BDW1 - Programmation web - CSS

Fabien Duchateau

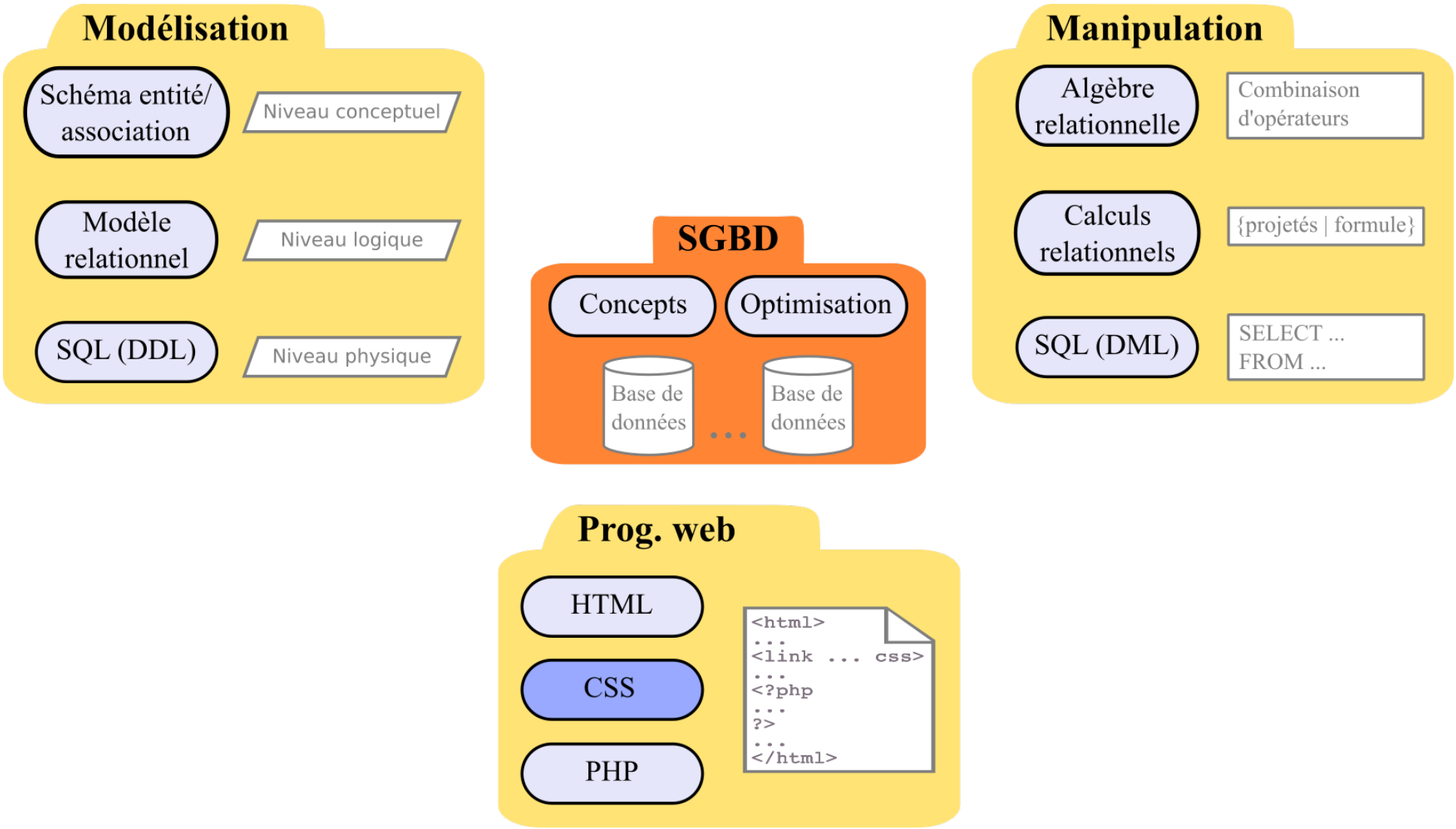
fabien.duchateau [at] univ-lyon1.fr

Université Claude Bernard Lyon 1

2018 - 2019

http://liris.cnrs.fr/fabien.duchateau/BDW1/

Positionnement dans BDW1



Une page web sans CSS, puis avec CSS

En tête

Menu

- Partie 1
- Partie 2
- Partie 3
- Partie 4

Titre de la section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce nec elit condimentum, ullamcorper purus at, pulvinar ante. Ut pellentesque convallis dui eu suscipit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Phasellus et venenatis sapien, ac rhoncus felis. Cras posuere sodales aliquet. Curabitur posuere nulla et ante faucibus lobortis. Integer quis rutrum justo. Vestibulum egestas ante vitae purus dapibus, eu dapibus risus suscipit.
Sed vel sodales massa. Proin in gravida leo. Morbi eget laoreet nunc. Etiam auctor feugiat ante ac vulputate. Curabitur id auctor dui. Cras ac bibendum metus, a commodo velit. Proin eleifend facilisis interdum. Aliquam semper suscipit nibh, fringilla molestie leo eleifend id. Quisque ac luctus tortor. Nulla et tincidunt turpis. Phasellus sagittis eros magna, id dictum elit commodo sit amet. Fusce quis eros mi.

Pied de Page - Texte généré par Lipsum

En tête

Menu

- Partie 1
- Partie 2
- Partie 3
- Partie 4

Titre de la section

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce nec elit condimentum, ullamcorper purus at, pulvinar ante. Ut pellentesque convallis dui eu suscipit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Phasellus et venenatis sapien, ac rhoncus felis. Cras posuere sodales aliquet. Curabitur posuere nulla et ante faucibus lobortis. Integer quis rutrum justo. Vestibulum egestas ante vitae purus dapibus, eu dapibus risus suscipit.
Sed vel sodales massa. Proin in gravida leo. Morbi eget laoreet nunc. Etiam auctor feugiat ante ac vulputate. Curabitur id auctor dui. Cras ac bibendum metus, a commodo velit. Proin eleifend facilisis interdum. Aliquam semper suscipit nibh, fringilla molestie leo eleifend id. Quisque ac luctus tortor. Nulla et tincidunt turpis. Phasellus sagittis eros magna, id dictum elit commodo sit amet. Fusce quis eros mi.

Pied de Page - Texte généré par Lipsum

Rappels

Balises structurantes en HTML :

- `header`, `footer`
- `nav`
- `section`, `article`
- `aside`
- `div` et `span`

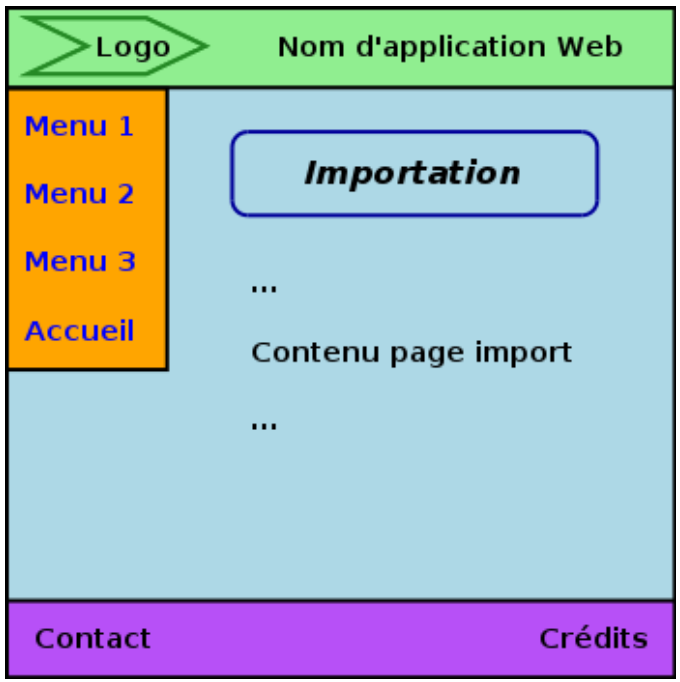
Ces balises étant purement sémantiques, comment réaliser la mise en page d'un document HTML ?

Notion de boîte

En HTML, toute balise est une boîte

De par son arborescence, un document HTML est une imbrication de boîtes

- Un exemple d'imbrication de boîtes ?
- Pourquoi les boîtes ne s'imbriquent pas de la même manière ?
- Comment spécifier que la zone verte prenne autant d'espace et soit collée en haut ?



Comment afficher un élément HTML ?

L'affichage d'un élément dépend de quatre paramètres optionnels appliqués à sa boite :

- Son **ordre d'apparition** dans le flux (niveau HTML)
- Ses **dimensions** (hauteur et largeur, mais aussi les marges intérieures/extérieures)
- Son **type de boîte** (affichage de la balise)
- Son **positionnement** (e.g., relatif, absolu, flex)

Dans ce cours, personnalisation de ces paramètres d'affichage

---

http://fr.openclassrooms.com/informatique/cours/apprenez-a-creer-votre-site-web-avec-html5-et-css3/

Plan

Notion de flux

Flux : ordre dans lequel les éléments HTML sont lus par le navigateur et affichés

Il est possible de sortir un élément du flux naturel, dans ce cas, l'espace libéré reste vacant

---

http://openweb.eu.org/articles/initiation_flux/

Exemple de flux

Plan

Dimensions d'un élément

Propriétés CSS pour les dimensions :

- Hauteur (`height`) et largeur (`width`)
- Hauteurs et largeurs minimale et maximale (`min-height`, `min-width`, `max-height`, `max-width`)
- Marges internes (`padding`) et externes (`margin`), qui s'ajoutent aux dimensions `width` et `height` et qui se déclinent :
  - `padding-top`, `padding-right`, `padding-bottom` et `padding-left`
  - `margin-top`, `margin-right`, `margin-bottom` et `margin-left`

Unités de mesure : **rem**, **em**, **%**, px, ex, cm, mm, in, pt, pc

---

http://www.w3.org/Style/Examples/007/units.fr.html

Exemples de dimensionnement

```
width: 50%; /* largeur de l'élément */
min-height: 400em; /* hauteur minimale de l'élément */
max-height: 60%; /* hauteur maximale de l'élément */
padding-top: 2rem; /* marge intérieure du haut */
padding-bottom: 1em; /* marge intérieure du bas */
margin: 1%; /* marge extérieures (haut, droite, bas, gauche) */
```

Gestion du contenu

Un contenu occupe une place plus ou moins grande à l'intérieur d'un élément aux dimensions fixées

```
overflow: visible | scroll | hidden | auto ;
word-wrap: normal | break-word ;
```

- La propriété `overflow` indique comment gérer le contenu d'un élément
- Valeurs : visible avec débordement (défaut), ascenseur, caché, automatique (i.e., choix par le navigateur)
- La propriété `word-wrap` avec la valeur *break-word* permet de spécifier une césure des mots trop longs (e.g., URL)

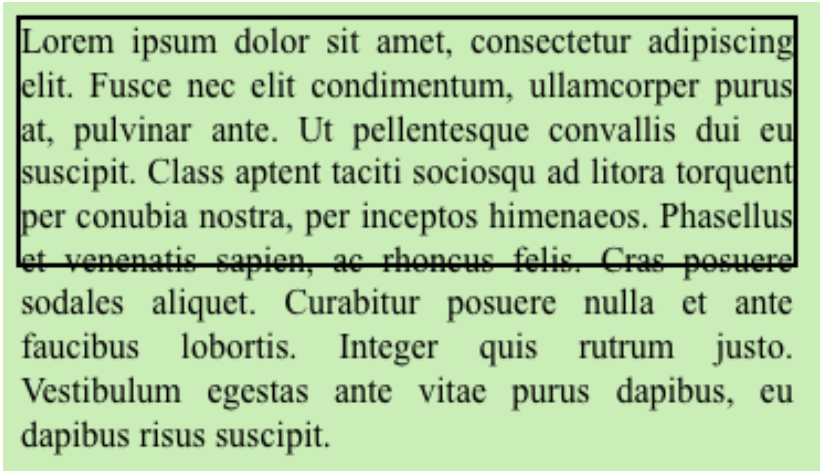
---

http://www.w3schools.com/css/css_overflow.asp

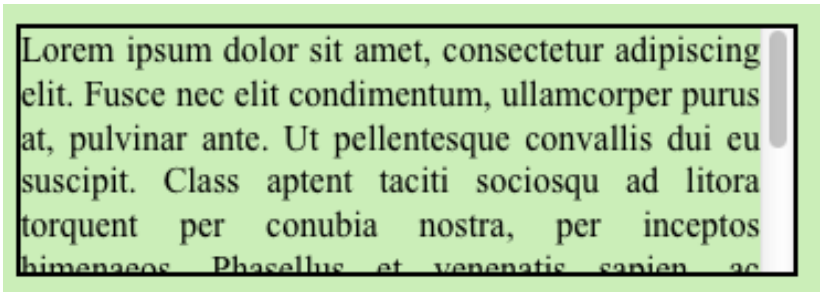
Exemples de gestion du contenu

```
#paragraphe1
{
  width: 350px;
  height: 110px;
  text-align: justify;
  border: 2px solid black;
}

#paragraphe2
{
  width: 350px;
  height: 110px;
  text-align: justify;
  border: 2px solid black;
  overflow: auto;
}
```



#paragraphe1



#paragraphe2

Plan

Notion de type de boîte

Type de boîte : comportement d'un élément HTML en termes d'affichage et de relation avec ses éléments voisins

display : block | inline | inline-block | list-item | table | table-row | table-cell | none | ... ;

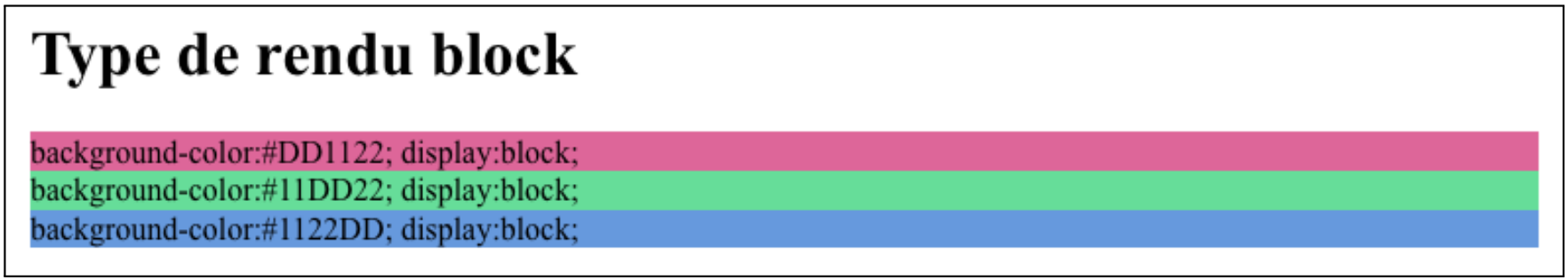
- Chaque balise HTML a un type de boîte par défaut
- La propriété `display` modifie le type de boîte d'un élément

https://developer.mozilla.org/en-US/docs/Web/CSS/display

Type de boîte *block*

display: block ;

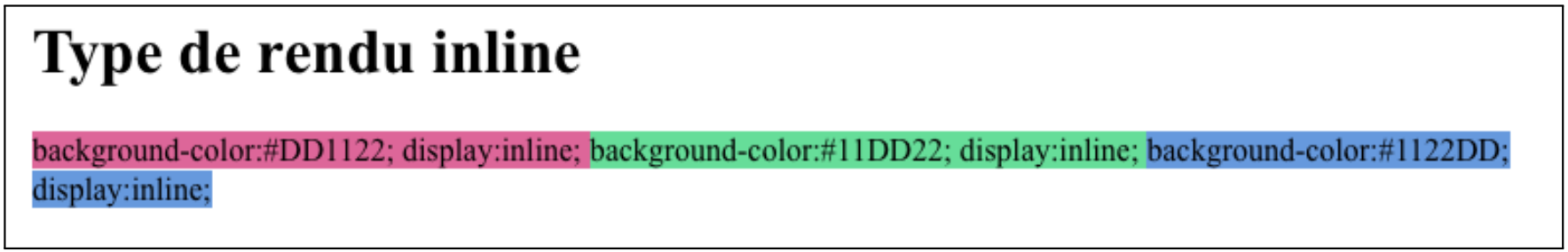
- Élément placé sous un autre (succession verticale) avec retour-chariots avant et après (e.g., `<p>`, `<h1>`, `<div>`)
- Largeur maximale disponible dans son conteneur parent
- Redimensionnable avec les propriétés `width`, `height`, `min-width`, `min-height`, `max-width` et `max-height`



Type de boîte *inline*

display: inline ;

- Élément placé à côté d'un autre (succession horizontale, e.g., `<a>`, `<img>`, `em`, `<span>`)
- Largeur déterminée par le contenu (texte, image, etc.)
- Non redimensionnable (en théorie)



Type de boîte *inline-block*

```
display: inline-block ;
```

- Identique au type de boîte *inline*, mais les éléments sont redimensionnables (cas de la balise `<input>`)
- Utilisée pour la mise en page (en combinaison avec la propriété d'alignement vertical `vertical-align`)

Type de rendu inline-block

background-color:#DD1122; display:inline-block; background-color:#11DD22; display:inline-block; background-color:#1122DD; display:inline-block;

Types de boîte *table*, *table-row*, *table-cell*

```
display: table | table-row | table-cell ;
```

- Même comportement que les balises `table`, `tr`, `td`, en particulier chaque cellule aura la même hauteur
- Utilisées pour la mise en page

Type de rendu table, table-row, table-cell

background-color:#DD1122; display:table;
background-color:#11DD22; display:table-row;background-color:#1122DD; display:table-cell;

Exemple de mise en page avec tableau

- Déclaration `display: table;` pour l'élément conteneur (parent)
- Déclaration `display: table-cell;` pour les zones enfant
- Avec ces tableaux émulés, les zones enfant ont la même hauteur quelque soit leur contenu

```
#main {
        display:table;
        margin:auto;
}
#menu  {
        display:table-cell;
        width:240px;
        background-color:#FF3366;
}
#contenu {
        display:table-cell;
        background-color:#9966FF;
}
```

http://css.mammouthland.net/mise-en-page-avec-display-table.php

Type de boîte *none*

```
display: none ;
```

- Élément caché, qui n'apparait pas et n'occupe plus d'espace sur la page
- Comportement différent de `visibility: hidden;`, qui conserve l'emplacement (vide) de l'élément à ne pas afficher

En résumé

Une vingtaine de valeurs possibles pour la propriété `display` !

https://developer.mozilla.org/en-US/docs/Web/CSS/display

Plan

Positionnement d'un élément

Par défaut, un élément se positionne selon son ordre d'arrivée dans le flux et selon son type de boîte et ses dimensions

Les propriétés `position` et `float` permettent de modifier ce positionnement :

- Statique (par défaut)
- Relatif
- Fixe
- Absolu
- Flottant

---

http://www.w3schools.com/css/css_positioning.asp
http://www.w3schools.com/css/css_float.asp

Positionnement statique

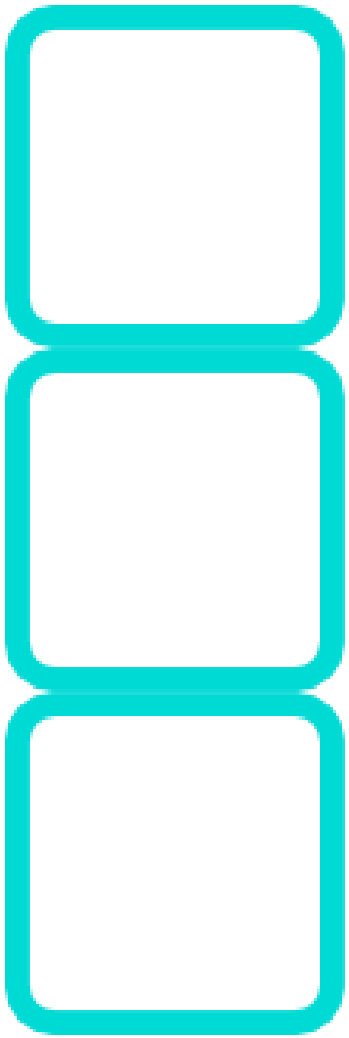
```
position: static ;
```

- La propriété `position` indique comment se positionne l'élément dans la page
- La valeur *static* est la valeur par défaut (positionnement selon l'ordre, le type de boîte et les dimensions)

---

http://www.w3schools.com/cssref/pr_class_position.asp

Exemple de positionnement statique



```
#element_1 {
  position: static;
  width: 200px;
  height: 200px;
}

#element_2 {
  position: static;
  width: 200px;
  height: 200px;
}

#element_3 {
  position: static;
  width: 200px;
  height: 200px;
}
```

http://blog.fourthbit.com/2013/11/27/essential-css-positioning

Positionnement relatif

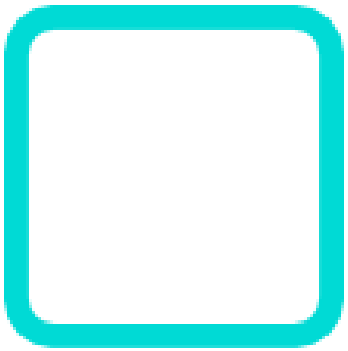
```
position: relative ;
```

- La valeur *relative* indique que l'élément est décalé par rapport à sa position initialement prévue dans le flux
- Les autres éléments considèrent que celui-ci est toujours à sa position initiale dans le flux
- Le décalage est précisé avec les propriétés `top`, `bottom`, `left` et `right`
- La propriété `z-index` permet si besoin de préciser l'ordre de superposition

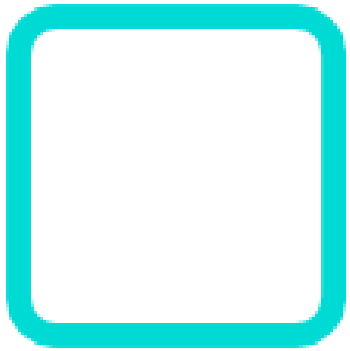
---

http://www.w3schools.com/cssref/pr_pos_z-index.asp

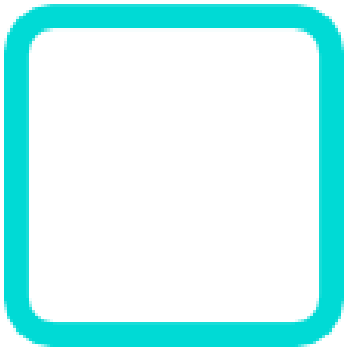
Exemple de positionnement relatif



```
#element_1 {
  position: relative;
  width: 200px;
  height: 200px;
}
```



```
#element_2 {
  position: relative;
  width: 200px;
  height: 200px;
  left: 200px;
}
```



```
#element_3 {
  position: relative;
  width: 200px;
  height: 200px;
}
```

http://blog.fourthbit.com/2013/11/27/essential-css-positioning

Positionnement absolu

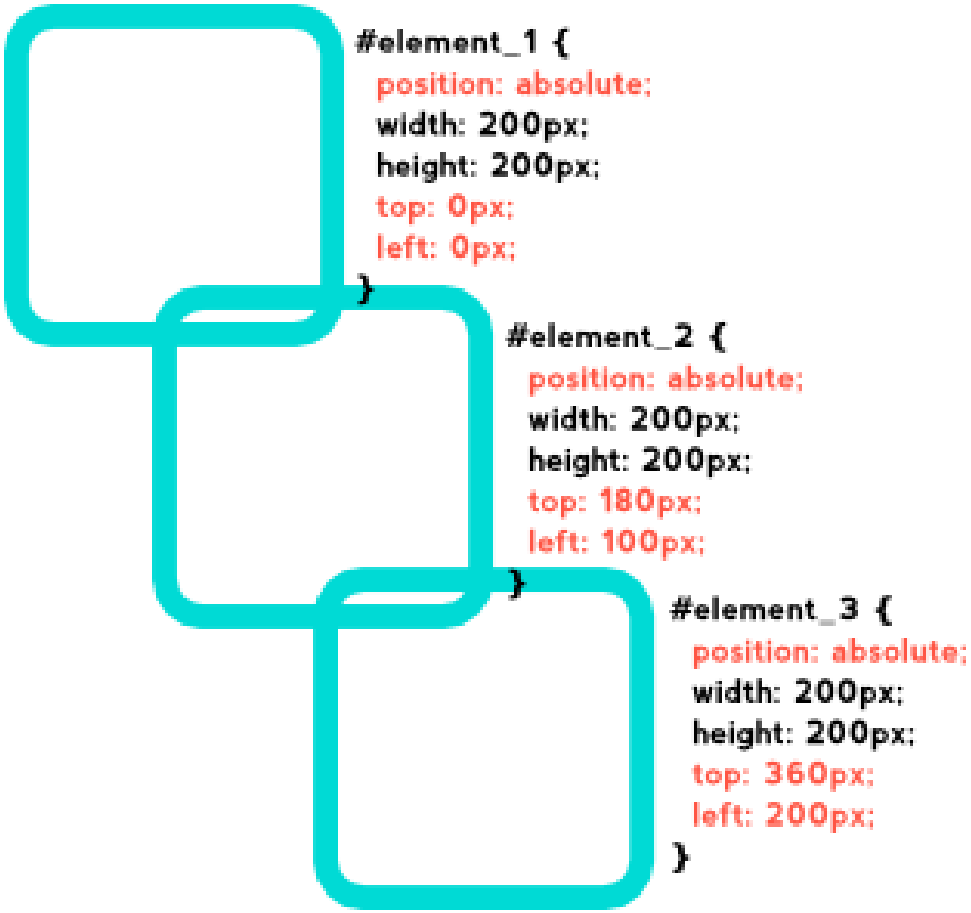
position: absolute ;

- La valeur *absolute* indique que l'élément est retiré du flux et positionné en décalage par rapport au premier ancêtre déjà positionné
- Le décalage est précisé avec les propriétés `top`, `bottom`, `left` et `right`
- La propriété `z-index` permet si besoin de préciser l'ordre de superposition

---

http://www.w3schools.com/cssref/pr_pos_z-index.asp

Exemple de positionnement absolu



http://blog.fourthbit.com/2013/11/27/essential-css-positioning

Positionnement fixe

position: fixed ;

- La valeur *fixed* est similaire à la valeur *absolute*, mais l'élément est positionné en décalage par rapport à la fenêtre du navigateur
- Le décalage est précisé avec les propriétés `top`, `bottom`, `left` et `right`
- Lors d'un défilement, l'élément fixe garde toujours la même position dans la page
- La propriété `z-index` permet si besoin de préciser l'ordre de superposition

Exemple de positionnement fixe et z-index

Page à l'état initial, puis après défilement (avec z-index)

Positionnement flottant

float: left | right | none | initial ;

- La propriété `float` indique que l'élément est flottant, i.e., il est retiré du flux et placé à gauche (valeur *left*) ou à droite (valeur *right*) de son élément parent
- Les éléments flottants se placent les uns à côté des autres, et en dessous si le conteneur parent n'est pas assez large

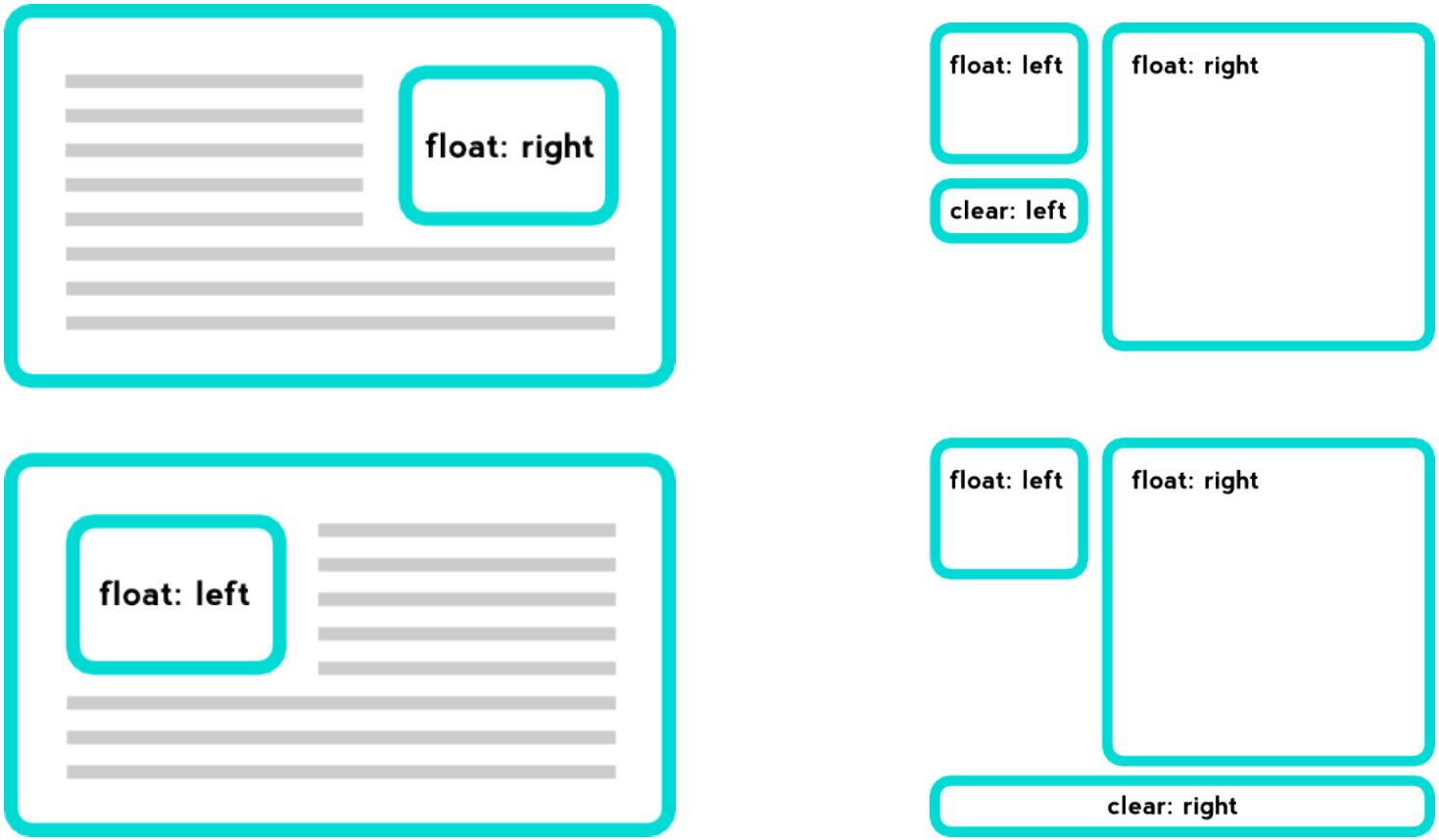
clear: left | right | both ;

- La propriété `clear` annule les conséquences d'un flottant pour les éléments qui suivent (ce qui permet de les placer dessous)
- Les valeurs *left*, *right* et *both* indiquent de placer l'élément sous des flottants situés à gauche, à droite ou des deux côtés

---

http://blog.fourthbit.com/2013/11/27/essential-css-positioning

Exemple de positionnement flottant



Des exemples flottants et d'utilisation de la propriété `clear`

http://blog.fourthbit.com/2013/11/27/essential-css-positioning

En résumé

- Le positionnement et le flottement (propriétés `position` et `float`) permettent de modifier l'affichage d'un élément par rapport au flux, à son type de boîte et à ses dimensions
- Ne pas abuser de ces positionnements (les flottants prévus initialement pour l'habillage de texte et non la mise en page)

---

http://www.alsacreations.com/article/lire/972-float-le-grand-bluff.html

Plan

Flexbox

- Modèle de boîtes "flex" proposé dans CSS3
- Agencement automatique et adaptatif à l'intérieur d'un composant
- Moins contraignant que les positionnements
- Complété par une grille *Grid* pour des projets plus complexes

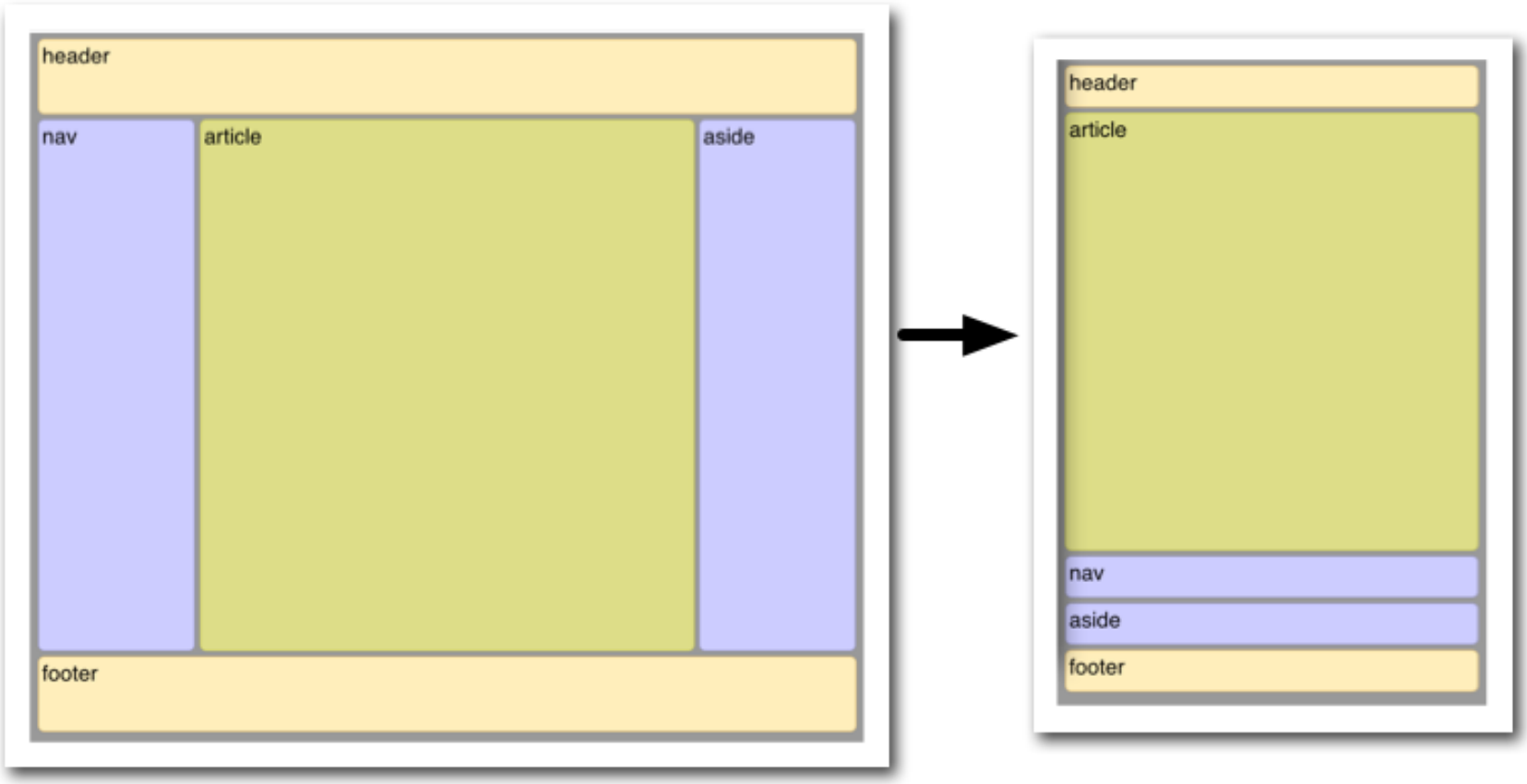
---

http://www.w3schools.com/css/css3_flexbox.asp

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Using_CSS_flexible_boxes

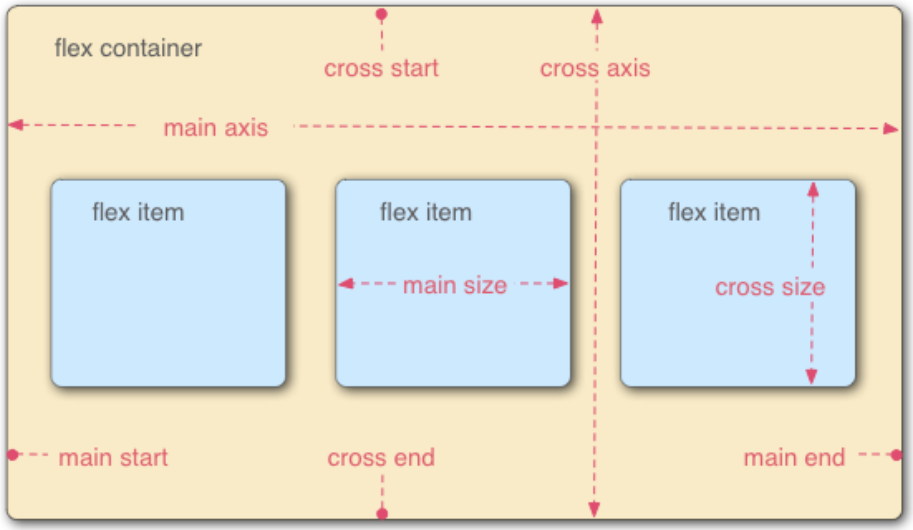
https://www.w3.org/TR/css-grid-1/

Adaptativité de Flexbox



Notions principales

- Un container flex définit un agencement pour les items flex qu'il contient
- Agencement selon l'axe principal (des items) et un axe secondaire (perpendiculaire à l'axe principal)
- Les autres éléments de la page sont affichés normalement



Container et items

```
display: flex ; /* container en mode bloc */
display: inline-flex ; /* container en mode inline */
```

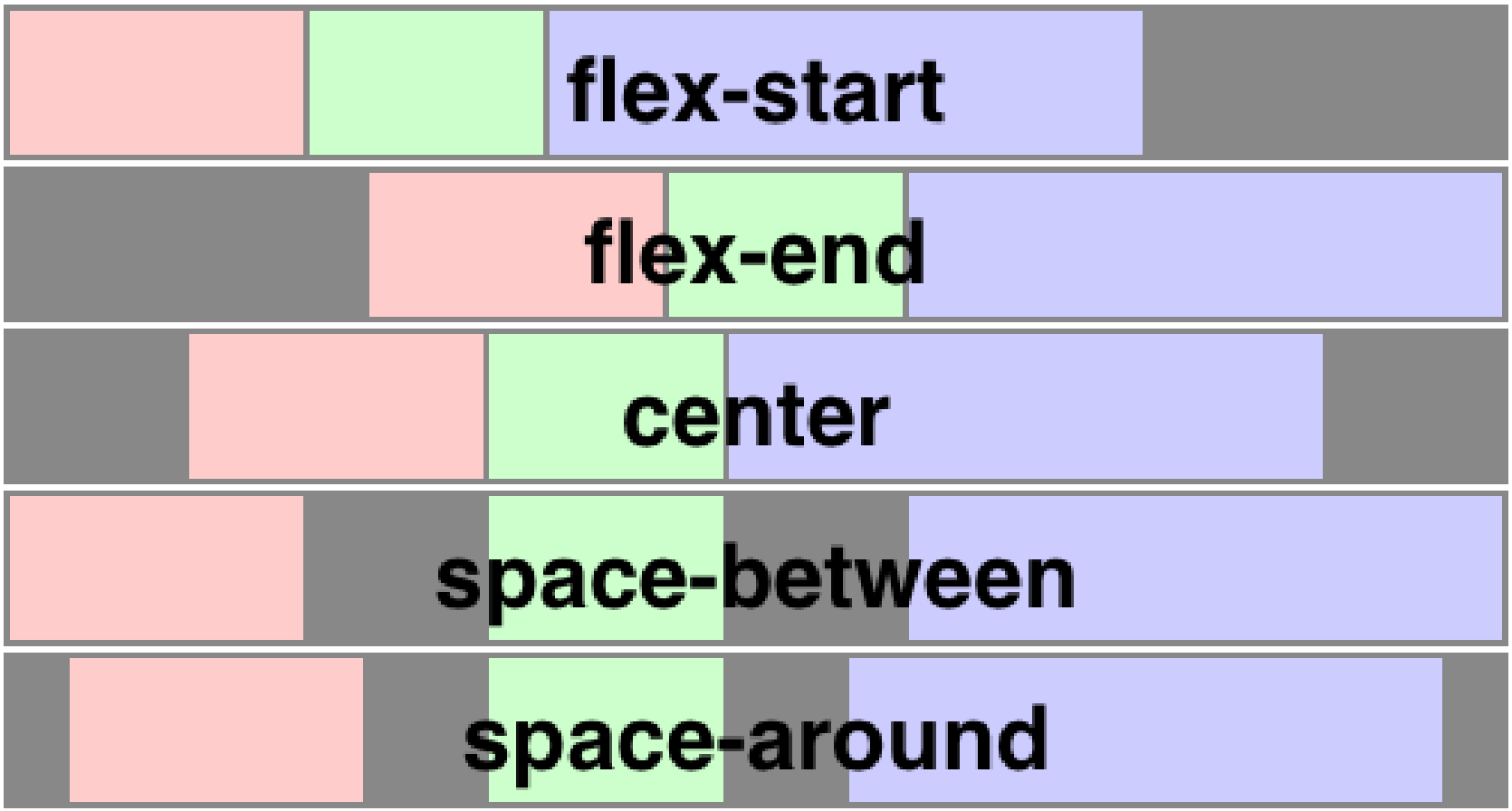
- La propriété `display` déclare un container flex avec un type de boîte *bloc* ou *inline*
- Tous les enfants directs du container sont des items flex
- Par défaut une "ligne" dans ce container pour accueillir les items flex

Alignement sur l'axe principal

justify-content: flex-start | flex-end | center | space-between | space-around ;

- La propriété `justify-content` aligne les items sur leur axe principal
- Valeurs : en début de container (défaut), en fin de container, au centre, avec des espaces entre items ou avec des espaces autour des items

Exemple d'alignement sur l'axe principal



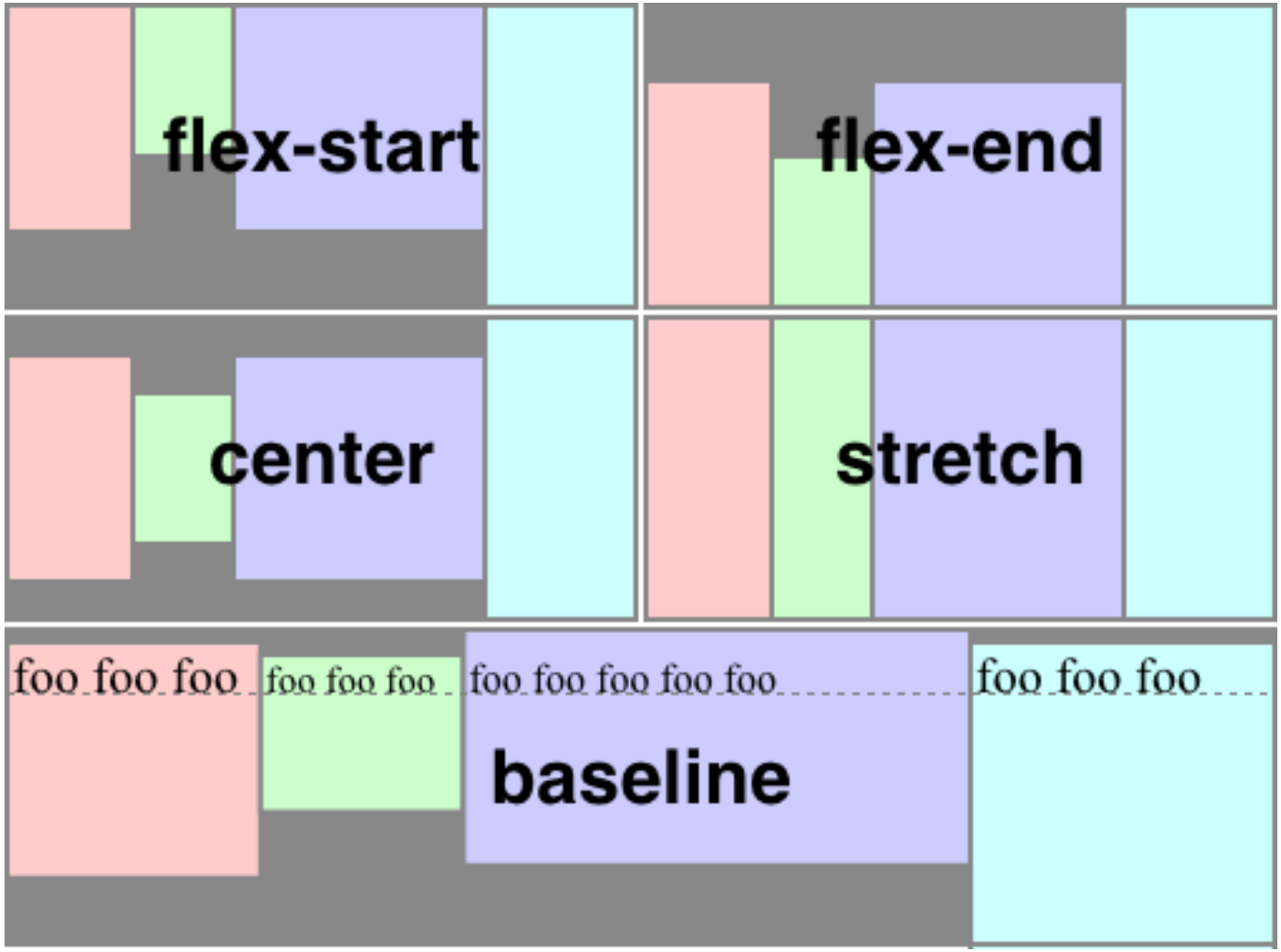
https://www.w3.org/TR/css-flexbox-1/

Alignement sur l'axe secondaire

```
align-items: flex-start | flex-end | center | strech | baseline ;
```

- La propriété `align-items` aligne les items sur leur axe secondaire
- Valeurs : en début de container, en fin de container, au centre, étiré au maximum (défaut) ou selon la ligne de base du texte

Exemple d'alignement sur l'axe secondaire



https://www.w3.org/TR/css-flexbox-1/

Autres propriétés de Flexbox

- L'axe principal des items est modifiable avec la propriété `flex-direction` (valeurs *row*, *row-reverse*, *column* et *column-reverse*)
- La direction des items du container est modifiable avec la propriété `direction` (valeurs *rtl* et *ltr*)
- Chaque item peut être ordonné différemment avec la propriété `order`
- Les propriétés `flex-wrap` et `align-content` permettent de préciser l'agencement des items sur plusieurs lignes
- La propriété `flex` définit la taille d'un item par rapport aux autres items du container
- ...

En résumé

Pour réaliser la mise en page avec le CSS :

- Privilégier l'ordre du flux et les dimensions
- Puis utiliser Flexbox (mais récent)
- Puis utiliser les types de boîtes, le positionnement et le flottement

Démo avec `demo2.html` (code source en ligne)