

# Conception Web

## HTML 5 / CSS 3

**Grégory Galli**

**Enseignant à l'Université de Nice Côte d'Azur (UCA)**  
**France**

# Présentation générale

Conception Web

# Conception Web - Présentation

---

- Format de données
- HTML5 / CSS3 / Javascript
- Spécifications du W3C
  - <http://www.w3.org/TR/html5/>

# Conception Web

---

- Problématiques
  - Ergonomie (UX / UI)
  - Résolutions
  - Performances
    - Du poste client
    - Du réseau
- Solutions
  - Best practices
  - Responsive Design (Media queries)
  - Architecture et Environnement technique adapté

# Outils

---

## ➤ Editeurs

- Notepad++
- Sublime Text
- Emacs
- VSCode



## ➤ IDE

- Netbeans
- Eclipse
- IntelliJ (Ultimate/spécifique)

## ➤ Versioning

- Git
- SVN



# Outils & Frameworks

---

## ➤ Préprocesseur CSS

- Sass
- Less
- Scss



## ➤ Frameworks

- PHP
  - CodeIgniter / Symfony / Zend / Laravel
- Java / Ruby / Groovy
  - JSF / Spring / Spring Boot
  - Ruby on Rails
  - Grails
- Python
  - Django / Flask



# Frameworks

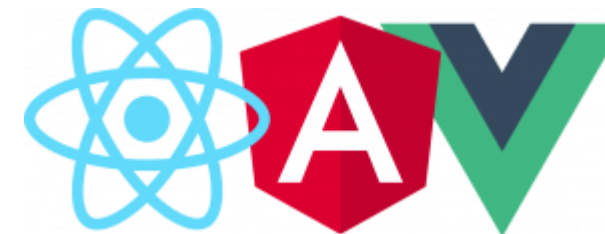
## ➤ CSS

- Bootstrap
- Foundation
- Bulma
- Pure
- Materialize
- Skeleton
- ...



## ➤ Javascript

- Angular
- React
- Vue.js
- Ember.js
- *jQuery*



# Frameworks

- Avantages
  - Mise à jour
  - Coût du développement
  - Cross-browser
  - Ergonomie
- Inconvénients
  - Surcouche
  - Dépendance
    - Bugs
    - Implémentation nouvelles fonctionnalités





# Conclusion

---

- Choix des technologies
- Equilibre
  - Besoins
  - Capacités
  - Contraintes
    - Temps
    - Budget
- Penser à l'avenir du projet
  - Maturité de technologies
  - Scalabilité
  - Migration ? MySQL / PostgreSQL / Oracle
  - Capacité à recruter

# HTML

Rappels

# Rappels - Structure

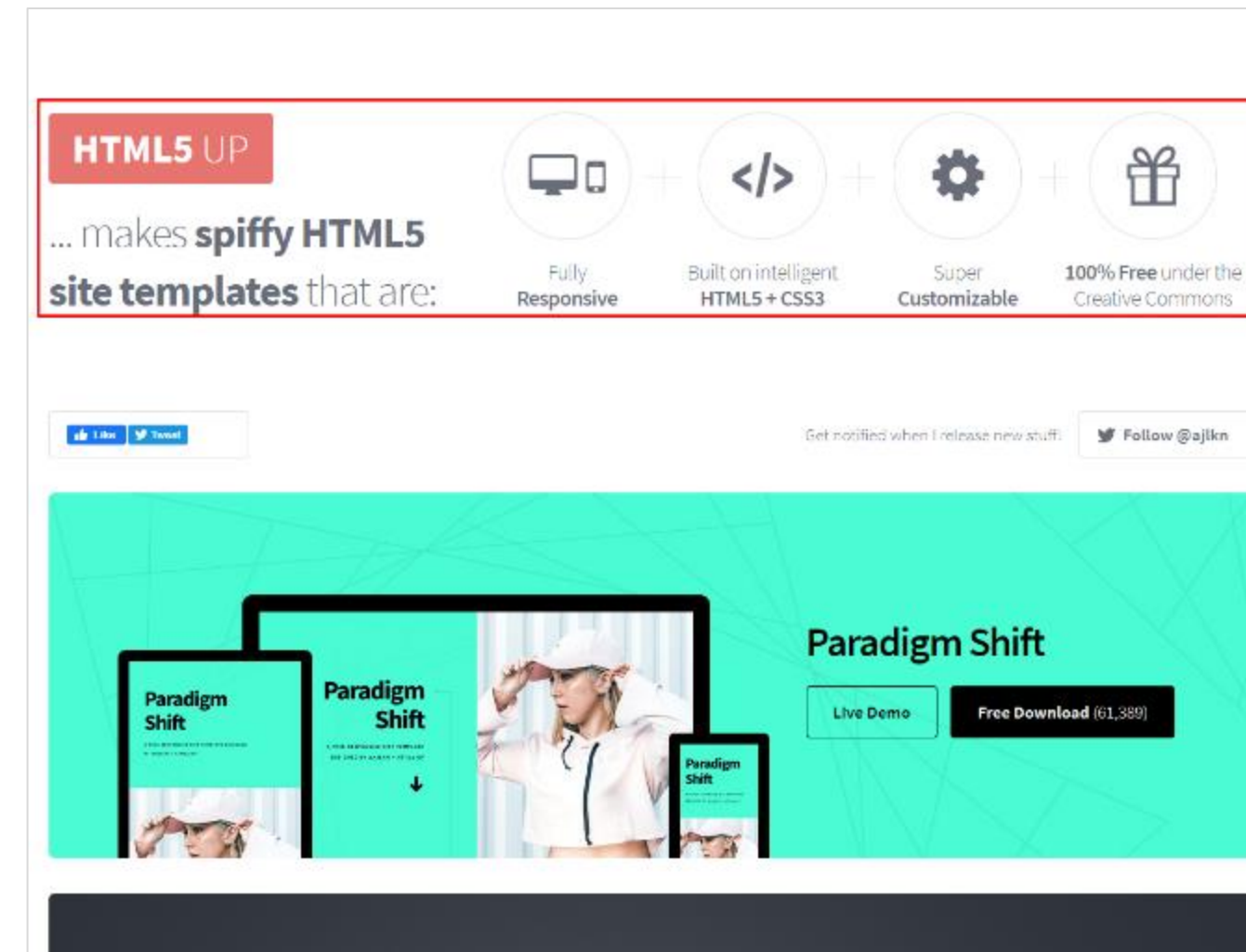
---

- Doctype
  - Document type
  - Spécification DTD (Document type definition)
  - HTML 5 

`<!DOCTYPE html>`
- HTML
  - Défini la racine du document
- Head
  - Méta data
  - Titre
  - Style
  - Inclusions
- Body
  - Contenu principal du document
  - Toute la partie visible

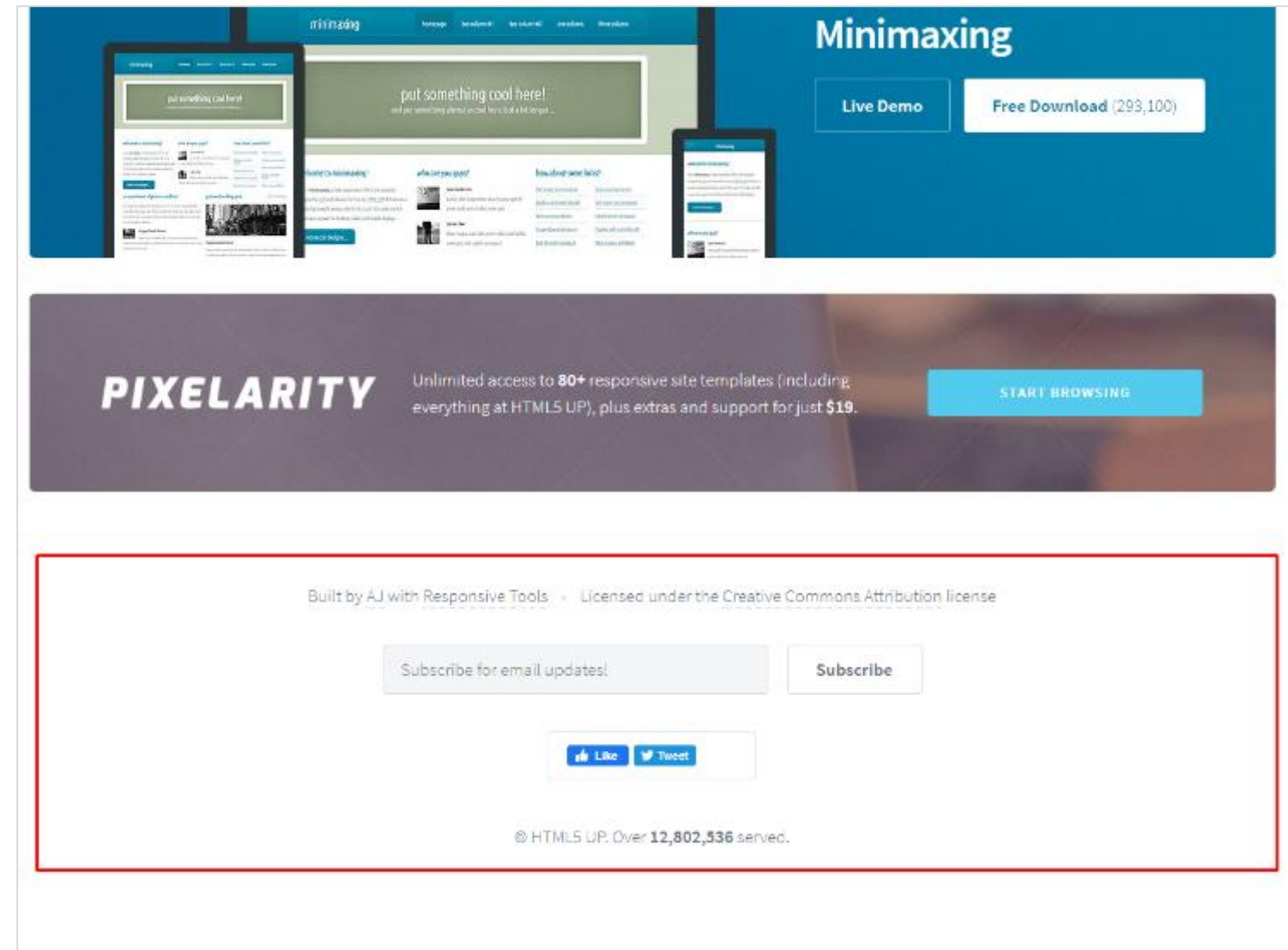
# Rappels - Structure HTML 5 - Header

- Contenu de l'en-tête du site
- Images
  - Logos
  - Illustrations
- Slogans
- Liens
  - Pas la navigation principale
- Login / Gestion de compte



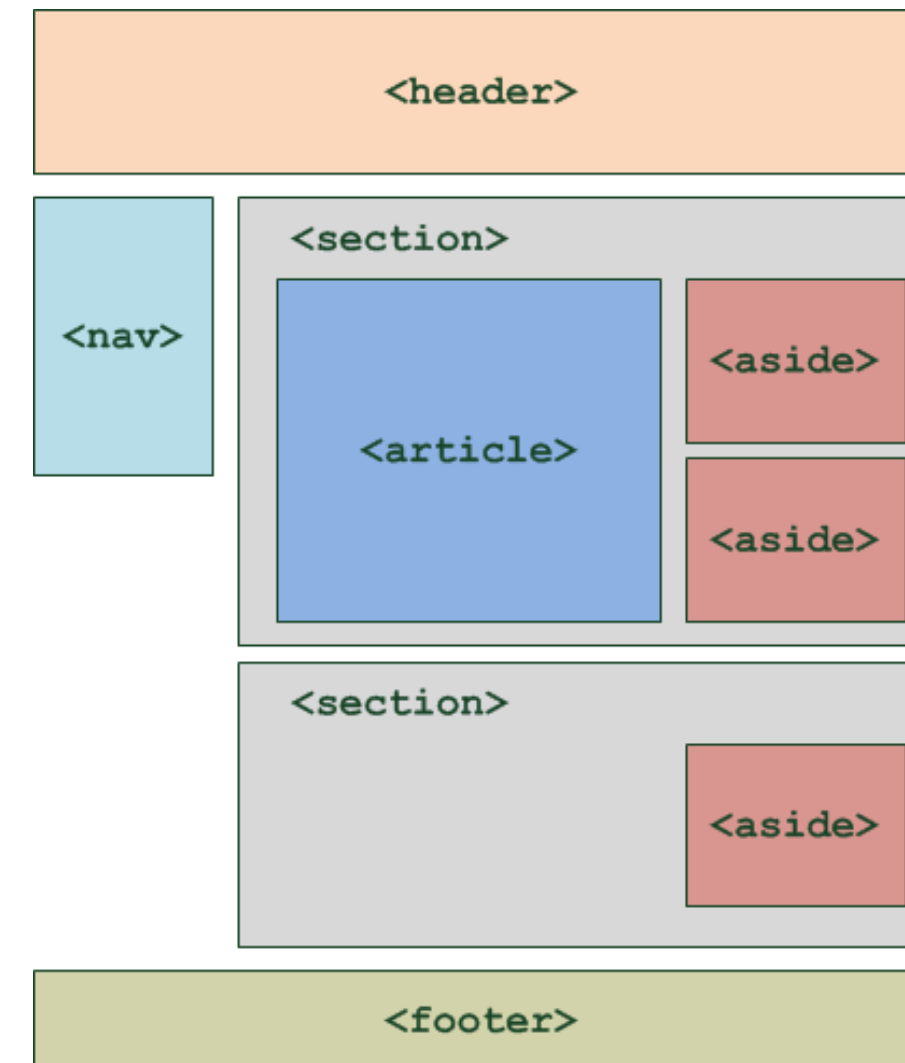
# Rappels - Structure HTML 5 - Footer

- Contenu du pied de page
- Contact
- Mentions légales
- Mentions d'auteurs



# Rappels - Structure HTML 5

- Nav
  - Principaux liens de navigation (Menu)
- Section
  - Bloc de contenu (thématique)
- Article
  - Contenu indépendant dans
    - Une page
    - Une section
    - Prévu pour être réutilisable
- Aside
  - Bloc de contenu complémentaire



# Rappels - Balises & Commentaires

- Balises encadrante / paire
  - `<div></div>`
  - `<h1></h1>`
- Balises orphelines
  - `<img src="" />`
  - `<br>` ou `<br />`
- Attributs
  - `<a href="..."></a>`
- Commentaire
  - `<!-- commentaire html -->`
  - Visible dans les sources de la page

# Rappels - Titres

---

- Balises h1 à h6
- Critique pour le référencement
  - Pas de sauts
  - Importance croissante pour le référencement (h1 > h2 ...)
  - Une seule balise h1
- h1
  - Unique
  - Pas trop longue (<100 caractères en général)



# Rappels - Listes

- Liste non ordonnée
  - Syntaxe : `<ul><li>élément</li></ul>`
  - Souvent utilisé pour construire des composants
    - Menu
    - Liste déroulantes (dropdown)
- Liste ordonnée
  - Syntaxe : `<ol><li>élément</li></ol>`

```
1 <ul>
2   <li>Element</li>
3   <li>Element</li>
4   <li>Element</li>
5   <li>Element</li>
6 </ul>
7 <ol>
8   <li>Element ordonné</li>
9   <li>Element ordonné</li>
10  <li>Element ordonné</li>
11  <li>Element ordonné</li>
12 </ol>
```

- Element
- Element
- Element
- Element

1. Element ordonné
2. Element ordonné
3. Element ordonné
4. Element ordonné

# Rappels - Liens

---

```
<!-- Lien classique-->
<a href="http://www.google.com/">Lien vers Google</a>
<!-- Ouvre le lien dans un nouvel onglet-->
<a href="http://www.google.com/" target="_blank">Lien vers Google</a>
<!-- Ancre-->
<a href="#footer">Ancre menant au bloc ayant l'id "footer"</a>
<!-- Mailto-->
<a href="mailto:greg.galli@gmail.com">Lien ouvrant l'application de mail par défaut</a>
<!-- Téléphone-->
<a href="tel:+33687179627">Lien ouvrant une application d'appel sur le numéro indiqué</a>

<!-- Liens spécifiques-->
<a href="callto:greg.galli.tokidev">Lien skype</a>
<a href="ts3server://ts3.serveur.com?port=9876">Lien TS3</a>
```

# Rappels - Formulaire

- Syntaxe : `<form></form>`
- **action**
  - URI du programme cible qui traitera les informations soumises par le formulaire
- **method**
  - Méthode HTTP à utiliser pour envoyer les données au serveur
  - **GET**
    - Toutes les valeurs du formulaire seront encodées dans l'url et concaténées à l'action après un « ? »
    - Limite de taille
  - **POST**
    - Toutes les valeurs du formulaire seront envoyées dans le corps de la requête
    - Passage obligé pour l'envoi de fichiers
    - Certains frameworks proposent des méthodes pour simuler des envoi de formulaires en PUT / DELETE mais ce n'est pas supporté par défaut par HTML
    - Pas de limite de taille
- Chaque input devrait être accompagné d'un label ciblant l'id de l'input correspondant

# Rappels - Formulaires






---

## ➤ **enctype**








- Attribut de formulaire accessible pour les formulaire en POST (method="POST")
- Permet de préciser l'encodage des form-data
- `application/x-www-form-urlencoded`
  - Les valeur sont encodées sous forme de couples clef-valeur séparés par un "&" avec un "=" entre la clef et la valeur
- `multipart/form-data`
  - Format permettant l'envoi de fichier, sans cet enctype défini, la soumission du formulaire ne transmettra pas les données binaires du fichier renseigné dans le formulaire
- `text/plain`
  - Format classique pour l'envoi de données textuelles

# Rappels - Formulaires - Input types

- `<input type="xxx" />`
  - Défini le champs et le type de donnée pour un affichage adapté par les navigateurs

Value	Description
<u><a href="#">button</a></u>	Defines a clickable button (mostly used with a JavaScript to activate a script)
<u><a href="#">checkbox</a></u>	Defines a checkbox
<u><a href="#">color</a></u>	 Defines a color picker
<u><a href="#">date</a></u>	 Defines a date control (year, month, day (no time))
<u><a href="#">datetime-local</a></u>	 Defines a date and time control (year, month, day, time (no timezone))
<u><a href="#">email</a></u>	 Defines a field for an e-mail address
<u><a href="#">file</a></u>	Defines a file-select field and a "Browse" button (for file uploads)
<u><a href="#">hidden</a></u>	Defines a hidden input field
<u><a href="#">image</a></u>	Defines an image as the submit button
<u><a href="#">month</a></u>	 Defines a month and year control (no timezone)

# Rappels - Formulaire - Input types

Value	Description
<u>number</u>	 Defines a field for entering a number
<u>password</u>	Defines a password field
<u>radio</u>	Defines a radio button
<u>range</u>	 Defines a range control (like a slider control)
<u>reset</u>	Defines a reset button
<u>search</u>	 Defines a text field for entering a search string
<u>submit</u>	Defines a submit button
<u>tel</u>	 Defines a field for entering a telephone number
<u>text</u>	Default. Defines a single-line text field
<u>time</u>	 Defines a control for entering a time (no timezone)
<u>url</u>	 Defines a field for entering a URL
<u>week</u>	 Defines a week and year control (no timezone)



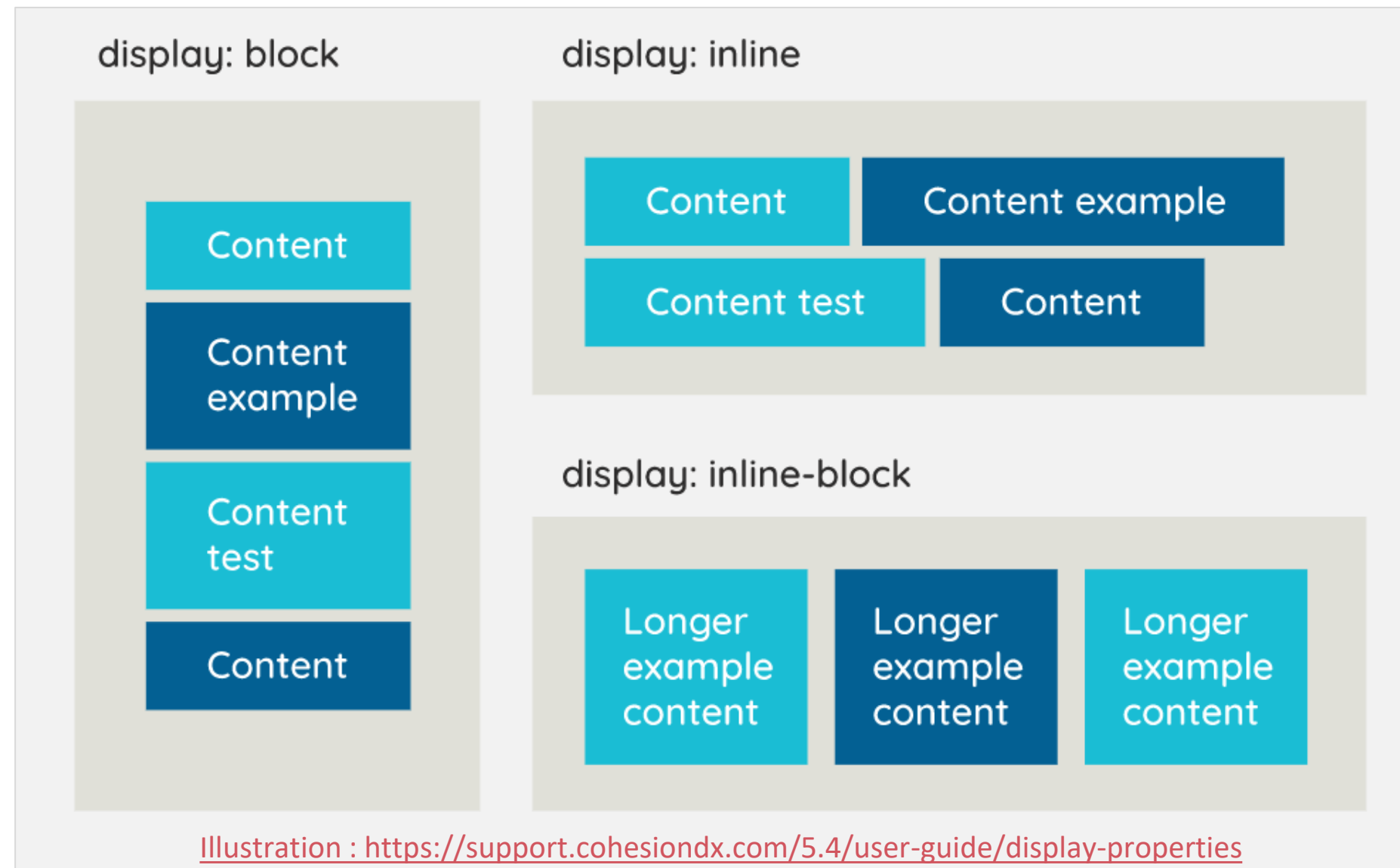
# Rappels - Types de balises

---

- Deux familles principales
- Type « **block** »
  - Éléments : div / h1 .. h6 / p / ul / ol / li / table / form ...
  - Retour à la ligne avant et après
  - Occupe toute la largeur de son parent
  - Occupe la hauteur de ses enfants
- Type « **inline** »
  - Éléments : span / img / input / a
  - Généralement dans un élément de type block (quelques exceptions)
  - Vient à la suite de l'élément précédent si possible (si type inline par exemple)
  - Possibilité de gérer le « vertical-align »
  - Occupe l'espace pris par le contenu
  - Ignore toute définition de hauteur / largeur

# Rappels - Types de balises

- Surcharge possible via la propriété CSS « display »





# CSS 3 & Responsive design

Rappels

- Feuilles de style en cascade (Cascading Style Sheets)
- Langage décrivant la présentation de documents HTML
- Plusieurs moyens de mise en œuvre
  - Dans la partie « head » du document HTML
    - Favorisé dans certains cas (merci la SEO)
  - Dans un attribut « style » directement dans les balises
    - Favorisé dans certains cas (merci la SEO bis)
  - Fichier référencé depuis le document HTML
    - Factorisation des déclarations
    - Mutualisation pour utiliser un même fichier depuis plusieurs / toutes les pages
    - Déclaration centralisées pour une mise à jour simple

**CSS**



# CSS - Objectifs

---

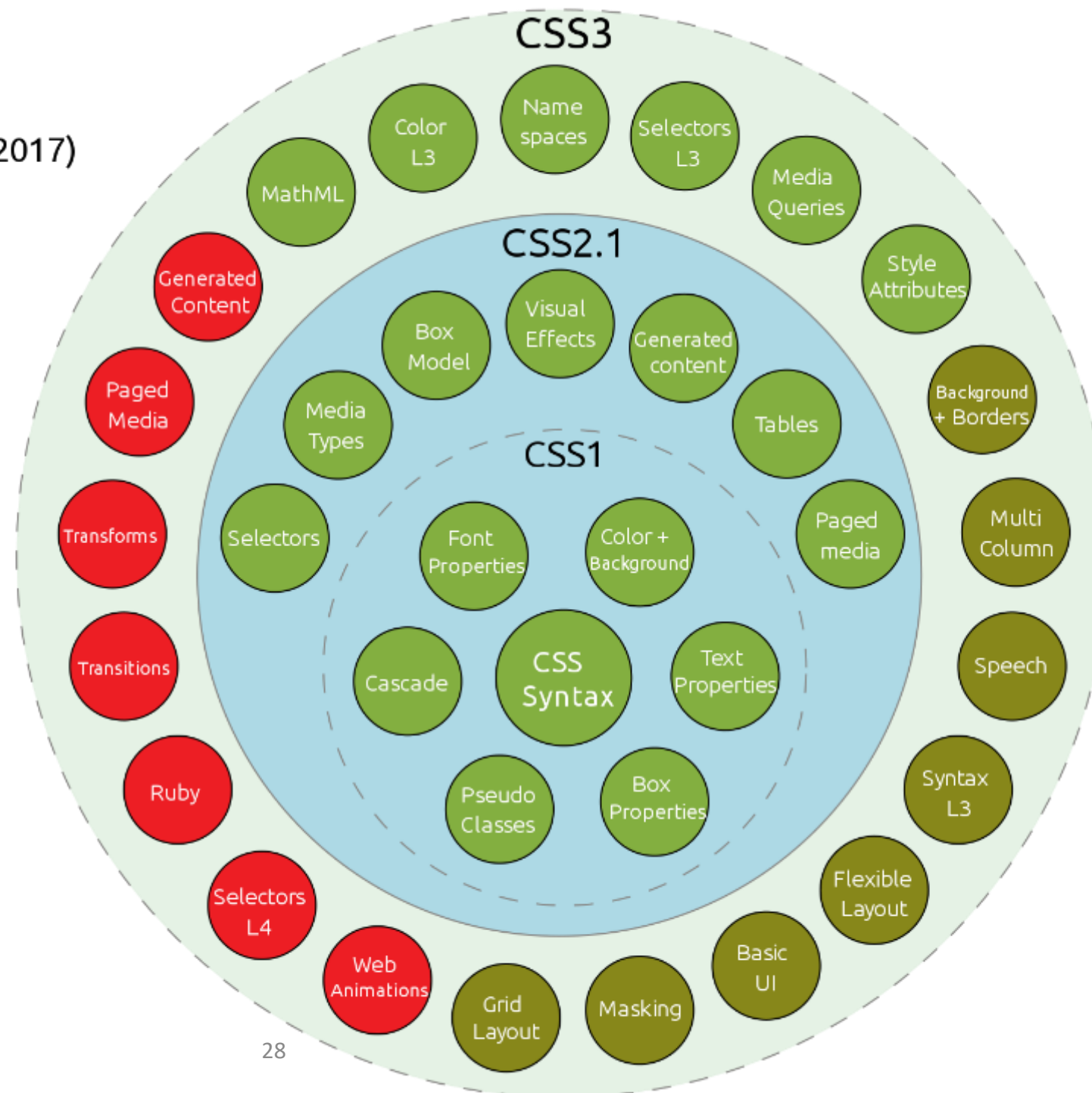
- Segmenter la structure du document de sa mise en forme
  - Pas de mélange de langages
  - Deux parties pouvant être gérées facilement par deux personnes différentes
- Diminue la taille des fichiers HTML
  - Favorise le référencement
- Limiter le recours aux solutions de contournement (animations javascript...)
  - Diminue la consommation de ressource du poste client
- Proposer des versions alternatives en fonction du moyen de consultation
  - Propose une ergonomie adaptée dans chaque cas

# CSS 3 - Apports

## CSS3

Taxonomy & Status (September 2017)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



# CSS 3 - Apports

---

- Avancée majeure pour le développement d'applications web mobiles ou responsives
- Media queries
  - Possibilité de définir des contraintes de résolution et types d'affichage pour la prise en compte de déclarations de styles
- Flexible layout
  - Introduction d'un nouveau mode d'affichage qui résoud de nombreux problème qui étaient jusque là « contournés »
- Grid layout
  - Intégration native à CSS de nombreuses fonctionnalités que l'on allait jusque là chercher dans des librairies et frameworks additionnels

# CSS - Principe & Sélecteurs

---

- Basé sur un système de sélecteurs auxquels sont appliqués des propriétés paramétrées
- Sélecteurs
  - Balise : on peut simplement cibler une balise depuis son nom
  - « id » d'un élément
    - L'« id » est sensé être unique au sein d'une même page
    - Permet de cibler précisément un élément
  - « class » d'un / plusieurs éléments
    - La classe est faite pour catégoriser un ensemble d'éléments ayant la même fonction / présentation
    - Permet de cibler tous les éléments concernés
  - Sélecteur d'état
    - Permet d'apporter des précisions sur les états ciblés (:hover)

# CSS - Sélecteurs avancés

```
h1, p
{
    /* Appliquera les règles aux titres de niveau 1 (h1) et aux paragraphes (p) */
}

h1 + p
{
    /* Appliquera les règles aux paragraphes (p) se trouvant après un titre de niveau 1 (h1) */
}

h1 span
{
    /* Appliquera les règles aux spans se trouvant dans un titre de niveau 1 (h1) */
}

a[title]
{
    /* Appliquera les règles aux liens (a) possédant un attribut "title" */
}
```

# CSS - Margin, Border, Padding





# CSS 3 - Responsive design

- Adaptation du site afin d'obtenir un rendu idéal et adapté à chaque résolution



# CSS 3 & Responsive design

Media Queries

# CSS 3 - Responsive design - Media Queries

---

- Introduit dans CSS 3
- Application de feuilles de style / règles en fonction
  - Du périphérique de consultation utilisé
  - De la résolution du périphérique de consultation
- Adaptation dynamique du design purement via CSS
- Une seule et même source pour
  - Tous les périphériques
  - Plusieurs ergonomies potentiellement très différentes

# CSS 3 - Responsive design - Media Queries

---

- Syntaxe : « @media » suivi du type de périphérique
- Expression initiale déterminant l'inclusion ou l'exclusion
  - Une media query est un test renvoyant « vrai » ou « faux » en fonction de l'environnement
- Utilisation d'opérateurs logiques pour définir les conditions
  - and : « et » logique
    - Permet d'enchaîner les conditions
  - only : uniquement
    - Permet de cacher les feuilles de styles pour les navigateurs les plus anciens
  - not : négation
    - Applique une négation sur l'expression qui suit
  - « , » : opérateur de combinaison, interprété comme un « ou » logique
    - Permet de spécifier plusieurs requêtes sur des cas distincts

# CSS 3 - Responsive design - Media Queries

- « @media » peut prendre les valeurs suivantes
  - all : tous les appareils
  - screen : tous les écrans
  - print : matériaux paginés, aperçu avant impression
  - speech : outils de synthèse vocale

```
/* Cible les écrans */  
@media screen { }  
  
/* Cible les écrans et les matériaux paginés */  
@media screen, print { }  
  
/* Cible tout sauf les écrans */  
@media not screen { }
```



# CSS 3 - Responsive design - Media Queries

- Les conditions se construisent sur des critères (features)
  - width : largeur de la zone d'affichage (viewport)
  - height : hauteur de la zone d'affichage (viewport)
  - aspect-ratio : rapport hauteur/largeur de la zone d'affichage (viewport)
  - orientation : orientation de la zone d'affichage (mobile)
  - resolution : densité en pixel
  - autres moins utilisées : scan, grid, update, overflow-block, overflow-inline, color, color-gamut, color-index, display-mode, monochrome, inverted-colors, pointer, hover, any-pointer, any-hover, light-level, prefers-reduced-motion, prefers-reduced-transparency, prefers-contrast, prefers-color-scheme, forced-colors, scripting
- Préfixes
  - « min- » et « max- » permettent de préfixer la plupart des critères qui acceptent des valeurs numériques

# CSS 3 - Responsive design - Media Queries

```
/* Cible les écrans avec une viewport inférieur à 640 pixels de largeur */  
/* Concernera généralement les navigateurs mobiles */  
@media screen and (max-width: 640px) { }  
  
/* Cible les écrans avec une viewport supérieur à 800 pixels de largeur */  
/* Concernera principalement les tablettes en orientation paysage */  
@media screen and (min-width: 800px) { }  
  
/* Cible les écrans avec une viewport de largeur entre 1024 et 1280 pixels */  
/* Concernera principalement les vieux écrans et tablettes en paysage */  
@media screen and (min-width: 1024px) and (max-width: 1280px) { }  
  
/* Cible les écrans en mode portrait, principalement mobile / tablette */  
@media screen and (orientation:portrait) { }
```

# CSS - Inclusion

---

- Inclusion dans les pages via une balise « link » dans la balise « <head> »
- Media query
  - Dans l'inclusion (1)
  - Dans le fichier CSS (2)
- Le CSS est lu de manière séquentielle
- Les déclarations qui viennent ensuite peuvent modifier une définition faite plus tôt



# CSS - Inclusion

```
// (1) Inclusion du fichier avec contrainte
// index.html
<link rel="stylesheet" type="text/css" href="style.css" media="screen and (max-width: 640px)" />

// style.css
body { background: red; }

// (2) Inclusion du fichier brut qui contiendra la media query
// index.html
<link rel="stylesheet" type="text/css" href="style.css" />

// style.css
@media screen and (max-width: 640px) {
    body { background: red; }
}
```

# CSS 3 & Responsive design

Flexible Layout

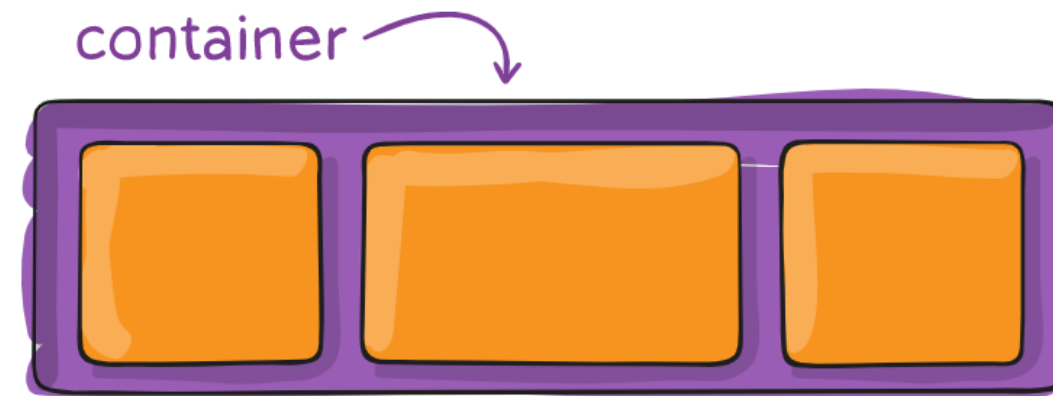
# CSS - Flexible Layout

---

- Ensemble de règles CSS
- Optimisation de l'espace
  - Adaptation automatique en fonction de l'affichage
- Facilite l'organisation des blocs composant la page
  - Gestion de l'ordre d'affichage des éléments
- Gestion de la direction
  - vertical
  - horizontal

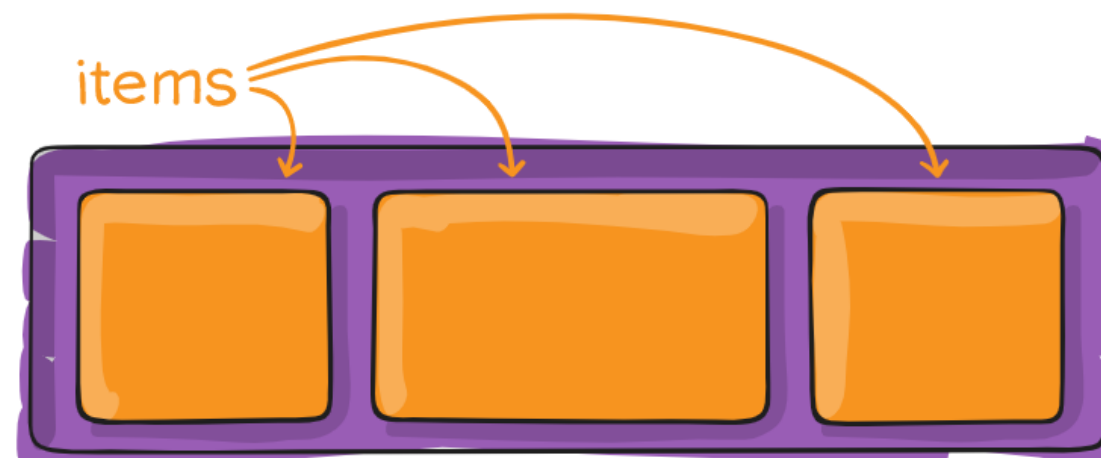
# CSS - Flexible Layout - Structure

## ➤ Container



**Properties for the Parent  
(flex container)**

## ➤ Items



**Properties for the Children  
(flex items)**

# CSS - Flexible Layout - Définitions

## ➤ Container : « display »

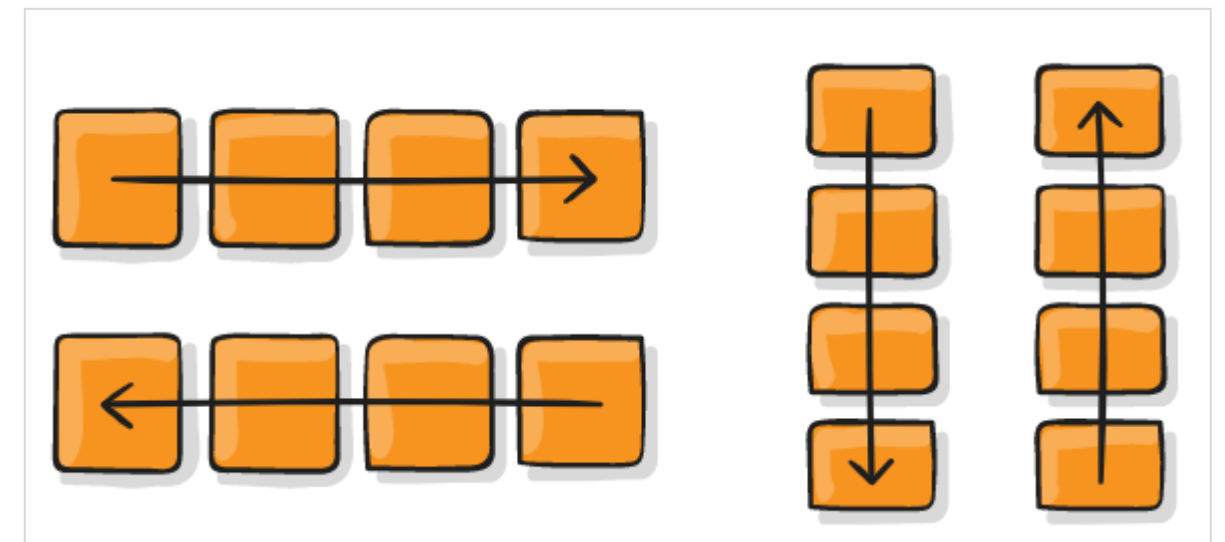
- flex
- inline-flex

```
.container {  
  display: flex; /* ou inline-flex */  
}
```

## ➤ Axe d'affichage : « flex-direction »

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

```
.container {  
  flex-direction: row | row-reverse | column | column-reverse;  
}
```



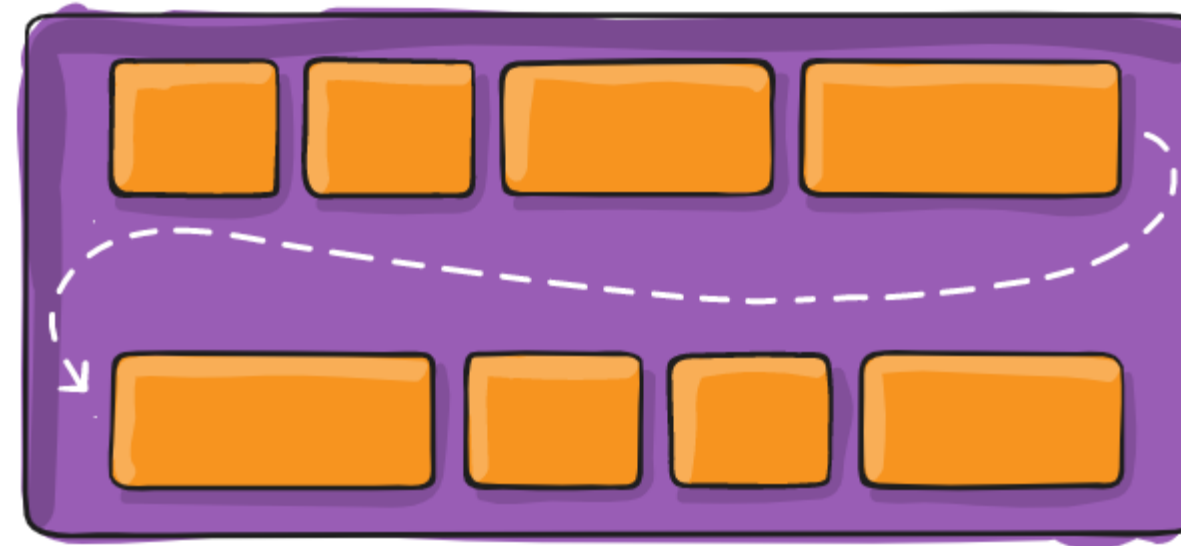
# CSS - Flexible Layout - Définitions

## ➤ Wrap

- Défini le comportement à suivre lorsque le contenu arrive en fin de ligne/colonne

## ➤ Options

- nowrap (défaut)
  - En une seule ligne
  - Peut entraîner un dépassement
- wrap
  - Sur plusieurs lignes
- wrap-reverse
  - Comme « wrap » mais inversé

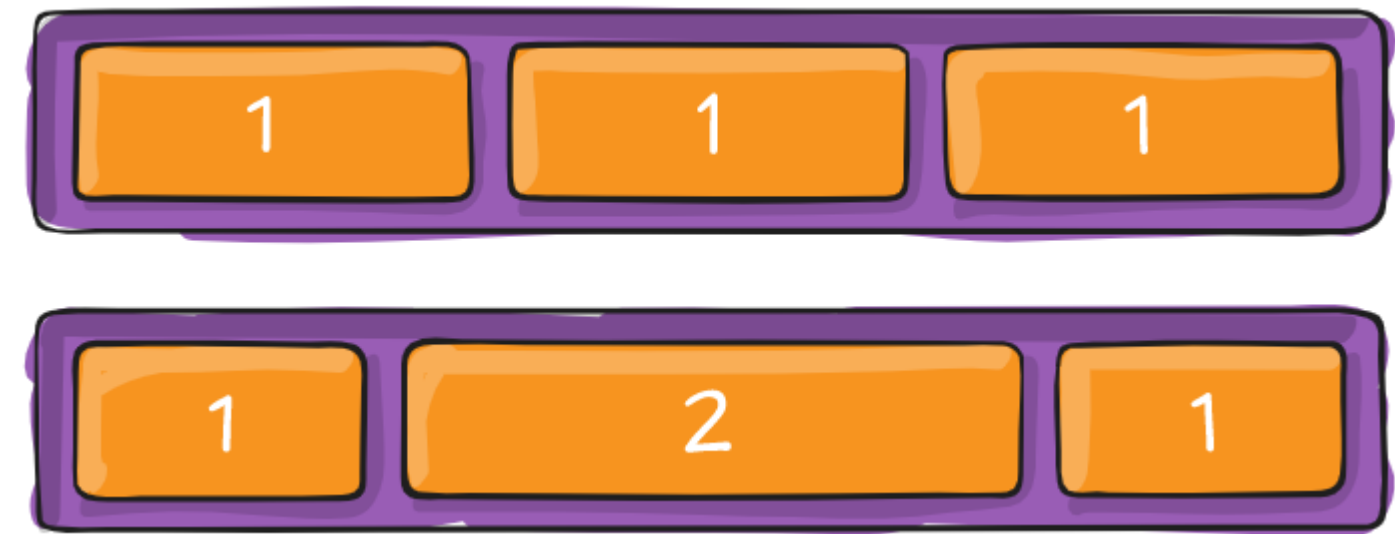


```
.container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

# CSS - Flexible Layout - Définition

- Gestion de l'espace via « flex-grow » et « flex-shrink »
- Défini la largeur / hauteur maximale d'un élément
- Fonctionne sur un système de pondération

```
.item {  
    flex-grow: <number>; /* default 0 */  
}  
  
.item {  
    flex-shrink: <number>; /* default 1 */  
}
```



# CSS - Flexible Layout - Définition

- « flex-basis » : Spécifie la taille par défaut d'un élément
- Intervient avant que l'espace restant soit distribué
- Intervient sur la width ou la height de l'élément

```
/* On définit une largeur */  
flex-basis: 10em;  
flex-basis: 3px;  
flex-basis: auto;  
  
/* On utilise les dimensions */  
/* intrinsèques avec des mots-clés */  
flex-basis: fill;  
flex-basis: fit-content;  
  
/* La taille se calcule automatiquement */  
/* en fonction du contenu de l'élément */  
flex-basis: content;
```



# CSS - Flexible Layout - Définition

- « justify-content »
  - Défini l'alignement sur l'axe principal
- Aide à distribuer l'espace restant

```
.container {  
    justify-content: flex-start | flex-end |  
    center | space-between | space-around | space-  
    evenly | start | end | left | right ... + safe |  
    unsafe;  
}
```

flex-start



flex-end



center



space-between



space-around



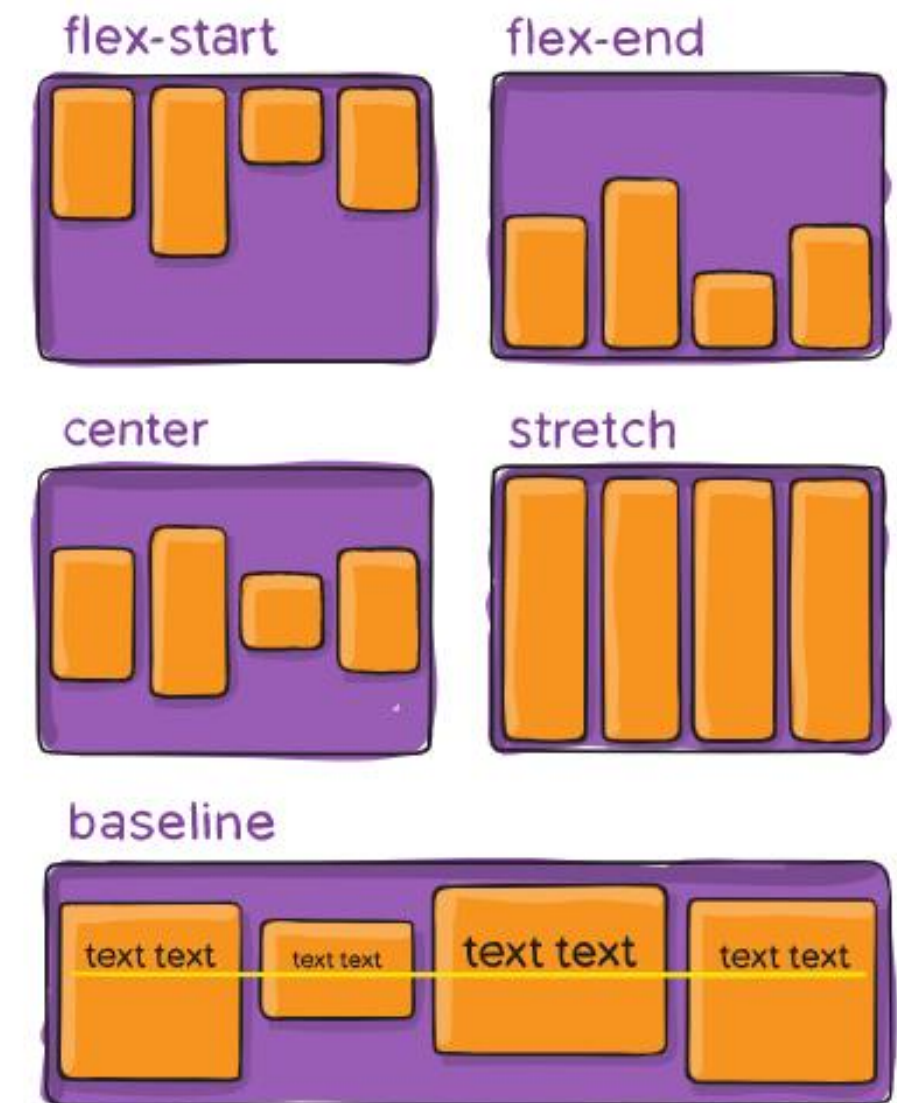
space-evenly



# CSS - Flexible Layout - Définition

- « align-items »
  - Défini l'alignement par rapport à l'axe perpendiculaire
- Aide à distribuer l'espace restant
- Equivalent à « justify-content » pour l'axe perpendiculaire

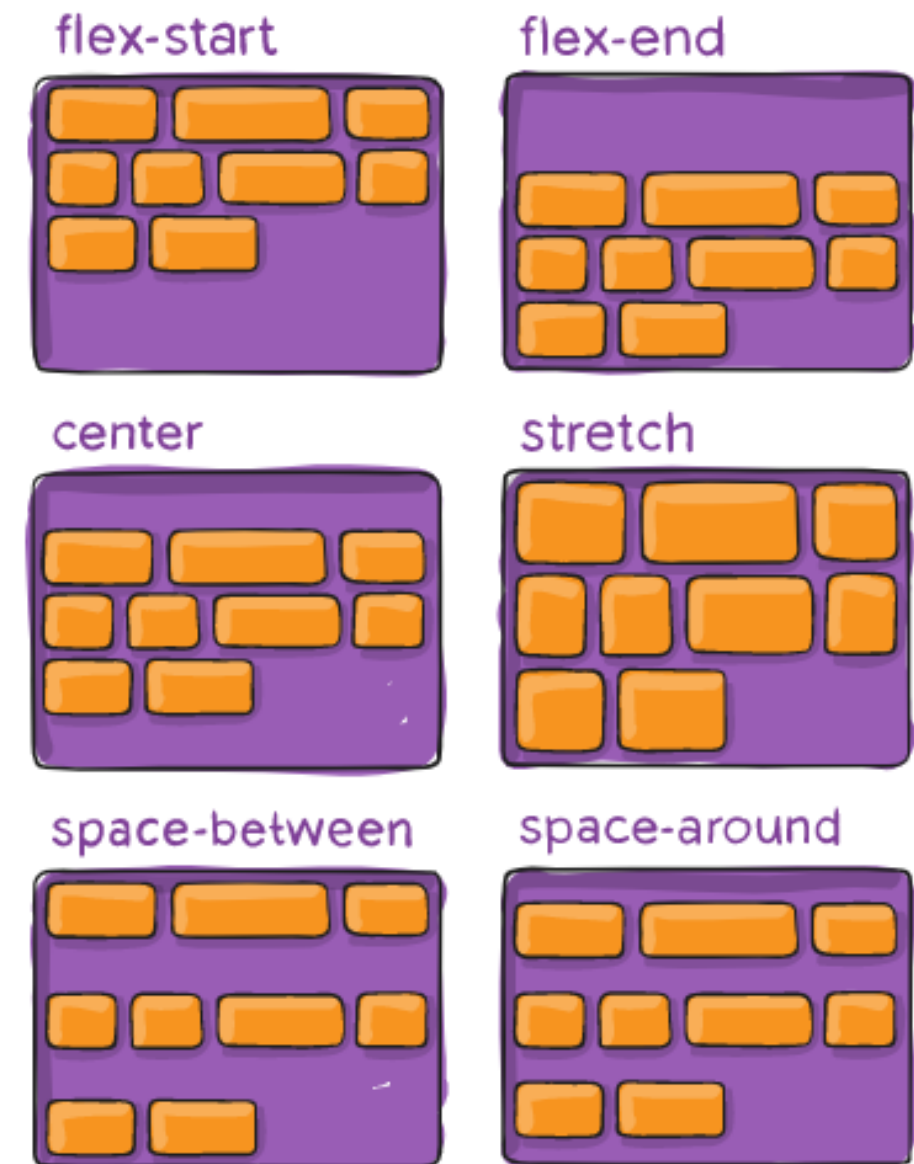
```
.container {  
    align-items: stretch | flex-start | flex-end  
    | center | baseline | first baseline | last  
    baseline | start | end | self-start | self-end  
    + ... safe | unsafe;  
}
```



# CSS - Flexible Layout - Définition

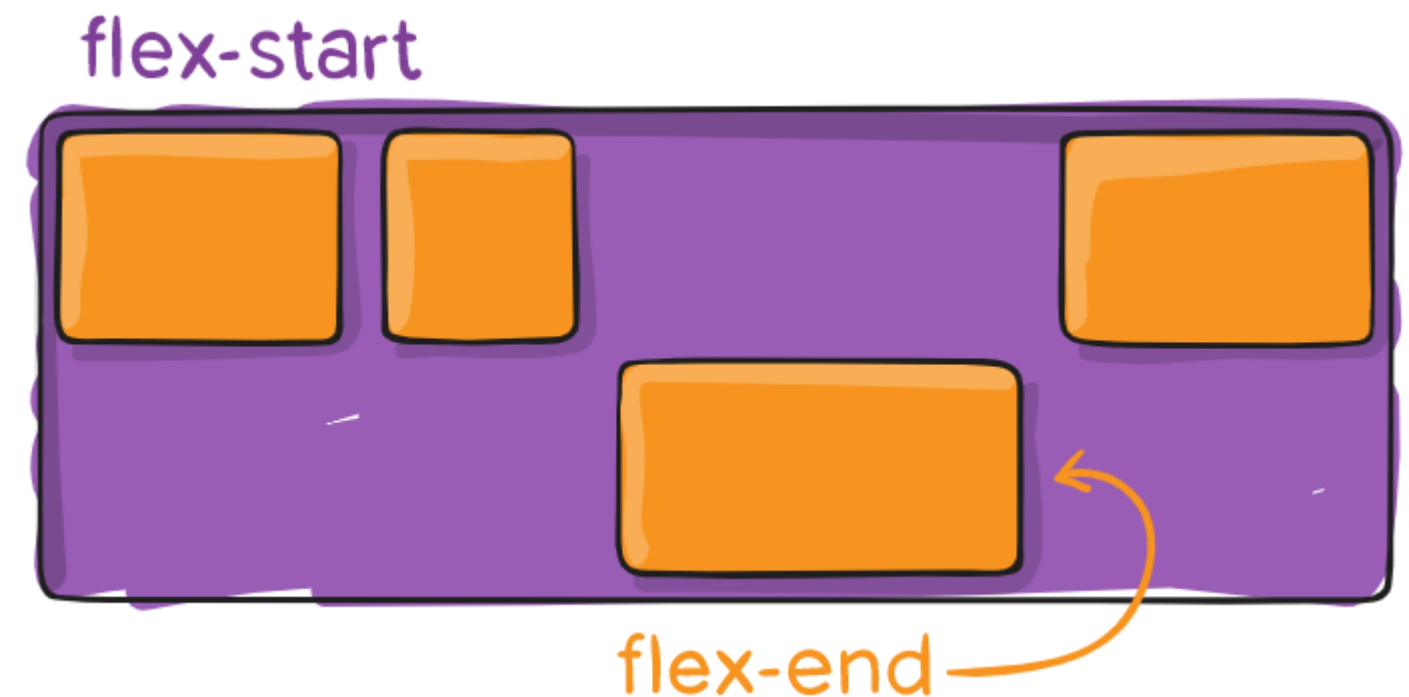
- « align-content »
  - Défini l'alignement des lignes d'un conteneur dans l'espace restant

```
.container {
  align-content: flex-start | flex-end |
  center | space-between | space-around | space-
  evenly | stretch | start | end | baseline |
  first baseline | last baseline + ... safe |
  unsafe;
}
```



# CSS - Flexible Layout - Définition

- « align-self »
  - Permet de surcharger l'alignement pour un élément
    - Alignement par défaut
    - Alignement depuis « align-items »



# CSS - Flexible Layout - Définition

- « flex » : Raccourcis vers les propriétés
  - « flex-grow »
  - « flex-shrink »
  - « flex-basis »
  - valeurs par défaut : « flex: 0 1 auto »

```
.item {  
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]  
}
```

# CSS - Flexible Layout - Définitions

- « flex-flow » : Raccourcis vers les propriétés
  - « flex-direction » : « row » par défaut
  - « flex-wrap » : « nowrap » par défaut

**flex-flow:** <'flex-direction'> || <'flex-wrap'>

```
/* flex-flow: <'flex-direction'> */  
flex-flow: row;
```

```
/* flex-flow: <'flex-wrap'> */  
flex-flow: nowrap;
```

```
/* flex-flow: <'flex-direction'> et <'flex-wrap'> */  
flex-flow: row nowrap;
```

# CSS 3 & Responsive design

Grid Layout

# CSS - Grid Layout - Introduction

---

- Système de layout le plus puissant disponible dans CSS
- Système basé sur deux dimensions (lignes / colonnes)
- Révolution totale dans le domaine de l'intégration web
  - Utilisation de tableaux
  - Utilisations de « div » avec des propriétés « float »
  - Utilisation des Flexible Layout qui n'était pas fait pour ça
  - Ces solutions étaient des contournements pour des problèmes quotidiens
- Vient remplacer de nombreuses librairies utilisées jusque là pour répondre à ce besoin



# CSS - Grid Layout - Introduction

- Pour commencer à utiliser les « Grid Layout » il faudra définir un conteneur avec une propriété « display » définie à la valeur « grid »

```
.container {  
    display: grid | inline-grid;  
}
```

- Il faudra ensuite définir les tailles des lignes et colonnes avec les propriétés suivantes

```
.container {  
    grid-template-columns: <track-size> ... | <line-name> <track-size> ...;  
    grid-template-rows: <track-size> ... | <line-name> <track-size> ...;  
}
```

# CSS - Grid Layout - Terminologies

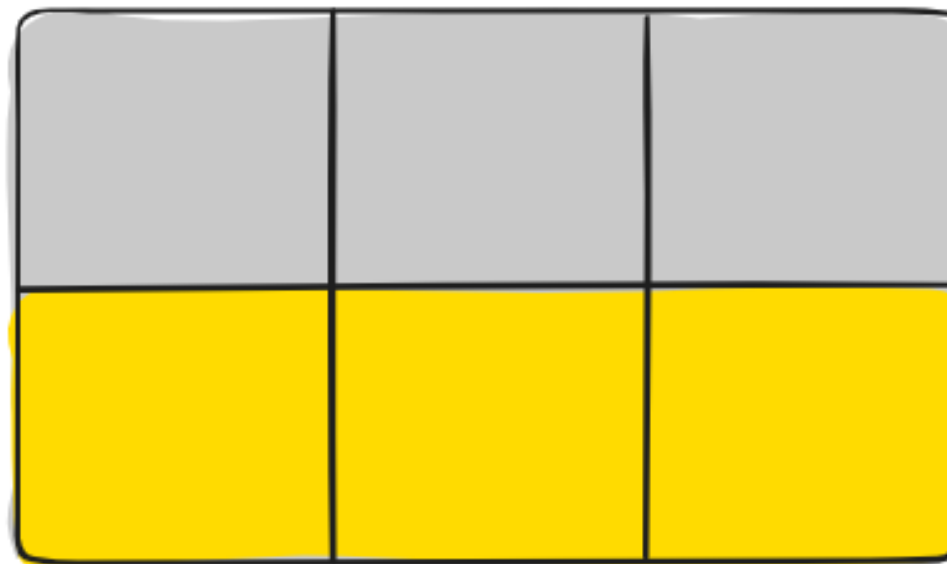
- Certaines terminologies sont importantes, voici les principales
- Grid Line : Ligne de grille
  - Il s'agit de la ligne à proprement parler, celle qui structure la grille
  - Elles peuvent être horizontales ou verticales selon s'il s'agit de ligne de « colonne » ou de ligne de « rangée » (rangée que l'on appelle souvent « ligne »)



# CSS - Grid Layout - Terminologies

---

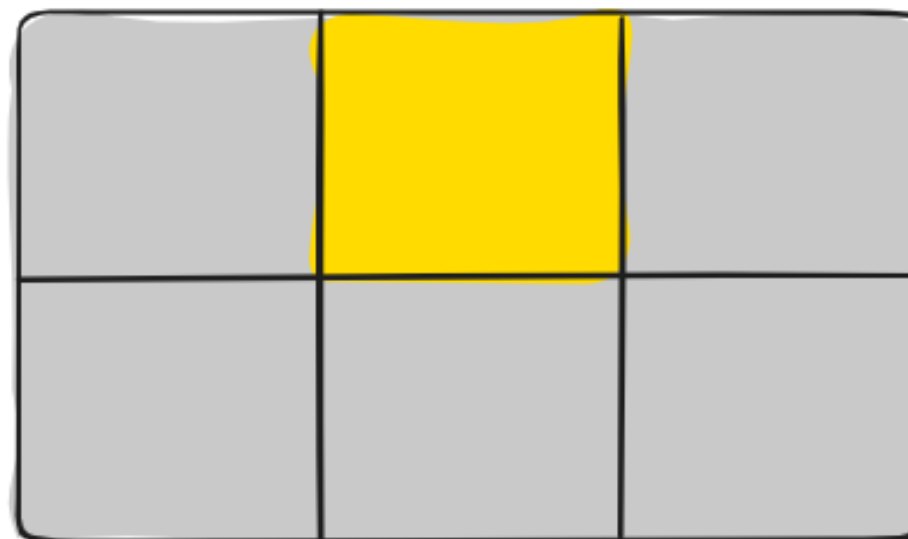
- Grid Track : Piste de la grille
  - Défini l'espace entre deux lignes adjacentes
  - Il peut s'agir d'une colonne ou d'une rangée de la grille
  - L'illustration ci-dessous met en avant la piste entre la 2<sup>ème</sup> et la 3<sup>ème</sup> ligne de la grille



# CSS - Grid Layout - Terminologies

---

- Grid Cell : Cellule de grille
  - Défini l'espace entre deux lignes de rangée adjacentes et deux lignes de colonne adjacentes
  - Il s'agit de la plus petite unité au sein d'une grille
  - L'illustration ci-dessous mets en avant la cellule entre les lignes de rangées 1 et 2 et les lignes de colonne 2 et 3



# CSS - Grid Layout - Columns & Rows

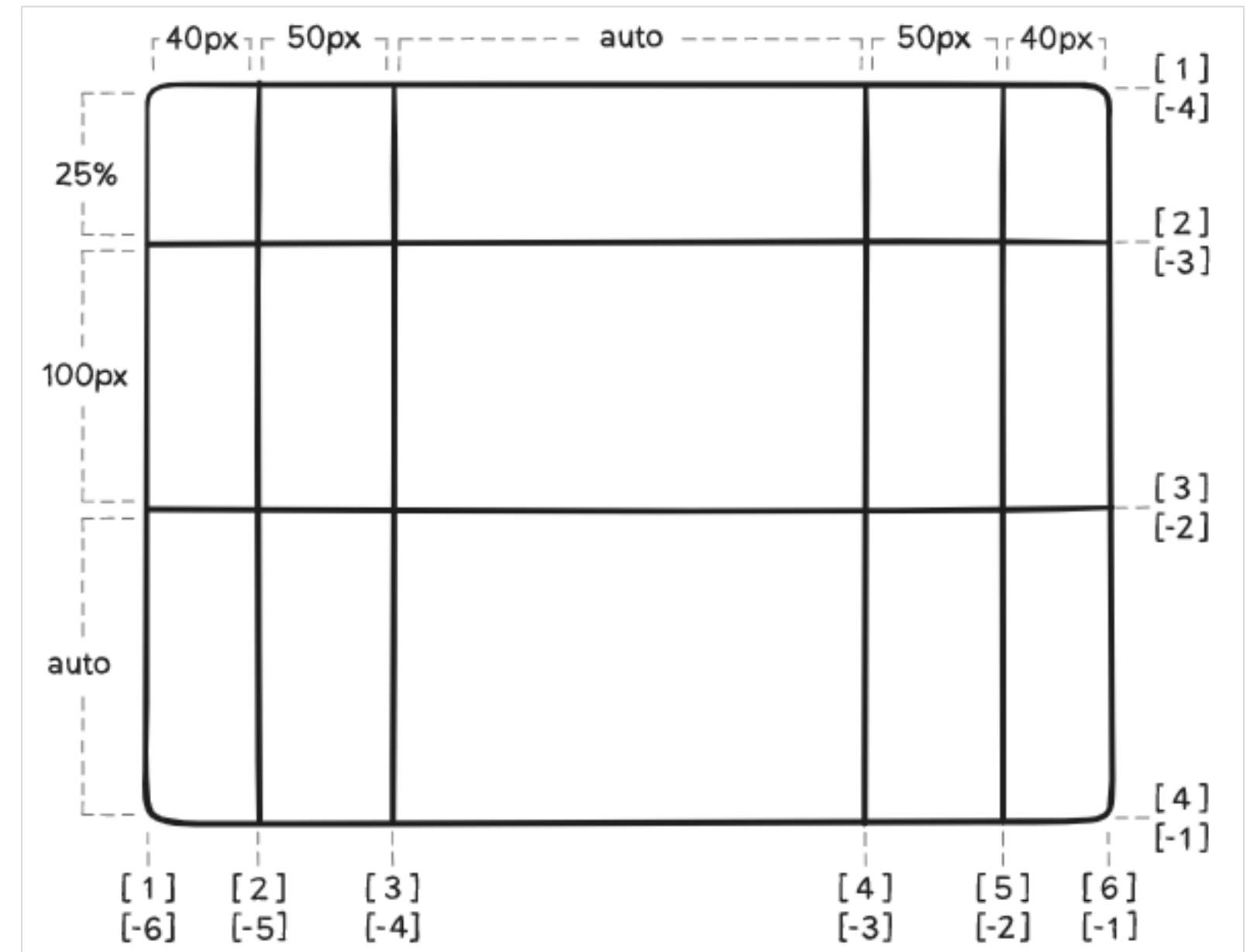
## ➤ Exemple

```
.container {
  grid-template-columns: 40px 50px auto 50px 40px;
  grid-template-rows: 25% 100px auto;
}
```

## ➤ Nommage possible

```
.container {
  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];
}
```



# CSS - Grid Layout - Columns & Rows

- Utilisation possible d'une propriété « repeat » pour dupliquer une déclaration

```
.container {  
  grid-template-columns: repeat(3, 20px [col-start]);  
  /* Est équivalent à */  
  grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];  
}
```

- L'unité « fr » permet de définir la taille d'une piste comme étant une fraction de l'espace libre dans le conteneur

```
.container {  
  /* Défini chaque piste comme occupant un tiers de la largeur du conteneur */  
  grid-template-columns: 1fr 1fr 1fr;  
}
```

# CSS - Grid Layout - Columns & Rows

- Si l'espace libre est réparti entre des **éléments flexibles** et d'autres **ne l'étant pas**, le calcul de l'espace disponible pour les **éléments flexibles** sera fait **après** avoir réservé l'espace pour **les éléments non flexibles**.

```
.container {  
    /* L'espace libre calculé exclu les 50 pixels réservés  
       pour le 2ème élément qui a une définition fixe  
    */  
    grid-template-columns: 1fr 50px 1fr 1fr;  
}
```

# CSS - Grid Layout - Columns & Rows

- Si l'espace libre est réparti entre des **éléments flexibles** et d'autres **ne l'étant pas**, le calcul de l'espace disponible pour les **éléments flexibles** sera fait **après** avoir réservé l'espace pour **les éléments non flexibles**.

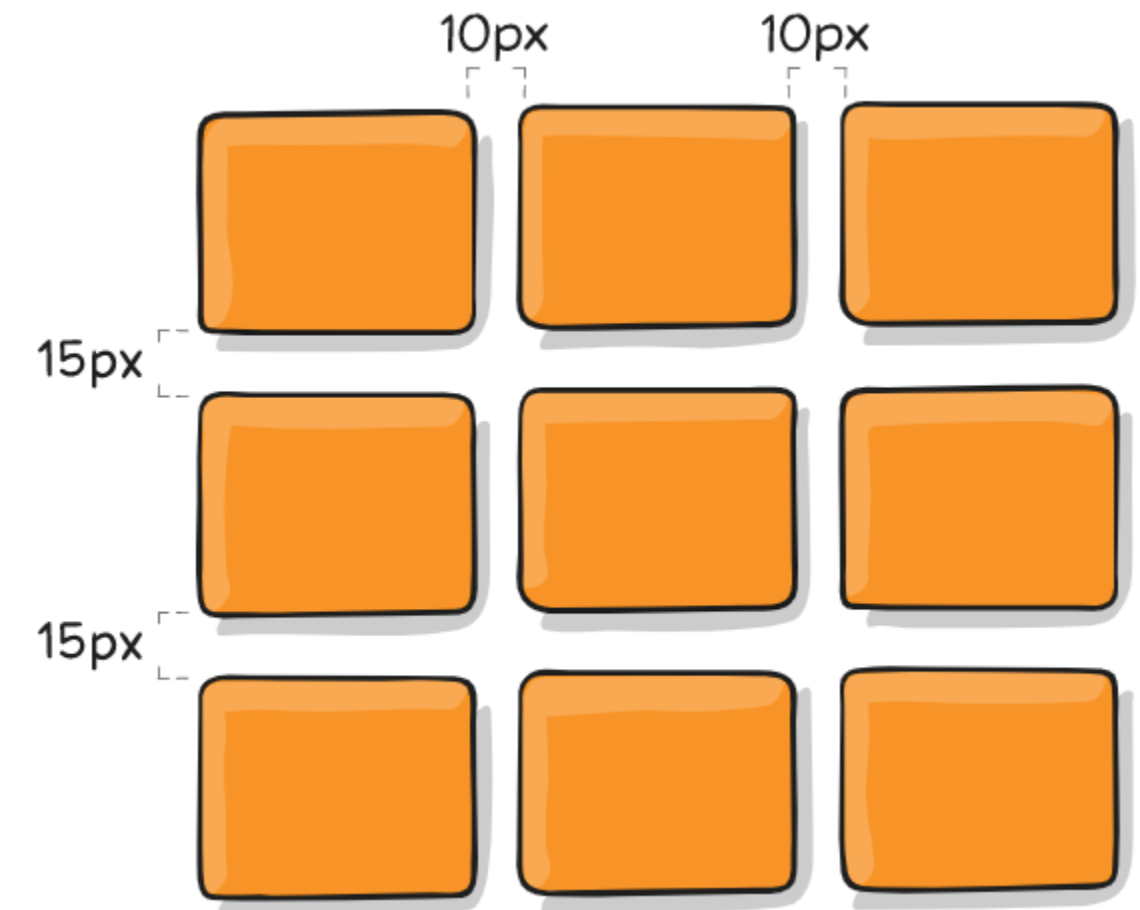
```
.container {  
    /* L'espace libre calculé exclu les 50 pixels réservés  
       pour le 2ème élément qui a une définition fixe  
    */  
    grid-template-columns: 1fr 50px 1fr 1fr;  
}
```



# CSS - Grid Layout - Columns & Rows

- Possibilité de spécifier les espaces entre les cellules
  - « grid-column-gap »
  - « grid-row-gap »
  - Concrètement augmente l'épaisseur des lignes séparant les cellules

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



# CSS - Grid Layout - Template Area

---

- Permet de faire le lien entre notre HTML et le CSS
- Le CSS déclare la structure de la grille mais n'a pas de lien concret avec le HTML et donc le contenu à afficher
- La propriété « grid-area » permet de faire le lien entre le HTML et le CSS
  - Les noms sont arbitraires
  - Une cellule n'ayant pas de bloc associé peut être remplacé par un « . »
- La propriété « grid-template-areas » permet de structurer la disposition de ces éléments

# CSS - Grid Layout - Template Area - Exemple

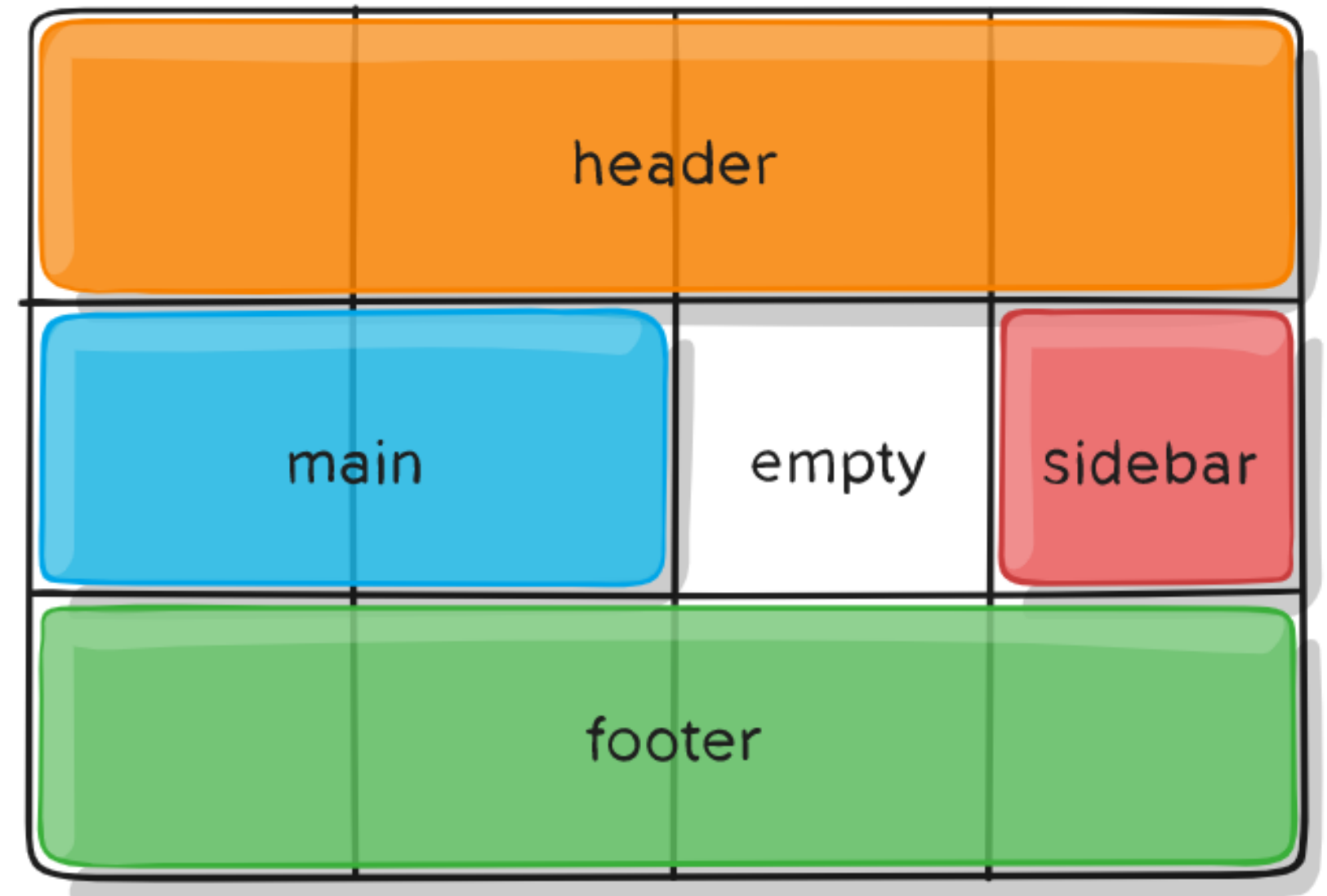
```
<div class="container">
  <div class="item-a">
    Header
  </div>
  <div class="item-b">
    Main
  </div>
  <div class="item-c">
    Sidebar
  </div>
  <div class="item-d">
    footer
  </div>
</div>
```

```
.item-a { grid-area: header; }
.item-b { grid-area: main; }
.item-c { grid-area: sidebar; }
.item-d { grid-area: footer; }

.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```

# CSS - Grid Layout - Template Area - Exemple

```
.item-a { grid-area: header; }  
.item-b { grid-area: main; }  
.item-c { grid-area: sidebar; }  
.item-d { grid-area: footer; }  
  
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}
```



# CSS - Grid Layout - Disposition - *justify-items*

- Permet de gérer l'alignement des éléments sur l'axe horizontal au sein d'une grille
- La déclaration s'applique systématiquement à tous les éléments de la grille
- Valeurs possibles
  - start : Aligne sur le début de la cellule
  - end : Aligne sur la fin de la cellule
  - center : Aligne le contenu pour qu'il soit centré dans la cellule
  - stretch : Etire le contenu pour qu'il remplisse la cellule (valeur par défaut)



# CSS - Grid Layout - Disposition - *align-items*

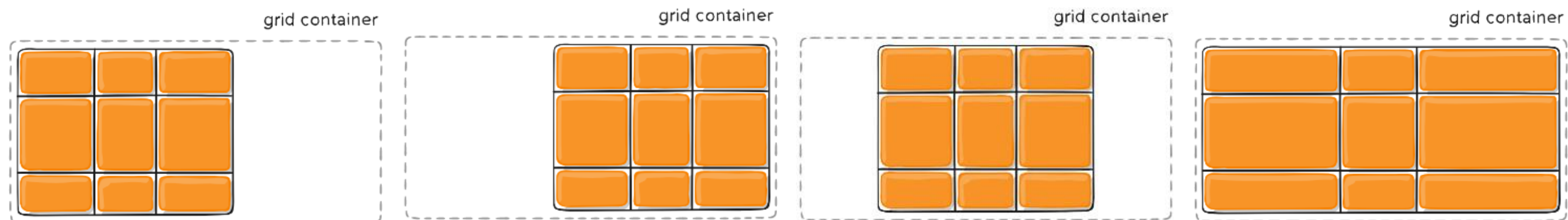
- Permet de gérer l'alignement des éléments sur l'axe vertical au sein d'une grille
- La déclaration s'applique systématiquement à tous les éléments de la grille
- Valeurs possibles
  - start : Aligne sur le début de la cellule
  - end : Aligne sur la fin de la cellule
  - center : Aligne le contenu pour qu'il soit centré dans la cellule
  - stretch : Etire le contenu pour qu'il remplisse la cellule (valeur par défaut)



# CSS - Grid Layout - Disposition - *justify-content*

- Dans les cas où la taille des éléments est inférieure à la taille du conteneur
- Définit l'alignement des éléments de la grille dans le conteneur sur l'axe horizontal

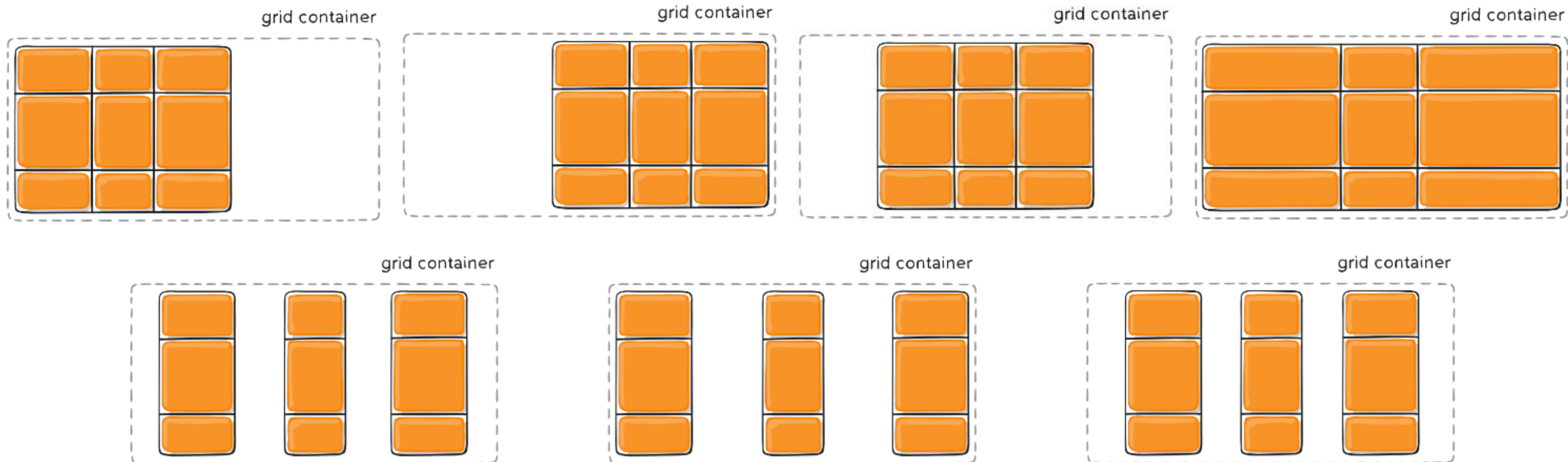
```
.container {  
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;  
}
```





# CSS - Grid Layout - Disposition - *justify-content*

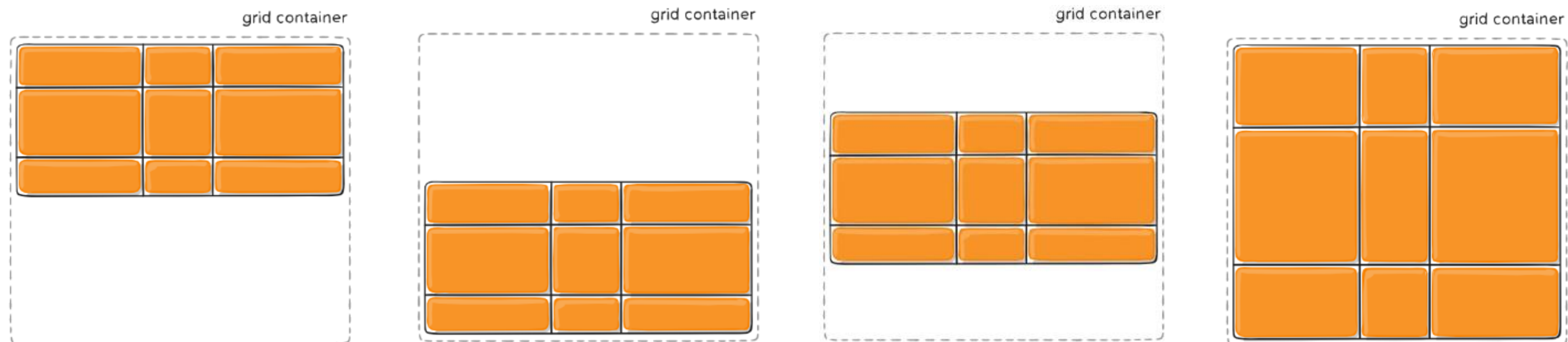
```
.container {  
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;  
}
```



# CSS - Grid Layout - Disposition - *align-content*

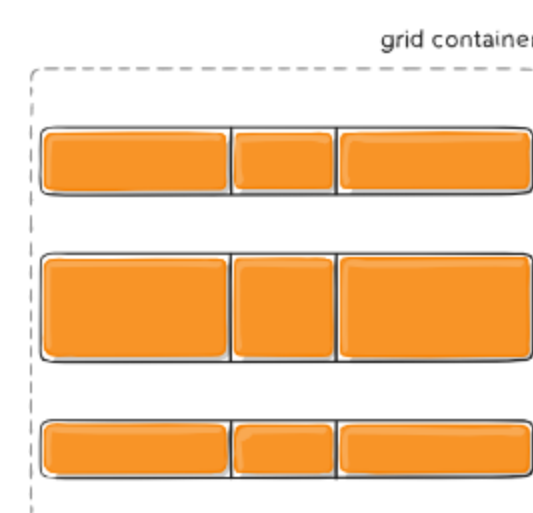
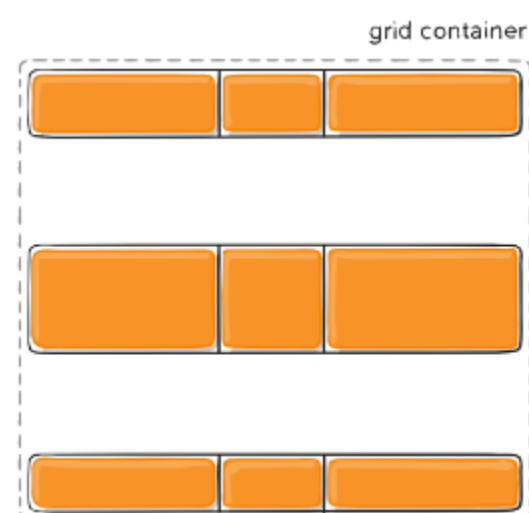
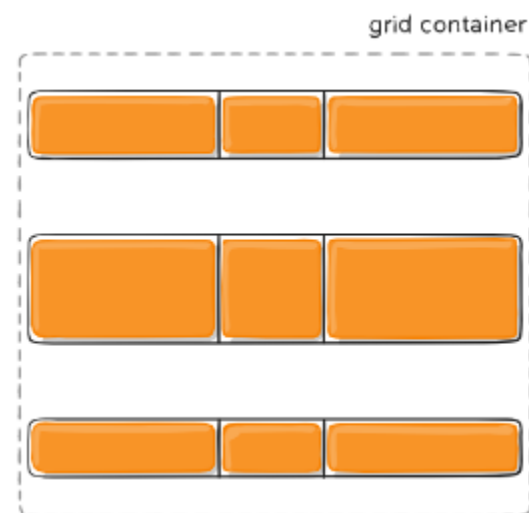
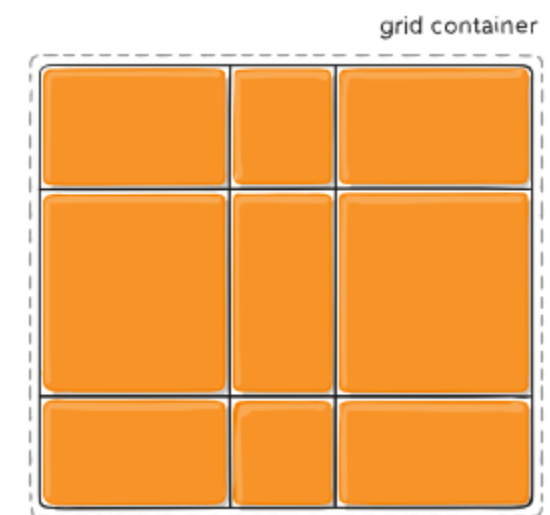
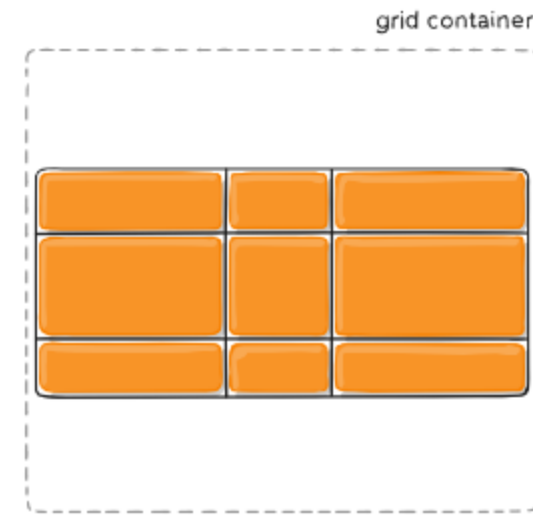
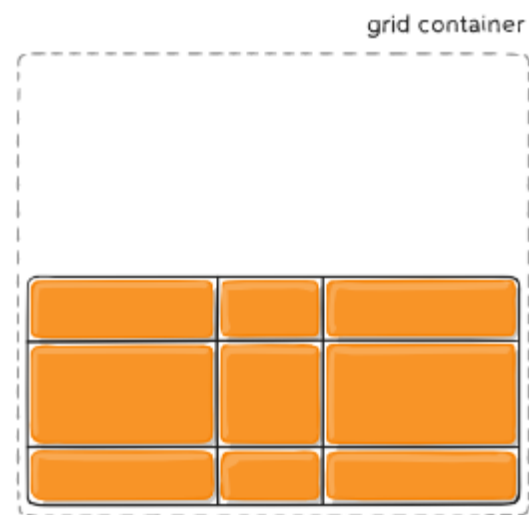
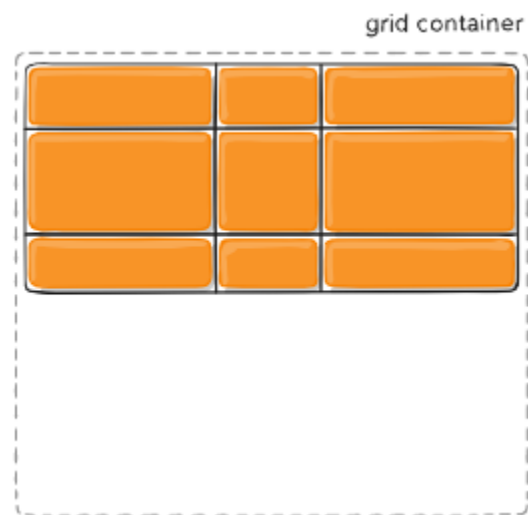
- Dans les cas où la taille des éléments est inférieure à la taille du conteneur
- Définit l'alignement des éléments de la grille dans le conteneur sur l'axe vertical

```
.container {
  align-content: start | end | center | stretch | space-around | space-between | space-evenly;
}
```



# CSS - Grid Layout - Disposition - *align-content*

```
.container {  
  align-content: start | end | center | stretch | space-around | space-between | space-evenly;  
}
```



# CSS - Grid Layout - Disposition

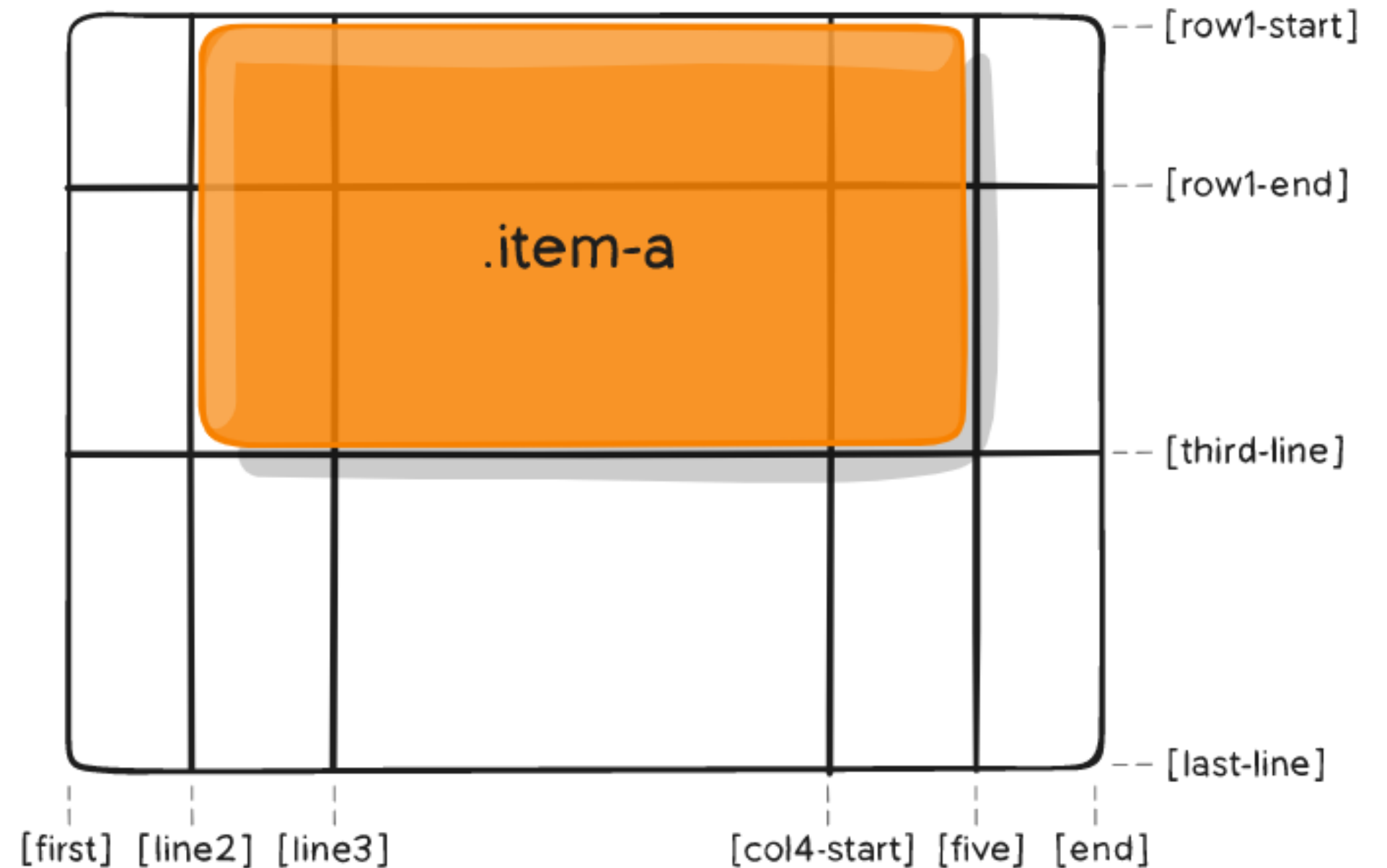
- « grid-column-start », « grid-column-end », « grid-row-start » et « grid-row-end » vous permettent de déterminer la position des éléments au sein d'une grille

```
.item {  
  grid-column-start: <number> | <name> | span <number> | span <name> | auto;  
  grid-column-end: <number> | <name> | span <number> | span <name> | auto;  
  grid-row-start: <number> | <name> | span <number> | span <name> | auto;  
  grid-row-end: <number> | <name> | span <number> | span <name> | auto;  
}
```

- number : le numéro de la ligne
- name : le nom de la ligne si les lignes ont été nommées
- span : « jusqu'à » la ligne désignée

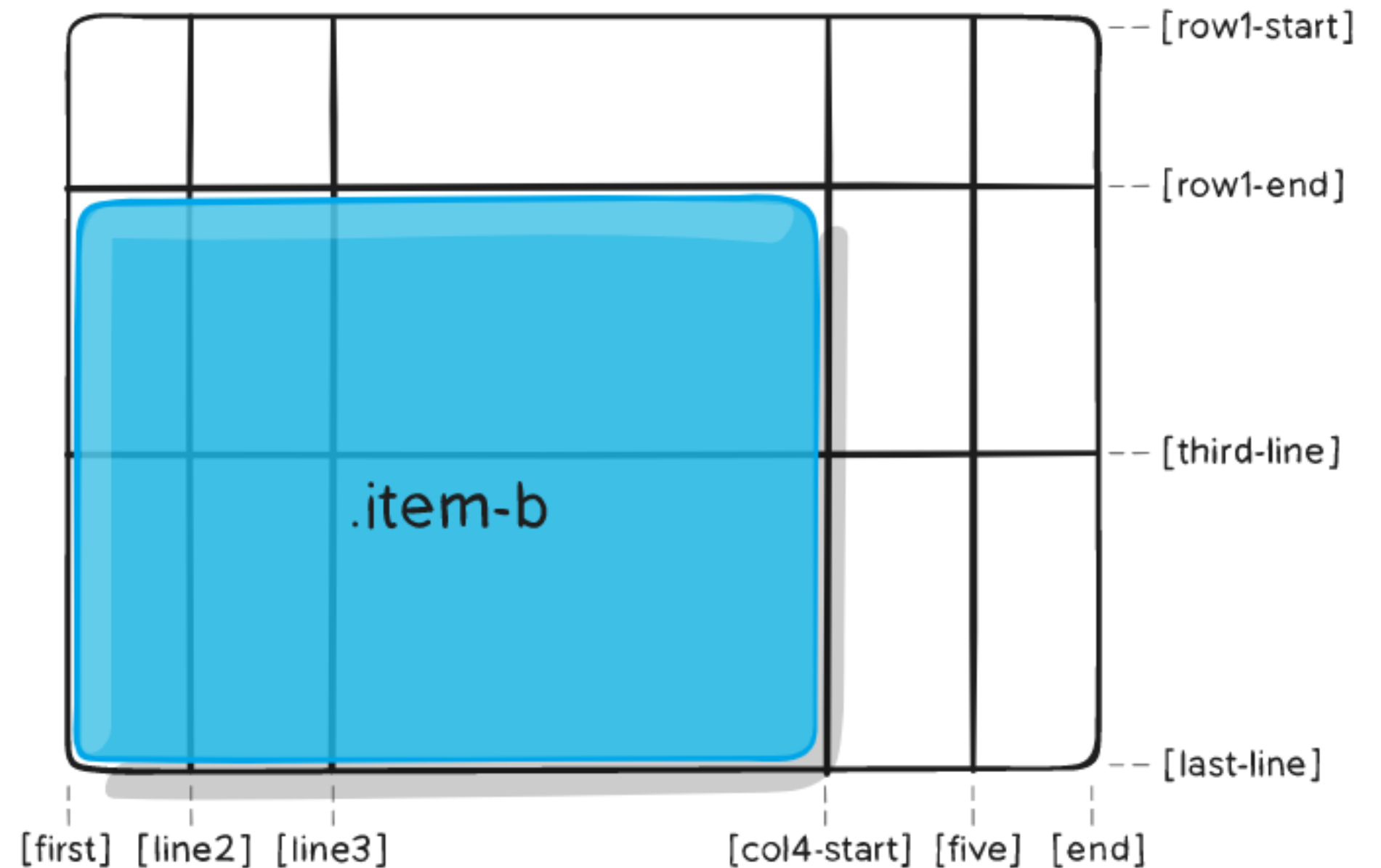
# CSS - Grid Layout - Disposition - Exemple

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```



# CSS - Grid Layout - Disposition - Exemple bis

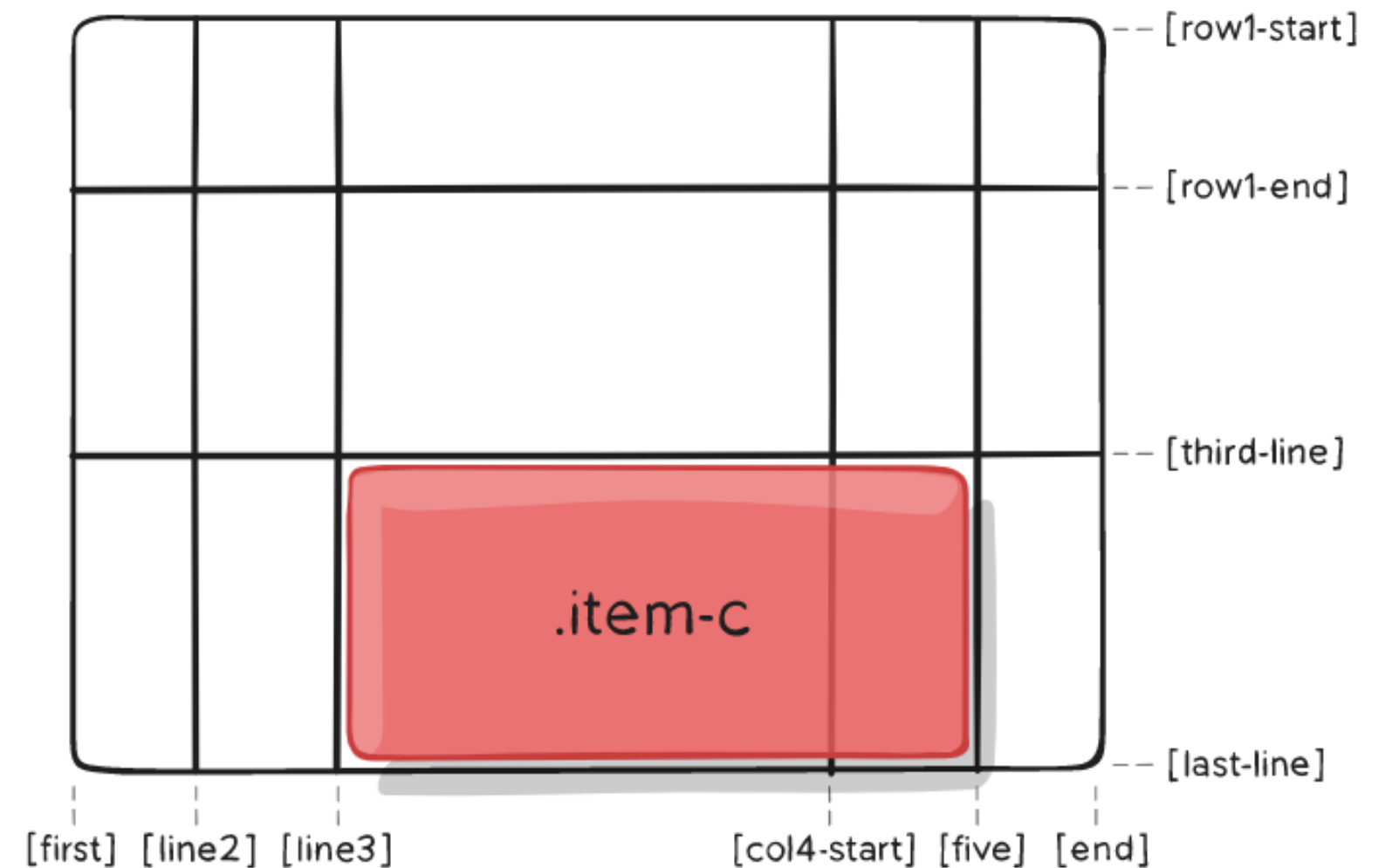
```
.item-b {  
  grid-column-start: 1;  
  grid-column-end: span col4-start;  
  grid-row-start: 2;  
  grid-row-end: span 2;  
}
```



# CSS - Grid Layout - Disposition - Raccourcis

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

```
.item-c {
  grid-column: 3 / span 2;
  grid-row: third-line / 4;
}
```



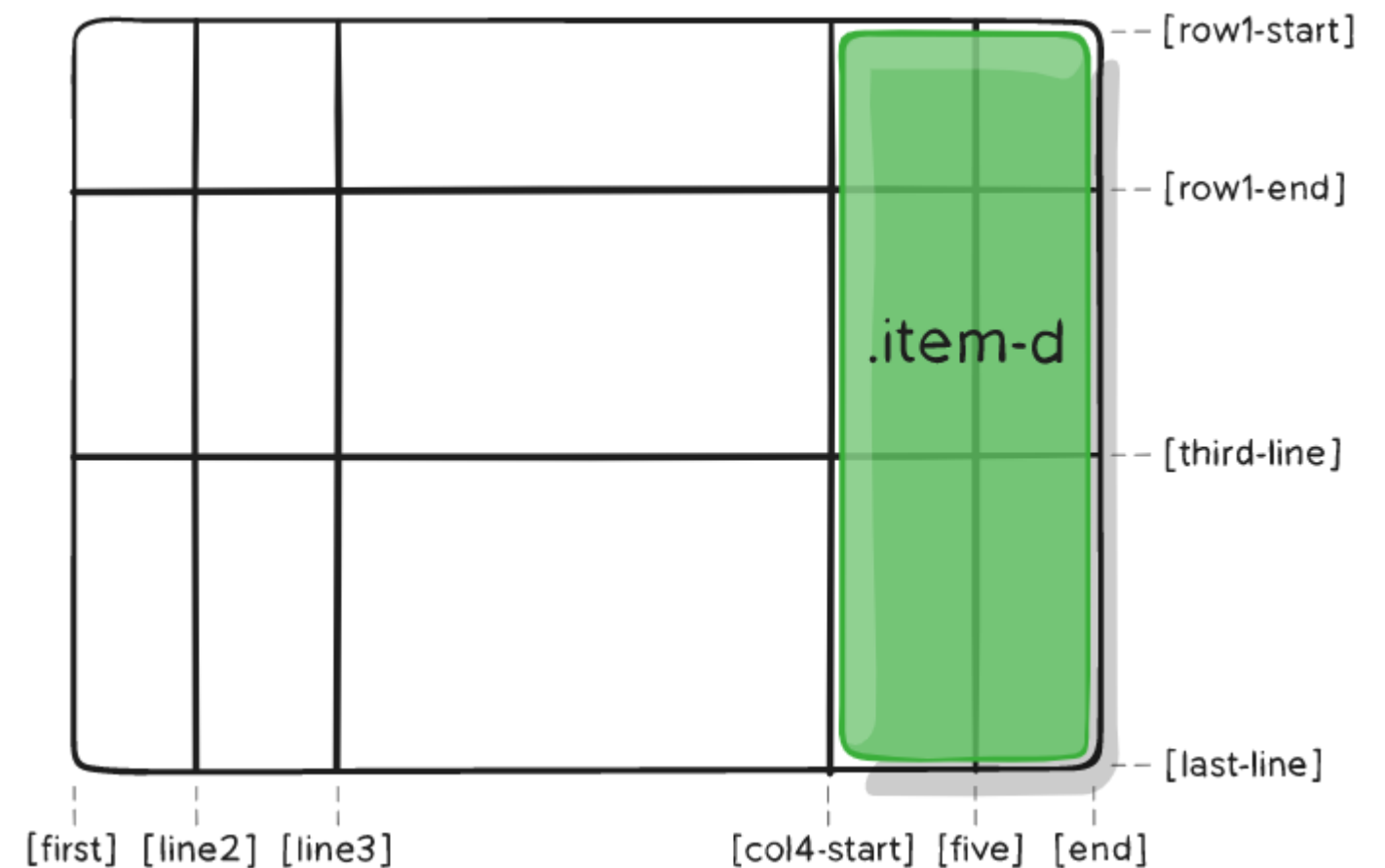


# CSS - Grid Layout - Disposition - *grid-area*

- Permet soit :
  - De nommer une aire pour pouvoir l'utiliser dans une définition de « grid-template-areas »
  - De définir une zone en lui fournissant les coordonnées (raccourcis)

```
.item {
  grid-area: < name > | < row-start > / < column-start > / < row-end > / < column-end >;
}
```

```
.item-d {
  grid-area: 1 / col4-start / last-line / 6;
}
```





# CSS - Grid Layout - Disposition

- Options de placements au sein d'une cellule
- Utile si le contenu est d'une taille inférieure à la taille de la cellule
- « justify-self » : aligne le contenu sur l'axe horizontal

```
.item {  
    justify-self: start | end | center | stretch;  
}
```

- « align-self » : aligne le contenu sur l'axe vertical

```
.item {  
    align-self: start | end | center | stretch;  
}
```

# Projet

Web



# Objectifs des 3 livrables

---

- Construire **deux** pages **statiques**
  - Page de **login** (1)
  - Page de **listing d'annonces** (2)
- **Proposer une version Desktop** de la page de listing d'annonces (3)
  - En utilisant les principes du responsive design
- Vous **devrez** utiliser
  - Les Grid Layout pour la structure principale
  - Les Flexible Layout pour le reste

# Login



Sign in



Welcome !

Please sign in to access content !

Username

Password

Sign in


All rights reserved Gregory Galli

# Liste d'annonces

Ad List

ALL ADS

MY ADS




Title

Description

1 hour ago

\$99,00



Title

Description

1 hour ago

\$99,00

All rights reserved Gregory Galli

# Références

---

- CSS3 badge  
Par Rudloff —  
[https://commons.wikimedia.org/wiki/File:CSS3\\_and\\_HTML5\\_badges.svg](https://commons.wikimedia.org/wiki/File:CSS3_and_HTML5_badges.svg), CC BY 3.0,  
<https://commons.wikimedia.org/w/index.php?curid=49121103>
- Avancée de la spécification CSS3  
Par Mercury999 — Travail personnel, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=36352662>
- Certaines sources et illustrations  
<https://css-tricks.com/>