



# MODULE HTML & CSS

Formation Développeur Web et Web Mobile

# PROGRAMME GÉNÉRAL DU MODULE

- 01 INTRODUCTION AUX LANGAGES DE BALISES
- 02 STRUCTURATION DES PAGES HTML
- 03 **MANIPULATIONS CSS**
- 04 INTRODUCTION AU SASS
- 05 BOOTSTRAP : EXEMPLE D'UN FRAMEWORK CSS



# LES SÉLECTEURS, LES RÈGLES

# PARTIE 3 – MANIPULATIONS CSS

01 **LES SÉLECTEURS, LES RÈGLES**

02 COULEURS ET MISE EN FORME DU TEXTE

03 COULEURS ET MISE EN FORME DES BOITES ET DES FONDS

04 FLEXBOX

05 GRID LAYOUT

06 MEDIA-QUERIES : RESPONSIVE DESIGN

# GÉNÉRALITÉS

## ∞ Rôle du CSS :

- Mise en page
- Appliqué des styles
- Cibler des éléments Html

## ∞ Les sélecteurs

- Permet de sélectionner les éléments html
- Il en existe des simples et des complexes

## ∞ Les propriétés CSS

- Gère les aspects des éléments sélectionnés
- Ex : color changera la couleur du texte



# SYNTAXE

∞ Sur l'exemple ci-contre :

- p : le selecteur
- color : la propriété
- blue : valeur

∞ Une déclaration est le couple propriété-valeur

∞ Chaque déclaration se termine par un point-virgule ;

∞ On place toutes les déclarations juste après le(s) sélecteur(s) dans un bloc défini par des accolades { }

```
/* Exemple : les paragraphes seront de couleur bleu avec une police sans-serif */  
  
p {  
    color: blue;  
    font-family: sans-serif;  
}
```

## PLACER LE CODE CSS : ÉLÉMENT <STYLE>

- ∞ La première façon est de placer le code dans le head du fichier html entre des balises <style>
- ∞ Fonctionne très bien si vous avez 1 ou 2 éléments à mettre en forme

```
<html lang="fr">

<head>

  <meta charset="UTF-8">
  <title>Document</title>
  <style>
    p {
      color: blue;
      font-family: sans-serif;
    }
  </style>
</head>

<body>
  <p>Lorem ipsum dolor sit amet consectetur adipiscing elit.</p>
</body>

</html>
```

## PLACER LE CODE CSS : L'ATTRIBUT STYLE

- ∞ La seconde façon est de placer le code en tant qu'attribut de l'élément à mettre en forme
- ∞ Vous en verrez énormément dans les sites, ils sont générés généralement en JavaScript

```
<p style="color: red; font-family: serif">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
```



# PLACER LE CODE CSS : LE FICHIER CSS

- ∞ La dernière façon est de placer le code css dans un fichier externe qu'on lie au fichier html par la balise <link> dans le head
- ∞ C'est la pratique la plus judicieuse pour gérer la mise en forme

```
<html lang="fr">

<head>

  <meta charset="UTF-8">

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <p>Lorem ipsum dolor sit amet adipisicing elit.</p>

</body>

</html>
```

```
body {
    background-color: lightgrey;
}

p {
    color: blue;
    font-family: sans-serif;
}
```

## BONNES PRATIQUES

- ∞ Mettre des commentaires : entre `/* */`
- ∞ Indenter correctement
- ∞ Aérer le code

```
body {  
    background-color: lightgrey;  
}  
  
/* Les paragraphes */  
p {  
    color: blue;  
    font-family: sans-serif;  
}  
  
/* Les headings */  
h1 {  
    color: red;  
    font-size: 16px;  
}
```

```
body { background-color: lightgrey;  
}p { color: blue; font-family: sans-serif;}h1{ color: red; font-size: 16px;}
```

# LES SÉLECTEURS SIMPLES

- ∞ Les sélecteurs simples correspondent au nom de la balise html
- ∞ On peut les séparer par des virgules afin d'appliquer la même règle à plusieurs éléments
- ∞ A utiliser en priorité : c'est ce qu'il y a de plus rapide pour le navigateur

```
/* Le texte en groupant les sélecteurs */
```

```
h1, h2, h3, h4, h5, h6, p {
```

```
    color: blue;
```

```
    font-family: sans-serif;
```

```
}
```

```
/* Les headings */
```

```
h1 {
```

```
    font-size: 16px;
```

```
}
```

```
/* Les paragraphes */
```

```
p {
```

```
    font-size: 12px;
```

```
}
```

## LES ATTRIBUTS ID ET CLASS

- ∞ Pour avoir plusieurs éléments HTML identiques avec des propriétés CSS différentes
- ∞ les id se notent avec un #valeurID et ils sont uniques
- ∞ les classes se notent .valeurClass
- ∞ On peut les associer
- ∞ En cas de conflit, c'est le sélecteur le plus précis qui gagne : id > class

```
<body>
  <p>Lorem ipsum.</p>
  <p id="redParagraph" class="italic">Lorem ipsum.</p>
  <p class="italic">Lorem ipsum.</p>
</body>
```

```
p {
  color: blue;
}

#redParagraph {
  color: red;
}

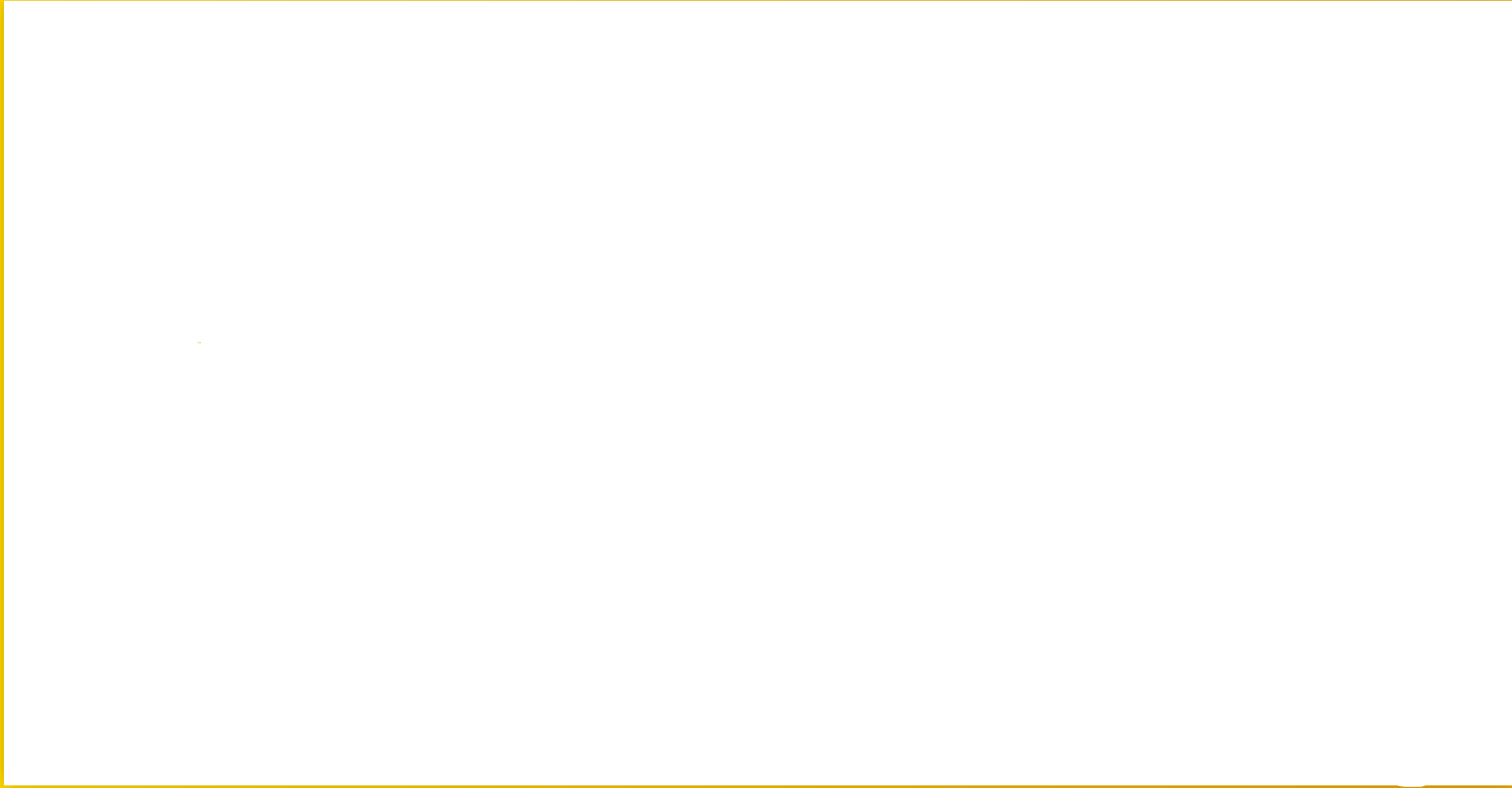
.italic {
  font-style: italic;
}
```

## L'HÉRITAGE EN CSS

- ∞ Tout élément HTML enfant va hériter, « en cascade », des styles de ses parents
- ∞ C'est l'origine du nom CSS : Cascading StyleSheet

```
body {  
    color: green;  
}  
  
.italic {  
    font-style: italic;  
}
```

```
<body>  
    <h1>Titre</h1>  
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>  
    <p class="italic">Lorem ipsum dolor sit amet consectetur adipisicing  
elit.</p>  
</body>
```



## LES SÉLECTEURS RELATIFS

- ∞ On peut sélectionner un élément par rapport à un autre
- `a b` : tous les `b` dans `a`
  - `a + b` : tous les `b` qui suivent directement un `a`
  - `a > b` : tous les `b` directement enfants de `a`
  - `a ~ b` : tous les éléments `b` qui suivent un élément `a`



– TP : slide 15 – les sélecteurs relatifs

```
/* Les paragraphes du conteneur */  
  
#conteneur p {  
  color: red;  
}  
  
/* Les paragraphes directement enfant du conteneur */  
  
#conteneur>p {  
  font-weight: bold;  
}  
  
/* Le paragraphe qui suit directement les h2 du conteneur */  
  
#conteneur h2+p {  
  background-color: blue;  
}
```

# LES SÉLECTEURS D'ATTRIBUTS

∞ A[attribut="valeur"] » va sélectionner tous les éléments A possédant un attribut en particulier avec une valeur précise.

🧐 – TP : **slide16 les sélecteurs d'attributs** :

- Mettre les liens avec l'attribut target présent en lightgreen et police grasse
- Celui avec l'attribut target="\_top" avec une taille de police à 3rem

```
/* Tous les liens avec l'attribut target présent */  
a[target] {  
    color: pink;  
    font-weight: bold;  
}  
  
/* Tous les liens avec l'attribut target="_top" */  
a[target="_top"] {  
    font-size: 3rem;  
}
```



## LES PSEUDO-CLASSES

∞ Une **pseudo-classe** est un mot-clé qui peut être ajouté à un sélecteur et qui cherche un état de l'élément,

∞ On a par exemple :

- :hover => au survol de la souris
- :checked => pour les formulaires
- :visited => lien visité
- :first-child => premier enfant
- :last-child => dernier enfant
- :nth-child(n) => le n<sup>ième</sup> enfant

```
/* Tous les liens */
a {
    color: pink;
}

/* Au survol d'un lien */
a:hover {
    color: purple;
    font-size: 1.5em;
}

/* Un lien déjà visité */
a:visited {
    color: lightgreen;
}
```

```
<p>Un <a href="https://www.humanbooster.com" target="_blank">lien</a></p>
```

```
<p>Un <a href="https://www.gregdesplaces.com" target="_blank">autre lien</a></p>
```

## LES 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,
- ∞ On a par exemple :
  - `::first-line` => la 1<sup>ère</sup> ligne (éléments de type block)
  - `::first-letter` => la 1<sup>ère</sup> lettre (éléments de type block)
  - `::selection` => partie sélectionnée par l'utilisateur
  - `::before` => permet d'insérer du contenu avant l'élément
  - `::after` => permet d'insérer du contenu après l'élément
- ∞ NB : les pseudo-éléments `::after` et `::before` sont utilisés généralement avec la propriété css **content**
- ∞ 🧠 => Entraînez vous ici : <https://flukeout.github.io/>



## — TP : PSEUDO-CLASSES ET PSEUDO-ÉLÉMENTS

∞ Faire le TP slide 19 :

- Mettre tous les liens en rose
- Au survol des liens, augmentez la police à 1.5rem et passez la couleur à jaune
- Mettre les liens déjà visités en rose
- Au survol d'un paragraphe, mettre le fond en rouge
- Faire une lettrine en augmentant la police de la première lettre des paragraphes à 1.8 rem
- Faire en sorte que les selections de l'utilisateur soient surlignées en jaune (vous pouvez commenter le survol du paragraphe en rouge)
- BONUS : Ajouter l'image [https://d1nhio0ox7pgb.cloudfront.net/\\_img/g\\_collection\\_png/standard/16x16/ok.png](https://d1nhio0ox7pgb.cloudfront.net/_img/g_collection_png/standard/16x16/ok.png) avant chaque titre de niveau en antune marge à droite de 6px

## PRIORITÉ ET ORDRE

- ∞ S'il y a plusieurs règles, le style appliqué est celui qui est le plus proche de l'élément,
- ∞ S'il y a plusieurs codes CSS, la priorité fonctionne comme cela :
  1. attribut style
  2. CSS dans balise <style>
  3. fichier CSS externe

```
<h1>Titre</h1>
<h2>Sous-titre</h2>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
<p>Lorem ipsum dolor <b>sit amet</b> consectetur adipisicing elit.</p>
```

```
body {
    color: green;
}

p {
    color: blue;
}

b {
    color: red;
}
```

# ELÉMENTS HTML BLOCK ET INLINE

## ∞ Block

- Commencent sur une nouvelle ligne
- Prennent tout la largeur de la page
- Les plus courants :
  - p
  - headings
  - listes
  - form
  - div : section, nav, footer, article

## ∞ inline

- S'insèrent dans la ligne actuelle
- Largeur nécessaire
- Les plus courants :
  - a
  - b, u
  - strong, em
  - img
  - span

## LES BALISES <DIV> ET <SPAN>

- ∞ Respectivement de type block et inline
- ∞ On leur attribue souvent des classes pour les cibler facilement


```
.div-paragraph-aqua {  
    background-color: aqua;  
}  
  
.red-bold {  
    color: red;  
    font-weight: bold;  
}
```

```
<div class="div-paragraph-aqua">  
    <p>Lorem ipsum <span class="red-bold">dolor    sit</span> amet consectetur adipisicing elit.</p>  
    <p>Lorem ipsum dolor sit amet consectetur    <span class="red-bold">adipisicing</span>    elit.</p>  
</div>
```

## – TP 23 : INLINE ET BLOCK

∞ NB : pour une bordure :

```
p {  
    border: 1px solid red;  
}  
  
b {  
    border: 1px dashed green;  
}
```

∞  – Test : ajouter la déclaration « display:block » au texte en gras et remarquez ce qui se passe !



# COULEURS ET MISE EN FORME DU TEXTE



## PARTIE 3 – MANIPULATIONS CSS

01 LES SÉLECTEURS, LES RÈGLES

02 **COULEURS ET MISE EN FORME DU TEXTE**

03 COULEURS ET MISE EN FORME DES BOITES ET DES FONDS

04 FLEXBOX

05 GRID LAYOUT

06 MEDIA-QUERIES : RESPONSIVE DESIGN

## PROPRIÉTÉ FONT-FAMILY

- ∞ Défini la police du texte
- ∞ ⚠ : les ordinateurs n'ont pas les mêmes polices installés, du coup on peut en lister plusieurs, séparées par des virgules
- ∞ Si le nom a des espaces, on l'écrit entre guillemets (ou apostrophes)
- ∞ Il existe des polices qui sont lues par tous les navigateurs : les **Web Safe Fonts** :

```
body {  
    font-family: 'Times New Roman', Times, serif;  
}  
  
h1, h2, h3, h4, h5, h6 {  
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;  
}
```

## UTILISER D'AUTRES POLICES : @FONT-FACE

- ∞ On peut ajouter d'autres polices à partir d'un dossier
- ∞ On utilisera alors un bloc @font-face
- ∞ Puis on la nomme dans la liste des polices en n'oubliant pas de mettre un style générique (serif, sans-serif, cursive, etc.)
- ∞ ⚠ : il faut faire attention aux copyrights des polices

### 🤖 – TP 27 : @font-face

```
@font-face {  
    font-family: "Ma Super Police";  
    src: url("font/MaSuperPolice.ttf");  
}  
  
span {  
    font-family: 'Ma Super Police', monospace;  
    font-size: 60px;  
}
```

# GOOGLE FONT

- ∞ Afin d'éviter d'écrire les @font-face, il existe des services en ligne
- ∞ On ajoute un <link> dans l'en-tête et zou la police est disponible
- ∞ La plupart de ces services services sont payants (ex : typekit, Cloud.typography), mais pas Google Font

## – TP 28 : Google font

```
<head>

  <meta charset="UTF-8">

  <title>Document</title>

  <link href="https://fonts.googleapis.com/css?fami
ly=Gloria+Hallelujah" rel="stylesheet">

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>Titre magnifique</h1>

  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. </p>

</body>
```

```
h1 {

  font-family: 'Gloria Hallelujah', cursive;

  font-size: 60px;

}
```

# PROPRIÉTÉ FONT-SIZE

∞ Change la taille du texte

∞ On utilise des valeurs absolues :

- les pixels : **px**
- les points : **pt**

∞ Ou des valeurs relatives

- les pourcentages : **%** (proportionnel à la taille du parent (ou du navigateur si le parent n'est pas spécifié))
- les ems : **em** (idem que %)
- les rems : **rem** (proportionnel à la taille de l'élément racine : <html> ou <body>) => le top pour l'accessibilité

∞ Ou des mots-clés basés sur la taille par défaut de l'utilisateur :

- xx-small, x-small, small, medium, large, x-large, xx-large



## PROPRIÉTÉ FONT-STYLE

∞ Force le style de police :

- normal (par défaut) ;
- italic (italique) ;
- oblique (penché) ;
- inherit (l'élément ciblé hérite du style de son élément parent).

## PROPRIÉTÉ FONT-WEIGHT

- ∞ Défini l'épaisseur d'une police
  - normal (valeur par défaut) ;
  - lighter (la police sera plus fine) ;
  - bold (la police sera plus épaisse) ;
  - bolder (la police sera très épaisse) ;
  - une centaine entre 100 (police très fine) et 900 (police très épaisse). 400 correspond à la valeur « normal » et 700 à « bold » ;
  - inherit (l'élément hérite du style de son parent) ;
  - initial (définit la propriété sur sa valeur d'origine).

## LA COULEUR DU TEXTE : PROPRIÉTÉ COLOR

∞ Elle prend différentes valeurs

- Un nom de couleur (green, blue, etc.) ;
- Une valeur hexadécimale (#AA8811, #FFFFFF.) ;
- Une valeur RGB ou RGBa
  - Les valeur de couleur Red, Green, Blue et alpha (la transparence)
  - Le valeur des couleurs vont de 0 à 255 : 0 est aucune couleur et 255 une teinte absolue
  - a, l'opacité va de 0 à 1 => 0 : transparent et 1 : opaque
  - Donc *color : rgba(255,0,0,1);* est du rouge pur



## LES PROPRIÉTÉS TEXT-...

Propriété	définition	valeurs
<b>text-align</b>	Alignement	left, right, center, justify, inherit
<b>text-transform</b>	Transforme la casse	Lowercase, Uppercase, Capitalize, Inherit, none
<b>text-decoration</b>	Décoration du texte	Underline, Overline, line-through, inherit, initial, none
<b>text-indent</b>	Indentation du texte	Une valeur en em, rem, %, px (positive ou négative)
<b>text-shadow</b>	Ombre du texte	4 valeurs en px : projection horizontal, vertical, rayon du flou, couleur

## AUTRES PROPRIÉTÉS

- ∞ line-height : la hauteur de la ligne
- ∞ letter-spacing : l'écartement entre les caractères
- ∞ word-spacing : l'écartement entre les mots

## PARTIE 3 – MANIPULATIONS CSS



### MISE EN FORME DES BOITES

## PARTIE 3 – MANIPULATIONS CSS

01 LES SÉLECTEURS, LES RÈGLES

02 COULEURS ET MISE EN FORME DU TEXTE

03 **COULEURS ET MISE EN FORME DES BOITES ET DES FONDS**

04 FLEXBOX

05 GRID LAYOUT

06 MEDIA-QUERIES : RESPONSIVE DESIGN

## PROPRIÉTÉS WIDTH ET HEIGHT

∞ Respectivement largeur et hauteur

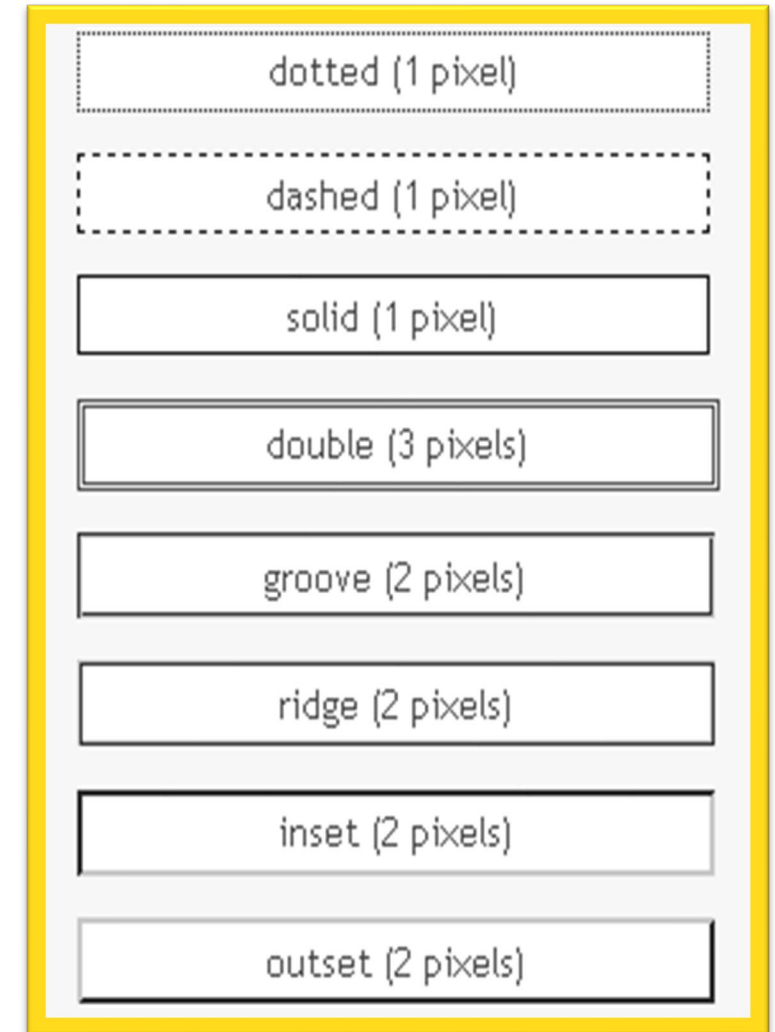
∞ Peuvent prendre des valeurs :

- en px, em, etc.
- la valeur auto (calculée par le navigateur en fonction du contenu)

∞ ⚠ : Il faut faire attention à ce que l'élément parent ne soit pas plus petit que les enfants

## LES BORDURES

- ∞ La propriété **border** qui prend 3 paramètres :
  - L'épaisseur (px, rem, em, etc.) ou **border-width**
  - Le style : voir ci-contre ou **border-style**
  - La couleur ou **border-color**
- ∞ Et chaque côté de manière isolée :
  - **border-top**
  - **border-right**
  - **border-bottom**
  - **border-left**
- ∞ On peut aussi arrondir les angles avec la propriété **border-radius** :
  - prend une valeur de taille (em, rem, px...) :
  - et aussi : **border-bottom-right-radius**, etc.



## LES MARGES

∞ Aussi bien pour **margin** que pour **padding**, on va pouvoir gérer chaque côté

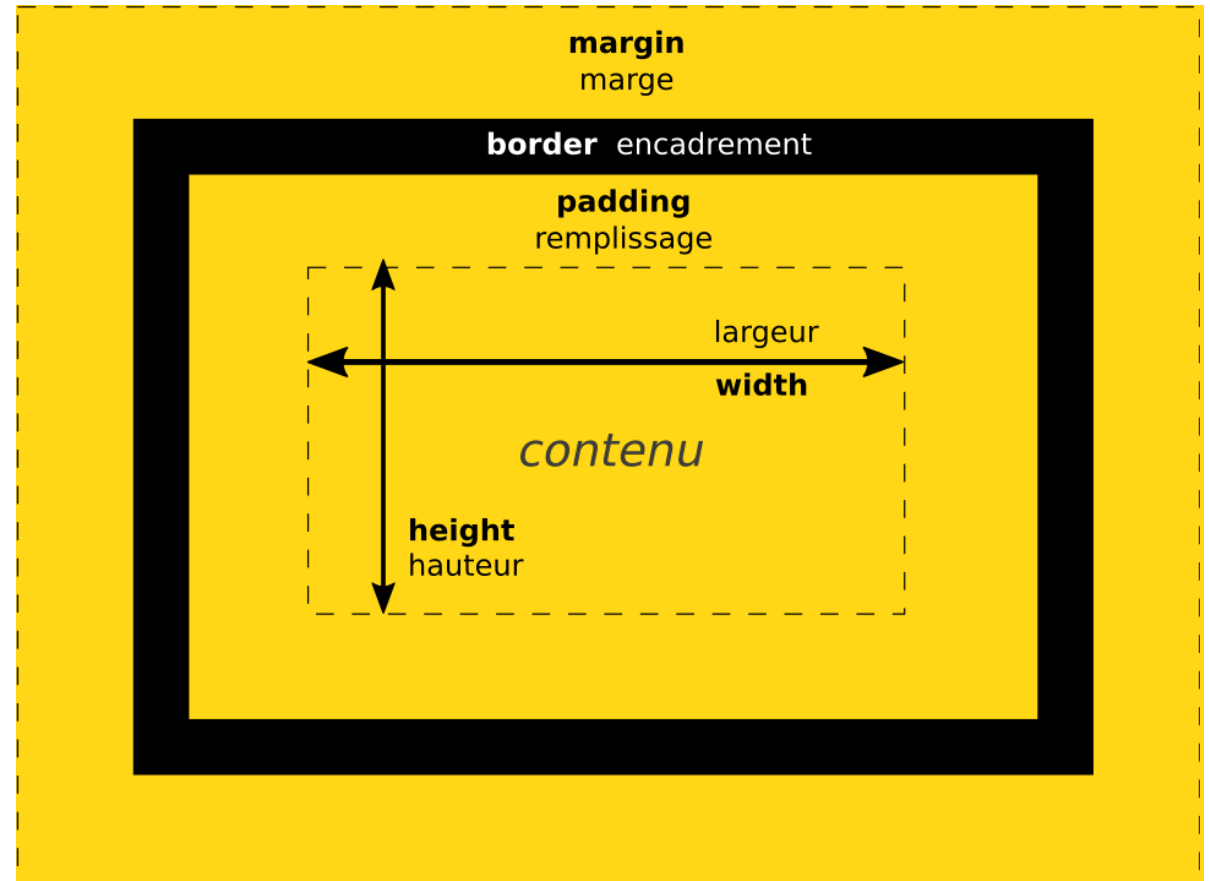
- **margin-top**
- **margin-right**
- **margin-bottom**
- **margin-left**

∞ Et aussi en une seule déclaration :

- **margin : [top] [right] [bottom] [left]**
- en partant du haut dans le sens trigonométrique (des aiguilles d'une montre)

# LE MODÈLE DE BOITES

- ∞ Chaque élément html est une boîte qui comprend :
- Le contenu
  - Une marge intérieure : **padding**
  - Une bordure : **border**
  - Une marge extérieure : **margin**







## TP – SLIDE 42 : FAIRE UNE BOITE

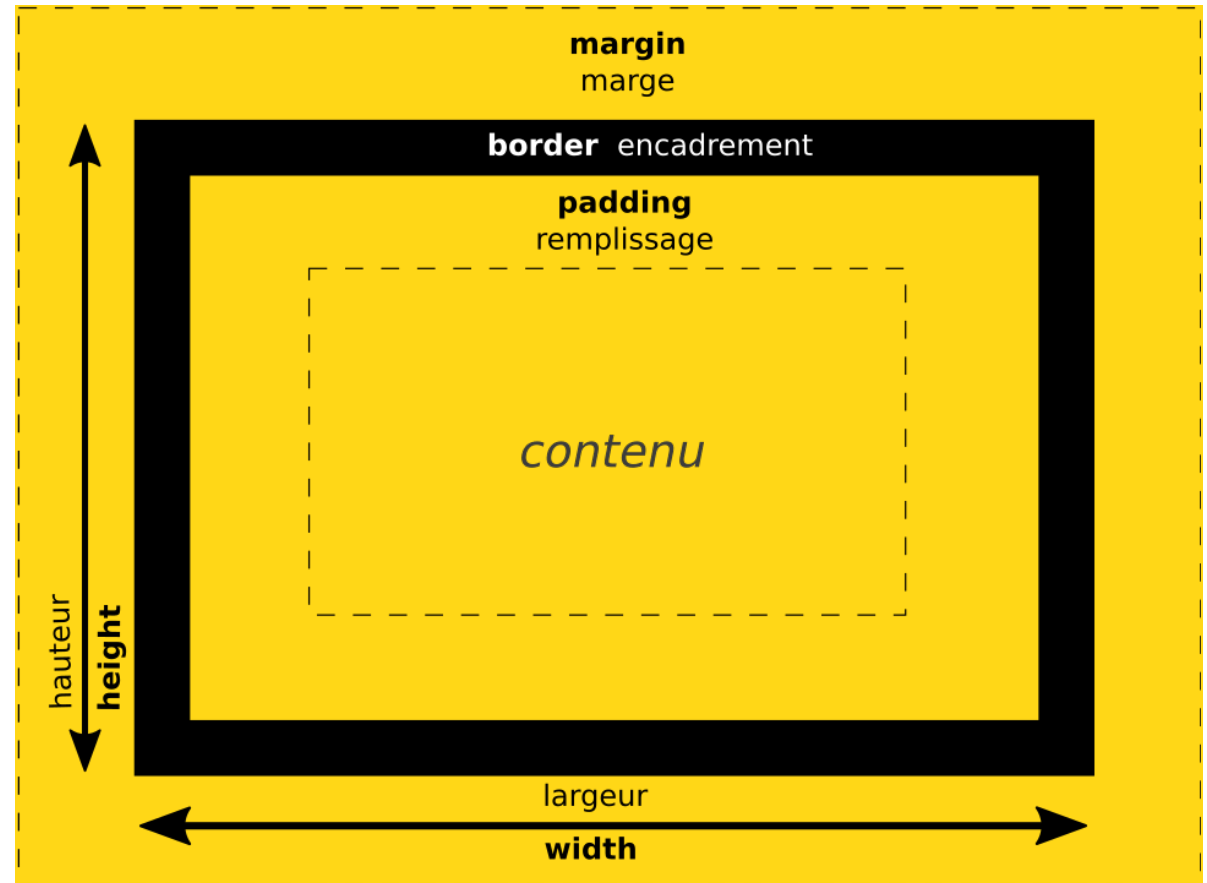
- ∞ Créer une div avec un titre et un paragraphe en lorem
  - de largeur (width) 500px
  - de couleur de fond (background-color) au choix
  - avec une bordure (border) 15px, de type au choix, de couleur au choix
  - avec une marge de 50px;

## PROPRIÉTÉ : BOX-SIZING: BORDER-BOX;

∞ On peut modifier le modèle de boîte par défaut avec la propriété **box-sizing**, qui peut prendre ces valeurs :

- **content-box** : la taille du contenu
- **border-box** : la taille jusqu'à la bordure

🤖 TP slide 43 – Box-sizing



## LES OMBRES : LE PROPRIÉTÉ BOX-SHADOW

On la définit grâce :

- ∞ À deux, trois ou quatre valeurs de longueur (<length>) :
  - Avec deux valeurs, celles-ci sont respectivement considérées comme les coordonnées de décalage de l'ombre : <offset-x> et <offset-y>
  - Si une troisième valeur est fournie, celle-ci correspondra au rayon du flou : <blur-radius>
  - Si une quatrième valeur est fournie, celle-ci correspondra au rayon d'étalement : <spread-radius>.
- ∞ Au mot-clé optionnel **inset**
- ∞ À une valeur de couleur (<color>) optionnelle.

## LA PROPRIÉTÉ DISPLAY

- ∞ Nous avons vu qu'il existait des éléments **block** et **inline**, avec la propriété `display`, on peut forcer des éléments à prendre ces valeurs
- ∞ On peut aussi forcer un élément à ne pas apparaître avec la valeur **`display: none;`**
- 🤖 **Voir slide 45 – display – exemple**

```
<div>

  <h1>La propriété display</h1>

  <p>Ceci est un paragraphe en display inline qui <span>
contient un span en display block</span> du coup ils
agissent de manière fort étrange !</p>

</div>
```

```
div {
  background-color: aqua;
}

p {
  background-color: green;
  display: inline;
}

span {
  background-color: red;
  display: block;
}
```

## DISPLAY : INLINE-BLOCK

- ∞ On peut créer un 3<sup>e</sup> type : **inline-block**
- ∞ Il permet de mettre des éléments côte à côte et de gérer précisément leur taille
- ∞ On s'en sert surtout pour les listes
- 🕒 Voir slide 46 – display inline-block – exemple

```
<ul>
  <li class="item1">item 1</li>
  <li class="item2">item 2</li>
  <li class="item3">item 3</li>
  <li class="item4">item 4</li>
</ul>
```

```
li {
    display: inline-block;
    background-color: aqua;
    padding: 1.3em;
    border: 0.2em solid blue;
}

.item1 {
    width: 125px;
}

.item2 {
    width: 100px;
}

.item3 {
    width: 150px;
}

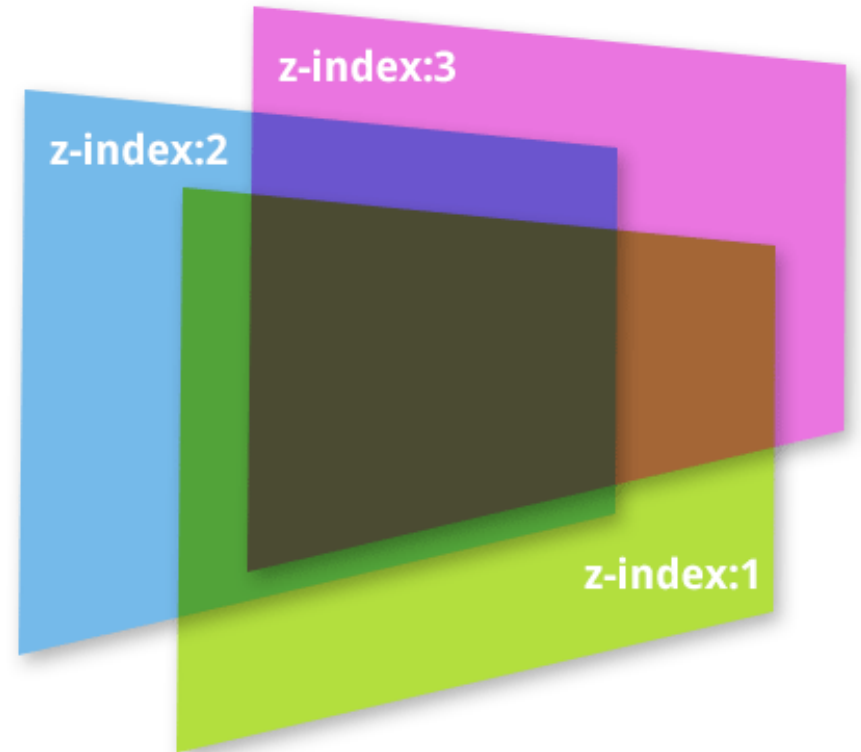
.item4 {
    width: 75px;
}
```

## POSITIONNER LES BOITES

- ∞ On utilise la propriété **position** couplé avec les propriétés **top**, **right**, **bottom**, **left**.
- ∞ **position** à 5 valeurs possibles:
  - **static**: par défaut, on n'utilisera pas **top**, **right**, **bottom**, **left**.
  - **relative**: on le décale par rapport à sa position normale sans affecter les autres éléments (on pourra utiliser **top**, **right**, **bottom**, **left**) => on l'utilisera plutôt pour utiliser la position absolue des enfants!
  - **absolute**: relative par rapport à son parent le plus proche qui en déclaré en position relative, il n'est plus dans le flux normal !
  - **fixed**: fixe par rapport à l'écran (sauf s'il existe une propriété transform d'un parent 😊), même en scrollant => utile pour les barres d'info par exemple
  - **sticky**: la position initial est comme static puis lors du scroll, il devient fixed jusqu'à que le scroll quitte le parent, pfffiou
- ∞ Si c'est pas clair (en même temps c'est normal 🤔) regardez cette vidéo :  
<https://www.youtube.com/watch?v=YGm1T7bVmJQ>
- 👤 => Voir slide 47 – position – exemple

## Z-INDEX

- ∞ Lorsque l'on a plusieurs boîtes en fixed ou absolute qui s'empilent, elles forment des sorte de calques.
- ∞ La propriété z-index permet de positionner en profondeur les différents éléments



## LE FLOTTEMENT : FLOAT ET CLEAR

- ∞ La propriété **float** indique qu'un élément doit être retiré du flux normal et doit être placé sur le côté droit ou sur le côté gauche de son conteneur.
- ∞ Voici ces valeurs :
  - *left*
  - *right*
  - *none*
- ∞ Les éléments suivants vont se positionner à côté : On annule ce comportement par la propriété **clear** qui prendra les valeurs *right*, *left* et *both*
- 👤 => Voir slide 49 – float et clear – exemple



## L'OVERFLOW

- ∞ Si un élément flottant est plus grand que son conteneur, il va déborder verticalement.
- ∞ On peut cacher ce qui dépasse avec **overflow** :
  - *Visible* : valeur par défaut (rien ne sera coupé)
  - *Hidden* : ce qui dépasse sera coupé
  - *Scroll* : coupe ce qui dépasse et ajoute une barre de défilement afin d'avoir accès à tout le contenu
- 🧐 => **Voir slide 50 – overflow – exemple**

## CENTRER EN CSS : 2 TECHNIQUES (OUTRE FLEXBOX)

∞ Centrer en mettant les marges droite et gauche auto

```
margin: 15px auto;
```

∞ Centrer en utilisant position absolute et mettant des pourcentages identiques en left et right:

```
position: absolute;  
text-align: center;  
right: 20%;  
left: 20%;
```

🧐 => Voir slide 51 – centrer en css – exemple

## LE BACKGROUND

- ∞ Pour choisir la couleur : **background-color**
- ∞ Pour mettre des images : **background-image**
- ∞ La position des images : **background-position**
- ∞ La répétition des images : **background-repeat**
- ∞ La taille des images de fond : **background-size**
- ∞ Le défilement : **background-attachment**
- ∞ `background : url([chemin]) [position-x] [position-y]`



## TP SLIDE 53 – SPRITES

- ∞ Télécharger un sprite social sur internet :  
<http://copperheadconsulting.com/wp-content/uploads/2015/02/spice-social-gadget-sprite.png>
- ∞ Ajouter des liens vers différents réseaux sociaux et faire en sorte :
  - Que le lien affiche une image grisée
  - Que l'image passent en couleur au survol



# FLEXBOX



## PARTIE 3 – MANIPULATIONS CSS

01 LES SÉLECTEURS, LES RÈGLES

02 COULEURS ET MISE EN FORME DU TEXTE

03 COULEURS ET MISE EN FORME DES BOITES ET DES FONDS

04 **FLEXBOX**

05 GRID LAYOUT

06 MEDIA-QUERIES : RESPONSIVE DESIGN

## UN AUTRE MODE DE MISE EN PAGE

∞ On l'active sur l'élément englobant en faisant **display:flex**

∞ Le FlexBox va permettre de gérer :

- La direction des éléments (ligne ou colonne, avec retour à la ligne ou non)
- L'alignement des éléments (vertical et horizontal) et leur répartition
- L'ordre des éléments
- La place prise par les éléments en fonction de l'espace disponible

## PRINCIPE

- ∞ Créer un conteneur et le mettre en flex
- ∞ Ca va agir sur les éléments enfants de ce conteneur
- ∞ Par défaut, il seront comme en inline-block



```
<div class="social">  
  <div class="facebook"></div>  
  <div class="twitter"></div>  
  <div class="youtube"></div>  
  <div class="linkedin"></div>  
</div>
```

```
.social {  
  display: flex;  
}  
  
.social div {  
  display: block; /*Pour que les liens réagissent en block*/  
  width: 84px;  
  height: 84px;  
  margin: 5px;  
  box-shadow: 5px 5px 5px lightgrey;  
}
```

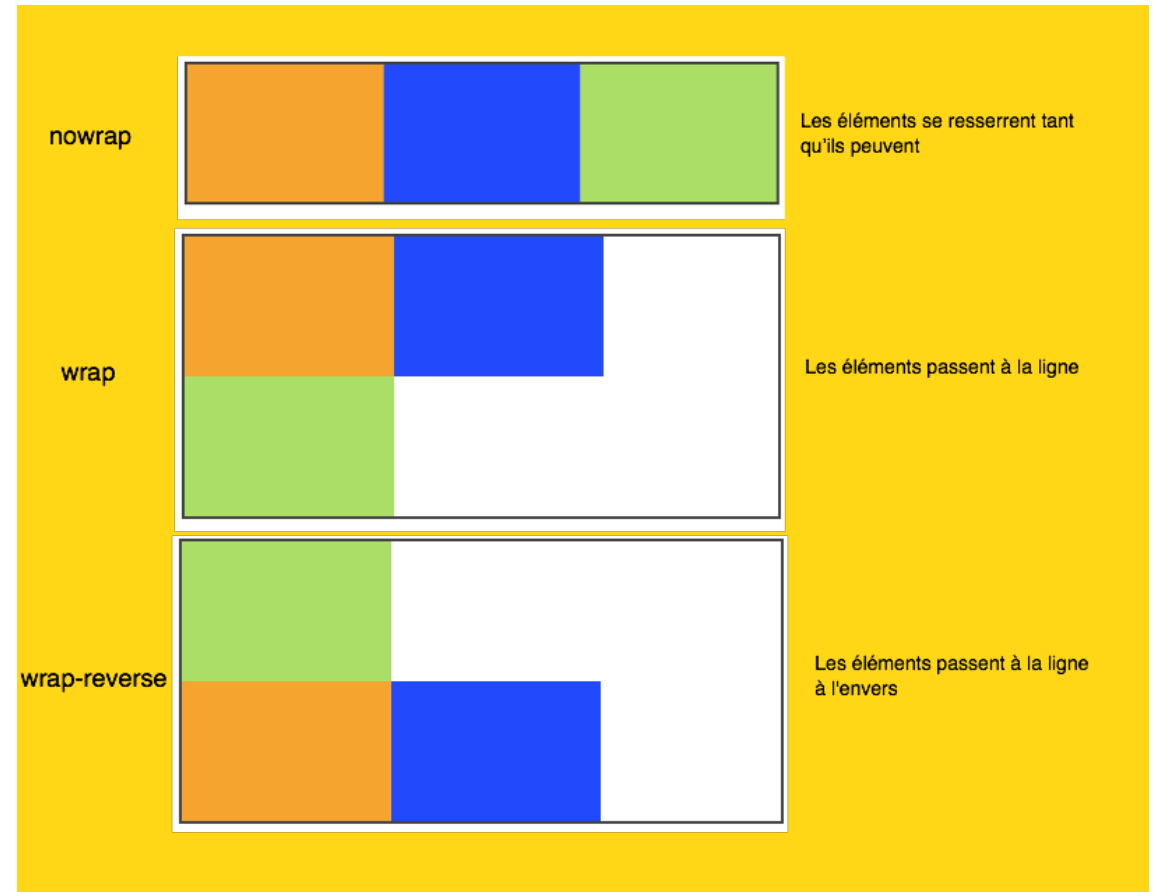


## FLEX-DIRECTION : DIRECTION ET SENS

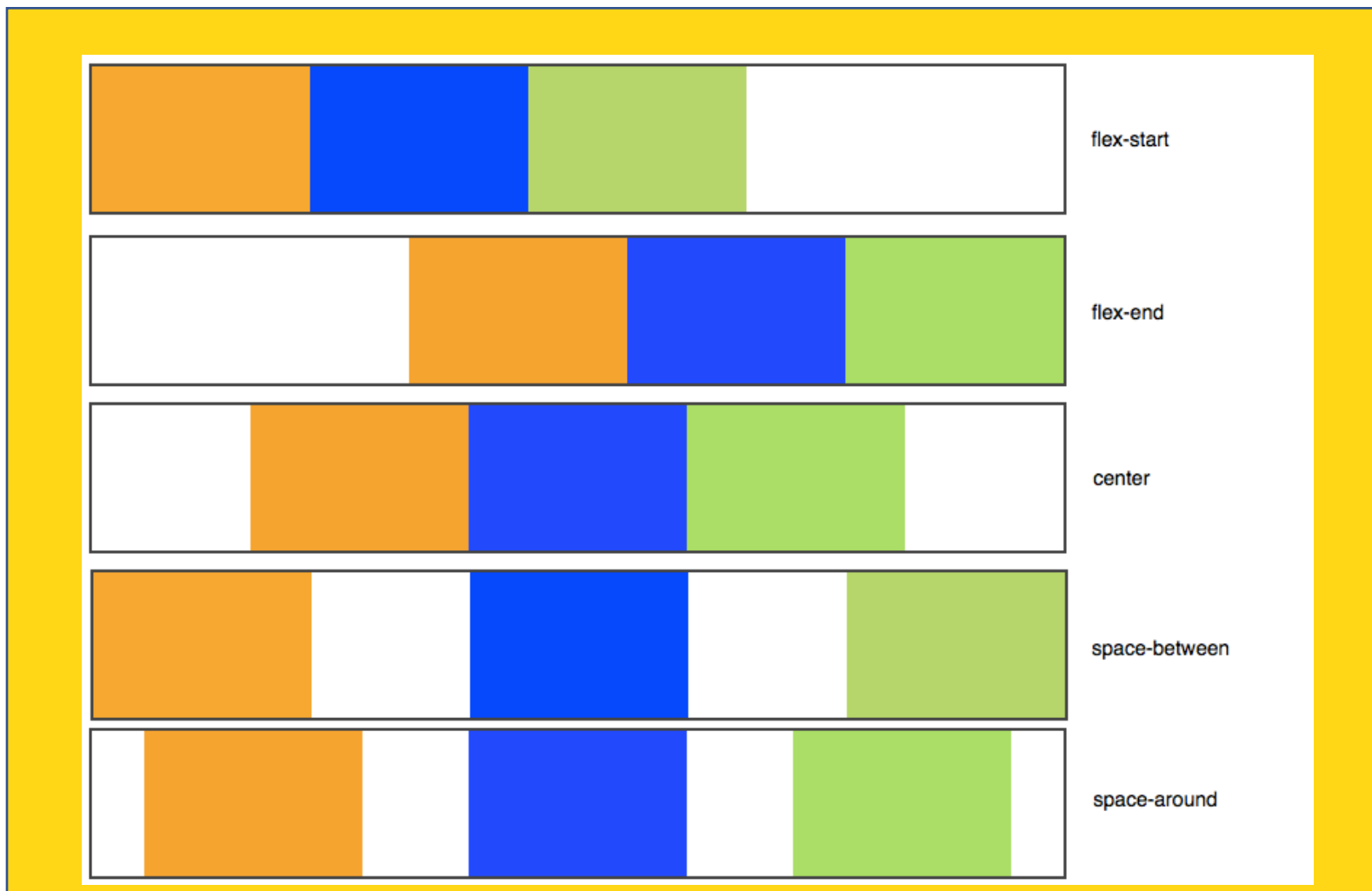
- ∞ la propriété **flex-direction** va positionner les éléments verticalement ou encore les inverser.
- ∞ Il peut prendre les valeurs suivantes :
  - *row*: organisés sur une ligne (par défaut)
  - *column*: organisés sur une colonne
  - *row-reverse*: organisés sur une ligne, mais en ordre inversé
  - *column-reverse*: organisés sur une colonne, mais en ordre inversé

# FLEX-WRAP : LE RETOUR À LA LIGNE

- ∞ Si les blocs sont trop grands pour le conteneur, on va utiliser la propriété **flex-wrap**, avec :
- **nowrap** : pas de retour à la ligne (par défaut)
  - **wrap** : les éléments vont à la ligne lorsqu'il n'y a plus la place
  - **wrap-reverse** : les éléments vont à la ligne lorsqu'il n'y a plus la place en sens inverse



# JUSTIFY-CONTENT : L'ALIGNEMENT PRINCIPAL



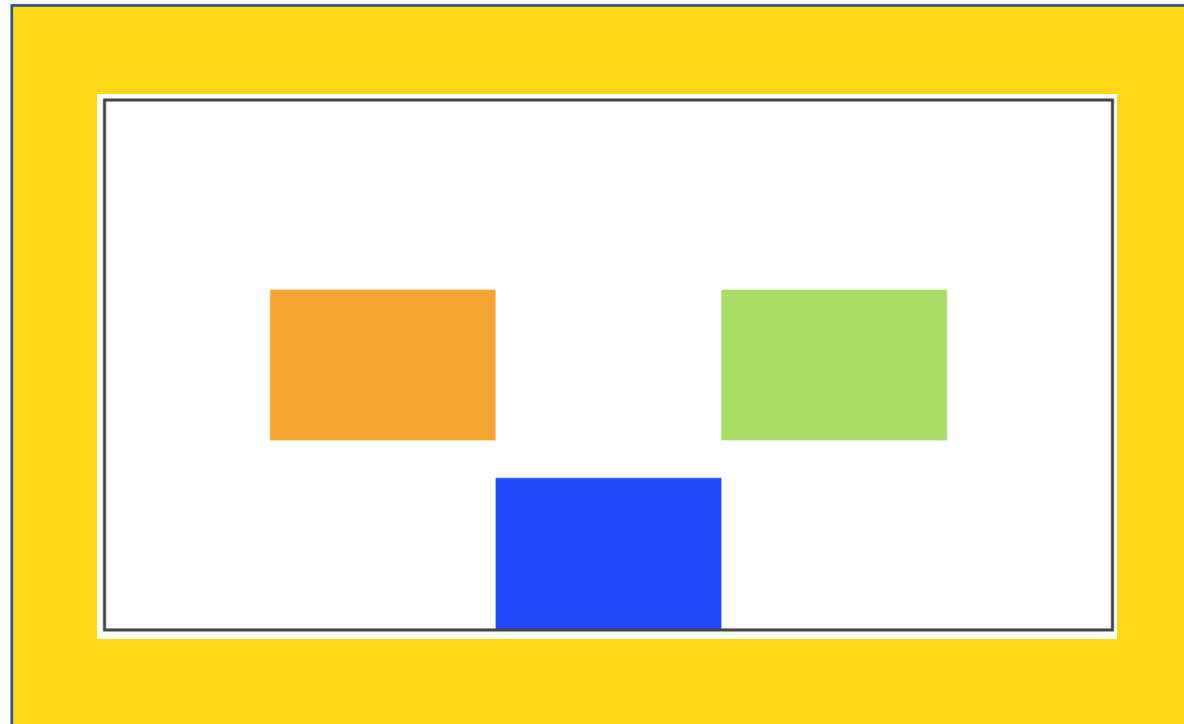
## ALIGN-ITEMS : ALIGNEMENT SECONDAIRE

∞ **align-items** peut prendre ces valeurs :

- *stretch*: les éléments sont étirés sur tout l'axe (valeur par défaut)
- *flex-start*: alignés au début
- *flex-end*: alignés à la fin
- *center*: alignés au centre
- *baseline*: alignés sur la ligne de base (semblable à flex-start)

## ALIGN-SELF : ALIGNEMENT D'UN SEUL ÉLÉMENT

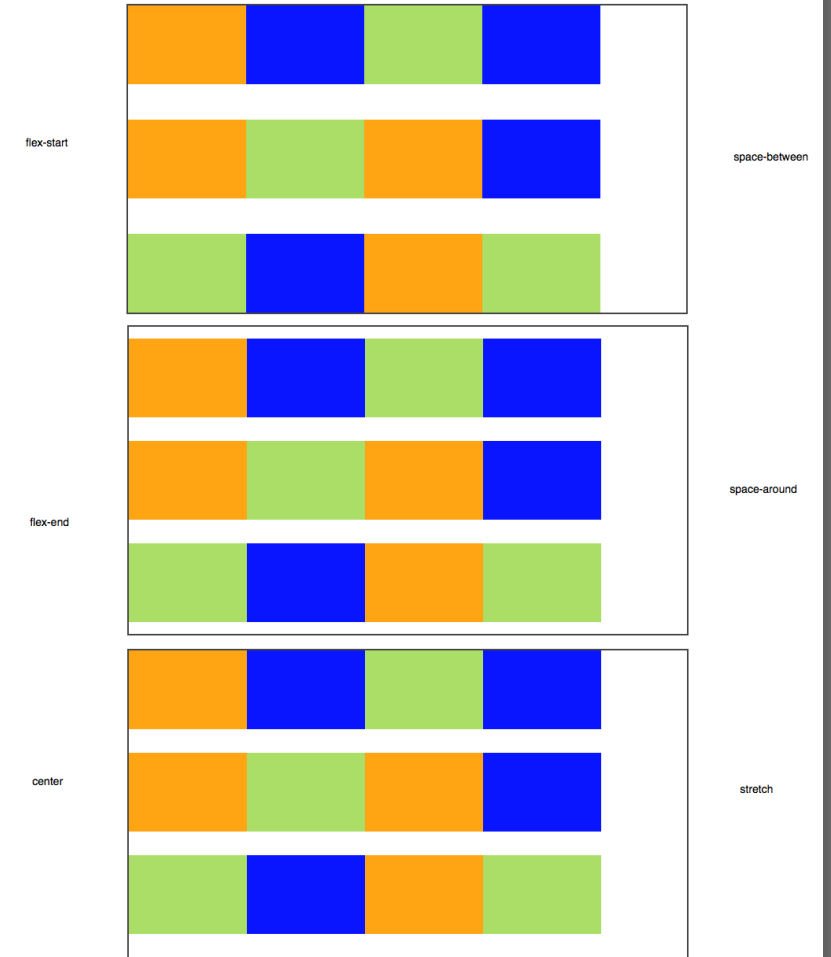
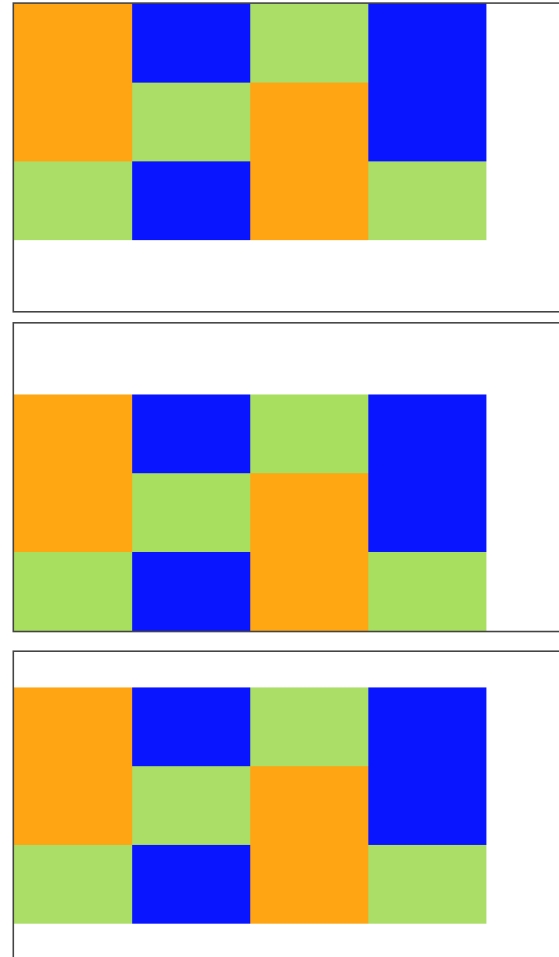
- ∞ On va agir sur un élément en particulier
- ∞ Il prend les mêmes valeurs que align-items



# ALIGN-CONTENT : RÉPARTITION DE PLUSIEURS LIGNES

∞ Prend les mêmes valeurs que **justify-content**, sauf qu'il va gérer la répartition des différentes lignes

- *flex-start*
- *flex-end*
- *center*
- *space-between*
- *space-around*
- *space-evenly*
- *stretch* (par défaut)



## PROPRIÉTÉS ORDER ET FLEX

∞ **order** prend un nombre et permet de choisir l'ordre de l'élément



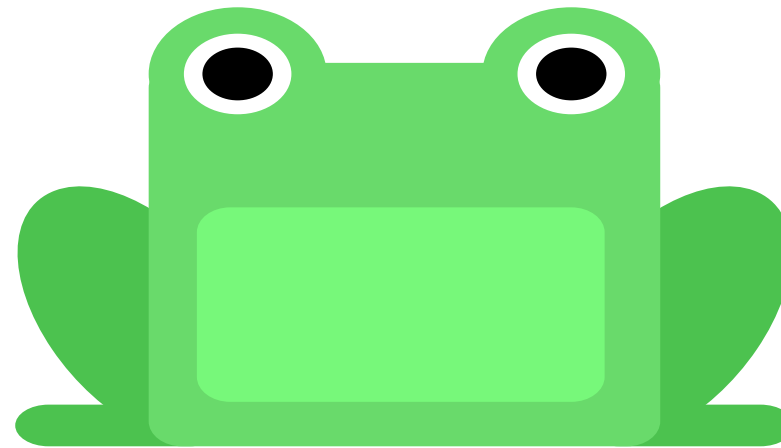
∞ **flex** va gérer sa taille, sa capacité à grossir ou à se retrécir



## JEU : AIDE FROGGY

∞ Se connecter ici : <https://flexboxfroggy.com/#fr>

∞ Et zou aider froggy !







## GRID LAYOUT

## PARTIE 3 – MANIPULATIONS CSS

01 LES SÉLECTEURS, LES RÈGLES

02 COULEURS ET MISE EN FORME DU TEXTE

03 COULEURS ET MISE EN FORME DES BOITES ET DES FONDS

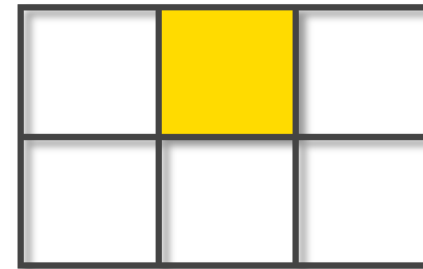
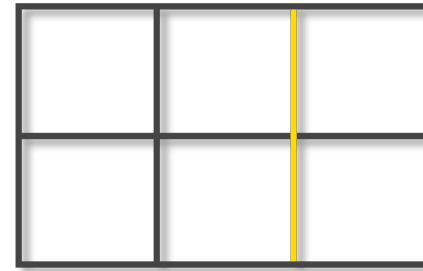
04 FLEXBOX

05 **GRID LAYOUT**

06 MEDIA-QUERIES : RESPONSIVE DESIGN

## LA GRILLE : LA VALEUR GRID DE DISPLAY

- ∞ Comme pour FlexBox, on crée un conteneur qui prendra la propriété **display:grid**
- ∞ Les enfants directes sont les items de la grille, ils vont se définir par rapport :
  - aux lignes de la grille : horizontale & vertical
  - aux pistes de la grille (grid track)
  - aux cellules de la grille
  - aux plages de cellules de la grille



## LES PISTES DE GRILLE

∞ On ajoute au parent en `display:grid` :

- **grid-template-columns**
- **grid-template-rows**

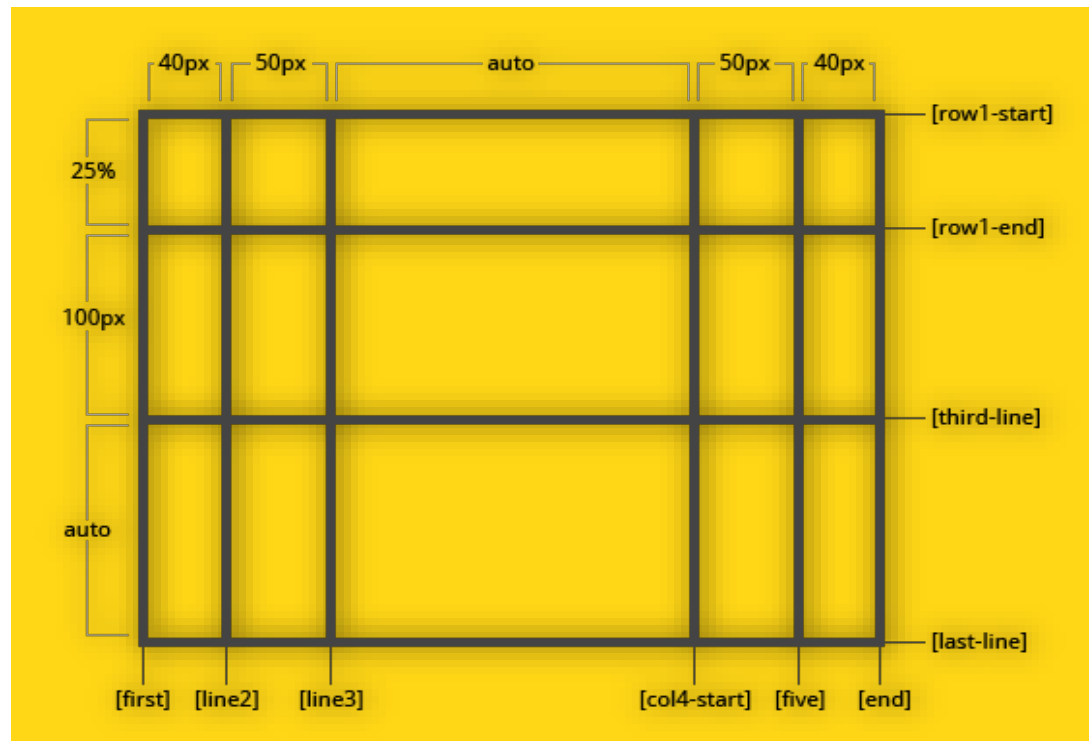
∞ Les deux fonctionnent de la même manière : on définit leurs dimensions

- Soit en unité de mesure : **15px 60px 12px**
- Soit en fraction de l'espace disponible : **1fr 3fr 2fr**
- Lorsque des tailles se répètent, on utilise **repeat(4, 1fr)**

∞ On pourra nommer les lignes

## LES PISTES DE GRILLES EXEMPLE

```
.grid {  
  
  grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];  
  
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];  
  
}
```



## GRID-TEMPLATE-AREAS

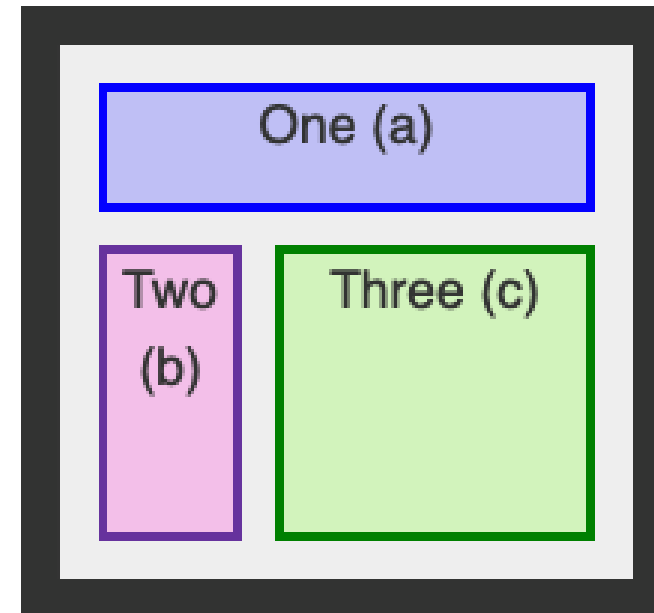
∞ Cette propriété permet de définir des zones de grille nommées.

```
grid-template-areas:
```

```
"a a a"
```

```
"b c c"
```

```
"b c c";
```



## GRID-TEMPLATE-AREAS

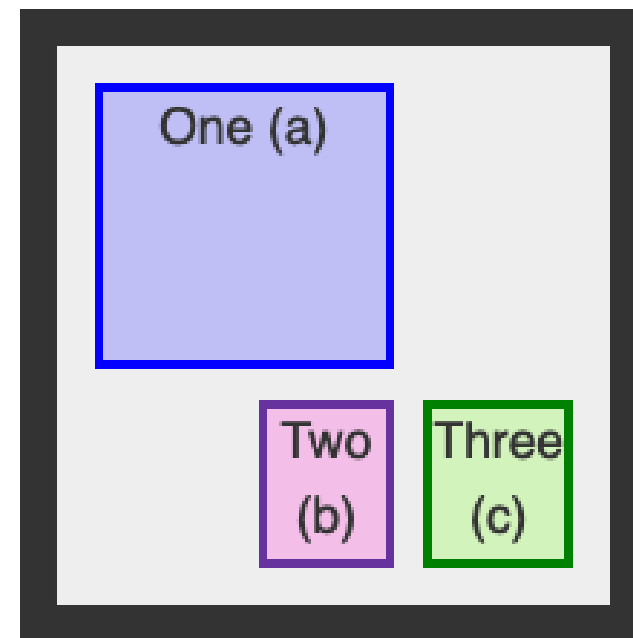
∞ Le point permet de définir une cellule vide

```
grid-template-areas:
```

```
"a a ."
```

```
"a a ."
```

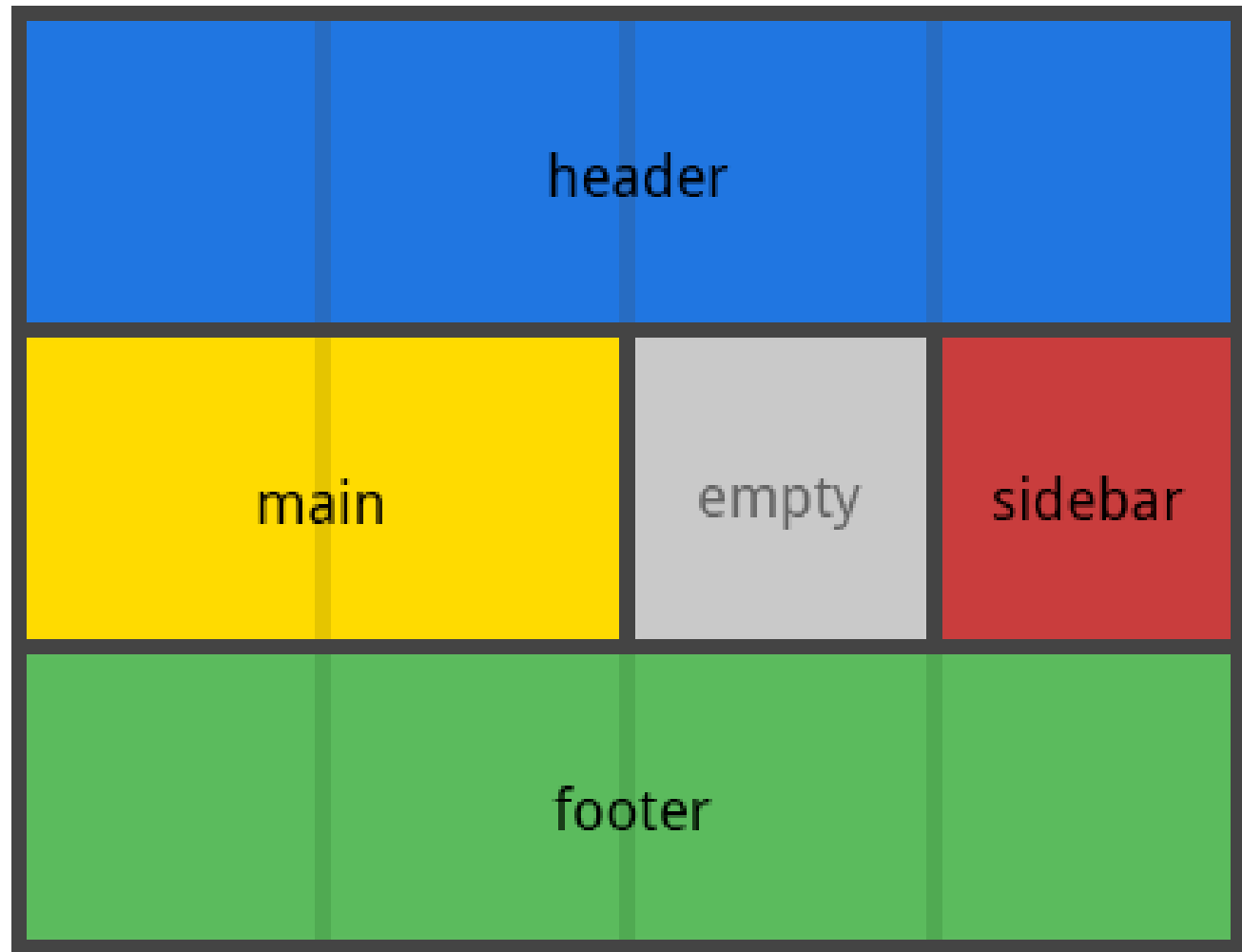
```
". b c";
```



```
.item-a {  
  grid-area: header;  
}  
.item-b {  
  grid-area: main;  
}  
.item-c {  
  grid-area: sidebar;  
}  
.item-d {  
  grid-area: footer;  
}  
.grid {  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}
```



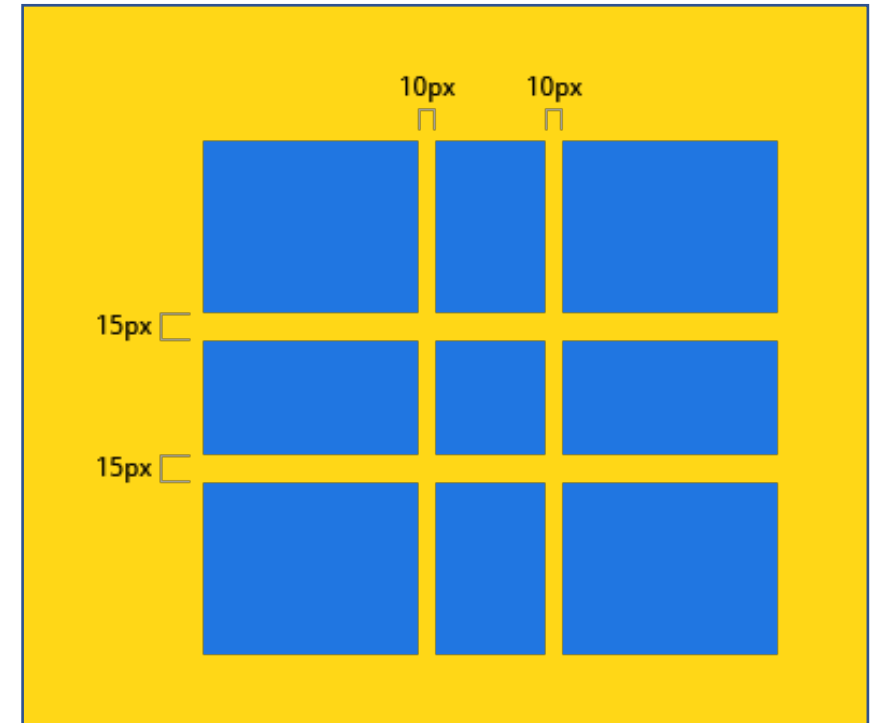
## GRID-TEMPLATE-AREAS



## LES ESPACES ENTRE LES LIGNES ET COLONNES

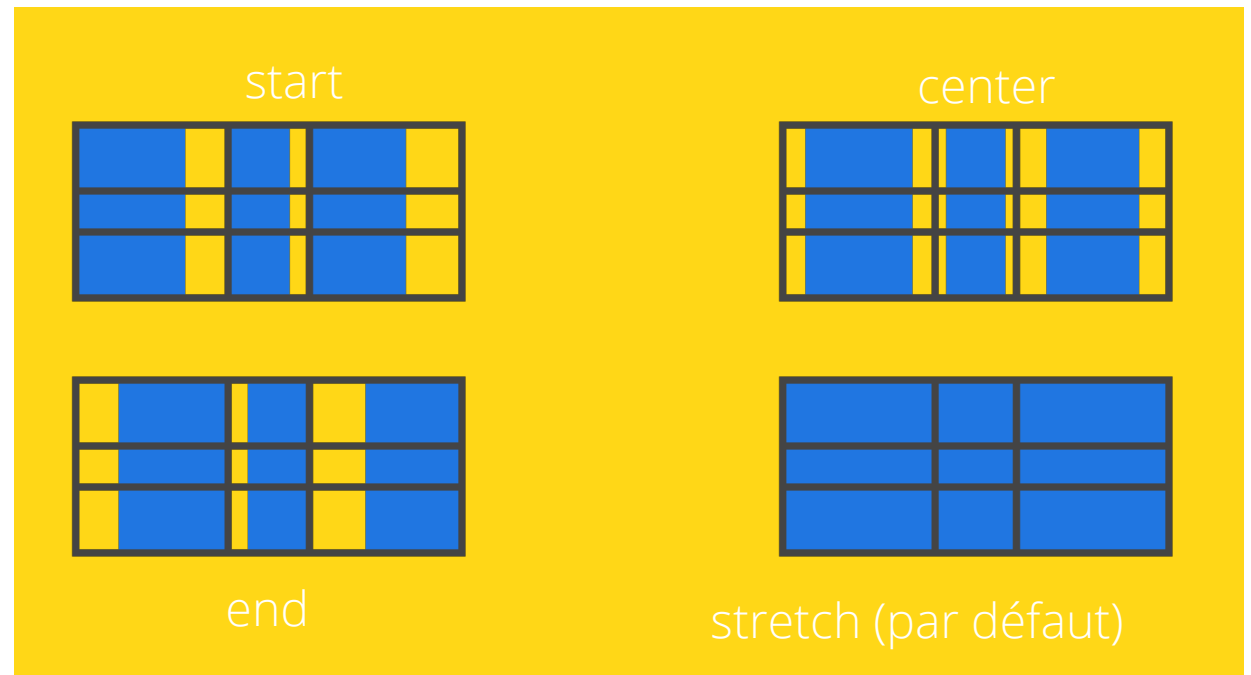
- ∞ On utilise **grid-column-gap** et **grid-row-gap**
- ∞ Ou plus simple **grid-gap** : **<grid-column-gap>** **<grid-row-gap>**

```
.container {  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: 80px auto 80px;  
  grid-gap: 10px 15px;  
}
```



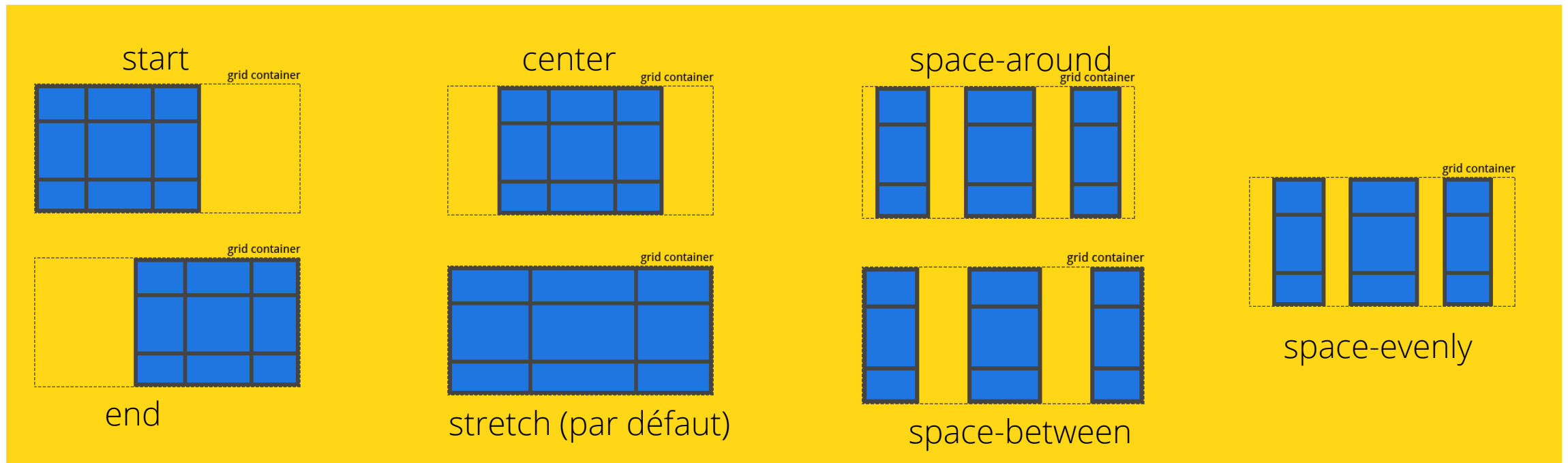
## L'ALIGNEMENT DES ITEMS DANS LES LIGNES ET COLONNES

- ∞ la propriété **justify-items** en horizontal
- ∞ la propriété **align-items** en vertical
- ∞ Ces propriétés peuvent prendre 4 valeurs



## L'ALIGNEMENT DU CONTENU DANS LE CONTENEUR LUI-MÊME

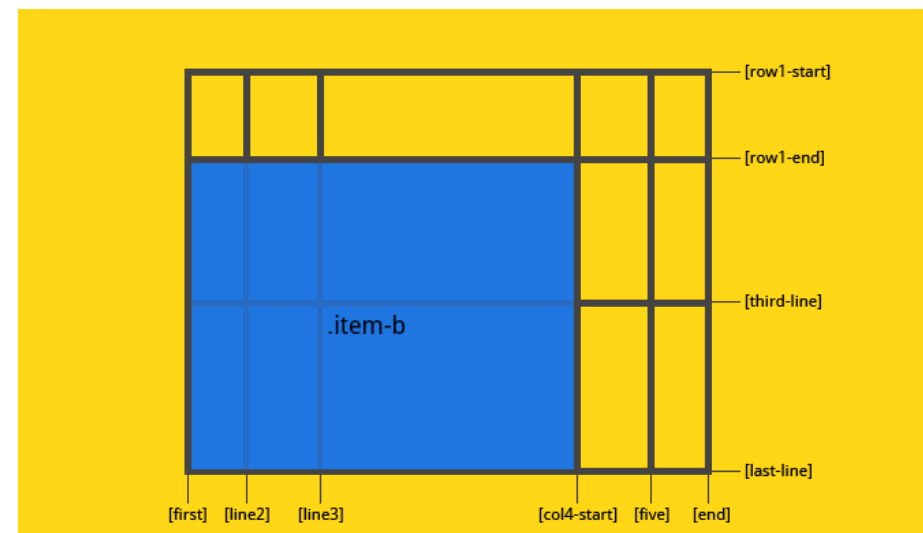
- ∞ la propriété `justify-content` en horizontal
- ∞ la propriété `align-content` en vertical
- ∞ Ces propriétés peuvent prendre 7 valeurs



## PROPRIÉTÉ DES ENFANTS

- ∞ Si l'on a pas utilisé **grid-template-areas**, les enfants se sont mis dans l'ordre chronologique
- ∞ On peut le placer avec :
  - en colonne : **grid-column-start** / **grid-column-end**
  - en ligne : **grid-row-start** / **grid-row-end**
- ∞ On utilise le numéro ou le nom de ligne et/ou l'étendu avec le mot clé span

```
.item-a {  
    grid-column-start: 1;  
    grid-column-end: span col4-start;  
    grid-row-start: 2;  
    grid-row-end: span 2;  
}
```



## LES RACCOURCIS

∞ On utilisera : **grid-column** et **grid-row**

∞ Voici leur syntaxe :

```
.item {  
  grid-column: <start-line> / <end-line> | <start-line> / span <value>;  
  grid-row: <start-line> / <end-line> | <start-line> / span <value>;  
}
```

## POSITIONNER LE CONTENU DES ENFANTS

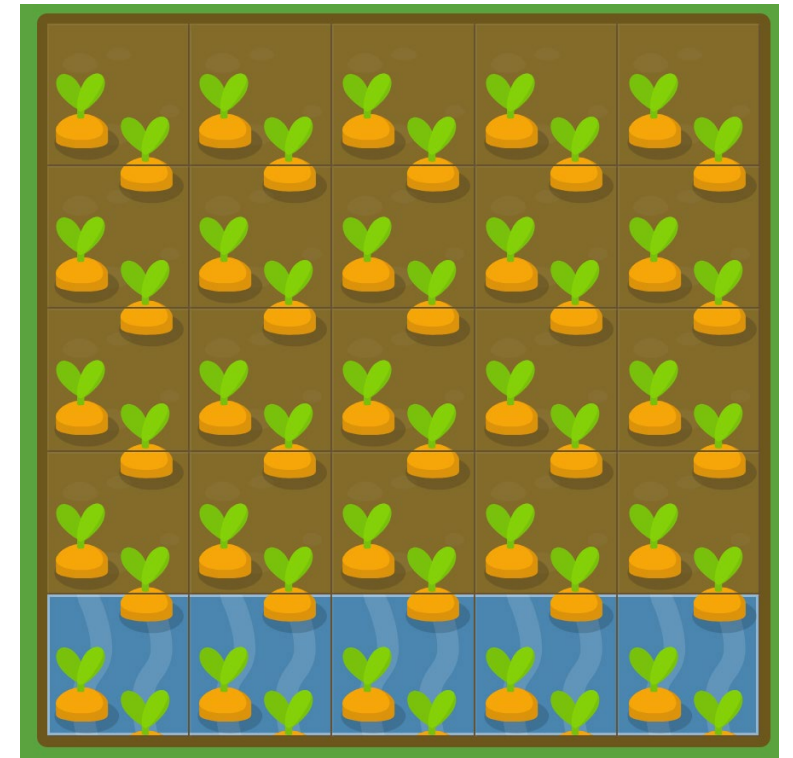
∞ Avec **justify-self** et **align-self** qui prennent les 4 valeurs :

- start - aligne le contenu sur la gauche de la zone de grille
- end - aligne le contenu sur la droite de la zone de grille
- center - aligne le contenu au centre de la zone de grille
- stretch - Remplit toute la largeur de la zone de grille (c'est la valeur par défaut).

## JEU : CULTIVONS, CULTIVONS !

∞ Se connecter ici : <http://cssgridgarden.com/#fr>

∞ Et zou, arrosez votre jardin !







## MEDIA-QUERIES : RESPONSIVE DESIGN

## PARTIE 3 – MANIPULATIONS CSS

01 LES SÉLECTEURS, LES RÈGLES

02 COULEURS ET MISE EN FORME DU TEXTE

03 COULEURS ET MISE EN FORME DES BOITES ET DES FONDS

04 FLEXBOX

05 GRID LAYOUT

06 **MEDIA-QUERIES : RESPONSIVE DESIGN**

## @MEDIA-SCREEN

∞ On va pouvoir définir un code spécifique en fonction de la taille et de l'orientation de l'écran utilisé

🤖 => **Faire slide 81 – @media**

```
@media screen and (max-width: 680px) {  
    body {  
        background-color: orange;  
    }  
}  
  
@media screen and (min-width: 681px) and (max-width: 1279px) {  
    body {  
        background-color: blue;  
    }  
}  
  
@media screen and (min-width: 1280px) {  
    body {  
        background-color: green;  
    }  
}
```

## LES PARAMÈTRES

- ∞ La taille max : `@media screen and (max-width: <valeur-min>px)`
- ∞ La taille min : `@media screen and (min-width: <valeur-max>px)`
- ∞ Un intervalle : `@media screen and (min-width: <valeur-min>px) and (max-width: <valeur-max>px)`
- ∞ L'orientation : `@media screen and (orientation:landscape)`



## TP : UTILISER LES MEDIA QUERIES

∞ Ouvrir le TP en faites en sorte que le fichier se redimensionne pour un écran inférieur à 685px

### zoom sur...



#### Soulfinger - Life, Love & Passion

Le producteur Soulfinger qui a posé ses valises à New York ces dernières années nous propose [...]

[lire la suite](#)



#### Dwele en concert le 24 avril 2013 au New Morning (Paris)

En 2006, Dwele nous faisait découvrir My People en live au New Morning. Je garde un très bon souvenir [...]

[lire la suite](#)



#### Justin Timberlake - The 20/20 Experience

Pourquoi n'y a-t-il aucun sujet sur cet album ? C'est de la soul.RnB, non ? Bon bref, je le trouve pesonnellement [...]

[lire la suite](#)



#### Allen Stone en concert à la Flèche d'or le 11 Mars

Après son dernier concert à la Maroquinerie, Allen Stone revient à Paris, cette fois-ci à la flèche [...]

[lire la suite](#)

À BIENTÔT

