LA GENÈSE

CSS (Cascading Style Sheets), c'est cet autre langage qui vient compléter le HTML.

Vous vous souvenez de son rôle ? Gérer la mise en forme de votre site.

Petit rappel : à quoi sert CSS ?

Il permet de styliser votre contenu

Le HTML est né en 1991, et CSS en 1996. Alors, vous vous dites sûrement : comment faisait-on la mise en forme, de 1991 à 1996 ? Eh bien, uniquement en HTML ! Il y avait en effet des balises HTML dédiées à la mise en forme. <font color="#aab1c3"> , par exemple, permettait de définir la couleur du texte.

Il existe plusieurs versions du CSS, aujourd'hui CSS3. Certains navigateurs ne comprennent pas certains «styles» (IE bien souvent) pensez à utiliser : https://www.caniuse.com/

LA GENÈSE

CSS (Cascading Style Sheets), c'est cet autre langage qui vient compléter le HTML.

Vous vous souvenez de son rôle ? Gérer la mise en forme de votre site.

Petit rappel : à quoi sert CSS ?

Il permet de styliser votre contenu

Le HTML est né en 1991, et CSS en 1996. Alors, vous vous dites sûrement : comment faisait-on la mise en forme, de 1991 à 1996 ? Eh bien, uniquement en HTML ! Il y avait en effet des balises HTML dédiées à la mise en forme. <font color="#aab1c3"> , par exemple, permettait de définir la couleur du texte.

Il existe plusieurs versions du CSS, aujourd'hui CSS3. Certains navigateurs ne comprennent pas certains «styles» (IE bien souvent) pensez à utiliser : https://www.caniuse.com/

AJOUTER DU CSS

3 Méthodes :

-Dans un fichier .css (méthode la plus recommandée car elle permet de réduire le code) ;
-Dans l'en-tête <head> du fichier HTML ;
-Directement dans les balises du fichier HTML via un attribut style (méthode la moins recommandée).[Utile pour tester rapidement]

Nous verrons ensemble les 3 méthodes (Voir fichier 1-CSS-XXXX.html)

Ordre de priorité : CSS sur balise>css interne>css externe

Supposons que votre projet comporte les 3 types d'ajout de CSS le css sur la balise prendre le dessus sur le css externe (A moins d'utiliser un hack CSS «!important», que nous verrons par la suite).
Je vous recommande d'utiliser la version externe

AJOUTER DU CSS

Pourquoi utiliser la méthode externe d'inclusion de CSS?

Supposons que votre projet comporte plusieurs pages (ce sera le cas à 99,99%)

Vous allez devoir spécifier le css sur chaque page HTML si vous utilisez une méthode interne

Cela va allourdir votre projet et rendre difficile la maintenance ...

Si j'inclus la même feuille sur mes pages, je peux modifier à un seul endroit et il y aura moins de code!!

Ce sera plus lisible pour les autres devs!

APPLIQUER UN STYLE

```
p {
    color: red;
    border:1px dashed #240a0a;
}
```

Ce code signifie toutes les balises p seront écrites en rouge avec une bordure pointillé + code couleur hexa

Trois éléments différents :

Des noms de balises : on écrit les noms des balises dont on veut modifier l'apparence. Par exemple, si je veux modifier l'apparence de tous les paragraphes <p> , je dois écrire p ;

Des propriétés CSS : les « effets de style » de la page sont rangés dans des propriétés. Il y a par exemple la propriété color qui permet d'indiquer la couleur du texte, font-size qui permet d'indiquer la taille du texte, etc. Il y a beaucoup de propriétés CSS et, comme je vous l'ai dit, je ne vous obligerai pas à les connaître toutes par cœur ;

Les valeurs : pour chaque propriété CSS, on doit indiquer une valeur. Par exemple, pour la propriété color , il faut indiquer le nom de la couleur. Pour font-size , il faut indiquer quelle taille on veut, etc.

APPLIQUER UN STYLE

```
1 h1
2 {
3     color: blue;
4 }
5
6 em
7 {
8     color: blue;
9 }
```

```
1 h1, em
2 {
3     color: blue;
4 }
```

Il s'agit de la même chose, cela permet de combiner un ensemble css. Il suffit de séparer les expressions cibles par une virgule.

```
8 p
9 {
10     color: blue; /* Les paragraphes seront en bleu */
11 }
```

/* TEXTE */ permet de placer un commentaire.

CTRL + : sur VSC permet de placer un commentaire

CLASS ET ID : CIBLE UN ELEMENT

On en avait parlé en HTML.

Comment faire pour que certains paragraphes seulement soient écrits d'une manière différente sans écrire le CSS directement sur le balise avec «style»?

En utilisant les ID et les CLASS

ID : l'ID doit être unique à la page exemple : Merci de votre soutien

Si j'ai dans ma page un id = annonce je ne peux plus en avoir un deuxieme avec le même nom

On cible l'ID dans le feuille CSS avec #annonce{font-weight:bold}

CLASS : Ici une classe peut être identique sur plusieurs éléments c'est d'ailleurs le principe. De plus un élément peut avoir plusieurs classes, ce qui n'est pas le cas avec l'id qui doit être unique et au nombre de 1.

Exemple : Il faut beau On cible la classe dans la feuille css avec .bleu{color:blue}

A noter on peut coupler les deux: exemple

Il faut beau

Les ID et les CLASS sont également utiles en javascript.

CLASS ET ID : EXERCICE PRATIQUE

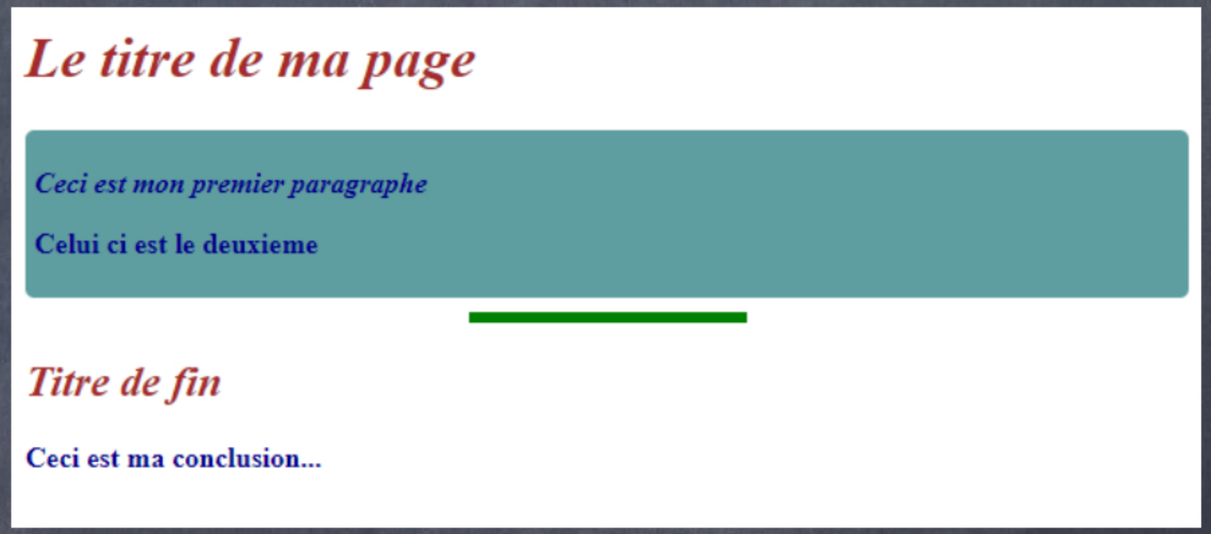
2-CSS > 2-id-class.html > html

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="assets/css/style2.css">
    <title>Document</title>
</head>
<body>
    <h1>Le titre de ma page</h1>
    <div id="entete">
        <p class="gras bleu italic">Ceci est mon premier paragraphe</p>
        <p class="gras bleu">Celui ci est le deuxieme</p>
    </div>
    <hr>
    <h2>Titre de fin</h2>
    <p class="gras bleu">Ceci est ma conclusion...</p>
</body>
</html>
```

Créer la feuille CSS en rapport avec ce code HTML

Pour que le rendu final ressemble à ça.

(A peu près...)



Utilisez ce site pour vous aider : https://www.w3schools.com/cssref/css3_pr_border-radius.asp

Dans google : exemple -> text gras css w3school

w3school est très simple à comprendre car beaucoup d'exemples.

Correction fichier : 2-id-class.html

CLASS ET ID : EXERCICE PRATIQUE

Si je veux mettre en couleur ou changer le contenu spécifique d'une balise je peux utiliser le «span»

```html
<p>Bonjour et <span class="salutations">bienvenue</span> sur mon site !</p>
```

```css
.salutations
{
    color: blue;
}
```

Vu en html : RAPPEL

Le span est une balise inline, il n'y a donc pas de retour à la ligne.

Autre balise inline : img,a,strong,i,em,input,...

Le div est une balise bloc, en l'utilisant il y a un retour à la ligne

Autre balise bloc : p,ul+li,h1,h2,form,...

LES SELECTEURS CSS

```
*{
    /* SELECTEUR UNIVERSEL */
}

div p{
    /* SELECTEUR CONTENU INTERNE */
}

div > p{
    /* SELECTEUR CONTENU INTERNE PREMIER ENFANT */
}

h1 + p {
    /* BALISE QUI EN SUIT UNE AUTRE */
}

input[type="text"]{
    /* SELECTEUR AVEC ATTRIBUT SPECIFIQUE */
}

p:first-child{
    /* PREMIER ENFANT DU PARENT */
}

#id{
    /* SELECTEUR ID */
}

.class{
    /* SELECTEUR CLASSE */
}
```

Selecteur universel : Sélectionne toutes les balises sans exception.

Contenu interne : je sélectionne toute les balises p contenu dans les div

Contenu interne > : je sélectionne toute les balises p contenu dans les div de premier niveau

Balise suivante : je sélectionne le premier <p> qui suit le h1

Selecteur attribut : Je sélectionne les input de type=»text»

Premier enfant : Sélectionne les premiers enfants <p> des balises parents

Voir : https://www.w3schools.com/cssref/tryit.asp?filename=trycss_sel_firstchild

id : Sélectionne la balise ayant id=XXXX (utiliser le #)

class : Sélectionne la ou les balises ayant class=XXX (utiliser le .)

PRIORITE DES SELECTEURS

Tous les sélecteurs n'ont pas la même priorité. En cas de contradiction entre deux règles, celle qui s'applique est celle qui a le sélecteur le plus prioritaire. De façon générale, les sélecteurs ont une priorité d'autant plus grande qu'ils sont plus précis. Le sélecteur très général * a une priorité de 0. A l'inverse, un sélecteur sur un identifiant , qui en principe ne concerne qu'un seul élément de la page, à une priorité de 100.

Une règle avec la mention !important a une priorité de 10000.

Une règle écrite dans l'attribut style d'une balise HTML a une priorité de 1000.

Une règle avec un sélecteur sur un identifiant (#) a une priorité de 100

Une règle avec un sélecteur sur une classe (.) ou des pseudo-class (:) a une priorité de 10.

Une règle avec un sélecteur sur un type d'élément (p) ou des pseudo-elements (::) a une priorité de 1.

Le sélecteur étoile (*) a une priorité de 0.

Lorsqu'un sélecteur comporte plusieurs parties combinées, les priorités de chacune des parties s'additionnent pour donner la priorité globale du sélecteur.

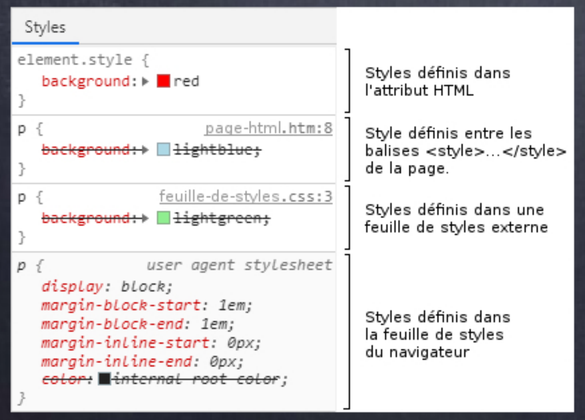
Une exception cependant : lorsque le sélecteur est composé de plusieurs parties séparées par une virgule, chaque partie est considérée comme un sélecteur à part entière. Chacune des parties peut éventuellement recevoir une priorité différente.

PRIORITE DES SELECTEURS

Enfin, lorsque deux règles contradictoires ont un sélecteur de même priorité, la dernière règle rencontrée remplace les valeurs définies par les règles précédentes.

```
/* Sélecteur sur un identifiant -> priorité = 100 */
#edito {color:blue;}

/* Sélecteur sur une pseudo-classe -> priorité = 10 */
:link {color:inherit;}

/* Sélecteur sur un type d'élément -> priorité = 1. */
p {font-size:1.1em;}

/* Règle comportant la mention !important -> priorité = 10000 */
p {color:silver!important}   Priorité = 10000.

/* Sélecteur sur un identifiant et un type -> priorité = 101 */
#edit img {width:25%;}

/* Sélecteur comportant 2 désignations de type -> priorité = 2 */
td>img {width:100%;}
```

Toujours utiliser la console en cas de problème pour vérifier qu'elle est la propriété activée.



Il s'agit d'un sujet vaste je vous invite à consulter :

http://stylescss.free.fr/v2-tutoriels/priorities.php

Si page non dispo -> Dans google

«Priorité selecteurs CSS»

CUSTOMISATION CSS

(Voir - 3-customisation.html)

Gestion de la taille : font-size : XX px (taille absolue) ou font-size : 1 em (se base par rapport au parent)

Gestion de la police : font-family : police1, police2, police3.

Si police un n'existe pas, utilisation de police 2 ...

Inclure une police personnalisée (dafont, etc...)

Dans votre feuille CSS

```
@font-face {
    font-family: 'MaSuperPolice';
   url('MaSuperPolice.eot') format('eot'),
   url('MaSuperPolice.woff') format('woff'),
   url('MaSuperPolice.ttf') format('truetype'),
   url('MaSuperPolice.svg') format('svg');
}
```

Plusieurs polices car le navigateur choisit directement celle de son choix (eot pour IE)

CUSTOMISATION CSS

Mettre en gras : utilisation de <b>, ou en css

p{ font-weight:bold}

Mettre en italique : utilisation de <i> ou en css

p{ font-style:italic}

Texte surligné : p { text-decoration : underline }

Texte barré : p {text-decoration : line-through }

Alignement : On ne peut modifier l'alignement que sur des balises bloc (<p>, ,<hn>)

p {text-align:left|center|right}

Technique de positionnement sur div : float.

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_float

CUSTOMISATION CSS

Gestion bloc, inline : Possibilité de transformer une balise inline ou bloc dans un autre type

p{display:inline|block|inline-block|none}

A noter : inline-block est identique à une balise de type block sauf qu'il n'y a pas de retour à la ligne

display:none permet de cacher un contenu

Taille d'une balise block ou inline-block

```
p{
 width:100px ou en %(en fonction de la taille de l'écran);
height:100px ou en %(en fonction de la taille de l'écran);
}
```

Espacement externe (balise inline-block ou block)

```
p{
   margin: 10px 10px 10px 10px; (dans le sens top,right,bottom,left)
   margin-top:5px (pour cibler un côté)
}
```

CUSTOMISATION CSS

Espacement interne (balise inline-block , inline, block)

```
p{
   padding: 10px 10px 10px 10px; (dans le sens top,right,bottom,left)
   padding-top:5px (pour cibler un côté)
}
```

Couleur du texte

```
p {
color:black|red|#XXXXXX  -> valeur hexa
}
```

On doit toujours commencer par écrire un dièse (#), suivi de six lettres ou chiffres allant de 0 à 9 et de A à F.

Ces lettres ou chiffres fonctionnent deux par deux. Les deux premiers indiquent une quantité de rouge, les deux suivants une quantité de vert et les deux derniers une quantité de bleu. En mélangeant ces quantités (qui sont les composantes Rouge-Vert-Bleu de la couleur) on peut obtenir la couleur qu'on veut.

CUSTOMISATION CSS

Gestion bordure

```
p{ border:3px solid #FF0000; /*Bordure de 3px pleine*/ Il existe d'autres valeurs (dotted|dasehd|double..)*/
border-radius:5px; (permet l'arrondi de la bordure d'un bloc)
}
```

Gestion du fond

```
.hero-image {
    background-image: url("XXXXX.jpg"); /* Une image de fond attention au chemin */
    background-color: #cccccc; /* Utilisé si l'image n'est pas dispo ou pas assez grande */
    height: 500px; /* Taille en hauteur */
    background-position: center; /* Image centrée */
    background-repeat: no-repeat; /* L'image n'est pas répétée */
    background-size: cover; /* Redimensionne le fond pour couvrir la surface */
  }
```

Vous rencontrerez également background: .... qui est une version raccourcie du dessus

https://developer.mozilla.org/fr/docs/Web/CSS/Propri%C3%A9t%C3%A9s_raccourcies

NOTION HÉRITAGE CSS

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge
    <meta name="viewport" content="width=device-width,
    <link rel="stylesheet" href="assets/css/style4.css"
    <title>Document</title>
</head>
<body>
    <h1>Mon titre 1</h1>
    <p>Ceci est un paragraphe</p>
</body>
</html>
```

```css
body{
    background-color: #FF0000;
    color: white;
    text-align:center;
}
h1{
    color: yellow;
}
.hero-image {
    background-image: url("XXXXX.jpg"); /* The image used */
    background-color: #cccccc; /* Used if the image is unav
    height: 500px; /* You must set a specified height */
    background-position: center; /* Center the image */
    background-repeat: no-repeat; /* Do not repeat the image
    background-size: cover; /* Resize the background image to
}
```

On remarque que le contenu de la balise <p> est blanc. Pourquoi?

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style.

La balise <body> contient entre autres les balises de paragraphe <p> et de titre <h1> .

Si j'applique une couleur de fond rouge et une couleur de texte blanche à la balise <body> , tous mes titres et paragraphes auront eux aussi un arrière-plan de couleur rouge et un texte de couleur blanche... C'est ce phénomène qu'on appelle l'héritage : on dit que les balises qui se trouvent à l'intérieur d'une autre balise « héritent » de ses propriétés.

Si je souhaite modifier la couleur héritée ou autre propriété je dois redéfinir sa valeur via CSS (voir le <h1> en jaune)

CSS DYNAMIQUE

voir 4-dynamique.html

S'applique sur les balises en utilisant son nom, son id ou sa classe

```
a /* Liens par défaut (non survolés) */
{
   text-decoration: none;
   color: red;
   font-style: italic;
}

a:hover /* Apparence au survol des liens */
{
   text-decoration: underline;
   color: green;
}

a:active /* Quand le visiteur clique sur le lien */
{
    background-color: #FFCC66;
}

a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color: #FFCC66;
}

a:visited /* Quand le visiteur a déjà vu la page concernée */
{
    color: #AAA; /* Appliquer une couleur grise */
}
```

Transition fluide

https://www.w3schools.com/css/css3_transitions.asp

```
/* TRANSITION */
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s;
  }

  div:hover {
    width: 300px;
  }
```

CSS ANIMATION

voir 5-animation.html

Permet d'animer un élément. Initialement on utilisait javascript

Plus d'infos ici

https://www.w3schools.com/css/css3_animations.asp

S > assets > css > # style6.css > @keyframes mymove

```css
div {
    width: 100px;
    height: 100px;
    background: red;
    position:relative;
    animation: mymove 5s infinite;
  }

  @keyframes mymove {
    0% {top: 0px;}
    50% {top: 100px;}
    100% {top: 0px;}
  }
```

2-CSS > 5-animation.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="assets/css/style6.css">
    <title>Document</title>
</head>
<body>
    <div></div>
</body>
</html>
```

POSITIONNEMENT

Unitialement on utilisait la méthode du tableau pour gérer sa mise en page. Avec le responsive design, il n'est plus possible d'utiliser cette technique.

Il en existe cependant plusieurs :

- Le positionnement 'float' (voir 6-float.html)
- Le fait de transformer des balise inline en bloc avec CSS (déjà vu)
- Le système de position (relative / absolu) (voir 7-position.html)
- Le flexbox (voir 8-flexbox.html)

On s'attardera sur le positionnement flexbox car il s'agit aujourd'hui de la meilleure façon de positionner son contenu.

FLEXBOX

Voir 8-flexbox.html

Flexbox : vous définissez un conteneur, et à l'intérieur vous placez plusieurs éléments que vous allez gérer via le CSS facilement.

```css
#conteneur
{
    display: flex;
}
.element{
  width:150px;
  height:150px;
}
.rouge{background: red}
.vert{background: green}
.bleu{background: blue}
```

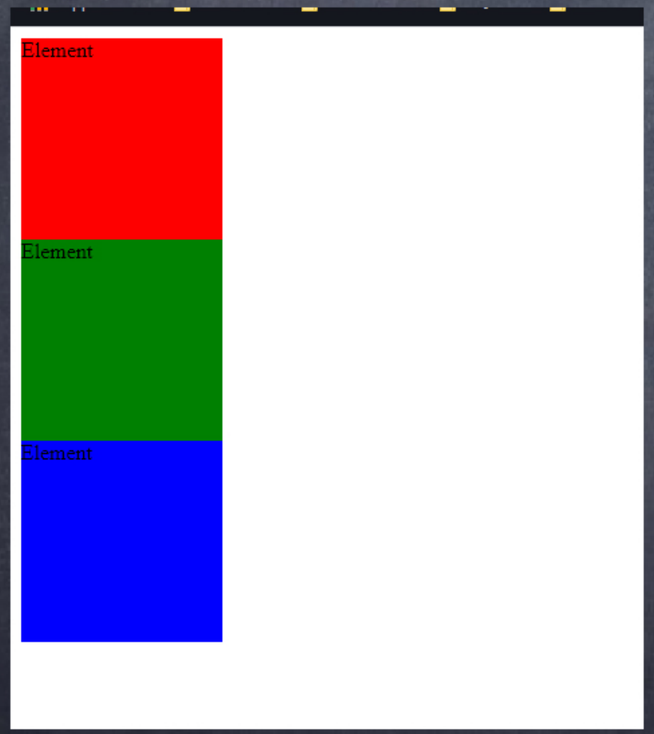
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, ini
    <link rel="stylesheet" href="assets/css/style7.css">
    <title>Document</title>
</head>
</head>
<body>
    <div id="conteneur">
        <div class="element rouge">Element </div>
        <div class="element vert">Element </div>
        <div class="element bleu">Element </div>
    </div>
</body>
</html>
```

FLEXBOX

Voir 8-flexbox.html

flex-direction:column - affichage des éléments en colonne

flex-direction:column-reserve - Affichage inversé




```
#conteneur
{
    display: flex;
    flex-direction:column /*Affichage colonne*/
}
.element{
  width:150px;
  height:150px;
}
.rouge{background: red}
.vert{background: green}
.bleu{background: blue}
```

FLEXBOX

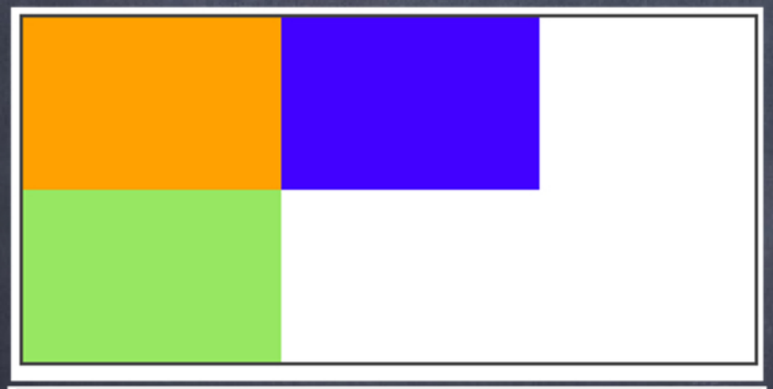
Voir 8-flexbox.html

Les blocs essaient de rester sur la même ligne s'ils n'ont pas la place. Si vous voulez, vous pouvez demander à ce que les blocs aillent à la ligne lorsqu'ils n'ont plus la place, avec flex-wrap qui peut prendre ces valeurs :

nowrap

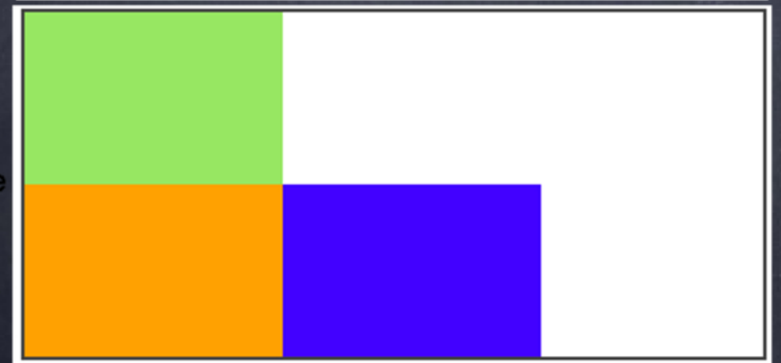
Les éléments se resserrent tant qu'ils peuvent

wrap



Les éléments passent à la ligne

wrap-reverse



Les éléments passent à la ligne à l'envers

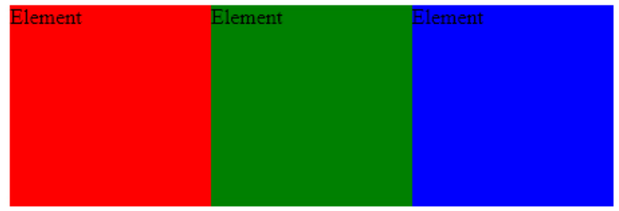
```
#conteneur
{
    display: flex;
    /*flex-direction:column;*/ /*Affichage colonne*/
     flex-wrap: wrap;  /*RETOUR LIGNE*/
}
.element{
  width:150px;
  height:150px;
}
.rouge{background: red}
.vert{background: green}
.bleu{background: blue}
```

FLEXBOX

Alignement : Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle l'axe principal. Il y a aussi un axe secondaire (cross axis) :

si vos éléments sont organisés horizontalement, l'axe secondaire est l'axe vertical ;

si vos éléments sont organisés verticalement, l'axe secondaire est l'axe horizontal.




```
#conteneur
{
    display: flex;
    /*flex-direction:column;*/ /*Affichage colonne*/
     flex-wrap: wrap;  /*RETOUR LIGNE*/
     justify-content: center; /*alignement principal*/
     align-items: center; /*alignement secondaire*/
     height:500px; /*il faut utiliser une largeur au conteneur*/
     border:1px solid black;
}
.element{
  width:150px;
  height:150px;
}
.rouge{background: red}
.vert{background: green}
```

FLEXBOX

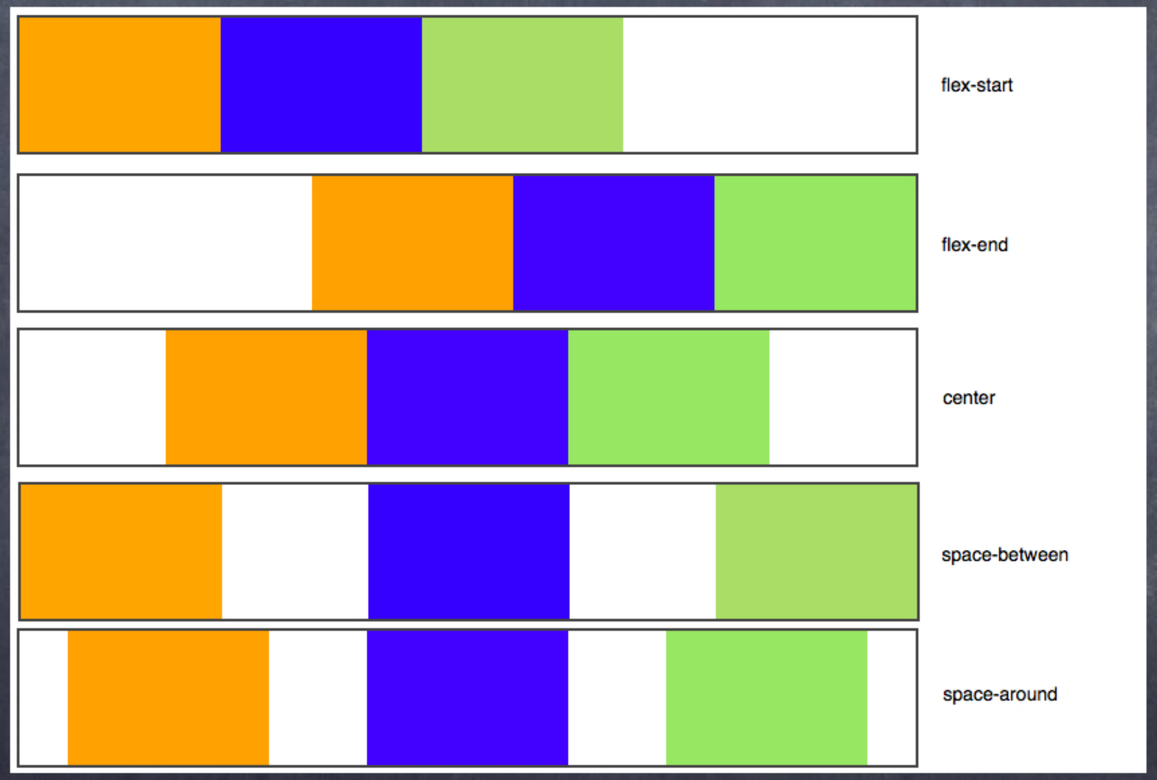
Valeur possible de justify-content et align-items.

Flexbox permet de gérer l'affichage sans changer le code HTML c'est la meilleure solution!!

Il existe encore quelques possibilités moins utilisées (changement d'ordre, position spécifique)

Retrouvez la bible ici :

https://css-tricks.com/snippets/css/a-guide-to-flexbox/



MEDIA-QUERY

(cf 9-responsive.html) L'utilisation des media-query permet de gérer un affichage différent en fonction de la taille de l'écran. Cela permet de proposer un affichage responsive (compatible sur l'ensemble des supports)

Il s'agit de la norme actuelle. On dit d'ailleurs que l'on doit travailler «Mobile First»

On travaille la version mobile en priorité.

Comment ça marche :

```
@media screen and (max-width: 1280px)
{
    /* Rédigez vos propriétés CSS ici */
}
```

Il est également possible de l'ajouter sur une feuille CSS externe complète mais cette méthode n'est pas intéressante

Règles dispos :

- color : gestion de la couleur (en bits/pixel) ;
- height : hauteur de la zone d'affichage (fenêtre) ;
- width : largeur de la zone d'affichage (fenêtre) ;
- device-height : hauteur du périphérique ;
- device-width : largeur du périphérique ;
- orientation : orientation du périphérique (portrait ou paysage) ;
- media : type d'écran de sortie. Quelques-unes des valeurs possibles :
    - screen ou all : écran « classique »,
    - handheld : périphérique mobile,
    - print : impression,
    - tv : télévision,
    - projection : projecteur,

MEDIA-QUERY

Exemple d'utilisation

```
/* Sur les écrans, quand la largeur de la fenêtre fait au maximum 1280px */
@media screen and (max-width: 1280px)

/* Sur tous types d'écran, quand la largeur de la fenêtre est comprise entre 1024px et 1280px */
@media all and (min-width: 1024px) and (max-width: 1280px)

/* Sur les téléviseurs */
@media tv

/* Sur tous types d'écrans orientés verticalement */
@media all and (orientation: portrait)
```

MEDIA-QUERY

Aucune utilisation de media-query, ce css sera appliqué par défaut

```
#conteneur
{
    display: flex;
    /*flex-direction:column;*/ /*Affic
     flex-wrap: wrap;  /*RETOUR LIGNE*
     justify-content: center; /*aligne
     align-items: flex-start; /*aligne
     height:500px; /*il faut utiliser
     border:1px solid black;
     background: #dddddd;
}
.element{
  width:150px;
  height:150px;
  font-size:25px;
  text-align: center;
  font-weight: bold;;
}
.rouge{background: red}
.vert{background: green}
.bleu{background: blue}
```

Sur l'ensemble des supports entre 900px et 1280px de largeur ce css surchargera le précédent

```
@media  all and (min-width: 900px) and (max-width: 1280px)
{
    #conteneur
    {
        border:3px solid black;
        background: #907d7d;
    }
    .element{
        width:100px;
        height:100px;
        font-size:18px;
        text-align: left;
      }
}
```

Sur des écrans avec maximum 899px de largeur ce css prendra le dessus

```
@media  all and (max-width: 899px)
{
    #conteneur
    {
        border:3px solid rgb(230, 81, 81);
        background: #4b4040;
    }
    .element{
        width:80px;
        height:110px;
        font-size:16px;
        text-align: left;
      }
}
```

Cela permet de disposer d'affichage spécifique en fonction de la taille des supports. Vous pouvez effectivement réaliser cela «from-scratch» (à la main) ou utiliser un framework comme bootstrap qui permet de gérer le responsive design plus facilement, plus rapidement et avec un rendu très sympa!

-Framework : Cadre de développement complet (il y a une logique à suivre) (symfony/bootstrap/angular)

-Bibliothèque : Moins formel, il s'agit d'un ensemble de fonctionnalité prédéfinie (React/Jquery)

BOOTSTRAP

Bootstrap est un framework permettant de gérer plus facilement la partie CSS de votre programme.

C'est un framework développé par l'équipe de Twitter. Nous utiliserons la version 5.X

https://getbootstrap.com/docs/5.0/getting-started/download/ (comment l'installer)

https://getbootstrap.com/docs/5.0/layout/grid/ (le principe de grille)

https://getbootstrap.com/docs/5.0/components/alerts/ (présentation des composants)

Sachez qu'utiliser bootstrap ne vous oblige pas à adopter le visuel prédéfini, vous pouvez ajouter votre feuille css pour surcharger les composants par défaut.

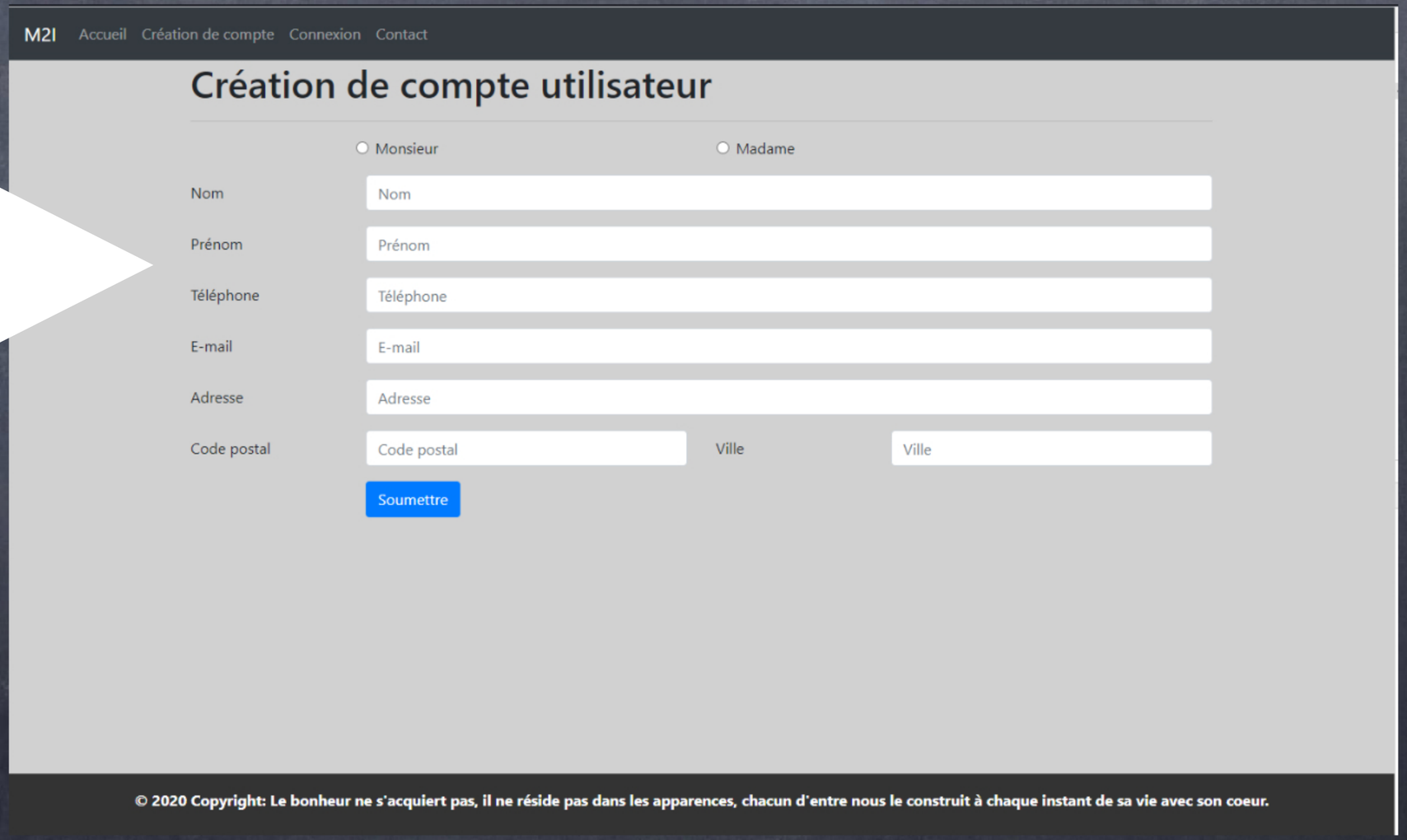
BOOTSTRAP

Votre but est d'avoir un rendu proche de celui ci en utilisant bootstrap + si besoin une feuille css supplémentaire.

Menu : Mettez des faux liens

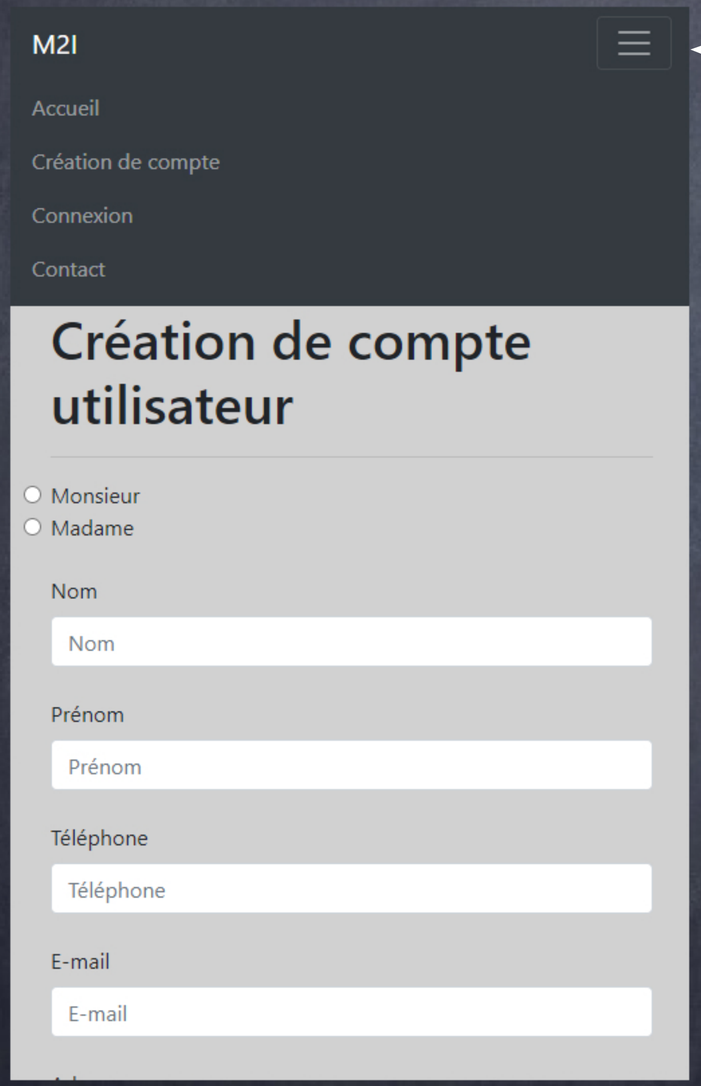
Formulaire

Footer



BOOTSTRAP

Version mobile



Menu pouvant s'ouvrir et se fermer

Ce n'est pas grave si votre version ne ressemble pas à l'identique à la mienne.

Vous ne connaissez pas bootstrap? C'est normal, une seule solution, consultez la documentation et les forums.

BOOTSTRAP

Aide :

Installer bootstrap soit en téléchargeant les sources ou en utilisant le CDN

https://getbootstrap.com/docs/5.0/getting-started/introduction/

```html
<html lang="en">
  <head>
    <title>Title</title>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS v5.0.2 -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"  integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao

  </head>
  <body>

    <!-- Bootstrap JavaScript Libraries -->
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAw
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H
  </body>
</html>
```

Ceci est un bon début!

BOOTSTRAP

Aide :

https://getbootstrap.com/docs/5.0/components/alerts/

Dans la partie «components» vous trouverez facilement une barre de navigation et des infos sur les formulaires.

Pour le principe de responsive design cela fonctionne avec «flex» et c'est ici que ça se passe

https://getbootstrap.com/docs/5.0/layout/grid/

Bootstrap utilise une grille de 12 cellules et des colonnes un peu comme un tableau.

En utilisant des classes css sur les éléments vous pouvez dire :

Cet élément prendra 2 cases quand l'affichage est petit mais 5 cases quand l'affichage est grand :

exemple :

 cela permet d'avoir un affichage très précis sans avoir à coder.

Passez un peu de temps à lire la documentation, bootstrap est très utilisé dans le monde professionnel.

Autre framework CSS : Tailwind, Bulma, modernize. Tailwind 3 possède un avantage (la purge CSS)