

# Cours : WEB

## 1. Introduction au WEB

Le "**World Wide Web**", plus communément appelé "**Web**" (littéralement la "toile d'envergure mondiale") a été développé au CERN (Conseil Européen pour la Recherche Nucléaire) par le Britannique **Sir Timothy John Berners-Lee** au début des années 90.

Pour faciliter les échanges d'information entre scientifiques, Tim Berners-Lee met au point le système **hypertexte**. Le système **hypertexte** permet, à partir d'un document, de consulter d'autres documents en cliquant sur des mots clés. Ces mots "cliquables" sont appelés **hyperliens**.

Tim Berners-Lee développe aussi le premier navigateur web (logiciel permettant de lire des pages contenant des hypertextes). C'est en 1993 avec l'arrivée du navigateur web "NCSA Mosaic" que le web commence à devenir populaire en dehors du monde de la recherche.

Le web est basé sur 3 notions essentielles :

- les **URL (Uniform Resource Locator)**
- le langage de description **HTML (HyperText Markup Language)**
- le protocole **HTTP (HyperText Transfer Protocol)**

⚠ Attention à ne pas confondre "web" et "internet" ⚠

Internet est un réseau de réseaux mondial, c'est-à-dire l'infrastructure globale, basée sur le protocole IP, et sur laquelle s'appuient différents services, dont le Web. Le web est un service qui utilise ce réseau. Le Web est le système permettant de naviguer de pages en pages en cliquant sur des liens dans un navigateur.

Donc Internet est le réseau, l'infrastructure. Le Web est un service sur ce réseau.

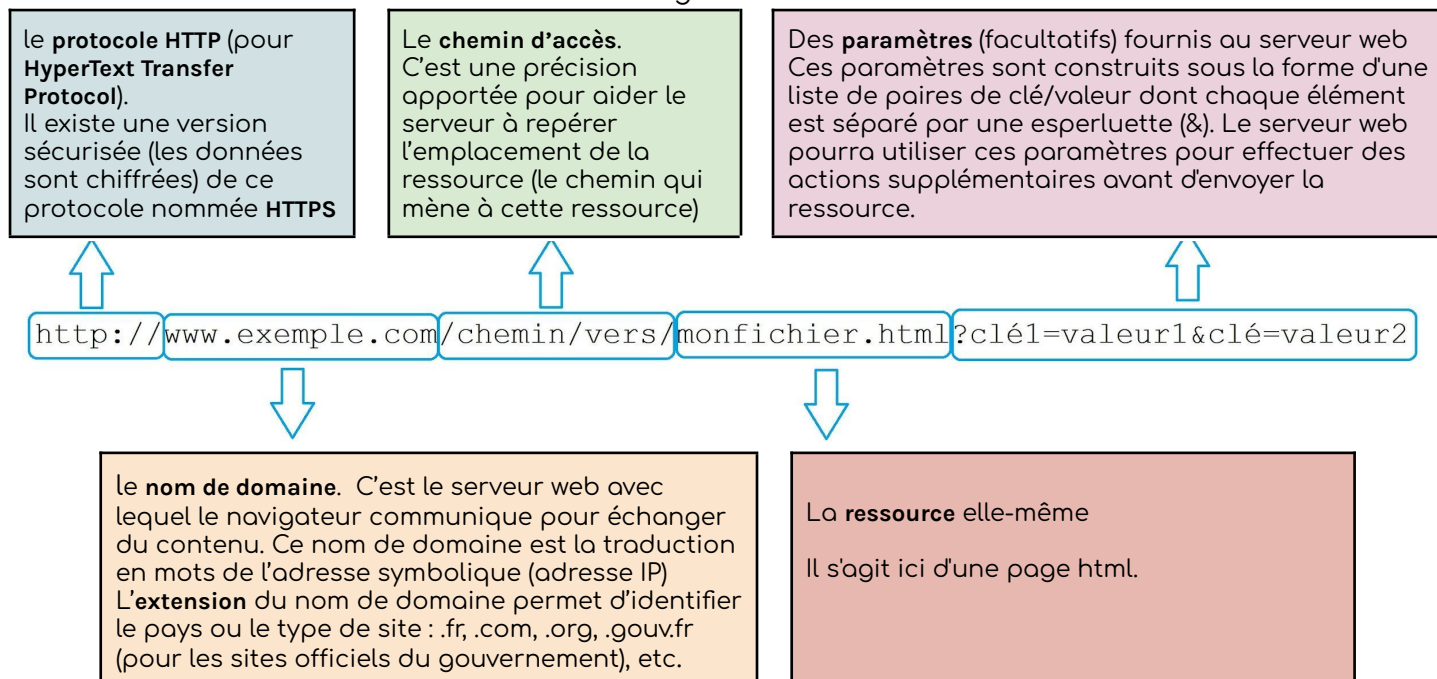
D'ailleurs, sur internet d'autres services autres que le web sont à disposition. Par exemple les mails (utilisant le protocole **SMTP : Single Mail Transfer Protocol**) ou bien les transferts de fichiers (utilisant le protocole **FTP : File Transfer Protocol**).

## 2. URL

Une **URL** est une adresse sur le web. Elle permet de trouver une page sur Internet.

Le sigle **URL** est l'acronyme de **Uniform Resource Locator**

L'**URL** est saisie dans la barre d'adresse d'un navigateur afin d'accéder à une ressource sur le Web.



### 3. Le couple HTML + CSS

Deux langages sont nécessaires pour écrire une page web :

- le langage **HTML** pour décrire la structure de la page
- le langage **CSS** pour décrire la mise en page de la page

#### a. HTML : le langage de balisage

L'**HyperText Markup Language**, est le langage de description à **balises (ou langage de balisage)** conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte (texte lié à une autre ressource du Web). **HTML** permet de structurer le contenu des pages, et d'y inclure des images, des vidéos, du son, des formulaires de saisie, des programmes informatiques.... Il est souvent utilisé conjointement avec des feuilles de style en cascade (**CSS**) et le langage de programmation **JavaScript**.

La version actuelle est le **HTML 5**.



#### i. Vocabulaire

Vocabulaire	Définition	Exemple
<b>Balise</b>	La structure d'un page <b>HTML</b> est donnée par des balises, qui sont les suites de caractères délimitées par < et >	<code>&lt;head&gt;</code> <code>&lt;/h1&gt;</code>
<b>Balise ouvrante</b>	Balise ne contenant pas le caractère /	<code>&lt;p&gt;</code>
<b>Balise fermante</b>	Balise contenant le caractère / à la suite du caractère <	<code>&lt;/title&gt;</code>
<b>Attribut, Valeur</b>	Chaque balise peut avoir un ou plusieurs <b>attributs</b> . Ces attributs sont des informations supplémentaires qui permettent de configurer les <b>balises</b> ou d'adapter leur comportement. Un attribut est, le plus souvent, associé à une valeur au moyen du caractère =. La valeur d'un attribut est donnée comme une chaîne de caractères contenues entre guillemets.	<code>&lt;html lang="fr"&gt;</code> attribut → lang valeur → fr  <code>&lt;label for="name"&gt;</code> attribut → for valeur → name
<b>Élément</b>	Une paire de balises ouvrante et fermante, ainsi que le contenu situé entre les deux, est appelé élément. Un élément peut contenir d'autres éléments.	<code>&lt;ul&gt;</code> <code>&lt;li&gt;Dentifrice&lt;/li&gt;</code> <code>&lt;li&gt;chocolat&lt;/li&gt;</code> <code>&lt;/ul&gt;</code>
<b>Balise orpheline</b>	(ou balise auto-fermante) Balise qui est à la fois ouvrante et fermante. Elle se compose comme suit <code>&lt; nom_balise /&gt;</code>	<code>&lt;meta charset="utf8" /&gt;</code> <code>&lt;/br&gt;</code>
<b>Commentaire</b>	Chaîne de caractère délimitée par <code>&lt;!--</code> et <code>--&gt;</code> et ignoré par le navigateur Web. Un commentaire sert de documentation à l'auteur du fichier.	<code>&lt;!-- Commentaire --&gt;</code>

#### ii. Structure d'un fichier HTML

<ul style="list-style-type: none"> <li>• Type de fichier : <code>&lt;!DOCTYPE html&gt;</code></li> <li>• Ouverture du fichier : <code>&lt;html&gt;</code></li> <li>• Une <b>en-tête</b> de fichier qui donne des précisions sur le fichier : <code>&lt;head&gt;</code></li> <li>• Un contenu qui s'affiche dans le navigateur : <code>&lt;body&gt;</code></li> </ul>	<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;!-- En-tête de la page --&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;!-- Corps de la page --&gt;   &lt;/body&gt; &lt;/html&gt;</pre>
--	---

### iii. Balises

**Balises d'en-tête <head>** Contient des informations sur le fichier HTML qui ne sont pas affichées sur la page

Balise	Description	Exemple
<link />	Liaison avec une feuille de style	<link rel="stylesheet" href="style.css">
<meta />	Métadonnées de la page web (charset, mots-clés etc..)	<meta charset="utf8" />
<script>	Code JavaScript	<script src="js3.js"></script>
<title>	Titre de la page (situé dans l'onglet)	<title> Programme NSI </title>

**Balises de corps <body>** Contient le contenu ou corps de la page

#### Balises de structuration du texte

Balise	Description
<b> ou <strong>	Gras
<i>	italique
<p>	paragraphe
 	retour à la ligne
<h1> ... <h6>	titre ... sous titre
<a>	lien
<img />	image
<figcaption>	description de l'image
<audio>	son
<video>	vidéo

#### Balises de formulaire

Balise	Description
<form>	formulaire
<fieldset>	groupe de champs
<legend>	titre d'un groupe de champs
<label>	libellé d'un champ
<input />	champ de formulaire (texte, mot de passe, case à cocher, bouton etc... )
<textarea>	zone de saisie multi-ligne
<select>	liste déroulante
<option>	élément d'une liste déroulante
<optgroup>	groupe d'éléments d'une liste déroulante

#### Balises de tableau

Balise	Description
<table>	tableau
<caption>	Titre du tableau
<tr>	ligne du tableau
<th>	cellule d'en-tête
<td>	Cellule
<thead>	Section de l'en-tête du tableau
<tbody>	Section du corps du tableau
<tfoot>	Section du pied du tableau

#### Balise de listes

Balise	Description
<ul>	liste à puces, non numérotée
<ol>	liste numérotée
<li>	élément de la liste à puces
<dl>	liste de définitions
<dt>	terme à définir
<dd>	définition du terme

**Balises sectionnantes** Structuration recommandée

Balise	Description	
<code>&lt;header&gt;</code>	Une en-tête de la page	
<code>&lt;nav&gt;</code>	Un menu navigation	
<code>&lt;section&gt;</code>	Une partie du contenu	
<code>&lt;article&gt;</code>	Des articles dans chaque section	
<code>&lt;aside&gt;</code>	Éventuellement un contenu sur un côté	
<code>&lt;footer&gt;</code>	Un pied de page	

**iv. Conventions**

- Nom des balises en **minuscule**
- Chaque balise ouvrante doit être **fermée** (sauf les balises auto-fermantes)
- Un **espace** entre chaque attribut
- Pas d'espace entre l'attribut, le = et la valeur
- Les valeurs entre " "
- Le code **HTML** doit être « propre », après chaque ouverture de balise, les contenus des balises sont écrits après un retour à la ligne et une tabulation, on parle alors d'**indentation**.
- La fin de la balise doit être à la même tabulation que la balise d'entrée afin de repérer rapidement une éventuelle erreur dans le code.
- Validation du W3C : <https://validator.w3.org/>

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première page Web</title>
  </head>
  <body>
    <!-- Menu de navigation du site -->
    <ul class="navbar">
      <li><a href="index.html">Home</a>
      <li><a href="profil.html">Profil</a>
      <li><a href="liens.html">Liens</a>
    </ul>
    <!-- Contenu principal -->
    <h1>Ma page web</h1>
    <p>Bienvenue sur ma page avec style!</p>
    <p>Il lui manque des images</p>
  </body>
</html>
```

**b. CSS : Le langage feuilles de style**

Le langage feuilles de styles en cascade (en anglais "**Cascading Style Sheets**", abrégé **CSS**) est un langage de description qui permet de gérer la présentation d'une page Web. Les styles permettent de définir des règles appliquées à un ou plusieurs documents **HTML**. Ces règles portent sur le positionnement des éléments, l'alignement, les polices de caractères, les couleurs, les marges et espacements, etc. La dernière version du **CSS** est le **CSS 3**.


**i. Avantages du CSS**

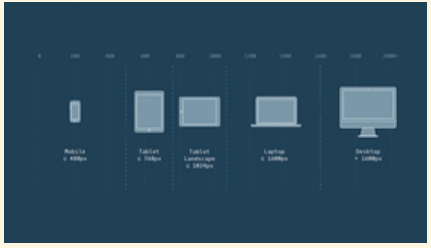
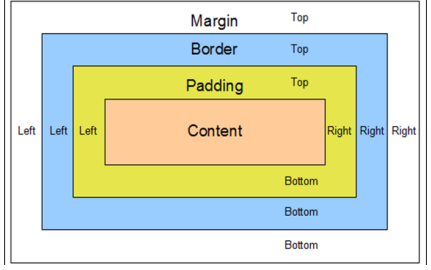

Au début du **HTML**, celui-ci mélange allégrement les balises structurantes avec la mise en forme de la balise. Cependant, le code est devenu rapidement trop compliqué et illisible. Se pose alors rapidement des problèmes de maintenance des sites web.

Depuis le **CSS**, tout le contenu est écrit en **HTML** et est séparé des éléments de mise en forme qui sont écrits en **CSS**. Il est recommandé d'écrire l'apparence commune pour toutes les pages d'un site web dans des fichiers css séparés.

**ii. Utilisation du CSS**

Syntaxe du CSS	
<p>La syntaxe du CSS est simple, elle est constituée :</p> <ul style="list-style-type: none"> <li>• d'un sélecteur qui sélectionne un ou plusieurs éléments du document HTML.</li> <li>• d'un bloc de déclaration entre deux { }.</li> <li>• une liste de couples : propriété : valeur(s) ;</li> </ul>	

<b>Sélecteurs de type élément</b>	<pre>h1 {   color : #f90;   font-size : 36 px; }</pre>
<ul style="list-style-type: none"> <li>Pour attribuer des styles à <b>tous les éléments</b> du même nom (même balise).</li> <li>A utiliser pour définir des styles très généraux.</li> <li>Syntaxe : <b>élément</b></li> </ul>	
<b>Sélecteurs de type identifiant</b>	<b>HTML</b> <pre>&lt;h3 id="film1"&gt;Avengers &lt;/h3&gt; &lt;p&gt; Film de superhéros &lt;/p&gt;</pre> <b>CSS</b> <pre>#film1{   font-family: sans-serif;   color: #555 }</pre>
<ul style="list-style-type: none"> <li>Permet d'attribuer des styles <b>uniquement</b> à l'élément <b>HTML</b> ayant la valeur identifiant pour son attribut <b>id</b>.</li> <li>Un <b>id</b> est unique dans un document <b>HTML</b></li> <li>Syntaxe : <b>#identifiant</b></li> </ul>	
<b>Sélecteurs de type classes</b>	<b>HTML</b> <pre>&lt;h3 class="film titre"&gt;Avengers &lt;/h3&gt; &lt;p class="film"&gt; Film de superhéros &lt;/p&gt;</pre> <b>CSS</b> <pre>.film{   font-style: italic;   background-color: #eee   padding: 1em   border: 1px solid black; }</pre>
<p>Dans le document <b>HTML</b> : attribut <b>class</b> permet d'attribuer un nom de classe à un élément</p> <p>Un élément peut appartenir à plusieurs classe</p> <p>Plusieurs éléments peuvent être attribués d'un même nom de classe</p> <ul style="list-style-type: none"> <li>Permet d'attribuer des styles à <b>tous les éléments de même classe</b> (attribut <b>class</b>)</li> <li>Une classe s'utilise pour définir des styles transverses à plusieurs éléments</li> <li>Syntaxe : <b>.nom_classe</b></li> </ul>	
<b>Pseudo classes et éléments</b>	<pre>div {   background-color : #4F5ED5 ; }</pre> <pre>div:hover {   background-color : red ; }</pre>
<p>Pseudo-classes pour attribuer un style en fonction de l'état de l'élément :</p> <ul style="list-style-type: none"> <li>Survolé <b>.hover</b></li> <li>Déjà visité <b>.visited</b></li> <li>Pas encore visité <b>.link</b></li> <li>Lien actif <b>.active</b></li> <li>Ayant le focus <b>.focus</b></li> </ul>	
<b>Regroupements de sélecteurs</b>	<pre>h1,h2{   font-weight : bold ;   font-family : Georgia, arial ; } ol li {   color : blue ; }</pre>
<ul style="list-style-type: none"> <li>On peut regrouper des sélecteurs ayant des caractéristiques communes en les séparant par des <b>virgules</b>.</li> <li>Ou caractériser un élément quand il est inclus dans un autre élément ( balise li incluse dans ol)</li> </ul>	
<b>Les couleurs</b>	
<p>Une couleur en CSS peut être défini par :</p> <ul style="list-style-type: none"> <li><b>rgb(R, G, B)</b> : R, G et B sont des valeurs entières entre 0 et 255 (un octet), pour le rouge, le vert et le bleu ou des pourcentages</li> <li><b>#RRGGBB</b> RR, GG et BB sont les valeurs hexadécimales (entre 00 et FF). Écriture abrégées <b>#RGB</b></li> <li>Quelques (16) couleurs prédéfinies : <b>black, red, green, blue,</b> etc...</li> <li>Voir <a href="http://www.code-couleur.com/">http://www.code-couleur.com/</a></li> </ul>	

Unités de mesure	
Le modèle de la boîte	
Positionnement	
<ul style="list-style-type: none"><li>Le «flux normal» représente le rendu de gauche à droite, de haut en bas des éléments.</li><li>Dans ce cas, l'agencement des boîtes est calculé par le navigateur en fonction des propriétés de chacune (inline, block)</li><li>Faire «flotter» un élément (float), c'est le sortir du flux normal. L'élément est placé le plus à droite ou à gauche de son conteneur et laisse s'écouler le flux le long de sa boîte.</li><li>La valeur "absolute" de la propriété position sort aussi complètement l'élément du flux normal : le positionnement se fait par rapport à la page (ou à une boîte conteneur positionnée).</li><li>La valeur "relative" de la propriété position sort aussi complètement l'élément du flux normal : le positionnement se fait par rapport à l'élément précédent.</li><li>La valeur "fixed" de la propriété position sort aussi complètement l'élément du flux normal : le positionnement est fixe dans la fenêtre.</li></ul>	<p><b>float : left ;</b></p> <p><b>position: absolute ; top: 100px ; left: 100px ;</b></p> <p><b>position: relative ; top: 100px ; right: 100px ;</b></p> <p><b>position: fixed ; bottom: 100px ; left: 100px ;</b></p>
Mode d'affichage d'un élément	<div>Transformer un élément inline en <b>block</b></div> <div>Transformer un élément block en <b>inline</b></div> <div>Effacer un élément</div> <div>display : block;</div> <div>display: inline;</div> <div>display: none;</div>
Ou écrire le CSS	<pre>&lt;head&gt;   &lt;meta charset="utf8" /&gt;   &lt;title&gt; NSI &lt;/title&gt;   &lt;link rel="stylesheet" type="text/css"     href="cours.css" /&gt; &lt;/head&gt;</pre>
<div>Dans un fichier à part avec l'extension .css</div> <div>Faire le lien entre le document HTML et le fichier .css</div>	

## 4. JavaScript

### a. Définition

**JavaScript** est un langage de programmation qui permet d'implémenter des mécanismes d'interactions complexes sur une page web. Il est interprété par un moteur **JavaScript** intégré dans le navigateur.

Le **JavaScript** a été créé en **1995** par **Brendan Eich**, il est utilisé pour programmer le navigateur côté client.



Attention à ne pas confondre le **javascript** et le **java**, ce sont des langages différents (bien qu'ils soient tous les deux basés sur du **C++**, donc, il est possible d'y trouver des similitudes). **JavaScript** est différent du **PHP**, qui est un autre langage d'interaction du web, mais du côté serveur.

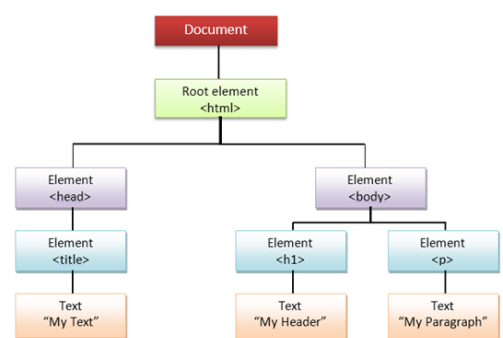
## b. Intégration de code javascript

Le code Javascript peut être directement intégré dans une page HTML avec la balise <b>&lt;script&gt;</b>	<pre>&lt;html&gt; ... &lt;body&gt;   &lt;script&gt; alert("Hello world !"); &lt;/script&gt; &lt;/body&gt; &lt;/html&gt;</pre>
Cependant, comme le CSS, il est recommandé de le déporter dans un fichier Javascript .js indépendant.	<p><b>HTML</b></p> <pre>&lt;html&gt; ... &lt;body&gt;   ...   &lt;script src="script.js"&gt;&lt;/script&gt; &lt;/body&gt; &lt;/html&gt;</pre> <p><b>JAVASCRIPT</b></p> <pre>alert("Hello world !");</pre>

## c. Construction d'un programme JavaScript

Variables	
<ul style="list-style-type: none"> <li>Le nom d'une variable ne peut contenir que des lettres de A à Z, des chiffres, l'underscore ( _ ) et le dollar ( \$ ). Le JavaScript est un langage sensible à la casse (différence entre majuscule et minuscule).</li> <li>Comme Python, le JavaScript est un langage typé dynamiquement. Par contre, il est recommandé de déclarer les variables avec le mot clé <b>var</b> soit en début de programme, soit à leur première utilisation.</li> </ul> <p>Les lignes de code Javascript se terminent toujours par ; (point virgule) Les "blocs de code" sont délimitées par { ... } (accolades)</p>	<pre>var n; n = 2;           // Type number  var nom = "Dupond"; // Type string  var etat=true;   // Type boolean</pre>
Commentaires	
<ul style="list-style-type: none"> <li>Il est possible d'intégrer des commentaires dans du code JavaScript, de la même manière que dans le CSS.</li> <li>Si votre commentaire tient sur une ligne, vous pouvez utiliser deux barres obliques pour indiquer un commentaire</li> </ul>	<pre>/* ceci est un commentaires sur plusieurs lignes*/  // commentaire sur une ligne</pre>
La structure conditionnelle	
<p><b>Rappel:</b> Les structures conditionnelles sont des éléments du code qui permettent de tester si une expression est vraie ou non et d'exécuter des instructions différentes selon le résultat.</p> <p>Opérateurs : ==, !=, &gt;, &gt;=, &lt;, &lt;=, &amp;&amp; (et),    (ou), ! (non)</p>	<pre>var note = prompt("saisir une note");  if (note &lt; 0    note &gt; 20) {   alert("Note non comprise entre 0 et 20"); } else {   alert("La note saisie est "+note); }</pre>
La boucle itérative bornée	
<p><b>Rappel:</b> Cette boucle permet de répéter des instructions un nombre de fois connu.</p>	<pre>var str = "" ;  for (var i = 0; i &lt; 10; i++) {   str = str + i ; }  confirm("les chiffres sont : "+str);</pre>
La boucle itérative non bornée	
<p><b>Rappel:</b> cette boucle s'exécute tant qu'une condition de test est vérifiée. La condition est évaluée avant d'exécuter l'instruction contenue dans la boucle.</p> <p>Dans l'exemple, on utilise des fonctions prédéfinies:</p>	<pre>var n = Math.round(Math.random()*10); var nombre = prompt("Devinez un nombre entre 1 et 10"); // prompt permet une saisie  while(nombre != n ){   nombre = prompt ("Faux : saisir un nouveau nombre");</pre>



<ul style="list-style-type: none"> <li>• <i>Math.random</i> qui renvoie un nombre entre [0,1[</li> <li>• <i>Math.round</i> qui arrondit à l'entier supérieur</li> </ul>	}
<b>Les fonctions</b>	
<p><b>Rappel:</b> Une fonction permet de créer une portion de code réutilisable. Elle exécute le code qu'elle contient quand elle est appelée. Il est possible d'appeler une fonction de n'importe où dans la page web à partir du moment où elle a été déclarée avant.</p> <p>En python, le mot clé <b>def</b> permet de déclarer une fonction. En JavaScript, c'est le mot clé <b>function</b>.</p>	<pre>function affichedate(){     alert("La date : "+Date() ); }</pre>
<b>Programmation événementielle</b>	
<ul style="list-style-type: none"> <li>• En Python, la programmation est <b>séquentielle</b> ou <b>impérative</b> (instructions exécutées dans l'ordre du programme).</li> <li>• Avec Javascript, on peut faire de la programmation <b>événementielle</b>. Les événements sont des structures de code qui « écoutent » ce qui se passe dans le navigateur et déclenchent du code en réponse.</li> <li>• La fonction qui est appelée sur l'événement est exécutée en séquentiel.</li> </ul>	<pre>&lt;body&gt; &lt;p&gt; Page avec javascript &lt;/p&gt; &lt;p onClick = "affichedate()"&gt; Cliquer pour voir la date du jour &lt;/p&gt; &lt;p onMouseover="affichedate()"&gt; Survoler pour voir la date du jour &lt;/p&gt; &lt;br /&gt; Taper sur une touche pour voir la date du jour: &lt;input onkeypress="affichedate()"&gt; &lt;/body&gt;</pre>
<b>Modifier le CSS avec l'id</b>	
<p>Sur des événements traités en Javascript, il est possible de modifier les propriétés CSS d'éléments HTML grâce à leur id.</p>	<b>CSS</b> <pre>#p1 {     background-color : red;     width: 300px;     height: 70 px; }</pre>
<b>HTML</b> <pre>&lt;html&gt; &lt;head&gt;     &lt;meta charset="utf8" /&gt;     &lt;link href="formats.css" rel="stylesheet" type ="text/css" /&gt; &lt;/head&gt; &lt;body&gt;     &lt;p id="p1" onClick="cssp1()"&gt; Page JavaScript &lt;/p&gt;     &lt;script src="script.js" &gt;&lt;/script&gt;</pre>	<b>JavaScript</b> <pre>function cssp1(){     p1.style.color="blue"; }</pre>
<b>Document Object Model : DOM</b>	
<ul style="list-style-type: none"> <li>• Le Document Object Model (<b>DOM</b>) est une interface de programmation pour les documents HTML.</li> <li>• Un <b>document HTML</b> chargé dans le navigateur devient un <b>objet Javascript</b>.</li> <li>• L'objet document donne accès à tous les <b>éléments de la page</b>, chaque élément devenant aussi un <b>objet Javascript</b>.</li> </ul> <p>On a un arbre d'objets.</p>	
<b>Accéder à un objet du DOM</b>	
<p>Il existe 4 méthodes pour accéder à un objet du DOM :</p> <ul style="list-style-type: none"> <li>• <b>getElementById</b> (par identifiant)</li> <li>• <b>getElementsByTagName</b> (par nom de balise)</li> <li>• <b>getElementsByClassName</b> (par classe)</li> <li>• <b>getElementsByName</b> (pour les éléments d'un formulaire)</li> </ul>	<pre>var elem=document.getElementById("p1"); var elems1=document.getElementsByTagName("p"); var elems2=document.getElementsByClassName("N"); var elems3=document.getElementsByName("up")</pre>
<p>Attention: Dans les 3 derniers cas, on récupère un tableau d'objets.</p>	



Modifier le contenu du HTML et du CSS avec le DOM	<pre>function cssp1(){   var elem=document.getElementById("p1");   elem.style.color="blue";   elem.innerHTML=Date(); } function effacer(){   var elem=document.getElementById("p1");   elem.style.display="none" } function reset(){   var elem=document.getElementById("nom");   elem.value=""; }</pre>
<p>La modification du contenu <b>HTML</b> se fait avec la propriété :</p> <ul style="list-style-type: none"> <li><b>innerHTML</b>, pour tout élément sauf les input</li> <li><b>value</b>, pour les éléments input des formulaires</li> </ul> <p>La modification du <b>CSS</b> se fait avec la propriété :</p> <ul style="list-style-type: none"> <li><b>style</b>, permet de définir les styles d'un élément</li> </ul>	
Les tableaux d'objets	<pre>function effacer(){   var elems =   document.getElementsByClassName("T");    for(i=0; i&lt;elems.length; i++){     elems[i].style.display="none"   } }</pre>
<p>Le tableau est un <b>type composé d'éléments d'un autre type</b>. Donc à un nom de variable correspond plusieurs éléments (valeurs). L'accès aux éléments du tableau se fait par un indice qui commence à <b>zéro</b> (comme en python)</p> <p>Après avoir récupéré un tableau d'objets du <b>DOM</b>, la modification ne peut se faire que sur un élément à la fois. Il faut donc utiliser une <b>boucle</b> pour <b>parcourir</b> ce tableau.</p> <p>La propriété <b>length</b> donne le nombre d'éléments dans un tableau</p>	

## 5. Le modèle client-serveur

Principe de fonctionnement du modèle client-serveur dans le cadre d'un échange de pages Web sur Internet :

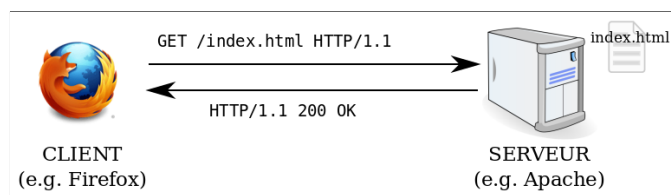
- **Client** : c'est le visiteur du site Web. Il demande la page Web au serveur. En pratique, vous êtes des clients quand vous surfez sur le Web. Plus précisément c'est le navigateur Web (Firefox, Chrome, Safari, IE, Edge, ...) qui est le client car c'est lui qui demande la page Web.
- **Serveur(s)** : ce sont les ordinateurs qui délivrent les sites Web aux internautes, c'est-à-dire aux clients.



Le client fait une requête au serveur, qui lui répond en lui envoyant le code **HTML** de la page Web.

### 1. Le navigateur comme client HTTP

- Quand on ouvre une **URL** commençant par **http://**, le navigateur va agir comme un client **HTTP**. Il va donc envoyer une requête **HTTP**.
- Le serveur **HTTP** (appelé aussi serveur **WEB**) renvoie une réponse **HTTP** avec le code **HTML** de la page demandée
- Le navigateur commence à interpréter ce code et à l'afficher.
- Dès qu'une ressource (**CSS**, image, ...) apparaît dans la page, une autre requête **HTTP** est envoyée au serveur
- Le serveur renvoie alors la ressource qui peut être affichée par le navigateur



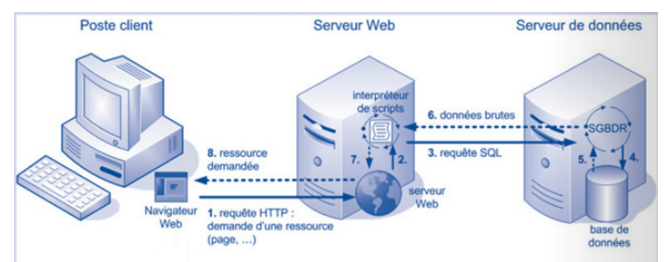
### Architecture à 2-tiers

On appelle architecture à 2-tiers, l'architecture client/serveur au sein de laquelle le client réclame une source et le serveur la lui donne directement sans même solliciter une autre application.

### Architecture 3-tiers, N-tiers

Le serveur web peut échanger avec des serveurs qui servent d'interface entre la partie web et une application spécifique.

Les sites web dynamiques sont souvent 3-tiers



## Sites Internet statiques vs dynamiques

Les sites *statiques* :

- Ce sont des sites réalisés uniquement à l'aide de **HTML/CSS**.
- Ils fonctionnent très bien mais leur contenu ne change pas.
- Les sites statiques sont donc bien adaptés pour réaliser des sites "vitrine".

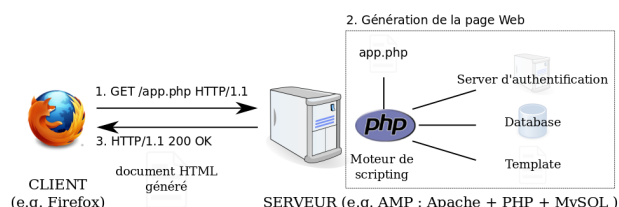
Les sites *dynamiques* :

- Ils utilisent d'autres langages tels que **PHP** pour générer du **HTML** et **CSS**.
- La plupart des sites Web que vous visitez sont dynamiques.
- Fonctionnalités typiques de sites dynamiques :

un espace membres, un forum, un compteur de visiteurs, des actualités, une newsletter.

Mécanisme de génération des pages dynamiques :

- Le client demande au serveur à voir une page Web (requête **HTTP**) ;
- Le serveur crée la page spécialement pour le client (en utilisant par exemple le langage PHP)
  - Le serveur répond au client en lui envoyant la page qu'il vient de générer (réponse **HTTP**).



## 6. Protocole HTTP

Un protocole de communication est un ensemble de règles qui permettent à des ordinateurs de communiquer ensemble. Le protocole **HTTP (HyperText Transfer Protocol)** va permettre au client d'effectuer des requêtes à destination d'un serveur web. En retour, le serveur web va envoyer une réponse.

### Requête HTTP (du client vers le serveur)

Ligne de requête	GET /monrepertoire/monFichier.html HTTP/1.1	Cette requête <b>HTTP</b> contient les informations suivantes : <ul style="list-style-type: none"> <li>• <b>GET</b> est la méthode employée</li> <li>• <b>/monrepertoire /monFichier.html</b> correspond au chemin de la ressource demandée</li> <li>• <b>HTTP/1.1</b> : la version utilisée du protocole HTTP (1.1 ici)</li> <li>• <b>Mozilla/5.0</b> : le navigateur web employé est Firefox de la société Mozilla</li> <li>• <b>text/html</b> : le client s'attend à recevoir du HTML</li> </ul>
Entête de requête	User-Agent : Mozilla/5.0 Accept : text/html	
Saut de ligne		
Corps de requête		

### Réponse du serveur à une requête HTTP (du serveur vers le client)

Une fois que le serveur a reçu la requête du client, le serveur va adresser une réponse au client.

Ligne de requête	HTTP/1.1 200 OK	Voici quelques explications sur la réponse envoyée par le serveur : <ul style="list-style-type: none"> <li>• <b>HTTP/1.1 200 OK</b> : Le code <b>200</b> signifie que la ressource est bien présente sur le serveur.</li> </ul>
Entête de requête	Date: Tue, 09 Jul 2019 19:43:31 GMT Server: Apache/2.0.54 (Debian GNU/Linux)SVN/1.1.4 Content-Type: text/html; Content-Length: 3521	

Saut de ligne		demandée par le client n'existe pas sur le serveur.
Corps de requête	<pre>&lt;!doctype html&gt; &lt;html lang="fr"&gt; &lt;head&gt; ...</pre>	<ul style="list-style-type: none"> <li>• <b>Server: Apache/2.0.54 (Debian GNU/Linux) SVN/1.1.4 :</b> On a ici une information sur la machine serveur : le type de serveur est un serveur Apache fonctionnant sur le système d'exploitation Linux.</li> </ul>

Le code de statut HTTP	Content-Type : type MIME
Un sous-ensemble des codes et textes des réponses: <ul style="list-style-type: none"> <li>• 200 OK</li> <li>• 301 MOVED PERMANENTLY</li> <li>• 308 PERMANENT REDIRECT</li> <li>• 401 UNAUTHORIZED</li> <li>• 403 FORBIDDEN</li> <li>• 404 NOT FOUND</li> <li>• 500 INTERNAL SERVER ERROR</li> </ul>	Un sous ensemble de type de contenu : <ul style="list-style-type: none"> <li>• text/plain (texte brut)</li> <li>• text/html</li> <li>• text/javascript</li> <li>• text/css</li> <li>• image/png</li> <li>• image/jpeg</li> <li>• video/mpeg</li> </ul>

## La version sécurisée de HTTP : HTTPS :

Pour garantir le chiffrement des données entre le client et le serveur, il existe une version sécurisée du protocole **HTTP**: le protocole **HTTPS**.

- le client (le navigateur Web) contacte un serveur et demande une connexion sécurisée en proposant une liste de méthodes de chiffrement
- Le serveur répond en choisissant dans cette liste une méthode de chiffrement et produit un certificat garantissant qu'il est bien le serveur en question et pas un serveur pirate déguisé.
- Les données échangées ensuite entre le client et le serveur sont ensuite chiffrées grâce à un algorithme de cryptographie.

## Méthode GET ou POST

Parmi les requêtes **HTTP** disponibles (**GET, HEAD, POST, OPTIONS, CONNECT, TRACE, PUT, PATCH, DELETE**)

- **GET** : C'est la méthode pour demander une ressource.
- **POST** : Cette méthode est utilisée pour soumettre des données en vue d'un traitement côté serveur.  
C'est la méthode employée lorsque l'on envoie au serveur les données issues d'un formulaire.

Formulaire : méthode GET	Formulaire : méthode POST
URL ⓘ Non sécurisé   claudia.com/contact.php?nom=toulouse31&satisf=1	URL ⓘ Non sécurisé   claudia.com/contact.php

## Requête HTTP

Ligne de requête	GET /contact.php?nom=toulouse31&satisf=1 HTTP/1.1
Entête de requête	Host: www.claudia.com Accept : text/html, text/CSS, image/jpg
Saut de ligne	
Corps de requête	

```
<h2> Evaluez votre visite </h2>
<form action="contact.php" method="get">
  <p> Pseudo: <input type="text" name="nom"/> </p>
  <p> Votre niveau de satisfaction : </br>
  <input type="radio" name="satisf" value="0"/> Pas satisfait </br>
  <input type="radio" name="satisf" value="1"/> Moyennement satisfait </br>
  <input type="radio" name="satisf" value="2"/> Très satisfait </br>
  <br/>
  <input type="submit" value="Envoyer">
</form>
```

## Requête HTTP

Ligne de requête	POST /contact.php HTTP/1.1
Entête de requête	Host: www.claudia.com Accept : text/html, text/CSS, image/jpg
Saut de ligne	
Corps de requête	nom=toulouse31&satisf=1

```
<h2> Evaluez votre visite </h2>
<form action="contact.php" method="post">
  <p> Pseudo: <input type="text" name="nom"/> </p>
  <p> Votre niveau de satisfaction : </br>
  <input type="radio" name="satisf" value="0"/> Pas satisfait </br>
  <input type="radio" name="satisf" value="1"/> Moyennement satisfait </br>
  <input type="radio" name="satisf" value="2"/> Très satisfait </br>
  <br/>
  <input type="submit" value="Envoyer">
</form>
```