LYCÉE ARISTIDE BERGÈS
SIO

CSS

131 CSS : structures, grilles et flexibilité

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

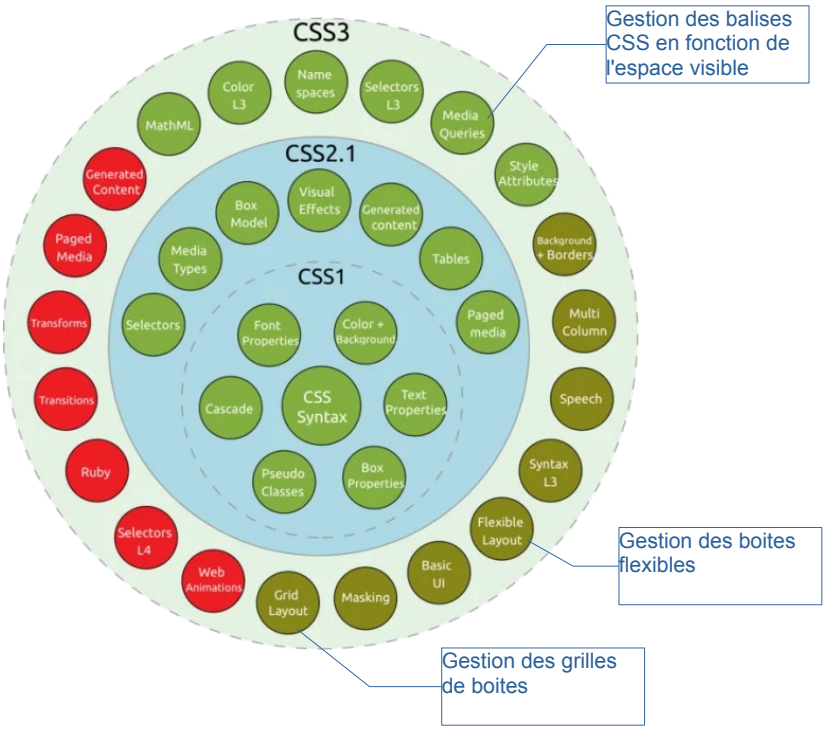
Date	Révision
2023-11-18	Ajout des exemples (en particulier, les animations)

Sommaire

A Structure CSS

Comme cela a été évoqué dans le cours précédent, CSS4 n'existe pas, car trop imposant pour pouvoir valider la conformité et la bonne implémentation de toutes les fonctionnalités.

On trouve donc actuellement une norme CSS3 et de nombreux modules :



Parmi ces modules de la norme CSS3, on trouve des modules de structuration.

- des configurations en fonctions de l'espace visible disponible (Media Queries)
- Des grilles CSS (Grid layout)
- des boites flexibles (Flexible layout)

Nous allons donc découvrir l'intérêt de ces fonctionnalités et leur usage.

A.1 Le module "Media Query"

A.1.1 Explications

Les requêtes média (media queries) permettent de modifier l'apparence d'un site ou d'une application en fonction du type d'appareil (impression ou écran par exemple) et de ses caractéristiques (la résolution d'écran ou la largeur de la zone d'affichage (viewport) par exemple).

A.1.2 Type de media

On peut prendre en compte un écran (`screen`) mais aussi une imprimante (`printer`) ou un descripteur vocal (`speech`).

Depuis l'évolution du module media query 4, les type suivant n'existe plus : `tty`, `tv`, `projection`, `handheld`, `braille`, `embossed` et `aural`.

Le contenu est alors différent selon le média. Ci-après, quelques exemples d'applications.

```
@media screen { code css  }
@media print { autre code css  }
@media screen, speech { code css commun a deux types }
```

A.1.3 Définition du media

CSS permet également de définir plusieurs définitions. Il est possible d'utiliser des mots-clés comme `aspect-ratio`, `min-width`, `max-width` , `min-height`, `max-height` ou `orientation`.

```
@media (min-width: 30em)
@media (aspect-ratio: 11/5)
@media (orientation: landscape)
```

Si la définition ne s'applique qu'à un seul type, il suffit d'ajouter l'opérateur `and` à la suite du type et de fournir les conditions d'applications.

```
@media screen and (min-width: 30em) and (orientation: landscape) { ... }
```

Enfin, pour simplifier l'usage, CSS4 propose pour ce module l'usage des signes `<`, `<=`, `>=` et `>` avec le mot-clé `width` ou `height` (sans les préfixes `max-` ou `min-`).

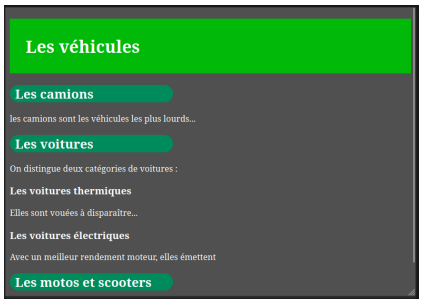
Exemple :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS @Media Queries</title>

    <style>
        body {
            background-color: #505050;
            color: white;
        }
        h2 {
            border-radius: 50px;
            padding-left: 10px;
            width: 300px;
        }
        @media screen and (width <= 300px) {
            h1 {
                background-color: rgb(185, 0, 0);
            }
            h2 {
                background-color:rgb(139, 51, 0);
            }
        }

        /* définition importante */
        @media screen and (width >= 700px) {
            h1 {
                background-color: rgb(0, 185, 9);
                padding: 30px;
            }
            h2 {
                background-color:rgb(0, 139, 93);
            }
        }
    </style>

</head>
<body>
    <h1>Les véhicules</h1>
    <h2>Les camions</h2>
    <p> les camions sont les véhicules les plus lourds...</p>
    <h2>Les voitures</h2>
    <p>On distingue deux catégories de voitures :</p>
    <h3>Les voitures thermiques</h3>
    <p>Elles sont vouées à disparaître...</p>
    <h3>Les voitures électriques</h3>
    <p>Avec un meilleur rendement moteur, elles émettent</p>
    <h2>Les motos et scooters</h2>
    <h2>Les trotinettes</h2>
</body>
</html>
```

Dans cet exemple, il y a une partie de CSS fixe (border-radius, padding, etc.) et deux parties distinctes : en dessous de 301 pixels de large, les titres sont en rouge (on est écrasé 😉) et passent en vert lorsqu'on dépasse 700 pixels de large.

Pour tester le fonctionnement du code, il faut passer en mode débogueur, puis sélectionner une taille d'affichage dans le navigateur :

Le résultat devrait être similaire aux captures ci-dessous :

Moins de 301 pixels	Entre 301 et 700 pixels	Au-delà de 700 pixels
		

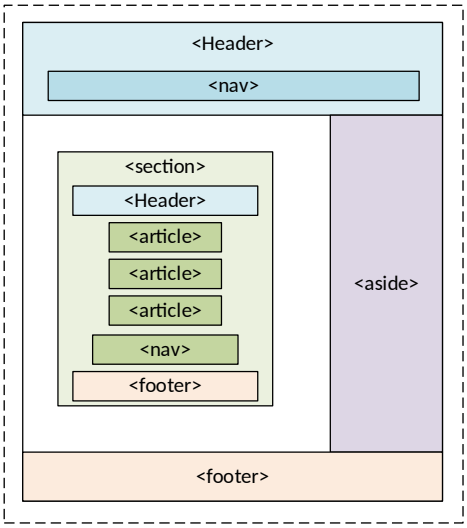
C'est la technique utilisée pour déterminer un affichage selon la taille et l'orientation de l'appareil.

La plupart des frameworks comme Bulma, Bootstrapp, Picnic, Foundation… utilisent ce module CSS pour la gestion des menus.

A.2 Éléments de structure

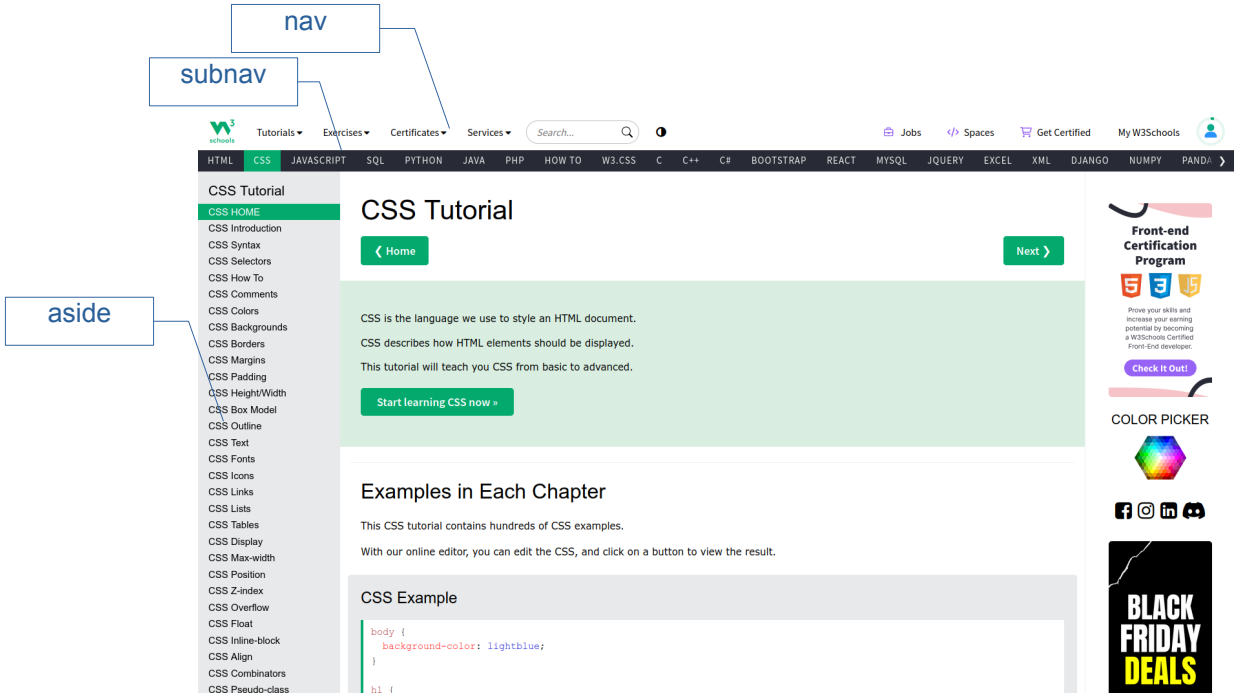
Bien que nous puissions utiliser n'importe quelle balise pour créer un bloc, un standard de fait est né : la logique d'une page web étant souvent la même, on retrouve les balises suivantes, implémentées dans HTML5.

- Menu utilisateur : utiliser `<nav>`
- Section : utiliser la balise `<section>`
- Entête : utiliser la balise `<header>`
- Pied de page : utiliser la balise `<footer>`
- Afficher un article : utiliser la balise `<article>`
- Bandeau vertical gauche ou droite : utiliser `<aside>`



De plus, cette norme formalise également certains éléments qui peuvent être utilisés à plusieurs endroits (header et footer peuvent apparaître dans la page comme dans une section).

Ainsi, la présentation des pages du site w3schools répond à une normalisation logique :



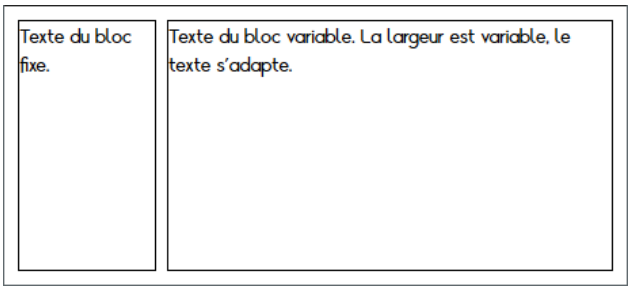
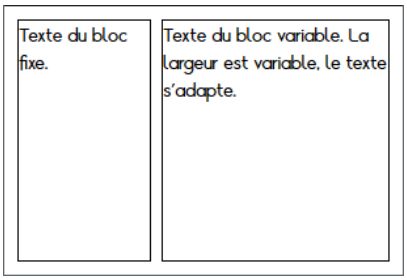
Note : Bien que les balises existent, les navigateurs ne savent pas les placer automatiquement. Il faudra donc toujours fournir un style CSS associé à ces balises. Il s'agit donc plutôt de balises sémantiques, permettant une lecture plus aisée de la page par un humain.

Voici un premier exemple de page utilisant ces balises.

Partie CSS	Partie HTML

aside {
position:fixed;
border:solid 1px;
width:100px;
top:10px;
left:10px;
bottom:10px;
}
article {
position:fixed;
border:solid 1px;
top:10px;
left:120px;
bottom:10px;
right:10px;
}
	<!doctype html>
	<html lang="fr">
	<head>
	<meta charset="UTF-8">
	<title>Document</title>
	<link rel="stylesheet" type="text/css" href="style.css">
	</head>
	<body>
	<aside>
	Texte du bloc fixe.
	</aside>
	<article>
	Texte du bloc variable. La largeur est variable, le texte s'adapte.
	</article>
	</body>
	</html>

Voici le résultat : Le premier bloc est à 10 pixels du bord et a une largeur de 100 pixels. Le deuxième bloc est placé à 120 pixels du bord, mais sa largeur n'est pas spécifiée. Par défaut un bloc occupe alors toute la largeur restante.

large	étroit
	

Pour créer une sorte de grille de boites, la position fixe devient complexe, car il faut calculer toutes les largeurs et ajouter tous les espaces entre les boites… mais en plus, ce site présenterait l'inconvénient de ne pas s'adapter à la largeur de la fenêtre !

A.3 Les grilles CSS

Ce module est une réponse pour faciliter la gestion des emplacements de conteneurs dans une page.

Il offre l'avantage de pouvoir placer les éléments sur une page, selon les deux axes (vertical et horizontal) et selon une logique simple (qui évite les calculs).

A.3.1 Fonctionnement

Pour simplifier et rapidement comprendre, l'espace visible dans la page web va être divisé par une grille multiple de 6 (généralement 6, 12, 24). Le choix mathématique du nombre 6 est simple : on peut le diviser par 1, 2, 3.

Le conteneur général aura une propriété `display: grid` ou `display: inline-grid`.

Note : dans une page web, Firefox dispose d'un inspecteur de grille qui permet d'afficher graphiquement la grille dans la page (espacement, colonne, etc.). Pour cela, il suffit de cliquer sur les symboles [grid] dans l'outil de débogage, onglet inspecteur. Le sous-onglet "mise en page" permet d'afficher des options (couleurs, titres et numéros des grilles).

Le comportement de `inline-grid` et `grid` est similaire à `inline-block` et `block`.

On peut ajouter les propriétés `grid-template-rows` ou `grid-template-columns` pour définir respectivement des rangées et des colonnes.

Bien que l'on puisse utiliser des unités absolues (px) pour définir les valeurs des rangées et colonnes, il existe une unité specifique fr (fraction) qui permet de définir des fractions de l'espace. Exemple :

```
grid-template-columns: 1fr 4fr 1fr;
```

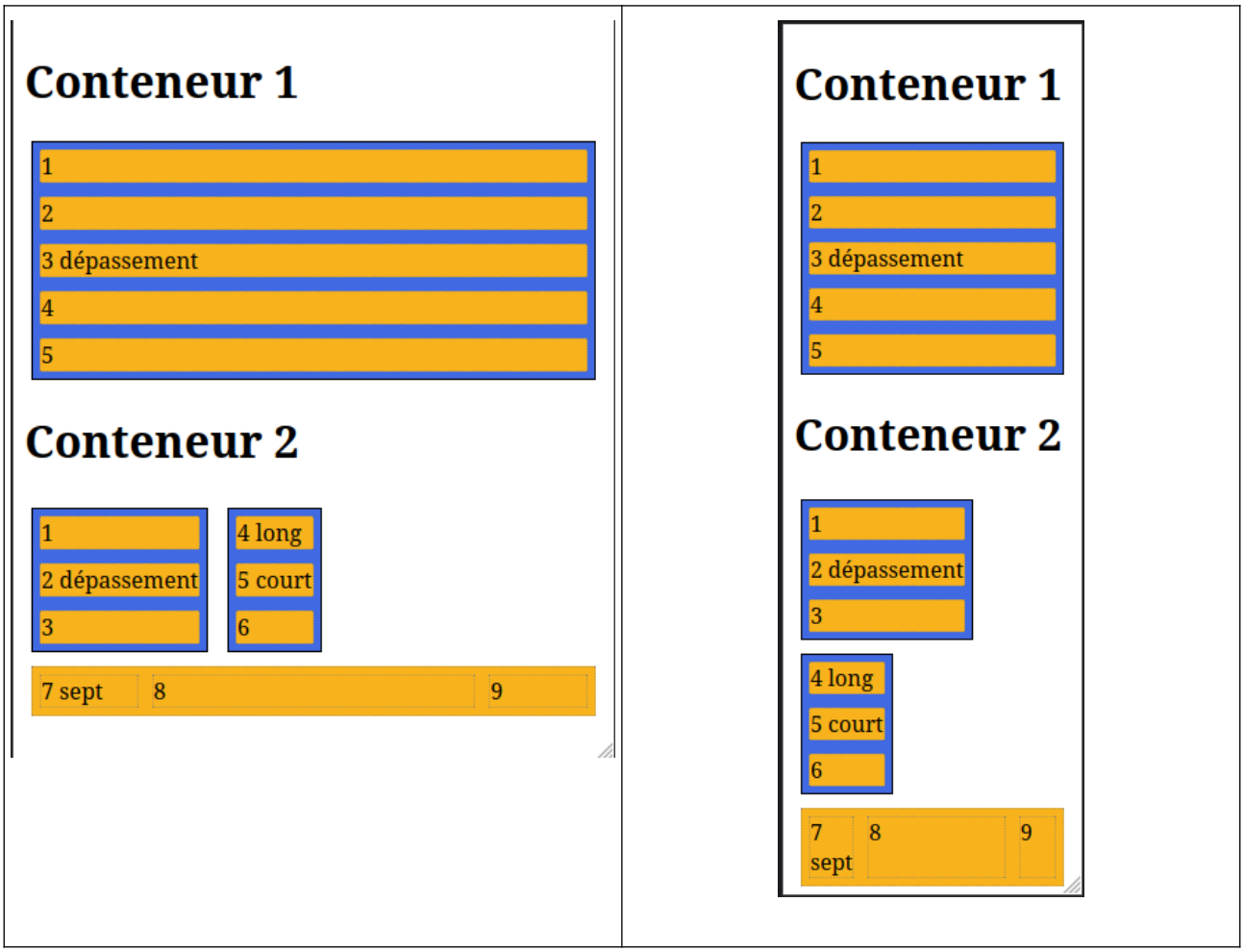
Dans ce cas, il existe 3 colonnes, la première prend 1/6^{e} de largeur, la deuxième mesure 4/6^{e} et la dernière fait également 1/6^{e}.

A.3.2 Exemple d'usage de grille

Voici un exemple simple, avec trois modèles de grilles :

```
<!DOCTYPE html>
<HTML lang="fr">
<HEAD>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS grid</title>
    <style>
        .grid-container1 {
            display: grid;
            border: solid black 1px;
            padding: 15 px;
            margin: 5px;
            background-color: royalblue;
        }
        .grid-container2 {
            display: inline-grid;
            border: solid black 1px;
            padding: 15 px;
            background-color: royalblue;
        }
        .grid-container3 {
            display:grid;
            grid-template-columns: 1fr 3fr 1fr;
        }
        div {
            background-color: rgb(248, 179, 28);
            margin: 5px;
            border: dotted grey 1px;
        }
    </style>
</HEAD>

<BODY>
    <h1>Conteneur 1</h1>
    <div class="grid-container1">
        <div class="c1">1</div>
        <div class="c1">2</div>
        <div class="c1">3 dépassement</div>
        <div class="c1">4</div>
        <div class="c1">5</div>
    </div>
    <h1>Conteneur 2</h1>
    <div class="grid-container2">
        <div class="c2">1</div>
        <div class="c2">2 dépassement</div>
        <div class="c2">3</div>
    </div>
    <div class="grid-container2">
        <div class="c2">4 long</div>
        <div class="c2">5 court</div>
        <div class="c2">6</div>
    </div>
    <div class="grid-container3">
        <div class="c2">7 sept</div>
        <div class="c2">8</div>
        <div class="c2">9</div>
    </div>
</BODY>
</HTML>
```

Le résultat reste dynamique et permet un affichage pratique et bien géré.



En effet, on constate que le `grid-template-columns` conserve ses propriétés. Lorsqu'il n'y a plus assez de place, les grilles en `inline-grid` se placent naturellement les unes sous les autres.

Ainsi, la page reste toujours visible depuis un téléphone mobile jusqu'au grand écran.

A.4 Les boites flexibles

Il s'agit d'un module permettant de structurer l'alignement, la direction, l'ordre et la taille des éléments.

Les conteneurs deviennent flexibles (ou souples) : ils ont donc la capacité de voir leur taille se contracter ou s'élargir, mais surtout, les contenants seront également flexibles.

A.4.1 Propriétés

Les modifications s'appliquent alors par rapport à un **axe** (`flex-direction: row` ou `flex-direction: columns`).

Il est possible d'utiliser un **découpage** automatique (`flex-wrap: nowrap` ou `flex-wrap: wrap` ou `flex-wrap: wrap-reverse`).

La modification simultanée de l'axe et du découpage utilise `flex-flow` (exemple, `flex-flow : row wrap;`).

Il est ensuite possible de **justifier** le contenu d'un conteneur (pour le centrer, par exemple). On utilise alors `justify-content: center`. Pour placer à gauche ou à droite, on utilise `justify-content: flex-start` ou `justify-content : flex-end`. On peut également répartir les contenus avec la propriété `justify-content: space-around`.

Enfin, il est possible d'**aligner** les contenus, horizontalement : si les contenus ont des hauteurs différentes, alors on utilisera `align-items:` avec les valeurs `baseline` (alignement par le milieu), `stretch` (tous les contenus auront la même hauteur), `flex-start` (alignement par le haut) et `flex-end` (alignement par le bas).

On notera que l'organisation flexible permet de centrer parfaitement (verticalement et horizontalement) un contenu dans un conteneur, en utilisant `justify-content: center` et `align-items: center`.

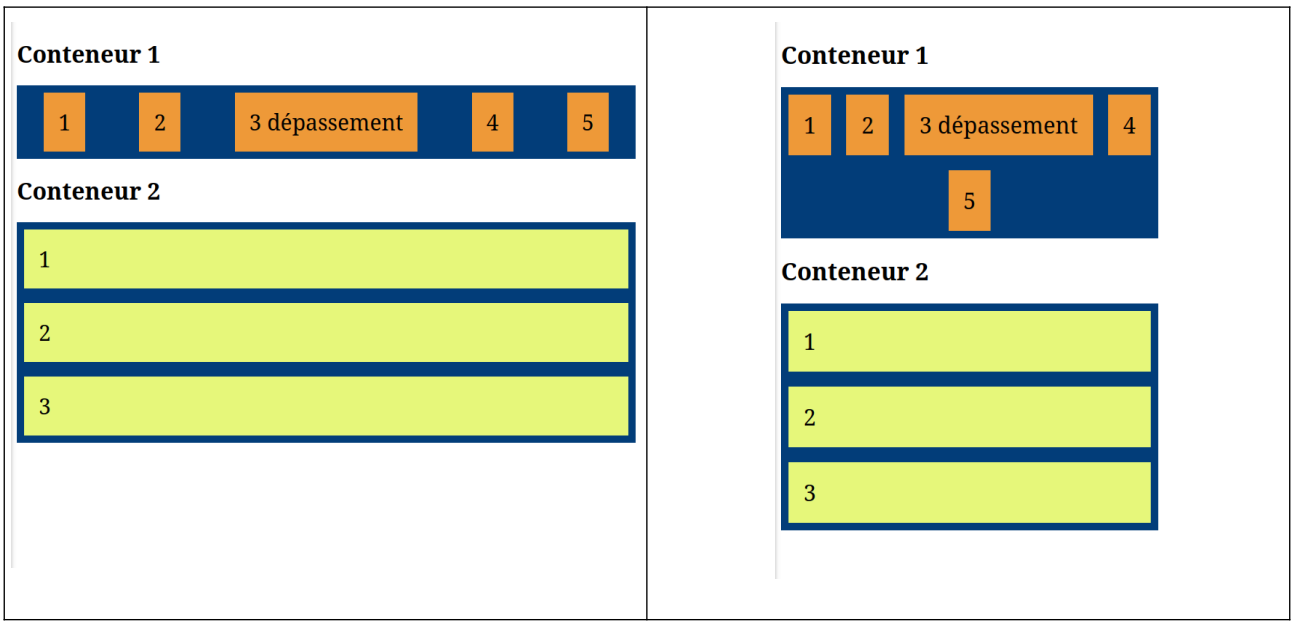
A.4.2 Exemple de contenu flexible

Voici un exemple simple de gestion de contenus.

```
<!DOCTYPE html>
<HTML lang="fr">
<HEAD>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS flex</title>
    <style>
        .flex-container1 {
            display: flex;
            background-color: rgb(2, 61, 121);
            flex-direction: row;
            /* flex-wrap: wrap; */
            justify-content:space-around;
        }
        .flex-container2 {
            display: flex;
            background-color: rgb(2, 61, 121);
            flex-direction: column;
        }

        /* style pour les div contenues dans flex-container */
        .flex-container1>div {
            background-color: #ee9938;
            margin: 10px;
            padding: 20px;
            font-size: 30px;
        }

        /* style pour les div contenues dans flex-container2 */
        .flex-container2>div {
            background-color: #e6f77a;
            margin: 10px;
            padding: 20px;
            font-size: 30px;
        }
    </style>
</HEAD>

<BODY>
    <h1>Conteneur 1</h1>
    <div class="flex-container1">
        <div class="c1">1</div>
        <div class="c1">2</div>
        <div class="c1">3 dépassement</div>
        <div class="c1">4</div>
        <div class="c1">5</div>
    </div>
    <h1>Conteneur 2</h1>
    <div class="flex-container2">
        <div class="c2">1</div>
        <div class="c2">2</div>
        <div class="c2">3</div>
    </div>
</BODY>
</HTML>
```

Ici, nous avons deux conteneurs différents. La gestion des `div` contenues montre la souplesse des propriétés pour les boites flexibles. Cela améliore l'agencement des pages web.

Dans l'exemple ci-dessous, la ligne `flex-wrap: wrap` ; a été dé-commentée.



Les éléments flex apportent une finesse plus importante dans l'affichage des informations et des éléments.

A.5 Que choisir : grid ou flex ?

Les deux modules sont intensément utilisés dans les frameworks CSS les plus importants. Ils n'ont pas les mêmes caractéristiques.

Grille	Flexible
• Capable de gérer 2 dimensions simultanément • Flexbox peut s'utiliser à l'intérieur d'une grille	• Adaptation automatique quelle que soit la taille de l'écran (remplissage de l'écran)
	• Ne gère qu'une dimension à la fois • Il n'est pas possible d'utiliser une grille dans un conteneur flexbox

Les deux modules restent plutôt complémentaires : le besoin détermine l'outil.