Javascript
Le langage du Front-End

Principale source : Documentation : https://developer.mozilla.org/fr/docs/Learn

I. TABLE DES MATIÈRES

Documentations

J'ai choisi deux sites (parmi tant d'autres …)

- L'officiel Documentation : https://developer.mozilla.org/fr/docs/Web (en Français)
- https://www.w3schools.com/ (en Anglais)

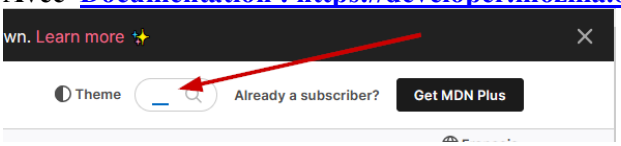
1. Liens

i. Voici les liens directs

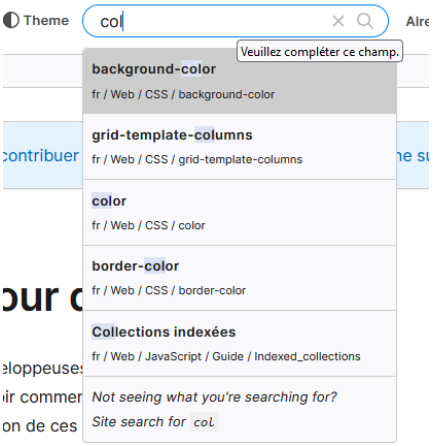
	Mozilla	w3school
HTML	Documentation : https://developer.mozilla.org/fr/docs/Web/HTML	https://www.w3schools.com/tags/default.asp
CSS	Documentation : https://developer.mozilla.org/fr/docs/Web/CSS	https://www.w3schools.com/cssref/default.asp
Javascript	Documentation : https://developer.mozilla.org/fr/docs/Web/JavaScript	https://www.w3schools.com/jsref/default.asp

ii. Comment rechercher une information ?

Avec Documentation : https://developer.mozilla.org/fr/docs/Web : la recherche se trouve là :



Puis la fenêtre s'agrandit :



Avec www.w3schools.com/ en haut à droite  ou en navigant sur la gauche.

2. Tester votre code en ligne

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Quelques fois, il est fastidieux de lancer tous les logiciels nécessaires pour tester un morceau de code ou vous êtes sur un ordinateur à la BU.

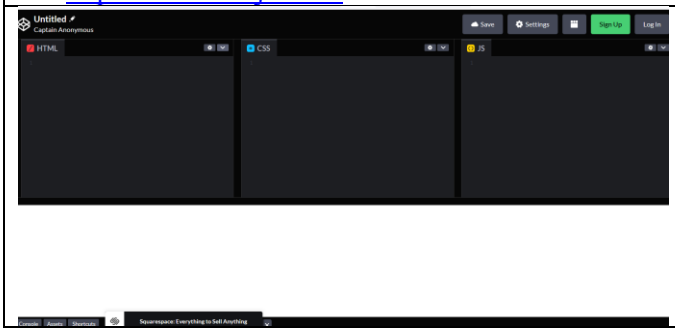
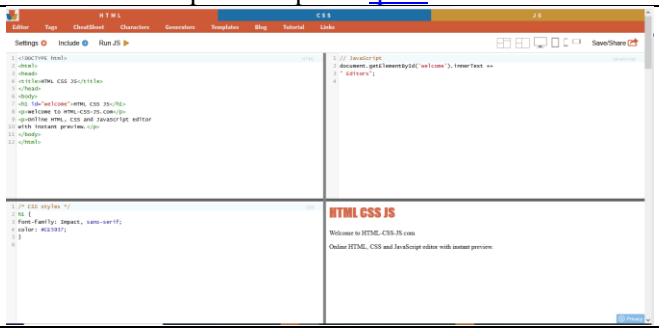
www.w3schools.com (et bien d'autres) permet de tester votre code dans un navigateur :

- Pur HTML : https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default
- CSS avec HTML : https://www.w3schools.com/css/tryit.asp?filename=trycss_default
- Javascript : https://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst

L'inconvénient, c'est que l'HTML, le CSS et le Javascript se trouve dans le même fichier :

Il y a d'autres sites (peut-être en connaissez-vous ?), qui proposent de travailler avec trois fichiers différents :
https://jsfiddle.net/
https://liveweave.com/
et mes deux préférés :

1. https://html-css-js.com/
2. Source : https://codepen.io/pen/

https://html-css-js.com/	Source : https://codepen.io/pen/
	

3. Exemples

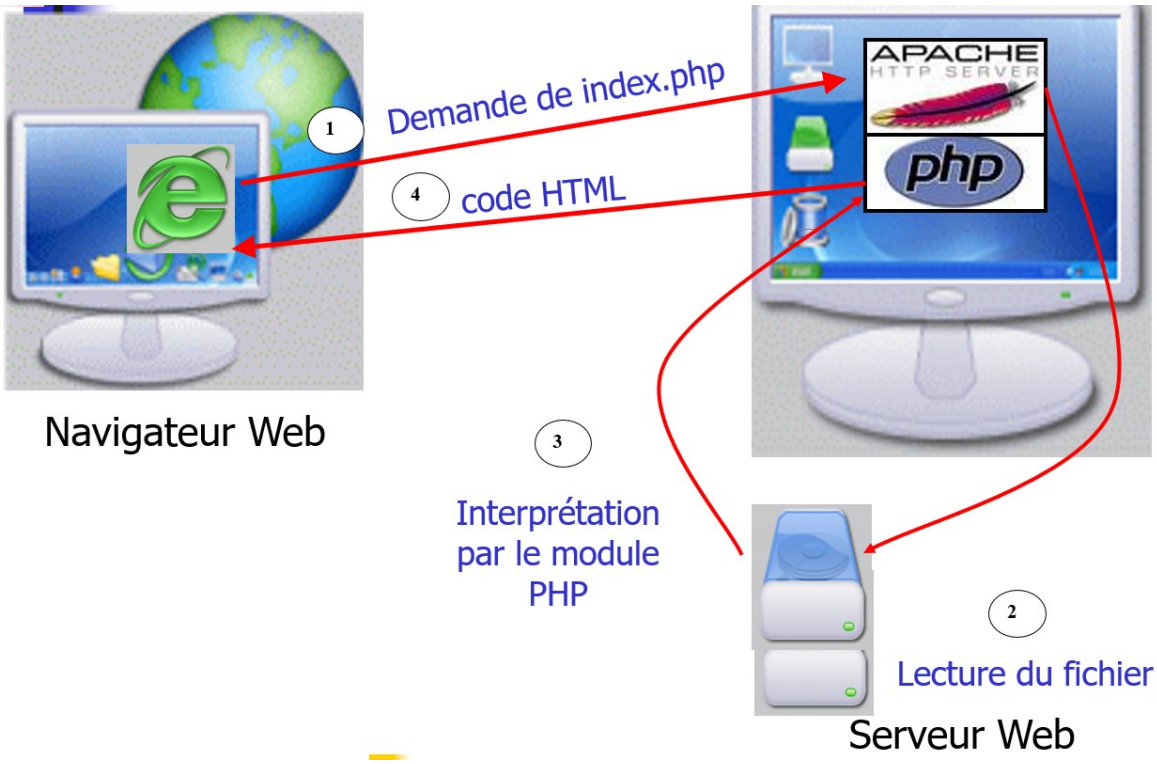
Le plus souvent possible un lien codepen est fourni il peut être utile en CM ou tout simplement pour relire le cours. Mais les fichiers ne peuvent être utilisés tels quels.

💣 Attention : dans le code HTML de codepen il n'y a que les lignes qui se trouvent dans la balise body. Il n'y a pas de référence par exemple aux fichier css ou javascript.

🤞 Github …
https://github.com/rtomczak/CoursJavascript
… construction….

I. Présentation de javascript

1. Comment fonctionne un site web



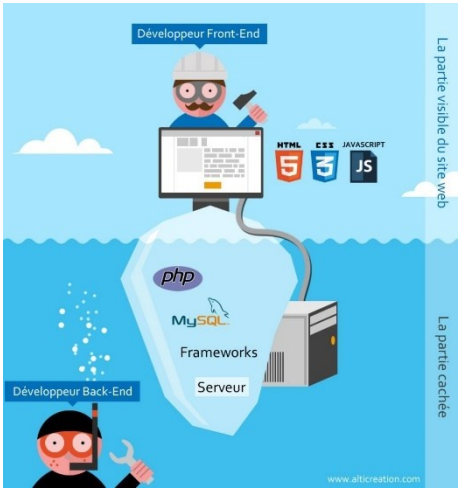
Lors de la demande d'une page web auprès d'un serveur, une demande est envoyée sur celui-ci vers un fichier PHP : par exemple *index.php* dans cet exemple. Le protocole de communication est le *http* ou *https* si la communication est sécurisée.
Le script PHP *index.php* est exécuté sur le **serveur** qui lui renvoie un fichier **HTML**.

Une page HTML texte composée de balises. Celles-ci permettent de mettre en forme le texte (gras, souligné etc…), les images, de créer des formulaires, etc…

Ainsi : votre navigateur (Chrome, Firefox, Safari …) va demander un fichier au serveur et ensuite il va recevoir un fichier **texte** de type HTML qui va interpréter, code HTML et CSS, pour vous l'afficher l'écran. C'est pourquoi, il se peut que l'affichage soit différent sous Chrome ou sous Firefox.

2. Front End vs Back End

Certainement qu'au début de cette relation client-serveur, il était considéré que le serveur, plus puissant, pouvait intégralement exécuter les requêtes, les demandes, et que le navigateur n'avait plus qu'à afficher la page HTML.



Cela à changé puisqu'on s'est aperçu qu'il était préférable, de filtrer les requêtes vers le serveur, qu'il était possible d'effectuer des modifications directement dans votre navigateur.

Le Back-End renvoie aux technologies (PHP, Java, C#, Go, SQL, et même javascript…) qui exécute le code sur le serveur.
Tandis que le Front-End, correspond à ce qui s'exécute sur votre navigateur, en local. Le langage principalement utilisé est *Javascript* et par extension HTML, CSS.

3. Qu'est-ce que Javascript ?

C'est un langage de programmation comme le langage C ou Python orienté objet et il peut être utilisé en Back-End ou en Front-End. Dans ce cours, nous parlerons uniquement de Front-End. Pour la petite histoire, Javascript (JS pour les intimes) a été créé en seulement 10 jours par Brendan Eich afin de donner de l'interactivité aux pages web qui étaient jusqu'à présent uniquement statique.

Encore une fois c'est le langage de programmation qu'il faut connaitre pour le développement web. D'ailleurs dans l'index TIOBE, qui donne un aperçu des langages utilisés dans le développement informatique, Javascript est en 7ème position au mois d'Août 2022 :

Aug 2022	Aug 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	15.42%	+3.56%
2	1	v	C	14.59%	+2.03%
3	3		Java	12.40%	+1.96%
4	4		C++	10.17%	+2.81%
5	5		C#	5.59%	+0.45%
6	6		Visual Basic	4.99%	+0.33%
7	7		JavaScript	2.33%	-0.61%

Une utilisation de JavaScript classique et de vérifier un formulaire dans votre page web avant de l'envoyer vers le serveur ainsi vous traitez les erreurs sur le poste et non sur le serveur ce qui permet ce qui est bien plus rapide et permet d'économiser de la bande passante.

☝ A retenir : Javascript est un langage utilisé pour les scripts côté client pour apporter un côté « dynamique » à la page. Il est interprété par le navigateur.

Le script JS est intégré à la page HTML avec le CSS.

☝ A retenir : Javascript fait parti du trio HTML/CSS/Javascript indispensable pour la l'écriture de page Web dynamique et intégrative côté client.

En résumé :
C'est un langage de programmation qui permet d'inclure du code dans une page Web. Ce programme peut modifier le contenu de la page web, interagir avec l'utilisateur sur un clic bouton …

4. Avantages / inconvénients

a) Avantages

- Ce langage est très populaire et une des raisons et qu'il est très facile à apprendre. La courbe d'apprentissage est très rapide. D'ailleurs beaucoup de jeunes codeurs ont débuté avec ce langage.
- JavaScript peut être utilisé côté serveur notamment avec note.js
- Il est dit que l'on peut tout faire avec JavaScript y compris des applications pour smartphone
- Un framework est un ensemble d'outils, de bibliothèques, qui facilite la vie du programmeur. Il en existe de tous types, dans de nombreux domaines. Et bien Javascript en a de très performantes qui font gagner beaucoup de temps. Les plus populaires, AngularJS, React, Node.js, React Native, sont massivement utilisés en entreprise.

b) Inconvénients

- C'est un langage de script qui est interprété, exécuté par le navigateur. Les langages interprétés (comme le Python) sont censés être moins rapides que les langages compilés (comme le langage C)
- Pour pouvoir utiliser toutes les possibilités de JS, il vous faut connaitre HTML et le CSS
- Son nom : cela porte très souvent en confusion JavaScript n'a rien à voir avec le langage Java. Souvent, certains recruteurs pensent que les programmeurs Javascript peuvent également développer en Java.
- Comme il est interprété par le navigateur, vous n'aurez pas forcément le même résultat avec votre navigateur Chrome, Firefox ou Safari. Dans certains cas il faut prévoir l'exécution du code en fonction de votre navigateur

5. Démarrer avec Javascript

a) Intégrer un script Javascript dans une page web

Il existe plusieurs méthodes pour intégrer un script JS :

- Écrire le code JS dans un fichier dont l'extension est *.js*. Par exemple *script.js*
- Le lier dans le fichier HTML par la ligne : `<script src="script.js"></script>`

> 💣 Dans notre exemple, le fichier `script.js` doit être dans le même répertoire que l'HTML. Il est possible d'utiliser des répertoires afin d'organiser au mieux l'ensemble des fichiers (par exemple en mettant les fichiers CSS dans le répertoire *scripts*.
> Ainsi le lien se fera par `<script src="/scripts/script.js"></script>`

b) Où le placer ?

L'écriture des scripts sur votre navigateur, côté client donc, peut se faire en javascript.

i. Dans le HEAD

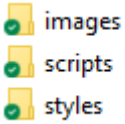
Pour bien séparer le code HTML de celui du Javascript (comme du CSS), il est fortement conseillé d'écrire dans un fichier en dehors de celui de HTML dans l'entête `<head>`

Pour ne pas l'oublier, et qu'il soit toujours visible, je préfère le déclarer dans l'entête avec le css ainsi :

```
<head>
    <meta charset="utf-8">
    <title>Premiers Pas</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
</head>
```

Attention, le fichier *script.js* (ainsi que *style.css)* doit être dans le même répertoire que votre fichier *html.*

Quelques fois il est plus lisible de créer des répertoires et de mettre les fichiers correspondants, par exemple les fichiers concernant le CSS dans répertoire *styles* et ceux Javascript dans *scripts :*



ii. A la fin du body

```
    ...<!-- Mis à la fin pour afficher une première fois -->
    <script src="CSS-JS-01.js"></script></html>
</body>
```

En mettant le code avec le , (normalement) l'affichage de la page se fait avant l'exécution du script JS.

6. Accès aux outils de développement

Source : https://codepen.io/rtomczak/pen/yLjgbmQ

Soit la page suivante :


```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Premiers Pas</title>
        <link rel="stylesheet" href="style.css">
        <script src="script.js"></script>
    </head>
    <body>
        <h1 id="msg">Hello World!</h1>
        <input type="text" value="Premiers pas avec
Javascript" size="30" id="label">
        <br><br>
        <input type="radio" name="choix"
value="1"  id="c1"> Choix N°1
        <input type="radio" name="choix" value="2" checked
id="c2"> Choix N°2
        <input type="radio" name="choix" value="3"
id="c3"> Choix N°3
        <br><br>
```

```
        <input type="password" value="password" id="pwd">
        <br><br>
        <input type="button" value="Ok" id="bp">
    </body>
</html>
```

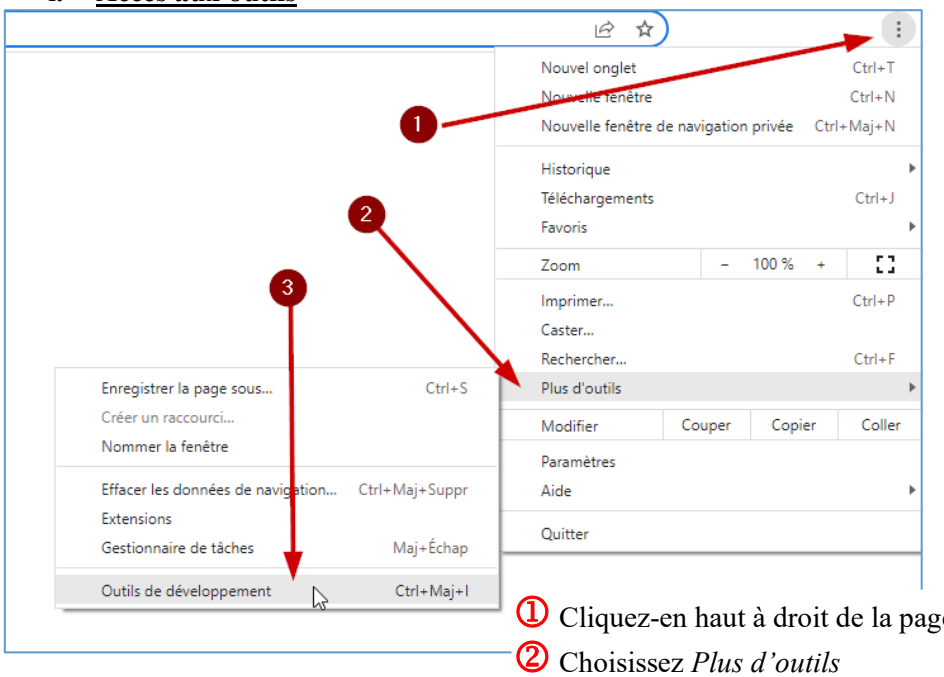
Les outils de développement vous nous d'analyser en profondeur, les pages web. Leurs fonctions sont nombreuses, notamment :

- Naviguer entre les différents rendus, tablette, smartphone …
- Lancer des commandes
- Visualiser le DOM
- Visualiser et modifier le CSS
- …

a) Sous Chrome

Attention : la version des outils de Chrome présentée ici est en Anglais.

i. Accès aux outils



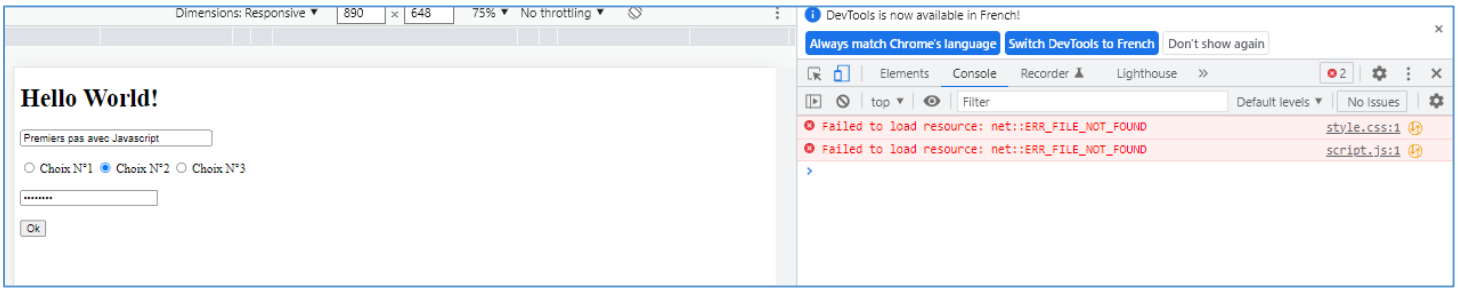
① Cliquez-en haut à droit de la page sur ⋮

② Choisissez *Plus d'outils*

③ Outils de développement

Ou la combinaison : 

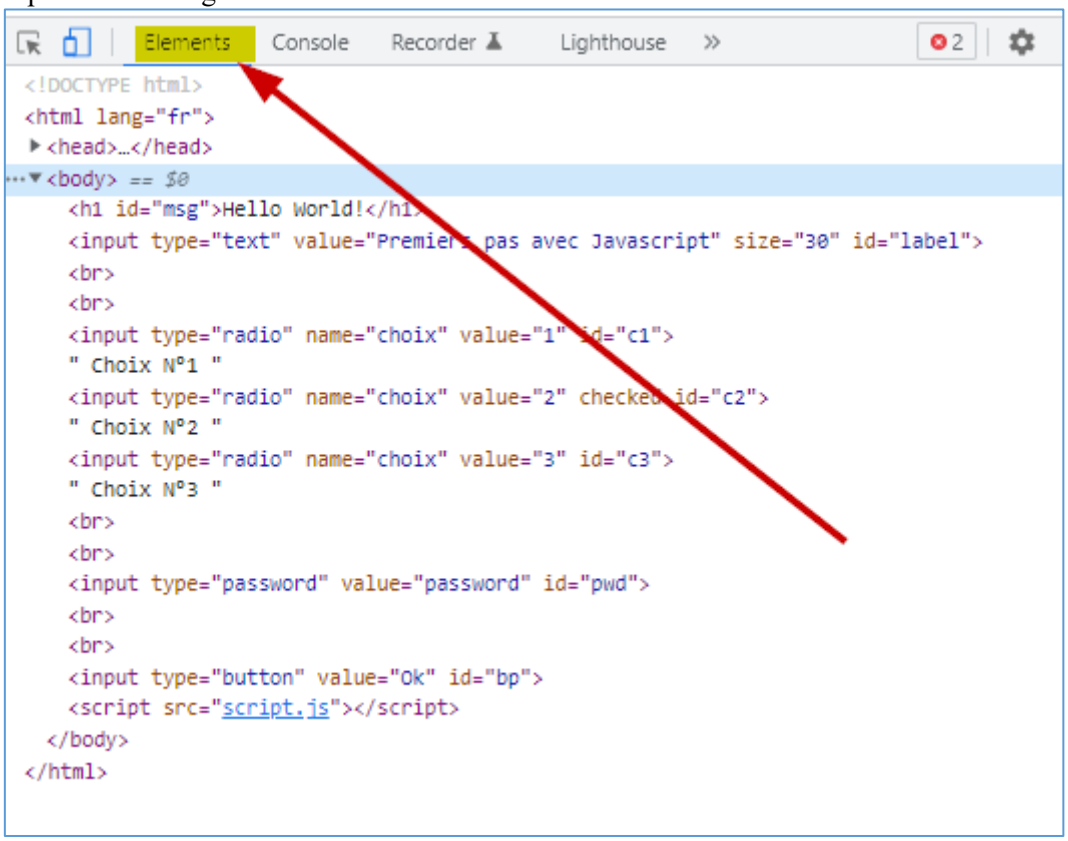
Vous pouvez voir que le navigateur a détecté deux erreurs : effectivement les fichiers *style.css* et *script.js* n'existent pas pour l'instant.



Dans ce cours, nous allons nous intéresser à deux onglets :

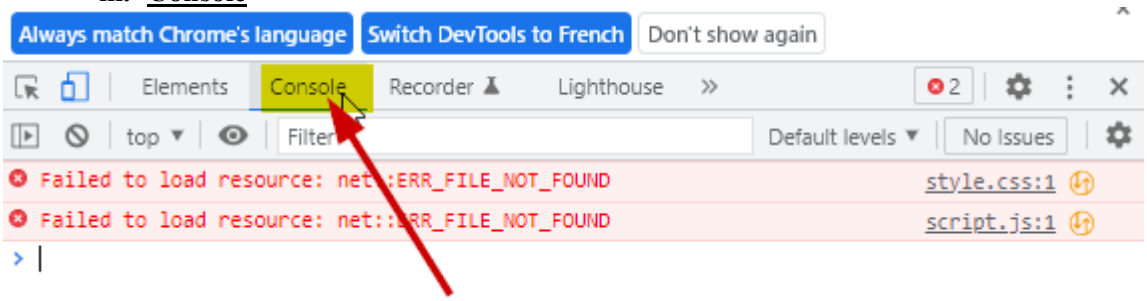
ii. Elements :

Cet onglet nous permet de naviguer dans le code HTML.



Cet onglet nous permet de naviguer dans le code HTML.

iii. Console



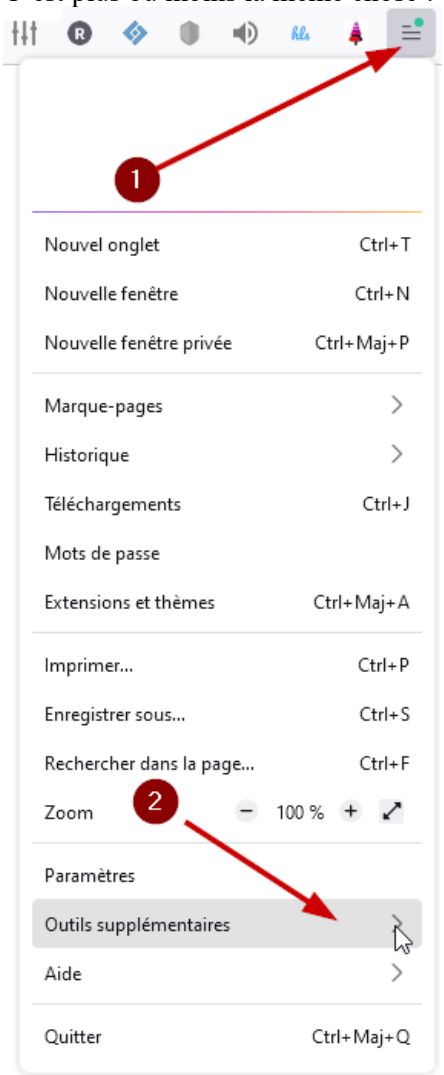
Avec lequel il nous sera possible d'entrer des commandes Javascript.

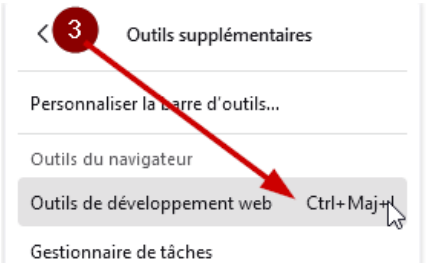
b) Avec Firefox

Attention : la version des outils de Firefox présentée ici est en Français.

i. Accès aux outils

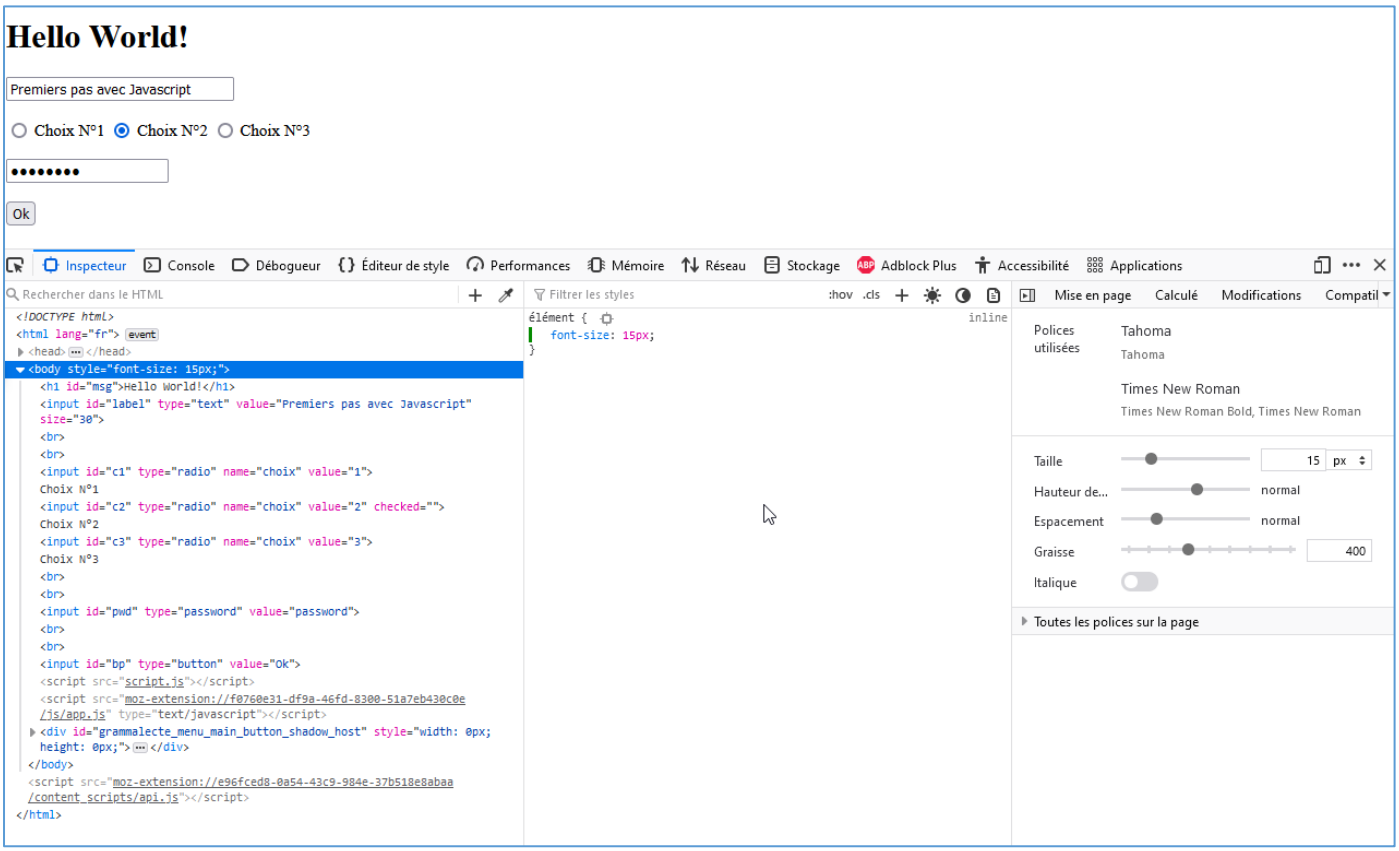
C'est plus ou moins la même chose :





① Cliquez-en haut à droit de la page sur ☰
② Choisissez *Outils supplémentaires*
③ Outils de développement web

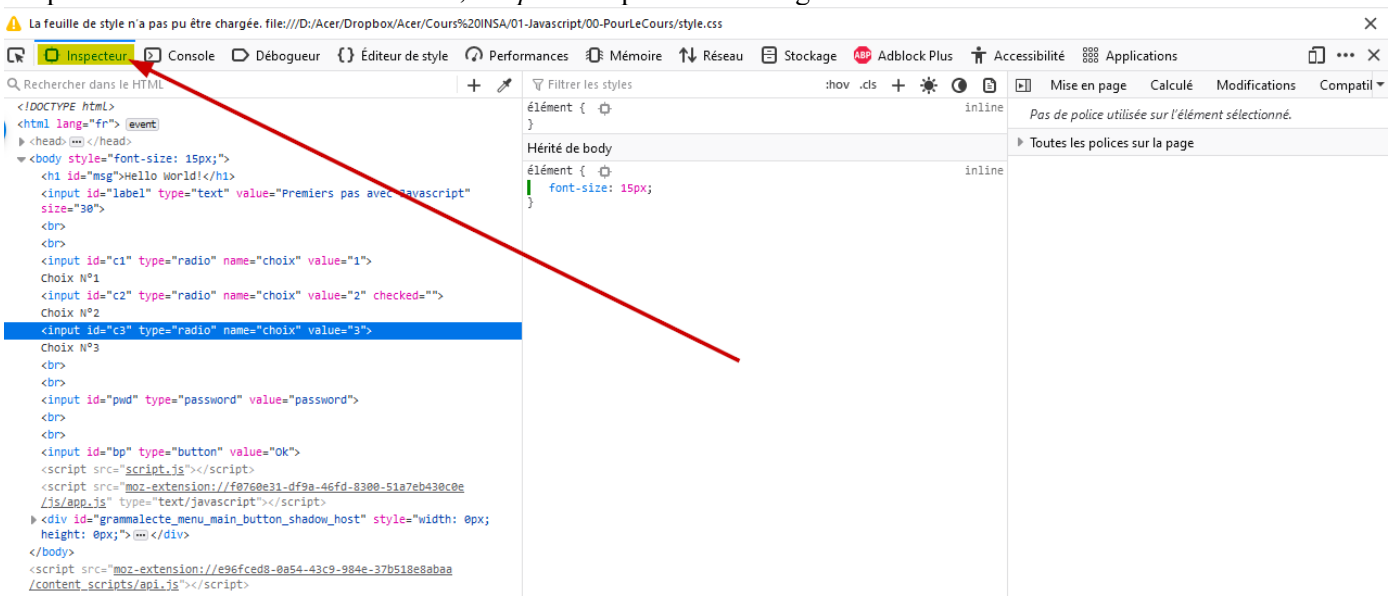
Ou sinon la combinaison : CTRL Maj I



Nous avons beaucoup plus d'informations mais nous allons essentiellement utiliser *Inspecteur* et *Console*

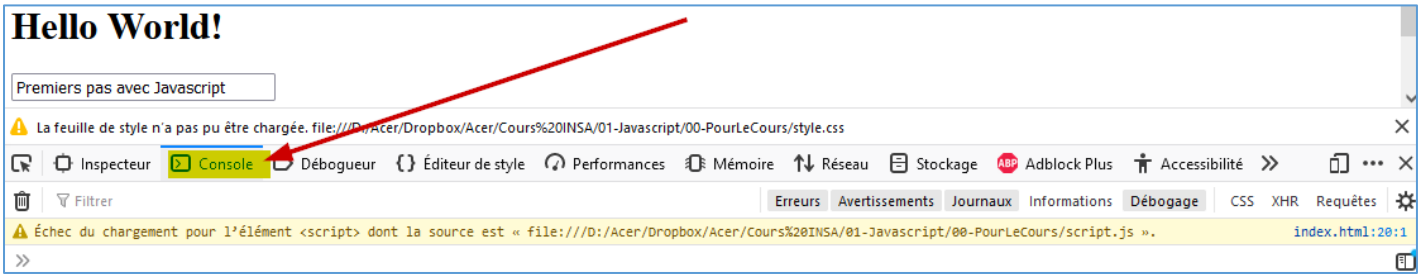
i. Inspecteur

Un peu comme Chrome avec *Elements*, *Inspecteur* permet de naviguer dans le code HTML et CSS



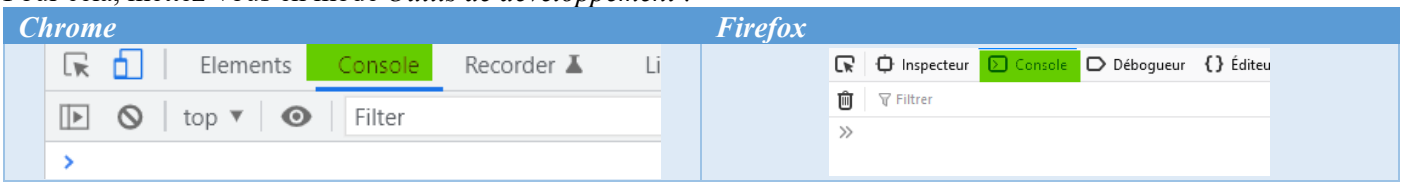
ii. Console

Comme avec Chrome, il est possible de lancer des commandes Javascript



7. Mode console

Le mode console est un mode interactif : vous entrez une commande elle s'exécute directement.
Pour cela, mettez-vous en mode *Outils de développement :*



Par exemple :

Ce mode est très utile pour tester votre code.

💣 Dans ce mode il n'est pas obligatoire de mettre le point virgule à la fin de la commande. Ce n'est pas le cas lorsque le code se trouve dans le fichier !!!

✍ *Les exemples suivants ont été réalisés dans la console*

II. Tour d'horizon du langage Javascript

Au lieu de refaire un cours sur des éléments déjà vus dans d'autres langage, je vais plutôt vous donner des exemples de code.
Le code se trouve dans le pen : [Source : https://codepen.io/rtomczak/pen/abGpwqN](https://codepen.io/rtomczak/pen/abGpwqN)

1. Les variables

La déclaration se fait par `let` :

```
let x; // Déclaration d'une variable x.
```

💣 il est possible de déclarer une variable comme ceci : `var x;` Dans ce cas cette variable est globale, cad qu'elle est visible dans tous les fichiers. Ce comportement est à éviter.

Voici un exemple d'affectation et de type des variable :

```
x = 0; // Maintenant la variable vaut zéro
x ;// retourne sa valeur cad 0
```

JavaScript supporte plusieurs types de variables :

```
x = 1; // des entiers.
x = 0.01; // réels.
x = "hello world"; // chaîne de caractères entre des guillemets doubles
x = 'JavaScript'; // ou des simples
x = true; // valeur booléenne vrai
x = false; // ou faux
x = null; // null est une valeur spéciale qui signifie "pas de valeur"
x = undefined; // Presque la même chose, non définie
```

💣Notez le point virgule à la fin de la ligne.

Les tableaux - listes

Les tableaux peuvent contenir plusieurs éléments :

```
let prix = [212.5, 301, 125, 250]; // un tableau de valeurs délimités par []
prix[0] // retourne 212.5 cad le premier élément à l'indice 0
prix[3] // retourne 250 le dernier élément à l'indice 3
prix[prix.length-1] // Même résultat
prix.length // retourne 4 car le nombre d'éléments est 4.
prix[4] = 119; // Ajoute un nouvel élément.
prix // retourne  Array(4) [ 212.5, 301, 125, 250, 119 ]
prix[4] = 0; // modification d'un éléments du tableau.
let tableauVide = []; // [] indique un tableau vide
tableauVide.length; // retourne 0
```

Remarque : un tableau peut contenir différents types de données

```
let tab = [1,"e"] ;
```

2. Des dictionnaires (appelés JavaScript Objects)

Les dictionnaires sont une sorte de collection d'Object
L'object est une collection de paires nom/valeur comme en Python qui se trouvent entre crochets {}

```
let ang2fr ={"one":"un"} ;
ang2fr["two"] = "deux"; // Ajout d'une paire nom/valeur
ang2fr; // Renvoie Object { one: "un", two: "deux" }
ang2fr.one; // Renvoie "un"
ang2fr.content = {}; // Le contenu est vidé
```

3. Listes et dictionnaires

a) Liste de dictionnaire

Il est possible d'inclure des dictionnaires dans une liste :

```
let pointsXY = [ // un tableau à deux dimensions
    {x:1, y:0},// l'élément est un object
    {x:10, y:10}
    ];
pointsXY[1] ;// Retourne Object { x: 10, y: 10 }
```

b) Des dictionnaires de tableaux

```
let vecteurs = {
    vect1:[ [1,2] , [3,4] ],
      vect2:[ [3,4] , [5,4]]
  };

  vecteurs["vect1"] // Renvoie Array [ [1,2] , [3,4] ]
  vecteurs["vect"] = [ [11,12], [21,22]]; // Modifie le deuxième
```

4. Des calculs

```
// Calculs
3 + 2 ;// retourne 5: addition
3 - 2 ;// retourne 1: soustraction
3 * 2 ;// retourne 6: multiplication
3 / 2 ;// retourne 1.5: division
"3" + "2" ;// retourne Attention le résultat est "32"
```

```
// Les opérateurs d'incrémentation et de décrémentation
let compteur = 0; // Defini un compteur
compteur++; // Incrémente le compteur
compteur--; // Décrémente le compteur
compteur += 2; // ajoute 2 à la variable : idem compteur = compteur + 2;
```

```
compteur *= 3; // Multiplie par 3: idem compteur = compteur * 3;
compteur // retourne 6
```

5. Conditions booléennes

```
// Condition booléenne
let x = 1, y = 2;
x === y ;// retourne false faux: égalité stricte
x !== y ;// retourne true vrai: inégalité
x < y ;// retourne true vrai
x <= y ;// retourne true vrai
x > y ;// retourne false faux
x >= y ;// retourne false faux
"un" === "deux" ;// retourne false faux
"un" > "deux" ;// retourne false faux car "un" est alphabétiquement plus grand que "de"
```

💣L'égalité faible (==) effectuera une conversion des deux éléments à comparer avant d'effectuer la comparaison
L'égalité stricte (===) effectuera la même comparaison mais sans conversion préalable (elle renverra toujours false si les types des deux valeurs comparées sont différents)

```
let num = 0, str="0";
num == str; // Retourne true Vrai !!!
num === str; // Retourne false Faux ouf .
```

6. Définition et utilisation de fonctions

```
// Les fonctions
function carre(x){ // La définition d'une fonction se fait par function
    return x*x; // on retourne le carré de x : x*x
  }
```

L'appel se fera tout simplement comme ceci :

```
carre(4); // Appelle la fonction carre avec comme paramètre 4. Cette fonction retourne 16
```

7. If alors sinon :

```
// Conditionnel
function abs(x){
    if (x<0){
        return -x;
    }
    else {
        return x;
    }
}
```

Puis les appels pouront être :

```
abs(3); // Retourne 3
abs(-3); // Retourne 3
```

💣 Vous avez remarqué l'indentation (tabulation) et les crochets {} qui entourent les instructions. Les tabulations ne sont pas obligatoires mais vivement conseillées par une meilleure lisibilité.
Par contre les crochets le sont !!!

8. Une boucle avec pour

Une variable de boucle, ici nommée `i` est déclarée puis utilisée avec `for` :

```
// Boucle pour
let somme = 0,i;
for (i=0;i<=3;i++) {
    somme = somme + i;
}
somme;// Renvoie 6 car 0+1+2+3 = 6
```

9. Boucle tant que

Ce code effectue le même calcul que précédemment :

```
// Boucle tant que
i = 0;
somme = 0;
while (i<=3){
    somme += i;
    i++;
}
somme;// Renvoie 6 car 0+1+2+3 = 6
```

10. For avec un tableau

```
// pour avec un tableau
let prix = [10,100,1000];

function calculSomme
(tableau) { // Calcule la somme des éléments de tableau
    let total = 0; // initialisation à zéro.
    // Boucle sur tableau et récupère chaque élément dans unPrix
    for(let unPrix of tableau) {
        total += unPrix; // Ajoute l'élément au total
    } // Fin de la boucle
}

somme(prix) // Renvoie 10+100+1000 = 1110
```

11. En résumé

Pour ceux qui connaissent le langage C, PHP, ou Java, Javascript est très proche notamment le point virgule à la fin des lignes.

Types de variables	
Déclaration	`let unEntier = 3;`
Affectation	`somme = 1.2 + 1.3;`
Chaînes	`let nom = "toto";` ou `let nom2 = 'tata'`
Booléens	`true` ou `false`
Opérateurs ou et non	`||` `&&` `!`
Tableaux	`t = [1,2,3,4]` et `t[1]` ou `t.lenght`
Dictionnaires	`dico = {a:1, b:2, c:3}` `d.a`
Éléments du langage	
Commentaire	`// Ceci est un commentaire`
Conditionnelle : si alors sinon	`if (condition) { instructions } else { instructions }`
Boucle pour	`if (condition) { instructions } else { instructions }`
Boucle tant que	`while (condition) { instructions }`
Fonction avec retour	`function Mafonction(paramètres) { instructions return qqchose }`

☝ L'opérande permettant la division entière (// en Python) n'existe pas en Javascript

III. Quelques rappels de HTML et CSS

1. HTML

Le langage HTML (HyperText Markup Language) permet de structurer le contenu de la page web à l'aide balises. On ne peut réellement parler de langage de programmation : c'est plutôt un langage de balisage qui indique aux navigateurs comment structurer, présenter la page web.

a) Structure d'une page HTML

Le langage HTML est un langage de balises.
Source : https://codepen.io/rtomczak/pen/VwxPWPP :

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Page de test</title>
    </head>
  <body>
    <p>Voici ma page web</p>
  </body>
</html>
```

`<!doctype html>` : type de document web
`<title>` et `</title>` : nom de la page, celle qui sera affiché en haut
`<html>` et `</html>` : balise racine contenant tout le code html
`<head>` et `</head>` : informations relatives à la page web mais qui ne s'affichera pas
`<body>` et `</body>` : relatif aux contenus visibles pour l'utilisateur

b) Les balises communes

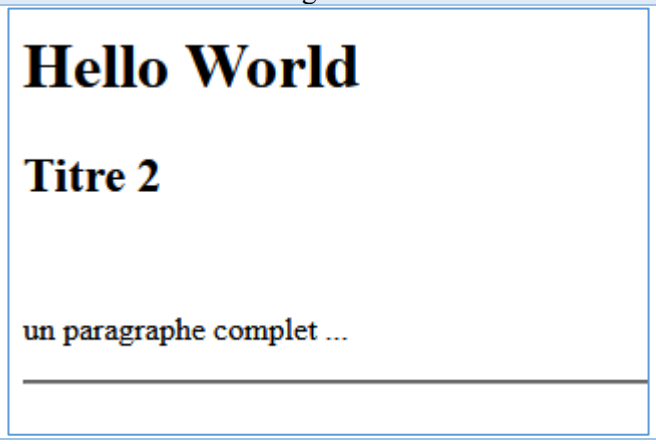
<h1> à <h6>	Niveaux de titre hx où représente le niveau
<p>	Définition d'un paragraphe
 	Saut de ligne
<hr>	Ligne horizontale
<!--...-->	Commentaire

Source : https://codepen.io/rtomczak/pen/zYjNEOj :

Exemple : balises communes

HTML	Page web

```
<h1>Hello World</h1>
<h2>Titre 2</h2>
<br>
<p>un paragraphe
complet
...
<hr>
```

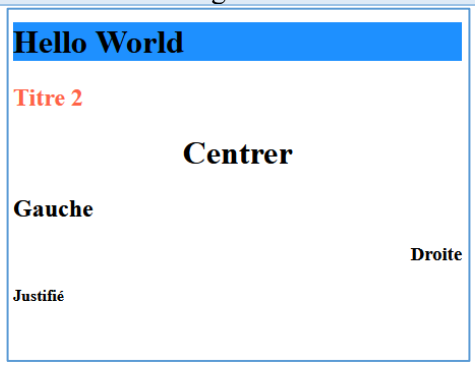


Avec un peu de CSS :
Source : https://codepen.io/rtomczak/pen/RwyKLwX

Avec du CSS

HTML	Page web

```
<h1 style="background-color:DodgerBlue;">Hello
World</h1>
<h2 style="color:Tomato;">Titre 2</h2>
<h1 style="text-align:center">Centrer</h1>
<h2 style="text-align:left">Gauche</h2>
<h3 style="text-align:right">Droite</h3>
<h4 style="text-align:justify">Justifié</h4>
```



c) Les commentaires

Comme tout code informatique, il est bon d'expliquer ou de mettre des remarques. En html les commentaires doivent être mis entre deux balises `<!--` et `-->`.

```
<!-- Sur une seule ligne-->
<!--
    Un très long
    commentaire
-->
```

Dans les exemples donnés dans ce cours, les commentaires sont quelques fois utilisés.

d) Mise en forme des textes et paragraphes

Source : https://codepen.io/rtomczak/pen/LYmxzbv

Texte et paragraphe

HTML

```
<p>The <abbr title="HyperText Markup Language">HTML</abbr> permet de créer des pages web.</p>
<p>This is normal text
<b>and this is bold text</b></p>
<p><code>var nombre=2</code> déclare une variable globale</p>
<p><strong>Ce texte est important!</strong></p>
<i>En italique</i>
<u>Souligné</u>
<s>Barré</s>
<sup>Exposant</sup>
<sub>Indice</sub>
```

Page web

The HTML permet de créer des pages web.

This is normal text **and this is bold text**

`var nombre=2` déclare une variable globale

Ce texte est important!

En italique Souligné ~~Barré~~ Exposant Indice

☝ Remarque : remarquez l'absence de la balise paragraphe `<p>` dans les cinq dernières lignes et le résultat sur la droite. Même si dans le fichier html il existe un retour de ligne, elle n'apparait pas à l'affichage. Il faut soit utiliser une balise `<br>` ou encore mieux formater votre texte avec un paragraphe `<p>`.

e) Les listes

Il existe deux types de listes :

- Liste ordonnée :
  - `<ol>` qui permet d'avoir une liste numérotée
  - `<li>` qui représente un item.
  - `<lh>` indique le titre de la liste.
- Liste non ordonnée.
  - `<ul>` à la place de `<ol>`

Source : https://codepen.io/rtomczak/pen/YzLNrrP :

Listes

HTML	Page web
`<h1>Avec la balise ul</h1>` `<ul>` `  <li>Thé</li>` `  <li>Café</li>` `  <li>Chocolat</li>` `</ul>` ` ` `<h1>Avec la balise ol</h1>` `<ol>` `  <li>Thé</li>` `  <li>Café</li>` `  <li>Chocolat</li>` `</ol>` ` `	**Avec la balise ul** • Thé • Café • Chocolat **Avec la balise ol** 1. Thé 2. Café 3. Chocolat

iii. Avec du CSS

Source : https://codepen.io/rtomczak/pen/GRdrMOZ :

Listes et CSS

HTML	Page web
`<h2>Espacement normal</h2>` `<ol>` `  <li>Thé</li>` `  <li>Café</li>` `  <li>Chocolat</li>` `</ol>` ` ` `<h2>80% de la taille d'une ligne</h2>` `<ol style="line-height:80%">` `  <li>Thé</li>` `  <li>Café</li>` `  <li>Chocolat</li>` `</ol>`	**Espacement normal** 1. Thé 2. Café 3. Chocolat **80% de la taille d'une ligne** 1. Thé 2. Café 3. Chocolat

f) Les tableaux

- La balise <table> qui indique au navigateur la création d'un tableau.
- La balise <th> définit l'en-tête de chaque colonne.
- La balise <tr> définit une ligne.
- La balise <td> définit une cellule.

Le code HTML est : [Source : https://codepen.io/rtomczak/pen/gOzgGoM](https://codepen.io/rtomczak/pen/gOzgGoM)

```
<head>
    <meta charset="utf-8">
    <style>
        table {
            border: 2px solid rgb(244, 8, 8);
        }
        th, td {
            border: 1px solid rgb(14, 14, 14);
        }
    </style>
</head>
<body>
    <table>
        <caption>Tableau simple</caption>
        <tr>
            <th>COLONNE 1</th>
            <th>COLONNE 2</th>
            <th>COLONNE 3</th>
        </tr>
        <tr>
            <td>Première ligne col1</td>
            <td>Première ligne col2</td>
            <td>Première ligne col3</td>
        </tr>
        <tr>
            <td>Deuxième ligne col1</td>
            <td>Deuxième ligne col2</td>
            <td>Deuxième ligne col3</td>
        </tr>
        <tr>
            <td>Troisième ligne col1</td>
            <td>Troisième ligne col2</td>
            <td>Troisième ligne col3</td>
        </tr>
        <tr>
            <td>....</td>
            <td>....</td>
            <td>....</td>
        </tr>
        <tr>
            <td>Dernière ligne col1</td>
            <td>Dernière ligne col2</td>
            <td>Dernière ligne col3</td>
        </tr>
    </table>
```

Un peu de CSS sinon il n'y aura aucun cadre lisible

Idem pour les lignes et les colonnes

Et le résultat

Tableau simple

COLONNE 1	COLONNE 2	COLONNE 3
Première ligne col1	Première ligne col2	Première ligne col3
Deuxième ligne col1	Deuxième ligne col2	Deuxième ligne col3
Troisième ligne col1	Troisième ligne col2	Troisième ligne col3
----	----	----
Dernière ligne col1	Dernière ligne col2	Dernière ligne col3

☝ Il existe la possibilité de regrouper des lignes ou des colonnes avec `rowspan` ou `colspan` (https://www.w3schools.com/html/html_table_colspan_rowspan.asp)

g) Les formulaires

Nous avons déjà présenté un formulaire qui était composé des éléments :

- Un champ de texte
- Trois boutons radios
- Un champ mot de passe
- Et un bouton

Source : https://codepen.io/rtomczak/pen/yLjgbmQ :


```
        <h1 id="msg">Hello World!</h1>
        <input type="text" value="Premiers pas avec
Javascript" size="30" id="label">
        <br><br>
        <input type="radio" name="choix"
value="1"  id="c1"> Choix N°1
        <input type="radio" name="choix" value="2" checked
id="c2"> Choix N°2
        <input type="radio" name="choix" value="3"
id="c3"> Choix N°3
        <br><br>
        <input type="password" value="password" id="pwd">
        <br><br>
        <input type="button" value="Ok" id="bp">
```

☝ Tous les éléments d'un formulaire débutent par la balise `<input>`

💣 Mais dans ce code, il manque l'essentiel : nous avons des éléments de formulaire, **mais pas de formulaire !!**

i. Vers un fichier php distant :

En effet, un formulaire débute toujours par la balise `<form>`. Par exemple ce formulaire envoie les informations contenus (..) vers le fichier *dd* qui se trouve sur le serveur web :

Source : https://codepen.io/rtomczak/pen/oNdBGdv

```
<form method="post" action="http://www.somesite.com/login.php">
    <p>Identifiant : <input type="text" name="login"></p>
    <p>Mot de passe : <input type="password" name="mdp"></p>
    <input type="submit" value="Valider">
  </form>
```

Et du côté serveur le php pourrait être quelque chose comme cela :

```
<?php
if(isset($_POST['login']) && isset($_POST['mdp'])) {
echo "Bienvenue ".$_POST['login'];
}
```

Le formulaire est validé par un bouton `input` de type `submit`. Toutes les valeurs des éléments `input` sont envoyées au serveur, à l'adresse définie dans `action` par `post` ou `get` indiqué dans l'attribut `method` .

💣 Dans ce cours, nous allons nous focaliser sur le code côté navigateur, c'est pourquoi il est très souvent omis de mettre `<form>`

ii. Les composants les plus utilisés :

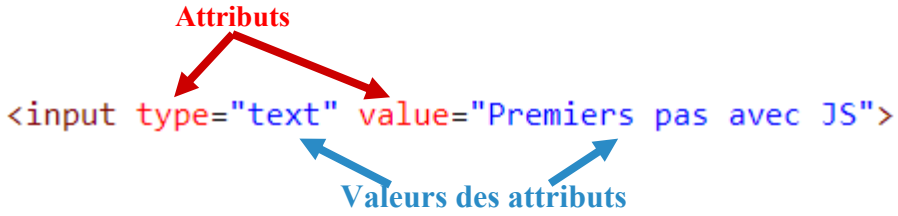
`<input type=`	**Description**
`"text"`	Zone de texte
`"password"`	Mot de passe
`"hidden"`	Composant invisible mais envoyé vers le serveur
`"radio"`	Boutons radios : un seul choix possible
`"checkbox"`	Cases à cocher : plusieurs choix possibles
`"button"`	Un bouton
`"reset"`	Bouton permettant la remise à zéro du formulaire
`"submit"`	LE bouton pour envoyer le formulaire
`"file"`	Choisir un fichier
…	

iii. « Nouveaux » composants qui facilite la vie :

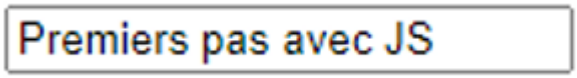
`<input type=`	**Description**
`"email"`	Zone de texte qui doit respecter le format d'email
`"date"`	Zone de texte qui doit respecter le format date
`" range"`	Curseur
`"code"`	Pour choisir une couleur
…	

iii. Attributs type et value

En HTML l'attribut `type` indique le type du composant et `value` sa valeur.
La syntaxe est la suivante :



L'affichage sera dans ce cas :



iv. Quels attributs pour quels composants ?

Il existe un très grand nombre d'attributs souvent différents pour chaque composant : il faut simplement consulter la documentation.

v. Label

Documentation : https://developer.mozilla.org/fr/docs/Web/HTML/Element/Label

Pour des questions d'accessibilité il est conseillé d'associer une balise `label` à vos composants :

Son attribut doit être identique à l'id du composant :

```
<label for="nom">Entrez votre nom</label>
<input id="nom" type="text" name="nom">
```

h) Images

Source : https://codepen.io/rtomczak/pen/mdLRBKE

C'est la balise bien connue `<img>` qui permet l'affiche d'images :

```
<!-- L'image dollar.jpf se trouve dans le même répertoire que la page HTML -->
<img src="dollar.jpg">
<!-- euro.png se trouve dans le répertoire images -->
<img src="./images/euro.png">
<!-- Image située sur un autre site -->
<img
src="https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png">
```

i) Liens

Source : https://codepen.io/rtomczak/pen/RwyKLBa

L'attribut détermine la cible du lien qui est l'adresse du document que votre navigateur va aller afficher : une page html bien sûr, mais aussi une image ou un document pdf ...

```
<a href="http://www.wikipedia.org"> La page principale de Wikipedia</a>
```

La page principale de Wikipedia

Il est possible de spécifier une cible avec `target`, par exemple un nouvel onglet :

```
<a href="http://www.wikipedia.org" target="_blank"> wiki </a>
```

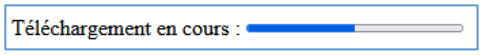
j) Barre de progression et gauge

Très utile dans notre cours, car sa valeur peut être modifiée en JS deux types de barres.
Source : https://codepen.io/rtomczak/pen/OJZWxBm

i. Barre de progression

Documentation : https://developer.mozilla.org/fr/docs/Web/HTML/Element/Progress
Comme son nom l'indique, affiche la progression d'un évènement, un téléchargement par exemple :

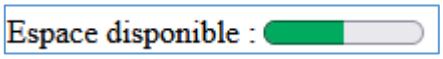
```
<label for="file">Téléchargement en cours : </label>
<progress id="file" value="50" max="100"> </progress>
```



ii. Une jauge :

Documentation : https://developer.mozilla.org/fr/docs/Web/HTML/Element/Meter
Une jauge indique plutôt un état qu'une progression : espace disponible par exemple.

```
<label for="espace_dispo">Espace disponible :</label>
<meter id="espace_dispo" value="0.5"></meter>
```



k) Grouper avec div et span

Documentation : https://developer.mozilla.org/fr/docs/Web/HTML/Element/span
https://developer.mozilla.org/fr/docs/Web/HTML/Element/div
Ces deux balises de regroupement sont utilisées en CSS avec les attributs class et id.

Avec la balise (`<p>`) le paragraphe est appelé un élément de bloc, qui peut contenir plusieurs éléments.

- `<span>` permet de définir ce que l'on appelle un élément de ligne, un seul élément.
- `<div>` quant à lui, comme , est un élément de bloc :

D'après https://www.pierre-giraud.com/html-css-apprendre-coder-cours/div-span/
https://codepen.io/rtomczak/pen/oNdBoGB

Html

```
<p>L'élément span qui suit est un <span
class="FormatageSpan">élément en ligne</span>
;
    son arrière-plan est coloré afin
d'illustrer la zone couverte par cet élément
en
    ligne.</p>
  <div id="bloc1">
    <p>Un premier paragraphe</p>
    <p>Un autre paragraphe</p>
    <ul>
      <li>Un élément de liste</li>
      <li>Un <span class="FormatageSpan">autre
élément</span> de liste</li>
    </ul>
  </div>
```

Css

```
.FormatageSpan {
    /* Une classe*/
    font-weight: bold; /*Les textes seront en
gras*/
    background-color: yellow; /*Fond Jaune*/
    color: black; /*Couleur du texte noire*/
  }

  #bloc1 {
    /* Avec un id*/
    font-weight: bold; /*Les textes seront en
gras*/
    background-color: blue; /*Fond Jaune*/
    color: white; /*Couleur du texte noire*/
    width: 40%; /*Définit la largeur à 5 */
  }
```

Résultat

L'élément span qui suit est un **élément en ligne** ; son arrière-plan est coloré afin d'illustrer la zone couverte par cet élément en ligne

Un premier paragraphe

Un autre paragraphe

- **Un élément de liste**
- **Un autre élément de liste**

2. CSS

Le CSS (Cascading Style Sheet) permet de définir l'aspect du site web : pour simplifier le contenu se trouve dans les fichiers HTML et la forme (couleurs, polices …) est définie dans le fichier CSS. Ainsi tout le site aura le même format (par exemple le titre 1 en gras, bleu…) et la modification de ce seul fichier modifiera l'aspect de toutes les pages HTML associées.

Par exemple, la feuille de style CSS `style.css` utilisée par une page HTML est définie dans la balise `<head>` par :

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

💣 Dans notre exemple, le fichier `style.css` doit être dans le même répertoire que l'HTML. Il est possible d'utiliser des répertoires afin d'organiser au mieux l'ensemble des fichiers (par exemple en mettant les fichiers CSS dans le répertoire *css*.
Ainsi le lien se fera par `<link href="/css/style.css" rel="stylesheet">`.
Plusieurs feuilles de style peuvent être utilisées.

a) Directement dans le fichier html

Quelques fois pour des questions de lisibilité ou de rapidité, on met le css directement dans le html :

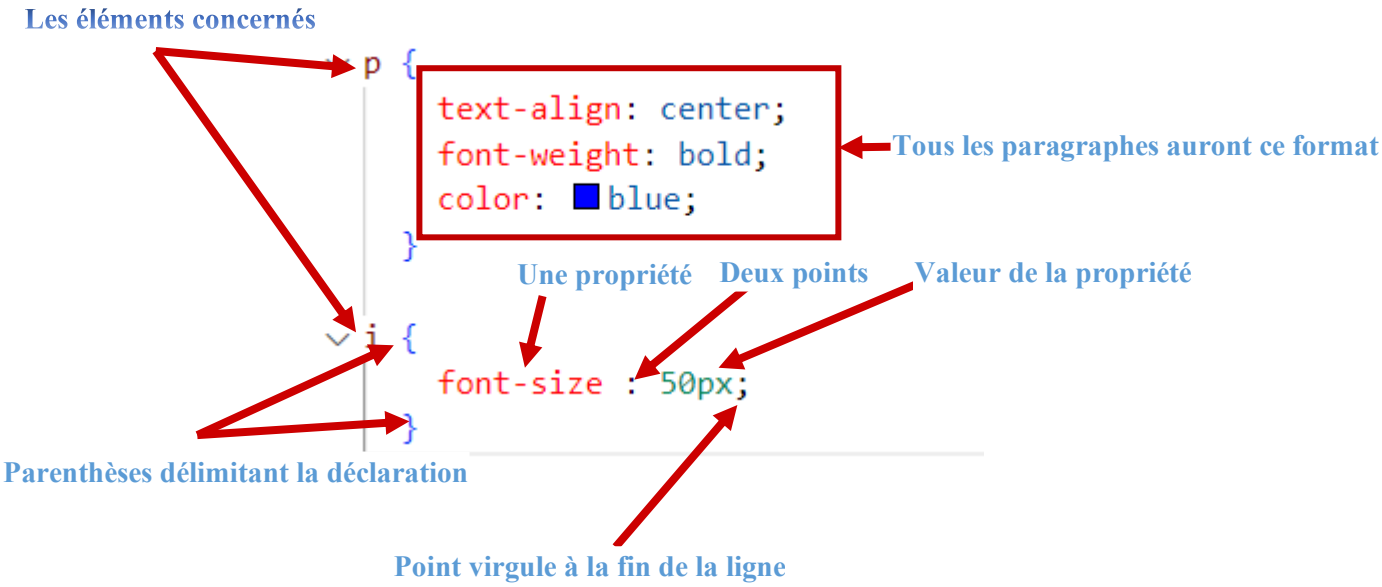
```
<h1 style="background-color: red;">Hello World!</h1>
```

☝ Il est possible de mettre le css directement dans le fichier html mais ce n'est pas conseillé : il faut essayer de conserver une différence entre le fond (hmtl) et la forme (css).

b) Avec les balises

Imaginez que vous êtes un webdesigner et que vous voulez que tous les titres 1, les paragraphes soient identiques et bien vous allez créer une feuille de style qui sera utilisée dans tout votre site.

Par exemple ce css :



https://codepen.io/rtomczak/pen/RwyKdvx

Avec le html:

```
<p>The <abbr title="HyperText Markup Language">HTML</abbr> permet de créer des pages
web.</p>
<p>This is normal text
  <b>and this is bold text</b>
</p>
<p><code>var nombre=2</code> déclare une variable globale</p>
<p><strong>Ce texte est important!</strong></p>
<i>En italique</i>
<u>Souligné</u>
<s>Barré</s>
<sup>Exposant</sup>
<sub>Indice</sub>
```

Le résultat sera :

The HTML permet de créer des pages web.

This is normal text and this is bold text

var nombre=2 déclare une variable globale

Ce texte est important!

En italique Souligné ~~Barré~~ Exposant Indice

c) Id et class

La différence est simple :

- L'`id` concerne un et un seul élément
- `class` regroupe un ensemble d'éléments

i. Soit le css :

```css
. formatage1{
   /* Une classe*/
   font-weight: bold; /*Les textes seront en gras*/
   background-color: yellow; /*Fond Jaune*/
   color: black; /*Couleur du texte noire*/
  }

  #bloc1 {
   /* Avec un id*/
   font-weight: bold; /*Les textes seront en gras*/
   background-color: blue; /*Fond Jaune*/
   color: white; /*Couleur du texte noire*/
   width: 40%; /*Définit la largeur à 5 */
  }
```

ii. Le html :

La classe est utilisée plusieurs fois

```html
<p>L'élément span qui suit est un <span class="formatage1">élément en ligne</span> ;
  son arrière-plan est coloré afin d'illustrer la zone couverte par cet élément en
  ligne.</p>
<div id="bloc1">
  <p>Un premier paragraphe</p>
  <p>Un autre paragraphe</p>
  <ul>
    <li>Un élément de liste</li>
    <li>Un <span class="formatage1">autre élément</span> de liste</li>
  </ul>
</div>
<p class="formatage1"> En dehors d'un span</p>
```

Bloc identifié de façon uniquement par l'id bloc1

d) Sélecteurs

Dans le fichier css, on parle de sélecteurs : ils indiquent les éléments qui sont concernés par le formatage. Voici les principaux

Type de sélecteur	Définition dans le css	Éléments concernés
D'élément	`p {}` `ul{}`	`<p>` `<ul>`
De classe	`.formatage1 {}`	Tous les éléments ayant comme attribut `class="formatage1"`
D'id	`#bloc1 {}`	L'élément ayant comme attribut `id="bloc1"`

D'autres exemples :	
`p.formatage1 {}`	Tous les paragraphes ayant comme classe `formatage1`
`h1,p {}`	Tous les `<h1>` et `<p>`
`input[type=text] {}`	Tous les éléments `<input type="texte">`
`#bloc1 .formatage1 span {}`	Les balises `<span>` contenu dans les classes `.formatage1` elles-mêmes contenues dans un élément dont l'identifiant est `bloc1`

Exemple : https://codepen.io/rtomczak/pen/mdLRYEb

HTML

```
<h1> Titre 1</h1>
<ul>
    <li>1er élement</li>
    <li>Second</li>
    <li>Troisième</li>
</ul>

<p id="bloc1">Bloc d'indentificateur bloc1 </p>
<span class="formatage1"> Span avec classe formatage1 </span>
<p class="formatage1">Comme avec span mais la couleur est maintenant cyan </p>
```

Css :

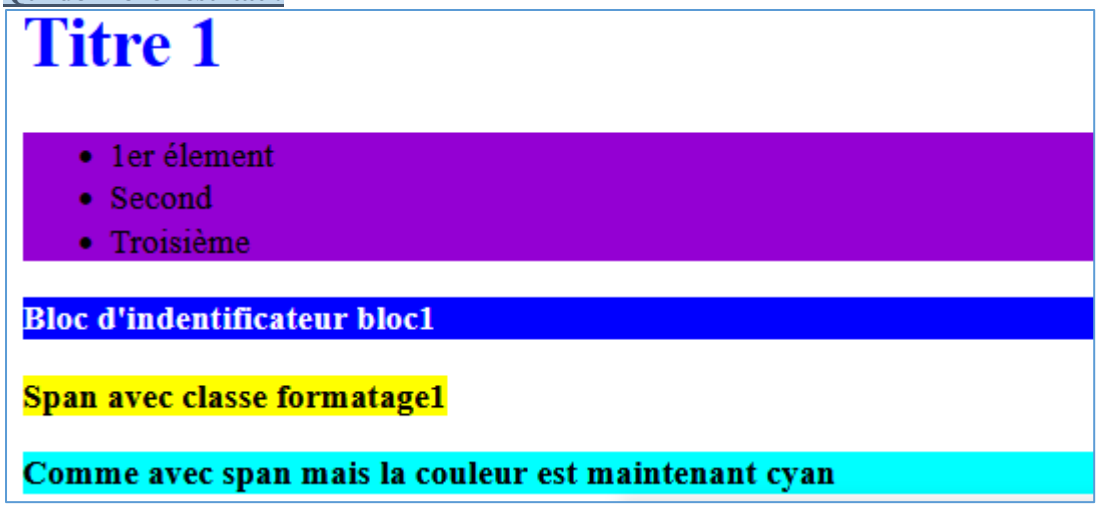
```
h1 {
    color: blue;
  }
  ul {
    background-color: darkviolet;
  }
  .formatage1 {
    font-weight: bold;
    background-color: yellow;
    color: black;
  }

  #bloc1 {
    font-weight: bold;
```

```
    background-color: blue;
    color: white;
    width: 40%;
  }

  p.formatage1 {
    background-color: cyan;
  }
```

Qui donne le résultat :



e) Les propriétés du texte

i. Les unités courantes

Vous avez peut-être remarqué la ligne : `width: 40%;` la `font-size : 50px;`
Ainsi la largeur a été définie comme faisant 40 % de celle du parent (la fenêtre par exemple) et la taille de la police est de 50 pixels.
La taille peut également s'exprimer en em, 1 em est la taille de la police du document (ou du parent si cela a été modifié auparavant), donc 0.5 em correspond à la moitié de la taille et 3em, trois fois plus grand.

ii. Quelques propriétés pour le texte :

Propriété	Signification
`color: aqua;`	Couleur du texte
`background-color : red;`	Couleur du fond
`font-family: 'Times New Roman', Times, serif;`	Type de police
`font-size: 150%;` `font-size: 2em;`	Taille de la police
`text-shadow: 0px 0px 9px rgb(245, 25, 25);`	Ombre
`text-outline`	Contour
`opacity: 0.5;`	Opacité
`text-align: center;`	Alignement
	…

iii. Exemple avec le texte:
https://codepen.io/rtomczak/pen/wvjgbZq

CSS :

```css
p{
    font-size: 2em;
    background-color : red;
}

p#grosseur{
    font-weight: bold;
}

div.tailleFamilleOpaciteCentre{
    font-size: 150%;
    font-family: 'Times New Roman', Times, serif;
    opacity: 0.5;
    text-align: center;
}
```

HTML :

```
Un exemple de texte
<p>dans un paragraphe la taille est doublée</p>
<p id="grosseur">Essai avec font-weight: bold </p>
<div class="tailleFamilleOpaciteCentre">
    <ul>
        <ol>taille de 150%</ol>
        <ol>police Times Nem Roman</ol>
        <ol>opacité de 0.5</ol>
        <ol>le texte est centré</ol>
    </ul>
```

Résultat :

Un exemple de texte

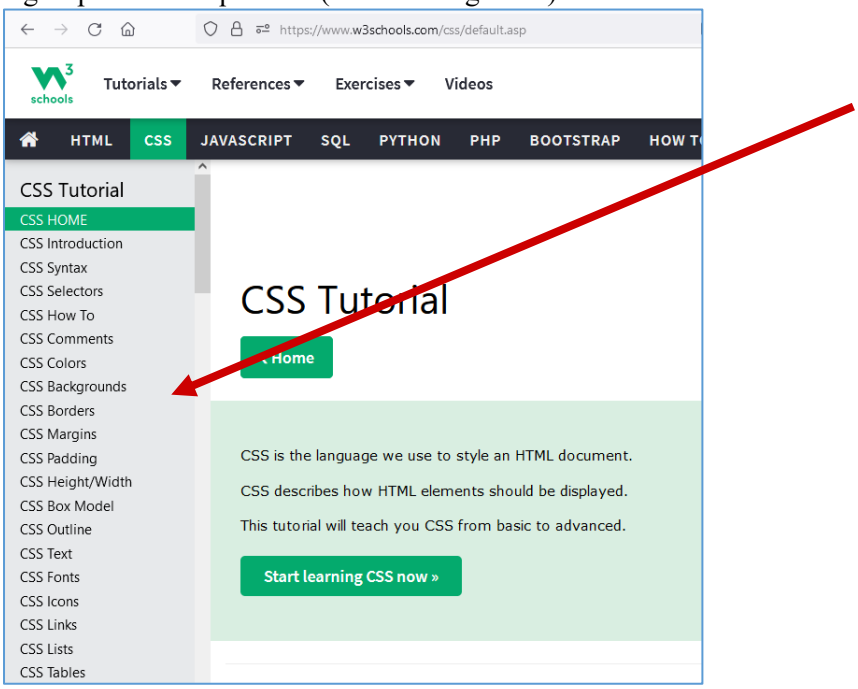
dans un paragraphe la taille est doublée

Essai avec font-weight: bold

taille de 150%
police Times Nem Roman
opacité de 0.5
le texte est centré

f) D'autres propriétés

Il existe tant de propriétés que je vous invite à les consulter sur https://www.w3schools.com/css/default.asp où ils sont regroupés thématiquement (colonne de gauche)



g) Conclusion

Vous avez vu l'incroyable puissance du CSS, le plus difficile est de trouvé la bonne propriété qui nous intéresse !!!

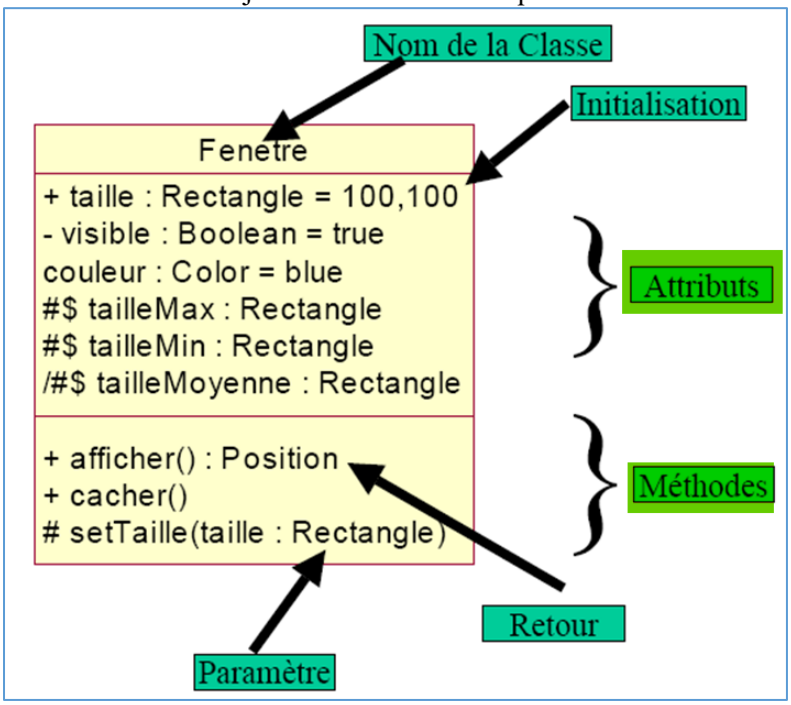
IV. Manipuler le DOM avec JS

1. Qu'est-ce qu'un objet ?

a) L'objet

Une classe est une description d'un ensemble d'objets ayant des propriétés communes.
Un objet est une instance d'une classe (une classe = plusieurs objets).

Les attributs d'un objet sont ces caractéristiques et une méthode est une fonction appartenant à la classe :



Dans la suite, nous parlerons d'objet, *window* mais surtout l'objet *document :*

- un exemple d'attribut est `document.body` qui est lui-même un objet. Repérez l'absence de parenthèses à la fin
- un exemple de méthode est `document.getElementById('monP');` Notez la présence d'un paramètre entre parenthèses.

2. Les objets window et document

L'objet *window* est un objet global qui représente la fenêtre dans le navigateur : JS est exécuté à partir de cet objet. L'appel est implicite, il n'est pas obligatoire de le spécifier, par exemple *alert* qui permet l'affichage d'une fenêtre, n'est pas une fonction comme on pourrait croire, mais une méthode de l'objet *window.*

```
// Ces deux appels sont équivalents
window.alert('Hello world!');
alert('Hello world!');
```

L'objet *document* est un objet fils de *window* c'est le plus utilisé car représente la page HTML (plus précisément la balise <html>).

3. Comprendre le DOM

Le DOM (*Document Object Model)* est une interface qui nous permet la programmation dans les documents HTML. Ainsi, le DOM sera utilisé par Javascript : par exemple ajouter un élément HTML, masquer un …

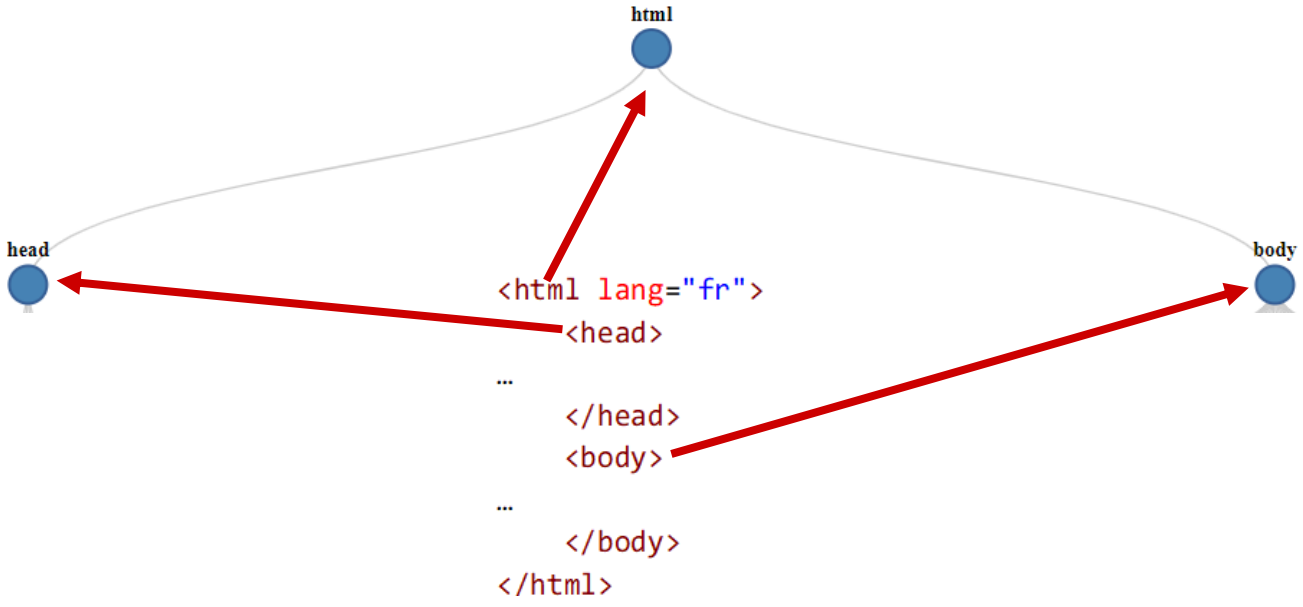
Le DOM d'une page HTML représente celle-ci souvent forme d'arbre structuré : des branches et des nœuds.

Soit le code HTML suivant :
Source : https://codepen.io/rtomczak/pen/NWMdgaO

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Premiers Pas</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <h1 id="msg">Hello World!</h1>
        <input type="text" value="Premiers pas avec Javascript" size="30" id="label">
        <br><br>
        <input type="radio" name="choix" value="1"  id="c1"> Choix N°1
        <input type="radio" name="choix" value="2" checked id="c2"> Choix N°2
        <input type="radio" name="choix" value="3" id="c3"> Choix N°3
        <br><br>
        <input type="password" value="password" id="pwd">
        <br><br>
        <input type="button" value="Ok" id="bp">
    </body>
    <script src="script.js"></script>
</html>
```

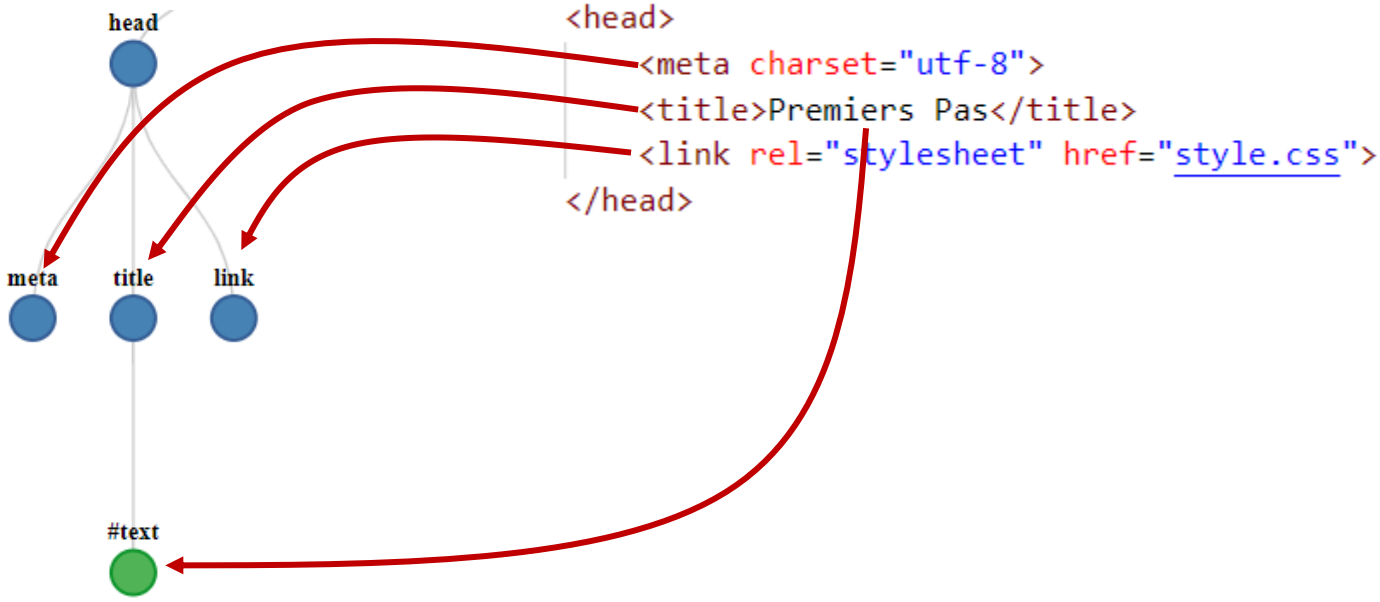
Construisons pas à pas le DOM :

a) La racine : la balise html



b) Base head

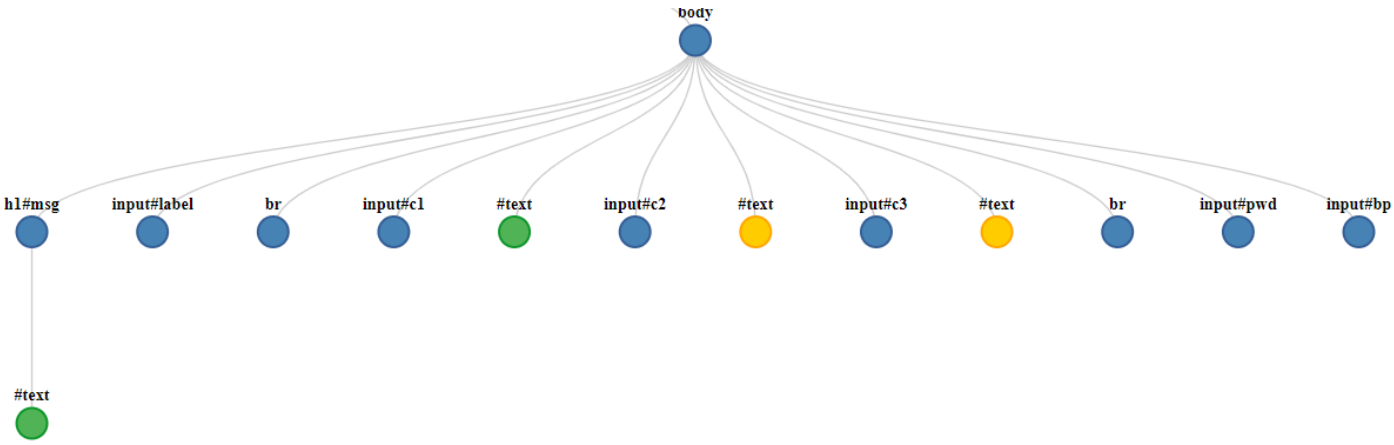
Ainsi la racine est la balise *html* puis deux branches vers *head* et *body*
Ensuite du côté gauche le code sous *head* la branche DOM correspondante est :



Dans le cas de la branche de gauche, les nœuds sont meta et link, le nœud du milieu *title* possède un feuille #text qui correspond au texte « Premiers pas »

c) Balise body

C'est bien sûr la balise dans laquelle il y a plus de contenu :




```
<body>
    <h1 id="msg">Hello World!</h1>
    <input type="text" value="Premiers pas avec Javascript" size="30" id="label">
    <br>
    <input type="radio" name="choix" value="1"  id="c1"> Choix N°1
    <input type="radio" name="choix" value="2" checked id="c2"> Choix N°2
    <input type="radio" name="choix" value="3" id="c3"> Choix N°3
    <br>
    <input type="password" value="password" id="pwd">
    <br>
    <input type="button" value="Ok" id="bp">
</body>
```

En détail cela donne :

h1#msg #text	`<h1 id="msg">Hello World!</h1>` L'`id` de `h1` se trouve juste après le **#text** correspond au texte `Hello World!`
input#label	`<input type="text" value="Premiers pas avec Javascript" size="30" id="label">` Idem, l'`id` de `input "label"` se trouve après le #
br	` `
input#c1	`<input type="radio" name="choix" value="1"  id="c1">`
#text	Le texte `Choix N°1`
input#c2	`<input type="radio" name="choix" value="3" id="c1">`
#text	`Choix N°2`
input#c3	`<input type="radio" name="choix" value="3" id="c3">`
#text	`Choix N°3`
br	` `

input#pwd	`<input type="password" value="password" id="pwd">`
input#bp	`<input type="button" value="Ok" id="bp">`

4. Utiliser le mode console

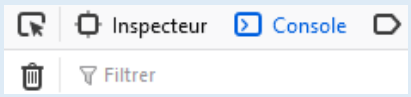
Vous avez deux possibilités :

- Vous suivez le code sur codepen. Dans ce cas il faut ouvrir la console.
- Si vous testez sur votre ordinateur le code, dans ce cas il faut ouvre les outils de développement pour Firefox ou Chrome (Touche F12). Revoyez ci nécessaire Accès aux outils de développement

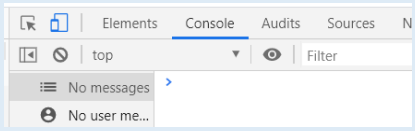
☝ Cette dernière méthode est celle préconisée pour plusieurs raisons :

- codepen permet de tester le code html sans se soucier des entêtes (body, head…) ou des noms des feuilles de style ou de javascript, ce que ne permet pas votre navigateur
- dans codepen il n'y uniquement le code se trouvant dans la balise body
- Utiliser un éditeur de texte qui vous facilitera la vie en complétant votre code ou en vous proposant des choix possibles (notepad++, Visual Studio …)
- Les outils de développement Firefox ou Chrome sont très puissants (touche F12)
- En TD et TP, vous allez devoir développer sur votre machine

Firefox

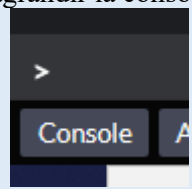


Chrome



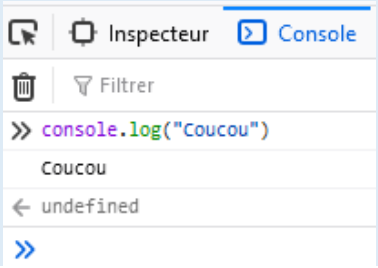
codepen

En bas à gauche, il faut peut-être agrandir la console

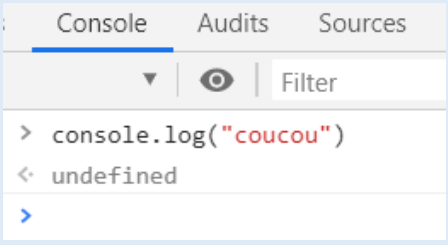


Testez en entrant la commande `console.log("coucou")` :

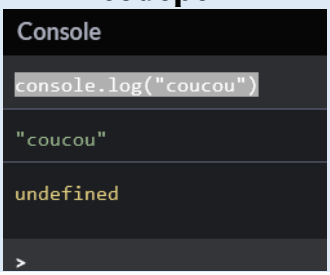
Firefox



Chrome



codepen



Les exemples sont souvent donnés avec Chrome dans cette partie.

5. Accéder aux éléments : getElementById(id)

Cette méthode est la plus utilisée !!!
Le fichier utilisé est Source : https://codepen.io/rtomczak/pen/NWMdgaO .

a) L'identifiant *id*

La valeur de l'attribut id définie de façon **unique** un élément d'une page HTML. Ainsi deux composants HTML ne pourront avoir le même *id.*

Par exemple, nous allons nommer *message* le texte qui sera mis en haut de la page web comme-ceci :

```
<h1 id="message">Hello World!</h1>
```

a) Fichier HTML

Reprenons l'exemple :


```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Premiers Pas</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <h1 id="message">Hello World!</h1>
        <input type="text" value="Premiers pas avec Javascript" size="30" id="label">
        <br><br>
        <input type="radio" name="choix" value="1"  id="c1"> Choix N°1
        <input type="radio" name="choix" value="2" checked id="c2"> Choix N°2
        <input type="radio" name="choix" value="3" id="c3"> Choix N°3
        <br><br>
        <input type="password" value="password" id="pwd">
        <br><br>
        <input type="button" value="Ok" id="bp">
    </body>
    <script src="script.js"></script>
</html>
```

b) getElementById(id)

La fonction Javascript ou plus précisément la méthode, *getElementById(id)* est l'une des plus utilisée pour la manipulation du DOM.
Ainsi il est possible de rendre dynamique, non plus sur le serveur, mais directement sur le navigateur.

Par exemple, prenons le bouton : 

, il s'appelle bp et sa valeur est « Ok », si vous entrez `document.getElementById("bp")`, l'affichage est `<input type="button" value="Ok" id="bp">`

```
> document.getElementById("bp")
<-   <input type="button" value="Ok" id="bp">
```

La commande renvoie l'élément identifié par l'id "bp".

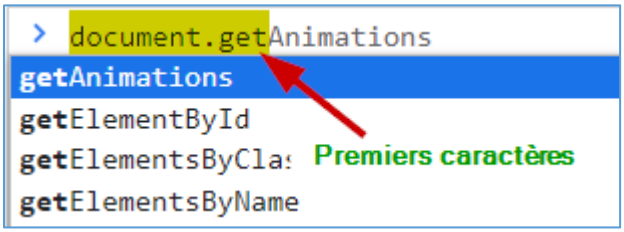
c) Modifier des éléments avec value

Ainsi pour modifier la valeur de cet élément, il suffit d'écrire :

```
document.getElementById("bp").value = "Cliquez sur moi"
```

Et le texte du bouton est modifié :

✋ Remarque : remarquez que lorsque vous écrivez dans la console, une liste de concordances vous est proposez. A l'aide des flèches et de la tabulation vous pouvez choisir une ligne.



Ainsi il est possible de modifier la zone de texte par `document.getElementById("label").value = "Coucou tout le monde"`

d) Autres attributs checked et innerHTML

Il est également possible de modifier d'autres attributs :

i. checked

Cet attribut permet de sélection un des boutons.

Le code html ressemble à

```
<input type="radio" name="choix" value="1"  id="c1"> Choix N°1
<input type="radio" name="choix" value="2" checked id="c2"> Choix N°2
<input type="radio" name="choix" value="3" id="c3"> Choix N°3
```

Avant	○ Choix N°1 ◉ Choix N°2 ○ Choix N°3
Javascript	`document.getElementById("c1").checked = true`
Après	◉ Choix N°1 ○ Choix N°2 ○ Choix N°3

Ce code permet de sélectionner le premier choix au lieu du second (bouton radio : un seul choix possible)

```
document.getElementById("c1").checked = true
```

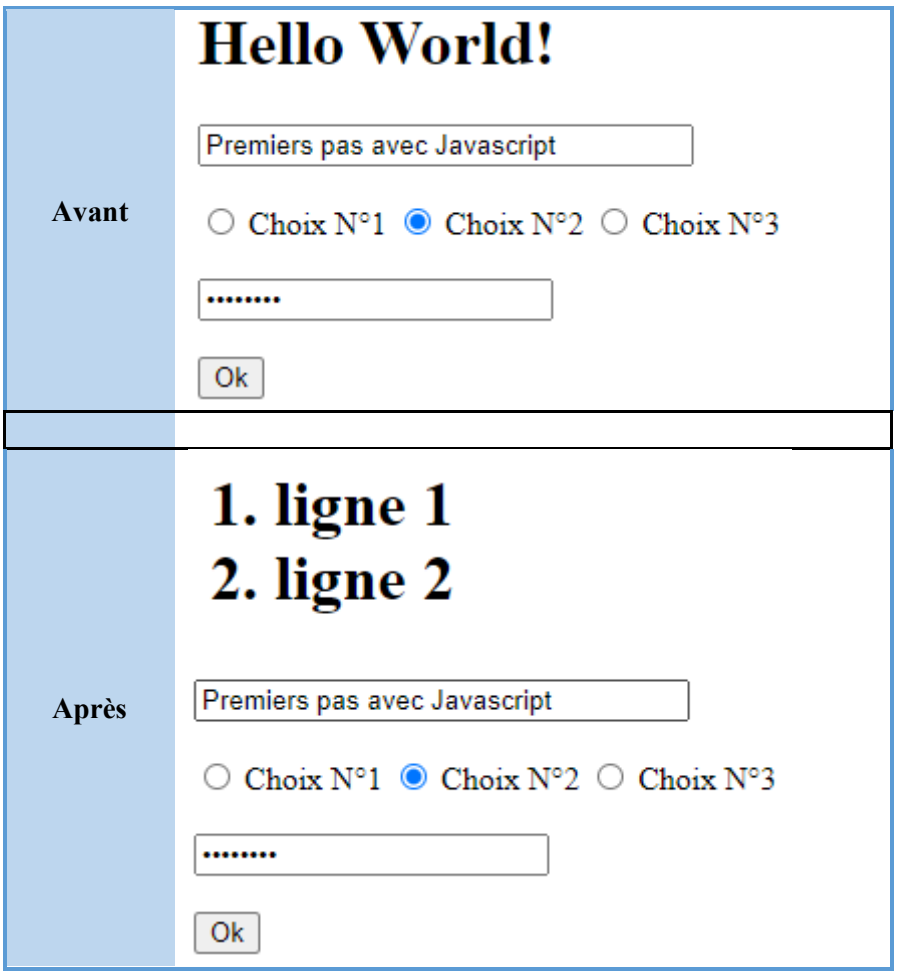
ii. innerHTML

La propriété innerHTML qui permet de récupérer le code HTML d'un élément sous forme de texte.

Ainsi cette commande va remplacer le code HTML dans msg :

```
document.getElementById("msg").innerHTML="<ol> <li>ligne 1</li> <li>ligne 2</li></ol>"
```

On a ainsi défini le code HTML `"<ol> <li>ligne 1</li> <li>ligne 2</li></ol>"` qui se trouve dans l'élément d'id `"msg"`.



6. Accéder aux éléments par le nom

A la place de l'ID, il est possible également de sélectionner par un nom (name) :
Dans notre exemple, nous avons trois boutons pourtant le même nom `choix` :

```
<input type="radio" name="choix" value="1"  id="c1"> Choix N°1
<input type="radio" name="choix" value="2" checked id="c2"> Choix N°2
<input type="radio" name="choix" value="3" id="c3"> Choix N°3
```

La commande `document.getElementsByName("choix")` renvoie un type qui s'appelle un *NodeList* ce qui signifie que c'est un nœud de type liste :

```
> document.getElementsByName("choix")
< ▼NodeList(3) [input#c1, input#c2, input#c3]
    ▶0: input#c1
    ▶1: input#c2
    ▶2: input#c3
      length: 3
    ▶[[Prototype]]: NodeList
```

Et nous avons bien une liste de trois éléments.
Pour plus de commodité, mettons-la dans une variable et affichons son premier élément, sa longueur, la valeur de l'élément d'indice 2, ainsi que son type :

Javascript

```
let radioChoix = document.getElementsByName("choix")
radioChoix[0]
radioChoix.length
radioChoix[2].value
radioChoix[2].type
```

Le résultat dans la console :

```
> let radioChoix = document.getElementsByName("choix")
<- undefined
> radioChoix[0]
<-   <input type="radio" name="choix" value="1" id="c1">
> radioChoix.length
<- 3
> radioChoix[2].value
<- '3'
> radioChoix[2].type
<- 'radio'
```

7. getElementsByTagName()

Cette méthode de *document* permet la récupération de tous les éléments HTML spécifié.
Par exemple, si on veut accéder à tous les éléments div :

```
document.getElementsByTagName('div')
```

```
> document.getElementsByTagName('div')
<- ▼HTMLCollection [div.formatage1]
     length: 1
   ▶0: div.formatage1
   ▶__proto__: HTMLCollection
```

On obtient une collection d'éléments comme un tableau.
Le caractère * permet de récupérer tous les éléments HTML :

```
document.getElementsByTagName('*')
```

```
> document.getElementsByTagName('*')
<- ▶HTMLCollection(16) [html, head, meta, title, link, body, h1, ul, li, li, li, p#bloc1, div.formatage1,
   p.formatage1, input, script, bloc1: p#bloc1]
```

Et on accède au deuxième élément, la balise *head* en mettant sa position (on commence à compter à zéro) entre crochets à la fin de la ligne :

```
document.getElementsByTagName('*')[1]
```

```
> document.getElementsByTagName('*')[1]
<- ▼<head>
     <meta charset="utf-8">
     <title>Premiers Pas</title>
     <link rel="stylesheet" href="CSS-JS-01.css">
   </head>
```

8. Manipulation des attributs

a) Récupérer un attribut avec getAttribute

Bien souvent, il y deux possibilités pour récupérer la valeur d'un attribut :

- Soit par `getAttribute()`
- Soit en le nommant directement

Par exemple pour :

HTML :

```
<input type="text" value="Premiers pas avec Javascript" size="30" id="label">
```

Javascript :

```
elt = document.getElementById("label")
elt.getAttribute("size") donnerait la valeur de l'attribut size donc 30
```

Ce qui est équivalent à

```
document.getElementById("label").getAttribute("size")
```

Console :

```
>> elt = document.getElementById("label")
<- ▶ <input id="label" type="text" value="Premiers pas avec Javascript" size="30">
>> elt.getAttribute("size")
<- "30"
>> document.getElementById("label").getAttribute("size")
<- "30"
```

b) Modifier un attribut avec setAttribute()

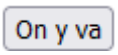
Prenons le bouton en bas de la page 

le code HTML est :

```
<input type="button" value="Ok" id="bp">
```

A l'aide de `setAttribute` on va modifier l'attribut `'value'` en `'On y va'`

```
document.getElementById("bp").setAttribute('value','On y va')
```

Le bouton est ainsi modifié : 

c) Les attributs accessibles directement

Souvent au lieu d'utiliser il est possible d'accéder directement à l'attribut.
Par exemple, pour récupérer la classe d'un élément, au lieu de :

```
unEnsemble[0].getAttribute("class"))
```

Il est possible d'utiliser directement `className`

```
unEnsemble[0].className
```

Même chose pour href,id etc..

9. Naviguer dans le DOM

a) Parcourir avec firstChild et lastChild

Comme leurs noms l'indiquent, ces attributs (pas des fonctions) permettent l'accès au premier ou dernier enfant.

i. Exemple 1 avec l'objet document

Pour toutes les pages web, le nœud body contient tous les éléments de la balise body :

JS :

```
document.body
```

Console :

```
>> document.body
<- ▼ <body>
     aLink: ""
     accessKey: ""
     accessKeyLabel: ""
     assignedSlot: null
   ▶ attributes: NamedNodeMap []
```

`document.body.innerHTML` permet de s'en convaincre :

```
>> document.body.innerHTML
<- "<h1 id=\"msg\">Hello World!</h1>\n<input
```

Affichons maintenant le premier enfant :

```
console.log(document.body.innerHTML)
console.log(document.body.firstChild)
```

```
>> console.log(document.body.firstChild)
   ▶ <h1 id="msg">
```

C'est bien le titre 1.

Et le dernier :

```
console.log(document.body.lastChild)
```

```
>> console.log(document.body.lastChild)
    ▶ <div id="grammalecte_menu_main_button_shadow_host" style="width: 0px; height: 0px;">
```

Vous l'avez peut-être remarqué, il n'est pas dans notre code HTML : c'est un div qui a été automatiquement ajouté lorsqu'on est dans le mode *Outils de développement.*

ii. Exemple 2 avec getElementById

Supposons que dans notre code, il existe un paragraphe dont l'id est monP :

```
let paragraphe = document.getElementById('monP');
```

Le premier enfant :

```
let premier = paragraphe.firstChild;
```

Et le dernier :

```
let dernier = paragraphe.lastChild
```

b) D'autres propriétés/attributs

- noteValue permet de récupérer la valeur (`let premier = paragraphe.firstChild.nodeValue;`)
- childNodes retourne un tableau contenant la liste des enfants (`paragraphe.childNodes;`)
- nextSibling et previousSibling sont deux attributs qui permettent d'accéder au nœud suivant ou au nœud précédent.

10. Manipuler le DOM

Nous allons voir ces méthodes assez rapidement. Elles seront développées en TD/TP.

a) Pour créer un élément : createElement

Tout débute par celle méthode qui permet de créer seulement un nœud DOM.

Par exemple (JS) :

```
nouveauDiv = document.createElement("div");
nouveauLien = document.createElement('a');
```

b) Affecter des attributs

i. Pour div

```
nouveauDiv.innerHTML = "<p>Je suis un <b>nouveau</b> div!</p>"
```

ii. Pour a

```
nouveauLien .id = 'lien1';
nouveauLien .href = 'https://www.tomczak.fr';
nouveauLien .title = 'Ancien site du prof !';
```

Ou bien :

```
nouveauLien .setAttribute('id', 'lien1');
nouveauLien .setAttribute('href', 'https://www.tomczak.fr');
nouveauLien .setAttribute('title', ''Ancien site du prof !');
```

c) Insérer un nœud appendChild et createTextNode

i. Balise avec du text

JS :

Pour rappel un nouvel élément à été créé puis on a ajouté un texte :

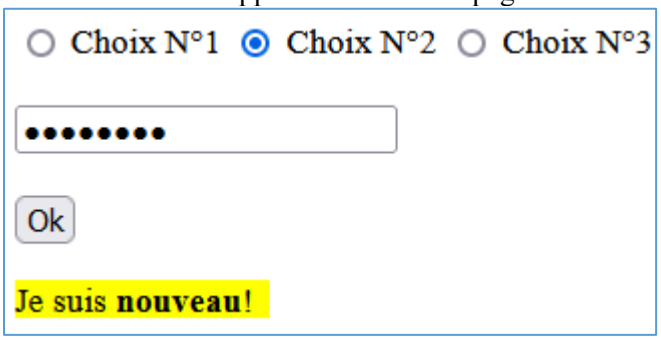
```
let nouveauDiv = document.createElement("div");
nouveauDiv.innerHTML = "<p>Je suis <b>nouveau</b>!</p>";
```

Maintenant, on va le mettre à la fin du body :

```
document.body.appendChild(nouveauDiv);
```

L'affichage :

Le nouvel élément apparait en bas de la page :



ii. Cas où il faut un context textuel

Lorsqu'on a créer un nœud, `appenChild()` insère ce nouveau nœud DOM « enfant » dans le « parent »

Par exemple (JS) :

Le nouvel élément est mis à la fin du document.

```
document.body.appendChild(nouveauDiv);
```

Et pour le lien, on va le mettre dans l'élément d'id `Fin`

```
document.getElementById("Fin").appendChild(nouveauLien);
```

Si vous observez la page avec cette dernière commande, rien ne se passe : c'est normal pour l'afficher il manque un élément DOM de type #text car il n'en possède pas au contraire du cas précédent où on avait ajouté du code HTML. Ce qui est appelé un nœud textuel et pour l'insérer rien de plus simple : une nouvelle commande `createTextNode`

JS :

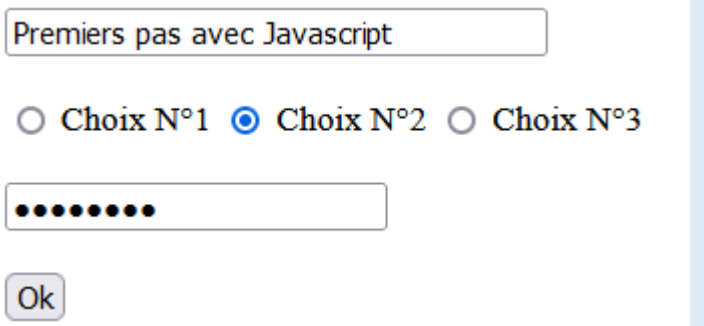
Création d'un lien textuel :

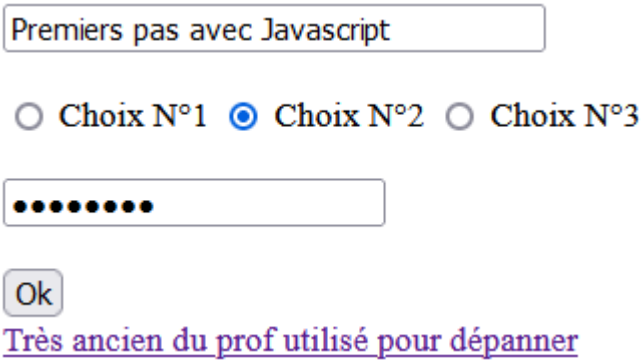
```
let texteLien = document.createTextNode("Très ancien du prof utilisé pour dépanner");
```

Et maintenant, on l'ajoute comme enfant à l'élément lien :

```
nouveauLien.appendChild(texteLien);
```

Affichage :

Avant	

Après	

d) Suppression avec removeChild

Et inversement, `removeChild` supprime un nœud.

A l'aide de `querySelector`, je recherche le tout premier lien puis je le supprime avec comme parent la div dont l'id est Fin:

JS :

```
let lien = document.querySelector("a");
document.getElementById("Fin").removeChild(lien)
```

Pour supprimer l'élément DIV je ne peux pas faire ainsi :

```
lien = document.querySelector('div');
document.body.removeChild(lien[1]);
```

Car il y a deux DIV, `querySelector` or renvoie le premier.

La solution est d'utiliser `querySelectorAll` ainsi :

```
let liens = document.querySelectorAll('div');
```

Dans `liens` il y a les deux div, je veux supprimer le second donc :

```
document.body.removeChild(liens[1]);
```

e) D'autres méthodes

i. insertBefore()

`appenChild()` insère un nœud DOM « enfant » dans le « parent tandis que `insertBefore()` l'insère avant le nœud parent :

ii. cloneNode()

Comme son nom l'indique, clone un élément pour l'insérer ailleurs par exemple.

iii. replaceChild()

Il remplace un enfant par un autre

iv. removeChild()

Pour supprimer un enfant.

v. hasChildNodes()

Vérifie la présence d'enfants.

vi. insertBefore() et insertAfter()

Insèrent avant ou après un enfant.

V. Manipuler le CSS avec JS

☝ Dans cette partie, au lieu d'utiliser la console, nous allons utiliser un fichier javascript qui va afficher dans la console.

1. Spécifier une classe avec getElementsByClassName

a) Récupérer un ensemble d'éléments par sa classe

Cette méthode renvoie un ensemble d'éléments spécifié par sa classe CSS.

script.js

```
let elements = document.getElementsByClassName("formatage1");
console.log(elements) // Deux éléments
console.log(elements[0]) //Le premier est span
console.log(elements[1]) //Le second p
```

HTML

```
<ul>
    <li>1er élement</li>
    <li>Second</li>
    <li>Troisième</li>
</ul>
<p id="bloc1">Bloc d'indentificateur bloc1 </p>
<span class="formatage1"> Span avec classe formatage1 </span>
<p class="formatage1">Comme avec span mais la couleur est maintenant cyan </p>
```

L'affichage dans la console

Il y a deux éléments de classe formatage1:

```
▼HTMLCollection(2) i
   length: 2
  ▶0: div.formatage1
  ▶1: p.formatage1
  ▶__proto__: HTMLCollection

▼<div class="formatage1">
    " Div avec classe formatage1
          "
  ▶<p class="formatage1">…</p>
    <!-- Mis à la fin pour afficher une première fois -->
    <script src="CSS-JS-01.js"></script>
  </div>

▼<p class="formatage1">
    "Comme avec div mais la couleur est maintenant cyan "
    <input type="submit" value="Un nouveau bouton">
  </p>
```

b) En modifier un

Ensuite comme avec `getElementById` il est possible de modifier ses attributs : HTML, valeur …

Par exemple, nous allons ajouter un bouton dans la div dont la classe est `formatage1`:
On récupère le code HTML se trouvant dans l'élément 1 :

```
let ancienCode = elements[1].innerHTML // L'ancien code html
```

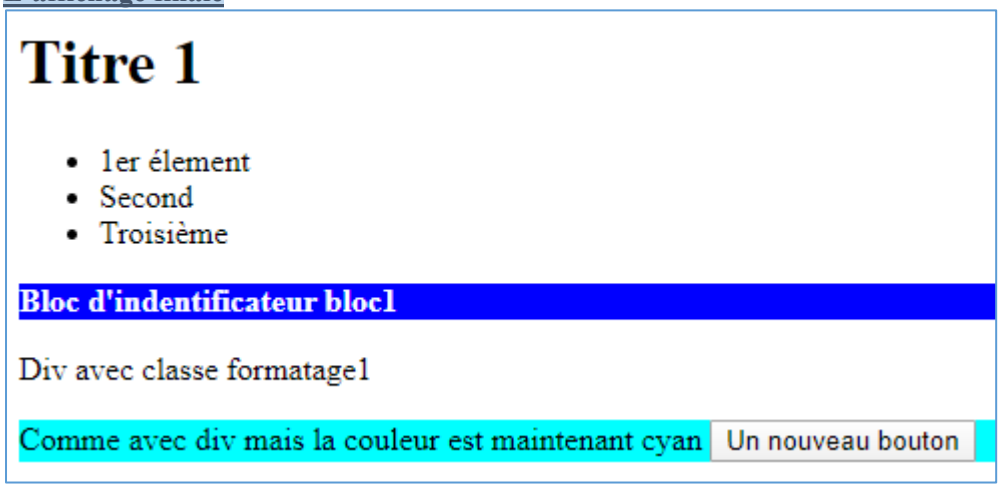
Puis on y ajoute un bouton :

```
elements[1].innerHTML =ancienCode + '<input type="submit" value="Un nouveau bouton">';
```

Le code JS

```
let ancienCode = elements[1].innerHTML // L'ancien code html
elements[1].innerHTML =ancienCode + '<input type="submit" value="Un nouveau bouton">';
```

L'affichage finale



2. N'importe quel sélecteur querySelector

a) Le premier avec querySelector()

Les sélecteurs sont utilisés dans le fichier CSS
Pour rappel un sélecteur peut-être :

Type de sélecteur	Définition dans le css	Éléments concernés
D'élément	`p {}` `ul{}`	`<p>` `<ul>`
De classe	`.formatage1 {}`	Tous les éléments ayant comme attribut `class="formatage1"`
D'id	`#bloc1 {}`	L'élément ayant comme attribut `id="bloc1"`

D'autres exemples :	
`p.formatage1 {}`	Tous les paragraphes ayant comme classe `formatage1`
`h1,p {}`	Tous les `<h1>` et `<p>`
`input[type=text] {}`	Tous les éléments `<input type="texte">`
`#bloc1 .formatage1 span {}`	Les balises `<span>` contenu dans les classes `.formatage1` elles-mêmes contenues dans un élément dont l'identifiant est `bloc1`

Simple mais efficace, c'est une méthode très intéressante, elle retourne le **premier élément** spécifie par le sélecteur se trouvant le css.

Dans le css, nous avons l'id bloc1

```
#bloc1 {
  font-weight: bold;
  background-color: blue;
  color: white;
  width: 40%;
}
```

Javascript va aller retourner le premier élément (et le seul)

Cet élément c'est `<p id="bloc1">Bloc d'indentificateur bloc1 </p>`

```
let unSelecteur = document.querySelector("#bloc1")
console.log(unSelecteur)
```

Dans la console nous avons donc :

```
<p id="bloc1">Bloc d'indentificateur bloc1 </p>
```

b) Tous avec querySelectorAll()

Au contraire de `querySelector` cette méthode retourne **tous les éléments** sous forme de nodeList.

Dans notre css nous avons une classe qui s'appelle `formatage1` :

```
.formatage1 {
  font-weight: bold;
  background-color: yellow;
  color: black;
}
```

La commande CSS :

Tous les éléments contenant le sélecteur css `.formatage1` seront retournés par `querySelectorAll(".formatage1")`

Et

```
let unEnsemble = document.querySelectorAll(".formatage1")
console.log(unSelecteur)
```

Affichera dans la console :

```
▼NodeList(2) [div.formatage1, p.formatage1]
    length: 2
  ▶0: div.formatage1
  ▶1: p.formatage1
  ▶__proto__: NodeList
```

Ce qui correspond à un *nodeList* composé de deux éléments.

3. Tant d'autres …

…