

Chap 07 : Le Javascript

Table des matières

1. Qu'est-ce que JavaScript et à quoi sert-il ?	1
2. Comment utiliser JavaScript avec HTML et CSS ?	2
2.1. Cas général :	2
2.2. Fichier js externalisé	2
3. La syntaxe	3
4. Boite de dialogue	3
5. Les variables	3
5.1. Déclarer une variable	3
5.2. Les types de variables	4
5.3. Les chaines de caractères	4
5.4. Tester l'existence de variables avec <code>typeof</code>	4
5.5. Les calculs	4
5.6. La concaténation	5
5.7. Interagir avec l'utilisateur	5
6. Les conditions	5
6.1. Les opérateur de condition	5
6.2. Les structures conditionnelles	6
6.2.1. La condition « <code>if else</code> »	6
6.2.2. La condition <code>switch</code>	6
7. Les opérateurs logiques	7
8. Les boucles	7
8.1. L'incrément	7
8.2. la boucle <code>while</code>	8
8.3. la boucle <code>do while</code>	8
8.4. La boucle <code>for</code>	8
9. Les fonctions	8
10. Modeler des pages web avec <code>js</code>	9
10.1. Manipuler les éléments HTML	9
10.2. Manipuler les éléments HTML en utilisant une sélection CSS	11
10.3. Modification de contenu de la page HTML avec la propriété <code>innerHTML</code>	12
10.4. Modification de style de la page HTML avec la propriété <code>style</code>	13
10.5. Autres manipulations d'éléments HTML	14
11. Interactions avec l'utilisateur	15
11.1. Liste des événements en JS:	15
11.2. La pratique	15
11.3. Pour aller plus loin avec <code>addEventListener</code>	17
12. Exercices	19

1. Qu'est-ce que JavaScript et à quoi sert-il ?

JavaScript est un langage de programmation de scripts côté client. Il est utilisé pour créer des pages web **interactives** et **dynamiques** en ajoutant des fonctionnalités telles que les **animations**, les **formulaires interactifs**, les **scripts de validation**, les **galeries d'images**, etc. sans avoir besoin de charger une nouvelle page du serveur.

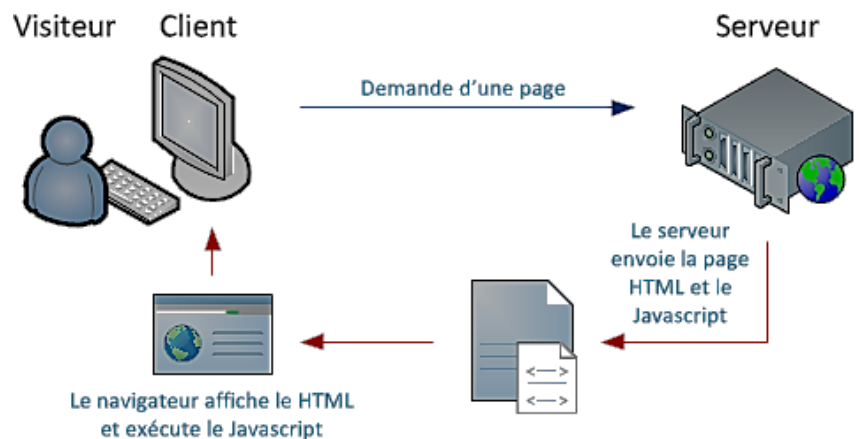
JavaScript peut être utilisé conjointement avec HTML et CSS pour créer des pages web complètes et riches en fonctionnalités. Il fonctionne dans les navigateurs web modernes sur les ordinateurs de bureau et les appareils mobiles, ce qui en fait un outil de développement web très puissant et largement utilisé.



Le Javascript est un langage dit **client-side** c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute.

Pour cela :

- votre ordinateur récupère le code source d'une page web sur un serveur.
- votre navigateur interprète la page et les scripts qu'elle contient.
- la page formatée s'affiche sur votre écran. Les scripts, quant à eux, sont mis en mémoire et seront lancés dès que l'événement attendu se produira



2. Comment utiliser JavaScript avec HTML et CSS ?

2.1. Cas général :

En intégrant le code JavaScript directement dans la page HTML, à l'intérieur de la balise `<script>` et `</script>`

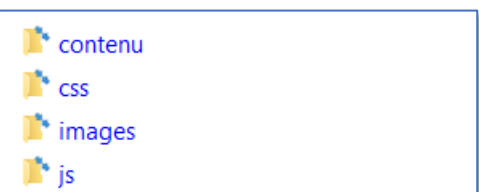
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      /* Votre feuille de style CSS */
    </style>
  </head>
  <body>
    <!-- Votre contenu HTML -->
    <script>
      // Votre code JavaScript
    </script>
  </body>
</html>
```

On peut les placer **soit dans l'en-tête** (`<head> ... </head>;`), **soit dans le corps** (`<body> ... </body>;`) de la page HTML.

2.2. Fichier js externalisé

Pour bien faire on externalise dans un dossier `js`.

On écrit le javascript dans un fichier `script.js` que l'on mettra aussi dans le dossier `js`.

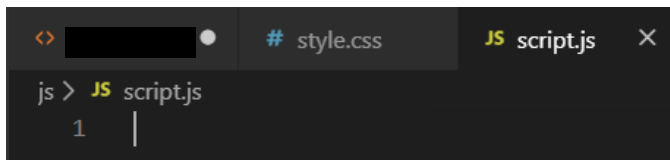


Activité n°1 : Dans la `index.html` rajouter le lien vers le fichier `js`

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="css/style.css" />
  <script type="text/javascript" src="js/script.js"></script>
  <title>Logique sur les passoires</title>
</head>
```

`type="text/javascript"` est facultatif

Créer un nouveau fichier vide avec l'éditeur. Enregistrer-le dans le dossier `js` sous le nom **`script.js`**



3. La syntaxe

- Les instructions sont séparées par des **points-virgules**
- Les commentaires se placent entre `/*` et `*/`
- Les scripts peuvent s'intégrer directement dans la page dans des balise `<script>` ou peuvent être dans un fichier `.js` par exemple : `<script src="js/hello.js"></script>`

4. Boîte de dialogue

Le fichier javascript est appelé depuis la page Web au moyen de l'élément `<script>` et `src` avec l'URL du fichier `.js`. La fonction `alert()` existe déjà dans javascript donc on peut directement l'écrire dans la page Web.

Activité n°2.: Dans la `index.html` rajouter n'importe où dans les balises `<body>`

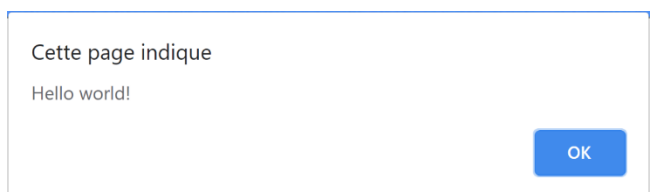
```
<script>
</script>
```

Enregistrer et observer la `index.html` dans Firefox.
Externalisons le code :

Activité n°3.: Dans le fichier `script.js` écrire le script suivant :

```
alert('Hello world!');
```

Enregistrer le tout et observer la `index.html` dans Firefox
La suite du cours sur le javascript se fera sur des fichiers extérieurs.



Activité n°4.: Dans une nouvelle page html que l'on appellera `exo_JS.html` rajouter le lien vers le fichier `js`. Le fichier sera enregistré dans le dossier `Documents\site`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <script type="text/javascript" src="js/exo.js"></script>
    <title>Page de tests du code js</title>
  </head>
  <body>
    Page de tests du code JS
  </body>
</html>
```

Créer un nouveau fichier vide avec l'éditeur. Enregistrer-le dans le dossier `js` sous le nom `exo.js`

5. Les variables

5.1. Déclarer une variable

`let` permet de déclarer des variables dont la **portée est limitée** à celle du bloc dans lequel elles sont déclarées. Le mot-clé `var`, quant à lui, permet de définir une **variable globale ou locale** à une fonction (sans distinction des blocs utilisés dans la fonction).

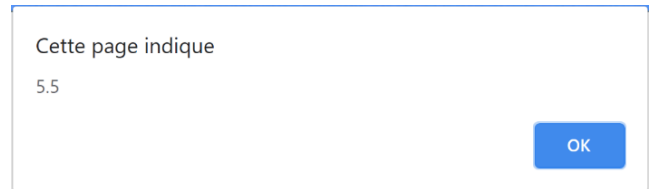
La déclaration d'une variable se fait avec le mot clé `var` ou `const` (mais elle a besoin d'une valeur initiale):

- ```
var maVariable ;
 maVariable = 5 ;
```

- ou `var message = "Bonjour, visiteur";`

**Activité n°5.:** Dans le fichier `exo.js` écrire le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var myVariable = 5.5;
alert(myVariable);
```



## 5.2. Les types de variables

Le Javascript est un langage typé dynamiquement : on n'a pas besoin de déclarer le type des variables. Il existe trois types principaux de variable :

- Les **numbers**
- Les **strings**
- Les **booleans**

On utilisera tout le long la fonction `alert()` qui permet d'ouvrir un popup avec le résultat on aurait pu aussi utiliser la fonction `console.log()` qui permet d'obtenir le résultat dans la console, sur Firefox Ctrl+Maj+I

## 5.3. Les chaînes de caractères

Pour inclure des guillemets " ou des apostrophes dans une chaînes, il faut utiliser le caractère d'échappement \ (antislash).

Exemple :

```
// deux chaînes de caractères
var message1 = "Ceci est un \"petit\" test (pas besoin d'antislash \).";
var message2 = 'Un autre "petit" test (attention à l\'antislash)';

// maintenant, on les affiche
alert(message1);
alert(message2);
```

On peut également insérer des retours à la ligne ainsi que des tabulations avec des caractères spéciaux :

- `\n` : retour à la ligne.
- `\t` : une tabulation (ne marche pas dans tous les cas)
- `\b` : pour insérer un backspace (touche "retour arrière")
- `\uXXXX` : pour insérer le caractère donc la valeur unicode est XXXX (en hexadécimales).

## 5.4. Tester l'existence de variables avec `typeof`

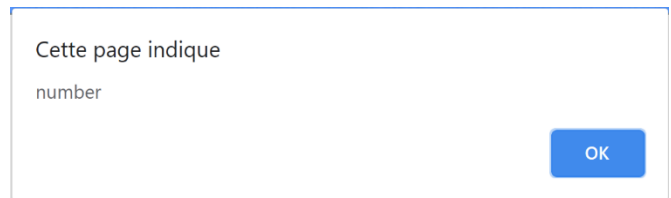
L'instruction `typeof` permet de tester l'existence d'une variable ou d'en vérifier son type. Par exemple :

**Activité n°6.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var number = 2;
alert(typeof number);

var text = 'Mon texte';
alert(typeof text);

var aBoolean = false;
alert(typeof aBoolean);
```



## 5.5. Les calculs

Tous les opérateurs classiques peuvent être utilisés. Ainsi on pourra écrire

**Activité n°7.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var divisor = 3, result1, result2, result3;
result1 = (16 + 8) / 2 - 2 ;
result2 = result1 / divisor;
result3 = result1 % divisor;

alert(result2);
alert(result3);
```

Cette page indique  
3.3333333333333335

OK

## 5.6. La concaténation

**Activité n°8.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var hi = 'Bonjour ', name = 'toi', result;
result = hi + name;
alert(result);
```

Cette page indique  
Bonjour toi

OK

On peut aussi concaténer des chaînes de caractères et un nombre.

## 5.7. Interagir avec l'utilisateur

Avec la fonction `prompt()`. Elle s'utilise comme `alert()`. Elle renvoie ce que l'utilisateur a écrit sous forme d'une chaîne de caractère.

**Activité n°9.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var userName = prompt('Entrez votre prénom :');
alert(userName);
```

Cette page indique  
Entrez votre prénom :

OK

Annuler

On peut également dire bonjour à nos visiteurs

**Activité n°10.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var start = 'Bonjour ', name, end = ' !', result;

name = prompt('Quel est votre prénom ?');
result = start + name + end;
alert(result);
```

Tout ce qui est récupéré avec `prompt()` est sous forme d'une chaîne de caractères. Pour convertir la chaîne de caractères en **nombre entier** on utilise la fonction `parseInt()`. On pourra utiliser la fonction `parseFloat()` pour convertir une chaîne de caractère en nombre décimal. Par exemple

**Activité n°11.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var first, second, result;
first = prompt('Entrez le premier chiffre :');
second = prompt('Entrez le second chiffre :');
result = parseInt(first) + parseInt(second);
alert(result);
```

## 6. Les conditions

### 6.1. Les opérateurs de condition

Les opérations de comparaison classiques sont les mêmes : `==` ; `!=` ; `<` ; `<=` etc

Pour pouvoir comparer 4 en tant que `number` et 4 en tant que `string` il faut utiliser d'autres opérateurs :

**Activité n°12.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var number = 4, text = '4', result;

result = number == text;
alert(result); // Affiche « true » alors que « number » est un nombre et « text » une chaîne de caractères

result = number === text;
alert(result); // Affiche « false » car cet opérateur compare aussi les types des variables en plus de leurs valeurs
```

## 6.2. Les structures conditionnelles

### 6.2.1. La condition « `if else` »

**Activité n°13.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var userName = prompt('Entrez votre prénom :');
if (2 < 8 && 8 >= 4) { // Cette condition renvoie « true », le code est donc exécuté
 alert('La condition est bien vérifiée.');
```

La fonction `confirm()`. On lui passe en paramètre une chaîne de caractère qui sera affichée à l'écran et elle retourne un booléen en fonction de l'action de l'utilisateur.

**Activité n°14.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
if (confirm('Voulez-vous exécuter le code JavaScript de cette page ?')) {
 alert('Le code a bien été exécuté !');
```

Relancer la Web `exo_JS.html` en choisissant l'autre proposition.

La structure `else` existe également et permet d'exécuter un certain code si la condition n'a pas été vérifiée. Par contre il est conseillé de l'écrire ainsi (directement après l'accolade de fermeture de la structure `if`) :

```
if (/* condition */) {
 // Du code...
} else {
 // Du code...
}
```

La structure `else if` peut être aussi utilisé ainsi

```
if (/* condition */) {
 // Du code...
} else if (/* condition */) {
 // Du code...
} else {
 // Du code...
}
```

### 6.2.2. La condition `switch`

Supposons qu'on est besoin de nombreux `else if` les uns à la suite des autres. Par exemple :

**Activité n°15.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var tiroir = parseInt(prompt('Choisissez le tiroir à ouvrir (1 à 4) :'));
if (tiroir == 1) {
 alert('Contient divers outils pour dessiner : du papier, des crayons, etc.');
```

```

} else if (tiroir == 3) {
 alert('Ah ? Ce tiroir est fermé à clé ! Dommage !');
} else if (tiroir == 4) {
 alert('Contient des vêtements : des chemises, des pantalons, etc.');
```

Avec `switch` c'est un peu plus facile :

**Activité n°16.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```

var tiroir = parseInt(prompt('Choisissez le tiroir à ouvrir (1 à 4) :'));

switch (tiroir) {
 case 1:
 alert('Contient divers outils pour dessiner : du papier, des crayons, etc.');
```

Tout ce qui suit les deux points d'un `case` sera exécuté si la variable analysée par le `switch` contient la valeur du `case`. A chaque fin d'un `case` on écrit l'instruction `break` pour casser le `switch` et éviter d'afficher les autres alertes. La partie `default` est optionnel.

## 7. Les opérateurs logiques

- L'opérateur ET se note `&&`
- L'opérateur OU se note `||` (Alt Gr +6)
- L'opérateur NON se note comme en Python avec `!`

## 8. Les boucles

### 8.1. L'incrémentement

L'incrémentement permet d'ajouter une unité à un nombre et à l'inverse, la décrémentation permet de soustraire une unité.

**Activité n°17.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```

var number1 = 0, number2 = 0;

number1++;
alert(number1);
```

On utilisera tout le long la fonction `alert()` qui permet d'ouvrir un popup avec le résultat on aurait pu aussi utiliser la fonction `console.log()` qui permet d'obtenir le résultat dans la console, sur Firefox Ctrl+Maj+I

```
number2--;
alert(number2) ;
```

La position de l'opérateur ++ est important si on veut récupérer le résultat de l'incrémement :

var number = 0; Il y a deux possibilités :

- var output = ++number; → retournera 1 car retourne la valeur de number incrémentée
- var output = number++; → retournera 0 car retourne la valeur de number avant incrémement

## 8.2. la boucle while

A chaque fois que la boucle se répète on parle d'itération. Par exemple :

**Activité n°18.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire // devant chaque ligne ou /\* et \*/ et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
var number = 1;

while (number < 10) {
 number++;
}
alert(number);
```

## 8.3. la boucle do while

La boucle do while ressemble très fortement à la boucle while, sauf que dans ce cas la boucle est toujours exécutée au moins une fois. La syntaxe d'une boucle do while:

```
do {
 instruction_1;
 instruction_2;
 instruction_3;
} while (condition);
```

## 8.4. La boucle for

Le schéma d'une boucle for :

```
for (initialisation; condition; incrémement) {
 instruction_1;
 instruction_2;
 instruction_3;
}
```

**Activité n°19.:** Dans le fichier `exo.js` passer les lignes précédentes en commentaire // devant chaque ligne ou /\* et \*/ et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
for (var iter = 0; iter < 5; iter++) {
 alert('Itération n°' + iter);
}
```

Attention les variables utilisées dans la boucle while ou dans la boucle for ne sont pas détruites (comme dans Python) une fois sortie de la boucle.

## 9. Les fonctions

Pour définir une fonction il faut le script suivant :

```
function myFunction(arguments) {
 // Le code que la fonction va devoir exécuter
}
```

**Activité n°20.:** Exemple de fonction sans argument : Dans le fichier `exo.js` passer les lignes précédentes en commentaire // devant chaque ligne ou /\* et \*/ et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
function showMsg() {
```



```
 alert('Et une première fonction, une !');
}
showMsg(); // On exécute ici le code contenu dans la fonction
```

La fonction `showMsg()` exécute elle-même une autre fonction qui n'est autre que `alert()` avec un message prédéfini. Bien sûr, tout code écrit dans une fonction ne s'exécute pas immédiatement, sinon aucun intérêt. C'est pourquoi à la **fin du code on appelle la fonction** afin de l'exécuter, ce qui affiche le message souhaité.

Toute variable déclarée dans une fonction n'est utilisable que dans cette même fonction. Ce sont les variables locales.

**Activité n°21.: Exemple de fonction avec argument :** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
function myFunction(arg) { // Notre argument est la variable « arg »
 // Une fois que l'argument a été passé à la fonction, vous allez le retrouver dans la variable « arg »
 alert('Votre argument : ' + arg);
}
myFunction('En voilà un beau test !');
```

**Activité n°22.: Exemple de fonction avec `prompt()` :** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
function myFunction(arg) {
 alert('Votre argument : ' + arg);
}
myFunction(prompt('Que souhaitez-vous passer en argument à la fonction ?'));
```

**Activité n°23.: Exemple de fonction avec un `return` :** Dans le fichier `exo.js` passer les lignes précédentes en commentaire `//` devant chaque ligne ou `/*` et `*/` et rajouter le script suivant. Enregistrer et observer le fichier `exo_JS.html` dans Firefox.

```
function sayHello() {
 return 'Bonjour !';
 alert('Attention ! Le texte arrive !');
}
alert(sayHello());
```

## 10. Modeler des pages web avec js

Le Javascript va modifier les pages html et css en accédant aux éléments cible du DOM (Document Objet Model).

### 10.1. Manipuler les éléments HTML

- **Sélection des éléments par ID** : pour sélectionner un élément HTML en utilisant son ID, nous pouvons utiliser la méthode `getElementById()`

```
<body>
 <p id = "titre"> je représente le titre qui va être modifié car j'ai le bon id</p>
 <script>
 var titre = document.getElementById("titre");
 titre.style.color = "blue";
 </script>
</body>
```

- **Sélection des éléments par nom de classe** : pour sélectionner plusieurs éléments HTML qui ont la même classe, nous pouvons utiliser la méthode `getElementsByClassName()`. Par exemple :

```

<body>
 <p class = "paragraphe"> je représente </br>le paragraphe </br> qui va être
modifié </br> car j'ai la bonne classe</p>
 <script>
 var paragraphes = document.getElementsByClassName("paragraphe");
 for (var i = 0; i < paragraphes.length; i++) {
 paragraphes[i].style.backgroundColor = "yellow";
 }
 </script>
</body>

```

Trouve tous les éléments ayant la classe « paragraphe »

**Activité n°24.:** Créer un fichier `interaction.html`, dans site et saisir le code ci-dessous.

```

<!DOCTYPE html>
<head>
 <meta charset="UTF-8">
 <title>Interaction html/css et JS</title>
</head>
<body>
 <h1>Voici un joli titre.</h1>
 <p id='important'>Ceci est un texte qu'il faut mettre en valeur !</p>
 <button onclick="change_couleur()">Cliquez ici !</button>
</body>
<script src="js/interaction.js"></script>
</html>

```

## Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

Cliquez ici !

Dans le code précédent, `<button onclick="change_couleur()">Cliquez ici !</button>` est une balise qui va créer un bouton.

Un click dessus va déclencher la fonction `change_couleur()`

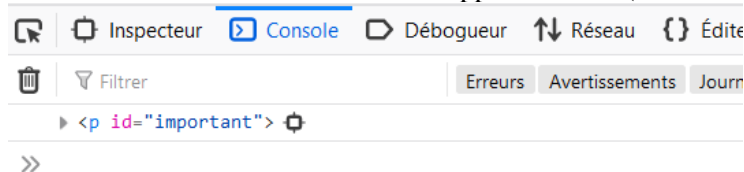
Cette fonction va être définie dans un fichier javascript (`interaction.js`) qui va être créé par ce qui suit ... Ensuite créez un fichier `interaction.js` dans le sous-dossier `js`. Saisir le code suivant. Puis enregistrer et observer le fichier `interaction.html` dans Firefox.

```

function change_couleur() {
 var paragraphe = document.getElementById("important");
 console.log(paragraphe);
}

```

Ouvrez la console des outils de développement web (CTRL Maj I). Cliquer sur le bouton et observer dans console :



Vous devez constater que la variable `paragraphe` contient maintenant la balise `paragraphe` et son contenu. Dans la variable `paragraphe`, l'élément d'ID (identifiant) `"important"` est stocké.

**Activité n°25.:** Ajouter le code ci-dessous à la fonction du fichier `js`. Puis enregistrer et observer le fichier `interaction.html` dans Firefox.

```

paragraphe.style.color="red";

```

Appuyez sur le bouton, et constatez le changement. Habituellement on utilise le CSS pour la mise en page.

## Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

Cliquez ici !



**Activité n°26.:** Créer un fichier `interaction.css` dans le dossier `css`. Rajouter le code suivant à la page CSS

```
h1{
 text-align: center;
 font-size: 40px;
}
.rouge {
 color:red;
 font-size:30px;
}
```

# Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

Cliquez ici !

Puis ajouter la ligne suivante dans le html pour lier le CSS avec le html.

```
<link rel="stylesheet" href="css/interaction.css">
```

Modifier le fichier `Js` comme suit. Puis enregistrer et observer le fichier `interaction.html` dans Firefox.

```
function change_couleur() {
 var paragraphe = document.getElementById("important");
 console.log(paragraphe);
 paragraphe.classList.add("rouge");
}
```

# Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

Ce code aura pour effet d'ajouter la classe "rouge" à notre élément "paragraphe" sélectionné.

On va à présent modifier la couleur

Cliquez ici !

**Activité n°27.:** Ajouter le bouton suivant dans le html

```
<button onclick="reset_couleur()">Reset</button>
```

Puis ajouter la fonction suivante dans le JS :

```
function reset_couleur() {
 var paragraphe = document.getElementById("important");
 console.log(paragraphe);
 paragraphe.classList.remove("rouge");
}
```

# Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

# Voici un joli titre.

Ceci est un texte qu'il faut mettre en valeur !

Cliquez ici !

Reset

Cliquez ici !

Reset

## 10.2. Manipuler les éléments HTML en utilisant une sélection CSS

Sélection des éléments avec `querySelector` et `querySelectorAll` : pour sélectionner des éléments HTML en utilisant une sélection CSS, nous pouvons utiliser les méthodes `document.querySelector()` et `document.querySelectorAll()`.

**Activité n°28.:**

```
<body>
```

```

<p class="paragraphe">je représente </br> un paragraphe</p>
<p class="paragraphe">je représente </br> un autre paragraphe</p>
<p class="paragraphe">je représente </br> un dernier paragraphe</p>
<script>
 var premierParagraphe = document.querySelector(".paragraphe");
 premierParagraphe.style.fontWeight = "bold";

 var tousLesParagraphes = document.querySelectorAll(".paragraphe");
 for (var i = 0; i < tousLesParagraphes.length; i++) {
 tousLesParagraphes[i].style.textAlign = "center";
 }
</script>
</body>

```

Une fois que nous avons sélectionné un ou plusieurs éléments HTML, nous pouvons les manipuler en utilisant leurs propriétés et méthodes.

### 10.3. Modification de contenu de la page HTML avec la propriété `innerHTML`

**Activité n°29.:** Modification du contenu d'un élément :

```

<body>
<h1 id="titre">je suis un titre avec un id</h1>
<script>
 var titre = document.getElementById("titre");
 titre.innerHTML = "Nouveau titre";
</script>
</body>

```

**Activité n°30.:** Ajout de contenu à la fin d'un élément :

```

<body>
<p id="paragraphe">je suis un paragraphe avec un id</p>
<script>
 var paragraphe = document.getElementById("paragraphe");
 paragraphe.innerHTML += "
Nouveau contenu";
</script>
</body>

```

**Activité n°31.:** Modification du contenu de plusieurs éléments avec une boucle :

```

<body>

<script>
 var listeElements = document.getElementsByTagName("li");
 for (var i = 0; i < listeElements.length; i++) {
 listeElements[i].innerHTML = "Element " + (i + 1);
 }
</script>
</body>

```

On cible chaque élément de la liste et on le remplace par « Element » et le numéro

**Activité n°32.:** Modification du contenu en HTML en utilisant des balises HTML :

```

<body>
<p id="paragraphe">ceci est un paragraphe</p>
<script>

```

```

 var paragraphe = document.getElementById("paragraphe");
 paragraphe.innerHTML = "Contenu en gras et <i>italique</i>";
 </script>
</body>

```

#### 10.4. Modification de modification de style de la page HTML avec la propriété style

**Activité n°33.:** Modification de la couleur de fond d'un élément :

```

<body>
 <p id="element">ceci est un paragraphe qui a un id</p>
 <script>
 var element = document.getElementById("element");
 element.style.backgroundColor = "yellow";
 </script>
</body>

```

**Activité n°34.:** Modification de la couleur de police d'un élément

```

<body>
 <p id="element">ceci est un paragraphe qui a un id</p>
 <script>
 var element = document.getElementById("element");
 element.style.color = "blue";
 </script>
</body>

```

**Activité n°35.:** Modification de la largeur d'un élément :

```

<body>
 <p id="element">ceci est un paragraphe qui a un id</p>
 <script>
 var element = document.getElementById("element");
 element.style.width = "70px";
 </script>
</body>

```

**Activité n°36.:** Modification de plusieurs styles en même temps :

```

<body>
 <p id="element">ceci est un paragraphe qui a un id</p>
 <script>
 var element = document.getElementById("element");
 element.style.backgroundColor = "yellow";
 element.style.color = "blue";
 element.style.width = "70px";
 </script>
</body>

```

**Activité n°37.:** Modification de styles en utilisant une boucle :

```

<body>

 framboise
 fraise
 pomme

 <script>
 var listeElements = document.getElementsByTagName("li");
 for (var i = 0; i < listeElements.length; i++) {
 listeElements[i].style.backgroundColor = "yellow";
 listeElements[i].style.color = "blue";
 }
 </script>

```

```
</body>
```

## 10.5. Autres manipulations d'éléments HTML

**Activité n°38.:** Ajout ou suppression de classes CSS :

```
<body>
 <p id="element">je suis un paragraphe avec un id</p>
 <script>
 var element = document.getElementById("element");
 element.classList.add("nvlclasse");

 var premierParagraphe = document.querySelector(".nvlclasse");
 premierParagraphe.style.fontWeight = "bold";

 element.classList.remove("nvlclasse");
 </script>
</body>
```

- Modification de l'attribut src d'une image pour changer son URL :

```
<body>

 <script>
 var image = document.getElementById("image");
 image.src = "nouvelle_image.jpg";
 </script>
</body>
```

**Activité n°39.:** Modification de l'attribut href d'un lien pour changer son URL :

```
<body>
 c'est la page wikipedia
 <script>
 var lien = document.getElementById("lien");
 lien.href = "http://www.google.fr";
 </script>
</body>
```

**Activité n°40.:** Ajout d'un événement (par exemple, un clic) à un élément :

```
<body>
 <p id="bouton">c'est un paragraphe avec un id il faut cliquer dessus</p>
 <script>
 var bouton = document.getElementById("bouton");
 bouton.addEventListener("click", function() {
 alert("Bouton cliqué");
 });
 </script>
</body>
```

**Activité n°41.:** Modification de la position d'un élément en utilisant les propriétés left, right, top, et bottom :

```
<body>
 <p id="element">c'est un paragraphe avec un id</p>
 <script>
 var element = document.getElementById("element");
 element.style.position = "absolute";
 element.style.left = "100px";
 element.style.top = "100px";
 </script>
</body>
```

```
</script>
</body>
```

## 11. Interactions avec l'utilisateur

Les événements permettent de déclencher une fonction selon qu'une action s'est produite ou non. Par exemple, faire apparaître une fenêtre `alert()` lorsque l'utilisateur survole une zone d'une page web, ou ajouter un élément lors d'un clic de bouton de la souris (ou du clavier).

### 11.1. Liste des événements en JS:

Il est important de noter que les événements peuvent être déclenchés par des actions de l'utilisateur ou par des actions du script lui-même.

**Résumé des événements JS :**  
<https://www.lehtml.com/js/even.htm>

<code>click</code>	Cliquer (appuyer puis relâcher) sur l'élément
<code>dblclick</code>	Double-cliquer sur l'élément
<code>mouseover</code>	Faire entrer le curseur sur l'élément
<code>mouseout</code>	Faire sortir le curseur de l'élément
<code>mousedown</code>	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
<code>mouseup</code>	Relâcher le bouton gauche de la souris sur l'élément
<code>mousemove</code>	Faire déplacer le curseur sur l'élément
<code>keydown</code>	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
<code>keyup</code>	Relâcher une touche de clavier sur l'élément
<code>keypress</code>	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
<code>focus</code>	« Cibler » l'élément
<code>blur</code>	Annuler le « ciblage » de l'élément
<code>change</code>	Changer la valeur d'un élément spécifique aux formulaires ( <code>input</code> , <code>checkbox</code> , etc.)
<code>input</code>	<a href="https://caniuse.com/#feat=input-event">Taper un caractère dans un champ de texte (son support n'est pas compvar sur tous les navigateurs) (https://caniuse.com/#feat=input-event)</a>
<code>select</code>	Sélectionner le contenu d'un champ de texte ( <code>input</code> , <code>textarea</code> , etc.)
<code>submit</code>	déclenché lorsque le formulaire est soumis.
<code>resize</code>	Resize de la fenêtre
<code>scroll</code>	Scroll de la fenêtre

### 11.2. La pratique

**Activité n°42.:** Créer un fichier (avec tout ce qu'il faut 😊) `evenement.html`, dans `Documents\site` et saisir le code ci-dessous.

```
<body>
 Cliquez ici !
</body>
```

Enregistrer et observer la page html dans Firefox.  
Ici la fonction `alert()` ne se déclenche que lors d'un click.

**Activité n°43.:** Afficher une alerte lorsque l'utilisateur clique sur un bouton :

```
<body>
 <button id="bouton">Cliquez ici !</button>
 <script>
 var bouton = document.getElementById("bouton");
 bouton.addEventListener("click", function() {
 alert("Bouton cliqué");
 });
 </script>
```

```
});
</script>
</body>
```

**Activité n°44.:** Changer la couleur de fond d'un élément lorsque la souris passe dessus

```
<body>
 <p id="maDiv">c'est un paragraphe avec un id</p>
 <script>
 var div = document.getElementById("maDiv");
 div.addEventListener("mouseover", function() {
 div.style.backgroundColor = "red";
 });
 </script>
</body>
```

**Activité n°45.:** Afficher le code d'une touche pressée lorsque l'utilisateur tape sur le clavier :

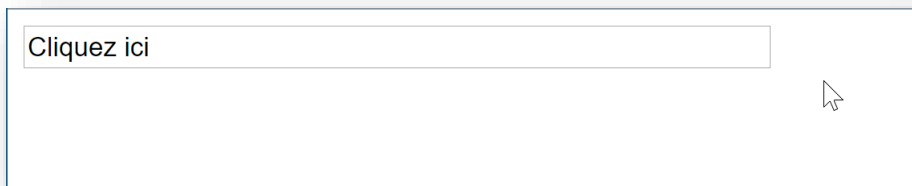
```
<body>
 <p>c'est un paragraphe </p>
 <script>
 document.addEventListener("keydown", function(event) {
 console.log("Touche pressée : " + event.keyCode);
 });
 </script>
</body>
```

Le mot-clé **this** : il s'agit d'une propriété pointant **sur l'objet actuellement en cours d'utilisation**. Donc, si vous faites appel à ce mot-clé lorsqu'un événement est déclenché, l'objet pointé sera l'élément qui a déclenché l'événement.

**Activité n°46.:**

```
<body>
 <input type="text" id="input" size="50" value="Cliquez ici"
 onfocus="this.value = 'Appuyez sur la tabulation pour perdre le focus'" onblur="
this.value= 'Cliquez ici !'">
</body>
```

Enregistrer et observer la page html dans Firefox.





Appuyez sur la tabulation pour perdre le focus

Le mot clé `this` permet de pointer sur l'objet qui a déclenché l'évènement.

- `onfocus` : se déclenchera si le "focus" est pris par la balise input.
- `onblur` : se déclenchera si le "focus" est perdu par la balise input.

On va séparer le code javascript du code html.

### 11.3. Pour aller plus loin avec `addEventListener`

**Activité n°47.:** Créer un fichier `plusloin.html`, dans *site* et saisir le code ci-dessous.

```
<!DOCTYPE html>
<html>
<body>
 <button id="clickIt">Cliquez ici !</button>
 <p id="hoverPara">Passez la souris sur ce texte !</p>
 <b id="effect">
 <script>
 const x = document.getElementById("clickIt");
 const y = document.getElementById("hoverPara");
 x.addEventListener("click", RespondClick);
 y.addEventListener("mouseover", RespondMouseOver);
 y.addEventListener("mouseout", RespondMouseOut);
 function RespondMouseOver() {
 document.getElementById("effect").innerHTML +=
 "MouseOver Event" + "
";
 }
 function RespondMouseOut() {
 document.getElementById("effect").innerHTML +=
 "MouseOut Event" + "
";
 }
 function RespondClick() {
 document.getElementById("effect").innerHTML +=
 "Click Event" + "
";
 }
 </script>
</body>
</html>
```

Enregistrer et observer la page html dans Firefox.

Cliquez ici !

Passez la souris sur ce texte !

**Click Event**

**Click Event**

**MouseOver Event**

**MouseOut Event**

**MouseOver Event**

Comme vous le voyez, il est possible d'ajouter plusieurs `EventListener` à un même élément html. Vous pouvez aussi associer une fonction à votre `eventListener` sans qu'elle soit intégrée dans les paramètres, cette fonction peut être définie par la suite (comme `RespondMouseOut`).

Pour aller plus loin ou avoir plus de détails : [https://www.w3schools.com/jsref/dom\\_obj\\_all.asp](https://www.w3schools.com/jsref/dom_obj_all.asp)

## 12. Exercices

Ecrire directement le script dans le html entre balise `<script>...</script>`. Les fichiers HTML `exercice_numéro.html` sont à compléter pour le dossier Ressources

### Exercice 1 : Ecrire une fonction simple

Vous utiliserez les fonctions `alert()` (qui affiche une chaîne de caractères) et `prompt()` (qui invite à la saisie d'une donnée), dont voici les syntaxes :

- `alert(Chaîne à afficher à l'écran)`
- `variable=prompt(Question à afficher à l'écran)`

1. Complétez le fichier `exercice1.html` pour que le click sur le bouton lance une fonction. Cette fonction demande la saisie d'une largeur, d'une longueur et affiche la surface du rectangle correspondant.
2. Remplacez ce calcul par celui du périmètre (double de la somme de la longueur et de la largeur).

### Exercice 2 : Ecrire une fonction renvoyant une valeur

1. Créez une fonction qui :
  - demande la saisie d'un rayon ;
  - retourne la surface du cercle de ce rayon
2. Affichez le résultat de l'appel à cette fonction en cliquant sur le bouton.

### Exercice 3 : Changements de couleurs

Les changements de couleurs d'une zone de texte font appel à `onmouseover`, `onmouseout`, `onclick` et `ondblclick`, ainsi qu'aux méthodes de contrôle du style par JavaScript appliquées à l'objet courant indiqué par `this`.

Exemple : Ceci est un texte dont la couleur va changer au passage de la souris.

Créer une page web comportant une phrase, dont un groupe de mots, de couleur noire au chargement, doit prendre la couleur :

1. rouge au passage de la souris ;
2. citron vert (lime) en réponse à un click ;
3. bleu marine (navy) en réponse à un double click.

### Exercice 4 : Manipulation des méthodes d'accès aux éléments

On a utilisé les méthodes `getElementById`, `getElementsByName` et `getElementsByTagName`.

1. Créer une fonction `modif_paragraphe()`, appelée en cliquant sur le titre. Cette fonction sélectionne le paragraphe en utilisant son identifiant, puis le modifie avec la propriété `innerHTML`, en remplaçant le mot **original** en caractères droit par le mot **corrigé**, en italique.
2. Créer une fonction `centrage_h1()`, appelée en cliquant sur le paragraphe. Cette fonction détecte d'abord les éléments portant le nom de balise `h1`. Elle sélectionne ensuite le premier d'entre eux et modifie son attribut `align`, en lui affectant la valeur `"center"`, à l'aide de la méthode `setAttribute....`

Aide :

- `titre.setAttribute("align", "center");`