jsref/dom_obj_style.asp

- **document.createElement():** Crea un tipus d'element.

Sintaxi:

```
var capçalera = document.createElement(element);
```

Exemple: Cream un element botó al body

```
<html>
   <body>
        <h1>Fes clic i es crearà un botó</h1>
        <button onclick="funciocrearbotó()">Fes click</button>
        <script>
            function funciocrearbotó() {
                var boto = document.createElement("BUTTON"); // Cream l'element botó
                document.body.appendChild(boto); // Aquest és un altre mètode que afegeix al body el botó que hem creat
            }
        </script>
    </body>
</html>
```

- **document.createTextNode():** Crea un node de tipus text.

Sintaxi:

```
var text = document.createTextNode("text");
```

Exemple: Afegeix element a una llista

```
<html>
   <body>
        <h1>Fes clic i s'afegirà elements a una llista</h1>
        <button onclick="afegirtext()">Fes click</button>
        <ol id="llista">
            <li>Un element</li>
            <li>Un altre element</li>
        </ol>
        <script>
            function afegirtext()
            {
                var elementnou = document.createElement("li"); // Crea un node <li>
                var text = document.createTextNode("un altre element"); // Crea un text node
                element.appendChild(text); // Afegeix text a <li>, funció que veurem a continuació
                document.getElementById("llista").appendChild(elementnou);
            }
        </script>
    </body>
</html>
```

Els dos mètodes anteriors, ens serveixen per crear elements i nodes de text, però si no les afegim a l'arbre, aquests elements no representaran bé el contingut de la pàgina web. Per poder afegir aquests elements tenim els següent mètodes:

- **appendChild():** Ens permet afegir un nou fill a l'arbre. Com podrem veure a la sintaxi aquest mètode se vincula directament al node que volem afegir el nou node.

Sintaxi:

```
elementpare.appendChild(nouelement);
```

Exemple: Afegir un paràgraf

```
<html>
    <head>
       <meta charset="UTF-8">
    </head>
    <body>
       <div>
          <p id="existent">Element  existent</p>
       </div>
       <button onclick=afegir()>clica</button>
       <script>
          function afegir(){
             alert ("afegirem un paràgraf nou");
             if (confirm("clica si vols fer-ho")==true) {
                var nouelement = document.createElement('p');
                nouelement.appendChild(document.createTextNode('El text de l'element nou'));
                var elementpare = document.getElementById('existent');
                elementpare.appendChild(nouelement);
              }
          }
       </script>
     </body>
</html>
```

- **insertBefore:** Ens permet afegir el nou element abans d'una altre.

Sintaxi:

```
elementpare.insertBefore(nodenou, elementfill);
```

Exemple: Inserim un paràgraf entre dos encapçalaments

```
<html>
  <body>
    <div id="undiv">
      <h1>Primer encapçalament</h1>
      <h1>Segon encapçalament</h1>
    </div>
    <script>// Cream un paragraf
      var noup = document.createElement('p'); //cream i guardam a una variable el nou paragraf
      noup.appendChild(document.createTextNode('Nou paragraf')); //Afegim el text
      var identif = document.getElementById('undiv'); // Guardam a una variable el punt a partir del qual hem de inserir
      var elementposterior = document.getElementsByTagName('h1')[1]; // Guardam a una variable l'element que se convertirà e
n element posterior al nou *
      identf.insertBefore(noup,elementposterior); //inserim el nou element
    </script>
  </body>
</html>
```

* document.getElementsByTagName('h1')[1] : Aquesta sintaxi ens permet triar el segon element de la llista que retorna el mètode document.getElementsByTagName

- **replaceChild():** Permet reemplaçar un element existent.

- **removeChild():** Permet esborrar un element.

Exemple: Esborrar elements d'una llista

```
<!DOCTYPE html>
<html>
<body>
    <ul id="llista">
        <li>Ana</li>
        <li>Pep</li>
        <li>Lluís</li>
        <li>Leire</li>
    </ul>
    <p>Clica al botó per eliminar elements a la llista</p>
    <button onclick="eliminar()">Fes click</button>
    <script>
        function eliminar() {
           var list = document.getElementById("llista"); //Guardam a una variable el punt a partir del qual hem d'eliminar
           list.removeChild(list.childNodes[0]); // Esborram el primer element. childNodes és una propietat que veurem a con
tinuació
        }
    </script>
</body>
</html>
```

- **cloneNode()**: Permet fer una copia exacte d'un element.

- **Propietats del nodes:**
  - **childNodes**: Ens retorna un array amb tots els nodes fill d'un element.

    Sintaxi:
    ```
    var variable = document.element.chilNodes;
    ```

    Exemple: Obtenir els nodes de l'element body

    ```
    var c = document.body.childNodes;
    ```

    Nota: Per accedir a cadascun dels fills haurem de fer-ho com un array ([])

  - **firstChild**: És de només lectura i retorna el primer node fill d'un node, i null si no té fills.

```
Exemple: Obtenir el primer fill d'un element span

<p id="ex">
  <span> això és un span </span>
</p>
<script>
  var x = document.getElementById('ex');
  var fill = x.firstChild;
</script>
```

- **innerHTML**: Ens permet recuperar el contingut d'un element o inserir nou contingut.

```
Sintaxi:
var contingut = document.getElementById(valor_atribut_id).innerHTML;
```

```
Exemple: Canviar el text d'un paràgraf

<!DOCTYPE html>
<html>
<body>
    <p id="inner" onclick="canviar()">Clica damunt aquest paràgraf per canviar el text.</p>
    <script>
        function canviar() {
          document.getElementById("inner").innerHTML = "Nou text";
        }
    </script>
</body>
</html>
```

- **lastChild**: Ens retorna el darrer fill directe de l'element (funciona com el firstChild).

- **nextSibling**: Ens retorna el germà següent al node o element que s'especifica.

Propietats dels elements

- **Modificar atributs.** Hi ha quatre mètodes per modificar atributs d'elements:

  - **hasAttributte():** Comprova si un element té un atribut concret. Retorna un booleà.

```
Sintaxi:
element.hasAttribute('nom_atribut');
```

```
Exemple: Comprovar que un element botó té un atribut determinat

<html>
  <body>
    <p>Clica para saber si l'element botó té l'element onclick.</p>
    <button id="boto" onclick="has()">clica</button>
    <p id="demo"></p>
    <script>
       function has()
       {
         var x = document.getElementById("boto").hasAttribute("onclick");
         if (x==true) {
            document.write("el botó té l'atribut onclik")
         }
         else {
          document.write("El botó no té l'atribut onclick")
         }
       }
    </script>
   /body>
</html>
```

  - **getAttribute():** Mostra el valor d'un atribut especificat o null.

```
Sintaxi:
element.getAttribute(nom_atribut);
```

```
Exemple: Mostrar valor d'un atribut

<!DOCTYPE html>
<html>
<head>
   <style>
      .vermell {
         color: red;
         background-color: rgb(0, 255, 255)
      }
   </style>
</head>
<body>
    <h1 class="vermell">Provam getAttribute</h1>
    <h1> Un mètode del dom</h1>
    <button onclick="comprovar()">Clica</button>
    <p id="afegirparagraf"></p>
    <p>afegirem mitjançant un paragraf, el valor de l'atribut class que du el primer encapçalament</p>
    <script>
        function comprovar() {
            var x = document.getElementsByTagName("h1")[0].getAttribute("class");  //guardam dins d'x el valor de l'atri
but class del 1er element h1
            document.getElementById("afegirparagraf").innerHTML = x; //innerHTML ens permet afegir contingut a un elemen
t
        }
    </script>
</body>
</html>
```

- **setAttribute():** Agrega o actualitza el valor d'un atribut especificat.

```
Sintaxi:
element.setAttribute(nom, valor);
```

```
Exemple: Aplicar classe

<!DOCTYPE html>
<html>
<head>
   <style>
      .vermell {
         color: red;
     }
</style>
</head>
<body>
   <h1>Aplicarem una classe a aquest encapçalament</h1>
   <button onclick="aplicarclasse()">clica</button>
   <script>
      function aplicarclasse() {
        document.getElementsByTagName("h1")[0].setAttribute("class", "vermell");
      }
   </script>
</body>
</html>
```

- **removeAttribute():** Elimina un atribut d'un element.

```
Sintaxi:
element.removeAttribute(valoratribut);
```

- **Modificar classes:** A JavaScript, tenim les propietats className i classList per treballar amb atributs de classes.
  - **className:** Obté o estableix un valor de classe.

```
Sintaxi:
element.className;
```

Exemple: Afegir una classe a h1

```
<html>
   <head>
      <meta charset="UTF-8">
      <title>Afegir classe</title>
      <style>
        .estilnou { color: green; }
      </style>
   </head>
   <body>
     <h1 id="ex">Afegim classe</h1>
     <script>
        document.getElementById("ex").className = "estilnou";
     </script>
   </body>
</html>
```

- **classList.add()**: Agrega un o més valors de classe.

Sintaxi:

```
element.classList.add(valor);
```

Exemple: afegir classes amb classList.add

```
<html>
<head>
   <style>
      .estil {
          width: 300px;
          height: 50px;
          background-color: red;
      }
      .estil2 {color: red; }
   </style>
</head>
<body>
   <button onclick="afegirclasse()">clica</button>
   <main id="nav"> <p>Text del main</p> </main>
   <script>
      function afegirclasse() {
         document.getElementById("nav").classList.add("estil","estil2");
      }
   </script>
</body>
</html>
```

- **classList.toggle()**: Pot tenir un o dos arguments, si n'hi ha un, alterna el valor de la classe, és a dir, si la classe existeix l'elimina i retorna false, i si no existeix l'afegeix i retorna true. Si s'indiquen els dos arguments, si el segon argument s'avalua com a true, s'afegeix la classe i si s'avalua com a false, elimina la classe.

Sintaxi:

```
element.classList.toggle(classe, valortrueofalse);
```

- **ClassList.contains():** Comprova si el valor de classe existeix.

Sintaxi:

```
element.classList.contains('active');
```

- **classList.replace():** Substitueix un valor de classe existent per un nou.

Sintaxi:

```
element.classListreplace('old','new');
```

- **classList.remove():** Elimina un valor de classe.

Sintaxi:

```
element.classList.remove('active');
```