Javascript cours n°3

Javascript et le navigateur

Le Document Object Model

Les événements

Insertion de code JS dans la page

- Directement dans le code HTML, balise `script`

- Deux possibilités :
- Code dans le fichier html (à éviter)
- Code dans un ou plusieurs fichiers js (à préférer)

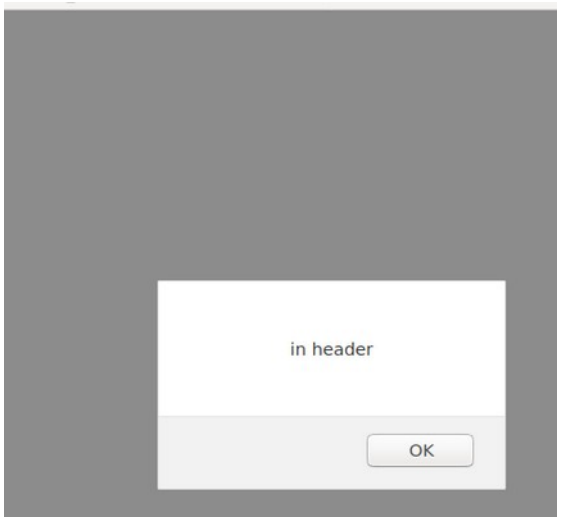
- Le code est exécuté au moment où il est lu

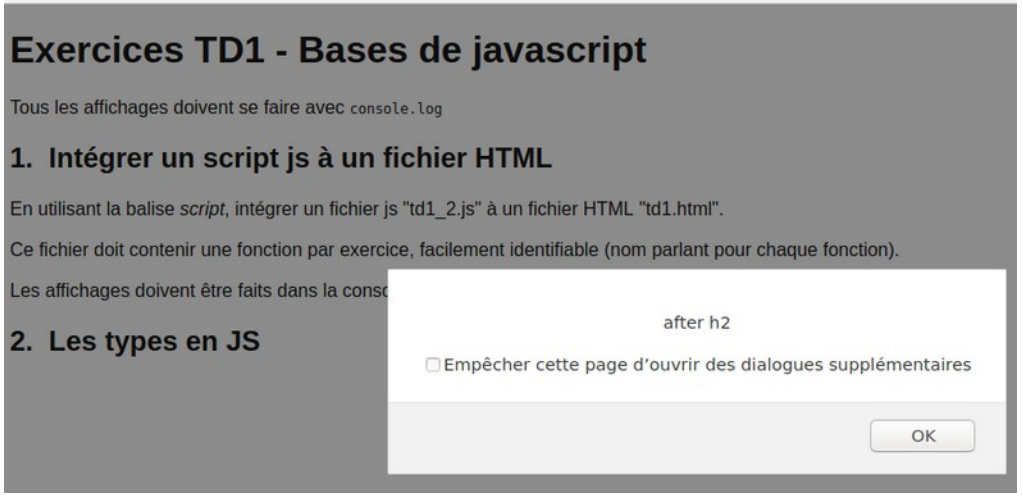
- Il interrompt donc l'interprétation du html
  → ralentit l'affichage de la page
  → qui est un facteur pris en compte en SEO

Insertion de code JS dans la page

- Exemples :

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sujet du td 1</title>
    <link rel="stylesheet" href="sujet
    <script>alert('in header')</script
</head>
<body>
<h1>Exercices TD1 - Bases de javascrip
<p>Tous les affichages doivent se fair
<h2>Intégrer un script js à un fichier
<p>En utilisant la balise <em>script</
<p>Ce fichier doit contenir une fonctio
<p>Les affichages doivent être faits d

<h2>Les types en JS</h2>
<script>alert('after h2')</script>
```





Insertion de code JS dans la page

- Script avec inclusion de fichier
- 2 version possibles
- La seconde (attribut `type`) est considérée comme obsolète

```
<body>
    <h1>...</h1>

    <script src="td1_3.js"></script>
    <script type="text/javascript" src="td1_3.js"></script>
</body>
```

Sécurité

- Les navigateurs exécutent le code JS dans une zone sécurisée : *sandbox*
- Aucun accès au disque dur local du visiteur du site
- Pas d’accès aux variables de registre Windows
- Seule la page html peut être modifiée

Document Object Model

- DOM = représentation de la page HTML dans laquelle le script JS est inséré
- Il s'agit d'un arbre, dans lequel on peut naviguer
- Chaque élément est un `Node`

Document Object Model

- Exemple d'accès aux éléments d'une page

```html
<body>
    <h1>LP WMCE</h1>
    <h2>Rentrée</h2>
    <p>Lundi 20 septembre, à 14h, en salle B09.</p>

    <h2>Liste des étudiants</h2>
    <ul id="myUL">
        <li>Barel Antoine</li>
        <li>Bonazzi Pierre Jean</li>
        <li>Boukada Adel</li>
        <li>Breuil Allan</li>
        <li>Calvet Yann</li>
    </ul>
    <script src="wmce.js"></script>
</body>
```

```
const explorePage = (noeud) => {

    for (let fils of noeud.childNodes) {
        if (fils.nodeType === Node.ELEMENT_NODE) {
            console.log(fils.nodeName);
            explorePage(fils)
        } else if (fils.nodeType === Node.TEXT_NODE) {
            console.log(fils.nodeName, fils.nodeValue,
                '(fils de ' + fils.parentNode.nodeName + ')')
        }
    }
}
```

```
#text
     (fils de BODY)
H1
#text LP WMCE (fils de H1)
#text
     (fils de BODY)
H2
#text Rentrée (fils de H2)
#text
     (fils de BODY)
P
#text Lundi 20 septembre, à 14h, en salle B09.
(fils de P)
#text

     (fils de BODY)
H2
#text Liste des étudiants (fils de H2)
#text
     (fils de BODY)
UL
#text
         (fils de UL)
LI
#text Barel Antoine (fils de LI)
#text
         (fils de UL)
```

Navigation dans le DOM

- `childNodes, parentNode` (page précédente)
- `children` (seulement les éléments nœuds, pas le texte)
- `firstChild, lastChild`
  `firstElementChild, lastElementChild`
- `nextSibling, previousSibling`
  `nextElementSibling,previousElementSibling`
  (voisins = éléments au même niveau)

Navigation dans le DOM

- Accès direct à un ou plusieurs éléments :
- Par son attribut `id`
- Par son attribut `name`
- Par son type de tag
- Par une de ses classes css

```
Titre de la liste :  Liste des étudiants
Nombre d'étudiants :  5
▸ NodeList []
▸ HTMLCollection { 0: span.local , length: 1 }
```

```
console.log("Titre de la liste : ", document.getElementById( elementId: "titre_liste").innerHTML);
console.log("Nombre d'étudiants : ",
    document.getElementById( elementId: "liste").getElementsByTagName( qualifiedName: "li").length);
console.log(document.getElementsByName( elementName: "unNom"));
console.log(document.getElementsByClassName( classNames: "local"));
```

```
<body>
    <h1>LP WMCE</h1>
    <h2>Rentrée</h2>
    <p>Lundi 20 septembre, à 14h, <span class="local">en salle B09</span>.</p>

    <h2 id="titre_liste">Liste des étudiants</h2>
    <ul id="liste">
        <li name="unNom">Barel Antoine</li>
```

Navigation dans le DOM

- Toutes ces fonctions sont applicables sur le document ou uniquement sur un élément du document
- Exemple :

```
<body>

<div id="undiv">
    <button id="btnInDiv">Clic ici</button>
</div>
<button id="envoi" onclick="verifFormulaire()">Envoyer</button>

</body>
```

```
btns doc
▼ HTMLCollection
  ▶ 0: <button id="btnInDiv">
  ▶ 1: <button id="envoi"
  onclick="verifFormulaire()">
  ▶ btnInDiv: <button id="btnInDiv">
  ▶ envoi: <button id="envoi"
  onclick="verifFormulaire()">
    length: 2
  ▶ <prototype>: HTMLCollectionPrototype {
  item: item(), namedItem: namedItem(), length:
  Getter, … }
btns div
▼ HTMLCollection
  ▶ 0: <button id="btnInDiv">
  ▶ btnInDiv: <button id="btnInDiv">
    length: 1
  ▶ <prototype>: HTMLCollectionPrototype {
  item: item(), namedItem: namedItem(), length:
  Getter, … }
```

```
let unDiv = document.getElementById( elementId: "undiv");
console.log("btns doc", document.getElementsByTagName( qualifiedName: "button"));
console.log("btns div", unDiv.getElementsByTagName( qualifiedName: "button"));
```

Navigation dans le DOM

- Alternative aux `getElement...` :
  `querySelectorAll, querySelector`

```
document.querySelector( selectors: "#liste")
document.querySelectorAll( selectors: ".as")
document.querySelectorAll( selectors: "LI")
document.querySelectorAll( selectors: "LI.as")
;
```

- `All` : liste de tous ceux qui correspondent, sinon, le premier trouvé dans l'arbre de l'élément dans lequel on cherche

- Différence vis-à-vis des `getElement…` :
  une fonction pour en remplacer quatre
  → algorithme moins efficace mais plus riche

Navigation dans le DOM

- Autre différence vis-à-vis des `getElement` : renvoie une liste *statique* alors que `getElement`… renvoient une liste *live*

```
let eltUL = document.getElementById( elementId: "myUL");
let listeLIqsa = eltUL.querySelectorAll( selectors: "LI");
let listeLIgebtn = eltUL.getElementsByTagName( qualifiedName: "LI");

console.log("lg qsa", listeLIqsa.length);
console.log("lg gebtn", listeLIgebtn.length);

const newLI = document.createElement( tagName: "li");
newLI.innerText = "Francois Matthieu";
eltUL.appendChild(newLI);

console.log("lg qsa", listeLIqsa.length);
console.log("lg gebtn", listeLIgebtn.length);
```

```
<ul id="myUL">
    <li>Alonso Fernando</li>
    <li>Andre Loic</li>
    <li>Diani Dorian</li>
    <li class="as">Duda Kévin</li>
    <li>Dumoulin Margot</li>
</ul>
```

```
lg qsa 5                tests.js:22
lg gebtn 5              tests.js:23
lg qsa 5                tests.js:29
lg gebtn 6              tests.js:30
```

Modification du DOM

- `innerHTML` : accès au contenu d'une balise HTML, en lecture ou écriture
- Exemple `h2`

```
console.log("Titre de la liste : ", document.getElementById( elementId: "titre_liste").innerHTML);
document.getElementById( elementId: "titre_liste").innerHTML = "Liste (incomplète)";
```

```
Titre de la liste :  Liste des étudiants
Titre de la liste :  Liste (incomplète)
```

- Ou tout le contenu d'un `ul`

```
document.getElementById( elementId: "liste").innerHTML = "<li>Un seul étudiant</li>"
```

Liste (incomplète)

- Alonso Fernando
- Andre Loïc
- Diani Dorian
- Duda Kévin
- Dumoulin Margot

Liste (incomplète)

- Un seul étudiant

Modification du DOM

- createElement
- appendChild, removeChild, replaceChild
- insertBefore

```
let li2 = document.createElement( tagName: "li");
li2.innerHTML = "Un 2eme étudiant";
document.getElementById( elementId: "liste").appendChild(li2);

let li3 = document.createElement( tagName: "li");
li3.innerHTML = "Encore un étudiant";
let li4 = document.createElement( tagName: "li");
li4.innerHTML = "Etudiant 0";

document.getElementById( elementId: "liste").insertBefore(li3,
    document.getElementById( elementId: "liste").children[0]);
document.getElementById( elementId: "liste").replaceChild(li4, li3);
document.getElementById( elementId: "liste").removeChild(li4);
```

Liste (incomplète)

- Un seul étudiant
- Un 2eme étudiant

Liste (incomplète)

- Encore un étudiant
- Un seul étudiant
- Un 2eme étudiant

Liste (incomplète)

- Etudiant 0
- Un seul étudiant
- Un 2eme étudiant

Liste (incomplète)

- Un seul étudiant
- Un 2eme étudiant

Modification du DOM

- Gestion du css
- Via les classes

```
btn.classList.add("btn1");
btn.classList.add("btn_gras", "btn_bleu");
btn.classList.remove( tokens: "btn1", "btn_gras");
```

- Via les styles

```
btn.style.backgroundColor = "blue";
btn.style.display = "none";
```

Modification du DOM… et efficacité

- Solution 1 : 11 secondes

```
let eltUL = document.getElementById( elementId: "myUL");
eltUL.innerHTML="";

for (let i=0; i < 3000; i++) {
    const newLI = "<li>Francois Matthieu</li>";
    eltUL.innerHTML += newLI;
}
```

→ Le *rendering* du DOM est coûteux !

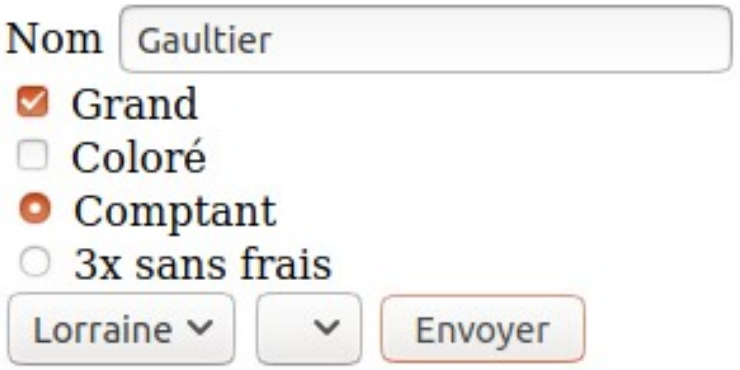
- Solution 2 : résultat immédiat

```
let eltUL = document.getElementById( elementId: "myUL");
let eltULinnerHTML="";

for (let i=0; i < 3000; i++) {
    const newLI = "<li>Francois Matthieu</li>";
    eltULinnerHTML += newLI;
}

eltUL.innerHTML=eltULinnerHTML;
```

Les formulaires

- Valeurs des champs de saisie
- `value, checked, selectedIndex`

```
console.log("text value", document.getElementById( elementId: "nom").value);
console.log("checkbox value", document.getElementById( elementId: "grand").value);
console.log("checkbox checked", document.getElementById( elementId: "grand").checked);
console.log("radio value", document.getElementById( elementId: "comptant").value);
console.log("radio checked", document.getElementById( elementId: "comptant").checked);
console.log("select value", document.getElementById( elementId: "regions").value);
console.log("select selectedIndex", document.getElementById( elementId: "regions").selectedIndex);
```




```
text value Gaultier
checkbox value on
checkbox checked true
radio value Comptant
radio checked true
select value 2
select selectedIndex 1
```

Les formulaires

- `<select>` multiple : `select.options[i].selected`
- `<textarea>` : utilisation de `value`

- Tous ces attributs sont accessibles en écriture

```
document.getElementById( elementId: "nom").value = "JP";
document.getElementById( elementId: "grand").checked = false;
document.getElementById( elementId: "colore").checked = true;
document.getElementById( elementId: "tfsf").checked = true;
document.getElementById( elementId: "regions").selectedIndex = 3;
```

Navigation DOM et formulaires

- L'accès à un élément d'un formulaire est possible comme ceci :

```
document.forms[0].elements[2].value
```

- (valeur du troisième champ du 1er formulaire)

- Tout s'écroule si on change la structure du document html (insertion / suppresion d'un formulaire, d'un champ, etc.)

- → à éviter, utiliser plutôt les *id*

Les événements

- `onclick` : clic sur un élément
- `onsubmit` : soumission d'un formulaire
- `onchange` : modification (nécessite la perte de focus sur certains éléments)
- `onkeyup, onkeydown, onkeypressed`
- `onload` : au chargement de la page

```
<div>
    <label for="long_texte">Conclusion : </label>
    <textarea id="long_texte" rows="3" cols="5" onchange="modif(this)"></textarea>
</div>
<button id="envoi" onclick="verifFormulaire()">Envoyer</button>
```

Les événements

- Généralement on préfère gérer les événements en JS, pour garder un code HTML plus propre

- → pas de onclick, etc. en HTML

```html
<div>
    <label for="long_texte">Conclusion : </label>
    <textarea id="long_texte" rows="3" cols="5"></textarea>
    <button id="envoi">Envoyer</button>
</div>
```

- → mais gestion en JS

```js
let btnEnvoi = document.getElementById( elementId: "envoi");
btnEnvoi.addEventListener( type: "click", listener: () => verifFormulaire());

let lgTxt = document.getElementById( elementId: "long_texte");
lgTxt.addEventListener( type: "change", listener: () => modif(lgTxt));
```

Les événements

- Le second paramètre est une fonction...
- ...Pas l'exécution d'une fonction

```
const btnReset = document.getElementById('btn_reset');

// OK !!!
btnReset.addEventListener('click', recommence);
btnReset.addEventListener('click', () => recommence());

// pas OK !!!
btnReset.addEventListener('click', recommence());
```