css grid layout

- mode de placement complémentaire à `flex` destiné à du placement en 2 dimensions,
- Des propriétés sur le conteneur : fixent les contraintes organisant le flux des items directement inclus
- Des propriétés sur les items : comportement particulier de chaque item
- Quelques références :
  - Introduction sur MDN
  - le guide CSS Tricks
  - alsacréations
  - des exemples, et d'autres exemples

css grid layout

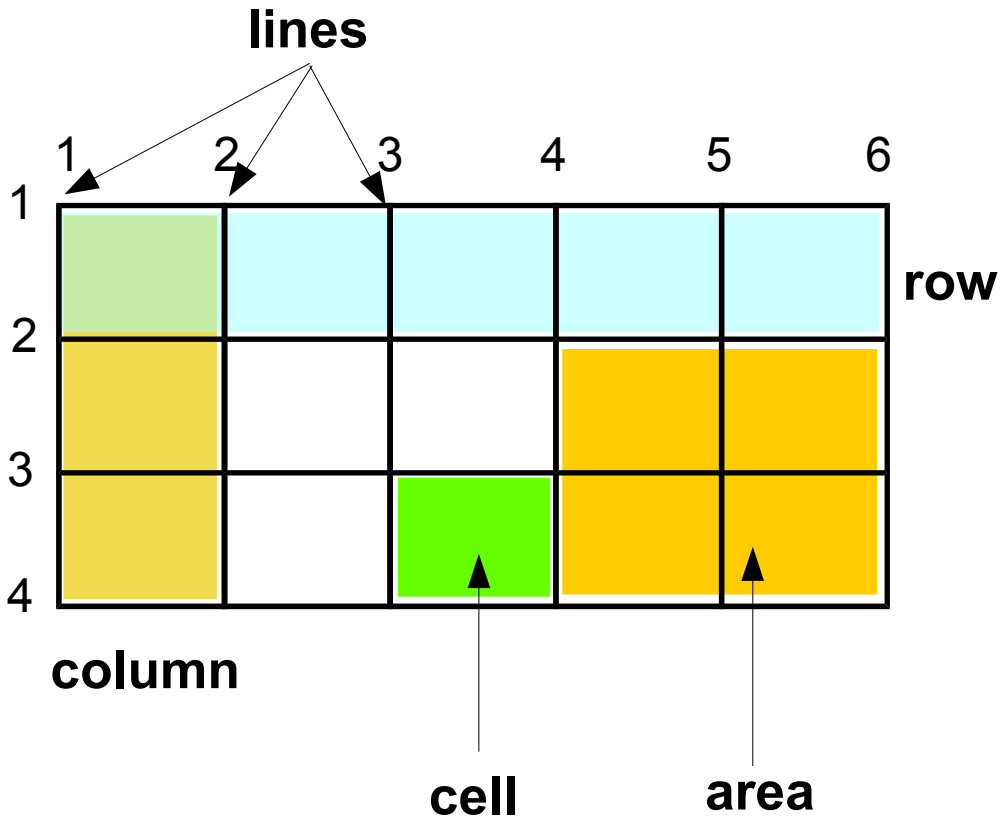
- Placement basé sur la définition d'1 grille au niveau du conteneur :
  - nbre et largeur des colonnes
  - nbre et hauteur des rangées
- les items sont placés sur cette grille :
  - **explicitement** : on précise le rectangle occupé par l'item sur la grille
  - **implicitement** : on précise uniquement la taille de l'item, et il est placé automatiquement
- on peut paramétrer le placement implicite :
  - en ligne, en colonne, occupation maximale
- exemple

```
.grid_container {
  display: grid;
  grid-template-columns : 10% 50px 15rem 20rem; // 4 colonnes
  grid-template-rows: 5rem 4rem;                // 2 rangées
  grid-auto-flow: row;                  // placemement implicite
                                        // en ligne
  grid-gap: .2em;  //gouttières
}



.cell_one {                             // placement explicite
  grid-column : 3 / 5;                  // sur la grille
  grid-row: 1/3;
}

.cell_two                       {       // placement implicite
  grid-column: span 2;                  // largeur 2 colonnes
}

.cell_three {                           // placement implicite
  background-color: green;

}
```

vocabulaire

- **container** : élément sur lequel on applique `display: grid,` parent direct des éléments à placer
- **item** : les descendants directs du container, à placer sur la grille
- **track** : zone comprise

  entre 2 lines successives, c'est donc 1 **row** ou 1 **column**
  - **row :** track horizontale
  - **column** : track verticale
- les **lines** structurent la grille et sont nommées, par défaut de 1 à n
- **cell** : espace entre 2 lines successives horizontales et et 2 lines successives verticales
- **area :** zone délimitée par 4 lines – une area est donc formée de cellules contiguës formant 1 rectangle
- **gap** : espace séparant les rows et les columns



la définition des grilles – propriétés du conteneur

- une grille est déclarée en spécifiant la structure des rows et columns
- cette structure peut
  - être **explicite** : le nombre et la largeur de chaque rangée/colonne est explicitement spécifié
  - prévoir la structure de tracks **implicites,** ajoutées automatiquement en fonction du nombre d'items à placer ; on peut déclarer la largeur des rangées/colonnes implicites
- on utilise souvent une structure mixte explicite/implicite
  - la partie explicite définit la structure générale de la grille,
  - la partie implicite gère le placement d'items supplémentaire ou placés à l'extérieur de la grille explicite
- les **unités et valeurs** pour dimensionner les tracks :
  - unités classiques : `rem, %, vw, vh,  px, ...`
  - **fractions** de l'espace disponible : `1fr, 2fr` …
  - `minmax(min, max)` : au moins `min`, au plus `max` → `minmax(100px, 20%)`
  - auto, min-content, max-content **:** valeurs liées au contenu

la définition des grilles – propriétés du conteneur

- **définition explicite :**
  - `grid-template-columns` : structure des *rangées*
  - `grid-template-rows` : structure des *colonnes*
  - raccourci : `grid-template:`
- **tracks implicites :**
  - `grid-auto-columns` : taille d'une *colonne* ajoutée automatiquement
  - `grid-auto-rows` : taille d'une *rangée* ajoutée automatiquement

```
.grid_container {
  display: grid;
  grid-template-columns : 20% 8rem 2fr 1fr;
  grid-template-rows: 5rem 5rem 5rem;
}
```

- 4 colonnes et 3 rangées
- les 2 dernières colonnes occupent l'espace laissé libre par les 2 premières
- la 3ème fait le double de la 1ère
- voir sur codepen

```
  grid-template-columns: 20% 8rem 2fr 1fr;
  grid-auto-rows: 5rem;
```

- 4 colonnes et des rangées de 5rem
- autant de rangées que nécessaires, créées au fur et à mesure de l'ajout d'items
- voir sur codepen

```
  grid-template-columns :repeat(3, 20%) repeat(3, 1fr);
  grid-auto-rows: 5rem ;
```

- 6 colonnes : 3 de 20 % + 3 occupant l'espace disponible
- `repeat(n, taille)` : répéter n fois un motif
- voir sur codepen

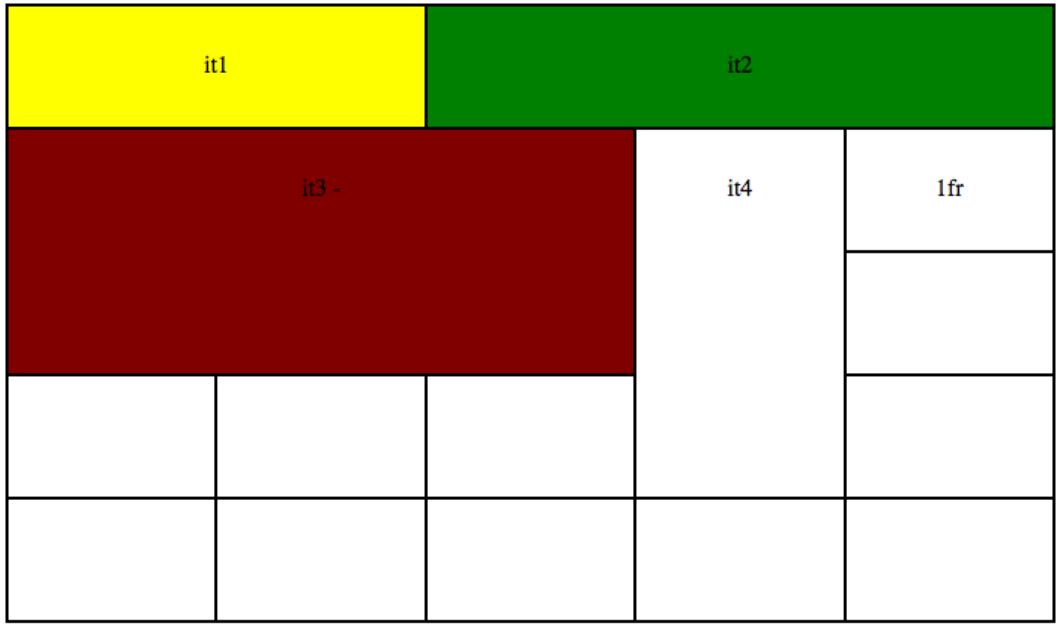
```
  grid-template-columns :minmax(5rem, 20%) repeat(5, 1fr);
  grid-auto-rows: 5rem ;
```

- 1 colonne au moins 5rem, au plus 20 %, suivie de 5 colonnes identiques sur l'espace libre
- voir sur codepen

positionnement des éléments sur la grille – propriétés des items

- **placement explicite** : on indique les lignes début et fin:
  - dans le sens horizontal : `grid-column-start, grid-column-end`
  - dans le sens vertical : `grid-row-start, grid-row-end`
  - raccourcis : `grid-column debut / fin ;`
- **les valeurs :**
  - nom de ligne : `grid-column-start: 1; grid-column-end : 3;`
  - nombre de lignes : `grid-column-end: span 3;`

```
.grid_container {
  display: grid;
  grid-template-columns :repeat(5, 1fr);
  grid-template-rows :  repeat(5, 5rem);
}
.it1 {
  grid-column-start: 1; grid-column-end: 3;
  background-color: yellow; }
.it2 {
  grid-colum-start: 3; grid-column-end: span 3;
  background-color: green }
.it3 {
  grid-column: 1 / span 3;
  grid-row-start : 2 ; grid-row-end: 4;
  background-color: maroon ; }
.it4 {
  grid-colum: 4 / span 1;
  grid-row: 2 / span 3; }
```



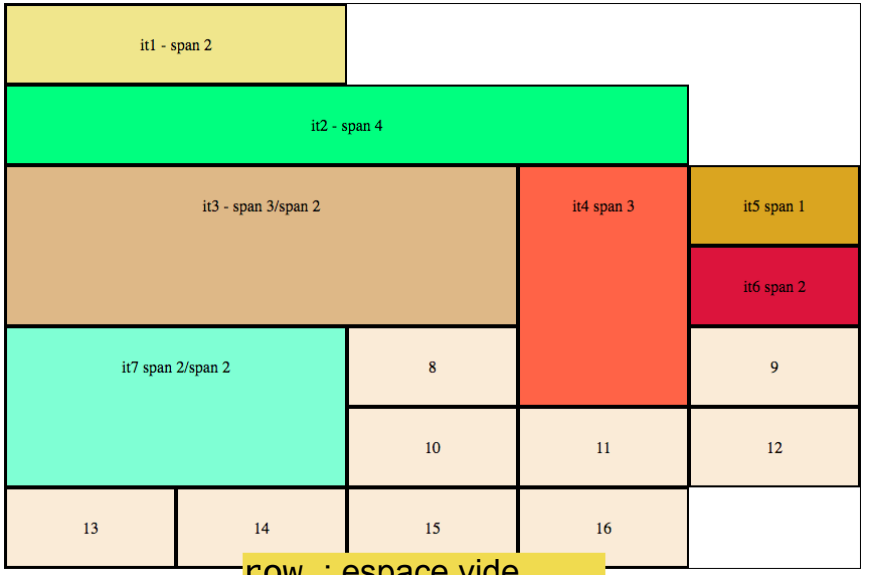
positionnement implicite des éléments

- les items peuvent être positionnés **implicitement** sur la grille :
- on précise uniquement leur **taille** si elle est différente de 1 en utilisant `grid-column-end:` et `grid-rox-end:`
- les items sont placés par un algorithme **dans l'ordre où ils apparaissent**,
- on peut donner des **préférences** à l'algorithme de placement en définissant la propriété `grid-auto-flow` sur le **conteneur :**
  - `grid-auto-flow: row` – remplissage des **rangées** dans l'ordre d'apparition des items avec ajout automatique de rangées si besoin
  - `grid-auto-flow: column` – idem, sur les **colonnes**
  - `grid-auto-flow: dense` – remplissage dense avec occupation au mieux de l'espace ; l'ordre d'apparition des items n'est pas forcément respecté
  - `grid-auto-flow: row dense` – remplissage dense des rangées
  - `grid-auto-flow: column dense` - remplissage dense des colonnes

- exemple : voir sur codepen

```
.grid_container {
  display: grid;
  grid-template:repeat(5, 5rem) /repeat(5, 1fr);
  grid-auto-rows: 5em;
  grid-auto-columns: 1fr;
  grid-auto-flow: row ;
                /* column ; */
                /* dense ;  */
}
```

row

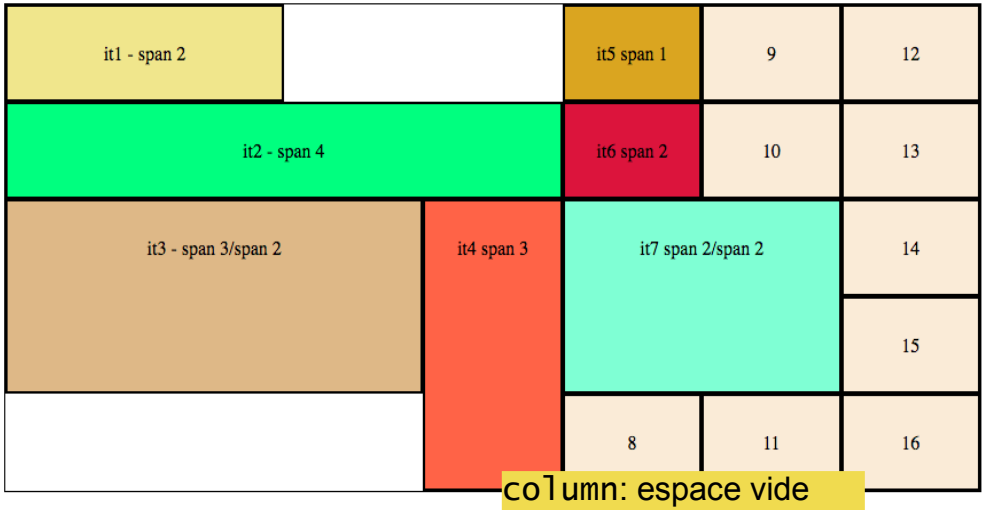


row : espace vide
+rangées automatiques

column

dense

dense: pas d'espace
vide +ordre non préservé



column: espace vide
+colonnes automatiques

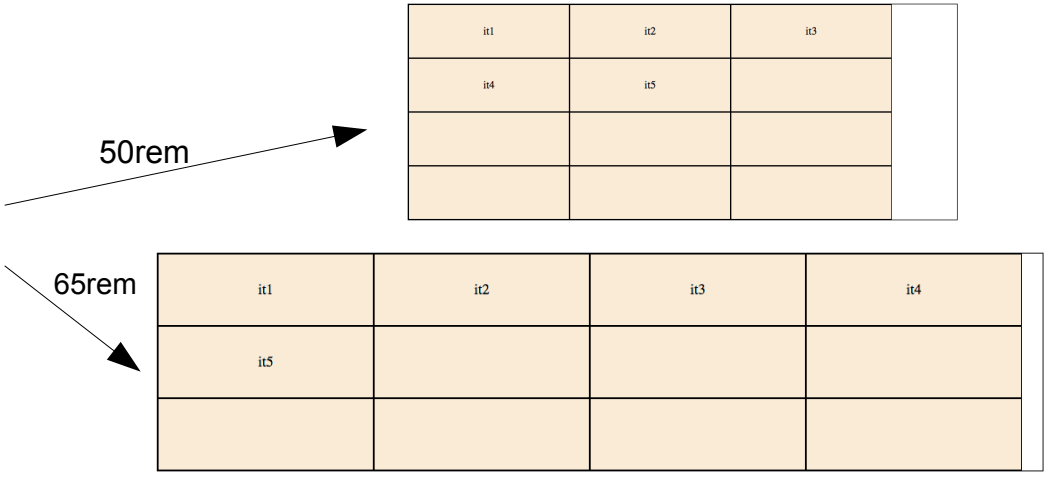
autres propriétés sur les conteneurs

- **gouttières** : il est possible d'espacer les cellules en définissant des gouttières
  - `grid-row-gap:` gouttières entre les rangées
  - `grid-column-gap:` gouttières entre les colonnes
  - `grid-gap:` raccourci
- **alignements** : par défaut, le contenu d'une cellule est étiré sur toute la cellule (stretch). On peut modifier ce comportement sur toute la grille :
  - `justify-items:` alignement le long de l'axe ligne
  - `align-items:` alignement le long de l'axe colonne
  - valeurs : `start | end | center | stretch`

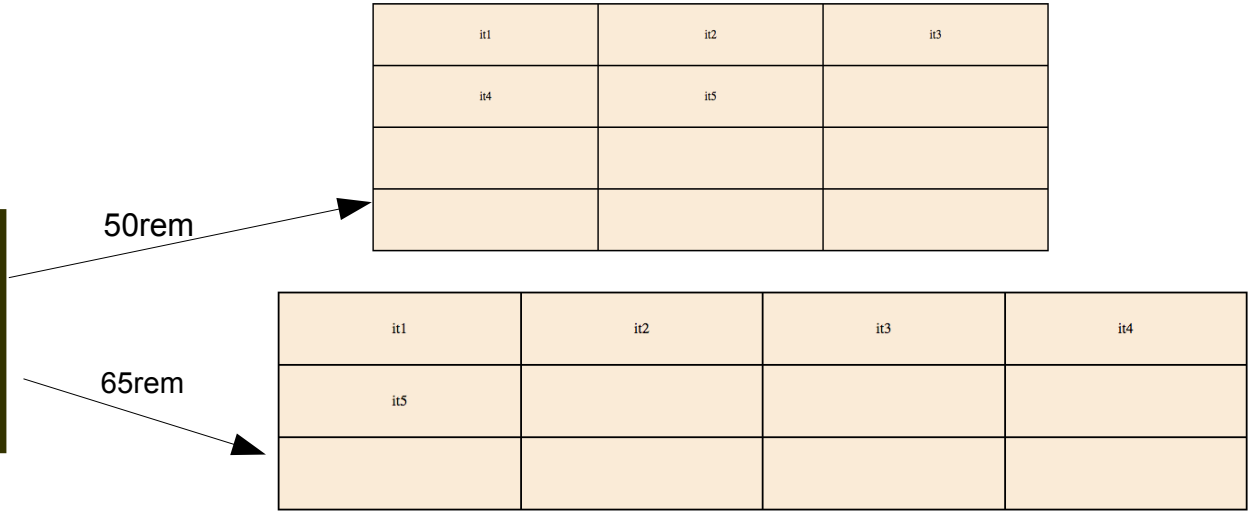
définition avancées de grilles

- **Répétitions automatiques** adaptée à la taille du conteneur dans un template :
  la fonction `repeat( n, motif )` accepte la valeur `auto-fit` qui permet de répéter le motif autant de fois que la place le permet.
  Elle permet donc de spécifier des grilles dont le nombre de colonnes s'adapte à la largeur du conteneur. (voir sur codepen)
  On peut combiner avec minmax() pour obtenir des grilles fluides et responsives. (voir sur codepen)

```
.grid_container {
  display: grid;
  grid-template-columns: repeat(auto-fit, 15rem);
  grid-auto-rows: 5rem;
}
```



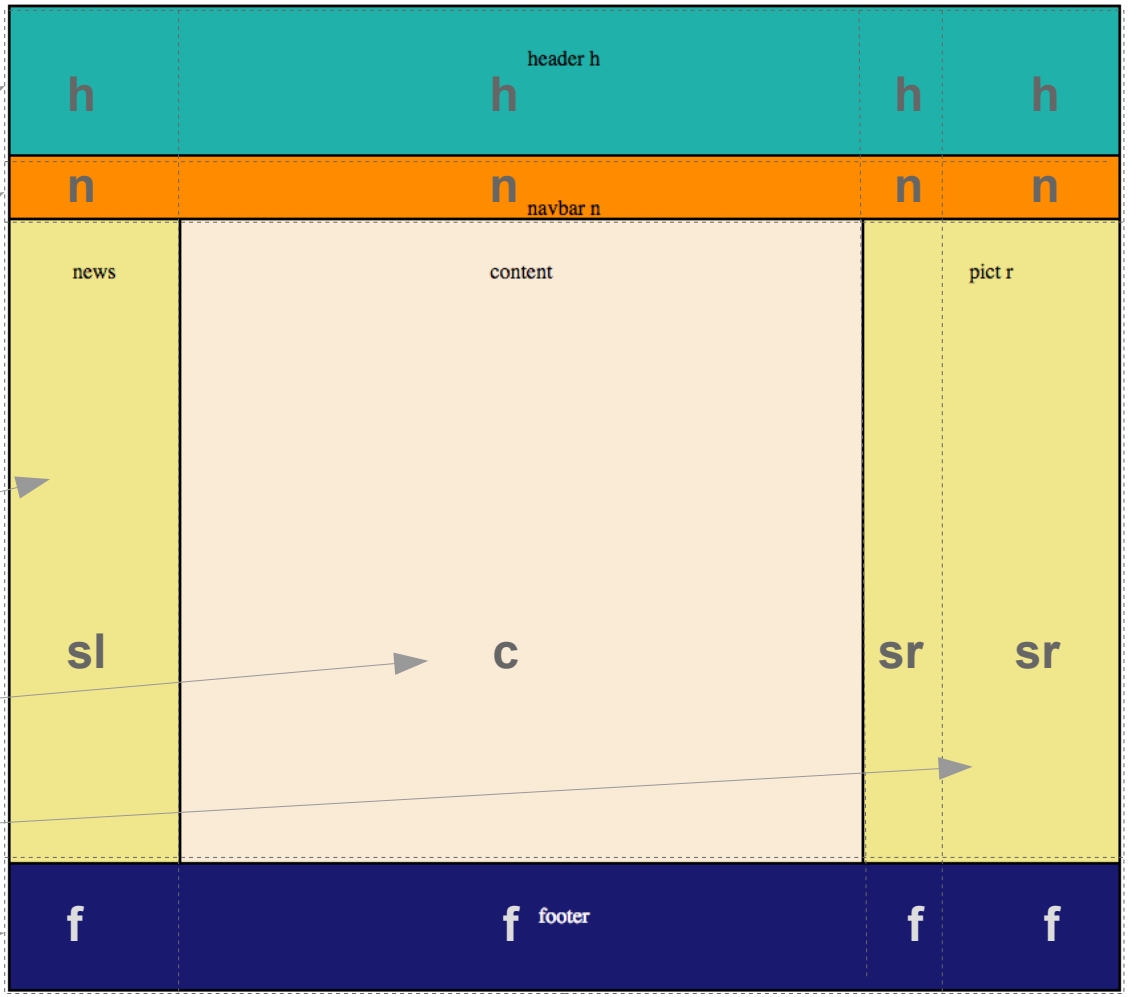

```
.grid_container {
  display: grid;
  grid-template-columns:
      repeat(auto-fit, minmax(15rem, 1fr));
  grid-auto-rows: 5rem;
}
```



définitions avancées de grilles

- **Zones nommées** : il est possible de nommer les différentes zones de la grille et d'utiliser ces noms pour placer explicitement des items
  - chaque cellule de la grille est nommée,
  - une zone est définie par plusieurs cellules portant le même nom,
  - une zone est formée de **plusieurs cellules contiguës** formant 1 **rectangle**
- propriété sur le conteneur :
  - **`grid-template-areas:`** permet de définir les noms des différentes zones de la grille
- propriété sur les items :
  - `grid-area:` permet de positionner explicitement l'item sur la zone indiquée ; l'item prend la taille de la zone

```css
.grid_container {
grid-template-columns : 1fr 4fr .5fr 1fr;
grid-template-rows: 15vh 3rem
                calc(85vh - 3rem)  6rem;
grid-template-areas :
            "h h h h"
            "n n n n"
            "sl c sr sr"
            "f f f f" ;
}

.header {
  grid-area: h;
}
.nav {
  grid-area: n;
}
.news {
  grid-area: sl;
}
.content {
  grid-area: c;
}
.pict {
  grid-area: sr;
}

.footer {
  grid-area: f;
}
```



voir sur codepen