Chapitre 1 : Interaction avec l'utilisateur dans une page Web grâce à JavaScript

I. Introduction

A. Qu'est-ce que JavaScript ?

JavaScript a été créé en 1995 par Brendan Eich.

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js.
On abrège souvent JavaScript par JS, ce qui sera parfois fait par la suite.
Avec les technologies HTML et CSS, JavaScript est parfois considéré comme l'une des technologies cœur du World Wide Web. Le langage JavaScript permet des pages web interactives, et à ce titre est une partie essentielle des applications web. Une grande majorité des sites web l'utilisent, et la majorité des navigateurs web disposent d'un moteur JavaScript dédié pour l'interpréter, indépendamment des considérations de sécurité qui peuvent se poser le cas échéant.
Du code JavaScript peut être intégré directement au sein des pages web, pour y être exécuté sur le poste client. C'est alors le navigateur web qui prend en charge l'exécution de ces programmes appelés scripts.

Dans une page Web (programme de 1ère NSI)

Généralement, JavaScript sert à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML via le DOM (Document Object Model). Il est aussi utilisé pour réaliser des applications dynamiques, des transitions, des animations ou manipuler des données réactives, à des fins ergonomiques ou cosmétiques.

Sur un serveur Web

JavaScript peut également être utilisé comme langage de programmation sur un serveur HTTP à l'image des langages comme PHP, Python, ASP, etc. La plateforme Node.js est actuellement très utilisée pour développer et déployer des applications Web entièrement basée sur JavaScript, que ce soit côté client ou côté serveur.

B. Premiers pas avec JavaScript

Comme indiqué précédemment, le JavaScript est un langage essentiellement utilisé avec le HTML, vous allez donc apprendre dans ce chapitre comment intégrer ce langage à vos pages Web pour les rendre dynamiques. Il y a deux façons d'écrire le code JS d'une page Web :

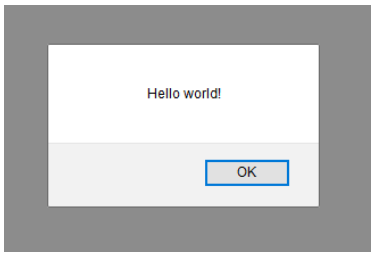
- soit directement dans le HTML (obsolète, mais encore présente dans beaucoup de pages Web anciennes)
- soit dans un fichier séparé au format .js (pratique recommandée notamment car le code est plus facilement maintenable et que l'on peut alors l'appliquer à plusieurs documents HTML)

On détaille ci-dessous les deux méthodes pour afficher le traditionnel texte « Hello World ! » (= « Bonjour le monde ! » en français) à l'utilisateur.

1ère méthode : écrire le JavaScript directement dans le code HTML (à éviter)

Pour écrire du JavaScript dans le code HTML, il suffit de l'écrire entre les balises `<script>` et `</script>`. Le code ci-dessous permet d'afficher le message « Hello World » à l'utilisateur grâce à la fonction `alert`.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre onglet</title>
  </head>
  <body>
    <script>
      alert("Hello World !");
    </script>
  </body>
</html>
```



Remarque : chaque instruction JavaScript doit se terminer par un point-virgule.

2ème méthode (recommandée) : séparer le JavaScript du HTML

Il est préférable d'externaliser le code JS du code HTML, on devra donc utiliser deux fichiers séparés. Pour afficher le message précédent de cette façon, il est nécessaire de :

- Créer un fichier d'extension « .js » dans lequel on écrit le code JS (fichier appelé « script.js » ici) ;
- Associer ce fichier JS au fichier HTML comme ci-dessous avec l'instruction surlignée en jaune.

Le code HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Titre onglet</title>
  </head>
  <body>
    <!-- code HTML ici -->
    <script src="script.js"></script>
  </body>
</html>
```

Le code JS écrit dans `script.js`

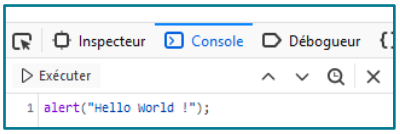
```javascript
alert("Hello World !");
```

Dans le HTML, on écrit la ligne surlignée en jaune pour faire référence au code JS, on utilise la balise `<script>` avec l'attribut `src` dont la valeur est le chemin du fichier dans lequel on écrit tout le code JS. **Cette ligne est à écrire juste avant la fermeture du `<body>`.**
Ici, seul le nom du fichier a été écrit, il faut donc que le fichier `script.js` soit dans le même répertoire que le fichier HTML.

Comment tester du code JavaScript ?

Pour écrire et tester du code JS on peut :

- Utiliser un **éditeur de texte** pour écrire le JS et le HTML (exemples : Geany, Atom, Notepad++, etc.) et un navigateur pour tester ;
- Utiliser un **bac à sable en ligne** comme repl.it (https://repl.it/) ou CodePen (https://codepen.io/) ou JSFiddle (https://jsfiddle.net/), etc. qui permettent de tester des codes directement en ligne sans enregistrer de fichiers sur votre ordinateur ;
- On peut également utiliser la **console JavaScript d'un navigateur**. Dans Firefox, on peut ouvrir la console Web par le raccourci Ctrl + Maj + K (ou dans Menu puis Développement web puis Console web). Il suffit alors d'écrire le code JS dans cette console puis l'exécuter. En phase de tests et de débogage il peut être intéressant d'utiliser la console du navigateur.



II. Interaction avec l'utilisateur dans une page Web

A faire **Activité : Défis JS**

Dans ce qui suit, comme dans les vidéos, on s'attache à écrire un code JavaScript simple et qui fonctionne dans la majorité des cas afin de ne pas entrer dans des considérations trop techniques du langage. En particulier, on utilise `var` pour déclarer toutes les variables, quelles que soient leur nature ; on utilise `querySelector` pour sélectionner un élément HTML, quel qu'il soit, etc.

A. Les événements

En JavaScript, un *événement* est une action qui se produit et qui possède deux caractéristiques essentielles :

- C'est une action qu'on peut « écouter », c'est-à-dire une action qu'on peut détecter car le système va nous informer qu'elle se produit ;
- C'est une action à laquelle on peut « répondre », c'est-à-dire qu'on va pouvoir attacher un code à cette action qui va s'exécuter dès qu'elle va se produire.

Un événement peut être associé à n'importe quel élément HTML (un bouton `<button>`, un paragraphe `<p>`, un titre `<h>`, un bloc `<div>` etc.). Il existe beaucoup d'événements (plus d'une centaine), en voici quelques-uns :

Evénement	Description
`click`	Cliquer sur l'élément
`dblclick`	Double-cliquer sur l'élément
`mouseover`	Faire entrer le curseur sur l'élément
`mouseout`	Faire sortir le curseur de l'élément
`keydown`	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
`keyup`	Relâcher une touche du clavier
`change`	Changer la valeur d'un champ de formulaire

B. Définir des gestionnaires d'événements

Pour écouter et répondre à un événement, on définit ce qu'on appelle des *gestionnaires d'événements* chargés à la fois d'écouter le déclenchement d'un événement et d'exécuter le code associé dès que l'événement se produit.

Aujourd'hui, en JavaScript, il existe trois grandes façons de définir un gestionnaire d'événements :

- On peut utiliser des attributs HTML de type événement (**obsolète** → à ne plus utiliser) ;
- On peut utiliser des propriétés JavaScript liées aux événements ;
- On peut utiliser la méthode `addEventListener()` (**recommandé** → c'est celle présentée dans les vidéos).

On détaille ci-dessous les trois méthodes permettant d'afficher le message « Hello World ! » dès lors que l'on clique sur un certain bouton de la page Web.

1ère méthode : Utiliser un attribut HTML de type événement (obsolète)

Le code HTML

```html
<body>
  <button onclick="afficheMsg()">Clic</button>
  <script src="script.js"></script>
</body>
```

Le code JS

```js
function afficheMsg() {
  alert("Hello World !");
}
```

Un attribut `onclick` est ajouté au bouton et sa valeur est la fonction JavaScript contenant le code à exécuter dès que l'événement survient. Souvent, les attributs possèdent le nom de l'événement qu'ils doivent écouter et gérer précédé par « `on` » (par exemple : `onclick`, `onmouseover`, `onmouseout`, etc.). En pratique, on ne procède plus ainsi mais il est possible de trouver cela dans des pages web plus anciennes (et même dans des sujets de BAC récents…).

2ème méthode : Utiliser les propriétés JavaScript liées aux événements

Le code HTML

```html
<body>
  <button id="btn">Clic</button>
  <script src="script.js"></script>
</body>
```

Le code JS

```js
var bouton = document.querySelector('#btn');

function afficheMsg() {
  alert("Hello World !");
}

bouton.onclick = afficheMsg;
```

3ème méthode : Utiliser la méthode `addEventListener()` (recommandée)

C'est le mécanisme d'événement le plus récent, le plus flexible et le plus performant (on ne rentrera pas dans les détails ici). C'est celui qui a été présenté dans les vidéos de l'activité et celui à privilégier aujourd'hui. Il fournit aux navigateurs une nouvelle fonction appelée `addEventListener()` qui fonctionne de la même manière que les propriétés du gestionnaire d'événement, mais la syntaxe est légèrement différente.

Le code HTML

```html
<body>
  <button id="btn">Clic</button>
  <script src="script.js"></script>
</body>
```

Le code JS

```js
var bouton = document.querySelector('#btn');

function afficheMsg() {
  alert("Hello World !");
}

bouton.addEventListener('click', afficheMsg);
```

Dans la fonction `addEventListener`, il faut spécifier deux paramètres : le nom de l'événement (ici `click` mais on peut remplacer par `mouseover` ou un autre) et le nom de la fonction à exécuter en réponse à cet événement.

Le code JS précédent peut s'écrire de manière équivalente de plusieurs façons, par exemple :

```js
function afficheMsg() {
  alert("Hello World !");
}

document.querySelector('#btn').addEventListener('click', afficheMsg);
```

Ou encore

```js
var bouton = document.querySelector('#btn');

bouton.addEventListener('click', function() {
  alert("Hello World !");
});
```

C. Modifier les éléments de la page

On considérera la page HTML ci-dessous pour les exemples qui suivent.

```
<body>
  <h1 id="monTitre">Généralités sur JS</h1>
  <p id="monPara">Voici une page Web
interactive grâce à JavaScript.</p>
  <button id="btn">Cliquez ici</button>
  <script src="script.js"></script>
</body>
```

Voici quelques exemples de fonctions permettant de modifier les propriétés des éléments de la page Web. On suppose que ces fonctions sont appelées lors du clic sur le bouton de la page comme on l'a vu dans le paragraphe précédent.

Modifier le style d'un élément : la propriété `style`

On peut modifier le style d'un élément HTML en utilisant la propriété `style` de cet élément. Par exemple, cette fonction Javascript

```
function changeStyles() {
  var para = document.querySelector("#monPara");
  var corps = document.querySelector("body");
  para.style.color="red";
  para.style.fontWeight="bold";
  corps.style.backgroundColor="rgb(255,255,0)";
}
```

permet de modifier le style du paragraphe : le texte passe en rouge et en gras ; le style du corps de la page : la couleur de fond passe en jaune.

Changer le texte d'une balise : la propriété `innerHTML`

On peut changer le texte d'un élément HTML en utilisant la propriété `innerHTML` de cet élément. Par exemple, cette fonction Javascript

```
function changeTexte() {
  var para = document.querySelector("#monPara");
  para.innerHTML = "Bonjour tout le monde !";
}
```

permet de modifier le texte du paragraphe.

Utiliser des champs de saisie : la propriété `value`

On ajoute dans le code HTML un élément `<input>` (=champ de saisie) avec la ligne

```
<input type="text" id="zoneDeSaisie"/>
```

dans lequel on demande à l'utilisateur d'écrire son prénom.
Il est alors possible de récupérer le prénom saisi en utilisant la propriété `value` de cet élément. Par exemple, si l'utilisateur a saisi le prénom « Brendan » alors cette fonction Javascript

```
function disBonjour() {
  var saisie = document.querySelector("#zoneDeSaisie");
  var para = document.querySelector("#monPara");
  let prenom = saisie.value;
  let texte = "Bonjour " + prenom;
  para.innerHTML = texte;
}
```

permet de récupérer le texte saisi grâce à la propriété `value` de l'élément `<input>` puis de construire la chaîne de caractères « Bonjour Brendan » et de l'écrire dans la paragraphe.

Bilan

- Lors de la navigation sur le Web, les internautes interagissent avec leur machine par le biais des pages Web.
- L'Interface Homme-Machine (IHM) repose sur la gestion d'événements associés à des éléments graphiques munis de méthodes algorithmiques.
- Un événement est une action qui se produit, comme un clic sur un bouton (`click`), le survol d'un élément avec la souris (`mouseover`), etc.
- On peut « écouter » un événement pour détecter lorsqu'il se produit et on peut lui « répondre » en lui associant un code qui va s'exécuter dès qu'il se produit.
- Le langage JavaScript, associé au HTML, permet d'implémenter la gestion de ces événements en utilisant ce qu'on appelle un *gestionnaire d'événement*.
- Aujourd'hui, on recommande d'utiliser la méthode `addEventListener` pour gérer les événements.

Ressources :

- Introduction aux événements, MDN Mozilla : https://developer.mozilla.org/fr/docs/Apprendre/JavaScript/Building_blocks/Ev%C3%A8nements
- Wikipédia : https://fr.wikipedia.org/wiki/JavaScript
- Cours de Pierre Giraud : https://www.pierre-giraud.com/javascript-apprendre-coder-cours/addeventlistener-gestion-evenement/
- Cours OpenClassrooms, Premiers pas en JavaScript : https://bit.ly/2UtkVOb