

Cascading Style Sheet



diginamic
FORMATION

- [1 Historique](#)
- [2 Concept de la cascade](#)
 - [2.1 Reset](#)
- [3 Liaison avec le HTML](#)
 - [3.1 En externe](#)
 - [3.2 En interne](#)
 - [3.3 En ligne](#)
- [4 Syntaxe de base](#)
- [5 Les sélecteurs](#)
 - [5.1 Sélection par type](#)
 - [5.2 Sélection par descendance](#)
 - [5.3 Groupement](#)
 - [5.4 Sélection par class et id](#)
 - [5.5 Sélection par pseudo class](#)

- [5.6 Sélecteur par attribut](#)
- [5.7 Sélecteur par attribut avec regexp](#)
- [5.8 Poids des sélecteurs](#)
- [6 Les propriétés de base](#)
 - [6.1 Positionnement \(à l'ancienne !\)](#)
 - [6.2 Propriétés de boîte](#)
 - [6.3 Dimensions](#)
 - [6.4 Fonds](#)
 - [6.5 Liste](#)
 - [6.6 Polices et textes](#)
- [7 Principaux apports CSS3](#)
 - [7.1 Border radius](#)
 - [7.2 Box shadow](#)
 - [7.3 Box Sizing](#)
 - [7.4 Transform](#)
 - [7.5 Transitions](#)
 - [7.6 Combinaison de transition et de transform](#)
 - [7.7 Key frames](#)
 - [7.7.1 Animation](#)
 - [7.7.2 Exemple :](#)
 - [7.7.3 Itérations :](#)
 - [7.8 Gradients](#)
 - [7.9 Media Queries](#)

- 7.10 Fonctions
- 8 Unités
- 9 Flex
 - 9.1 display: flex; Création du conteneur flexible
 - 9.2 flex-direction : Choix de l'axe
 - 9.3 flex-wrap : conteneur flexible sur plusieurs lignes
 - 9.4 flex-flow : la propriété raccourcie pour flex-direction et flex-wrap
 - 9.5 Propriétés s'appliquant aux éléments flexibles
 - 9.5.1.1 flex-grow : facteur d'expansion d'un élément flexible
 - 9.5.1.2 flex-shrink : facteur de rétrécissement d'un élément flexible.
 - 9.5.1.3 flex-basis : taille initiale des éléments flexibles
 - 9.5.1.4 flex : la propriété raccourcie pour flex-grow, flex-shrink et flex-basis
 - 9.6 justify-content et align-items
 - 9.6.1.1 justify-content
 - 9.6.1.2 align-items
 - 9.7 align-content : Répartition de l'espace entre et autour des éléments le long de l'axe transversal
 - 9.8 align-self

- [9.9 gap](#)
- [9.10 Order](#)
- [9.11 Exercice 1](#)
 - [9.11.1.1 Trouver le code css adapté pour obtenir le résultat classique ci-dessus :](#)
- [9.12 Exercice 2 : Création d'une grille 960 avec flex](#)
- [10 Glyphicons avec fontello](#)

1 Historique

Les feuilles de style ont toujours existé sous différentes formes depuis la naissance du SGML (Standard Generalized Markup Language).



Robert Caillau, collaborateur au CERN de Tim Berners-Lee

à la fin des années 80, avait pour ambition de séparer la structure de la présentation.

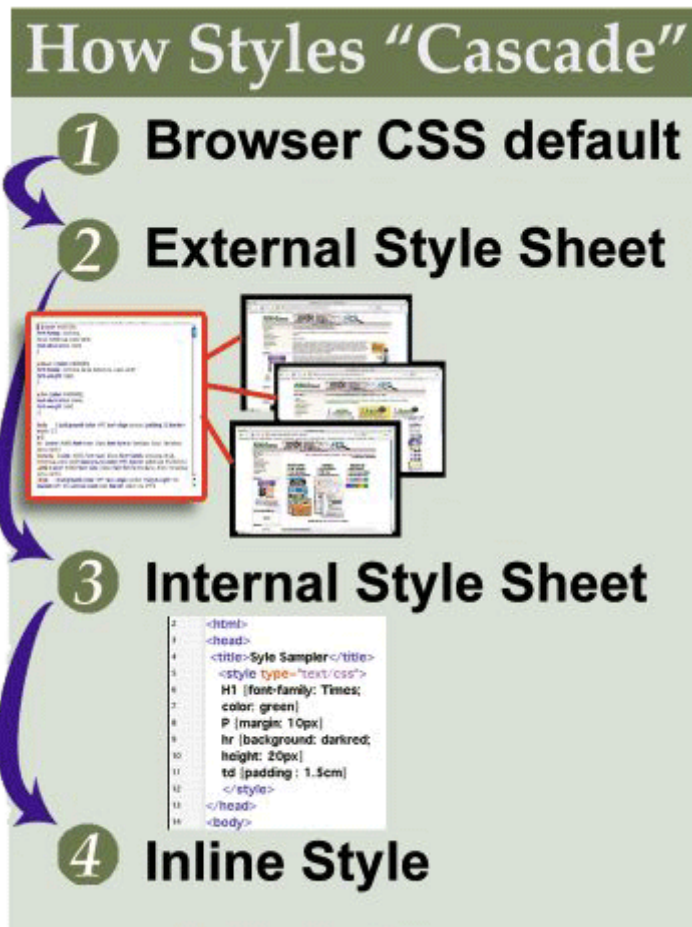


Plusieurs pistes de langage ont été proposées au W3C avant que le CHSS puis le **CSS** de **Håkon Wium Lie** (chef de développement pour Opéra) ne soit choisi.

- CSS1 : publié en 1996
- CSS2 : publié en 1998
- CSS3 : divisé en modules : 2012 (Media Queries, transitions et animation, flexbox et grid, box sizing, gradients, ...)

2 Concept de la cascade

options de configuration Ouvert



Par défaut, les navigateurs ont leur propres feuilles de style.

Le travail "classique" de l'intégrateur consiste à créer des fichiers ".css" que l'on reliera via la balise "" aux fichiers

HTML. L'intérêt est de mettre en commun ces feuilles de style si bien que la modification d'un fichier css aura potentiellement un impact sur un grand nombres de fichiers html.

Les feuilles de style par défaut des navigateurs ne sont pas les mêmes (surtout sur les vieux navigateurs).

2.1 Reset

[Pour s'assurer que les navigateurs aient tous les mêmes styles de base, on peut utiliser le "reset" d'Eric Meyer](#)

Il est également possible, mais c'est déconseillé, d'utiliser des feuilles de style internes (via la balise "style") aux fichiers html voire des styles "en ligne" via l'attribut de balise "style".

Nous verrons dans le chapitre suivant qu'il est indispensable de bien comprendre le mécanismes des "poids des sélecteurs" pour savoir quel style s'applique finalement.

3 Liaison avec le HTML

3.1 En externe

```
<link type="text/css" href=" ./mystyle.css" />
```

3.2 En interne


```

<head>
  <style type="text/css">
    hr {color:sienna;}
    p {margin-left:20px;}
    body {background-
image:url("images/back40.gif");}
  </style>
</head>

```

3.3 En ligne

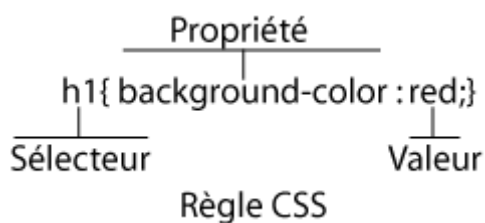
```

<p style="color:blue;margin-left:20px;">This is a
paragraph.</p>

```

4 Syntaxe de base

- Selecteur
- bloc de code
- propriété
- valeur



5 Les sélecteurs

5.1 Sélection par type

ou par balise ex : header

5.2 Sélection par descendance

section header article

section>header>article

5.3 Groupement

h2, h3

5.4 Sélection par class et id

.actu

#actu1

5.5 Sélection par pseudo class

:hover

:after

:before

:nth-child(2n+1)// attention, tous les éléments "frères"
comptent !

5.6 Sélecteur par attribut

article[title="bonjour"]





















5.7 Sélecteur par attribut avec regexp

`[class^="icon-"], [class*=" icon-"]`

5.8 Poids des sélecteurs

Les poids sont attribués par ordre d'importance :

1. !important
2. **(id en html)**
3. . (class en html)
4. type (de balise en comptant la descendance)
5. selecteur universel (*)

| | | | |
|---|--|---|---|
| <p>*</p>  <p>Sélecteur universel 0-0-0-0</p> | <p>DIV</p>  <p>1 élément 0-0-0-1</p> | <p>LI > UL</p>  <p>2 éléments 0-0-0-2</p> | <p>BODY DIV ... UL LI P A</p>  <p>12 éléments 0-0-0-12</p> |
| <p>.CLASS</p>  <p>1 class 0-0-1-0</p> | <p>*.CLASS</p>  <p>1 sélecteur universel 1 class 0-0-1-0</p> | <p>[TYPE=CHECKBOX]</p>  <p>1 attribut 0-0-1-0</p> | <p>:ONLY-OF-TYPE</p>  <p>1 pseudo-class 0-0-1-0</p> |
| <p>LI.CLASS</p>  <p>1 élément 1 class 0-0-1-1</p> | <p>LI[ATTR]</p>  <p>1 élément 1 attribut 0-0-1-1</p> | <p>LI:NTH-OF-TYPE(3N) ~ LI</p>  <p>2 éléments 1 pseudo-class 0-0-1-2</p> | <p>FORM INPUT[TYPE=EMAIL]</p>  <p>2 éléments 1 attribut 0-0-1-2</p> |
| <p>LI.CLASS:NTH-OF-TYPE(3N)</p>  <p>1 élément 1 class 1 pseudo-class 0-0-2-1</p> | <p>INPUT[TYPE]:NOT(.CLASS)</p>  <p>1 élément 1 class 1 attribut 0-0-2-1</p> | <p>.CLASS[ATTR]:NTH-CHILD()...</p>  <p>10 class / pseudo-class / attributs 0-0-10-0</p> | <p>#ID</p>  <p>1 identifiant 0-1-0-0</p> |
| <p>#ID LI.CLASS A[Href]</p>  <p>2 éléments 1 class - 1 attribut 1 identifiant 0-1-2-2</p> | <p>#KOOKIES #ID A</p>  <p>1 élément 2 identifiants 0-2-0-1</p> | <p>STYLE = " "</p>  <p>inline style 1-0-0-0</p> | <p>!IMPORTANT</p>  <p>!important 1-0-0-0-0</p> |



6 Les propriétés de base

6.1 Positionnement (à l'ancienne !)

- float (css1)
- clear (css1)
 - A utiliser pour faire passer des boîtes sous les éléments flottants (float)
- display (css1)
 - valeurs possibles :
 - none (pas d'affichage)
 - block (par défaut pour la grande majorité des balises) : prend toute la largeur
 - inline (par défaut pour img, span, b, strong, em, q, abbr, ... prend le minimum de place dans le flux du texte sans marge en haut et en bas
 - table (pour les tableaux)
 - inline-block (un mélange de block et inline
 - [flex](#),
 - ...
 - Attention, cette propriété est importante pour permettre à un élément en ligne d'occuper tout l'espace de son conteneur : display:block; height : 100%;
- position (css2)

Lorsque vous appliquez à un élément les règles :
position: absolute;

top: 0;

left: 0; L'élément en question sera positionné sur un autre calque et en haut à gauche du premier élément ancêtre qui aura pour règle : position: relative Si aucun de ses ancêtres n'a de position en relative, le "body" sera alors la référence.

Attention si l'on veut voir au dessus de l'élément positionné en absolute ses éléments frères, il faut leur appliquer la règle suivante : position: relative;

- Attention, dans le cas où on positionne un élément sur une image qui est en display "**inline**", un espace correspondant au "**descender**"

apparaît : <https://stackoverflow.com/questions/31444891/mystery-white-space-underneath-image-tag/31445364#31445364>

- top, right, bottom, left (css2)
- z-index (css2)
- :after{ /CSS2 / content: ""; display: block; clear:both; }

6.2 Propriétés de boîte

- Modèle de boîte



- Attention si les valeurs des propriétés de boîte sont exprimées en %, elles sont relatives à leur conteneur
- margin (css1)
- padding (css1)
- [border](#)
- [border-radius \(css3\)](#)
 - border-radius:8px 8px 8px 8px;

6.3 Dimensions

- width (css1)
 - S'applique sur tous les éléments block
 - En fonction de son contenant
 - Attention, le width dépend par défaut du padding, du border et du margin qui s'ajoutent au calcul. Pour corriger cela et faire en sorte que seul le

margin s'ajoute : utiliser la propriété

: <https://developer.mozilla.org/fr/docs/Web/CSS/box-sizing>

- height (css1)
 - S'applique sur tous les éléments block
- line-height (css1)
 - Hauteur de ligne des éléments inline
cf <https://www.paris-web.fr/2010/conferences/macrotypographie-page-web.php>
- max-width (css2)
 - En fonction de son contenant
- max-height (css2)

6.4 Fonds

- background-color (css1)
cf <https://color.adobe.com/fr/create/color-wheel>
- background-image (css1)
- background-repeat (css1)
- background-position (css1)
- background (css1)

```
p{ background:#FFCCCC url(.. /images/fond.png) no-repeat 50% 100%; } /* le repère de la position
```

vient du selecteur (ici p) et non pas de son conteneur */

- repeat-x(css1)
- repeat-y(css1)

6.5 Liste

- list-style (CSS1)
- list-style-type (css1)

6.6 Polices et textes

- font-family (css1)
- font-size (css1)
- font-style (css1)
- font-variant (css1)
- font-weight (css1)
- font (css1)
- color (css1)
- text-align (css1)
- line-height (css1)
- letter-spacing (css1)
- word-spacing (css1)
- white-space (css1)

7 Principaux apports CSS3

7.1 Border radius

<https://www.cssmatic.com/border-radius>

```
border-radius: 26px solid #000000;
```

7.2 Box shadow

<https://www.cssmatic.com/box-shadow>

```
box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);
```

7.3 Box Sizing

La propriété `box-sizing` permet de contrôler la manière dont la largeur et la hauteur d'un élément sont calculées, ce qui facilite la mise en page. Ce qui est en général utilisé, c'est la valeur `border-box` qui permet de s'assurer que le padding et le border sont inclus dans le calcul de la largeur (width).

```
div { box-sizing: border-box; }
```

7.4 Transform

<https://developer.mozilla.org/fr/docs/Web/CSS/transform>

La propriété transform modifie l'espace de coordonnées utilisé pour la mise en forme visuelle. Grâce à cette propriété, il est possible de translater les éléments, de les tourner, d'appliquer des homothéties, de les distordre pour en changer la perspective.

Ex :

`transform: scale(2, 2);` / multiplie par deux la taille d'une image par exemple /

`transform: rotate(0.5turn);` / Rotation de 180° /

`transform: translate(120px, 50%);` / Translation de 120px sur l'axe des x et de 50% sur l'axe des y /

7.5 Transitions

https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions

Les transitions CSS permettent de contrôler la vitesse d'animation lorsque les propriétés CSS sont modifiées. Plutôt que le changement soit immédiat, on peut l'étaler sur une certaine période.

Ex :

```
.box {  
    border-style: solid;  
    border-width: 1px;
```

```
display: block;
width: 100px;
height: 100px;
background-color: #0000FF;
transition: width 2s ease, height 2s ease,
background-color 2s ease, transform 2s ease;
}
.box:hover { background-color: #FFCCCC; width:
200px; height: 200px; transform: rotate(180deg); }
```

``

Spécifications de la vitesse de la courbe de transition :

- ease - Début lent puis rapide puis fin lente
- linear - Rapidité constante
- ease-in - Début lent
- ease-out - Fin lente
- ease-in-out - Début et fin lentes
- cubic-bezier(n,n,n,n) - Pour définir ses propres valeurs

7.6 Combinaison de transition et de transform

Ex :

```
#wrapper-img img {
  transition: transform 2s ease;
```

```
}  
#wrapper-img img:hover {  
  transform: scale(1.2, 1.2);  
  /* transform: translate(-50%, -50%); */  
  cursor: pointer;  
}
```

7.7 Key frames

<https://developer.mozilla.org/fr/docs/Web/CSS/@keyframes>

7.7.1 Animation

La règle `@keyframes` permet de définir les étapes qui composent la séquence d'une animation CSS. Cela permet de contrôler une animation plus finement que ce qu'on pourrait obtenir avec [les transitions](#).

Chaque règle `@keyframes` contient une liste de sélecteurs d'étapes dont chacun contient le pourcentage d'avancement de l'animation auquel il correspond ainsi que les informations de styles qui correspondent à cette étape..

Les étapes peuvent être listées dans n'importe quel ordre. Elles seront enchaînées dans l'ordre indiqué par le pourcentage d'avancement.

Si les étapes décrivent des propriétés qui ne peuvent pas être animées, elles seront ignorées mais les autres

propriétés seront bien animées.

Seules les propriétés qui sont définies sur les étapes de début (0%) et de fin (100%) seront animées. Toutes les propriétés qui ne sont pas incluses dans les descriptions de ces étapes conserveront leurs valeurs de départ au cours de l'animation.

!important dans une étape

Les déclarations qui utilisent !important dans une description d'étape sont ignorées

7.7.2 Exemple :

```
p {  
  animation-duration: 25s;  
  animation-name: slidein;  
}  
  
@keyframes slidein {  
  from {  
    margin-left: 100%;  
    width: 300%;  
  }  
  75% {  
    font-size: 300%;  
    margin-left: 25%;  
    width: 150%;  
  }  
}
```

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

7.7.3 Itérations :

animation-iteration-count permet de spécifier le nombre de fois où l'animation devra se jouer.

Référence : <https://developer.mozilla.org/en-US/docs/Web/CSS/animation-iteration-count>

7.8 Gradients

Les gradients permettent de créer des transitions douces entre plusieurs couleurs.

```
`/* Exemple de gradient linéaire */  
div { background: linear-gradient(to right, red,  
orange, yellow, green, blue, indigo, violet); }`
```

7.9 Media Queries

- @media screen and (max-width:960px) and (min-width : 450px){ / css ici /}

7.10 Fonctions

- [calc](#)
- [clamp](#)
- [max](#)
- [min](#)
- [variables](#)
- ...

8 Unités

options de configuration Ouvert

- 0
- px
- % (en fonction de la taille de l'élément parent)
- em (hauteur de la lettre "M" de l'élément parent)
- [rem \("r" pour root, ce qui signifie que la taille se base sur la taille de la police de root = élément html\)](#) Ex en sass :

```
:root {  
  font-size: 62.5%; /* 10px */  
  body {  
    font-size: 1.6rem; /* 16px */  
  }  
}
```

[Pour s'y retrouver plus facilement dans les conversions de taille de police de caractères]
(<http://pxtoem.com/>)

- vw et vh : [tailles relatives à la taille \(largeur et hauteur\)](#) de l'écran

9 Flex

options de configuration Ouvert

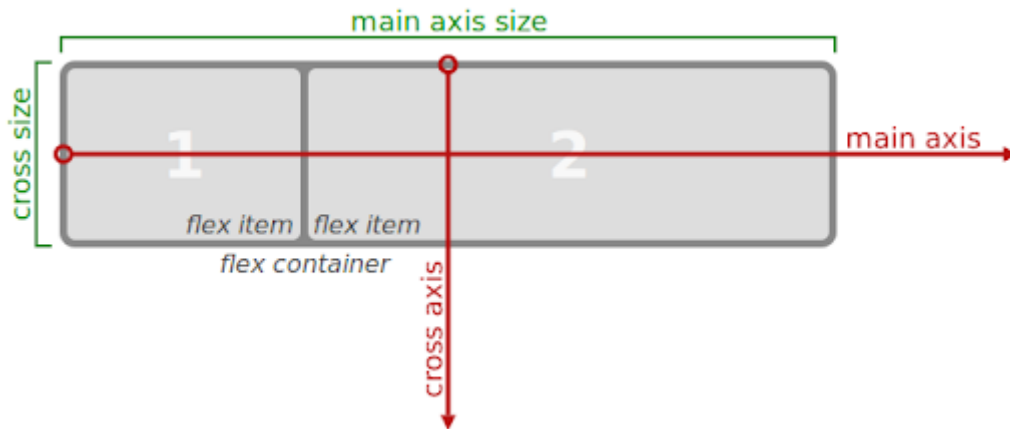
Tutos :

- https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Flexible_Box_Layout/Concepts_de_base_flexbox
- <https://la-cascade.io/flexbox-guide-complet/>
- <https://flexboxfroggy.com/>
- <https://codepen.io/enxaneta/full/adLPwv/> pour bien comprendre la différence entre align-content et align-items

9.1 display: flex; Création du conteneur flexible

```
section {  
  display : flex;  
}
```

L'élément section dispose maintenant d'un axe principal (**main axis**) qui est par défaut horizontal. L'axe principal d'un conteneur flexbox peut être horizontal ou vertical, mais par défaut il est horizontal. Le conteneur dispose également d'un **axe transversal** (cross axis) qui est par défaut vertical.



9.2 flex-direction : Choix de l'axe

On définit l'axe principal grâce à la propriété "flex-direction".

Les valeurs possibles sont :

- **row,**
- **row-reverse,**
- **column,**
- **column-reverse**

Exemple de code :

HTML

```
<section>
  <article>Contenu 1</article>
  <article>Contenu 2</article>
  <article>Contenu 3</article>
  <article>Contenu 4</article>
</section>
```

CSS

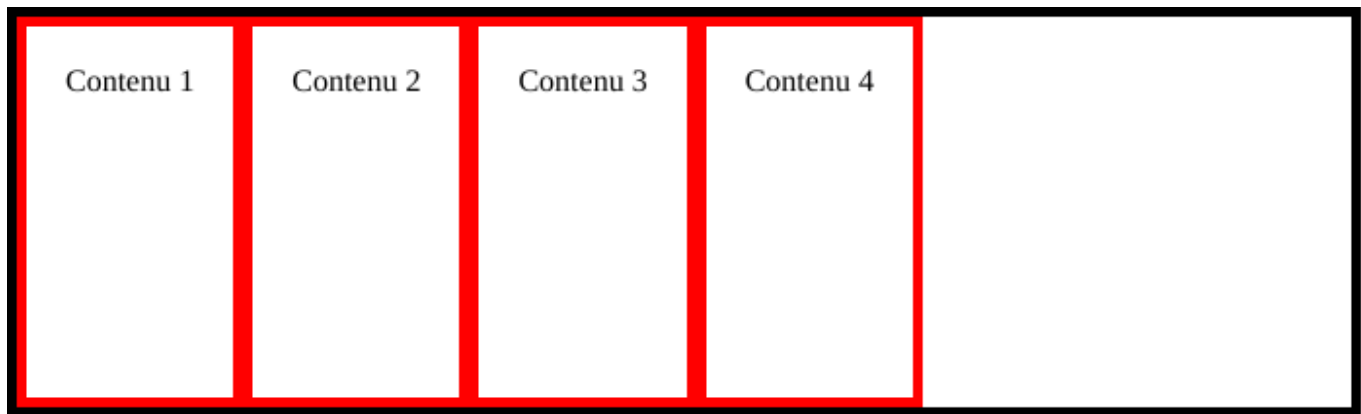
```
/*Élément parent (conteneur flexible)*/

section {
  display: flex;
  flex-direction: row;
  border: 5px solid black;
  height: 400px;
}

/*Éléments enfants (éléments flexibles) */

section > article {
  border: 5px solid red;
  padding: 20px;
}
```

Résultat



On observe que :

- les éléments enfants sont maintenant côte à côte sur une même ligne. Si l'on avait choisi "column" comme valeur à la propriété flex-direction, les articles seraient les uns sous les autres
- même si la largeur du navigateur est inférieure à la somme des largeurs des éléments enfants, ceux ci restent sur la même ligne, quitte à faire apparaître un ascenseur horizontal
- Si l'on agrandit la hauteur du conteneur, les éléments flexibles suivent cette hauteur. Les éléments sont donc étirés le long de l'axe secondaire afin d'occuper l'espace sur cet axe.
- La propriété flex-basis vaut auto (voir plus bas)
- La propriété flex-wrap vaut nowrap (voir plus bas)

9.3 flex-wrap : conteneur flexible sur plusieurs lignes

Par défaut la propriété flex-wrap vaut "nowrap". En modifiant cette valeur, les éléments enfants flexibles vont pouvoir passer à la ligne lorsque la largeur de leur conteneur ne sera plus suffisante.

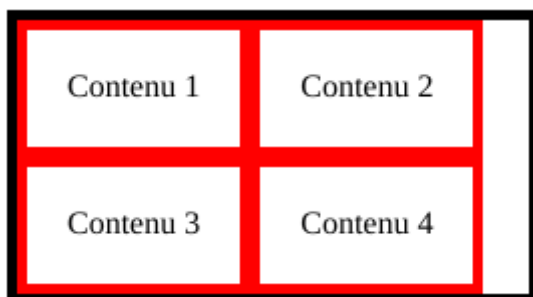
Les valeurs possibles sont :

- **wrap**
- **nowrap**
- **wrap-reverse**

CSS :

```
section {  
  display: flex;  
  flex-direction: row;  
  border: 5px solid black;  
  flex-wrap: wrap;  
}
```

Résultat :



9.4 flex-flow : la propriété raccourcie pour flex-direction et flex-wrap

Flex flow permet simplement de regrouper sur une même ligne les valeurs des propriétés flex-direction et flex-wrap.

Ainsi le code css du dessus peut être raccourci comme ceci :

```
section {  
    display: flex;  
    border: 5px solid black;  
    flex-flow: row wrap;  
}
```

9.5 Propriétés s'appliquant aux éléments flexibles

9.5.1.1 flex-grow : facteur d'expansion d'un élément flexible

flex-grow va permettre de savoir comment l'élément doit "grossir" en fonction :

- de l'espace disponible sur l'axe principal,
- de la valeur de flex-grow des autres éléments flexibles qui sont sur le même axe principal.

Les valeurs possibles sont : un **nombre positif** qui correspond au facteur de grossissement utilisé. Plus la valeur est élevée, plus l'élément sera étendu si nécessaire au regard et en proportion de la valeur de flex-grow des autres éléments.

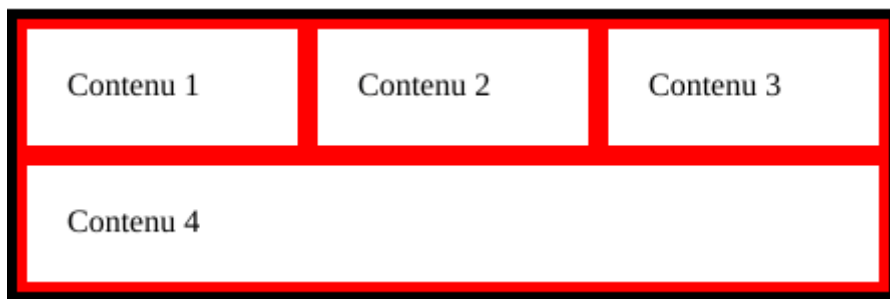
Exemple 1:

```
/* éléments enfants */  
section > article {  
  border: 5px solid red;  
  padding: 20px;  
  
  flex-grow: 1; /* ici tous les éléments auront le  
même comportement */  
}
```

Résultat dans le cas où tous les éléments peuvent se trouver sur la même ligne :



Résultat dans le cas où tous les éléments ne peuvent se trouver sur la même ligne :



Flex-grow différents

```
/* éléments enfants flexibles */  
section > article {  
  border: 5px solid red;  
  padding: 20px;  
  flex-grow: 1;  
}  
section > article:first-child {  
  flex-grow: 2;  
}
```

Dans ce cas, le premier élément va prendre 2 fois plus de place le long de l'axe principal :



9.5.1.2 flex-shrink : facteur de rétrécissement d'un élément flexible.

flex-shrink se comporte de la même façon que flex-grow mais s'applique dans le cas où il n'y a pas assez de place pour les éléments flexibles sur l'axe principal.

Les valeurs possibles sont : **un nombre positif** qui correspond au facteur de rétrécissement utilisé. Plus la valeur est élevée, plus l'élément sera rétréci si nécessaire au regard et en proportion de la valeur de flex-shrink des autres éléments.

9.5.1.3 flex-basis : taille initiale des éléments flexibles

[Tuto pour comprendre en profondeur flex-basis](#)

flex-basis détermine la base de flexibilité utilisée comme taille initiale principale pour un élément flexible. Cette propriété détermine la taille de la boîte de contenu mais son comportement peut être modifié si min-width et max-width sont définis.

Globalement, voici comment est déterminé la largeur d'un élément flex :

content —> width —> flex-basis (limité par max|min-width)

Il faut donc comprendre cette formule de la façon suivante :

- si rien n'est défini la largeur de l'élément dépend de son contenu.
- si seul width est défini, c'est lui qui s'applique

- sinon c'est flex-basis qui prend le dessus mais limité par max-width et min-width

Les principales valeurs possibles pour flex-basis sont : auto (par défaut), 0 (prend le minimum de place),

Exemples de valeurs possibles :

- flex-basis: 30%; / *relatif* /
- flex-basis: 3px;
- flex-basis: auto; / *par défaut* /
- flex-basis: content; / *en fonction du contenu de l'élément* /
- ...

Ex :

```
/* éléments enfants flexibles */
section > article {
  border: 5px solid red;
  padding: 20px;
  flex-grow: 1;
  flex-basis: auto;
}
section > article:first-child {
  flex-grow: 1;
  flex-basis: 0;
}
```

Résultat :

| | | | |
|--------------|-----------|-----------|-----------|
| Contenu 1 | Contenu 2 | Contenu 3 | Contenu 4 |
|--------------|-----------|-----------|-----------|

On note que la base de calcul de la taille des éléments n'étant pas la même, le premier élément est moins large que les autres. C'est d'autant plus visible lorsque l'espace disponible dans le conteneur parent n'est plus suffisant.

9.5.1.4 flex : la propriété raccourcie pour flex-grow, flex-shrink et flex-basis

La propriété raccourcie flex permet de définir les valeurs de cette propriété dans cet ordre : **flex-grow**, **flex-shrink**, **flex-basis**.

Ex :

```
/* éléments enfants flexibles */
section > article {
  border: 5px solid red;
  padding: 20px;
  flex: 1 1 auto;
}
```

La propriété **flex** permet également d'utiliser des valeurs synthétiques qui couvrent la majorité des scénarios.

- flex: initial
- flex: auto
- flex: none
- flex: nombre-positif

Avec flex: initial, les éléments récupèrent les valeurs initiales pour les différentes propriétés du modèle de boîte flexible. Cette valeur permettra d'obtenir le même comportement que flex: **0 1 auto**. Ici, flex-grow vaut 0 et les éléments ne s'agrandiront pas au-delà de la taille flex-basis. flex-shrink vaut 1 et les éléments pourront rétrécir si besoin plutôt que de dépasser du conteneur. flex-basis vaut auto et les éléments utiliseront donc la taille qui leur a été définie sur l'axe principale ou la taille déterminée à partir du contenu.

Avec flex: auto, on obtient le même comportement que flex: **1 1 auto**, la seule différence avec flex:initial est que les éléments peuvent s'étirer si besoin.

Avec flex: none, les éléments ne seront pas flexibles. Cette valeur est synonyme de flex: **0 0 auto**. Les éléments ne peuvent ni s'agrandir, ni se rétrécir mais seront disposés avec flex-basis: auto.

On voit aussi souvent des valeurs comme flex: 1 ou flex: 2, etc. Cela correspond à flex: **1 1 0**. Les éléments peuvent s'agrandir ou bien rétrécir à partir d'une taille de base égale à 0.

9.6 justify-content et align-items

Alignement, justification et distribution de l'espace disponible entre les éléments

9.6.1.1 justify-content

La propriété justify-content est utilisée afin **d'aligner** les éléments **le long de l'axe principal** dans la direction définie par flex-direction. La **valeur initiale est flex-start** qui **place les éléments à partir de la ligne de début du conteneur sur l'axe principal**. La valeur flex-end permet de les placer vers la fin et la valeur center permet de les centrer le long de l'axe principal.

Attention : notez que tous les flex-items doivent avoir la propriété flex-grow: 0; afin que la propriété justify-content fonctionne.

Les valeurs possibles sont :

- **flex-start**
- **flex-end**
- **center**
- **space-around**
- **space-between**
- **space-evenly** (même chose que "space-between" sauf que l'espace est le même à gauche du premier

élément, à droite du dernier élément et entre les éléments)

Exemple :

```
/* élément parent */
section {
  display: flex;
  flex-flow: row wrap;
  align-items: stretch;
  justify-content: space-around;
}
/* éléments enfants flexibles */
section > article {
  padding: 0;
  flex: 0 0 auto;
  width: 23%;
  background-color: blueviolet;
}
/* éléments enfants des enfants flexibles */
section > article > div {
  padding: 1rem;
}
```

Résultat :

Contenu 1

Contenu 2 plus
grand que les
autres pour tester

Contenu 3

Contenu 4

9.6.1.2 align-items

La propriété align-items permet d'**aligner les éléments le long de l'axe secondaire**.

La valeur initiale de cette propriété est **stretch**, ce qui explique pourquoi, par défaut, les éléments flexibles sont étirés sur l'axe perpendiculaire afin d'avoir la même taille que l'élément le plus grand dans cet axe (qui définit la taille du conteneur sur cet axe).

Les valeurs possibles sont :

- **stretch**,
- **flex-start**,
- **flex-end**,
- **center**

9.7 align-content : Répartition de l'espace entre et autour des éléments le long de l'axe transversal

<https://developer.mozilla.org/fr/docs/Web/CSS/align-content>

Cette propriété n'a de sens que si les éléments flexibles passent à la ligne (flex-wrap: wrap)

align-content définit la façon dont l'espace est réparti entre et autour des éléments le long de l'**axe transversal** lorsque

celui-ci est un conteneur de boîte flexible.

Les valeurs possibles sont :

- **center;** / Les éléments sont groupés au centre /
- **start;** / Les éléments sont groupés au début /
- **end;** / Les éléments sont groupés à la fin /
- **flex-start;** / Les éléments flexibles sont groupés au début /
- **flex-end;** / Les éléments flexibles sont groupés à la fin /
- **space-between;** / Espace réparti entre les éléments mais le premier et le dernier élément sont "collés" au conteneur /
- **space-around;** / L'espace est réparti entre les éléments /
- **normal** / default /

Attention, le comportement par défaut est un peu étrange. Si on prend le cas où l'axe principal est horizontal, la première ligne va être "collée" en haut du container. S'il y a plusieurs lignes et qu'il reste de l'espace, ce dernier sera réparti à part égale entre les lignes. C'est un comportement très proche de

- "space between" sauf que de l'espace sera réparti après la dernière ligne,

- "space around" sauf qu'il n'y aura pas d'espace réparti avant la première ligne.

9.8 align-self

Quand on veut changer les propriétés d'un enfant en particulier sur l'axe secondaire.

Les valeurs possibles sont :

- **auto**
- **flex-start**
- **flex-end**
- **center**
- **baseline**
- **stretch**

9.9 gap

Gère l'espace de la gouttière sur l'axe secondaire (permet de ne plus utiliser les margin négatifs).

Ex pour une gouttière de 20px verticale lorsque l'axe principal est "row" :

gap: 20px 0;

9.10 Order

<https://developer.mozilla.org/fr/docs/Web/CSS/order>

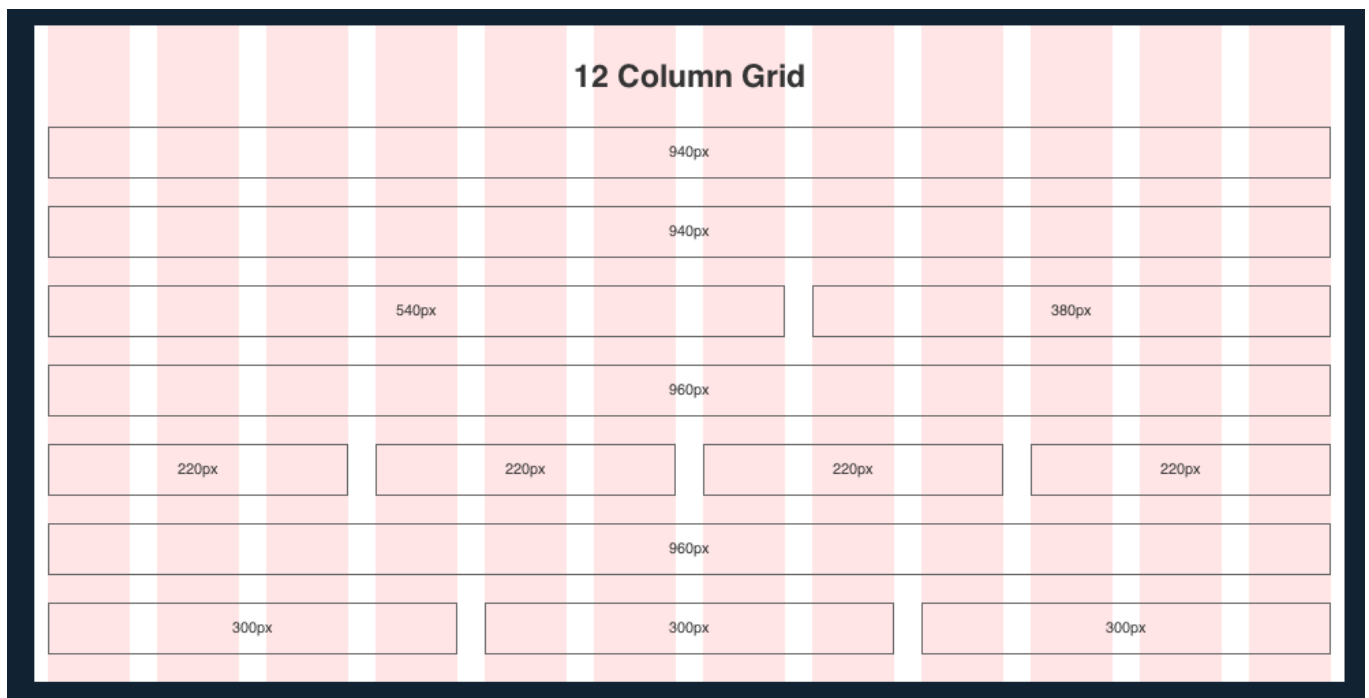
La propriété `order` définit l'ordre avec lequel on dessine les éléments d'un conteneur d'éléments flexibles ou d'une grille d'éléments. Les éléments sont appliqués dans l'ordre croissant des valeurs de `order`. Les éléments ayant la même valeur pour `order` seront appliqués dans l'ordre selon lequel ils apparaissent dans le code source du document.

9.11 Exercice 1

| | | |
|---------|---|---------|
| Texte 1 | Texte 2 qsd fsqdfsqdf qsd fsqdfqdsf | Texte 3 |
| Texte 4 | Lorem ipsum dolor sit, amet consectetur adipisicing elit. Minima ullam laudantium, veniam, dolore similique at illo fugit molestiae, porro molestias impedit maxime. Rerum eius a distinctio dolorum doloribus id iusto! 5 | Texte 6 |
| Texte 7 | Texte 8 | Texte 9 |

9.11.1.1 Trouver le code css adapté pour obtenir le résultat classique ci-dessus :

9.12 Exercice 2 : Création d'une grille 960 avec flex



10 Glyphicons avec fontello

options de configuration Ouvert

- [Rendez vous sur fontello](#)
- Choisir les icônes qui vous intéressent
- Télécharger l'archive zippée
- Dézipper dans le répertoire fontello de votre arborescence web
- Faire un "lien" vers fontello.css depuis votre page html

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

- Utiliser les mêmes class que dans le fichier demo.html.

Ex :

```
```css  
<li class="icon-right-open">
```

## Exercice

Affichez sur une page web les principales icônes des réseaux sociaux en leur donnant des couleurs qui correspondent aux couleurs des différentes marques (bleu pour facebook). Au survol, la couleur et la taille de chaque icône change.