Chapitre 4 : Interactions homme/machine sur le Web

I. Introduction

1) Histoire du Web *(prise de notes)*

Quelques événements marquants résumés dans le tableau ci-dessous. Les dates ne sont pas à retenir mais les grandes étapes sont importantes.

Année	Évènement
1958	Laboratoires Bell invente le modem, permettant de transmettre des données binaires sur une ligne téléphonique.
1962	Initiative américaine DARPA : Licklider avance l'idée d'un réseau continental d'ordinateurs, en partie pour se défendre durant la guerre froide.
1968	Protocole d'échange de données par commutation par paquets. On n'échange plus un octet de données par un octet mais des blocs d'octets. Cela va plus vite!
1969	Connexion des premiers "ordinateurs" (noeuds) entre quatre universités américaines (réseau ARPAnet). Les ordinateurs sont des ordinateurs centraux reliés à de multiples terminaux au sein de l'université.
1971	23 ordinateurs sont reliés sur ARPANET. Envoi du premier courriel par Ray Tomlinson.
1973	Définition du protocole TCP/IP : TCP (Transmission Control Protocol) et IP (Internet Protocol).
1983	Adoption du protocole TCP/IP et du mot « Internet ».
1984	1 000 ordinateurs connectés.
1989	100 000 ordinateurs inter-connectés.
1991	Annonce publique du World Wide Web (Tim Berners-Lee).
1992	1 000 000 ordinateurs connectés.

Le "World Wide Web", plus communément appelé "Web" a été développé au CERN (Conseil Européen pour la Recherche Nucléaire) par le Britannique Sir Timothy John Berners-Lee et le Belge Robert Cailliau au début des années 90.

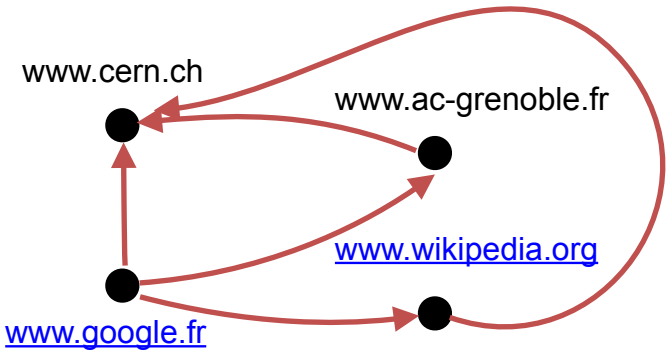
À cette époque les principaux centres de recherche mondiaux étaient déjà connectés les uns aux autres (notamment via ARPAnet aux USA, Cyclades en France et X.25 à une échelle européenne), mais pour faciliter les échanges d'information Tim Berners-Lee met au point le **système hypertexte**.

Le système hypertexte permet, à partir d'un document, de consulter d'autres documents en cliquant sur des mots clés. Ces mots "cliquables" sont appelés **hyperliens** et sont par

défaut soulignés en bleu. Ces hyperliens sont maintenant connus sous le simple terme de "liens".

On passe d'une vision par blocs déconnectés (google est déconnecté de cern.ch ou de wikipedia.org) à une vision de type graphe :

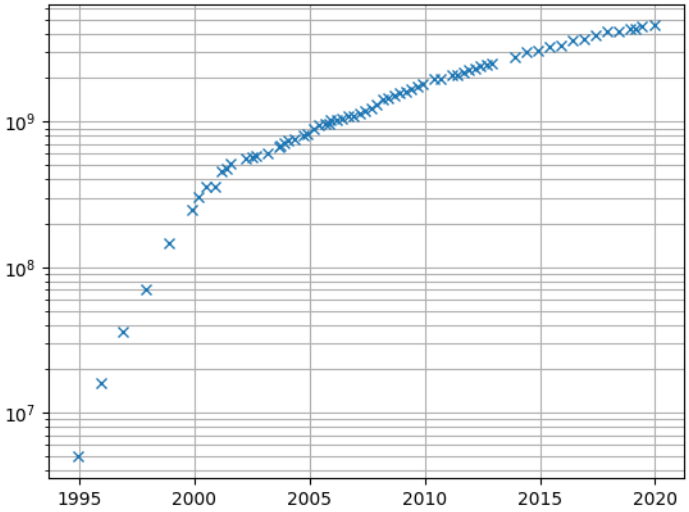
www.cern.ch	www.ac-grenoble.fr
www.google.fr	www.wikipedia.org



La première page web est toujours consultable à l'adresse suivante : http://info.cern.ch/hypertext/WWW/TheProject.html

Sir Timothy Berners-Lee développe aussi le premier logiciel permettant de lire des pages contenant des hypertextes : c'est le navigateur web. Il l'appelle simplement "WorldWideWeb". **Face Palm** : la première idée de Tim avait été d'appeler le WWW "The Information Mine"… Les initiales auraient fait TIM !

Il faudra attendre 1993 et l'arrivée du navigateur web "NCSA Mosaic" pour que le web commence à devenir populaire en dehors du petit monde de la recherche.



Depuis, le nombre d'utilisateurs et de serveurs en activité dans le monde n'ont cessé de croitre. L'information devient virtuellement universellement accessible.

En 2000, on note la présence d'un plateau lié à la crise économique provoquée par une croissance trop rapide des technologies d'internet….

JS -> 1995, CSS -> 1993.

2) Vocabulaire et standards du Web *(prise de notes)*

Il existe une confusion entre "web" et "internet". Même si le "web" "s'appuie" sur internet, les deux choses **n'ont rien à voir** :

- le réseau "internet" est un "réseau de réseau" s'appuyant sur le protocole IP
- le **"web" est la combinaison de trois technologies** : HTTP, URL et HTML.

D'ailleurs on trouve autre chose que le "web" sur internet, par exemple, les emails avec le protocole SMTP (Simple Mail Transfert Protocol) et les transferts de fichiers avec le protocole FTP (File Transfert Protocol). En Peer-To-Peer, on ne passe pas non plus par le web. On utilise seulement le réseau Internet.

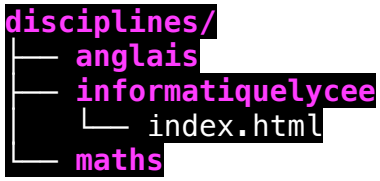
Techniquement le web se base sur trois composants :
- le protocole HTTP (HyperText Transfert Protocol)
- les URL (Uniform Resource Locator)
- le langage de description HTML (HyperText Markup Language).

Nous allons rapidement discuter des URL. Prenons par exemple l'adresse. C'est une URL :

http://www.ac-grenoble.fr/disciplines/informatiquelycee/index.html

Il y a 3 éléments dans cette "phrase" :
- l'indication du protocole : ici, http://
- le nom du serveur : www.ac-grenoble.fr
- le location de la ressource dans le serveur : /disciplines/informatiquelycee/index.html .

En plus du protocole et du nom du serveur, l'URL donne la position de la page que l'on cherche à accéder dans l'arborescence du site Web. Ainsi, par rapport à notre exemple, il est fort à parier que le fichier HTML index.html que l'on souhaite atteindre se trouve sur le serveur ac-grenoble.fr dans un dossier racine **disciplines** et plus précisément dans le sous-dossier **informatiquelycee** :



Autre exemple :
http://www.michel-mounier.fr/wp-content/uploads/2020/06/2004-Topo-Bloc-Bastille.pdf

Notons que lorsque l'on connaît l'arborescence du site web, on peut se promener où l'on veut dans les ressources… Ce n'est pas très sécurisé !

Pour éviter que l'on connaisse cette arborescences, l'URL ses sites internet comporte souvent des "Slug" après le nom de serveur (chaines de caractères + nombre aléatoire)

Exemple :
https://www.education.gouv.fr/fonctionnement-de-l-ecole-41510

Slug

Vous avez déjà étudié le langage de description HTML et nous allons compléter vos connaissances à la prochaine séance.

Le protocole HTTP sera abordé chapitre 8.

II. Interactivité dans une page web, côté client

1) Clients et serveurs *(prise de notes)*

Deux ordinateurs en réseau peuvent s'échanger des données. Pour des raisons d'efficacité, dans la plupart des cas, ces échanges ne sont pas "symétriques" : en effet un ordinateur A va souvent se contenter de demander des ressources (fichiers contenant du texte, photos, vidéos, sons...) à un ordinateur B.
L'ordinateur B va lui se contenter de fournir des ressources à tous les ordinateurs qui lui en feront la demande.

Définition : On dit que l'ordinateur A, qui demande des ressources, est un client alors que l'ordinateur B, celui qui fournit les ressources, est un serveur.

En tapant «http://www.google.fr», votre machine va chercher à entrer en communication avec le serveur portant le nom «www.google.fr» *(en fait, c'est plus compliqué, pour les puristes nous dirons donc que la communication va être établie avec le serveur www du domaine google.fr, mais bon, pour la suite nous pourrons nous contenter de l'explication « simplifiée »)*.

Leur montrer la vidéo : https://www.youtube.com/watch?v=CIhalbnBgA4

Une fois la liaison établie, le client et le serveur vont échanger des informations en dialoguant (de manière schématique):

client : bonjour www.google.fr (ou bonjour www se trouvant dans le domaine google.fr), pourrais-tu m'envoyer le fichier index.html

serveur : OK client, voici le fichier index.html

client : je constate que des images, du code css sont utilisés, peux-tu me les envoyer ?

serveur : OK, les voici

Sur internet, ce modèle client/serveur domine assez largement, même s'il existe des cas où un ordinateur pourra jouer tour à tour le rôle de client et le rôle de serveur, très souvent, des ordinateurs (les clients) passeront leur temps à demander des ressources à d'autres ordinateurs (les serveurs) . Par exemple, comme expliqué dans l'exemple ci-dessus on retrouve cet échange client/serveur à chaque fois que l'on visite une page web. Il y a de fortes chances pour que votre ordinateur personnel joue quasi exclusivement le rôle de client (sauf dans le cas du "peer to peer").

N'importe quel type d'ordinateur peut jouer le rôle de serveur, mais dans le monde professionnel les serveurs sont des machines spécialisées conçues pour fonctionner 24h sur 24h. Ils peuvent aussi avoir une grosse capacité de stockage afin de stocker un grand nombre de ressources (vidéos, sons,...).

Afin assurer une continuité de service, dans les sociétés, plusieurs serveurs assurent exactement le même rôle (on parle de redondance). Vous vous doutez bien que Google ne possède pas qu'un seul serveur, en effet, en moyenne, chaque seconde, c'est environ 65000 clients qui se connectent aux serveurs du moteur de recherche de Google.

Aucun serveur, même extrêmement performant, ne serait capable de répondre à toutes ces requêtes. Google, Amazon ou encore Facebook possèdent un très grand nombre de serveurs afin de pouvoir satisfaire les demandes des utilisateurs en permanence. Ces entreprises possèdent d'immenses salles contenant chacune des centaines ou des milliers de serveurs (ces serveurs sont rangés dans des armoires appelées "baie serveur").

Souvent les serveurs sont spécialisés dans certaines tâches, par exemple, les serveurs qui envoient aux clients des pages au format HTML sont appelés "serveur web".

II. Interactivité dans une page web, côté client

Il y a quelques années, le Web était statique et tous les utilisateurs voyaient la même page web. Vous avez réalisé cela pendant les vacances. De nos jours, pratiquement toutes les pages web sont **dynamiques** et intègrent d'autres composants que le simple code HTML5.

1) Interactions grâce à CSS — ressources Fondation Mozilla

Une manière très simple d'interagir avec un utilisateur consiste à le faire grâce aux fichiers CSS (Cascading Style Sheet). Cette interaction, grâce à l'existence des classes et des pseudo-classes en CSS, permet de faire réagir n'importe quel élément HTML à des actions d'un utilisateur.

Rappel : une classe en CSS est définie grâce à un point suivi d'un **sélecteur**. Par exemple, la classe appelée couleur s'écrit :

```
.couleur {
    color: red;
    background-color: aliceblue;}
```

Pour l'appeler, on utilisera une balise **<h1 class=couleur> le titre </h1>**.

Les pseudo-classes permettent de modifier les classes suite à des **interactions** entre l'homme et le navigateur. Sur le site de **Mozilla**, recherchez quelques **pseudo-classes** CSS qui semblent intéressantes pour nous et notez la syntaxe ci-dessous.

.....

.....

❐ — Exercice **1** —

Téléchargez le dossier zippé IMH.zip sur bouillotvincent.github.io et dézippez-le immédiatement : allez dans l'exercice 1. On souhaite créer un recueil de poèmes interactifs.

Si vous double-cliquez sur index.html, la page web se lance et vous pouvez tester votre page HTML. Pour la modifier, faites un clic droit "Ouvrir avec" puis choisissez Brackets/Geany.

On dispose de trois images de poèmes de Baudelaire que l'on veut mettre côte à côte. Lorsque l'on passe notre curseur sur un de ces poèmes, on souhaite qu'il se **transform**(e) à l'aide de **CSS** (cherchez ces mots-clés sur Mozilla…) :

- en s'agrandissant d'un facteur 5 ;
- en se translatant sur l'axe des X de 40% ;
- en se translatant sur l'axe des Y de 20%.

Dans le fichier CSS, on utilisera la classe zoom et la pseudo-classe **hover** appliquée à zoom. Dans le fichier HTML, on n'oubliera pas d'appliquer la classe **zoom** aux images.

❐ — Exercice **2** —

On veut maintenant jouer un peu avec le texte.

1) On dispose d'un poème de Verlaine qui est bien trop long. On ne souhaite donc afficher que les 100 premiers pixels verticaux du poème.
Pour cela, modifiez la classe hide en ajoutant un attribut appelé **overflow**. Cherchez comment faire sur le site de Mozilla.

2) Cela a trop bien marché… On ne peux plus lire le poème ! On veut donc utiliser la pseudo-classe **active** appliquée à la classe hide pour afficher le poème tant que l'utilisateur garde le bouton gauche de la souris appuyé.

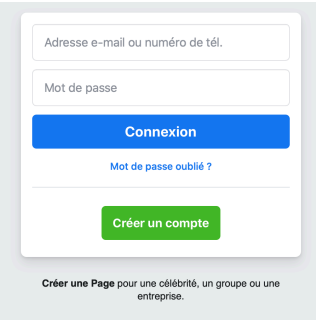
3) Comme dans l'exemple projeté au tableau, on souhaite conserver l'encadrement de telle manière à ce que celui-ci s'adapte à la nouvelle taille du texte. Pour cela on va changer **max-height.** Recherchez cette propriété pour savoir comment adapter la taille verticale!

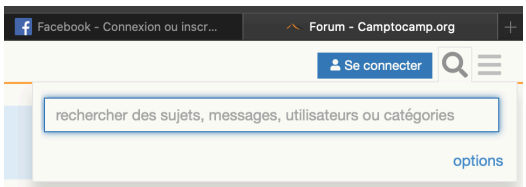
Conclusion : Les CSS permettent de faire de superbes effets visuels mais demandent un investissement très important en terme de temps de travail. C'est en fait un métier plutôt bien payé : 52 691$ en moyenne aux USA.

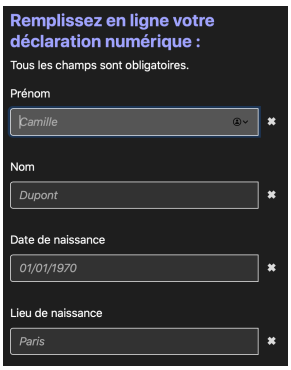
2) Interactions grâce à HTML

Même si, en apparence, cela n'est pas très sexy, HTML permet d'interagir avec l'utilisateur grâce à toute une ribambelle **d'inputs** dans des formulaires.

Les formulaires sont les éléments les plus importants des sites internet car ils permettent à l'utilisateur de transmettre des informations au serveur. Ils se retrouvent partout :







En HTML, un formulaire est délimité par une balise **form** (même si ce n'est pas une obligation) :
<form>
…..
</form>

Trois types de balises sont utilisés pour construire un formulaire : **select**, **input** et **label**.
- label possède un attribut **for** permettant de contrôler un input.
- input possède un attribut **type** qui lui permet de jouer des rôles différents.
- select permet de créer des listes déroulantes et utilise la balise **option** pour créer les différents items de la liste déroulante.

❐ — Exercice **3** —

Dans le dossier IMH, allez dans l'exercice 2 et en recherchant les mots-clés ci-dessus sur https://developer.mozilla.org/fr/docs/Web/HTML/Element , explorez les différentes possibilités de formulaire. Créez un clone de la page proposée dans l'exercice 2.

Si vous avez fini en avance, créez un CSS permettant de mettre en forme votre formulaire de création de Pokémon !

Conclusion : dans cette partie, nous avons créé un superbe formulaire. Résumez en un mot ce que fait ce formulaire pour l'instant : _____
Il va donc falloir qu'un serveur récupère notre formulaire pour en faire quelque chose. Mais cela sera pour un autre chapitre !

3) Interactions grâce à JavaScript *(une introduction)*

Un développeur internet ne peut aujourd'hui pas se passer de Javascript. Toutefois, en NSI, le but n'est pas d'apprendre ce nouveau langage de programmation, mais juste de comprends comment Javascript peut dynamiser vos sites web au travers de quelques exemples. Il est vrai que c'est un peu dommage de ne pas aller plus loin…

Un peu d'histoire : JavaScript a été créé en **dix jours** par Brendan Eich en 1995. Malgré son nom, JavaScript n'a rien à voir avec le langage Java. Il s'appelait initialement LiveScript mais a pris le nom de JavaScript pour profiter de la notoriété de Java : c'est donc un coup de pub !

❐ — Exercice **4** —

Dans le dossier Exo3, vous avez trois fichiers : index.html, style.css et script.js.

Ouvrez le fichier index.html.
Y'a-t-il une ligne nouvelle ? D'après vous, que fait cette ligne ? _____

_____

❒ — Exercice **5** —

Dans le dossier Exo3, à l'aide de Brackets, modifiez le fichier **script.js** et saisissez le code ci-dessous :

```
alert("Le JavaScript fonctionne !")
```

Rechargez la page index.html sur le navigateur . Que remarquez-vous ?

_____

Rappel : Le but n'est pas d'apprendre à programmer en JavaScript : nous nous contenterons pour le moment de cette simple instruction "alert" qui crée un popup. Evidemment, JavaScript permet de faire bien plus de choses : variable, condition, boucle, fonction… Si vous voulez en apprendre plus sur la programmation en JavaScript, je vous invite à consulter le site de Mozilla!

Dans l'exemple ci-dessus, l'instruction "alert" est exécutée dès l'ouverture de la page web, il est tout à fait possible de lier l'exécution de certaines instructions JavaScript à l'action d'un utilisateur (par exemple un clic sur un bouton).

❒ — Exercice **6** —

Dans le code HTML, ajoutez un bouton dans le corps du fichier à l'aide de l'instruction :

```
<button onclick="maFonction()">Cliquer ici</button>
```

Modifiez le code JavaScript dans le bon fichier comme suit :

```
function maFonction() {
    alert("Le JavaScript fonctionne !")
}
```

A quel moment le code JavaScript est-il exécuté ? _____

Décrivez avec vos propres mots ce qu'il se passe lorsque l'on ouvre le site internet **puis** que l'on appuie sur le bouton.

_____

_____

_____

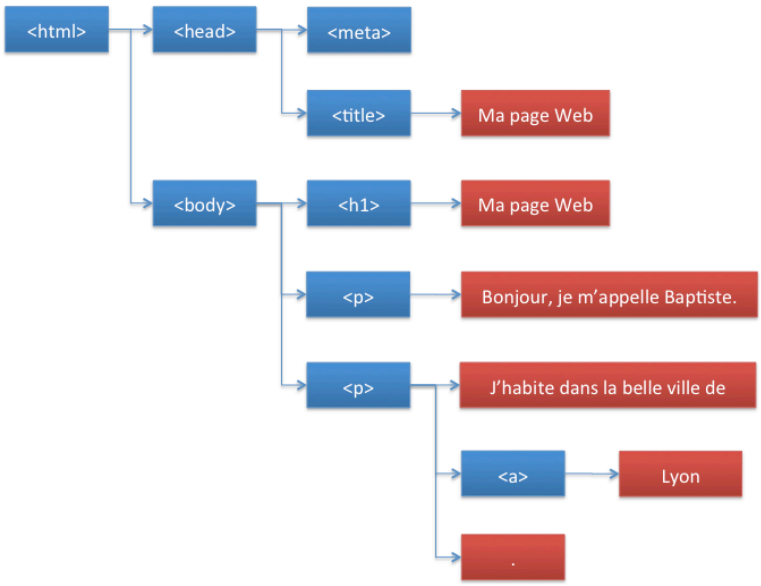
_____

4) Interactions grâce à JavaScript *(pour aller plus loin)*

Pour afficher un résultat dans un navigateur, ce dernier parcourt le code HTML d'une page afin de construire une représentation de sa structure. Ensuite, il utilise cette représentation pour afficher les différents éléments de la page.

Cette représentation de la structure d'une page web offerte par un navigateur est exploitable via JavaScript : elle est appelée **DOM**, pour **Document Object Model**. Le DOM définit la structure d'une page et le moyen d'interagir avec elle : il s'agit d'une interface de programmation, ou API (Application Programming Interface).

De manière très schématique, le DOM est l'ensemble des balises imbriquées les unes dans les autres de votre site web. On peut la représenter sous une forme d'arborescence qui ressemble à quelque chose comme cela :



En accédant aux éléments noeuds de cette arborescence (pas les feuilles finales — terminales — de l'arbre), il est possible de faire des choses bien plus complexes que l'affichage d'un simple pop-up avec JavaScript.
Par exemple, il est possible de modifier le style d'une balise, de modifier la classe (CSS) d'une balise ou encore de modifier le contenu d'une balise.

❐ — Exercice **7** —

Dans le code HTML de l'exercice 3, ajoutez un autre paragraphe :

```
<p id="para2">J'aime le rouge.</p>
<button onclick="maFonction2()">Cliquer ici</button>
```

Modifiez le code JavaScript comme suit :

```
function maFonction2() {
    document.querySelector("#para2").style.color="red";
}
```

Testez votre page HTML.

En sachant que le style CSS pour changer la couleur s'appelle "color", décrivez le fonctionnement de la ligne de code de la fonction maFonction() :

_____

_____

_____

Modifier le fichier JS afin d'afficher aussi une bordure avec des tirets après le clic :

_____

Dans cet exemple, on modifie le style directement de l'id #para2 directement dans le fichier HTML **(cela n'est pas interdit mais fortement déconseillé!)**. Il est possible de travailler plus "proprement" en utilisant les classes CSS :

❐ — Exercice **8** —

Dans le code HTML de l'exercice 3, ajoutez un autre paragraphe et deux boutons :

```
<p id="para3">J'aime encore plus le CSS.</p>
<button onclick="foncRouge()">Rouge</button>
<button>Noir</button>
```

Modifiez le code JavaScript comme suit :

```
function foncRouge() {
    document.querySelector("#para3").classList.remove("vert");
    document.querySelector("#para3").classList.add("rouge");
}
```

Ajoutez au CSS les lignes ci-dessous :

```
.rouge {
    color:red;
    font-size:20px;
}
```

Testez votre page HTML. Que remarquez-vous ?

_____

Explication : La fonction JavaScript "foncRouge()" permet de modifier la classe CSS de la balise ayant pour id "para3".

Dans un premier temps, la ligne

document.querySelector("#para3").classList.remove("vert")

permet de supprimer **l'association** entre la balise d'id "para3" et la classe CSS "vert" (si cette association n'existe pas, cette ligne n'a aucun effet). Dans un deuxième temps, on associe la classe CSS "rouge" avec la balise d'id "para3" avec la ligne

document.querySelector("#monPara").classList.add("rouge");

❐ — Exercice **9** —

Modifiez votre code HTML, JS et CSS afin de formater en noir et en caractère taille 20 l'id "para3" suite à un clic sur le bouton Noir.

Testez votre page HTML.

Il est également possible de modifier le contenu d'une balise HTML !

❐ — Exercice **10** —

Dans le code HTML de l'exercice 3, ajoutez un nouveau paragraphe et un bouton :

```
<p id="para4">J'adore cliquer.</p>
<button onclick="modifMessage()"> Click! </button>
```

Modifiez le code JavaScript comme suit :

```
var nombreClick=0;
function modifMessage() {
    document.querySelector("#para4").innerHTML = "Bravo, vous avez cliqué x fois sur le bouton !";
}
```

Testez votre page HTML.
Que fait la variable globale nombreClick ?

_____

On veut que mon message change en fonction du nombre de clicks et affiche : "Bravo, vous avez cliqué 1 fois sur le bouton !"
"Bravo, vous avez cliqué 2 fois sur le bouton !"
"Bravo, vous avez cliqué 3 fois sur le bouton !"
etc.

En transposant vos connaissances (concaténation de chaine de caractères…) en Python au Javascript, faites les modifications nécessaires dans le fichier JS.

Bien sur, il existe énormément d'autres événements HTML que "onclick". Il est possible de détecter le "survol" par le curseur de la souris d'un élément HTML.

❐ — Exercice **11** —

Dans le code HTML de l'exercice 3, rajoutez un dernier paragraphe :
Survolez-moi

Modifiez le code JavaScript comme suit :

```
function foncEntre(){
    document.querySelector("#maDiv").classList.remove("blanc");
    document.querySelector("#maDiv").classList.add("rouge2");
}
function foncQuitte() {
    document.querySelector("#maDiv").classList.remove("rouge2");
    document.querySelector("#maDiv").classList.add("blanc");
}
```

Ajoutez les lignes suivantes au code CSS :

```
#maDiv{
    width : 200px;
    height : 100px;
    margin : 0 auto;
    border : 2px solid black;
}
```

Ajoutez deux classes CSS rouge2 (respectivement blanc) dont la couleur du fond est rouge (resp. blanc).

Après avoir réalisé les modifications et analysé le code ci-dessus, testez cette nouvelle page en cliquant sur le fichier index.html

Conclusion : le code HTML permet de générer des éléments graphiques qui seront affichés par un navigateur web, mais pas seulement : il est aussi possible de mettre en place dans le code HTML des **événements**.

Un événement donné pourra déclencher l'exécution d'instructions JavaScript : onclick, onmouseover, onkeydown, addEventListener sont de ceux-là.

Exo 4 : approfondissement : recherchez comment ajouter une applet Géogebra dans une page web à l'aide de Javascript. Le but est de faire apparaitre une telle applet à la suite d'un clic sur un bouton (voir imprim' écran dans l'énoncé dans l'exercice.)