TP 8: Manipuler le Web avec DOM

Objectifs:

- Exécuter du code JavaScript depuis une page HTML
- Manipuler une page HTML depuis un programme JavaScript
- Introduction aux concepts d'API et d'événements
- Développer une calculatrice dans le navigateur web

Référence:

- Référence JavaScript du Mozilla Developer Network (en Français)

Aide-mémoires:

- OverAPI's JavaScript Cheatsheet (liste des propriétés et fonctions JavaScript/DOM, classées par type)
- JavaScript Cheat Sheet by DaveChild (liste des principales fonctions et propriétés JavaScript/DOM)

---

1. JavaScript et le DOM

Dans les chapitres précédents, nous avons exécuté des programmes JavaScript simples de manière interactive, depuis la console de notre navigateur.

Dans ce chapitre, nous allons voir:

- comment associer un programme JavaScript à une page web,
- et comment modifier cette page dynamiquement depuis notre programme.

Terminologie: quelques rappels sur le Web

- WWW: *World Wide Web*, c'est le nom donné à l'ensemble des pages liées entre elles sur Internet via le protocole HTTP (nous en reparlerons dans le chapitre suivant). La plupart de ces pages sont décrites en langage HTML.
- HTML: *HyperText Markup Language*, c'est un langage qui permet de décrire la structure et le contenu d'une page web, en utilisant des éléments (balises).

- CSS: *Cascading Style Sheets*, c'est un langage qui permet de mettre en page et styliser les éléments HTML d'une page web en leur appliquant des propriétés.
- JavaScript: c'est le seul langage qui permet de donner des instructions exécutables depuis une page web au format HTML.
- Navigateur Web / *Web Browser*: c'est un logiciel d'affichage de pages web (généralement composées de fichiers HTML, CSS et JavaScript), et permettant à l'utilisateur d'interagir avec celles-ci.
- DOM: *Document Object Model*, c'est à la fois le nom qu'on donne à l'ensemble des éléments qui constituent une page Web ouverte dans le navigateur (telle qu'elle est structurée en mémoire), et à l'API qui permet de manipuler ces éléments.
- API: *Application Programming Interface*, est un ensemble de fonctions fournies par un logiciel (par exemple: un navigateur Web, ou un serveur Web) qui permettent à d'autres programmes d'interagir / échanger des informations avec lui.

Associer un programme JavaScript à une page web

Vous devez savoir qu'une page web peut être associée à une feuille de style CSS. Pour cela, le code source HTML de cette page doit contenir un élément `<link>` donnant l'URL du fichier CSS correspondant.
De la même façon, pour associer un programme JavaScript à une page web, il suffit d'ajouter un élément `<script>` donnant l'URL du programme en question (dont le fichier porte généralement l'extension `.js`).
Exemple de code source HTML d'une page web:

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Bonjour !</h1>
    <script src="script.js"></script>
  </body>
</html>
```

Trois choses importantes à remarquer:

- l'élément `<script>` peut être défini dans le `<head>` ou dans le `<body>`, mais il est généralement recommandé de l'insérer juste avant la fin du `<body>`;
- l'URL du script doit être fournie via l'attribut `src`;
- mais surtout, contrairement aux éléments `<link>`, les éléments `<script>` ne doivent pas être exprimés sous forme d'une balise auto-fermante (finissant par `/>`) => il est impératif d'**utiliser une balise fermante `</script>` après chaque balise ouvrante `<script>`**.

Les scripts ainsi intégrés dans le `<body>` de la page seront exécutés au fur et à mesure qu'ils sont découverts et chargés par le navigateur.

Remarque, il est aussi possible d'intégrer directement notre programme JavaScript entre les balises `<script>` et `</script>`, pour éviter de le stocker dans un fichier séparé. Cette méthode n'est pas recommandée car elle peut causer des erreurs de syntaxe.

Application: Dire bonjour au monde

Vous allez créer une page web qui affiche `Bonjour le monde !` dans une fenêtre à l'aide de la fonction `alert()`.
Pour cela:

1. Créez un fichier `bonjour.html` définissant une page web minimale mais valide, contenant seulement un titre de type `<h1>`.
2. Ouvrez cette page dans votre navigateur, pour vérifier qu'elle s'affiche correctement.
3. Créez un fichier `bonjour.js` contenant un programme JavaScript permettant d'afficher le message demandé.
4. Testez votre programme JavaScript en le copiant dans la console du navigateur, depuis n'importe quelle page.
5. Associez le fichier `bonjour.js` à la page `bonjour.html`, à l'aide d'un élément `<script>`.
6. Rafraîchissez la page `bonjour.html` dans votre navigateur, pour vérifier que le message demandé s'affiche bien dès l'affichage de la page.
7. En cas de problème, vérifiez la présence d'erreurs dans la console JavaScript du navigateur, corrigez-les, et ré-essayez.

Accéder aux éléments de la page Web depuis JavaScript

Les navigateurs Web (tels que Google Chrome) donnent accès à une *API* (voir définition plus haut) permettant à nos programmes JavaScript d'interagir avec le *DOM* de la page Web à laquelle ils sont liés.

C'est à dire qu'un script intégré dans une page peut utiliser des fonctions permettant de manipuler le contenu de cette page. Par exemple: pour récupérer des informations saisies par l'utilisateur dans des champs de la page, ou encore modifier le contenu et/ou l'affichage de la page.

Pour accéder à un élément de la page, il faut identifier précisément (c.a.d. sans ambiguïté) cet élément auprès du navigateur. Par exemple: en l'adressant par son identifiant unique (attribut `id` de l'élément).

Pour cela, l'API du DOM met à notre disposition un *objet* (cf chapitre précédent) appelé `document`, et cet objet contient plusieurs fonctions. Nous allons d'abord nous intéresser à la fonction `getElementById()` qui permet d'accéder à l'objet représentant un élément de la page, en fonction de son identifiant unique.
Supposons que notre page Web contienne les éléments suivants:

```
<body>
  <p id="premier-paragraphe">Bonjour</p>
  <p id="deuxieme-paragraphe">le monde</p>
</body>
```

Nous pouvons alors accéder au premier paragraphe en JavaScript (ex: depuis un script rattaché à cette page, ou depuis la console du navigateur) de la manière suivante:

```
document.getElementById('premier-paragraphe');
// => retourne un objet qui représente l'élément HTML correspondant
```

En exécutant cet appel de fonction dans la console, on voit s'afficher ce qu'elle retourne: un objet JavaScript qui représente l'élément HTML ayant `premier-paragraphe` comme identifiant.
Nous allons voir plus bas que l'objet `document` fourni par l'API du DOM contient d'autre fonctions permettant d'accéder à des éléments, comme `getElementsByClassName()` ou encore `querySelector()`.

Application:

1. allez sur le site web de votre choix,
2. utilisez l'onglet "Éléments" de Chrome Dev Tools (barre latérale dans laquelle se trouve aussi la console JavaScript),
3. repérez un élément qui possède un attribut `id`,
4. dans la console, utilisez `getElementById()` de manière à afficher l'objet JavaScript représentant cet élément.

Astuce: pour accéder plus rapidement à la partie du DOM qui représente un élément d'une page Web, effectuer un clic-droit sur cet élément de la page, puis cliquez sur "Inspecter". Vous vous retrouverez immédiatement dans l'onglet "Éléments" de la page, à l'endroit où est défini cet élément.

Récupérer la valeur d'un champ de saisie `<input>`

Maintenant que nous savons accéder à l'objet JavaScript correspondant à un élément HTML de la page, nous allons voir comment récupérer des données de cet élément.

Pour cela, nous allons utiliser la propriété `value` de l'objet JavaScript représentant un élément HTML.
Imaginons une page HTML contenant le formulaire suivant:

```
<body>
  <form>
```

```
    <label for="nom">Nom:</label>
    <input id="nom" value="Michel" />
    <label for="prenom">Prénom:</label>
    <input id="prenom" value="Jean" />
  </form>
</body>
```

Pour accéder à l'objet JavaScript représentant le champ portant l'identifiant `nom`, nous allons utiliser la fonction `getElementById()` de cette manière:

```
document.getElementById('nom');
```

Enfin, pour récupérer la valeur actuelle de ce champ, il suffit d'utiliser la propriété `value` de cet objet:

```
document.getElementById('nom').value;
```

Dans notre exemple, l'exécution de cette instruction JavaScript retournera la valeur du champ correspondant: `"Michel"`.
Si on exécute à nouveau cette instruction après que l'utilisateur a modifié la valeur du champ, la valeur retournée correspondra à la valeur actuelle du champ (après modification).

Pratique: Récupération des valeurs du formulaire

Afin de mettre en pratique l'association d'un programme JavaScript à une page HTML, l'accès à un élément en JavaScript, et la récupération de la valeur d'un champ de saisie, vous allez:

1. Créer un dossier `JS-FORM-1`;
2. Dans ce dossier, créer un fichier `index.html` valide contenant deux champs de saisie, tel que définis dans l'exemple ci-dessus (avec identifiants `nom` et `prenom`);
3. Toujours dans ce dossier, créer un fichier `index.js` contenant un programme JavaScript qui affichera avec `alert()` la valeur du premier champ de la page, puis celle du deuxième champ;
4. Associez votre programme JavaScript (fichier `index.js`) à votre page HTML (fichier `index.html`), de manière à ce qu'il soit exécuté quand on ouvre la page, et vérifier que les deux valeurs sont bien affichées dans des `alert` lors du chargement de la page dans votre navigateur;
5. En cas de malfonctionnement, corrigez les erreurs dans vos fichiers, en vous aidant de la console JavaScript.

Réagir aux actions de l'utilisateur sur la page

Maintenant que nous savons accéder aux données d'une page HTML depuis un programme JavaScript, et faire en sorte que ce programme s'exécute au chargement de la page, nous allons voir comment exécuter des instructions JavaScript en réponse à une action de l'utilisateur sur la page.

À chaque fois que l'utilisateur interagit avec une page Web, le navigateur déclenche des *événements*. Ces événements sont mis à disposition par l'API du DOM, afin qu'un programme JavaScript puisse les intercepter et réagir à certains d'entre eux.

Quelques exemples d'événements:

- `click`: l'utilisateur a cliqué sur un élément
- `change`: l'utilisateur a changé la valeur d'un champ de saisie
- `mouseover`: l'utilisateur a survolé un élément à l'aide de la souris

Vous trouverez la liste des évènements standard sur cette page: Event reference - MDN.

On peut définir le comportement (la réaction) que doit adopter le navigateur lorsqu'un événement survient, en y associant une fonction JavaScript.

Par exemple, nous pourrions définir une fonction direBonjour() contenant alert('bonjour !'), puis demander au navigateur d'appeler cette fonction à chaque fois que l'utilisateur clique sur un bouton.

Il existe plusieurs moyens d'intercepter des événements en y attachant une fonction:

- la fonction `addEventListener()` (que nous verrons peut-être plus tard)
- et les propriétés `on*` associées à chaque nom d'événement, ex: `onclick`, `onchange`, `onmouseover`...

Pour l'instant, nous allons employer la méthode la plus simple: affecter une fonction à la propriété d'un élément correspondante à l'événement choisi.

Imaginons une page HTML contenant un bouton:

```
<body>
  <button id="mon-bouton">Mon Beau Bouton</button>
</body>
```

Pour afficher un `alert` à chaque fois que l'utilisateur cliquera sur ce bouton (événement `click`), nous devons affecter une fonction à la propriété `onclick` de l'objet JavaScript représentant ce bouton:

```
document.getElementById('mon-bouton').onclick = function direBonjour() {
  alert('bonjour !');
};
```

Exercice: Calculatrice

Afin de pratiquer la récupération de valeurs d'un formulaire et l'exécution de code JavaScript lorsque l'utilisateur clique, nous allons développer une calculatrice simple en HTML + JavaScript.

La page HTML a développer doit contenir:

- deux champs `<input>` portant les valeurs d'`id`: "`premier-nombre`" et "`deuxieme-nombre`";
- un troisième champ portant l'`id`: "`resultat`";
- et un bouton portant l'`id`: "`mon-bouton`".

Ensuite, y associer un fichier JavaScript qui permettra à l'utilisateur, à chaque fois qu'il cliquera sur le bouton `mon-bouton`, d'obtenir le résultat de l'addition des nombres qu'il aura saisis dans les champs `premier-nombre` et `deuxieme-nombre`, dans le champ `resultat`. Vous aurez probablement besoin d'utiliser la fonction `parseInt()` ou `parseFloat()` pour convertir des chaînes de caractères en véritables nombres.

Par ailleurs, pour éviter que votre bouton ne soumette votre formulaire quand vous cliquerez dessus, vous aurez peut-être besoin d'ajouter l'attribut `type="button"` à cet élément. (Explications: ici)

Exercice Bonus: Calculatrice multi-opérations

Sur la base de la calculatrice ci-dessus, ajouter un composant de sélection de l'opération qui sera effectuée entre les deux nombres lorsque l'utilisateur cliquera sur le bouton: addition, soustraction, multiplication et division.