
DOM ET EVENTS
Ludovic, Ayyoub, Alexandre, Nicolas
FMS

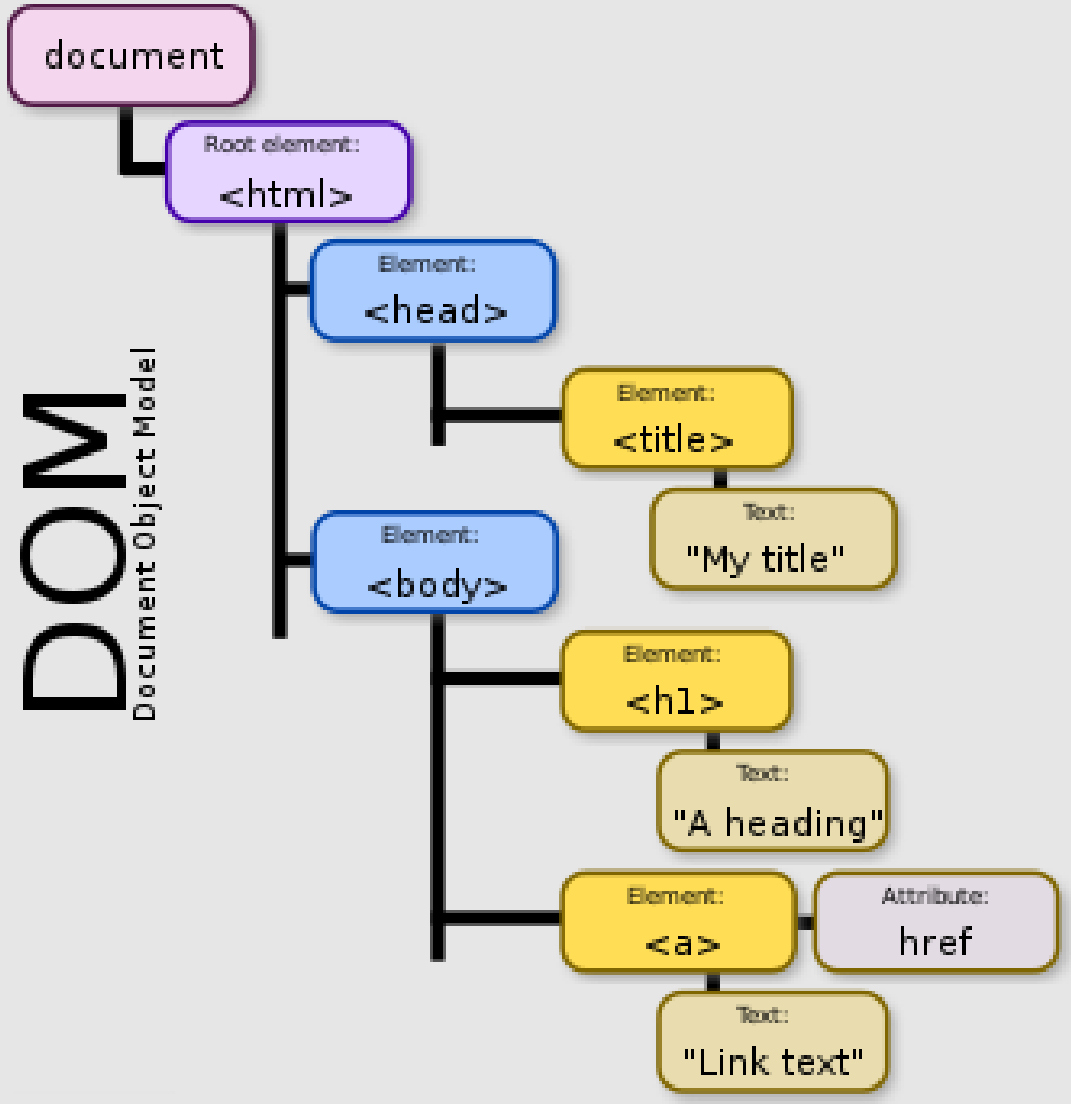
Document Object Model

Quand une page web est téléchargée , le navigateur crée un DOM de la page.

Un HTML DOM model est construit comme un arbre d'objets, comme ci-contre :

Avec le DOM, JS peut accéder et changer les éléments d'un document HTML, ainsi que les événements :

Par exemple, la mise en page ou la réaction au clique de souris.



L’object « document » et ces méthodes

- **getElementById(x)**. Renvoie un objet élément représentant l’élément à l’index « x »
- **innerHTML**. Définir une valeur pour vous permettre de remplacer le contenu d’un élément.
- **getElementsByName(x)**. Renvoie une collection d’éléments avec un attribut « nom » dans le document.
- **item(n)**. Revoie l'élément en position n dans une NodeList.
- **firstChild**. Renvoie le premier noeuf enfant de l’arbre ou null, s’il n’y en a pas.
- **nextSibling**. Renvoie le nœud (node) suivant immédiatement après le nœud spécifié dans la liste

Méthodes de modification essentielles

- **createElement(type, nom)**. Crée un élément et renvoie un objet Element (un type de Node).
- **appendChild(Node)**. Ajoute un élément à l'instance, en tant que dernier enfant.
- **insertBefore(new_Node, ref_Node)**. Insertion avant la référence d'un node en tant qu'enfant
- **removeChild(Node)**. Supprime un Node
- **setAttribute(nom, valeur)**. Ajoute un attribut à l'élément.

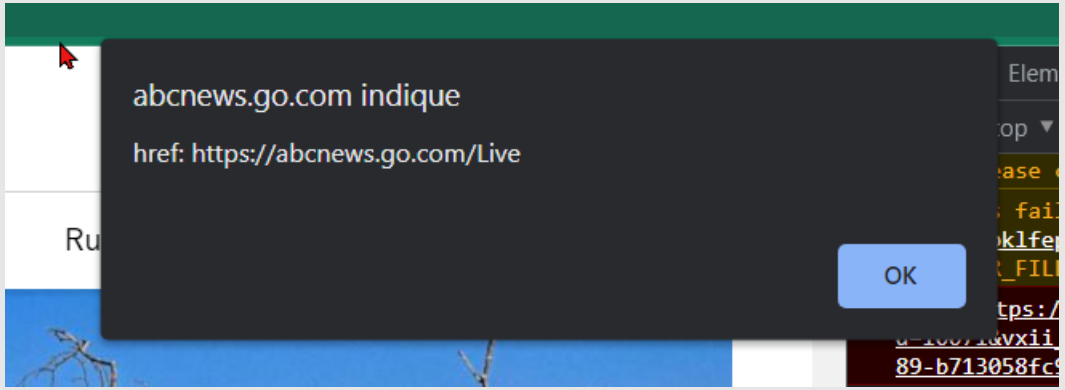
```
<html>
  <head>
    <script>
       // run this function when the document is loaded
       window.onload = function() {

         // create a couple of elements in an otherwise empty HTML page
         const heading = document.createElement("h1");
         const heading_text = document.createTextNode("Gros titre !");
         heading.appendChild(heading_text);
         document.body.appendChild(heading);
       }
    </script>
  </head>
  <body>
  </body>
</html>
```

Comment utiliser DOM avec JS?

Cet exemple parse une page web pour trouver des liens et les afficher.

- *document* est un objet défini dans le coeur de dom pour représenter le document.
- *getElementsByTagName* est une méthode de l'objet qui construit un tableau avec chaque balise correspondant au paramètre, "a" en l'occurrence.
- *href* est une propriété du DOM HTML.
- *anchorList* est une variable déclarée.

```
var anchorList = document.getElementsByTagName("a") ;
for (var i = 0; i < anchorList.length ; i++)
{
 alert("href: " + anchorList[i].href + "\n");
}
```



Exemple d'altération du DOM

Dans cet exemple, on peut voir la page web être altérer par le biai d'un fonction dans le script.

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Page</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

1 Noeud "p" avec l'id "demo"

2 La fonction dans le script, imprime un texte HTML dans la document au noeud associé

Les évenements

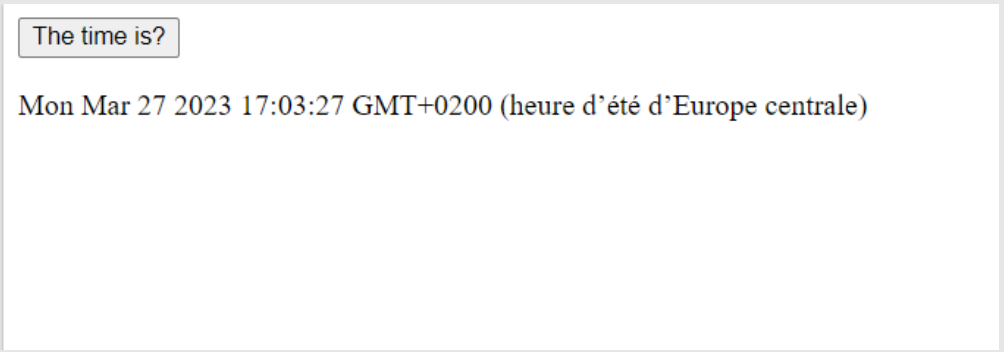
Définition : Les évènements HTML sont les « choses » qui arrivent sur les éléments HTML.

Par exemple: un clic de souris sur un bouton, la page a finit d'être télécharger, un champ a été rempli

Quand JS est utilisé dans des pages HTML, JS peut « réagir » à ces évènements.

Exemple : Ici un bouton cliqué renvoie l'heure, avec détail du méridien

```
<!DOCTYPE html>
<html>
<body>

<button onclick="document.getElementById('demo').innerHTML=Date()">The time is?</button>

<p id="demo"></p>

</body>
</html>
```



Evenements communs

Evenements	Description
onchange	Un élément HTML a été changé
onclick	L'utilisateur a cliqué l'élément HTML
onmouseover	L'uilisateur a survolé l'élément HMTL
onmouseout	L'utilisateur a éloigné la souris de l'élément
onkeydown	L'utilisateur appuie sur une clef du clavier
onload	Le navigateur a finit de télécharger la page

EventTarget.addEventListener()

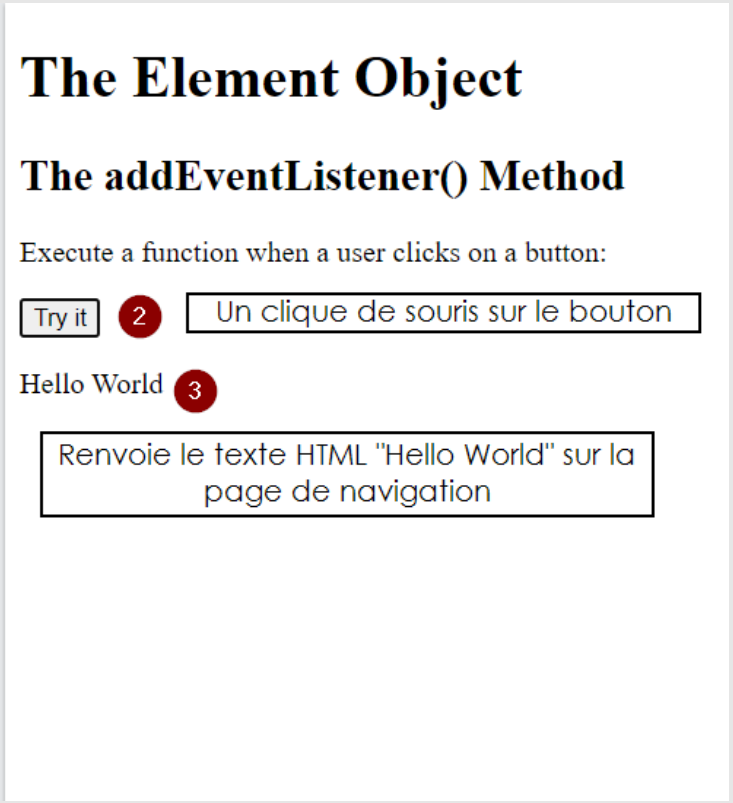
- Une méthode qui attache une fonction à appeler chaque fois que l'évènement spécifié envoyé à la cible
- Cibles courantes : un élément, le document lui-même ou une fenêtre.
- Agit en ajoutant une fonction ou un objet qui implémente « EventListener » à la liste des gestionnaire d'évènements spécifiés sur la cible (« EventTarget ») à partir de laquelle il est appelé.
- Syntax : « `element.addEventListener(event, function, useCapture)` »
- Paramètres:

Paramètres	Description
event	Le nom de l'évenement. Nécéssaire
function	La fonction à lancer. Nécéssaire
useCapture	Modèle de capture. Optionnel

Exemple de « EventListener »

```html
<!DOCTYPE html>
<html>
<body>

<h1>The Element Object</h1>
<h2>The addEventListener() Method</h2>

<p>Execute a function when a user clicks on a button:</p>

<button id="myBtn">Try it</button>

<p id="demo">

<script>
const element = document.getElementById("myBtn");
element.addEventListener("click", myFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "Hello World";
}
</script>

</body>
</html>
```

1 Ici la fonction écrite dans le script pour écrire du texte lors d'un clique



Ce qu'il faut retenir :

- Le DOM est un object généré par le navigateur, doté d'une interface de programmation, par lequel on peut altérer le HTML via des scripts et/ou des languages de programmation.
- Le DOM représente le document comme un ensemble de nœuds et d'ojets possédant des propriétés et des méthodes.
- Les nœuds sont des points uniques dans l'arbre que représente le DOM.
- Les nœuds peuvent être associées à des gestionnaires d' évènements.
- Une fois un évenement est déclenché, les gestionnaires d' évènements sont exécutés.

Ressources:

- https://www.w3schools.com/
- https://en.wikipedia.org/wiki/Document_Object_Model
- https://developer.mozilla.org/fr/
- https://stackoverflow.com/