



# BTS DIGITAL CONTENT

Module WEB – Premier semestre 2021

Schlessen Claude  
claude@schlessen.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"

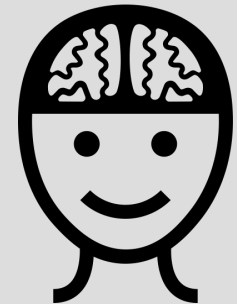


### 3) Comprendre comment développer un site web

- 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

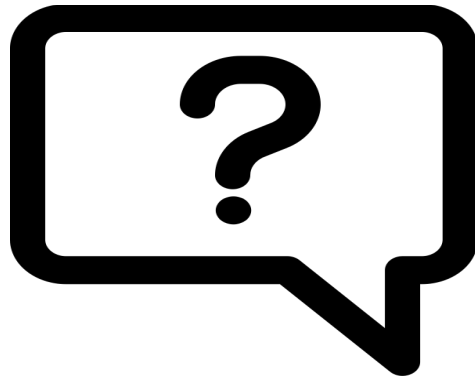


# Télécharger ce cours en ligne

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

## Exemples CodePen

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



## Questions ?

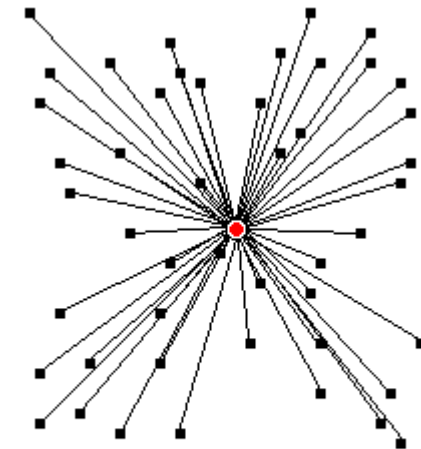
Claude Schlessor  
claude@schlessor.lu  
621.222.887

# 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

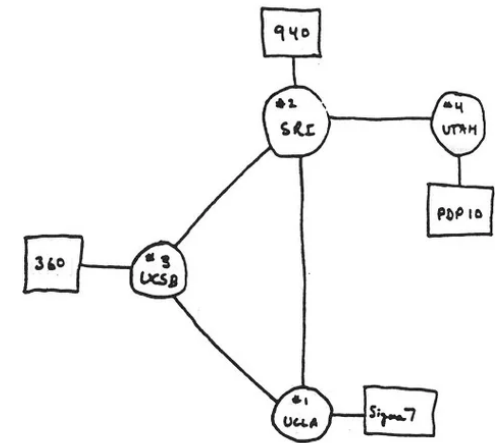
Il suffit de détruire le centre  
pour mettre le réseau entier  
hors service.



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 limitée 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).



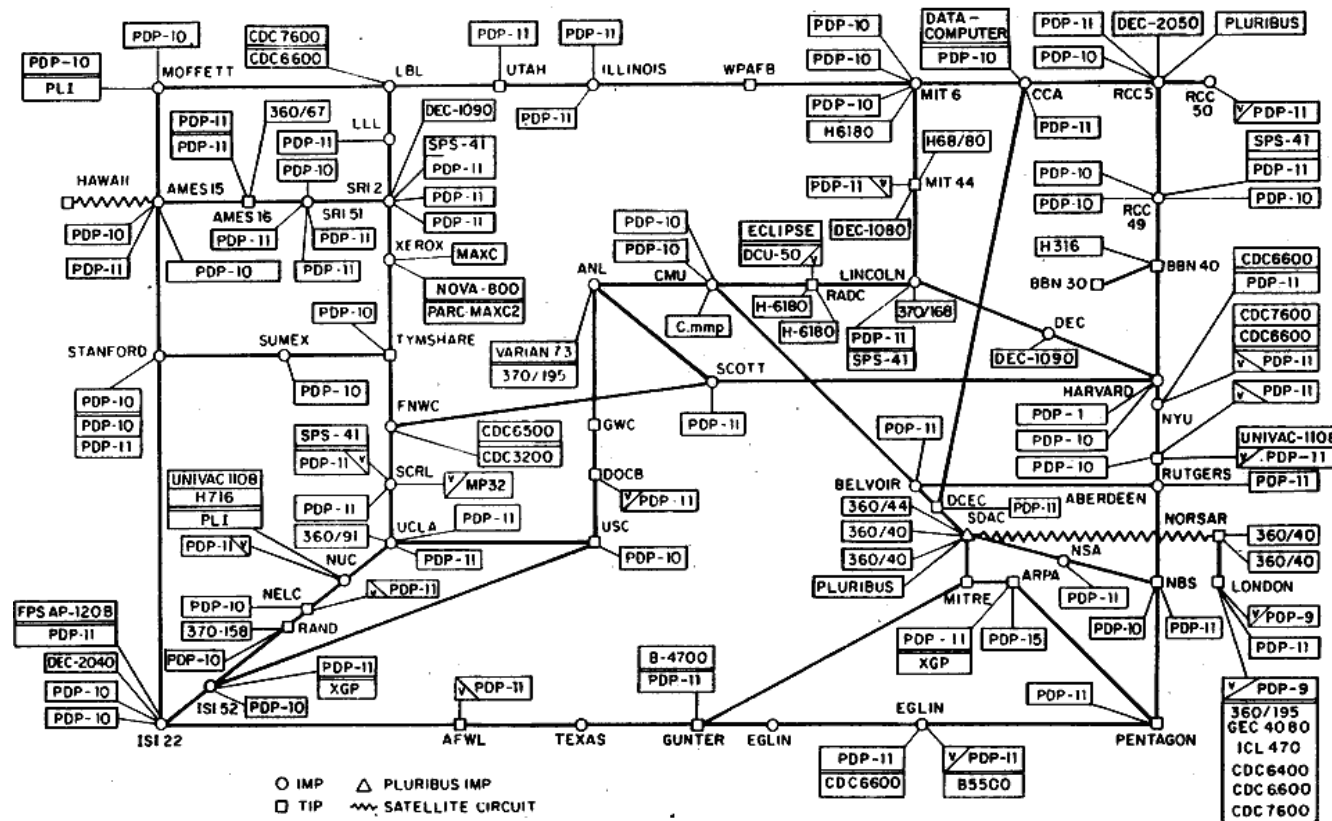
THE ARPA NETWORK

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



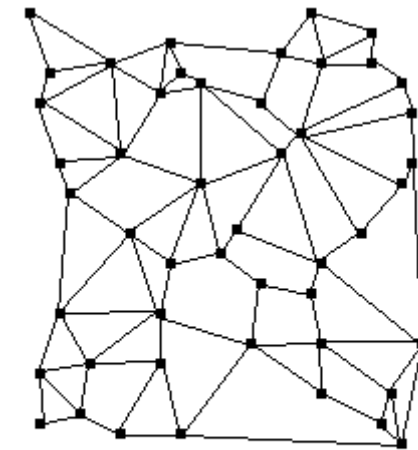
## 1.2) Quelle est l'infrastructure d'Internet ?

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 défectueux 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](http://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**

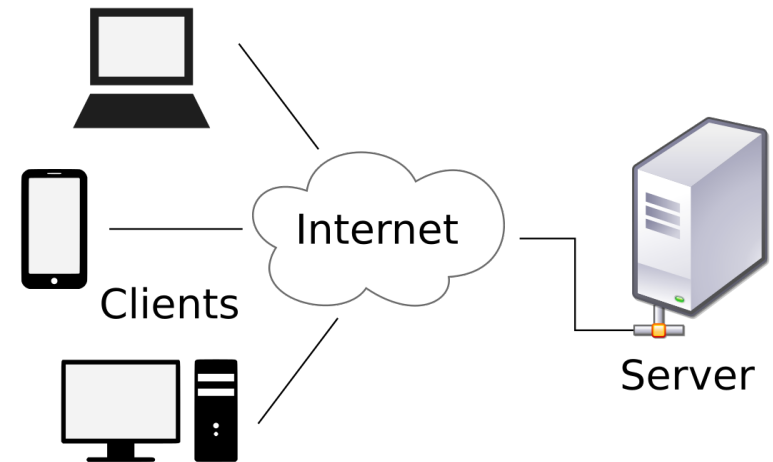
## 1.3) Qu'est-ce-qu'un serveur et à quoi sert-il ?

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 ?

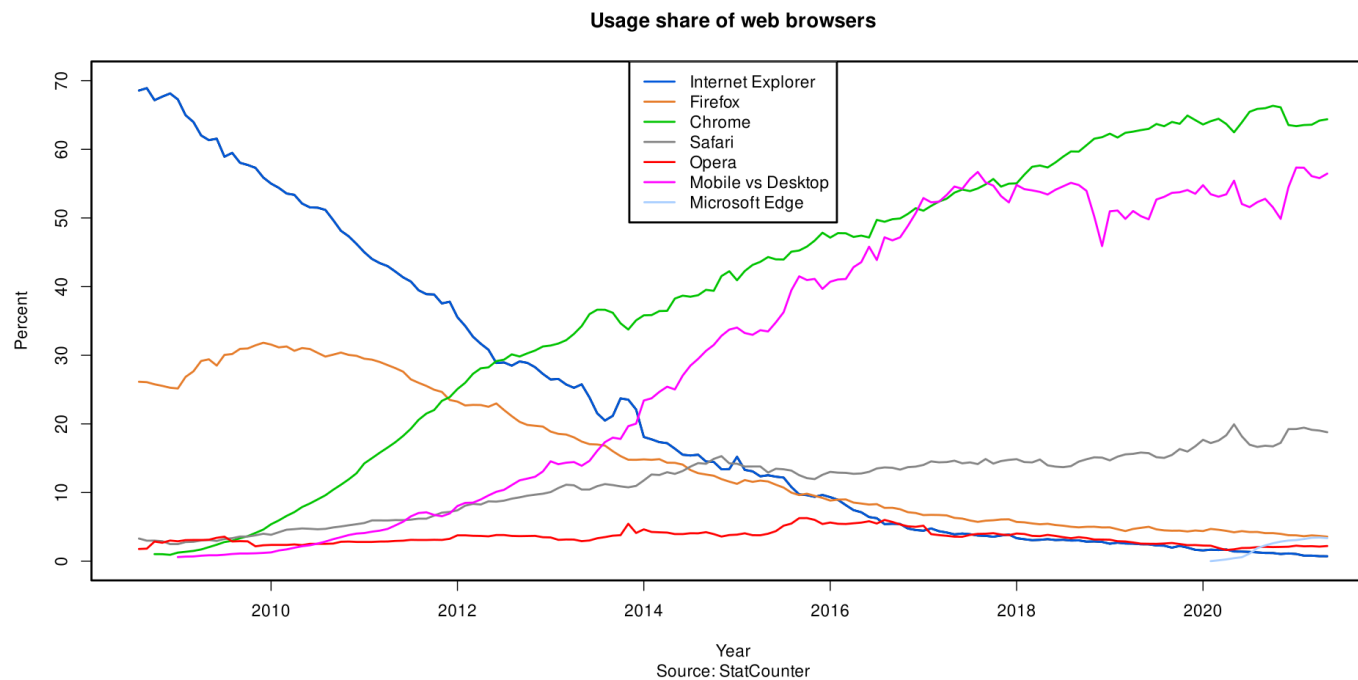
Internet	World Wide Web
On appelle Internet le réseau informatique mondial qui se base sur le protocole TCP/IP et sur lequel s'appuient différents services, dont le World Wide Web.	Système qui permet de naviguer de pages en pages en cliquant sur des liens dans un navigateur.

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.**

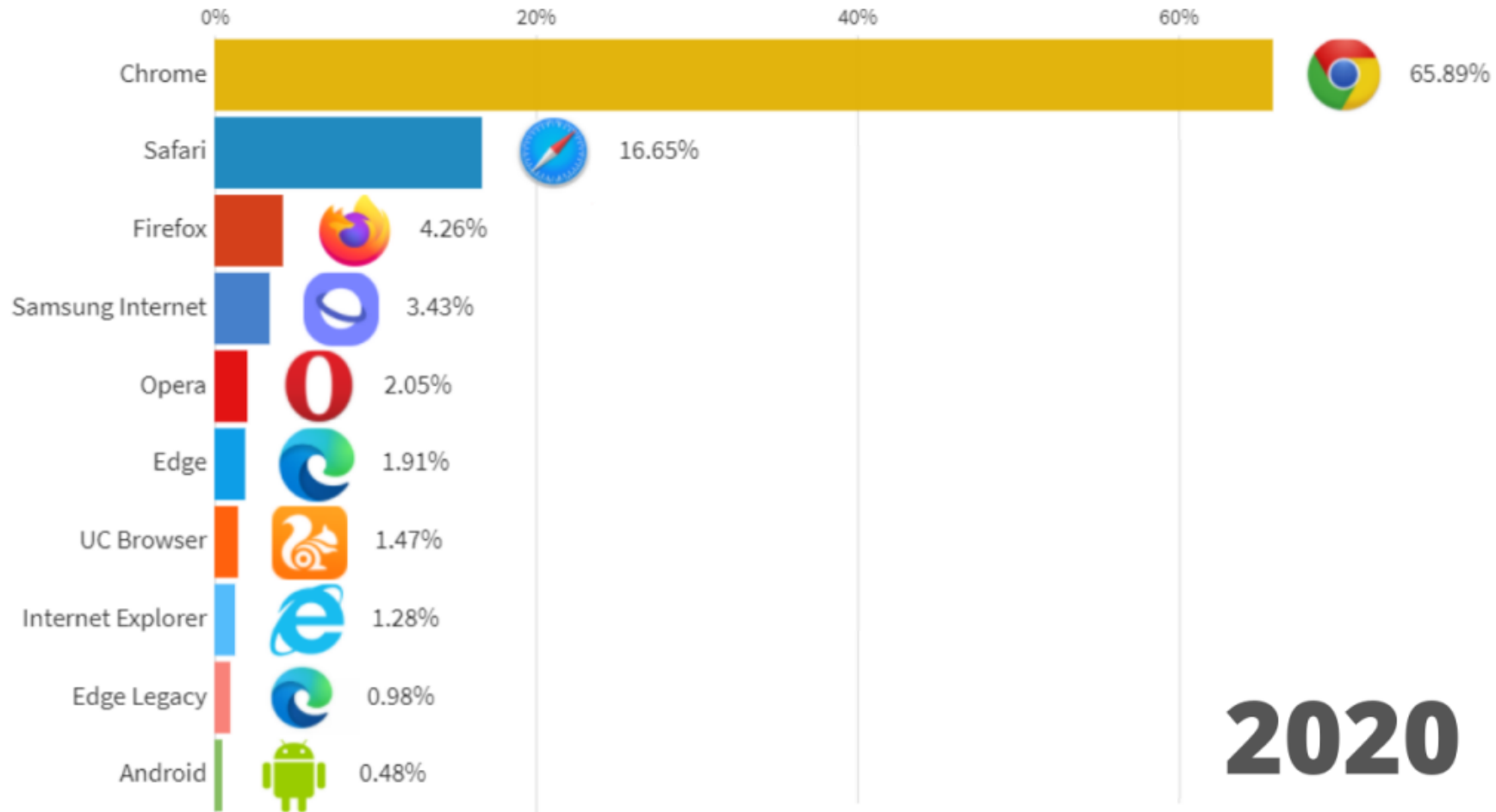
## 2) Comprendre comment une page web est affichée

### 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



## 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.

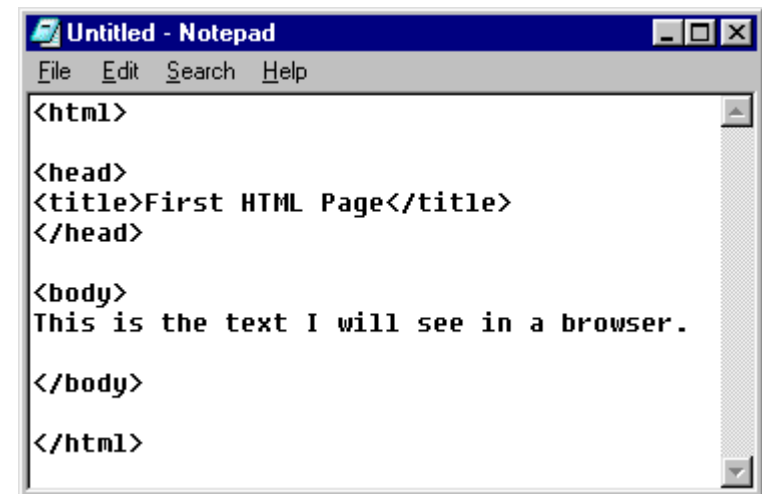


## - Quel éditeur de texte choisir ?

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.

A screenshot of a Windows Notepad application window. The title bar reads 'Untitled - Notepad'. The menu bar includes 'File', 'Edit', 'Search', and 'Help'. The text area contains the following HTML code:

```
<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 par défaut	Éditeur tiers
<b>Windows</b>	- <a href="#">Bloc-notes</a>	- <a href="#">Notepad++</a> - <a href="#">Visual Studio Code</a> - <a href="#">Web Storm</a> - <a href="#">Brackets</a> - <a href="#">ShiftEdit</a> - <a href="#">Sublime Text</a>
<b>Mac OS</b>	- <a href="#">TextEdit</a>	- <a href="#">TextWrangler</a> - <a href="#">Visual Studio Code</a> - <a href="#">Brackets</a> - <a href="#">ShiftEdit</a> - <a href="#">Sublime Text</a>
<b>Linux</b>	- <a href="#">Vi</a> (tout système UNIX) - <a href="#">GEdit</a> (Gnome) - <a href="#">Kate</a> (KDE) - <a href="#">LeafPad</a> (Xfce)	- <a href="#">Emacs</a> - <a href="#">Vim</a> - <a href="#">Brackets</a> - <a href="#">Sublime Text</a>
<b>Chrome OS</b>		- <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



## - Implémentation de la hiérarchie structurale

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>**.

# **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>
```

Code source: <https://codepen.io/xsccsx/pen/LYjNVVE>

**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 balise de titre dans l'ordre correct et logique : **<h1>** puis **<h2>**, puis **<h3>** et ainsi de suite.
- Bien qu'il y ait 6 niveaux de titre (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.