HE2B ESI
Haute École Bruxelles-Brabant
Campus de l'École Supérieure d'Informatique
Bachelier en Informatique

2024-2025

2WEB1A – Laboratoires de développement WEB 1

TD 5 – Boutons et introduction à JavaScript

Résumé

Nous allons ajouter des boutons à nos pages, définir leur style et programmer ce qui doit se passer lorsqu'on clique dessus.

Sommaire

Ce document est distribué sous licence
Creative Commons Paternité - Partage à l'Identique 2.0 Belgique (http://creativecommons.org/licenses/by-sa/2.0/be/).
Les autorisations au-delà du champ de cette licence peuvent être demandées à esi-2web1a-list@he2b.be.

Version du document
13 février 2025

1 Des boutons qui ont du style

Ajouter un bouton à une page est très facile ; il suffit d'utiliser la balise `<button>`.

```
<button>OK</button>
```

OK

Nous connaissons déjà quelques propriétés utiles pour modifier le style du bouton. Par exemple :

```
background-color: teal;
color: white;
border: 1px solid black;
```

OK

Examinons d'autres propriétés utiles pour les boutons.

1.1 Arrondir les angles

Pour le moment, nos cadres ont des angles droits ; on peut obtenir des angles arrondis via la propriété `border-radius`.

```
border-radius: 8px;
```

Dans les cas les plus simples, il s'agit du rayon du cercle. Autrement dit, la distance entre le coin et le moment où la courbure commence. Cf. https://developer.mozilla.org/en-US/docs/Web/CSS/border-radius pour les détails.

1.2 Une part d'ombre

Il est également possible d'ajouter une ombre pour donner l'illusion que l'élément (un bouton dans notre cas) est surélevé par rapport à la page.

```
box-shadow: 2px 2px 5px gray;
```

OK

Il n'est pas facile de se rappeler le sens de chaque valeur pour cette propriété et surtout il est difficile d'imaginer le résultat. Il existe des outils pour nous aider dans cette tâche, notamment : https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_backgrounds_and_borders/Box-shadow_generator.

Exercice 1 **Se familiariser avec les ombres**

Utilisez l'outil proposé pour trouver comment obtenir une ombre proche de celle-ci :

OK

1.3 Il faudra me passer dessus

Si vous prenez le style par défaut pour un bouton, vous pouvez constater que ce style change légèrement quand la souris passe sur le bouton. Il y a également un effet visuel suite au clic de la souris. Comment configurer ces effets ? Via des *pseudo-classes*.

```
button:hover{  // Lorsque la souris est sur le bouton
   box-shadow: 4px 4px gray;      // Propriété qui s'ajoute aux propriétés de base
}
button:active{  // Lorsque le bouton est cliqué
   box-shadow: 0px;
}
```

Exercice 2 **Se familiariser avec la dynamique du bouton**

Adaptez votre style de sorte qu'un bouton ressemble aux captures ci-dessous où on montre un bouton de base, lorsque la souris passe dessus et lorsqu'il est cliqué.

OK OK OK

Une feuille de style pour les boutons

Il est utile de pouvoir récupérer des éléments de style de projet en projet. Nous ne pouvons que vous encourager à créer un fichier (par exemple `button.css`) qui contient le CSS lié aux boutons. Vous pourrez l'enrichir (par exemple en créant des classes pour des boutons d'alerte, des boutons avec icônes…). Il vous sera ainsi facile de le réutiliser pour avoir des boutons stylés dans tous vos TDs, dans le projet et dans tous les cours liés aux web (`3web2a`, `4web3d`).

2 Des boutons qui ont un effet

Pour le moment, cliquer sur un bouton n'a aucun effet. Et c'est normal, nous n'avons pas dit quoi faire.

Tutoriel 1 Un bouton qui dit bonjour

Nous allons définir un bouton qui affiche un petit texte lorsqu'on clique dessus.

- Partez d'un nouveau projet (un dossier vide donc) ouvert dans Code.
- Créez un fichier `index.html` de base.
- Ajoutez le code pour définir un bouton.

index.html
```
<button id="goBtn">GO</button>
```

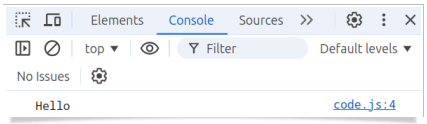
- Créez un fichier `code.js` avec le contenu suivant :

code.js
```
"use strict";

function go() {
    console.log("Hello");
}

document.getElementById("goBtn").addEventListener("click", go);
```

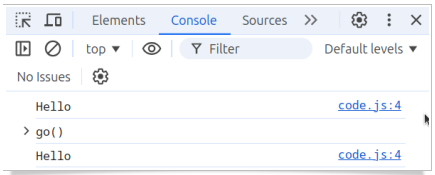
- Ajoutez dans le fichier HTML (dans la partie `<head>`) la balise qui incorpore le JavaScript (raccourci clavier : `script:src`).

index.html
```
<script src="code.js" defer></script>
```

- Lancez la page et cliquez sur le bouton. Vous ne voyez rien s'afficher ? C'est normal.
- Ouvrez l'outil du développeur et sélectionnez l'outil `Console` ; vous verrez le message.

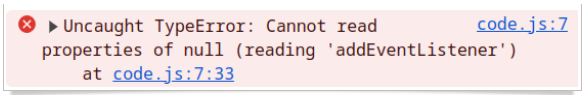


- La console permet également de taper interactivement des commandes. Vous pouvez par exemple appeler la fonction `go` ce qui est pratique pour les tests.



C'est aussi dans la console que vous verrez les messages d'erreurs liés au JavaScript.

- Par curiosité, enlevez le `defer` de la balise `<script>` et relancez.



Comprenez-vous pourquoi vous avez cette erreur (vous pouvez chercher sur MDN l'aide de la balise `<script>`) ?

Tutoriel 2 **Un bouton qui dit bonjour dans la page**

Modifions l'exemple afin que le message apparaisse sur la page.

- Partez du projet obtenu dans le tutoriel précédent.
- Ajoutez le code pour définir un bouton et prévoir un champ pour la réponse.

index.html
```
<button id="goBtn">GO</button>
<p>Je vous dit : <span id="answer">...</span> !</p>
```

- Dans le code JavaScript, la fonction `go` devient

code.js
```
function go() {
    const answerTag = document.getElementById("answer");
    answerTag.textContent = "hello";
}
```

- Testez ! Cette fois-ci, appuyer sur le bouton fait apparaitre le mot `Hello` sur la page.

3 Introduction à JavaScript

Attardons-nous sur les bases du langage JavaScript.

Les bases de JavaScript

Voici une très brève introduction à JavaScript :

- La bonne pratique demande à commencer le fichier par *"use strict"*;
- Une **constante** se déclare ainsi (on ne spécifie pas le type) : `const nom_de_variable = value;`
- Idem pour une **variable** : `let nom_de_variable = initial_value;`
- Pour une **alternative**, la syntaxe est la même que celle de Java.

```
if (someTest) {
    // code when test is true
} else {
    // code when test is false
}
```

- On peut répéter des instructions grâce au `while` et au `for`.

```
while (a_boolean_expression) {
    // code to repeat
}
```

```
for (let i = 0; i < 10; i++) {
    // code to repeat
}
```

- Pour définir une **fonction**.

```
function function_name(argument_1, [...], argument_n) {
    // Function code
    return return_value;
}
```

- Utiliser les **objets** via la ***notation pointée***
  - ▷ `unObjet.property` pour accéder à une propriété d'un objet.
  - ▷ `unObjet.method()` pour appeler une fonction d'un objet. (on parle de **méthode**)
  - ▷ Exemple : `document.getElementById("unId").value`
    - `document` est un objet (qui représente la page)
    - `getElementById(...)` est une méthode de cet objet (qui recherche dans le document un élément d'id donné). Cet élément trouvé est lui même un objet.
    - `value` est une propriété de ce second objet (qui contient la valeur entrée dans le champ de saisie [a])
- La notation `[1,2,3]` représente un **tableau** contenant les valeur 1, 2 et 3.
  - ▷ `arr[i]` donne l'élément en position i (à partir de 0) dans le tableau `arr`.
  - ▷ `arr.length` donne le nombre d'élément dans le tableau `arr`.
  - ▷ `arr.push(val)` ajoute la valeur `val` à la fin du tableau `arr`.
  - ▷ `arr.pop()` enlève et retourne la **dernière** valeur du tableau `arr`.
  - ▷ `arr.shift()` enlève et retourne la **première** valeur du tableau `arr`.
  - ▷ `arr.unshift(val)` ajoute la valeur `val` au début du tableau `arr`.
- Un tableau peut se parcourir avec une variante du `for`, le ***for-of***.

```
for (const elem of array) {
    // elem est un élément du tableau, à chaque fois différent
}
```

a. On suppose ici que c'est bien à un champ de saisie, `<input>`, que correspond l'identifiant `unId` donné.

Organisation de votre code

Pour les exercices qui suivent.

- Vous pouvez tout coder dans un seul fichier `.js`.
- Vous pouvez également avoir un seul fichier `.html`.
- Dans la page, séparez clairement chaque exercice par un titre qui reprend le numéro de l'exercice.
- Pour la lisibilité, incorporez le numéro de l'exercice dans les noms choisis pour les `id`.
  Exemple : `<button id="ex1Btn">`

Exercice 3 Addition de 2 nombres

Reprenez le tutoriel du TD2 sur l'addition de deux nombres et vérifiez si vous comprenez bien tout le code.

Exercice 4 Parité d'un nombre

La page offre un champ de saisie numérique. Lorsqu'on clique sur un bouton, un message indique si le nombre est pair ou impair.

Exercice 5 Compter

La page offre un champ de saisie numérique. Lorsqu'on clique sur un bouton, un message apparait avec les nombres de 1 à n (où n est le nombre saisi). Si le nombre saisi est inférieur à 1, on affichera plutôt un message d'erreur.

Exercice 6 Mélanger

La page offre un champ de saisie numérique. Lorsqu'on clique sur un bouton, un code JavaScript va :

1. Créer un tableau avec les nombres de 1 à n (où n est le nombre saisi).
2. Mélanger ce tableau.
3. Afficher à l'écran les nombres mélangés en utilisant le *for-of*.

Shuffle

Voici une fonction JavaScript qui mélange le tableau passé en paramètre.

```
// Shuffles array in place. ES6 version
// @param {Array} a - An array containing the items.
function shuffle(a) {
    for (let i = a.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [a[i], a[j]] = [a[j], a[i]]; // Échange les éléments en position i et j
    }
}
```

4 Modifier la classe

La code JavaScript peut être utilisé pour modifier les classes associées à un élément.

classList

L'objet `elem.classList` offre des méthodes pour manipuler les classes associées à `elem`.

- `elem.classList.add("aClass")` ajoute la classe donnée à `elem`.
- `elem.classList.remove("aClass")` enlève la classe donnée de `elem`.
- `elem.classList.contains("aClass")` retourne `true` si la classe donnée est présente dans `elem`.
- `elem.classList.toggle("aClass")` bascule la classe donnée (l'ajoute si elle n'est pas présente ; l'enlève si elle l'est).

Exercice 7 Barrer un élément de liste

Codez une page qui présente un paragraphe de texte. Lorsqu'on clique dessus, le texte est barré. Lorsqu'on re-clique dessus, le texte n'est plus barré.

5 Un bouton inactif

Vous avez déjà très certainement rencontré des boutons inactifs dans les pages web que vous visitez. Comment faire pour activer/désactiver un bouton et préciser le style d'un bouton inactif ?

HTML. Un `<button>` possède un attribut `disabled` pour le rendre inactif.

```
<button disabled>OK</button>
```

Attribut booléen

Un attribut comme `disabled` est un **attribut booléen**. Il n'est pas de la forme `nom=valeur`. Sa simple présence fait qu'il est vrai ; son absence le rend faux.

CSS. On peut préciser qu'on spécifie le style d'un bouton inactif via la *pseudo classe* `:disabled`

```
button {...}    // s'applique à tous les boutons (inactifs ou pas)
button:disabled {...} // s'applique à tous les boutons inactifs
```

JavaScript. Il est facile de modifier dynamiquement la propriété `disabled` via du code JavaScript.

```
unBouton.disabled = true; // ou false
```

Exercice 8 **On/Off**

Ajoutez deux boutons à une page. Le bouton `On` est actif et le bouton `Off` ne l'est pas. Lorsqu'on clique sur le bouton actif, il devient inactif et le bouton inactif devient actif.

Avec les notions de JavaScript vues jusqu'à présent, vous avez probablement du écrire **deux** fonctions (au code similaire) pour résoudre cet exercice. Nous verrons dans un prochain TD comment faire pour n'avoir qu'une seule méthode mais vous pouvez déjà chercher à le faire s'il vous reste du temps.

Compétences à l'issue de ce TD

Quelles sont les compétences qu'on attend de vous à l'issue de ce TD ?

- Être capable d'ajouter des boutons à vos pages et d'en définir le style : couleur, angles, ombre.
- Pourvoir définir un style différent quand la souris passe sur le bouton ou que le bouton est cliqué.
- Ajouter un *écouteur* à un bouton de telle sorte qu'une fonction JavaScript soit exécutée quand on clique dessus.
- Écrire un code JavaScript dans un fichier dédié et l'inclure correctement dans la page (`defer`).
- Connaitre les bases du langage JavaScript : définir une constante, une variable, écrire des alternative, des boucles, des fonctions et manipuler des tableaux.
- Écrire un code JavaScript qui ajoute dynamiquement du texte à dans un élément de la page.
- Écrire un code JavaScript qui modifie les classes associées à un élément de la page.
- Comprendre la notion de bouton *inactif* et pouvoir rendre un bouton actif/inactif en JavaScript.