TRAVAUX DIRIGÉS
POSITIONNEMENT AVANCÉ GRID LAYOUT

1. INTRODUCTION

le but de ce TD est de découvrir les possibilités de positionnement offert par les propriétés Grid Layout de CSS3

2. SITOGRAPHIE

— https ://la-cascade.io/css-grid-layout-guide-complet/
— https ://grid.malven.co

3. EXERCICES :

3.1. **première grille.** Nous allons créer une grille de 2 lignes et 3 colonnes dans laquelle nous allons insérer des éléments. Pour déclarer la grille, il faut un container comme en Flexbox. De la même façon, Ce container va posséder les propriétés du parent gérant l'organisation. Déclarer les propriétés suivantes :

(1) **display : grid** : initie le container grid (les autres valeurs possibles sont inline-grid, et subgrid)
(2) **grid-template-rows :** défini le nombre et les hauteurs des lignes. chaque ligne est définie par une dimension en pixel, en % ou en fraction du contenant. par exemple :
    grid-template-rows : 480 px 200px 360px défini 3 lignes avec des hauteurs fixe
    grid-template-rows : 1fr 1fr 3fr défini 3 lignes, la 3ème possédant une hauteur 3 fois plus grande que les deux autres
(3) **display :grid-template-columns** de la même façon, défini le nombre de colonnes avec leur dimension
(4) Les propriétés **row-gap** et **column-gap** permettent de définir l'espacement horizontal et vertical entre chaque rangée et colonne
(5) les propriétés d'alignement :
    horizontal : *justify-content* l'alignement horizontal des éléments, et *justify-items* l'alignement horizontal dans les éléments
    vertical : *align-content* l'alignement vertical des éléments, et *align-items* l'alignement vertical dans les éléments
    Dans ce container, insérer 4 sections possédant des couleurs différentes.

3.2. **positionnement explicite dans une grille.** Le positionnement par défaut est assez réducteur et ne présente pas toutes les fonctionnalités de Grid. Nous allons en fait pouvoir faire prendre à différents éléments des tailles plus importantes que d'occuper juste une case de la grille définie. les propriétés permettant de définir les positions sont **grid-row-start, grid-row-end, grid-column-start** et **grid-column-end**. Les valeurs correspondent aux gouttières visées, pas aux blocs, donc pour 3 colonnes, vous avez 4 gouttières, Vous pouvez aussi utiliser des valeurs négatives pour les positions, cela revient à calculer la position à partir de la fin des colonnes ou lignes. Dans notre exemple, la valeur -1 pour les colonnes correspond à 4. Vous allez définir que le premier bloc occupe 2 éléments en largeur, le second, 2 en hauteur positionné à droite de premier bloc, le 3ème occupe 1 en largeur et en hauteur mais situé au centre, et enfin, le dernier occupe tout l'espace en bas.

3.3. **grille implicite.** Nous avons défini un schéma ou le nombre de lignes et de colonnes ont été défini directement. Quand les dimensions le permettent (dimension fixe ou pourcentage dont la somme est inférieure à 100%, si nous précisons des valeurs de propriétés **grid-row** ou **grid-column** hors des valeurs existante, des colonnes ou lignes supplémentaires sont créées. Essayer de créer un bloc avec les valeurs raccourcies
    vertical : *grid-row : 3 /5* la première valeur correspond à *grid-row-start* la seconde à *grid-row-end*
    horizontal : *grid-column : 2 / 3* idem mais pour les colonnes

On peux remarquer que les dimensions dépassent légèrement des dimensions actuelles. Elles sont liées au contenu. Par contre nous pouvons anticiper ces cas et dimensionner à l'avances les colonnes et lignes qui vont se rajouter. Il faut alors utiliser au niveau du conteneur les propriétés suivantes

vertical : *grid-auto-rows* dimensionne la hauteur des lignes

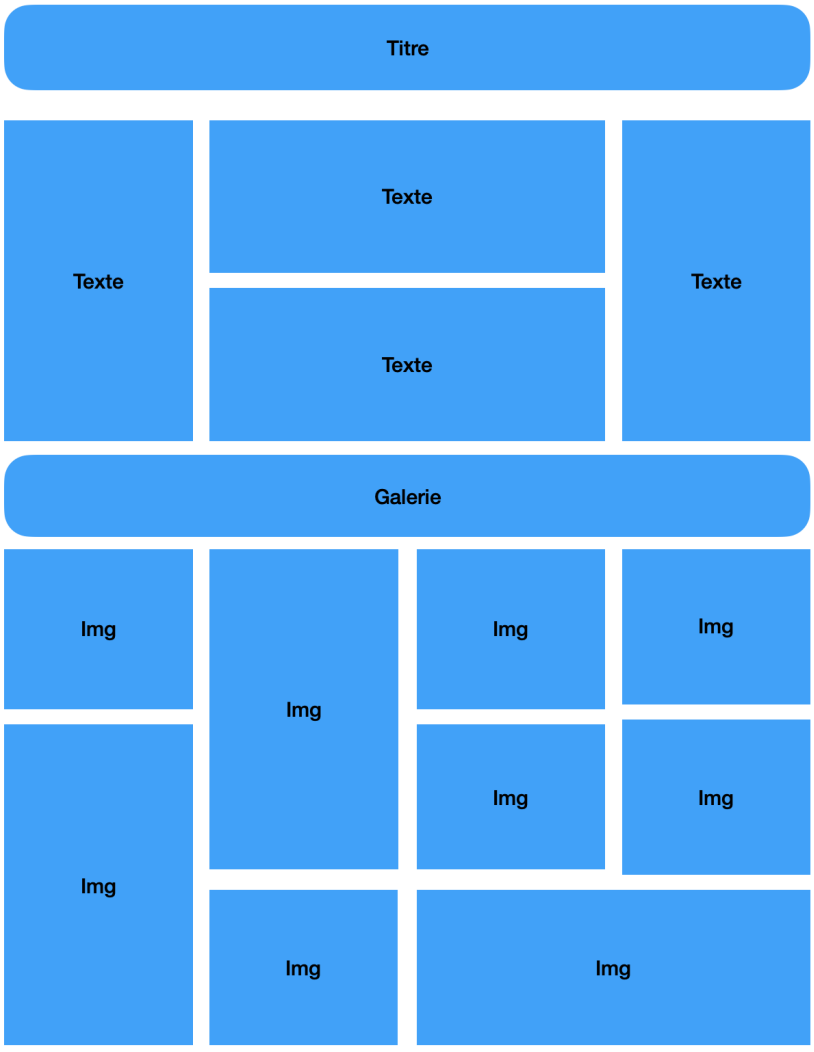
horizontal : *grid-auto-columns* idem mais pour les colonnes

Poser les dimensions des lignes et colonnes supplémentaires à 200px de large et 150px de hauteur

3.4. **axe d'organisation.** Comme pour Flexbox, il est possible de est possible de définir l'ordre d'empilement des éléments, horizontalement et/ou verticalement. La propriété définissant cela est **grid-auto-flow** prenant les valeurs row (par défaut), column ou dense. Par contre, cela n'a d'effet que sur les éléments qui ne sont pas placer explicitement. Ajouter un nouveau bloc sans positionnement explicite et tester les valeurs de cette propriété.

4. PROBLÈME

Voici un schéma à reproduire. Vous utiliserez le formalisme Grid pour le reproduire. Les dimensions



des colonnes seront de 200px et celle des gouttières de 10px. en hauteur, vous fixerez les zones de texte à 250px et celle des images à 200px (images carrées sur le plus petite unité.