Positionnement flottant

- A l'origine, le **flottement** sert à pousser, à droite ou à gauche, du texte ou une image, et à remplir l'espace libre avec le reste du contenu
- Cet usage a été **détourné pour positionner** des éléments, mais il peut varier selon le navigateur d'où des erreurs d'affichage
- Un élément flottant est sorti du flux et placé à l'extrême gauche (**float:left**) ou droite (**float:right**) de son conteneur tout en restant sur sa hauteur de ligne initiale, le reste s'écoule autour.

float:left – float:right

- L'élément défini avec **float:left** est fixé à gauche et la suite du contenu occupe tout le reste à droite

- Avec **float:right**, on a le comportement inverse
- **float:none**, par défaut

Nouveautés à propos de float

- La norme CSS3 évolue sans cesse : d'autres valeurs pour le flottement ont été ajoutées

  inline-start et **inline-end**

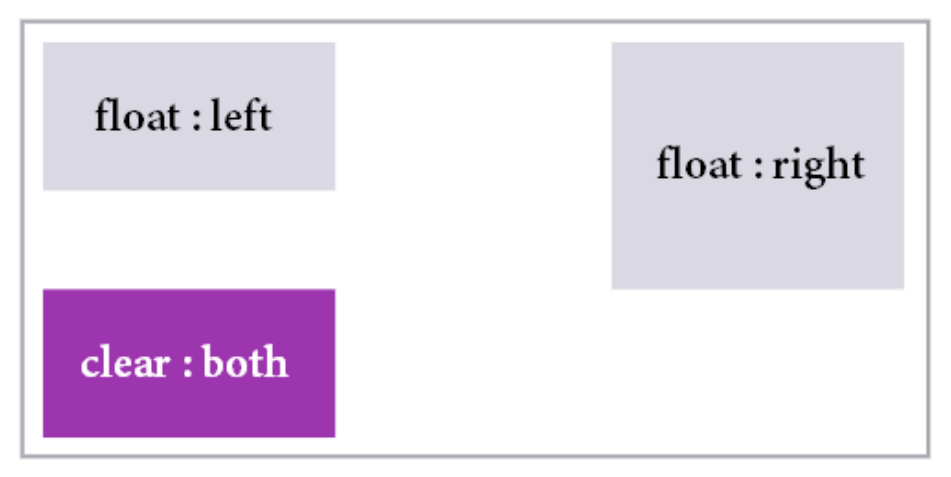
  exemple pour la prise en charge des langues
- On parle aussi des propriétés logiques (voir MDN) qui sont influencées par les propriétés **direction** (MDN) et **writing-mode** (MDN)
- Pour plus d'explications détaillées sur float
- Support variable selon les navigateurs (voir MDN et CanIUse.com)

Annuler le flottement

- La propriété **clear** interdit à un élément de se placer sur la même ligne qu'un bloc flottant et le force à passer en-dessous
- On annule les flottants à gauche (**clear:left**) ou à droite (**clear:right**) ou les deux (**clear:both**)
- **clear** permet, d'**empêcher les dépassements** de flottants, ou de placer un élément **toujours en bas** du flottant le plus long

clear:both

- Avec la propriété **clear:both**, le dernier élément ne tient plus compte des positionnements flottants définis pour les éléments précédents
- Il reprend donc la position par défaut



Plusieurs float

- On peut positionner plusieurs éléments avec la propriété **float:left** ce qui permet d'afficher des blocs successifs sous forme de colonnes.
- Si l'addition des largeurs des différents blocs dépasse la largeur totale, le dernier bloc passe sous les autres.

Des blocs côte à côte

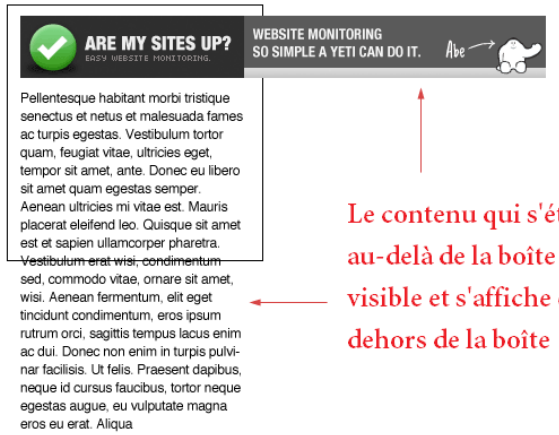
- Un élément **flottant** prend par défaut la **largeur de son contenu**. Si le contenu est important, il vaut mieux fixer avec width ou max-width
- S'il y a un **2ème élément flottant**, il reste sur le même plan et donc se place à ses côtés
- Attention, les flottants sortent du flux, donc le parent sera vide et le contenu va dépasser. Il faut donc **fixer la hauteur** du parent ou placer un **élément non flottant** après.
  => Solution : le « clearfix hack » (ex1) (ex2)
  => Mieux : utiliser Flexbox ou CSS Grid

Dépassement

- La propriété **overflow** définit la méthode de dépassement des textes de leur conteneur.
  - **visible** : le contenu qui sort est visible.
  - **hidden** : le contenu est rogné aux limites du conteneur, et pas de barre de défilement
  - **scroll** : le contenu est rogné aux limites du conteneur, et des barres de défilement
  - **auto** : le choix est laissé aux navigateurs.
- CSS3 étend ce mécanisme en proposant les propriétés, **overflow-x** et **overflow-y**

Valeurs pour overflow

```
.box { overflow: visible; }
```

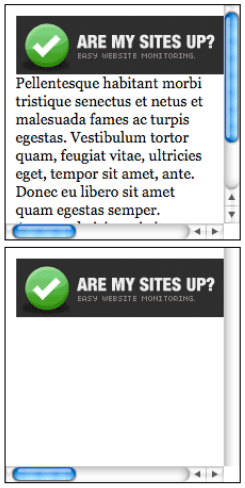


Le contenu qui s'étend au-delà de la boîte est visible et s'affiche en dehors de la boîte

```
.box { overflow: hidden; }
```

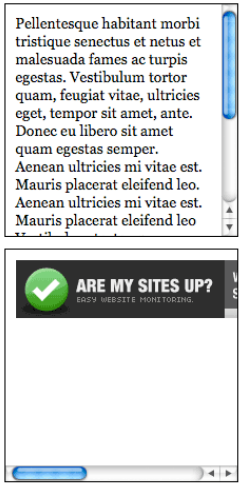
Tout contenu qui s'étend au-delà de la boîte est complètement caché

```
.box { overflow: scroll; }
```



Mettre la valeur scroll à la propriété overflow va provoquer l'affichage des scrollbars aussi bien verticalement que horizontalement (même si elles ne sont pas nécessaires dans les deux directions)

```
.box { overflow: auto; }
```



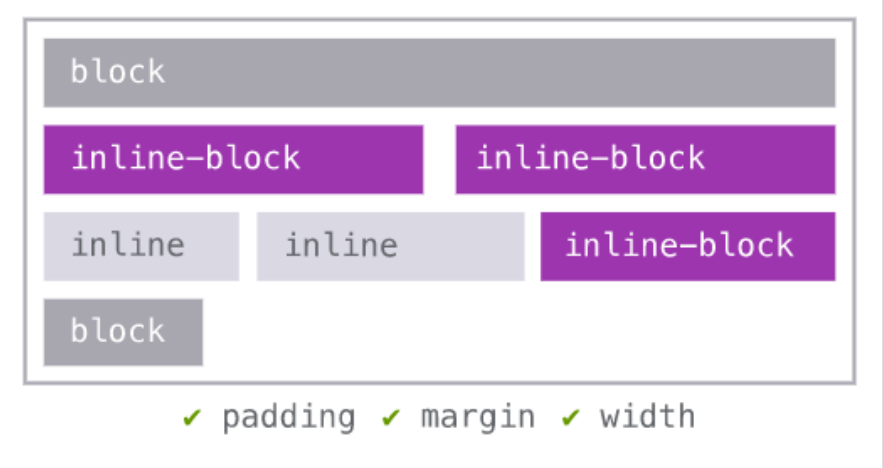
Mettre la valeur auto à la propriété overflow va provoquer l'affichage des scrollbars uniquement si elles sont nécessaires

Evolution de overflow

- La norme CSS3 continue ici aussi d'évoluer pour améliorer l'affichage des informations (W3C)
- Pour d'autres explications détaillées sur overflow et ses variantes
- Propriétés à ne pas confondre avec la propriété **text-overflow**
- Récemment, les propriétés **overflow-inline** et **overflow-block** ont été ajoutées mais ne sont pas encore bien supportées sur les navigateurs

Inline-block

- Avec la propriété **display:inline-block;** on peut combiner les avantages des 2 types principaux
  - Inline : des blocs peuvent se placer l'un à côté de l'autre et rester verticalement alignés
  - Block : pouvoir être dimensionnés
- Avec la propriété **vertical-align,** on peut régler l'alignement des blocs selon la valeur (top, baseline, bottom, middle,...)



Rendu de tableau en CSS

- Il est possible de faire une **mise en page** sous forme de **tableau** et sans utiliser les balises <table>, <tr>, <td> !
- Mais avec des valeurs spéciales pour **display**
  - **table, table-row, table-cell**
  - **table-header-group, table-footer-group**
  - **table-caption,…**
- Parfois utilisé pour aligner des éléments façon tableau (ex : formulaire) mais il faut modifier le conteneur et tous ses éléments internes