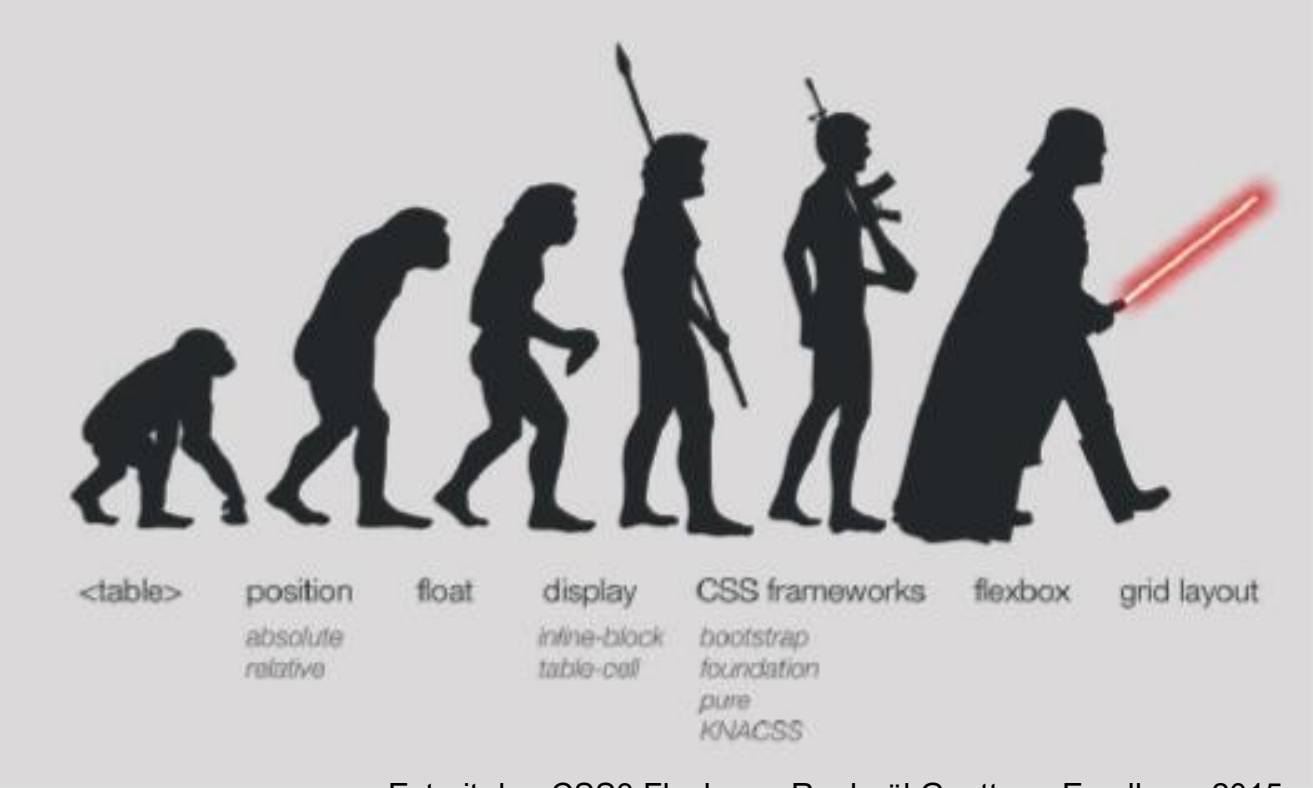
CSS3 – Grid Layout

- Flexbox ou Grid Layout ou Bootstrap ?
- Concepts de Grid Layout
- Terminologie
  - Container, Item
  - Line, Track, Cell, Area
- Les propriétés de grille
- Unité fr et fonction repeat()

Evolution de CSS



Extrait de : CSS3 Flexbox – Raphaël Goetter – Eyrolles – 2015

Flexbox et Grid Layout,
vous allez enfin aimer CSS !

Grid Layout ou Flexbox ?

- **Flexbox (2015)**
  pour gérer les composants, les contenus et les éléments internes dans la page
- **Grid Layout (2017)**
  pour la construction globale des gabarits, les grilles de mise en forme et leurs gouttières
  c-à-d concevoir des pages web +/- complexes de manière simple, intuitive et sans code inutile.

Flexbox et Grid Layout sont complémentaires

Différences entre Flexbox et Grid

- Différence fondamentale entre Grid Layout et Flexbox : la **gestion des axes** horizontaux et verticaux de façon simultanée.
- **Flexbox** est optimal dans **une seule direction**, tandis que **Grid** gère parfaitement les **rangées et colonnes** sans artifice ni conteneurs intermédiaires.
- Dans leur nature même : **Grid** conçu pour une prise en compte globale de la page, tandis que **Flexbox** est prévu pour agencer les composants internes et gérer leur fluidité.

Pourquoi Grid Layout ?

- C’est un framework de **positionnement** à lui tout seul, conçu pour **agencer les pages web**.
- Ni **position**, ni **float**, ni **inline-block**, ni même **Flexbox** n’ont été élaborés pour créer des designs de page. On les utilise en les détournant de leur objectif initial.
- **Parce que** Grid Layout est fait pour ça, il est plus rapide à écrire, il est plus performant et il est plus facile à maintenir

Grid Layout ou Bootstrap ?

- **Bootstrap** : framework créé en mélangeant diverses techniques CSS et JavaScript
- **Grid Layout** : spécification officielle du W3C.
- Bootstrap est un outil polyvalent, intégrant une convention de nommage, un fichier CSS «reset», des composants JavaScript et *un modèle de positionnement par grille*, d'où la confrontation

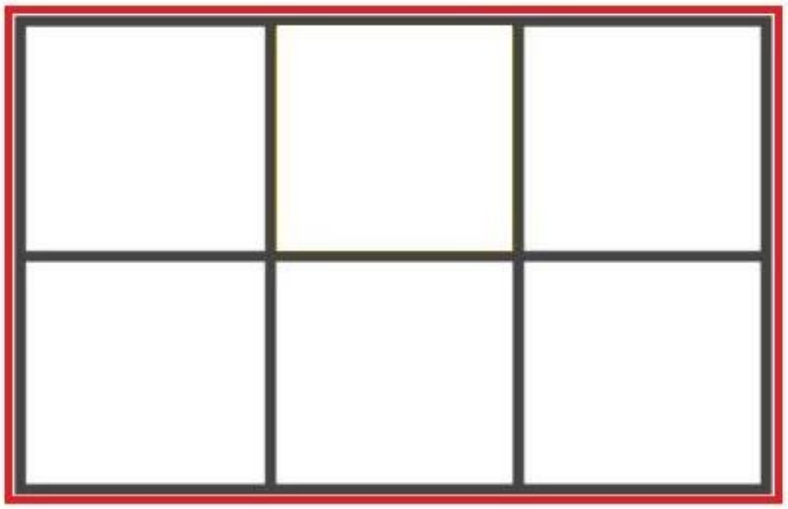
  https://css-tricks.com/css-grid-replace-flexbox/

  https://hackernoon.com/how-css-grid-beats-bootstrap-85d5881cf163

Concepts de Grid Layout

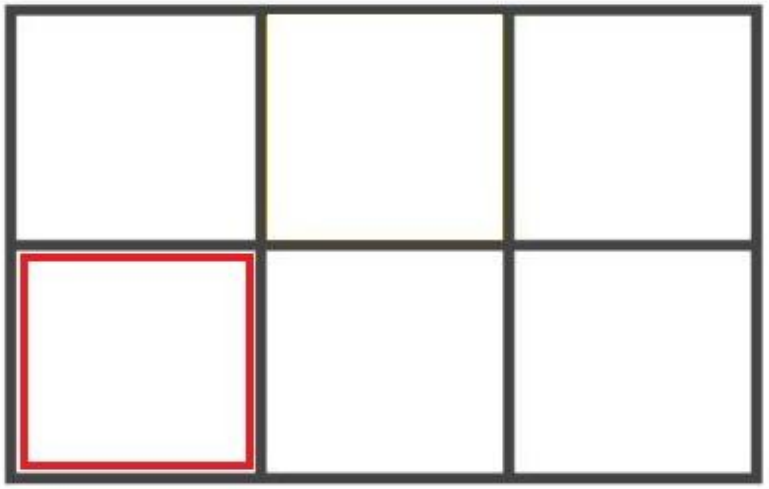
- Eléments physiques dans une grille (présents dans le code HTML)
  - **Grid Container** : le conteneur
  - **Grid Item** : les éléments enfants
- Eléments logiques dans une grille (pour la construction virtuelle des rangées et colonnes)
  - **Grid Line** : les lignes
  - **Grid Track** : les pistes
  - **Grid Cell** : les cellules
  - **Grid Area** : les zones

Les éléments physiques



GRID - CONTAINER

GRID-ITEM



Grid Container

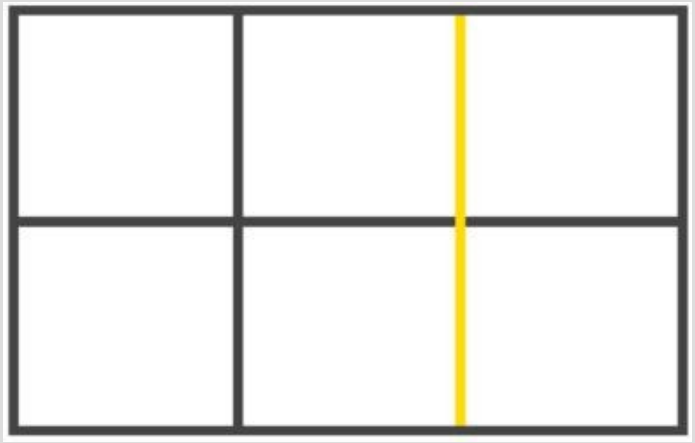
- 2 types d'éléments physiques sont nécessaires pour une grille : le **parent** et son/ses **enfant**(s)
- Ce schéma est nommé « **contexte de grille** »
- Attention, les **petits-enfants et le reste** de la descendance, ne sont **pas concernés** par ce contexte, ni par les propriétés associées
- Un **Grid Container** est l'élément **parent**, le **conteneur** de la grille. On lui donne la propriété

display:grid *ou* ***display:inline-grid***

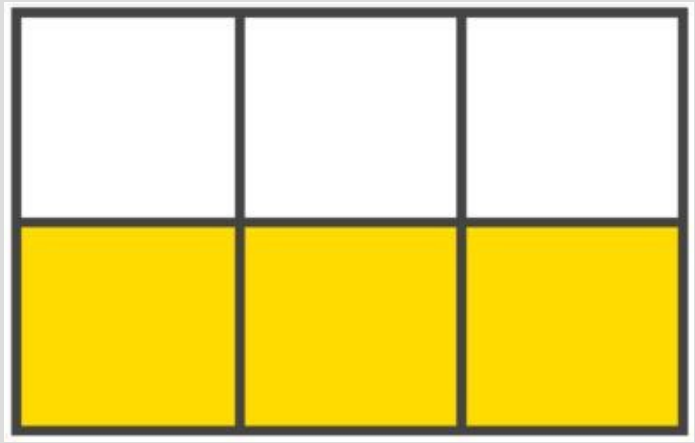
Grid Item

- Un **Grid Item** est le(s) élément(s) **enfant(s) direct(s)** d'un Grid Container
- Le terme « **direct** » est important : les petits-enfants et autres descendants du Grid Container **ne participent pas** au modèle de grille et **conservent leur modèle de positionnement** naturel (block, inline, etc.)
- Pas besoin d'appliquer une propriété CSS pour **transformer un élément en Grid Item**. Il suffit que le parent direct crée un contexte de grille.

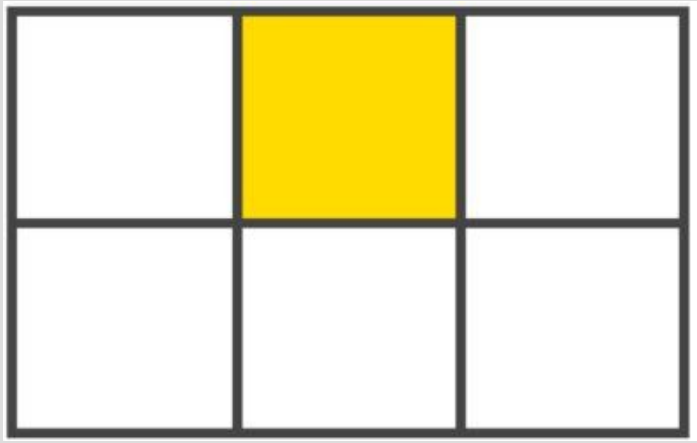
Les éléments logiques



GRID-LINE



GRID-TRACK



GRID-CELL

GRID-AREA

Grid Line

- **D’autres éléments** participent au contexte de grille **sans être présents physiquement** dans le code : les éléments logiques de la grille
- Une **Grid Line** (« ligne ») est une **ligne virtuelle** horizontale ou verticale **divisant la grille**
- Les propriétés de positionnement font référence aux « Grid Line » pour **placer des éléments au sein d’une grille**

Grid Track

- Une **Grid Track** (« piste »), c’est **l’espace** représenté **entre deux lignes** (Grid Line)
- Grid Track **horizontale** représente une **rangée**
- Grid Track **verticale** représente une **colonne**
- Les propriétés CSS **grid-template-columns** et **grid-template-rows**, servent à créer des pistes

Grid Cell

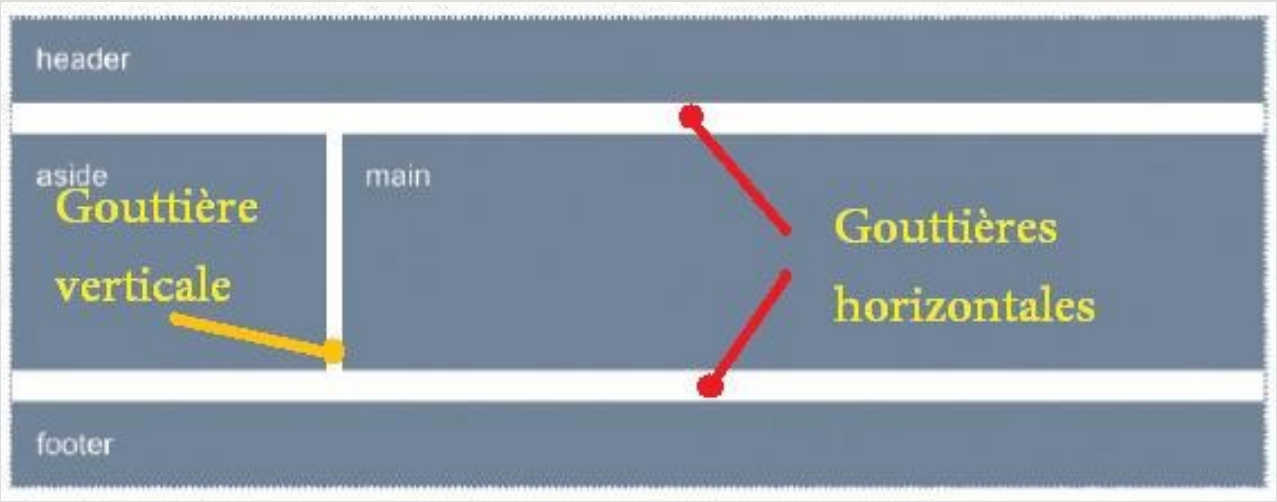
- Une **Grid Cell** (« cellule »), est une **intersection entre deux pistes**, c’est-à-dire entre une colonne et une rangée.
- C’est la **plus petite entité** de positionnement dans une grille

Grid Area

- Une **Grid Area** (« zone ») est **l'emplacement** dans lequel se positionne **un Grid Item**.
- Il est **constitué d'une ou plusieurs Grid Cell** et ne peut avoir qu'une **forme rectangulaire**

Gouttière

- Une **gouttière** est un **espace homogène entre chaque piste** verticale ou horizontale, ajouté pour aérer les compositions, (comme on fait habituellement dans les publications papier)
- Elément très employé en webdesign moderne



row-gap: 20px;

column-gap: 10px;

gap : 20px 10px;

Propriétés de grille

- Pour définir une grille, on a 3 propriétés :
  - **display:grid;** permet de définir un **contexte de grille** pour un élément qui sera de type **block**
  - **display:inline-grid;** permet de définir un **contexte de grille** pour un élément qui sera de type **inline** (et 2 conteneurs seront côte à côte)
  - **display:subgrid;** permet de transformer un grid item en grid container (la grille enfant a les mêmes caractéristiques que la grille parente) Ce mode est en cours de développement

Exemple 1 – HTML

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>CSS Grid Layout</title>
    <link type="text/css" rel="stylesheet" href="grid.css" />
</head>
<body>
    <div id="global">
        <header>Entête</header>
        <nav>Menu</nav>
        <main>Contenu</main>
        <aside>Barre latérale</aside>
        <footer>Pied de page</footer>
    </div>
</body>
</html>
```

Exemple 1 – CSS

```css
#global {
    display:grid;
}

header {
    background-color: aqua;
}

nav {
    background-color: aquamarine;
}

main {
    background-color: coral;
}

aside {
    background-color: lightgray;
}

footer {
    background-color: black;
    color:antiquewhite;
}
```

Colonnes et rangées

- Avec ***display:grid*** les enfants **directs** deviennent des **éléments de grille** affichés sur une colonne
- Pour construire la grille, on définit les colonnes et les rangées (les pistes) :
  - On peut choisir le nombre et la dimension des **colonnes** avec **grid-template-columns**
  - On peut choisir le nombre et la dimension des **rangées** avec **grid-template-rows**
  - Comme **unité**, on peut utiliser les unités absolues ou relatives (px, %, vh, em,…) ainsi que auto et une nouvelle unité (**fr** pour fraction)

Exemple 2

```
#global {
    display:grid;
    grid-template-columns: 100px 60% auto;
    grid-template-rows: 50px auto;
}
```

- Par exemple, cette grille est composée de :
  - **3 colonnes** :
    la 1ère de 100 pixels de large,
    la 2ème de 60% de la largeur de la fenêtre et
    la 3ème occupe automatiquement ce qui reste
  - **2 lignes** :
    la 1ère de 50 pixels de haut et
    la 2ème occupe automatiquement ce qui reste

Unité fr

- Les **pistes** (ligne, colonne) peuvent être définies à l'aide de **n'importe quelle unité** de mesure.
- Les grilles proposent une nouvelle unité de mesure pour créer des **pistes flexibles**. Cette unité, **fr**, représente une **fraction** de l'espace disponible dans le conteneur de la grille
- Exemple : **grid-template-columns : 1fr 3fr 1fr;** permet de diviser la grille en 3 colonnes, la 1ère occupe une fraction, la 2ème occupe 3 fractions et la 3ème occupe 1 fraction

Fonction repeat( )

- Pour éviter de répéter tout ou une partie des pistes définies, on peut utiliser la fonction **repeat()**
- Par exemple :
  - *grid-template-columns : 1fr 1fr 1fr 1fr;* remplacé par **grid-template-columns : repeat(4, 1fr);**
  - *grid-template-columns : 50px 1fr 1fr 1fr 50px;* ou **grid-template-columns : 50px repeat(3, 1fr) 50px;**
  - *grid-template-columns : 1fr 2fr 1fr 2fr 1fr 2fr;* ou **grid-template-columns : repeat(3, 1fr 2fr);**