



# BTS DIGITAL CONTENT

Module WEB – Premier semestre 2021

Schlessier Claude  
claude@schlessier.lu

# Contenu du cours

## 1) Comprendre comment fonctionne Internet

- Comment est né Internet ?
- Quelle est l'infrastructure d'Internet ?
- Qu'est-ce-qu'un serveur et à quoi sert-il ?
- Quelle est la différence entre Internet et World Wide Web ?

### // Objectifs

- Comprendre les grandes lignes de ce qu'est Internet
- Savoir ce qu'est un nom de domaine
- Savoir ce qu'est une adresse IP et pourquoi elle est utilisée
- Connaître les types de serveurs DNS, MAIL, WEB, BD
- Pouvoir faire la différence entre Internet et World Wide Web



## 2) Comprendre comment une page web est affichée

- Comment accède-t-on au World Wide Web ?
- Qu'est-ce qui se passe derrière les coulisses quand on ouvre une page web ?
- Comment et sous quel format les données sont-elles transmises ?
- Quelles sont les technologies sous-jacentes d'un site web ?
- Quelle est la différence entre un site statique et un site dynamique ?

### // Objectif

- Savoir comment accéder au World Wide Web
- Connaître les principaux navigateurs
- Reconnaître les protocoles HTTP, HTTPS
- Savoir faire la différence entre client et serveur
- Savoir faire la différence entre site statique et site dynamique
- Savoir expliquer ce que signifie "hébergement"

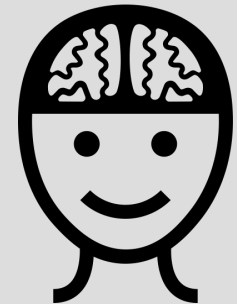


### 3) Comprendre comment développer un site web

- Qu'est-ce qu'un langage de balisage ?
- Qu'est-ce qu'un langage de programmation ?
- De quels outils a-t-on besoin pour créer un site web ?

#### // Objectif

- Savoir faire la différence entre langage de programmation / balisage
- Reconnaître des termes comme HTML, CSS, PHP, JavaScript
- Connaître les outils basiques pour créer un site
- Reconnaître le protocole FTP



## 4) Développer un site web statique

- Quelle est la structure d'un document HTML ?
- Quelles sont les différentes balises HTML ?
- Pourquoi et comment utiliser une feuille de style ?
- A quoi sert un framework et pourquoi l'utiliser ?
- A quoi sert le Responsive Design ?

### // Objectif

- Créer un site web basique
- Acquérir une bonne connaissance de la structure d'un document HTML
- Connaître les balises HTML
- Connaître la structure d'un fichier CSS
- Savoir formater un document HTML en utilisant du CSS
- Savoir dire à quoi servent les Media Queries et comment les utiliser
- Pouvoir nommer un framework et expliquer à quoi il sert



## 5) Utiliser du JavaScript pour rendre le site dynamique

- A quoi sert le Javascript et comment l'utiliser dans un document HTML ?
- Qu'est-ce qu'un langage interprété ?
- Qu'est ce qu'une variable ?
- Comment et quand utilise-t-on une boucle ou condition ?
- Qu'est ce qu'une fonction ?
- Découvrir le framework jQuery.

### // Objectif

- Rendre un site web dynamique
- Savoir dire à quoi sert le JavaScript
- Acquérir des connaissances basiques en JavaScript
- Savoir dire ce qu'est une variable, condition, boucle et fonction
- Pouvoir écrire un programme JavaScript simple

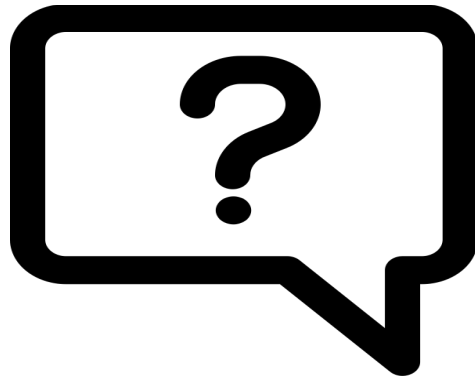


# Télécharger ce cours en ligne

<https://github.com/SchlessorClaude/bts-dc-web>

## Exemples CodePen

<https://codepen.io/xscctx/>



## Questions ?

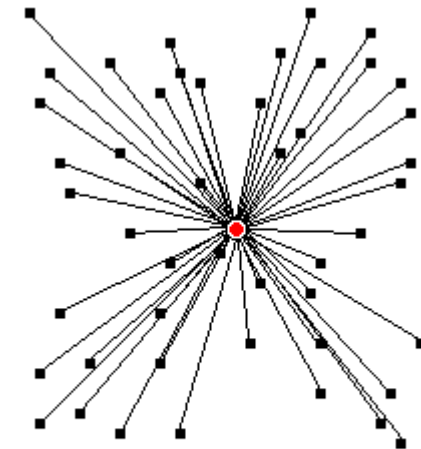
Claude Schlessor  
claude@schlessor.lu  
621.222.887

# 1) Comprendre comment fonctionne Internet

## 1.1) Comment est né Internet ?

A l'époque de la guerre froide, le gouvernement américain se demandait comment protéger l'appareil d'État en **créant un réseau de communication qui pouvait résister à une attaque** potentielle des Soviétiques.

La solution est venue de la Rand Corporation, le groupe d'experts de la guerre froide. En 1964, un chercheur du nom de Paul Baran proposa de **mettre en place un réseau de communication qui n'aurait aucun centre** et donc pas de point unique de défaillance.



**Centralized  
Network**

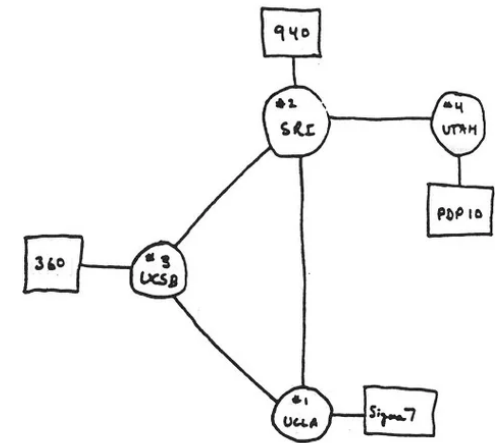
Il suffit de détruire le centre  
pour mettre le réseau entier  
hors service.



À l'origine, le réseau était censé **permettre aux chercheurs de l'Arpa (*Advanced Research Projects Agency*) de faire des calculs à distance**, c'est-à-dire sur des logiciels qu'ils ne possédaient pas, mais que leurs collègues, à l'autre bout du pays, pouvaient avoir sur leurs ordinateurs.

Au cours des années 1970, les chercheurs branchés sur l'ArpaNet ont trouvé une utilité nouvelle au réseau. Ils se sont mis à correspondre avec leurs collègues. **L'utilisation du réseau n'était donc plus limité aux calculs mais est devenu un moyen de télécommunication.**

Au fil du temps, **des universités américaines se sont progressivement reliées au réseau**. Chacune est devenu un nouveau nœud et a profité de l'occasion pour publier les travaux de ses chercheurs sur son répertoire FTP (File Transfer Protocol).



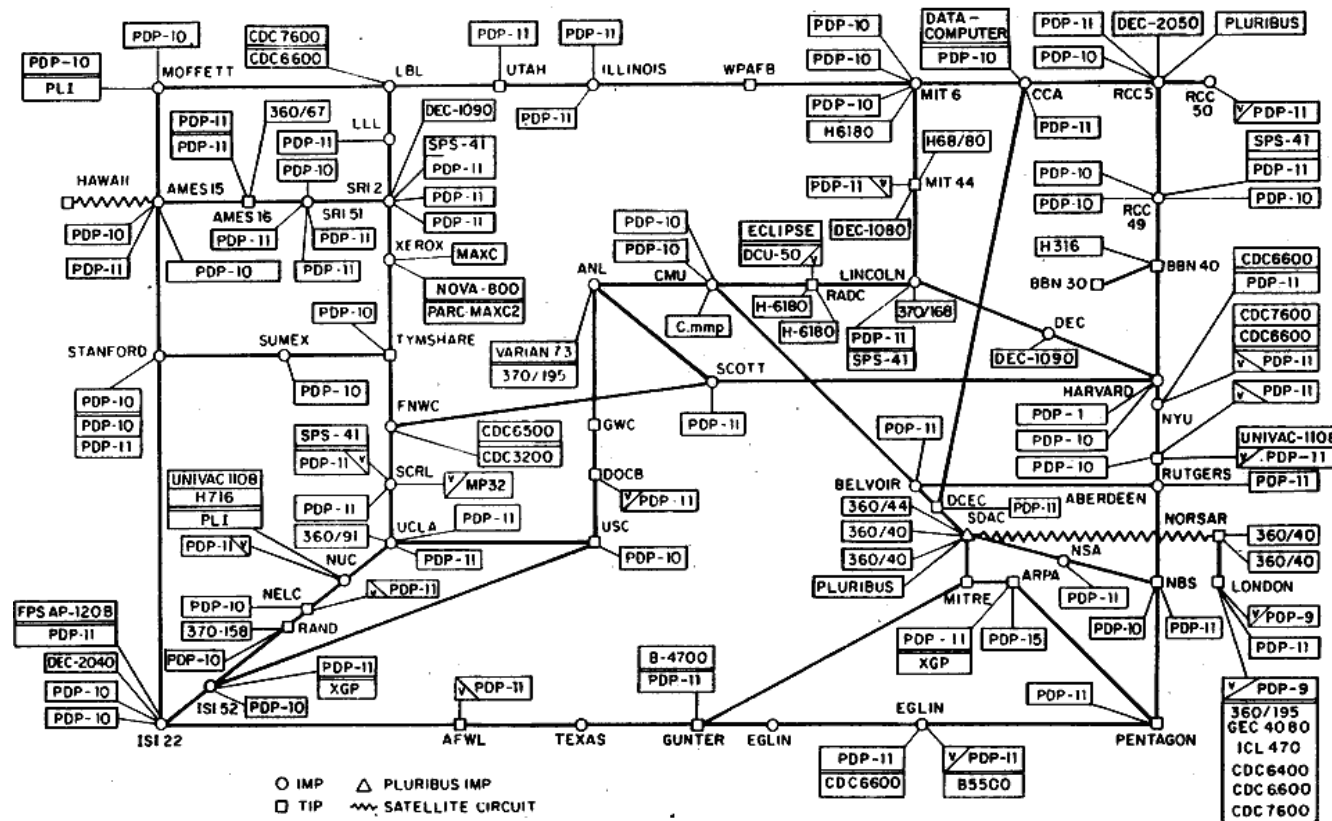
THE ARPA NETWORK

DEC 1969

4 NODES

En 1983, ArpaNet se détache du reste du réseau, qui devient alors l'Internet (*International Network* ou *Interconnected Network*).

ARPANET LOGICAL MAP, MARCH 1977



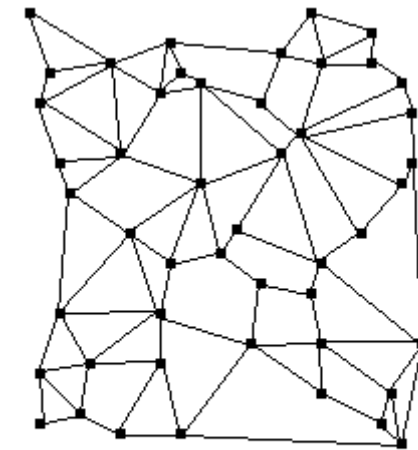
## 1.2) Quelle est l'infrastructure d'Internet ?

L'Internet est constitué de nœuds (*nodes* en anglais), tous égaux et reliés les uns aux autres. Ainsi, même si plusieurs d'entre eux sont défaillants ou détruits, le réseau entier reste fonctionnel.

**On appelle ceci un réseau distribué.**

Chaque machine connectée au réseau est identifiée par une adresse IP (Internet Protocol) unique, par exemple 168.212.226.204

Comme les adresses IP sont difficiles à retenir par les êtres humains, on utilise des **noms de domaine**, comme par exemple `www.google.fr`, dans le cadre du World Wide Web. Des serveurs DNS traduisent ces noms en adresses IP utilisées dans la communication entre clients et serveurs.



**Distributed  
Network**

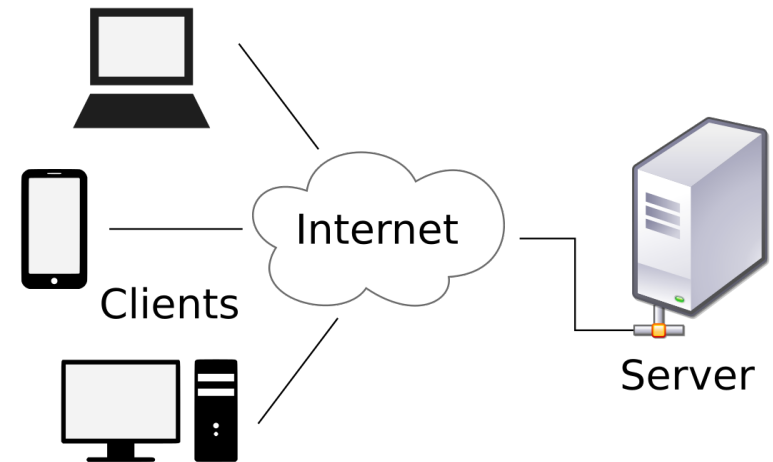
## 1.3) Qu'est-ce-qu'un serveur et à quoi sert-il ?

Le terme serveur désigne le rôle joué par un ordinateur connecté à un réseau Internet ou intranet et destiné à offrir des services à des clients.

Les services que peut rendre un serveur sont nombreux.

### **Voici quelques exemples :**

- Serveur FTP (Partage de fichiers)
- Serveur Mail (Envoi et collecte de courriels)
- Serveur de base de données (Stockage et mise à disposition de données)
- Serveur Web (Affichage de sites Internet)
- Serveur DNS (Traduction de nom de domaine en adresse IP)



## 1.4) Quelle est la différence entre Internet et World Wide Web ?

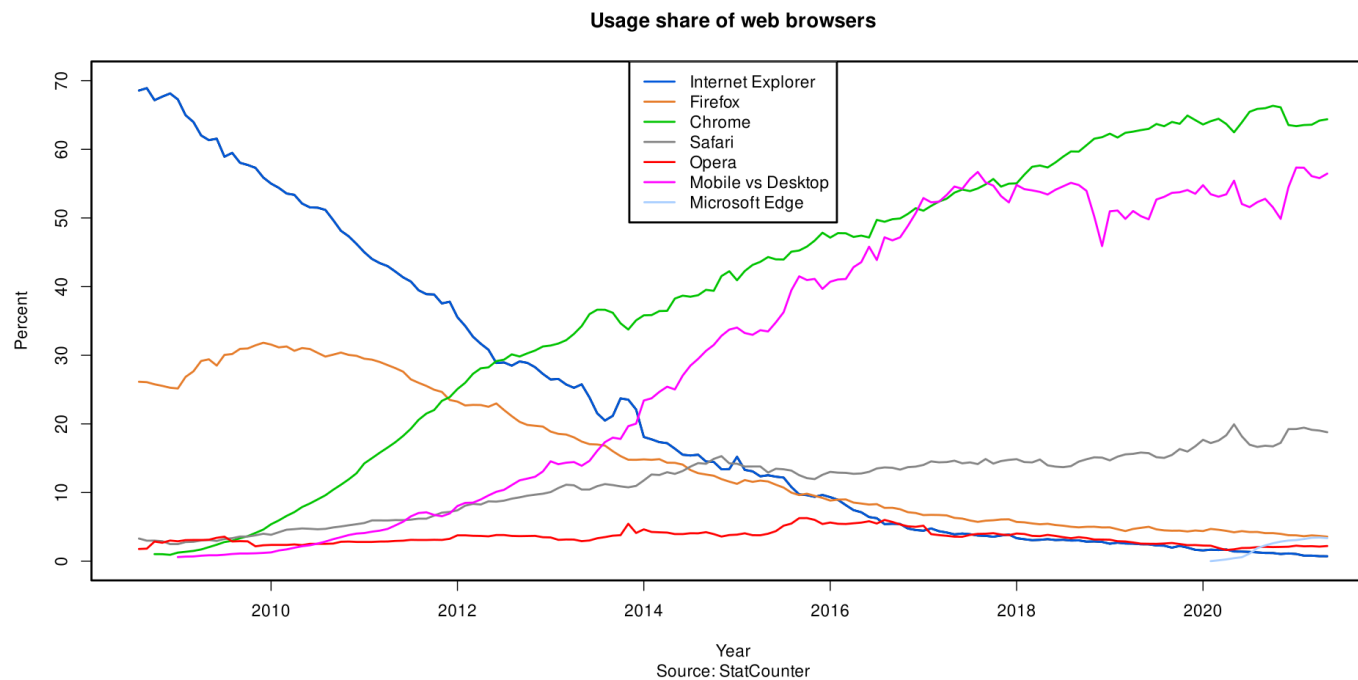
Internet	World Wide Web
On appelle Internet le réseau informatique mondial qui se base sur le protocole TCP/IP et sur lequel s'appuient différents services, dont le World Wide Web.	Système qui permet de naviguer de pages en pages en cliquant sur des liens dans un navigateur.

Quand on parle d'Internet, on parle uniquement du réseau, c'est-à-dire de l'infrastructure. Le World Wide Web n'est qu'un service entre autres proposé sur ce réseau. **Internet peut exister sans le Web, mais pas l'inverse.**

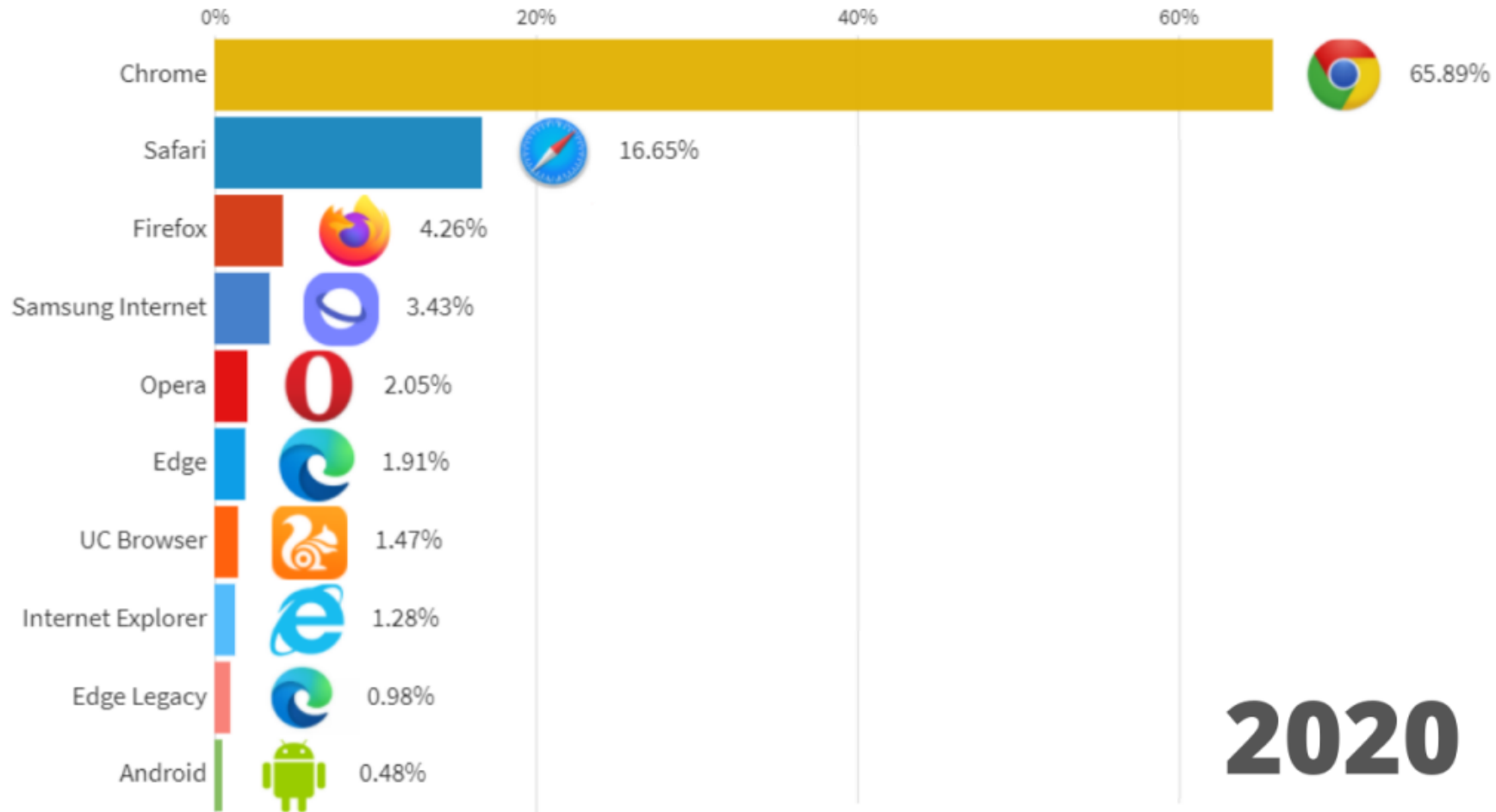
## 2) Comprendre comment une page web est affichée

### 2.1) Comment accède-t-on au World Wide Web ?

Pour accéder au World Wide Web, il faut disposer d'un ordinateur ou téléphone mobile connecté sur Internet et utiliser un logiciel qu'on appelle navigateur Web (browser en anglais) pour visualiser le contenu d'une page Internet.



## Part de marché des navigateurs Web dans le monde



**2020**

## 2.2) Qu'est-ce qui se passe derrière les coulisses quand on ouvre une page web ?

D'une manière générale, le navigateur Web va extraire des ressources et informations du Web en les téléchargeant (contenu HTML, CSS, Javascript, images, vidéos...) afin de les afficher sur l'écran de l'utilisateur.

Chaque page du World Wide Web est accessible via une **URL** (Uniform Resource Locator, littéralement *localisateur uniforme de ressource*). La façon la plus simple pour charger une page Web est de saisir l'adresse en question dans la **barre d'adresse** du navigateur.

Une URL est composée de plusieurs parties qui correspondent en général au schéma suivant :

Protocole	Service	Domaine de second niveau	Domaine de premier niveau	Dossier	Fichier
http://	www.	exemple	.com	/repertoire	/fichier.html



Après avoir inséré une URL dans la barre d'adresse du navigateur, celle-ci est envoyée à un routeur. Le **routeur** a ensuite la tâche de localiser l'adresse IP correspondante. Pour ceci, il va faire appel à un **serveur DNS** qui va traduire l'adresse Internet en adresse IP.

Lorsque le routeur a déterminé l'adresse IP du site Web désiré, celui-ci demande alors les données nécessaires pour afficher la page Web concernée. Cette requête se réalise via HTTP/HTTPS sous la forme d'un paquet de données. Celui-ci comporte toutes les informations dont le serveur Web a besoin pour construire le contenu.

Si la requête fonctionne, le serveur retournera les informations nécessaires pour afficher le contenu dans le navigateur.

Si le serveur n'arrive pas à localiser la page Internet, il affichera un code **erreur HTTP 404** (page introuvable) ou redirigera l'utilisateur vers une autre URL.

## 2.3) Comment et sous quel format les données sont-elles transmises ?

La plupart des informations transitant sur Internet (courriers électroniques, pages web...) dépassent largement les tailles maximales d'un paquet IP (taille moyenne 128 ou 256 octets).

Elles doivent donc être découpées en plusieurs paquets de taille appropriée par l'ordinateur expéditeur et reconstituées par l'ordinateur destinataire.

Pour rendre cela possible, le protocole TCP se charge de découper les données à envoyer en un ensemble de paquets IP.

Du côté du récepteur, le protocole TCP va réordonner les paquets IP reçus, en accuser la réception ou, au contraire, redemander ceux qui se seraient perdus et les réassembler pour reconstituer les données initiales.



## 2.4) Quelles sont les technologies sous-jacentes d'un site web ?

Les éléments invisibles aux yeux des internautes s'appellent le code source d'une page. Il est destiné aux navigateurs et codé grâce à plusieurs langages. Les 3 principaux sont **HTML**, **CSS** et **JavaScript**.

Ces langages indiquent au navigateur comment afficher les différents morceaux de la page web. Ils assurent la mise en page du contenu et de l'interface.

D'un point de vue technique, une page web est un fichier texte envoyé par un serveur à un navigateur.



## 2.5) Quelle est la différence entre un site statique et un site dynamique ?

**Les pages Web statiques** affichent exactement les mêmes informations chaque fois que quelqu'un les visite. Les pages Web statiques ne sont pas forcément que du texte. Elles peuvent contenir un design multimédia et même des vidéos.

Cependant, chaque visiteur sera accueilli par le même texte, le même design ou vidéo sauf si le code source de cette page est modifié.

**Les pages Web dynamiques** peuvent afficher un contenu différent pour chaque visiteur et ceci à partir du même de code source. Le site Web peut par exemple afficher un formulaire de connexion aux utilisateurs non-connectés et les données du compte aux utilisateurs connectés.

Le travail du webmaster est de concevoir une programmation qui permettra de présenter telle ou telle information selon la requête de l'internaute.

## 3) Comprendre comment développer un site web

### 3.1) Qu'est-ce qu'un langage de balisage ?

En informatique, un langage de balisage est un langage permettant de structurer (*ranger*) ou mettre en forme (*présenter*) des données en les organisant à l'aide de *b alises*.

**Pour construire des pages Web on utilise principalement le HTML et le CSS et dans certains cas le JSON et le XML.**

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a heading (first one) </h1>
<p> The paragraph </p>

</body>
</html>
```

Un langage de balisage n'est pas un langage de programmation: on dit que c'est un langage de *description* parce qu'il ne sert pas à écrire des programmes mais à *décrire* des données.

## 3.2) Qu'est-ce qu'un langage de programmation ?

Contrairement au langage de balisage, le langage de programmation dit précisément à un ordinateur ce qu'il doit faire et comment.

Ces instructions se font en utilisant des structures de contrôle comme par exemple des *conditions*, des *boucles* ou encore des *fonctions*.

Le langage de programmation ne décrit donc pas la structure des données, mais donne à l'ordinateur des **instructions** comment traiter ces données.

**Les langages de programmation les plus connues pour construire des pages Web sont le PHP et le JavaScript**

```
1 <?php
2 $str_isset = "";
3 $bol_isset = isset($str_isset);
4
5 If ($bol_isset){
6     echo "The variable is set";
7 }
8 else {
9     echo "The variable is not set";
10 }
11 ?>
12
```

### 3.3) De quoi a-t-on besoin pour créer un site web ?

La plupart des logiciels nécessaires au développement d'un site Web peuvent être téléchargés gratuitement sur Internet. Vous aurez besoin des outils suivants:

- **Editeur texte**  
Pour rédiger le code source.
- **Navigateur Web**  
Pour visualiser votre page Web.
- **Client FTP**  
Pour transférer vos fichiers vers un serveur web externe.

Tous les systèmes d'exploitation possèdent par défaut un éditeur de texte et un navigateur Web.

Seul l'outil qui permet de transférer les fichiers vers votre serveur Web pourrait manquer à l'appel.

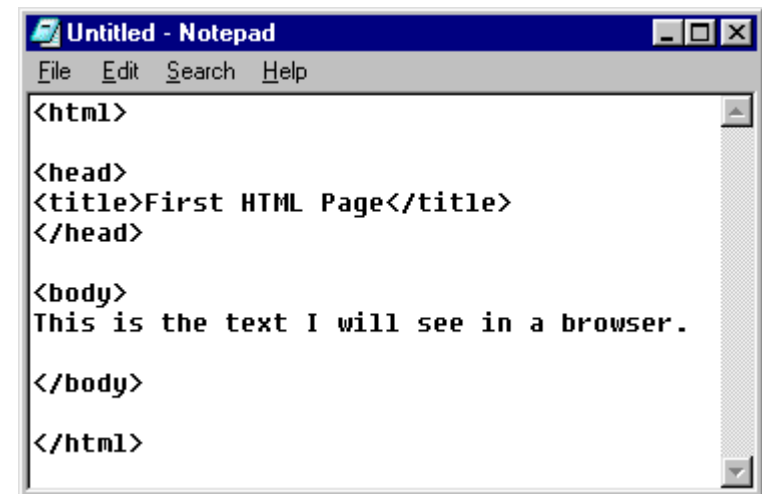


## - Quel éditeur de texte choisir ?

Les éditeurs de texte permettent de créer et de modifier des fichiers dont le contenu est du texte, **sans aucune mise en forme**. Les éditeurs qui permettent d'ajouter une mise en forme (comme le gras ou le soulignement) ne sont pas utilisables pour écrire des pages Web.

Presque tous les systèmes d'exploitations possèdent un éditeur de texte basique par défaut. Ces éditeurs sont plutôt simples à manipuler mais n'ont pas certaines fonctionnalités utiles au développement web.

Si vous souhaitez choisir un autre éditeur que celui par défaut, il y en a une myriade qui sont disponibles, dont certains gratuits.



```
<html>

<head>
<title>First HTML Page</title>
</head>

<body>
This is the text I will see in a browser.
</body>

</html>
```

*Les éditeurs de texte plus évolués proposent des fonctionnalités supplémentaires comme la coloration syntaxique, l'auto-complétion, le repli de sections, la recherche avancée, etc.*



Système d'exploitation	Éditeur natif	Éditeur tiers
Windows	- <a href="#">Bloc-notes</a>	<ul style="list-style-type: none"> <li>- <a href="#">Notepad++</a></li> <li>- <a href="#">Visual Studio Code</a></li> <li>- <a href="#">Web Storm</a></li> <li>- <a href="#">Brackets</a></li> <li>- <a href="#">ShiftEdit</a></li> <li>- <a href="#">Sublime Text</a></li> </ul>
Mac OS	- <a href="#">TextEdit</a>	<ul style="list-style-type: none"> <li>- <a href="#">TextWrangler</a></li> <li>- <a href="#">Visual Studio Code</a></li> <li>- <a href="#">Brackets</a></li> <li>- <a href="#">ShiftEdit</a></li> <li>- <a href="#">Sublime Text</a></li> </ul>
Linux	<ul style="list-style-type: none"> <li>- <a href="#">Vi</a> (tout système UNIX)</li> <li>- <a href="#">GEdit</a> (Gnome)</li> <li>- <a href="#">Kate</a> (KDE)</li> <li>- <a href="#">LeafPad</a> (Xfce)</li> </ul>	<ul style="list-style-type: none"> <li>- <a href="#">Emacs</a></li> <li>- <a href="#">Vim</a></li> <li>- <a href="#">Brackets</a></li> <li>- <a href="#">Sublime Text</a></li> </ul>
Chrome OS		- <a href="#">ShiftEdit</a>

## - Comment transférer mes fichiers vers un serveur Web ?

Lorsque votre site Web est prêt à être publié, vous devrez transférer (*uploader*) vos fichiers vers votre serveur web. Les informations d'accès FTP (pour *File Transfer Protocol* ou protocole de transfert de fichiers) se composent généralement d'une URL SFTP, d'un nom d'utilisateur et d'un mot de passe.

Le FTP est une technique vieillissante qui est souvent remplacée par des systèmes comme [RSync](#) ou encore [Git & Github](#).

Système d'exploitation	Client FTP	
Windows	- <a href="#">WinSCP</a>	- <a href="#">FileZilla</a> (tout système d'exploitation)
Linux	- <a href="#">Nautilus</a> (Gnome) - <a href="#">Konqueror</a> (KDE)	
Mac OS		

## 4) Développer un site Web statique

### 4.1) Quelle est la structure d'un document HTML ?

HTML (HyperText Markup Language) n'est pas un langage de programmation : c'est un *langage de balisage* qui sert à indiquer au navigateur comment structurer les pages Web visitées.

**Il peut être aussi compliqué ou aussi simple que le développeur web souhaite qu'il soit.**

Le HTML se compose d'une série d'éléments avec lesquels vous pouvez encadrer, envelopper ou *baliser* différentes parties du contenu pour les faire apparaître d'une certaine manière.

Des balises encadrantes peuvent transformer une petite partie de contenu en un lien vers une autre page sur le Web, mettre des mots en italique, etc.



Par exemple, prenons la phrase suivante :

```
Mon chat est très grincheux
```

Si nous voulons que cette ligne soit affichée en gras par le navigateur Web, nous pouvons l'envelopper dans une balise *strong*.

```
<strong>Mon chat est très grincheux</strong>
```

**Note :** Les éléments HTML sont insensibles à la casse, c'est-à-dire qu'ils peuvent être écrits en majuscules ou en minuscules. Par exemple, un élément `<strong>` peut être écrit `<strong>`, `<STRONG>`, `<sTrOnG>`, `<STRong>`, etc. et il fonctionnera parfaitement. La meilleure pratique, cependant, est d'écrire tous les éléments en minuscules pour des raisons de cohérence et de lisibilité.

## - Quelle est l'anatomie d'un élément HTML ?

(1) Élément HTML		
(2) Balise ouvrante	(3) Contenu	(4) Balise fermante
<code>&lt;strong&gt;</code>	Mon chat est très grincheux	<code>&lt;/strong&gt;</code>

### 1. L'élément

L'ensemble balise ouvrante, balise fermante et contenu constituent l'élément.

### 2. La balise ouvrante

Il s'agit du nom de l'élément (dans ce cas, `strong`), encadré par un **chevron ouvrant** (`<`) et un **chevron fermant** (`>`). La balise ouvrante indique où l'élément commence ou commence à prendre effet — dans ce cas où commence le texte en gras ;

### 3. Le contenu

Il s'agit du contenu de l'élément. Dans notre cas, c'est simplement du texte ;

### 4. La balise fermante

C'est la même que la balise ouvrante, sauf qu'elle comprend une **barre oblique** (`/`) avant le nom de l'élément. Elle indique la fin de l'élément — dans ce cas, la fin du texte en gras. Ne pas inclure une balise de fermeture est une erreur fréquente chez les débutants, et peut amener des résultats étranges ;

## - Créez votre premier élément HTML

a) Allez sur la page suivante: <https://codepen.io/pen/>

b) Saisissez l'élément suivant dans la zone HTML :

```
<strong>Mon chat est très grincheux</strong>
```

c) Le résultat devrait s'afficher la manière suivante :

**Mon chat est très grincheux**

d) Essayez maintenant de modifier le code afin d'afficher le mot *très* en italique (balise *em*).

e) Le résultat devrait s'afficher de la manière suivante :

**Mon chat est *très* grincheux**

## - Qu'est-ce-qu'on appelle des éléments imbriqués ?

Dans l'exercice précédent on a vu qu'on peut mettre des éléments à *l'intérieur* d'autres éléments — cela s'appelle **l'imbrication**.

```
<strong>Mon chat est <em>très</em> grincheux</strong>
```

**Vous devez vous assurer que vos éléments soient correctement imbriqués.** Dans l'exemple ci-dessus, nous avons ouvert l'élément **strong** en premier, puis l'élément **em**. Nonc nous devons fermer l'élément **em** d'abord, puis l'élément **strong**.

Ce qui suit est incorrect :

```
<strong>Mon chat est <em>très</strong> grincheux</em>
```

Si les balises se chevauchent comme dans l'exemple ci-dessus, votre navigateur Web essaiera de deviner ce que vous vouliez dire, et vous allez obtenir des résultats inattendus.

## - Pourquoi indenter son code ?

Pour garder un œil sur le niveau d'imbrication, on peut **indenter** (*indentation*) le code.



Indenter correspond à créer des retraits en début de ligne dans votre éditeur et ceci de façon cohérente et logique.

Le but de l'indentation est de pouvoir **discerner plus facilement les différents éléments** ou parties du code et de **détecter les erreurs** potentielles. Indenter va également nous permettre d'avoir un **code plus propre et plus lisible**, donc plus compréhensible.

**Note :** Les retours à la ligne et l'indentation créés dans l'éditeur n'affectent pas le résultat final dans le navigateur Web.



Sans indentation :

```
<!DOCTYPE html>
<html>
<head>
<title>Ma première page HTML</title>
<meta charset="utf-8">
</head>
<body>
<h1>Je viens d'écrire un titre en HTML !</h1>
<p>Et voilà mon premier paragraphe :)</p>
</body>
</html>
```

Le même code avec indentation :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première page HTML</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>Je viens d'écrire un titre en HTML !</h1>
    <p>Et voilà mon premier paragraphe :)</p>
  </body>
</html>
```

## - Les retours à la ligne et espaces dans un fichier HTML

Vous pouvez ajouter autant d'espaces que vous le voulez au sein d'un paragraphe ou d'un titre ou effectuer des retours à la ligne dans votre code, ceux-ci ne seront jamais affichés visuellement dans votre navigateur.

**En HTML, les espaces (espaces, tabulations, retours à la ligne) consécutifs sont équivalents à un seul espace.**

Pour insérer plusieurs espaces consécutifs, il est nécessaire d'utiliser une entité HTML spécifique, appelée espace insécable (&nbsp; pour *non-breaking space*), dont le code HTML est le suivant :

```
&nbsp;
```

Exemple:

```
<strong>Mon chat est <em>très</em> &nbsp;&nbsp;&nbsp;&nbsp; grincheux</strong>
```

## - Quelle est la différence entre éléments bloc et éléments en ligne ?

Il existe deux catégories importantes d'éléments en HTML que vous devez connaître :

**Les éléments de niveau bloc** (*block* en anglais) forment un bloc visible sur une page.

[https://developer.mozilla.org/fr/docs/Web/HTML/Block-level\\_elements](https://developer.mozilla.org/fr/docs/Web/HTML/Block-level_elements)

**Les éléments en ligne** (*inline* en anglais) entourent des petites parties du contenu du document et sont imbriqués dans des éléments de niveau bloc.

[https://developer.mozilla.org/fr/docs/Web/HTML/Inline\\_elements](https://developer.mozilla.org/fr/docs/Web/HTML/Inline_elements)

### Exemples d'éléments bloc :

```
<div>
```

```
<p>
```

```
<section>
```

```
<h1> ... <h6>
```

### Exemples d'éléments en ligne :

```
<strong>
```

```
<a>
```

```
<em>
```

```
<span>
```

```
<small>
```

## - Les éléments de niveau bloc

Les éléments de niveau bloc forment **un bloc visible sur une page** — ils apparaissent sur une nouvelle ligne quel que soit le contenu précédant et tout contenu qui les suit apparaît également sur une nouvelle ligne. Exemple : `<div> ... </div>`, `<p> ... </p>`.

Les éléments de niveau de bloc sont souvent des éléments structurels de la page et représentent, par exemple, *des paragraphes, des listes, des menus de navigation, des pieds de page*, etc

```
<p>
  Le chat est grincheux.
</p>
<p>
  La souris est contente.
</p>
```

**Important :** Un élément de niveau bloc ne peut pas être imbriqué dans un élément en ligne, mais il peut être imbriqué dans un autre élément de niveau bloc.

## - Les éléments en ligne

Les éléments en ligne sont **contenus dans des éléments de niveau bloc**. Ils entourent seulement des petites parties du contenu du document, ni des paragraphes entiers, ni des regroupements de contenu.

Un élément en ligne ne fait pas apparaître une nouvelle ligne dans le document. Il apparaît généralement dans un paragraphe de texte, par exemple un élément (hyperlien) **<a>** ou des éléments de mise en évidence tels que **<em>** ou **<strong>**.

```
<p>  
  <strong>Le chat est grincheux.</strong>  
</p>  
<p>  
  <em>La souris est contente.</em>  
</p>
```

**Important :** Un élément en ligne doit être imbriqué dans un élément bloc et peut être imbriqué dans un autre élément en ligne.

## Exercices E1

Allez sur la page suivante: <https://codepen.io/pen/> et essayez de reproduire le contenu ci-dessous.

a) Reproduisez l'affichage suivant :

```
Le  
chat  
est  
très  
grincheux
```

b) Reproduisez l'affichage suivant :

```
Le  
chat  
est  
très  
grincheux
```

c) Reproduisez l'affichage suivant :

Le chat  
est très *grincheux*

d) Reproduisez l'affichage suivant (nouveau élément bloc : `<hr />`) :

Le chat  

---

est très *grincheux*

e) Reproduisez l'affichage suivant (nouveau élément en ligne: `<small>...</small>`) :

La souris est très *petite*

f) Reproduisez l'affichage suivant :

La souris est                      très *petite*

g) Reproduisez l'affichage suivant (nouveaux éléments bloc: **<h1>**, **<h2>** ...**<h6>**) :

```
1  
2  
3  
4  
5  
6
```

<https://codepen.io/collection/jbPRgM>



## - Hiérarchie structurale des documents HTML

Dans une histoire, la balise `<h1>` représenterait le titre de l'histoire, les balises `<h2>` représenteraient les titres des chapitres, les balises `<h3>` les sous-sections des chapitres, en poursuivant ainsi jusqu'à la balise `<h6>`. Voici un exemple :

# **L'ennui mortel**

par Chris Mills

## **Chapitre I : La nuit noire**

Il faisait nuit noire. Quelque part une chouette ululait. La pluie tombait sur ...

## **Chapitre II : Le silence éternel**

Notre protagoniste ne pouvait même pas murmurer à l'ombre de la silhouette...

### **Le spectre parle**

Plusieurs heures s'étaient écoulées, quand soudain le spectre assis se releva et s'exclama : « S'il vous plaît, ayez pitié de mon âme ! »...

```
<h1>
    L'ennui mortel
</h1>
<p>
    par Chris Mills
</p>

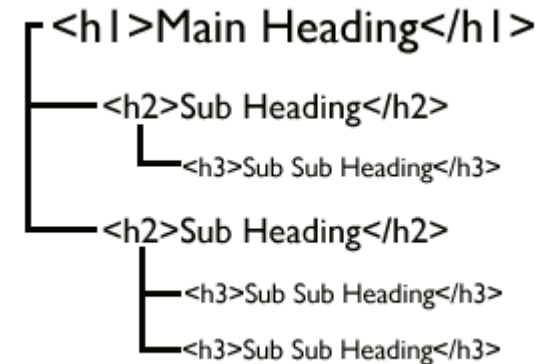
<h2>
    Chapitre I : La nuit noire
</h2>
<p>
    Il faisait nuit noire. Quelque part une chouette ululait. La pluie tombait sur ...
</p>

<h2>
    Chapitre II : Le silence éternel
</h2>
<p>
    Notre protagoniste ne pouvait même pas murmurer à l'ombre de la silhouette...
</p>

<h3>
    Le spectre parle
</h3>
<p>
    Plusieurs heures s'étaient écoulées, quand soudain le spectre assis se releva et
    s'exclama : « S'il vous plaît, ayez pitié de mon âme ! »...
</p>
```

**C'est vous qui décidez** ce que représentent les éléments utilisés tant que la hiérarchie a du sens. Vous devez cependant garder à l'esprit quelques bonnes pratiques lorsque vous créez de telles structures :

- Il est préférable de n'utiliser qu'**un seul <h1> par page**. C'est le niveau principal, et tous les autres devraient être de niveau inférieur.
- Assurez-vous d'utiliser les balises de titre dans l'**ordre correct et logique**. En premier lieu **<h1>**, après **<h2>**, puis **<h3>** et ainsi de suite.

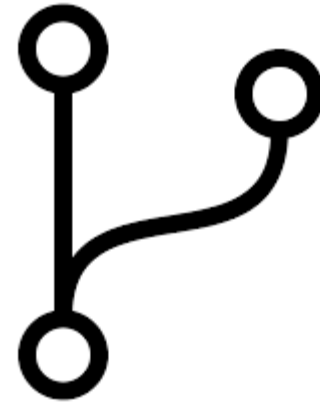


Bien qu'il y ait six niveaux de titres (de **<h1>** à **<h6>**), **il est déconseillé d'utiliser plus de trois niveaux** dans une page. Les documents avec beaucoup de niveaux deviennent complexes et difficiles à parcourir. Dans ce cas, il est préférable de partager le contenu sur plusieurs pages.

## Exercice E2

Allez d'abord sur la page <https://codepen.io/xscsx/pen/LYjNVVE> et créez une copie du code en cliquant sur le bouton **Fork** en bas à droite.

Le **Fork** va créer une copie du code et l'ajouter à votre espace CodePen personnel. Vous pouvez y apporter des modifications sans changer le code source originel.



**a)** Rajoutez le paragraphe suivant au **Chapitre I** :

La lune éclaircissait les arbres.

**b)** Rajoutez le **Chapitre III : Le soleil se lève** avec les deux paragraphes suivants :

Le soleil se levait à l'horizon.

Une nouvelle journée commença.

## 4.2) Éléments HTML

### • Les éléments `<h1>` ... `<h6>`

Les éléments `<h1>` à `<h6>` sont des éléments bloc qui permettent de définir certains textes comme des **titres ou sous-titres** pour le contenu. D'une certaine façon, ceux-ci fonctionnent comme pour un livre : on a le titre du livre puis les titres des différents chapitres et parfois des sous-titres au sein de ces chapitres.

```
<h1>Mon titre principal</h1>  
<h2>Mon titre de section</h2>  
<h3>Mon sous-titre</h3>
```

<https://codepen.io/xscsx/pen/RwZMmzb>

### • L'élément `<p>`

Les éléments `<p>` sont des éléments bloc qui sont utilisés pour contenir des paragraphes de texte. Vous les utiliserez fréquemment pour placer du texte sur une page.

```
<p>Le chat est grincheux.</p>
```

<https://codepen.io/xscsx/pen/XWaEwLY>

## • L'élément **<strong>**

L'élément **<strong>** est un élément en ligne qui indique que le texte a une importance particulière. Cela se traduit généralement par un affichage en **gras**.

```
Le chat est <strong>très</strong> grincheux.
```

<https://codepen.io/xsccsx/pen/yLoKdBP>

## • L'élément **<em>**

L'élément **<em>** (*emphase*) est un élément en ligne qui est utilisé afin de marquer un texte sur lequel on veut insister. Cela se traduit généralement par un affichage en *italique*.

```
Le chat a faim. Il faut lui donner à manger <em>maintenant</em>.
```

<https://codepen.io/xsccsx/pen/XWaELrY>

## • L'élément **<hr>**

L'élément **<hr>** représente un **changement thématique** entre des éléments de paragraphe (par exemple, un changement de décor dans un récit, un changement de sujet au sein d'une section). C'est un élément HTML vide qui ne peut pas avoir de nœud enfant.

```
<h1>
  The main languages of the Web
</h1>
<p>
  HTML is the standard markup language for creating Web pages. HTML describes the structure of
  a Web page, and consists of a series of elements. HTML elements tell the browser how to
  display the content.
</p>

<hr />

<p>
  CSS is a language that describes how HTML elements are to be displayed on screen, paper, or
  in other media. CSS saves a lot of work, because it can control the layout of multiple web
  pages all at once.
</p>
```

<https://codepen.io/xsccsx/pen/PoKRvvp>

## • L'élément **<div>**

Les éléments **<div>** ressemblent fortement aux éléments **<p>**, leur différence est de niveau sémantique.

- L'élément **<p>** sert à décrire un **paragraphe de contenu** – comme un paragraphe dans un livre. Les principaux navigateurs Web afficheront une marge au-dessus et en dessous du paragraphe.
- L'élément **<div>** sert à décrire un **conteneur de données** – comme le cadre d'un tableau. Une **<div>** balise sera affichée sans aucune marge.

```
<div class="recette">
  <h1>Recette pain de campagne</h1>
  <p>
    Pour faire du pain on a besoin de 500g de
    farine, 20g de levure, 250ml d'eau et d'1
    cuillère à café de sel.
  </p>
</div>
```

```
.recette{
  border:1px solid black;
  padding: 10px;
}
```

<https://codepen.io/xsccsx/pen/ExvEBad>

- Si vous avez besoin d'un **conteneur générique** pour la mise en page, utilisez un **<div>**
- Si vous avez besoin d'un élément pour **décrire un paragraphe de contenu**, utilisez un **<p>**



## • L'élément `<span>`

L'élément `<span>` sert à décrire un conteneur de données, exactement comme l'élément `<div>`. La différence est que l'élément `<div>` est un élément bloc, alors que `<span>` est un élément en ligne.

Les éléments `<div>` et `<span>` sont utilisés pour regrouper des éléments afin de les mettre en forme, et ceci en utilisant des attributs `class` ou `id` et des règles CSS.

```
<div class="recette">
  <h1>Recette pain de campagne</h1>
  <p>
    Pour faire du pain on a besoin de <span
class="ingredient">500g de farine</span>,
    <span class="ingredient">20g de
    levure</span>, <span class="ingredient">250ml
    d'eau</span> et d'<span
class="ingredient">1 cuillère à café de
    sel</span>.
  </p>
</div>
```

```
.recette{
  border:1px solid black;
  padding: 10px;
}

.ingredient{
  color: red;
  font-weight:bold;
}
```

<https://codepen.io/xsccsx/pen/porLmOP>

**Exercice :** Affichez toutes les valeurs (500g, 20g ... ) en bleu

## • L'élément `<ul>`

L'élément HTML `<ul>` (*unordered list*) représente une liste d'éléments pour lesquelles **l'ordre n'a pas d'importance**, comme par exemple une liste de courses. Chaque objet de la liste doit être enveloppé dans un élément HTML `<li>` (*list item*).

Dans les menus et les listes non ordonnées, les différents éléments sont habituellement affichés en utilisant des puces.

```
<ul>
  <li>farine</li>
  <li>levure</li>
  <li>eau</li>
  <li>sel</li>
</ul>
```

<https://codepen.io/xsccsx/pen/GRvxbQN>

**Exercice** : Affichez les ingrédients de la recette de pain en tant que liste non ordonnée.

## • L'élément **<ol>**

L'élément HTML **<ol>** (*ordered list*) représente une liste d'éléments pour lesquels **l'ordre a une importance**, comme par exemple la liste des pays les plus grands du monde. Chaque objet doit être enveloppé dans un élément HTML **<li>** (*list item*).

Dans les listes ordonnées, les différents éléments sont habituellement affichés avec un compteur croissant à gauche, tel qu'un nombre ou une lettre.

```
<ul>
  <li>Russie - 17 1251 91 km2</li>
  <li>Canada - 9 984 670 km2</li>
  <li>États-Unis - 9 629 091 km2</li>
  <li>Chine - 9 600 000 km2</li>
</ul>
```

<https://codepen.io/xsccsx/pen/KKvojoR>

## • L'élément **<table>**

L'élément HTML **<table>** permet de créer un ensemble structuré de données présentées en lignes et colonnes (tableau).

```
<table>
  <thead>
    <tr>
      <th>Personne</th>
      <th>Âge</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Chris</td>
      <td>25</td>
    </tr>
    <tr>
      <td>Sarah</td>
      <td>29</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Moyenne</td>
      <td>27</td>
    </tr>
  </tfoot>
</table>
```

```
table,
td {
  border: 1px solid black;
}

thead{
  background-color: black;
  color: white;
}

td,th{
  width: 200px;
  text-align:center;
  height: 40px;
}

tfoot{
  background-color: gray;
  color: white;
}
```

<https://codepen.io/xsccsx/pen/XWaEvVX>

Un tableau vous permet de retrouver rapidement et facilement des valeurs au croisement entre différents types de données, par exemple : une personne et son âge, ou un jour et la semaine, ou les horaires d'ouverture de la piscine du quartier.

**Les tableaux HTML ne doivent être utilisés que pour des données tabulaires.** Il ne faut pas utiliser les tableaux HTML pour organiser des pages Web, par exemple : une ligne pour contenir l'en-tête, une ligne pour les colonnes de contenu, une ligne pour le pied de page, etc.

## Liste des éléments HTML qui constituent un tableau :

Élément	Description
<b>table</b>	Définit un <b>tableau</b> .
<b>tr</b>	Définit une <b>ligne de cellules</b> d'un tableau. Une ligne peut être constituée d'éléments <b>&lt;td&gt;</b> ou d'éléments <b>&lt;th&gt;</b> .
<b>th</b>	Définit une cellule d'un tableau comme une <b>cellule d'entête</b> .
<b>td</b>	Définit une cellule d'un tableau comme une <b>cellule normale</b> .
<b>thead</b>	Permet de regrouper un ou plusieurs éléments <b>&lt;tr&gt;</b> afin de former l' <b>entête du tableau</b> .
<b>tbody</b>	Permet de regrouper un ou plusieurs éléments <b>&lt;tr&gt;</b> afin de former le <b>corps du tableau</b> .
<b>tfoot</b>	Permet de regrouper un ou plusieurs éléments <b>&lt;tr&gt;</b> afin de former le <b>résumé du tableau</b> .

# Application pratique

1) Allez sur la page <https://codepen.io/xscctx/pen/porLMOz> et créez une copie du code en cliquant sur le bouton **Fork**.

2) Le plus petit conteneur d'un tableau est la cellule ; elle est créée avec l'élément **<td>**.

Ajoutez ceci entre les balises du tableau :

```
<td>Première cellule</td>
```

3) Si nous voulons une rangée de quatre cellules, nous devons copier la première trois fois.

Mettez à jour le contenu du tableau et rajoutez les cellules suivantes :

```
<td>Deuxième cellule</td>
```

```
<td>Troisième cellule</td>
```

```
<td>Quatrième cellule</td>
```

Comme vous le verrez, les cellules ne sont pas placées les unes en dessous des autres, mais elles sont automatiquement affichées dans une même ligne. Chaque élément **<td>** crée une cellule simple et ensemble elles forment la première ligne. Toutes les cellules que nous ajoutons allongent la ligne.

#### 4) Pour placer des cellules sur des lignes distinctes, nous devons utiliser la balise **<tr>**.

Regroupez les cellules en utilisant des balises **<tr>** ... **</tr>** :

```
<tr>
  <td>Première cellule</td>
  <td>Deuxième cellule</td>
</tr>
<tr>
  <td>Troisième cellule</td>
  <td>Quatrième cellule</td>
</tr>
```

#### 5) Pour augmenter la lisibilité, nous pouvons ajouter des en-têtes de tableau en utilisant la balise **<th>**.

Cette balise fonctionne exactement comme la balise **<td>**, à ceci près qu'elle indique une en-tête et non une cellule normale.

Ajoutez code HTML suivant directement après la balise **<table>** :

```
<tr>
  <th>Mes cellules</th>
</tr>
```



Vous allez constater que l'entête ne couvre qu'une seule colonne.

**6) Si vous voulez qu'une cellule (<td> / <th>) couvre plusieurs colonnes, vous devez ajouter l'attribut *colspan*="x".**

La valeur **x** de l'attribut est un nombre entier qui représente le nombre de *colonnes* à couvrir.

Modifiez l'entête de votre tableau ainsi :

```
<tr>
  <th colspan="2">Mes cellules</th>
</tr>
```

Une colonne peut couvrir plusieurs cellules de la même ligne mais aussi plusieurs cellules sur des lignes différentes.

**7) Si vous voulez qu'une cellule (<td> / <th>) couvre plusieurs lignes, vous pouvez utiliser l'attribut *rowspan*="y".**

La valeur **y** de l'attribut est un nombre entier qui représente le nombre de *lignes* à couvrir.

Modifiez la première colonne <td> de votre tableau ainsi :

```
<td rowspan="2">Première cellule</td>
```

### 8) Avez-vous compris l'utilisation de l'attribut colspan ?

Allez sur la page <https://codepen.io/xscsxp/pen/porLMOz>, créez une copie du code en cliquant sur le bouton **Fork** et reproduisez la table suivante :

### Ma première table

Entête	Entête
Colonne	Colonne
Colonne	
Colonne	Colonne

### 9) Avez-vous compris l'utilisation de l'attribut rowspan ?

Allez sur la page <https://codepen.io/xscsxp/pen/porLMOz>, créez une copie du code en cliquant sur le bouton **Fork** et reproduisez la table suivante :

### Ma première table

Entête	Entête
Colonne	Colonne
	Colonne
Colonne	Colonne

## Exercice

Entête 1	Entête 2
Colonne1 rowspan="3"	Colonne2
	Colonne3
	Colonne4
Colonne5	Colonne6 rowspan="2"
Colonne7	
Colonne8 colspan="2"	
Colonne9	Colonne10

- Allez sur la page <https://codepen.io/xsccsx/pen/gOxyogo> créez une copie du code en cliquant sur le bouton **Fork** et reproduisez la table à côté.
- Vous pouvez trouver une liste des couleurs sous [https://www.w3schools.com/tags/ref\\_colornames.asp](https://www.w3schools.com/tags/ref_colornames.asp)

## • L'élément **<a>**

L'élément **<a>** nous permet de créer des **hyperliens**. Ces liens nous permettent de lier nos documents à n'importe quel autre document (ou autre ressource) voulu ; nous pouvons faire des liens vers des parties précises de documents et rendre des applications disponibles à une simple adresse web.

À peu près tout contenu web peut être converti en lien, de sorte que cliqué (ou activé autrement), il dirigera le navigateur vers une autre adresse web (URL).

**Une URL peut pointer vers des fichiers HTML, des fichiers textes, des images, des documents textuels, des fichiers vidéo ou audio et tout ce qui peut exister sur le Web.** Si le navigateur Web ne sait pas comment afficher ou gérer un fichier, il vous demande si vous voulez ouvrir le fichier (dans ce cas, la responsabilité de l'ouverture et de la gestion du fichier incombe à l'application native adéquate sur l'appareil) ou bien télécharger le fichier (auquel cas, vous pouvez essayer de vous en occuper plus tard).

## - L'attribut *href*

Un lien élémentaire se crée en intégrant le texte (ou tout autre contenu) que vous voulez transformer en lien dans un élément **<a>** et en lui affectant un attribut **href** (également connu comme étant une **Hypertext Reference**) contenant l'adresse web vers laquelle vous voulez que le lien pointe.

Je fais mes études au **<a href="https://www.lnbd.lu/">**Lycée Nic-Biever**</a>**

## - L'attribut *title*

L'autre attribut qu'il est possible d'ajouter à un lien est **title** ; il est destiné à contenir des informations utiles supplémentaires à propos du lien, comme par exemple les informations contenues dans la page de destination. Par exemple :

```
Je fais mes études au <a title="Informations" href="https://www.lnbd.lu/">Lycée Nic-Biever</a>
```

## - Différence entre URL absolue et URL relative

Les deux termes que vous rencontrerez sur le Web sont **URL absolue** et **URL relative** :

Une **URL absolue** pointe sur un emplacement défini de manière *absolue* sur le web, y compris en précisant le *protocole* et le *nom de domaine*, comme par exemple:

Fichier qui contient le lien :	http://www.exemple.lu/index.html
Lien absolu compris dans ce fichier :	<a href="http://www.google.de">Google</a>
Page sur laquelle l'utilisateur se trouve après avoir cliqué sur ce lien :	http://www.google.de

<http://www.example.com/projects/index.html>

<http://www.google.de>

<http://www.rtl.lu/news>

Une **URL relative** pointe vers un emplacement qui est *relatif* au fichier à partir duquel vous établissez le lien. Donc, si nous voulons créer un lien depuis un fichier vers un autre fichier PDF dans le **même** dossier, l'URL sera seulement le nom de ce fichier.

Fichier qui contient le lien :	http://www.exemple.lu/index.html
Lien relatif compris dans ce fichier :	<a href="informations.html">Google</a>
Page sur laquelle l'utilisateur se trouve après avoir cliqué sur ce lien :	http://www.exemple.lu/informations.html

index.html  
./cours/matieres.html