Positionnement CSS

Interface Homme Machine

Nathanaël Martel

Position des blocs en CSS

Voici un aperçu des méthodes disponible

- Table
- Absolute
- Float
- Flex
- Grid

Rappel

Design fluide :

- La mise en forme s'étire et se déforme avec les dimensions de l'écran

Utilisation de :

- `max-width, max-height`
- `vw, vh, %`
- `fr (fraction de l'espace restant)`

Rappel

Design responsive :

- La mise en forme et modifié en fonction des dimensions de l'écran

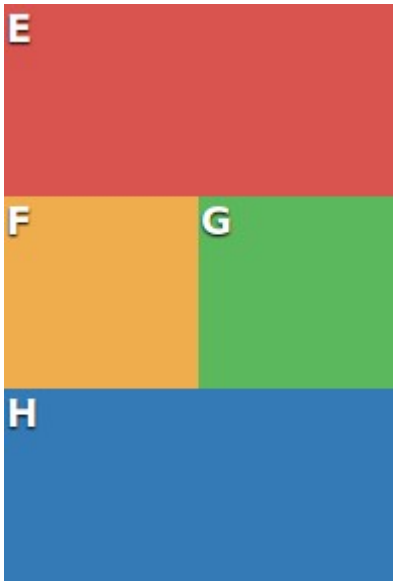
Utilisation de :

- Media query

Table

```html
<table>
    <tr>
        <td class="red" colspan="2">E</td>
    </tr>
    <tr>
        <td class="yellow">F</td>
        <td class="green">G</td>
    </tr>
    <tr>
        <td class="blue" colspan="2">H</td>
    </tr>
</table>
```

```css
table {border-spacing:0;}
.red, .green, .yellow {
    width:5em;
    height:5em;
}
.blue {
    height:5em;
}
```



<Table>

Avantages :

- **Très simple**
- **Ne nécessite pas de CSS**
- **Excellent support, en particulier pour les clients de messagerie**

Inconvénients :

- **Pas du tout adapté**
- **Sémantique fausse**
- **Adaptation mobile difficile**

→ ne jamais utiliser !

(sauf pour les mails ou support CSS faible)

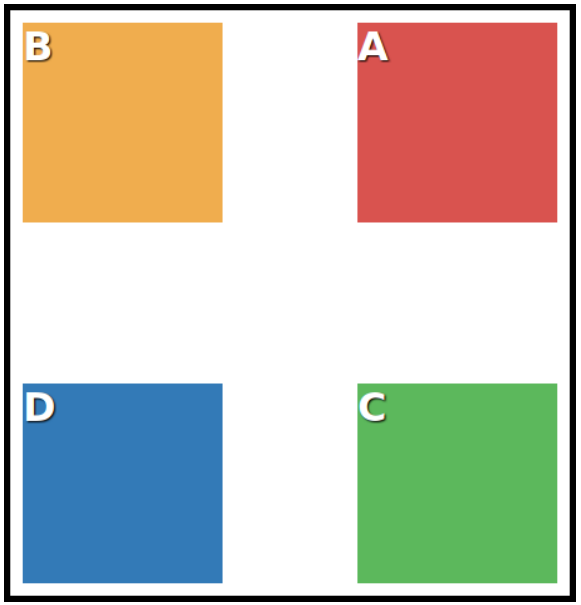
Rappel

La balise `<div>`

- Élément neutre d'un point de vue sémantique
- De type « block » (`display:block`)
- Prend la largeur de l'espace disponible
- S'adapte à la hauteur de son contenu
- Est positionné dans le flux, en dessous de l'élément suivant

Absolute

```html
<div class="relative-container">
    <div class="red">A</div>
    <div class="yellow">B</div>
    <div class="green">C</div>
    <div class="blue">D</div>
</div>
```




```css
.relative-container {
    border:5px solid black;
    position:relative;
    height:90vh;
}
.red, .yellow, .green, .blue {
    position:absolute;
    width:5em;
    height:5em;
}
.red {top:10px;right:10px;}
.yellow {top:10px;left:10px;}
.green {bottom:10px;right:10px;}
.blue {bottom:10px;left:10px;}
```

`Display:absolute`

- L’élément est automatiquement en `display:block`
- La largeur et la hauteur s’adapte à son contenu
- L’élément sort du flux

`position:absolute`

Avantages :

- **Possibilité de mette un élément à peu près n'importe où**

Inconvénients :

- **Nécessite beaucoup de code en responsive ou fluide**
- **Superposition possible des éléments**
  - **Peut-être géré avec : `z-index`**

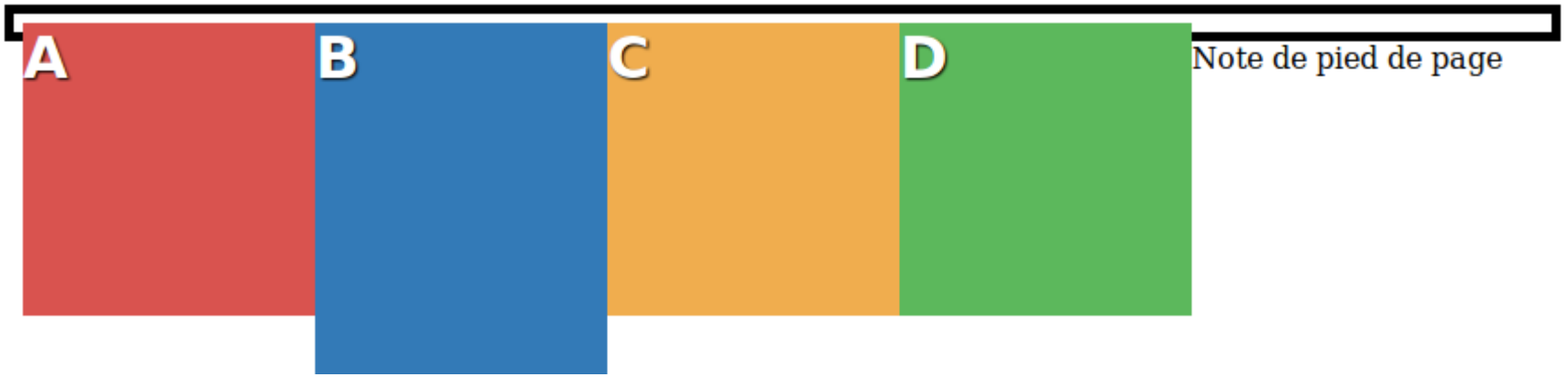
→ à utiliser avec parcimonie

Float

```html
<div class="container">
    <div class="red">A</div>
    <div class="blue">B</div>
    <div class="yellow">C</div>
    <div class="green">D</div>
</div>

Note de pied de page
```

```css
.red, .green, .yellow, .blue {
    width:5em;
    height:5em;
    float:left;
}
.blue {height:6em;}
.container {
    border:5px solid black;padding:5px;
}
```



Float

```html
<div class="container">
    <div class="red">A</div>
    <div class="blue">B</div>
    <div class="yellow">C</div>
    <div class="green">D</div>
    <div class="clear"></div>
</div>

Note de pied de page
```

```css
.red, .green, .yellow, .blue {
    width:5em;
    height:5em;
    float:left;
}
.blue {height:6em;}
.clear {
    clear:both;
}
```

Note de pied de page

`Float:left`

- L'élément est automatiquement en `display:block`
- Mais sa largeur et sa hauteur s'adapte à son contenu
- L'élément sort du flux
- Le flux peut attendre la fin des `float` avec `clear:both`
- Conçu pour faire courir le texte autour d'une image

`Float:left`

Avantages :

- **Relativement simple**

Inconvénients :

- **Les éléments ne sont pas lié entre eux, donc la hauteur des « colonnes » ne peut être lié**

→ le plus courant en 2017

Flex

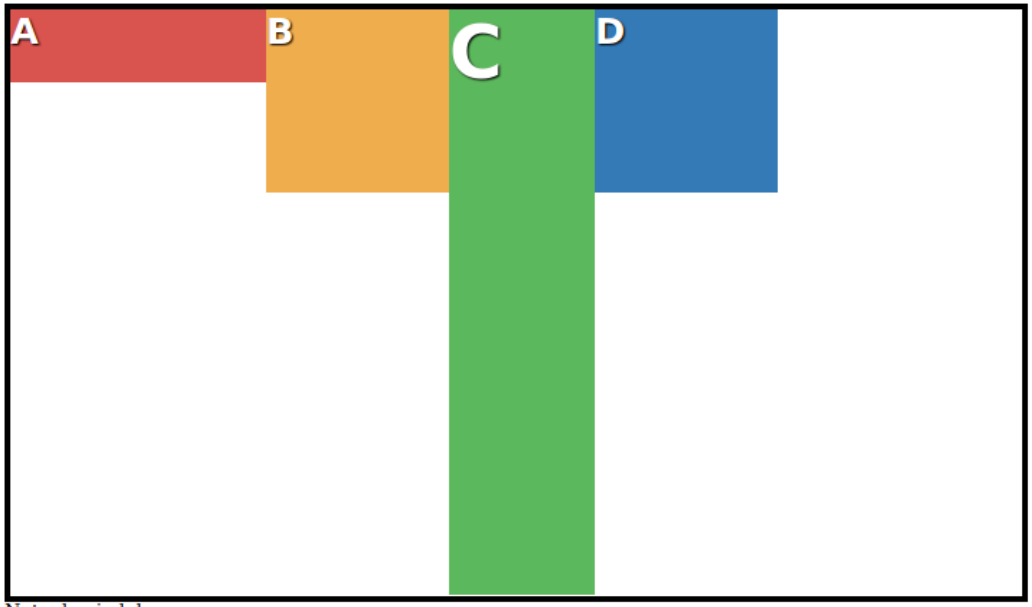
```
<div class="flex-container">
    <div class="red">A</div>
    <div class="yellow">B</div>
    <div class="green">C</div>
    <div class="blue">D</div>
</div>
```

Note de pied de page

```
.red, .green, .yellow, .blue {
    width:5em;
    height:5em;
}
.red {
    height:2em;
    width:7em;
}
.green {
    height:8em;
    width:2em;
    font-size:4em;
}
.flex-container {
    border:5px solid black;paddin
```

Flex

```
.flex-container {
    display:flex;
}
```



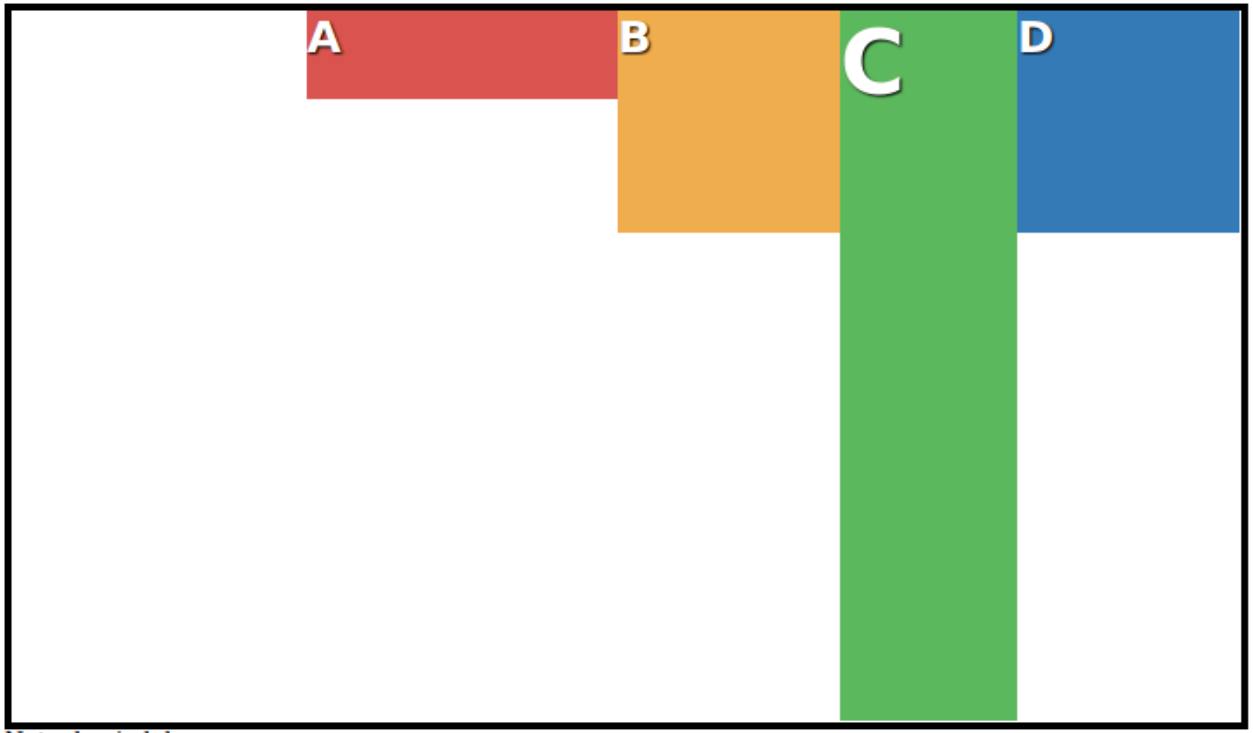
Note de pied de page

Flex

Répartition sur l'axe principale :

```
.flex-container {
  display:flex;
  justify-content:end;
}
```

- End
- Start
- Center
- Space-arround
- Space-between



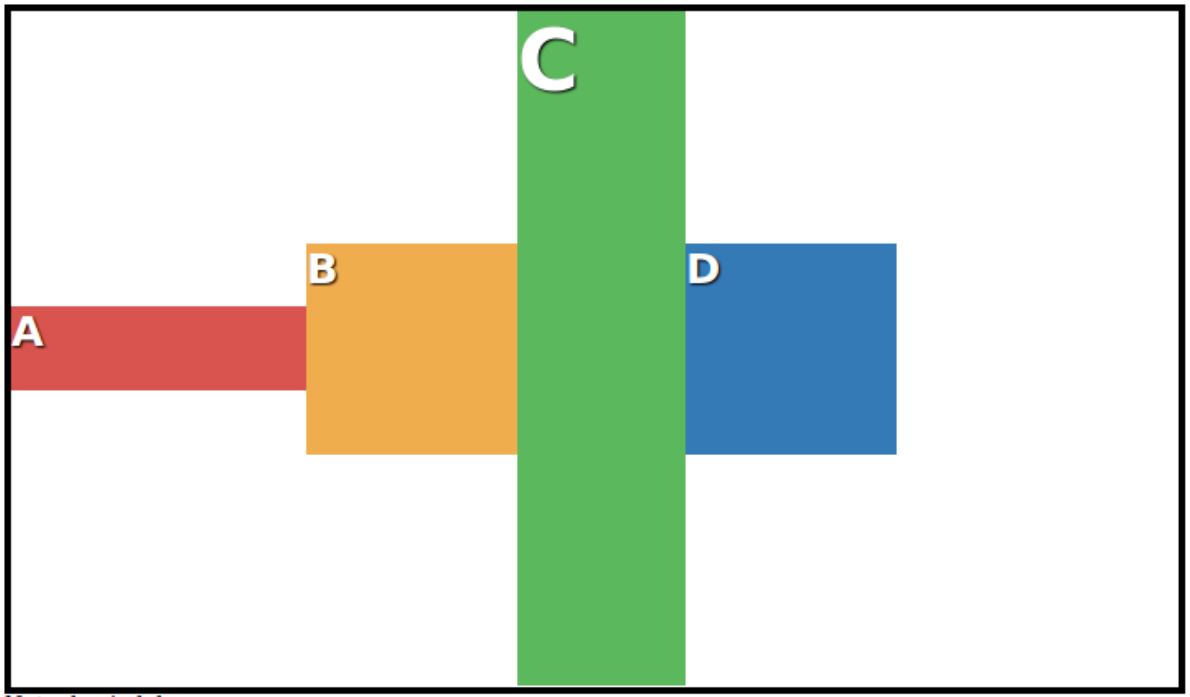
Note de pied de page

Flex

Répartition sur l'axe secondaire :

```
.flex-container {
  display:flex;
  align-items:center ;
}
```

- End
- Start
- Center
- baseline



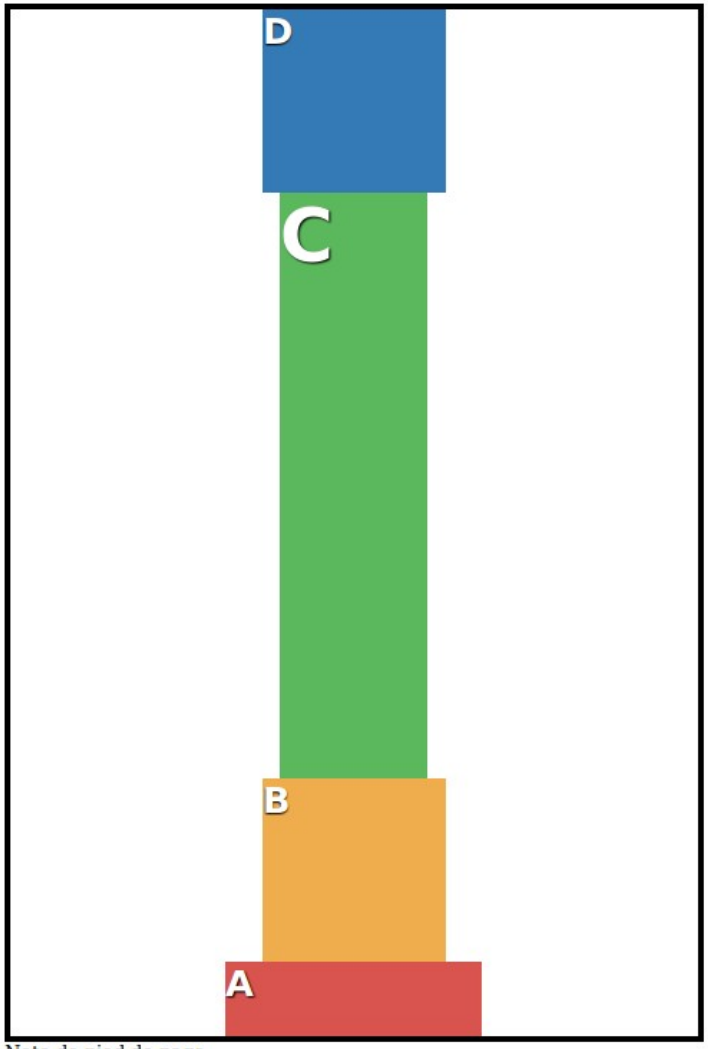
Note de pied de page

Flex

Changement des axes

```
.flex-container {
  display:flex;
  flex-direction: column-reverse;
  align-items: center;
}
```

- Column-reverse (de bas en haut)
- Column (de haut en bas)
- Row (de gauche à droite)
- Row-revers (de droite à gauche)



Note de pied de page

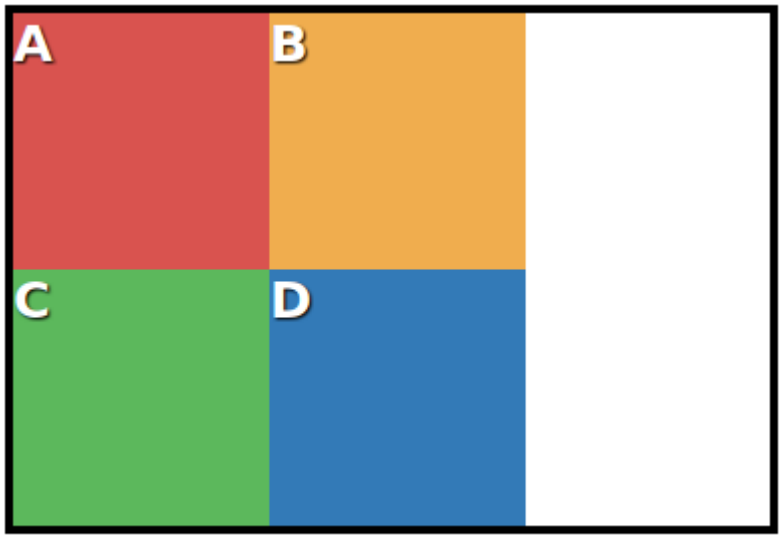
Flex

Sur plusieurs lignes

```
.red, .green, .yellow, .blue {
    width: 5em; height: 5em; font-size: 2em;
}

.flex-container {
  display:flex;
  flex-wrap:wrap;
}
```

- Wrap
- Nowrap
- Wrap-reverse



Note de pied de page

Flex

Répartition des blocs

```
.red, .green, .yellow, .blue {
    height: 5em; font-size: 2em;
    flex:1 ;
}
.green {
    flex:2 ;
}

.flex-container {
  display:flex;
}
```

Note de pied de page

- Répartition sur tout l'espace disponible
- En fonction du poids de chaque enfant

Display:flex

Avantages :

- **Conçu pour ça !**
- **Possibilité changer l'axe et le sens**
- **Possibilité de centrer (sur tous les axes)**
- **Très facile à manier**
- **Adapté pour le responsive et le fluide**
- **Idéale pour les listes**

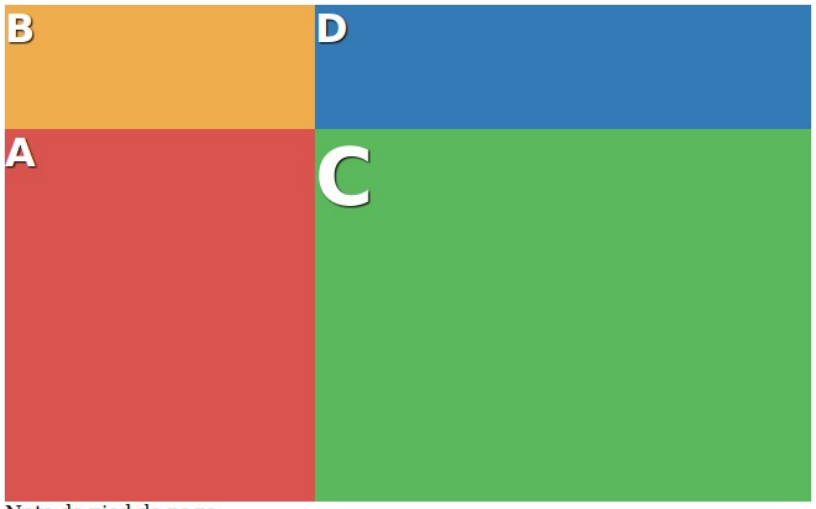
Inconvénient :

- **Tous les enfants de l'élément `flex` se positionnent ensemble**

Grid

```html
<div class="grid-container">
    <div class="red">A</div>
    <div class="yellow">B</div>
    <div class="green">C</div>
    <div class="blue">D</div>
</div>
```

Note de pied de page

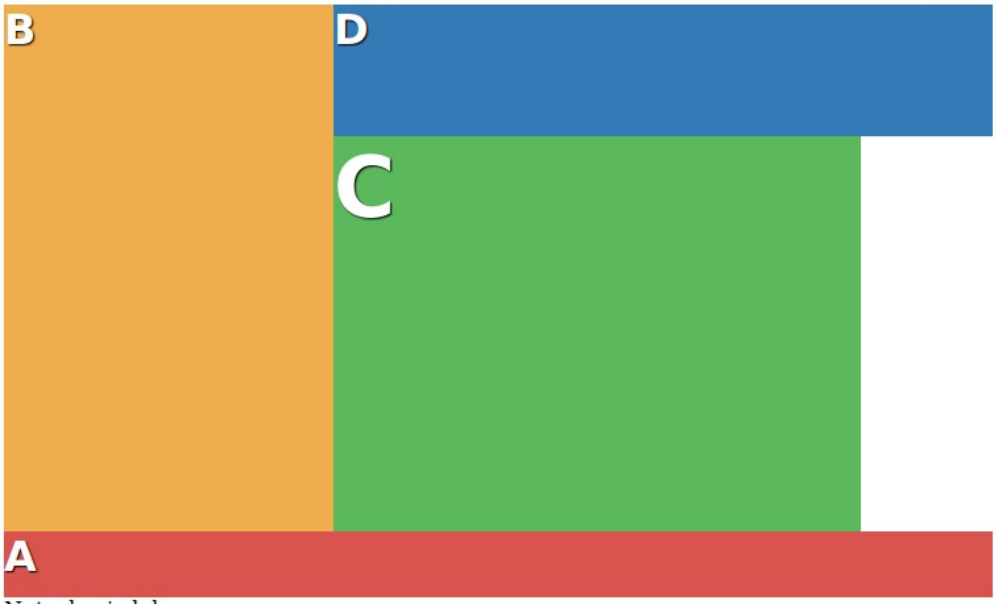


Note de pied de page

```css
.grid-container {
    display: grid;
    grid-template-columns: 250px 400px;
    grid-template-rows: 100px 300px;
}

.yellow {
    grid-column: 1;
    grid-row: 1;
}
.blue {
    grid-column: 2;grid-row: 1;
}
.red {
    grid-column: 1;grid-row: 2;
}
.green {
    grid-column: 2;grid-row: 2;
}
```

Grid



Note de pied de page

```
.grid-container {
    display: grid;
    grid-template-columns: 250px 400px 100px;
    grid-template-rows: 100px 300px 50px;
}

.yellow {
    grid-column: 1;grid-row: 1 / span 2;
}
.blue {
    grid-column: 2 / span 2;grid-row: 1;
}
.red {
    grid-column: 1 / span 3;grid-row: 3;
}
.green {
    grid-column: 2;grid-row: 2;
}
```

Grid


```
.grid-container {
    display: grid;
    grid-template-areas
    "h h"
    "n c"
    "f f";
}

.yellow {
    grid-area: h;
}
.blue {
    grid-area: f;
}
.red {
    grid-area: n;
}
.green {
    grid-area: c;
}
```

Grid

Avantages :

- **Conçu pour ça !**
- **Possibilité d'indiquer où vont les éléments a posteriori**
- **Adapté pour le responsive et le fluide**
- **Idéale pour la mise en page global**

Inconvénient :

- **Support des navigateurs encore restreint**
  - **85 % en 2019**

Alignement vertical

- `display:table-cell ;`
  - Alors vous pouvez faire `vertical-align:middle;`
  - Inconvéniant : la taille et le positionnement d'une `table-cell` difficile à gérer
- `line-height :xx ;`
  - Retirer le padding et augmenter la hauteur de la ligne
  - Inconvéniant : ne fonctionne que s'il y a une seule ligne
- Flex
  - `align-items:center;`

Alignement horizontal

- Éléments in-line
  - text-align:left;
  - Left, right, center
- Éléments de type blocs
  - Margin-left:auto;
    margin-right:auto;
    max-width:300px
- Éléments Flex
  - Justify-content:center

Alignement horizontal et vertical

- `Cellule de tableau`
  - `display:table-cell;`
    `text-align:center ;`
    `vertical-align:middle ;`
    `height:100px;width:100px ;`
  - **Inconvéniant : la taille doit être géré en px**
- **Éléments Flex**
  - `Display:flex ;`
    `Justify-content:center ;`
    `align-item:center ;`

Responsive

- Adapter en fonction des caractéristiques du support
  - Largeur de l'écran
  - Hauteur de l'écran
  - Orientation : portrait / paysage
  - Media : print, screen…
  - Autres : aspect-ratio, resolution, scan, grid, update, color, color-gamut, color-index, display-mode, inverted-color, pointer, hover…

Responsive

Choix du media

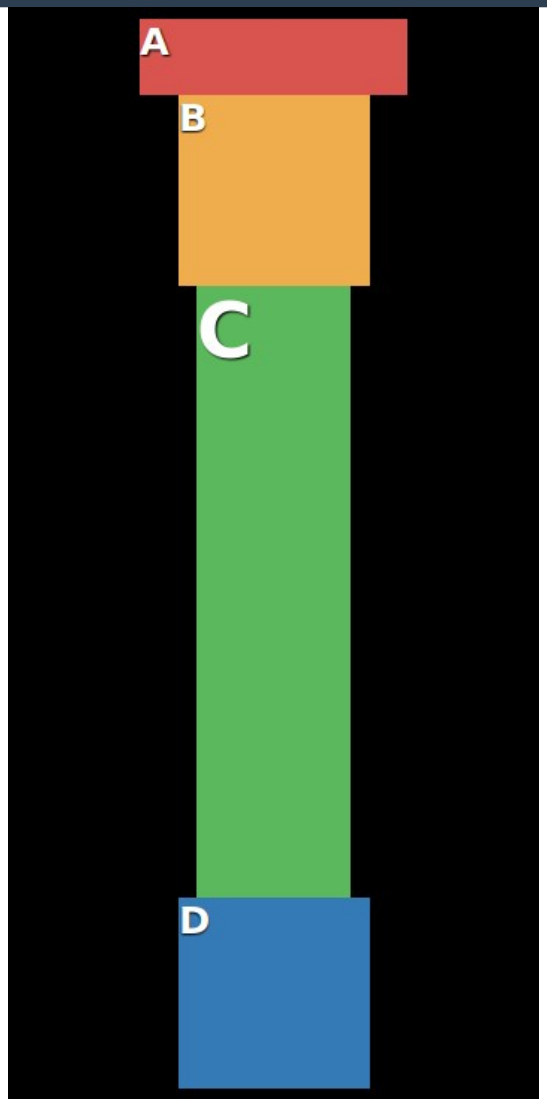
- **Dans le CSS :**

```
@media print {
  #menu, #footer, aside {
    display:none;
  }
}
```

- **Import**
  - `@import url("fineprint.css") print;`
- **Dans le HTML :**

```
<link rel="stylesheet" media="print" href="print.css"
type="text/css" />
```

Responsive + flex




```
@media (max-width:500px) {
    .flex-container {
        background:black;
        flex-direction: column;
    }
}
```

Responsive + grid

Note de pied de page

```
@media (max-width:500px) {
    .grid-container {
        grid-template-columns: 100%;
        grid-template-rows: auto auto auto auto;
    }

    .yellow {
        grid-column: 1;grid-row: 3;
    }
    .blue {
        grid-column: 1;grid-row: 2;
    }
    .red {
        grid-column: 1;grid-row: 1;
    }
    .green {
        grid-column: 1;grid-row: 4;
    }
}
```

Conclusion

Faites attention à utiliser une méthode robuste et sémantique