Le modèle des grilles CSS

Rédigé par: Jean-Luc Colson
Date première rédaction: Janvier 2020

SUIVI DES MODIFICATIONS A LA FICHE			
Série	Date	Page(s) modifiée(s)	Raison

Contents

Introduction aux grilles CSS

Dans cette nouvelle partie, nous allons étudier un dernier modèle de disposition : le modèle des grilles.

Implémenté en 2017, le modèle des grilles est le modèle de disposition le plus récent et également le plus puissant que nous allons pouvoir utiliser en CSS.

La plupart des propriétés qu'on va pouvoir utiliser avec ce modèle vont ressembler aux propriétés des boites flexibles car ces deux modèles possèdent des principes de base communs.

Première définition du modèle des grilles et différences avec le flexbox

Le modèle des grilles est un modèle bidimensionnel, ce qui signifie que c'est un modèle qui va nous permettre de placer nos éléments en fonction de deux axes.

A la différence du modèle des boites flexibles, il n'est plus question ici d'axe principal et d'axe secondaire. Avec le modèle des grilles, les deux axes peuvent être manipulés de la même façon et vont être définis de cette manière :

- Un axe de bloc qu'on pourra également appeler (pour simplifier) axe vertical ou axe des colonnes ;
- Un axe en ligne ou axe horizontal ou encore axe des rangées.

Le modèle des grilles va donc s'avérer encore plus puissant que le modèle des boites flexibles qui était principalement unidimensionnel. Pour illustrer cela, imaginons la situation suivante :

D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-01.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

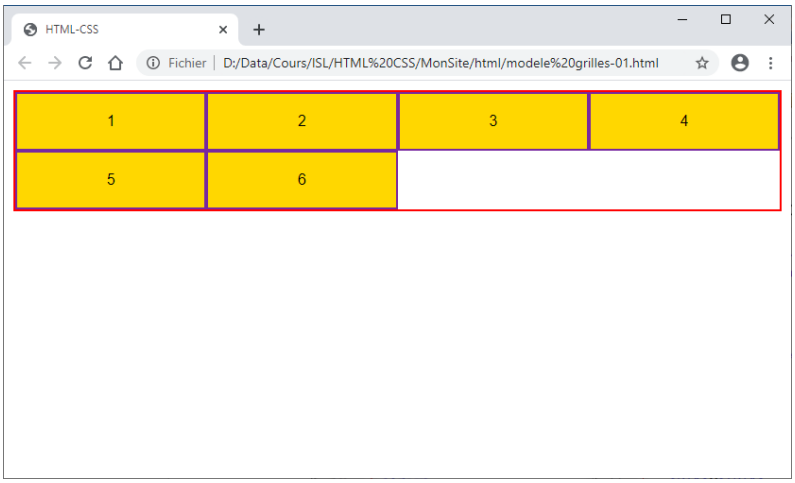
modele grilles-01.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-01.css">
</head>
<body>

<div class="conteneur-flex">
   <div>1</div>
   <div>2</div>
   <div>3</div>
   <div>4</div>
   <div>5</div>
   <div>6</div>
</div>

</body>
</html>
```

modele grilles-01.css

```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}

.conteneur-flex{
  display: flex;
  flex-flow: row wrap;
  margin: 10px;
  border: 2px solid red;
}
.conteneur-flex > div{
  flex: 0 0 25%;
  padding: 20px 0px;
  background-color: gold;
  border: 2px solid RGB(120,40,160);
}
```

Hyper Text M length : 322 lines : 21 Ln : 21 Col : 8 Sel : 0 | 0 Windows (CR LF) UTF-8 INS



Ici, j'ai défini un conteneur flexible dont l'axe principal est l'axe horizontal et qui possède 6 éléments flexibles. On définit une taille de base de 25% pour les éléments flexibles et on ne leur laisse pas la possibilité de s'agrandir ou dé rétrécir. En revanche, on donne le droit aux éléments d'aller à la ligne si nécessaire.

Nos éléments vont donc se placer les uns à côté des autres en partant du début du conteneur puis passer à la ligne. Avec les boites flexibles, il faut bien comprendre que chaque ligne agit comme un conteneur flexible indépendant.

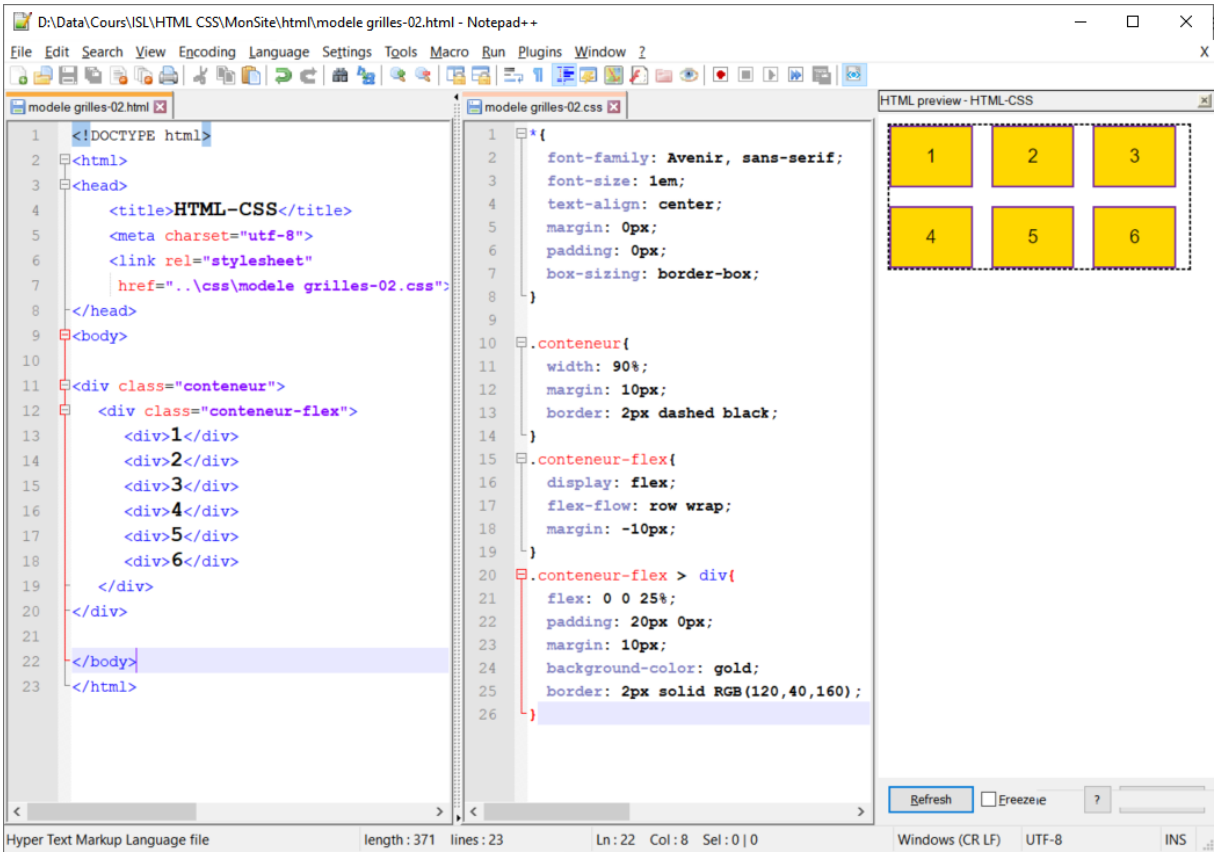
Maintenant, imaginons que je souhaite aligner mon sixième élément flexible dans la ligne. Cela va être impossible avec le flexbox puisqu'il n'existe pas de propriété comme justify-self dans ce modèle.

Les grilles ne vont pas posséder cette limitation et nous permettre au contraire de définir l'alignement de chaque élément de grille selon l'axe de bloc et l'axe en ligne.

Autre limitation du flexbox par rapport aux grilles : le modèle des boites flexibles ne possède pas au jour d'aujourd'hui de propriété nous permettant de définir la taille des gouttières d'un élément.

Une gouttière est l'équivalent d'une marge mais uniquement entre deux éléments flexibles ou de grille et non pas entre un élément et son conteneur.

Avec le flexbox, on était donc obligé d'utiliser la propriété margin pour espacer les éléments flexibles les uns des autres puis d'utiliser des marges négatives sur le conteneur pour supprimer les marges créées entre le conteneur et les éléments.



Non seulement cette solution n'est pas optimale d'un point de vue propreté du code mais en plus le fait de rajouter des marges aux éléments flexibles risque de poser des problèmes dans la disposition des éléments puisque les marges externes vont venir s'ajouter à la taille des éléments.

Les grilles possèdent elles un set de propriétés qui va nous permettre de définir les gouttières des éléments d'une bien meilleure façon.

Vous pouvez donc retenir l'idée suivante pour définir s'il est préférable d'utiliser le flexbox ou les grilles : si vous avez besoin de contrôler la disposition des éléments selon les deux axes ou si vous avez besoin d'espacer précisément les différents éléments, alors utilisez les grilles. Sinon, utilisez le flexbox.

Notez par ailleurs que nous allons tout à fait pouvoir utiliser ces deux modèles conjointement et créer des dispositions de page complexes, un créant des éléments de grille qui vont contenir des éléments flexibles par exemple.

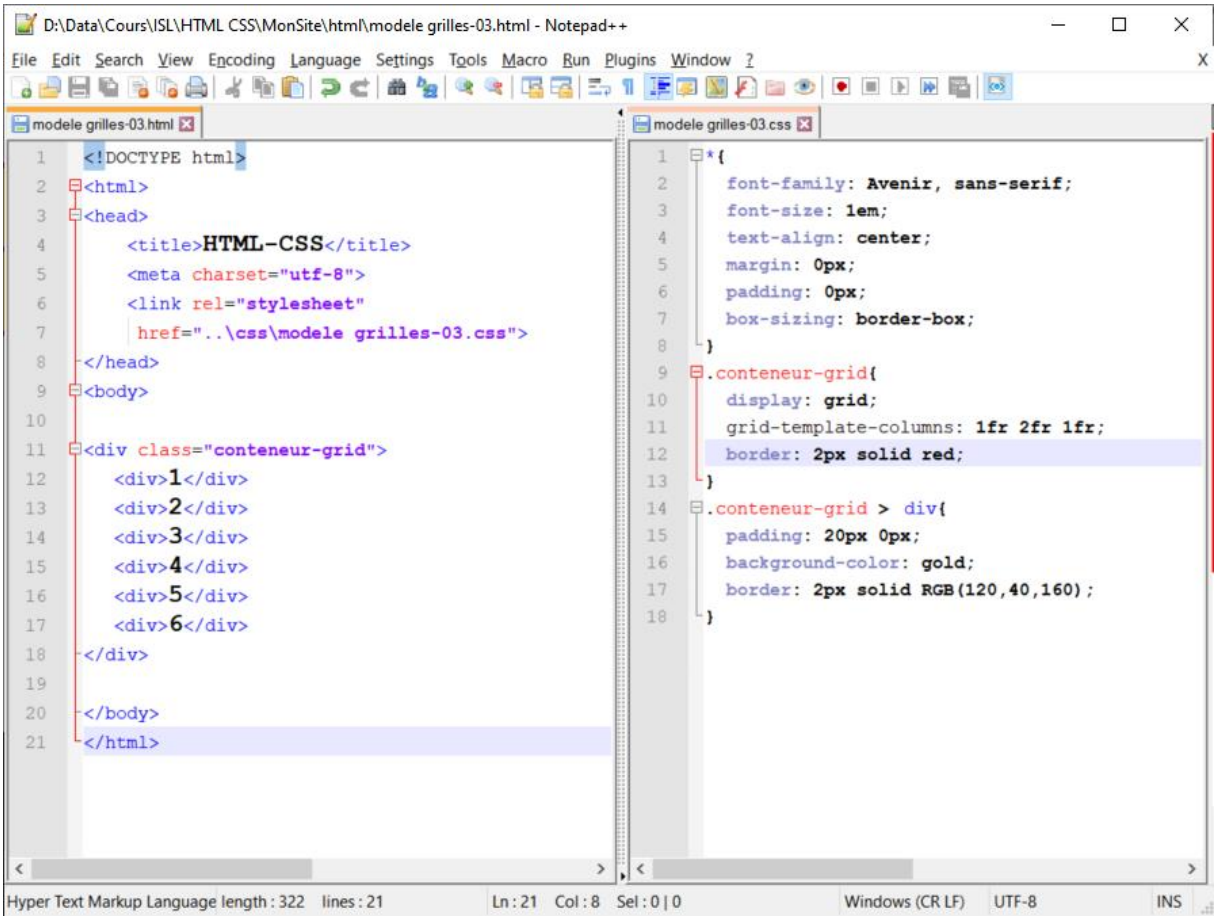
Élément grille conteneur et éléments de grille

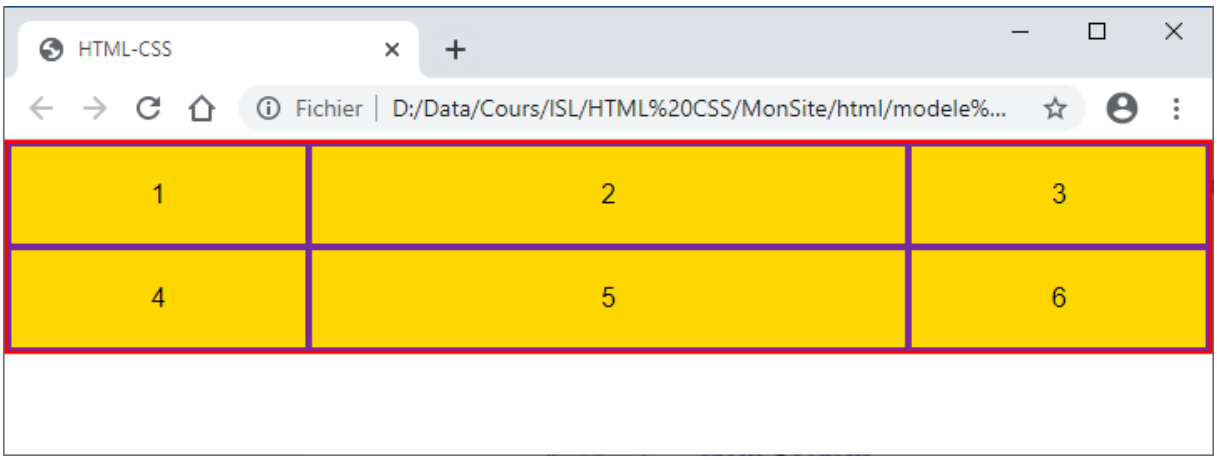
Pour définir une grille, nous allons devoir appliquer un display : grid (la grille sera de type block) ou un display : inline-grid (la grille sera de niveau inline) à un élément.

L'élément auquel on applique un display : grid ou display : inline- grid va automatiquement devenir un élément grille conteneur.

De manière similaire au modèle des boites flexibles, tous les enfants directs de notre élément grille conteneur (et seulement les enfants directs) vont automatiquement être des éléments de grille.

Voici ci-dessous une première grille :





Ici, nous définissons un conteneur de grille avec display : grid. Les enfants directs du conteneur vont ainsi automatiquement devenir des éléments de grille. Ensuite, je définis les colonnes de ma grille avec grid-template-columns. N'essayez pas de tout comprendre tout de suite, nous aurons l'occasion d'étudier cette propriété par la suite.

Le vocabulaire des grilles

Les grilles sont des structures relativement complexes et il est donc essentiel de définir précisément les différentes parties d'une grille que nous allons être amenés à manipuler ou qui vont pouvoir nous être utiles.

Le premier concept à comprendre est qu'une grille est constituée de lignes ou de « droites » horizontales et verticales. Ces lignes ne sont pas visibles à l'écran et il faut donc se les imaginer.

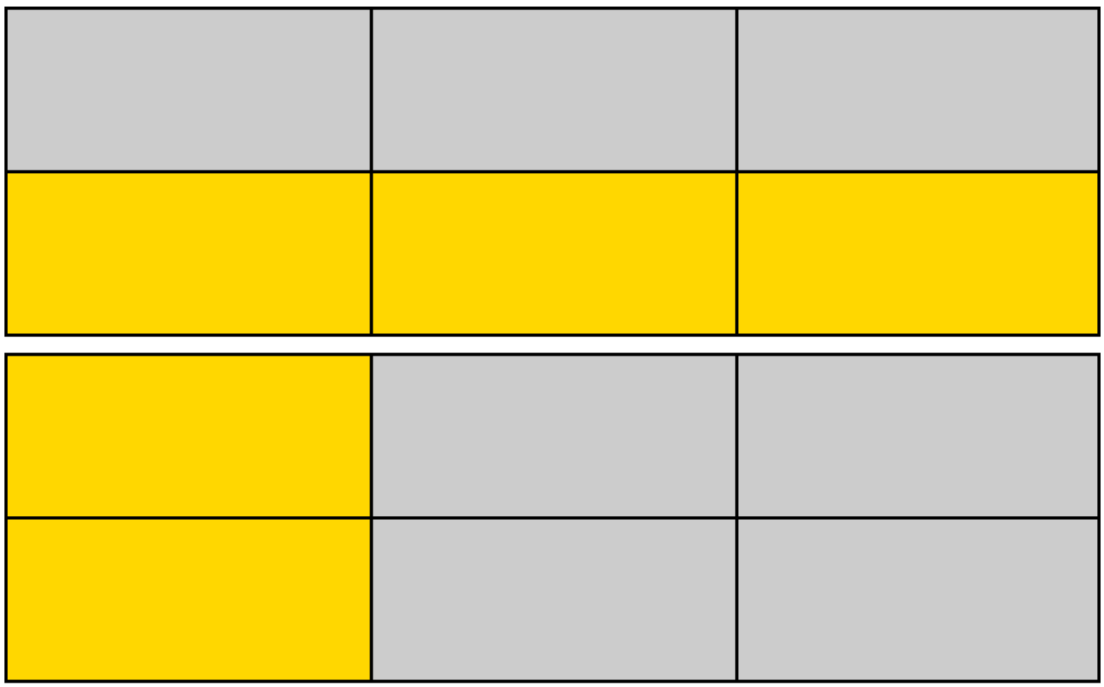
Ces lignes vont être disposées de chaque côté des colonnes et des rangées d'une grille.

Note : Pour les grilles, nous parlerons de « rangées » pour définir les « rows » en anglais et non pas de lignes afin de ne pas les confondre avec les lignes que nous venons de définir ci-dessus.

Une grille possédant 3 colonnes et 2 rangées va donc posséder 4 lignes verticales et 3 lignes horizontales, une grille possédant 4 colonnes et 4 rangées va posséder 5 lignes verticales et 5 lignes horizontales et etc.

Ci-dessous, vous pouvez retrouver une représentation visuelle des lignes d'une grille. Ma grille contient ici 3 colonnes et 2 rangées. Les lignes verticales ont été dessinées en bleu et les lignes horizontales ont été dessinées en rouge.

L'espace entre deux lignes adjacentes est ce qu'on appelle une piste. Le terme piste sert donc tout simplement à désigner indifféremment une colonne ou une rangée dans une grille.

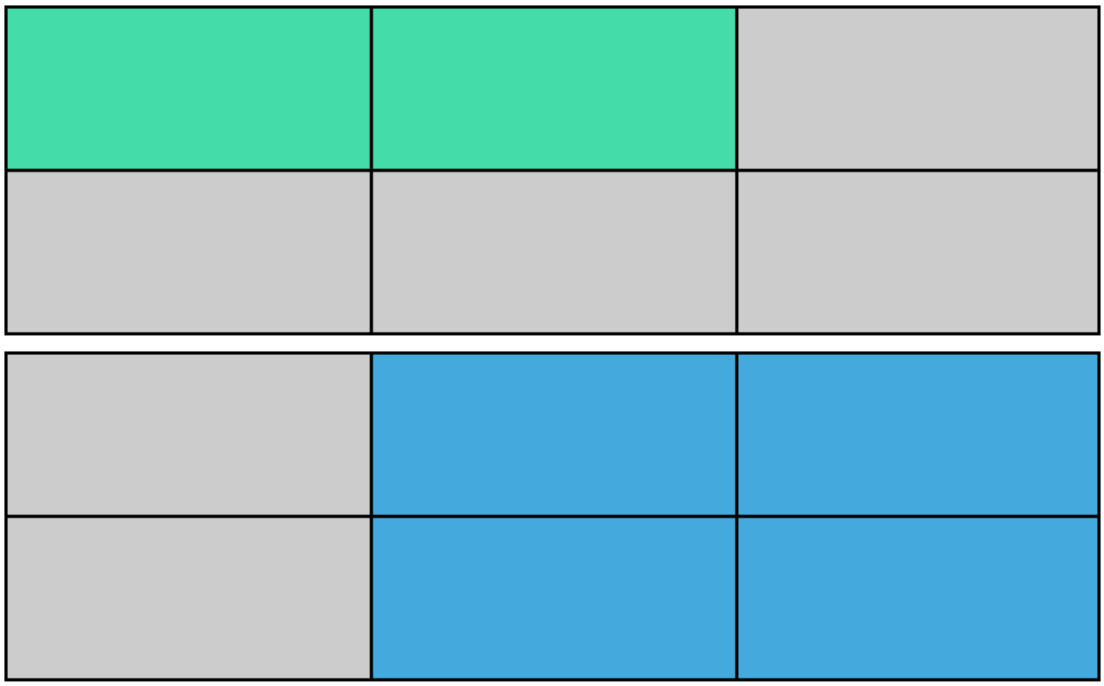


Ci-dessus, j'ai créé deux grilles composées de 3 colonnes et de 2 rangées. En jaune, j'ai colorié une piste pour chacune des deux grilles.

Une piste est composée de cellules. Tout comme pour les tableaux, une cellule correspond visuellement à l'espace délimité par deux lignes de colonnes et deux lignes de rangées adjacentes.

Ci-dessus, j'ai colorié 3 cellules de ma grille en jaune, vert et bleu.

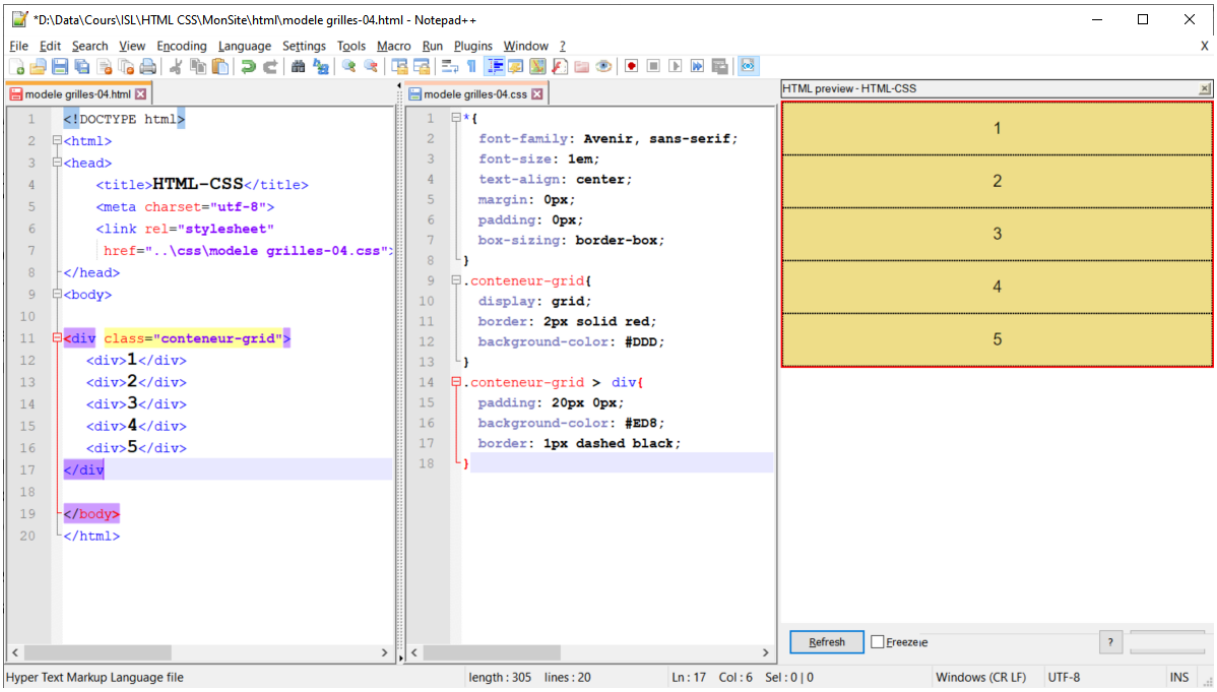
Finalement, une zone de grille correspond à l'espace délimité par deux lignes de colonnes et deux lignes de rangées qui ne sont pas nécessairement adjacentes. Ci-dessous, j'ai dessiné une zone dans chacune de mes deux grilles (la première couvre 2 cellules et la deuxième couvre 4 cellules).



Créer une grille et définir des pistes de grille

Dans la leçon précédente, nous avons vu qu'il suffisait d'appliquer un display : grid à un élément pour le transformer en un élément grille conteneur et ainsi définir notre grille.

Notre grille ainsi créée ne va par défaut être composée que d'une seule colonne et d'autant de rangées qu'il y a d'éléments de grille.



Ici, on dit que les pistes sont définies de manière implicite (car elles sont créées par la grille elle-même). Nous allons cependant également pouvoir définir les pistes de nos grilles nous-mêmes. Dans ce cas-là, on dira que les pistes ont été définies explicitement.

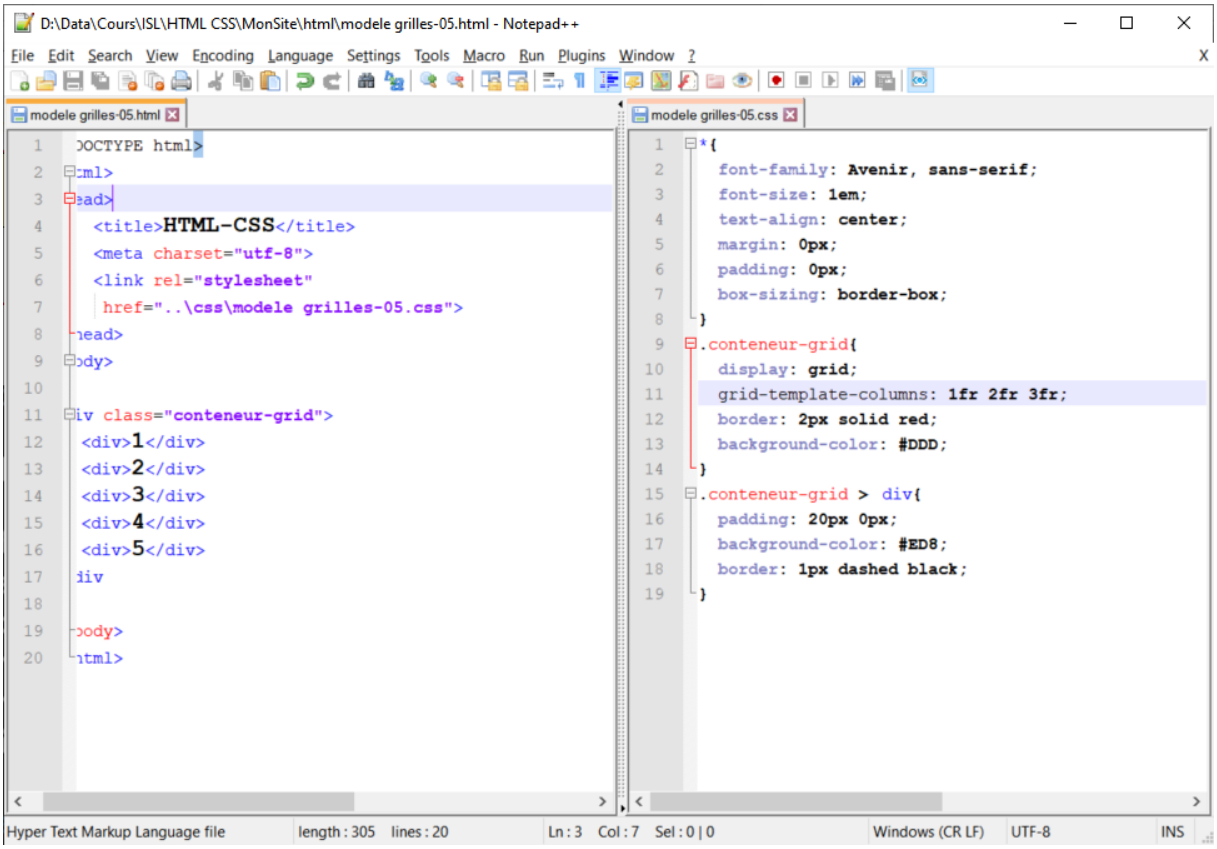
Nous allons apprendre dans cette leçon à définir explicitement le nombre et la taille des pistes de nos grilles ainsi qu'à maitriser le comportement des pistes définies implicitement.

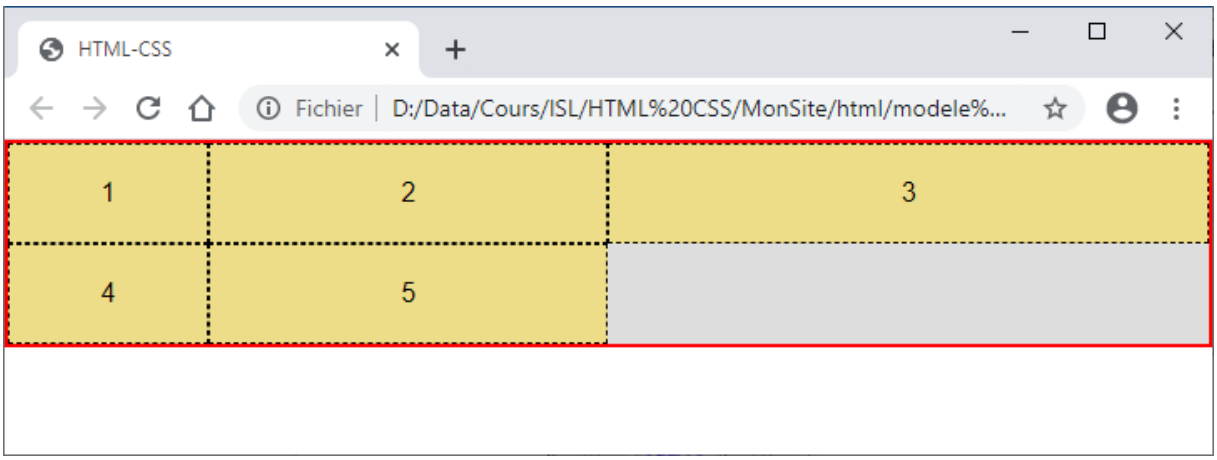
Définir explicitement les colonnes et les rangées d'une grille

Pour définir explicitement le nombre et les dimensions de colonnes et de rangées de notre grille, nous allons pouvoir utiliser les propriétés grid-template-columns et grid-template-rows.

Ces propriétés vont pouvoir accepter n'importe quelle valeur de type dimension ainsi qu'un type de valeurs spécifique aux grilles : le fr qui représente une fraction de l'espace disponible dans le conteneur et qui va donc nous permettre de définir la taille d'une piste en fonction des autres.

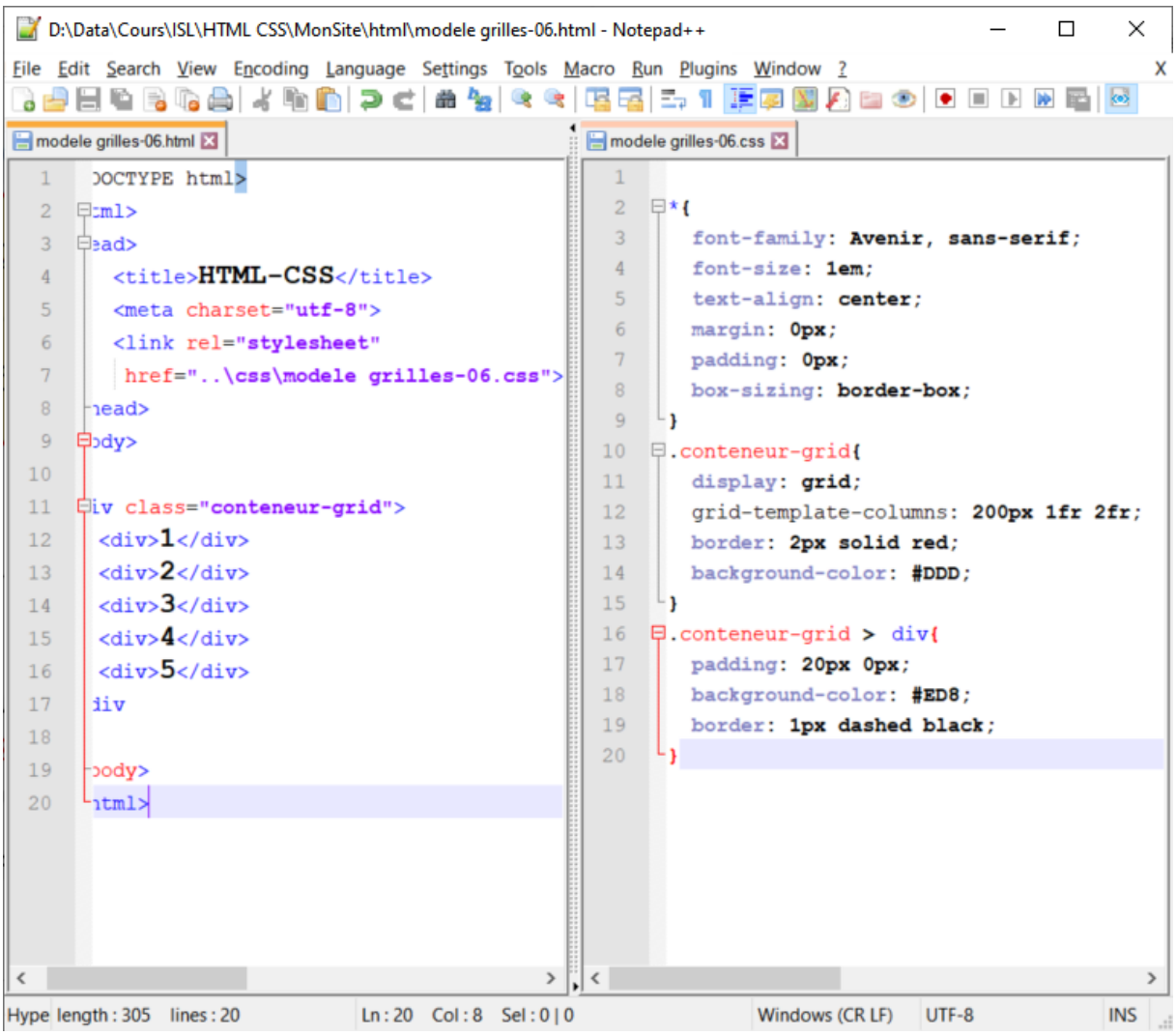
Le nombre de valeurs passées à grid-template-columns et à grid-template-rows va déterminer le nombre de colonnes et de rangées de notre grille.

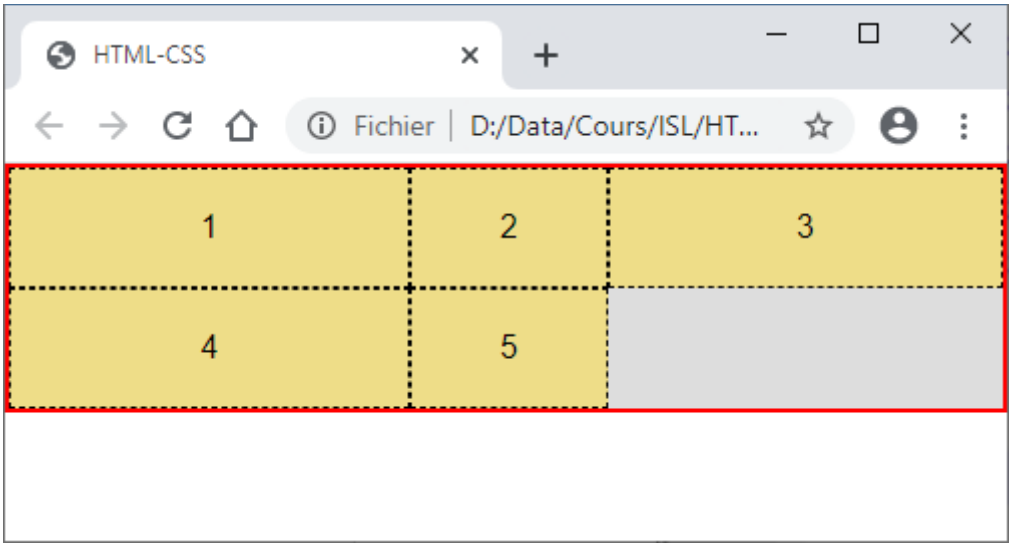




Ci-dessus, nous avons une grille qui contient 5 éléments. Nous créons trois colonnes dans notre grille avec grid-template-columns : 1fr 2fr 3fr. Dans ce cas, la deuxième colonne occupera deux fois plus d'espace que la première tandis que la troisième occupera trois fois plus d'espace que la première.

Notez qu'on va également tout à fait pouvoir définir nos pistes en utilisant un mélange de différents types d'unités.

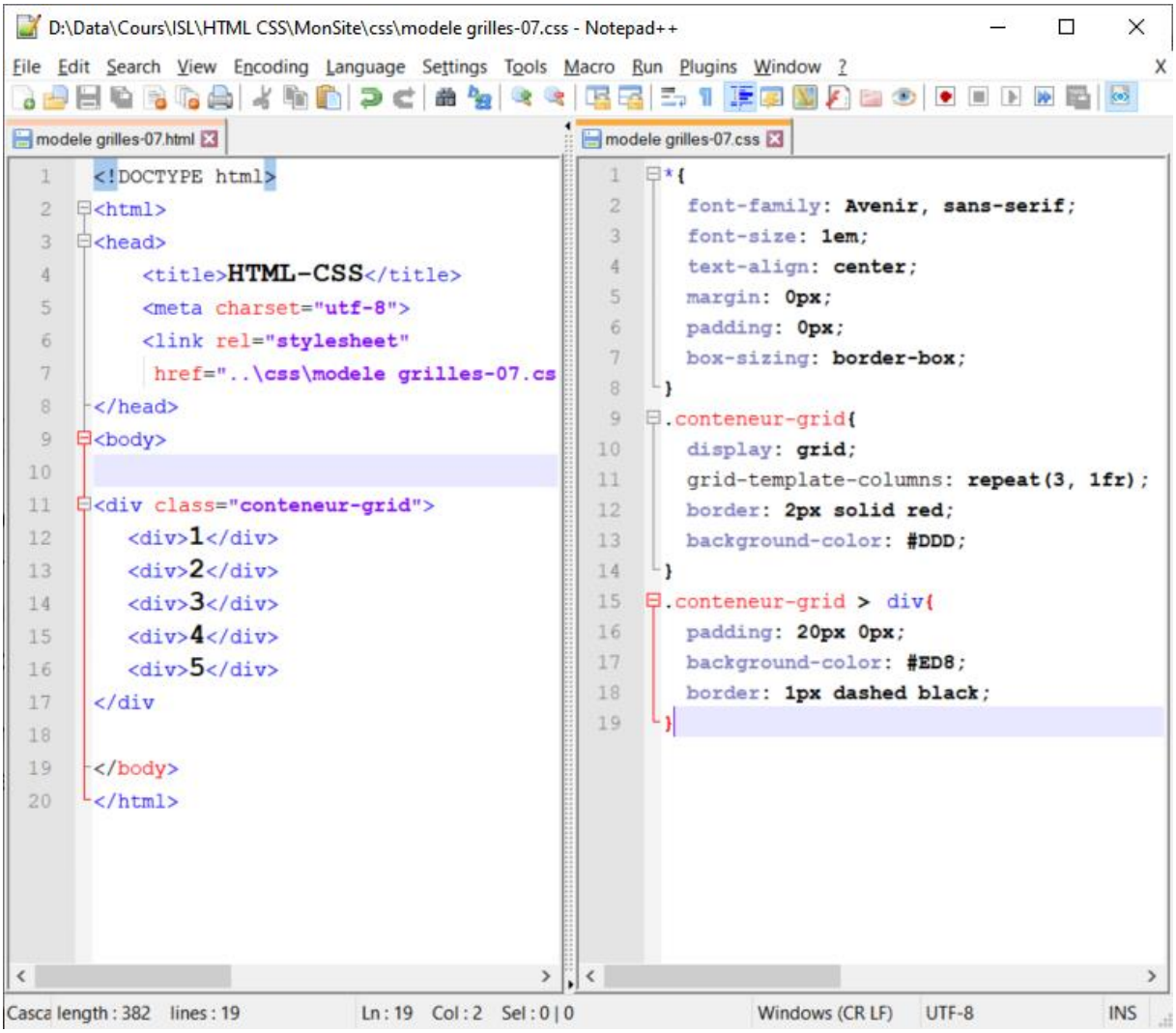




Dans cet exemple, on crée à nouveau une grille à trois colonnes. Cette fois-ci, on demande à ce que notre première colonne occupe un espace de 200px en largeur. L'espace restant dans le conteneur

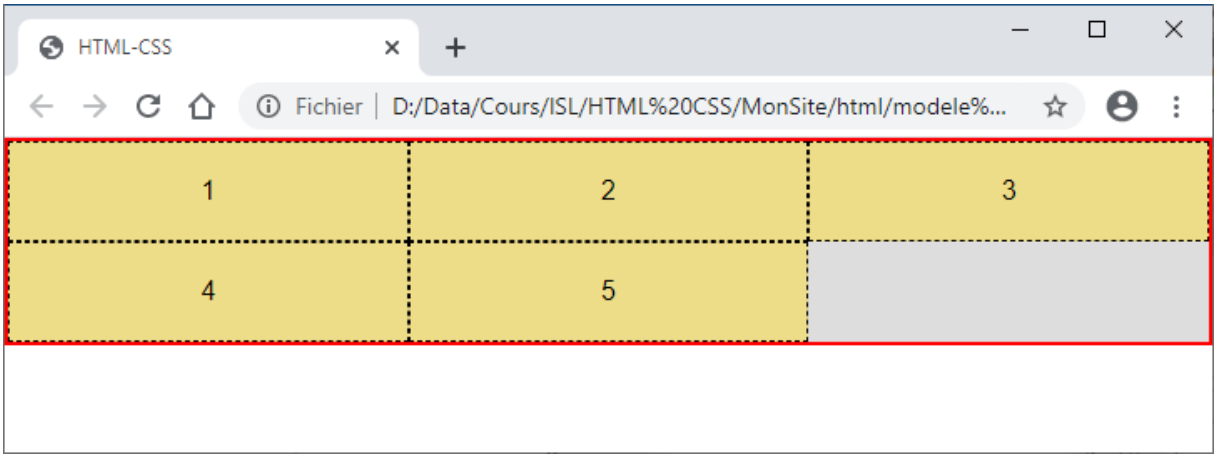
sera ensuite partagé entre les deux autres colonnes : 1/3 de l'espace restant pour la deuxième colonne et les deux autres tiers pour la dernière colonne.

Nous allons encore pouvoir passer une fonction repeat() à grid-template-columns et à grid-template-rows pour créer rapidement plusieurs pistes aux caractéristiques similaires. Cette fonction va accepter deux valeurs : le nombre de pistes à créer ainsi que la taille de chaque piste.

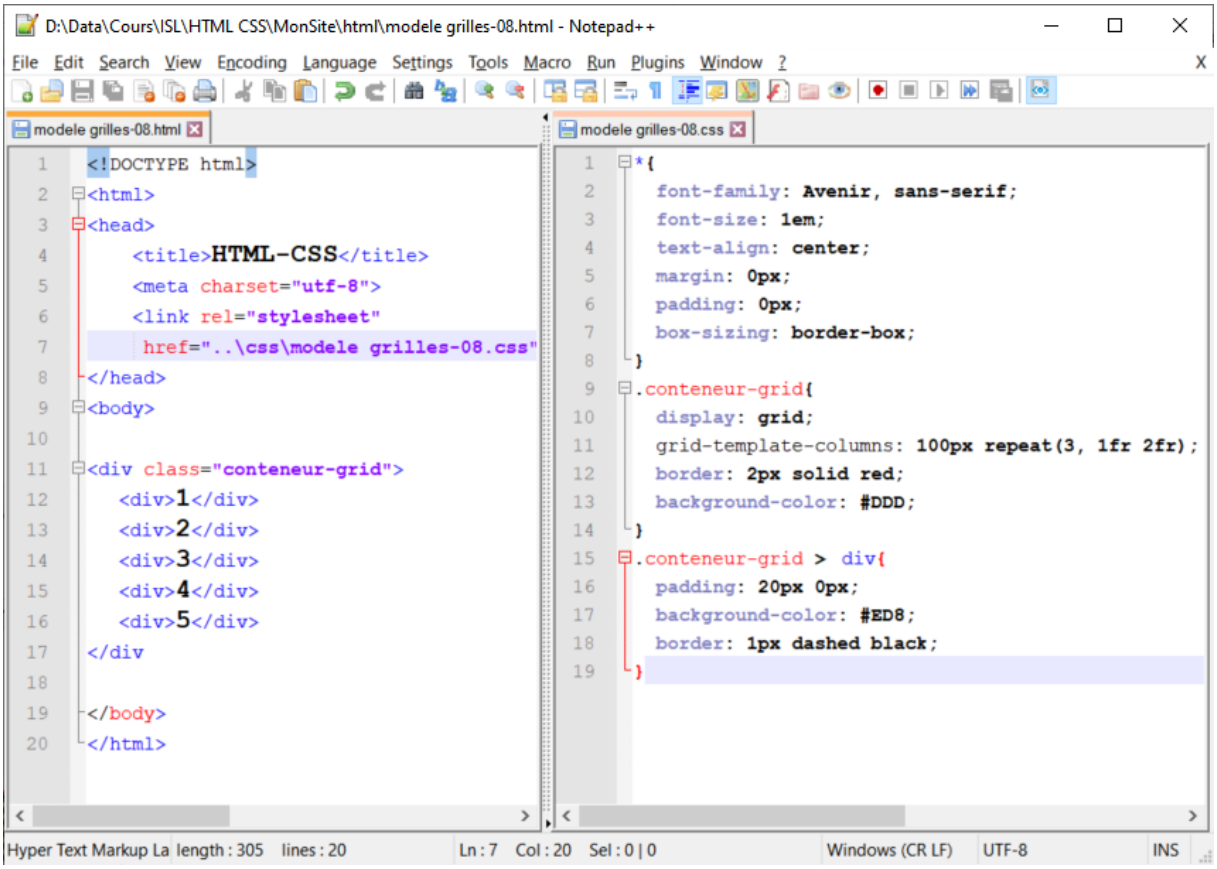


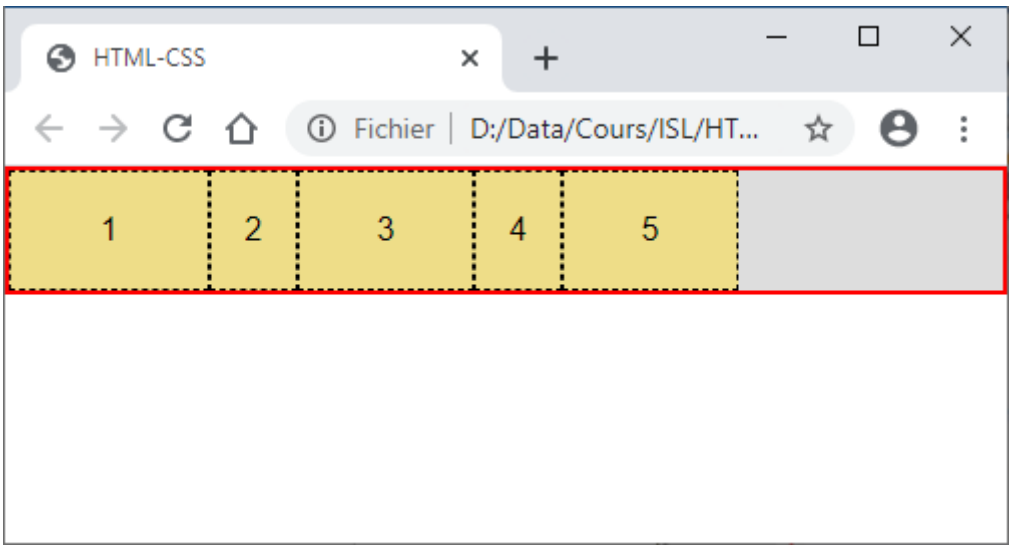

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-07.cs
</head>
<body>

<div class="conteneur-grid">
   <div>1</div>
   <div>2</div>
   <div>3</div>
   <div>4</div>
   <div>5</div>
</div

</body>
</html>
```

```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  border: 2px solid red;
  background-color: #DDD;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
```



Ici, on définit à nouveau trois colonnes dans notre grille. Chaque colonne occupera la même place. Notez qu'on va pouvoir utiliser la fonction repeat() pour définir des motifs. Notez également qu'on va pouvoir définir certaines pistes avec repeat() et certaines pistes individuellement.





Ici, on crée une première colonne qui va occuper une place de 100px en largeur puis on définit un motif de 2 colonnes à répéter 3 fois avec repeat(). A chaque fois, la première colonne aura une taille de 1fr et la seconde une taille de 2fr.

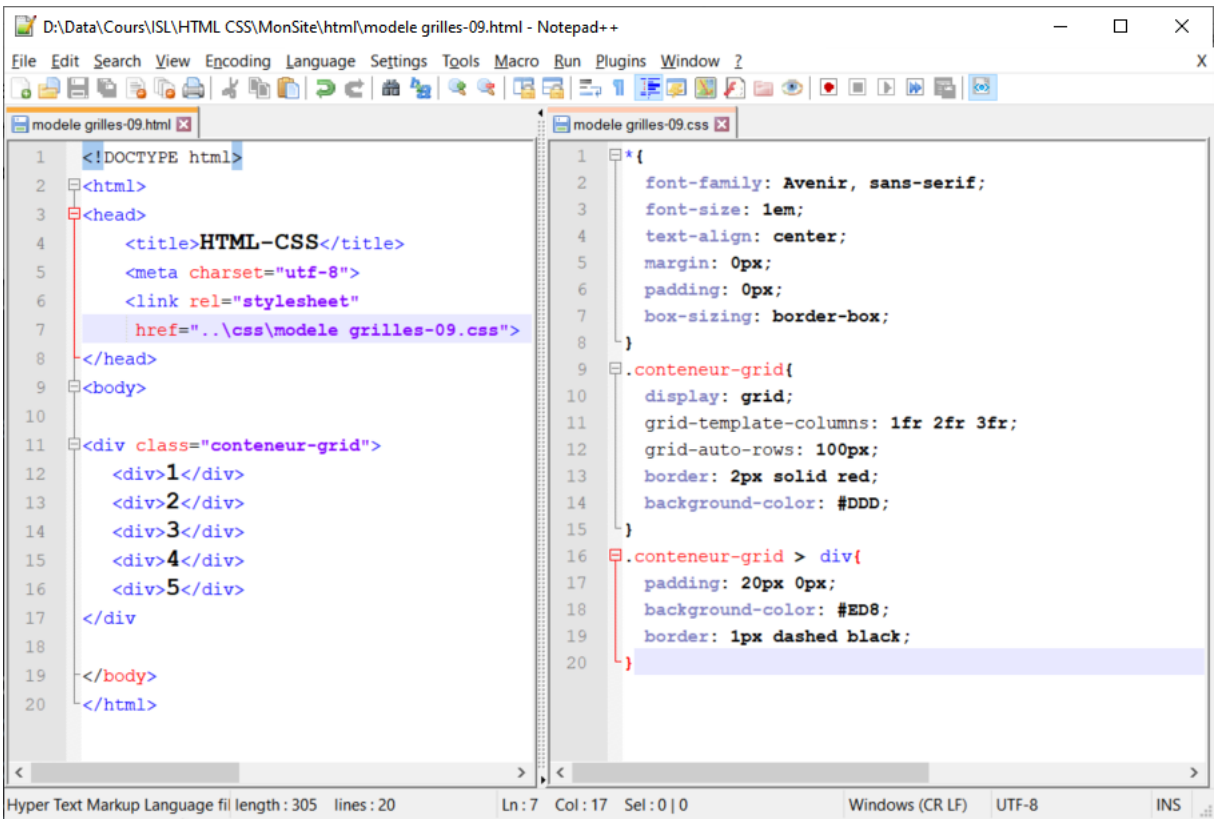
Règles de création et taille des pistes implicites

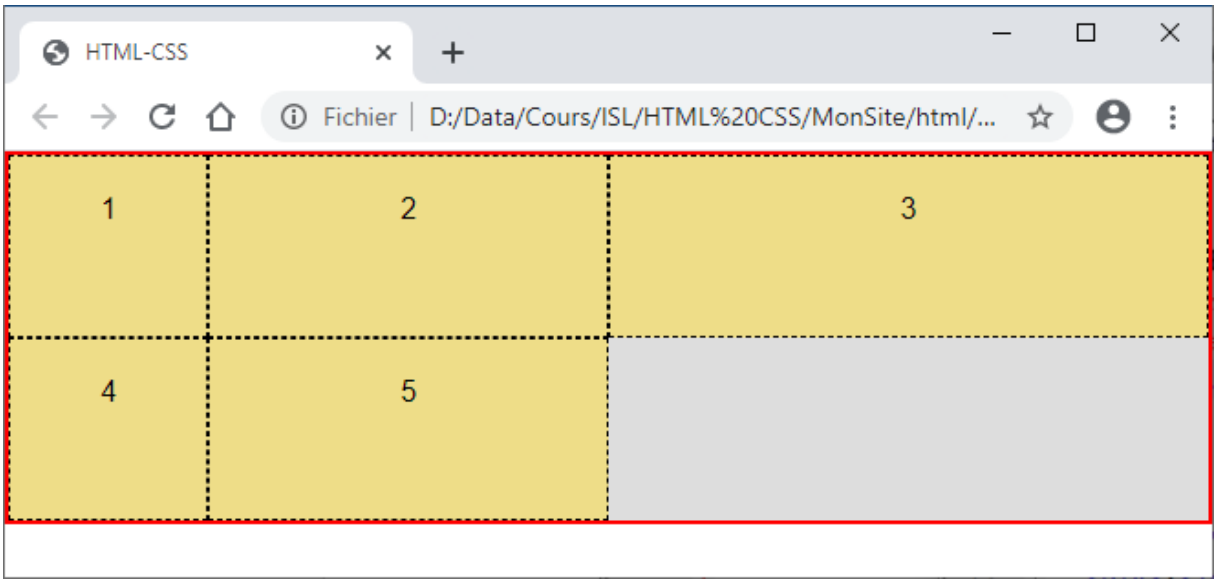
Dans les exemples ci-dessus, nous n'avons à chaque fois défini que les colonnes de notre grille et avons laissé la grille définir implicitement son nombre de rangées.

On dit qu'une piste est définie implicitement dès qu'elle n'a pas été créée avec grid-template-columns ou grid-template-rows.

Une grille va ainsi créer implicitement de nouvelles rangées ou de nouvelles colonnes dès qu'elle va devenir trop petite pour contenir un élément. Cela peut arriver dans deux situations différentes : si on a tenté de placer explicitement un élément en dehors de la grille ou si nos éléments possèdent trop de contenu pour que celui-ci rentre dans la grille.

Par défaut, les dimensions des pistes créées implicitement auront la valeur auto ce qui signifie que les pistes ajusteront leur taille selon leur contenu. On va pouvoir définir un autre comportement et une taille pour les pistes créées implicitement grâce aux propriétés grid-auto-rows et grid-auto-columns.





Dans l'exemple ci-dessus, on crée une grille et on définit explicitement 3 colonnes avec grid-template-columns : 1fr 2fr 3fr. On ne précise pas de propriété grid-template-rows : les rangées vont donc être créées implicitement.

Nous allons donc renseigner une propriété grid-auto-rows pour maitriser la hauteur des rangées créées implicitement par la grille.

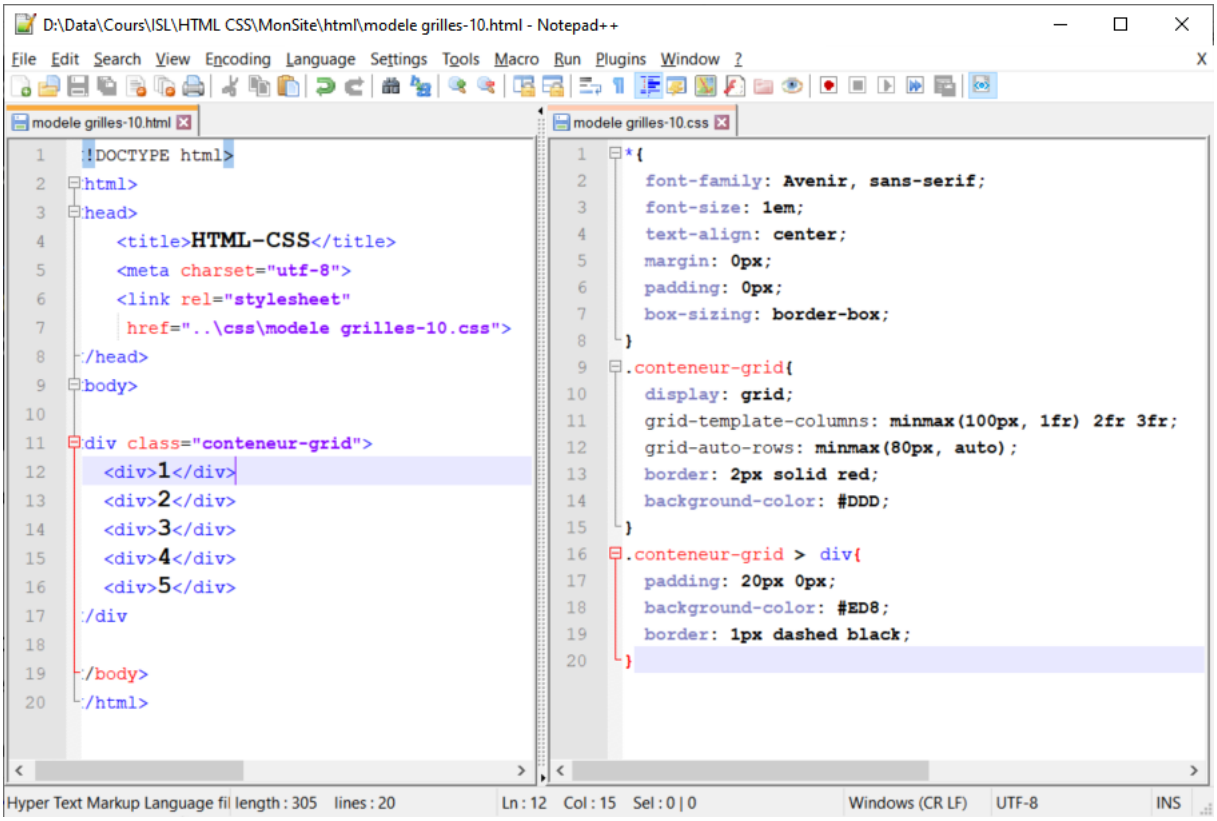
Définir un intervalle de tailles valides pour les pistes d'une grille

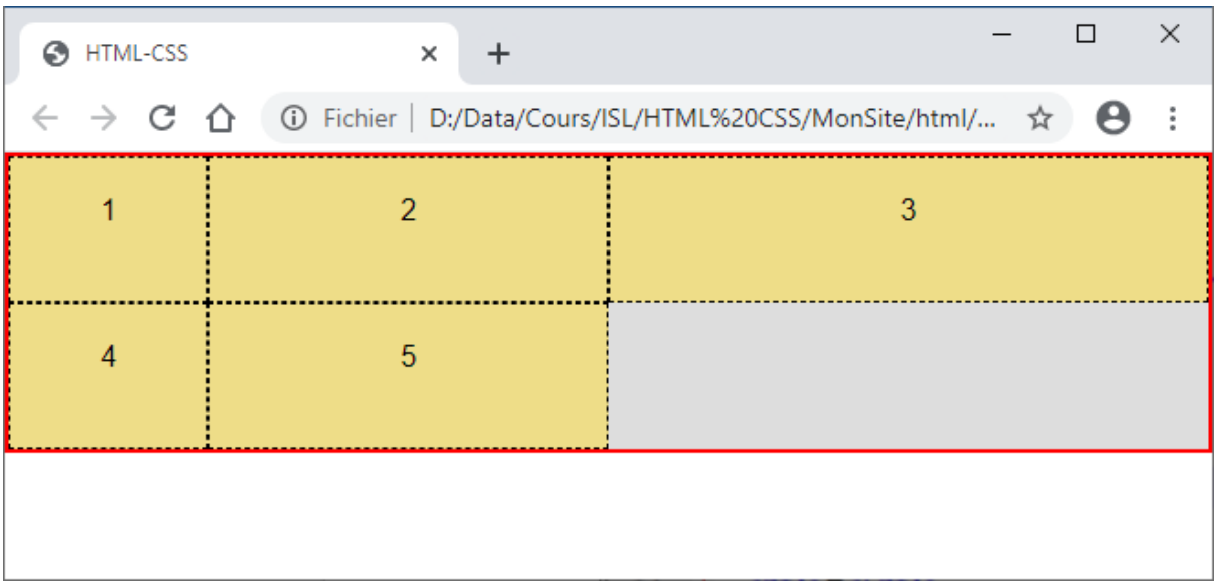
Les propriétés grid-template-columns, grid-template-rows, grid-auto-columns et grid-auto-rows vont également pouvoir accepter une fonction minmax() en valeur pour une ou plusieurs pistes.

La fonction minmax() va s'avérer très intéressante puisqu'elle va nous permettre de définir des bornes, c'est-à-dire un intervalle de dimensions valides pour nos pistes, que celles-ci aient été définies explicitement ou implicitement.

Nous allons passer deux valeurs à cette fonction qui vont correspondre à la borne basse et à la borne haute. Les types de valeurs acceptés par minmax() sont les suivantes :

- Une longueur en px par exemple ;
- Un pourcentage ;
- Une unité de fraction fr ;
- Le mot clef max-content qui va représenter la taille idéale de la piste c'est-à-dire la plus petite taille permettant d'afficher tout le contenu sur une ligne ;
- Le mot clef min-content qui va représenter la plus petite taille que le piste peut avoir sans que son contenu ne déborde (avec un contenu éventuellement sur plusieurs lignes) ;
- Le mot clef auto qui va laisser la piste s'adapter en fonction de son contenu et par rapport à son autre borne.





Dans l'exemple précédent, on utilise grid-auto-rows : minmax(80px, auto) pour indiquer que les rangées créées implicitement ne peuvent pas faire moins de 80px de hauteur et doivent s'adapter à leur contenu si celui-ci est plus grand que la taille minimale.

On va également utiliser minmax() en valeur de notre propriété grid-template-columns pour indiquer que notre première colonne doit occuper une largeur minimum de 100px et maximum de 1fr.

Ici, dans le cas où la borne haute (1fr) s'avère plus petite que la borne basse (100px), alors elle sera ignorée et minmax() ne servira qu'à définir une taille minimale.

Positionner des éléments dans une grille

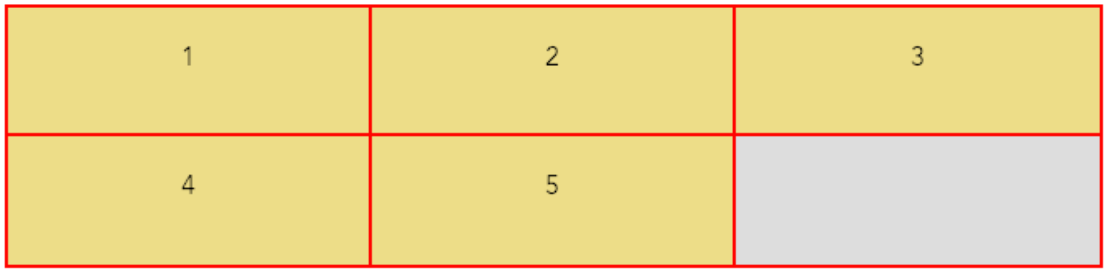
Nous savons désormais comment créer des grilles et comment définir précisément les colonnes et les rangées dont elles se composent.

Dans cette leçon, nous allons apprendre à positionner les éléments de grille dans une grille.

Les fondamentaux du positionnement des éléments de grille

Les éléments de grille vont être positionnés à partir des lignes invisibles créées automatiquement par la grille. Pour rappel, ces lignes vont se trouver de chaque côté des différentes pistes d'une grille. Une grille possédant 3 colonnes et 2 rangées va ainsi disposer de 4 lignes verticales et de 3 lignes horizontales.

Dans l'illustration ci-dessous, j'ai tracé en rouge les différentes lignes d'une telle grille qui possède ici 5 éléments :



Ces lignes vont par défaut être numérotées dans chacun des deux axes. Elles vont donc posséder un ordre sur lequel on va pouvoir se baser pour positionner nos éléments de grille.

La numérotation va dépendre du sens de l'écriture du document. Pour un langage qui se lit de gauche à droite, par exemple, la ligne tout à gauche sera la ligne n°1, celle à sa droite sera la ligne n°2 et etc. dans l'axe de bloc. De même, la ligne tout en haut sera la n°1, celle en dessous la n°2 et etc. dans l'axe en ligne.

Pour positionner les éléments de grille dans la grille, nous allons pouvoir indiquer 4 lignes : de lignes de départ (horizontal / vertical) à partir desquelles l'élément doit être placé et deux lignes d'arrivée (horizontal / vertical) où l'élément doit finir.

Nous allons donc indiquer une surface qu'un élément doit couvrir et, en même temps qu'on définit son positionnement, définir de facto la taille de l'élément. Cette façon de faire est différente de ce qu'on a pu voir jusqu'à présent et va imposer aux éléments de grille de couvrir toujours un nombre entier de cellules.

Ce que vous devez bien comprendre ici est que la définition des tailles / dimensions va se faire au moment où on définit les pistes de notre grille. Pour cette raison, il faudra bien définir au départ si on veut créer une grille à 3, 6, 12, etc. colonnes et également bien définir le comportement des rangées.

Ensuite, nous n'allons faire que placer des éléments dans la grille. Fonctionner comme cela peut sembler contraignant mais c'est au final une excellente méthode pour organiser ses éléments et créer des designs complexes puisque toutes les contraintes se font sur la grille et cela va nous éviter d'avoir à envisager toutes les situations de comportement non voulu des différents éléments.

Positionner des éléments de grille en pratique

Par défaut, les éléments de grille vont se positionner à la suite les uns des autres selon le sens de l'écriture du document. Chaque élément va occuper une cellule, c'est-à-dire ne va s'étendre que sur une rangée et sur une colonne.

Pour positionner manuellement (ou explicitement) les éléments dans la grille, nous allons pouvoir utiliser les propriétés suivantes :

- grid-column-start : Indique la ligne de départ de l'élément selon l'axe de bloc (ligne verticale);
- grid-column-end : Indique la ligne de fin de l'élément selon l'axe de bloc (ligne verticale) ;
- grid-row-start : Indique la ligne de départ de l'élément selon l'axe en ligne (ligne horizontale);
- grid-row-end : Indique la ligne de fin de l'élément selon l'axe en ligne (ligne horizontale) ;
- grid-area : Indique une zone de grille dans laquelle l'élément doit de placer.

Nous allons également pouvoir utiliser les versions raccourcies grid-column, grid-row.

Positionner les éléments de grille en utilisant les lignes des grilles

Utiliser la numérotation des lignes pour placer les éléments

Nous allons déjà pouvoir passer un chiffre aux propriétés grid-column-start, grid-column-end, grid-row-start et grid-row-end.

Le chiffre passé va indiquer le numéro de la ligne où l'élément de grille doit commencer (pour les propriétés -start) ou la ligne où il doit s'arrêter (pour les propriétés –end).

Commençons avec un exemple simple d'une grille possédant 5 éléments tous positionnés explicitement :

*D:\Data\Cours\ISL\HTML CSS\MonSite\css\modele grilles-11.css - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

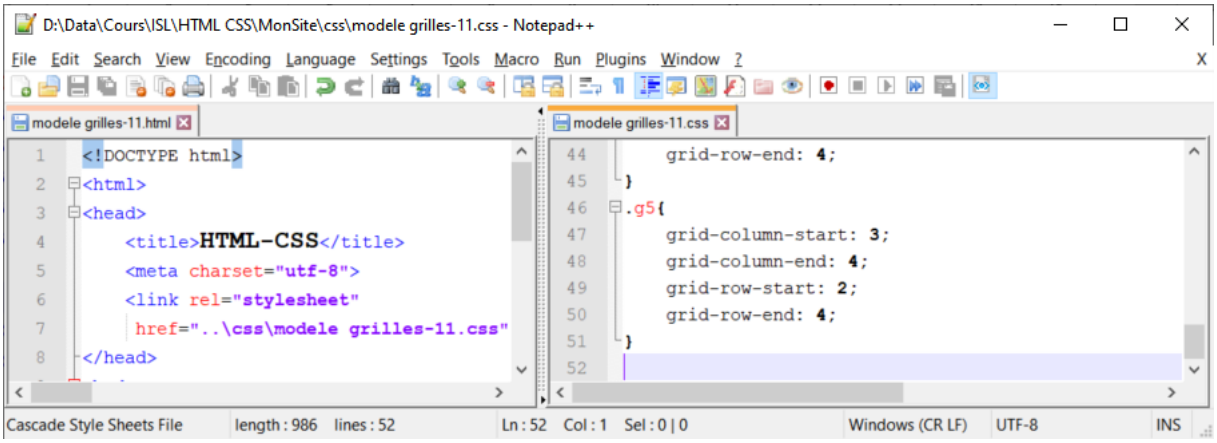
modele grilles-11.html

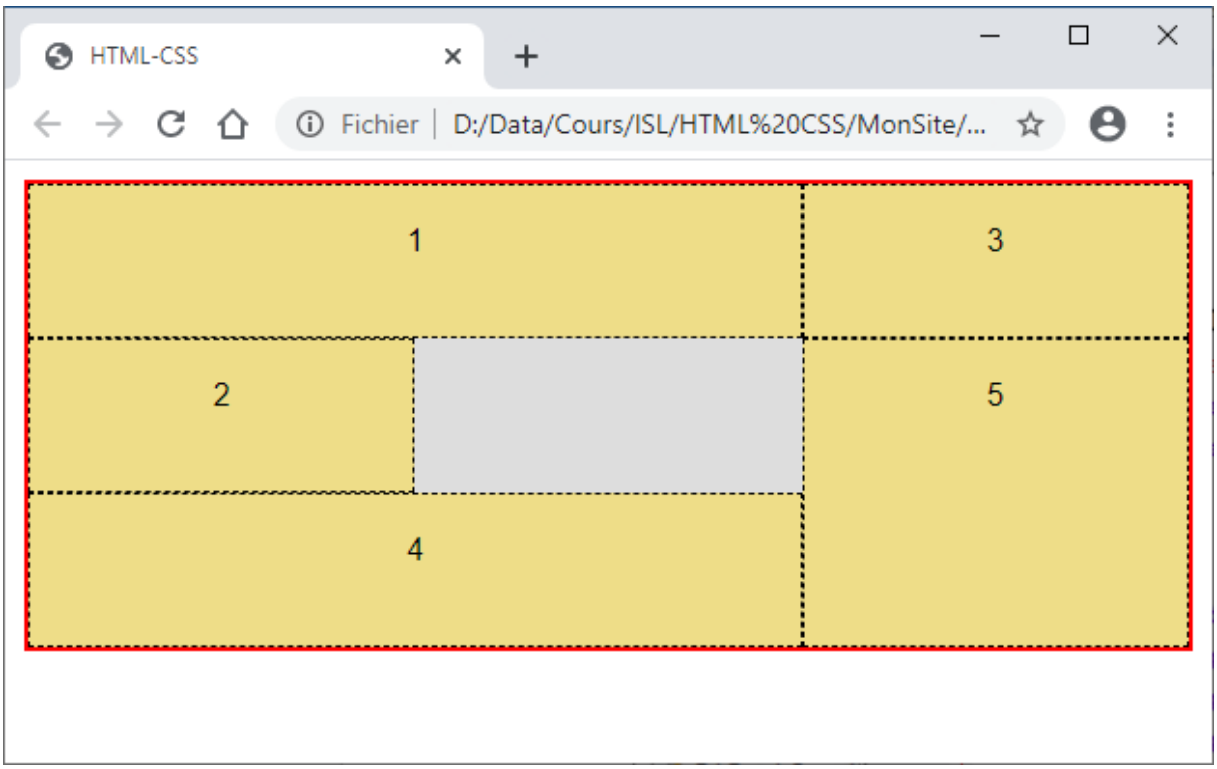
```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-11.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-11.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,minmax(80px,1fr));
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 1;
    grid-row-end: 2;
}
.g2{
    grid-column-start: 1;
    grid-column-end: 2;
    grid-row-start: 2;
    grid-row-end: 3;
}
.g3{
    grid-colmun-start: 3;
    grid-column-end: 4;
    grid-row-start: 1;
    grid-row-end: 2;
}
.g4{
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 3;
    grid-row-end: 4;
}
```

Cascade Style Sheets File | length : 986 lines : 52 | Ln : 52 Col : 1 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS





Ici, on indique que notre premier élément doit de positionner à partir de la première ligne verticale et couvrir l'espace jusqu'à la troisième ligne verticale. L'élément sera également compris entre la première et la deuxième ligne horizontale.

Notre deuxième élément devra se positionner entre la première et la deuxième ligne verticale et entre la deuxième et la troisième ligne horizontale.

On demande à l'élément 3 de se positionner entre la troisième et la quatrième ligne verticale et entre la première et la deuxième ligne horizontale.

Notre quatrième élément de grille va se placer entre la première et la troisième ligne verticale et entre la troisième et la quatrième ligne horizontale.

Finalement, notre cinquième et dernier élément de grille va recouvrir l'espace entre la troisième et la quatrième ligne verticale et entre la deuxième et la quatrième ligne horizontale de la grille.

Notez qu'on ne va pas être obligé de préciser les 4 propriétés pour positionner un élément dans une grille. Pour être tout à fait précis, seule une des deux propriétés grid-column-start et grid-row-start est strictement nécessaire.

Attention cependant : si certaines propriétés sont omises pour certains éléments mais précisées pour d'autres, alors les éléments dont les propriétés ont été précisées viendront se placer avant les autres dans la grille et cela peut décaler les autres éléments.

En omettant l'une des deux propriétés grid-column-start ou grid-row-start, l'élément sera positionné selon sa position par défaut par rapport à l'axe non défini.

En omettant les propriétés grid-column-end et / ou grid-row-end, l'élément suivra son comportement par défaut qui est de n'occuper qu'une piste dans l'axe non défini.

Pour le moment, contentons-nous simplement d'omettre les propriétés -end lorsque l'élément n'est positionné que sur une piste. Nous verrons les cas de positionnement plus complexes plus tard.

D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-12.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

modele grilles-12.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-12.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-12.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,minmax(80px,1fr));
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 1;
}
.g2{
    grid-column-start: 1;
    grid-row-start: 2;
}
.g3{
    grid-column-start: 3;
    grid-row-start: 1;
}
.g4{
    grid-column-start: 1;
    grid-column-end: 3;
    grid-row-start: 3;
}
```

Hyper Text Markup Language fil length : 361 lines : 20 Ln : 5 Col : 11 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-12.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

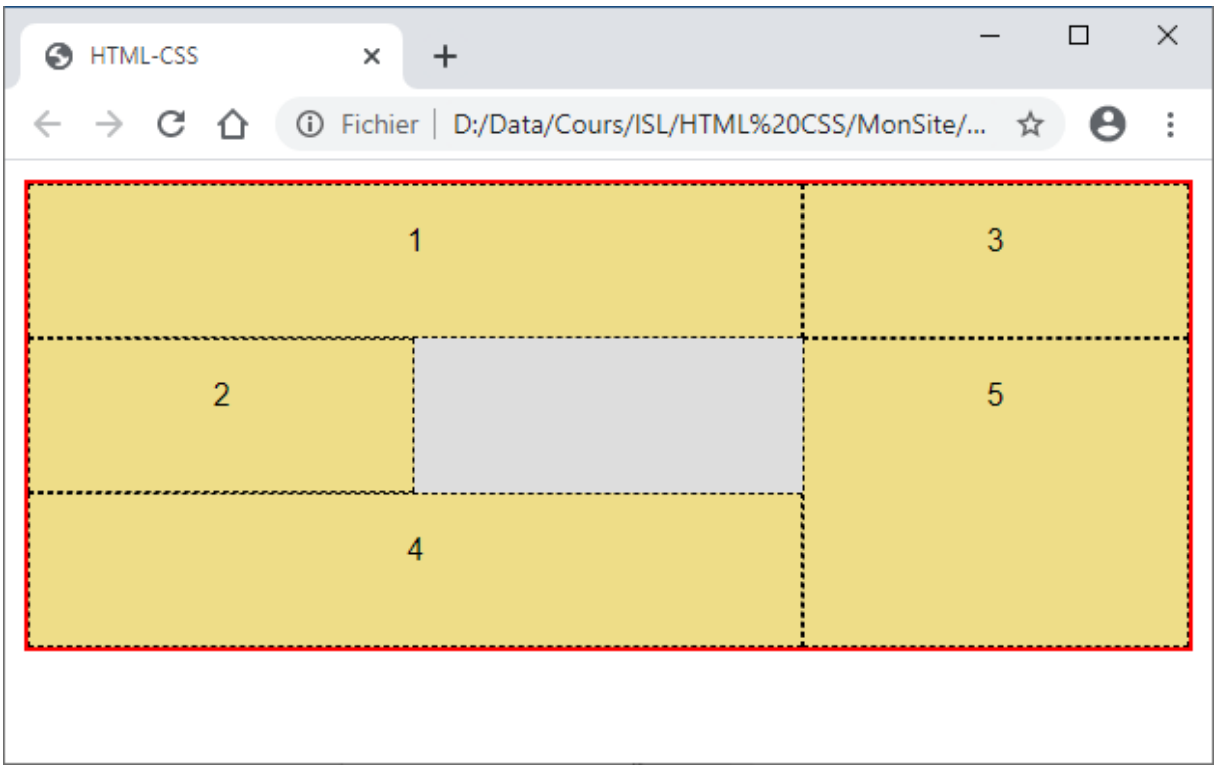
modele grilles-12.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-12.css"
```

modele grilles-12.css

```css
    grid-row-start: 3;
}
.g5{
    grid-column-start: 3;
    grid-row-start: 2;
    grid-row-end: 4;
}
```

Hyper Text Markup Language fil length : 361 lines : 20 Ln : 5 Col : 11 Sel : 0 | 0 Windows (CR LF) UTF-8 INS



Finalement, notez également qu'on va pouvoir placer un élément à partir de la fin de la grille en indiquant des numéros de lignes négatifs. Dans ce cas, -1 correspond à la dernière ligne définie explicitement, -2 à l'avant dernière etc. En pratique, cette notation est très peu utilisée et va répondre à des besoins très spécifiques.

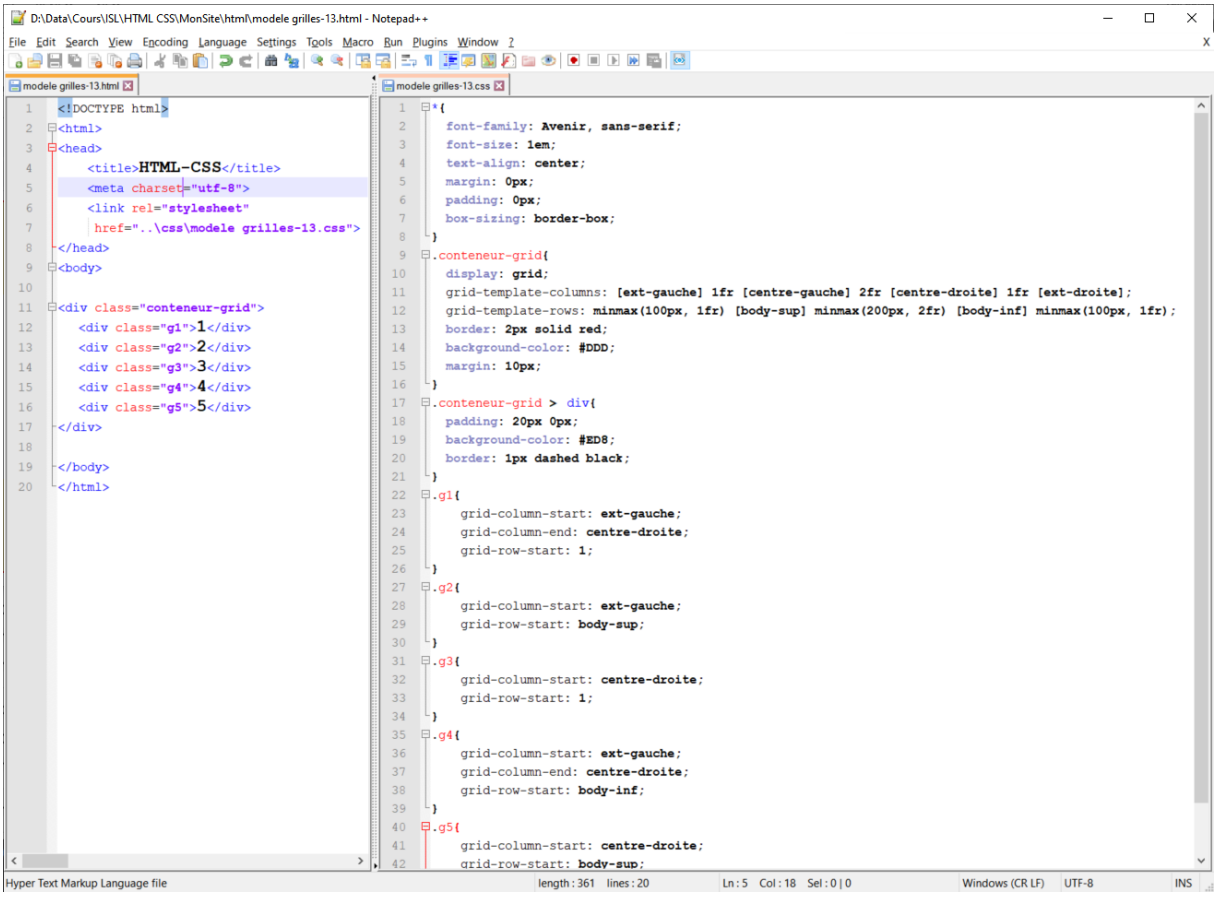
Nommer les lignes pour positionner les éléments

Jusqu'à présent, nous nous sommes contentés d'utiliser le système de numérotation des lignes créé automatiquement par les grilles.

Nous allons également pouvoir nommer nos lignes pour pouvoir ensuite les manipuler plus facilement.

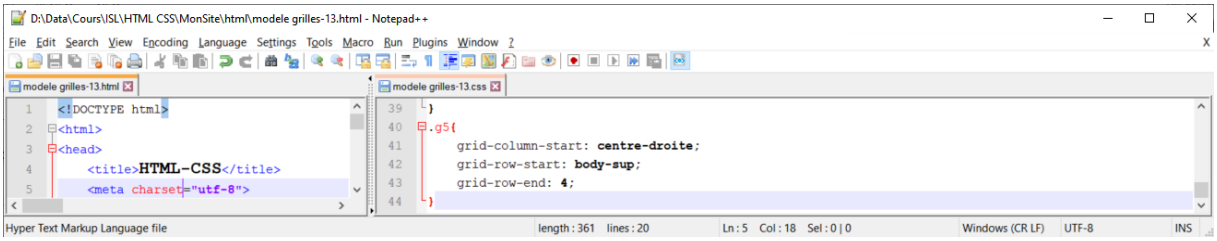
Pour cela, nous allons devoir indiquer le nom des lignes entre crochets dans la définition des pistes de la grille avec les propriétés grid-template-columns et grid-template-rows que nous avons étudié précédemment.

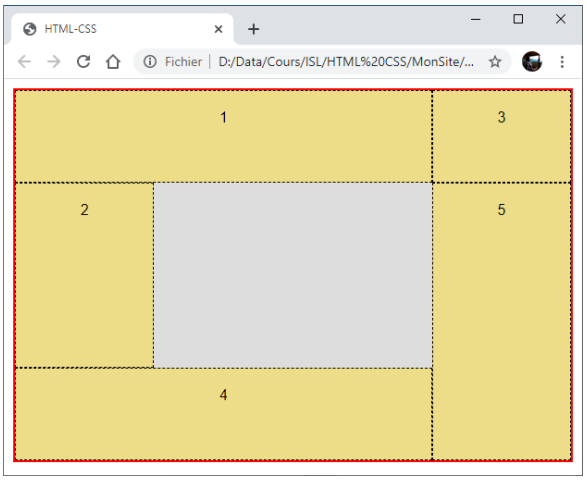
Reprenons notre grille précédente et nommons des lignes :


D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-13.html - Notepad++
modele grilles-13.html
modele grilles-13.css
<!DOCTYPE html>
<html>
<head>
<title>HTML-CSS</title>
<meta charset="utf-8">
<link rel="stylesheet"
href="..\css\modele grilles-13.css">
</head>
<body>

1
2
3
4
5

</body>
</html>
*{
font-family: Avenir, sans-serif;
font-size: 1em;
text-align: center;
margin: 0px;
padding: 0px;
box-sizing: border-box;
}
.conteneur-grid{
display: grid;
grid-template-columns: [ext-gauche] 1fr [centre-gauche] 2fr [centre-droite] 1fr [ext-droite];
grid-template-rows: minmax(100px, 1fr) [body-sup] minmax(200px, 2fr) [body-inf] minmax(100px, 1fr);
border: 2px solid red;
background-color: #DDD;
margin: 10px;
}
.conteneur-grid > div{
padding: 20px 0px;
background-color: #ED8;
border: 1px dashed black;
}
.g1{
grid-column-start: ext-gauche;
grid-column-end: centre-droite;
grid-row-start: 1;
}
.g2{
grid-column-start: ext-gauche;
grid-row-start: body-sup;
}
.g3{
grid-column-start: centre-droite;
grid-row-start: 1;
}
.g4{
grid-column-start: ext-gauche;
grid-column-end: centre-droite;
grid-row-start: body-inf;
}
.g5{
grid-column-start: centre-droite;
Hyper Text Markup Language file
length : 361 lines : 20
Ln : 5 Col : 18 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS


D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-13.html - Notepad++
modele grilles-13.html
modele grilles-13.css
<!DOCTYPE html>
<html>
<head>
<title>HTML-CSS</title>
<meta charset="utf-8">
}
.g5{
grid-column-start: centre-droite;
grid-row-start: body-sup;
grid-row-end: 4;
}
Hyper Text Markup Language file
length : 361 lines : 20
Ln : 5 Col : 18 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS


HTML-CSS
Fichier | D:/Data/Cours/ISL/HTML%20CSS/MonSite/...
1
3
2
5
4

Ici, ma grille possède à nouveau 3 colonnes et 3 rangées, donc 4 lignes verticales et 4 lignes horizontales. Ici, vous devez bien comprendre que l'on va à la fois nommer les lignes et définir nos pistes avec les propriétés grid-template-columns et grid-template-rows.

Regardons de plus près la déclaration relative à grid-template-columns : je commence par donner un nom à ma première ligne, puis je définis la taille de ma première colonne, puis je passe le nom de ma deuxième ligne, puis la taille de ma deuxième colonne et etc. jusqu'au nom de ma dernière ligne.

Bien sûr, nous ne sommes pas obligés de nommer toutes les lignes. Par exemple, je n'ai donné de nom qu'au deux lignes horizontales les plus à l'intérieur de ma grille dans grid-template-rows.

Ensuite, nous allons pouvoir utiliser ces noms plutôt que les numéros pour positionner nos éléments de grille.

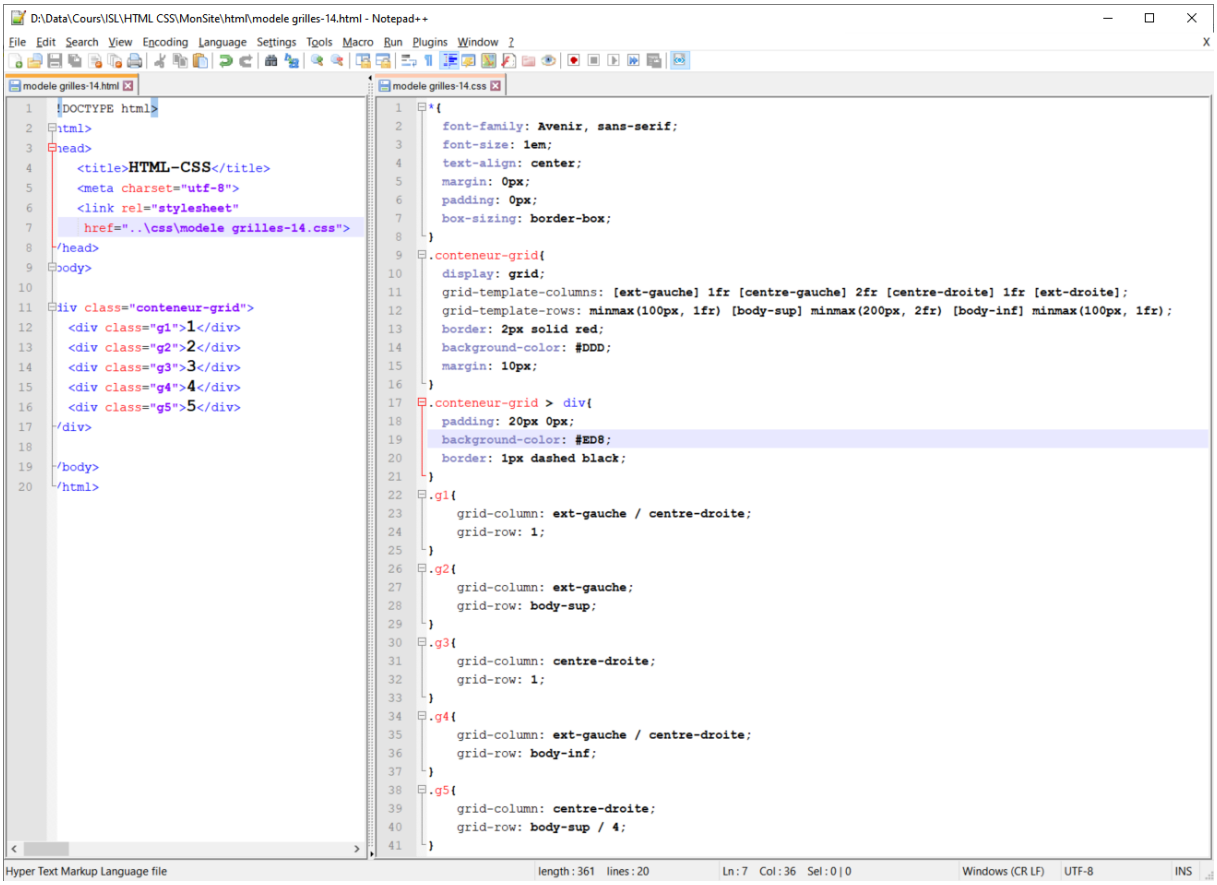
Utiliser les propriétés raccourcies grid-column et grid-row pour positionner les éléments dans la grille en fonction des lignes

La propriété grid-column est la version raccourcie des propriétés grid-column-start et grid-column-end.

La propriété grid-row est la version raccourcie des propriétés grid-row-start et grid-row-end.

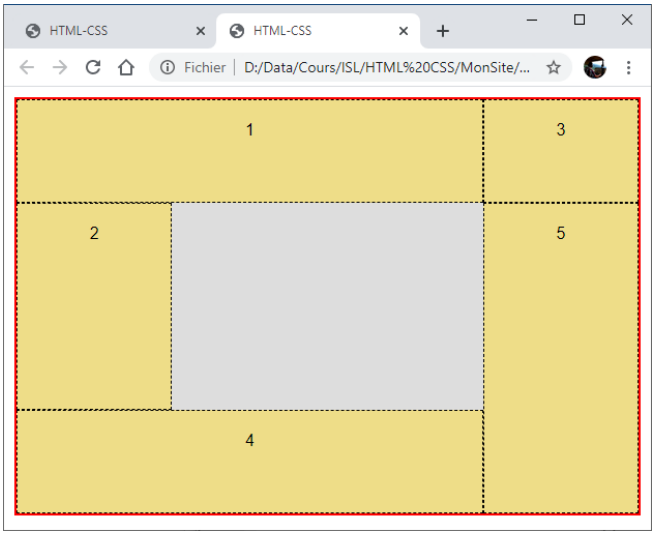
Nous allons donc pouvoir passer deux valeurs à chacune de ces propriétés. Les valeurs devront être séparées par un slash /. Les valeurs vont pouvoir être numérotées ou être des noms de lignes. Dans le cas où une seule valeur est passée, elle sera considérée comme étant la valeur de départ.

En reprenant notre grille précédente, on va ainsi pouvoir écrire :



```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-14.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>

</body>
</html>
```

```
*{
   font-family: Avenir, sans-serif;
   font-size: 1em;
   text-align: center;
   margin: 0px;
   padding: 0px;
   box-sizing: border-box;
}
.conteneur-grid{
   display: grid;
   grid-template-columns: [ext-gauche] 1fr [centre-gauche] 2fr [centre-droite] 1fr [ext-droite];
   grid-template-rows: minmax(100px, 1fr) [body-sup] minmax(200px, 2fr) [body-inf] minmax(100px, 1fr);
   border: 2px solid red;
   background-color: #DDD;
   margin: 10px;
}
.conteneur-grid > div{
   padding: 20px 0px;
   background-color: #ED8;
   border: 1px dashed black;
}
.g1{
    grid-column: ext-gauche / centre-droite;
    grid-row: 1;
}
.g2{
    grid-column: ext-gauche;
    grid-row: body-sup;
}
.g3{
    grid-column: centre-droite;
    grid-row: 1;
}
.g4{
    grid-column: ext-gauche / centre-droite;
    grid-row: body-inf;
}
.g5{
    grid-column: centre-droite;
    grid-row: body-sup / 4;
}
```



Positionner les éléments en utilisant les zones des grilles

Voyons à présent la notion de zone.

Utiliser la numérotation des lignes pour définir des zones et placer les éléments

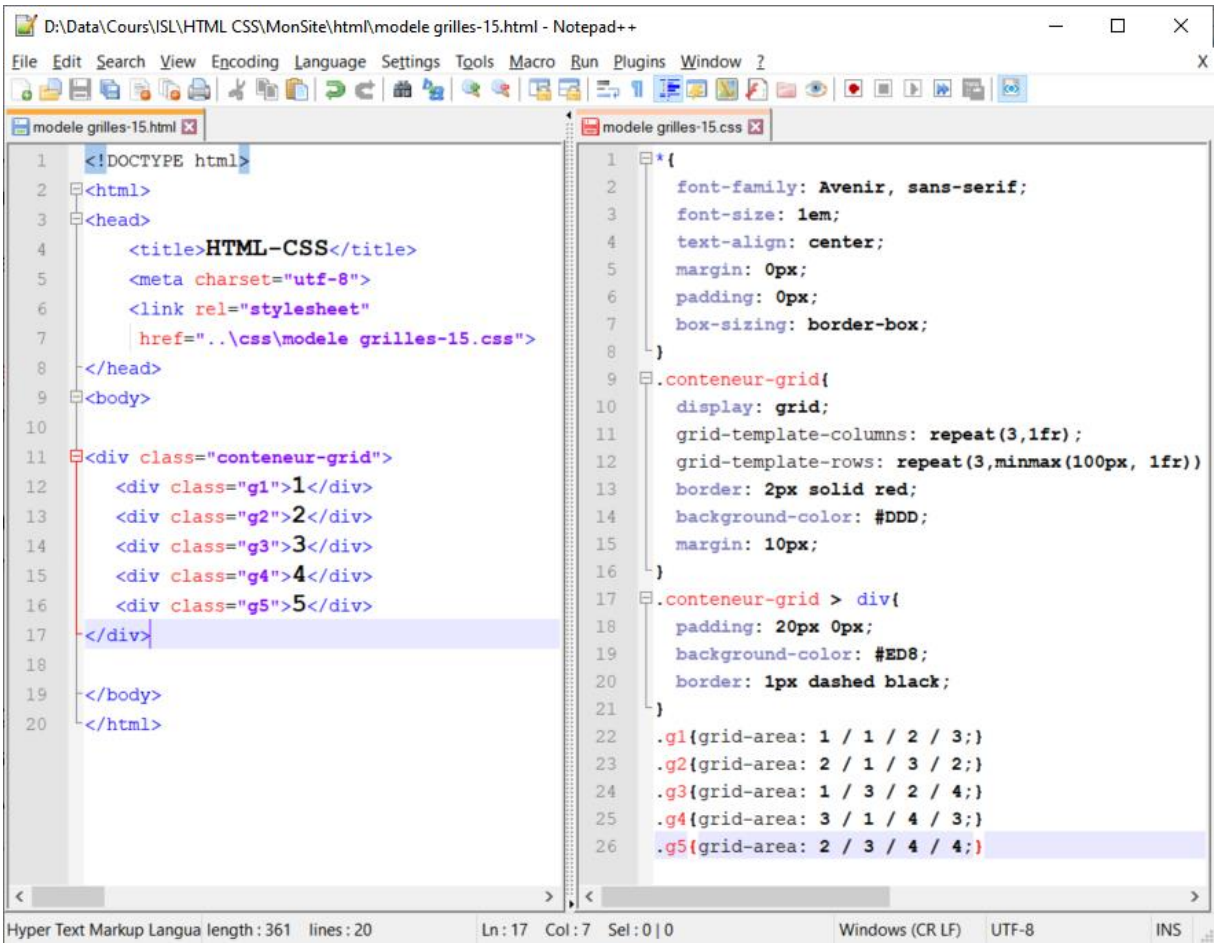
La propriété grid-area va nous permettre de définir des zones dans une grille. Pour cela, nous allons déjà pouvoir lui passer 4 numéros en valeur séparés par des slash.

La première valeur correspond au numéro de la ligne horizontale où la zone doit commencer, la deuxième valeur correspond au numéro de la ligne verticale où la zone doit commencer, la troisième valeur correspond au numéro de la ligne horizontale où la zone se termine et la quatrième valeur correspond au numéro de la ligne verticale où la zone se termine.

L'ordre des valeurs (pour un langage qui s'écrit de gauche à droite) est donc un ordre antihoraire : haut, gauche, bas, droite.

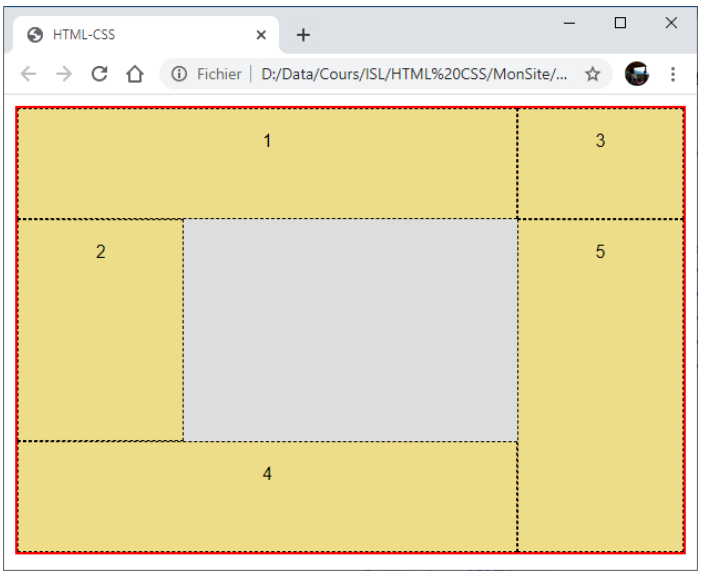
La propriété grid-area va s'appliquer aux éléments de grille et ainsi également définir leur taille. Elle peut ainsi remplacer les propriétés grid-column-start, grid-column-end, grid-row-start et grid-row-end.

Illustrons cela en reprenant à nouveau notre grille précédente et en utilisant plutôt grid-area pour placer les éléments dans la grille.




```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-15.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>

</body>
</html>
```

```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,minmax(100px, 1fr))
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{grid-area: 1 / 1 / 2 / 3;}
.g2{grid-area: 2 / 1 / 3 / 2;}
.g3{grid-area: 1 / 3 / 2 / 4;}
.g4{grid-area: 3 / 1 / 4 / 3;}
.g5{grid-area: 2 / 3 / 4 / 4;}
```



Nommer des zones pour positionner les éléments

Nous allons également pouvoir nommer nos zones de grille pour ensuite placer les éléments. Pour cela, il va falloir procéder en deux étapes.

Nous allons déjà passer les différents noms de zones à grid-area puis ensuite les utiliser avec la propriété grid-template-areas pour définir les zones en soi dans notre grille.

Nous allons passer à grid-template-areas le nom de nos zones en définissant pour chaque cellule la zone à laquelle elle appartient. Nous allons entourer les valeurs pour chaque rangée de la grille avec un couple de guillemets "". Pour laisser une cellule hors zone, nous indiquerons un point . en valeur.

Notez que les zones crées doivent toujours être rectangulaires. D'autres formes comme une forme en « L » ne sont à l'heure actuelle pas permises.

D:\Data\Cours\ISL\HTML CSS\MonSite\css\modele grilles-16.css - Notepad++

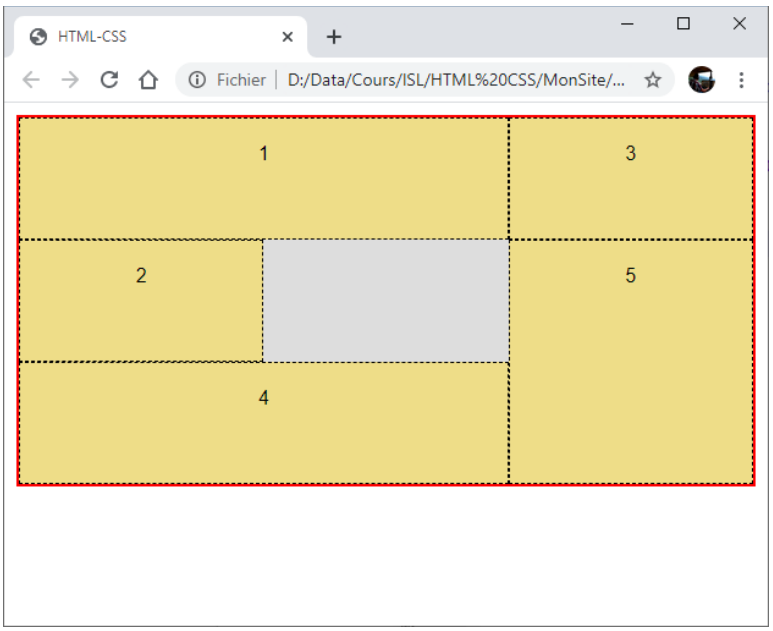
modele grilles-16.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-16.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-16.css

```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,minmax(100px, 1fr));
  grid-template-areas: "hg hg hd" "mg . bd" "bg bg bd";
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{grid-area: hg;}
.g2{grid-area: mg;}
.g3{grid-area: hd;}
.g4{grid-area: bg;}
.g5{grid-area: bd;}
```

Cascade Style Sheets File | length : 613 | lines : 27 | Ln : 27 | Col : 20 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



Ici, la zone hg recouvre les deux premières cellules de la première rangée de notre grille tandis que la zone hd recouvre la troisième cellule de cette rangée.

La zone mg recouvre la première cellule de la deuxième rangée de notre grille.

La zone bg recouvre les deux premières cellules de la troisième rangée de la grille.

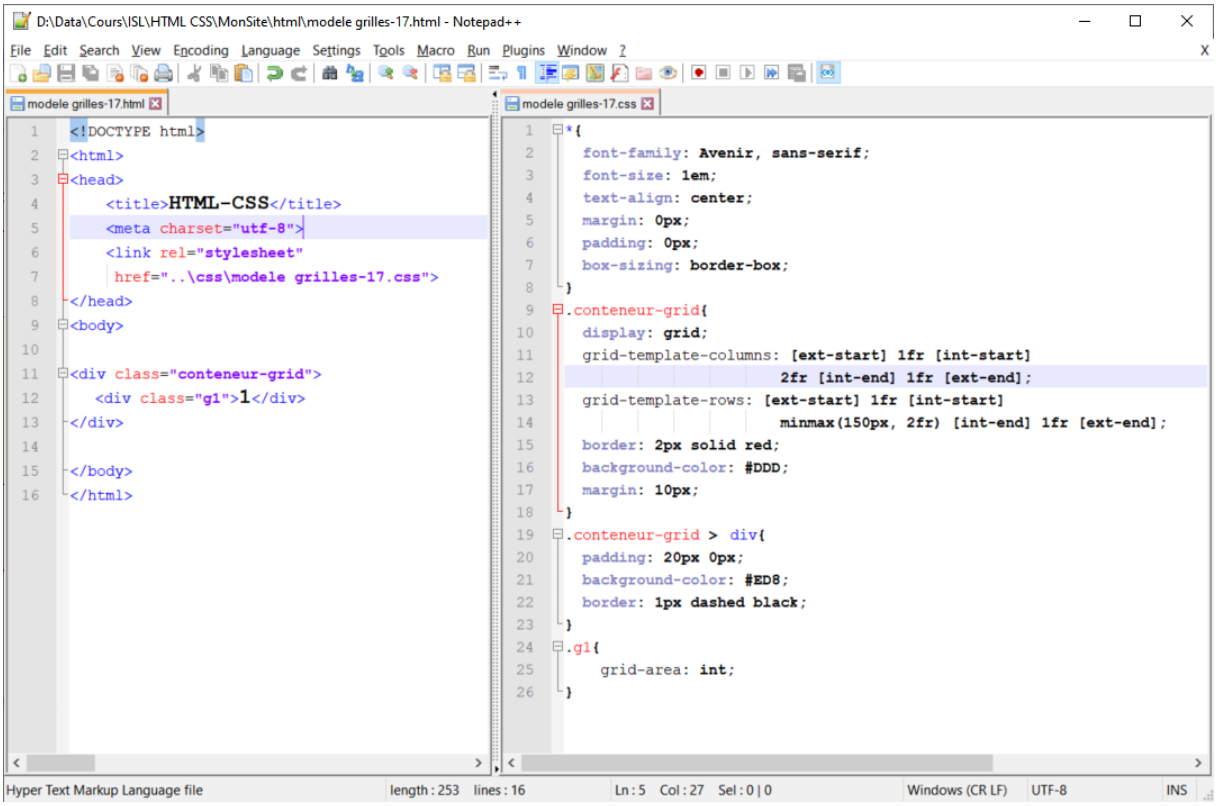
Finalement, la zone bd recouvre la dernière cellule de la deuxième rangée de la grille ainsi que la dernière cellule de la troisième rangée de la grille.

La deuxième cellule de la deuxième rangée de notre grille n'appartient à aucune zone définie explicitement.

Lignes nommées et zones implicites, zones nommées et lignes implicites

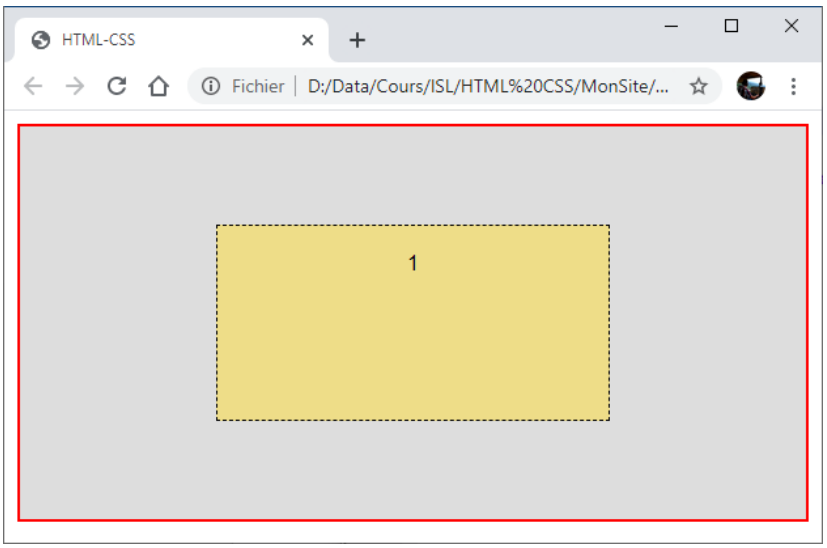
Nous allons pouvoir donner n'importe quel nom à nos lignes et à nos zones. Cependant, certains noms de lignes vont déclencher des mécanismes de création de zones nommées et à l'inverse la création d'une zone nommée va automatiquement définir des noms pour les lignes qui la définissent.

Par exemple, si on utilise un même nom avec les suffixes -start et -end pour nommer les lignes de départ et d'arrivée qui peuvent définir une zone, alors la grille va automatiquement créer la zone en question et va lui donner le même nom sans préfixe. Nous allons ensuite pouvoir utiliser ce nom pour placer nos éléments de grille.



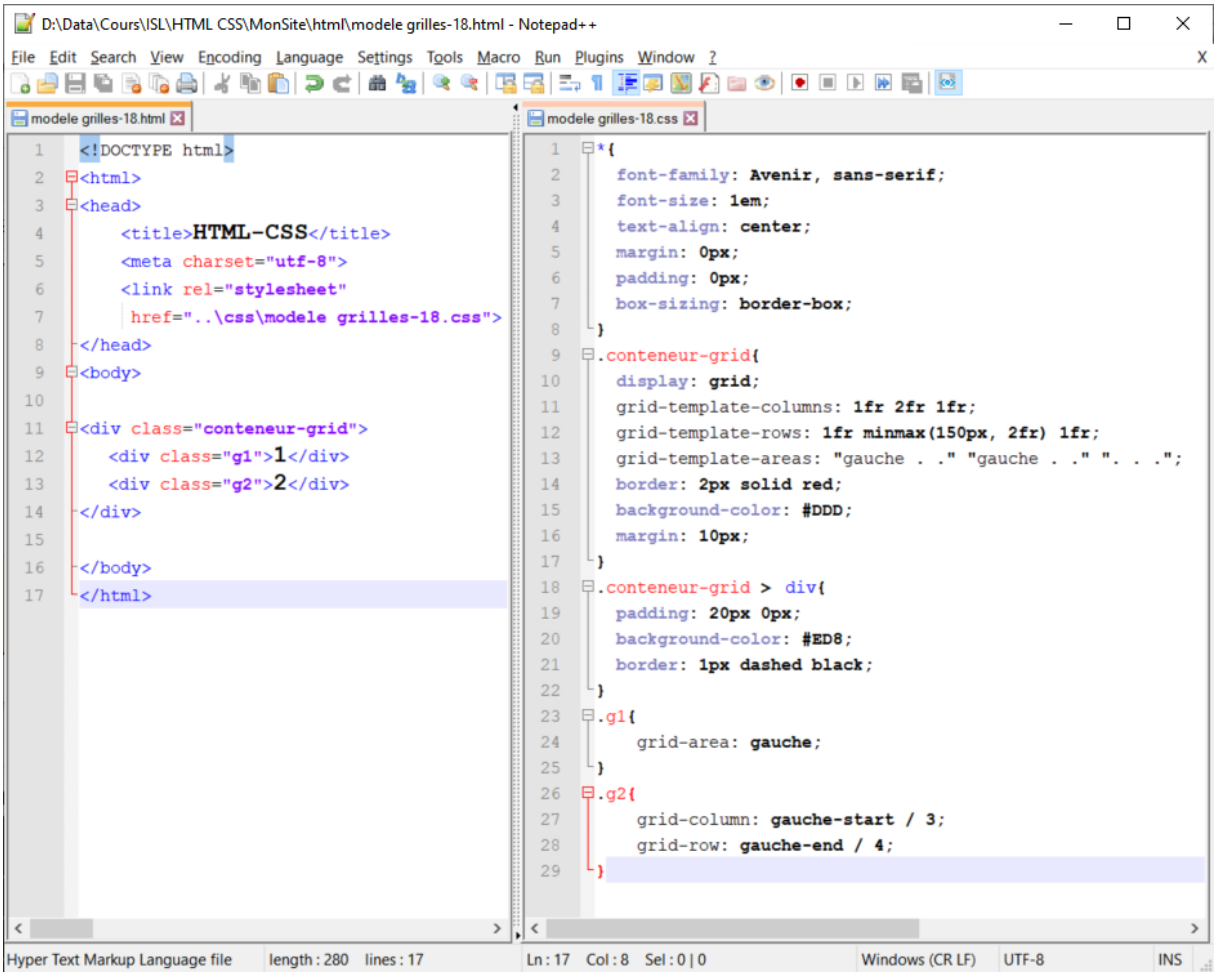

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-17.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
</div>

</body>
</html>
```

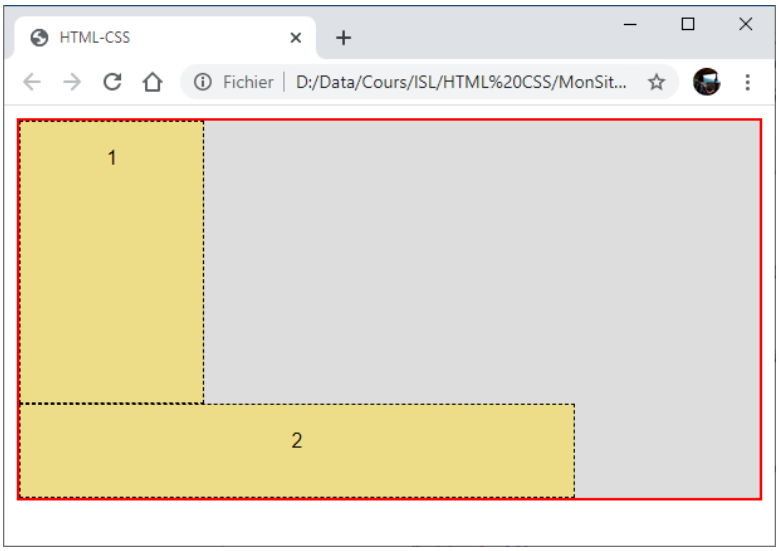
```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: [ext-start] 1fr [int-start]
                         2fr [int-end] 1fr [ext-end];
  grid-template-rows: [ext-start] 1fr [int-start]
                      minmax(150px, 2fr) [int-end] 1fr [ext-end];
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{
    grid-area: int;
}
```



Ici, la grille va automatiquement créer une zone ext et une zone int à partir de la définition de nos lignes. On se sert de la zone int pour positionner notre élément de grille.

A l'inverse, en créant une zone nommée, la grille va automatiquement nommer les lignes qui servent à la définir en leur ajoutant les suffixes -start pour la ligne de départ et -end pour la ligne d'arrivée.

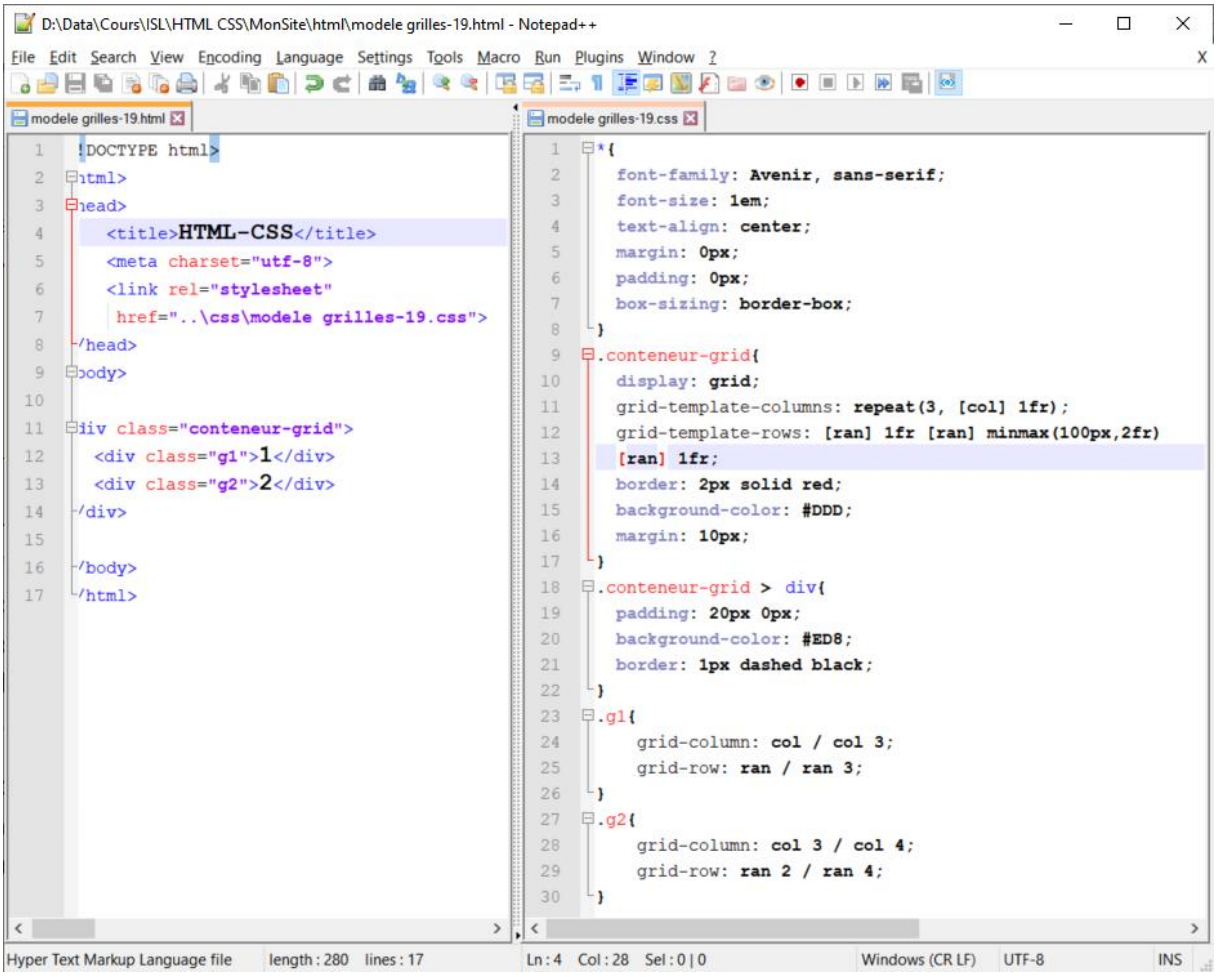


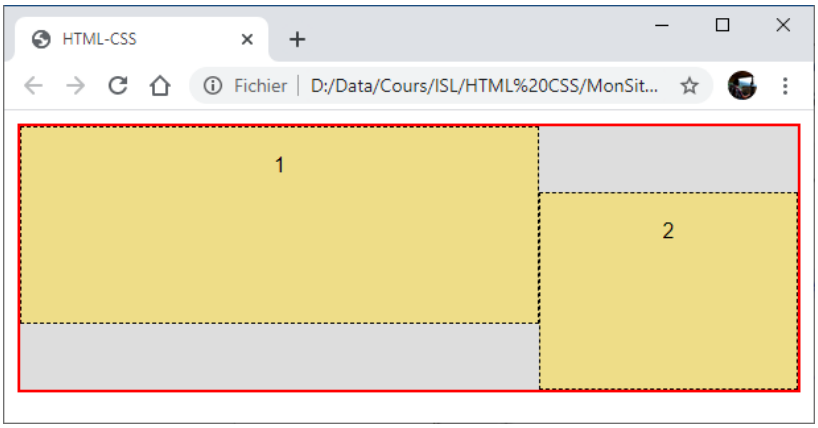


Ici, on définit une zone gauche qui recouvre la première cellule des première et deuxième rangées de la grille. Les lignes gauche-start et gauche-end sont alors automatiquement créées par la grille. On s'en sert pour positionner notre deuxième élément de grille.

Notez qu'il n'est pas gênant qu'une ligne verticale possède le même nom qu'une ligne horizontale ou même que deux lignes verticales ou deux lignes horizontales (ou plus) possèdent le même nom. Cela est parfaitement autorisé et peut faire gagner du temps de développement si on souhaite créer des pistes similaires.

En cas d'ambiguïté sur le nom de la ligne, la grille utilisera toujours la première des lignes qui portent le même nom. Pour lever l'ambiguïté, on peut préciser le numéro de la ligne derrière son nom pour la cibler en particulier.





Ici, on crée une grille possédant à nouveau 3 colonnes et 3 rangées. Toutes les lignes verticales vont porter le nom col (observez bien la syntaxe dans repeat() tandis que toutes les lignes horizontales vont porter le nom ran.

Pour placer un élément dans la grille à partir ou jusqu'à une ligne qui n'est pas la première, il faut préciser le numéro de ligne après son nom pour lever l'ambiguïté.

Les propriétés raccourcies grid-template et grid

Notez qu'il existe deux autres propriétés raccourcies qui peuvent servir à positionner les éléments dans une grille : les propriétés grid-template et grid.

La propriété grid-template est la version raccourcie des propriétés grid-template-rows, grid-template-columns et grid-template-areas.

La propriété grid est une version qui condense encore plus de propriété puisque c'est la version raccourcie de grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns et grid-auto-areas.

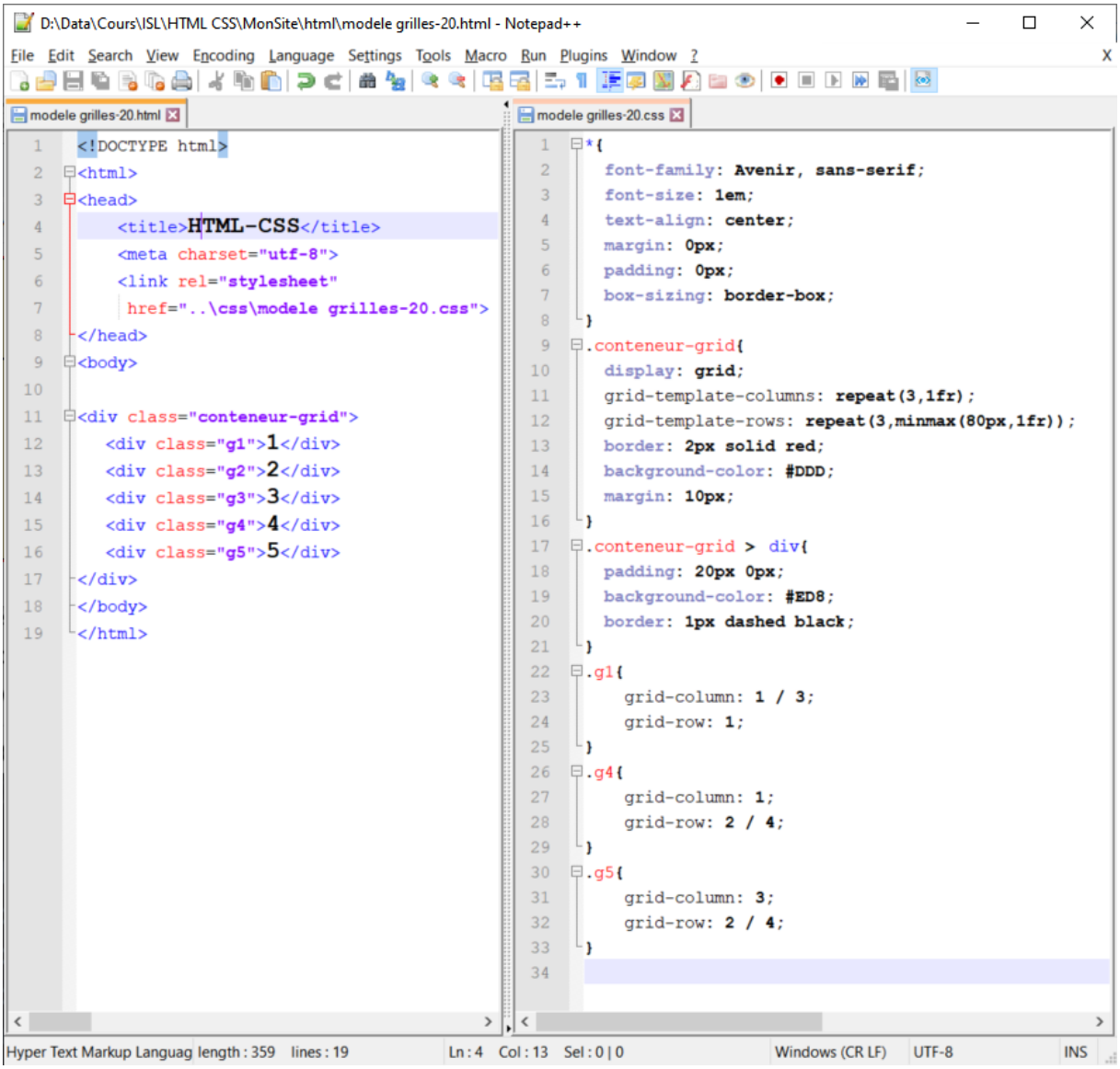
Personnellement, je vous déconseille pour des raisons évidentes de lisibilité d d'utiliser ces deux versions raccourcies qui rendre le code très compliqué à comprendre.

Contrôler le positionnement automatique des éléments dans la grille

Mélange de positionnement explicite et implicite

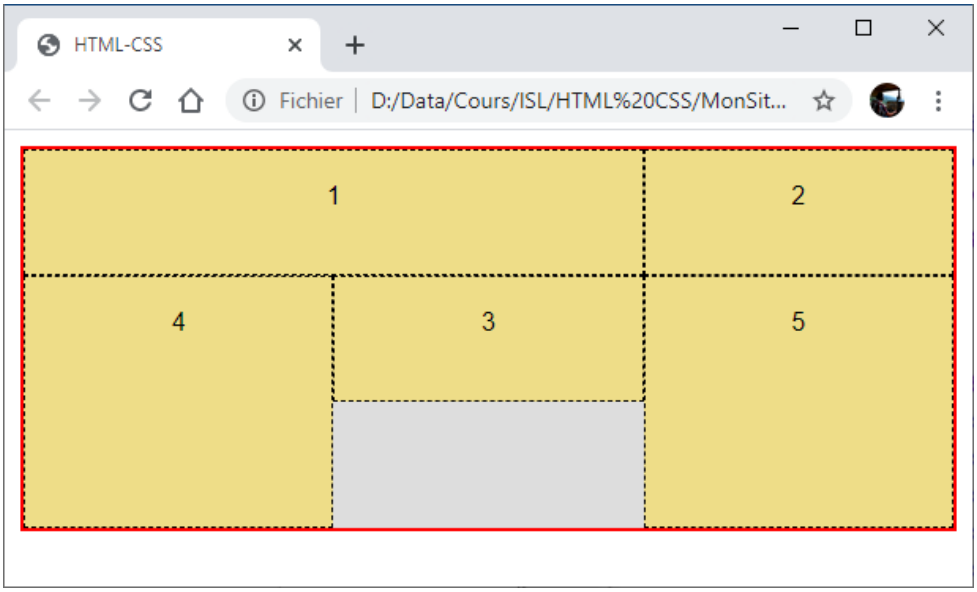
On ne va pas être obligé de positionner explicitement (c'est-à-dire en utilisant une des propriétés vues précédemment) tous les éléments d'une grille.

Dans le cas où certains éléments sont positionnés explicitement et d'autres non, la grille va commencer par placer les éléments positionnés explicitement puis placera ensuite les autres éléments en utilisant le positionnement par défaut.




```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-20.css">
</head>
<body>

<div class="conteneur-grid">
   <div class="g1">1</div>
   <div class="g2">2</div>
   <div class="g3">3</div>
   <div class="g4">4</div>
   <div class="g5">5</div>
</div>
</body>
</html>
```

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(3,minmax(80px,1fr));
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  padding: 20px 0px;
  background-color: #ED8;
  border: 1px dashed black;
}
.g1{
    grid-column: 1 / 3;
    grid-row: 1;
}
.g4{
    grid-column: 1;
    grid-row: 2 / 4;
}
.g5{
    grid-column: 3;
    grid-row: 2 / 4;
}
```



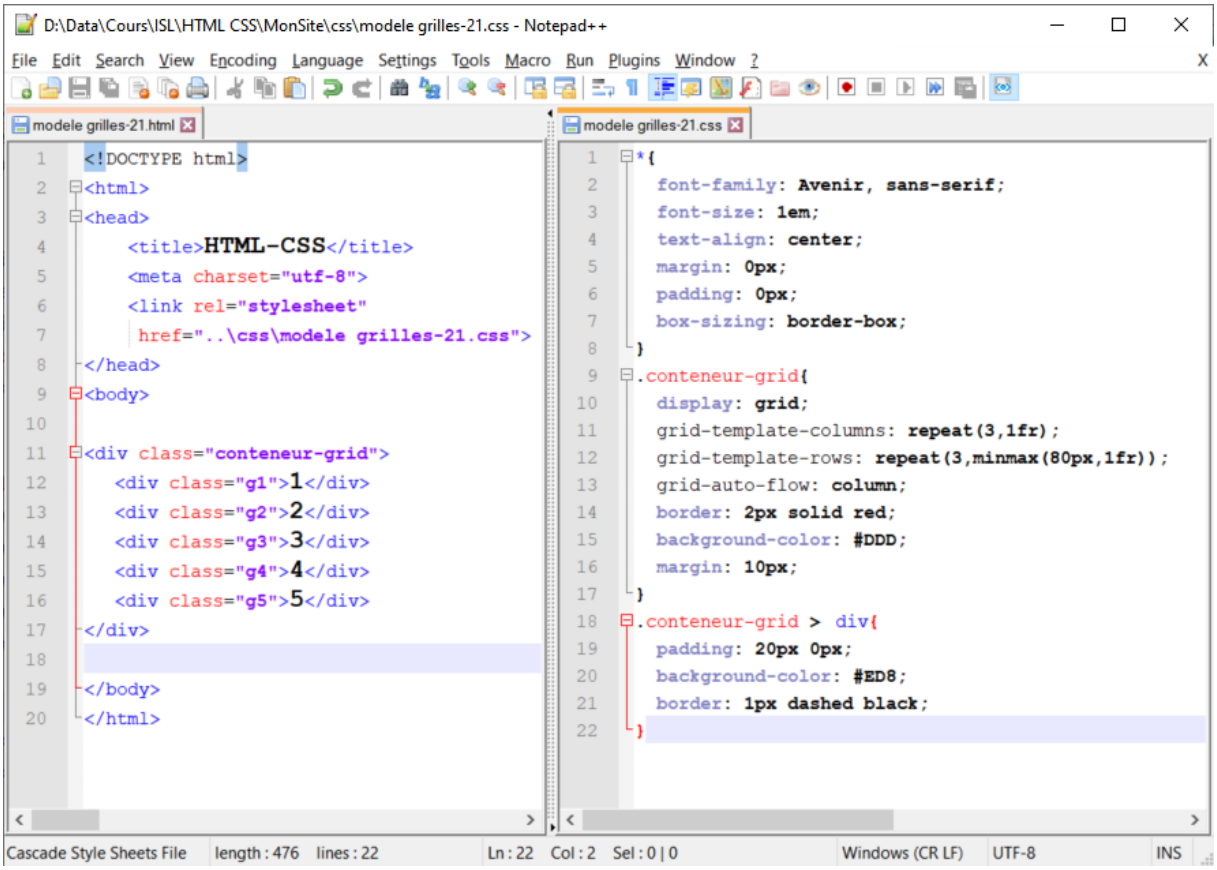
Ici, on ne positionne explicitement que nos éléments 1, 4 et 5. Ce seront donc les premiers éléments positionnés dans la grille. Ensuite, la grille va positionner automatiquement les éléments 2 et 3 en utilisant le positionnement par défaut dès qu'il y aura un espace suffisant dans la grille.

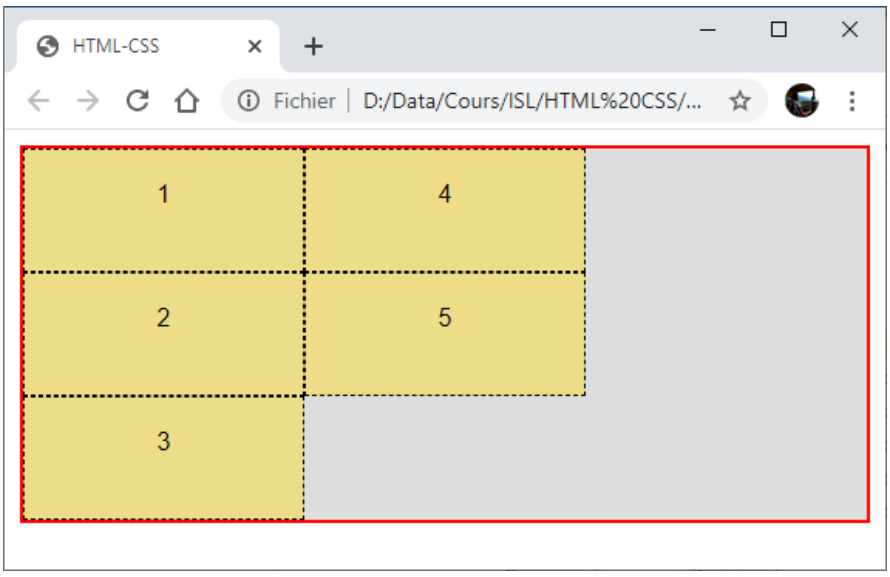
L'élément n°2 est le premier élément non positionné explicitement et va donc se positionner dans le premier espace disponible. L'élément n°3 est le deuxième élément non positionné et va donc se positionner dans le premier espace disponible suivant l'élément n°2.

Placer les éléments automatiquement en colonne

Le placement automatique des éléments est défini par la propriété grid-auto-flow. La valeur par défaut est row qui signifie que les éléments positionnés automatiquement viendront se placer à côté des autres tant que possible.

Nous allons également pouvoir définir un placement automatique en colonne en utilisant la valeur column de cette même propriété.

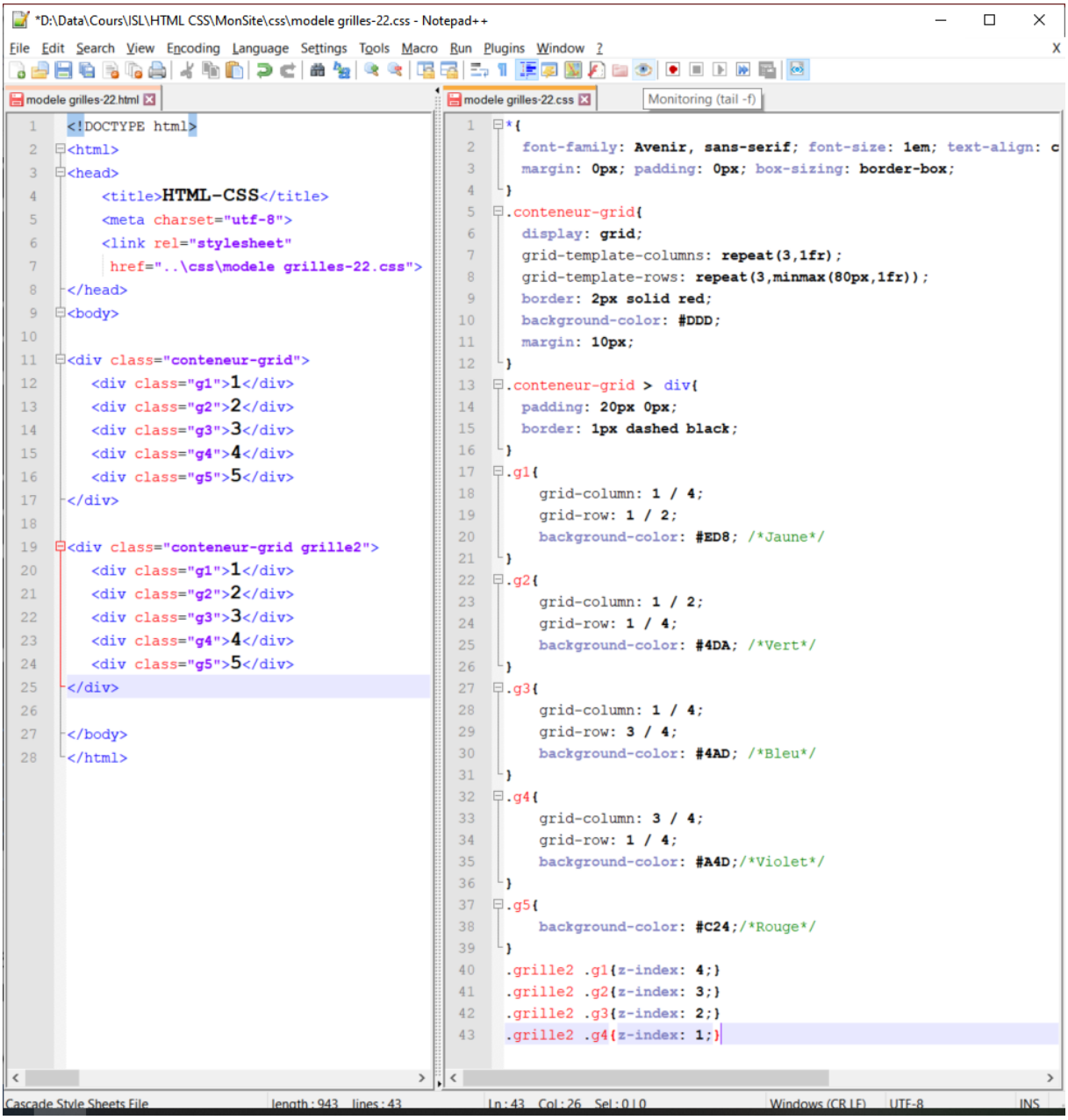


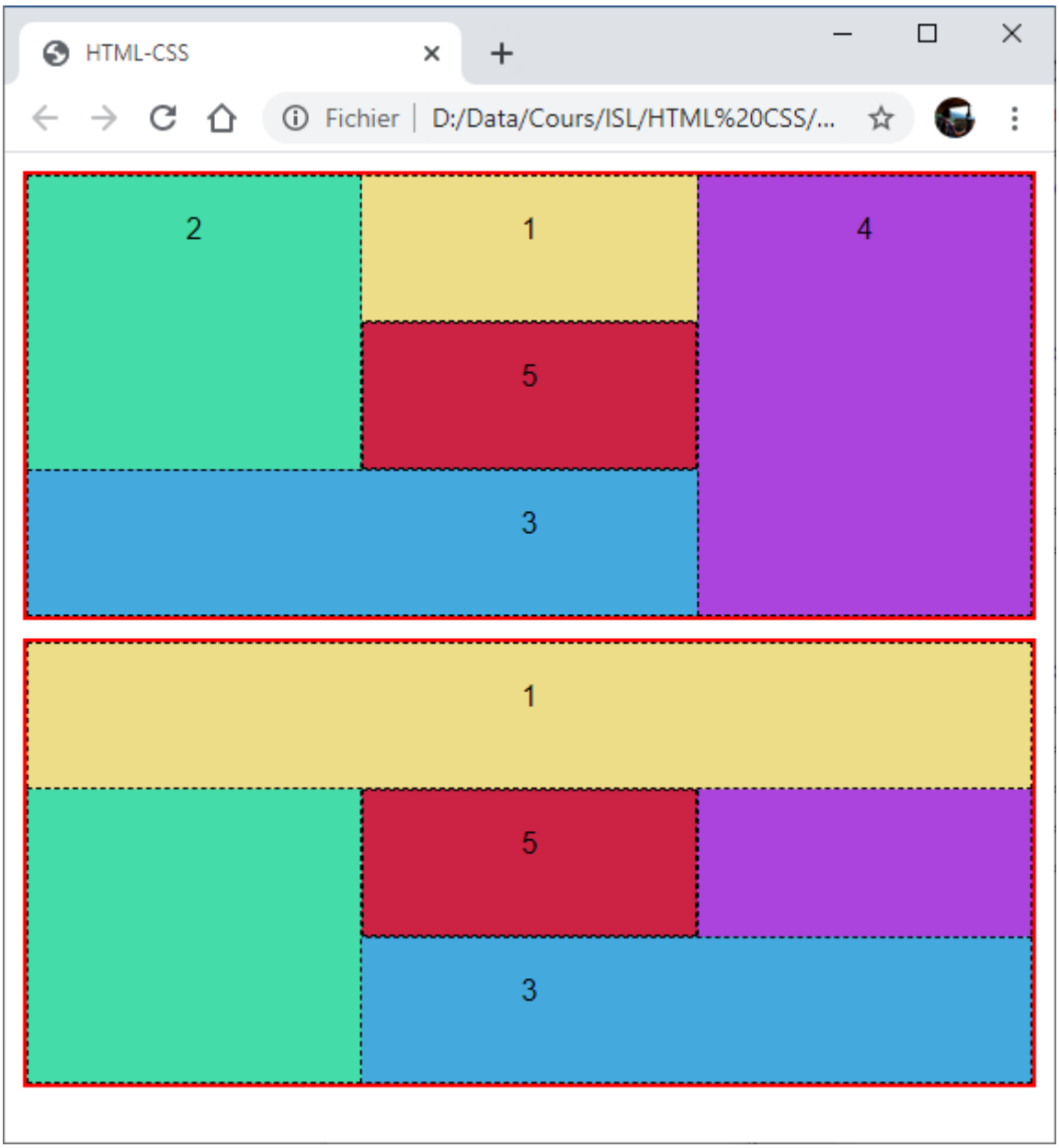


Contrôler le chevauchement des éléments dans une grille

Notez également que nous allons pouvoir positionner plusieurs éléments de grille dans les mêmes cellules. Par défaut, le dernier élément déclaré se placera au-dessus des autres.

Nous allons alors pouvoir modifier ce comportement et choisir quel élément devra apparaitre au-dessus de quel autre en définissant un z-index pour les différents éléments se chevauchant.





Ici, nous créons deux grilles identiques. Les éléments 1, 2, 3 et 4 des deux grilles vont se chevaucher. La première grille suit le comportement par défaut des éléments : le deuxième élément déclaré dans le code viendra se positionner au-dessus du premier, le troisième au-dessus du deuxième et etc.

Nous ajoutons ensuite des z-index aux éléments de notre deuxième grille afin de modifier l'ordre d'empilement des éléments.

Aligner et espacer les éléments de grille

Dans la leçon précédente, nous avons vu comme positionner ou placer des éléments dans une grille.

Il nous reste plus que deux choses à voir pour être totalement opérationnel avec nos grilles : comment aligner les éléments et comment les espacer.

En effet, par défaut, les éléments de grille vont s'étirer pour remplir tout l'espace dans lequel on les a positionnés quel que soit leur contenu. Nous allons pouvoir modifier ce comportement grâce aux propriétés d'alignement.

La plupart des propriétés d'alignement que nous allons voir ici devraient vous rappeler celles déjà vues dans la partie liée au modèle des boites flexibles et c'est tout à fait normal puisque le module CSS relatif à l'alignement (dans la spécification officielle) est commun à ces deux modèles.

Les axes des grilles

Le modèle des boites flexibles était un modèle unidimensionnel : c'est un modèle qui fonctionne avant tout selon un axe principal.*

Le modèle des grilles est lui un modèle bidimensionnel : il va fonctionner selon deux axes qu'on va pouvoir manipuler de façon égale. On ne parlera donc plus d'axe « principal » et d'axe « secondaire ». A la place, nous parlerons d'axe de bloc et d'axe en ligne.

L'axe de bloc est l'axe selon lequel les éléments de type block sont disposés, c'est-à-dire l'axe vertical puisque les éléments de type block se placent par défaut les uns en dessous des autres.

L'axe en ligne est l'axe selon lequel les éléments de type inline sont disposés, c'est-à-dire l'axe horizontal puisque les éléments de type inline vont par défaut se placer les uns à côté des autres.

Avec les grilles, nous allons pouvoir gérer indifféremment l'alignement des éléments selon l'axe de bloc et l'axe en ligne.

Aligner les éléments ou les pistes d'une grille par rapport à l'axe de bloc

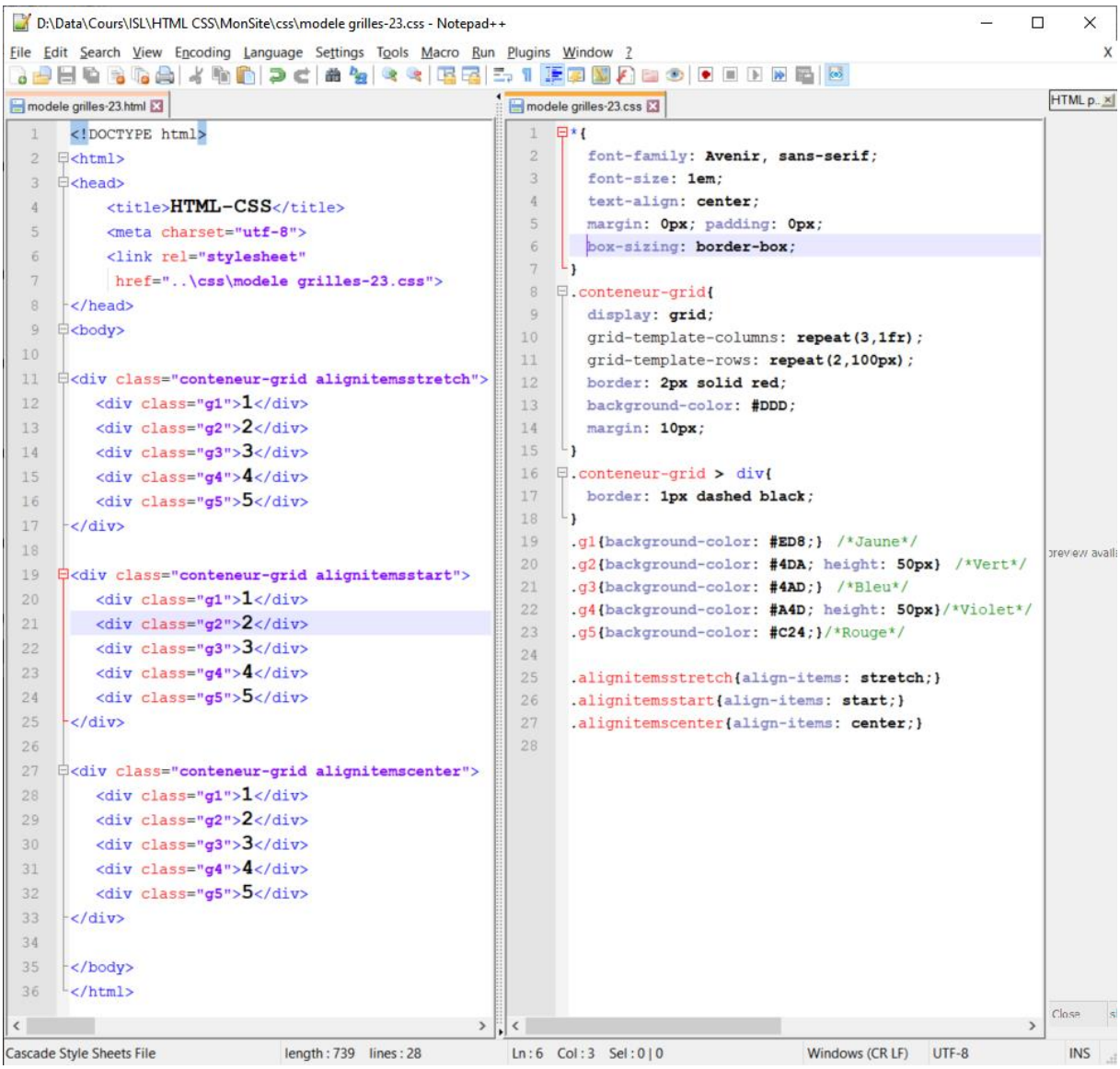
Commençons avec l'alignement des éléments ou des pistes selon l'axe de bloc. Nous allons pouvoir gérer cet alignement en utilisant les propriétés align-items, align-self et align-content.

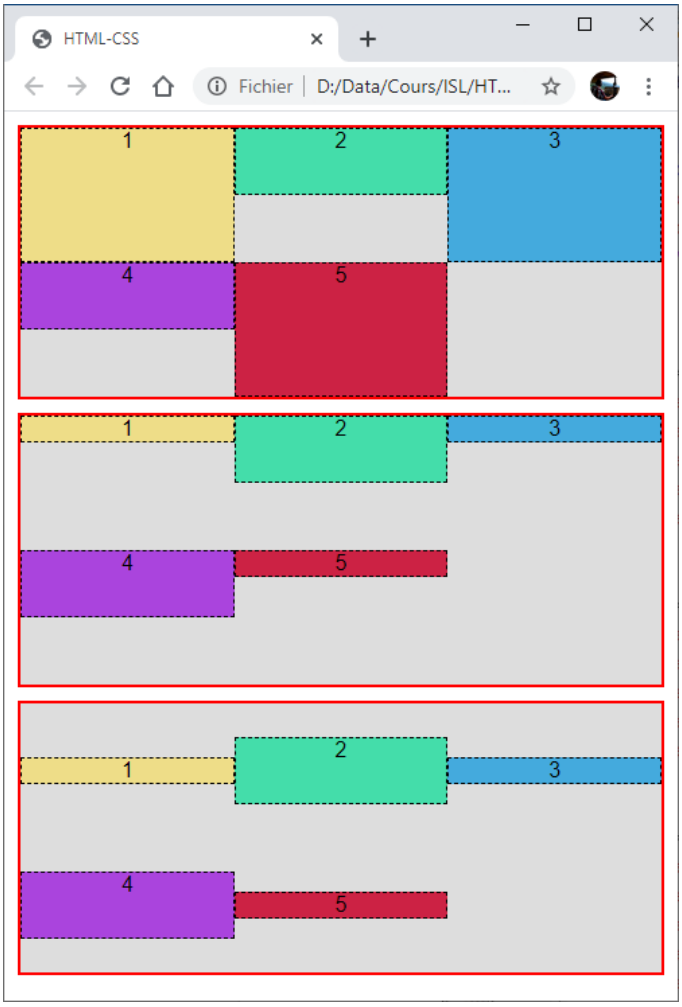
La propriété align-items permet de contrôler l'alignement des éléments de grille le long de l'axe de bloc.

On va devoir appliquer cette propriété à l'élément conteneur de grille et nous allons pouvoir lui passer les valeurs suivantes :

- stretch : Valeur par défaut. Les éléments de grille vont s'étirer pour occuper toute la hauteur des cellules dans lesquelles ils sont placés ;

- start : La hauteur des éléments va être déterminée par leur contenu et les éléments vont s'aligner à partir du départ (en haut) de l'espace dans lequel ils sont définis ;
- end : La hauteur des éléments va être déterminée par leur contenu et les éléments vont s'aligner à partir de la fin (en bas) de l'espace dans lequel ils sont définis ;
- center : La hauteur des éléments va être déterminée par leur contenu et les éléments vont s'aligner au milieu de l'espace dans lequel ils sont définis ;
- baseline : Les éléments vont être alignés de telle sorte à ce que leurs lignes de base soient alignées les unes par rapport aux autres.





Ici, on impose à nos rangées de faire 100px de haut. La hauteur intrinsèque de nos éléments de grille, qui ne contiennent qu'un chiffre, est inférieure à la hauteur des rangées. On va donc pouvoir utiliser align-items pour aligner les éléments en hauteur.

On définit également des éléments de hauteurs différentes afin de bien voir l'effet des propriétés d'alignement que nous étudions. Notez ici que la valeur stretch n'a pas la capacité d'étirer un élément qui possède une taille explicite.

La propriété align-self quant à elle, permet de contrôler l'alignement d'un élément de grille en particulier dans l'axe de bloc. On va devoir cette fois-ci l'appliquer à l'élément qu'on souhaite aligner. Cette propriété va pouvoir prendre les mêmes valeurs que la propriété align-items.

D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-24.html - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

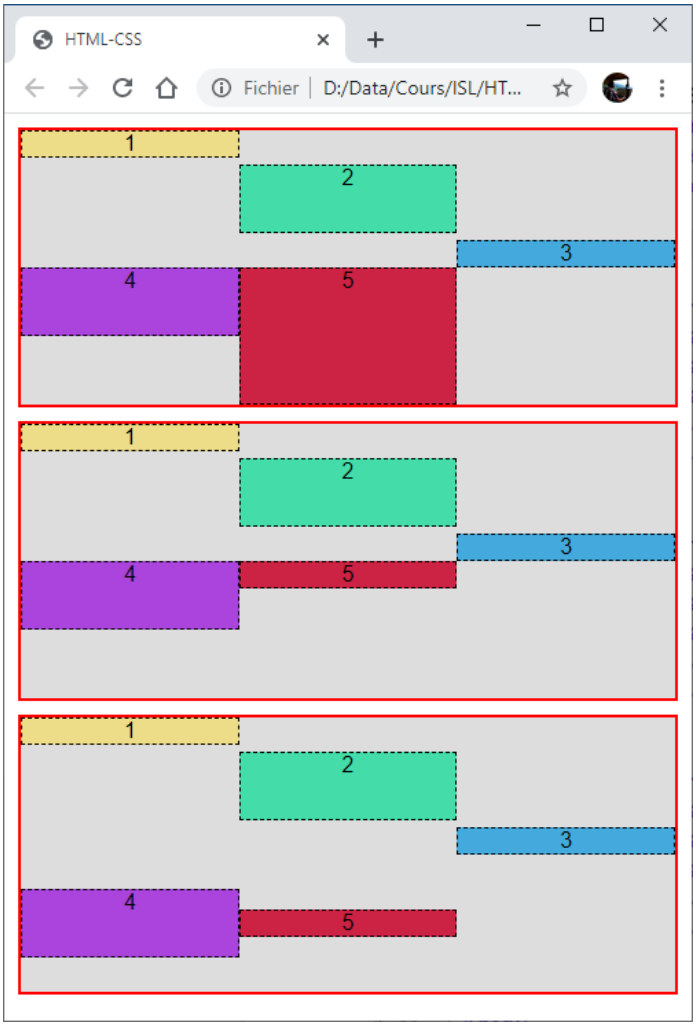
modele grilles-24.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-24.css">
</head>
<body>

<div class="conteneur-grid alignitemsstretch">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

<div class="conteneur-grid alignitemsstart">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

<div class="conteneur-grid alignitemscenter">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-24.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,100px);
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  border: 1px dashed black;
}
.g1{background-color: #ED8;} /*Jaune*/
.g2{background-color: #4DA; height: 50px} /*Vert*/
.g3{background-color: #4AD;} /*Bleu*/
.g4{background-color: #A4D; height: 50px}/*Violet*
.g5{background-color: #C24;}/*Rouge*/

.alignitemsstretch{align-items: stretch;}
.alignitemsstart{align-items: start;}
.alignitemscenter{align-items: center;}

.alignselfstart{align-self: start;}
.alignselfcenter{align-self: center;}
.alignselfend{align-self: end;}
```

Hyper Text Markup Language file | length : 894 | lines : 36 | Ln : 7 | Col : 36 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



Si les pistes d'une grille n'occupent pas toute la hauteur dans la grille, alors on va pouvoir les aligner dans le conteneur selon l'axe de bloc, c'est-à-dire choisir comment distribuer l'espace restant en hauteur. Nous allons pouvoir faire cela en utilisant la propriété align-content.

Cette propriété va être appliquée à l'élément conteneur et nous allons pouvoir lui passer les valeurs suivantes :

- start : La hauteur des pistes va être définie par la hauteur des éléments et les pistes vont s'empiler au début du conteneur selon l'axe de bloc c'est-à-dire en haut pour un langage dont l'écriture se fait de haut en bas ;
- end : La hauteur des pistes va être définie par la hauteur des éléments et les pistes vont s'empiler à la fin du conteneur selon l'axe de bloc c'est-à-dire en bas pour un langage dont l'écriture se fait de haut en bas ;
- center : La hauteur des pistes va être définie par la hauteur des éléments et les pistes vont s'empiler au centre du conteneur selon l'axe de bloc ;

- baseline : Les pistes vont être alignées de telles sortes à ce que les lignes de base soient alignées selon l'axe de bloc ;
- stretch : Les pistes vont s'étirer pour prendre toute la hauteur disponible dans le conteneur (sauf si une hauteur autre que auto a été explicitement définie) ;
- space-between : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes selon l'axe de bloc. La première et la dernière pistes vont être collées contre les bords du conteneur ;
- space-around : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes selon l'axe de bloc. L'espace entre les bords du conteneur et la première et la dernière piste sera deux fois moins important qu'entre deux pistes ;
- space-evenly : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes et entre une piste et un bord du conteneur selon l'axe de bloc.

D:\Data\Cours\ISL\HTML CSS\MonSite\css\modele grilles-25.css - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

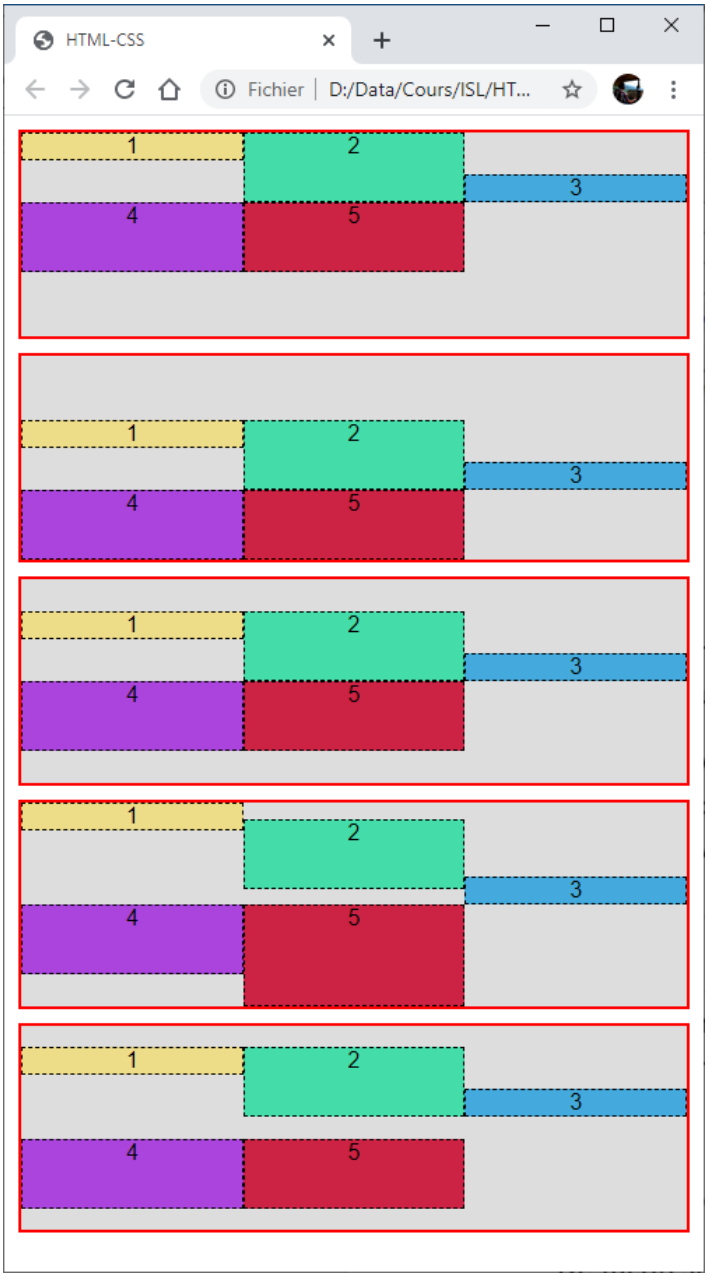
modele grilles-25.html

```
</head>
<body>

<div class="conteneur-grid aligncontentstart">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid aligncontentend">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid aligncontentcenter">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid aligncontentstretch">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid aligncontentevenly">
    <div class="g1 alignselfstart">1</div>
    <div class="g2 alignselfcenter">2</div>
    <div class="g3 alignselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
```

modele grilles-25.css

```
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,auto);
  height: 150px;
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  border: 1px dashed black;
}
.g1{background-color: #ED8;} /*Jaune*/
.g2{background-color: #4DA; height: 50px} /*Vert*/
.g3{background-color: #4AD;} /*Bleu*/
.g4{background-color: #A4D; height: 50px}/*Violet*/
.g5{background-color: #C24;}/*Rouge*/

.alignitemsstretch{align-items: stretch;}
.alignitemsstart{align-items: start;}
.alignitemscenter{align-items: center;}

.alignselfstart{align-self: start;}
.alignselfcenter{align-self: center;}
.alignselfend{align-self: end;}

.aligncontentstart{align-content: start;}
.aligncontentend{align-content: end;}
.aligncontentcenter{align-content: center;}
.aligncontentstretch{align-content: stretch;}
.aligncontentevenly{align-content: space-evenly;}
```

Cascade Style Sheets File | length : 1.096 lines : 39 | Ln : 7 Col : 3 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



Ici, on définit des conteneurs de 150px de haut et une hauteur auto pour nos rangées qui vont donc adapter leur hauteur à leur contenu.

J'utilise la valeur auto ici car c'est la seule valeur que align-content : stretch va pouvoir surcharger et car je sais que la hauteur de mes éléments est inférieure à la hauteur du conteneur et qu'on pourra donc bien voir l'effet de la propriété align-content.

J'ai également conservé les propriétés d'alignement des éléments dans les pistes et les éléments de différentes tailles afin que vous puissiez bien voir l'effet de chaque propriété.

Je vous conseille ici d'inspecter vos conteneurs avec les outils pour développeur de votre navigateur afin de bien voir la hauteur et l'alignement de nos différentes pistes.

Aligner les éléments ou les pistes d'une grille par rapport à l'axe en ligne

De façon similaire à l'alignement selon l'axe de bloc, nous allons pouvoir gérer l'alignement ou plus exactement la justification selon l'axe en ligne des éléments et des pistes en utilisant trois propriétés qui sont justify-items, justify-self et justify-content.

La propriété justify-items va nous permettre de contrôler l'alignement de tous les éléments dans le conteneur selon l'axe en ligne.

Nous allons devoir appliquer cette propriété à notre élément grille conteneur et allons pouvoir lui passer les mêmes valeurs qu'à la propriété align-items à savoir :

- stretch : Valeur par défaut. Les éléments de grille vont s'étirer pour occuper toute la largeur des cellules dans lesquelles ils sont placés ;
- start : Les éléments vont se placer au début du conteneur selon l'axe en ligne (à gauche pour un langage qui va de gauche à droite) ;
- end : Les éléments vont se placer à la fin du conteneur selon l'axe en ligne (à droite pour un langage qui va de gauche à droite) ;
- center : Les éléments vont être centrés dans le conteneur selon l'axe en ligne ;
- baseline : La ligne de base des éléments va être alignée.

D:\Data\Cours\ISL\HTML CSS\MonSite\html\modele grilles-26.html - Notepad++

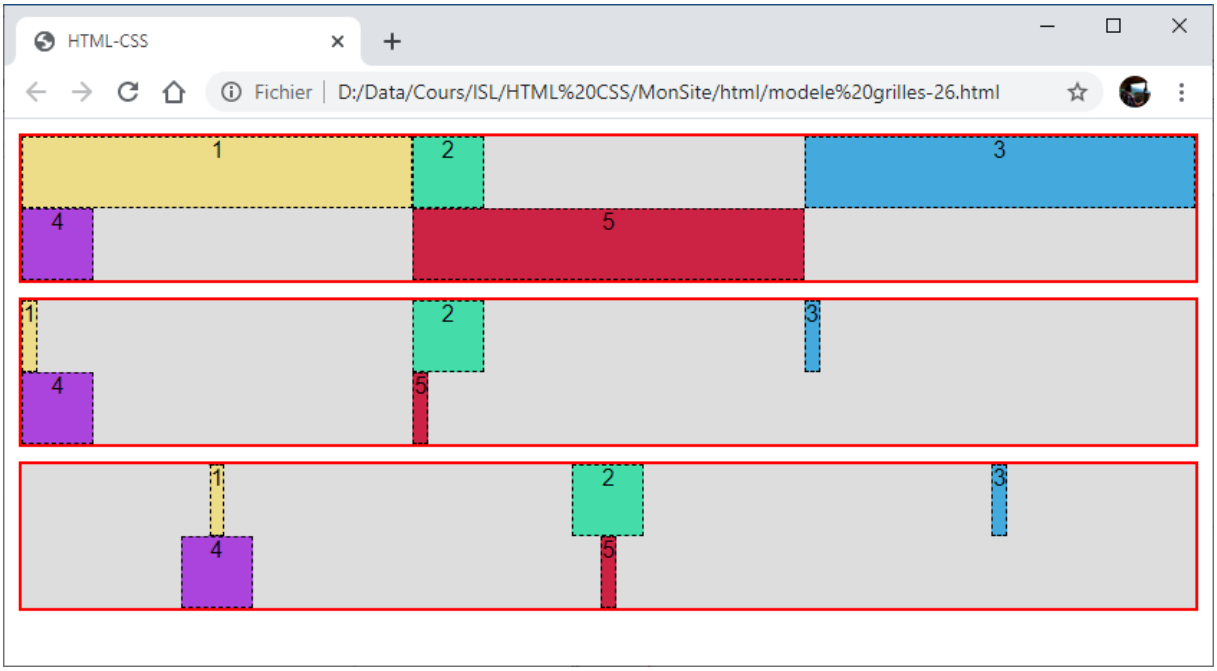
modele grilles-26.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-26.css">
</head>
<body>

<div class="conteneur-grid justifyitemsstretch">
    <div class="g1">1</div>
    <div class="g2">2</div>
    <div class="g3">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifyitemsstart">
    <div class="g1">1</div>
    <div class="g2">2</div>
    <div class="g3">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifyitemscenter">
    <div class="g1">1</div>
    <div class="g2">2</div>
    <div class="g3">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-26.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,50px);
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  border: 1px dashed black;
}
.g1{background-color: #ED8;} /*Jaune*/
.g2{background-color: #4DA; width: 50px} /*Vert*/
.g3{background-color: #4AD;} /*Bleu*/
.g4{background-color: #A4D; width: 50px}/*Violet*/
.g5{background-color: #C24;}/*Rouge*/

.justifyitemsstretch{justify-items: stretch;}
.justifyitemsstart{justify-items: start;}
.justifyitemscenter{justify-items: center;}
```

Hyper Text Markup Language file | length : 764 | lines : 34 | Ln : 2 | Col : 7 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



Cette fois-ci, afin de bien vous montrer l'effet des propriétés d'alignement dans l'axe en ligne, je définis des éléments de grille de largeurs diverses. Notez qu'une nouvelle fois la valeur stretch ne va pas pouvoir surcharger une largeur définie explicitement.

La propriété justify-self va elle nous permettre de contrôler l'alignement selon l'axe en ligne d'un élément en particulier. Nous allons appliquer cette propriété à l'élément de grille qu'on souhaite aligner et allons pouvoir lui passer les mêmes valeurs qu'à justify-items.

D:\Data\Cours\ISL\HTML CSS\MonSite\css\modele grilles-27.css - Notepad++

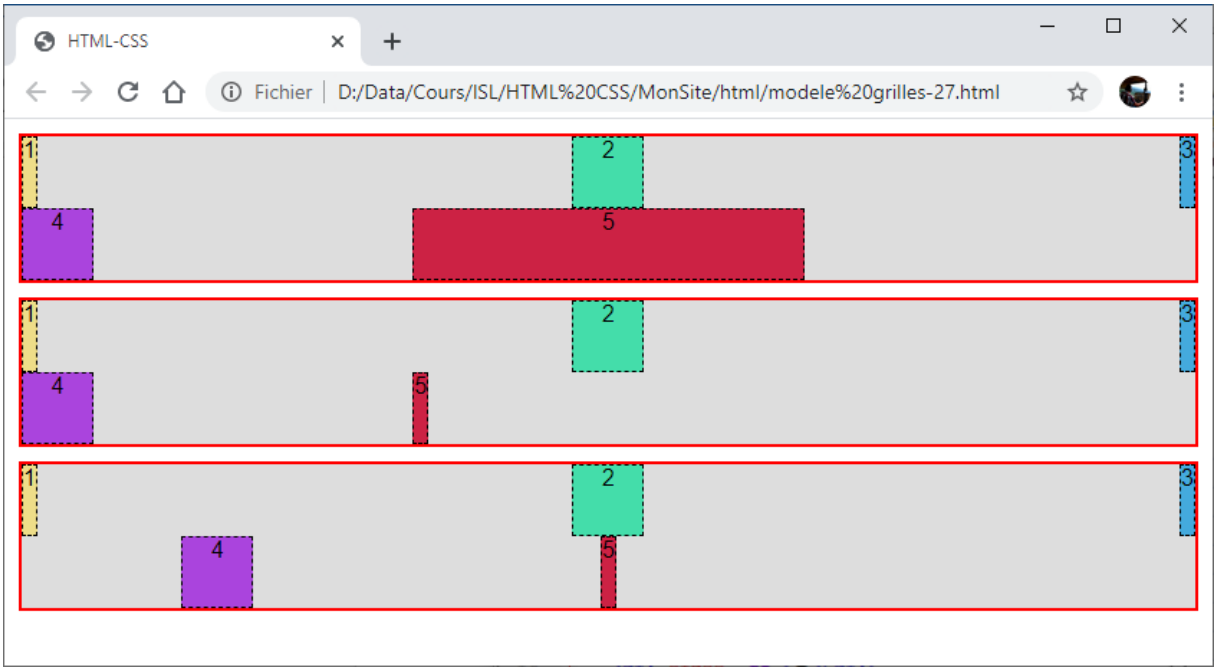
modele grilles-27.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-27.css">
</head>
<body>

<div class="conteneur-grid justifyitemsstretch">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifyitemsstart">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifyitemscenter">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-27.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,1fr);
  grid-template-rows: repeat(2,50px);
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.conteneur-grid > div{
  border: 1px dashed black;
}
.g1{background-color: #ED8;} /*Jaune*/
.g2{background-color: #4DA; width: 50px} /*Vert*/
.g3{background-color: #4AD;} /*Bleu*/
.g4{background-color: #A4D; width: 50px}/*Violet*/
.g5{background-color: #C24;}/*Rouge*/

.justifyitemsstretch{justify-items: stretch;}
.justifyitemsstart{justify-items: start;}
.justifyitemscenter{justify-items: center;}

.justifyselfstart{justify-self: start;}
.justifyselfcenter{justify-self: center;}
.justifyselfend{justify-self: end;}
```

Cascade Style Sheets File | length : 873 | lines : 32 | Ln : 7 | Col : 3 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



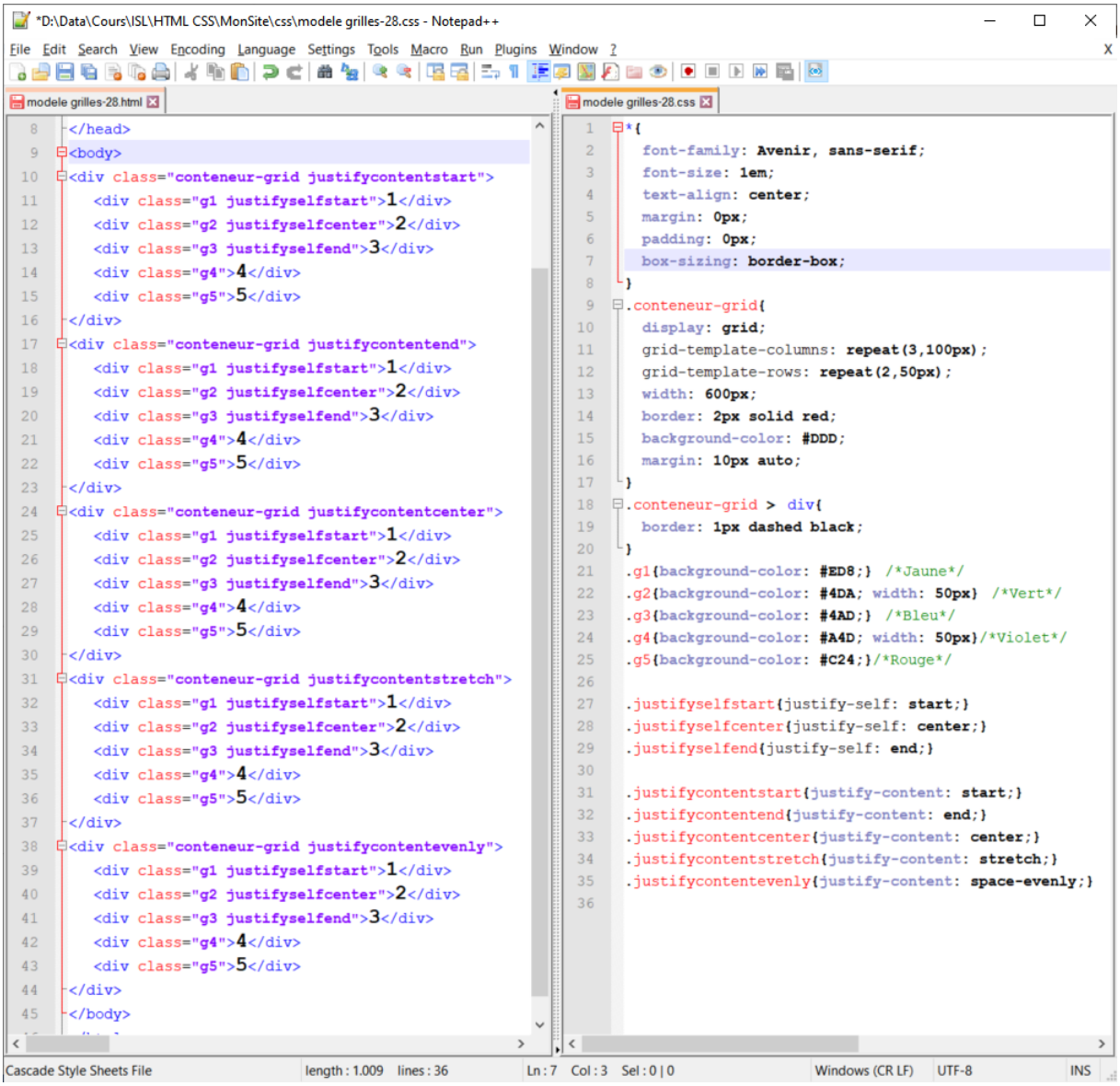
En, cas de conflit avec justify-items, la valeur de justify-self va être prioritaire puisque la sélection est plus précise.

Finalement, si les pistes d'une grille n'occupent pas tout l'espace en largeur dans le conteneur, alors nous allons pouvoir les aligner c'est-à-dire choisir comment distribuer l'espace restant selon l'axe en ligne.

Nous allons pouvoir faire cela en utilisant la propriété justify-content qu'on va devoir appliquer à l'élément grille conteneur. Nous allons pouvoir lui passer les mêmes valeurs qu'à la propriété align-content à savoir :

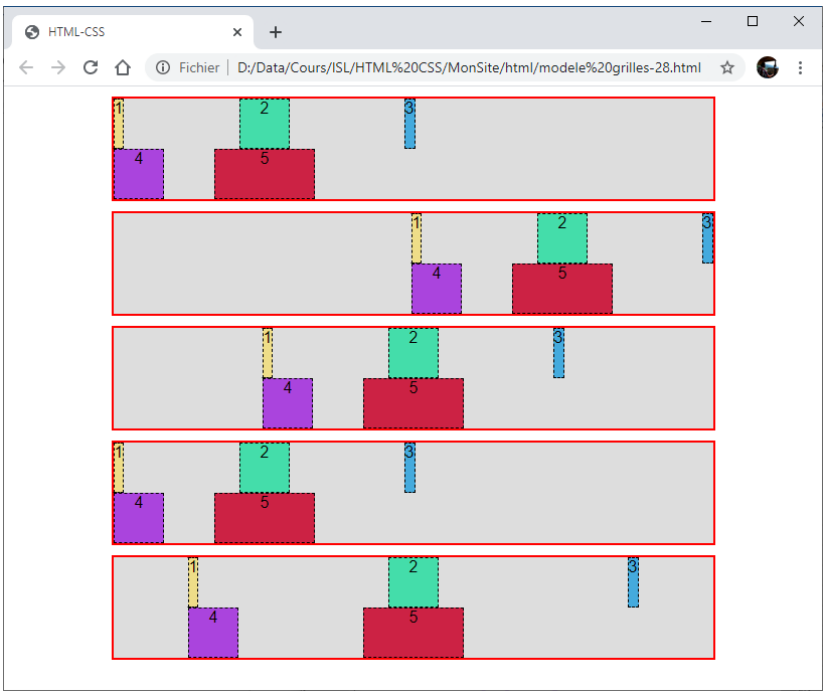
- stretch : Les pistes vont s'étendre pour occuper toute la largeur disponible dans le conteneur;
- start : Les pistes vont se grouper en début de conteneur (à gauche pour un langage dont l'écriture se fait de gauche à droite) ;
- end : Les pistes vont se grouper en fin de conteneur (à droite pour un langage dont l'écriture se fait de gauche à droite) ;
- center : Les pistes vont se grouper au centre du conteneur selon l'axe en ligne ;
- baseline : Les pistes vont s'aligner de telle sorte à ce que leurs lignes de bases soient alignées selon l'axe en ligne ;
- space-between : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes selon l'axe en ligne. La première et la dernière pistes vont être collées contre les bords du conteneur ;

- space-around : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes selon l'axe en ligne. L'espace entre les bords du conteneur et la première et la dernière piste sera deux fois moins important qu'entre deux pistes ;
- space-evenly : L'espace disponible dans le conteneur va être reparti équitablement entre les différentes pistes et entre une piste et un bord du conteneur selon l'axe en ligne.




```
</head>
<body>
<div class="conteneur-grid justifycontentstart">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifycontentend">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifycontentcenter">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifycontentstretch">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid justifycontentevenly">
    <div class="g1 justifyselfstart">1</div>
    <div class="g2 justifyselfcenter">2</div>
    <div class="g3 justifyselfend">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
</body>
```

```
*{
    font-family: Avenir, sans-serif;
    font-size: 1em;
    text-align: center;
    margin: 0px;
    padding: 0px;
    box-sizing: border-box;
}
.conteneur-grid{
    display: grid;
    grid-template-columns: repeat(3,100px);
    grid-template-rows: repeat(2,50px);
    width: 600px;
    border: 2px solid red;
    background-color: #DDD;
    margin: 10px auto;
}
.conteneur-grid > div{
    border: 1px dashed black;
}
.g1{background-color: #ED8;} /*Jaune*/
.g2{background-color: #4DA; width: 50px} /*Vert*/
.g3{background-color: #4AD;} /*Bleu*/
.g4{background-color: #A4D; width: 50px}/*Violet*/
.g5{background-color: #C24;}/*Rouge*/

.justifyselfstart{justify-self: start;}
.justifyselfcenter{justify-self: center;}
.justifyselfend{justify-self: end;}

.justifycontentstart{justify-content: start;}
.justifycontentend{justify-content: end;}
.justifycontentcenter{justify-content: center;}
.justifycontentstretch{justify-content: stretch;}
.justifycontentevenly{justify-content: space-evenly;}
```



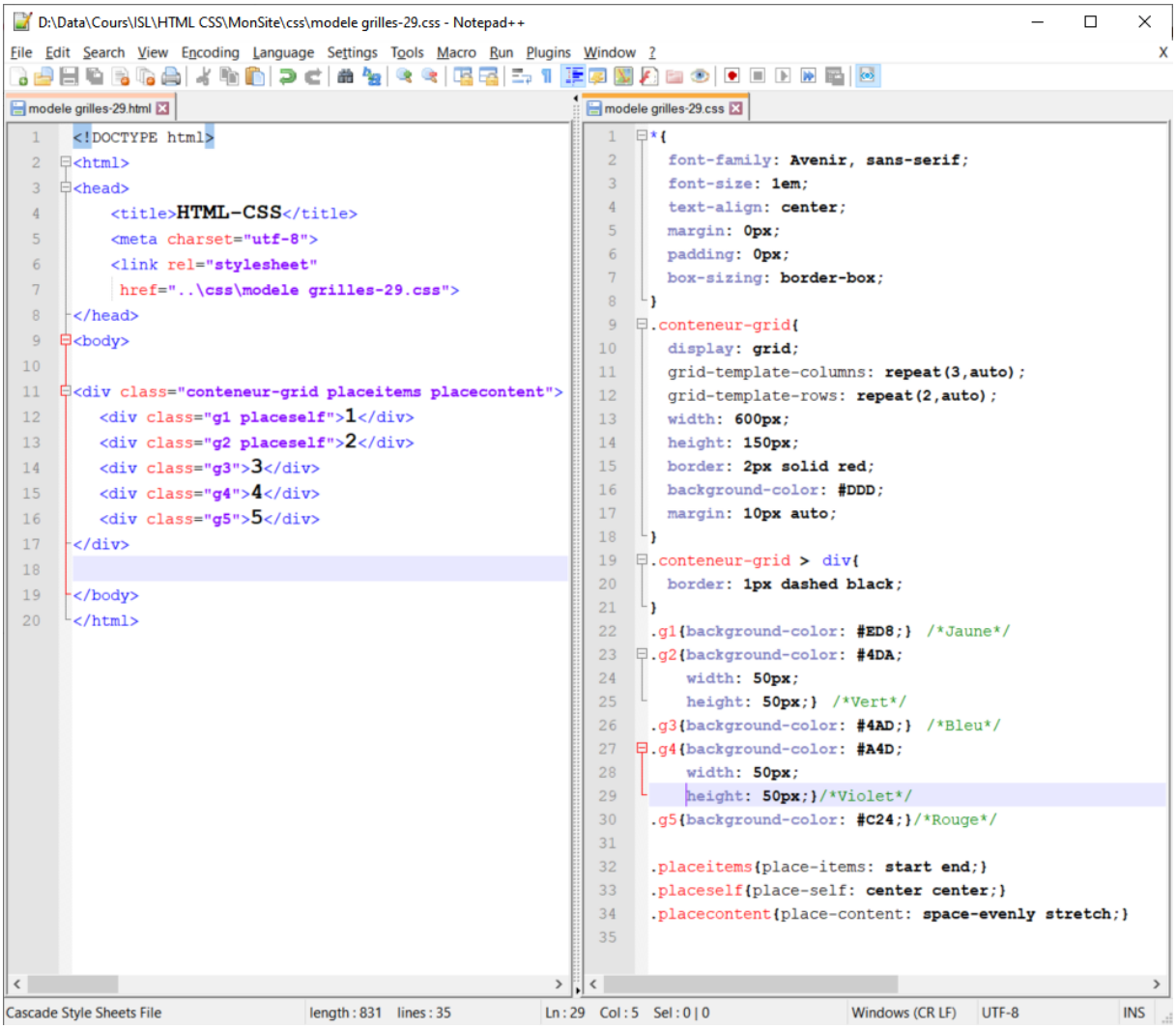
Pour voir l'effet de justify-content, il faut que les pistes de notre grille n'occupent pas toute la largeur du conteneur. Pour cela, on définit ici une grille de 3 colonnes faisant chacune 100px de large ainsi qu'une taille fixe de 600px pour le conteneur.

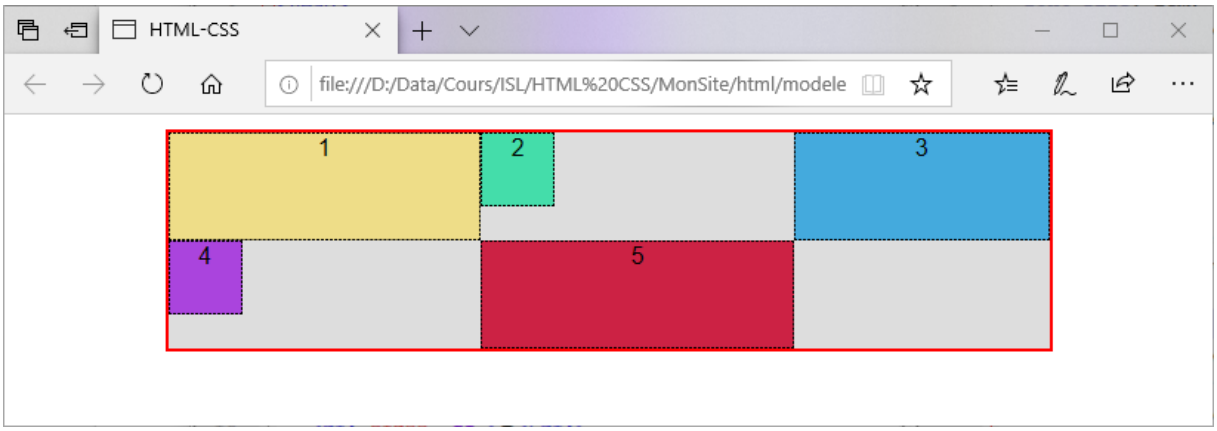
Ensuite, on utilise justify-content pour distribuer l'espace restant dans le conteneur. La valeur start va regrouper nos différentes colonnes au début du conteneur par exemple, c'est-à-dire à gauche pour une écriture de gauche à droite.

Propriétés d'alignement raccourcies

On va également pouvoir préciser l'alignement des éléments ou des pistes sur les deux axes en une seule fois grâce aux propriétés d'alignement raccourcies suivantes :

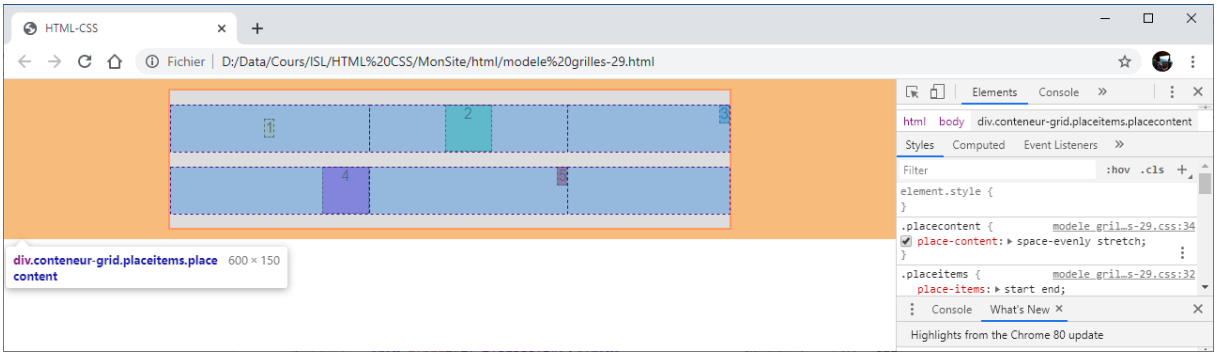
- place-items : notation raccourcie de align-items et justify-items ;
- place-self : notation raccourcie de align-self et justify-self ;
- place-content : notation raccourcie de align-content et justify-content.





Ici, on définit une grille de 3 colonnes et 2 rangées avec des tailles auto. Notre conteneur de grille a des dimensions de 600 * 150px.

Les éléments vont être placés en haut à droite dans leur cellule grâce à place-items: start end à l'exception des éléments auxquels on applique place-self: center center qui vont eux être centrés. L'espace en hauteur va être réparti équitablement entre les pistes tandis que l'espace en largeur va être absorbé grâce à la valeur stretch.



Ci-dessus, j'ai affiché le découpage de ma grille et donc les pistes (délimitées par des tirets noirs) en utilisant les outils pour développeurs de Chrome (clic droit > inspecter l'élément) afin que vous puissiez bien voir où sont alignés les éléments dans les pistes et les pistes dans le conteneur.

Définir les tailles des gouttières pour espacer les éléments de grille

Finalement, le modèle des grilles nous offre des propriétés pour éloigner les éléments de grille les uns des autres sans que les éléments ne s'éloignent des bords du conteneur de la grille.

On appelle cet espace entre les éléments des gouttières. La dimension des gouttières des cellules d'une grille va pouvoir être définie à l'aide des propriétés column-gap et row-gap ou avec la propriété raccourcie gap. Il faudra appliquer ces propriétés au conteneur.

D:\Data\Cours\ISL\HTML CSS\MonSite\css\modele grilles-30.css - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

modele grilles-30.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>HTML-CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet"
     href="..\css\modele grilles-30.css">
</head>
<body>

<div class="conteneur-grid">
    <div class="g1">1</div>
    <div class="g2">2</div>
    <div class="g3">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>
<div class="conteneur-grid grille2">
    <div class="g1">1</div>
    <div class="g2">2</div>
    <div class="g3">3</div>
    <div class="g4">4</div>
    <div class="g5">5</div>
</div>

</body>
</html>
```

modele grilles-30.css

```css
*{
  font-family: Avenir, sans-serif;
  font-size: 1em;
  text-align: center;
  margin: 0px;
  padding: 0px;
  box-sizing: border-box;
}
.conteneur-grid{
  display: grid;
  grid-template-columns: repeat(3,auto);
  grid-template-rows: repeat(2,auto);
  row-gap: 5px;
  column-gap: 10px;
  height: 150px;
  border: 2px solid red;
  background-color: #DDD;
  margin: 10px;
}
.grille2{
    gap: 5px 10px; /*row-gap column-gap*/
}
.conteneur-grid > div{
  border: 1px dashed black;
}
.g1{grid-row: 1 / 4;
    background-color: #ED8;} /*Jaune*/
.g2{grid-column: 2;
    background-color: #4DA;} /*Vert*/
.g3{grid-column: 2;
    background-color: #4AD;} /*Bleu*/
.g4{grid-column: 2;
    background-color: #A4D;}/*Violet*/
.g5{grid-column: 3;
    grid-row: 1 / 4;
    background-color: #C24;}/*Rouge*/
```

Cascade Style length : 831 lines : 36 Ln : 36 Col : 38 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Comme vous pouvez le voir, les gouttières sont des espaces qui n'apparaissent qu'entre deux éléments. On peut ainsi espacer les éléments de grille les uns des autres sans les éloigner des bords du conteneur de grille.