Cours NSI - Javascript

Créé en 1995 par Brendan Eich. Standardisé sous le nom d'**ECMAScript**.
Langage de programmation de scripts principalement employé dans les pages web interactives. Avec HTML et CSS, au cœur du `www`. Permet des pages web interactives et des applications web.
C'est un **langage interprété** : Le script (fichier texte) est lu par un interpréteur. Les navigateurs (Firefox, Chrome...) incluent un interpréteur.

Les navigateurs ont des différences ce qui peut occasionner des difficultés (ex : même commande → pas le même effet)

le DOM

Document Object Model = structure du document `html`. Il est organisé comme un arbre où chaque item a un parent et éventuellement des enfants. Chaque paragraphe, bouton, lien... est associé à un nœud dans l'arbre.

JavaScript permet d'accéder aux différents nœuds du DOM ce qui permet de le modifier (*ajouter un paragraphe, changer une couleur...*) et de réagir aux actions sur le DOM (*clic sur un bouton*)

Les événements

JavaScript permet de faire les mêmes choses que Python. Mais on va s'intéresser aux événements sur une page web. On parle de **programmation événementielle** : le navigateur se met à l'écoute et réagit à certains événements comme un clic de souris, le survol d'une zone avec le curseur, l'arrivée de données... L'événement déclenche alors une fonction.

Voici une liste (non exhaustive) des événements possibles. À chaque fois, l'événement est associé à un nœud du DOM.

- **`load` :** Quand le nœud est chargé. Permet d'attendre que la page soit chargée avant de commencer le traitement.
- **`click` :** clic souris sur le nœud.
- **`dblclick` :** double clic sur le nœud.
- **`mouseover` :** le curseur de la souris passe au-dessus du nœud.
- **`mouseout` :** le curseur arrête d'être au-dessus du nœud.
- **`input` :** quand on modifie le contenu d'un champ de saisie.
- **`change` :** comme le précédent mais après avoir validé par la touche *Entrée*.
- **`focus` :** quand le nœud gagne le *focus*. Dans une page où on trouve plusieurs zones de saisie, celle dans laquelle on est en train d'écrire est celle qui a le focus.
- **`blur` :** quand le nœud perd le focus.
- **`submit` :** dans le cas d'un formulaire (ensemble de champ de saisie avec un bouton "soumettre" ou "Envoyer" ou "ok" ou...) quand le formulaire est validé et peut-être envoyé.

Ajouter du Javascript dans une page - option 1 : dans la balise

```
<a href="mom_lien" onclick="alert('coucou !');">bla bla</a>
```

Cette méthode est simple mais n'a que des défauts : pas pratique si beaucoup de programmes, mélange le contenu html avec la forme...

Ajouter du Javascript dans une page - option 2 : bloc de script

```
<script language="javascript">
function traitement_du_clic(e){
  // le code de la fonction
}
</script>
```

À noter que ce bloc n'est pas forcément dans la zone *<head>*. Il peut être plus judicieux de le placer en fin de page de sorte que le script n'est examiné qu'à la fin du chargement de la page, quand le DOM est terminé.

On peut continuer à attacher les événements dans les balises. Par exemple ici :

```
<a href="mom_lien" onclick="traitement_du_clic">bla bla</a>
```

On peut aussi attacher l'événement à la balise *<a>* via le script :

```
document.getElementById("id_du_lien").addEventListener("click",
    traitement_du_clic);
```

Dans ce cas, il faut pouvoir identifier la balise. On l'a fait ici à l'aide de *id*, ce qui veut dire que le lien s'écrira ainsi dans le html :

```
<a href="mom_lien" id="id_du_lien">bla bla</a>
```

Il y a beaucoup de méthodes pour atteindre un nœud particulier du DOM.

Ajouter du Javascript dans une page - option 3 : fichier à part

On pousse la logique du cas précédent jusqu'au bout. On place le code JavaScript dans un fichier `mon_script.js` et on ajoute dans la page html :

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

Comparaison Python / JavaScript

Fin d'instruction : fin de ligne	Fin d'instruction : `;`
`:` et indentation délimite bloc	`{` et `}` délimitent les blocs
Commentaires après `#`	Commentaires après `//`
Pas de déclaration de variables	mieux de déclarer `let x = ...`
`def mafonction(...):`	`function mafonction(...) {` ou : `mafonction = function(...) {`

```
if test:
  instructions
else:
  instructions
```

```
if (test) {
  instructions
} else {
  instructions
}
```

```
while test:
  instructions
```

```
while (test) {
  instructions
}
```

```
for i in range(n):
  instructions sur i
```

```
for (let i = 0; index < n; index++) {
  instructions sur i
}
```

```
for item in tableau:
  instructions sur item
```

```
for (let item of tableau) {
  instructions sur item
}
```