

Initiation HTML et CSS

Développeurs JAVA

Formateur : Jean-Paul BLAIRON

sep-oct 2020

- Réseau de base avec 2 ordinateurs
- Comment relier 10 ordinateurs ensemble ?
- Comment relier des milliards d'ordinateurs entre eux ?
- Des clients et des serveurs
- Clients et serveurs web
- Client et serveur sur la même machine ?
- Développement web : balises, HTML et éditeur
- Installation du serveur web de développement

Pour que deux ordinateurs puissent communiquer entre eux, ils doivent être liés soit par un **lien physique** (généralement par un câble Ethernet), **soit sans fil** (par exemple, via WiFi ou Bluetooth).

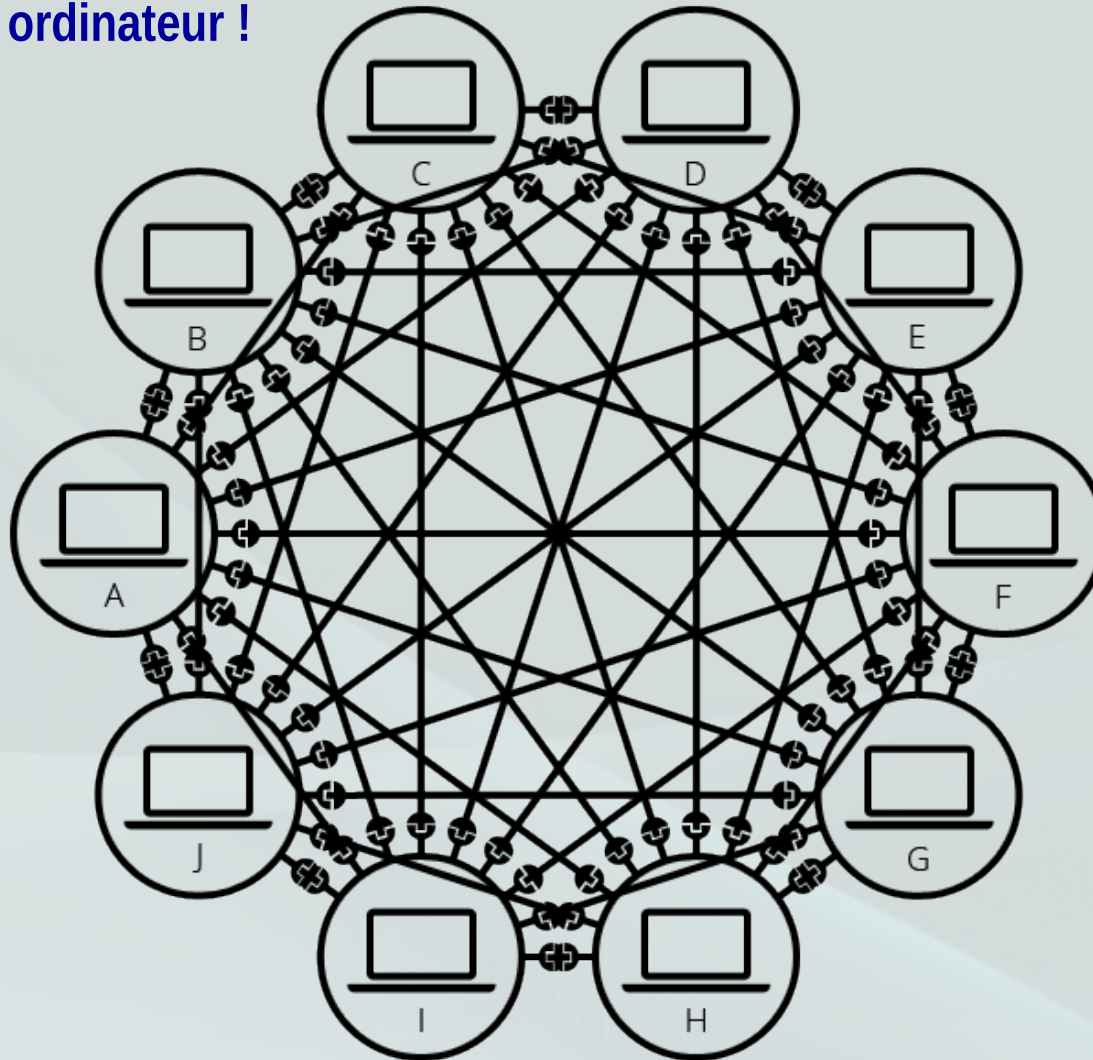
Tous ces types de connexions sont possibles sur les ordinateurs modernes.



Référence : https://developer.mozilla.org/fr/docs/Apprendre/Fonctionnement_Internet

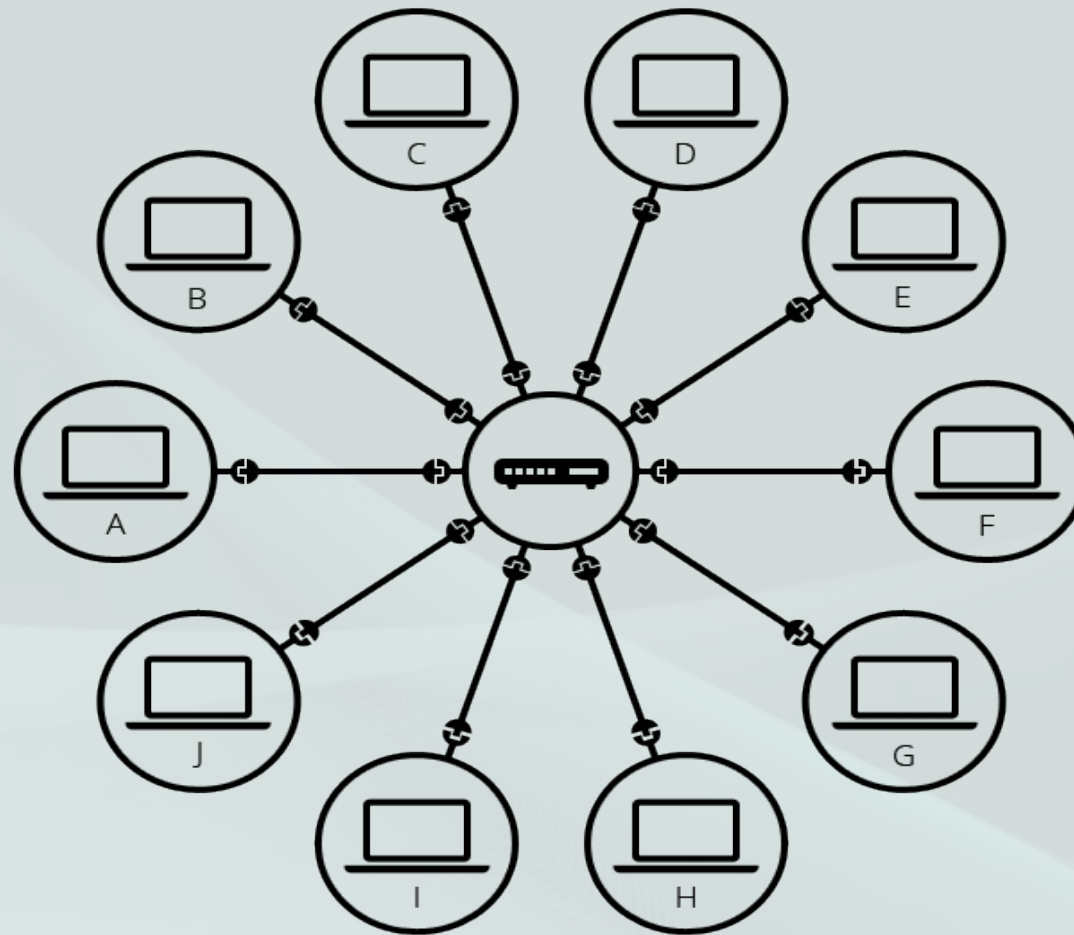
Comment relier 10 ordinateurs ensemble ?

En suivant la même logique que pour 2 ordinateurs, nous aurions besoin de **45 câbles** et de **9 prises** sur chaque ordinateur !

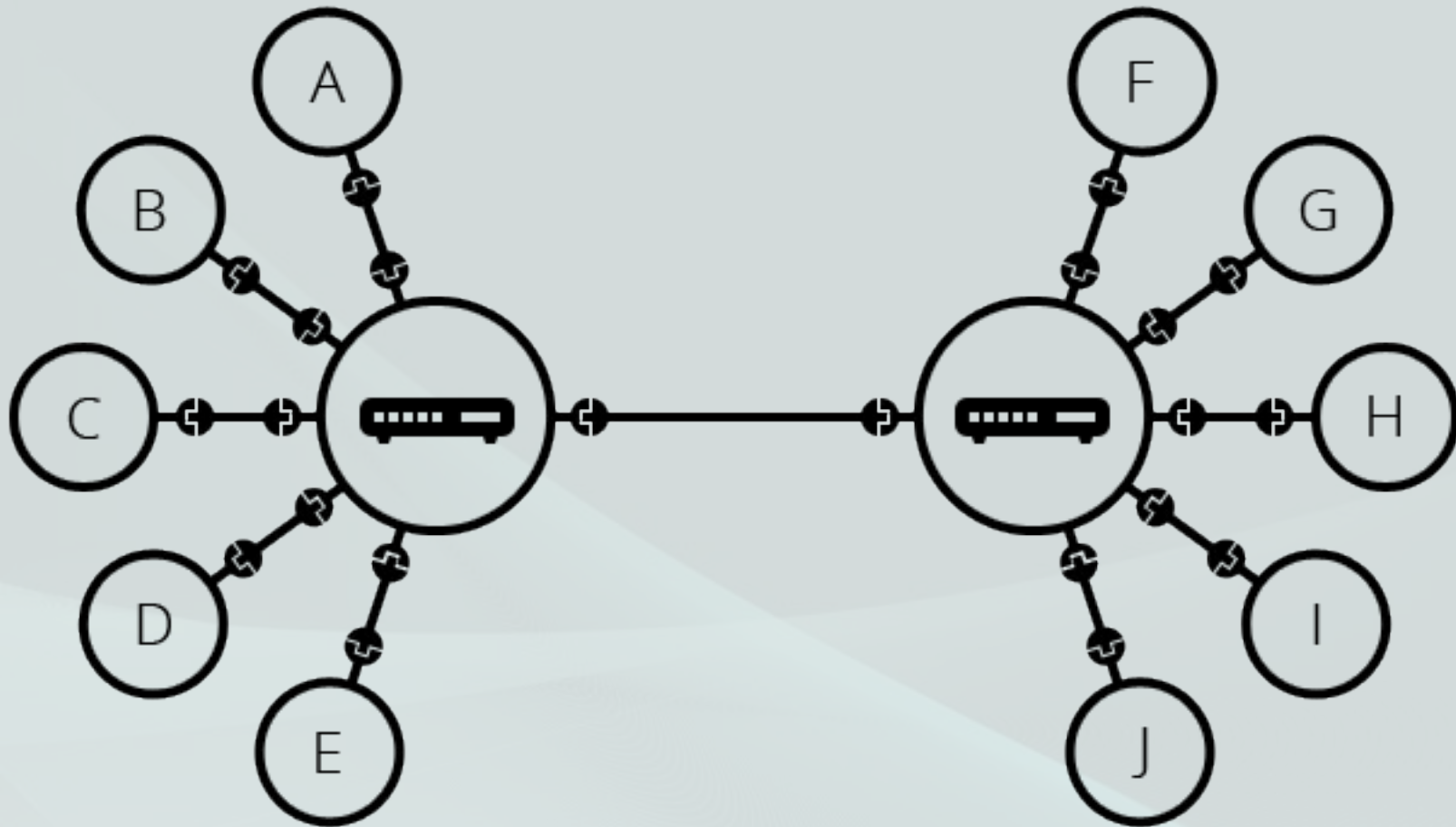


Comment relier 10 ordinateurs ensemble ?

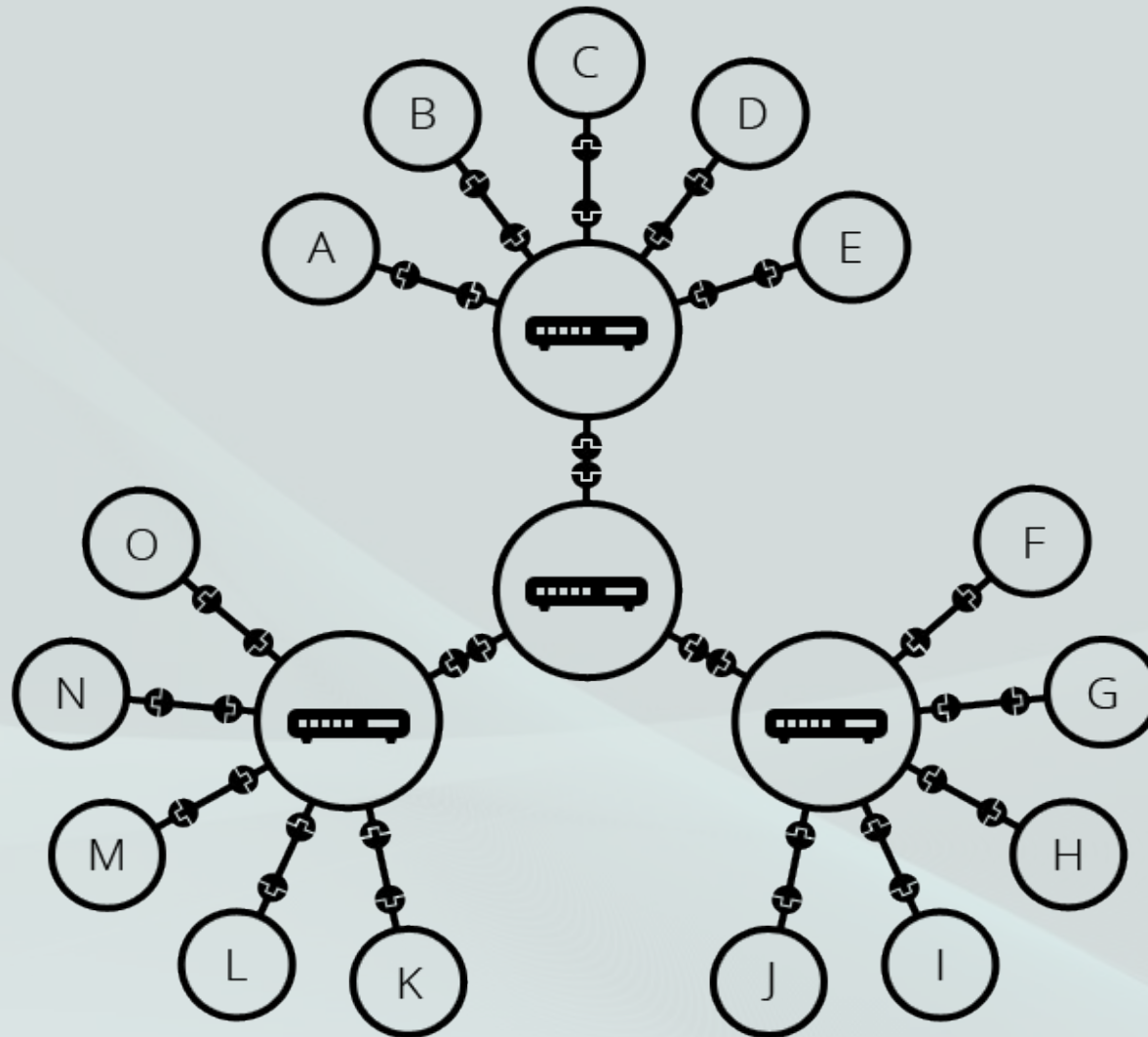
Afin de résoudre ce problème, chaque ordinateur du réseau est relié à un ordinateur spécial que l'on appelle **routeur**. Ce routeur n'a qu'une seule fonction : **assurer** que les messages transmis par un ordinateur donné se rendent au **bon ordinateur destinataire**.



Commençons par **relier 2 routeurs** entre eux.

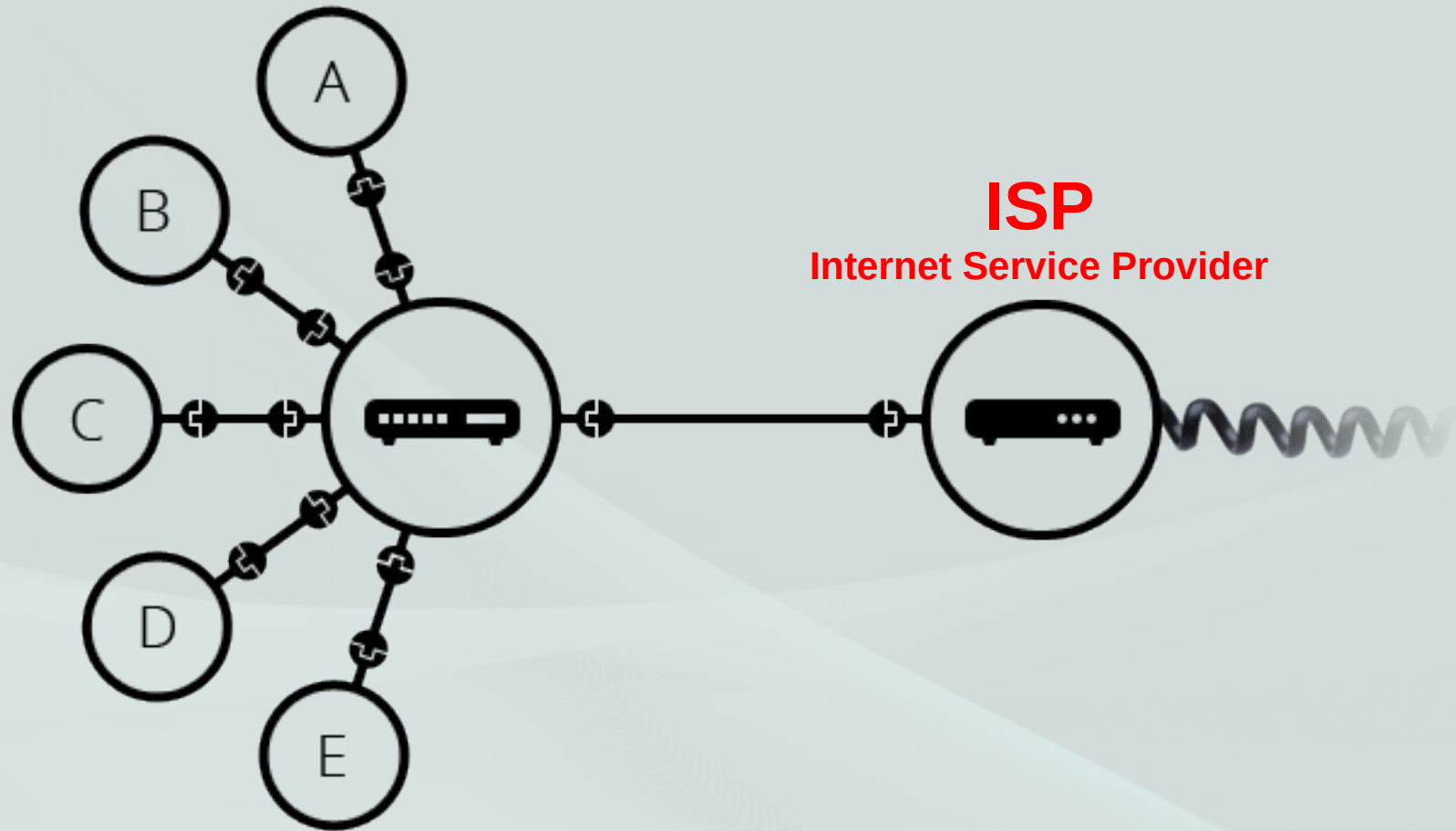


Nous avons ensuite la capacité d'**étendre** ce réseau **indéfiniment**.

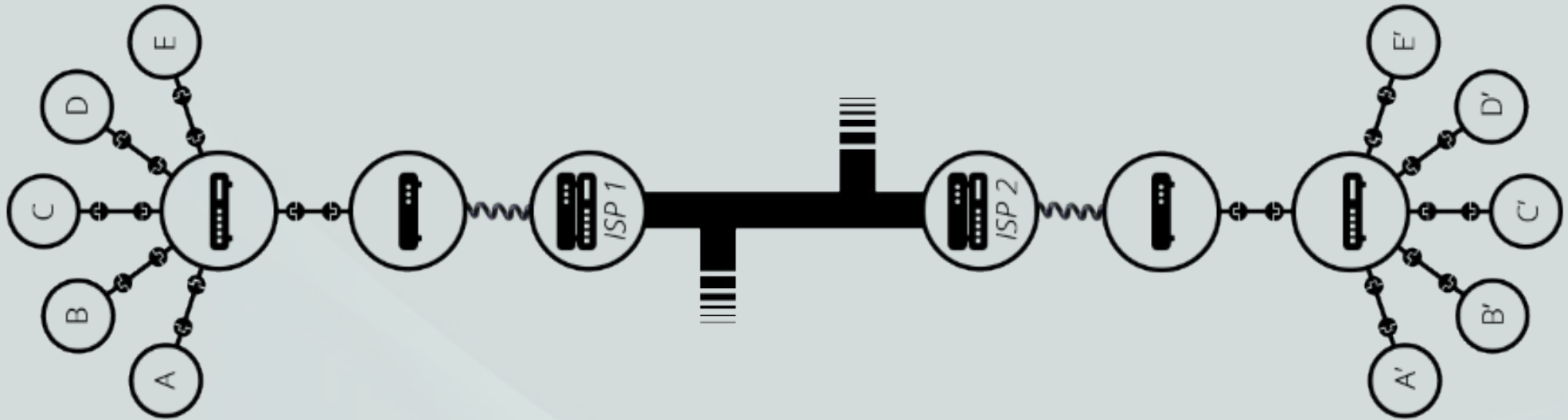


Comment relier des milliards d'ordinateurs entre eux ?

Cependant, il vous est **impossible** de brancher des **câbles** entre votre **maison** et le **reste de la planète**, alors vous allez lier votre réseau à un Fournisseur d'accès à Internet (**FAI**).



Et les **ISP** sont ensuite **reliés** aussi entre eux.



Lorsque nous souhaitons transmettre un message à un ordinateur, nous devons préciser de quel ordinateur il s'agit. Nous utilisons pour cela des **adresses IP** (Internet Protocol). Essayons la commande suivante dans une fenêtre « **cmd** » (**terminal texte**)

ping 13.225.238.96

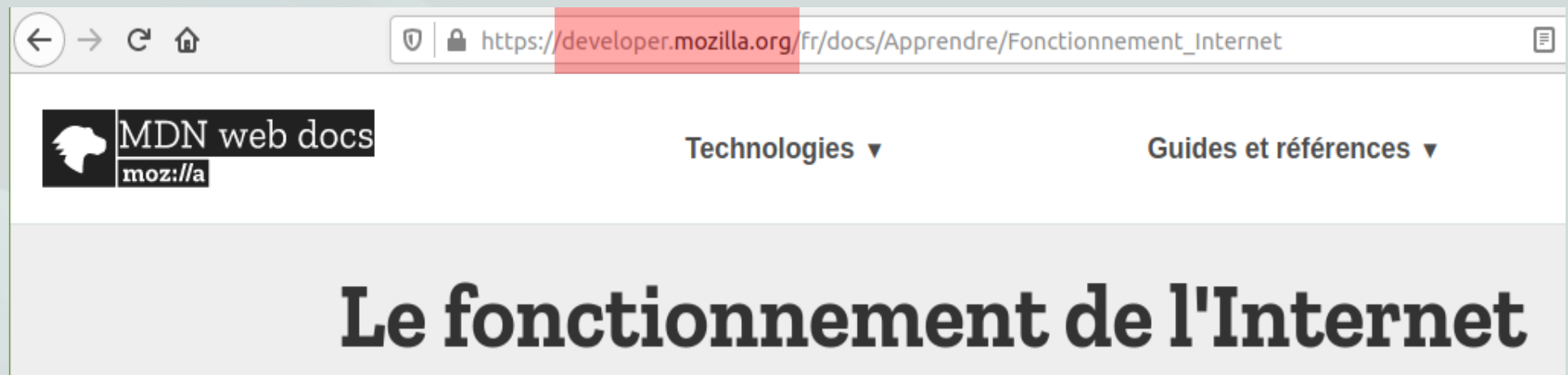
PING 13.225.238.96 (13.225.238.96) 56(84) bytes of data.

64 bytes from 13.225.238.96: icmp_seq=1 ttl=246 time=13.9 ms

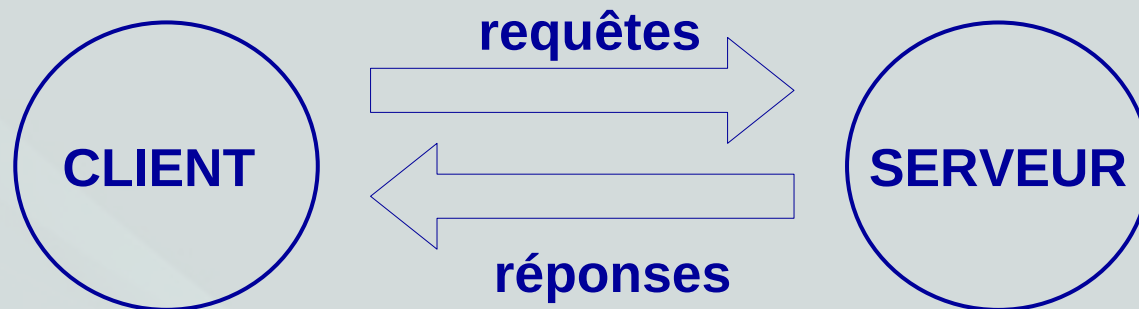
Nous venons de tester notre connectivité avec une machine sur internet. Essayons maintenant la commande suivante :

ping developer.mozilla.org

Il est plus facile de se rappeler du « **nom de domaine** » developer.mozilla.org que de l'adresse IP **13.225.238.96**, mais les **ordinateurs** utilisent entre eux des **adresses IP**.

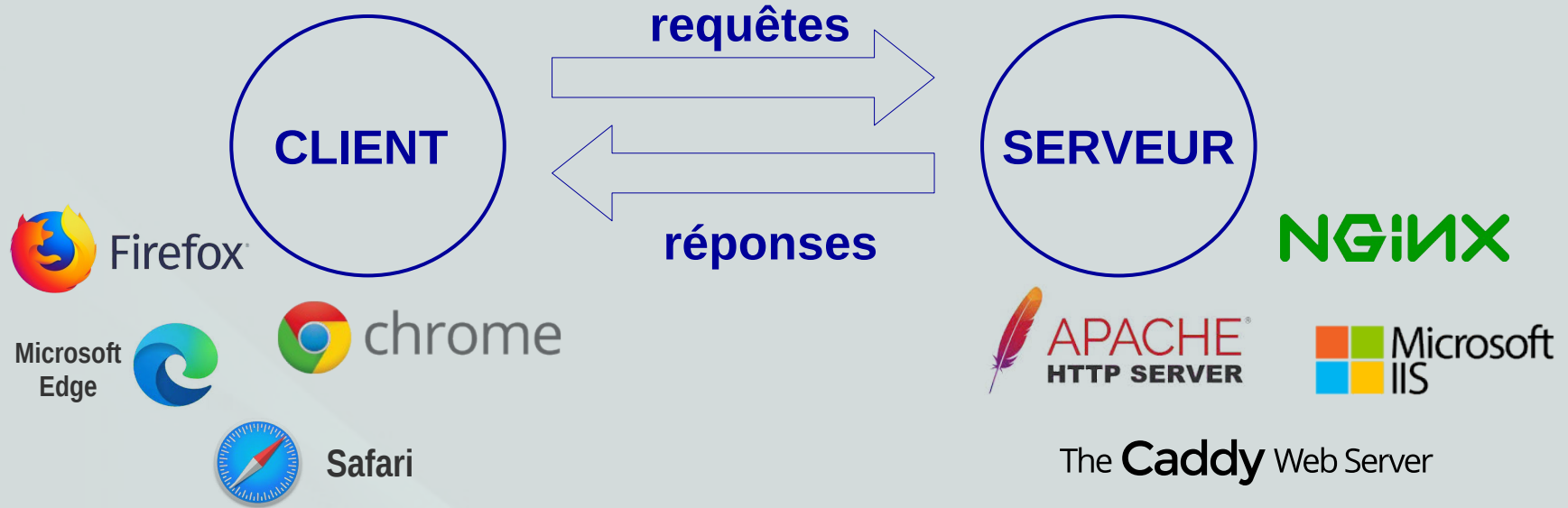


L'environnement **client-serveur** désigne un mode de communication à travers un réseau entre plusieurs **programmes** : l'un, qualifié de **client**, **envoie des requêtes** ; l'autre ou les autres, qualifiés de **serveurs**, **attendent les requêtes des clients et y répondent**.



Par exemple, lorsque vous avez tapé la commande « **ping developer.mozilla.org** », votre application « cmd » (**application cliente**) a envoyé une requête de résolution du nom de domaine « **developer.mozilla.org** » à un serveur DNS (**application serveur**) qui lui a communiqué l'adresse IP correspondant à ce nom de domaine.

Par **extension**, le **client** désigne également l'**ordinateur** ou la machine virtuelle sur lequel est exécuté le logiciel client, et le **serveur**, l'**ordinateur** ou la machine virtuelle sur lequel est exécuté le logiciel serveur.



Afin d'**éviter** de devoir configurer une **machine** « serveur » **spécifique** en phase de développement, nous installons généralement l'**application** « **cliente** » (ici le navigateur web) **et l'application** « **serveur** » (ici un serveur web) sur notre **machine de développement**.

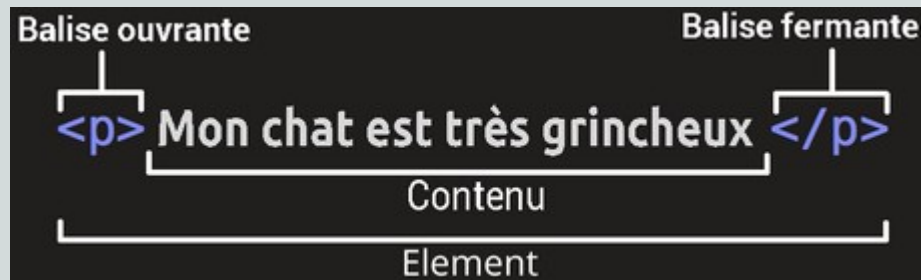
Du point de vue **pratique**, nous utiliserons une adresse IP particulière qui est « **127.0.0.1** » et dont le nom de domaine est « **localhost** ». Pourquoi ?

1. votre système sait que le nom « localhost » correspond à « 127.0.0.1 » **sans avoir besoin** d'interroger un serveur **DNS** (pas de configuration nécessaire)
2. les requêtes sur « **127.0.0.1** » et « **localhost** » **ne transitent jamais sur le réseau** car elle désignent la machine sur laquelle vous vous trouvez. C'est en fait une adresse « logique » qui est toujours disponible même en l'absence d'un câble réseau ou d'une connexion WiFi .

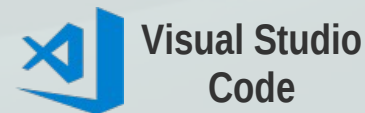
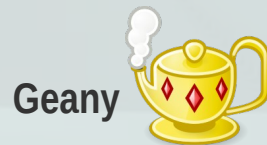


Historiquement, les **langages** à base de **balises** servent surtout à **structurer** ou **formater** des documents. L'inclusion de balises permet de **transférer** à la fois la **structure** du document et son **contenu**.

HTML est le langage de **balisage standard** pour les pages **Web**. Avec HTML, vous pouvez créer votre propre site Web :



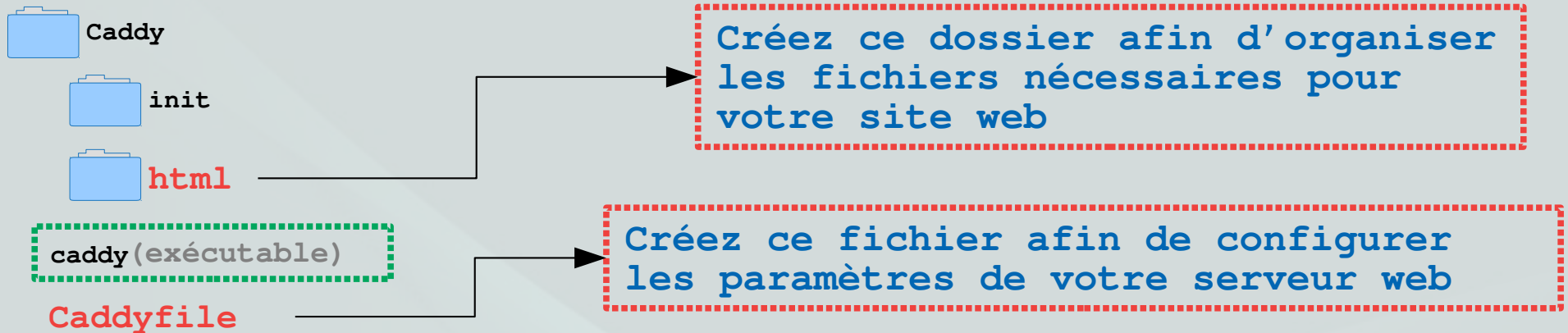
Vous pouvez choisir n'importe quel **éditeur de texte** pour créer des pages HTML, mais en voici quelques-uns qui devraient vous faciliter la tâche :



- Téléchargement du serveur web "Caddy"

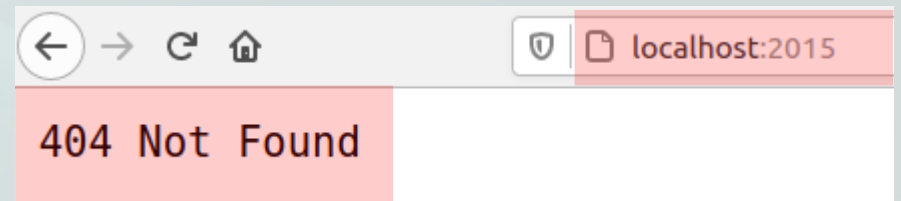
Version "**Personal (free)**" depuis le site <https://caddyserver.com/>

- Organisation du dossier "Caddy"



- Lancez l'exécutable « caddy » et vérifiez le résultat dans un navigateur

```
Activating privacy features... done.  
Serving HTTP on port 2015  
http://:2015
```



- Adresses IP et numéros de port

En plus de l'adresse IP, l'application cliente doit spécifier le « **port** » sur lequel l'**application serveur** « **écoute** » (ici localhost:**2015**). C'est une sorte d'**aiguillage** qui permet à la **machine serveur** de transférer la requête d'un client vers la bonne **application serveur**.

Sur **internet**, les serveurs web « **écoutent** » généralement par défaut sur les ports « **80** » (**HTTP** non chiffré) et « **443** » (**HTTPS** c.-à-d. du HTTP chiffré). **Caddy** écoute par défaut sur le port « **2015** », mais nous allons le configurer sur le port « **8080** » (pratique courante).

- Configuration du serveur Caddy

Arrêtez votre serveur **Caddy** et **éditez** le fichier « **Caddyfile** »

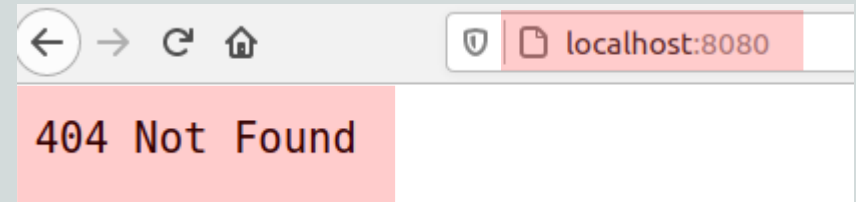
```
localhost:8080 {  
    root html  
}
```

← Numéro de port

← Nom du dossier dans lequel se trouveront les fichiers du site

Démarrez à nouveau votre serveur Caddy.

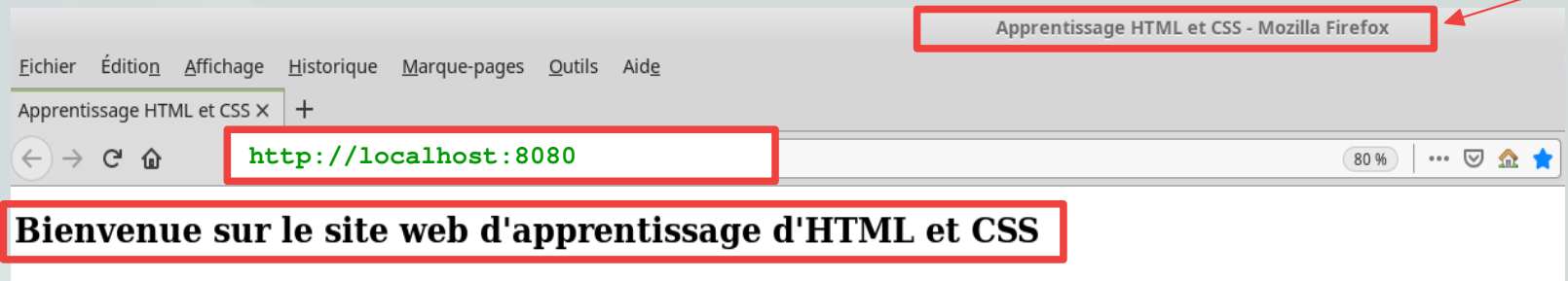
```
Activating privacy features... done.  
Serving HTTP on port 8080  
http://localhost:8080
```



Le serveur écoute maintenant bien sur le port « **8080** », mais la page d'accueil du site n'est pas trouvée (Erreur **404**). En réalité, le serveur Caddy essaie de trouver un **fichier** « **index.html** » (dans le **dossier** « **html** »), ce qui correspond à la **page d'accueil** du site.

- Création d'un fichier index.html basique
- Utilisation du site w3schools
- Gestion des Titres et des paragraphes
- Commentaires et listes
- Exercices sur les concepts de base


```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Apprentissage HTML et CSS
    </title>
    <meta charset="UTF-8">
  </head>
  <body>
    Bienvenue sur le site web d'apprentissage d'HTML et CSS
  </body>
</html>
```

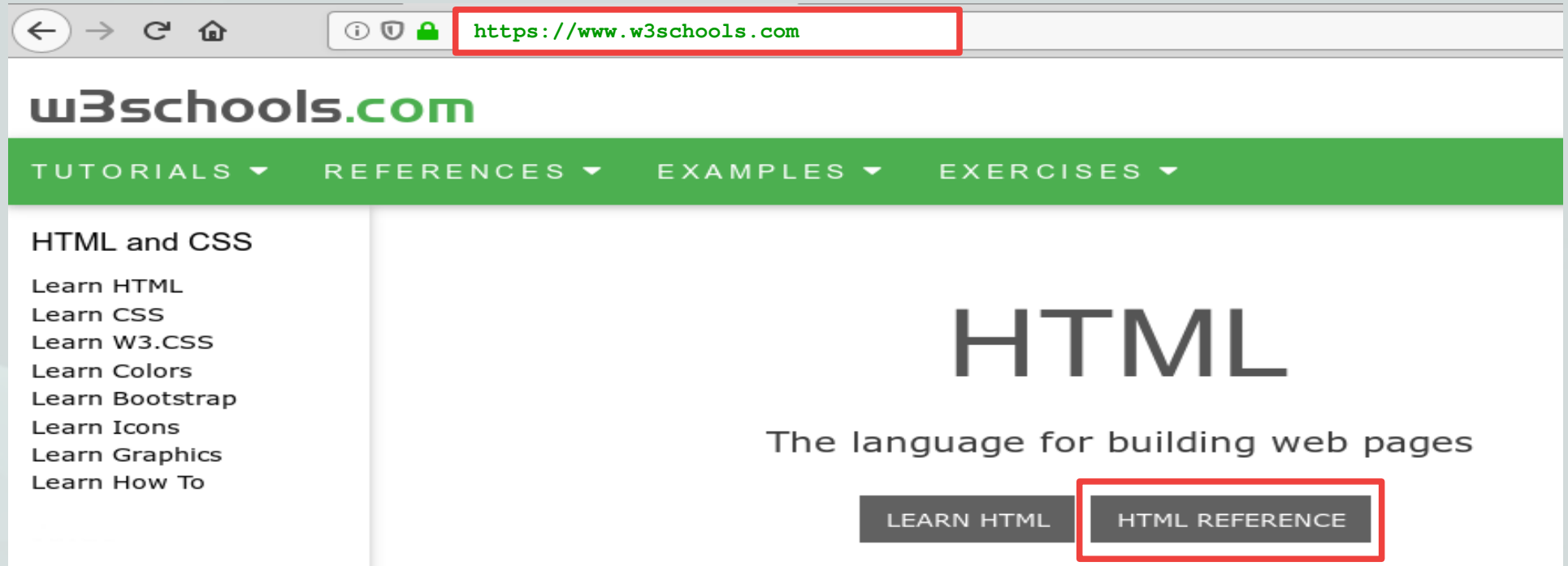


Remarques : La déclaration **<!DOCTYPE>** est une instruction spécifiant au navigateur la **version d'HTML** utilisée (ici HTML5). Ce n'est donc pas une balise HTML et doit être la première ligne du document. La balise **<html>** spécifie le **début** du document et **</html>** la **fin** du document HTML.

Remarque : Les **métadonnées** sont situées entre les balises `<head>` et `</head>`

Remarque : Les **données** (contenu) sont situées entre `<body>` et `</body>`

- Utilisation du site **w3schools**



Remarque : créez un **raccourci** vers la page "**HTML REFERENCE**" dans votre navigateur

- Structurer les titres et sous-titres

`<h1>Bienvenue sur le site web d'apprentissage d'HTML et CSS</h1>`

- Qu'est-ce qui a changé depuis que le texte est devenu un titre?

- Analysez les 6 "niveaux" de titres : www.w3schools.com/tags/tag_hn.asp

Remarque : ne confondez pas la balise `<title>` (titre de **barre** du navigateur) avec les balises `<h1>` à `<h6>` (titres de **contenu**).

- Structurer le contenu "texte" en paragraphes

`<p>Ceci est notre premier paragraphe. Son contenu "coulera" automatiquement de ligne en ligne jusqu'à ce qu'il soit terminé avec la balise de fermeture de paragraphe.</p>`

`<p>Notre deuxième paragraphe. Vous pouvez constater que la balise d'ouverture du paragraphe a automatiquement ajouté un "retour à la ligne, alors qu'un retour à la ligne dans votre éditeur HTML est complètement ignoré par votre navigateur</p>`

- Ajouter des commentaires dans le code HTML

```
<head>  
  <!--site web modifié le 08-09-2019-->  
  ...
```

Remarque : les commentaires sont acceptés n'importe où entre `<html></html>`

- Ajouter des listes dans le code HTML

```
<h2>Une liste ordonnée :</h2>  
<ol><!--Ordered List-->  
  <li>Bruxelles</li>  
  <li>Paris</li>  
  <li>Berlin</li>  
</ol>  
  
<h2>Une liste non ordonnée :</h2>  
<ul><!--Unordered List-->  
  <li>Lundi</li>  
  <li>Mardi</li>  
  <li>Mercredi</li>  
</ul>
```

Exercice 1 : gestion des métadonnées entre `<head>` et `</head>`

- quels sont les éléments pouvant être inclus entre `<head>` et `</head>`?
- Ajoutez une balise `<meta name="author" content="Tux le Manchot">`
- Ajoutez une balise `<meta name="description" content="Notre site d'apprentissage des langages HTML et CSS">`

Remarque : `<meta>` est une balise "orpheline" (pas de balise `</meta>`)

Remarque : pour chaque élément, il est important de vérifier la compatibilité avec les principaux navigateurs web (Browser Support).

Exercice 2 :

- modifiez la "liste ordonnée" pour que le premier `` commence à 50 plutôt que 1
- chaque `` suivant devra décrémenter de 1, plutôt qu'incrémenter de 1

Exercice 3 : à l'aide de la souris, redimensionnez la fenêtre du navigateur :

- comment se comporte le texte entre les balises `<h1>` et `</h1>` ?
- comment se comporte le texte entre les balises `<p>` et `</p>` ?

- Gestion et affichage des images
- Modifier la taille et la résolution des images
- Rogner une image selon une zone de sélection
- Utilisation du site "Can I Use"
- Ajouter des éléments "audio" et "vidéo"
- Gestion des liens vers d'autres pages
- Création de l'arborescence de base du site
- Gérer l'accès aux différentes pages

- Création d'un dossier séparé pour les images

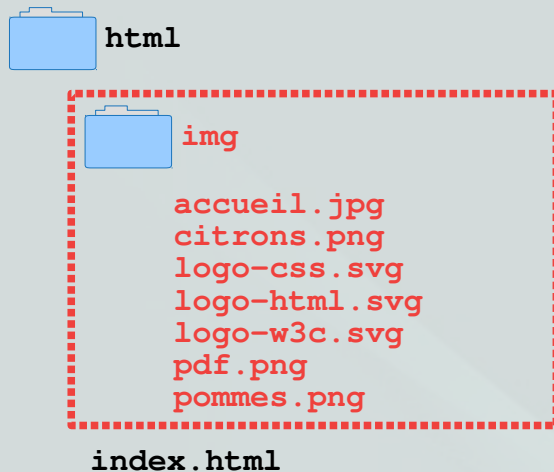


Image **bitmap** (raster)

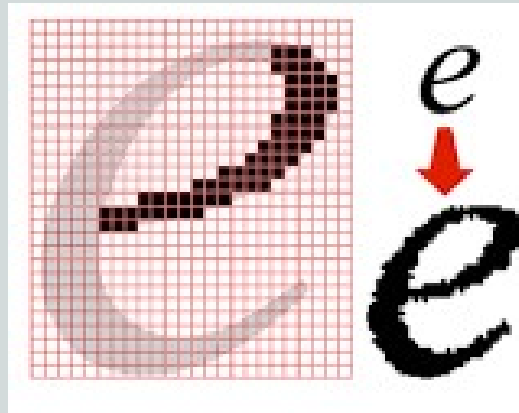
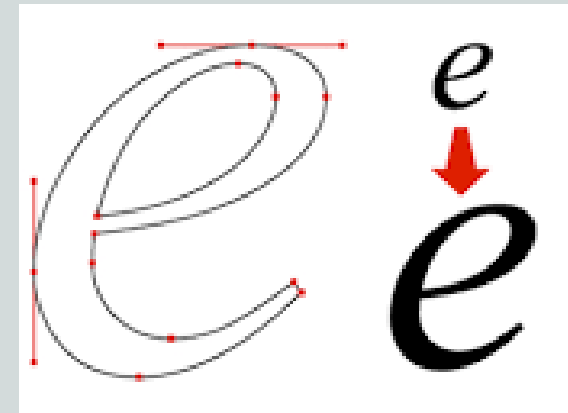


Image **vectorielle**



Remarque : les images de type "**bitmap**" sont généralement enregistrées dans les formats suivants :

. **jpg** ou **jpeg** pour les photographies et les images réalistes (compression avec pertes, taille des fichiers bien adaptées au web, pas de transparence).

. **png** pour les images de type "dessin au trait", les images représentant du texte, les images avec peu de couleurs (compression sans perte, taille des fichiers plus grande, transparence).

Remarque : les images "**vectorielles**" utilisent généralement le format **.svg**

- **Balise d'affichage des images**

`` (en dessous de `<h1>`)

Remarque : le chemin menant au dossier "**racine**" de votre site varie d'une machine à une autre. Le chemin sur votre machine de développement est presque toujours différent de celui d'un serveur de "production"

Machine de **développement**



D:\html

Serveur de **production**



/var/www/html



img

accueil.jpg



img

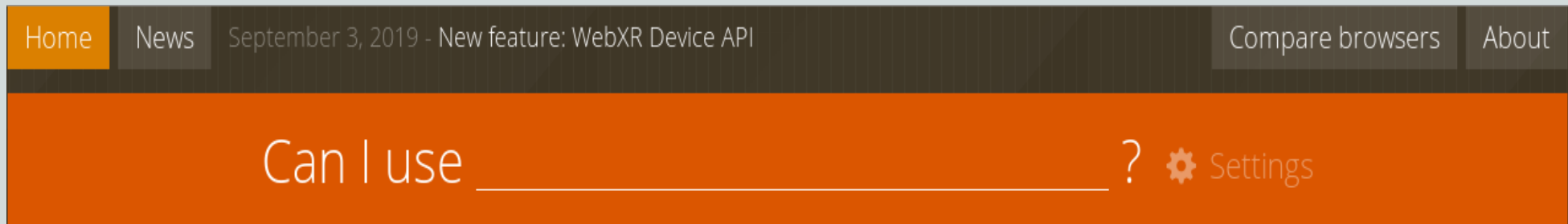
accueil.jpg

Remarque : tous les **chemins** vers les images doivent idéalement être "**relatifs**" à la racine de votre site (`img/accueil.jpg` plutôt que `D:\html\img\accueil.jpg`) afin d'éviter les différences entre l'arborescence de développement et celle de production.

Remarque : l'attribut `alt=` permet de présenter un texte explicatif si l'image ne peut pas être chargée (problème de permissions d'accès, chemin erroné ...). L'attribut "`alt`" est également utile si l'utilisateur utilise la navigation par synthèse vocale ou s'il a désactivé l'affichage des images dans la configuration de son navigateur.

- Vérifier la **compatibilité** de votre code avec les différents **navigateurs**

`https://caniuse.com`



Exercice : recherchons si les navigateurs sont compatibles avec les codecs suivants

- oga ("codec" audio numérique, sans brevet et plus performant que mp3)
- mp3 ("codec" audio numérique populaire)
- ogv ("codec" vidéo numérique, sans brevet et plus performant que mp4)
- mp4 ("codec" vidéo numérique populaire)

Remarque : Est-ce que tous les **navigateurs** reconnaissent les formats **oga** et **ogv**?

- Ajouter la lecture d'un fichier audio

<h2>Voici un exemple de fichier audio :</h2>

<audio controls>

<source src="audio/perception.oga" type="audio/ogg">

<source src="audio/perception.mp3" type="audio/mpeg">

</audio>

Remarque : les navigateurs compatibles chargeront en priorité le fichier .oga et les autres navigateurs (non compatibles) chargeront le fichier .mp3

<h2>Voici un exemple de fichier vidéo :</h2>

<video muted controls>

<source src="video/welcome.ogv" type="video/ogg">

<source src="video/welcome.mp4" type="video/mp4">

</video>

Remarque : les navigateurs compatibles chargeront en priorité le fichier .ogv et les autres navigateurs (non compatibles) chargeront le fichier .mp4 .

Remarque : les fichiers .oga et .ogv peuvent aussi être nommés .ogg

Hyperliens vers d'autres pages (entre `<img/accueil.jpg>` et le premier `<p>`)

```
<a href="https://www.w3schools.com/tags/default.asp">HTML</a>
```

Remarque : la balise `<a>` définit un "**hyperlien**", c.-à-d. un lien pointant vers une **autre page** de votre site **ou** d'un autre site (Internet).

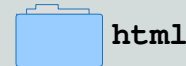
Remarque : si vous souhaitez afficher une **image** à la place du texte "**HTML**", il suffit de remplacer le texte par une balise ``.

```
<a href="https://www.w3schools.com/tags/default.asp">  
  </a>
```

```
<a href="https://www.w3schools.com/cssref/default.asp">  
  </a>
```

Remarque : nous constatons que les images sont **trop grandes** et le résultat n'est pas agréable esthétiquement. Nous allons devoir aborder bientôt la gestion du **style** avec **CSS** ;-)

- Création de nouvelles pages HTML



citrons.html
contact.html
index.html
pommes.html
sandbox.html

```
<h1>Bienvenue sur la page des citrons</h1>

```

```
<h1>Bienvenue sur la page de contact</h1>
```

```
<h1>Bienvenue sur la page des pommes</h1>

```

```
<h1>Bienvenue dans le bac à sable</h1>
```

- Accès aux différentes pages à l'aide d'**URL**



- Création d'un **menu** d'accès aux différentes pages

```
<a href="index.html">Accueil</a>  
<a href="citrons.html">Citrons</a>  
<a href="pommes.html">Pommes</a>  
<a href="contact.html">Contact</a>  
<a href="sandbox.html">Sandbox</a>
```

- La **plupart** des **balises** HTML dispose d'une **valeur d'affichage** par défaut.

Les 2 valeurs principales sont : **block** ou **inline**

- Comportement des balises en mode "**block**"

Elles commencent toujours sur une **nouvelle ligne** et prennent la **largeur maximale** disponible (s'étendent à gauche et à droite autant que possible).

- Quelques balises dont le comportement est **généralement** de type "**block**"

<u><div></u>	<u><aside></u>	<u><fieldset></u>	<u><footer></u>	<u><form></u>
<u><h1>-<h6></u>	<u><header></u>	<u><hr></u>	<u></u>	<u><main></u>
<u><nav></u>	<u></u>	<u><p></u>	<u><section></u>	<u></u>

- Comportement des balises en mode "**inline**"

Elles ne commencent **pas** sur une **nouvelle ligne** et ne prennent que la **largeur minimale** nécessaire.

- Quelques balises dont le comportement est généralement de type "inline"

<u><code><a></code></u>	<u><code>
</code></u>	<u><code><button></code></u>	<u><code></code></u>	<u><code><input></code></u>
<u><code></code></u>	<u><code><textarea></code></u>			

Remarque : le comportement par défaut d'une balise peut **varier** en fonction du **navigateur** web utilisé.

Remarque : lors de l'étude du langage **CSS**, nous verrons que la propriété "**display**" permettra de **modifier** le **comportement** par défaut d'une balise.

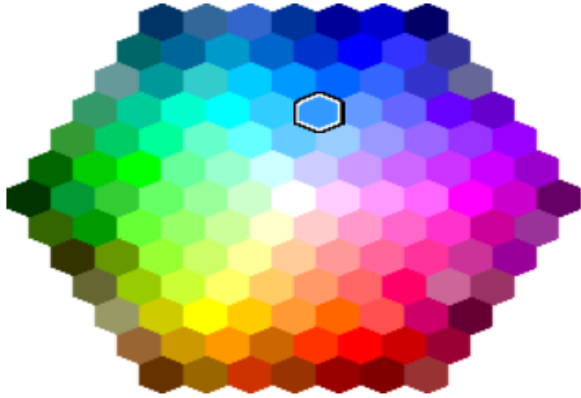
Analyse de votre code actuel

- les balises `<p>`, `<h1>`, `<h2>`, ``, `` et `` sont-elles en mode "**block**" ?
- les balises `<a>` et `` sont-elles en mode "**inline**" ?
- ajoutez "Texte collé à gauche" entre `</h2>` et `<audio>`. La balise `<audio>` est-elle en mode "**block**" ou "**inline**" ?

- Trouver les valeurs digitales des couleurs
- Une page web = HTML + CSS
- Premiers pas dans la gestion du style avec le CSS
- Gestion de polices spécifiques avec le CSS
- Utilisation de classes CSS génériques
- Mise en évidence de mots avec CSS
- Gestion du contenu des pages citrons.html et pommes.html
- Modèle de boîte CSS (marge - bordure - padding - contenu)
- Gestion du style du menu principal

- Apprenez à utiliser le "Color Picker" depuis le site www.w3schools.com

Pick a Color:



Or Enter a Color:

Selected Color:

Black Text

Shadow

White Text



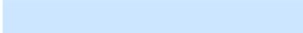











Shadow

#3399ff

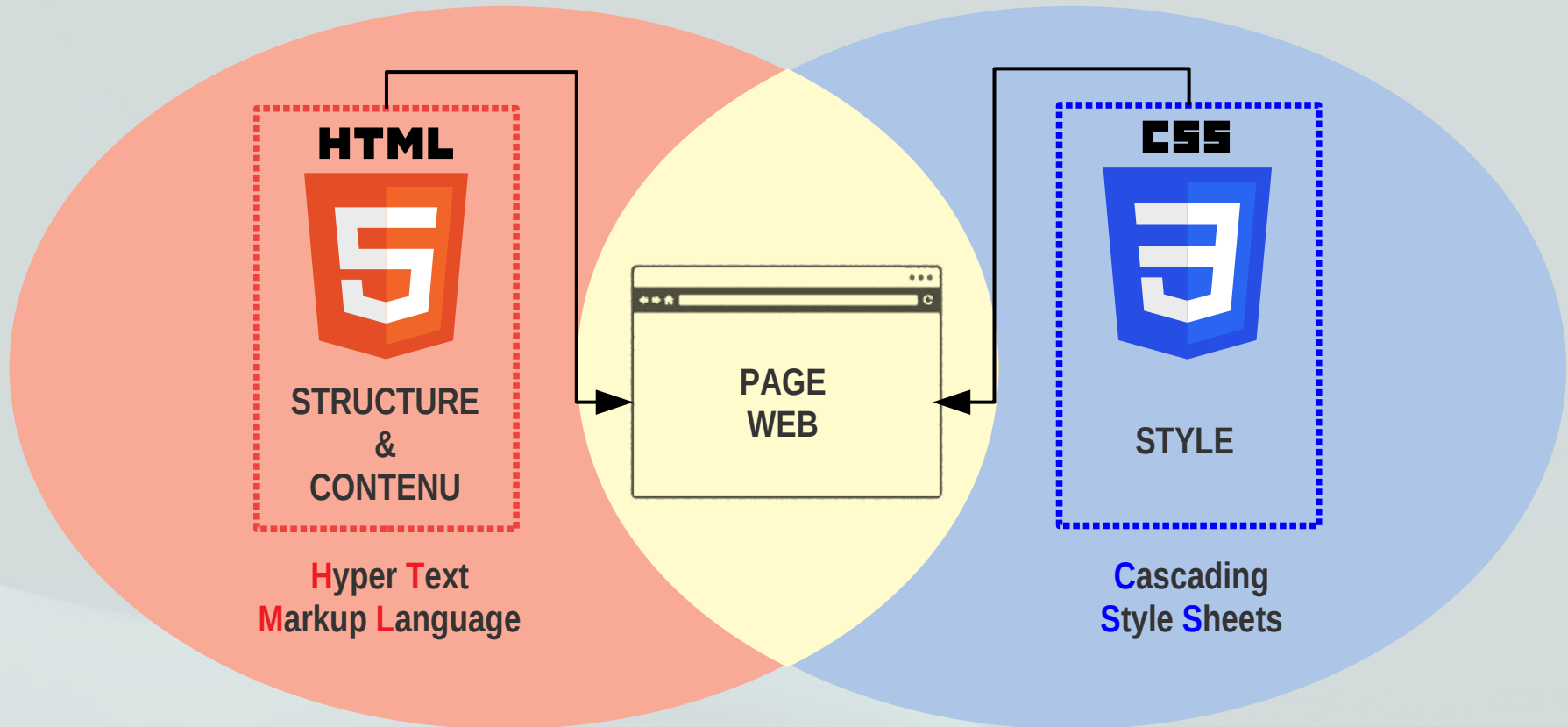
rgb(51, 153, 255)

hsl(210, 100%, 60%)

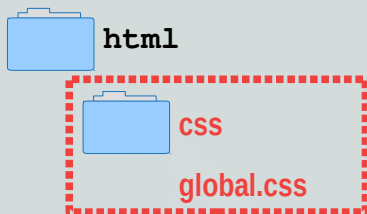
Lighter / Darker:

100%		#ffffff
95%		#e6f2ff
90%		#cce6ff
85%		#b3d9ff
80%		#99ccff
75%		#80bfff
70%		#66b3ff
65%		#4da6ff
60%		#3399ff
55%		#1a8cff
50%		#0080ff
45%		#0073e6
40%		#0066cc
35%		#0059b3

Une **page** web = **HTML** + **CSS**



- Création d'un nouveau dossier et d'un fichier css



Remarque : les "CSS" ou "feuilles de style en cascade" (Cascading Style Sheets) contiennent du code informatique qui décrit le style de présentation des documents HTML.

- Ajouter un pointeur dans le code HTML vers le fichier .css

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

- Ajouter du style avec des propriétés CSS de base

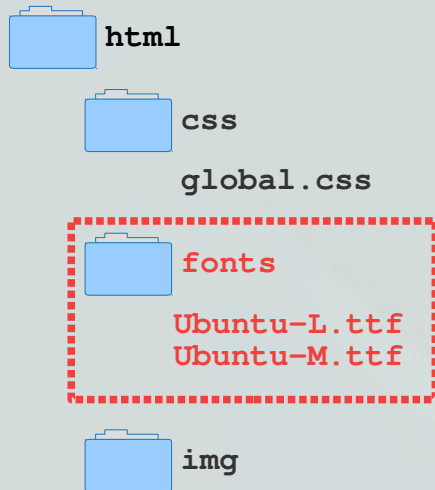
```
html {  
  --gris: #333333; /*https://www.w3schools.com/css/css3_variables.asp*/  
  --gris-clair: #f2f2f2;  
  --bleu: #004d99;  
  --orange: #ff6600;  
  --orange-clair: #ffe0cc;  
  --rouge: #ff0000;  
  --vert: #009973;  
  --vert-clair: #009973;  
}
```

Remarque : les commentaires sont situés entre /* et */

```
body {  
  background-color: white; /*https://www.w3schools.com/colors/colors_names.asp*/  
  font-size: 100%;  
  font-weight: normal;  
}
```

Remarque : ici les "sélecteurs CSS" correspondent à des balises HTML

- Ajouter des fichiers de polices de caractères spécifiques



avec
empattement

Serif

avec empattement

The word 'Serif' is shown in a serif font. The letter 'S' is circled in red, and a line connects it to the word 'Serif'. Another red circle is around the bottom of the 'S', with a line connecting it to the text 'avec empattement'.

sans
empattement

Sans-Serif

sans empattement

The words 'Sans-Serif' and 'S' are shown in a sans-serif font. The 'S' is circled in red, and a line connects it to the text 'sans empattement'.

Remarque : les polices "**sans-serif**" (**sans empattements**) sont généralement plus lisibles à l'écran (exemples : Ubuntu, Helvetica, Arial).

- Spécifier le nom et l'**URL** des **polices** de caractères **spécifiques**

```
@font-face {  
  font-family: "Ubuntu-L";  
  src: url(/fonts/Ubuntu-L.ttf) format('truetype');  
}
```

```
@font-face {  
  font-family: "Ubuntu-M";  
  src: url(/fonts/Ubuntu-M.ttf) format('truetype');  
}
```

- Gérer la **priorité** entre les **polices** de caractères

```
body {  
  ...  
  font-family: "Ubuntu-L", Helvetica, Arial, sans-serif;  
}
```

police prioritaire → priorité la plus basse

Remarque : Helvetica (Linotype), Arial (Microsoft), sans-serif (police générique)

- Ajouter le **style** pour les titres **<h1>**, **<h2>** et **<h3>**

```
h1 {  
    font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;  
    font-size: 150%;  
    font-weight: bold;  
}  
  
h2 {  
    font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;  
    font-size: 120%;  
    font-weight: bold;  
}  
  
h3 {  
    font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;  
    font-size: 100%;  
    font-weight: bold;  
}
```

- Ajouter des **classes génériques** pour la mise en évidence de texte (**global.css**)

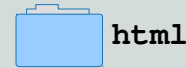
```
.gris {  
    color: var(--gris);  
}  
  
.bleu {  
    color: var(--bleu);  
}  
  
.orange {  
    color: var(--orange);  
}  
  
.rouge {  
    color: var(--rouge);  
}  
  
.vert {  
    color: var(--vert);  
}
```

```
.gras {  
  font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;  
  font-weight: bold;  
}  
  
.leger {  
  font-family: "Ubuntu-L", Helvetica, Arial, sans-serif;  
  font-weight: normal;  
}
```

Remarque : nous venons de définir le style de "**classes**" qui ne s'appliqueront que pour les **balises** ayant l'**attribut** `class="NOM_DE_LA_CLASSE"` dans le code **HTML**.

Les **autres** balises ne bénéficieront **pas** du style de cette classe. Ce type de sélecteur est utilisé pour donner du style à un **ensemble** de **balises** faisant partie de la classe spécifiée.

- Les balises `` permettent de mettre en évidence des **mots** ou des parties de phrases.



citrons.html
contact.html
index.html
pommes.html
sandbox.html

```
<h1 class="bleu">Bienvenue sur la page des <span  
class="orange">citrons</span></h1>
```

```
<h1 class="bleu">Bienvenue dans le formulaire de  
<span class="orange">contact</span></h1>
```

```
<h1 class="bleu">Bienvenue sur le site web  
d'apprentissage d'<span class="orange">HTML et  
CSS</span></h1>
```

```
<h1 class="bleu">Bienvenue sur la page des <span  
class="orange">pommes</span></h1>
```

```
<h1 class="bleu">Bienvenue dans le <span  
class="orange">bac à sable</span></h1>
```

- Mise en **évidence** des listes de la page **index.html**

```
<h2 class="orange gras">Une liste ordonnée :</h2>
<ol start="50" reversed class="orange gras"><!--Ordered List-->
  <li><span class="gris leger">Bruxelles</span></li>
  <li><span class="gris leger">Paris</span></li>
  <li><span class="gris leger">Berlin</span></li>
</ol>
```

```
<h2 class="vert gras">Une liste non ordonnée :</h2>
<ul class="vert gras"><!--Unordered List-->
  <li><span class="gris leger">Lundi</span></li>
  <li><span class="gris leger">Mardi</span></li>
  <li><span class="gris leger">Mercredi</span></li>
</ul>
```

Remarque : nous venons d'appliquer **plusieurs styles** pour chacune des balises.

- Spécification d'un **même style** pour **2 balises** différentes

```
audio, video {
  width: 70%;
}
```


- **Exercice :**

Utilisez le fichier "citrons.docx" (extrait du site Wikipedia) afin de remplir le contenu de la page "citrons.html".

Utilisez le fichier "pommes.docx" (extrait du site Wikipedia) afin de remplir le contenu de la page "pommes.html".

Veillez à respecter les points suivants :

- découper les paragraphes de manière similaire
- pour les titres de niveau 2 (par exemple : "Histoire"), utilisez une hauteur de police plus grande et "grasse" et respectez la couleur du fichier « docx »
- pour les titres de niveau 3 (par exemple : "- Eureka"), utilisez une police "grasse" et respectez la couleur du fichier « docx »
- respectez une zone "blanche" en dessous des titres

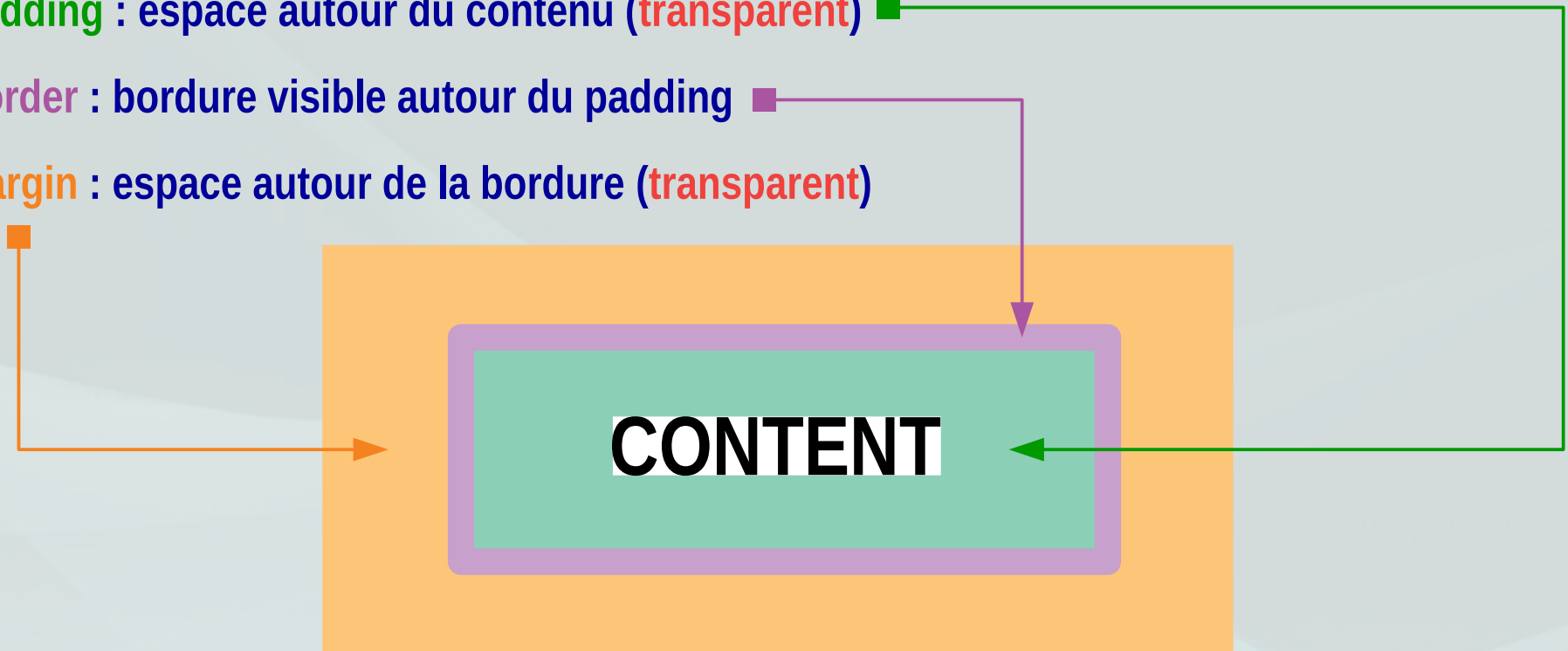
- Lors de l'utilisation du langage **CSS** pour gérer le **style** d'un document HTML, tous les **éléments HTML** peuvent être considérés comme des "**boîtes**".

Quel que soit le **contenu**, cette boîte peut être **entourée** (complétée) par **une ou plusieurs** des 3 **propriétés CSS** suivantes :

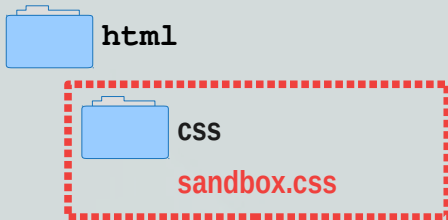
Padding : espace autour du contenu (**transparent**) ■

Border : bordure visible autour du padding ■

Margin : espace autour de la bordure (**transparent**) ■



- Ajouter un fichier .css pour gérer le style du bac à sable



- Ajouter un pointeur dans le code HTML vers le fichier .css

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

- Exemple de modèle de boîte dans le code HTML du bac à sable

```
<h2 class="bleu">Le modèle classique de boites CSS</h2>  
  

```

- Ajoutons un fond pour visualiser l'espace pris par nos images

```
.boiteDemo {  
  background-color: yellow;  
}
```

- Ajoutons maintenant un "**padding**" de 20 pixels **autour** de l'**image**

`padding: 20px;`

- Ajoutons maintenant une "**bordure**" bleue de 10 pixels **autour** du **padding**

`border: 10px solid blue;`

- Ajoutons enfin une "**marge**" de 30 pixels **autour** de la **bordure**

`margin: 30px;`

- Calcul de la **place** prise par **une image** sur la page

- La **taille** d'une **l'image** (contenu) fait **200 x 230** pixels (click droit + examiner l'élément)

- la **place prise** par une image = **contenu + padding + bordure + marge**

- **largeur** prise par une image = $200 + (2 \times 20) + (2 \times 10) + (2 \times 30) = 320$ pixels

- **hauteur** prise par une image = $230 + (2 \times 20) + (2 \times 10) + (2 \times 30) = 350$ pixels

Référence : https://www.w3schools.com/css/css_boxmodel.asp

- Utiliser des valeurs différentes pour chaque côté

```
padding-top: 10px; (padding du haut)
padding-right: 20px; (padding de droite)
padding-bottom: 30px; (padding du bas)
padding-left: 40px; (padding de gauche)
padding : 10px 20px 30px 40px ; (top - right - bottom - left)
```

```
border-top: 10px dotted blue;
border-right: 5px solid red;
border-bottom: 10px dotted red;
border-left: 5px solid blue;
```

```
margin-top: 10px;
margin-right: 30px;
margin-bottom: 10px;
margin-left: 0px;
margin: 10px 30px 10px 0px;
```

Analyse : effectuez un "click droit" sur une photo puis "Examiner l'élément" (Firefox).

- Ajouter le **style** du **menu principal** de navigation (global.css)

```
.menuPrincipal {  
    background-color: var(--vert);  
    padding: 10px;  
}
```

- Ajouter le **style** des **hyperliens** constituant le menu

```
.lienMenu {  
    font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;  
    font-weight: bold;  
    color: white;  
    text-decoration: none;  
    margin-right: 5px;  
}
```

Remarque : ici nous voulons donner un style particulier aux liens `<a>` situés dans le **menu principal**, sans que les autres balises `<a>` ne soient impactées.

Remarque : les balises **membres** d'une **classe** ne doivent **pas forcément** être de **même type**.

- **Modification** du code **HTML** afin d'utiliser les **classes** **.menuPrincipal** et

```
<div class="menuPrincipal">
  <a href="index.html" class="lienMenu">Accueil</a>
  <a href="citrons.html" class="lienMenu">Citrons</a>
  <a href="pommes.html" class="lienMenu">Pommes</a>
  <a href="sandbox.html" class="lienMenu">Sandbox</a>
  <a href="contact.html" class="lienMenu">Contact</a>
</div><!--Fin .menuPrincipal-->
```

Remarques : les balises `<a>` ont la fâcheuse tendance à ajouter un **espace** derrière elles, ce qui est gênant lorsqu'on les utilisent pour créer un menu horizontal (testez en ajoutant `background-color: blue;` pour la classe `.lienMenu`).

De manière générale, les balises `<div></div>` définissent une **division** ou une **section** dans le document **HTML**. Nous assurons ici une couleur de fond homogène en incluant les balises `<a>` du menu dans la division `<div class="menuPrincipal">`.

Lorsqu'on utilise plusieurs `<div>` dans le code, il est pratique d'ajouter un **commentaire** comme `<!-- Fin de .menuPrincipal-->` derrière les `</div>` pour permettre d'identifier facilement la division que l'on ferme.

- **Utilité des formulaires HTML**
- **Création d'un formulaire de contact simple en HTML**
- **Ajouter un fichier .css pour gérer le style du formulaire**
- **Analyse de l'attribut CSS "display"**
- **Ajouter un pointeur dans le code HTML vers le fichier.css**
- **Ajouter le style du formulaire de contact**
- **Ajouter la page de traitement du formulaire**

- Utilité des formulaires HTML

Les formulaires HTML permettent l'**interaction** entre un **utilisateur** et un **site web** ou une application. Ils permettent à l'utilisateur d'**envoyer des données** au site web.

Un formulaire HTML est composé d'un ou plusieurs **widgets**. Ceux-ci peuvent être des **zones de texte** (sur une seule ligne ou plusieurs lignes), des boîtes à **sélection**, des **boutons**, des **cases à cocher** ou des boutons **radio**.

La principale différence entre un formulaire HTML et un document HTML habituel réside dans le fait que, **généralement**, les **données** collectées par le formulaire sont **envoyées** vers un **serveur web**.

- Création d'un formulaire de contact simple en HTML

```
<form action="traitement-formulaire.html">
  Votre prénom<span class="rouge"> *</span>
  <input type="text" value="Votre prénom ici" class="contenuChampContact">
  Votre nom<span class="rouge"> *</span>
  <input type="text" value="Votre nom ici" class="contenuChampContact">
```

```
Votre adresse email<span class="rouge"> *</span>
<input type="text" value="Votre courriel ici" class="contenuChampContact">
Votre message <span class="rouge"> *</span>
<textarea class="contenuChampContact" rows="4" cols="50"
  maxlength="10" placeholder="Votre message"></textarea>
<p>
  <input type="reset" value="Réinitialiser">
  <input type="submit" value="Envoyer">
</p>
</form>
```

- Ajouter un fichier **.css** pour gérer le style du **formulaire**



- Analyse de l'attribut CSS **"display"**

https://www.w3schools.com/cssref/pr_class_display.asp

- Ajouter un **pointeur** dans le code HTML **vers** le fichier **.css**

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

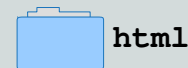
- Ajouter le **style** du **formulaire** de contact

```
.formulaireContact {  
  background-color: var(--orange-clair);  
  float: none;  
  margin-top: 1%;  
  width: 90%;  
  font-family: "Ubuntu-L", Helvetica, Arial, sans-serif;  
  color: var(--gris);  
  margin-bottom: 1%;  
  padding: 1%;  
}
```



```
.contenuChampContact {  
  display: block;  
  width: 90%;  
  font-family: "Ubuntu-L", Helvetica, Arial, sans-serif;  
  color: var(--gris);  
  margin-bottom: 2%;  
}
```

- Ajouter la page de traitement du formulaire



citrons.html
contact.html
index.html
pommes.html
sandbox.html
traitement-formulaire.html

`<h2 class="rouge">Votre formulaire de contact
a bien été envoyé au serveur!</h2>`

Remarque : quand nous appuyons sur le bouton "Envoyer", la page "contact.html" envoie le **contenu** du formulaire à la page "traitement-formulaire.html" qui **ne fait** en réalité **rien d'autre** que d'afficher le **message** "Votre formulaire de contact a bien été envoyé au serveur!", **sans rien traiter**.

Ceci est **normal**, car le **traitement** doit être effectué **par** le **serveur** qui doit vérifier le contenu du formulaire et **prendre** des **actions** réelles (par exemple créer un nouvel utilisateur dans une base de données). Il est nécessaire d'utiliser un **langage** de programmation "**côté serveur**" (PHP, Node.js, .NET, Java), ce qui ne fait **pas** partie des **sujets abordés** dans l'introduction HTML-CSS.

De plus, l'internaute moderne est habitué à recevoir la réponse du serveur sans quitter la page du formulaire. Ceci demande l'étude de requêtes "asynchrones" comme **AJAX** (**A**synchronous **J**avascript and **X**ML).

Remarque : afin de garantir une **meilleure expérience utilisateur** (User e**X**perience), il sera **également nécessaire** d'étudier un autre langage comme **JS** (**J**avascript), pour **aiguiller** et **corriger** l'internaute lorsqu'il remplit le formulaire.

- **Flex-container et flex-items**
- **FlexBox : axe principal de positionnement**
- **Propriétés CSS des "enfants" Flexbox**

- FlexBox (**Flex**ible **Box** Layout Module) est un nouveau modèle de "boîtes" CSS qui permet de positionner correctement et facilement les différents éléments sur une page HTML.
- Flex-container et flex-item

On déclare simplement un flex-container avec la propriété CSS "display"

```
display: flex;
```

Tous les éléments à l'intérieur de ce flex-container sont des flex-items

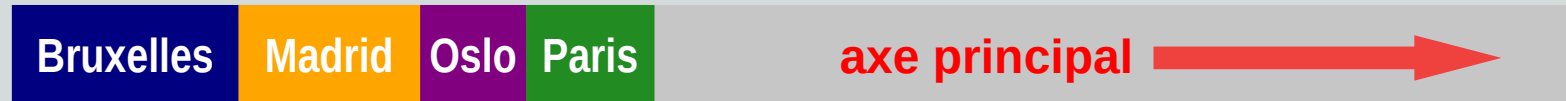


Remarque : n'importe quel élément HTML peut devenir un flex-container

Remarque : les flex-items sont donc les "enfants" directs d'un flex-container

FlexBox : axe principal de positionnement

```
flex-direction: row;
```



```
flex-direction: column;
```



Modifications initiales dans le fichier **sandbox.css**

```
.flexFieldset {  
  width : 50%;  
  font-weight: bold;  
  margin-bottom: 2%;  
}  
  
.flexContainer {  
  display: flex;  
  width: 100%;  
  background-color: lightgrey;  
}  
  
.flexContainerHeight {  
  height: 70px;  
}  
  
.flexItem {  
  color: white;  
  font-size: 150%;  
  font-weight: bold;  
  padding: 5px;  
}
```



```
.itemBruxelles {  
  background-color: navy;  
  color: white;  
}  
  
.itemMadrid {  
  background-color: orange;  
  color: white;  
}  
  
.itemOslo {  
  background-color: purple;  
  color: white;  
}  
  
.itemParis {  
  background-color: forestGreen;  
  color: white;  
}  
  
.directionRow {  
  flex-direction: row; /*direction par défaut*/  
}
```

```
.directionColumn {  
  flex-direction: column;  
}
```

- Code de démonstration de **flex-direction: row** dans [sandbox.html](#)

```
<h2 class="bleu">Le modèle FlexBox</h2>
```

```
<fieldset class="flexFieldset">
```

```
  <legend><span class="rouge gras">flex-direction:</span> row (défaut)</legend>
```

```
  <div class="flexContainer directionRow">
```

```
    <span class="flexItem itemBruxelles">Bruxelles</span>
```

```
    <span class="flexItem itemMadrid">Madrid</span>
```

```
    <span class="flexItem itemOslo">Oslo</span>
```

```
    <span class="flexItem itemParis">Paris</span>
```

```
  </div><!--Fin .conteneurFlex-->
```

```
</fieldset>
```

- Code de démonstration de **flex-direction: column** dans [sandbox.html](#)

Copier-collez le code du « fieldset » ci-dessus et modifiez-le comme suit :

```
<legend>flex-direction: column</legend>
```

```
<div class="flexContainer directionColumn">
```

FlexBox - justification des flex-items sur l'axe principal

Bruxelles Madrid Oslo Paris ← `justify-content: flex-start;`

`justify-content: center;` → Bruxelles Madrid Oslo Paris ←

`justify-content: flex-end;` → Bruxelles Madrid Oslo Paris

`justify-content: space-between;`

Bruxelles ← → Madrid ← → Oslo ← → Paris

`justify-content: space-around;`

→ Bruxelles ← → Madrid ← → Oslo ← → Paris ←

- Ajouts dans le fichier `sandbox.css`

```
.justifyStart {  
    justify-content: flex-start; /*justification par défaut*/  
}  
  
.justifyCenter {  
    justify-content: center;  
}  
  
.justifyEnd {  
    justify-content: flex-end;  
}  
  
.justifySpaceBetween {  
    justify-content: space-between;  
}  
  
.justifySpaceAround {  
    justify-content: space-around;  
}
```

- Code de démonstration de `justify-content: flex-start` dans `sandbox.html`

```
<legend><span class="rouge gras">justify-content:</span> flex-start (défaut)</legend>
<div class="flexContainer">
```

- Code de démonstration de `justify-content: center` dans `sandbox.html`

```
<legend>justify-content: center</legend>
<div class="flexContainer justifyCenter">
```

- Code de démonstration de `justify-content: flex-end` dans `sandbox.html`

```
<legend>justify-content: flex-end</legend>
<div class="flexContainer justifyEnd">
```

- Code de démonstration de `justify-content: space-between` dans `sandbox.html`

```
<legend>justify-content: space-between</legend>
<div class="flexContainer justifySpaceBetween">
```

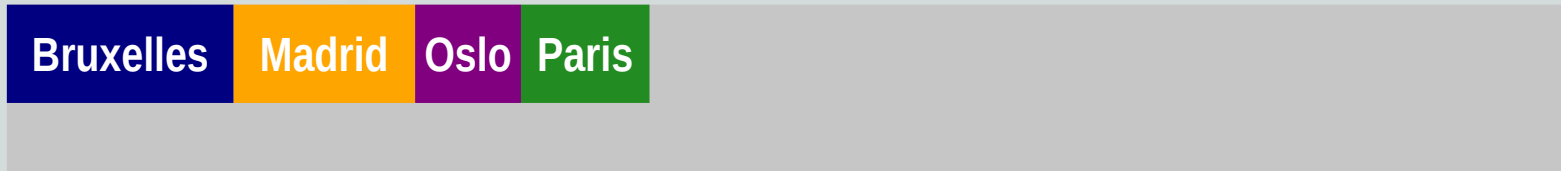
- Code de démonstration de `justify-content: space-around` dans `sandbox.html`

```
<legend>justify-content: space-around</legend>
<div class="flexContainer justifySpaceAround">
```

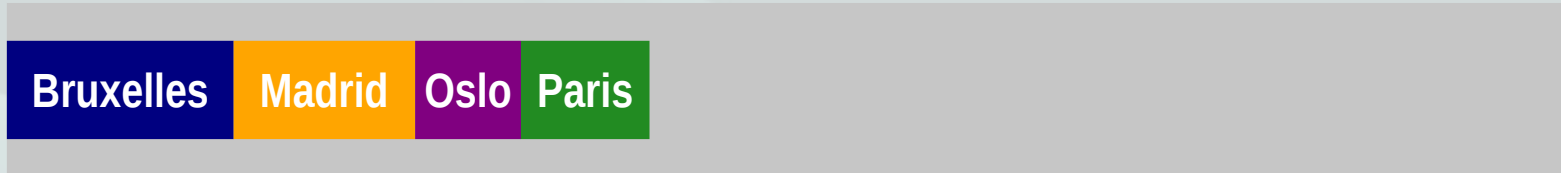
`align-items: stretch;`



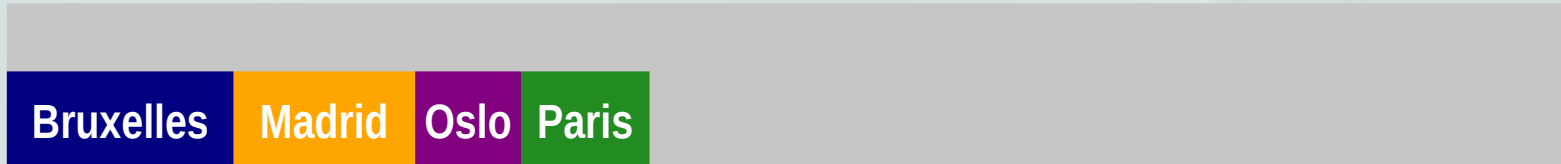
`align-items: flex-start;`



`align-items: center;`



`align-items: flex-end;`



- Ajouts dans le fichier `sandbox.css`

```
.alignStretch {  
    align-items: stretch; /*alignement axe secondaire par défaut*/  
}  
  
.alignStart {  
    align-items: flex-start;  
}  
  
.alignCenter {  
    align-items: center;  
}  
  
.alignEnd {  
    align-items: flex-end;  
}
```

- Code de démonstration de `align-items: stretch` dans `sandbox.html`

```
<legend><span class="rouge gras">align-items:</span> stretch</legend>  
<div class="flexContainer flexContainerHeight">
```

- Code de démonstration de `align-items: flex-start` dans `sandbox.html`

```
<legend>align-items: flex-start</legend>  
<div class="flexContainer flexContainerHeight alignStart">
```

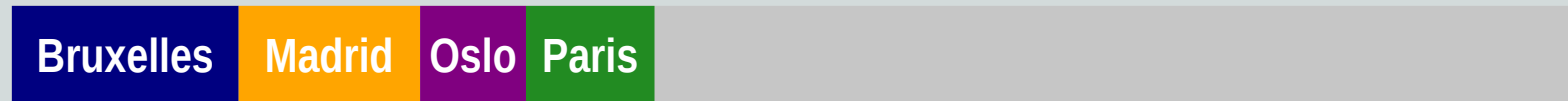
- Code de démonstration de `align-items: center` dans `sandbox.html`

```
<legend>align-items: center</legend>  
<div class="flexContainer flexContainerHeight alignCenter">
```

- Code de démonstration de `align-items: flex-end` dans `sandbox.html`

```
<legend>align-items: flex-end</legend>  
<div class="flexContainer flexContainerHeight alignEnd">
```

`flex-grow: 0;`



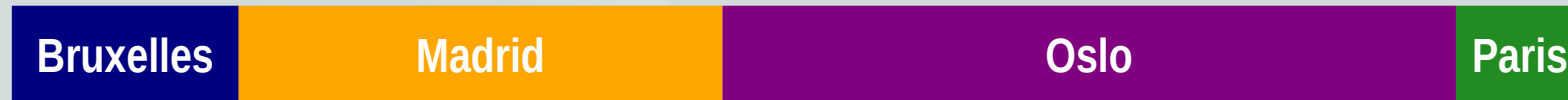
Remarque : chaque flex-item ne prend **que l'espace nécessaire** en fonction de son contenu

`flex-grow: 1 (Madrid);`



Remarque : le flex-item "Madrid" s'étend en prenant tout l'espace libre

`flex-grow: 1 (Madrid) et 2 (Oslo);`



Remarque : "Madrid" s'étend en prenant moins d'espace libre que "Oslo"

`flex-grow: 1 (tous les flex-items);`



Remarque : chaque flex-item s'étend en prenant une partie proportionnelle de l'espace libre

- Ajouts dans le fichier `sandbox.css`

```
.flexGrow0 {  
    flex-grow: 0; /*Facteur d'élargissement par défaut des flex-items*/  
}  
  
.flexGrow1 {  
    flex-grow: 1;  
}  
  
.flexGrow2 {  
    flex-grow: 2;  
}
```

- Code de démonstration de `flex-grow: 0` dans `sandbox.html`

```
<legend><span class="rouge gras">flex-grow:</span> 0 (défaut)</legend>  
<div class="flexContainer">
```

- Code de démonstration de `flex-grow: 1` dans `sandbox.html`

```
<legend>flex-grow: 1 (Madrid)</legend>  
    <span class="flexItem itemMadrid flexGrow1">Madrid</span>
```

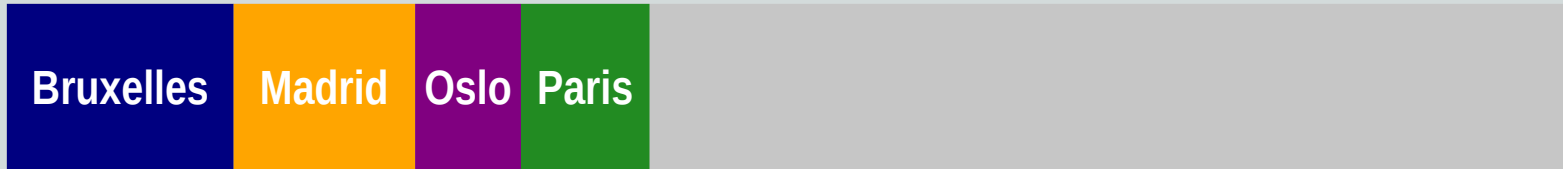
- Code de démonstration de **flex-grow: 1 et 2** dans [sandbox.html](#)

```
<legend>flex-grow: 1 (Madrid) et 2 (Oslo)</legend>  
<span class="flexItem itemMadrid flexGrow1">Madrid</span>  
<span class="flexItem itemOslo flexGrow2">Oslo</span>
```

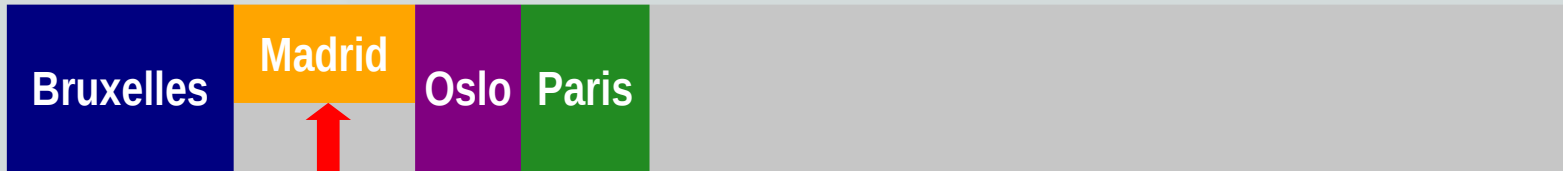
- Code de démonstration de **flex-grow: 1** pour tous les flex-items dans [sandbox.html](#)

```
<legend>flex-grow: 1 (tous les flex-items)</legend>  
<span class="flexItem itemBruxelles flexGrow1">Bruxelles</span>  
<span class="flexItem itemMadrid flexGrow1">Madrid</span>  
<span class="flexItem itemOslo flexGrow1">Oslo</span>  
<span class="flexItem itemParis flexGrow1">Paris</span>
```

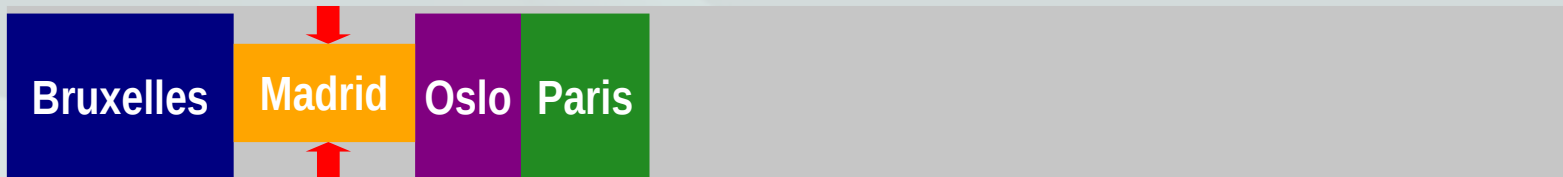
`align-self: stretch;`



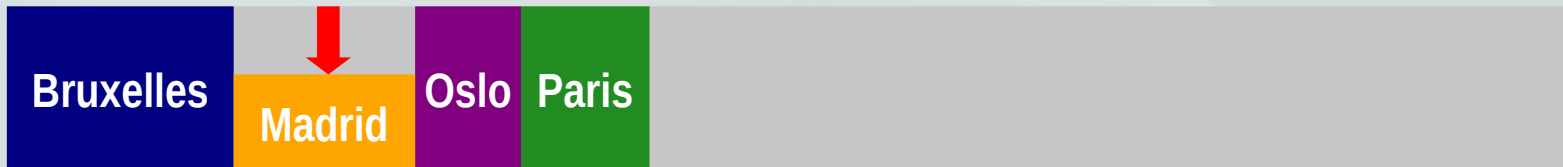
`align-self: flex-start;`



`align-self: center;`



`align-self: flex-end;`



- Ajouts dans le fichier **sandbox.css**

```
.alignSelfStretch {  
    align-self: stretch; /*Alignement par défaut par rapport aux "frères"*/  
}  
  
.alignSelfStart {  
    align-self: flex-start;  
}  
  
.alignSelfCenter {  
    align-self: center;  
}  
  
.alignSelfEnd {  
    align-self: flex-end;  
}
```

- Code de démonstration de **align-self: stretch** dans sandbox.html

```
<legend><span class="rouge gras">align-self:</span> stretch (défaut)</legend>  
<div class="flexContainer flexContainerHeight">
```

- Code de démonstration de **align-self: flex-start** dans sandbox.html

```
<legend>align-self: flex-start</legend>
<div class="flexContainer flexContainerHeight">
  <span class="flexItem itemMadrid alignSelfStart">Madrid</span>
```

- Code de démonstration de **align-self: center** dans sandbox.html

```
<legend>align-self: center</legend>
<div class="flexContainer flexContainerHeight">
  <span class="flexItem itemMadrid alignSelfCenter">Madrid</span>
```

- Code de démonstration de **align-self: flex-end** dans sandbox.html

```
<legend>align-self: flex-end</legend>
<div class="flexContainer flexContainerHeight">
  <span class="flexItem itemMadrid alignSelfEnd">Madrid</span>
```

- **Sémantique et éléments HTML sémantiques**
- **Nouvelle structure générale de nos pages**

- La sémantique est l'étude de la **signification** des **mots** et des **phrases** d'un langage.
- Un **élément HTML sémantique** est un élément dont le **nom décrit** clairement le type de **contenu** inclus, aussi bien pour les **développeurs** que pour les **navigateurs** et les **moteurs** de recherche.

Les balises `<div>` ou `` ne sont **pas sémantiques**, car leur nom ne dit rien à propos de leur contenu. Il est nécessaire de rajouter des attributs comme `class="contenuPrincipal"` mais dont le sens ne sera **pas forcément compris** par les navigateurs et les moteurs de recherche.

Avec **HTML5**, de nombreuses **balises sémantiques** sont apparues :

<code><article></code>	<code><aside></code>	<code><details></code>	<code><figcaption></code>	<code><figure></code>
<code><footer></code>	<code><header></code>	<code><main></code>	<code><mark></code>	<code><nav></code>
<code><section></code>	<code><summary></code>	<code><time></code>		

Les balises `<div>` ne seront dorénavant utilisées que dans le **cas** ou **aucune balise sémantique** n'est **appropriée**, par exemple pour créer un **"conteneur"** dont le seul but est de servir de **flex-container**.

`<body>` **FLEX** ↓

`<nav>` **FLEX** → ou ↓

Accueil Citrons Pommes Sandbox Contact

`<div class="content">` **FLEX** → ou ↓

`<main>`

`<aside>`

`<footer>` **FLEX** → ou ↓

`<address>`

```
<html>
...
<body>
  <nav>
    Contenu du menu de navigation ...
  </nav>

  <div class="contenu">
    <main>
      Contenu principal des pages ...
    </main>

    <aside>
      Contenu secondaire des pages ...
    </aside>
  </div><!--Fin de .contenu-->

  <footer>
    Contenu du pied de page ...
  </footer>
</body>
</html>
```


- Nouveau code **HTML** pour le contenu secondaire de toutes les pages

```
<aside>
  <p class="gras">Vous appréciez la réalisation de ce site?</p>
  <p>Téléchargez mon CV ci-dessous.</p>
  <a href="documents/moncv.pdf"></a>
  <p class="gras">Vous désirez me contacter?</p>
  <p>Utilisez le <a href="contact.html">formulaire</a> de contact.</p>
  <p class="gras">Pour approfondir HTML et CSS</p>
  <p>Visitez le site <a href="https://www.w3schools.com" target="_blank">
    www.w3schools.com</a></p>
</aside>
```

- Nouveau code **HTML** pour le pied de page de toutes les pages

```
<footer>
  <address>
    Éditeur responsable <a href="mailto:jules.cesar@romeantique.it"
      class="adresseCourriel">Jules CESAR</a>
    <p>Rue du Colisée - Rome - Italie antique</p>
  </address>
</footer>
```

Modification du fichier `global.css`

```
html {
  --vert: #2f4f4f;
  --vert-clair: #bfd9d9;
  --vert-tres-clair: #eff5f5;
}

body {
  display: flex;
  flex-direction: column;
}

nav {
  display: flex;
  background-color: var(--vert);
  padding: 5px;
}

.lienMenu {
  padding-right: 1%;
}
```

```
.contenu {
  display: flex;
}

main {
  background-color: aqua; /* Only necessary for FlexBox setup */
  width: 70vw;
  padding: 1%;
}

aside {
  background-color: var(--vert-clair);
  flex-grow: 1;
  padding: 1%;
}

audio, video {
  width: 100%;
}

footer {
  display : flex ;
  background-color: var(--vert);
  color: white;
  padding: 1%;
}
```

```
.adresseCourriel {  
  color: white;  
}
```

```
.imageAccueil {  
  max-width: 70vw;  
}
```

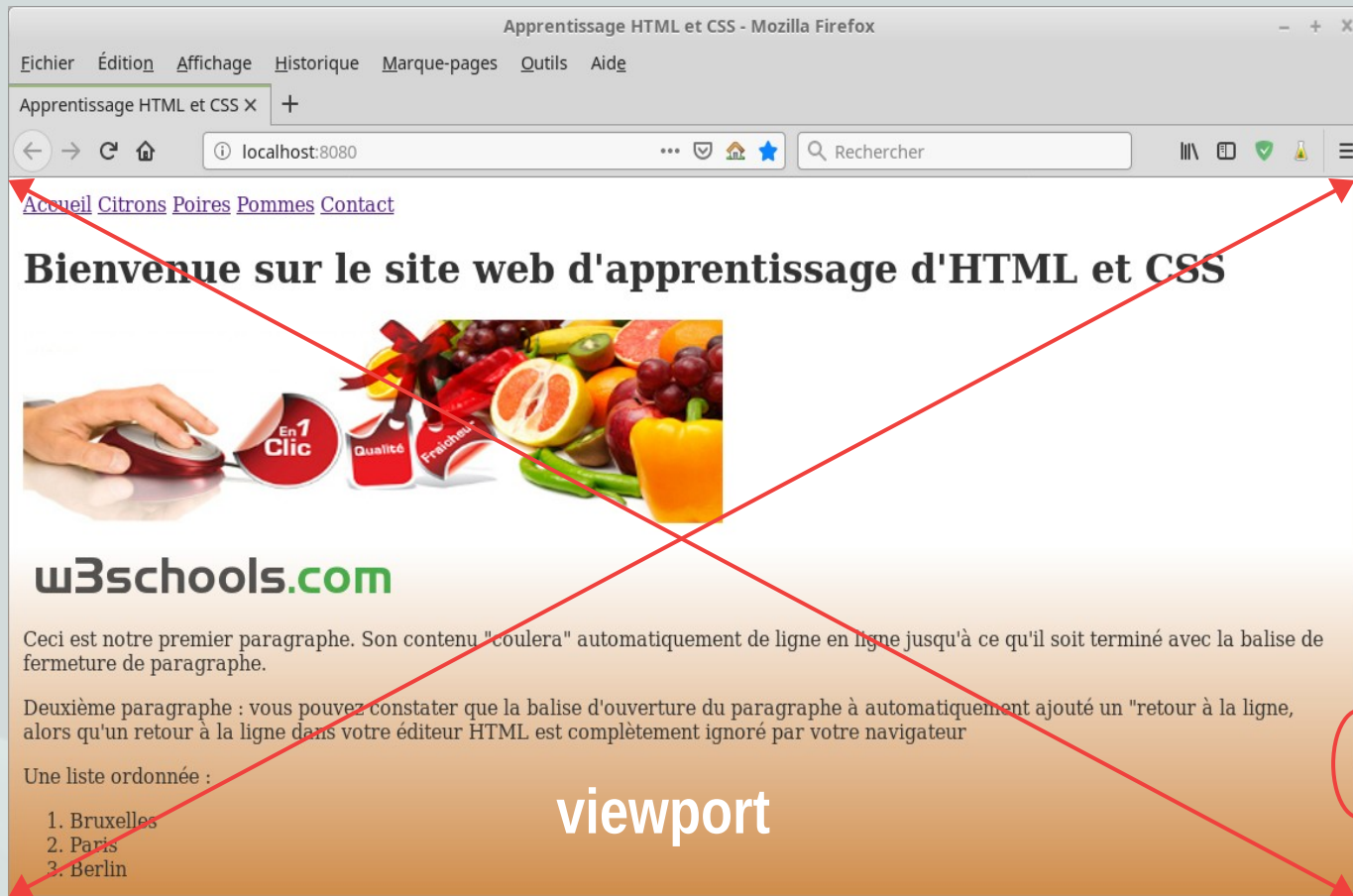
Modification du fichier [index.html](#)

```

```

- Le "viewport" ou la zone visible d'une page web
- Adapter le style en fonction des différents "viewports"

Le "viewport" (zone visible sans utilisation des barres de défilement)



- Adapter le style en fonction des différents "viewports"

Les "**media queries**" de CSS3 permettent d'adapter le style d'une page HTML à des conditions particulières, en demandant au navigateur utilisé (téléphone, ordinateur) de communiquer des informations telles que la taille du "**viewport**" ou d'autres informations comme les conditions de luminosité.

Concrètement, nous définirons des "points de transition" (**break points**) correspondant à des "**viewports**" spécifiques et pour chacun d'eux nous définirons un style à appliquer (ex. taille des caractères, la taille des marges ...).

Remarque : l'expérience montre que pour gérer efficacement les différences entre "viewports", il est recommandé de toujours définir en premier le style pour les petits écrans et d'assurer un nombre suffisant de "break points" vers les "viewports" de plus en plus grands.

```
@media screen and (width: 600px) {  
    font-size: 10pt;  
}
```

Analysons sur le site www.w3schools.com les différents "**types de vérifications**" (width ...) et les différents "**media types**" (screen ...) disponibles.

https://www.w3schools.com/css/css3_mediaqueries.asp

- Démonstration d'utilisation des media queries

```
main {  
    background-color: yellow;  
    width: 70vw;  
    padding: 1%;  
}  
@media screen and (min-width: 800px) {  
    main {  
        background-color: aqua;  
    }  
}
```

- Modification de la taille des caractères du « body » avec les media queries

```
body {  
    display: flex;  
    flex-direction: column;  
    min-height: 100vh;  
    background-color: white;  
    font-size: 90%;  
    font-weight: normal;  
    font-family: "Ubuntu-L", Helvetica, Arial, sans-serif;  
}
```

- ```
@media screen and (min-width: 800px) {
 body {
 font-size: 100%;
 }
}
```

- Modification du menu de navigation avec les media queries

```
nav {
 display: flex;
 background-color: var(--vert);
 font-size: 90%;
 padding: 5px;
}
```

```
@media screen and (min-width: 800px) {
 nav {
 font-size: 100%;
 }
}
```

- Déplacement de la zone « aside » avec les media queries

```
.contenu {
 background-color: var(--vert-clair);
 display: flex;
 flex-direction: column;
 align-items: center;
}
@media screen and (min-width: 800px) {
 .contenu {
 flex-direction: row;
 align-items: flex-start;
 }
}
```

- Adaptation de la largeur de la zone « main » avec les media queries

```
main {
 background-color: var(--vert-tres-clair);
 width: 98%;
 padding: 1%;
}
```

```
@media screen and (min-width: 800px) {
 main {
 width: 70%;
 }
}
```

- Adaptation de la largeur de la zone « aside » avec les media queries

```
aside {
 background-color: var(--vert-clair);
 padding: 1%;
}

@media screen and (min-width: 800px) {
 aside {
 flex-grow: 1;
 }
}
```

## Modification des balises <h1></h1> avec les media queries

```
h1 {
 font-family: "Ubuntu-M", Helvetica, Arial, sans-serif;
 font-size: 100%;
 font-weight: bold;
}

@media screen and (min-width: 800px){
 h1 {
 font-size: 130%;
 }
}
```



## Gérer les images de manière « responsive » avec les média queries

```
.imageFruits {
 width: 200px;
}

@media screen and (min-width: 800px) {
 .imageFruits {
 width: 300px;
 }
}

.logoHtmlCss {
 width: 60px;
}

@media screen and (min-width: 800px) {
 .logoHtmlCss {
 width: 80px;
 }
}
```

```
.imagePDF {
 width: 60px;
}

@media screen and (min-width: 800px) {
 .imagePDF {
 width: 80px;
 }
}
```

### Modifications dans le fichier « index.html »

```

```

```

```

### Modifications dans les fichiers « citrons.html » et « pommes.html »

```

```

```

```