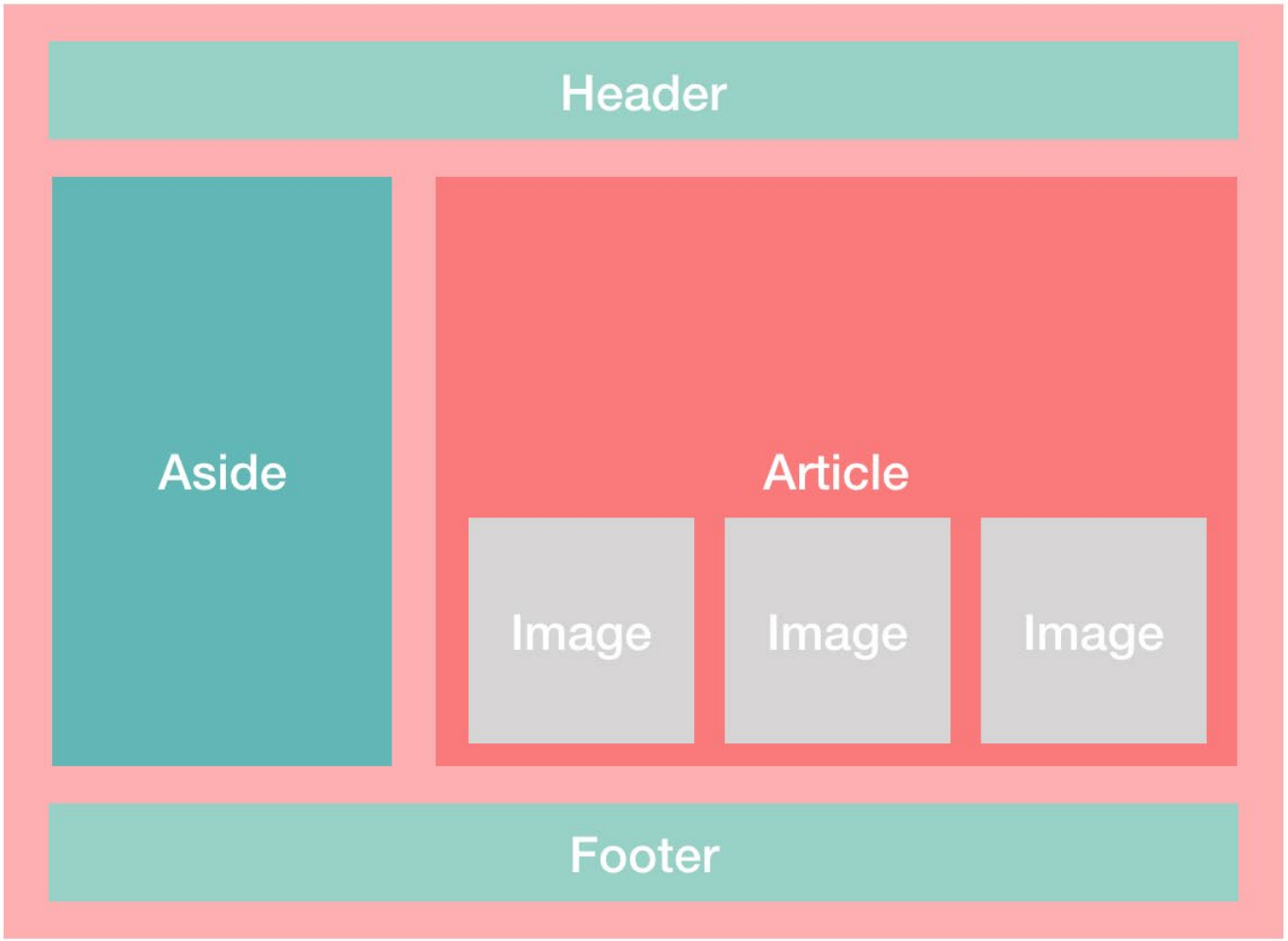
How I use CSS Grid and Flexbox to Create a One-page Website

CSS Grid for page layout and Flexbox for UI elements

Bien que nous puissions utiliser soit la grille CSS soit Flexbox pour la mise en page d'une page Web, la grille CSS tend à être utilisée pour la mise en page tandis que Flexbox tend à être utilisé pour l'alignement des éléments de l'interface utilisateur. Créons une page Web pour voir comment cela se passe dans le code réel. Je vais utiliser un design commun comme ci-dessous :



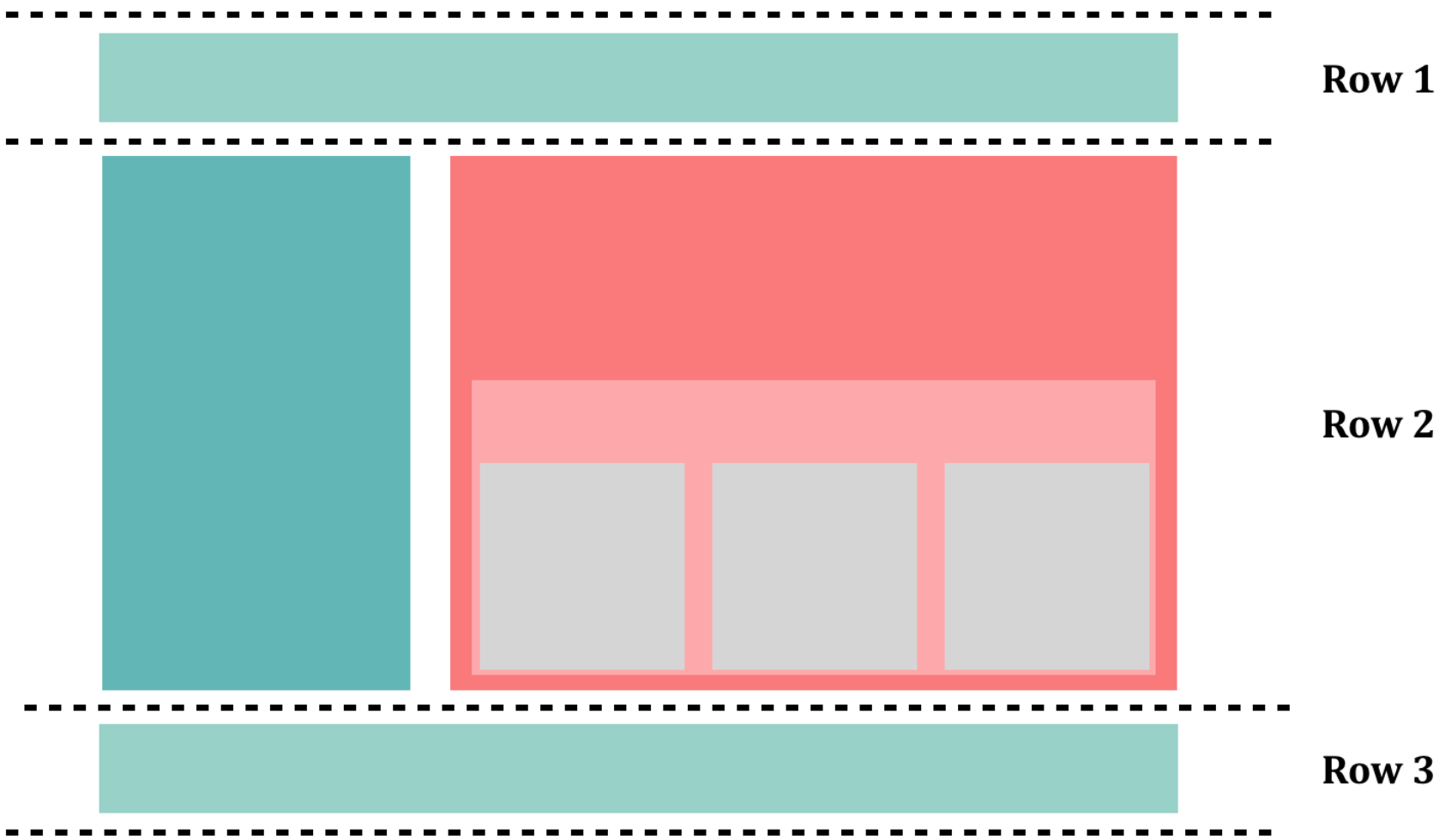
Etape 1 - Planifier avant le codage

L'astuce pour coder les mises en page est de planifier à l'avance. Je vais d'abord identifier les éléments parents et enfants en me basant sur la méthode de mise en page. Voici mon fonctionnement basé sur la conception.

Déterminer le nombre de lignes

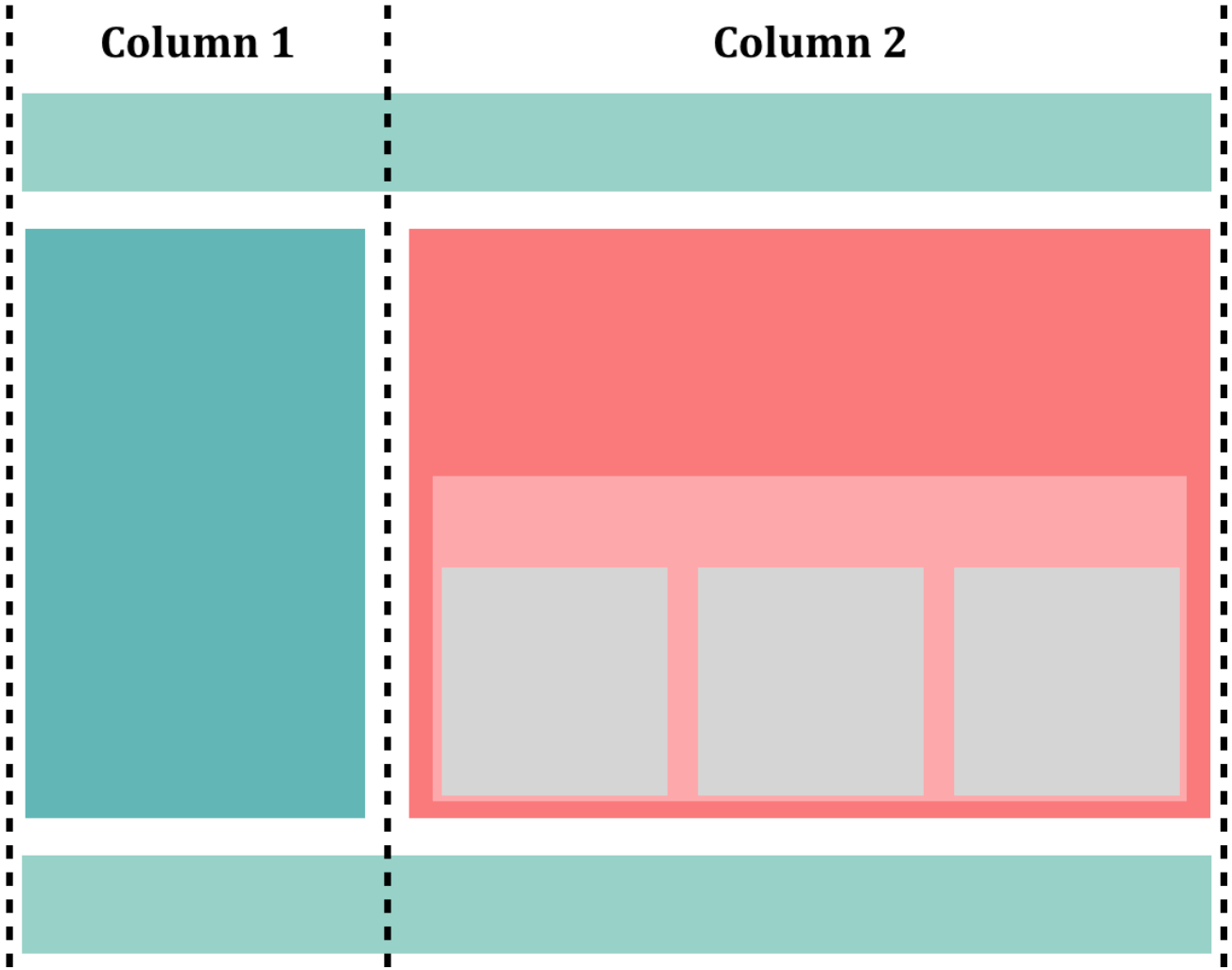
Je vais trier les éléments de la page en groupes horizontalement. Les éléments sont représentés sous forme de boîtes, et ceux qui se trouvent sur la même ligne (imaginaire) sont considérés comme étant sur la même rangée, par exemple le côté et l'article. S'ils devaient commencer ou se terminer sur une ligne différente, je les considérerais comme étant sur des lignes différentes.

Notez que les trois éléments et son conteneur dans l'article sont ignorés ici car ils ne font pas partie de la grille.



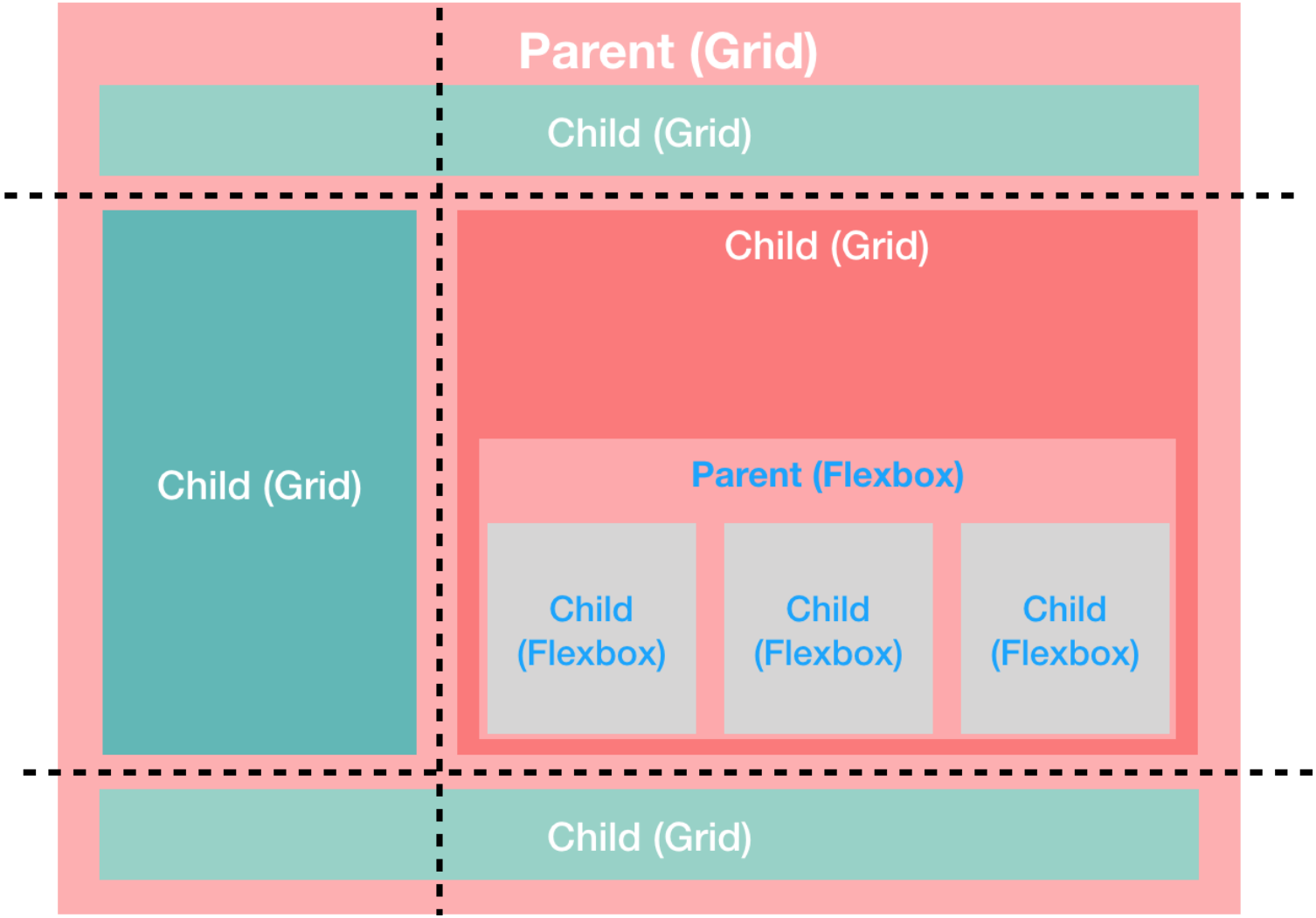
Déterminer le nombre de lignes.

Je vais trier les éléments en groupes verticalement. Les éléments situés le long d'une même ligne (imaginaire) seront considérés dans la même colonne.



Créer un guide

Après avoir déterminé le nombre de colonnes et de lignes, j'ai maintenant un guide que je peux suivre lors du codage. Dans ce guide, je vais aussi marquer les lignes de la grille et noter les éléments parents et enfants.



Etape 2 - Créer le fichier HTML

Pour une mise en page simple, je vais taper tout le html avant de travailler sur le CSS.

```
<body>
  <div class="grid-wrapper">
    <header>
      Header
    </header>
    <aside>
      <h4>Aside</h4>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque reprehenderit
inventore ipsum est omnis, totam deserunt temporibus rem optio quia obcaecati excepturi
consectetur, harum facilis alias veniam laborum iure! Expedita?</p>
    </aside>
    <article>
      <h1>Article</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore commodi
ratione pariatur eius et ullam odio temporibus corrupti autem atque quaerat soluta
saepe, ex dolorum laboriosam dolore minus repudiandae. Iste! Odit velit amet quisquam ut
        placeat quia tempore molestiae consectetur ab facilis eius necessitatibus veniam
autem, veritatis magnam magni ea tempora et. Perferendis molestiae cum saepe voluptas
tenetur aperiam optio! Corporis officia consectetur ipsa explicabo ipsum debitis.
        Voluptate excepturi aspernatur ipsa recusandae aliquam tempore officiis ullam
quis, enim eveniet magnam amet unde temporibus veniam, culpa reiciendis aliquid
corrupti! Quaerat, magni.</p>
      <div class="flex-wrapper">
        <span class="image"></span>
        <span class="image"></span>
        <span class="image"></span>
      </div>
    </article>
    <footer>Footer</footer>
  </div>
</body>
```

Étape 3 - Créer un fichier CSS

Après quoi, je vais créer le fichier CSS et le lier à mon fichier html.

Mise en place

J'enlève la marge et le padding par défaut des éléments html et body, et j'ajoute une hauteur de 100%.

Créer la grille

La grille CSS peut être intimidante au début car il existe plusieurs façons de créer une grille. Il est donc moins déroutant de s'en tenir à une syntaxe particulière lors de l'apprentissage.je déclare display : grid sur le div avec la classe de grid-wrapper. Cela transforme le div en un conteneur de grille. Ses enfants directs sont maintenant des éléments de grille. Les éléments qui sont dans ses enfants directs ne seront pas affectés.

Ensuite, je spécifie la largeur des lignes de la grille et la hauteur des colonnes de la grille.

Les lignes : Comme header et le footer ne contiennent que le contenu, et que je veux que aside et article se développent pour couvrir le reste de la page tout en poussant footer vers la fin de la page, j'ajoute grid-template-rows: min-content auto min-content.

Des colonnes :

Comme je veux prendre aside 1/4 de la largeur de la page et article pour prendre le reste, j'utilise grid-template-columns: 1fr 3fr. L'unité fr représente une fraction de l'espace disponible dans le conteneur de la grille. Quant au header et au footer, je n'ai pas à y penser puisqu'ils s'étalent sur les deux colonnes, occupant une largeur de 4fr.

Insérer des éléments dans les cellules de la grille

Je spécifie la taille et la position de chaque élément de la grille en utilisant les propriétés abrégées des colonnes et des rangées de la grille.l'en-tête et le pied de page couvriront la surface de 2 colonnes. La longueur de la grid-column: 1/3 est :

```
grid-column-start: 1;
grid-column-end: 3;
```

De même, la déclaration de la ligne de grille : 2/3 signifie qu'elle commence sur la deuxième ligne horizontale et se termine sur la troisième ligne horizontale.

```
grid-row-start: 2;
grid-row-end: 3;
```

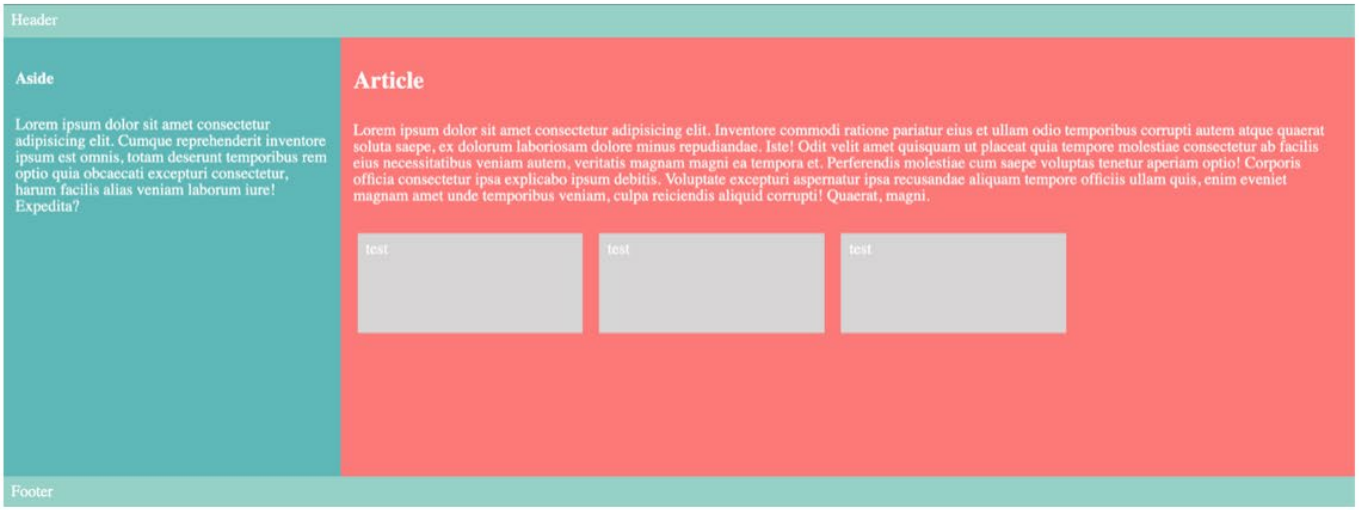
```css
aside {
  grid-column: 1/2;
  grid-row: 2/3;
  background-color: #5EB7B7;
}

header {
  grid-column: 1/3;
  grid-row: 1/2;
  background-color: #96D1C7;
}

footer {
  grid-column: 1/3;
  grid-row: 3/4;
  background-color: #96D1C7;
}

article {
  grid-column: 2/3;
  grid-row: 2/3;
  background-color: #FC7978;
}
```

Vous trouverez ci-dessous le résultat final de l'utilisation de la grille CSS :

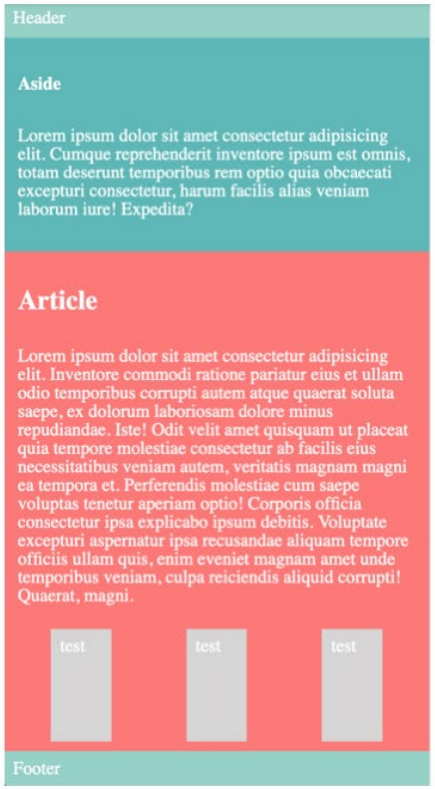


En créant flex-container et flex-child

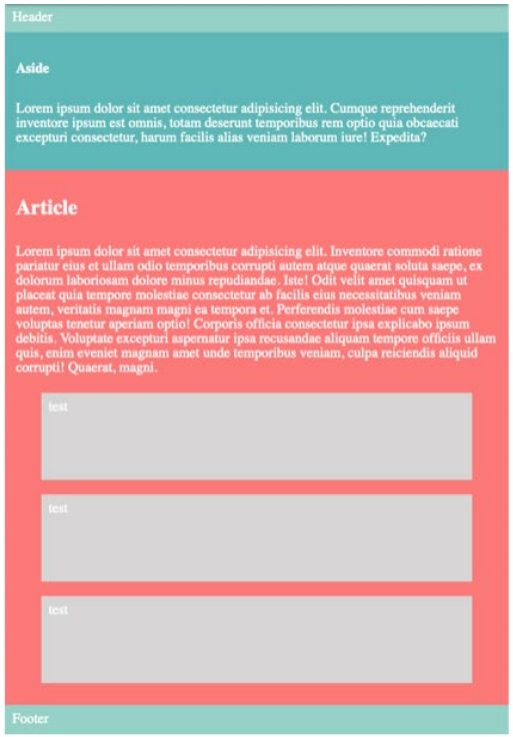
Ensuite, je travaille sur l'espacement des éléments d'image, qui sont représentés par des boîtes grises. Pour le transformer en flex-container, j'ajoute l'affichage : flex. Pour améliorer la réactivité de la page, je veux que la largeur de mes flex-items s'ajuste en fonction de la largeur de la page. Ceci peut être réalisé en réglant flex-grow : 1 sur les trois images.

La vue mobile

Enfin, je travaille sur la vue mobile. En mobile, on a tendance à disposer les choses vers le bas plutôt que sur le côté car la taille de l'écran est limitée. Pour cela, je peux le faire en une étape : Ajouter une requête média pour appliquer l'affichage display: block sur mon grid-container quand la taille de l'écran est petite. C'est une façon assez soignée, non ?



Les éléments flexibles à l'intérieur de article ont besoin de plus de travail car ils sont toujours disposés en ligne. Pour changer de rangée à colonne, j'ajoute flex-direction: column au flex-container. Pour des raisons de style, j'ajoute aussi une margin-bottom : 20px à chaque flex-item pour créer un espace entre les images.



Comme toujours, la dernière étape sera de vérifier s'il y a des problèmes d'alignement aux différents points de rupture en changeant la largeur du navigateur.

Summary

To recap, the layout properties we've used in this article are:

Grid

`display: grid`
`grid-template-columns` and `grid-template-rows` (note the plural form)
`grid-column` and `grid-row` (note the singular form)

Flexbox

`display: flex`

`flex-direction`
`flex-grow`

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>

<body>
  <div class="grid-wrapper">
    <header>
      Header
    </header>
    <aside>
      <h4>Aside</h4>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque reprehenderit inventore ipsum est omnis, totam deserunt temporibus rem optio quia obcaecati excepturi consectetur, harum facilis alias veniam laborum iure! Expedita?</p>
    </aside>
    <article>
      <h1>Article</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore commodi ratione pariatur eius et ullam odio temporibus corrupti autem atque quaerat soluta saepe, ex dolorum laboriosam dolore minus repudiandae. Iste! Odit velit amet quisquam ut
        placeat quia tempore molestiae consectetur ab facilis eius necessitatibus veniam autem, veritatis magnam magni ea tempora et. Perferendis molestiae cum saepe voluptas tenetur aperiam optio! Corporis officia consectetur ipsa explicabo ipsum debitis.
        Voluptate excepturi aspernatur ipsa recusandae aliquam tempore officiis ullam quis, enim eveniet magnam
amet unde temporibus veniam, culpa reiciendis aliquid corrupti! Quaerat, magni.</p>
      <div class="flex-wrapper">
        <span class="image">test</span>
        <span class="image">test</span>
        <span class="image">test</span>
      </div>
    </article>
    <footer>Footer</footer>
  </div>
</body>

</html>
```

CSS

```
aside, header, footer, article, span {
  padding: 10px;
}

html, body {
  height: 100%;
  margin: 0;
}

body {
  color: white;
  font-family: san-serif;
  font-size: 1.2em;
}

.grid-wrapper {
  /*changed the height from 100% to auto as the layout breaks when the height of the content
'overflows'.*/
  height: auto;
  display: grid;
  grid-template-columns: 1fr 3fr;
  grid-template-rows: min-content auto min-content;
  background-color: #FFAFB0;
}

aside {
  grid-column: 1/2;
  grid-row: 2/3;
  background-color: #5EB7B7;
}

header {
  grid-column: 1/3;
  grid-row: 1/2;
  background-color: #96D1C7;
}

footer {
  grid-column: 1/3;
  grid-row: 3/4;
  background-color: #96D1C7;
}

article {
  grid-column: 2/3;
  grid-row: 2/3;
/*   background-color: #FC7978; */
  border: 1px solid black;
}

.flex-wrapper {

 flex-direction: column;

  }

.image {

    margin-bottom: 20px;

   }

}
```