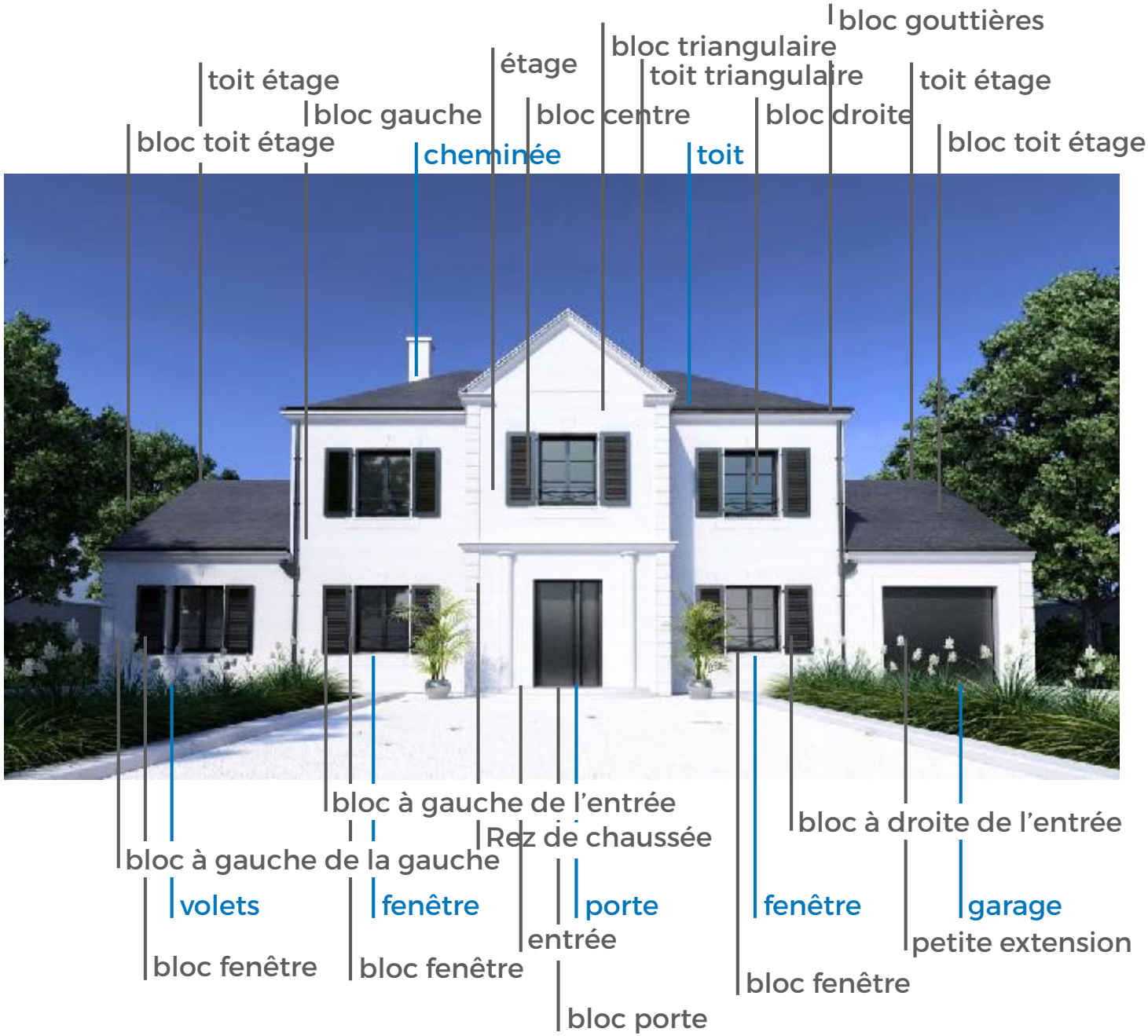
Flex & Grid

ou comment mettre en forme nos pages web

Ce que nous faisons trop souvent

il serait temps de changer


bloc gouttières
bloc triangulaire
étage
toit triangulaire
toit étage
toit étage
bloc gauche
bloc centre
bloc droite
bloc toit étage
bloc toit étage
cheminée
toit
bloc à gauche de l'entrée
bloc à droite de l'entrée
Rez de chaussée
bloc à gauche de la gauche
volets
fenêtre
porte
fenêtre
garage
entrée
petite extension
bloc fenêtre
bloc fenêtre
bloc fenêtre
bloc porte

```
<div class="header_top_bottom_holder">
    <div class="header_top clearfix" style="background-color:rgba(255, 255, 255, 1);">
        <div class="left">
            <div class="inner">
                <div class="header-widget widget_text_icl header-left-widget">
                    <div class="textwidget">
                        <div style="text-align: center; padding-top:165px;">
                            <h1 class="maptitle" style="color:#fff">Boite dans le monde</h1>
                            <a itemprop="url" href="http://www.lesaffre.com/" target="_blank"
class="qbutton  center default">Retrouvez nous sur monsite.com</a></div>
                    </div>
                </div>
            </div>
        </div>
        <div class="right">
            <div class="inner">
            </div>
        </div>
    </div>
```

Oui mais...

ça marche quand même !

Et le client...

Adaptabilité ?

Accessibilité ?

Performance ?

Maintenabilité ?

Coût de non qualité ?

Soyons professionnel !

Historique

Les navigateurs <3 le CSS

Prise en compte parfois tardive

Spécifications pas toujours supportées

Des éléments/propriétés utilisés pour des mises en page

Des propriétés parfois mal aimées

Aujourd'hui

enfin depuis quelque temps déjà

Flexible box layout module

W3C Candidate Recommendation

tadaaaaam !

Mais qu’est-ce donc ?

diantre

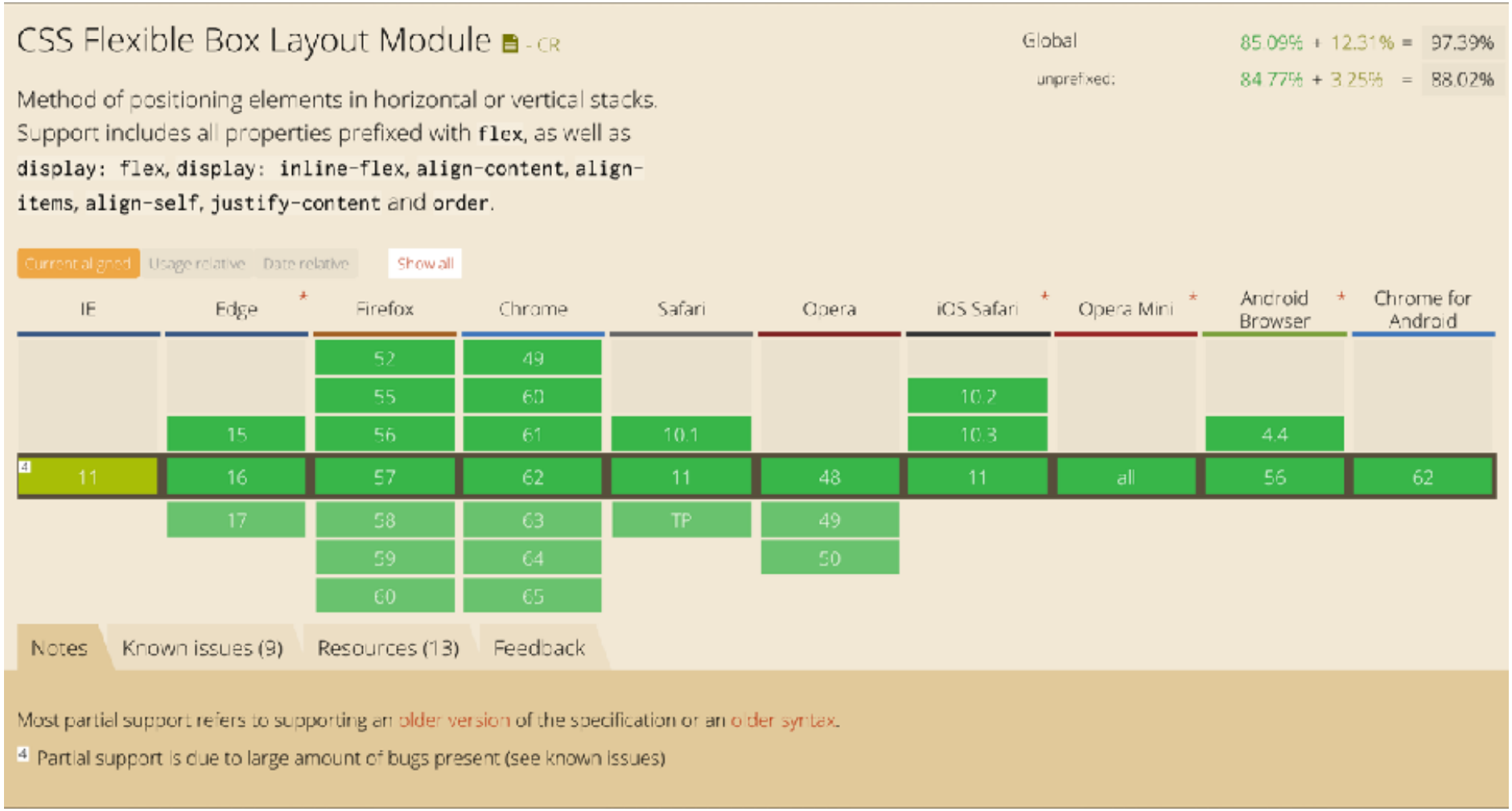
Un **module** de mise en forme **uni-directionnel**

Status **candidate recommendation**

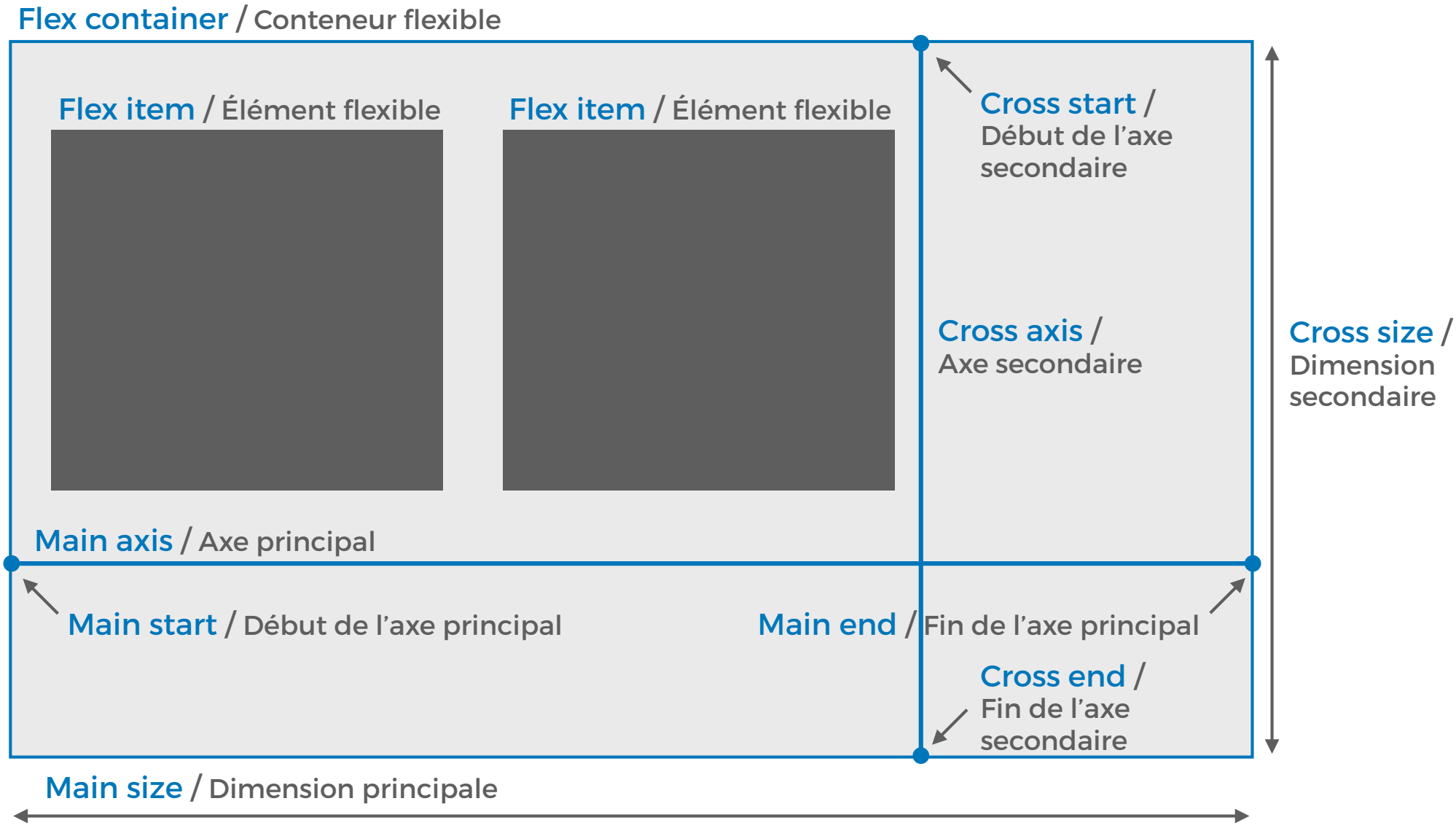
A son propre flux (**flex flow**)

Impact un **conteneur** et ses **enfants directs**

Un module supporté



Can I Use https://caniuse.com/#feat=flexbox


Flex container / Conteneur flexible
Flex item / Élément flexible
Flex item / Élément flexible
Cross start / Début de l'axe secondaire
Cross axis / Axe secondaire
Cross size / Dimension secondaire
Main axis / Axe principal
Main start / Début de l'axe principal
Main end / Fin de l'axe principal
Cross end / Fin de l'axe secondaire
Main size / Dimension principale

Propriétés du conteneur

display: flex | inline-flex;

Déclaration de l'élément en flexible

flex (type bloc)

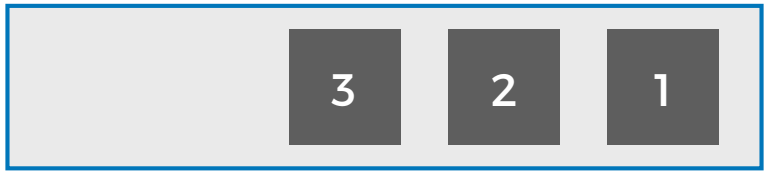
inline-flex (type inline)

flex-direction

Défini l'axe principal (main-axis) et sa direction

row*

row-reverse



column

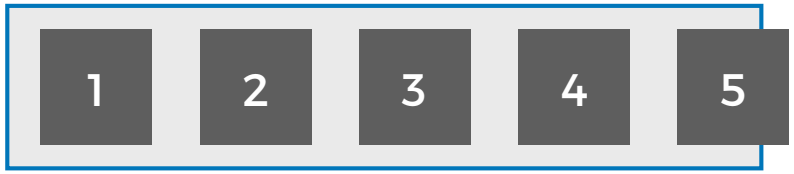
column-reverse

* valeur par défaut

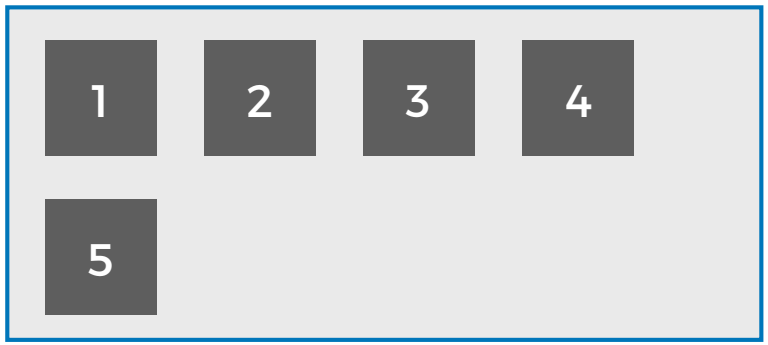
flex-wrap

Les flex items restent sur une ligne ou non

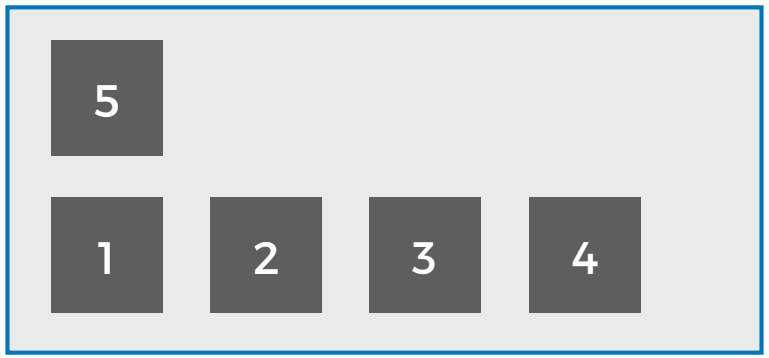
nowrap*



wrap



wrap-reverse



* valeur par défaut

flex-flow

Propriété raccourcie de flex-direction et flex-wrap

flex-flow: <flex-direction> <flex-wrap>;

justify-content

Répartition des espaces entre et autour des éléments flexibles sur l'axe principal (main-axis)

flex-start*

flex-end

space-between

space-around

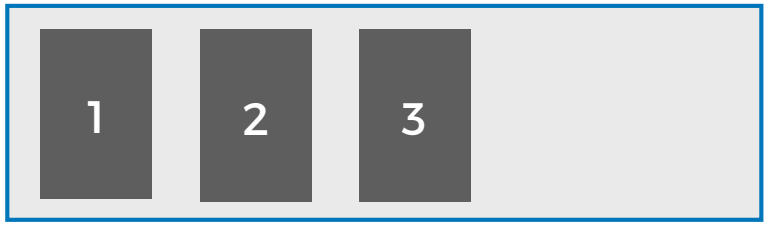
center

* valeur par défaut

align-items

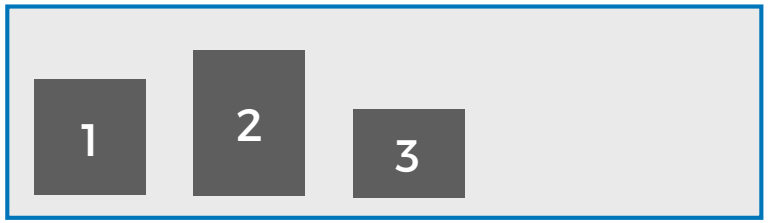
Répartition des espaces autour des éléments flexibles sur l'axe secondaire (cross-axis)

normal/stretch*



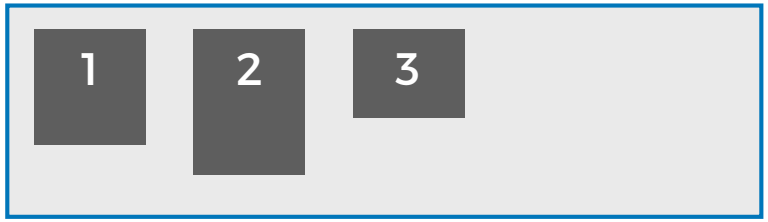
flex-start

flex-end



center

baseline

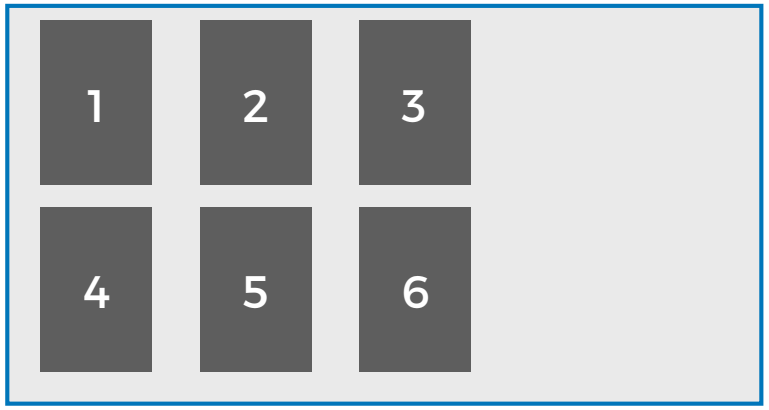


* valeur par défaut

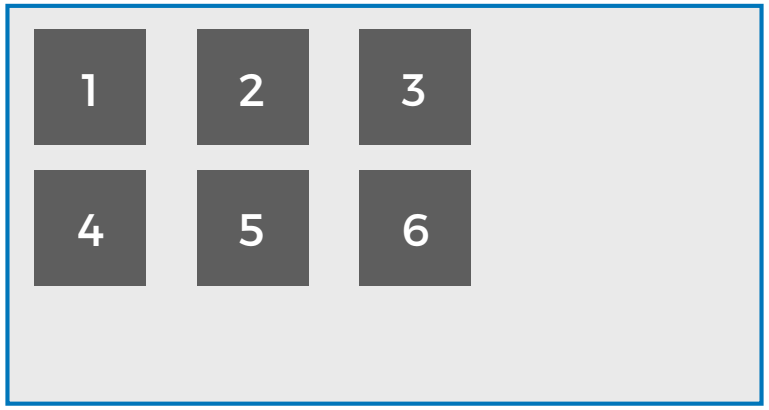
align-content

Répartition des espaces autour et entre des éléments flexibles sur l'axe secondaire (cross-axis)

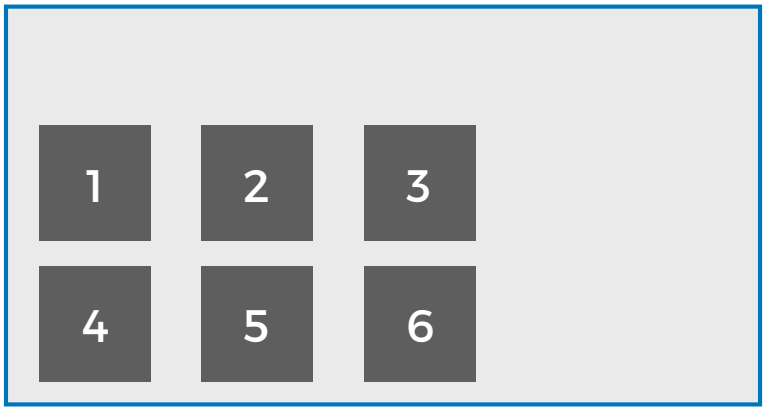
normal/stretch*



flex-start



flex-end

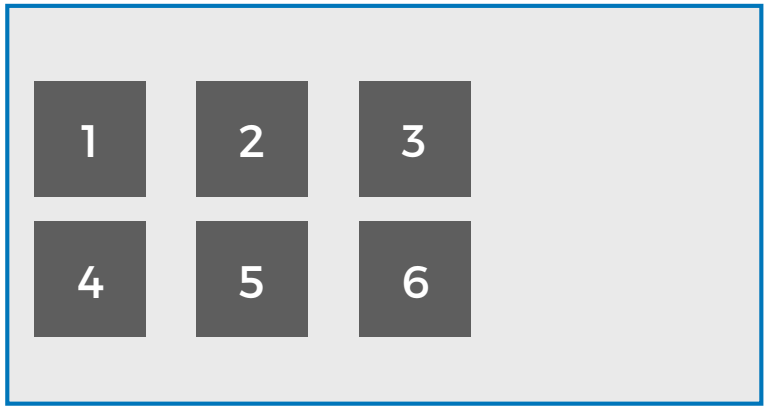


* valeur par défaut

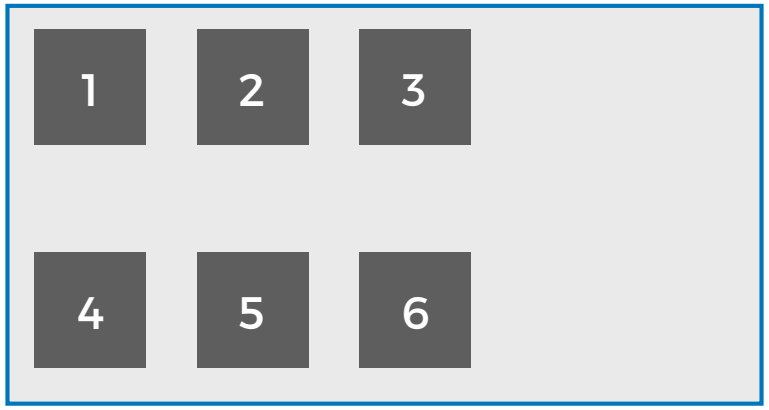
align-content

Répartition des espaces autour et entre des éléments flexibles sur l'axe secondaire (cross-axis)

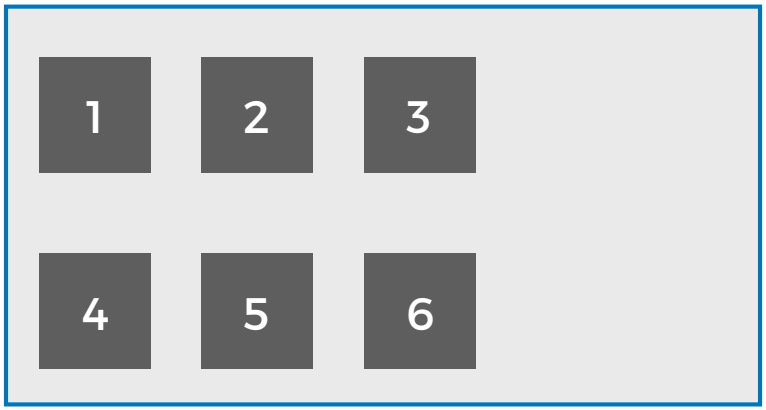
center



space-between



space-around

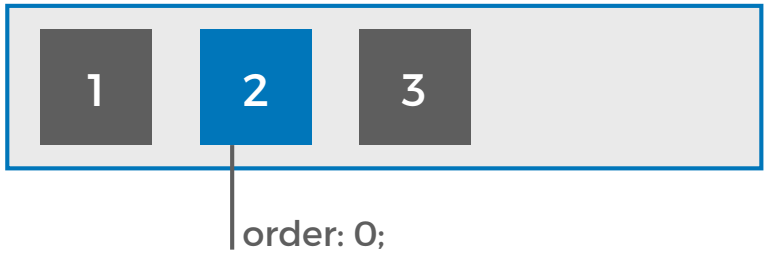


Propriétés des éléments flexibles

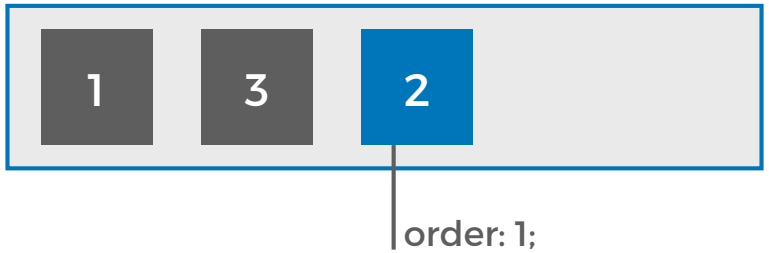
order

Défini l'orde d'apparition des éléments (nombre entier)

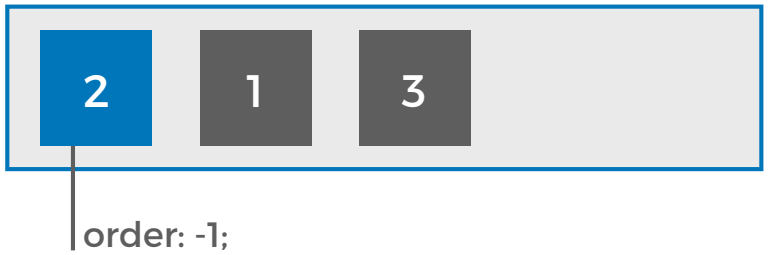
0*



n



-n



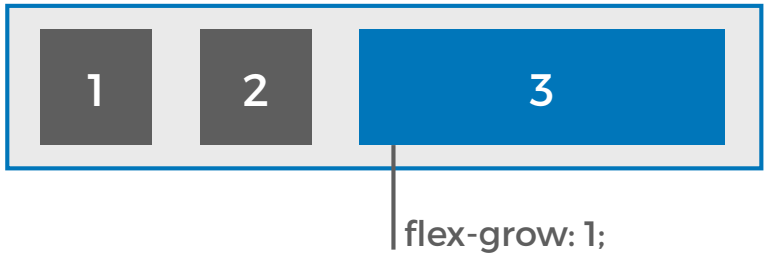
Attention à l'accessibilité des éléments

* valeur par défaut

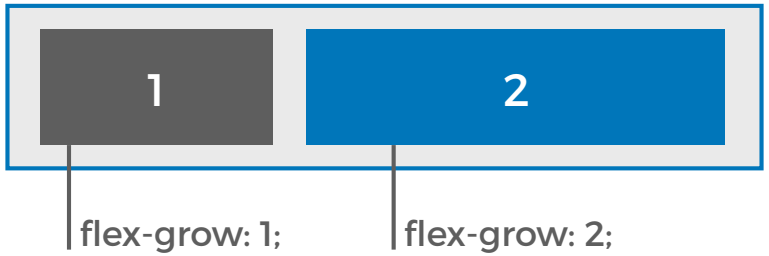
flex-grow

Indique le facteur de grossissement d'un élément flexible (nombre positif)

1



2



flex-shrink

Indique le facteur de rétrécissement d'un élément flexible (nombre positif)

flex-basis

Détermine la taille de l'élément flexible sur l'axe principal (main axis)

auto* (défini par la valeur de width ou height)

largeur

90%

200px

10rem

15vh

mots clés

content

min-content

max-content

flex

Propriété raccourcie de flex-grow, flex-shrink et flex-basis

flex: <flex-grow> <flex-shrink> <flex-basis>;

let’s go dude !

Une toolbar en flex

let’s go dude !

https://codepen.io/mynameistom/pen/MOdKJm

Aujourd'hui

enfin depuis quelque temps déjà

Grid layout module

W3C Candidate Recommendation

Mais qu’est-ce donc ?

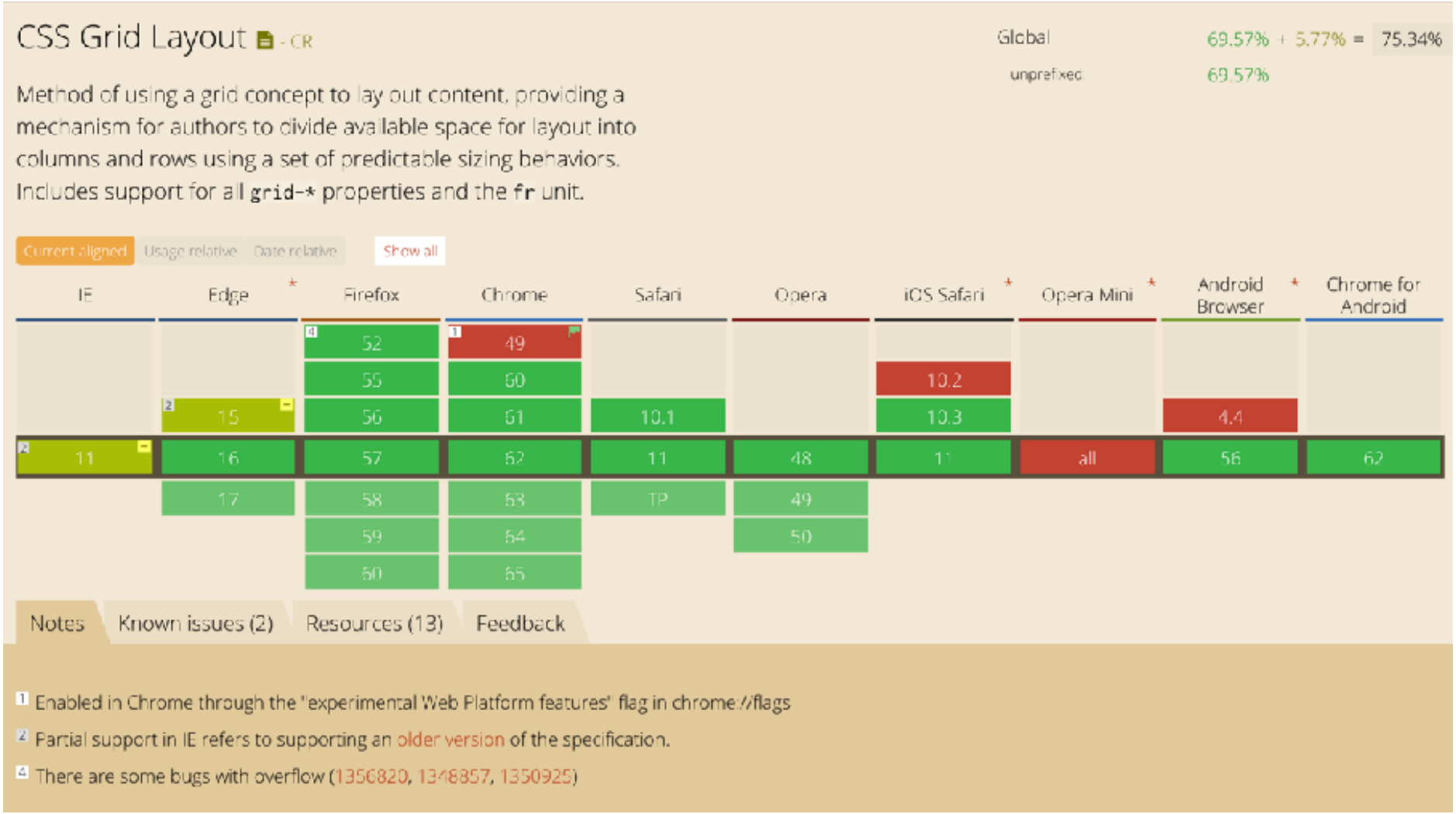
diantre

Un **module** de mise en forme **multi-directionnel**

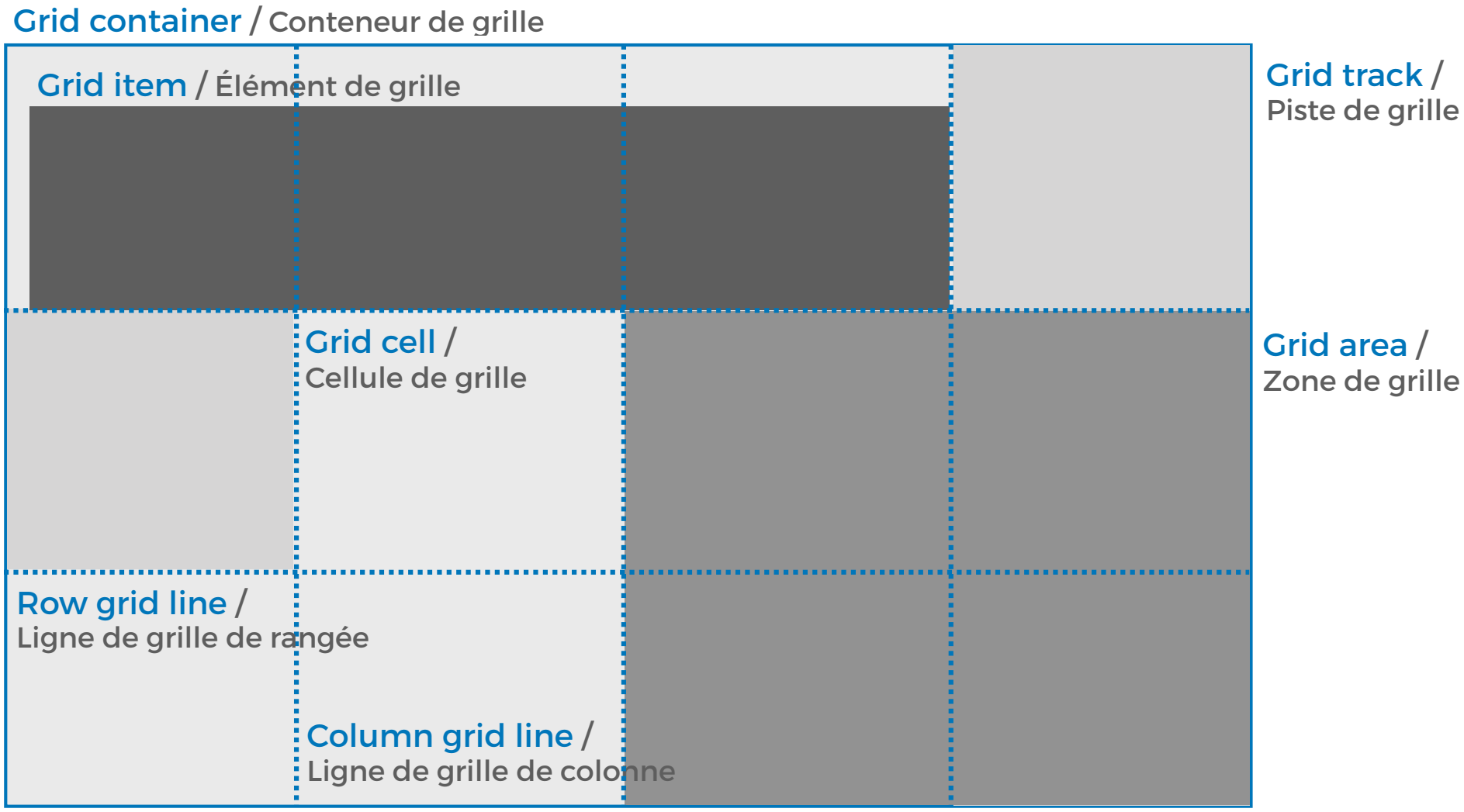
Status **candidate recommendation**

Impact un **conteneur** et ses **enfants directs**

Un module supporté



Can I Use https://caniuse.com/#feat=css-grid


Grid container / Conteneur de grille
Grid item / Élément de grille
Grid track /
Piste de grille
Grid cell /
Cellule de grille
Grid area /
Zone de grille
Row grid line /
Ligne de grille de rangée
Column grid line /
Ligne de grille de colonne

Propriétés de la grille

display: grid | inline-grid;

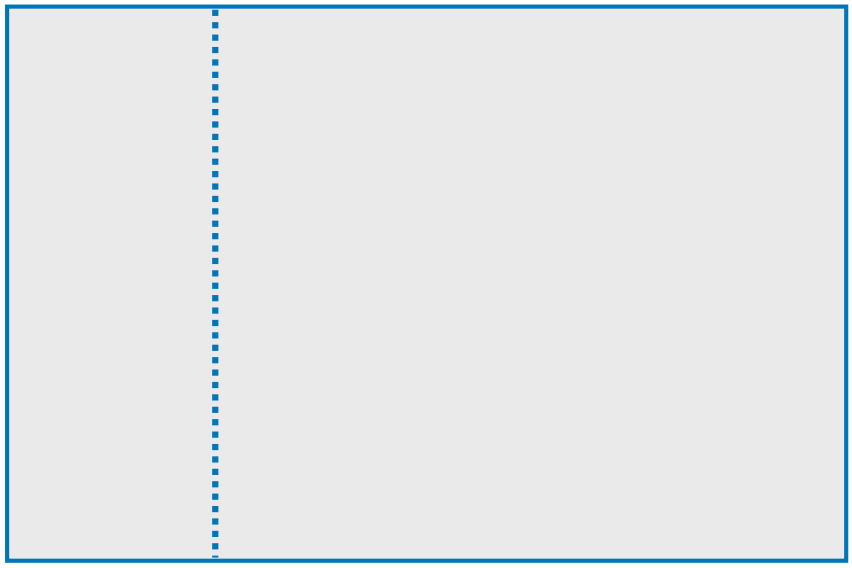
Déclaration de l'élément en grille

grid (type bloc)

inline-grid (type inline)

grid-template-columns

Défini le nom et les tailles des colonnes de la grille



none* (pas de grille explicite)

```
grid-template-columns: none;
```

<longueur> (px, %, em, fr, ...)

```
grid-template-columns: 100px;
```

[nom]

```
grid-template-columns: [start] 100px;
```

repeat()

```
grid-template-columns: repeat(3, 1fr);
```

minmax()

```
grid-template-columns: minmax(100px, 30%);
```

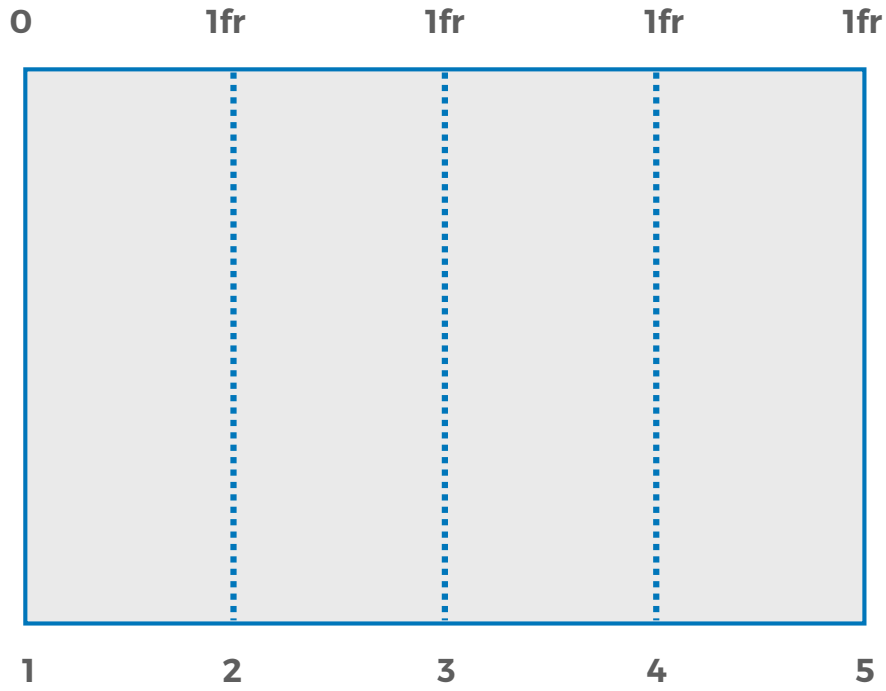
* valeur par défaut

grid-template-columns

grid-template-columns: [col1-start] 1fr [col1-end col2] 1fr [col3] 1fr 1fr;

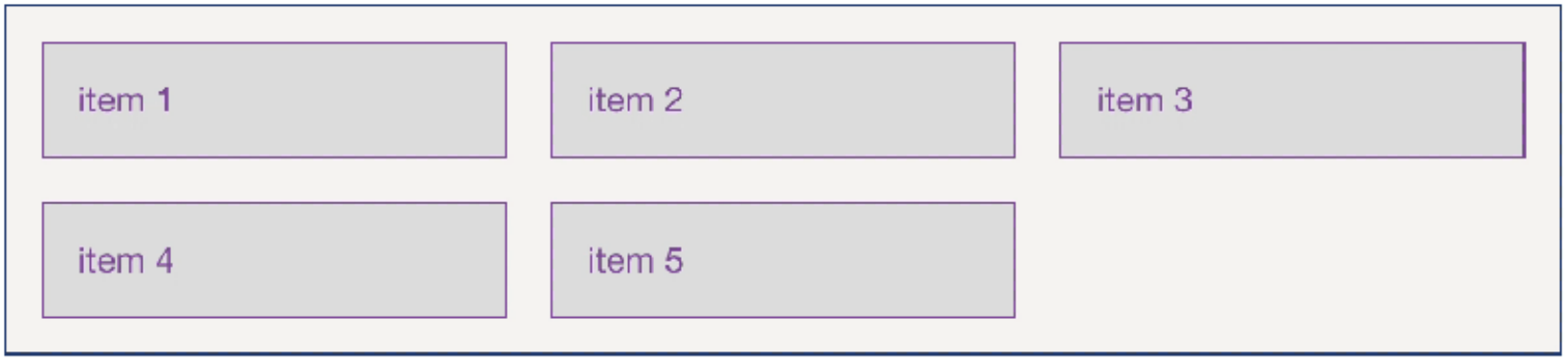
ou

grid-template-columns: repeat(4, 1fr);



grid-template-columns

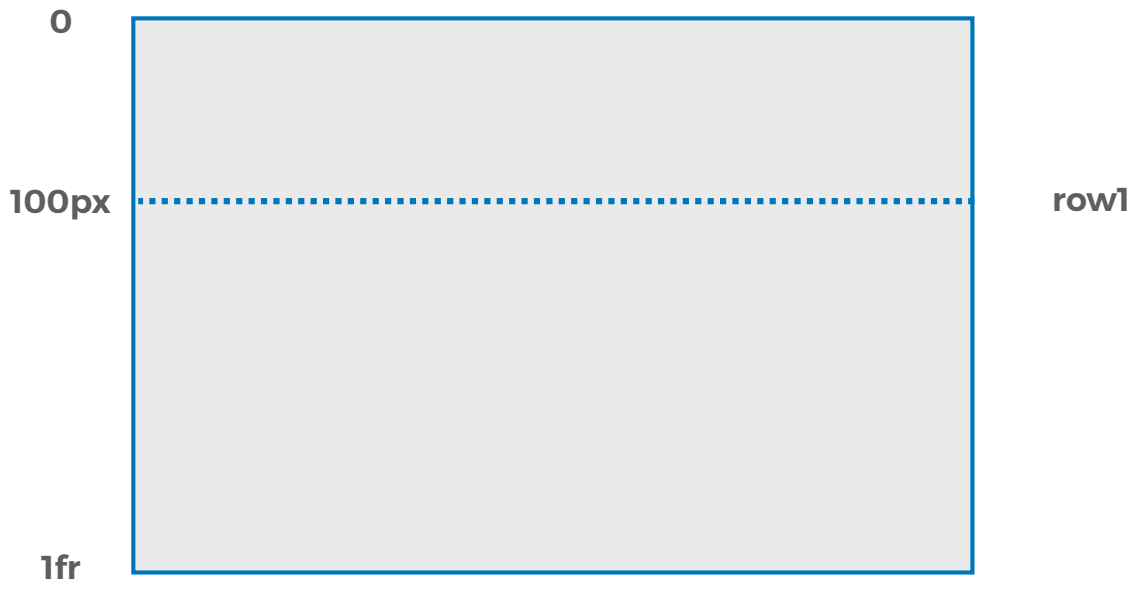
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr);



grid-template-rows

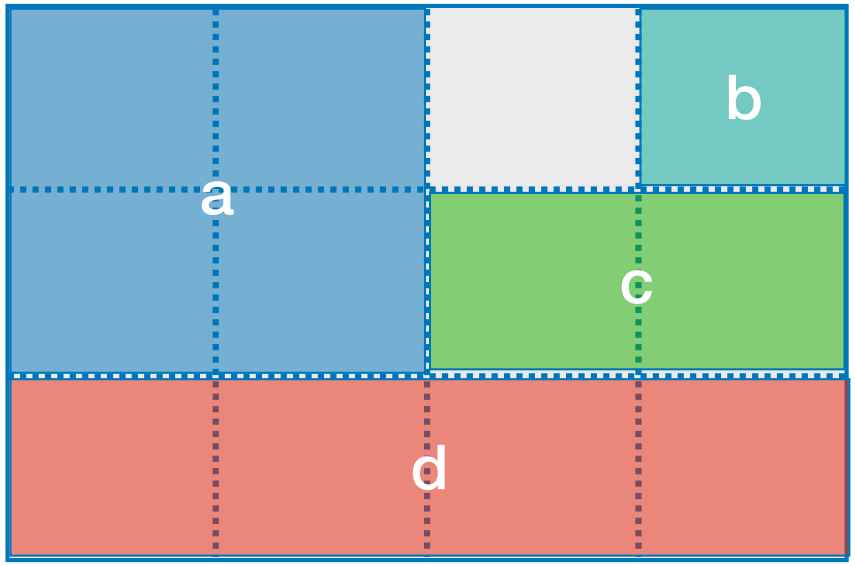
Défini le nom et les tailles des lignes de la grille

grid-template-rows: [row1] 100px minmax(100px, 1fr);



grid-template-areas

Nomme des zones de grilles

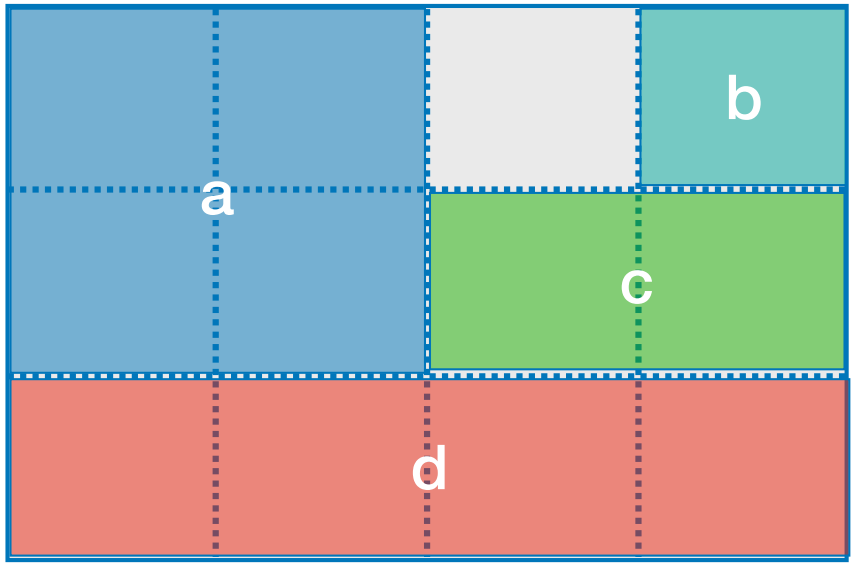



```
grid-template-columns: repeat(4, 1fr);
grid-template-rows: repeat(3, 1fr);
grid-template-areas: "a a . b"
                     "a a c c"
                     "d d d d";
```

grid-template

Propriété raccourcie de grid-template-columns,

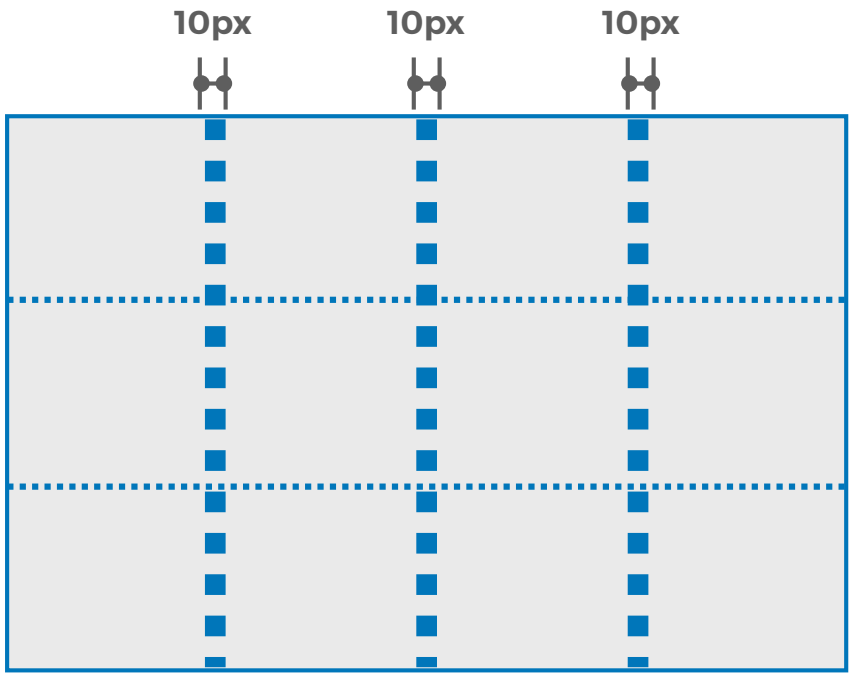
grid-template-rows et grid-template-areas




```
grid-template: "a a . b" 1fr
               "a a c c" 1fr
               "d d d d" 1fr
               / repeat(4, 1fr);
```

grid-column-gap

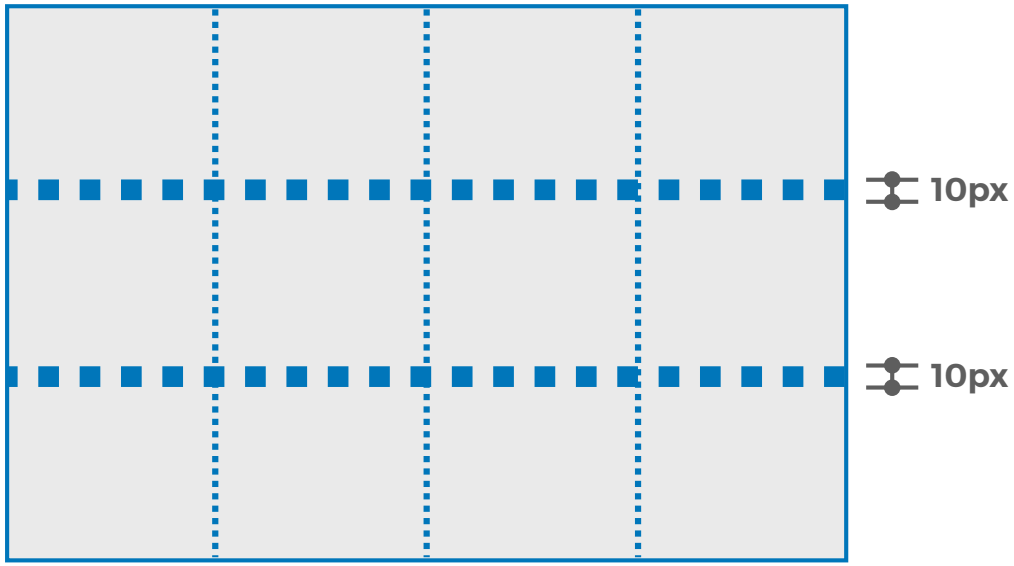
Défini l’espacement entre les colonnes de la grille



grid-column-gap: 10px;

grid-row-gap

Défini l’espacement entre les lignes de la grille



grid-row-gap: 10px;

grid-gap

Propriété raccourcie de grid-column-gap et grid-row-gap

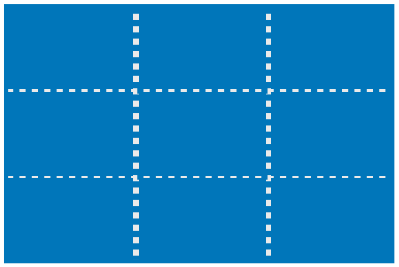
grid-gap: <grid-column-gap> <grid-row-gap>;

justify-content

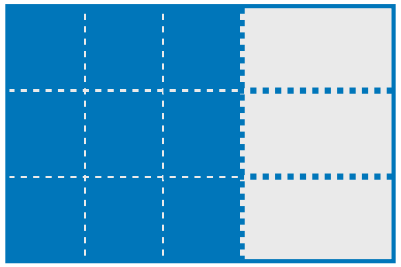
Alignement des colonnes de la grille dans le conteneur

(si celle-ci est inférieur au conteneur)

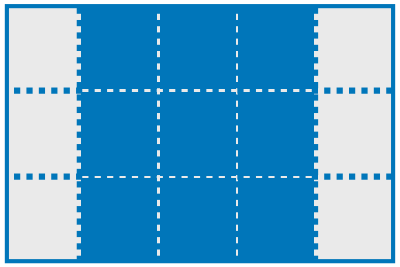
stretch



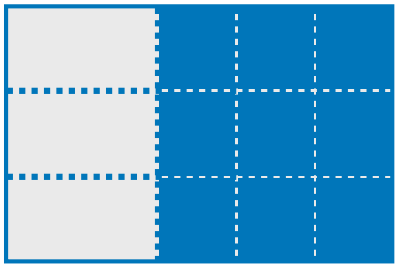
start



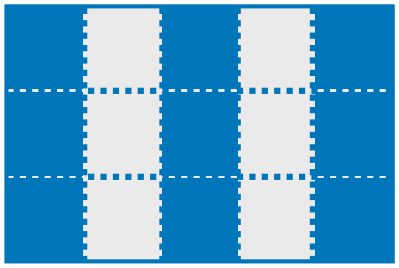
center



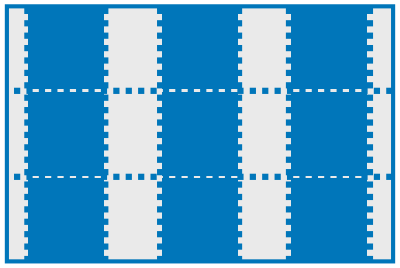
end



space-between



space-around/ space-evenly

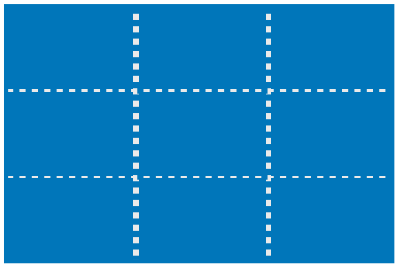


align-content

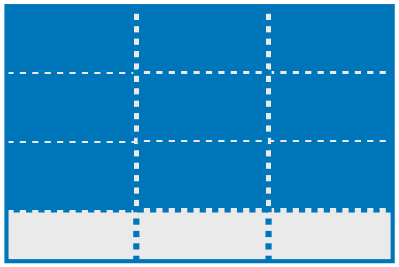
Alignement des lignes de la grille dans le conteneur

(si celle-ci est inférieur au conteneur)

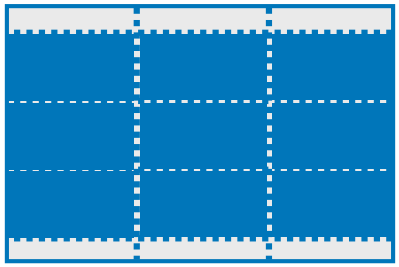
stretch



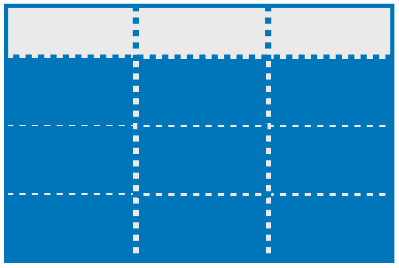
start



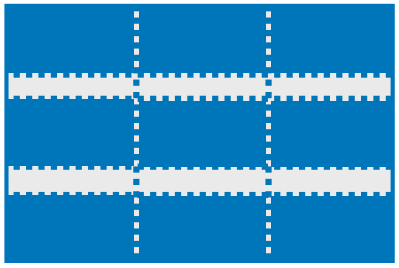
center



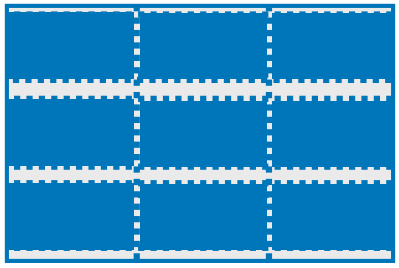
end



space-between



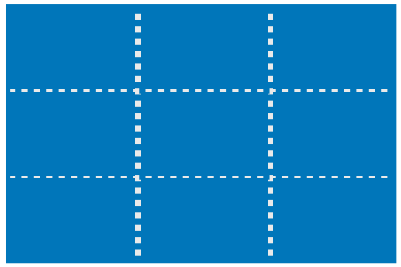
space-around/
space-evenly



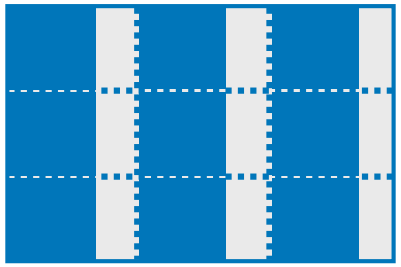
justify-items

Alignement des contenus sur les colonnes de la grille

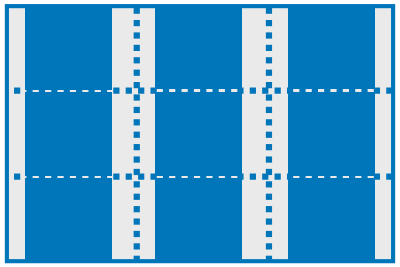
stretch*



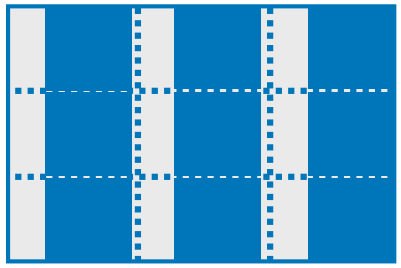
start



center



end

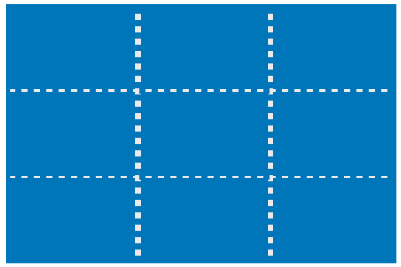


* valeur par défaut

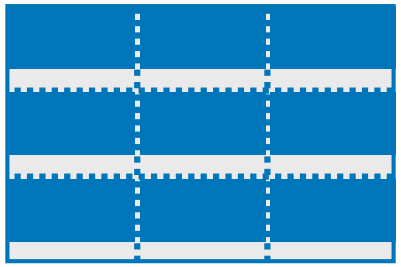
align-items

Alignement des contenus sur les lignes de la grille

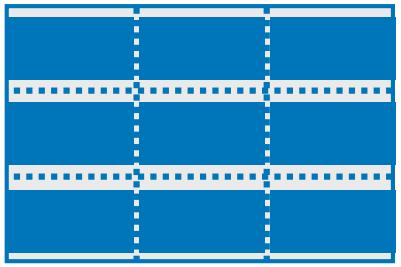
stretch*



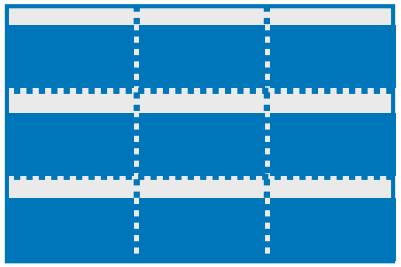
start



center



end

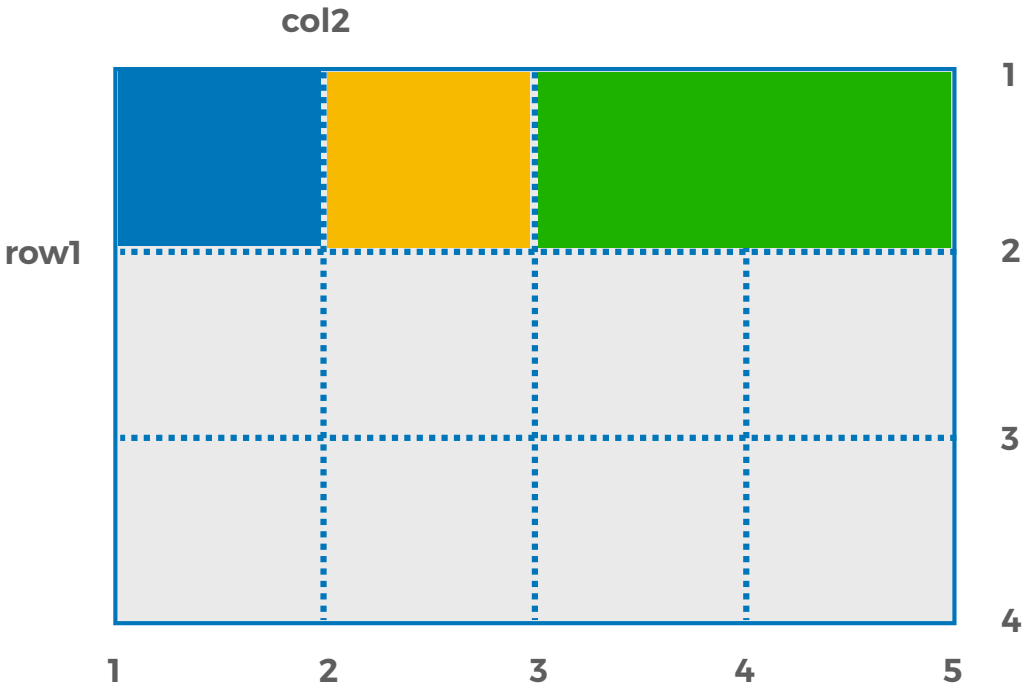


* valeur par défaut

Propriétés des éléments de grille

grid-column-start

Départ de l'item sur l'axe des colonnes



grid-column-start: 1;

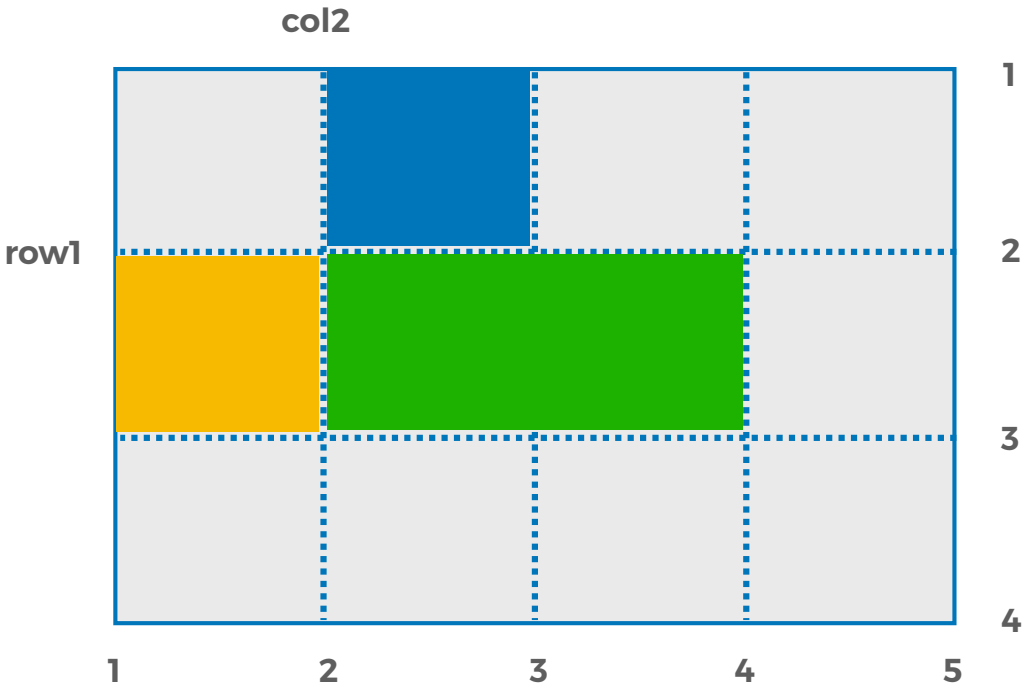
grid-column-start: col2;

grid-column-start: span 2;

s'étend de ...

grid-column-end

Arrivée de l'item sur l'axe des colonnes



grid-column-end: 3;

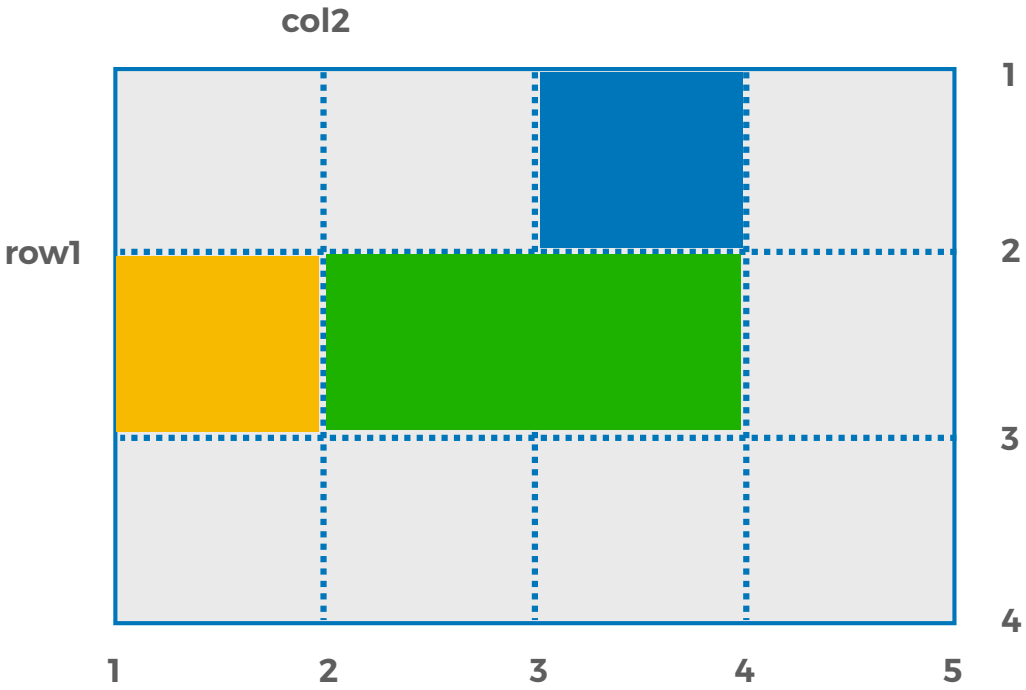
grid-column-end: col2;

grid-column-end: span 2;

grid-column

Taille et emplacement de l'item sur l'axe des colonnes

grid-column: <grid-column-start> / <grid-column-end>



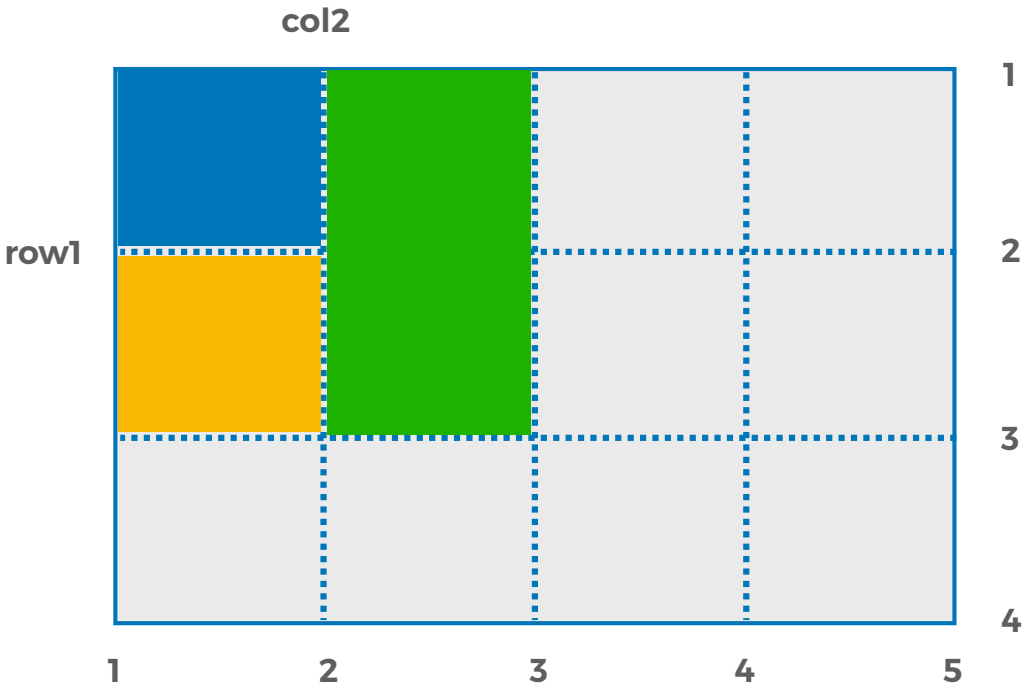
grid-column: 3;

grid-column: auto / col2;

grid-column: 2 / span 2;

grid-row-start

Départ de l'item sur l'axe des lignes



grid-row-start: 1;

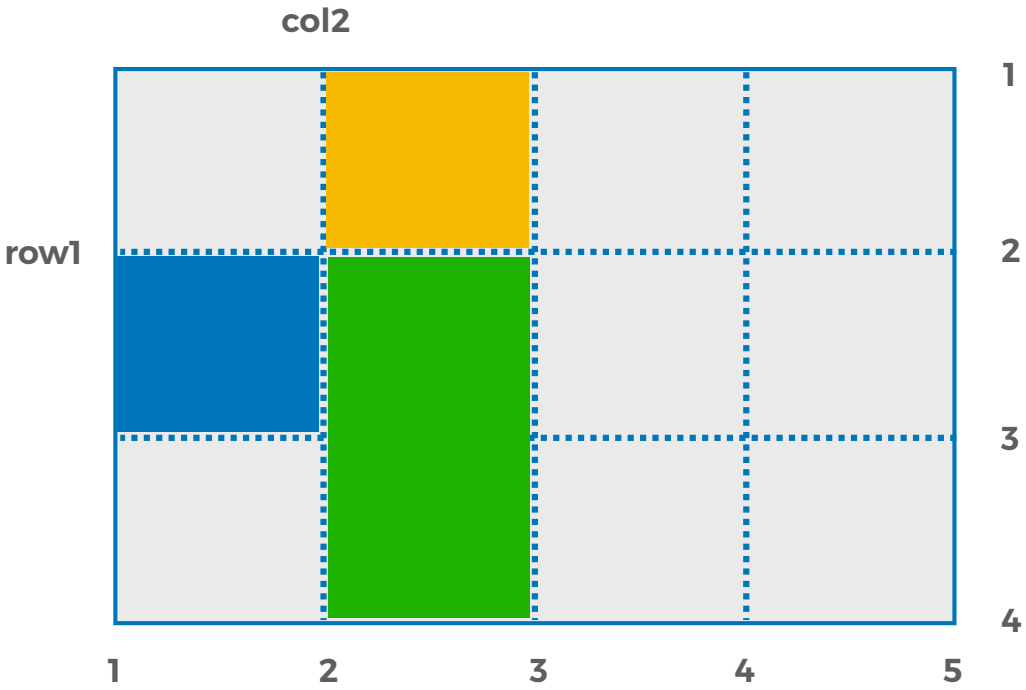
grid-row-start: row1;

grid-row-start: span 2;

s'étend de ...

grid-row-end

Arrivée de l'item sur l'axe des lignes



grid-row-end: 3;

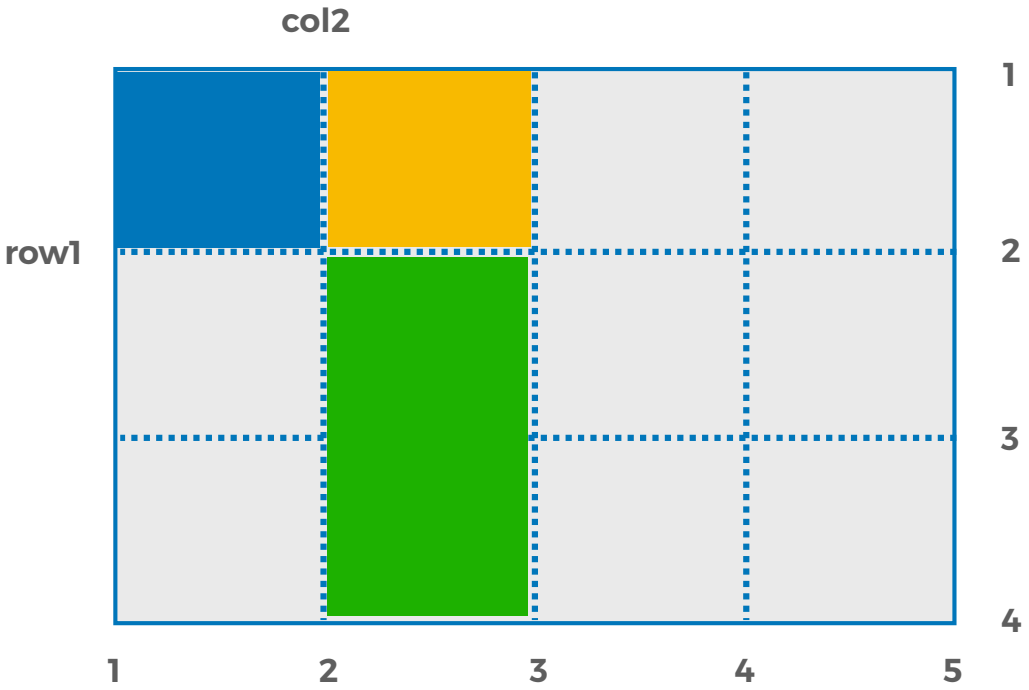
grid-row-end: row1;

grid-row-end: span 2;

grid-row

Taille et emplacement de l'item sur l'axe des lignes

grid-row: <grid-row-start> / <grid-row-end>



grid-row: 1;

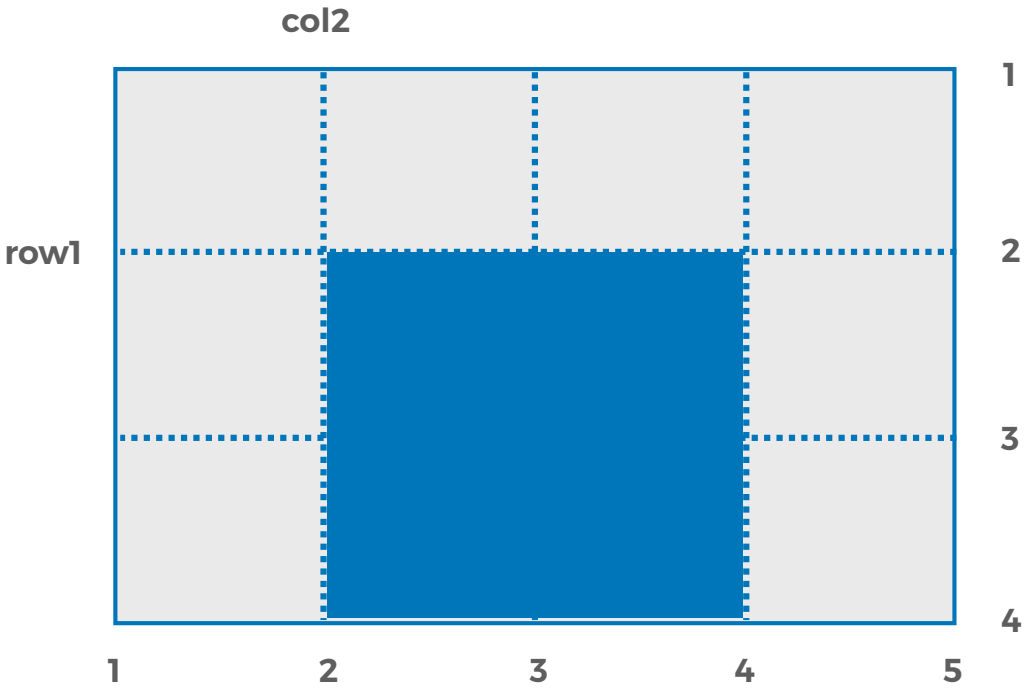
grid-row: auto / row1;

grid-row: 2 / span 2;

grid-area

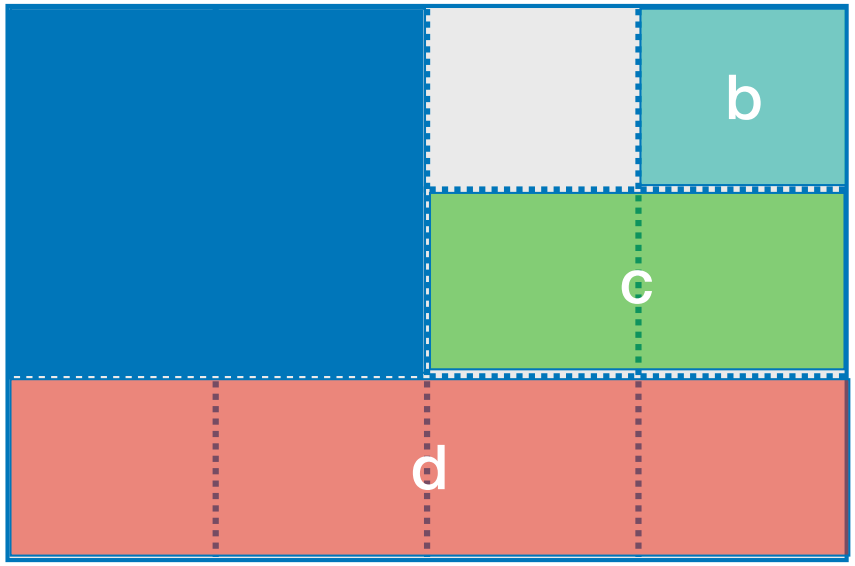
propriété raccourcie de :

grid-area: <grid-row-start> / <grid-column-start> / <grid-row-end> / <grid-column-end>



grid-area: row1 / col2 / 4 / span 2;

grid-area




```
.container {
   grid-template: "a a . b" 1fr
                  "a a c c" 1fr
                  "d d d d" 1fr
                   / repeat(4, 1fr);
}

.element {
   grid-area: a;
}
```

z-index

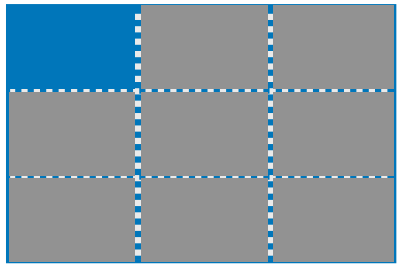
Les gird items peuvent se chevaucher.

```
.container {
  grid-template: repeat(3, 1fr) / repeat(4, 1fr);
}
.blue {
  grid-area: 1 / 1 / span 2 / span 3;
  z-index: 1;
}
.green {
  grid-area: 2 / 2 / span 2 / span 3;
}
```

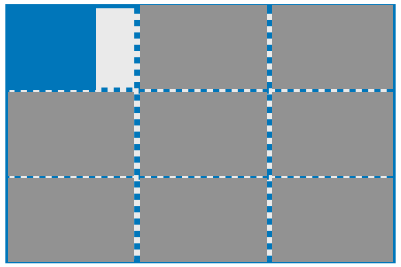
justify-self

Alignement de l'item sur les colonnes de la grille

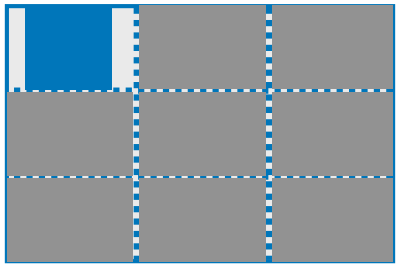
stretch*



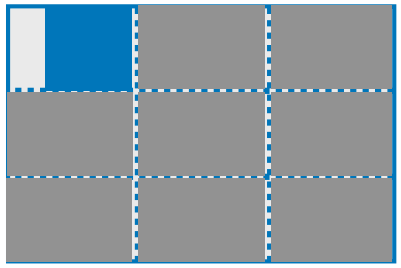
start



center



end

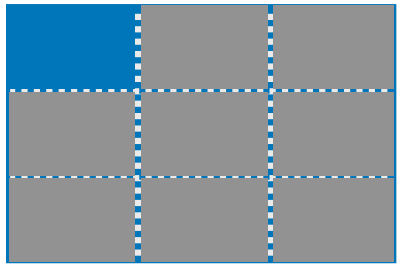


* valeur par défaut

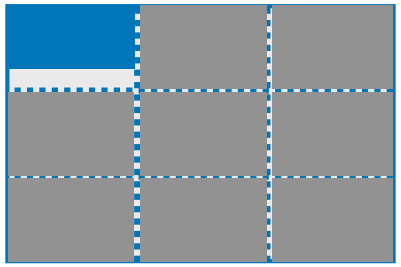
align-self

Alignement de l'item sur les lignes de la grille

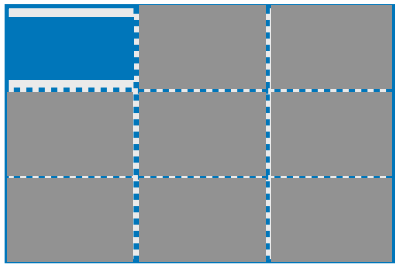
stretch*



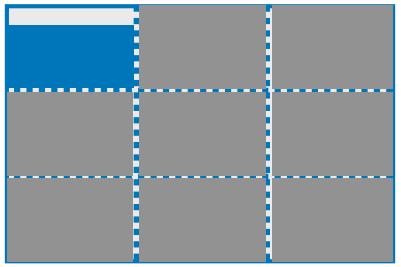
start



center



end

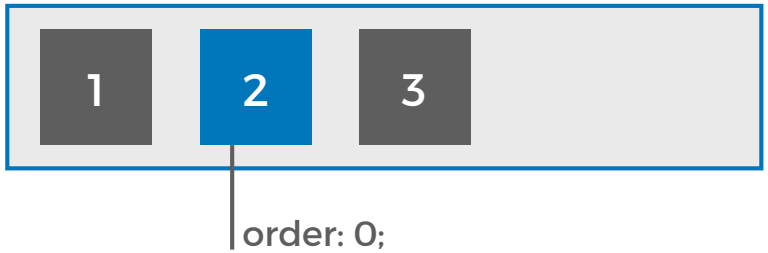


* valeur par défaut

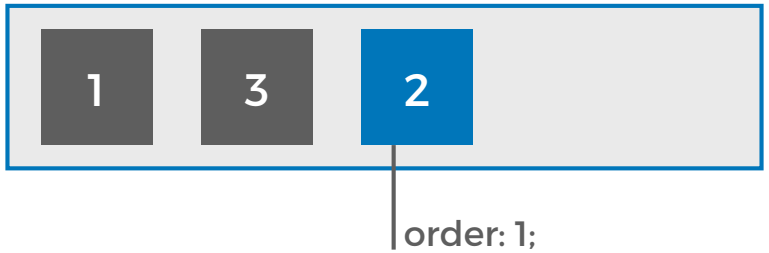
order

Défini l'orde d'apparition des éléments (nombre entier)

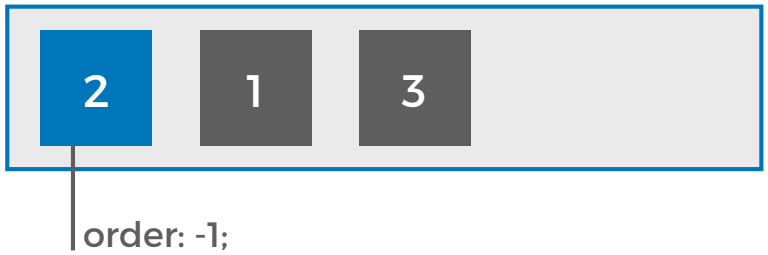
0*



n



-n



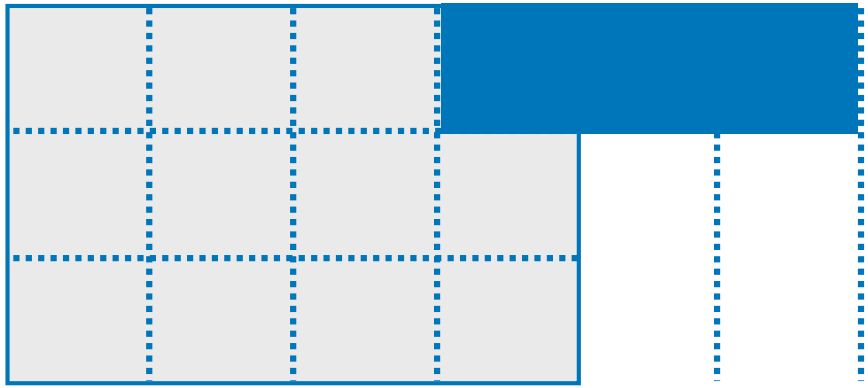
Attention à l'accessibilité des éléments

* valeur par défaut

Autres

grid-auto-columns

Taille des colonnes implicites



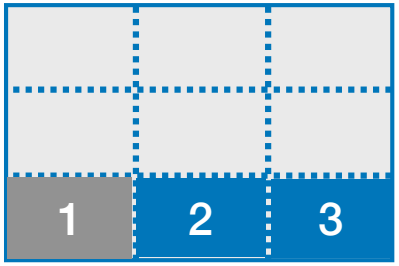
```
.container {
  display: grid;
  grid-template: repeat(3,100px) /
repeat(4, 100px);
  grid-auto-columns: 100px;
}
.element {
  grid-column: 3 / 6;
}
```

grid-auto-rows (lignes implicites)

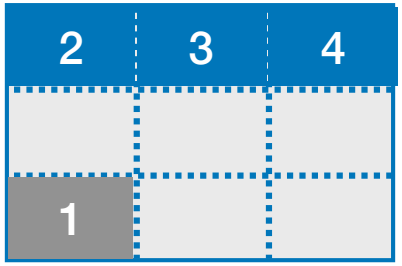
grid-auto-flow

Placement automatique les items non placés de façon explicites

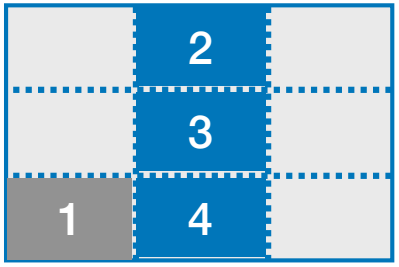
row*



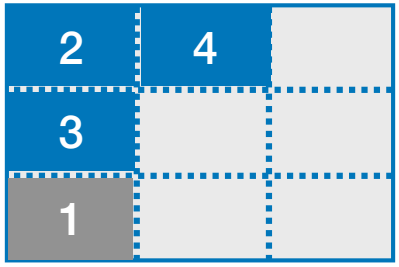
row dense



column



column dense



* valeur par défaut

Firefox vous aime

Firefox nous montre la grille

grid

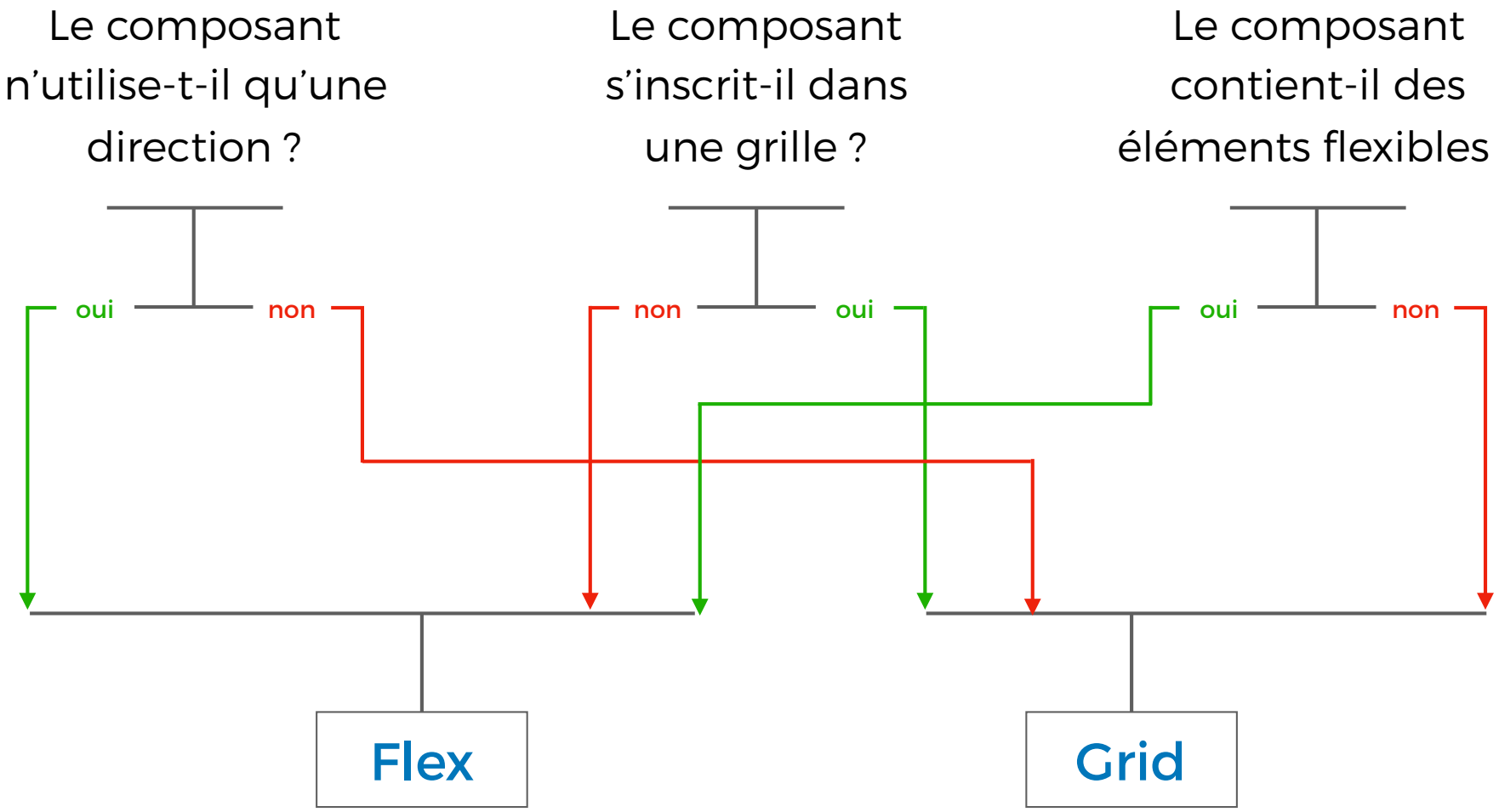
grid: <grid-template> | <grid-template-rows> / [auto-flow && dense?] <grid-auto-columns>? | [auto-flow && dense?] <grid-auto-rows>? / <grid-template-columns>;

https://developer.mozilla.org/fr/docs/Web/CSS/grid

let’s go dude !

https://codepen.io/mynameistom/pen/LOoGWJ

Que choisir ?


Le composant n'utilise-t-il qu'une direction ?
oui
non
Le composant s'inscrit-il dans une grille ?
non
oui
Le composant contient-il des éléments flexibles ?
oui
non
Flex
Grid

Compatibilités

Utiliser un développement mobile first

Certains vieux navigateurs auront la version mobile

Transformer (si besoin) les flex/grid items en inline-block

Les éléments se mettrons les uns à côté des autres

Utiliser la règle @supports

@supports (display:grid) {}

Prévoir un développement standard + ajouter les grid en supplément


© Walter Menegazzi