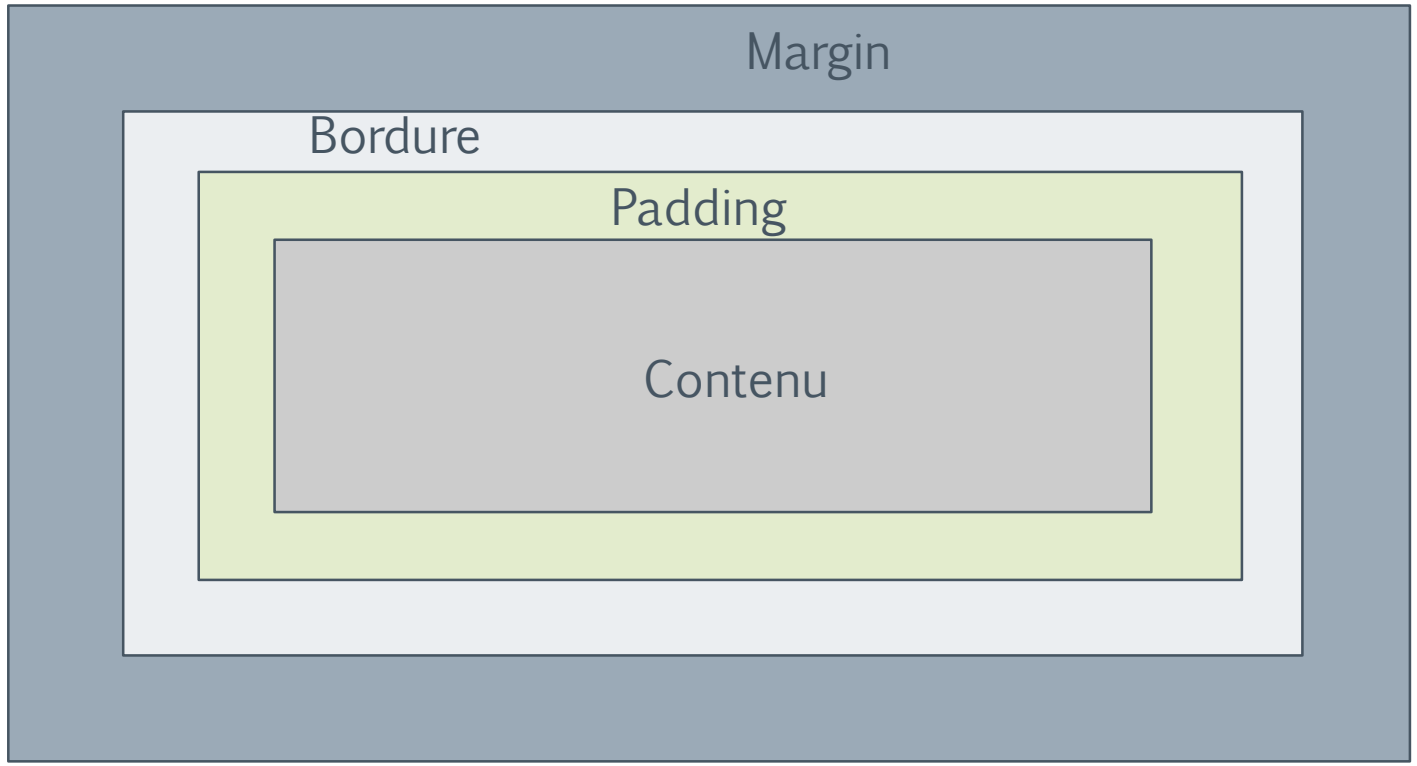
Création des pages web

CM 3 : CSS avancé

Modèle de boite CSS



Taille de la boite

› Le **contenu** peut être redimensionné à l'aide de *width* et *height:*

```
<style>
.size {
  border: 1px solid black;
  width: 100px;
  height: 200px;
  background-color: lightgrey;
}
</style>


<div class="size">Size test</div>
```

Attention: La largeur totale de l'élément sera width+padding+margin

Positionnement

› Il est possible de changer la position d'un élément sur la page à l'aide de la propriété *position* et *left, right, top, bottom*:

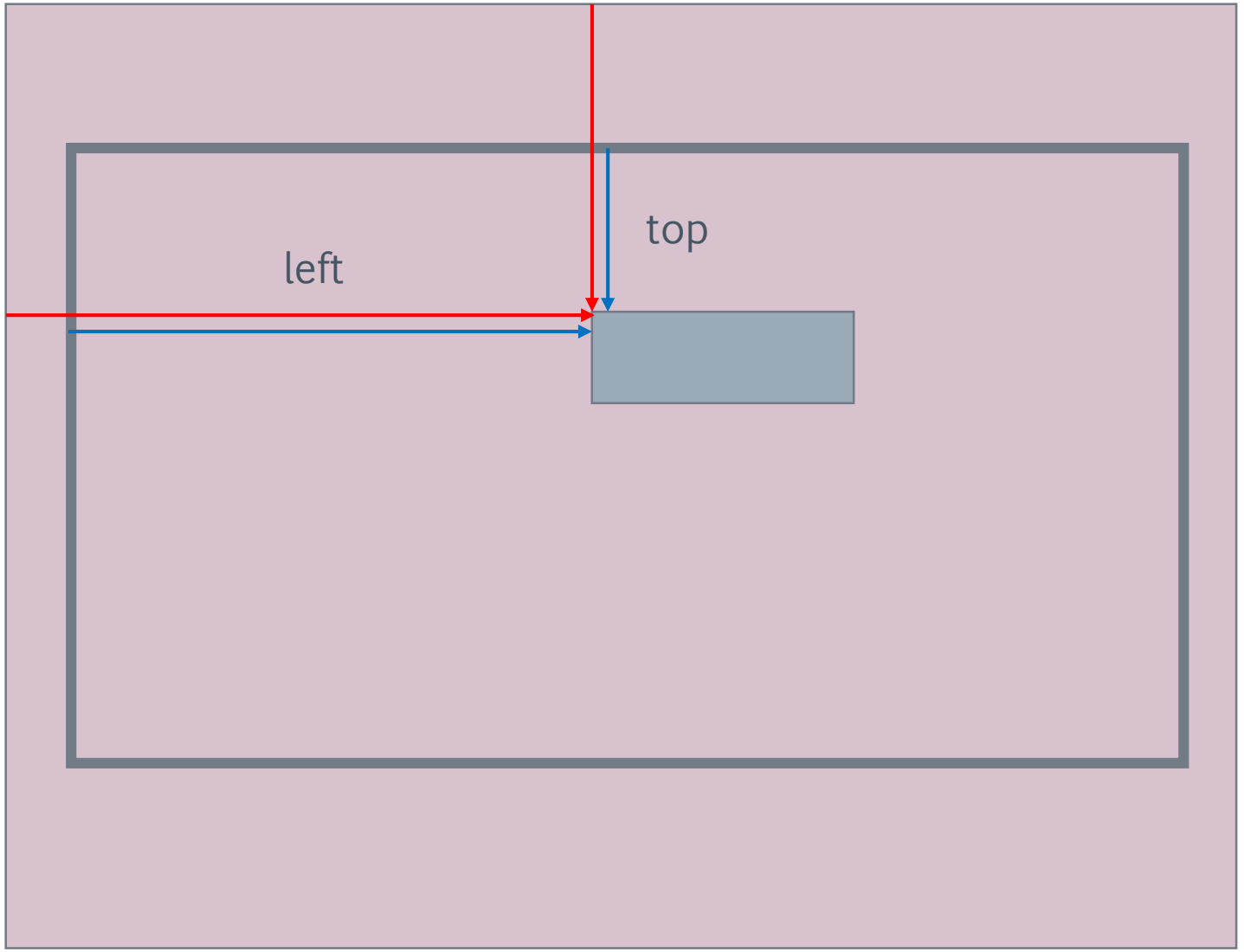
```
<style>
.position {
  border: 1px solid black;
  width: 100px;
  height: 200px;
  background-color: lightgrey;
  position: relative;
  left: 20px;
}
</style>


<div class="position">test position</div>
```

Types de position

static	Valeur par défaut (pas de possibilité de déplacement).
relative	Valeur relative par rapport à la position normale (en utilisant left/top/etc)
fixed	Valeur fixe par rapport à l'écran. Left/top/etc donnent les coordonnées.
absolute	Valeur fixe par rapport à la page. Left/top/etc donnent les coordonnées.
sticky	Relative ou fixe en dépendance du défilement de l'écran.

z-index – indice numérique pour superposer des éléments.

Fixed et absolute



Exemple

```
<style>
.position {
  border: 1px solid black;
  width: 100px;
  height: 100px;
  background-color: lightgrey;
  position: fixed;
  left: 200px;
  top: 200px;
}
</style>

<div class="position">test position</div>
```

Types de blocks

› Des éléments html peuvent avoir un comportement différent lors du placement. Leur type est dirigé par le paramètre display:

none	L'élément n'est pas affiché.
block	L'élément commence à une nouvelle ligne et prend toute la largeur de la page (vers la gauche et vers la droite). Exemples: <p>, , <h1>, <section>
inline	L'élément ne commence pas à une nouvelle ligne et prend autant de place que nécessaire. Exemples: , <a>, <img>
inline-block	L'élément se comporte comme inline, mais il est possible de lui appliquer des dimensions (width et height)
flex	L'élément correspond à un conteneur flexbox
grid	L'élément correspond à un conteneur grid
…	

Exemple

```
<style>
.position {
  border: 1px solid black;
  width: 100px;
  height: 100px;
  background-color: lightgrey;
  display: inline-block;
}
</style>


<p>Texte avant <div class="position">test
position</div> <p>texte apres</p>
```

Exemple: bouton

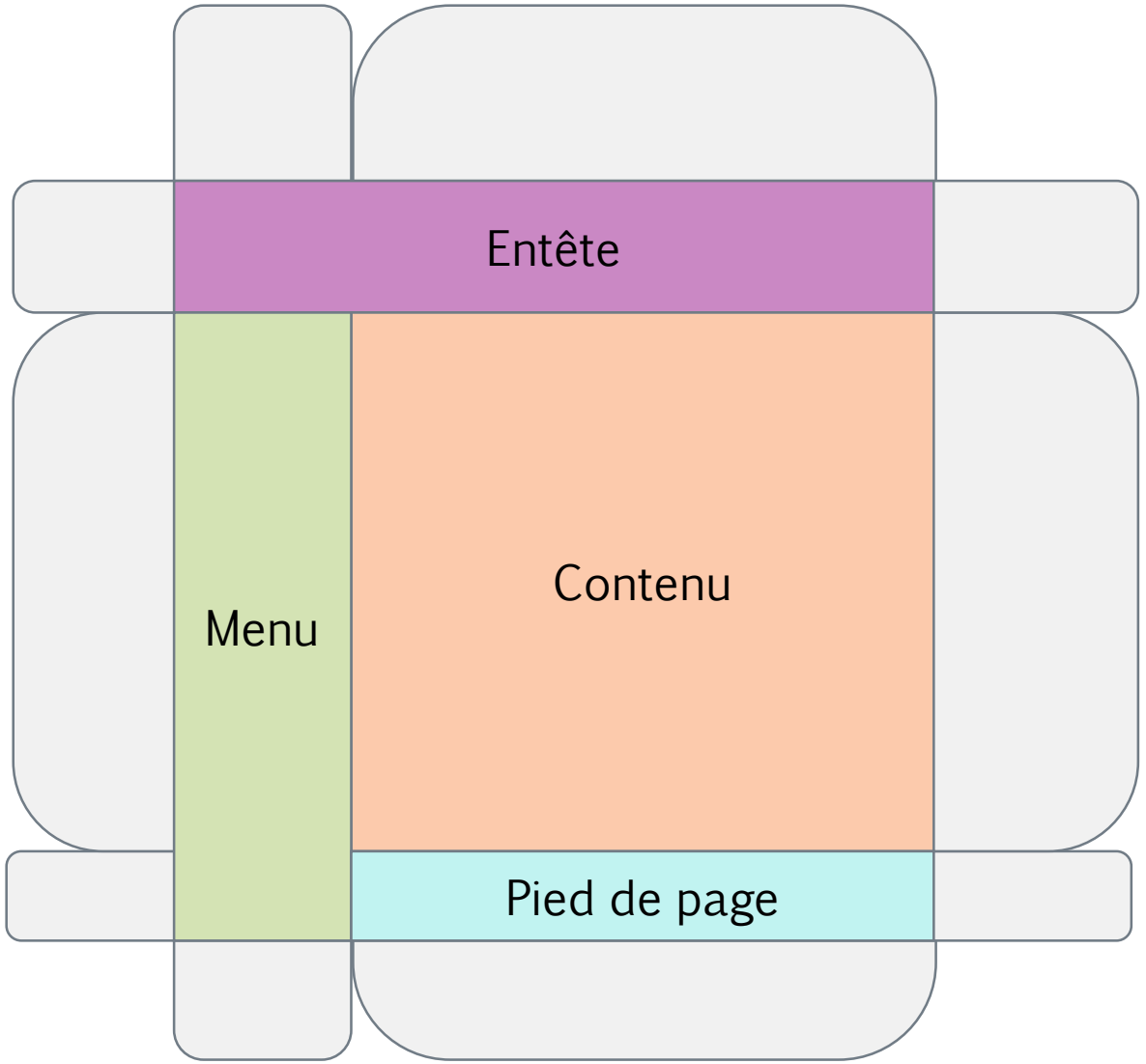
```
.btn {
        background-color: red;
        border: 1px solid black;
        padding: 5px;
        text-decoration: none;
        color: black;
        width: 50px;
        height: 100px;
        display: inline-block;
    }
```

```
<a class="btn" href="https://www.u-ec.fr/">UPEC</a>
```

Conteneur grid

- Permet de définir la structure de la page en termes de lignes et colonnes (en indiquant précisément leur nombre et leur tailles).

Exemple



Implémentation

- Dans conteneur:
  - `display: grid`
  - grid-template-rows: *dimensions des lignes*
    exemple de 3 lignes (1ère à 100px, 3e à 50px et la 2e prend l'espace restant) :
    ```
    grid-template-rows: 100px auto 50px
    ```
  - grid-template-columns: *dimensions des colonnes*
    exemple de 2 colonnes (1ère à ¼ de la largeur et la 2e à ¾ de largeur):
    ```
    grid-template-columns: 3fr 9fr;
    ```

- Pour chaque élément:
  - grid-row ou grid-column pour préciser l'emplacement
    exemple début colonne 1 et fin colonne 2:
    ```
    grid-column: 1 / 3
    ```
    exemple 2 lignes à partir de la ligne 2 :
    ```
    grid-row:  2 / span 2;
    ```

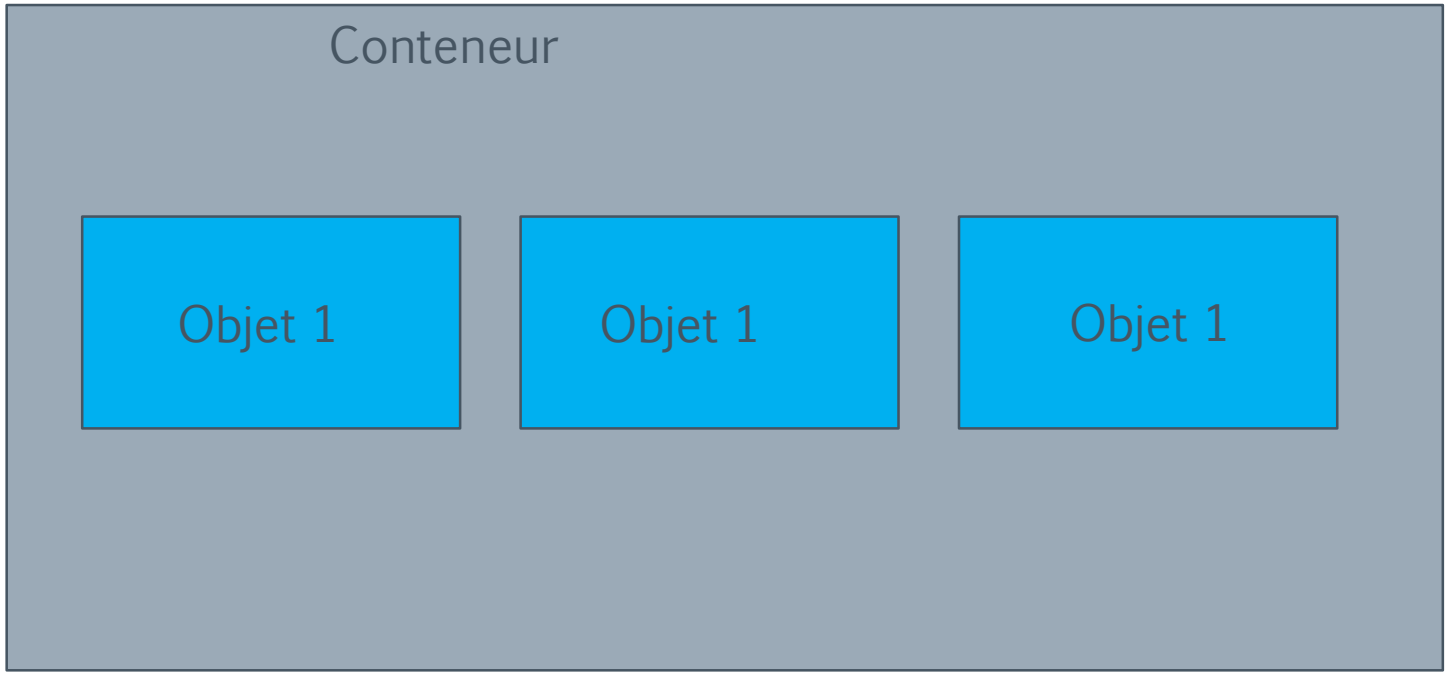
Exemple (code html)

```
<html>
   <head>
      <title> Trame de page. </title>
      <meta charset="utf-8" />
      <style> …  </style>
    </head>
    <body>
      <div id="main">
        <header>Entête</header>
        <nav>Menu</nav>
        <section>Contenu</section>
        <footer>Pied de page</footer>
      </div>
    </body>
</html>
```

Exemple (styles css)

```css
header {
   background-color: violet;
   grid-column: 1 / 3;
 }

 nav {
   background-color: lightgreen;
   grid-row: 2 / span 2;

 }

 section {
   background-color: lightsalmon;
   height: 400px;
 }

 footer {
   background-color: lightblue;
 }

 #main {
   display: grid;
   grid-template-rows: 100px auto 50px;
   grid-template-columns: minmax(200px, 3fr) 9fr;
 }

body {
  margin: 0px;
}
```

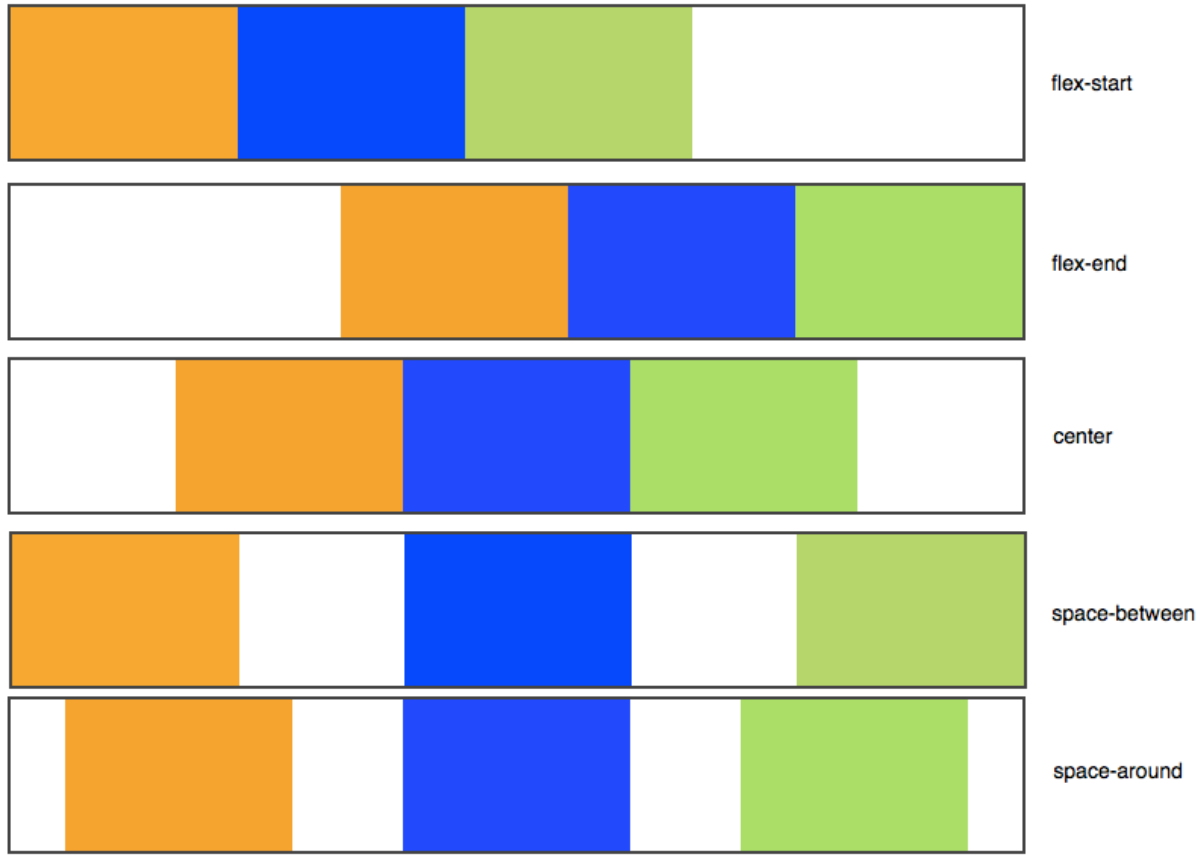
Conteneurs Flexbox



Flexbox

```
<div id="conteneur">
    <div class="element">Elément 1</div>
    <div class="element">Elément 2</div>
    <div class="element">Elément 3</div>
</div>

#conteneur
{
    display: flex;
    flex-direction: column; /* row */
}
```

Flex: justify-content



Flex: align-items

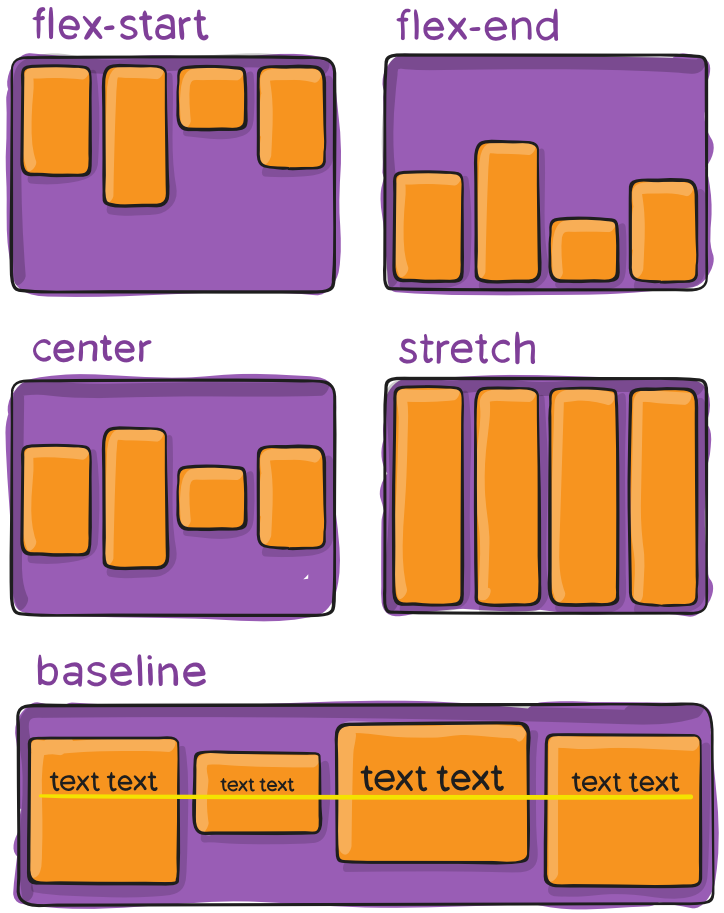


Image de: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Centrer avec flexbox

```
#conteneur{
        display: flex;
        flex-direction: row;
        justify-content: center;
        align-items: center;
}
```

```
#conteneur {       display: flex; }
```

```
.element {       margin: auto; }
```

Centrer avec margin (limité)

```
#container {
        margin: auto;
}
```

Flexbox & grid

- On peut évidemment emboiter les flexbox et les grid.
- On obtient donc un outil très puissant permettant à placer les éléments précisément.

Transformations 2d/3d

- Propriété ***transform***
  - *translate(), rotate(), scaleX(), scaleY(), skew(), matrix(), ...*
  - *rotateX(), rotateY(), rotateZ(), matrix3d, scale3d(), translate3d(),...*

Exemple:

```
div {
    transform: rotate(20deg);
}
```

Animations

- Utiliser transform, transition-duration et :hover

```
.anim:hover {
   transform: rotate(90deg);
   transition-duration: 500ms;
}
```

Utiliser animation et @keyframes:

```
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}

.anim:hover  {
  animation-name: example;
  animation-duration: 4s;
}
```

Créer un menu

- Horizontalement – une liste avec flexbox
- Verticalement – des liens avec flexbox
- Déclenchement à :hover

Créer un menu (code html)

```
<nav id="menu">
  <ul>
    <li><a href="#">Item 1</a></li>
    <li><a href="#">Item 2</a></li>
    <li class="dropdown">
      <a href="#">Item 3</a>
      <div class="dd-content">
        <a href="#"> DD item 1</a>
        <a href="#"> DD item 2</a>
        <a href="#"> DD item 3</a>
      </div>
    </li>
    <li><a href="#">Item 4</a></li>
  </ul>
</nav>
```

Styler la partie horizontale

```
#menu {
    background-color: black;
    font-size: large;
}
#menu > ul {
    list-style-type: none;
    display: flex;
    flex-direction: row;
    margin: 0;
    padding: 0;
}
#menu li a {
    background-color: black;
    text-align: center;
    text-decoration: none;
    color: white;
    text-align: left;
    padding: 0 5px;
}
#menu li a:hover {
    background-color: red;
}
```

Styler la partie verticale

```
#menu .dropdown .dd-content {
  display: none;
  position: absolute;
}

#menu .dropdown:hover .dd-content {
  display: flex;
  flex-direction: column;
}

#menu .dropdown:hover .dd-content a {
  background-color: lightgrey;
  color: black;
  padding: 10px 0px;
}

#menu .dropdown:hover .dd-content a:hover {
  background-color: darkgrey;
}
```

Formulaires HTML

- HTML contient également des éléments d'entrée utilisés pour l'interaction regroupés dans des formulaires:
  - Champs d'entrée texte.
  - Champs d'entrée numériques.
  - Listes déroulantes.
  - Boutons radio.
  - Cases à cocher.
  - Boutons.
- Ils s'affichent et se stylent comme tous les autres éléments HTML.
- Chaque formulaire est englobé dans la balise *form*.

Formulaires HTML

<input type="text">	Entrée texte
<input type="number">	Entrée numérique
<input type="date">	Entrée date
<input type="password">	Entrée mdp
<input type="radio">	Buton radio
<input type="checkbox">	Case à cocher
<textarea>...</textarea>	Entrée paragraphe
<select> + <option>	Liste déroulante
<button> ou <input type="button">	Bouton

Exemple

```
<form action="https://...">
   <p>Login: <input type="text"></p>
   <p>Mdp: <input type="password"></p>
   <p><button>Login</button></p>
 </form>
```