**Les sites et le Javascript**

Le JavaScript, ce langage de programmation dont on entend toujours parler combiné à du HTML ainsi que du CSS et qui forment ensemble le trio de base des outils pour créer un site, a souvent des utilités qui paraissent floues aux débutants.

Nous allons voir ici quelques utilisations basiques qui pourraient vous donner des idées pour votre prochain site.

# Index

1. La programmation
2. L'interaction avec l'HTML
3. L'interaction avec le CSS
4. Les évènements
5. Stockage simple
6. Mise en pratique

## **1. La programmation**

La base du JavaScript est d'être un langage de programmation c'est à dire qu'il nous donne accès aux capacités basiques suivantes :

Enregistrer des données

**let nom = "Jean Kirstein";**

Recevoir des informations de l'utilisateur

// Demander l'âge et l'enregistrer sous le nom age

**let age = prompt("Quel est votre âge ?");** la réponse ne sera pas directement affichée, il faudra l’appeler via un console.log par exemple

## Faire des actions

// A l'exemple précédent **prompt** était une action qui permettait d'ouvrir une boîte pour que l'utilisateur puisse passer des informations. Cette action existe de base dans JavaScript. comme pour console.log ou date

// On peut également créer nous-mêmes une action que l'on pourra utiliser autant de fois qu'on le souhaite.

**function saluer() { alert("Hello world !");** // Affiche "Hello world !" dans une boîte.

**} saluer();** // Lancer la fonction

## Varier ces actions selon les informations qui lui sont données

// Ici notre action pourra prendre une information qui sera enregistrée sous le nom **age** à l'intérieur de cette action

**function direAge(age) { alert(age);**

**} let ageUtilisateur = prompt("Quel est votre âge ?"); direAge(ageUtilisateur);**

## Répéter des actions

**// Trois parties : d'abord on crée un compteur, puis on pose une condition qui sera vérifiée à chaque répétition et enfin on augmente la valeur de compteur de 1 à la fin de chaque répétition (grâce au "++" collé à**

**"compteur") for (let compteur = 0; compteur < 3; compteur++) { alert(compteur**); // On affiche la valeur stockée dans compteur

}

Prendre des actions différentes selon des conditions posées

**let age = prompt("Quel est votre âge ?"); if (age < 18) { // Si l'âge est inférieur à 18 alors faire les actions du bloc alert("Vous n'avez pas l'âge requis pour consulter ce contenu");**

**} else { // Sinon faire les actions de ce bloc-ci alert("Chargement du contenu demandé");**

**}**

## Faire des manipulations de nombre et de chaînes de caractères **alert(2 + 17); // Affiche 19**

**alert(8424 \* 24905); // Affiche 209799720, aucune difficulté à accomplir des calculs compliqués**

**alert("Hello " + "World " + "!"); // Regroupe des chaînes de caractères ensemble grâce au "+" et alert affichera donc "Hello World !" alert("Bonjour ".repeat(2)); // Affiche "Bonjour " répété 2 fois alert("B" + "o".repeat(5) + "njour "); // Affiche "Booooonjour"**

## Organiser les informations grâce à des structures de données

**// Première structure de données, les tableaux servent à ordonner des données et chacune de ces données est stockée à une place numérique ("index") commençant à partir de 0 et sont principalement utilisées pour regrouper et ordonner des valeurs de même type (Ex: les notes des élèves d'une classe) let tableauMultipleDeux = [0, 2, 4, 6, 8]; alert(tableauMultipleDeux); // Affiche mon tableau entier alert(tableauMultipleDeux[0]); // Affiche le premier élément du tableau alert(tableauMultipleDeux[4]); // Affiche le dernier élément du tableau**

**// Deuxième structure de données, les objets servent à stocker des paires de clefs/valeurs (un peu comme un dictionnaire, on a un mot qui correspond à une valeur) et sont principalement utilisées pour regrouper des valeurs ayant des liens les unes avec les autres**

**let objetVoiture = {marque: "BMW", annee: 2016, couleur: "Rouge", prix: 100000}; // On crée ici une représentation d'une voiture grâce à un objet**

**// On ne peut pas afficher directement tout le contenu d'un objet avec alert mais on peut afficher chacune des valeurs alert(objetVoiture.marque); // Affiche la marque alert(objetVoiture.annee); // Affiche l'année**

### 2. L'interaction avec l'HTML

Afin que l'on puisse interagir avec le HTML a été créé le DOM ou Document Object Model (modèle d'objet de document en français) où le document est en fait les balises **<html>** et leur contenu. Cela permet d'avoir chacun des éléments du HTML sous forme d'objet à partir desquels on peut récupérer des informations ou les changer car toute modification faite sur ces objets aura un impact directement sur l'HTML.

Mais avant de pouvoir utiliser ces objets il va falloir d'abord les sélectionner l'élément dont on a besoin. On utilisera alors la fonction **document.querySelector** qui prend en argument une chaîne de caractères sous forme d'un sélecteur CSS et cette fonction va nous renvoyer l'élément si trouvé ou alors la valeur **null** pour indiquer qu'aucun élément n'a été trouvé.

Exemple sur la page d'accueil de **google.fr** :

**let elem = document.querySelector("#hplogo");**

**// Est enregistré dans la variable elem la valeur suivante (tronquée car trop longue)**

**<img id="hplogo" alt="Google" ... width="272" height="92">**

Cela ne ressemble pas trop aux objets évoqués plus haut mais il s'agit bien d'un objet bien que d'un type spécial qui permet d'avoir une représentation comme s'il s'agissait de HTML directement. L'important est que cela s'utilisera comme un objet et c'est là le plus important.

On peut voir toutes les clefs/valeurs que l'objet contient en l'affichant dans la console en tapant son nom puis en le déroulant. Il est conseillé que vous le fassiez sur une votre page de votre choix avant de continuer car la liste de ce que contient cet objet est impressionnante et trop longue pour être affichée ici.

Il s'agit en fait de de chacun des attributs possibles de l'élément mais également d'autre choses utiles tel que le texte à l'intérieur, les éléments enfants, les évènements, etc.

Pour récupérer une valeur de cet objet on écrira, comme avec tous les objets, le nom de l'objet suivi d'un point et de la clef correspondant à la valeur. Par exemple imaginons qu'un élément HTML **<img id="monImg" src="https://media.giphy.com/media/LXONhtCmN32YU/source.gif">** soit présent et qu'on veuille récupérer la source de l'image, c'est à dire le contenu de son attribut **src**. On fera alors ce qui suit : **let img = document.querySelector("#monImg");**

**let imgSrc = img.src; // La variable imgSrc contient dorénavant la valeur de l'attribut src de l'élément sélectionné**

Pour modifier une valeur on utilisera la même méthode mais en utilisant le signe égal "**=**" pour changer le contenu de la clef/variable choisie. L'exemple ci-dessus repris afin de modifier la source d'une image :

**// Dans l'exemple ci-dessous google utilise l'attribut srcSet à la place de src pour avoir plusieurs images de qualités différentes disponibles let img = document.querySelector("#hplogo");**

**img.srcset = "https://i.ytimg.com/vi/4zoLyXSno4s/hqdefault.jpg"; // L'image actuellement affiché sur la page sera changée afin de correspondre à la celle de la nouvelle source**

Dans le cas où on veut sélectionner plusieurs éléments d'un coup on pourra utiliser la function **document.querySelectorAll** qui fonctionne exactement de la même façon que **document.querySelector** mais qui retourne à la place quelque chose qui se comporte de façon générale comme un array contenant chacun des éléments correspondants au sélecteur fourni. Si on prend une situation où on aurait plusieurs images qui possède une même classe **galerie** on pourrait alors tous les sélectionner avant d'accomplir ce dont on a besoin :

**let imgListe = document.querySelectorAll(".galerie");**

**// imgListe.length (length signifie "longueur" en français) renvoie la longueur de imgListe afin de savoir combien de fois on doit répéter une opération for (let compteur = 0; compteur < imgList.length; compteur++) {**

## **imgListe[compteur].width = "480"; // Récupère l'élément à la**

**position de la valeur actuelle de compteur dans imgListe et change sa largeur (width est "largeur" en français) en une valeur de 480 pixels}**

Il existe également d'autres possibilités d'interagir avec du HTML mais qu'on croise moins souvent et il faudra choisir la fonction appropriée par rapport au but recherché.

Une fonction intéressante à connaître est **element.insertAdjacentHTML** qui permet de créer un nouvel élément HTML par rapport à un élément déjà existant sélectionné et qui prend deux arguments sous forme de chaîne de caractères, d'abord une position et ensuite un élément sous forme HTML. Le **element** dans cette fonction doit être remplacé par un élément sélectionné.

Les différentes positions possibles sont :

**'beforebegin' :** Avant l'élément lui-même.

**'afterbegin' :** Juste à l'intérieur de l’élément, avant son premier enfant.

**'beforeend' :** Juste à l'intérieur de l’élément, après son dernier enfant.

**'afterend' :** Après élément lui-même.

**Exemple avec <div id="maDiv"></div>: let divContainer = document.querySelector("#maDiv");**

**// Insère comme premier enfant de la div un paragraphe contenant du texte divContainer.insertAdjacentHTML("afterbegin", "<p>HELLO!</p>");**

### 3. L'interaction avec le CSS

L'interaction entre le CSS et le JavaScript se fera plus rarement directement vers le CSS mais on passera le plus souvent par le HTML avec l'attribut **style** qui nous permet de changer facilement le style d'un élément.

Attention de se rappeler que cet attribut est du CSS *Inline* (“en ligne” en français) ce qui signifie que le CSS directement sur un élément a une précédence sur les autres types de CSS.

Comme dans la partie précédente on récupère l'élément souhaité puis on modifie la valeur du style en mettant un point et en notant la variable **style** suivi d'un égal de d'une chaîne de caractères représentant le CSS.

**Exemple avec un <p id=”texteRouge”>Bonjour</p> :**

**let monParagraphe = document.querySelector(“#texteRouge”); monParagraphe.style = “color: red;”**

On peut aussi utiliser directement les feuilles de style et les modifier mais cette méthode est plus compliquée et moins pratique. Pour récupérer la liste des feuilles de style on fera appel à **document.styleSheets** mais cette méthode ne sera pas expliquée ici.

Dans le cas où votre besoin est de jouer avec les animations CSS avec l'aide du JavaScript (par exemple pour accès à l'ensemble des capacités liées à la programmation disponibles) il existe également un ensemble de fonctions qui permettent d'interagir de façon indirecte avec le CSS et de jouer et définir des animations directement depuis JavaScript.

Il s'agit de plusieurs fonctions et objets mais de notamment **element.animate** qui joue une animation sur un élément en particulier. Cette fonction prend deux arguments qui sont d'abord un ensemble d'images-clefs puis les options pour celles-ci.

**Exemple sur google.fr qui permet de faire tourner le logo :**

**// Sélectionne le tag <img> qui affiche le logo let googleLogo = document.querySelector("#hplogo");**

**// Anime le logo en fournissant un ensemble d'image-clefs (l'ensemble est un array et chaque image-clef est contenue dans un objet), et des options qui sont dans ce cas simplement une durée en millisecondes**

**googleLogo.animate( [{transform: "rotate(0deg)"}, {transform: "rotate(360deg)"}] , 2000);**

### 4. Les évènements

Les évènements sont un moyen pour le navigateur de dire ce qu'il se passe sur la page (notamment ce que l'utilisateur fait : cliquer, bouger son curseur à certains endroits, etc.) mais la partie intéressante est qu'on peut utiliser ces évènements dans le Javascript pour accomplir des actions selon l'évènement, par exemple réagir à un clic de souris ou enregistrer des informations dans une base de données etc.

Il existe deux façons d'utiliser les évènements en Javascript :

La première est d'utiliser directement comme vu précédemment le HTML pour utiliser les attributs liés aux types d'évènements (**onclick**, **onmouseover**, **onmouseout** et bien d'autres).

Option 1, on écrit directement le code dans l'attribut qui porte le nom de l'évènement **dans** le HTML.

**Exemple :**

**<h1 onclick="alert('Bonjour !');">Je dis bonjour si tu me cliques</h1>**

Option 2, on récupère l'élément dans du JavaScript en dehors du HTML et on lui modifie son attribut qui porte le bon nom d'évènement. Mais attention, il y a une différence entre écrire directement dans du HTML et le modifier à partir du JavaScript dans cas ! Ici il va falloir attribuer à cet attribut une **fonction** qui contiendra tous les opérations que vous voulez effectuez.

**Exemple avec le logo google.fr : let googleLogo = document.querySelector("#hplogo"); googleLogo.onclick = function onclick() { alert("C'est le logo Google");**

**} // L'attribut onclick est donc une fonction qui est appelée chaque fois qu'on clique sur le logo**

Un autre point à remarquer est que modifier l'attribut onclick de cette façon ne met pas à jour le HTML et que s’il est important d'avoir le HTML à jour on préférera utiliser la fonction **setAttribute** sur l'élément qui s'utilisera alors comme ceci :

**let googleLogo = document.querySelector("#hplogo");**

**googleLogo.setAttribute('onclick', `alert("C'est le logo Google")`); // Cette fois le HTML sera modifié pour être à jour**

Mais cette première méthode ne permet d'attribuer qu'une seule fonction par évènement. Bien qu'on pourrait modifier à nouveau la fonction à chaque nouvelle chose à faire cela reste au final plutôt difficile d'enlever une partie des actions à effectuer.

C'est ici qu'interviennent les fonction **addEventListener & removeEventListener** qui vont prendre un type d'évènement et une fonction en arguments. On appellera ces fonctions sur un élément sélectionné ce qui le fera réagir. Une différence importante est que lorsqu'on fournira le type d'évènements on omettra le "on" qui se trouve devant ("onclick" deviendra "click").

On va pouvoir appeler plusieurs fois l’événement

On va pouvoir modifier plus facilement

On va pouvoir enlever des évènements facilement

**Exemple toujours sur le logo de google.fr**

**function direBonjour() { alert("Bonjour");**

**} function direAuRevoir() { alert("Au revoir");**

**}**

**let googleLogo = document.querySelector("#hplogo"); // Lancera la fonction direBonjour quand le logo est cliqué googleLogo.addEventListener('click', direBonjour); // Lancera la fonction direAuRevoir quand le logo est cliqué googleLogo.addEventListener('click', direAuRevoir);**

**// Après avoir expérimenter on pourra enlever une fonction et retester le résultat googleLogo.removeEventListener('click', direBonjour);**

### 5. Stockage simple

Parfois il est nécessaire d'enregistrer des informations pour les réutiliser. Il existe un moyen très facile de stocker des informations simples dans le navigateur en utilisant **sessionStorage & localStorage**.

Le premier sert à enregistrer des informations pour la session (jusqu'à ce qu'on ferme le navigateur) et le second les enregistre jusqu'au moment où on se décide à supprimer les informations nous-mêmes.

Pour avoir les bases en main il est seulement nécessaire de connaître 3 fonctions: **setItem** (set signifie fixer/installer et item veut dire objet/article) qui prend d'abord un nom de clef et puis la valeur à stocker (on peut aussi utiliser cette technique pour mettre à jour la valeur d'une clef), **getItem** (get signifie obtenir) qui prend un nom de clef et renvoie la valeur qui correspond, et **removeItem** (remove signifie supprimer) qui prend également un nom de clef et supprime cette clef et sa valeur. On fera bien entendu précéder cette fonction du type de stockage à utiliser.

**Exemple :**

**// On demande à enregistrer la donnée "Jean" sous la clef "nom" sessionStorage.setItem("nom", "Jean");**

**// On récupère la valeur de la clef "nom" sessionStorage.getItem("nom");**

**// On supprime la clef "nom" et sa valeur sessionStorage.removeItem("nom");**

**// On essaie de récupérer la valeur de la clef "nom" mais la clef n'existe pas sessionStorage.getItem("nom");**

### 6. Exemple

Quoi de mieux qu'une mise en pratique pour comprendre ? A partir d'un site faites les opérations suivantes :

1. Permettez à l'utilisateur de choisir 2 couleurs qui seront utilisées comme couleur de texte & d'arrière-plan respectivement. Ces couleurs doivent être persistantes à travers les pages.
2. Rendez les titres cliquables ce qui fera défiler la fenêtre jusqu'à ce que ces titres se trouvent en haut de la partie visible de la fenêtre.
3. Créez un moyen de naviguer entre des sections d'une page comme le point 2 mais cette fois en utilisant un scroll de souris comme évènement déclencheur