R1.02

Développement d’interfaces web

(Version 13 septembre 2023)

Introduction et HTML

Objectif du module :

- Savoir structurer et présenter des contenus numériques

Contenus :

- *La séparation contenu-structure-présentation*
- *Les technologies du Web (World Wide Web) pour la description de documents et d'interfaces: HTML (HyperText Markup Language), CSS (Cascading Style Sheets),* chartes graphiques
- Sensibilisation à l'ergonomie
- Outils bureautiques de production de documents numériques
- Conformité des sites Web aux standards d'accessibilité W3C / WAI (World Wide Web Consortium / Web Accessibility)

Répartition temporelle

- CM : 3 x 1h20
    1. Introduction + HTML 5
    2. CSS 3 : Base (Sélecteurs, règles et modèle de boite, positionnement, flottants)
    3. CSS 3 : Complément (FlexBox, CSS Grid)

- TD : 4 x 2h40
    1. HTML : Les bases + les formulaires
    2. CSS 3 : Les Bases
    3. CSS3 : Compléments
    4. CSS3 : Compléments suite + début DM
- Evaluations
    5. Evaluation CSS + HTML
    6. Rendu et soutenance DM

Intervenants et supports

- Jean-François Berdjugin
- Nicolas Lance

https://madoc.univ-nantes.fr/course/view.php?id=49138

Progression

- R1.02 : Développement d'interfaces web
- R1.05 : Introduction aux bases de données et SQL

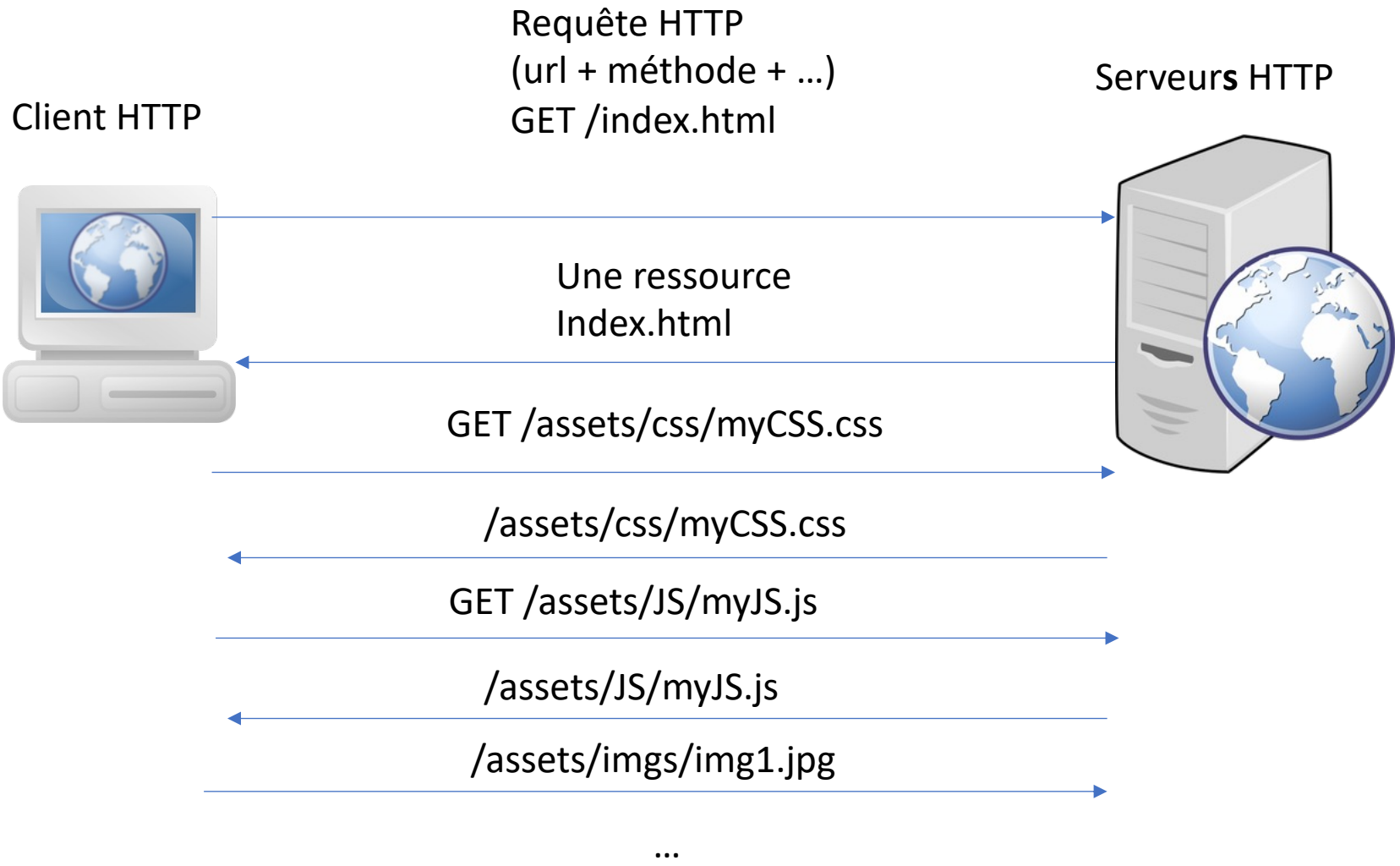
- R2.01 : Développement orienté objets
- R2.05 : Introduction aux services réseaux
- R2.06 : Exploitation d'une base de données

- R3.01 : Développement web =>PHP
- R3.06 : Architecture des réseaux
- R3.07: SQL dans un langage de programmation

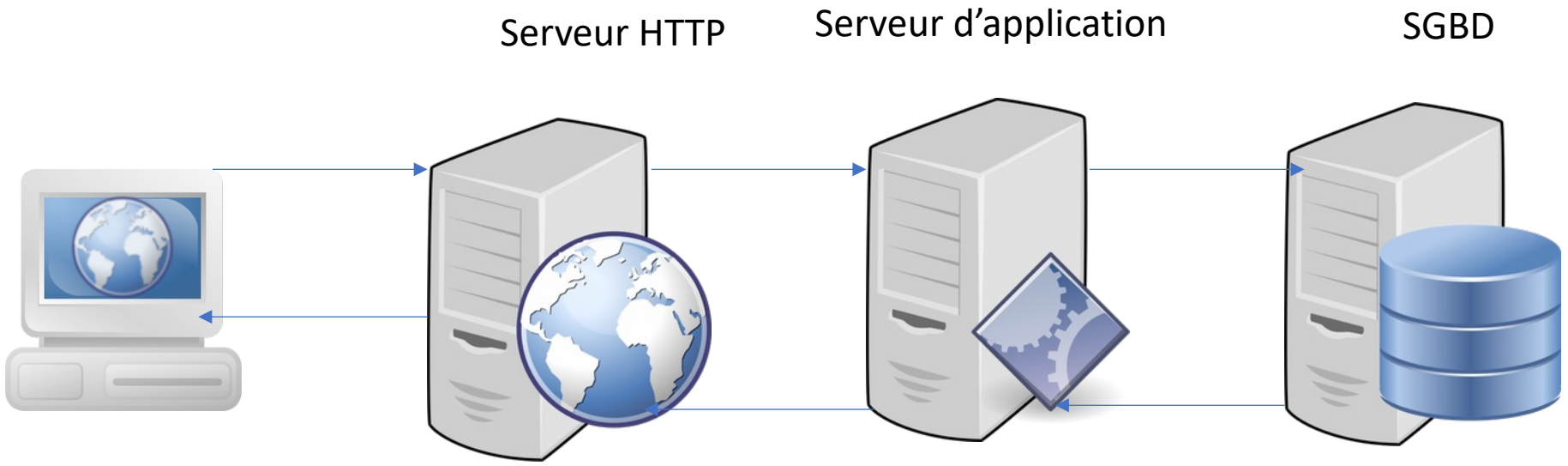
- R4.03:Qualité et au-delà du relationnel
- R4.Real.10 : Complément web =>JS

- R5.Real.05 : Programmation avancée => JEE

Une application web de type : statique



Une application web de type : dynamique



Toujours des ressources mais générées dynamiquement

HTML 5

- Rôle
- Un peu de culture
- Sites de références
- Une première page
- Les balises (principales)
  - Premier niveau
  - D’en-tête
  - D’architecture
  - De structuration du texte
  - De liste
  - De tableau
  - De formulaire
  - Multimédia
- Un exemple avec des ancres
- Un exemple de formulaire
  - Communication avec le serveur
  - Validation

Rôle

Structurer le contenu et non la présentation d’une page web, la présentation sera réalisée avec la CSS

Un peu de culture

Chronologie

- 1986 : SGML (Standard Generalized Markup Language), le premier langage à balise
- 1991 : Tim Berners-lee
  - HTTP (Hyper Text Transfer Protocol),
  - HTML (Hyper Text Markup Language).
  - Naissance du World Wide Web (WWW).
- 1996 :
  - HTML 2.0,
  - Hâkon Wium Lie: CSS (Cascade Style Sheet),
  - W3C (World Wide Web Concortium)
- 1997-1999 :
  - HTML 3.2, 4.0, 4.01
  - CSS 2.0
  - XML (eXtensible Markup Language)

2000 : XHTML 1.0 conforme à XML

Abandon de XML

2012 : HTML5 doit devenir « recommandation candidate »

2016 : HTML5.1

Un peu de culture

Les organismes de normalisation :

W3C : World Wide Web Concortium

organisme de standardisation à chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML5 et CSS

https://html.spec.whatwg.org/

IETF : Internet Engineering Task Force, élabore et promeut des standards Internet

WHATWG : Le Web Hypertext Application Technology Working Groupe

une collaboration non officielle des différents développeurs de navigateurs web ayant pour but le développement de nouvelles technologies destinées à faciliter l'écriture et le déploiement d'applications à travers le Web (Apple, Mozilla, opera, …)

Un site de référence pour ce cours

https://developer.mozilla.org/fr/docs/Web/HTML

Seul site accessible le jour de l'examen

https://validator.w3.org/

Pour valider vos pages

Il est également possible de rajouter des extension dans vos navigateurs

Une première page HTML5 : valide

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

- Le **doctype** indique le type de document ici HTML5
- Balises
  - html, head, meta (orpheline), title, body
- Attributs
  - Lang, charset
- Les éléments sont des regroupement de balises

Une première page HTML5 : invalide

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <meta charset="UTF-8">
</head>
<body>

</body>
</html>
```

- Un doctype implique une grammaire et une sémantique, ici la balise head doit contenir une balise title.

Balises : premier niveau

- html
- head
- body

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
</head>
<body>

</body>
</html>
```

Balises : balises d’en-tête

- title : le titre
- meta : information complémentaire sur la page
- link : une relation vers une ressource externe (css, favicon, rss, …)
- style : charger une css
- script, noscript : charger un script (du JS) ou afficher une erreur

- base

```
<head>
    <meta charset="UTF-8">
    <title>Etape1 : un menu </title>
    <link href="assets/css/etape1.css" rel="stylesheet">
    <script src="javascript.js"></script>
</head>
```

Balises : balises d’architecture

Permettent de définir la sémantique du document, utiles pour le référencement

- <header> : définit l'en-tête d'une section ou d'une page
- <footer> :définit le bas d'une section ou d'une page
- <hgroup> : définit les informations d'en-tete d'une section ou d'une page
- <details> : définit les détails d'un élément
- <summary> : définit l'en-tête des details d'un élément
- <menu> : définit un menu en forme de liste
- <section> : définit une section
- <article> : définit un article
- <aside> : définit un élément latéral
- <nav> : définit un groupe de liens de navigations
- <iframe> : Introduis une page html dans une frame
- : Calque ou section
- : Section de type inline

Balises : balises de structuration du texte

<h1> à <h6> : titre

<p> Paragraphe

<a> Lien

<b> Texte en gras

<meter> Mesure

<pre> Texte préformaté

 Saut de ligne

<wbr> Empl pr saut 2 ligne

<blockquote> Longue citation

<q> Courte citation

<samp> Echantillon

<var> Variable

<kbd> Raccourci clavier

<bdo> Sens du texte

 Mise en exergue

  Italique en exergue

<i> italique

<mark> Marqueur de texte

<small> Rétrécis le texte

<sub> Mise en indice

<sup> Mise en Exposant

<adress> Définit une adresse

<cite> Citation

<abbr> Abréviation

<dfn> Définition

<del> Texte supprimé

<ins> Texte ajouté

<time> Date / Horaire

<code> Portion de code

Balises : balises de liste

 Liste non-ordonné

 Liste ordonné

 Elément de liste

<dl> Liste de définition

<dt> Terme à definir

<dd> Définition du terme

```
<ul>
  <li> un </li>
  <li> deux    <ol>
                 <li> deux un </li>
               </ol>
  </li>
</ul>
```

Balises : balises de tableau

<table> Tableau
<caption> Titre du tableau
<thead> En-tête du tableau
<tbody> Corps du tableau
<tr> Ligne du tableau
<th> Cellule d'en-tête
<td> Cellule du tableau
<tfoot> Bas du tableau
<col> Colonne du tableau
<colgroup> Groupe de colonne

```
<table>
  <thead>
    <tr>
      <th colspan="2">The table header</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>The table body</td>
      <td>with two columns</td>
    </tr>
  </tbody>
</table>
```

Balises : balises de formulaire

<form> Formulaire

<label>Titre d'un élément

<datalist> Liste déroulante

<select> Liste selectionnable

<option> Elémént d'une liste

<optgroup> Grp d'élmnts d'une list

<textarea> Zone de texte

<button> Bouton cliquable

<output> Définit un type de sortie

<input /> saisie de type : text/ email/ password / submit /checkbox/ button / hidden / file / radio / hidden / range / time / color / image / reset / url / date / month / search / datetime /number

<fieldset> Regroupe plusieurs éléments du formulaire

<legend> Titre d'un groupe

```
<form action="" method="get" class="form-example">
  <div class="form-example">
    <label for="name">Enter your name: </label>
    <input type="text" name="name" id="name" required>
  </div>
  <div class="form-example">
    <label for="email">Enter your email: </label>
    <input type="email" name="email" id="email" required>
  </div>
  <div class="form-example">
    <input type="submit" value="Subscribe!">
  </div>
</form>
```

Balises : balises multimédia

<img > Image ou photo

<canvas> Graphique

<progress> Progression

<figure> Groupe d'element multimédia

<ficaption> Légende du groupe d'élément

<video> Vidéo

<source> Source du media

<area> Zone cliquable à l'interieur d'une image

<audio> Contenu audio

<map> Carte / image

<param> Parametre d'objet

<embed > Contenu exterieur

<object> Objet du contenu exterieur

```
<figure>
   <img src="media/examples/elephant-660-480.jpg"
        alt="Elephant at sunset">
   <figcaption>An elephant at sunset</figcaption>
</figure>
```

Un exemple avec des ancres

demo_ancre.html

-  Cochon
-  Furet

demo_ancre_cochon.html

Cochon

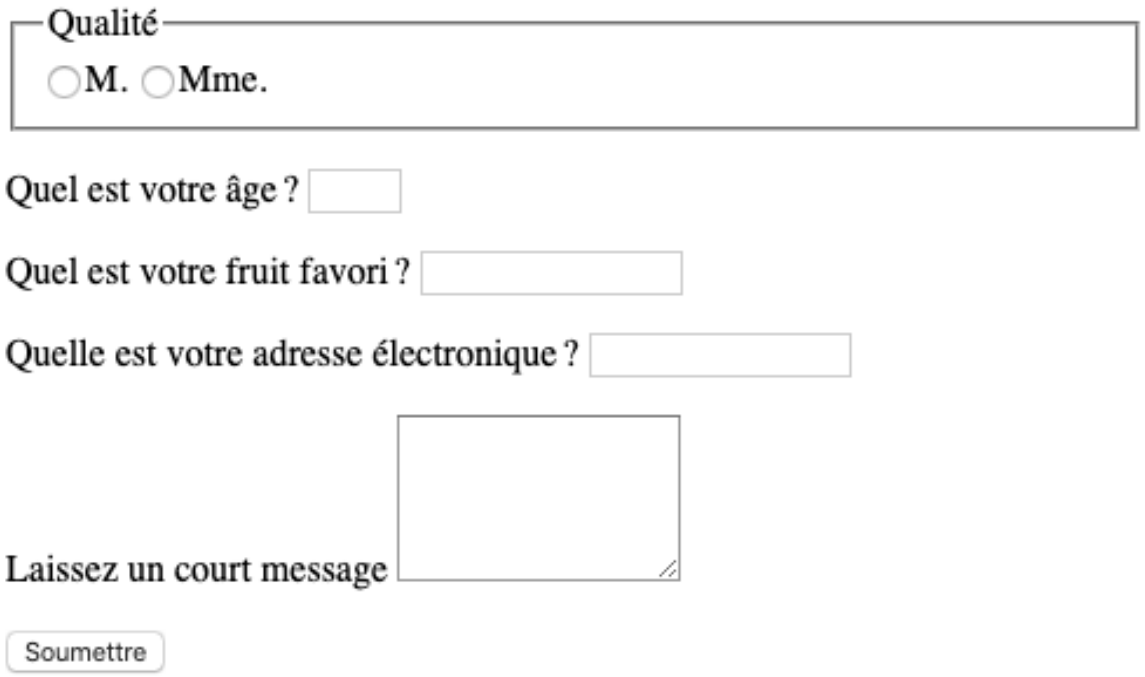
Home

demo_ancre_furet.html

Furet

Home

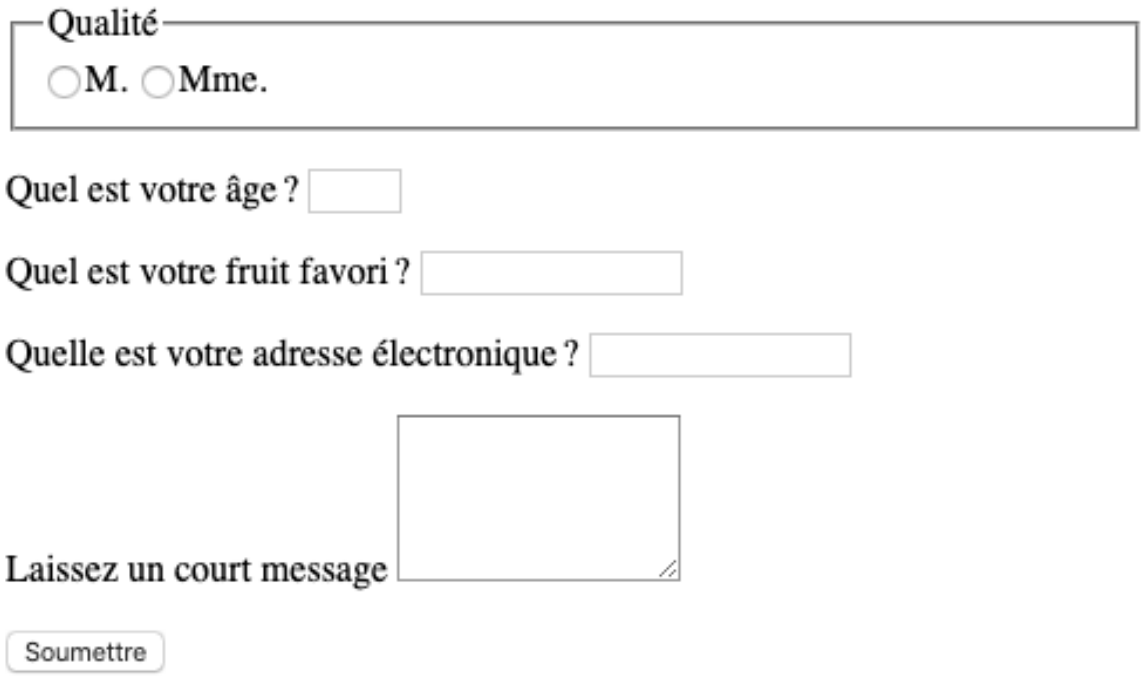
Un exemple de formulaire : objectif de communication



Communication :

- Les informations seront envoyée avec la méthode POST au script info_request.php s'exécutant sur localhost et écoutant sur le port 8080
  - La qualité sera reçue comme title
  - L'âge comme age
  - Le fruit comme fruit
  - L'adresse électronique comme email
  - Le message court comme msg

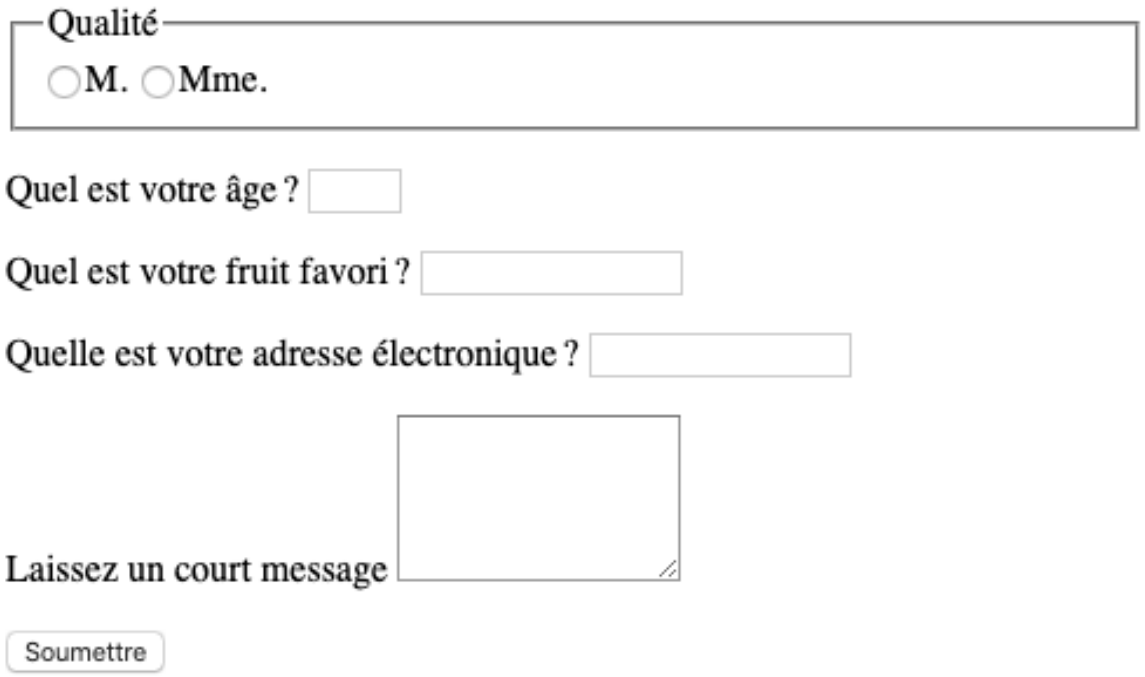
Un exemple de formulaire : objectif d'ergonomie



Ergonomie :

- Les clics sur les champs texte transmettrons le focus input correspondant
- Le formulaire pourra être envoyer avec un enter
- Des informations seront affichée si la souris demeure sur « Qualité » et « quelle est votre adresse électronique »

Un exemple de formulaire : objectif de validation



Validation :

- La qualité est obligatoire
- L'âge doit être compris entre 12 et 120
- Les fruit possibles sont Banane, banane, Cerise, cerise, Citron, citron, Fraise, fraise, Orange, orange, Pomme, pomme
- L'adresse électronique doit être valide

Un exemple de formulaire : code

```
<form action="http://localhost:8080/info_request.php" method="post">

   <fieldset>
      <legend title="Ce champ est obligatoire">Qualité </legend>
      <input type="radio" required name="title" id="r1" value="Mr"><label for="r1">M.</label>
      <input type="radio" required name="title" id="r2" value="Ms"><label for="r2">Mme.</label>
   </fieldset>

   <p>
      <label for="n1">Quel est votre âge ?</label>
      <input type="number" min="12" max="120" step="1" id="n1" name="age">
   </p>
   <p>
      <label for="t1" title="Ce champ est obligatoire"> Quel est votre fruit favori ?</label>
      <input type="text" id="t1" name="fruit" list="l1" required
            pattern="[Bb]anane|[Cc]erise|[Cc]itron|[Ff]raise|[Oo]range|[Pp]omme">
      <datalist id="l1">
         <option>Banane</option>
         <option>Cerise</option>
         <option>Citron</option>
         <option>Fraise</option>
         <option>Orange</option>
         <option>Pomme</option>
      </datalist>
   </p>
   <p>
      <label for="t2">Quelle est votre adresse électronique ?</label>
      <input type="email" id="t2" name="email">
   </p>
   <p>
      <label for="t3">Laissez un court message</label>
      <textarea id="t3" name="msg" maxlength="140" rows="5"></textarea>
   </p>
   <p>
      <input type="submit" value=" Soumettre " >
   </p>
</form>
```

CSS 3 : partie 1 Sélecteurs, règles et modèle de boîte

- Rôle
- Vocabulaire : sélecteurs et règles
- Chargement d'une CSS
- Sélecteurs
  - simples
  - combinateurs
  - pseudo classe
  - pseudo éléments
- Modèle de boîtes
  - content, padding, border, margin
  - taille
  - débordement
  - fusion des marges
  - display : inline, inline-block, block
  - unités
- Exemples
  - un body qui occupe la hauteur du viewport
  - un premier gabarit 3 colonnes
- Propriétés
  - de listes
  - de couleur et de fond
  - propriétés de boîte
  - propriétés de positionnement et d'affichage
  - Application au gabarit trois colonnes

CSS 3 : rôle

Les Cascading Style Sheets ont pour rôle la mise en forme (la disposition) de documents HTML.

Style Sheets : en imprimerie, un ensemble d'attributs de caractères et de formats de paragraphes pouvant être appliqués en une seule opération à un ou plusieurs paragraphes, à un livre.

Cascading : Comment les styles CSS se combinent entre eux

et se transmettent de balises en balises : en cascade

CSS3 : sélecteur, règle

Commentaire : /* */

Un ensemble de règles

sélecteur {

règle1;

…

}

Avec une règle : propriété : valeur 1 …


```
<!DOCTYPE html>
<html lang= »fr">
<head>
   <meta charset="UTF-8">
   <title>Title</title>
   <style>
      p {
         background-color: aqua; /*prop background-color valeur aqua */
         border: black 1px solid;
      }
   </style>
</head>
<body>
<p> blabla </p>
</body>
</html>
```

CSS3 : Comment la charger

- Via une balise link (dans head)

<link rel="stylesheet » href="myCSS">

Rem : les feuilles de styles peuvent également importer d'autres feuilles de style via @import

- Dans une balise style

<style>

p {color : aqua}

</style>

- Dans une balise avec l'attribut style

<p style="color : aqua">

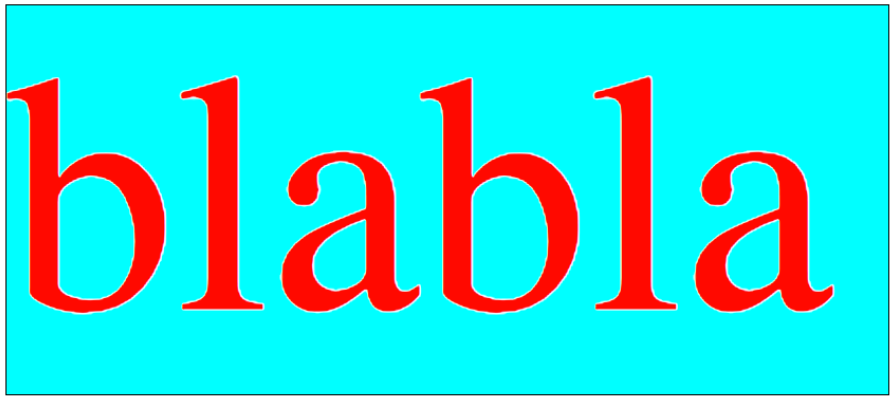
CSS3 : comment la charger

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        p {
            background-color: aqua;
            border: black 1px solid;
        }
    </style>
    <link rel="stylesheet" href="base.css">
</head>
<body>

<p style="color : red"> blabla </p>

</body>
</html>
```

```
/* base.css */
body {
    font-size: 20em;
}
```

=> Utiliser link

Rem : différant en JS



CSS3 : les sélecteurs simples

* : sélecteur **universel** tous les éléments de la pages

Elément : sélecteur de **type**, un élément particulier comme par exemple un paragraphe

.nomDeLaclasse : les éléments appartenant à la **classe**

```
* { color: red }

p { color : red }

<p class="classe1 classe2" > </p>
<div class="classe1"> </div>
.classe1{/* p et div */}
.classe2{/* p */}
```

CSS3 : les sélecteurs simples

#nomDeLIdentifiant : l'élément d'**identifiant** correspondant

```
<p class="classe1 classe2" id="id1" > </p>

<div class="classe1"> </div>

#id1 {/* p */}
```

CSS3 : les sélecteurs simples

[attr], [attr=valeur], [attr~=valeur], [attr|=valeur], [attr^=valeur], [attr$=valeur], [attr*=valeur] : sélecteurs **d'attributs** () avec :

* qui contient

$ qui finit par

~ correspondance exacte parmi une liste

| correspondance exacte ou qui commence par valeur_

^ qui commence

Possibilité d'ajouter i(I) ou s(S) pour la sensibilité à la casse

```
<style>
   a { color: blue; }
   a[href^="#"] { background-color: gold; }
   a[href*="example"] { background-color: silver;}
   a[href*="insensitive" i] {color: cyan;}
   a[href*="cAsE" s] {color: pink;}
   a[href$=".org"] {color: red;}
</style>
```

```
<ul>
   <li><a href="#internal">Lien interne<a></li>
   <li><a href="http://example.com">Lien
d'exemple</a></li>
   <li><a href="#InSensitive">Lien interne insensible à
la casse</a></li>
   <li><a href="http://example.org">Lien vers
example.org</a></li>
</ul>
```

CSS3 : les sélecteurs simples

[attr], [attr=valeur], [attr~=valeur], [attr|=valeur], [attr^=valeur], [attr$=valeur], [attr*=valeur] : sélecteurs **d'attributs** () avec :

* qui contient

$ qui finit par

~ correspondance exacte parmi une liste

| correspondance exacte ou qui commence par valeur_

^ qui commence

Possibilité d'ajouter i(I) ou s(S) pour la sensibilité à la casse

- Lien interne
- Lien d'exemple
- Lien interne insensible à la casse
- Lien vers example.org

```
<style>
   a { color: blue; }
   a[href^="#"] { background-color: gold; }
   a[href*="example"] { background-color: silver;}
   a[href*="insensitive" i] {color: cyan;}
   a[href*="cAsE" s] {color: pink;}
   a[href$=".org"] {color: red;}
</style>
```

```
<ul>
   <li><a href="#internal">Lien interne<a></li>
   <li><a href="http://example.com">Lien
d'exemple</a></li>
   <li><a href="#InSensitive">Lien interne insensible à
la casse</a></li>
   <li><a href="http://example.org">Lien vers
example.org</a></li>
</ul>
```

CSS3 : les sélecteurs combinateurs

A B : descendants

A + B : voisin direct

A ~ B : voisins

A > B : fils

Rem : d'autres spécifiques aux tableaux

Séléceteur1 , Sélécteur2 : le sélécteur1 et le sélécteu2

```
<style>
   ul {color: red}
   ul ul {color : blue}
   ul ~p {color : aqua}
   ul+p {color:green}
   span {color: black}
   p > span {color: brown}
</style>
```

```
<ul>
   <li> item 1 </li>
   <li> item 2</li>
   <li> <ul>
         <li id="toto"> item 3.1</li>
      `  <li> item 3.2</li>
         <li> item 3.3</li>
      </ul>
   </li>
</ul>
<p> hhhh </p>
<p> jjjjj
   <span> fils
      <span>
         petit fils <span> arriere </span>
      </span>
   </span>
</p>
```

CSS3 : les sélecteurs combinateurs

A B : descendants

A + B : voisin direct

A ~ B : voisins

A > B : fils

Rem : d'autres spécifiques aux tableaux

Séléceteur1 , Sélécteur2 : le sélécteur1 et le sélécteu2

- item 1
- item 2
- item 3.1
  - item 3.2
  - item 3.3

hhhh

jjjjj fils petit fils arriere

```
<style>
   ul {color: red}
   ul ul {color : blue}
   ul ~p {color : aqua}
   ul+p {color:green}
   span {color: black}
   p > span {color: brown}
</style>


<ul>
   <li> item 1 </li>
   <li> item 2</li>
   <li> <ul>
         <li id="toto"> item 3.1</li>
      `  <li> item 3.2</li>
         <li> item 3.3</li>
      </ul>
   </li>
</ul>
<p> hhhh </p>
<p> jjjjj
   <span> fils
      <span>
         petit fils <span> arriere </span>
      </span>
   </span>
</p>
```

CSS3 : les sélecteurs pseudo classe

- Une **pseudo-classe** est un mot-clé qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la déclaration.

- Exemple a:hover une ancre reçoit le focus

:**active**, :any-link, :blank, :checked, :current, :default, :defined, :dir(), :disabled, :drop, :empty, :enabled

:first; :first-child, :first-of-type,

:fullscreen, :future; :**focus**, :focus-visible , :**focus-within**

:has(), :host, :host(), :host-context()

:**hover**, :indeterminate, :in-range, :invalid, :is, :lang(), :last-child,

:last-of-type, :left, :link, :local-link, :not(),

:nth-child(), :nth-col(), :nth-last-child(), :nth-last-col(), :nth-last-of-type(), :nth-of_type()

:only-child, :only-of-type, :optional, :out-of-range,:past , :placeholder-shown ,:read-only,:read-write

:required,:right,:root,:scope,:target,:target-within ,:user-invalid

:valid, :**visited**, :where()

CSS3 : les sélecteurs pseudo éléments

Un pseudo-élément est un mot-clé ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle. Ainsi, le pseudo-élément ::first-line permettra de ne cibler que la première ligne d'un élément visé par le sélecteur.

/* La première ligne de chaque élément <p>. */ p::first-line { color: blue; text-transform: uppercase; }

CSS3 : les sélecteurs pseudo éléments

::**after** (:after)
::backdrop
::**before** (:before)
::cue (:cue)
::first-letter (:first-letter)
::first-line (:first-line)
::grammar-error
::marker
::part
::placeholder
::selection
::slotted()
::spelling-error

Un exemple utile plus tard

```
<style>
   div {
      background-color: brown;
   }
   div::before {
      content: 'avant';
   }
   div::after {
      content: 'after';
   }
</style>


<div>
   contenu
</div>
```

CSS 3 spécificité du sélecteur

- On compte le nombre d'ID => priorité au nombre le plus élevé
- On compte le nombre de classes et pseudo-classes => priorité au nombre le plus élevé
- On compte le nombre d'éléments et pseudo-éléments => priorité au nombre le plus élevé
- On ignore le sélecteur universel (*)

Rem :

!important permet de surpasser les règles

```
#content  nav li a.on:hover {
 color: red;
}
```

1 ID (#content), 2 classe+pseudo-classe (.on :hover), 3 éléments (nav li a) => 123

Il surpassera ce sélecteur:

```
nav.verte li.gras a.on:hover {
 color: green;
}
```

0 ID, 4 classe+pseudo-classe (.on :hover .verte .gras), 3 éléments (nav li a) => 33

Bonnes pratiques : Don’t repeat yourself

- Identifier ce qui se répète, le factoriser et utiliser plusieurs class par élément

```
.btn {
  background-color: #001534;
  color: #15DEA5;
  padding: 1.5rem;
}

et
.btn-wide {
  width: 100%;
}

/* And... */
.btn-rounded {
  border-radius: 3rem;
}

< button class="btn btn-rounded btn-wide" >...</button>
```

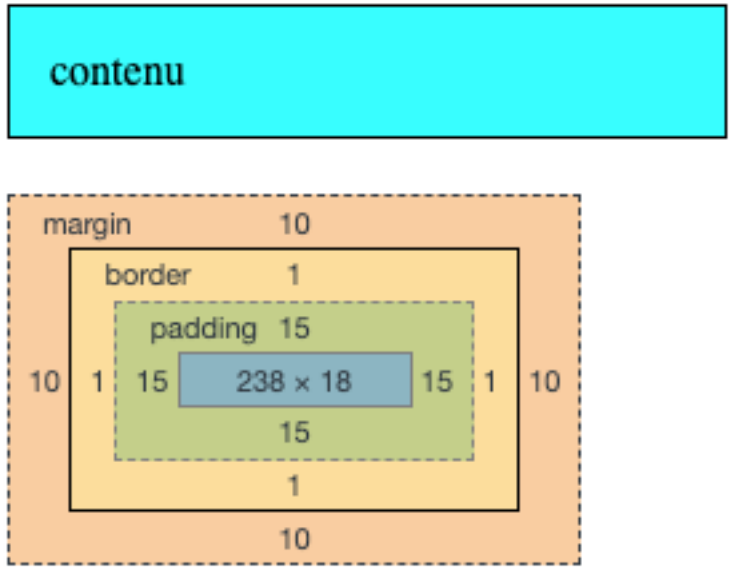
Bonnes pratiques : bloc, élément, modificateur

[block]__[element]--[modifier]

```
<figure class="photo">
        <img class="photo__img photo__img--highlighted" src="me.jpg">
        <figcaption class="photo__caption">
        <blockquote class="photo__quote">
                Look at me!
        </blockquote>
        </figcaption>
</figure>
<style>
        .photo { }
        .photo__img { }
        .photo__caption { }
        .photo__quote { }
        .photo__img--highlighted { }
</style>
```

CSS3 : Modèle de boîte content, padding, border, margin box

Basic Box Model en anglais : un module CSS qui décrit les boîtes rectangulaires avec leur zone de remplissage et leur marge



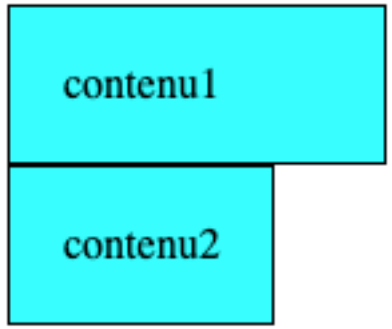

```
<style>
   div {
      margin: 10px;
      padding: 15px;
      border: thin solid black;
      background-color: aqua;
   }
</style>


<div>
   contenu
</div>
```

CSS3 : Modèle de boîte la taille

Par défaut

dimension : width et height

Possibilité d'inclure le padding et le border mais pas la marge.



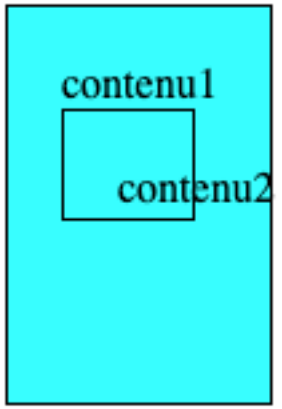
Rem : outline pour un trait autour de la marge

```
<style>
   div {
      width: 100px;
      padding: 20px;
      border: thin solid black;
      background-color: aqua;
      box-sizing: border-box;
   }
   div:nth-child(1) {
      box-sizing: content-box ;
   }
   div:nth-child(2) {
      box-sizing: border-box;
   }
</style>

<div>
   contenu1
</div>
<div>
   contenu2
</div>
```

CSS3 : modèle de boite des boîtes dans des boîtes tout peu déborder

```
<style>
   div {

      padding: 20px;
      border: thin solid black;
      background-color: aqua;
      box-sizing: border-box;
   }
   div#id1 {
      width: 100px;
      height: 150px;
      box-sizing: border-box ;
   }
   div#id2 {
      width: 50px;
      height: 25px;
      box-sizing: border-box;
   }
</style>
```

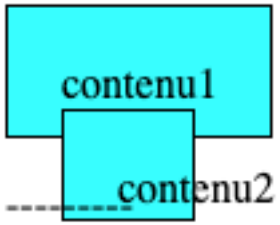
```
<body>
   <div id="id1">
      contenu1
   <div id="id2">
      contenu2
   </div>
   </div>
</body>
```



CSS3 : modèle de boite des boîtes dans des boîtes tout peu déborder

```
<style>
   div {

      padding: 20px;
      border: thin solid black;
      background-color: aqua;
      box-sizing: border-box;
   }
   div#id1 {
      width: 100px;
      height: 50px;
      box-sizing: border-box ;
   }
   div#id2 {
      width: 50px;
      height: 25px;
      box-sizing: border-box;
   }
</style>
```

```
<body>
   <div id="id1">
      contenu1
   <div id="id2">
      contenu2
   </div>
   </div>

<p>---------</p>
</body>
```



=> Laisser les boîtes se dimensionner en fonction de leur contenu et utiliser des pourcentages

CSS3 : modèle de boite la fusion des marges

- Les marges peuvent fusionner

```
<style>
   div {
      margin: 2rem 0;
      background: lavender;
   }

   p {
      margin: .4rem 0 1.2rem 0;
      background: yellow;
   }
</style>
```

```
<p>La marge basse de ce paragraphe est fusionnée…</p>
<p>… avec la marge haute de ce paragraphe. On a donc une marge
   de <code>1.2rem</code> entre les deux.</p>

<div>Cet élément contient deux paragraphes !
   <p>Celui-ci a une marge de <code>.4rem</code> par rapport au texte ci-dessus.</p>
   <p>La marge basse de cet élément fusionne avec la marge basse
      de l'élément parent. On a donc <code>2rem</code> de marge.
   </p>
</div>
```

La marge basse de ce paragraphe est fusionnée…

… avec la marge haute de ce paragraphe. On a donc une marge de `1.2rem` entre les deux.

Cet élément contient deux paragraphes !

Celui-ci a une marge de `.4rem` par rapport au texte ci-dessus.

La marge basse de cet élément fusionne avec la marge basse de l'élément parent. On a donc `2rem` de marge.

Bip bap bop.

CSS3 : modèle de boîte display inline, inline-block, block

La propriété **display** définit le type d'affichage utilisée pour le rendu d'un élément. Nous allons commencer par les valeurs block, inline et inline-block

Dans le flux les block sont les uns sous les autres et occupent la largeur de la page, les inline sont dans la direction en incise et on la largeur du contenu.

Les block sont redimensionnables, les inline ne le sont pas.

Les inline-block sont dans la direction en incise et redimensionnables

CSS3 : modèle de boîte display inline, inline-block, block

```
<style>
    span, div {
        background-color: aqua;
        width: 20%;
    }
   .classe1{
       display: inline-block;
   }
</style>
```

```
<span> inline </span>
<span> inline </span>
<div> block </div>
<div> block </div>
<div class="classe1"> inline-block
</div>
<div class="classe1"> inline-block
</div>
```

CSS3 : Les unités

- **Unités de longueur relatives**
  - Longueurs liées à la police : **em**, **rem**, cap, ch,ex, ic,lh, rlh
  - Longueurs liées au *viewport : vh, vb, vi, vw, vmin, vmax*
- **Unités de longueur absolues**
  - **px**, mm, cm, Q, in, pt,pc

Rem : % par rapport au conteneur, auto, calc

- Une div de 80% de la largeur du body centrée horizontalement

CSS3 : Les unités

- Une div de 80% de la largeur du body centrée horizontalement

```
<style>
    div {
        width: 80%;
        margin-left: auto;
        margin-right: auto;
        background-color: aqua;
    }
</style>
```

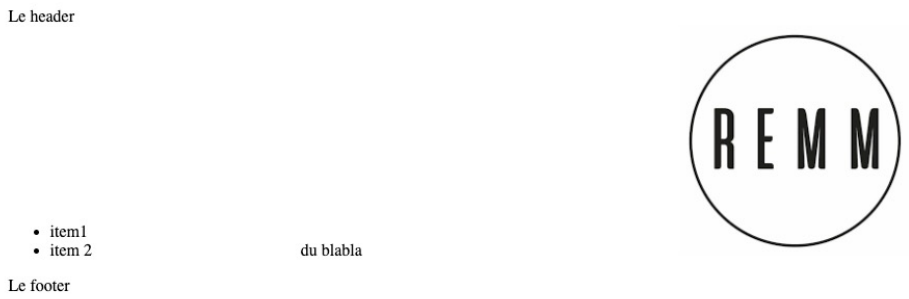
```
<div>
    un peu de contenu
</div>
```

CSS3 : un body qui occupe la hauteur du viewport

Dimensionner html puis le body sinon il s'adaptera au contenu

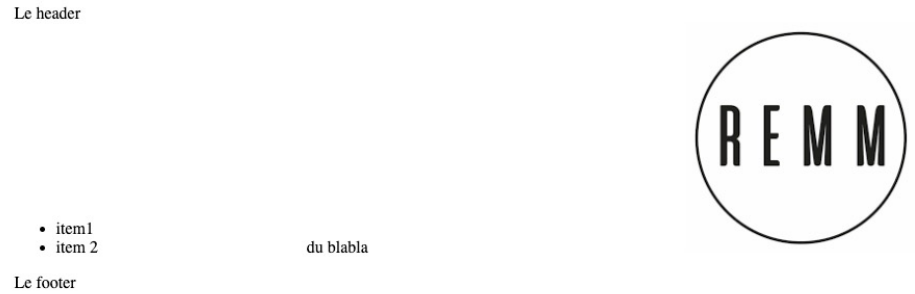
```
<style>
    html {
        height: 100%;
    }
    body {
        height: 100%;
        margin: 0;
        padding: 0;
        background-color: beige;
    }

</style>
```

CSS3: template gabarit 3 colonnes




```
<body>
   <div class="container">
      <div class="header"> Le header </div>
      <div class="content">
         <div class="left">
            <ul>
               <li> item1 </li>
               <li> item 2</li>
            </ul>
         </div>
         <div class="center">
            <p> du blabla </p>
         </div>
         <div class="right">
            <img src="assets/imgs/images.png" alt="remm">
         </div>
      </div>
      <div class="footer"> Le footer</div>
   </div>
</body>
```

CSS3: template gabarit 3 colonnes




```
<body>
  <div class="container">
    <div class="header"> Le header </div>
    <div class="content">
      <div class="left">
        <ul>
          <li> item1 </li>
          <li> item 2</li>
        </ul>
      </div>
      <div class="center">
        <p> du blabla </p>
      </div>
      <div class="right">
        <img src="assets/imgs/images.png" alt="remm">
      </div>
    </div>
    <div class="footer"> Le footer</div>
  </div>
</body>
```

```
.container {
    width: 80%;
    margin-left: auto;  margin-right: auto;
}
.left {
    display: inline-block; width: 30%;
}
.center{
    display: inline-block; width: 39%;
}
.right{
    display: inline-block; width: 30%;
}

.right img {
    width: 80%; height: auto;
}
```

CSS3 : propriétés du texte (openclassrooms)

font-family	*police1, police2, police3*, serif, sans-serif, monospace	Nom de police
@font-face	*Nom et source de la police*	Police personnalisée
font-size	1.3em, 16px, 120%...	Taille du texte
font-weight	bold, normal	Gras
font-style	italic, oblique, normal	Italique
text-decoration	underline, overline, line-through, blink, none	Soulignement, ligne au-dessus, barré ou clignotant
font-variant	small-caps, normal	Petites capitales
text-transform	capitalize, lowercase, uppercase	Capitales
font	-	Super propriété de police. Combine : font-weight, font-style, font-size, font-variant, font-family.
text-align	left, center, right, justify	Alignement horizontal
vertical-align	baseline, middle, sub, super, top, bottom	Alignement vertical (cellules de tableau ou éléments inline-block uniquement)
line-height	18px, 120%, normal...	Hauteur de ligne
text-indent	25px	Alinéa
white-space	pre, nowrap, normal	Césure
word-wrap	break-word, normal	Césure forcée
text-shadow	5px 5px 2px blue *(horizontale, verticale, fondu, couleur)*	Ombre de texte

CSS3 : propriétés du texte (openclassrooms)

Le header

- item1
- item 2

du blabla

Le footer

CSS3 : propriétés du texte (openclassrooms)

Le header

- item1
- item 2

du blabla

Le footer

```
.header, .footer{
    text-align: center;
}

.right, .center {
    vertical-align: top;
}
```

CSS3 : propriétés des listes

list-style-type	disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	Type de liste
list-style-position	inside, outside	Position en retrait
list-style-image	url('puce.png')	Puce personnalisée
list-style	-	Super-propriété de liste. Combine list-style-type, list-style-position, list-style-image.

CSS3 : propriétés de couleur et de fond

color	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur du texte
background-color	*Identique à color*	Couleur de fond
background-image	url('image.png')	Image de fond
background-attachment	fixed, scroll	Fond fixe
background-repeat	repeat-x, repeat-y, no-repeat, repeat	Répétition du fond
background-position	*(x y)*, top, center, bottom, left, right	Position du fond
background	-	Super propriété du fond. Combine : background-image, background-repeat, background-attachment, background-position
opacity	0.5	Transparence

CSS3 : propriétés des boîtes

width	150px, 80%...	Largeur
height	150px, 80%...	Hauteur
min-width	150px, 80%...	Largeur minimale
max-width	150px, 80%...	Largeur maximale
min-height	150px, 80%...	Hauteur minimale
max-height	150px, 80%...	Hauteur maximale
margin-top	23px	Marge en haut
margin-left	23px	Marge à gauche
margin-right	23px	Marge à droite
margin-bottom	23px	Marge en bas
margin	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge. Combine : margin-top, margin-right, margin-bottom, margin-left.
padding-left	23px	Marge intérieure à gauche
padding-right	23px	Marge intérieure à droite
padding-bottom	23px	Marge intérieure en bas
padding-top	23px	Marge intérieure en haut
padding	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge intérieure. Combine : padding-top, padding-right, padding-bottom, padding-left.
border-width	3px	Épaisseur de bordure
border-color	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur de bordure
border-style	solid, dotted, dashed, double, groove, ridge, inset, outset	Type de bordure
border	3px solid black	Super-propriété de bordure. Combine border-width, border-color, border-style. Existe aussi en version border-top, border-right, border-bottom, border-left.
border-radius	5px	Bordure arrondie
box-shadow	6px 6px 0px black *(horizontale, verticale, fondu, couleur)*	Ombre de boîte

CSS3 : propriétés de positionnement et d'affichage

display	block, inline, inline-block, table, table-cell, none...	Type d'élément (block, inline, inline-block, none…)
visibility	visible, hidden	Visibilité
clip	rect (0px, 60px, 30px, 0px) *rect (haut, droite, bas, gauche)*	Affichage d'une partie de l'élément
overflow	auto, scroll, visible, hidden	Comportement en cas de dépassement
float	left, right, none	Flottant
clear	left, right, both, none	Arrêt d'un flottant
position	relative, absolute, static	Positionnement
top	20px	Position par rapport au haut
bottom	20px	Position par rapport au bas
left	20px	Position par rapport à la gauche
right	20px	Position par rapport à la droite
z-index	10	Ordre d'affichage en cas de superposition. La plus grande valeur est affichée par-dessus les autres.

CSS3 : design final

Accueil

item1 +
item2 +

du blabla

REMM

Le footer

CSS3 : Le positionnement flux, flottant, relatif, absolu, fixe

- Notion de flux normal
- Flottant :
  - float
  - float et taille
  - clear
  - Exemples :
    - Comment dimensionner un block
    - Image et texte
    - gabarit
- Un premier système de grille
- Positionnement
  - relatif
  - absolu
  - fixe
  - Collé (sticky)
- Exemples de menu déroulants

CSS3 : positionnement flux normal

Mozilla : Simplifié à l'extrême, un document HTML est un document texte organisé avec des balises. Dans un tel document, le texte « coule » sur les différentes lignes. Autrement dit, le texte est affiché dans le sens de lecture (de gauche à droite pour les langages latins comme le français ou l'italien) et est fragmenté automatiquement pour passer à la ligne à chaque fois que le texte atteint le bord du document.

Chaque élément du document pourra modifier ce flux de texte :

- Certains éléments suivront simplement le flux, comme s'ils n'existaient pas
- Certains éléments pourront forcer le passage à la ligne, que le texte ait atteint la limite du document ou non
- Certains éléments pourront créer un nouveau flux de texte pour leur contenu, flux indépendant du flux de texte « extérieur ».

CSS3 : positionnement flux normal

Nous avons vu **display** :

none : Cette valeur retire l'élément du flux, comme si l'élément et son contenu n'existaient pas.

inline : Cette valeur rend l'élément transparent au sens où il s'inscrit dans le flux de texte global, il est donc associé au texte l'environnant.

Block : Cette valeur cassera le flux de texte pour insérer l'élément. Cela provoquera donc un saut de ligne avant et après. Le contenu de cet élément ne fait donc pas partie du flux global et suit donc les contraintes de l'élément définies par le modèle de boîte.

inline-block : Cette valeur est en quelque sorte un intermédiaire entre inline et block. Comme avec inline, les boîtes seront placées dans le flux global mais , comme avec block, le contenu ne fera pas partie du texte environnant..

CSS3 : positionnement flux normal

Nous avons vu comment modifier le flux pour le texte :

Ces propriétés sont : hyphens, text-align, text-align-last, text-indent, vertical-align, white-space, word-break et word-wrap.

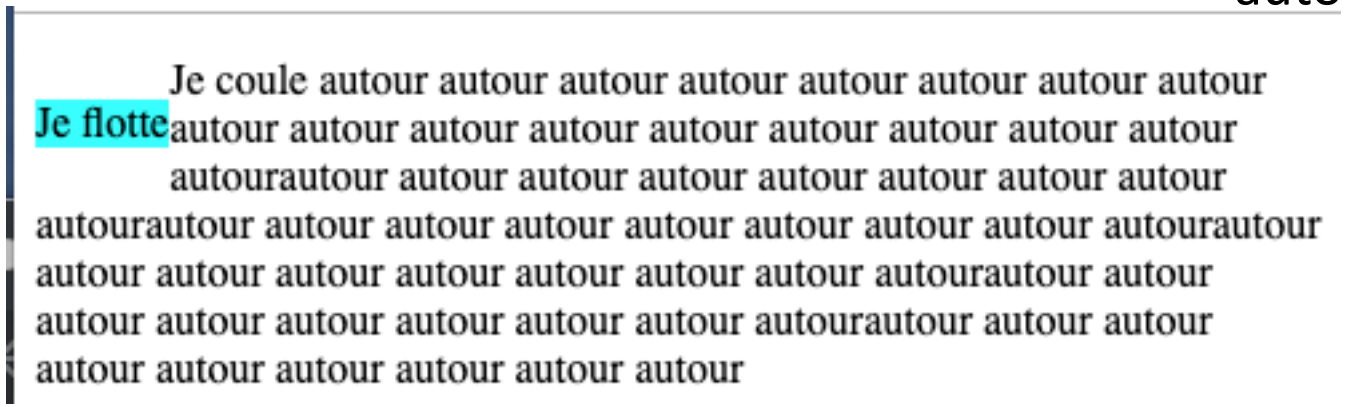
Nous allons maintenant utiliser des boîte toujours attachées au flux mais dont le texte va couler autour

CSS3 : float la base

```
<style>
   #id1{
      display: block;
      float:left;
      background-color: aqua;
   }
</style>
```

Rem : float: left ou right

```
<p id="id1"> Je flotte </p>
<p> Je coule autour autour autour autour
autour autour autour autour autour autour
autour autour autour autour autour autour
autour autourautour autour autour autour
autour autour autour autour autourautour
autour autour autour autour autour autour
autour autourautour autour autour autour
autour autour autour autour autourautour
autour autour autour autour autour autour
autour autourautour autour autour autour
autour autour autour autour autour</p>
```



CSS3 : float et taille

- La hauteur d'une div ne comprend pas la hauteur des flottants qui la composent

Exemple ok :


```
<style>
   #id1,#id2{
      display: block;
      background-color: aqua;
      margin: 0;
   }
   #id1{float: left}
   #id2{float: right}
   div{
      background-color: blue;
      margin: 0;
   }
</style>

<div>
   <p id="id1"> Je flotte </p>
   <p id="id2"> Je flotte </p>
   <p> Je coule </p>

</div>
```

CSS3 : float et taille

- La hauteur d'une div ne comprend pas la hauteur des flottants qui la composent
- Exemple flottants trop grands


```
<style>
   #id1,#id2{
      display: block;
      background-color: aqua;
}
   #id1{float: left}
   #id2{float: right}
   div{
      background-color: blue;
      margin: 0;
   }
</style>

<div>
   <p id="id1"> Je flotte </p>
   <p id="id2"> Je flotte </p>
   <p> Je coule </p>

</div>
```

CSS3 : float

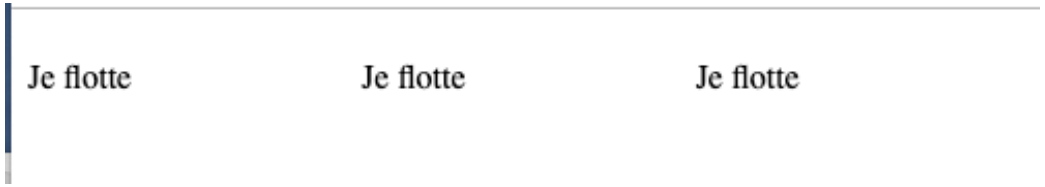
- La propriété **clear** indique si un élément peut être situé à côté d'éléments flottants qui le précèdent ou s'il doit être déplacé vers le bas pour être en dessous de ces éléments. La propriété clear s'applique aux éléments flottants comme aux éléments non-flottants.

- Valeurs: none, right, both, left

```
<style>
   #id1,#id2{
      display: block;
      background-color: aqua;

   }
   #id1{float: left}
   #id2{float: right}
   div{
      background-color: blue;
      margin: 0;
   }
   #id3{ clear:both }
</style>


<div>
   <p id="id1"> Je flotte </p>
   <p id="id2"> Je flotte </p>
   <p id="id3"> Je coule </p>

</div>
```

CSS3 : float comment dimensionner un block avec des flottants

- En utilisant un pseudo élément



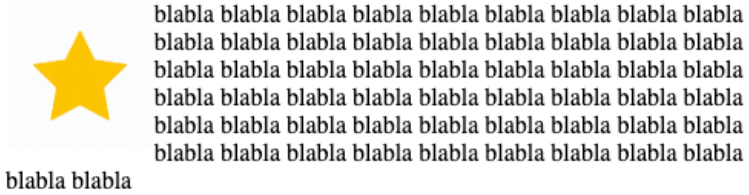

```
<style>
   div {
      background-color: blue;
   }
  p {width : 33.33%;
  float: left}
</style>


<div>
   <p id="id1"> Je flotte </p>
   <p id="id2"> Je flotte </p>
   <p id="id3"> Je flotte </p>
</div>
```

CSS3 : float comment dimensionner un block avec des flottants

- En utilisant un pseudo élément


```
<style>
   div {
      background-color: blue;
   }
  p {width : 33.33%;
  float: left}
</style>

div::after{
   content:'';
   display: block;
   clear: both;
}

<div>
   <p id="id1"> Je flotte </p>
   <p id="id2"> Je flotte </p>
   <p id="id3"> Je flotte </p>
</div>
```

CSS3 Float : image et texte




```
<style>
img{
    float: left;
    width: 20%;
}
</style>
```

```
<p> <img
src="assets/imgs/star.png">blabl
a  blabla blabla … </p>
```

CSS3 Float : gabarit 1

Accueil

item1 +
item2 +

du blabla

REMM

Le footer

CSS3 : Float un premier modèle de grille

- Les col doivent être dans des grid

```
*, *:after, *:before {
  box-sizing: border-box;
}

[class*='col-'] {
   float: left;
}

.grid:after {
   content: "";
   display: block;
   clear: both;
}

[class*='col-'] {
   padding-right: 20px;
}
[class*='col-']:last-of-type {
   padding-right: 0;
}

.grid-pad {
   padding: 20px 0 20px 20px;
}

.grid-pad > [class*='col-']:last-of-type {
   padding-right: 20px;
}

.col-1 {
   width: 8.33%;
}
.col-2 {
   width: 16.66%;
}
...

.col-12 {
   width: 100%;
}
```

CSS3 : position

position: static | relative | absolute | fixed | sticky

Un **élément positionné** est un élément dont la propriété de
position calculée est relative, absolute, fixed ou sticky
.

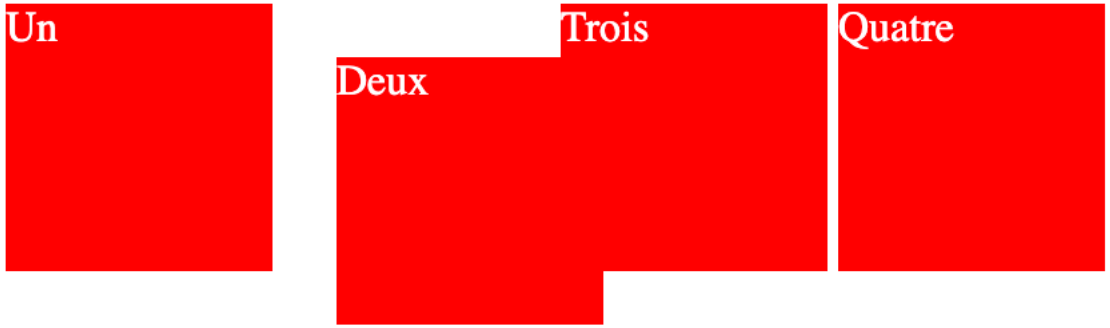
Un **élément positionné de façon relative** est un élément dont la propriété de position calculée est relative. Dans ce cas, les propriétés top ou bottom indiquent le décalage vertical à appliquer et left ou right indiquent le décalage horizontal.

Un **élément positionné de façon absolue** est un élément dont la propriété de position calculée est absolute ou fixed. Dans ce cas, les propriétés top, bottom, right et left indiquent les distances entre les bords de l'élément et les bords du bloc englobant (c'est-à-dire l'ancêtre par rapport auquel l'élément est positionné). Si l'élément possède des marges, elles sont ajoutées aux décalages.

Un **élément positionné en adhérence** est un élément dont la propriété de position calculée vaut sticky. Un tel élément se comporte comme un élément positionné de façon relative jusqu'à ce que son bloc englobant dépasse un seuil donné (par exemple fourni par la valeur de top) au sein du conteneur puis il se comporte ensuite comme un élément fixe jusqu'à atteindre le bord opposé du bloc englobant.

CSS3 : position relative

- L'espace dans le flux est conservé, la boite est positionnée

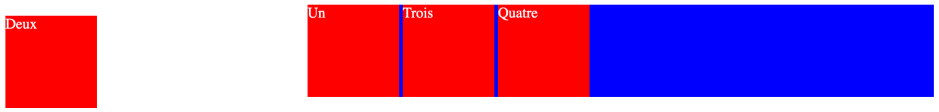



```
<style>
   .box {
      display: inline-block;
      background: red;
      width: 100px;
      height: 100px;
      color: white;
   }

   #deux {
      position: relative;
      top: 20px;
      left: 20px;
   }
</style>

<div class="box" id="un">Un</div>
<div class="box" id="deux">Deux</div>
<div class="box" id="trois">Trois</div>
<div class="box" id="quatre">Quatre</div>
```

CSS3 : position absolute 1

Position calculée à partir du dernier élément positionné, non pris en compte dans le flux

```
<div class="container">
    <div class="box" id="un">Un</div>
    <div class="box" id="deux">Deux</div>
    <div class="box" id="trois">Trois</div>
    <div class="box" id="quatre">Quatre</div>
</div>
```




```
<style>
    .container {
        margin: 0 auto;
        width: 50%;
        background-color: blue;
    }
    .box {
        display: inline-block;
        background: red;
        width: 100px;
        height: 100px;
        color: white;
    }

    #deux {
        position: absolute;
        top: 20px;
        left: 20px;
    }
</style>
```

CSS3 : position absolute 2

Position calculée à partir du dernier élément positionné, non pris en compte dans le flux

```
<div class="container">
   <div class="box" id="un">Un</div>
   <div class="box" id="deux">Deux</div>
   <div class="box" id="trois">Trois</div>
   <div class="box" id="quatre">Quatre</div>
</div>
```


```
<style>
   .container {
      margin: 0 auto;
      width: 50%;
      background-color: blue;

      position: relative;
   }
   .box {
      display: inline-block;
      background: red;
      width: 100px;
      height: 100px;
      color: white;
   }

   #deux {
      position: absolute;
      top: 20px;
      left: 20px;
   }
</style>
```

CSS3 : fixed

Le positionnement fixe est semblable au positionnement absolu sauf qu'ici, le bloc englobant correspond au *viewport* si aucun ancêtre de l'élément ne possède une propriété transform, perspective o u filter qui est différente de none.

On utilise souvent ce positionnement pour créer un élément flottant qui reste à la même position, même lorsqu'on fait défiler la page. Dans l'exemple qui suit, la boîte « Un » est fixée à 80 pixels du haut de la page et à 20 pixels du bord gauche.

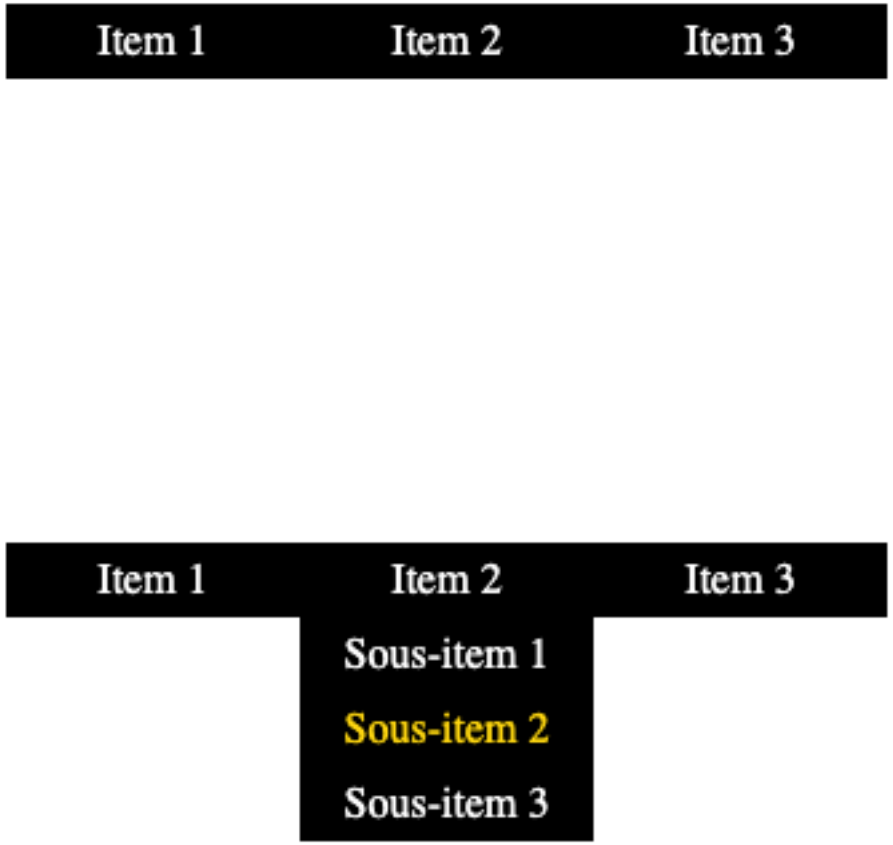
Rem : consommateur en CPU

CSS3 : position sticky

Le positionnement adhérent est un mélange de positionnement relatif et de positionnement fixe. L'élément est considéré comme positionné de façon relative, jusqu'à ce qu'un seuil soit franchi. À partir de ce seuil, l'élément est positionné de façon fixe.

CSS3 : menu horizontal déroulant

```
<div id="menu">
   <ul>
      <li><a href="#">Item 1</a></li>
      <li><a href="#">Item 2</a>
         <ul>
            <li><a href="#">Sous-item 1</a></li>
            <li><a href="#">Sous-item 2</a></li>
            <li><a href="#">Sous-item 3</a></li>
         </ul>
      </li>
      <li><a href="#">Item 3</a></li>
   </ul>
</div>
```



CSS3 : menu horizontal déroulant flottant

```
#menu {
   height:50px;
}
#menu ul {
   margin:0;
   padding:0;
   list-style-type:none;
   text-align:center;
}
#menu li {
   float:left;
   margin:auto;
   padding:0;
   background-color:black;
}
#menu li a {
   display:block;
   width:100px;
   color:white;
   text-decoration:none;
   padding:5px;
}
```

```
#menu li a:hover {
   color:#FFD700;
}
#menu ul li ul {
   display:none;
}
#menu ul li:hover ul {
   display:block;
}
#menu li:hover ul li {
   float:none;
}
```

CSS3 : menu horizontal déroulant inline-block

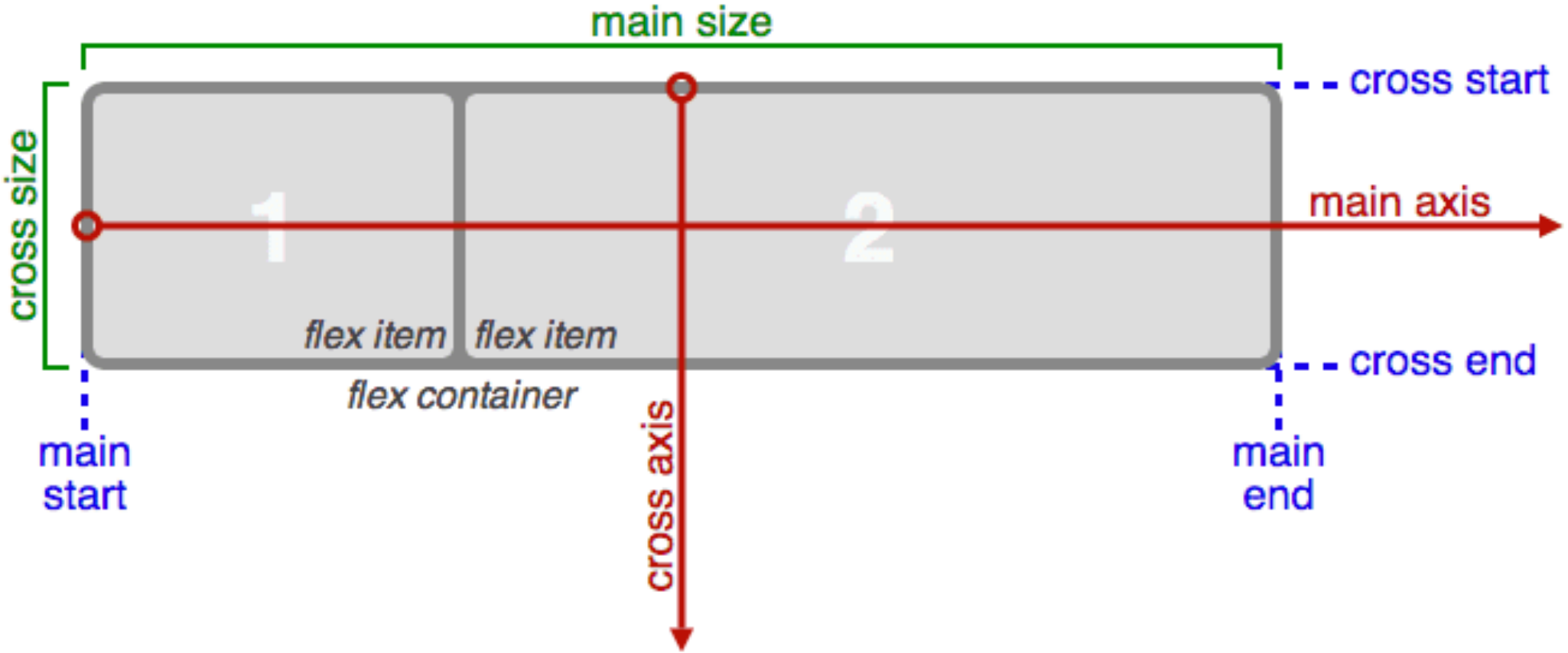
```
#menu {
   height:50px;
}
#menu ul {
   margin:0;
   padding:0;
   list-style-type:none;

}
#menu li {
   display: inline-block;
   text-align:center;
   margin:auto;
   padding:0;
   background-color:black;
}
#menu li a {
   display:block;
   width:100px;
   color:white;
   text-decoration:none;
   padding:5px;
}
```

```
#menu li a:hover {
   color:#FFD700;
}
#menu ul li ul {
   display:none;
   position: absolute;
}
#menu ul li:hover ul {
   display:block;
}
#menu li:hover ul li {
   display: block;
}
```

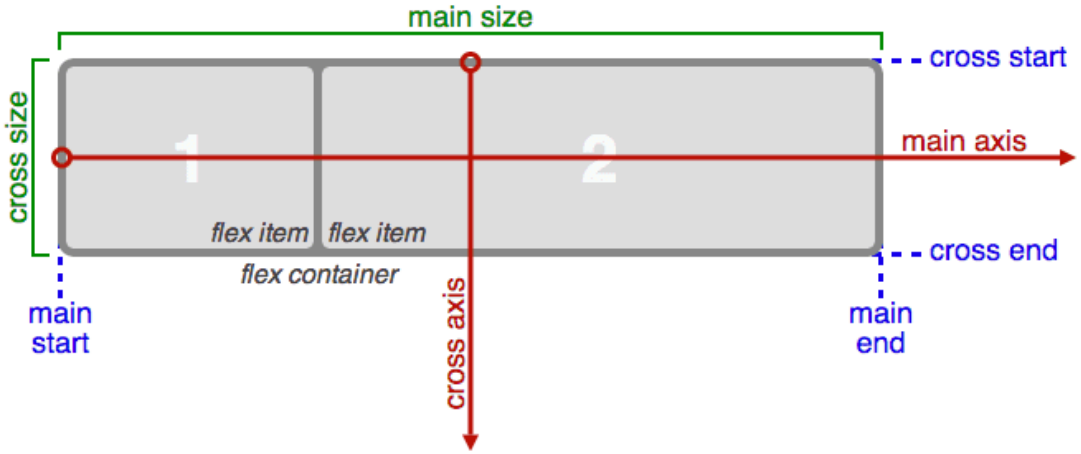
CSS3 : FlexBox et Grid CSS

Un changement de philosophie, la disposition est choisie au niveau du conteneur et non de l’élément.

CSS3 : FlexBox



CSS3 FlexBox



- **main axis** - L'axe principal d'un container flex est l'axe primaire sur lequel les items sont disposés. Attention, il n'est pas forcément horizontal, tout dépendra de la propriété justify-content (voir ci-dessous).
- **main-start | main-end** - À l'intérieur du container, les items flex sont placés entre un point de départ (main-start) et un point d'arrivée (main-end).
- **main size** - La taille d'un item flex qui se trouve dans l'axe principal, qu'il s'agisse de la hauteur ou de la largeur, est la taille principale (main size). La propriété main size est soit "width", soit "height", selon l'orientation.
- **cross axis** - L'axe perpendiculaire à l'axe principal est appelé cross axis. Sa direction dépend de la direction de l'axe principal.
- **cross-start | cross-end** - Les lignes sont remplies avec les items et sont placées dans le container en partant du côté cross-start< et en allant vers cross-end.
- **cross-size** - La largeur ou la hauteur d'un item, selon la dimension dans laquelle on se trouve (même principe que main size).

Source la cascade

CSS3 : FlexBox container

- **display**: flex | inline-flex;
- **flex-direction**: row | row-reverse | column | column-reverse
- **flex-wrap**: nowrap | wrap | wrap-reverse
- **flex-flow**: <‘flex-direction’> || <‘flex-wrap’>
- **justify-content**: flex-start | flex-end | center | space-between | space-around
- **align-items**: flex-start | flex-end | center | baseline | stretch
- **align-content**: flex-start | flex-end | center | space-between | space-around | stretch

CSS3 FlexBox item

- flex-grow: <nombre entier> (par défaut = 0)
- flex-shrink: <nombre entier> (par défaut = 1)
- flex-basis: <longueur> | auto (par défaut = auto)
- flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]
- order: <nombre entier>
- align-self: auto | flex-start | flex-end | center | baseline | stretch

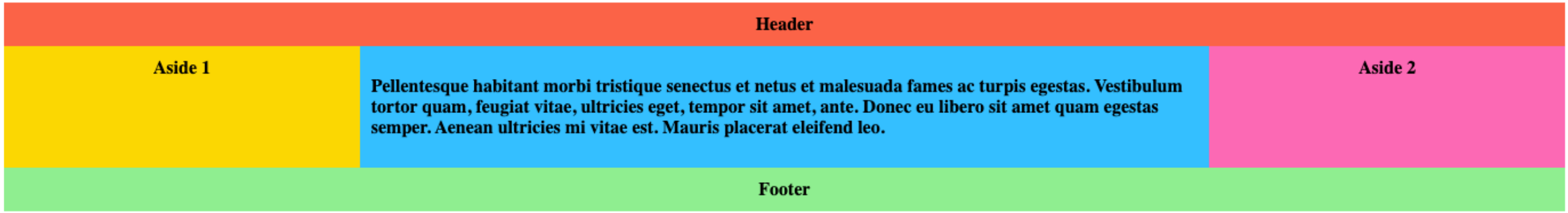
CSS3 : Flex jouons avec les grenouilles

- https://flexboxfroggy.com/#fr

- Les grenouilles sont des items flex

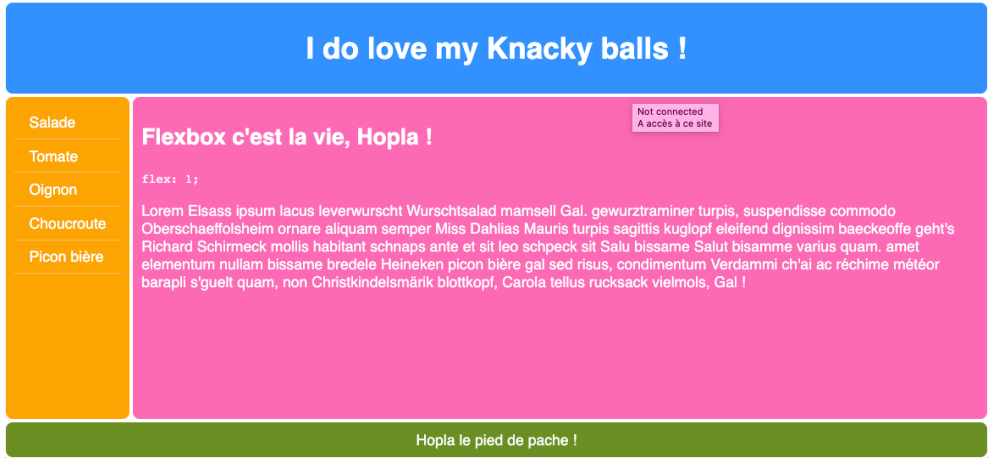
- Les nénuphars des images de fond

CSS3 : FlexBox un premier gabarit




```
<div class="wrapper">
   <header class="header">Header</header>
   <article class="main">
      <p>Pellentesque </p>
   </article>
   <aside class="aside aside-1">Aside 1</aside>
   <aside class="aside aside-2">Aside 2</aside>
   <footer class="footer">Footer</footer>
</div>
```

CSS3 : FlexBox un deuxième gabrit




```
<header>
  <h1>I do love my Knacky balls !</h1>
</header>

<div class="wrapper">
  <nav id="navigation" role="navigation">
    <a href="#">Salade</a>
    <a href="#">Picon bière</a>
  </nav>
  <section class="content">
    <h2>Flexbox c'est la vie, Hopla !</h2>
    <p><code>flex: 1;</code></p>
    <p>Lorem </p>
  </section>
</div> <!-- /wrapper -->

<footer>Hopla le pied de pache !</footer>
```

CSS3 : CSS Grid

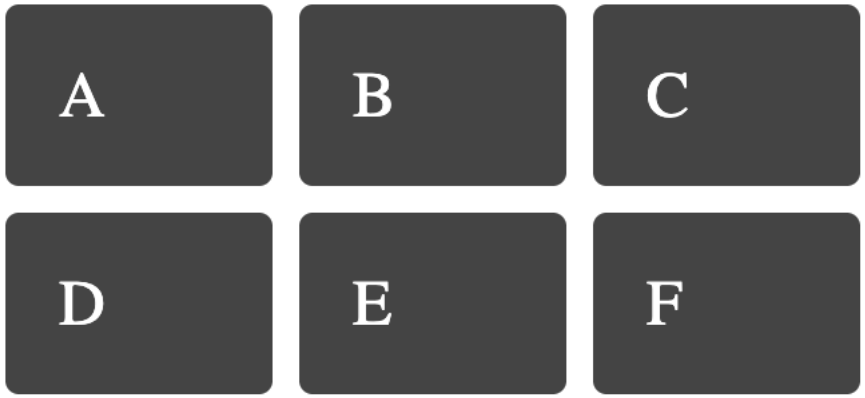
- Un modèle en grille avec un type de données flex
- display : grid ou inline-grid
- Des propriétés
- Des fonctions

CSS3 : Grid CSS Exemple 1 grid-template-columns

```html
<div class="wrapper">
  <div class="box a">A</div>
  <div class="box b">B</div>
  <div class="box c">C</div>
  <div class="box d">D</div>
  <div class="box e">E</div>
  <div class="box f">F</div>
</div>
```

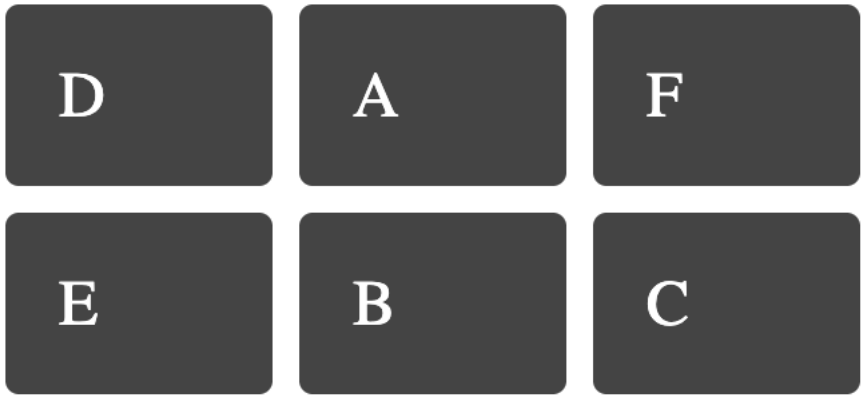
```css
.wrapper {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-gap: 10px;
}
```

Source : https://gridbyexample.com/examples/



CSS3 : Grid CSS Exemple 2 grid-column/grid-row

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
   <div class="box e">E</div>
   <div class="box f">F</div>
</div>
```



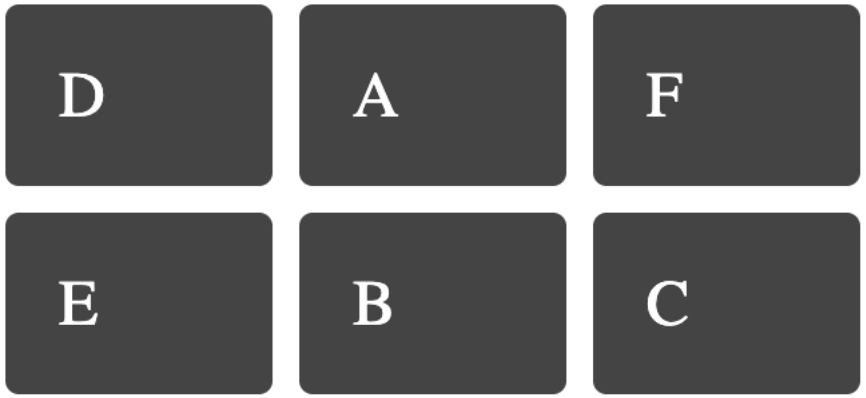

```css
.wrapper {
   display: grid;
   grid-template-columns: 100px 100px 100px;
   grid-gap: 10px;
}
 .a {
    grid-column-start: 2;
    grid-column-end: 3;
    grid-row-start: 1;
    grid-row-end: 2;
 }
 .b {
    grid-column-start: 2;
    grid-column-end: 3;
    grid-row-start: 2;
    grid-row-end: 3;
 }
```

```css
.c {
   grid-column-start: 3;
   grid-column-end: 4;
   grid-row-start: 2;
   grid-row-end: 3;
}
.d {
   grid-column-start: 1;
   grid-column-end: 2;
   grid-row-start: 1;
   grid-row-end: 2;
}
```

```css
.e {
   grid-column-start: 1;
   grid-column-end: 2;
   grid-row-start: 2;
   grid-row-end: 3;
}
.f {
   grid-column-start: 3;
   grid-column-end: 4;
   grid-row-start: 1;
   grid-row-end: 2;
}
```

CSS3 : Grid CSS Exemple 2bis grid-column/ grid-row

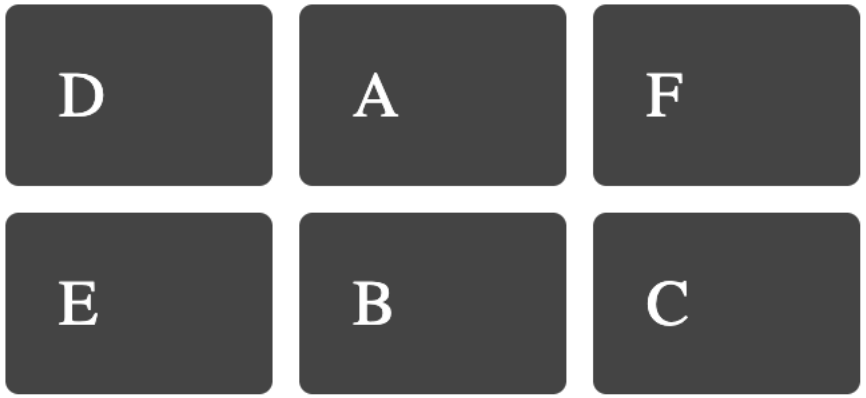
```
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
   <div class="box e">E</div>
   <div class="box f">F</div>
</div>
```




```
.wrapper {
   display: grid;
   grid-template-columns: 100px 100px 100px;
   grid-gap: 10px;
}

   .a {
      grid-column: 2 / 3;
      grid-row: 1 / 2;
   }
   .b {
      grid-column: 2 / 3;
      grid-row: 2 / 3;
   }
   .c {
      grid-column: 3 / 4;
      grid-row: 2 / 3;
   }
   .d {
      grid-column: 1 / 2;
      grid-row: 1 / 2;
   }
   .e {
      grid-column: 1 / 2;
      grid-row: 2 / 3;
   }
   .f {
      grid-column: 3 / 4;
      grid-row: 1 / 2;
   }
```

CSS3 : Grid CSS Exemple 3 grid-area

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
   <div class="box e">E</div>
   <div class="box f">F</div>
</div>
```

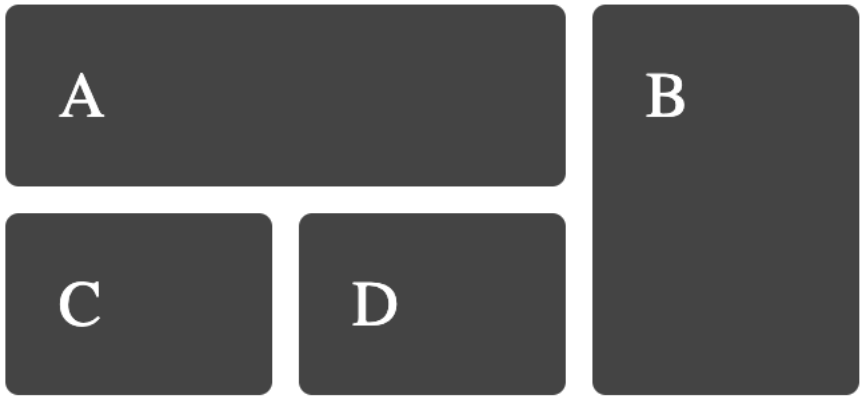



```css
.wrapper {
   display: grid;
   grid-template-columns: 100px 100px 100px;
   grid-gap: 10px;
}

   .a {
      grid-area: 1 / 2 / 2 / 3;
   }
   .b {
      grid-area: 2 / 2 / 3 / 3;
   }
   .c {
      grid-area: 2 / 3 / 3 / 4;
   }
   .d {
      grid-area: 1 / 1 / 2 / 2;
   }
   .e {
      grid-area: 2 / 1 / 3 / 2;
   }
   .f {
      grid-area: 1 / 3 / 2 / 4;
   }
```

CSS3 : Grid CSS Exemple 4 grid-row

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
</div>
```

```css
.wrapper {
   display: grid;
   grid-template-columns: 100px 100px 100px;
   grid-gap: 10px;
}
```



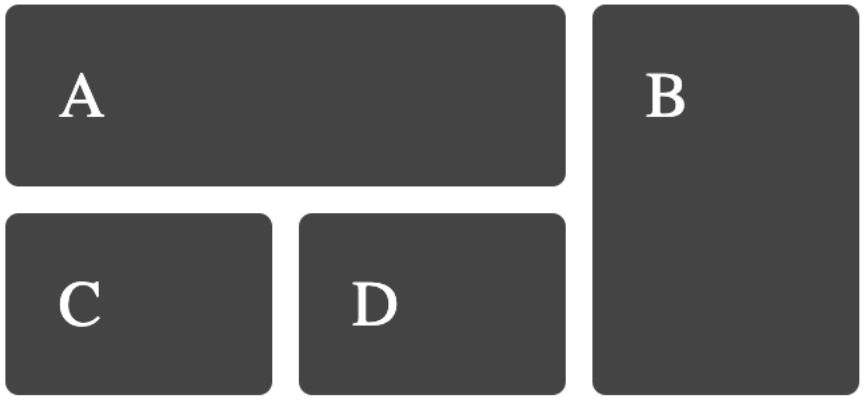

```css
.a {
   grid-column: 1 / 3;
   grid-row: 1;
}
.b {
   grid-column: 3 ;
   grid-row: 1 / 3;
}
```

```css
.c {
   grid-column: 1 ;
   grid-row: 2 ;
}
.d {
   grid-column: 2;
   grid-row: 2;
}
```

CSS3 : Grid CSS Exemple 5 span

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
</div>
```

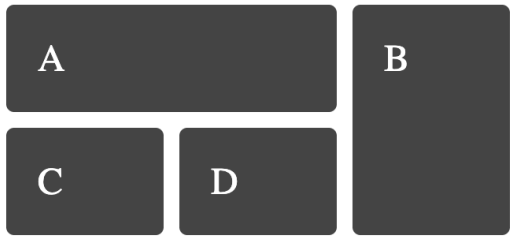
```css
.wrapper {
   display: grid;
   grid-template-columns: 100px 100px 100px;
   grid-gap: 10px;
}
```




```css
.a {
   grid-column: 1 / span 2;
}
.b {
   grid-column: 3 ;
   grid-row: 1 / span 2;
}
```

```css
.c {
   grid-column: 1 ;
   grid-row: 2 ;
}
.d {
   grid-column: 2 ;
   grid-row: 2 ;
}
```

CSS3 : Grid CSS Exemple 6 nommage




```
.wrapper {
   display: grid;
   grid-template-columns: [col1-start] 100px  [col2-start] 100px  [col3-start] 100px [col3-end];
   grid-template-rows: [row1-start] auto [row2-start] auto [row2-end];
   grid-gap: 10px;
}
```

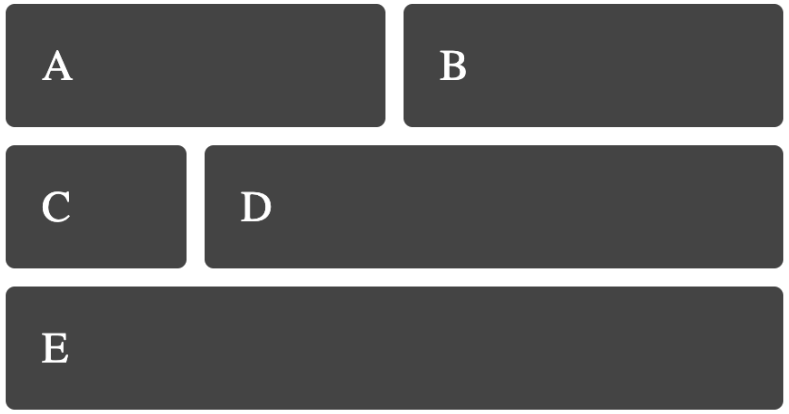
```
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
</div>
```

```
.a {
   grid-column: col1-start / col3-start;
   grid-row: row1-start ;
}
.b {
   grid-column: col3-start ;
   grid-row: row1-start / row2-end;
}
```

```
.c {
   grid-column: col1-start;
   grid-row: row2-start ;
}
.d {
   grid-column: col2-start ;
   grid-row: row2-start ;
}
```

CSS3 : Grid CSS Exemple 7 reapeat

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
   <div class="box e">E</div>
</div>
```




```css
.wrapper {
   display: grid;
   grid-template-columns: repeat(4, [col] 100px ) ;
   grid-template-rows: repeat(3, [row] auto  );
   grid-gap: 10px;
}

.a {
   grid-column: col / span 2;
   grid-row: row ;
}
.b {
   grid-column: col 3 / span 2 ;
   grid-row: row ;
}

.c {
   grid-column: col ;
   grid-row: row 2 ;
}
.d {
   grid-column: col 2 / span 3 ;
   grid-row: row 2 ;
}

.e {
   grid-column: col / span 4;
   grid-row: row 3;
}
```

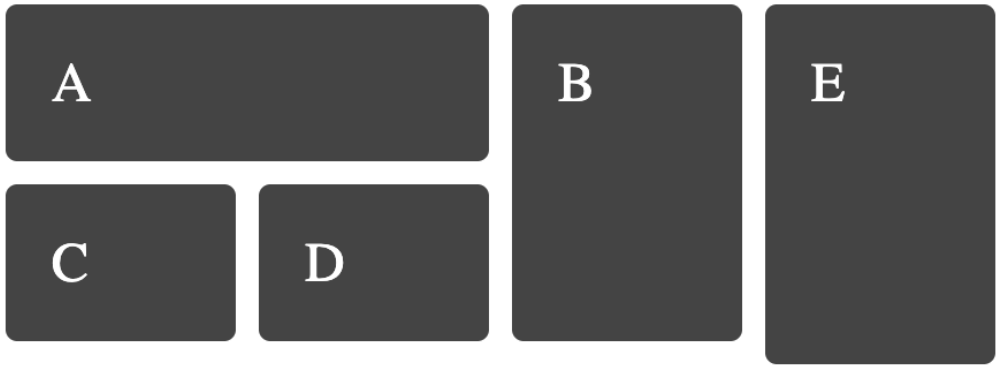
CSS3 : Grid CSS Exemple 8 grille implicite

```html
<div class="wrapper">
   <div class="box a">A</div>
   <div class="box b">B</div>
   <div class="box c">C</div>
   <div class="box d">D</div>
   <div class="box e">E</div>
</div>
```

```css
.wrapper {
   display: grid;
   grid-gap: 10px;
   grid-template-columns: 100px 100px 100px;
   grid-auto-columns: 100px;

}
```

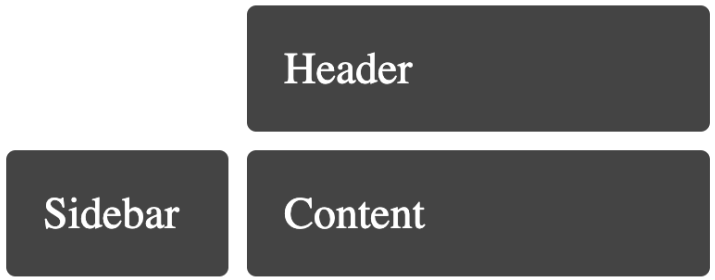
```css
.a {
   grid-column: 1 / 3;
   grid-row: 1;
}
.b {
   grid-column: 3 ;
   grid-row: 1 / 3;
}
.c {
   grid-column: 1 ;
   grid-row: 2 ;
}
```

```css
.d {
   grid-column: 2;
   grid-row: 2;
}

.e {
   grid-column: 4 / 5;
   grid-row: 1 / 4;
}
```



CSS3 : Grid CSS Exemple 9 grille template area

```html
<div class="wrapper">
   <div class="box header">Header</div>
   <div class="box sidebar">Sidebar</div>
   <div class="box content">Content</div>
</div>
```




```css
.wrapper {
   display: grid;
   grid-gap: 10px;
   grid-template-columns: 120px 120px 120px;
   grid-template-areas: "..... header  header"
                        "sidebar content content";
}

.sidebar {
   grid-area: sidebar;
}

.content {
   grid-area: content;
}

.header {
   grid-area: header;
}
```

CSS 3 : Grid CSS

D’autres propriétés ici :
https://developer.mozilla.org/fr/docs/Web/CSS/grid-template

CSS 3 : FIN

- Tansition, animation, transformation
- Z-index
- Compteur et variables

- Les média queries
- Les CSS compilées : sass et bootstrap

CSS3 : transitions

- Permettent de définir des transitions entre des valeurs de propriétés CSS

transition-delay : le délai avant le début de la transition

transition-duration : la durée

transition-property : la ou les propriétés modifiées

transition-timing-function : fonction de transition

CSS3 : transition exemple

```
<style>
   div {
      width: 100px;
      height: 100px;
      background: red;
      transition-property: width;
      transition-duration: 2s;
      transition-timing-function: linear;
      transition-delay: 1s;
   }

   div:hover {
      width: 300px;
   }

</style>
```

CSS3 : Les animations

Les animations sont des transitions entre deux états de mise en forme.

Une animation possède :

- Des propriétés propres
- Un ensemble d'étapes de l'état initial à l'état final (*keyframes)*

CSS3 : animations paramètres

animation-delay : délai entre chargement et début

animation-direction : alternance entre deux directions

animation-duration : la durée d’un cycle

animation-fill-mode : indique les valeurs qui doivent être appliquées aux propriétés avant et après l'exécution de l'animation.

animation-iteration-count : nombre de cycle (peut être infini)

animation-name : déclaration d’un nom qui pourra être utilisé comme référence à l'animation pour la règle @keyframes.

animation-play-state : pause ou reprise d’une animation

animation-timing-function : le minutage d’une animation

CSS 3 : animations keyframe

```
p {
   animation-duration: 3s;
   animation-name: slidein;
}

@keyframes slidein {
   from {
      margin-left: 100%;
      width: 300%;
   }

   to {
      margin-left: 0%;
      width: 100%;
   }
}
```

```
p {
   animation-duration: 3s;
   animation-name: slidein;
}

@keyframes slidein {
   5% {
      margin-left: 100%;
      width: 300%;
   }

   10% {
      margin-left: 0%;
      width: 100%;
   }

   100%{
      margin-left: 100%;
      width: 300%;
   }
}
```

CSS3 : transformation

La possibilité de réaliser des transformations du plan

Transform :

none | <transform-list>

où

<transform-list> = <transform-function>+

où

<transform-function> = <matrix()> | <translate()> | <translateX()> | <translateY()> | <scale()> | <scaleX()> | <scaleY()> | <rotate()> | <skew()> | <skewX()> | <skewY()> | <matrix3d()> | <translate3d()> | <translateZ()> | <scale3d()> | <scaleZ()> | <rotate3d()> | <rotateX()> | <rotateY()> | <rotateZ()> | <perspective()>

CSS3 : transformation exemple

```
p {
   border: solid red;
   transform: translate(100px) rotate(20deg);
   transform-origin: 0 -250px;
}
```

CSS3 : animation + transformation

```
.h1 {
    color: darkturquoise;
    font-family: lato;
    font-weight: 500;
    font-size: 60px;
    text-align: center;
    text-transform: uppercase;
    animation: bounceIn 1.5s both 1s;
    margin-top: 200px;
}
```

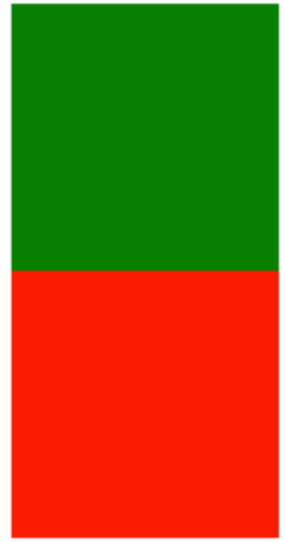
```
@keyframes bounceIn{
    0% {
        transform: scale(0.1);
        opacity: 0;
    }

    60% {
        transform: scale(1.2);
        opacity: 1;
    }

    100% {
        transform: scale(1);
        }
}
```

CSS3 : z-index

La propriété **z-index** définit le « *z-order* » d'un élément positionné et de ses éléments fils ou de ses éléments flexibles (les enfants d'un élément avec display: flex). Lorsque des éléments se chevauchent, le *z-order* détermine l'ordre des différentes couches que formeront les éléments. Généralement, un élément couvrira un autre élément si sa valeur de z-index est supérieure à celle du deuxième élément.

CSS3: z-index

```
<style>
   div {height: 100px; width: 100px; }
   div#test1 {position: absolute; z-index: 4; background-color:
green}
   div#test2 {position: absolute; z-index: 2; background-color:
grey}
   div#test3 {z-index: 10; background-color: aqua}
   div#test4 {position: relative; z-index: 8; background-color:
red}
</style>
```



```
 <div id="test1"></div>
 <div id="test2"></div>
 <div id="test3"></div>
 <div id="test4"></div>
```

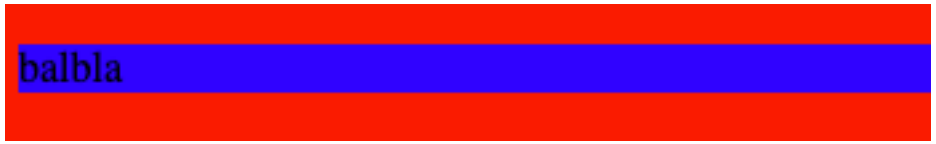
CSS3: variables

Les propriétés personnalisées CSS (*custom properties* en anglais, aussi parfois appelés **variables CSS**) sont des entités définies par les développeurs ou les utilisateurs d'une page Web, contenant des valeurs spécifiques utilisables à travers le document. Elles sont initialisées avec des propriétés personnalisées (par exemple **--main-color: black;**) et accessibles en utilisant la notation spécifique var()

- Une variable ne peut-être définie en fonction d'elle même
- C'est un langage déclaratif => il n'y a pas de notion de séquence
- Une variable dans un élément fils est redéfinie => la valeur du père ne change pas

CSS3 : variable

```
:root {
   --bg-color: red;
}

body {
   background-color: var(--bg-color);
}

div {
   --bg-color: blue;
}
p {
   background-color: var(--bg-color);
}
```




```
<div>
   <p> balbla </p>
</div>
```

CSS3 : compteurs

- Les **compteurs CSS** sont des variables dont les valeurs sont incrémentées par les règles CSS et qui permettent de savoir combien de fois elles sont utilisées.

Section 1 : Introduction

Section 2 : Corps

Section 3 : Conclusion

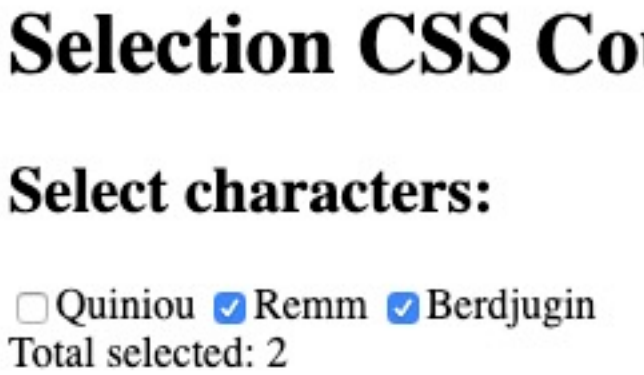
```
body {
   counter-reset: section;              /* On initialise le compteur à 0 */
}

h3::before {
   counter-increment: section;          /* On incrémente le compteur section */
   content: "Section " counter(section) " : "; /* On affiche le compteur */
}
```

```
<h3>Introduction</h3>
<h3>Corps</h3>
<h3>Conclusion</h3>
```

CSS3 : compteurs

```
<section class="characters">
   <h2>Select characters:</h2>
   <input id="q" type="checkbox"><label
for="q">Quiniou</label>
   <input id="r" type="checkbox"><label
for="r">Remm</label>
   <input id="b" type="checkbox"><label
for="b">Berdjugin</label>

</section>
```

```
body {
   counter-reset: characters;
}

input:checked {
   counter-increment: characters;
}

.total::after {
   content: counter(characters);
}
```



CSS3 : media queries

Les requêtes média (*media queries*) permettent de modifier l'apparence d'un site ou d'une application en fonction du type d'appareil (impression ou écran par exemple) et de ses caractéristiques (la résolution d'écran ou la largeur de la zone d'affichage (*viewport*) par exemple).

Les Media Queries permettent de cibler :

- Le type de média
- La taille de l'écran
- La taille de la fenêtre
- La résolution
- Le nombre de couleurs
- L'orientation
- ...

Exemple : @media all and (**max-width**: 1024**px**)
{ …}

CSS compilée

Un préprocesseur CSS est un outil (ou programme) informatique permettant de générer dynamiquement des fichiers CSS. L'objectif est d'améliorer l'écriture de ces fichiers, en apportant plus de flexibilité au développeur web.

- Exemple: less, sass, stylus, …

Exemple d’utilisation de sass

```
$mycolor : rgb(random(255), random(255), random(255));
ul {
        list-style: none;
        li {
        display: inline;
        &:hover {
                color: $mycolor;
                }
        }
}
```

```
ul {
  list-style: none;
}
ul li {
  display: inline;
}
ul li:hover {
  color: #ae5ff7;
}
```

Sass

- Des imbrications
- Des variable et des types
- Des structures de contrôle
- Des fonctions
- Des moyens de factoriser le code :
  - Inclusions
  - Mixins
  - Héritage

…

Dans la pratique

Des framework souvent compilés :

- Bootstrap
- Materialize
- Material UI
- Foundation
- Semantic UI
- KNACSS

…