Architecture et langages Web
(NFA040)

CSS : grid (tutoriel sur les gabarits)

Remarque. Ce tutoriel est extrait de la section de MDN intitulée : Gabarits de zones de grille. Les sources des exemples accompagnent ce sujet.

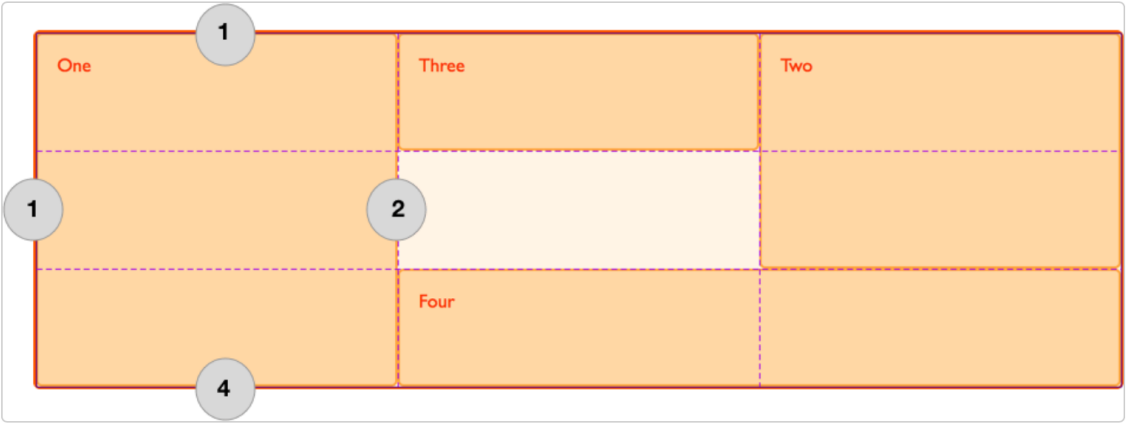
Dans le guide précédent, on a étudié les lignes formées par une grille et comment positionner des objets sur ces lignes. Lorsqu'on utilise une grille CSS, on a toujours ces lignes et celles-ci permettent d'avoir une disposition simple. Toutefois, il existe une autre méthode de disposition avec les grilles, qu'on peut utiliser seule ou combinée avec les lignes. Avec cette méthode, on place les éléments sur des *zones* de la grille. Nous allons voir dans ce guide comment cela fonctionne voire comment on peut faire de l'ASCII-art en CSS avec les grilles !

1 Donner un nom à une zone de grille

On a déjà utilisé la propriété `grid-area` précédemment. C'est cette propriété qui utilise les numéros des lignes comme valeur pour positionner une zone de grille :

```
.box1 {
   grid-area: 1 / 1 / 4 / 2;
}
```

Ici, on définit les quatre lignes qui entourent la zone en question :



On peut également définir une zone en lui donnant un nom puis en définissant l'emplacement de cette zone grâce à la propriété `grid-template-areas`. Vous pouvez choisir les noms de vos zones, on peut par exemple créer une disposition avec quatre zones :

- Un en-tête (*header*)
- Un pied de page (*footer*)
- Une barre latérale (*sidebar*)
- Le contenu principale (*content*)

Avec `grid-area`, on affecte un nom à chacune de ces zones. Pour le moment, aucune disposition n'a été créée mais on a des noms qu'on pourra utiliser dans notre disposition :

```
.header {
  grid-area: hd;
}
.footer {
  grid-area: ft;
}
.content {
  grid-area: main;
}
.sidebar {
  grid-area: sd;
}
```

Grâce à ces noms, on peut créer l'organisation. Cette fois, plutôt que de placer les objets grâce aux numéros de ligne, on définit la disposition dans le conteneur de la grille :

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-auto-rows: minmax(100px, auto);
  grid-template-areas:
    "hd hd hd hd   hd   hd   hd   hd   hd"
    "sd sd sd main main main main main main"
    "ft ft ft ft   ft   ft   ft   ft   ft";
}

<div class="wrapper">
  <div class="header">En-tête</div>
  <div class="sidebar">Barre latérale</div>
  <div class="content">Contenu</div>
  <div class="footer">Pied de page</div>
</div>
```

Grâce à cette méthode, il n'est pas nécessaire de gérer chacun des éléments individuellement. Tout est organisé au travers du conteneur. La disposition est décrite grâce à la propriété `grid-template-areas`.

2 Laisser une cellule vide

Dans l'exemple précédent, toute la grille est occupée... On peut également utiliser cette méthode pour laisser des cellules vides. Pour cela, il faut utiliser un point à la place d'un nom de zone. Aussi, si on veut que le pied de page soit uniquement affiché sous le contenu, il faudra avoir trois cellules vides sous la barre latérale.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-auto-rows: minmax(100px, auto);
  grid-template-areas:
      "hd hd hd hd   hd   hd   hd   hd   hd"
      "sd sd sd main main main main main main"
      ".  .  .  ft   ft   ft   ft   ft   ft";
}
```

Si on veut que la disposition soit bien représentée, on peut utiliser plusieurs points. Tant que ceux-ci ne sont pas séparés par un espace, ils compteront pour une seule cellule. Dans le cas d'une disposition complexe, cela permet d'avoir des lignes et colonnes clairement alignées, y compris dans la règle CSS.

3 Occuper plusieurs cellules

Dans notre exemple, chacune des zones occupe plusieurs cellules car on a répété le nom de la zone avec des espaces entre (on peut ajouter plus d'espaces si besoin, afin d'avoir une disposition lisible, c'est ce qu'on a fait précédemment pour que `hd` et `ft` soient alignés avec `main`).

La zone qu'on crée avec les noms doit être rectangulaires. Actuellement, il n'existe pas de méthode pour créer une zone avec une forme de L (bien que la spécification indique qu'une prochaine version pourrait couvrir cette fonctionnalité). On peut toutefois agrandir des lignes horizontales aussi simplement que des colonnes. Par exemple, on pourrait avoir la barre latérale qui descend jusqu'en bas en remplaçant les points par `sd`.

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-auto-rows: minmax(100px, auto);
  grid-template-areas:
    "hd hd hd hd   hd   hd   hd   hd   hd"
    "sd sd sd main main main main main main"
    "sd sd sd  ft  ft   ft   ft   ft   ft";
}
```

La valeur utilisée pour `grid-template-areas` doit obligatoirement décrire une grille complète, sinon elle est considérée invalide et la propriété est ignorée. Cela signifie qu'il faut le même nombre de cellules pour chaque ligne (si une cellule est vide, on l'indiquera avec un point). Si des zones ne sont pas rectangulaires, cela sera également considéré comme invalide.

4 Redéfinir une grille avec des *media queries*

Notre disposition fait désormais partie de notre feuille de style CSS. On peut donc l'adapter très facilement pour différentes résolutions. On peut redéfinir la position des objets sur la grille ou la grille elle-même, ou les deux simultanément.

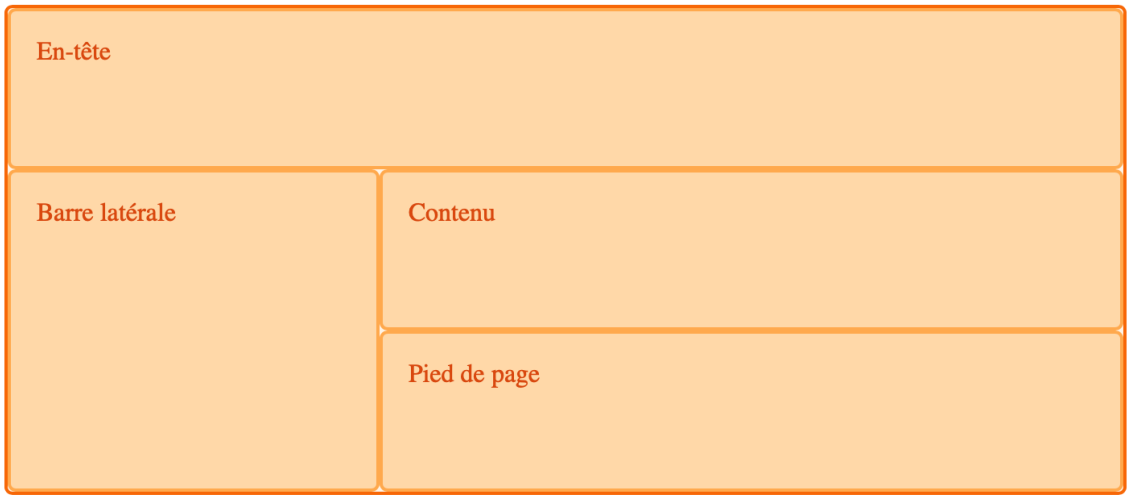
Pour ce faire, on définit les noms des zones en dehors de toute *media query* afin de pouvoir y accéder quel que soit l'endroit où la zone sera placée.

Pour la disposition vue précédemment, on définit ici une disposition par défaut sur une seule colonne pour les affichages étroits. On a donc une seule piste sur laquelle s'empilent les objets :

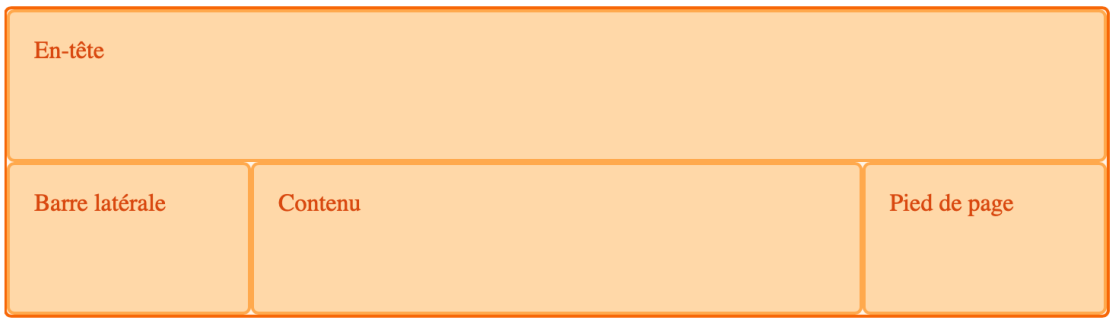
```
.wrapper {
  display: grid;
  grid-auto-rows: minmax(100px, auto);
  grid-template-columns: 1fr;
  grid-template-areas:
    "hd"
    "main"
    "sd"
    "ft";
}
```

On peut ensuite redéfinir la disposition à l'intérieur des différentes *media queries* utilisées pour avoir une disposition sur deux colonnes, voire trois lorsque l'espace le permet. On notera que pour la disposition la plus large, on a une grille organisée sur 9 colonnes/pistes et on redéfinit l'emplacement des objets avec `grid-template-areas`.

```
@media (min-width: 500px) {
  .wrapper {
    grid-template-columns: repeat(9, 1fr);
    grid-template-areas:
      "hd hd hd hd   hd   hd   hd   hd   hd"
      "sd sd sd main main main main main main"
      "sd sd sd  ft  ft   ft   ft   ft   ft";
  }
}
```

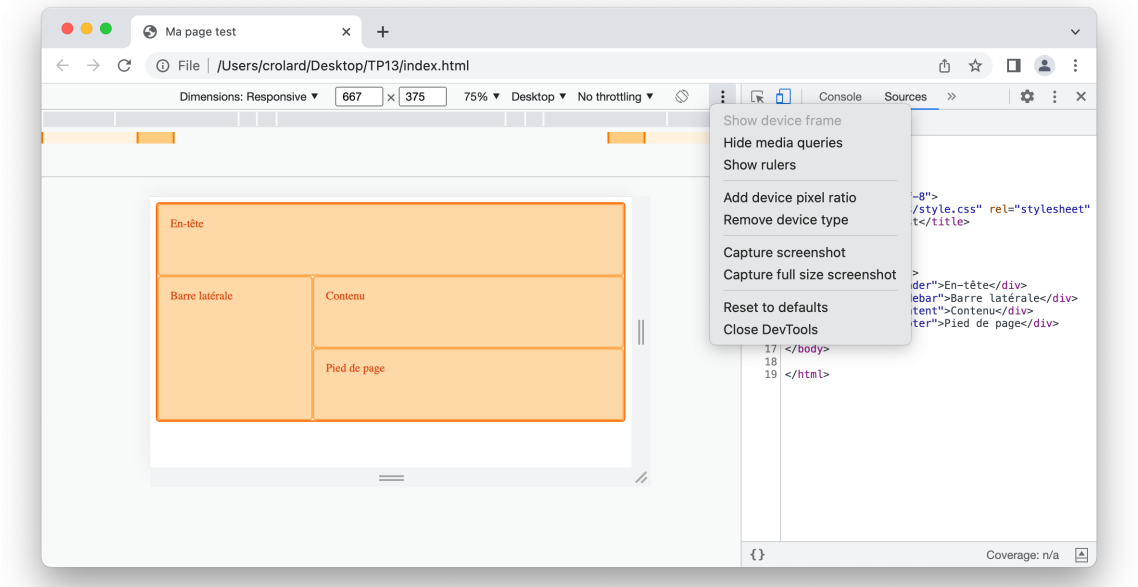



```
@media (min-width: 700px) {
  .wrapper {
    grid-template-areas:
      "hd hd hd   hd   hd   hd   hd   hd hd"
      "sd sd main main main main main ft ft";
  }
}
```


En-tête
Barre latérale
Contenu
Pied de page

Annexe

Pour tester les points de rupture avec Google Chrome, activez le mode responsive (icon en bleu sur la copie d'écran, qui représente un mobile et une tablette), puis affichez le type de périphérique (« Add device type ») dans le menu qui s'affiche en cliquant sur les trois points.



Vous devez alors choisir « Desktop » comme type de périphérique, puis tester en changeant la taille ou l'orientation de l'écran qui est représenté (il est préférable de fixer le niveau de zoom aussi, comme ci-dessous à 75% par exemple).