

## Extrait du programme

### THÈME : INTERACTIONS ENTRE L'HOMME ET LA MACHINE SUR LE WEB

- **Contenus :**

Modalités de l'interaction entre l'homme et la machine

Événements

- **Capacités attendus :**

Identifier les différents composants graphiques permettant d'interagir avec une application Web.

Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.

- **Commentaires :**

Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.



- **Contenus :**

Interaction avec l'utilisateur dans une page Web

- **Capacités attendus :**

Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page Web.

- **Contenus :**

Interaction client-serveur.  
Requêtes HTTP, réponses du serveur

- **Capacités attendus :**

Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre.  
Distinguer ce qui est mémorisé dans le client et retransmis au serveur.  
Reconnaître quand et pourquoi la transmission est chiffrée.

- **Commentaires :**

Il s'agit de faire le lien avec ce qui a été vu en classe de seconde et d'expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP.

- **Contenus :**

Formulaire d'une page Web

- **Capacités attendus :**

Analyser le fonctionnement d'un formulaire simple.  
Distinguer les transmissions de paramètres par les requêtes POST ou GET.

- **Commentaires :**

Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.



### a) Liens pour aller plus loin

Le site *MDN web docs* de Mozilla pour tout apprendre sur le développement web :

- la page principale : <https://developer.mozilla.org/fr/docs/Apprendre>
- la page sur le HTML <https://developer.mozilla.org/fr/docs/Apprendre/HTML>
- la page sur CSS <https://developer.mozilla.org/fr/docs/Apprendre/CSS>
- la page JavaScript <https://developer.mozilla.org/fr/docs/Apprendre/JavaScript>

Un ensemble de cours complet en ligne : <https://www.pierre-giraud.com/>. On peut y trouver du HTML, CSS, du JavaScript, du PHP, MySQL, du Python et même plus encore (mais je dois avouer que je n'ai pas encore testé)

- pour le HTML et CSS : <https://www.pierre-giraud.com/html-css-apprendre-coder-cours/>
- pour le JavaScript : <https://www.pierre-giraud.com/javascript-apprendre-coder-cours/>

### b) Liens vers un éditeur HTML+CSS+JavaScript en ligne

<https://jsfiddle.net/> ou encore <https://jsbin.com/>.

## I. Introduction

### a) Web $\neq$ Internet

Internet, c'est le réseau. Le Web, c'est une application utilisant Internet. Comme tant d'autres :

- les e-mails ;
- FTP ;
- le pair-à-pair ;
- le streaming ;
- ...

On confond souvent les deux puisque l'on passe souvent par le Web pour accéder aux autres applications.

### b) Le Web, c'est quoi ?

Le World Wide Web est un ensemble de fichiers (pages, images, sons, vidéos...) reliés par des liens hypertextes ... accessibles par l'internet.

### c) Une petite histoire de l'Internet

- 1966 : Conception d'ARPANET, l'ancêtre de l'Internet par Lawrence G. Roberts
- 1972 : Ray Tomlinson met au point la première application importante : le courrier électronique
- 1991 : Tim Berners-Lee invente les notions d'HTML, de navigateur et d'HTTP
- 1993 : Le Web passe dans le domaine public
- 1994 : Yahoo
- 1995 : Amazon
- 1998 : Google
- 2001 : Wikipedia
- 2004 : Facebook
- 2005 : Youtube
- 2010 : HTML5

Sur le lien suivant <http://info.cern.ch/hypertext/WWW/TheProject.html> vous pouvez retrouver la toute première page Web de l'histoire !

## World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#) , [Policy](#) , November's [W3 news](#) , [Frequently Asked Questions](#) .

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#) , etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) , [X11 Viola](#) , [NeXTStep](#) , [Servers](#) , [Tools](#) , [Mail robot](#) , [Library](#) )

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.

### d) Comment accéder au Web ?

- On utilise un navigateur (Firefox, Chrome, Edge, Safari...), appelé client.
- On rentre une URL, par exemple <http://info.cern.ch/hypertext/WWW/TheProject.html>
- Le navigateur utilise le protocole HTTP (HyperText Transert Protocol), ou HTTPs, pour obtenir la page.
- L'ordinateur contacté, appelé serveur, génère la page.

- Le navigateur reçoit puis interprète le code HTML.
- Il demande les autres fichiers nécessaires (images, sons, scripts...).
- Il affiche la page.
- On parle de modèle client-serveur.

### e) Principe d'URL

Un URL (pour *Uniform Resource Locator*) est composée de 3 parties : comment, où et quoi.

<http://info.cern.ch/hypertext/WWW/TheProject.html>

- **comment** : on utilise le protocole **HTTP**
- **où** : on se connecte à l'ordinateur **info.cern.ch**
- **quoi** : on veut le fichier **TheProject.html** qui est dans le dossier **WWW** qui est dans le dossier **hypertext** qui est la base du site internet.

## II. Le HTML, CSS et JavaScript

Le **HTML** (*Hypertext Markup Language*) est un langage de balisage qui sert à représenter les pages Web (WWW : *World Wide Web*).

Le rôle du **HTML** est uniquement de donner la structure d'une page Web. C'est pour cela qu'on l'associe généralement à

- des feuilles de style **CSS** (*Cascading Style Sheets*) pour la mise en page (la présentation, les couleurs, le design) ;
- des scripts en langage **JavaScript** pour obtenir un comportement dynamique

Les langages **HTML** et **CSS** font partie des langages de description. Ils se contentent de décrire ce qu'il faut afficher et comment. Au contraire, le JavaScript est un langage de programmation et c'est lui qui nous permettra de dynamiser notre site.

## III. HTML



Le **HTML** (*HyperText Markup Language*) est un langage de programmation qui a fait son apparition en 1991. Il permet de créer des sites web. Dans ce cours, nous utiliserons la dernière version, le HTML5.

Pour écrire le code permettant de créer un site web, il suffit d'utiliser le **Bloc-notes** !... et oui ce petit logiciel ridicule. C'est pour dire à quel point le HTML est simple **MAIS** efficace.

Il existe aussi des logiciels beaucoup plus puissants mais aussi beaucoup plus complexes à utiliser. Dans notre cas, nous utiliserons le logiciel **NotePad++**<sup>1</sup> ou le logiciel **Brackets**<sup>2</sup> qui ressemble à un **Bloc-notes** avec quelques fonctionnalités supplémentaires. La principale ? La coloration syntaxique.

```

essais - Bloc-notes
Fichier Edition Format Affichage ?

<!DOCTYPE html>
<html>
  <head>
    <!-- En-tête de la page -->
    <meta charset="utf-8" />
    <title>Mon super site Web</title>
  </head>
  <body>
    <!-- Corps de la page -->
    <h1>ça c'est du titre</h1>
    <p>Bienvenue sur mon super site web !<br>
    Il est vraiment super !</p>
    <p>Qu'en pensez-vous ?</p>
  </body>
</html>

```

```

*C:\Users\Laurent Chaudet\Documents\Math\...
Fichier Édition Recherche Affichage Encodage Langage Paramétrage
Macro Exécution Compléments Documents ?
essais.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- En-tête de la page -->
5     <meta charset="utf-8" />
6     <title>Mon super site Web</title>
7   </head>
8   <body>
9     <!-- Corps de la page -->
10    <h1>ça c'est du titre</h1>
11    <p>Bienvenue sur mon site web !<br>
12    Il est vraiment super !
13    </p>
14    <p>Qu'en pensez-vous ?</p>
15  </body>
16 </html>
Ln: 18 Col: 8 Sel: 0|0 Dos/Windows UTF-8 INS

```

1. Que l'on peut télécharger gratuitement en cliquant sur le lien suivant. Vous choisirez la dernière version.

<https://notepad-plus-plus.org/downloads/>

Vous pouvez choisir la version pour votre ordinateur (**Installer**) ou la version portable pour votre clef USB **Portable (zip)**.

2. On peut le télécharger gratuitement en cliquant sur le lien suivant :

<https://brackets.io/>

Pour la version portable, à installer sur votre clef USB, la solution se trouve à l'adresse suivante :

<https://sourceforge.net/projects/bracketsportable/files/1.13.0.2/Brackets%201.13%20version%202.zip/download>.

Enfin, nous utiliserons un navigateur pour admirer le résultat de notre code. Chaque navigateur ayant ses spécificités, il est conseillé, avant de poster un site web, de le tester avec les navigateurs les plus courants (Google Chrome, Mozilla Firefox, Safari, Edge) et ainsi de s'assurer que votre travail sera lisible ou que la présentation choisie sera respectée pour la très grande majorité des utilisateurs.

## IV. Ma première page Web

Ouvrir le logiciel **NotePadd++** ou **Brackets** et recopier le code suivant :


```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- En-tête de la page -->
5     <meta charset="utf-8" />
6     <title>Mon super site Web</title>
7   </head>
8
9   <body>
10    <!-- Corps de la page -->
11    <h1>ça c'est du titre</h1>
12    <p>Bienvenue sur mon site web !<br />
13    Il est vraiment super !
14    </p>
15
16    <p>Qu'en pensez-vous ?</p>
17  </body>
18 </html>
```

Sauvegarder ensuite ce travail en précisant bien l'extension **.html**

On pourra par exemple l'appeler **essais.html**

Vous devrez remarquer que votre éditeur (**NotePad++** ou **Brackets**) reconnaît alors le langage **HTML** et colorie les mots clefs.

Il ne reste plus qu'à ouvrir le fichier avec un navigateur en double cliquant sur l'icône du fichier **essais.html**

Nom	Modifié le	Type	Taille
 essais	13/03/2018 17:20	Firefox HTML Doc...	1 Ko

On obtient alors le résultat suivant :



## V. Des explications

Tout d'abord, le langage **HTML** est ce qu'on appelle un langage à balises, c'est-à-dire que (presque) toutes les commandes vont être encadrées par une balise ouvrante (par exemple `<html>`) et une balise fermante (par exemple `</html>`).

Les balises ouvrantes sont du style `<...>` et les balises fermantes `</...>`

Mais rentrons un peu plus dans le détail :

**LIGNE 1** : Le doctype `<!DOCTYPE html>` est essentiel. C'est la balise pour indiquer qu'il s'agit d'un document écrit en **HTML5**.

**LIGNE 2** et **LIGNE 18** : La balise ouvrante `<html>` indique le début du document **HTML** et la balise fermante `</html>` indique la fin.

### a) L'en-tête du document

```

3     <head>
4         <!-- En-tête de la page -->
5         <meta charset="utf-8" />
6         <title>Mon super site Web</title>
7     </head>

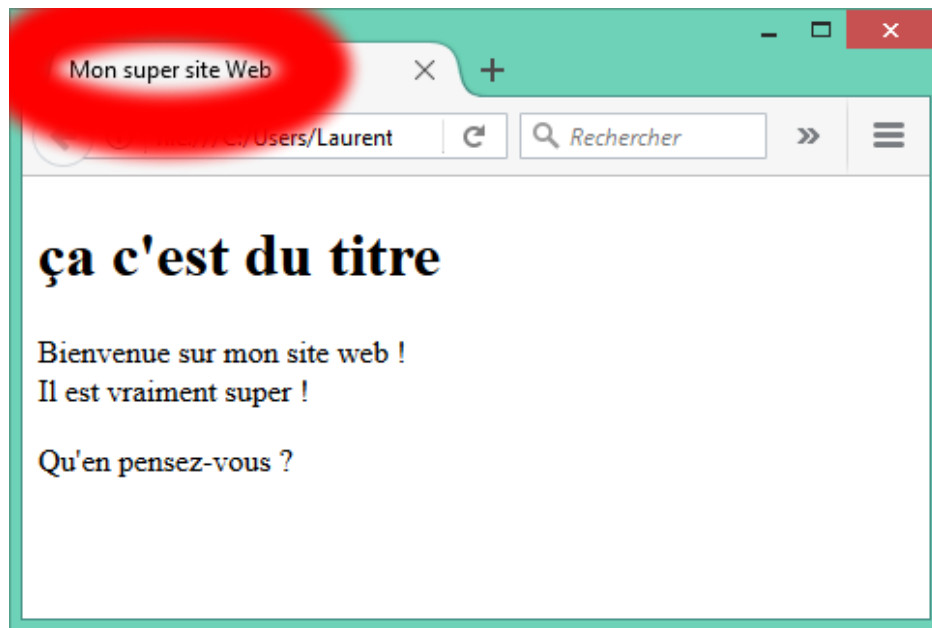
```

**LIGNE 3** et **LIGNE 7** : Entre les balises `<head>` et `</head>` nous avons **l'en-tête** du document. Dans l'en-tête nous donnons toutes les options et caractéristiques de notre page web.

**LIGNE 5** : La balise `<meta charset="utf-8" />` indique l'encodage utilisé pour votre fichier HTML. C'est ce qui permet d'afficher les caractères spéciaux. Dans notre cas, l'encodage choisi est **utf-8** car il permet d'afficher pratiquement tous les symboles de toutes les langues.

Si on supprime cette ligne, les caractères spéciaux (sur cette page, il s'agira du « ç ») ne seront pas bien interprétés par le navigateur (vous pouvez faire l'essai).

**LIGNE 6 :** Entre les balises `<title>` et `</title>` nous indiquons le titre de la page web.  
C'est ce titre que nous retrouvons dans l'onglet du navigateur.



## b) Le corps du document

```

9      <body>
10         <!-- Corps de la page -->
11         <h1>ça c'est du titre</h1>
12         <p>Bienvenue sur mon site web !<br />
13         Il est vraiment super !
14         </p>
15
16         <p>Qu'en pensez-vous ?</p>
17     </body>

```

**LIGNE 9** et **LIGNE 17** : Entre les balises `<body>` et `</body>` nous avons **le corps** du document.  
C'est la partie principale de la page. Tout ce que nous écrivons (ou presque) sera affiché sur le navigateur.

**LIGNE 11** : Entre les balises `<h1>` et `</h1>` on écrit un titre.  
Il existe plusieurs niveau d'importance d'un titre :

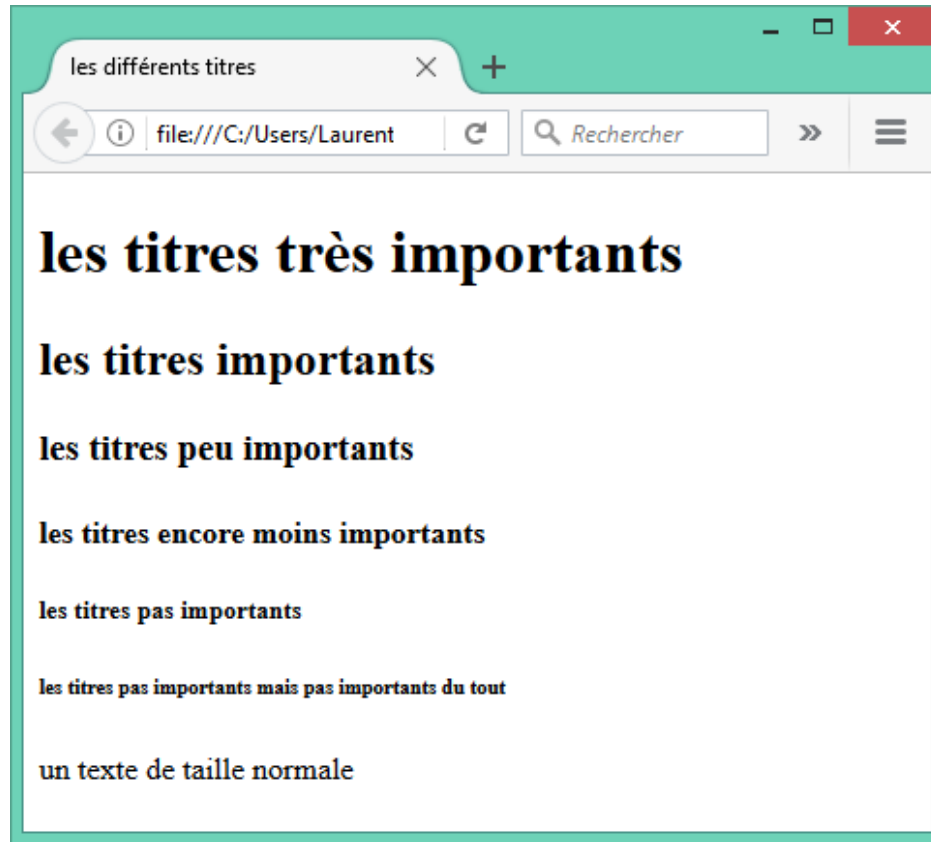
```

<h1> les titres très importants </h1>
<h2> les titres importants </h2>
<h3> les titres peu importants </h3>
<h4> les titres encore moins importants </h4>
<h5> les titres pas importants </h5>
<h6> les titres pas importants mais pas importants du tout </h6>
un texte de taille normale

```

ce qui donne le résultat suivant :





**LIGNE 12** et **LIGNE 14** : Entre les balises `<p>` et `</p>` nous avons le texte d'un paragraphe.

Nous retrouvons les mêmes balises à la ligne 16. Entre chaque paragraphe le navigateur effectue un saut avec un retour à la ligne.

**LIGNE 12** : Nous avons aussi la balise `<br />`. Elle est toute seule, c'est que l'on appelle une balise orpheline (on trouve aussi le terme d'*autofermante*).

Cette balise sert à sauter une ligne, tout en restant dans le même paragraphe.

**LIGNE 10** : Nous avons une balise particulière : `<!-- Corps de la page -->`.

Cette balise permet de faire des commentaires qui ne seront pas interprétés par le navigateur. Ces commentaires sont malgré tout très utiles pour pouvoir se rappeler de ce que nous voulions faire sur notre page Web.

Nous avons utilisé la même balise dans l'en-tête, à la ligne 4 : `<!-- En-tête de la page -->`.

## VI. D'autres balises utiles

Le site <https://www.w3schools.com/tags/default.asp> donne toutes les balises référencées.

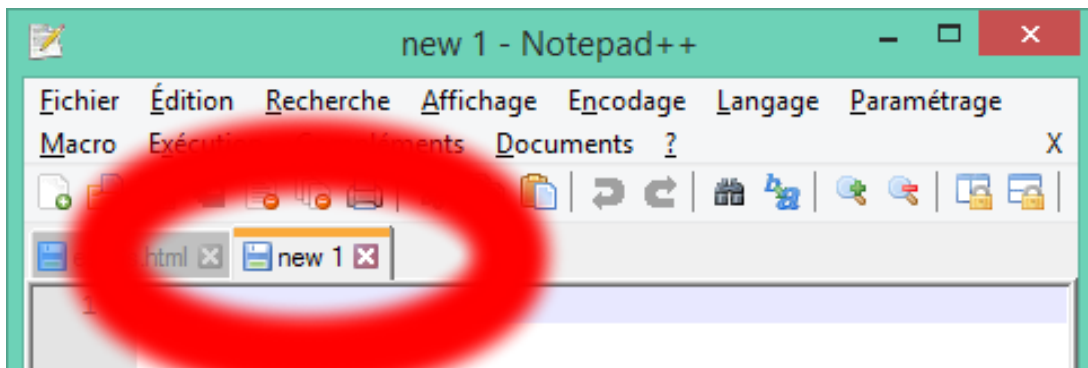
Utilisez-le pour remplir le tableau ci-dessous :

Balise	Description	Exemple d'utilisation
<!-- -->	Commentaire	<!-- un commentaire -->
<p>	Paragraphe	<p>un paragraphe.</p>
<h1> à <h6>		
<strong>		
<em>		
<sub>		
<sup>		
<ul> <li>		
<ol> <li>		
<table> <tr> <th> <td>		
<a>		
<img>		
<hr>		

## VII. Créer des liens

C'est l'intérêt principal du langage HTML, créer des liens entre les différentes pages. Enfin, nous allons voir comment procéder.

Pour commencer, créer un nouveau fichier sur **NotePad++** ou sur **Brackets**. Un nouvel onglet apparaît.

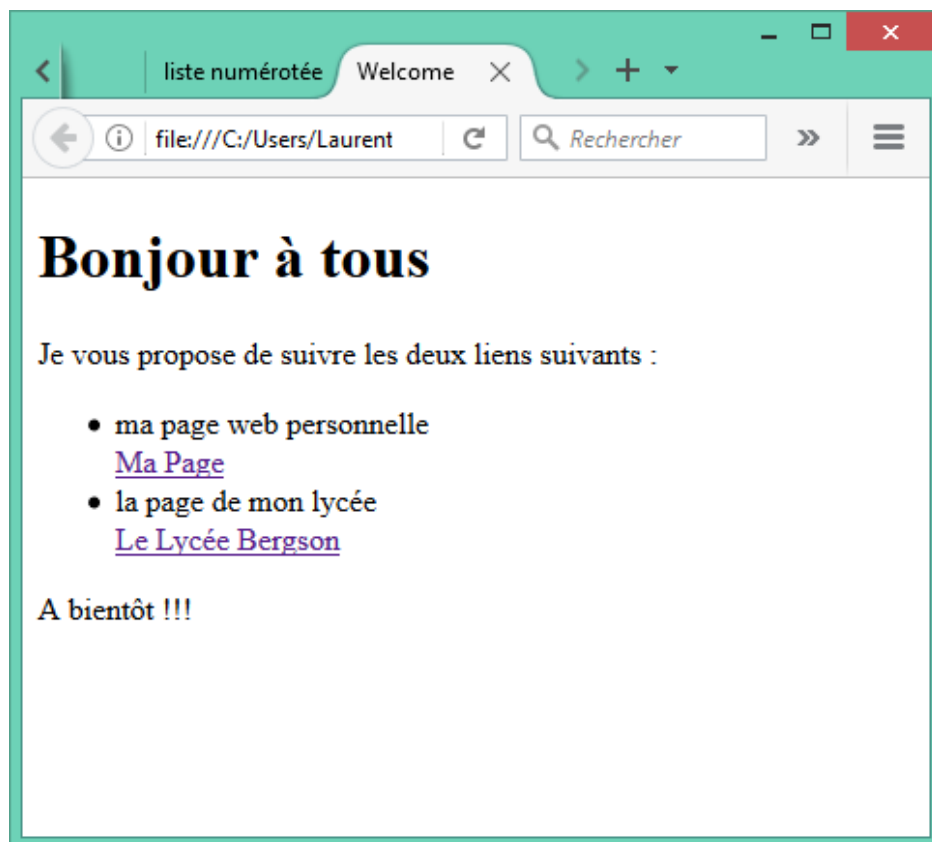


Avec ces onglets, nous pouvons avoir un accès rapide à toutes les pages de notre projet.

Recopier le code suivant et sauvegarder le sous le nom `maPage.html` dans le même répertoire que le fichier précédent :

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- En-tête de la page -->
5     <meta charset="utf-8" />
6     <title>Welcome</title>
7   </head>
8
9   <body>
10    <!-- Corps de la page -->
11    <h1>Bonjour à tous</h1>
12    <p>
13      Je vous propose de suivre les deux liens suivants :
14    <ul>
15      <li>
16        ma page web personnelle <br />
17        <a href="essais.html"> Ma Page </a>
18      </li>
19
20      <li>
21        la page de mon lycée <br />
22        <a href="http://bergson.paysdelaloire.e-lyco.fr"> Le Lycée Bergson </a>
23      </li>
24    </ul>
25    A bientôt !!!
26  </p>
27 </body>
28 </html>
```

Nous obtenons le résultat suivant :



Les liens hypertextes sont écrits aux lignes 17 et 22, le reste a déjà été vu précédemment et vous devriez comprendre les balises qui ont été utilisées.

Voici quelques explications :

```
17      <a href="essais.html"> Ma Page </a>
```

**LIGNE 17 :** La balise ouvrante `<a href="essais.html">` indique l'adresse du lien hypertexte.

Nous avons écrit le nom du premier fichier créer : `essais.html`.

On écrit ensuite le texte **Ma Page** qui sera visible sur le navigateur. Ce texte sera souligné et écrit en bleu (si on n'a jamais visité le lien) ou en violet (si on a déjà visité le lien).

On ferme avec la balise fermante `</a>`.

```
22      <a href="http://bergson.paysdelaloire.e-lyco.fr"> Le Lycée Bergson </a>
```

**LIGNE 22 :** C'est ici le même principe que précédemment. La différence est que nous avons créer un lien vers un site extérieur à celui de notre propre site web.

## VIII. Afficher des images

Pour afficher des images, il est conseillé de placer toutes les images dans un même dossier. Par commodité, on l'appellera **images**. On utilisera ensuite la **balise orpheline** :

```

```

Explications :

- `img src="images/stallman.jpg"` signifie que l'on va chercher l'image `stallman.jpg` dans le dossier `images`, pour ensuite l'afficher.
- `alt="Richard Stallman : le père du logiciel libre"` est la description de la photo. Ce n'est pas obligatoire, mais est fortement conseillé. En effet si la connexion internet est lente, on affichera d'abord la description le temps que l'image soit téléchargée puis affichée. Cette description sera aussi utile pour les mal-voyants qui ont des liseuses vocales. Ainsi, les personnes mal voyantes savent qu'il y a une image et connaissent sa description.



## IX. Un outil de vérification de la syntaxe

Le navigateur n'étant pas un débogueur, si votre code est faux, vous n'aurez pas de message d'erreur. Le navigateur fera ce qu'il peut pour afficher ce qu'il a compris, rien de plus<sup>3</sup>. Pour vérifier que votre page Web est conforme aux spécifications **HTML5**, rendez-vous sur le site du **W3C** (*World Wide Web Consortium*) : <http://validator.w3.org>.

Pour une page Web locale (pas encore publiée sur le Web) :

Validate by File Upload → Check

S'il y a des erreurs, elles vous seront indiquées, avec des explications (en anglais, of course).

Conseil d'ami : vérifier et corriger systématiquement vos pages Web avec cet outil.

C'est contraignant au début, mais cela vous fera prendre rapidement de bonnes habitudes.



## X. CSS

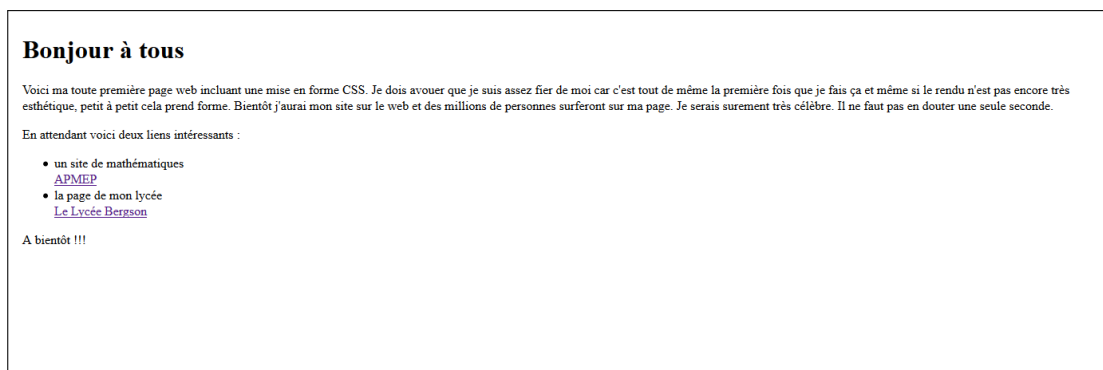


Nous l'avons vu, le **HTML** est le langage qui permet de programmer le contenu d'un site web. Mais il faut bien le dire le résultat est assez décevant. En effet, il nous manque le **CSS** (*Cascading Style Sheets*) qui va s'occuper de la mise en forme de la page web (taille des caractères, couleurs, ...). Le CSS est apparu un peu plus tard que le HTML, en 1996. La version utilisée actuellement est le CSS3.

Pour écrire le code, nous utiliserons à nouveau le logiciel gratuit **NotePad++** ou **Brackets** en sauvegardant le fichier avec l'extension **.css**.

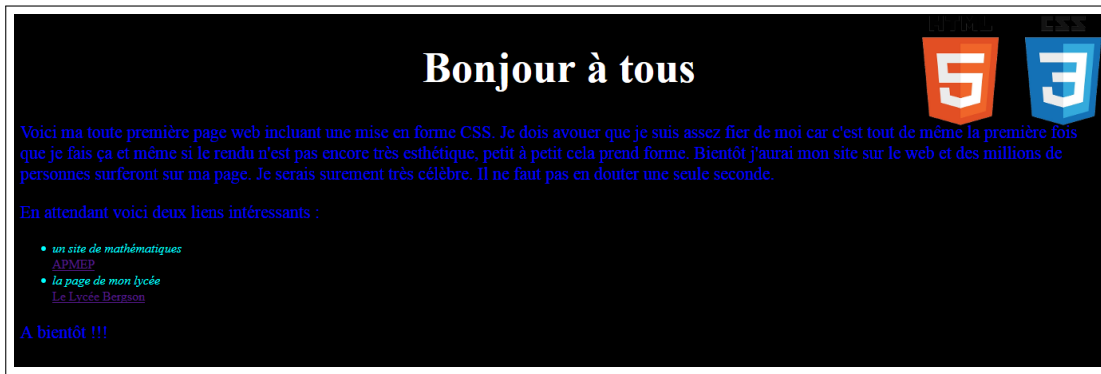
Voici un fichier HTML interprété par un navigateur sans, puis avec un fichier CSS.

Sans :



3. D'un autre côté, lorsque vous interprétez un programme écrit en Python, s'il y a une erreur, le programme ne s'exécute pas. Imaginez si c'était la même chose avec votre navigateur, si le code est faux, il ne se passerait rien... Au moins, on a quelque-chose.

Avec :



## a) Le CSS

Pour avoir une unité graphique de notre site web, le fichier `.css` informe le navigateur (Safari, Mozilla Firefox, Chrome, Edge, ...) de la couleur du fond, de la taille des caractères, de la façon d'afficher les titres, de la façon d'afficher les paragraphes, ... Pour chaque fichier `.html`, il suffit de préciser dans l'en-tête (entre les balises `<head>` et `</head>`) qu'il faut aller lire le fichier `.css` et respecter ses instructions. C'est ce qui est fait ci-dessous à la ligne 6 avec le code suivant :

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

On indique alors que la mise en forme de cette page HTML est gérée par un fichier appelé `monStyle.css`. L'intérêt majeur de cette technique réside dans le fait que sur l'ensemble de votre site Web vous avez une certaine unité graphique, les titres étant toujours de la même couleur, de la même taille, ..., même chose pour les liens hypertextes, les paragraphes, ... Plutôt que de le dire à chaque fois, on l'écrit une fois pour toute dans un fichier à part.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- En-tête de la page -->
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="monStyle.css" />
7     <title>Mon super site Web</title>
8   </head>
9
10  <body>
11    <!-- Corps de la page -->
12    (...)
13  </body>
14 </html>
```

Il ne reste plus qu'à créer ce fichier `monStyle.css`.

```
1 h1
2 {
3   color : black; /*les titres h1 sont en noirs*/
4   font-size :36px; /*titres de 40 pixels*/
5 }
6 p
7 {
8   color : blue; /* les paragraphes sont en bleu */
9   font-size : 14 px; /* paragraphes de 14 pixels */
10 }
```

Sauvegardez le document sous le nom `monStyle.css` puis ouvrez la page web, vous obtiendrez le résultat suivant :



Vous pouvez modifier la couleur, la taille des caractères et voir le résultat actualisant la lecture de votre page HTML sur votre navigateur en appuyant sur la touche **F5**.

Voici tout de même quelques explications sur le code **.css**.

Pour chaque balise, vous donnez son nom, puis entre accolades vous expliquez les caractéristiques particulières que vous lui donnez (couleurs, taille, ...)

Ainsi ligne 1 on va décrire la balise **h1** (sa couleur et sa taille en pixels).

A la ligne 6 on va décrire la balise **p** (à nouveau sa couleur et sa taille).

## XI. Différentes commandes CSS

### a) La couleur du texte : *color*

```
1 balise
2 {
3     color : blue;
4 }
```

Il existe 16 couleurs normalisées, c'est-à-dire reconnues par tous les navigateurs. D'autres existent mais elles ne sont malheureusement pas reconnues par tous.

#00FFFF Aqua	#000000 Black	#0000FF Blue	#FF00FF Fuchsia	#808080 Gray	#008000 Green	#00FF00 Lime	#800000 Maroon
#000080 Navy	#808000 Olive	#800080 Purple	#FF0000 Red	#C0C0C0 Silver	#008080 Teal	#FFFFFF White	#FFFF00 Yellow

Pour d'autres couleurs, vous pouvez faire appel à un système **RGB** (Red Green Blue) de trois nombres entre 0 et 255 en donnant la composition en rouge, vert et bleu du mélange ou leur version en hexadécimal avec le symbole **#**. Ainsi (255,0,0) correspond à **#FF0000**.

```
1 balise
2 {
3     color : rgb(255,255,0);
4 }
```

Pour récupérer la composante en rouge, vert et bleu de la couleur de votre choix, vous pouvez par exemple aller sur le site [https://www.w3schools.com/colors/colors\\_picker.asp](https://www.w3schools.com/colors/colors_picker.asp).

### b) La taille du texte : *font-size*

```
1 balise
2 {
3     font-size : 16px; /*les caractères ont une hauteur de 16 pixels*/
4 }
```

### c) La police : *font-family*

```
1 balise
2 {
3     font-family : Arial;
4 }
```

Attention, l'utilisation d'une police dépend de celles qui sont installées sur l'ordinateur de la personne qui vient visiter votre site. Si vous utilisez une police un peu trop exotique, le navigateur se rabattra sur la police standard **sans-serif**.

Les polices que l'on retrouve le plus communément sont : Arial, Arial Black, Comis Sans MS, Courier New, Georgia, Impact, Times New Roman, Trebuchet MS et Verdana.

Texte en Arial

**Texte en Arial Black**

Texte en Comic Sans MS

Texte en Courier New

Texte en Georgia

**Texte en Impact**

Texte en Times New Roman

Texte en Trebuchet MS

Texte en Verdana

### d) Italique, gras, souligné : *font-style*

```
1 balise
2 {
3     font-style : italic;
4 }
```

Les différents style que vous pouvez utiliser sont les suivants :

- **italic**,
- **oblique** (légèrement différent de l'italic),
- **normal**,

Attention, d'autres styles existent mais ils ne sont pas compris par tous les navigateurs.



### e) L'alignement : *text-align*

```
1 balise
2 {
3     text-align : center;
4 }
```

Les différents type d'alignement sont les suivants :

- **left**, le texte est aligné à gauche (c'est le cas par défaut) ;
- **center**, le texte sera centré ;
- **right**, le texte sera aligné à droite ;
- **justify**, le texte sera justifié, c'est-à-dire que le texte prendra toute la largeur possible sans laisser de blanc en fin de ligne. C'est ce que l'on retrouve dans les journaux ou les livres par exemple.

### f) Le corps de la feuille : *body*

```
1 body
2 {
3     background-color : black;
4     color : green;
5 }
```

Avec la balise `body`, on travaille sur l'ensemble de toute la page HTML. Ainsi, dans cet exemple, le fond de la page sera noir. Les textes, eux, seront en vert sauf si on précise des couleurs différentes pour certaines balises.

On peut aussi placer une image de fond dans le corps de la page :

```
1 body
2 {
3     background-image : url("monImage.png");
4 }
```

Cette image devra être placée dans le même répertoire que votre fichier `.css` pour pouvoir être lu, puis chargé par le navigateur.

Il existe plusieurs options concernant l'affichage de l'image de fond :

- **background-attachment** pour fixer ou non le fond.
  - fixed** : l'image de fond reste fixe.
  - scroll** : l'image de fond défile avec le texte (option par défaut).

```
1 body
2 {
3     background-image : url("monImage.png");
4     background-attachment : fixed;
5 }
```

- **background-repeat** répétition du fond.
  - Par défaut l'image de fond est répétée en mosaïque.
  - no-repeat** : le fond ne sera pas répété. Il n'y aura qu'une seule image sur la page.
  - repeat-x** : le fond sera répété uniquement sur la première ligne, horizontalement.
  - repeat-y** : le fond sera répété uniquement sur la première colonne, verticalement.
  - repeat** : le fond sera répété (option par défaut).

```
1 body
2 {
3     background-image : url("monImage.png");
4     background-repeat : no-repeat;
5 }
```

- **background-position** : si l'image de fond est fixe et ne se répète pas on peut choisir sa position.  
**top** : en haut.  
**bottom** : en bas.  
**left** : à gauche.  
**center** : centrée.  
**right** : à droite.

```
1 body
2 {
3     background-image : url("monImage.png");
4     background-attachment : fixed; /*le fond reste fixe*/
5     background-repeat : no-repeat; /*le fond n'est pas répété*/
6     background-position : top right; /*le fond sera placé en haut à droite*/
7 }
```

### g) D'autres commandes ?

Vous pouvez retrouver toutes les commandes officiellement reconnus par tous les navigateur en suivant le lien : <https://www.w3schools.com/cssref/default.asp>.

### h) Un debugueur ?

Tout comme pour votre page HTML, il n'y a pas de debugueur sur votre navigateur pour votre style CSS. On peut trouver un validateur en ligne à l'adresse suivante :

<http://jigsaw.w3.org/css-validator>



## XII. Des formulaires pour interagir

### a) Un premier formulaire

Un formulaire HTML est indiqué par la balise `<form>`.

```
1 <form>
2 Prénom:<br />
3 <input type="text" name="prenom"><br />
4 NOM:<br />
5 <input type="text" name="nom">
6 </form>
```

Prénom:

NOM:

Les balises `input` autorisent différentes valeurs pour l'attribut `type`. Par exemple :

- si le champ doit contenir un mot de passe, il est souhaitable qu'il ne s'affiche pas : `type="password"`
- si on doit faire un choix dans une liste, le type « radio » peut être une solution : `type="radio"`
- ...

Des formulaires plus complets? Voici une liste des différentes balises :

[https://www.w3schools.com/html/html\\_form\\_input\\_types.asp](https://www.w3schools.com/html/html_form_input_types.asp)

### b) Envoyer

Une fois rempli, on envoie le formulaire souvent à l'aide d'un bouton :

```
1 <form>
2 Prénom:<br />
3 <input type="text" name="prenom"><br />
4 NOM:<br />
5 <input type="text" name="nom"><br />
6 <input type="submit" value="Envoyer">
7 <input type="reset" value="Tout effacer">
8 </form>
```

Prénom:

NOM:

Envoyer

Tout effacer

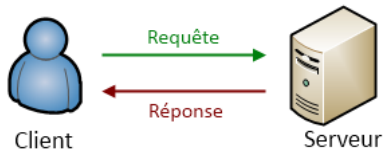
Dans cette situation, il convient de prévoir de réceptionner les réponses du formulaires, donc aussi d'envoyer, quand on clique sur le bouton, les résultats au « bon endroit ».

### XIII. Modèle client/serveur Web

Le **client** est le navigateur avec lequel on peut visualiser une page web (Mozilla Firefox, Google Chrome, ...).

Le **serveur** est l'ordinateur distant qui contient les informations (pages HTML, CSS, images, sons, programmes JavaScript, ...) qui seront envoyés au client en fonction de ses requêtes.

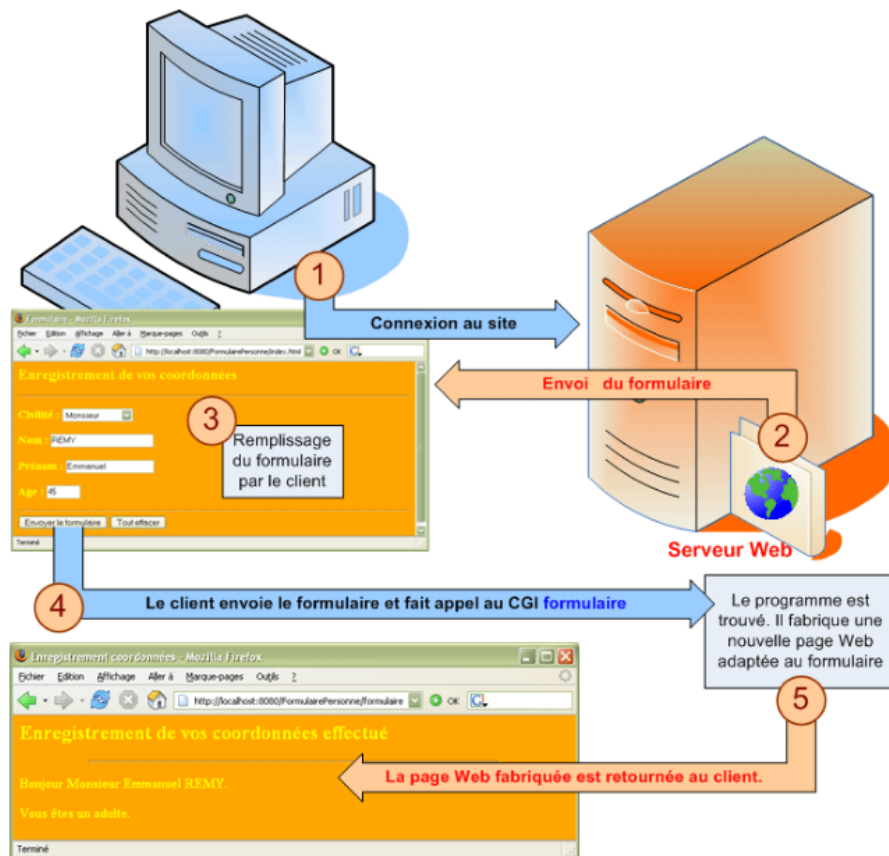
#### a) Le modèle général



En première approximation,

- si on tape l'adresse **URL**<sup>4</sup> `https://fr.wikipedia.org/wiki/Wikipédia` sur le navigateur, le client, cherche à entrer en communication avec le serveur `fr.wikipedia.org/`.
- Une fois la connexion établie il demande au serveur la ressource Wikipédia.
- Le serveur envoie au client la ressource demandée.

#### b) Communication sur un formulaire



Entre l'étape 4 et l'étape 5, le serveur se sert d'un programme informatique pour traiter l'information transmise par le client. Le serveur construit alors une nouvelle page, personnalisée avec les réponses du formulaire. Le plus souvent ce programme est écrit en php<sup>5</sup>, mais on peut trouver n'importe quel autre langage<sup>6</sup>.

4. URL pour *Uniform Resource Locator*, localiseur uniforme de ressources.

5. `https://fr.wikipedia.org/wiki/PHP`

6. On trouve de plus en plus de programme écrit en Python avec par exemple le petit framework flask ou le framework plus complet django.

`https://fr.wikipedia.org/wiki/Flask_(framework)`

`https://fr.wikipedia.org/wiki/Django_(framework)`

### c) Mode de passage des paramètres

Le mode de passage des paramètres par un formulaire est défini par l'attribut `method` de la balise `form`. Cet attribut peut-être `GET` ou `POST`

### d) La méthode GET

Une fois les champs saisis, la méthode `GET` envoie toute l'information nécessaire dans l'URL. Il est donc possible de mémoriser uniquement cette URL pour s'en servir plus tard.

Par exemple, si je recherche des images sur le thème NSI en licence *Reproduction et partage sans but commercial*, après avoir écrit NSI dans le champ de recherche, cliquer sur le bouton Images et sélectionner *Reproduction et partage sans but commercial*, la méthode `GET` envoie une requête directement avec l'adresse URL :

`https://www.qwant.com/?q=NSI&t=images&license=share`

On peut y lire la question : `q=NSI`, le type : `t=images` et la licence : `license=share` reliés par des `&`.

La limite principale de la méthode `GET` est le manque de confidentialité. En effet, considérons le formulaire suivant de saisie de mot de passe :

```
1 <form action="https://natureletchaud.fr" method="get">
2     Identifiant:<br />
3     <input type="text" name="ident"><br />
4     Mot de Passe:<br />
5     <input type="password" name="pass"><br />
6     <input type="submit" value="Se connecter">
7 </form>
```

Identifiant:

moi

Mot de Passe:

●●●●●●

Se connecter

Le mot de passe est bien masqué, cependant lorsque l'on clique sur le bouton, le navigateur exécute la requête et affiche dans sa barre d'adresse l'URL :

`https://natureletchaud.fr/?ident=moi&pass=bidule`

Si quelqu'un lit par dessus mon épaule, il obtient mon mot de passe qui est :

.....

### e) La méthode POST

Lors d'une soumission par la méthode `POST`, les paramètres ne sont plus présent l'adresse URL, mais sont passés dans le corps de la requête via le protocole HTTP.

```
1 <form action="https://natureletchaud.fr" method="post">
2     Identifiant:<br />
3     <input type="text" name="ident"><br />
4     Mot de Passe:<br />
5     <input type="password" name="pass"><br />
6     <input type="submit" value="Se connecter">
7 </form>
```

Identifiant:

Mot de Passe:

Le mot de passe est bien masqué et lorsque l'on clique sur le bouton, le navigateur exécute la requête avec les paramètres. L'adresse l'URL ne contient plus le mot de passe :

`https://natureletchaud.fr/`

Par contre si on utilise le protocole HTTP et non pas HTTPS, le mot de passe sera transmis dans la requête sans être chiffré, ce qui veut dire que l'on pourrait le récupérer.

## f) Le web côté serveur

Imaginons que Clément et moi-même souhaitons nous connecter sur la même plate-forme `http://www.france-ioi.org/`.

1. Le serveur Web reçoit une requête HTTP demandant une certaine ressource (la connexion en fonction de l'identifiant et du mot du passe).
2. Le serveur détermine alors que la ressource ne doit pas être renvoyée telle quelle au client (le navigateur) mais « évaluée ». Une manière de déterminer cela peut être l'extension du fichier (ici `.php` de l'adresse `http://www.france-ioi.org/index.php`) mais ce n'est pas la seule manière de faire.
3. Le serveur Web appelle un programme externe (par exemple l'interprète du langage php) pour évaluer le fichier. Il met à disposition de ce programme les paramètres de la requête ainsi que toutes les informations qui sont liées à cette dernière.
4. Le programme effectue un traitement en utilisant ces paramètres.
5. Le programme écrit (dans un fichier de sortie ou sur une sortie standard) le contenu de la réponse de la requête. En général c'est un document HTML, mais on peut aussi générer du CSV, du PDF, ...
6. Le serveur Web prend le fichier généré et le renvoie alors comme réponse au navigateur qui a fait la requête.

Dans ce modèle, un programme est exécuté côté serveur pour chaque requête HTTP. Mais si on revient à notre exemple :

1. Le site peut faire la différence entre Clément et moi. Mieux que ça, il personnalise ses réponses en fonction de l'identité du client.  
Ainsi Clément est connecté en mode élève, quant à moi, je suis connecté en mode professeur et j'ai accès aux résultats de chacun de mes élèves.
2. Plus important, entre deux requêtes HTTP distinctes, le serveur se souvient de « l'état » dans lequel était un client particulier lors de la dernière requête.  
Par exemple, j'ai accès à tous les thèmes validés sur France IOI et je peux même retrouver mes programmes. Évidemment, Clément n'y a pas accès. Il a accès lui à ses propres programmes.

## XIV. JavaScript



Le langage **JavaScript** a été créé dans les années 1995 par Brendan Eich, un informaticien Américain alors employé de l'éditeur de logiciels Netscape Communications. L'un des produits phare de cette société était le navigateur Web **Netscape Navigator**, qui est le premier à intégrer JavaScript. A l'époque, le système Windows de Microsoft et son navigateur intégré **Internet Explorer** sont omniprésents. Netscape Navigator ne peut pas survivre en tant que logiciel commercial. Son code est ouvert et il sert de base au navigateur Firefox, toujours en développement actuellement. Dès 1996, l'adoption de JavaScript, tant par les utilisateurs de **Netscape Navigator** que par ceux d'**Internet Explorer** en fait un langage incontournable pour la programmation Web côté client.

Vous pourrez tester tous vos codes sur la plate-forme <https://jsfiddle.net/> ou encore <https://jsbin.com/> qui permettent d'afficher le code HTML, le code CSS, le code JavaScript et obtenir le résultat sur une seule fenêtre. Vous pouvez aussi faire exactement la même chose avec le logiciel **Brackets**.

### a) Quelques notions de JavaScript

- Chaque instruction est terminée par un point virgule.
- L'indentation n'est pas obligatoire, mais vivement conseillée.
- Chaque bloc d'instructions est encadré par des accolades { et }.
- Les commentaires s'écrivent après deux slashes //.

### Variables

Bien que les variables soient typées, on ne précise pas le type. On se contente uniquement de déclarer la variable avec l'instruction `let`<sup>7</sup> :

```
1 let x, y;  
2 x = 7;  
3 y = 7 ** 2;
```

Il est à noter que JavaScript est assez tolérant sur les types :

```
1 let x = "2";      //x est de type string  
2 x = x + x;        //le résultat est "22" car on effectue une concaténation  
3 let y = x*2;       //le résultat est le nombre 44 car on ne peut pas multiplier une  
   chaine par un nombre  
4 let texte = "bonjour vous " + 5;    //le résultat est la chaîne "bonjour vous 5"
```

Pour une meilleure clarté de vos programmes, il vaut mieux éviter ces modifications et ou mélanges de type.

---

7. On peut encore trouver la précédente notation `var` qui est malgré tout de moins en moins utilisée.

## La fonction `console.log()`

La fonction `console.log()` permet d’afficher dans la console ce qui est entre parenthèses. Par exemple

```
1 let x, y ;
2 x = 7;
3 y = 7 ** 2 ;
4 console.log(y);
```

affichera dans la console le nombre 49.

Attention, il n’y aura aucun affichage sur la page web. Cette fonction est donc surtout un outil (très utile) de débbugage.

## Conditionnelles

La condition s’écrit entre parenthèses.

```
1 if (x < y) {
2     console.log("ok");
3 }
4 else {
5     console.log("problème !");
6 }
```

Les opérateurs de comparaison sont :

Opérateur	Signification
<code>==</code>	égal à <sup>8</sup>
<code>!=</code>	différent de
<code>&gt;</code>	supérieur à
<code>&gt;=</code>	supérieur ou égal à
<code>&lt;</code>	inférieur à
<code>&lt;=</code>	inférieur ou égal à

Les opérateurs logiques :

Opérateur	Type de logique	Utilisation
<code>&amp;&amp;</code>	ET	<code>valeur1 &amp;&amp; valeur2</code>
<code>  </code>	OU	<code>valeur1    valeur2</code>
<code>!</code>	NON	<code>!valeur</code>

8. Pour les petits curieux, il existe aussi l’opérateur `===` qui permet de vérifier l’égalité des valeurs **ET** du type. Ainsi `2=='2'` renvoie `true` alors que `2===2` renvoie `false` car 2 est de type number alors que '2' est de type string.



## Création et appel de fonction

On écrit le mot clef **function** puis le nom de la fonction avec ses parenthèses.

L'ensemble des instructions est encadré par des accolades.

La dernière instruction exécutée est un **return**.

```
1 //définition de la fonction double
2 function double(x) {
3     return 2*x;
4 }
5
6 console.log(double(7));
```

## Boucles bornées

Dans l'instruction d'une boucle **POUR** on définit entre parenthèses les paramètres de notre compteur :

- le point de départ (dans notre exemple `let i=0`, c'est 0)
- le point d'arrêt (dans notre exemple `i<9` c'est juste avant 9)
- l'incréméntation (dans l'exemple on a écrit l'abréviation `i++` qui signifie que l'on va de 1 en 1. On aurait aussi pu écrire `i = i + 1`)

Chacun de ces 3 paramètres sont séparés par un point virgule.

```
1 for(let i=0; i<9; i++) {
2     console.log(double(i));
3 }
```

## Boucles non bornées

La condition pour continuer à boucler avec une structure **TANT...QUE** s'écrit entre parenthèses.

```
1 let n=0;
2 while (n<3) {
3     console.log(n);
4     n++;           //on peut aussi écrire n = n + 1;
5 }
```

## D'autres références en JavaScript

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference>

## b) Intégrer un script JavaScript à une page HTML

Voici un exemple simple de ce que l'on peut faire.

Dans un premier temps voici notre fichier écrit en HTML :

```

1 <!--DOCUMENT html-->
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>Le dé</title>
6   </head>
7
8   <body>
9     <p>
10      Nous allons lancer un dé.<br \>
11      <form name="formulaire">
12      Combien a-t-il de face ? :
13      <input id="nb_faces" type="number" min="4" max="30"/> <br />
14      </form>
15    </p>
16    <button onclick="alea()">Lancer le dé</button>
17
18    <p id="a_remplir"></p>
19
20    <p>Merci !</p>
21
22    <!-- un lien vers le programme écrit en JavaScript -->
23    <script type="text/javascript" src="aleatoire.js"></script>
24
25  </body>
26 </html>

```

- Lignes 13 et 18, on donne des identités `id` à certaines balises.  
C'est à l'aide des ces `id` que notre programme écrit en JavaScript pourra interagir avec la page Web.  
Les interactions se feront donc sur ces 2 balises identifiées. Il faut donc que les `id` soient des **identifiants uniques**.
- Ligne 16, on informe que lorsque l'on cliquera sur le bouton `onclick` on devra exécuter la fonction `alea()`.  
Mais où est-elle ?
- Ligne 23, on indique que l'on a un script écrit en JavaScript `script type="text/javascript"`.  
On indique ensuite où le trouver. La source `src` se situe dans le fichier `aleatoire.js`.

Voici maintenant le fichier `aleatoire.js` associé à notre page web.

```
1 function alea() {  
2     let reponse = document.getElementById("nb_faces");  
3     let nb = reponse.value;  
4     let lancer = Math.floor((Math.random() * nb) + 1);  
5  
6  
7     let paragraphe = document.getElementById("a_remplir");  
8     let texte = "Le résultat du lancer est "+lancer+"<br /> La classe !"  
9     paragraphe.innerHTML = texte;  
10 }
```

- Lignes 2 et 7, on crée un lien entre le document HTML et certaines balises repérées par leur `id` à l'aide de la commande `document.getElementById()`.  
Le résultat est un objet assez complexe que nous ne détaillerons pas ici. Il contient de nombreuses informations.
- Ligne 3, on récupère la valeur de la réponse à l'aide de la méthode `value`.  
Le résultat est du type `string`.  
Autrement dit, le résultat du formulaire qui est ici le nombre de faces choisi par l'utilisateur.
- Ligne 4 on calcule le résultat du lancer. On utilise deux fonctions `random()` et `floor()` présentes dans la bibliothèque `Math`.  
La première renvoie un nombre décimal entre 0 et 1 (exclu).  
La deuxième donne la partie entière d'un nombre décimal.
- Ligne 8, on prépare notre réponse avec éventuellement des balises HTML.
- Ligne 9, à l'aide de la méthode `innerHTML` on pourra insérer le texte tel quel à l'intérieur de notre document HTML sur la balise correspondante.  
Donc notre cas, le texte sera insérer pour la balise d'`id="a_remplir"`.

### Exercice 1

Écrire une page HTML et son compagnon en JavaScript correspondant à une page Web demandant le nom, le prénom, la taille et le poids de l'utilisateur et renvoie son IMC<sup>9</sup>.

### Exercice 2

Améliorer votre page précédente en interprétant le résultat du calcul obtenu (normal, surpoids, ...)

### Exercice 3

On reprend l'exemple du lancer de dé en le modifiant légèrement.  
L'utilisateur doit choisir le nombre de lancers qu'il veut effectuer.

---

9. [https://fr.wikipedia.org/wiki/Indice\\_de\\_masse\\_corporelle](https://fr.wikipedia.org/wiki/Indice_de_masse_corporelle)