



Conception de pages Web (HTML/CSS)

Agencement

Enzo Doyen

enzo.doyen@unistra.fr

2026 - LGA1DMT2

Plan

- I. Éléments « block » et « inline »
- II. Unités de mesure
- III. Conteneurs
- IV. Modèles de boîte (*box models*)
- V. Positionnement

Récapitulatif du cours précédent

Nous avons vu comment styliser nos pages Web pour les rendre encore plus attrayantes à l'aide de CSS.

Ce cours-ci est une extension de ce que nous avons vu précédemment. Nous allons voir comment agencer et positionner les éléments de nos pages Web pour les rendre encore plus personnalisables.



I. Éléments « block » et « inline »

Éléments « **block** » et « **inline** »

Tous les éléments HTML ont, par défaut, une propriété CSS `display`. Celle-ci définit l’affichage de ces éléments.

La propriété `display` peut prendre plusieurs valeurs, mais les plus courantes sont **block** et **inline**.

Éléments « block »

Les éléments de type **block** commencent toujours sur une nouvelle ligne et prennent toute la largeur disponible.

Nous en avons vu plusieurs jusqu'à présent : les balises `p`, `ul` (pour les listes) ou encore `h1` à `h6` (pour les titres) sont des éléments de type **block**.

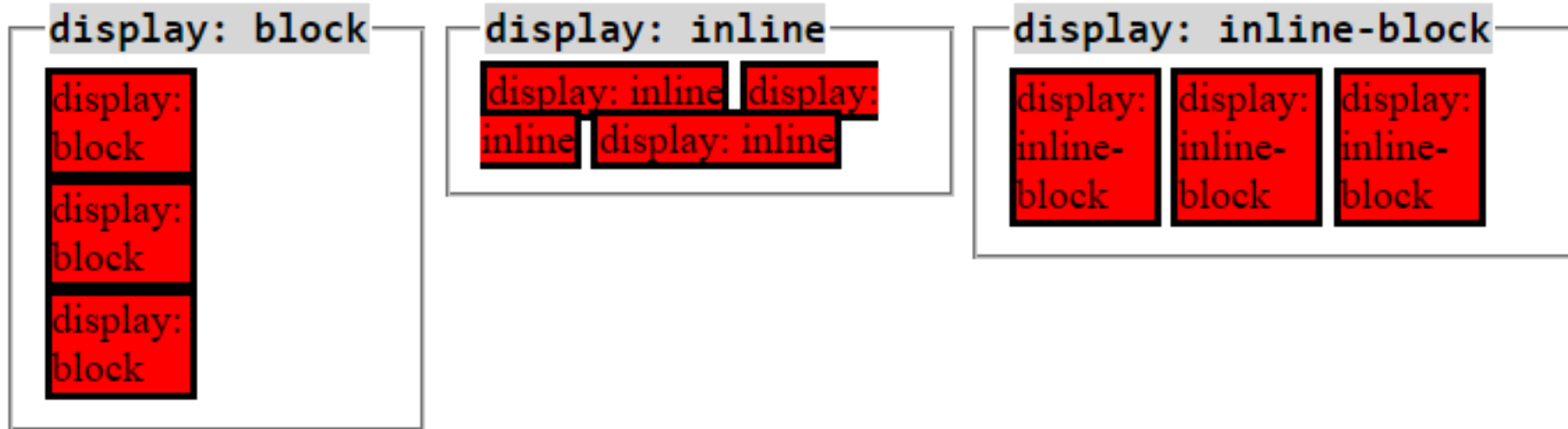
Éléments « inline »

Les éléments de type **inline** ne commencent pas sur une nouvelle ligne et ne prennent que la largeur nécessaire.

Nous avons aussi vu plusieurs éléments de type **inline** : les balises `a`, `strong` ou encore `img`.

Agencement selon la valeur de `display`

Il est possible de modifier la valeur de la propriété `display` de n'importe quel élément, peu importe sa valeur initiale.





II. Unités de mesure

Unités absolues et relatives

On distingue deux types d'unités de mesure en HTML/CSS : les unités **absolues** et les unités **relatives**.

Unités absolues

On distingue deux types d'unités de mesure en HTML/CSS : les unités **absolues** et les unités **relatives**.

Les unités absolues restent identiques peu importe les éléments autour. Par exemple, px est une unité absolue car le nombre de pixels est défini à l'avance et fixe.

Unités absolues communément utilisées :

- ♦ px (pixels) ;
- ♦ cm (centimètres), mm (millimètres) ;
- ♦ in (pouces) ;
- ♦ pt (points).

Unités absolues : exemple

```
1  .box {  
2    width: 5cm; /* largeur */  
3    height: 20px; /* hauteur */  
4    font-size: 14pt; /* taille police */  
5  }
```



Unités relatives

On distingue deux types d'unités de mesure en HTML/CSS : les unités **absolues** et les unités **relatives**.

Les unités relatives s'ajustent en fonction des autres éléments (éléments parents) ou de la taille de la fenêtre.

Unités relatives communément utilisées :

- ♦ em (relatif à la taille de police de l'élément parent) ;
- ♦ rem (relatif à la taille de police de l'élément racine <html>) ;
- ♦ % (relatif à la taille de l'élément parent) ;
- ♦ vh (% de la hauteur de la fenêtre) ;
- ♦ vw (% de la largeur de la fenêtre).

Unités relatives : exemple

```
1  .box {  
2    width: 50vw; /* 50 % de la largeur de la fenêtre  
   */  
3    height: 20%; /* 20 % de la hauteur de l'élément  
   parent */  
4    font-size: 1.3em; /* 1.3 fois la taille de  
   police de l'élément parent */  
5  }
```





III. Conteneurs

Conteneurs

Deux éléments HTML sont fréquemment utilisés comme « conteneurs », c'est-à-dire pour imbriquer/regrouper d'autres éléments : `div` et `span`.

div

La balise `div` est un élément de type **block** qui permet de regrouper plusieurs éléments ensemble **sous forme de bloc**.

```
1 <div class="section">
2   <h1>Titre</h1>
3   <p>Paragraphe</p>
4 </div>
```



span

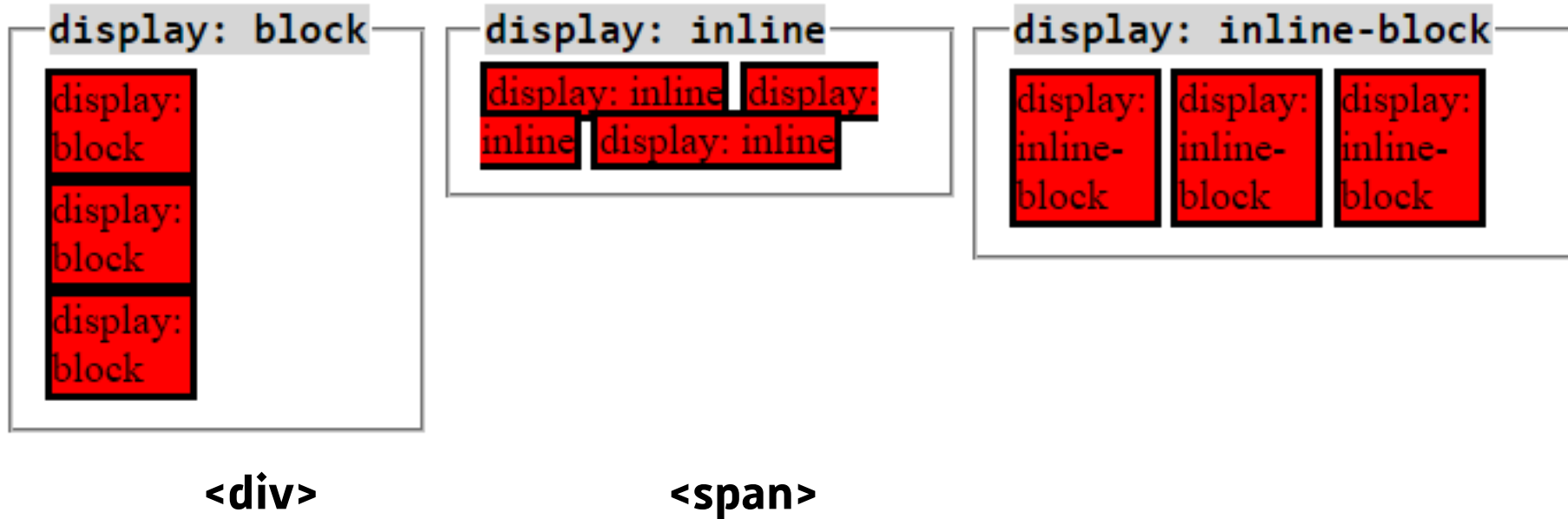
La balise span est un élément de type **inline** qui est généralement utilisé comme **conteneur** pour du texte, ou bien pour regrouper plusieurs éléments **en ligne**.

Elle est très pratique pour appliquer des styles à une partie du texte.

```
1 <h1>Mon <span class="highlight">super</span> titre</h1>
```



Rappel sur l'agencement des éléments





IV. Modèles de boîte (*box models*)

Modèle de boîte

Chaque élément sur une page Web est considéré comme une boîte rectangulaire, avec une hauteur et une largeur définies, celles-ci étant généralement proportionnelles au contenu de l'élément.

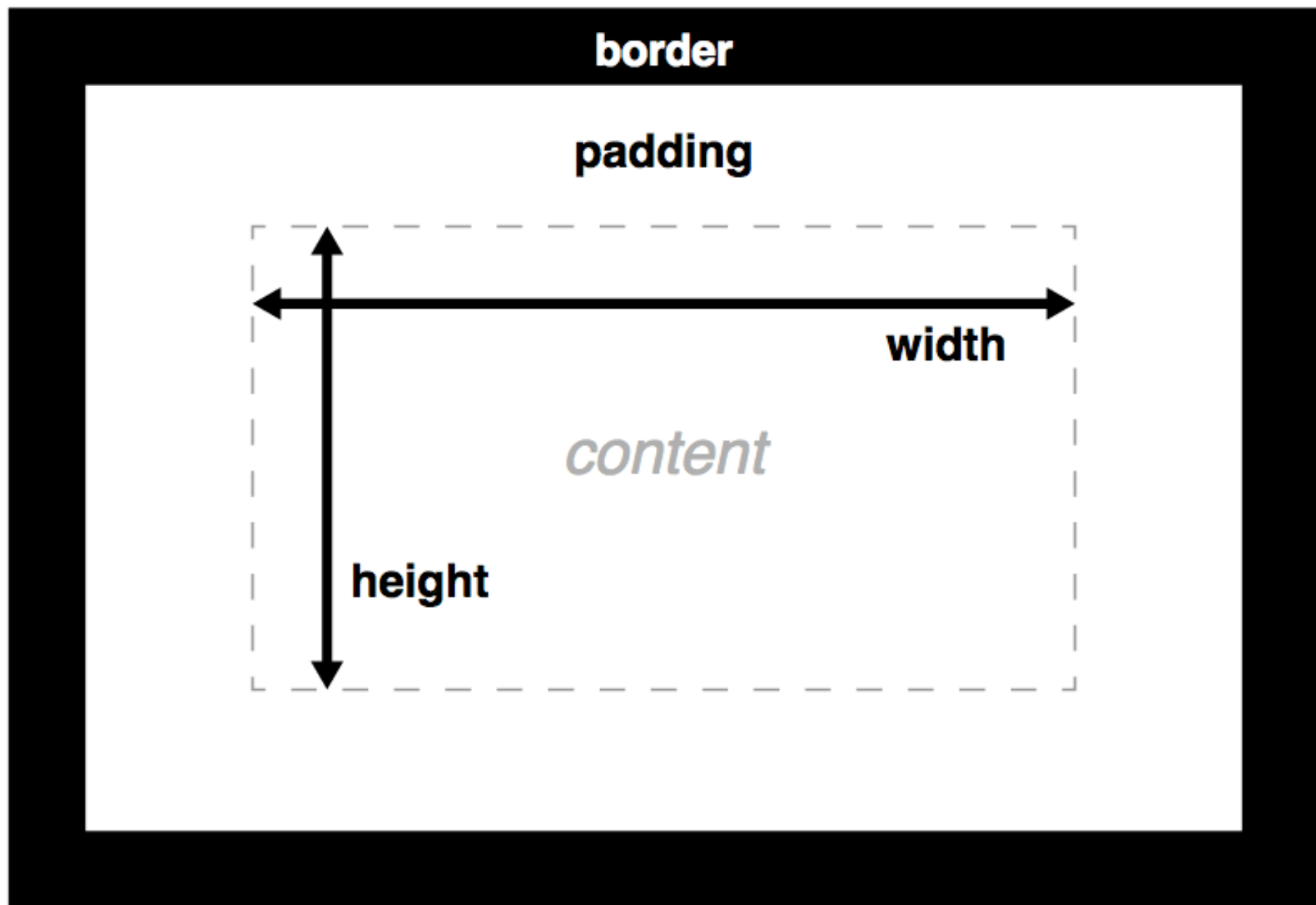
border

padding

width

content

height



Modèle de boîte

Il est possible d'appliquer plusieurs propriétés CSS à ces boîtes : on utilise notamment les propriétés `border`, `margin`, `padding`.

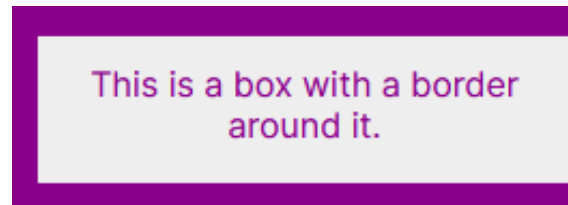
`border` sur MDN : <https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Properties/border>

`margin` sur MDN : <https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Properties/margin>

`padding` sur MDN : <https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Properties/padding>

Modèle de boîte : border

```
1  .box {  
2    border: 1rem solid #b0008b; /* bordure de 1rem  
   de large, de couleur #b0008b (violet) */  
3    background: #eee; /* couleur de fond */  
4  }
```



Modèle de boîte : `margin`

La propriété `margin` permet de définir l'espace **entre la bordure de l'élément et les éléments autour**. Il s'agit de l'espace **après la bordure**.

[Voir les exemples dans la documentation MDN](#)

Modèle de boîte : `margin`

Cette propriété peut prendre jusqu'à quatre valeurs pour définir l'espace autour de l'élément.

```
1 /* haut | droite | bas | gauche */  
2 .box {  
3     margin: 2px 1em 0 auto;  
4 }
```



Modèle de boîte : `margin`

Si une seule valeur est fournie, elle sera utilisée pour les quatre côtés.

```
1  .box {  
2    margin: 1em;  
3  }
```



Modèle de boîte : margin

On peut aussi définir l'espace autour de l'élément de chaque côté avec `margin-top`, `margin-right`, `margin-bottom` et `margin-left` :

```
1  .box {  
2    margin-top: 1em;  
3    margin-left: 2em;  
4  }
```



Modèle de boîte : padding

La propriété padding permet de définir l'espace **entre l'élément et la bordure**. Il s'agit de l'espace **avant la bordure**.

Pour le nombre de valeurs, les mêmes règles que pour margin s'appliquent. On peut aussi utiliser padding-top, padding-right, padding-bottom et padding-left.

[Voir les exemples dans la documentation MDN](#)

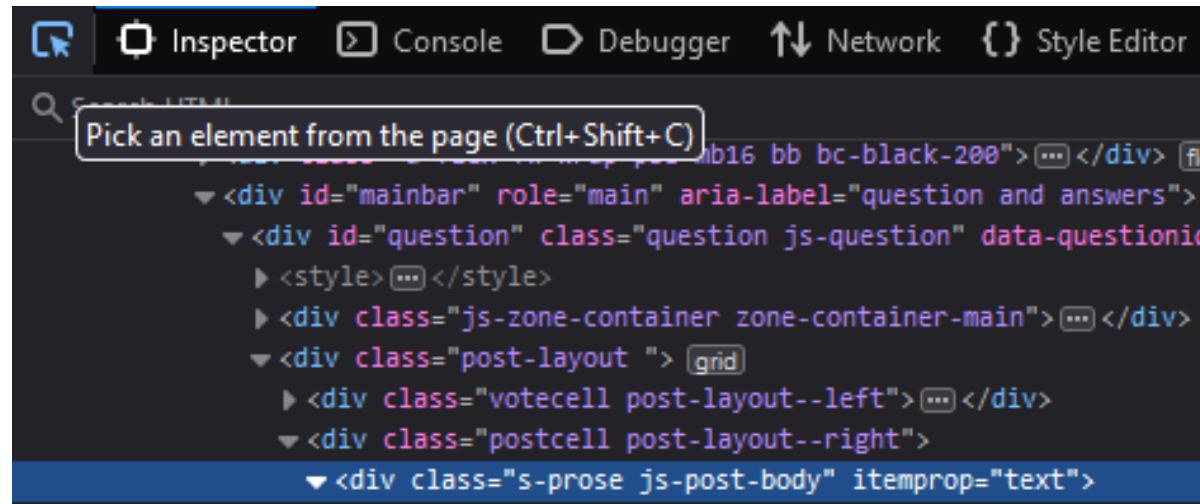
Modèle de boîte : `margin` OU `padding` ?

La plupart du temps, `margin` et `padding` permettent d'arriver au même résultat ; il y a néanmoins quelques subtilités :

- ♦ `margin` définit l'espace **autour de l'élément**, tandis que `padding` définit l'espace **à l'intérieur de l'élément** ;
- ♦ `padding` fait **partie de l'élément**, contrairement à `margin`.

Voir les modèles de boîte sur les pages Web

Grâce à la console de développement (F12), il est possible de visualiser les modèles de boîte.



Voir les modèles de boîte sur les pages Web

Advice

div#content | 1100 x 2715.03 | Flex Item

Does the order of ingredients while cooking matter to an extent that it changes the overall taste of the food?

Ask Question

Asked 2 days ago Modified today Viewed 382 times

▲

3

▼

🔖

🕒

Let's say I am making a meat curry. In first case, I boiled the meat in water, added the pepper first, let it cook for a while, then added some chilli powder. In the second case, I added chilli first, let it boil for some time, and then added pepper. Will there be a discernible change in the taste? I think what I am asking is does the law of superposition hold in cooking? It may not be observable since the width of each flavor layer would be infinitely miniscule, but is it discernible in the overall taste of the food? Scientific, anecdotal, all answers are appreciated!

Edit: Following the suggestions, I will narrow it down. The sequence of ingredients should mean that the meat is first exposed to those chemical reactions and the flavor fog(that it will absorb) of one ingredient, then of the other ingredient. Do you think this results in a first-absorbed-last-tasted flavor stack that we may not always put our hand on, but taste subconsciously?

Featured on Meta

🔖

Bringing clarity to status tag usage on meta sites

🔖

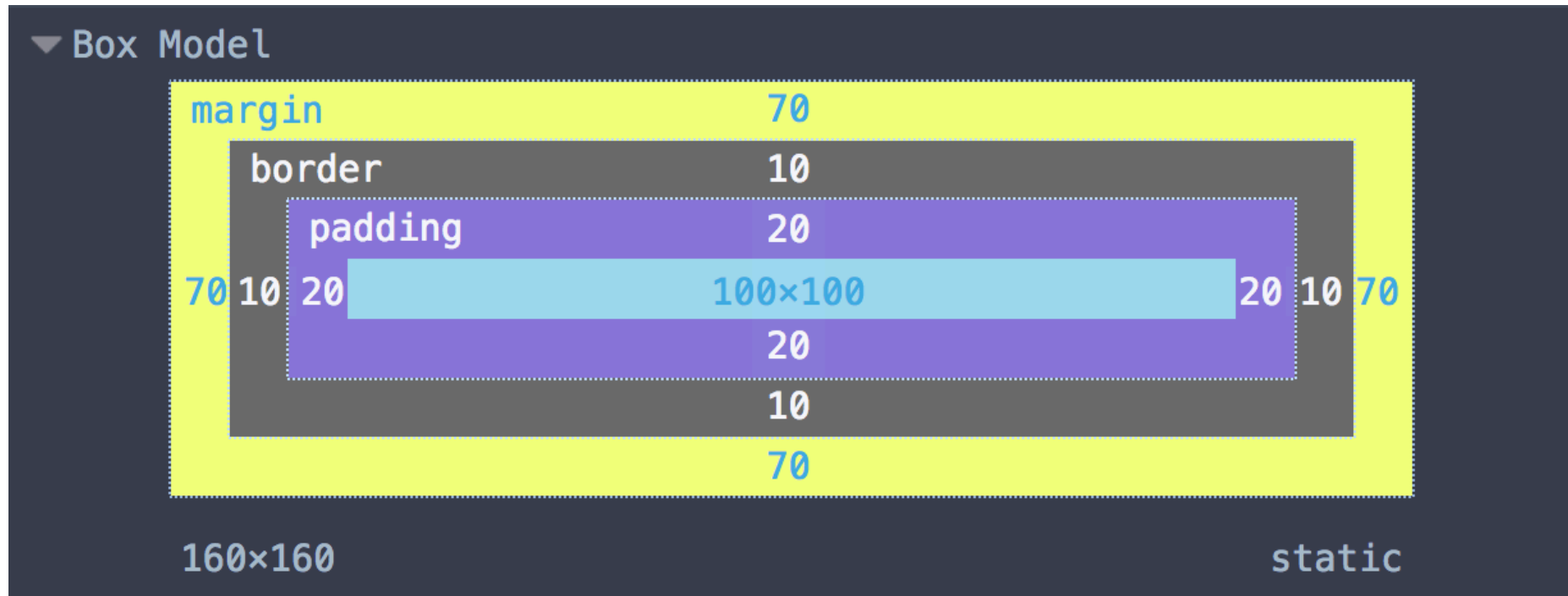
Announcing a change to the data-dump process

Related

26

How does the way that I cut my garlic affect the taste of my food?

Voir les modèles de boîte sur les pages Web





V. Positionnement

Positionnement

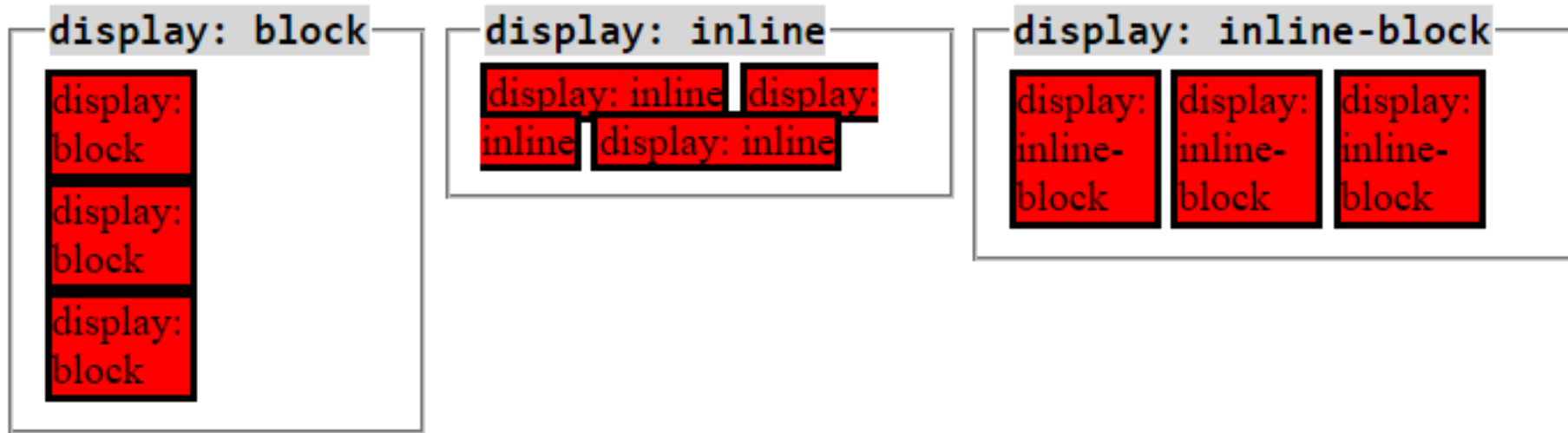
Par défaut, les éléments HTML sont affichés les uns en dessous des autres : ils suivent en fait l'ordre dans lequel ils sont écrits dans le fichier HTML.

Toutefois, on peut modifier cela en utilisant des propriétés CSS.

Les propriétés CSS `display`, `position` et `float` sont les plus couramment utilisées.

Positionnement : display

Nous l'avons déjà vu plus tôt :



Positionnement : display

On peut aussi utiliser `display: none` pour cacher un élément de la page. Les autres éléments dans le conteneur seront repositionnés pour combler l'espace laissé par l'élément caché.

```
1 .hidden {  
2     display: none;  
3 }
```



Positionnement : position

La propriété `position` permet de définir la méthode de positionnement d'un élément. Les valeurs principales sont :

- ♦ `static` (par défaut) ;
- ♦ `relative` ;
- ♦ `absolute` ;
- ♦ `sticky`.

[Voir la documentation MDN](#)

Positionnement : `position`

Les positions autres que `static` permettent de changer la position d'un élément par rapport à son emplacement initial, en utilisant les propriétés `top`, `right`, `bottom` et `left`.

Positionnement : position: relative

Un élément avec position: relative est positionné par rapport à sa position initiale.

```
1 .relative {  
2     position: relative;  
3     top: 10px; /* décalage de 10px vers le bas */  
4     left: 20px; /* décalage de 20px vers la droite  
5 }
```



Positionnement : `position: absolute`

Un élément avec `position: absolute` est positionné par rapport à son premier élément parent. Il n'est plus dans le flux normal du document, et, en conséquence, peut se superposer à d'autres éléments.

Positionnement : `position: sticky`

Un élément avec `position: sticky` est positionné par rapport à la fenêtre du navigateur. Il reste à la même position même lorsqu'on défile la page.

Positionnement : float

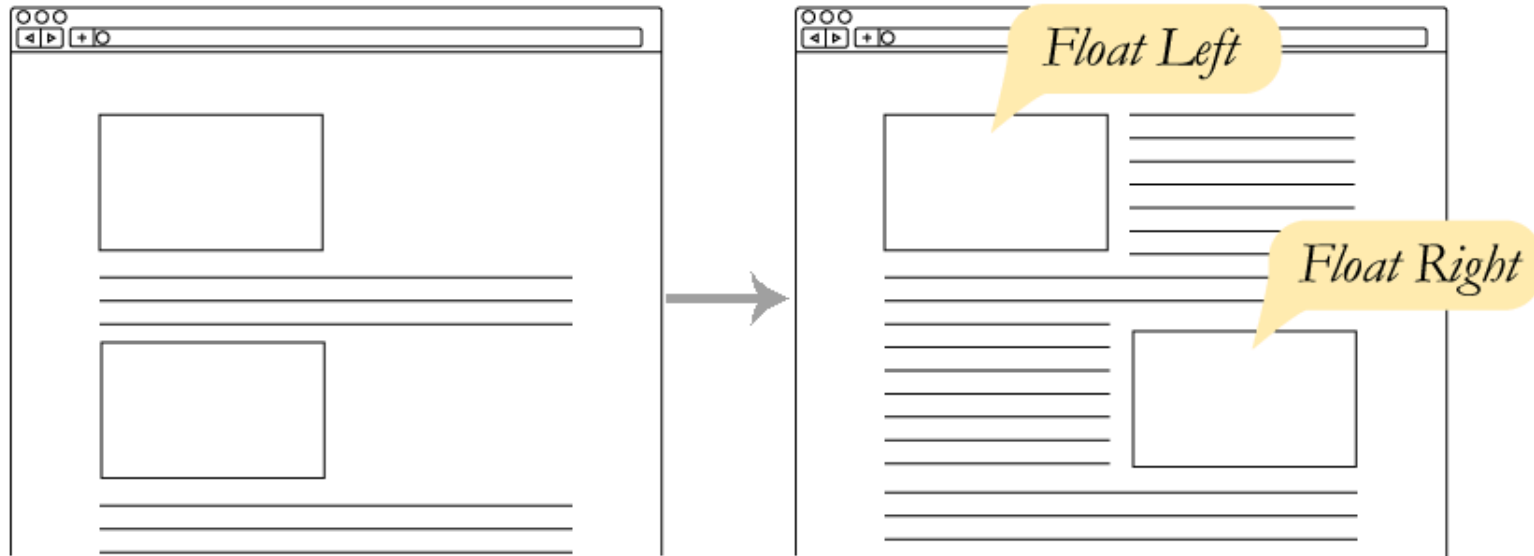
La propriété `float` permet de déplacer un élément sur le côté de son conteneur : à gauche (`float: left`) ou à droite (`float: right`).

```
1 .float {  
2     float: right;  
3 }
```



Positionnement : float

Cette propriété est utile pour agencer images et texte :



Dans ce cas par exemple, la propriété `float` s'applique aux images.

Positionnement : float

```
1 <body>
2   
3   <p>
4     ...
5   </p>
6 </body>
```



```
1 img.image {
2   float: left; /* ou float: right; */
3 }
```

