

# Intégration Web

MMI 2 – TP#4 S3

Danielo **JEAN-LOUIS**  
Michele **LINARDI**

# Mise en page - Historique

- Au début, les développeurs utilisaient des tableaux (<table>) pour faire la mise en page
  - **Ne jamais, jamais utiliser ceci !**
  - A utiliser uniquement :
    - Pour afficher des données tabulaires
    - Faire des e-mails

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/float>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/table>
- <https://www.alsacreations.com/article/lire/1209-display-inline-block.html>
- <https://putaindecode.io/podcasts/s02e04-css-layout> – podcast en français

# Mise en page - Historique

- Ensuite arriva la propriété "float"
  - **A éviter. Crée plein d'effets de bord fastidieux à corriger**

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/float>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/table>
- <https://www.alsacreations.com/article/lire/1209-display-inline-block.html>
- <https://putaindecode.io/podcasts/s02e04-css-layout> – podcast en français

# Mise en page - Historique

- Puis la valeur "inline-block" pour la propriété "display"
  - Très limité dans l'usage
- Et enfin le CSS3 apporta **flexbox** (vu en S1)
  - Et **grid** qui est le sujet du jour

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/float>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/table>
- <https://www.alsacreations.com/article/lire/1209-display-inline-block.html>
- <https://putaindecode.io/podcasts/s02e04-css-layout> – podcast en français

# CSS Grid

- Nouveauté CSS 3
  - Nécessite un navigateur moderne
- Petit frère de flexbox
- Permet la mise en page en grille
  - Pallie aux limites de flexbox
- Divise l'espace en régions de tailles égales ou différentes

## Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout)

# CSS Grid

- Gestion du placement des éléments sur deux dimensions (ligne et colonne)
  - Flexbox ne gère qu'une dimension
- Géré par tous les navigateurs modernes
  - PC et terminaux mobiles
- Affichage en “bloc”
  - Occupe toute la largeur disponible

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout)

# CSS Grid

- Possibilité de faire une grille dans une autre
- Permet de faire des grilles inégales
  - Lignes et colonnes de tailles différentes
  - Ajoute du dynamisme à la mise en page du site

## Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout)



# CSS Grid

- Comme pour flexbox, c'est le parent qui définit ce type de mise en page grâce à la propriété CSS "display"

A code editor window with a dark background and a pink border. The title bar shows three colored circles (red, yellow, green) and a blue icon with a white 'E' followed by the text 'css-grid'. The code inside is:

```
.mon-selcteur {  
  display: grid;  
  // [ ... ]  
}
```

Notre sélecteur est de type grid maintenant

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout)

# CSS Grid – Propriétés (liste non exhaustive)

- grid-template-rows : définit les lignes
- grid-template-columns : définit les colonnes
- gap : espacement des gouttières
  - Fonctionne également avec flexbox
- align-items, justify-items : comme dans flexbox

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout#propri%C3%A9t%C3%A9s\\_css](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout#propri%C3%A9t%C3%A9s_css)

## grid-template-rows

- Définit les lignes de la grille
  - Nombre infini (comme pour les colonnes)
- Les enfants occupent naturellement les lignes (en fonction des colonnes)

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-template-rows>

# grid-template-rows



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-rows: 250px 400px;  
}
```

Ici on définit deux lignes, une de 250px et une autre de 400px

## grid-template-rows

- S'il y a plus d'éléments que de lignes définies, les éléments en plus auront une hauteur égale à leur contenu

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-template-rows>

**Essayons avec plus de lignes**

# grid-template-rows



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-rows: 250px 400px 400px 400px 400px 250px;  
}
```

Maintenant, nous avons six colonnes mais...

**...on se répète**



## Fonction repeat()

- Ne fonctionne qu'avec les propriétés “grid-template-columns” et “grid-template-rows”
- Permet de répéter une valeur de grille
- Accepte deux arguments :
  - Nombre de répétitions
  - Valeurs à répéter, séparée par un espace

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>

# Fonction repeat()



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-rows: 250px repeat(4, 400px) 250px;  
}
```

Même code qu'auparavant mais sans répétition de code

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>

## Fonction repeat()

- Permet également de gérer un nombre infini de répétition
  - auto-fit : Prend plus d'espace s'il y en a
  - auto-fill : Ne prend pas plus d'espace s'il y en a

La différence se constate plutôt avec les colonnes (columns), pas les lignes (rows)

Source(s) :

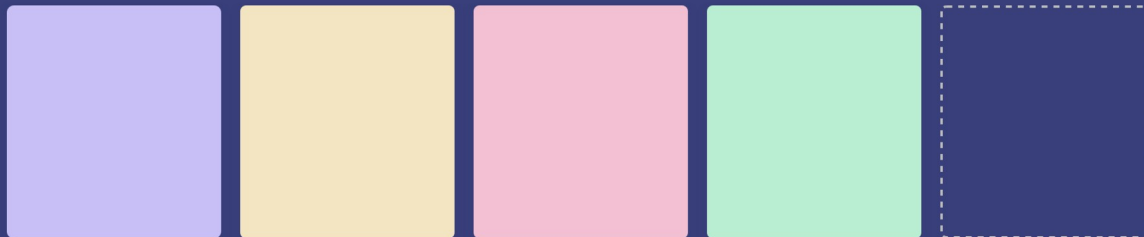
- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>
- <https://defensivecss.dev/tip/auto-fit-fill/>

# Fonction repeat()

```
repeat(auto-fill, minmax(200px, 1fr));
```

Don't auto expand the cols when more space is available

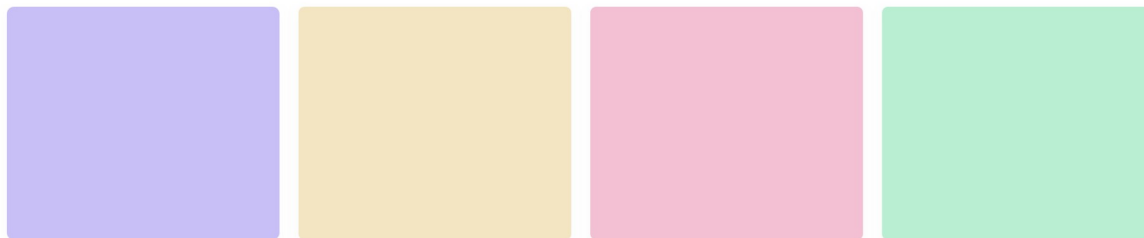
auto-fill



```
repeat(auto-fit, minmax(200px, 1fr));
```

Expand the cols when more space is available

auto-fit



La fonction minmax() indique les valeurs applicables, ici c'est au minimum 200px et au maximum 1fr – Crédits illustration defensivcss.dev

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>
- <https://defensivcss.dev/tip/auto-fit-fill/>

## Fonction repeat()

- auto-fit / auto-fill peuvent fonctionner sans minmax()
  - Dans ce cas le comportement habituel de ces deux valeurs ne sera pas pris en compte

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>

# Fonction repeat()



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, 120px);  
}
```

On définit un nombre infini de colonnes de 120px de largeur.  
Si ça dépasse, on retourne à la ligne.

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/repeat>

## grid-template-columns

- Définit les colonnes de la grille
  - Nombre infini (comme pour les lignes)
- La valeur “none” annule la grille

## grid-template-columns

- Les enfants occupent naturellement la grille
  - Ex : trois colonnes → **Premier** élément, **première** colonne. **Deuxième** élément, **deuxième** colonne. **Troisième** élément, **troisième** colonne, etc.



# grid-template-rows



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-columns: 250px 400px;  
}
```

Ici on définit deux colonnes, une de 250px et une autre de 400px

## Unité fr

- Représente une fraction de l'espace restant
  - $1\text{fr} = 100\%$  de l'espace restant
- Accepte une valeur décimale positive
- Prend en compte les gouttières dans le calcul
  - Contrairement à l'unité %
- A préférer l'unité fr à l'unité %
- Unité adaptative

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout/Basic\\_concepts\\_of\\_grid\\_layout#lunit%C3%A9\\_fr](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout#lunit%C3%A9_fr)

# Unité fr



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
}
```

Ici on définit quatre colonnes de tailles égales

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout/Basic\\_concepts\\_of\\_grid\\_layout#lunit%C3%A9\\_fr](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout#lunit%C3%A9_fr)

# Placer les éléments



css-grid

```
.mon-selecteur {  
  display: grid;  
  grid-template-columns: 250px 400px;  
}
```

Reprenons notre exemple

Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>

# Placer les éléments



Voilà ce que génère notre navigateur

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>

# Placer les éléments



Voilà que notre navigateur interprète. Il crée des “points” et étire les éléments en fonction de la configuration de la grille

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>

# Placer les éléments



Notre élément en bleu s'étire de 1 à 2, il est possible de changer ceci

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>

# Placer les éléments

- Le comptage commence à 1 et finit au nombre de colonnes / lignes + 1
  - Exemple : Trois colonnes = 1 à 4
- On peut compter
  - De gauche à droite : nombres positifs
  - De droite à gauche : nombres négatifs

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>



# Placer les éléments



css-grid

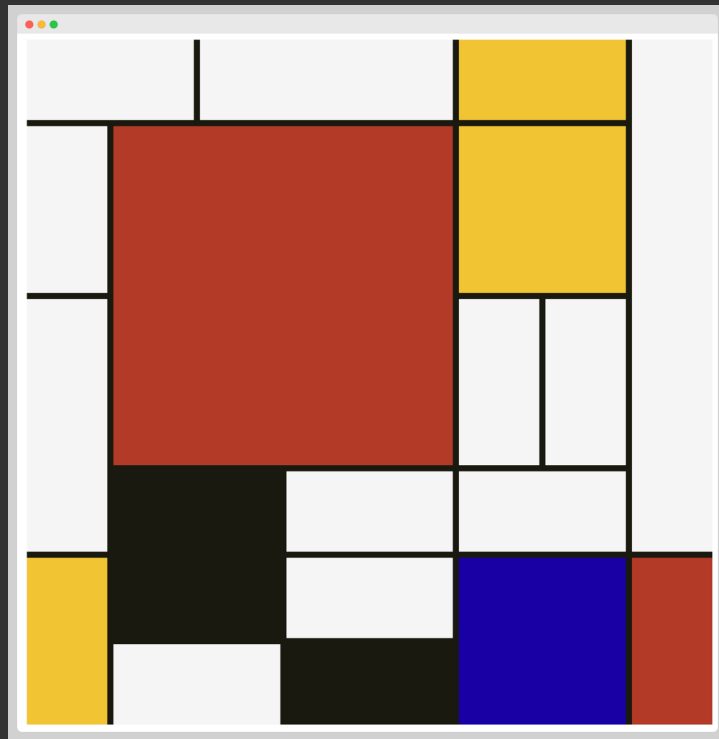
```
.mon-selecteur {  
  grid-column-start: 1;  
  grid-column-end: 3;  
}
```

Notre élément occupera deux colonnes grâce à ce code  
(le principe est le même avec les lignes et la propriété “grid-row-start” et “grid-row-end”)

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>

# Placer les éléments



Exemple de réalisation avec CSS Grid (voir sources)

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row-end>
- [https://github.com/DanYellow/cours/raw/refs/heads/main/integration-web-s2/cours-magistraux/numero-2/integration-web-s2\\_cours-magistraux\\_numero-2.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/integration-web-s2/cours-magistraux/numero-2/integration-web-s2_cours-magistraux_numero-2.ressources.zip)

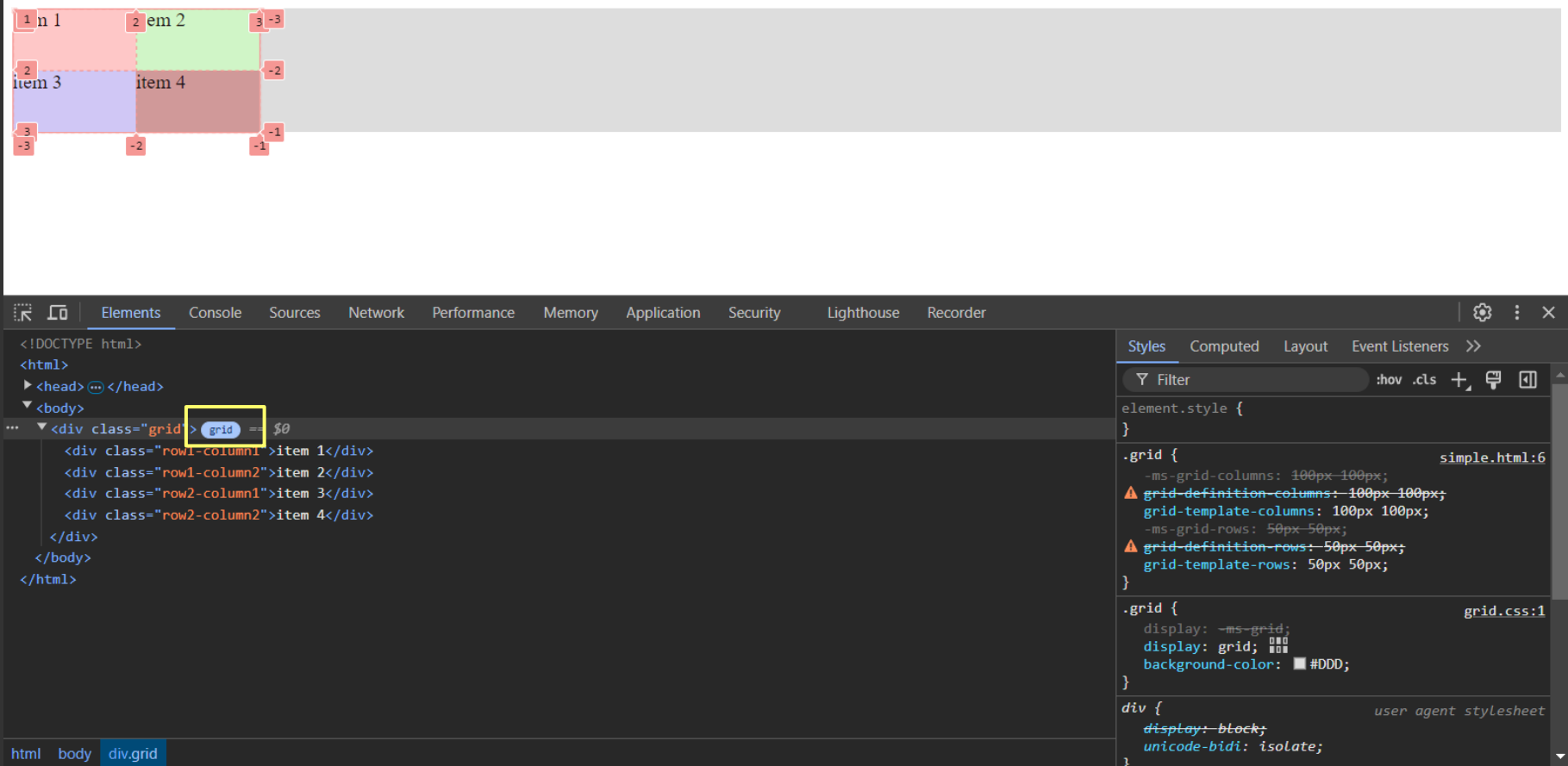
# Navigateur

- Chrome et Firefox proposent des aides visuelles pour afficher la grille du conteneur

## Source(s) :

- <https://developer.chrome.com/docs/devtools/css/grid?hl=fr>
- [https://firefox-source-docs.mozilla.org/devtools-user/page\\_inspector/how\\_to/examine\\_grid\\_layouts/index.html](https://firefox-source-docs.mozilla.org/devtools-user/page_inspector/how_to/examine_grid_layouts/index.html)

# Navigateur



Après avoir cliqué sur “grid”, notre navigateur affiche notre grille (Chrome)

# Pratiquons ! - CSS Grid (Partie 1/2)

Pré-requis :

- Avoir la ressource `ressources/css-grid`

A télécharger ici :

[https://github.com/DanYellow/cours/raw/refs/heads/main/integration-web-s3/travaux-pratiques/numero-5/integration-web-s3\\_travaux-pratiques\\_numero-5.ressources.zip](https://github.com/DanYellow/cours/raw/refs/heads/main/integration-web-s3/travaux-pratiques/numero-5/integration-web-s3_travaux-pratiques_numero-5.ressources.zip)

# Valeur subgrid

- Permet à un enfant lui-même grid d'avoir la même grille que le parent
  - Alignement parfait entre les deux
- Ne s'applique que pour les propriétés CSS
  - grid-template-columns et grid-template-rows

Source(s) :

- [https://developer.mozilla.org/fr/docs/Web/CSS/CSS\\_grid\\_layout/Subgrid](https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout/Subgrid)

# Fonctionnalités supplémentaires

- Possibilité de faire une mise en page en “brique”
  - Propriété “masonry” pas encore gérée par les navigateurs (2024)
- Possibilité de nommer les cellules (propriété “grid-template-areas”)
- ...

## Source(s) :

- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-template-areas>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-row>
- <https://developer.mozilla.org/fr/docs/Web/CSS/grid-column>

# Exemples de mises en page avec CSS Grid

- <https://gridbyexample.com/examples/>
- <https://igalia.github.io/css-grid-layout/>
- <https://grid.layoutit.com/>
  - Générateur de code pour CSS Grid



# Conclusion

- CSS Grid permet de faire des mises en page complexes
- Couvre les 20 % que Flexbox ne peut pas gérer
  - Peut se substituer à Flexbox à 100 %

**Questions ?**

