1NSI	IHM & Web	TP – De l'interactivité avec JavaScript
	JavaScript	

Objectifs :

- ⇨ Apprendre les bases du langage JavaScript
- ⇨ Créer des pages web dynamiques avec du code JavaScript

I – Le langage javascript

1) Caractéristiques

JavaScript est un langage de programmation coté client[1] (il est exécuté par le navigateur sur la machine de l'utilisateur) à contrario de PHP qui lui est déporté coté serveur (et s'exécute donc sur la machine serveur distante).
C'est un langage interprété orienté objet et dont la syntaxe est très proche du C ou de Java.
Il possède un typage faible et dynamique : le type d'une variable est déduit du contexte (par exemple `let a = 3;` créera une variable de type entier – c'est le même comportement que python) et le langage procède à des conversions de type implicite dès que nécessaire.
Exemple :

```
let a = 4 ;          //Crée une variable a de type entier
let b = "abc" ;           //Crée une variable b de type chaîne de caractère
a = a + b ;
```

Ici la variable `a` vaudra `"4abc"` (donc elle prend le type chaîne de caractère) à la fin de ce bout de code. JavaScript va interpréter le signe « `+` » comme une opération de concaténation et donc convertir la valeur de `a` en la chaîne de caractère `"4"` pour ensuite la concaténer avec `b` et enfin stocker le résultat dans `a`.

2) Syntaxe

Toutes les lignes de code qui doivent être évaluées doivent se terminer par le signe « `;` ». Cela permet (même si ce n'est pas recommandé) de mettre plusieurs instructions sur la même ligne. Javascript est cependant tolérant et prendra quand même en compte une instruction à partir du moment où on met un saut de ligne après (sans « `;` »).
Les blocs d'instructions (dans les boucles, fonctions, structures conditionnelles) sont mis entre accolades « `{}` ». L'indentation n'est pas obligatoire (et ignorée par l'interpréteur), mais fortement recommandée pour la lisibilité du code.
Les variables JavaScript doivent être déclarées avec les instructions `var` ou `let` (il existe une différence subtile que nous ne détaillerons pas ici[2]) avant de les utiliser. On peut ensuite les modifier comme on le souhaite.

II – Utilisation dans une page

1) Déclaration des scripts

Pour utiliser du code Javascript dans une page HTML, il existe, comme pour CSS, 3 possibilités :

1. Dans l'en-tête du fichier HTML (déclaré par une balise `<script>`). Dans ce cas ces instructions sont exécutées avant de lire le corps (`body`) de la page. L'habitude est de déclarer les scripts à la fin de la section `<head>`.

[1] Il existe des versions de Javascript qui peuvent s'exécuter côté serveur, mais nous n'en parlerons pas dans ce TP.
[2] Pour les curieux, l'explication de la différence est là : https://medium.com/@vincent.bocquet/var-let-const-en-js-quelles-diff%C3%A9rences-b0f14caa2049

2. Ex :
```
<head>
  <script>
     alert("Démarrage de la page html");
  </script>
</head>
```

3. Dans le corps de la page html avec une balise `<script>`. Dans ce cas on place généralement le script à la toute fin du `body` au moment où tous les éléments de la page ont été examinés (et donc sont accessibles à JavaScript).

4. Dans un fichier `.js` dont on donne la référence dans la balise `<script>`. Dans ce cas les instructions sont valables pour toutes les pages qui font référence à ce fichier (souvent pour tout le site).
   Dans ce cas, l'attribut `src` permet de spécifier l'emplacement et le nom du fichier JavaScript. Ce fichier est chargé puis exécuté immédiatement avant de continuer l'analyse de la page. On peut changer ce comportement avec l'attribut `async` qui permet d'exécuter le script *pendant* que l'analyse de la page continue ou `defer` qui n'exécutera le script qu'une fois l'analyse de la page terminée.

   Ex :
```
<head>
  <script src="monScript.js" defer></script>
</head>
```
   Le script situé dans le fichier monScript.js sera chargé et exécuté une fois que la page aura été complètement analysée.

On peut utiliser ces 3 méthodes simultanément dans le même document et même utiliser plusieurs fichiers `.js` pour une même page html.

2) Le DOM

Pour pouvoir agir sur la page html, Javascript utilise un objet appelé DOM (Document Object Model) et noté `document`, qui est une interface permettant d'obtenir des informations sur la page web et de la manipuler. On pourra ainsi rajouter du contenu ou le modifier.
De même on peut agir sur les feuilles de style CSS avec l'objet `document.styleSheets` ou sur la fenêtre avec l'objet `window`.

a. Sélection d'un élément dans une page

Il existe de nombreuses méthodes du DOM pour référencer un élément de la page :

Instruction	Type de contenu
`document.getElementById(`*`id`*`)`	Sélectionne l'élément dont l'`id` est *id*. `let balise_avec_l_id_msg = document.getElementById("msg");`
`document.getElementsByTagName(`*`balise`*`)`	Sélectionne l'ensemble des balises du type *balise* (renvoi un tableau d'éléments). `let tab_balises_h2 = document.getElementsByTagName("h2");`
`document.getElementsByClassName(`*`classe`*`)`	Sélectionne l'ensemble des balises de la classe *classe* (renvoi un tableau d'éléments). `let tab_balises_classe_article = document.getElementsByClassName("article");`
`document.querySelector(`*`balise`*`)`	Sélectionne la première balise du type *balise* (renvoi un seul élément). `let premiere_balise_h2 = document.querySelector("h2");`
`this`	**fait référence à l'élément actuel (celui dans lequel est appelé une fonction par exemple)**

b. Modification d'un élément

Une fois l'élément récupéré, on peut adresser son contenu, ses attributs ou son style.

Instruction	Type de contenu
`element.innerHTML`	Contenu HTML de la balise (peut contenir d'autres balises)
`element.textContent`	Contenu texte de la balise (n'est pas interprété comme du code HTML)
`element.attribut`	Valeur de l'attribut `attribut` de l'élément
`element.style.propriété`	Valeur de la propriété `propriété` du style de l'élément

c. Gestion des évènements

On peut attribuer une fonction ou des instructions Javascript à divers évènements liés à un élément. Les évènements peuvent être par exemple : `click`, `mouseover`, `mouseout`, `mousedown`, `mouseup`, `change`, `focus`, `load`, `keypress`, …[3]

Pour assigner des instructions à un évènement survenant à un élément, on peut utiliser deux méthodes :

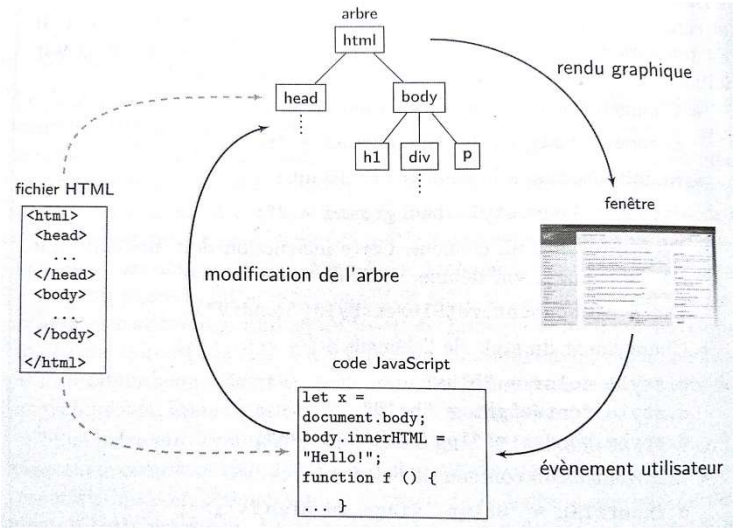
- L'assigner en tant que propriété dans la balise ouvrante (il faut alors rajouter « on » devant le nom de l'évènement).
  Ex : `<p onmouseover="this.style.color='red' ;">Paragraphe à survoler</p>`

- Ajouter un gestionnaire d'évènement à un élément existant grâce à la méthode `addEventListener(évènement, fonction)`.
  Ex : `document.getElementById("btn").addEventListener("click",ecritMessage);`

  Cette méthode est préférable car elle permet de ne pas mélanger le code javascript (dans le fichier .js) et la structure du document (dans le fichier .html), mais elle a l'inconvénient de ne pas pouvoir passer de paramètre à la fonction.

d. Evaluation du code JavaScript

Le code JavaScript est évalué par le navigateur à chaque fois qu'il rencontre une balise `<script>` jusqu'à ce que la fin du document soit atteinte et que l'arbre DOM ait été complété.
Le navigateur effectue le rendu graphique puis attend que des évènements déclenchent des fonctions JavaScript. Ces fonctions peuvent modifier le DOM qui est actualisé. Puis le navigateur attend l'évènement suivant et ainsi de suite.

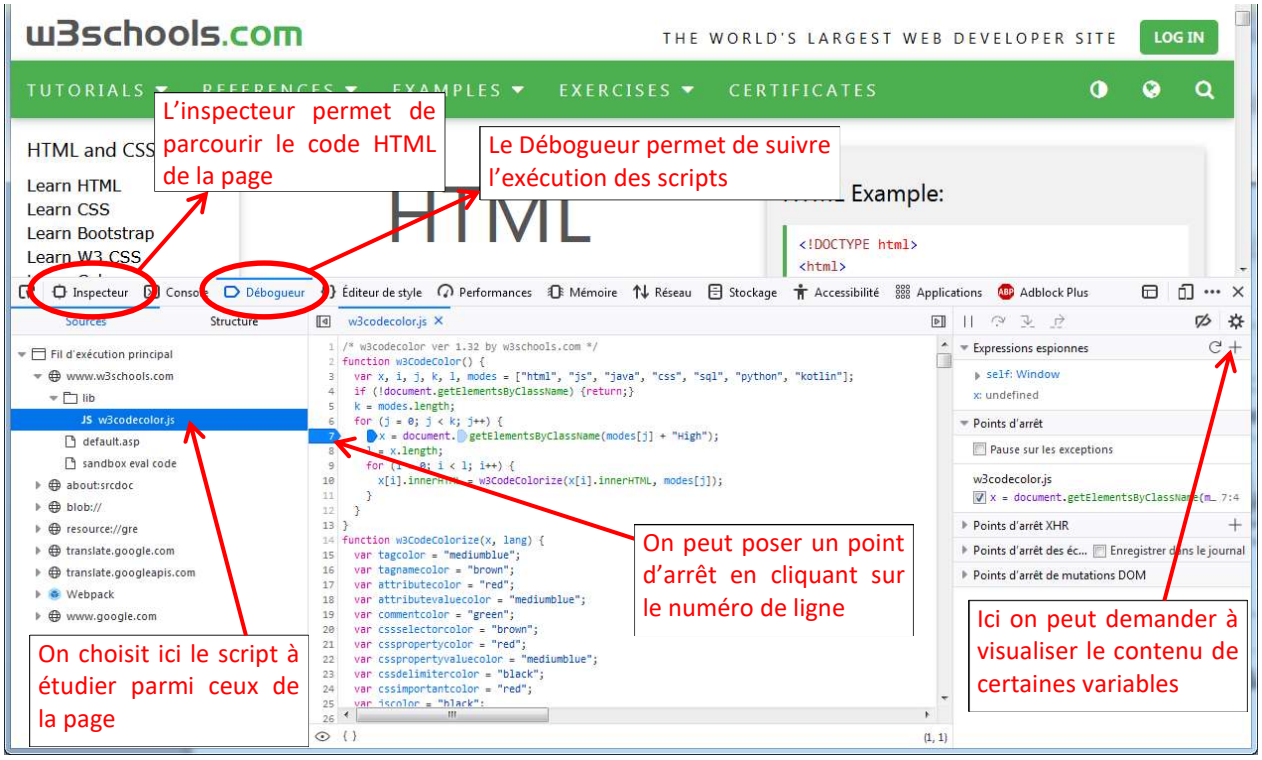


D'après « Numérique et sciences informatiques – 1ère », ed. Ellipses

3) Codage et débogage

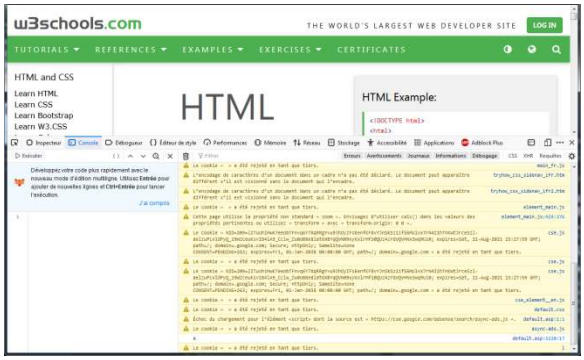
Pour programmer en Javascript on peut comme pour tout langage de programmation utiliser n'importe quel éditeur de texte (par exemple notepad++) pour créer le code source, puis l'exécuter dans le navigateur. Ensuite la touche F12 permet d'accéder aux outils de développement.

[3] Une liste plus exhaustive peut être trouvée sur https://www.w3schools.com/jsref/dom_obj_event.asp

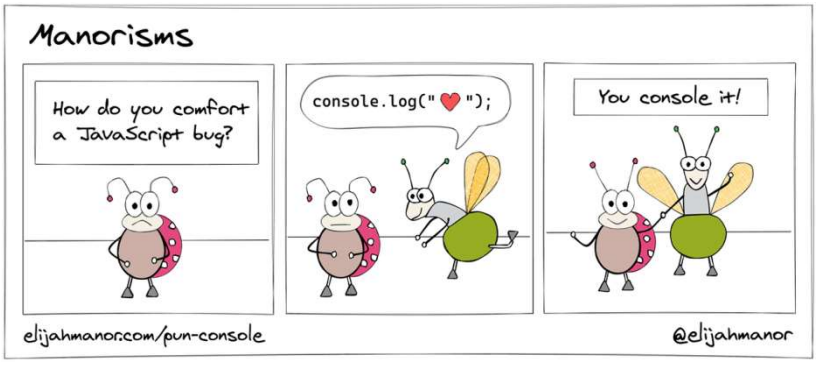


On a accès à différents outils, comme l'inspecteur qui permet de sélectionner des éléments de la page et de voir le code html associé, voire de modifier celui-ci (la modification se fait sur la copie en mémoire et pas sur la version stockée sur le serveur et disparaîtra si la page est rechargée).

Il y a également un débogueur qui permet d'arrêter l'exécution et de faire du pas à pas. Pour rappel, l'exécution des scripts est déclenchée automatiquement au moment où le navigateur tombe sur la balise `<script>` ou qu'un évènement précis est déclenché. Il n'y a donc pas de bouton pour « démarrer » JavaScript.



Enfin on a accès à la console qui affiche les messages d'erreur (partie de droite) et dans laquelle on peut également taper des commandes (partie de gauche) un peu comme dans la console python.
Ces messages d'erreur ne sont pas affichées par le navigateur en temps normal et ne sont visibles que si on ouvre la console.



Il est également possible d'écrire des informations de débogage avec la commande `console.log("texte");`.

On pourra rajouter des commentaires dans ses programmes en utilisant « `//` » pour un commentaire uniligne (équivalent de « `#` » en python), ou « `/*` …. `*/` » pour un commentaire multiligne (équivalent des guillemets triples en python).

III – Applications

Maintenant c'est à vous de jouer. En utilisant notepad++ et les outils de développement, jsbin ou codePen, faites les exercices suivants.

Références :

https://www.w3schools.com/js/DEFAULT.asp
https://www.pierre-giraud.com/javascript/cours-complet/javascript-presentation.php
https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector

Question 1 :
On considère la page html ci-contre.
1) D'après vous qu'affiche la page ?
2) En fait l'exécution du script provoque une erreur. Pourquoi ?
3) Trouver une solution à ce problème et la mettre en œuvre pour que la page s'affiche comme prévu.

```
<html>
 <head>
  <script>
   let coul = prompt("Choisir une couleur)");
   document.getElementById("titre1").style.color = coul;
  </script>
 </head>
 <body>
   <h1 id="titre1">Ceci est un titre en couleur</h1>
 </body>
</html>
```

Dans la pratique on solutionne généralement le problème en plaçant les scripts dans un fichier .js séparé que l'on exécutera une fois la page complètement chargée avec le mot-clé `defer`. On peut également mettre toutes les initialisations dans une fonction « `init` » et rajouter l'attribut « `onload="init"` » à la balise `<body>`.

Question 2 :
On utilisera pour la suite du TP la page html « Question2.html ».
Rajouter une fonction dans le script (et éventuellement changer la page HTML) pour que le compteur de la première ligne s'incrémente de 1 à chaque fois qu'on clique sur le bouton « +1 ».

Aide : Il faut créer une fonction « `incrementeCompteur` » puis attacher un gestionnaire d'évènement « click » au bouton « +1 » avec la ligne : `document.getElementById("compteur").addEventListener("click",incrementeValeur);`
On peut à la place rajouter un attribut « `onclick="incrementeValeur()"` » au bouton « +1 ».

Question 3 :
On utilisera pour la suite du TP la page html « Question2.html ».
Rajouter une fonction dans le script pour changer la couleur du mot « JavaScript » dans le titre, puis modifier la page HTML pour que la couleur du mot « JavaScript » change lorsqu'on survole les cases du tableau (chaque case change la couleur à la valeur qu'elle indique (la case « Rouge » change en `"red"`, « Bleu » en `"blue"`, etc…).

Question 4 :
1) Sur la page précédente, rajouter un bouton « Montrer le texte secret » qui lorsqu'on clique dessus affiche le texte caché (la balise div dont l'id est `"secret"`).
2) Modifier votre code pour qu'une fois le texte affiché, le bouton « Montrer le texte secret » se transforme en bouton « Cacher le texte secret » et qu'un clic dessus permette de re-cacher le texte (et de revenir au bouton « Montrer le texte caché »).

Aide : Il faut commencer par écrire une fonction « `montreTexteSecret` » qui modifie l'attribut `"display"` du div caché pour lui attribuer la valeur `"block"`.

Question 5 :
Sur la page précédente, rajouter un bouton « Animation couleur » qui lorsqu'on clique dessus fait cycler les couleurs du mot « JavaScript » parmi la liste suivante :
`["red", "orange", "yellow", "lightgreen", "cyan", "blue", "purple"]`
Si vous avez le temps, essayer de faire en sorte que l'animation s'arrête si on re-clique sur le bouton.

Aide : Il n'existe pas de fonction pour « attendre un certain temps » en JavaScript car il n'y a qu'un seul processus JavaScript et s'il devait se mettre en pause, cela bloquerait tous les autres évènements JavaScript. On doit donc *planifier* la prochaine exécution de la fonction qui change la couleur. Elle sera ainsi déclenchée par un évènement de type « minuteur » et entre temps JavaScript aura pu gérer tous les évènements se présentant.
Pour ce faire, on doit utiliser la procédure :
`window.setTimeout(fonction, délai[, arguments_de_la_fonction])`.
Elle permet de déclencher la fonction `fonction` au bout de `délai` millisecondes. On peut après les deux premiers placer des arguments optionnels qui seront passés comme paramètres à la fonction.

TRAVAIL A FAIRE POUR LA PROCHAINE SEANCE :

Suivre le cours sur le tutoriel (en anglais) : https://jgthms.com/javascript-in-14-minutes