

**HTML**



**CSS**



**JS**

# Intégration web

# Introduction / Les outils du front end

- Structure : HTML
- Mise en forme : CSS
- Interactivité, logique : Javascript

# Le HTML 5 - rappels

## ■ Le langage html

- signifie Hyper Text Markup Language
- est le langage de balisage standard pour la création de pages Web
- décrit la **structure** d'une page Web
- se compose d'une série d'éléments (**balises**)
- Les éléments HTML indiquent au navigateur comment afficher le contenu
- Les éléments HTML étiquettent des éléments de contenu tels que "ceci est un titre", "ceci est un paragraphe", "ceci est un lien", etc.
- date de 1991

# Le HTML 5 - rappels

- Les balises html

<tagname> Le contenu va ici ... </tagname >

<h1> Mon premier cap </h1 >

Certaines balises sont vides par définition...

<br>

[https://www.w3schools.com/html/tryit.asp?filename=tryhtml\\_intro](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro)

# Le HTML 5 - rappels

- Les attributs html

```
<tagname attr="value">... </tagname >
```

# HTML - rappels / La structure de base

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
    <meta charset="utf-8">
  </head>
  <body>
    </body>
</html>
```

# Le HTML 5 - rappels

## ■ Les balises de la structure html de base

- `<!DOCTYPE html>` définit que ce document est un document HTML5
- `<html>` est l'élément racine d'une page HTML
- `<head>` contient des méta-information sur la page HTML
- `<title>` spécifie un titre pour la page HTML (qui est affiché dans la barre de titre du navigateur ou dans l'onglet de la page)
- `<body>` définit le corps du document et est un conteneur pour tous les contenus visibles, tels que les en-têtes, les paragraphes, les images, les hyperliens, les tableaux, les listes, etc.
- `<h1>` définit un grand titre
- `<p>` définit un paragraphe
- `<meta>` définit les métadonnées d'un document HTML

# Le HTML 5 - rappels

- Les différents types d'images
  - **JPEG** 16 millions de couleurs (compressé)
  - **PNG** 16 millions plus transparence (le plus récent)
  - **GIF** 256 couleurs, peut être animé
  - **SVG** vectoriel

# Les balises HTML indispensables

## Les balises de base

- <!DOCTYPE> Définit le type de document
- <html> Définit un document HTML
- <head> Définit un ensemble d'informations sur le document
- <title> Définit le titre du document
- <body> Définit le corps du document
- <h1> à <h6> Définit les titres HTML
- <p> Définit un paragraphe
- <br> Insère un saut de ligne [auto fermante]
- <hr> Insère une ligne horizontale [auto fermante]
- <!--...--> Définit un commentaire

# Les balises HTML indispensables

## Mise en forme

- <address> Définit les infos de contact de l'auteur d'un document/article
- <b> Définit du texte en gras
- <blockquote> Définit une citation
- <code> Définit un bout de code informatique
- <em> Définit du texte mis en avant (apparaît en italique)
- <i> Définit du texte en italique
- <mark> Définit du texte mis en avant (apparaît surligné)
- <strong> Définit du texte important (apparaît en gras)
- <sub> Définit du texte au format indice
- <sup> Définit du texte au format exposant
- <time> Définit une date ou une heure
- <u> Définit du texte souligné

# Les balises HTML indispensables

## Les liens

- <a> Définit un lien hypertexte
- <link> Définit une relation entre un document et une ressource externe (surtout utilisé pour les feuilles de style css) [auto fermante]
- <nav> Définit les liens de navigation

## Les listes

- <ul> Définit une liste non ordonnée
- <ol> Définit une liste ordonnée
- <li> Définit l'élément d'une liste

# Les balises HTML indispensables

## Les images

- <img> Définit une image [auto fermante]
- <canvas> Utilisé pour dessiner des graphiques via Javascript
- <figcaption> Définit la légende d'un élément <figure>
- <figure> Caractérise son propre contenu
- <picture> Permet de définir plusieurs sources pour une image

## Audio Video

- <audio> Définit du contenu audio
- <source> Définit des ressources multiples pour les éléments media (<video> et <audio>)
- <track> Définit des titres de piste pour les éléments media (<video> et <audio>)
- <video> Définit du contenu video

# Les balises HTML indispensables

## Les formulaires

- <form> Définit un formulaire HTML
- <input> Définit un champ de saisie [auto fermante]
- <textarea> Définit un champ de saisie (zone de texte multiligne)
- <button> Définit un bouton cliquable
- <select> Définit une liste déroulante
- <optgroup> Définit un groupe d'options dans une liste déroulante
- <option> Définit une option dans une liste déroulante
- <label> Définit une étiquette pour un élément <input>
- <fieldset> Regroupe des éléments dans un formulaire
- <legend> Définit une légende pour un élément <fieldset>

# Les balises HTML indispensables

## Styles et sémantique

- <style> Définit des informations de style pour le document
- <div> Définit un bloc
- <span> Définit une section en ligne
- <header> Définit l'entête d'un document ou d'une section
- <footer> Définit le pied de page d'un document ou d'une section
- <main> Spécifie le contenu principal d'un document
- <section> Définit une section
- <article> Définit un article
- <aside> Définit le contenu à côté du contenu de la page

# Les balises HTML indispensables

## Les tables

- <table> Définit un tableau
- <th> Définit une cellule d'entête dans un tableau
- <tr> Définit une ligne
- <td> Définit une cellule
- <thead> Regroupe le contenu de l'entête
- <tbody> Regroupe le corps du contenu
- <tfoot> Regroupe le contenu du pied de page

# Les balises HTML indispensables

## Meta Info

- <head> Définit les informations sur le document
- <meta> Définit les metadata sur le document [auto fermante]

## Autres balises

- <script> Définit un script côté client
- <noscript> Définit un contenu alternatif en cas de désactivation du Javascript côté client
- <iframe> Intègre un cadre dans le document acceptant du contenu externe

# Le HTML 5

- Les principales balises structurantes html5

< <b>header</b> >	Entête
< <b>main</b> >	Contenu principal
< <b>footer</b> >	Pied de page
< <b>nav</b> >	Principaux liens de navigation
< <b>section</b> >	Section dans un document

# Le HTML 5

- Les autres balises html5 importantes

<**article**> Un article dans un document

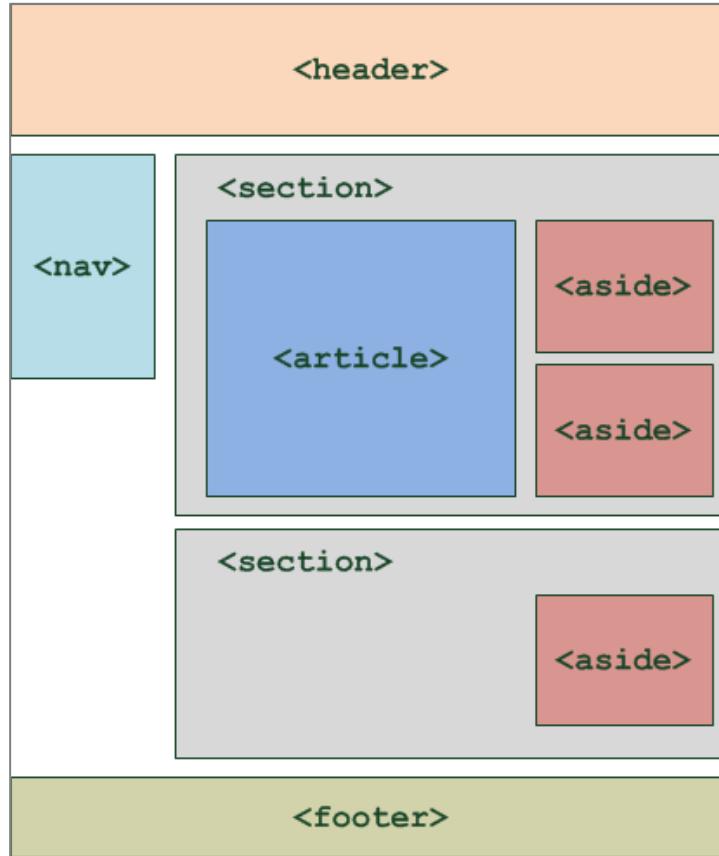
<**aside**> Le contenu annexe de la page

<**figure**> Contenu autonome (image et légende)

<**figcaption**> Légende pour un élément <figure>

<**time**> Date ou heure

# Le HTML 5



Quelle balise manque  
à cette page ?

# HTML et SEO

La structuration du contenu selon des balises « signifiantes » est considérée comme une **bonne pratique**. Notamment pour le référencement par les moteurs de recherche.

En plus de ces considérations de SEO, une telle structuration est encouragée pour rendre le contenu facilement accessible aux clients « machines » présents et à venir.

Les balises structurantes du html5 doivent être utilisées au détriment des balises <div>.

A ces balises, il faut ajouter les balises classiques <strong>, <em> et <h1> qui jouent un rôle important dans l'indexation d'une page.

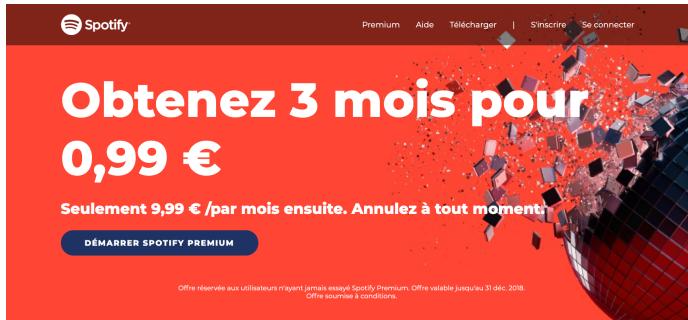
# Toutes les balises et attributs HTML

<https://www.w3schools.com/html/default.asp>

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

# Exercice 1

Structure html à partir d'un template graphique.



[Premium](#) [Aide](#) [Télécharger](#) | [S'inscrire](#) [Se connecter](#)

**Obtenez 3 mois pour 0,99 €**

Seulement 9,99 € /par mois ensuite. Annulez à tout moment.

[Démarrer Spotify Premium](#)

Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2018. Offre soumise à conditions.

**Pourquoi passer à Spotify Premium ?**

Téléchargez votre musique.

Profitez-en même sans connexion internet

**Écoutez gratuitement ou abonnez-vous à Spotify Premium.**

Spotify Free

**0,00 € /mois**

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zappez les titres sans limite
- ✓ Écouter hors connexion
- ✓ Écoutez les titres de votre choix
- ✓ Son de qualité supérieure

[DÉMARRER](#)

Spotify Premium

**3 mois pour 0,99 €**

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zappez les titres sans limite
- ✓ Écouter hors connexion
- ✓ Écoutez les titres de votre choix
- ✓ Son de qualité supérieure

[DÉMARRER SPOTIFY PREMIUM](#)

Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2018. Offre soumise à conditions.

[Légal](#) [Cookies](#) [À propos des pubs](#)

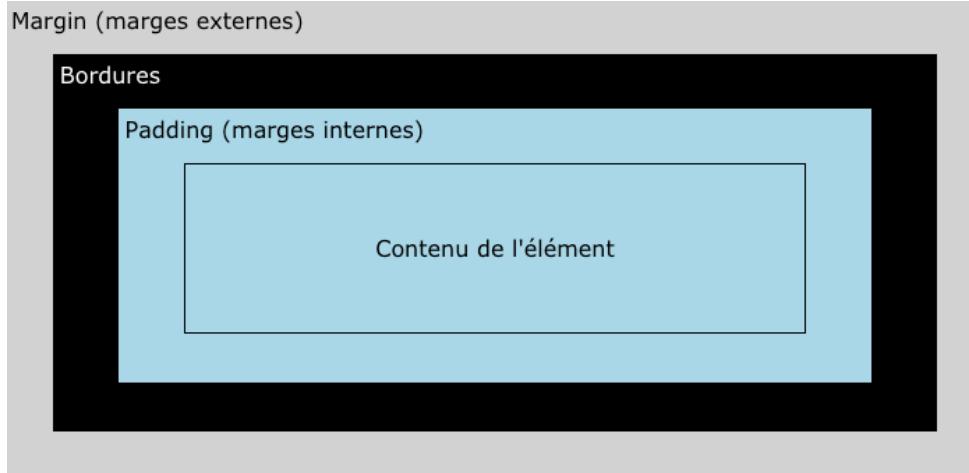
© 2018 Spotify AB

# CSS - rappels

- Lier une feuille de style externe

```
<link rel="stylesheet" type="text/css" href="css/style.css">
```

2 grands types de balises : inline et block



**box-sizing**

content-box,  
border-box

**margin : auto**

Centre horizontalement

# Le CSS - rappels

- Positionnement

Le positionnement flottant : **float**

Le type d'affichage : **display**

Le type de position : **position**

# Le positionnement en CSS

## ■ **float**

**none** (défaut), **right**, **left**

Si différent de none, élément retiré du flux normal, se place à droite ou à gauche de son conteneur. Demeure tout de même dans le flux.

Implique l'utilisation d'une disposition en bloc : change la valeur du display.

Sa hauteur n'est pas prise en compte par son parent.

Nécessite un **clearfix**

# Le positionnement en CSS

- **display**

**block, inline**

**none** : élément pas affiché

**inline-block** : en ligne mais forme un rectangle (ne va pas à la ligne)

**flex** : se comporte comme un conteneur flexible

list-item, table, grid...

# Le positionnement en CSS

## ▪ **position**

**static** (défaut) : élément positionné dans le flux avec sa position par défaut

**relative** : positionné dans le flux, mais peut être décalé avec les propriétés top, right, bottom, left

**absolute** : sorti du flux, positionné avec top, right, bottom, left, par rapport à son plus proche ancêtre positionné.

**fixed** : sorti du flux, positionné avec top, right, bottom, left, par rapport à la fenêtre (ne bouge pas lors du scroll).

**sticky** : positionné dans le flux mais sorti du flux et positionné de façon fixe par rapport au parent ("collé" en haut).

# CSS - rappels

- Quelques sélecteurs

selector1 > selector2

selector1 + selector2

Pseudo classes :

:hover

:checked

:active

:first-child

:last-child

:nth-child(n)

:not(selector)

# CSS - rappels

Pseudo éléments :

::after

::before

::first-letter

::first-line

::selection

# CSS - rappels

- Quelques propriétés css

clear : both;

display : table;

transform : translateY(-50%);

transition : all 1s;

z-index : 5;

# CSS - rappels

```
box-shadow: 10px 10px 5px #888888;
```

```
selector {  
    background-image: url( ... );  
    background-position: center;  
    background-size: cover;  
    background-repeat: no-repeat;  
}
```

```
selector {  
    border: 5px solid red;  
    border-radius: 1em;  
}
```

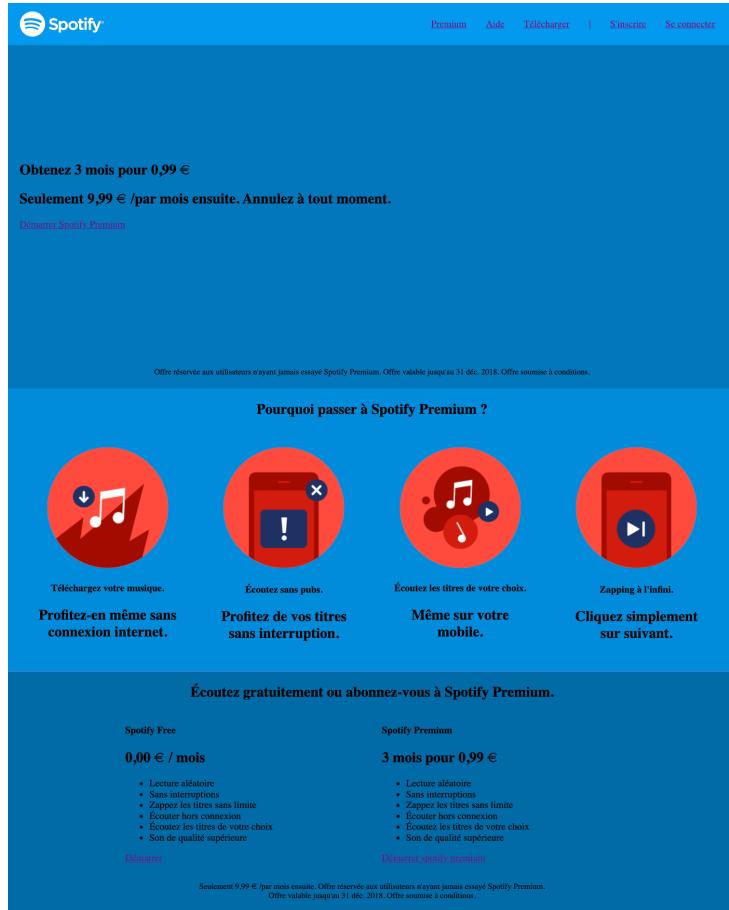
```
img{  
    object-fit: cover;  
}
```

# CSS - rappels

```
@font-face {  
    font-family: maPolice;  
    src: url(sansation_light.woff);  
}  
  
@import url('https://fonts.googleapis.com/css?family=Roboto:300,400');  
  
p {  
    font-family: 'Roboto', sans-serif;  
}
```

# Exercice 1 (suite)

Styles CSS de position et dimension à partir d'un template graphique.



Sans flexbox, avec des **flottants**

#0099EE;  
**Notions importantes**

- Utiliser un **container**
- Contrôler le padding avec le **box-sizing**
- Contrôler les **flottants** avec un **clearfix**
- Contrôler le **margin-collapse** (**clearfix**)
- Contrôler le **positionnement vertical** (**transform**)
- Utiliser **calc()** et **nth-child()**

# Le positionnement en CSS - flexboxes

- Utiliser des conteneurs flex

Un **conteneur flex** (flexbox) a la propriété **display : flex** (ou inline-flex).

Son utilisation permet de :

- Dimensionner les blocs enfants indépendamment de leur contenu.
- Les présenter horizontalement (row) ou verticalement (column).
- Simplifier l'alignement des éléments (axe vertical).

# Le positionnement en CSS - flexboxes

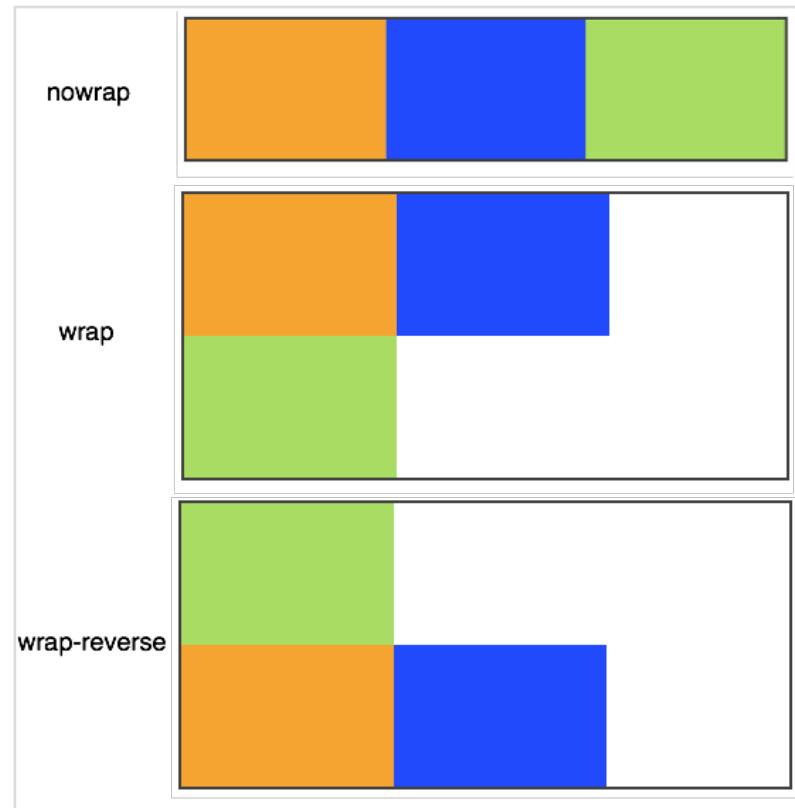
- Propriétés de base des flexboxes

**flex-direction** : en ligne ou en colonne

row (défaut), column, row-reverse, column-reverse

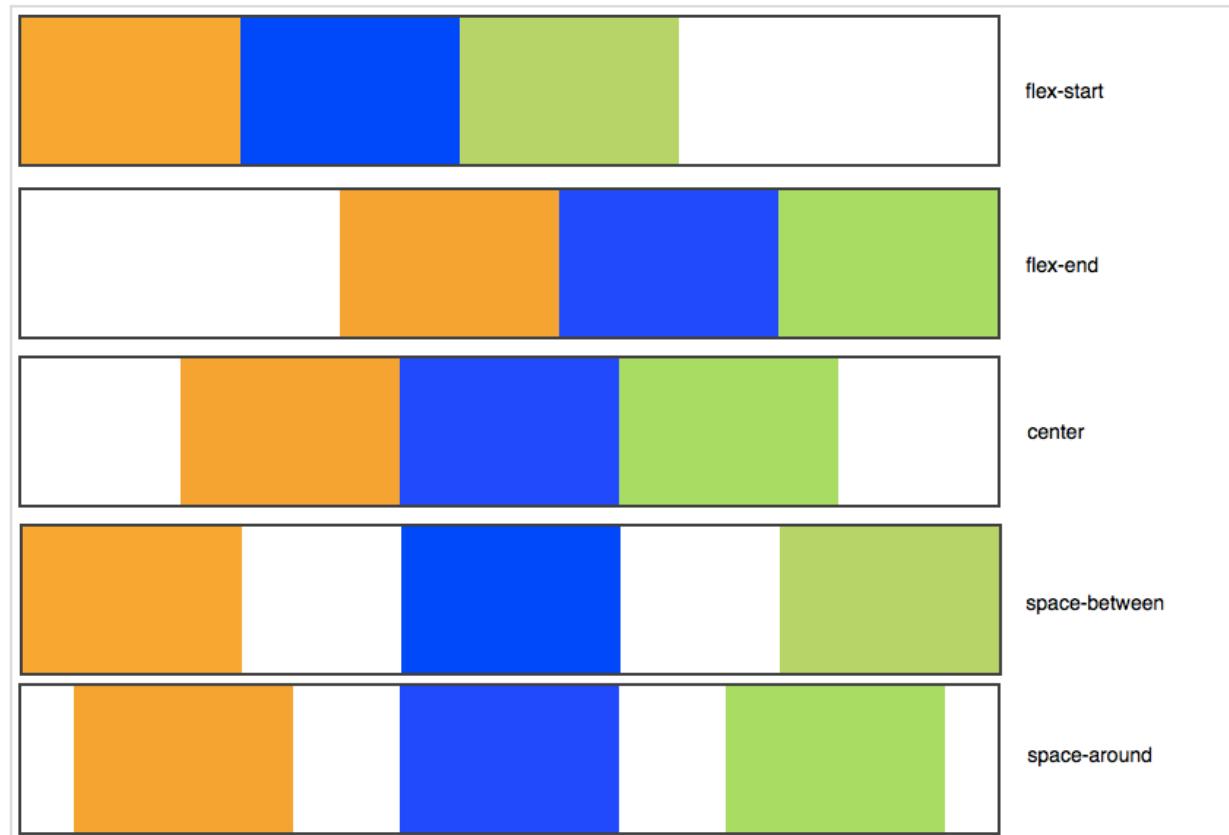
**flex-wrap** : les éléments peuvent aller à la ligne ou pas

nowrap (défaut), wrap, wrap-reverse



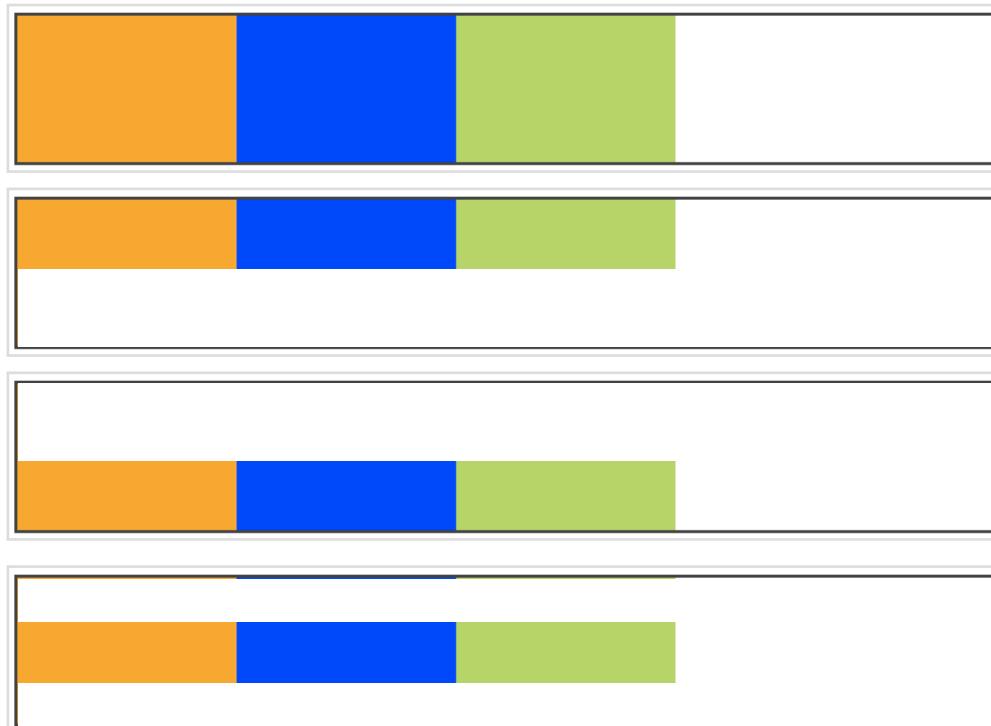
# Le positionnement en CSS - flexboxes

**justify-content** : alignement des enfants sur l'axe principal  
flex-start(défaut), flex-end, center, space-between, space-around



# Le positionnement en CSS - flexboxes

**align-items** : alignement des enfants sur l'axe secondaire  
stretch(défaut), flex-start, flex-end, center



# Le positionnement en CSS - flexboxes

Chaque élément contenu dans une flexbox est un **élément flex**.

On peut lui appliquer la propriété **flex** qui regroupe 3 propriétés :

**flex-grow** : comment l'élément va grossir par rapport aux autres.

**flex-shrink** : comment l'élément va rétrécir par rapport aux autres.

**flex-basis** : la « longueur » de l'élément. Peut valoir auto, inherit ou encore une valeur absolue (px) ou relative (%), em).

Valeur par défaut

**flex : 0 | auto**

Si une seule valeur donnée, indique le poids relatif de l'élément flex

**flex : 1**

# Le positionnement en CSS - flexboxes

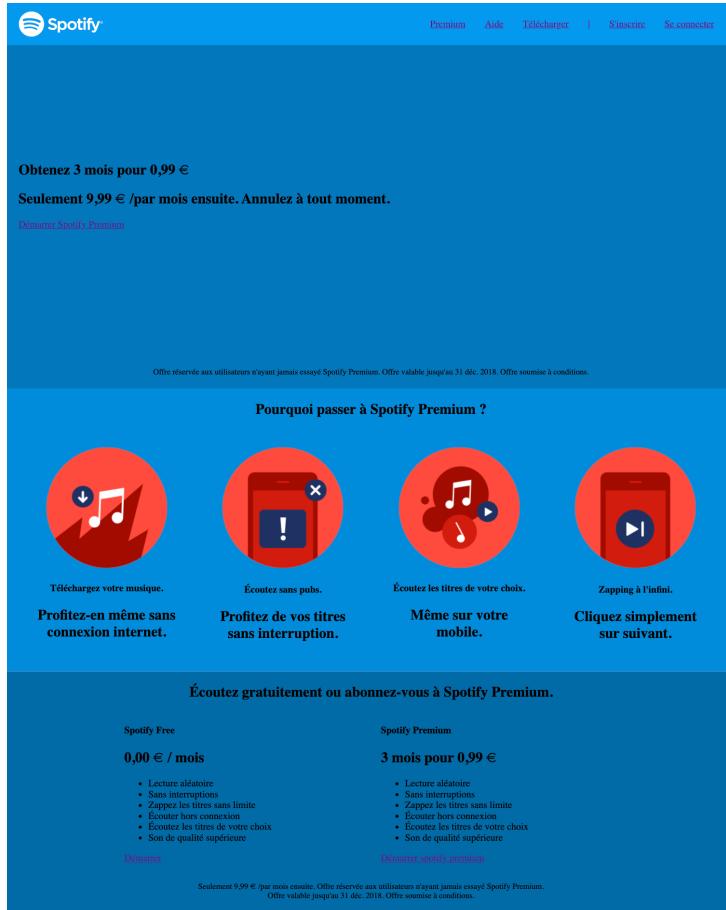
- Autres propriétés des éléments flex

**align-self** : capacité de l'élément à s'aligner indépendamment sur l'axe secondaire.

**order** : capacité de l'élément à se positionner dans le conteneur. Pour être efficace, il faut que tous les éléments aient une valeur order.

# Exercice 1 (suite)

Styles CSS de position et dimension à partir d'un template graphique.



Avec **flexbox**, sans flottants

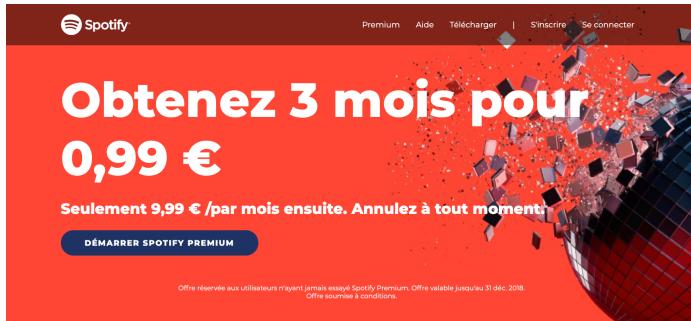
## Notions importantes

- flex-direction
- flex-wrap
- flex-direction
- justify-content
- align-items

- flex-grow
- flex-shrink
- flex-basis
- order

# Exercice 1 (Fin)

Styles CSS mise en forme du texte et des visuels.



Cf.TP3 sur myGES

## Pourquoi passer à Spotify Premium ?



Téléchargez votre musique.  
Profitez-en même sans connexion internet.

Écoutez sans pubs.  
Profitez de vos titres sans interruption.

Écoutez les titres de votre choix.  
Même sur votre mobile.

Zapping à l'infini.  
Cliquez simplement sur suivant.

## Écoutez gratuitement ou abonnez-vous à Spotify Premium.

**Spotify Free**  
**0,00 €/mois**

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zapper les titres sans limite
- ✓ Écouter hors connexion
- ✓ Ecoutez les titres de votre choix
- ✓ Son de qualité supérieure

**DÉMARRER**

**Spotify Premium**  
**3 mois pour 0,99 €**

- ✓ Lecture aléatoire
- ✓ Sans interruptions
- ✓ Zappez les titres sans limite
- ✓ Écouter hors connexion
- ✓ Ecoutez les titres de votre choix
- ✓ Son de qualité supérieure

**DÉMARRER SPOTIFY PREMIUM**

Seulement 0,99 €/par mois ensuite. Offre réservée aux utilisateurs n'ayant jamais essayé Spotify Premium. Offre valable jusqu'au 31 déc. 2018. Offre soumise à conditions.

# CSS / les animations

## Les transitions css

Les transitions css3 permettent de passer progressivement d'une valeur à une autre.

La transition est déclenchée à chaque modification de la propriété ciblée.

```
.maclasse {  
    transition : propriete duree;  
}
```

```
.maclasse {  
    transition : all 1s;  
}
```

```
.maclasse {  
    transition : opacity 0.3s ease-out, width 2s;  
}
```

```
.maclasse {  
    transition: height 0.3s ease-out, opacity 0.3s ease 0.5s;  
}
```

# CSS / les animations

```
.maclasse {  
    transition-property: width, height;  
    transition-duration: 2s;  
    transition-timing-function: linear;  
    transition-delay: 1s;  
}
```

=

```
.maclasse {  
    transition: width height 2s linear 1s;  
}
```

# CSS / les animations

## Les animations CSS

Les animations dans les pages web se font de plus en plus à l'aide de propriétés css, au détriment des animations en javascript.

**@keyframes** permet de créer une animation au cours de laquelle les propriétés css sont graduellement modifiées. Pour utiliser cette animation, il faut appeler les propriétés **animation-name** et **animation-duration**.

```
@keyframes mon_animation{  
    from {propriete1 : valeur1; propriete2 : valeur2}  
    to {propriete1 : valeur3; propriete2 : valeur4}  
}  
.maclasse{  
    animation-name : mon_animation;  
    animation-duration : 4s;  
}
```

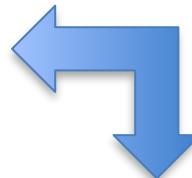
# CSS / les animations

On peut définir des images-clé tout au long de l'animation :

```
@keyframes mon_animation{  
    0% {background-color: red;}  
    25% {background-color: yellow;}  
    50% {background-color: blue;}  
    100% {background-color: green;}  
}
```

```
div {  
    animation-name: mon_animation;  
    animation-duration: 5s;  
    animation-timing-function: linear;  
    animation-delay: 2s;  
    animation-iteration-count: infinite;  
    animation-direction: alternate;  
}
```

```
div {  
    animation: mon_animation 5s linear 2s infinite alternate;  
}
```



Equivalent

# CSS / @media queries

Les @media queries permettent de créer des styles pour certains types de sortie ou taille d'écran : le « **responsive design** ».

On peut créer des styles pour l'écran, l'impression... ou encore pour un affichage en portrait ou un écran de moins de 640px de large.

```
@media screen and (max-width: 640px){  
    /* mes styles pour un viewport de moins de 640px */  
}
```

Ces @media queries permettent de modifier l'agencement d'une page en fonction de sa largeur. Associées à une grille, elles permettent de créer des points d'arrêt (« **breakpoints** ») pour chaque type de mise en forme : xsmall, medium, large...

Note : Pour une bonne adaptation au format téléphone, il faut prendre en compte le zoom par défaut des navigateurs mobiles.

`<meta name='viewport' content='width=device-width, initial-scale=1.0'>`

Attention à user-scalable (cf SEO)

# CSS / @media queries

Deux méthodes d'application :

- en chargeant une feuille de style .css différente en fonction de la règle

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

OU

```
<link rel="stylesheet" media="screen and (max-width: 1280px)" href="mobile.css" />
```

- en écrivant la règle directement dans le fichier .css habituel

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

# CSS / @media queries

## Les règles disponibles :

- 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 : écran « classique » ;
- handheld : périphérique mobile ;
- print : impression ;
- tv : télévision ;
- projection : projecteur ;
- all : tous les types d'écran.

# CSS / @media queries

## Exemples d'utilisation

- **only** : « uniquement »
- **and** : « et »
- **not** : « non »

```
/* 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)
```

# CSS / @media queries

## Exemple de breakpoints (Bootstrap)

- Extra small <576px
- Small ≥576px (largeur max. du conteneur : 540px)
- Medium ≥768px (largeur max. du conteneur : 720px)
- Large ≥992px (largeur max. du conteneur : 960px)
- Extra large ≥1200px (largeur max. du conteneur : 1140px)

```
@media screen and (min-width : 576px){  
    .conteneur{  
        max-width : 540px;  
    }  
}
```

Ces *breakpoints* reposant sur des (min-width:) est caractéristique du design « **mobile first** ».

Beaucoup de grilles css reposent sur des breakpoints et mettent en place des blocs flottants, contenus grâce au « **clearfix** ».

# Les grilles CSS / Principe

Héritée de la presse, la grille web est rendue nécessaire par les considérations suivantes :

- l'ergonomie (lisibilité)
- le responsive design

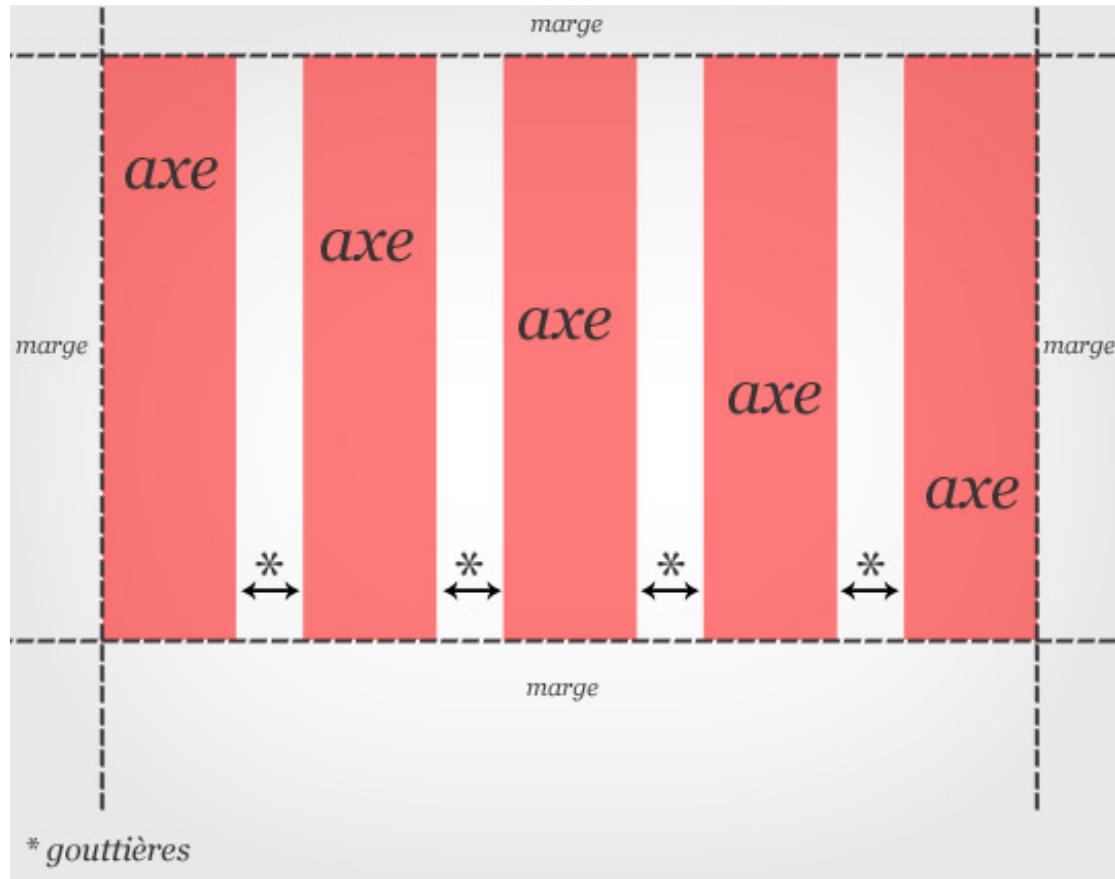
Il y a longtemps, les pages web étaient constituées de tableaux (`<table>`) qui ne peuvent pas s'adapter aisément aux différents formats.

Aujourd'hui, elles sont faites de blocs (`<div>`, `<main>`, `<section>`...) disposés selon une grille.

A screenshot of a French newspaper website (likely Le Monde) demonstrating a traditional grid layout. The page is divided into several columns containing text articles and small images. The layout is fixed and does not adapt to different screen sizes.

A screenshot of The New York Times website, illustrating a modern grid-based layout. The page features a sidebar on the right with a "World" section and other news categories. The main content area uses a grid to organize news stories, each with a thumbnail image and a brief headline. This layout is designed to be responsive and adaptable across different devices.

# Les grilles CSS / Principe

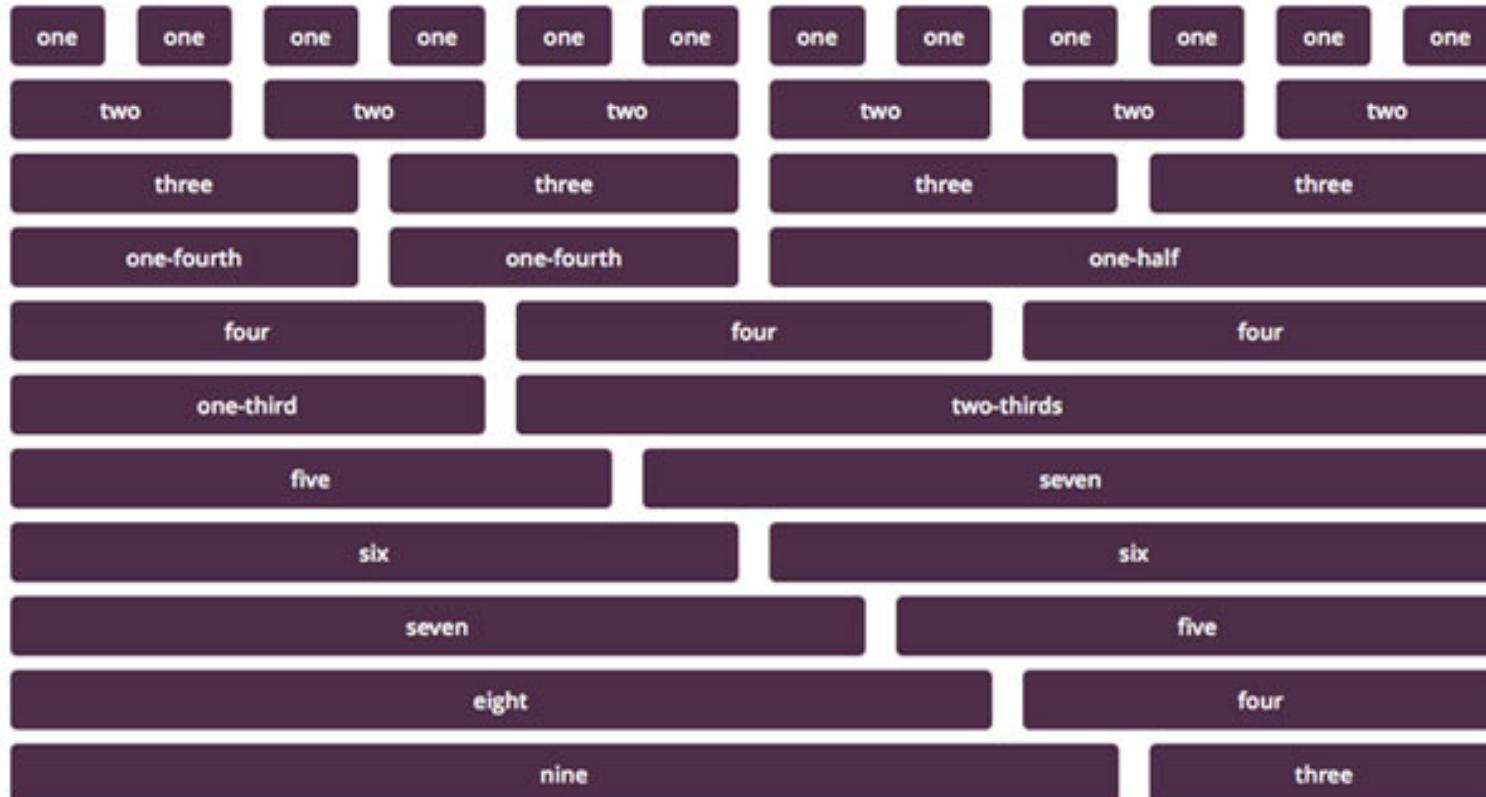


Une grille est constituée des éléments suivants :

- Des **colonnes** ou axes (au nombre de 12, la plupart du temps)
- Des **gouttières**
- Des **marges** latérales

# Les grilles CSS / Principe

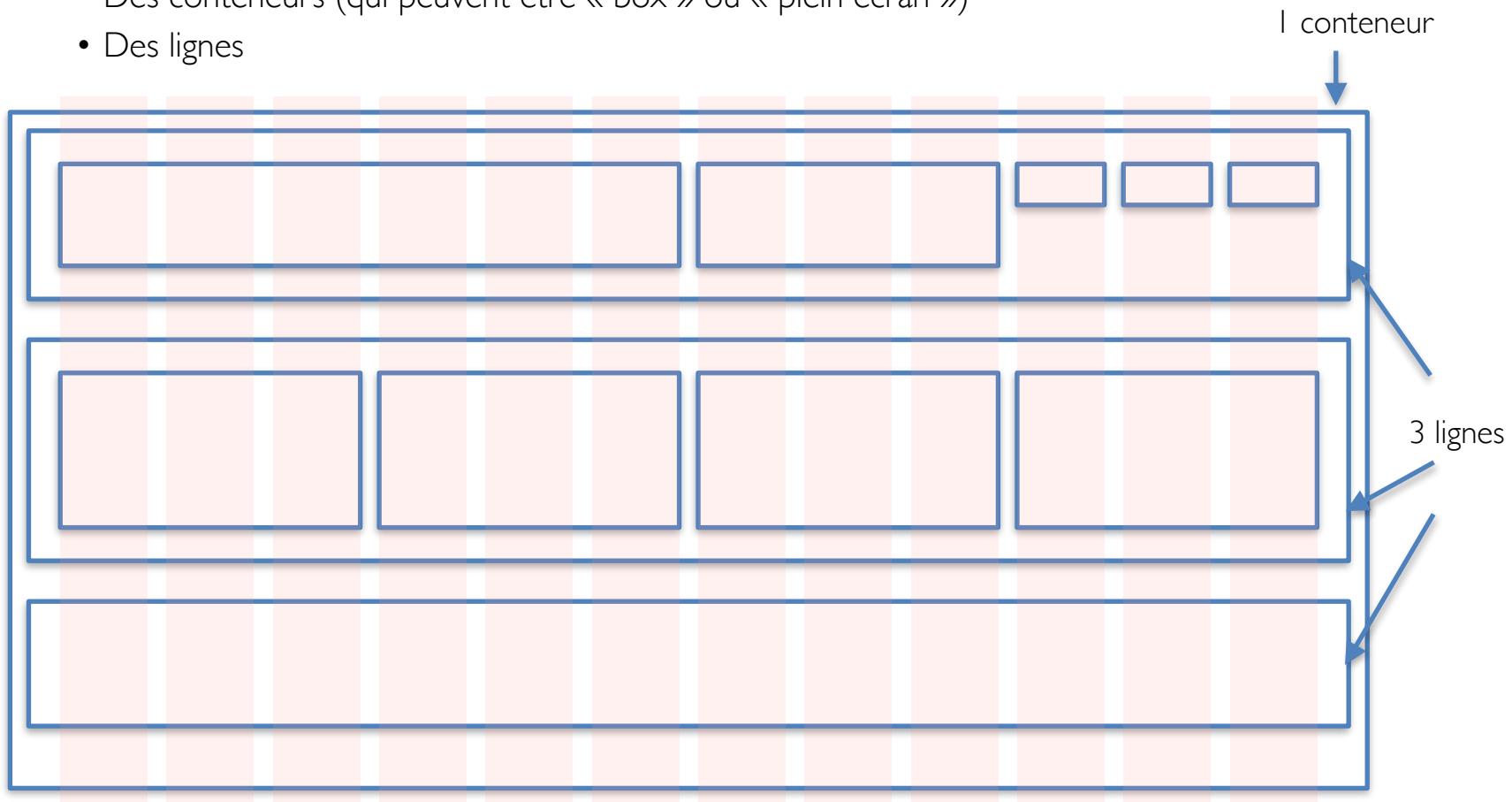
Chaque élément se positionne dans la grille selon sa largeur (exprimée en nombre de colonnes)



# Les grilles CSS / Principe

Pour simplifier la mise en page, on ajoute :

- Des conteneurs (qui peuvent être « box » ou « plein écran »)
- Des lignes



# Les grilles / exemples de framework CSS

Un framework CSS est un ensemble de fichiers comportant :

- les **styles de grille**
- les **styles des éléments standard** (formulaire, bouton...) dans différents états (alerte, succès...)
- des **composants UI** à base javascript (slider, accordéon, fenêtre modale...)
- des **polices de caractères**
- des **icônes**

## Les plus connus

### Bootstrap

- v3 : éléments flottants (float : left) avec des marges
- v4 : flexboxes avec du padding



### Zurb Foundation

### UIKit

### Semantic UI



# CSS / Les variables

## Variables CSS

```
/* Déclaration d'une variable */
--color-clickable: red;
/* Utilisation de la variable CSS */
color: var(--color-clickable);
```

Héritage (cascade) et spécificité

```
/* Default styles */
html {
    --color-clickable: red;
}
a {
    color: var(--color-clickable);
}
/* Footer styles */
.footer {
    --color-clickable: lightgrey;
    background: black;
    color: white;
}
```

Sinon :

```
/* Default styles */
a {
    color: red;
}
/* Footer styles */
.footer {
    background: black;
    color: white;
}
.footer a {
    color: lightgrey;
}
```