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, %`

Rappel

Design responsive :

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

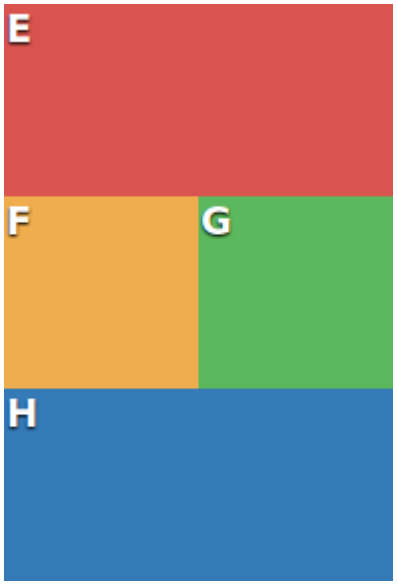
Utilisation de :

- 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 quasiment 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)

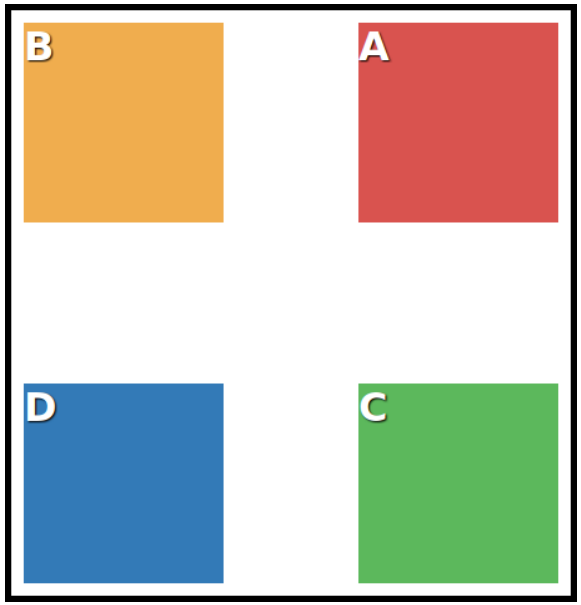
Rappel

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`
- Mais sa largeur et sa 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**

→ à utiliser avec parcimonie

Float

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

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

`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`

`Float:left`

Avantages :

- **Relativement simple**

Inconvénients :

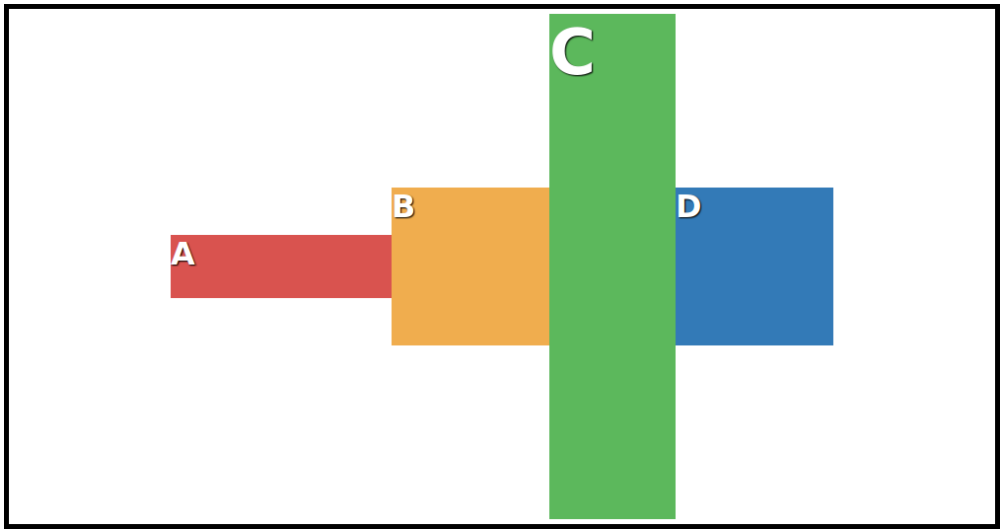
- **Les éléments ne sont pas lié entre eux, donc la hauteur des « colonnes » ne peut être lié**

→ le plus courant en 2017

→ c'est comme ça que fonctionne bootstrap

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>
```




```
.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;padding:5px;
    display:flex;
    flex-direction: row;
    justify-content: center;/*space-between;*/
    align-items:center;
}
```

Flex

Présentation de Hubert Sablonnière

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**

Inconvénients :

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

Grid

Voir le dossier sur alsacreation

Grid

Avantages :

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

Inconvénients :

- **Support des navigateurs encore restreint**

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
  - `Justify-content:center`

Conclusion

Faites attention à utiliser une méthode robuste et sémantique