TP 4 – DOM, interactions et avancés ES6

1. Rappels des notions

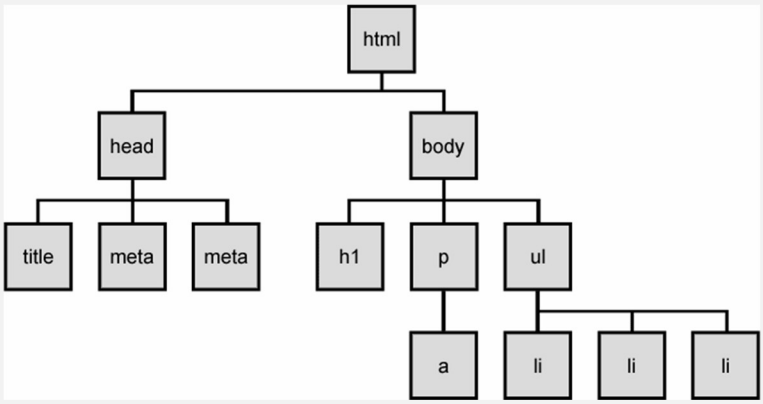
1. Document Object Model (DOM)

Le DOM est une API d'interaction avec des documents HTML (ou XML). Il comporte une représentation d'une page HTML sous forme d'arbre et offre de nombreux moyens d'interaction pour la parcourir, modifier, créer ou supprimer des éléments à l'intérieur.
Les navigateurs commencent toujours par convertir un fichier HTML en DOM puis font un affichage du DOM en utilisant les projets des nœuds de l'arbre et des styles associés. Certaines librairies javascript (comme jQuery) permettent de modifier simplement le DOM alors que d'autres utilisent un DOM virtuel (comme react) et ne font que les mises à jour nécessaires du DOM.

Le code HTML suivant donne la représentation sous forme d'arbre ci-dessous :

```
<html>
<head>
  <title>Page test</title>
  <meta charset="UTF-8">
  <meta name="description" content="DOM">
</head>
<body>
  <h1>Titre</h1>
  <p id="titre">voici un <a href="lien.html">lien</a></p>
  <ul>
    <li>element 1</li>
    <li>element 2</li>
    <li>element 3</li>
</ul>
</body>
</html>
```

Il y a plusieurs représentations selon que l'on indique le texte ou pas, les attributs, etc. Ici on a une représentation des objets « Element » uniquement, donc le texte n'apparait pas.



Les objets classiques pour interagir avec une page sont :

- Node : un Node est un nœud quelconque du DOM et peut correspondre à une balise, à un commentaire, à du texte dans une balise, etc.
- Element : un Element correspond à une balise. Dans la plupart des cas on ne fera pas vraiment la différence avec un Node car Element hérite de Node. Ces objets offrent des possibilités de modification du nœud ou des parcours vers d'autres nœuds de l'arbre.
- Document : c'est un Node particulier qui est le point d'entrée dans le contenu de la page. Permet de récupérer des informations sur le document (encodage de caractères, URL, feuilles de style et scripts utilisées) et offre des fonctions pour rechercher des balises dans la page. Une variable document est accessible à tout moment.

2. Parcours du DOM

LIVE signifie que la variable retournée sera modifiée automatiquement si de nouvelles balises sont créées dynamiquement.

```
// récupérer UN élément par son identifiant

// récupérer DES éléments par leur classe, idem pour document (LIVE)
var elements = element_parent.getElementsByClassName(name);

// récupérer DES éléments par leur nom de balise, idem pour document (LIVE)
var elements = element_parent.getElementsByTagName(name);

// récupérer DES éléments avec des sélecteurs CSS
document.querySelectorAll("p.x") // tous les paragraphes de classe x

// récupérer le contenu HTML (sérialisé) d'une balise
var contenu = element.innerHTML;

// idem en incluant la balise elle-même
var contenu_et_balise = element.outerHTML;

// récupérer le premier / dernier fils d'un Node / Element
var premier = node.firstChild; / element.firstElementChild;
var dernier = node.lastChild / element.lastElementChild;

// récupérer toutes les éléments fils d'un (LIVE)
var fils = node.childNodes;  / element.children;

// récupérer le frère suivant d'un Node / Element
var next = node.nextSibling / element.nextElementSibling;

// récupérer le parent d'un Node / Element
var parent = node.parentNode / element.parentElement;
```

Dans le DOM de l'exercice 1, vous devez savoir récupérer

- Le contenu HTML de la balise <p id="titre"> ?
- la deuxième balise <li> ?
- la balise <a> sachant qu'une variable balise_p contient la balise p ?
- la balise <p> sachant qu'une variable balise_h1 contient la balise h1 ?
- le contenu HTML de la deuxième balise <li> de 2 manières différentes sachant qu'une variable balise_ul contient la balise ul ?

- la dernière balise <li> sachant qu'une variable balise_a contient la balise a ?
- toutes les balises p contenues dans une balise d'identifiant x donné

Exercice 1. Soit la balise <h id="h">test</h>. Que retourne children et childNodes sur cet Element ?

3. Modification du DOM

```
// créer un Element correspondant à une balise p
var balise = document.createElement("p");

// créer un Node contenant du texte
var contenu = document.createTextNode("test");

// ajouter une balise fille (à la fin)
balise_pere.appendChild(balise_fille);
document.body.appendChild(balise);

// ajouter une nouvelle balise fille d'une balise donnée avant une autre balise fille reference (insère à la fin si balise_reference==null)
balise_pere.insertBefore(balise_fille, balise_reference);

// supprimer une balise
var balise_supprimee = balise_pere.removeChild(balise_fille);
```

4. Modification de balises

```
// Modifier le contenu HTML d'une balise
element.innerHTML="<p>test</p>";

// Modifier le style d'une balise
element.style.style_css = valeur_css

// classes d'une balise
element.classList; // retourne un objet DOMTokenList contenant les classes
element.classList.remove("class1");
element.classList.add("class2");

// attributs HTML d'une balise
var valeur = element.hasAttribute(attribut);
var valeur = element.getAttribute(attribut);
element.setAttribute(attribut, valeur);

// propriétés d'un Element du DOM
var valeur = element.value; // valeur d'un champ input
var checked = element.checked; // case checkbox ou radio cochée ?
var link = element.href; // destination d'un lien
```

5. Evénements

Liste des événements de base :

- Souris : click, mousover, mouseout, mousemove
- Clavier : keydown, keyup
- Formulaires : blur, change, focus, select, submit
- Page : load

```
// Ajouter un événement à un Element (par exemple obtenu avec getElementById)
// Attention, chaque couple (evenement, fonction) doit être unique
Element.addEventListener(evenement, fonction);
Element.onevent = function() {...} ; // remplacer event par le bon événement

// Supprimer un événement créé précédemment via le nom de la fonction
// N'est pas utilisable si on a créé l'événement avec une fonction anonyme
Element.removeEventListener(evenement, fonction);
```

Tous ces événements peuvent être utilisés directement en HTML. Attention la syntaxe est légèrement différente selon la manière dont l'événement est défini.

```
// directement en HTML
<p onclick="...">paragraphe cliquable</p>

// via la propriété onclick de l'Element
obj.onclick = function() {...};

// en ajoutant un Listener sur l'Element
function funx(event) {...}
obj.addEventListener("click", funx);
```

Quand une fonction est appelée en réponse à un événement, elle reçoit un objet Event qui contient entre autres :

```
obj.addEventListener("click", function(event) {console.log(event.xxx);});
// le Node cible de l'événement : event.target
// la position de l'événement : event.x, event.y
// le chemin de la racine du DOM : event.path
// est-ce que la touche Shift est enfoncée : event.shiftKey
// ...
```

6. Minutage des événements

```
// exécute la fonction toutes les duree millisecondes
let intervalle = window.setInterval(fonction, duree);

// stoppe l'appel de la fonction
window.clearInterval(intervalle);

// Exécute la fonction une fois après duree millisecondes
let timer = window.setTimeout(fonction, duree);

// Abandonne le timer
window.clearTimeout(timer);
```

7. les « templates string »

Vous avez dû remarquer que sur la norme « airbnb », vous ne pouviez pas concaténer une chaine avec le contenu d'une variable ou la valeur de retour d'une fonction. Vous devez utiliser le principe des *template strings* ou aussi *template literals*. Le côté *template* est justement pensé pour constituer des *modèles* de blocs de texte réutilisables.

```
const ami = 'georges';

//  bad
console.log('mon ami s\'appelle ' + ami);

//  good
console.log(`mon ami s'appelle ${ami}`);
```

Les templates string seront utiliser dans du code JavaScript *vanilla* et avec des outils tels que jQuery, Angular, React.

8. Objet littéral

```
const user = {
  name: 'pierre',
  mail: 'pierre@gmail.com',
  age: 32,
  friends: ['bob', 'mireille'],
};
```

User est un objet littéral contenant ici 4 propriétés. Chaque propriété peut contenir un type de variable différent number, boolean, object

```
//  modification d'une propriété
user.name = 'georges';

//  affichage
console.log(user.name);
```

Un objet littéral peut contenir une ou plusieurs méthode(s)

```
const user = {
  name: 'pierre',
  mail: 'pierre@gmail.com',
  age: 32,
  friends: ['bob', 'mireille'],
  printFriends() {
    this.friends.forEach((x) => (
      console.log(`${this.name} a comme ami ${x}`)
    ));
  },
};
```

La méthode printFriends() permet d'afficher la liste des différents amis.

```
//  affichage
user.printFriends();
```

Que pouvez dire 'this' utilisé ici deux fois

9. Destructuring

Le destructuring consiste à assigner des variables provenant d'un objet ou tableau en reposant sur leur structure.

```
// sur les tableaux
const arr = [1, 2, 3, 4];

const [first, second] = arr;
```

2 variables seront créées first et second avec les valeurs 1 et 2

```
// sur les tableaux
const personne = {
  name: 'dupont',
  age: 34,
}

const {name, age} = personne;
```

Quel est le type de chaque variable ?

```
let a = 10;
let b = 20;

[a, b] = [b, a];
```

A quoi sert cette écriture ?

10. Spread (…)

L'opérateur spread a une écriture identique que les « paramètres rest »
IL permet de décomposer (développer) un objet itérable, comme un tableau.

```
function sum(x, y, z) {
  return x + y + z;
}

const numbers = [1, 2, 3];

console.log(sum(...numbers));
```

```
// sur les tableaux
const arr = [1, 2, 3, 4];

const [first, …tab] = arr;   // tab => [2, 3, 4]
```

```
const articulations = ['épaules', 'genoux'];
const corps = ['têtes', ...articulations, 'bras', 'pieds'];
// ["têtes", "épaules", "genoux", "bras", "pieds"]
```

Il permet également de faire une copie de tableau
Si nous faisons le code suivant :

```
const t1 = [1, 2, 3];
const t2 = t1;

t2.push(4);
console.log(t1);
console.log(t2);
```

Que vaut t1 et t2 ? et pourquoi ?

Pour remédier à ce problème, nous utilisons la syntaxe suivante

```
const t1 = [1, 2, 3];
const t2 = [...t1];

t2.push(4);
console.log(t1);
console.log(t2);
```

11. map(), filter(), reduce()

Ces 3 méthodes de la classe Array sont aujourd'hui très utilisées.
Avez-vous bien compris leur fonctionnement

Reportez-vous aux documentations :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/map
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array/filter
https://medium.com/@hkairi/reduce-le-couteau-suisse-du-d%C3%A9veloppeur-javascript-8cf4b6f98304

2. Exercices

Cette première partie a pour but de vous rappelez des notions INDISPENSABLES pour manipuler le DOM et les évènements. Prenez votre temps pour tester plusieurs méthodes.

Vous trouverez sous moodle des fichiers à compléter permettant d'appliquer les notions de cours ci-dessus

Tous les exercices doivent être validés au format « airbnb-base »