

CSS3

Synthèse de cours

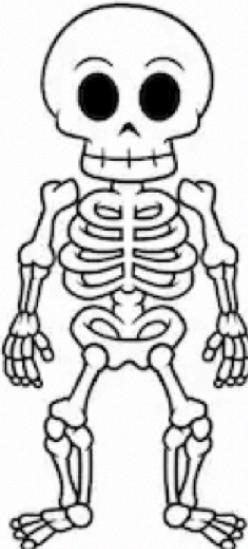
Mots clé : html5, CSS3

- JD.TOULOUSE -

CSS - Synthèse de cours

1. PETITE HISTOIRE DU CSS

CSS est un langage de programmation déclaratif qui permet aux concepteurs Web, aux développeurs, aux blogueurs, etc. de rendre les sites Web uniques et attrayants. CSS nous donne la possibilité de jouer avec une mise en page, d'ajuster les couleurs et les polices, d'ajouter des effets aux images.



HTML



HTML + CSS



HTML+CSS+JavaScript

CSS (Cascading Style Sheets) a été proposé pour la première fois par Håkon Wium Lie le 10 octobre 1994 alors qu'il travaillait avec Tim Berners-Lee au CERN et le reste appartient à l'histoire. CSS n'était pas le seul langage de style en développement à l'époque, mais l'élément même de la séquence en cascade et en développement est ce qui le distingue des autres.

CSS est un langage de programmation déclaratif spécifique à un domaine ; il décrit comment les éléments HTML doivent être affichés à l'écran, sur papier ou sur d'autres supports ; il peut contrôler la mise en page, les couleurs et les polices de plusieurs pages Web à la fois.

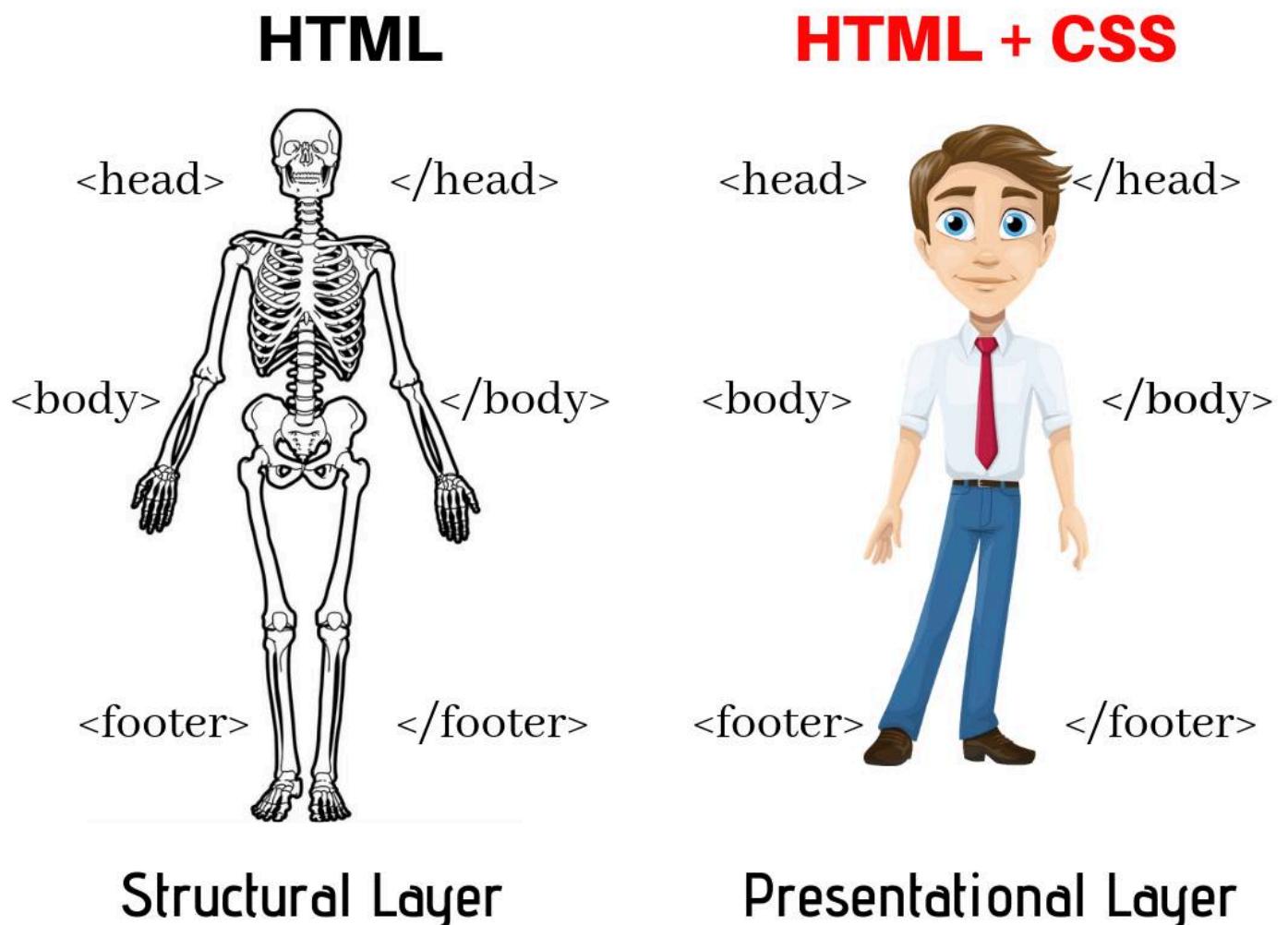
1.1. Avantages du CSS

- Plus facile à entretenir et à mettre à jour
- Une plus grande cohérence dans la conception
- Plus d'options de formatage
- Code léger

- Temps de téléchargement plus rapides
 - Avantages de l'optimisation des moteurs de recherche
 - Facilité de présentation de différents styles à différents spectateurs
 - Une plus grande accessibilité

CSS vous donne la possibilité de créer des sites qui ont un aspect très différent d'une page à l'autre, sans beaucoup de codage approfondi.

HTML Vs CSS

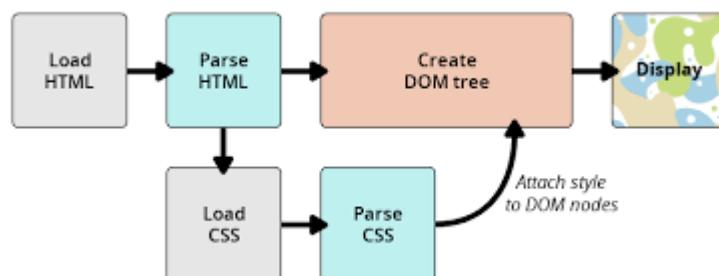


1.2. Les versions du CSS

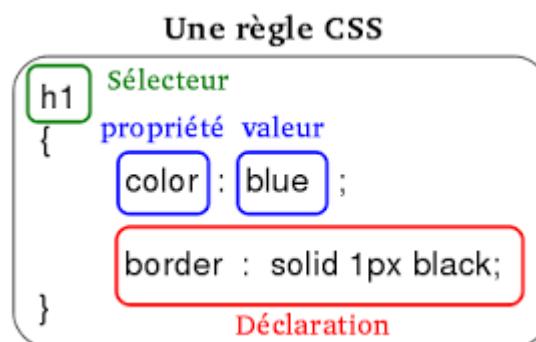
Les fonctionnalités CSS sont contrôlées et développées par le W3C au fil des ans à travers de multiples spécifications.

- CSS 1 est sorti en 1996 et republié avec des corrections en 1999.
- CSS 2 est apparu en 1998 et basé sur CSS 1. CSS 2 ajoute la prise en charge de différents supports de sortie.
- Après CSS 2.1, CSS lui-même a été modularisé, de sorte que chaque module peut être développé indépendamment du reste.
- Le terme CSS3 fait référence à tout ce qui a été publié après CSS 2.1, il n'y a pas de spécification CSS4 intégrée unique.
- Le groupe de travail CSS publie parfois des instantanés, une collection de modules entiers et des parties d'autres brouillons qui sont considérés comme suffisamment stables pour être implémentés par les développeurs de navigateurs.

2. ARBRE DE RENDU



3. DECLARATION CSS





4. COMMENTAIRES

```
/*Un premier commentaire CSS/
body{
    background-color: orange;
}

/*Un deuxième commentaire
 *sur plusieurs
 *lignes*/
p{
    /*color: blue;*/
    font-size: 20px;
}
```

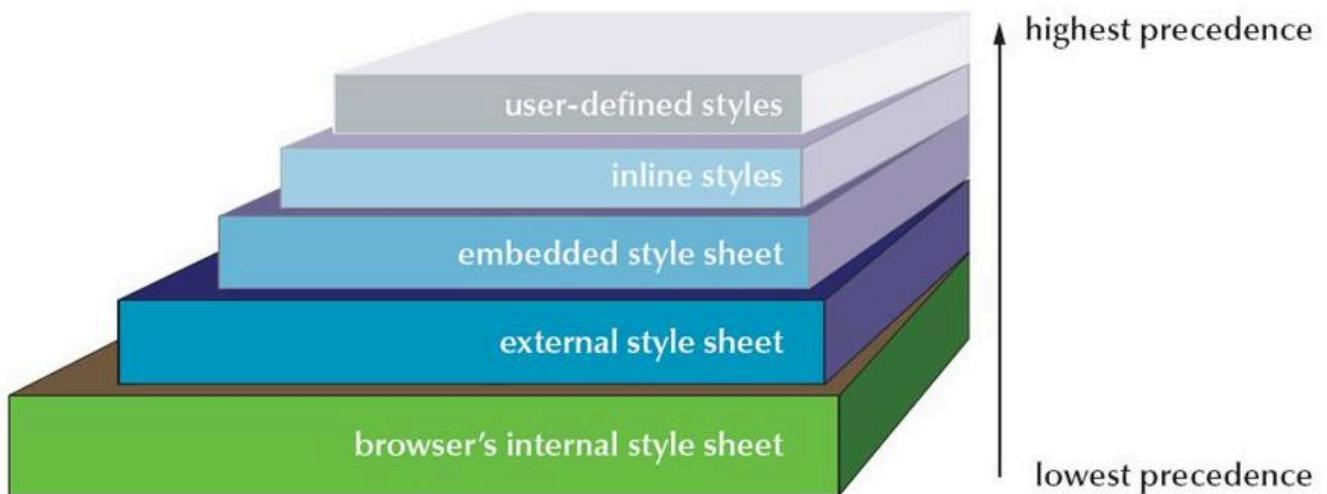
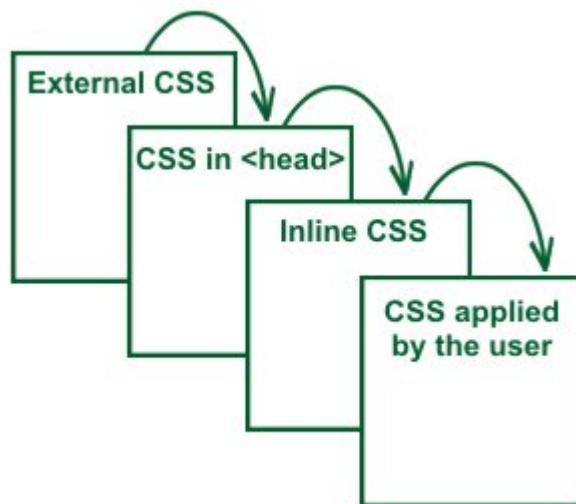
5. LA CASCADE

CSS est l'acronyme de **Cascading Style Sheets**, qu'on peut traduire par *feuilles de style en cascade* et la compréhension de ce premier mot *cascading* est cruciale — la façon dont la cascade se comporte est la clé de la compréhension du CSS.

À un moment donné, vous travaillerez sur un projet et vous constaterez que le CSS que vous pensiez appliquer à un élément ne fonctionne pas. En général, le problème est que vous avez créé deux règles qui pourraient potentiellement s'appliquer au même élément. La **cascade**, et le concept étroitement lié de spécificité, sont des mécanismes qui contrôlent quelle règle s'applique lorsqu'il y a un tel conflit. La règle qui régit votre élément n'est peut-être pas celle à laquelle vous vous attendiez, vous devez donc comprendre comment ces mécanismes fonctionnent.

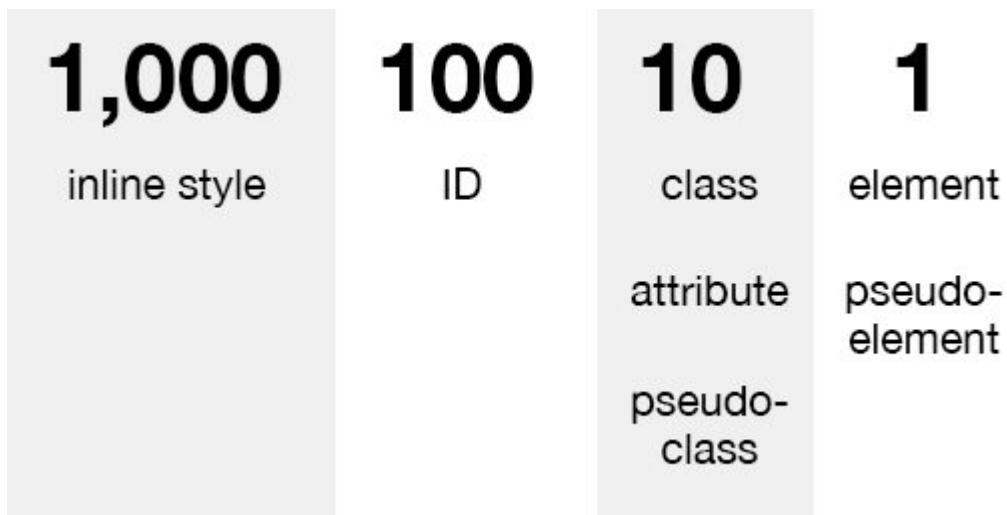
- À un niveau élémentaire, la **cascade** des styles signifie que l'ordre d'apparition des règles dans le CSS a une importance

- Quand deux règles applicables ont la même spécificité, c'est la dernière déclarée qui sera utilisée pour la mise en forme.



5.1. Spécificité

Quand des règles avec des sélecteurs différents s'appliquent sur un même élément, le navigateur choisit la règle qui a la plus grande spécificité.



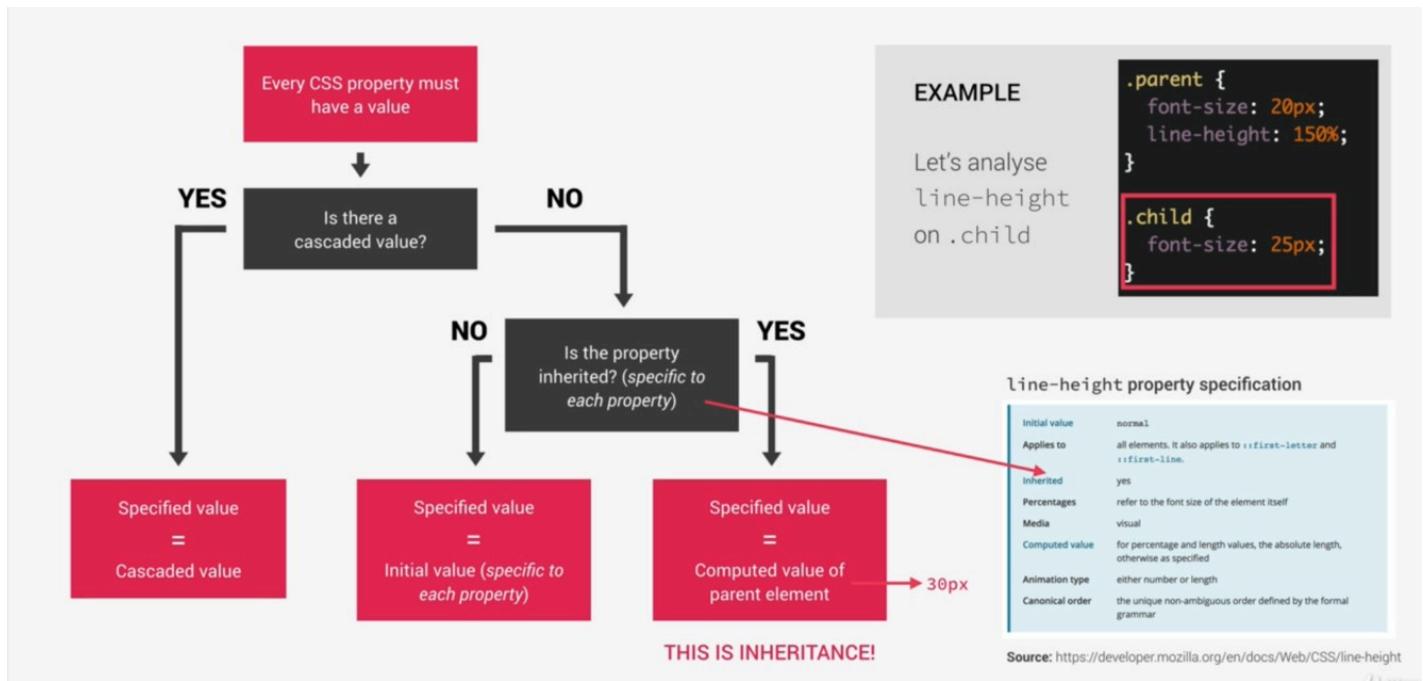
Selectors	Specificity
*{}	0,0,0,0
li{}	0,0,0,1
li:first-line {}	0,0,0,2
ul li{}	0,0,0,2
ul ol+li{}	0,0,0,3
h1 + *[rel=up]{}	0,0,1,1
ul ol li.red {}	0,0,1,3
li.red.level {}	0,0,2,1
#exampleId{}	0,1,0,0
style=""	1,0,0,0

6. L'HERITAGE

Le concept d'**héritage** est aussi à garder en tête dans ces situations : certaines propriétés CSS par défaut héritent de la valeur donnée à l'élément parent de l'élément courant, d'autres non. Cela peut être à l'origine de comportements inattendus.

Pour [chaque propriété CSS](#), la spécification indique si, par défaut, cette propriété est héritée ou non. Cela permet de définir le comportement qu'on observera lorsqu'aucune valeur n'est spécifiée pour une propriété pour un élément donné.

L'héritage provient toujours de l'élément parent par rapport à l'arbre du document, même si cet élément n'est pas le bloc englobant.



6.1. Contrôler l'héritage

CSS propose quatre valeurs spéciales universelles pour contrôler l'héritage. Ces valeurs sont applicables à toute propriété CSS.

- **initial** : applique la valeur initiale d'une propriété à un élément. La valeur initiale est fournie par le navigateur et peut être utilisée pour chaque propriété CSS. Cette propriété prendra alors [la valeur initiale](#) spécifiée pour cette propriété.
- **inherit** : valeur qui peut être utilisée pour qu'une propriété prenne [la valeur calculée](#) de la propriété pour l'élément parent.
- **unset** : combinaison des mots-clés **initial** et **inherit**. Ce mot-clé indique que toutes les propriétés qui s'appliquent à l'élément ou à son parent prendront la valeur déclarée pour le parent si elles peuvent être héritées ou la valeur initiale sinon.
- **revert** : permet de remonter la cascade afin que la propriété puisse prendre la valeur qui aurait été utilisée sans mise en forme particulière (la valeur qu'elle aurait eu sans mise en forme appliquée par la feuille de style de l'auteur, de l'utilisateur ou de l'agent utilisateur). Ainsi, si la propriété hérite de son parent, elle prendra la valeur héritée et sinon la valeur par défaut de l'agent utilisateur (ou de la feuille de style utilisateur).
- **all** : permet de réinitialiser toutes les propriétés, à l'exception de unicode-bidi et direction, avec leurs valeurs initiales, héritées ou qui proviennent d'une autre feuille de style.

6.2. Propriété héritée

Le mot-clé `inherit` est une valeur qui peut être utilisée pour qu'une propriété prenne [la valeur calculée](#) de la propriété pour l'élément parent.

6.3. Valeur initiale

La **valeur initiale** d'une [propriété CSS](#) est définie par la spécification et varie selon [qu'une propriété est héritée ou non](#).

6.4. Valeur définie

La **valeur définie** d'une propriété CSS est celle explicitement définie dans la feuille de style ou grâce au style de son élément parent.

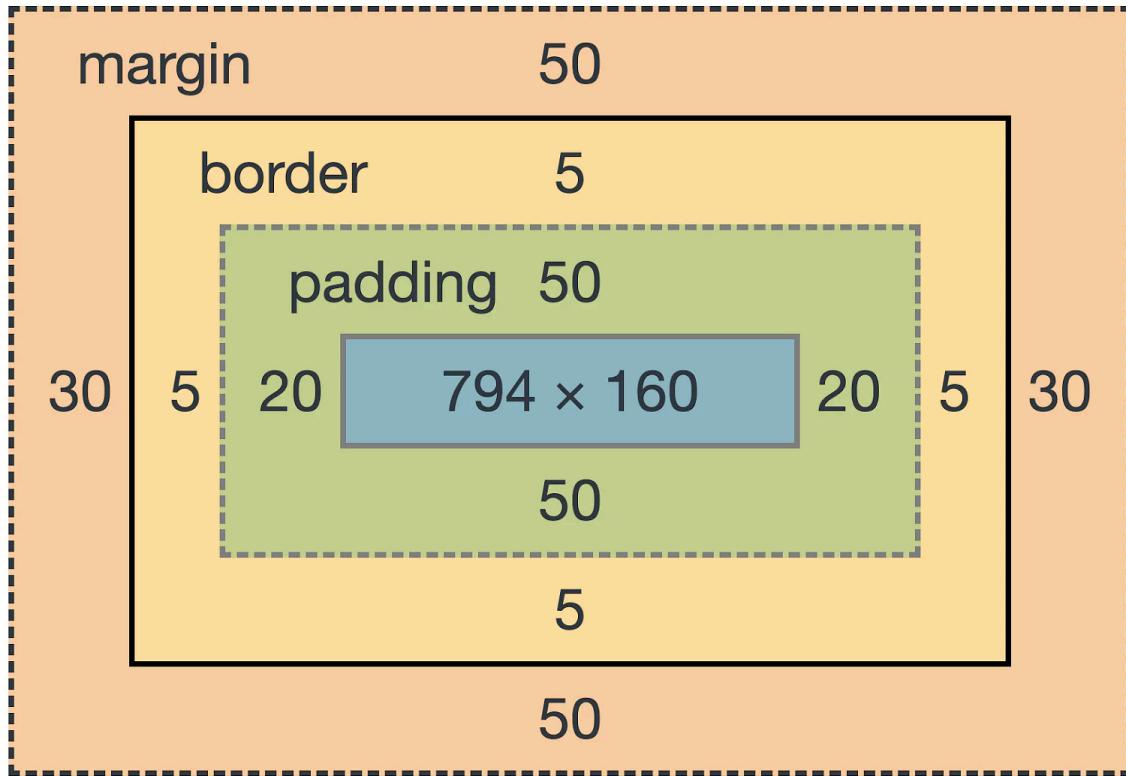
6.5. Valeur calculée

La **valeur calculée** d'une propriété CSS est calculée à partir de [la valeur définie](#) :

1. En gérant les valeurs spéciales `inherit`, `initial`, `unset` et `revert`.
2. En effectuant les calculs décrits dans la section « Valeur calculée » de chaque résumé de propriété.

7. LE MODELE DE BOITE

Le modèle de boîte définit comment chaque paramètre de la boîte fonctionne pour former l'aspect final affiché à l'écran. Le modèle de boîte CSS (Basic Box Model en anglais) est un module CSS qui définit les boîtes rectangulaires (y compris leurs zones de remplissage (padding) et de marges) qui sont générées pour disposer les éléments selon leur modèle de mise en forme visuelle.



7.1. La propriété box-sizing

<p>CSS Box Sizing</p> <p>content-box border-box</p> <pre> box-sizing: content-box; width: 100px; height: 100px; padding: 20px; box-sizing: border-box; width: 100px; height: 100px; padding: 20px; </pre> <p> ● Padding ● Content area ○ Element border </p>	<p>CSS3 Box Model</p> <p>box-sizing : content-box; Content-box model exclude the border, padding and margin</p> <p>box-sizing : border-box; In border-box model all included inside with (margin, padding, border)</p>
---	---

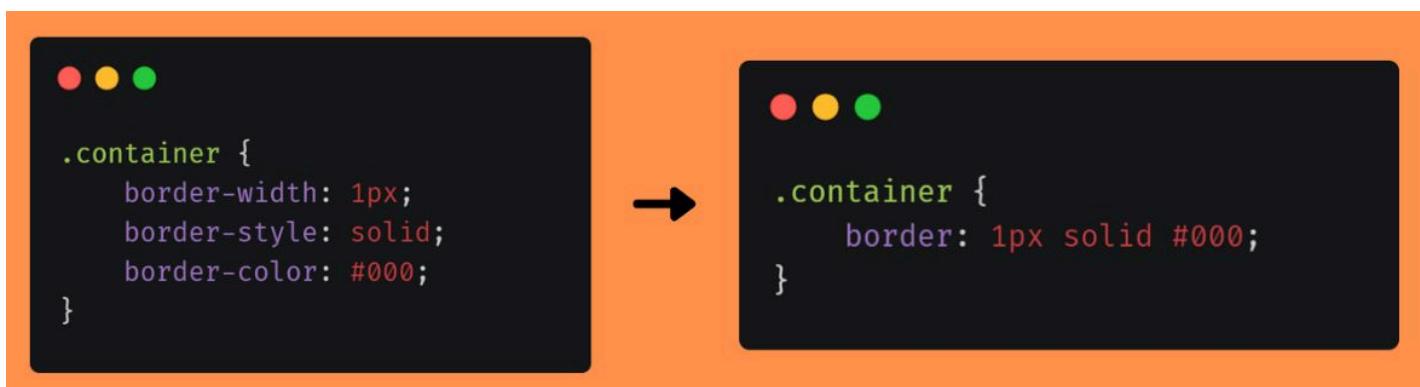
7.1.1. Les propriétés qui définissent le flux du contenu dans une boîte

- [overflow](#)
- [overflow-x](#)
- [overflow-y](#)

7.1.2. Les propriétés qui définissent la taille d'une boîte

- `height`
- `width`
- `max-height`
- `max-width`
- `min-height`
- `min-width`

7.2. Les propriétés qui définissent les bordures d'une boîte



Les différents styles possibles sont :

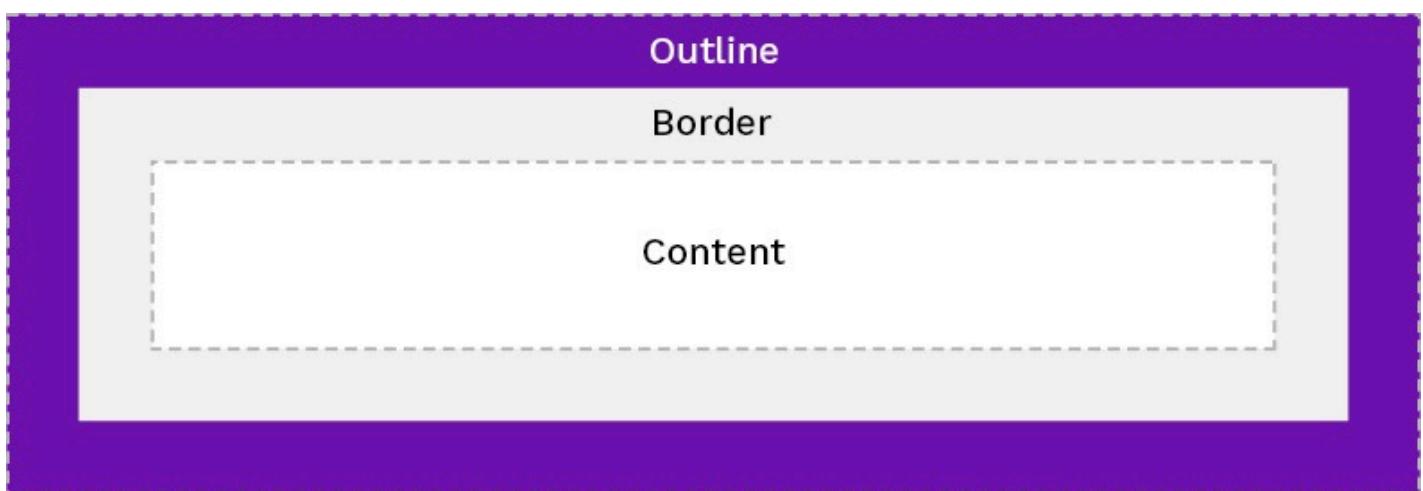
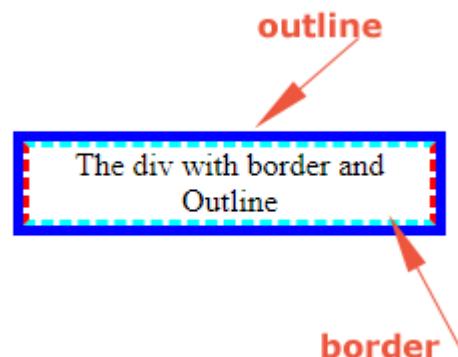
- none
- hidden
- dotted
- dashed
- solid
- double
- groove
- ridge
- inset
- outset

7.3. Les propriétés qui définissent les extérieurs des bordures (outline)

La propriété `outline` :

- est une ligne tracée autour un élément, à l'extérieur le Bordure CSS
- n'appartiennent pas au Modèle de boîte CSS

- ne répond pas au border-radius



This paragraph has a 6px dotted outline 0.5 cm away from the border edge.

```
p {
    border: 2px solid gray;
    outline: 6px dotted blue;
}
```

7.4. Les propriétés qui définissent les marges d'une boîte



7.5. Les propriétés qui définissent le remplissage (*padding*) d'une boîte



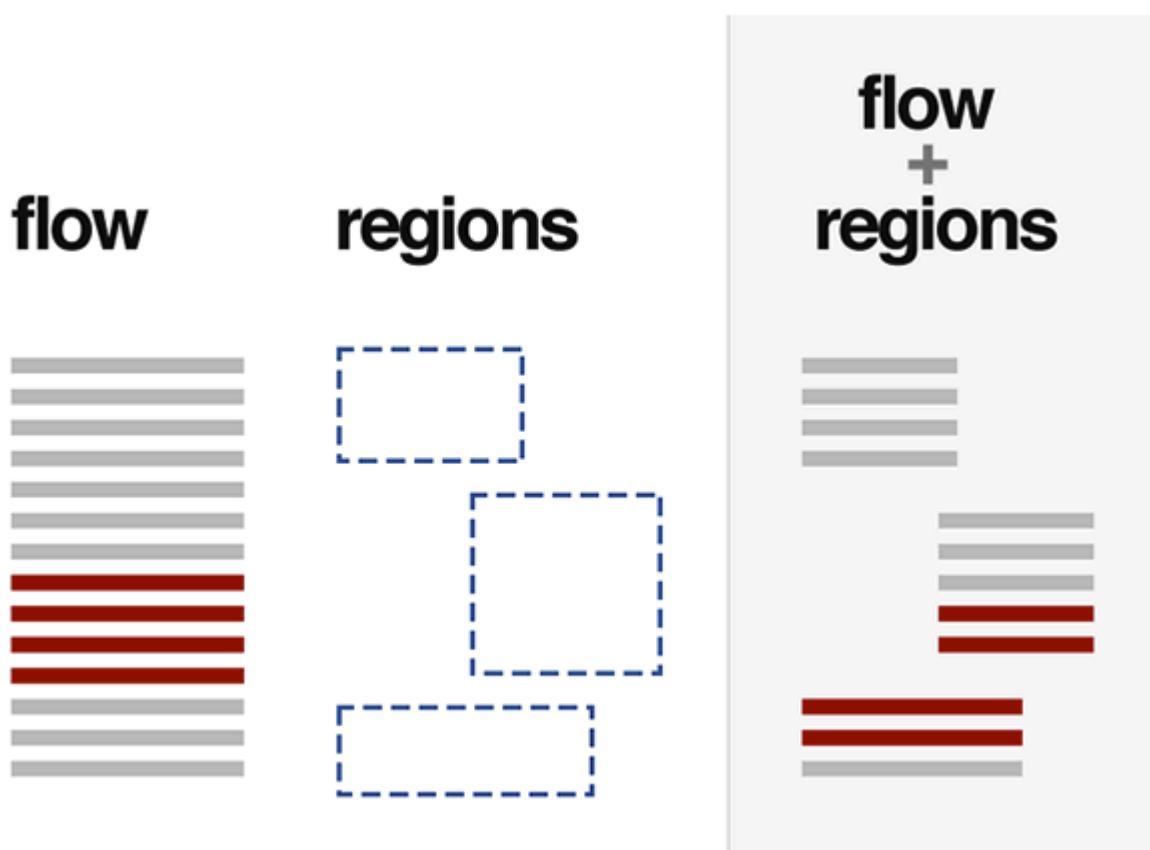
7.5.1. Les autres propriétés

- [box-shadow](#)
- [visibility](#)

8. LE FLUX

8.1. Définition

Le *flux* correspond à l'écoulement des informations (ou données) dans le processus d'interprétation des navigateurs.



8.2. Propriété display

display: block



display: inline



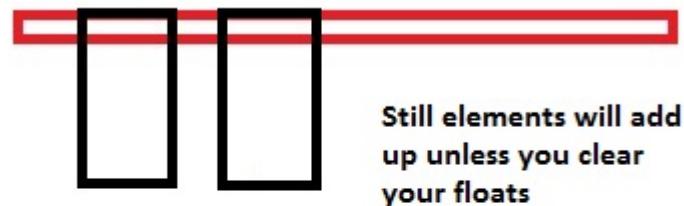
display: inline-block



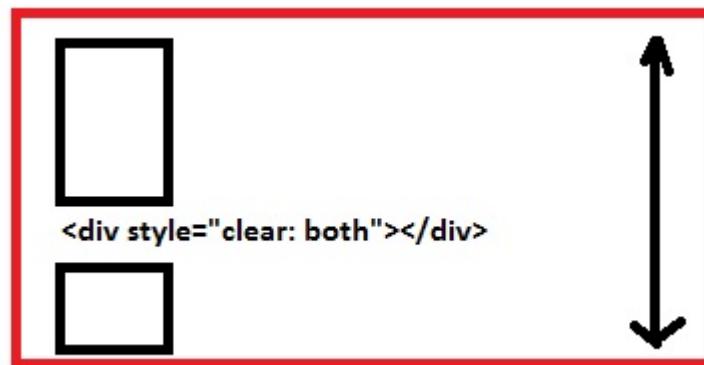
8.3. La propriété float et les éléments flottants

La propriété `float` indique qu'un élément doit être retiré du flux normal et doit être placé sur le côté droit ou sur le côté gauche de son conteneur. Le texte et les autres éléments en ligne (inline) entoureront alors l'élément flottant. L'élément est **retiré du flux normal** de la page mais s'inscrit toujours dans le flux (contrairement au positionnement absolu).

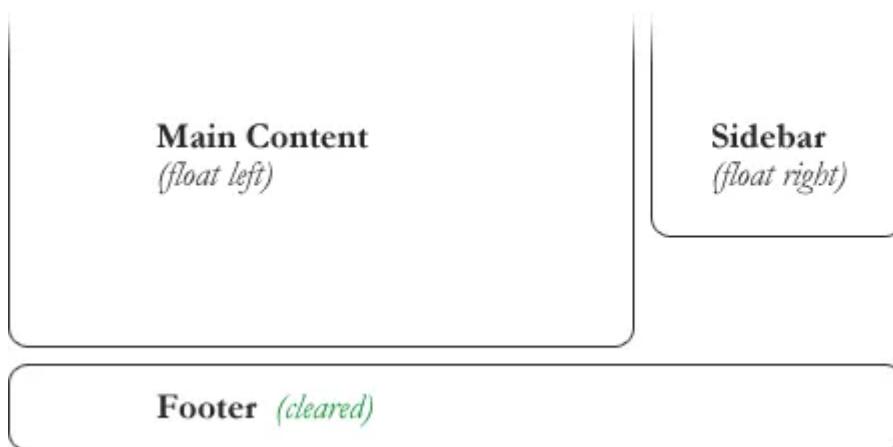




Add any other element and it will shift to the right of the floated div if any space is left

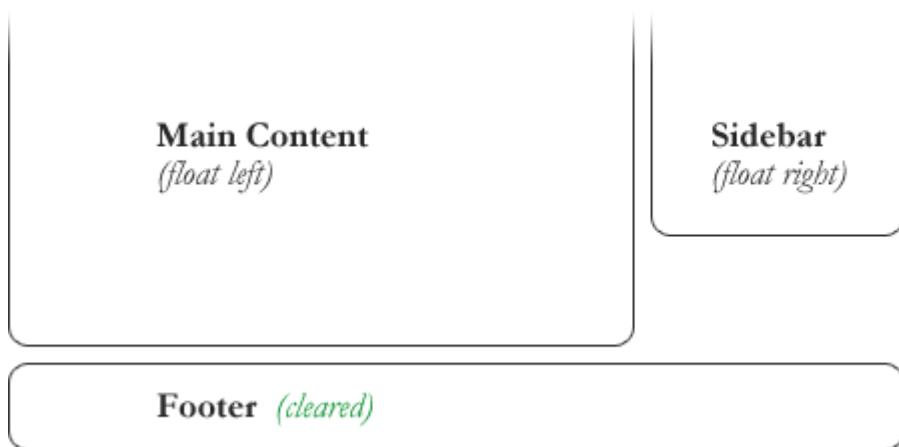


Voici un exemple simple de mise en page construite avec des flottants, qui pourrait être problématique pour le pied de page :



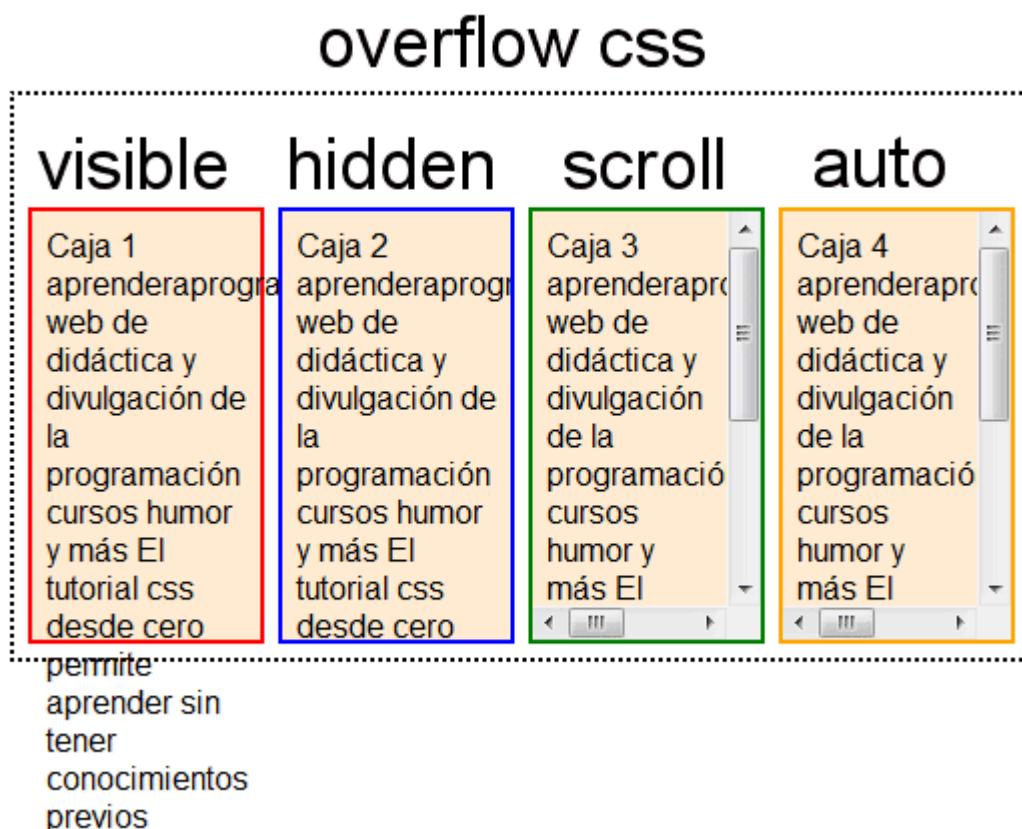
Mais en effaçant l'élément de pied de page, la mise en page s'enclenche :

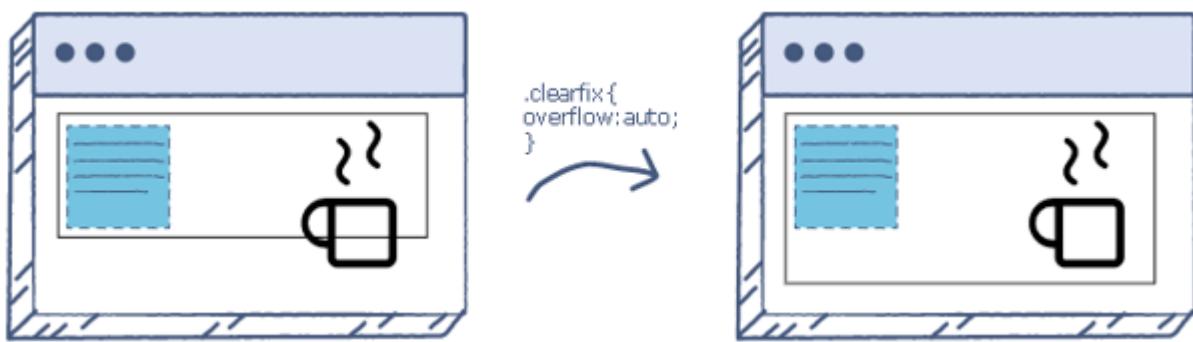
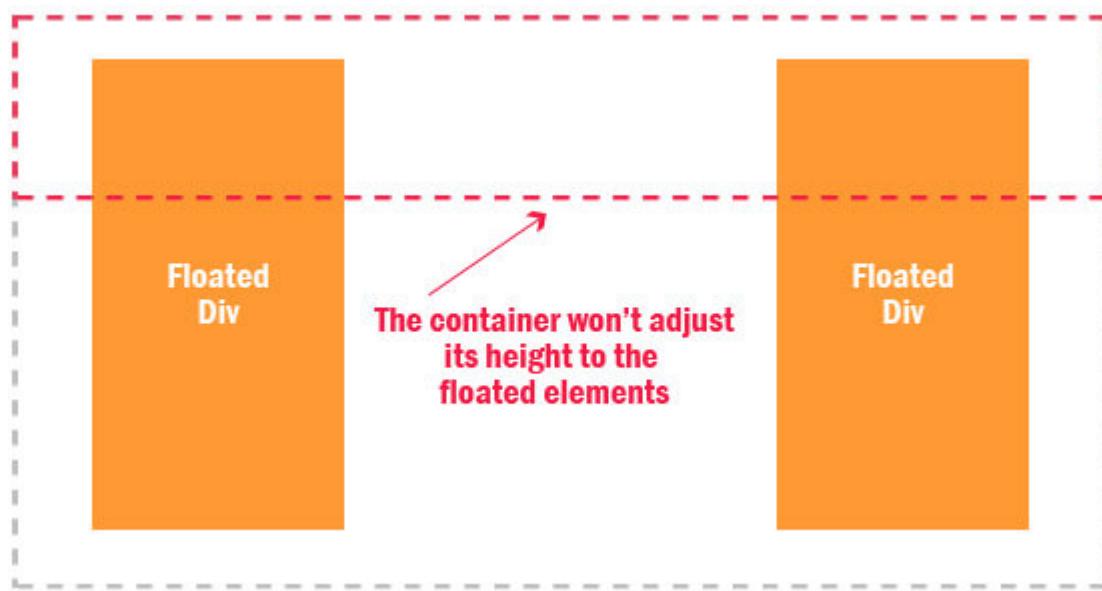
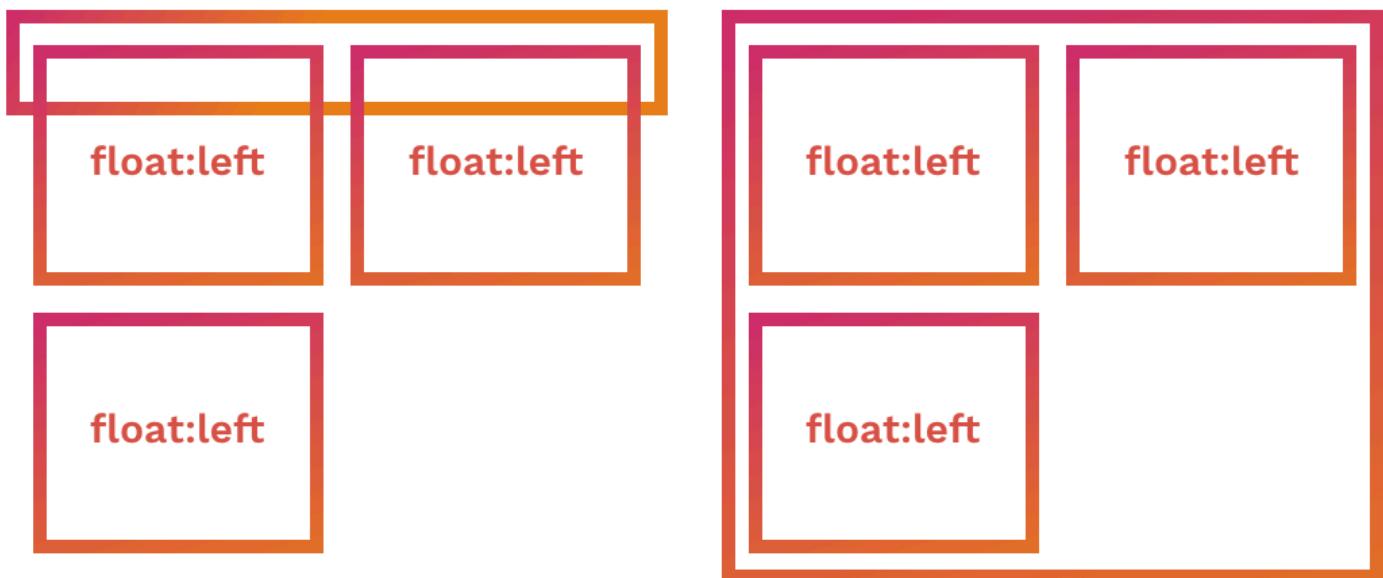
```
#footer {
  clear: both;
}
```



8.4. La propriété overflow

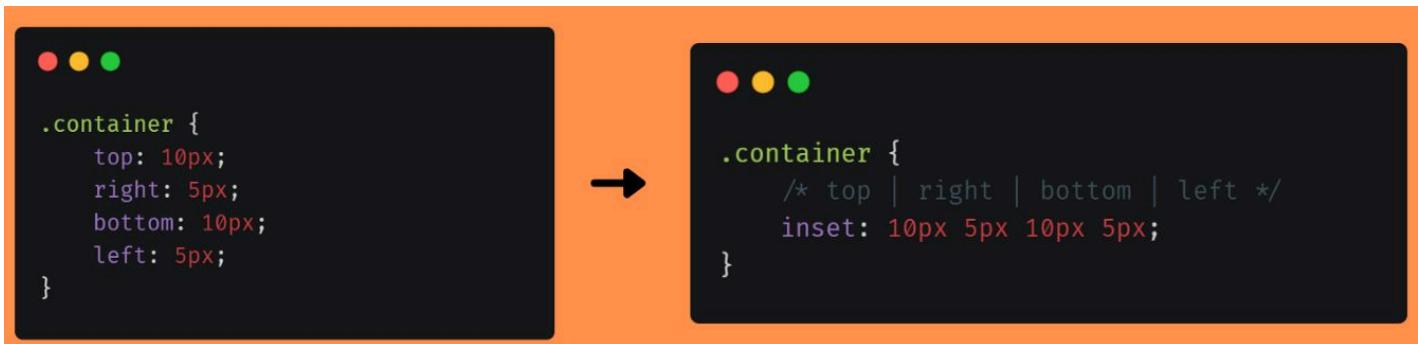
La propriété CSS `overflow` est une [propriété raccourcie](#) qui définit comment gérer le dépassement du contenu d'un élément dans son bloc. Elle définit les valeurs des propriétés [`overflow-x`](#) et [`overflow-y`](#).





9. LE POSITIONNEMENT EN CSS

- La propriété CSS position définit la façon dont un élément est positionné dans un document.
- Les propriétés top, right, bottom et left déterminent l'emplacement final de l'élément positionné.



```
<h1>Positionnement</h1>
<p>Je suis un élément de niveau bloc de base.</p>
<p class="positioned">Je suis un élément de niveau bloc de base.</p>
<p>Je suis un élément de niveau bloc de base.</p>
```

9.1. Positionnement statique

Valeur par défaut reçue par chaque élément — il signifie simplement « placer l'élément à sa position normale suivant le cours normal de mise en page du document — rien de spécial à constater ici ».

Je suis un élément de niveau bloc de base.

Je suis un élément de niveau bloc de base.

Je suis un élément de niveau bloc de base.

```
body {
    width: 500px;
    margin: 0 auto;
}

p {
    background-color: rgb(207, 232, 220);
    border: 2px solid rgb(79, 185, 227);
    padding: 10px;
    margin: 10px;
    border-radius: 5px;
}
```

9.2. Positionnement relatif

Modification de la position d'un élément dans la page — il est déplacé par rapport à sa position dans le cours normal — y compris la possibilité de chevaucher d'autres éléments de la page.

Je suis un élément de niveau bloc de base.

Voici un élément avec un positionnement relatif.

Je suis un élément de niveau bloc de base.

```
.positioned {
    position: relative;
    background: rgba(255, 84, 104, .3);
    border: 2px solid rgb(255, 84, 104);
    top: 30px;
    left: 30px;
}
```

9.3. Positionnement absolu

Déplacement d'un élément indépendamment du cours normal de positionnement, comme s'il était placé sur un calque séparé. À partir de là, vous pouvez le fixer à une position relative au bord de la page de l'élément `<html>` (ou de son plus proche élément ancêtre positionné). Ce positionnement est utile pour créer des effets de mise en page complexes tels que des boîtes à onglets où différents panneaux de contenu sont superposés, affichés ou cachés comme vous le souhaitez, ou des panneaux d'information hors de l'écran par défaut, mais qui peuvent glisser à l'écran à l'aide d'un bouton de contrôle.

Positionnement absolu

Voici un élément avec un positionnement absolu.

Je suis un élément de niveau bloc de base.

Je suis un élément de niveau bloc de base.

```
.positioned {
    position: absolute;
    background: rgba(255,84,104,.3);
    border: 2px solid rgb(255,84,104);
    top: 30px;
    left: 30px;
}
```

9.4. Positionnement fixed

Tout à fait semblable au précédent, à l'exception que l'élément est fixé par rapport à la vue du navigateur et non d'un autre élément. C'est très pratique pour créer des effets tels qu'un menu de navigation persistant, toujours au même endroit sur l'écran, quand l'utilisateur fait défiler le reste de la page.

```
.positioned {
    position: fixed;
    top: 30px;
    left: 30px;
}
```

```
<h1>Positionnement fixé</h1>
<div class="positioned">Fixé</div>
<p>Paragraphe 1.</p>
<p>Paragraphe 2.</p>
<p>Paragraphe 3.</p>
```

Positionnement fixé

Fixé

Paragraphe 1.

Paragraphe 2.

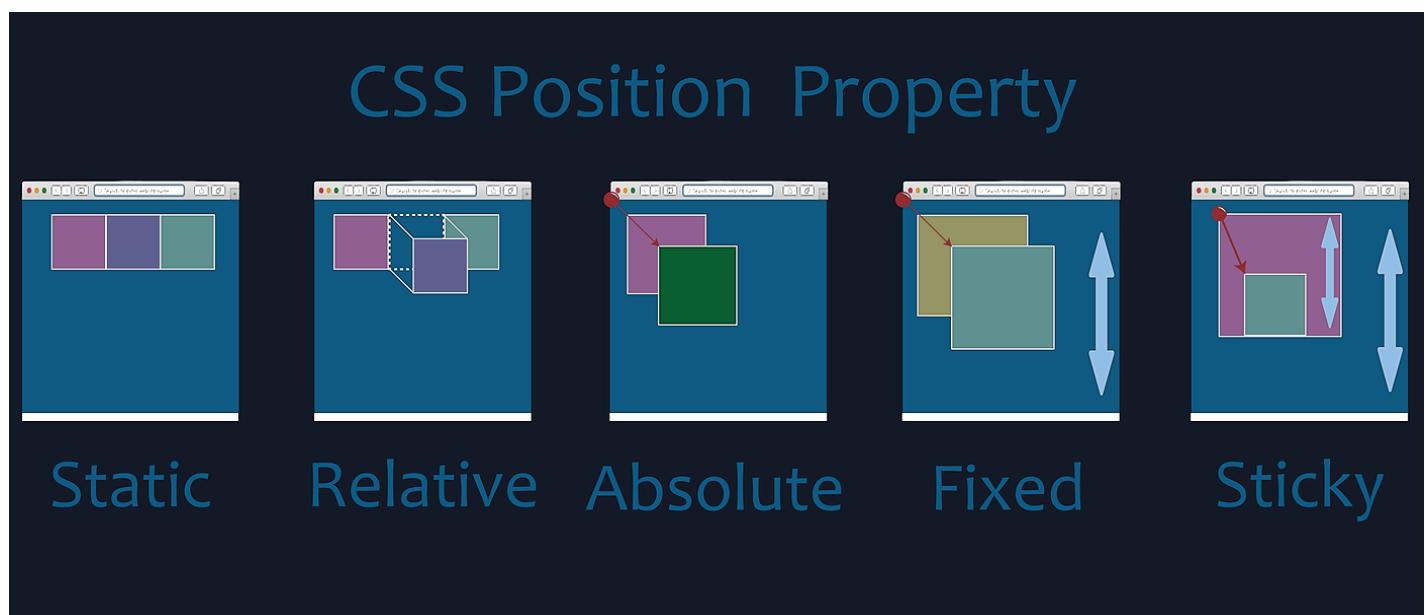
Paragraphe 3.

9.5. Positionnement sticky

nouvelle méthode de positionnement qui fait qu'un élément se comporte comme `position: static` jusqu'à un certain décalage de la vue au delà duquel il se comporte comme si sa propriété était `position: fixed`.

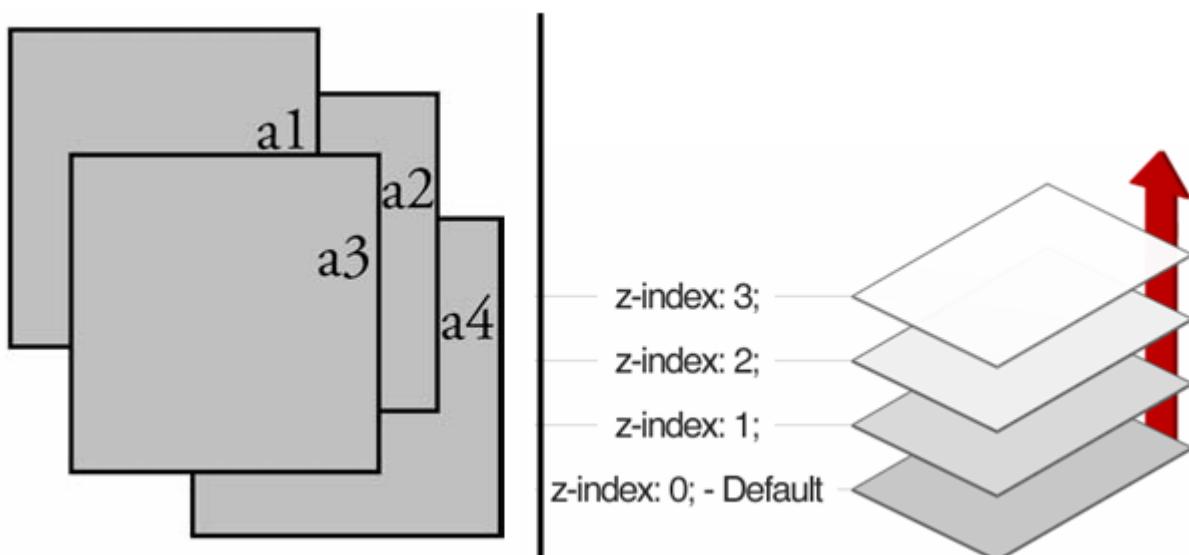
```
.positioned {
    position: sticky;
    top: 30px;
    left: 30px;
}
```

En résumé :



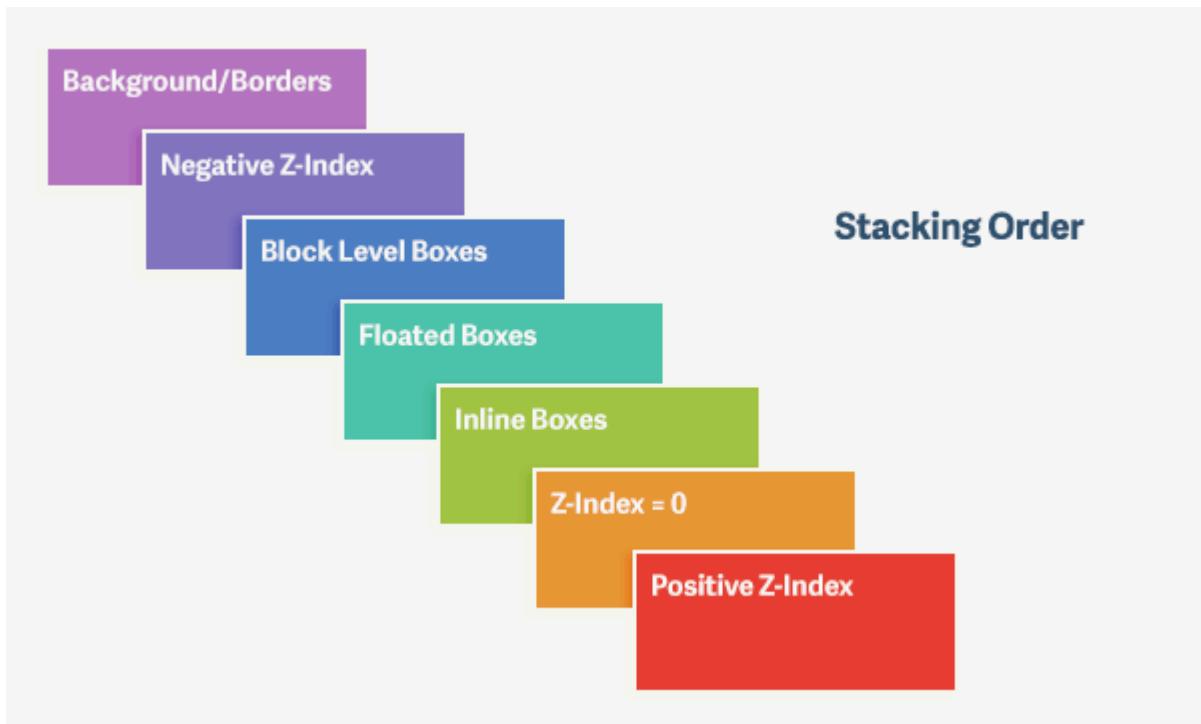
10. EMPILEMENT : L'ART DES CALQUES

La propriété **z-index** définit le « *z-order* » (NdT : « ordre z » n'est pas usité) d'un élément positionné et de ses éléments enfants ou de ses éléments flexibles. Lorsque des éléments se chevauchent, le *z-order* détermine l'ordre des différentes couches que formeront les éléments. Généralement, un élément couvrira un autre élément si sa valeur de **z-index** est supérieure à celle du deuxième élément.



Alors que l'exemple précédent ne montre que deux niveaux hiérarchiques, dans chaque contexte hiérarchique, il existe jusqu'à **sept niveaux possibles**, répertoriés ci-dessous.

1. **Arrière-plan et bordures** - de l'élément qui constitue le contexte hiérarchique. Le niveau le plus bas de la hiérarchie.
2. **z-index négatif** - les contextes hiérarchiques des éléments descendants avec une valeur z-index négative.
3. **Bloquer les conteneurs** - flux entrant, non en ligne et descendants non placés.
4. **Conteneurs flottants** - éléments flottants non positionnés.
5. **Conteneurs en ligne** — dans le flux, en ligne et descendants non placés.
6. **z-index : 0** - éléments positionnés. Ceux-ci génèrent un nouveau contexte hiérarchique.
7. **indice z positif** - éléments positionnés. Le plus haut niveau de la hiérarchie.



11. SELECTEURS CSS

Les sélecteurs sont utilisés pour sélectionner des éléments dans un document HTML, afin de leur attacher des propriétés (de style).

Les sélecteurs CSS les plus utilisés sont by **class** et by **id**, néanmoins, il existe de nombreux sélecteurs que vous pouvez utiliser facilement et équitablement pour ajouter des styles dans un ensemble d'éléments.

11.1. I - Les selecteurs de base

11.1.1. Sélecteur de type

```
h1 {
    font-size: 26px;
}
```

11.1.2. Sélecteur de classe

Les sélecteurs de classe sont désignés par `a.` (période) avant le nom de la classe lorsqu'il est utilisé dans un sélecteur. Supposons que nous ayons un document HTML qui contient l'extrait de code suivant:

```
<p class ="citation">La seule chose que nous devons craindre est la peur
elle-même.</p>
<span class ="auteur">Franklin D. Roosevelt</span>
```

11.1.3. Sélecteur d'ID

Contrairement aux sélecteurs de classe, un sélecteur d'ID utilise un hachage (#) lorsqu'il est utilisé dans un sélecteur.

```
<h1 id ="titre1">Feuilles de style en cascade</h1>
```

11.1.4. Sélecteur universel

Les sélecteurs universels sélectionneront tous les éléments d'une balise. L'exemple suivant sélectionnera chaque élément qui se trouve dans une div.

```
div * {
    color: gray;
    font-size: 14px;
}
```

11.2. Les combinateurs

11.2.1. Sélecteur descendant

```
div p {  
    background-color: #fdc7d5;  
}
```

11.2.2. Sélecteur universel *

```
div * {  
    background-color: #fdc7d5;  
}
```

11.2.3. Sélecteur de frères et sœurs adjacents +

Sélectionne un élément qui suit directement un autre élément.

```
div + p {  
    background-color: #fdc7d5;  
}
```

11.2.4. Sélecteur général de frères et sœur ~

Sélectionne **tous les frères et sœurs** d'un élément qui le suit.

```
div ~ p {  
    background-color: #fdc7d5;  
}
```

11.2.5. Sélecteur enfant

Sélectionne les enfants directs d'un élément. Vous pouvez sélectionner des éléments qui sont des enfants directs d'autres éléments.

```
div > p {  
    background-color: #fdc7d5;  
}
```

11.3. Les sélecteurs d'attributs1. [attr=valeur]

11.3.1. [attr=valeur]

Permet de cibler un élément qui possède un attribut **attr** dont la valeur **est exactement valeur**.

Exemple :

Sélectionner tous les éléments `input` de type case à cocher.

```
input[type="checkbox"] {
    font-size: 18px;
    margin-top: 3rem;
}
```

11.3.2. [attr^=valeur]

Permet de cibler un élément qui possède un attribut **attr** dont la valeur **commence par valeur**.

Exemple :

Sélectionne les éléments avec `class="bonjour"`, `class="bonsoir"`, `class="bonus"` ...

```
[class^="bon"] {
    background: yellow;
}
```

11.3.3. [attr\$=valeur]

Permet de cibler un élément qui possède un attribut **attr** dont la valeur **se termine par valeur**.

Exemple :

Sélectionne tous les paragraphes qui se terminent par `ion`.

```
[class$="ion"] {
    background: skyblue;
}
```

11.3.4. [attr~="valeur"]

Permet de cibler un élément qui possède un attribut **attr** dont la valeur est **valeur**. Cette forme permet de fournir une liste de valeurs, séparées par des blancs, à tester. Si au moins une de ces valeurs est égale à celle de l'attribut, l'élément sera ciblé.

Exemple :

Tous les divs en espagnol d'Espagne seront bleus.

```
div[lang~="es-ES"] {
    color: blue;
}
```

```
<div lang="en-us en-gb en-au en-nz">Hello world!</div>
<div lang="pt">Olá Mundo!</div>
<div lang="es-ES">Hola Mundo ! </div>
<div lang="zh-TW">世界您好! </div>
<div data-lang="zh-TW">世界您好! </div>
```

11.3.5. [attr|=valeur]

Permet de cibler un élément qui possède un attribut **attr** dont la valeur **est exactement valeur** ou dont la valeur **commence par valeur** suivi immédiatement d'un tiret (U+002D). Cela peut notamment être utilisé pour effectuer des correspondances avec des codes de langues.

Exemple :

Tous les classes commençant par "top-" auront une couleur de fond jaune.

```
[class|=top] {
    background: yellow;
}
```

```
<h1 class="top-header">Welcome</h1> /* fond jaune */
<p class="top-text">Hello world!</p> /* fond jaune */
<p class="topcontent">Are you learning CSS?</p>
```

11.3.6. [attr*=valeur]

Permet de cibler un élément qui possède un attribut **attr** et dont la valeur contient au moins une occurrence de **valeur** dans la chaîne de caractères.

Exemple :

Liens avec "example" n'importe où dans l'URL.

```
a[href*="example"] {
    color: red;
}
```

```
<ul>
    <li><a href="#internal">Lien interne</a></li>
    <li><a href="http://example.com">Lien d'exemple</a></li> /* red */
    <li><a href="#Insensitive">Lien interne insensible à la casse</a></li>
    <li><a href="http://example.org">Lien vers example.org</a></li>
</ul>
```

11.3.7. [attr opérateur valeur i]

On peut ajouter un **i** (ou **I**) avant le crochet de fin. Dans ce cas, la casse ne sera pas prise en compte (pour les caractères contenus sur l'intervalle ASCII).

Exemple :

Liens avec "insensitive" n'importe où dans l'URL, quelle que soit la casse.

```
a[href*="insensitive" i] {
    color: cyan;
}
```

11.3.8. [attr opérateur valeur s]

Ajouter un **s** (ou **S**) avant le crochet fermant permettra d'effectuer une comparaison de valeur sensible à la casse (pour les caractères ASCII).

Exemple :

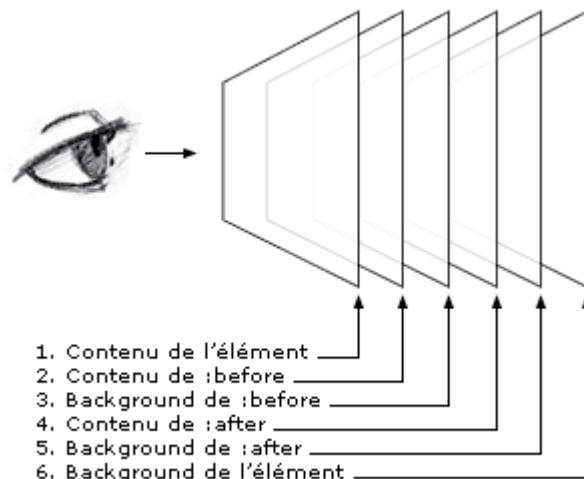
Liens avec "SenSitiVe" n'importe où dans l'URL et avec cette casse donnée.

```
a[href*="SenSitive" s] {
    color: red;
}
```

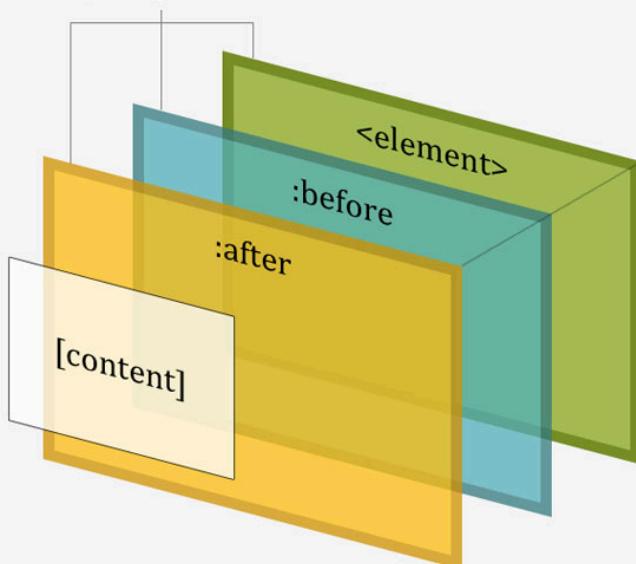
```
<ul>
    <li><a href="#Sensitive">Lien interne insensible à la casse</a></li>
    <li><a href="#Sensitive">Lien interne sensible à la casse</a></li>
</ul>
```

11.4. Les pseudo-éléments

Un **pseudo-élément** est un mot-clé ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle. Ainsi, le pseudo-élément `::first-line` permettra de ne cibler que la première ligne d'un élément visé par le sélecteur.



Multiple boxes from one HTML element allow
for multiple backgrounds and borders



Exemple 1:

```
div::before {  
    content: "before";  
}  
div::after {  
    content: "after";  
}
```

```
<div>  
    before  
  
    after  
</div>
```

CSS Box Model Properties

::after
element
::before

Exemple 2 :

```
p::first-letter {  
    color: blue;  
}
```

```
<p>Bonjour !</p>
```

:before Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec volutpat viverra cursus. Integer porta porttitor eros, non fringilla risus eleifend at. Duis laoreet tincidunt sem sed laoreet. Phasellus ante orci, suscipit non ornare eu, luctus in enim. Donec facilisis ultricies neque, sed iaculis elit aliquet a. Vivamus iaculis luctus mauris a sodales. Maecenas tincidunt arcu vitae nisi tempor laoreet pharetra nunc gravida. Donec tincidunt semper lobortis. Sed eu magna nec nisi viverra vol :after

:before
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec volutpat viverra cursus. Integer porta porttitor eros, non fringilla risus eleifend at. Duis laoreet tincidunt sem sed laoreet. Phasellus ante orci, suscipit non ornare eu, luctus in enim. Donec facilisis ultricies neque, sed iaculis elit aliquet a. Vivamus

:after

11.5. Les pseudo-classes

Une **pseudo-classe** est un mot-clé qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la déclaration. La pseudo-classe `:hover`, par exemple, permettra d'appliquer une mise en forme spécifique lorsque l'utilisateur survole l'élément ciblé par le sélecteur (changer la couleur d'un bouton par exemple).

Exemple :

```
.survol:hover {
    background-color: palegreen;
}
```

```
div>
<p class="survol">
    Bonjour !
</p>
</div>
```

12. LA COULEUR

12.1. Notation hexadécimale #rrggbb

Les valeurs RGB, ou **hex**, commencent par un signe numérique suivi de six caractères, en utilisant une combinaison de nombres de 0 à 9 et les lettres A-F .

Cette forme indique une couleur opaque dont les deux premiers chiffres hexadécimaux indiquent la composante rouge (`rr`---), les deux chiffres suivants indiquent la composante verte (`--gg`--) et les deux derniers chiffres indiquent la composante bleue (`----bb`).

Exemple :

On pourra représenter un bleu pur et opaque avec les chaînes de caractères `"#0000ff"` ou `"#00f"`.

```
h1 {
    color: #0000ff;
}
```

12.2. Notation héxadécimale #rrggbbaa

Exemple :

Pour représenter un bleu pur opaque à 25%, on utilisera "#0000ff44" ou "#00f4".

```
h1 {
    color: #00f4;
}
```

12.3. La notation fonctionnelle rgb()

Exemple :

On représentera un rouge pur à moitié opaque grâce à `rgb(255, 0, 0, 0.5)` ou `rgb(100%, 0, 0, 50%)`.

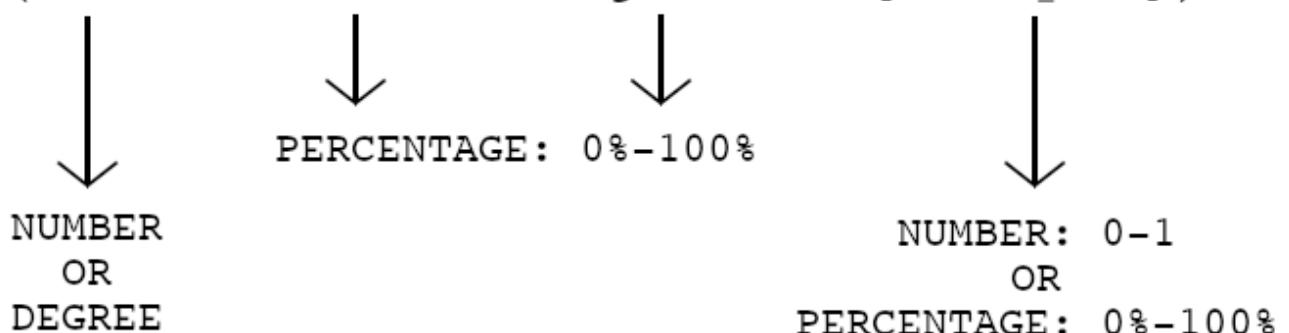
```
h1 {
    background-color: rgb(255, 0, 0, 0.5);
}
```

12.4. La notation fonctionnelle HSL

Exemple :

```
h1 {
    background-color: hsl(90deg, 100%, 50%)
}
```

`hsl(hue saturation lightness [/ alpha])`



`hsl(90deg, 100%, 50%)`

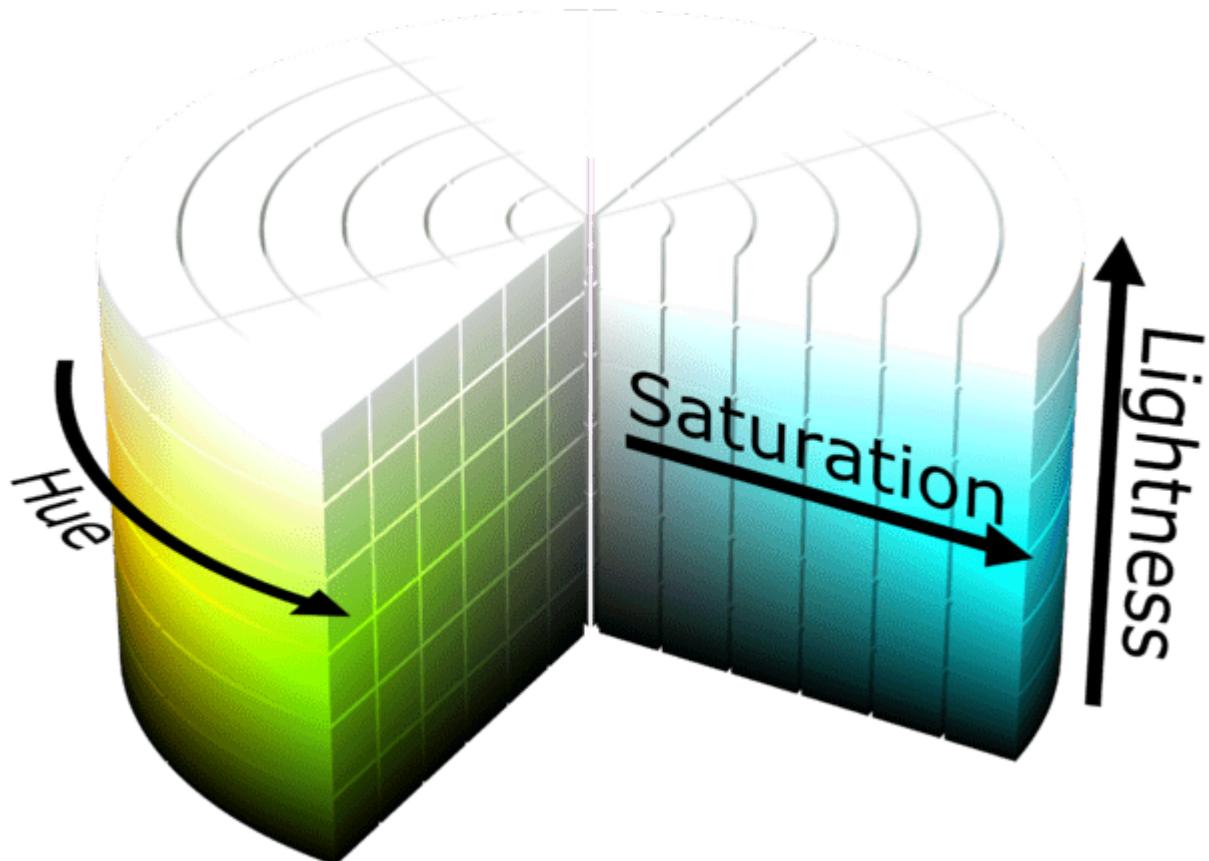
`hsl(90, 100%, 50%)`

`hsl(0.15turn, 50%, 75%)`

`hsl(0.15turn, 90%, 75%)`

`hsl(0.15turn, 90%, 50%)`

`hsl(270deg, 90%, 50%)`



12.5. La propriété currentColor

Exemple :

Nous allons définir la couleur du texte et laisser les autres propriétés à l'intérieur du lien `a` hériter de la couleur de la balise `p` via `currentColor`

```
p { color: #333; }

p a {
    text-decoration: none;
    color: currentcolor;
    border-bottom: 1px solid currentcolor;
}
```

13. LE GRADIENT

Les **dégradés CSS** sont représentés par le type de donnée `gradient` qui est un sous-ensemble du type `image`. L'utilisation de dégradés CSS permet d'afficher des transitions douces entre deux couleurs ou plus. Il existe trois sortes de dégradés :

- Les dégradés linéaires (créés avec la fonction [linear-gradient\(\)](#))
- Les dégradés radiaux (créés avec la fonction [radial-gradient\(\)\(en-US\)](#))
- Les dégradés coniques (créés avec la fonction [conic-gradient\(\)\(en-US\)](#))

13.1. Dégradé linéaire

Pour définir un dégradé sous sa forme la plus simple, il suffit d'avoir deux couleurs. Celles-ci permettent de placer ce qu'on appellera des arrêts de couleur (*color stops* en anglais). Il est nécessaire d'en avoir au moins deux, mais il est possible d'en avoir plus.

```
<div class="lineaire-simple"></div>
```

```
div {
    width: 120px;
    height: 120px;
}

.lineaire-simple {
    background: linear-gradient(blue, pink);
}
```

résultat :



13.2. Dégradé radial

Les dégradés radiaux sont similaires aux dégradés linéaires mais permettent d'obtenir un effet qui rayonne à partir d'un point. Il est possible de créer des dégradés circulaires ou elliptiques.

```
<div class="radial-simple"></div>
```

```
.radial-simple {  
    background: radial-gradient(red, blue);  
}  
  
div {  
    width: 120px;  
    height: 120px;  
}
```

résultat :

13.2.1. Positionner les points d'arrêt

```
<div class="degrade-radial"></div>
```

```


résultat :



### 13.3. Dégradés coniques



La fonction conic-gradient() permet de créer une image composée d'un dégradé de couleurs tournant autour d'un point (plutôt qu'une progression radiale). On pourra ainsi utiliser des dégradés coniques pour créer des camemberts ou des éventails de couleurs.



La syntaxe de conic-gradient() est semblable à celle de radial-gradient() mais les arrêts de couleur seront placés le long d'un arc plutôt que le long de la ligne émise depuis le centre. Les arrêts de couleur seront exprimés en pourcentages ou en degrés, ils ne pourront pas être exprimés sous forme de longueurs absolues.



Pour un dégradé radial, la transition entre les couleurs forme une ellipse qui progresse vers l'extérieur dans toutes les directions. Un dégradé conique verra la transition progresser le long de l'arc autour du cercle, dans le sens horaire. À l'instar des dégradés radiaux, il est possible de positionner le centre du dégradé et à l'instar des dégradés linéaires, on peut modifier l'angle du dégradé.



#### 13.3.1. dégradé conique simple



---



Comme pour les dégradés linéaires et radiaux, il suffit de deux couleurs pour créer un dégradé conique. Par défaut, le centre du dégradé sera situé au centre (point 50% 50%) et le début du dégradé commencera vers le haut :



```
<div class="degrade-radial"></div>
```



38 / 71 - JD.TOULOUSE


```

```

div {
    width: 120px;
    height: 120px;
}

.simple-conic {
    background: conic-gradient(red, blue);
}

```

13.4. Répétition de dégradé

https://developer.mozilla.org/fr/docs/Web/CSS/CSS/Images/Using_CSS_gradients

14. VARIABLES CSS

Exemple :

```

:root {
    --h1-bg-color: lightgrey;

h1 {
    background-color: var(--h1-bg-color);
}

```

15. BOX-SHADOW (boîte avec ombrage porté)

documentation : <https://developer.mozilla.org/fr/docs/Web/CSS/box-shadow>

Box Shadow

```
offset-x    offset-y    blur-radius  spread-radius  color
          ↘          ↘          ↘          ↘          ↘
box-shadow: 0 0 10px 5px #ff000066;
```

Outer Shadow



```
box-shadow: 0 4px 6px #77777736,
           0 4px 6px #77777761;
```

Inner Shadow



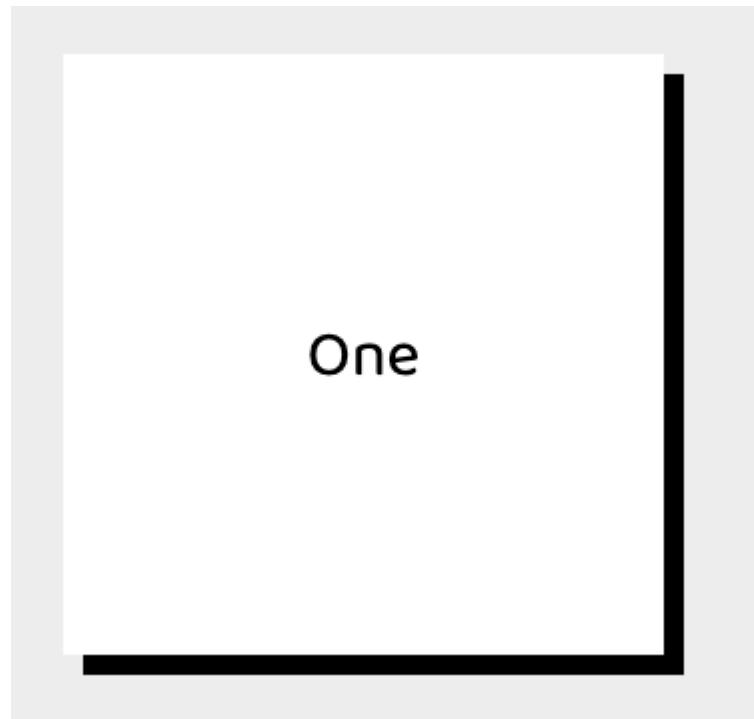
```
box-shadow: inset 5px 5px 10px rgba(166, 188, 200, 0.75),
           inset -5px -5px 15px rgba(255, 255, 255, 0.75);
```

15.1. Ajouter une ombre de boîte à une div

```
<div class='one'>One</div>
```

```
.one {
  box-shadow: 5px 5px;
}
```

résultat :



15.2. Ajouter une couleur à l'ombre de boîte d'une div

```
<div class='one'>Two</div>
```

```
.two {  
  
    box-shadow: 5px 5px red;  
}
```

résultat :



15.3. Ajouter un aspect flouté à l'ombre de boîte d'une div

```
<div class='one'>Three</div>
```

```
.three {  
  
    box-shadow: 5px 5px red;  
}
```

résultat :



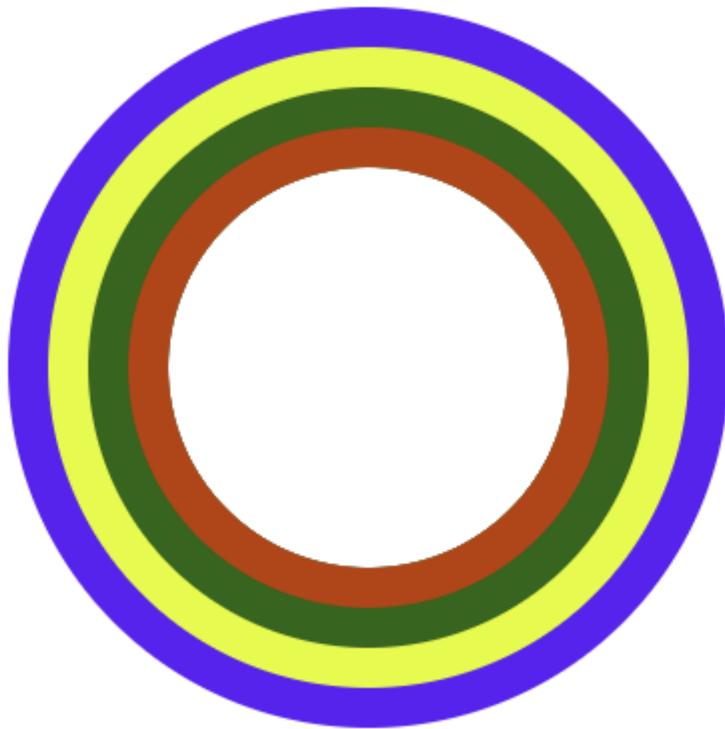
15.4. Amusez-vous avec les box-shadow

```
<div></div>
```

```
body {  
  
    display: flex;  
    height: 100vh;  
    justify-content: center;  
    align-items: center;  
}
```

```
div {  
    border-radius: 50%;  
    box-shadow: 0px 0px 0px 10px red, 0px 0px 0px 20px green, 0px 0px 0px 30px  
yellow, 0px 0px 0px 40px blue, 0px 0px 0px 50px skyblue, 0px 0px 0px 60px  
pink;  
    width: 100px;  
    height:100px;  
    margin: 3em;  
}
```

résultat :



16. UNITES CSS

Il existe plusieurs types de valeur numérique que vous pourrez utiliser en CSS.

ABSOLUTE

Pixels (px)

Inches (in)

Centimeters (cm)

Millimeters (mm)

Points (pt)

Picas (pc)



RELATIVE

Percentages (%)

Font sizes (em, rem)

Character sizes (ex, ch)

Viewport dimensions (vh, vw)

Viewport max (vmax)

Viewport min (vmin)

16.1. Unités absolues

La liste qui suit contient uniquement des unités de longueur **absolue**. Ces quantités ne sont pas relatives à quoi que ce soit d'autre et leur taille sera considérée comme constante.

Unité	Nom	Équivalent à
cm	Centimètres	$1\text{cm} = 38\text{px} = 25/64\text{in}$
mm	Millimètres	$1\text{mm} = 1/10\text{th of } 1\text{cm}$
Q	Quarts de millimètre	$1\text{Q} = 1/40\text{th of } 1\text{cm}$
in	Pouces (<i>inches</i>)	$1\text{in} = 2.54\text{cm} = 96\text{px}$
pc	Picas	$1\text{pc} = 1/6\text{e de } 1\text{in}$
pt	Points	$1\text{pt} = 1/72\text{e de } 1\text{in}$
px	Pixels	$1\text{px} = 1/96\text{th de } 1\text{in}$

La plupart de ces unités sont utiles pour l'impression plutôt que pour l'affichage sur un écran. Ainsi, on n'utilise généralement pas `cm` (centimètres) sur un écran. La seule unité ici que vous rencontrerez fréquemment est `px` (pixels).

16.2. Unités relatives

Les unités de longueur relative permettent d'exprimer des quantités relatives à quelque chose d'autre comme la taille de la police de l'élément parent ou la taille de la zone d'affichage (viewport). L'avantage d'utiliser des unités relatives est qu'avec un peu d'organisation, on peut faire que la taille du texte ou d'autres éléments se mette à l'échelle, relativement à quelque chose d'autre sur la page. La plupart des unités les plus utiles pour le développement web sont présentes dans le tableau qui suit.

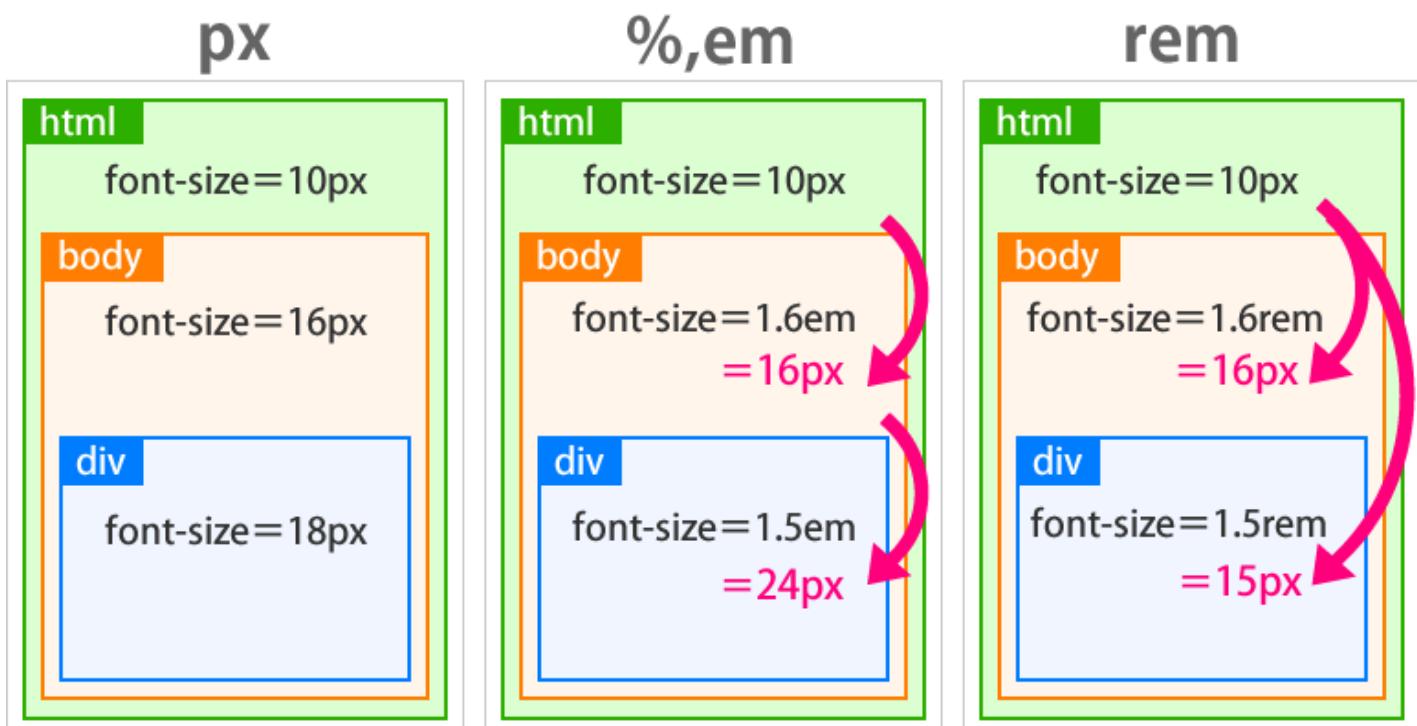
Unité	Relative à
em	Pour les propriétés typographiques comme <code>font-size</code> , relative à la taille de la police de l'élément parent. Pour les autres propriétés comme <code>width</code> , relative à la taille de la police de l'élément;
ex	La hauteur d'x de la police de l'élément.
ch	La chasse/avance du glyphe « 0 » pour la police de l'élément.
rem	La taille de la police pour l'élément racine.
lh	La hauteur de ligne pour l'élément.
vw	1% de la largeur du <i>viewport</i> (la zone d'affichage).
vh	1% de la hauteur du <i>viewport</i> (la zone d'affichage).
vmin	1% de la plus petite dimension du <i>viewport</i> (la zone d'affichage).
vmax	1% de la plus grande dimension du <i>viewport</i> (la zone d'affichage).

16.3. REM vs EM vs % vs PX

source : <https://www.aleksandrkhannisan.com/blog/respecting-font-size-preferences-rems-62-5-percent/>

```
html {
    font-size: 62.5%;
```

```
body {  
  
    font-size: 1.6rem;  
}  
  
h1 {  
  
    font-size: 1.8rem;  
}
```



REM

HTML (36PX)

BODY ($0.8\text{rem} = 0.8 * 36 = 28.8\text{px}$)

DIV($0.8\text{rem} = 0.8 * 36 = 28.8\text{px}$)

DIV($0.8\text{rem} = 0.8 * 36 = 28.8\text{px}$)

EM

HTML (36PX)

BODY ($0.8\text{em} = 0.8 * 36 = 28.8\text{px}$)

DIV($0.8\text{em} = 0.8 * 28.8 = 23.04$)

DIV($0.8\text{em} = 0.8 * 23.04 = 18.43$)

17. MANIPULER LE TEXTE

17.1. Indentation

La propriété **text-indent** définit la longueur qui doit être laissée avant le début de la première ligne d'un élément contenant du texte.

Exemple :

```
p {
    text-indent: 5em;
    background: powderblue;
}
```

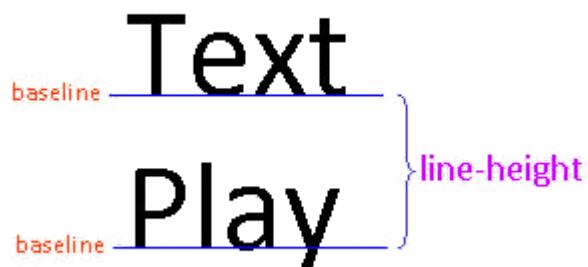
Lorem ipsum dolor sit amet, consecetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consecetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

17.2. Hauteur de ligne

<https://iamvdo.me/blog/css-avance-metriques-des-fontes-line-height-et-vertical-align>

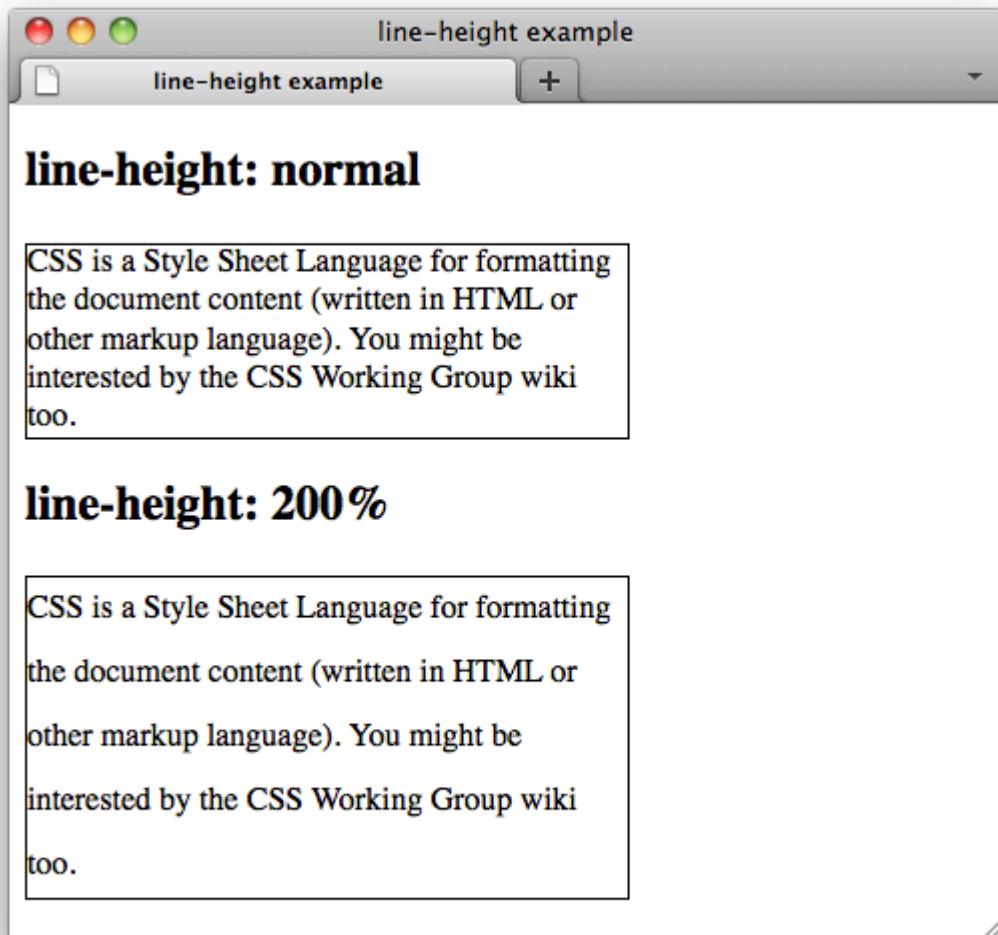
La propriété **line-height** définit la longueur qui doit être laissée avant le début de la première ligne (ligne de base) d'un élément contenant du texte.



Exemple :

Toutes les règles qui suivent fourniront un résultat équivalent :

```
div { line-height: 1.2; font-size: 10pt }
div { line-height: 1.2em; font-size: 10pt }
div { line-height: 120%; font-size: 10pt }
```



Exemple 2 :

Astuce pour center un texte dans un cadre en pure CSS.

```
.center {  
    line-height: 150px;  
    height: 150px;  
    border: 3px solid green;  
    text-align: center;  
}
```

```
<p class="center">Astuce pour center un texte dans un cadre en pure CSS.</p>
```

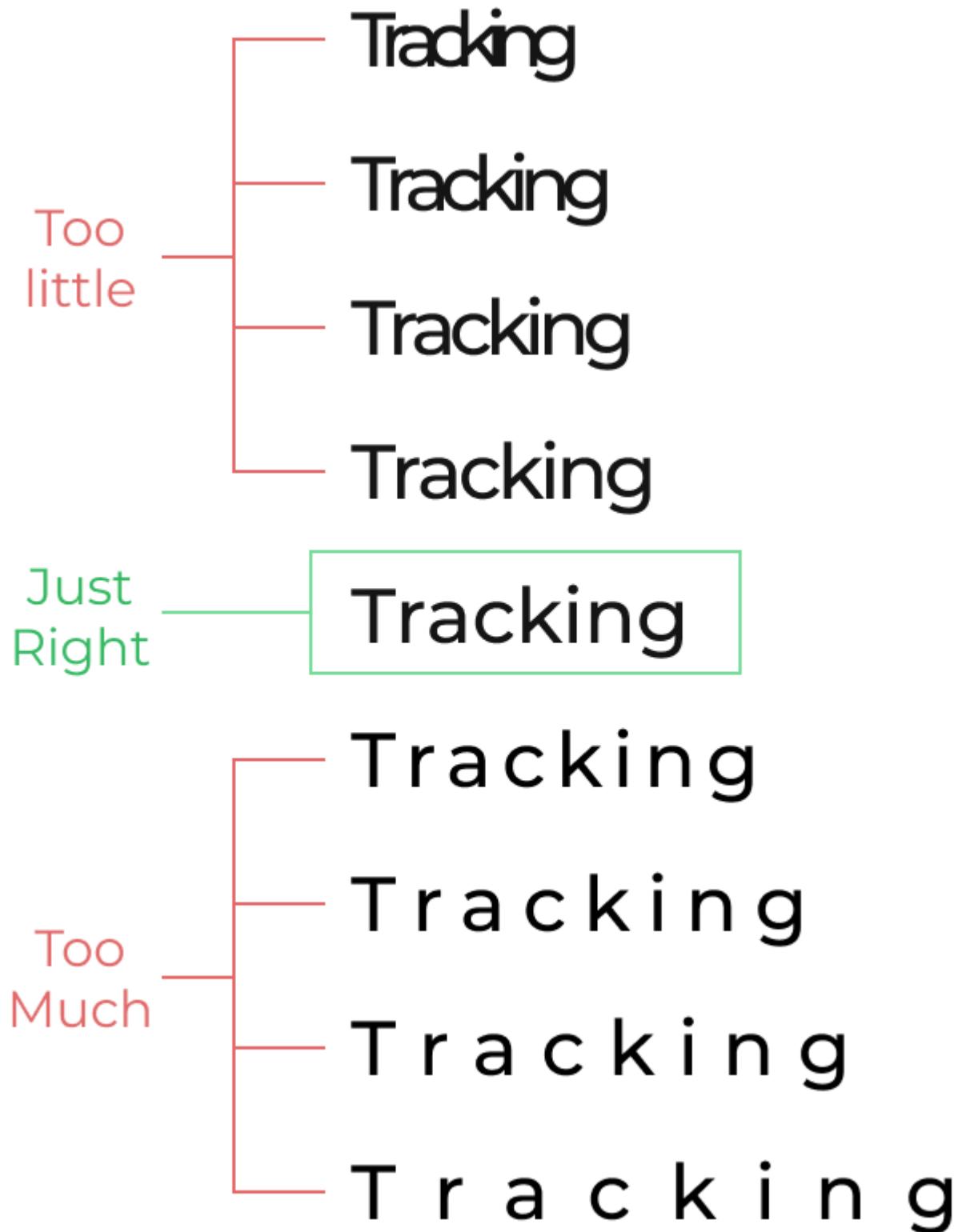
Astuce pour centrer un texte dans un cadre en pure CSS.

17.3. Espacement entre les lettres : letter-spacing

La propriété **letter-spacing** définit [l'interlettre](#) utilisée pour les caractères qui composent le texte.

Exemple :

```
.ls10 {  
letter-spacing: 10px  
}
```



17.4. Espacement entre les mots : `word-spacing`

La propriété `word-spacing` définit la règle d'espacement utilisée entre les balises et entre les mots.

Exemple :

```
p {
    word-spacing: 2em;
}
```

This line has **normal** word-spacing

This line has a word-spacing of **2em**

This line has a word-spacing of **5px**

17.5. Césure : word-break

La propriété **word-break** est utilisée pour définir la façon dont la césure s'applique pour les endroits où le texte dépasserait de sa boîte de contenu.

normal : Le passage à la ligne classique est utilisé.

break-all : La césure peut être insérée après n'importe quel caractère (ne s'applique pas pour les textes en chinois, japonais et coréen).

 Lorem ipsum dolor sit amet consectetur adipisicing elit. Porro odit dolor illum incident molestiae. Unde nihil dolor em in suscipit pariatur.

 Lorem ipsum dolor, sit amet consectetur adipisicing elit. Architecto omnis molestias rem doloremque quo voluptate, ab quam! Aliquam, error dolore.

17.6. Gestion des blancs white-space

La propriété **white-space** est utilisée pour décrire la façon dont les blancs sont gérés au sein de l'élément.

Exemple :

```
white-space: normal;
```

But ere she from the church-door stepped She smiled and told us why: 'It was a wicked woman's curse,' Quoth she, 'and what care I?' She smiled, and smiled, and passed it off Ere from the door she stept—

```
white-space: nowrap;
```

But ere she from the church-door stepped

17.7. Tronquer une phrase : text-overflow

La propriété **text-overflow** définit la façon dont le contenu textuel qui dépasse d'une boîte est signalé pour les utilisateurs.

Exemple :

```
.truncate {
  width: 200px;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

```
<p class="truncate">
  Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque culpa accusantium totam sunt delectus assumenda facilis possimus obcaecati dolores neque voluptatibus fugiat blanditiis dicta, molestiae ea dignissimos quisquam molestias voluptates!
</p>
```

Lorem ipsum dolor sit amet ...

18. TYPOGRAPHIE CSS

18.1. FONT

CSS Fonts est un module CSS qui définit des propriétés relatives aux polices de caractères et la façon dont les ressources des polices sont chargées. Il permet de définir le style d'une police, comme sa famille, sa taille ou sa graisse ainsi que la variante du glyphe à utiliser dans le cas des polices disposant de plusieurs glyphes par caractère. Il permet également de définir la hauteur d'une ligne.

La propriété **font** est :

- une propriété raccourcie permettant de définir rapidement [font-style](#), [font-variant](#), [font-weight](#), [font-stretch](#), [font-size](#), [line-height](#) et [font-family](#)

SYNTAXE

```
font: italic small-caps bolder condensed 16px/3 cursive;
```



18.2. font-weight

La propriété [CSS](#) **font-weight** permet de définir la graisse utilisée pour le texte. Les niveaux de graisse disponibles dépendent de la police (cf. [font-family](#)). Certaines fontes n'existent qu'avec les niveaux de graisses **normal** et **bold**.

SYNTAXE

```
font-weight: normal;
font-weight: bold;
```

```
font-weight: lighter;
font-weight: bolder;

font-weight: 100;
font-weight: 200;
font-weight: 300;
font-weight: 400;
font-weight: 500;
font-weight: 600;
font-weight: 700;
font-weight: 800;
font-weight: 900;
```

Correspondance entre les valeurs numériques et les noms communément utilisés :

Valeur	Nom communément utilisé
100	<i>Thin (Hairline)</i>
200	<i>Extra Light (Ultra Light)</i>
300	<i>Light</i>
400	<i>Normal</i>
500	<i>Medium</i>
600	<i>Semi Bold (Demi Bold)</i>
700	<i>Bold</i>
800	<i>Extra Bold (Ultra Bold)</i>
900	<i>Black (Heavy)</i>
950	<i>Extra Black (Ultra Black)</i>

18.3. INSERER UNE FONTS

18.3.1. Avec la règle @import

La règle **@import** est utilisée afin d'importer des règles à partir d'autres feuilles de style. Ces règles **@** doivent être utilisées avant toutes les autres règles, à l'exception de **@charset**.

@import n'est pas une instruction imbriquée et ne peut donc pas être utilisée à l'intérieur de groupe de règles conditionnelles.

Exemple :

```
@import url('https://fonts.googleapis.com/css?family=Montserrat');
```

18.3.2. @font-face

La règle **@font-face** permet de définir les polices d'écriture à utiliser pour afficher le texte de pages web. Cette police peut être chargée depuis un serveur distant ou depuis l'ordinateur de l'utilisateur. Si la fonction **local()** est utilisée, elle indique à l'agent utilisateur de prendre en compte une police présente sur le poste de l'utilisateur.

Exemple :

```
@font-face {
    font-family: 'quadranta';
    src: url('fonts/quadranta.eot?') format('eot'),
         url('fonts/quadranta.otf') format('truetype'),
         url('fonts/quadranta.woff2') format('woff2'),
         url('fonts/quadranta.woff') format('woff'),
         url('fonts/quadranta.ttf') format('truetype'),
         url('fonts/quadranta.svg#QuadrantaBold') format('svg');
    font-weight: normal;
    font-style: normal;
}

body {
    font-family: quadranta, sans-serif;
}
```

<p> Et voici la police quadrata.</p>

18.3.3. Depuis le fichier html

```
<link href="https://fonts.googleapis.com/css?family=Montserrat"
rel="stylesheet" type="text/css">
```

19. DISPOSITION MULTI COLONNES

La propriété `columns` est une propriété raccourcie permettant de définir les deux propriétés `column-width` (qui définit la largeur des colonnes) et `column-count` (qui définit le nombre de colonnes) en même temps.

Exemple 1:

```
.container {
  column-count: 3;
  column-gap: 20px;
  column-rule: 4px dotted rgb(79, 185, 227);
}
```

The screenshot shows a website header with the title 'The Envato Times'. Below the header is a navigation bar with links for 'HOME', 'ABOUT', 'WEBSITES', and 'CONTACT'. The main content area features a subheading 'THIS IS A SUBHEADING TITLE' followed by two columns of text. The left column contains approximately 150 words of placeholder text (Lorem ipsum). The right column also contains approximately 150 words of placeholder text. The text is styled with a serif font and a dotted border rule between the columns.

THIS IS A SUBHEADING TITLE

Cum porta risus odio tincidunt? Ultrices, nascetur lundium tincidunt ridiculus? Enim nisi ac nec nec vel in ac pid ultricies nunc, cras porta ultrices. Hac lacus nisi non! Ut turpis, tempor eros? Tortor et, platea turpis scelerisque duis porttitor augue ridiculus nec et mid, sociis? Est ac a? Arcu elementum eros in ultrices dignissim pulvinar tincidunt elementum, hac platea hac. Purus et egestas vel, elementum facilisis, turpis sit augue ultricies, porta augue, enim massa cursus ac augue! Porta sed cras. Ut a augue ac in, dis pulvinar? Cras amet aenean magna adipiscing turpis mattis purus placerat, placerat nunc

Dis? Rhoncus ultricies porta purus habitasse cum dictumst urna dictumst sociis, porttitor augue rhoncus enim! Lundium, magnis adipiscing ut! Lorem in nascetur lorem sagittis urna! Quis eu rhoncus mid platea cras facilisis! Eros in facilisis! Urna sit ac montes. Rhoncus et eros urna lectus, adipiscing lorem ultrices natoque placerat, pid tincidunt magna mus, placerat pid velit tristique sit amet. Pulvinar nec, parturient eros cras, parturient nec lorem, dolor purus phasellus porttitor. In porta cras, dolor odio ac porta sociis et cras quis, augue quis cum nisi aliquam lorem, ut. Cras mattis, phasellus dignissim augue enim massa nascetur, sit ut.

Turpis purus aliquet. Habitasse placerat odio ac sociis mid, auctor cursus parturient eros! Lundium, vut, urna ridiculus? Penatibus ac et mus elementum. Sed magna augue, velit odio magna pellentesque a, lacus, aliquam elementum, in massa! Dapibus turpis, placerat purus, lectus in sagittis porta massa aenean! Turpis enim non adipiscing lundium parturient, a natoque egestas. Mid dictumst porta. Risus lorem aliquam velit mid, placerat! Et lectus sociis scelerisque ac turpis. Magna porta? Augue, augue in. Porta dolor, et amet, ac. Cum scelerisque tincidunt dignissim in, pellentesque habitasse! Elit? Mus porttitor in a nec, augue! Elit nisi purus? Porta platea integer.

20. TRANFORMATION

La propriété **transform** modifie l'espace de coordonnées utilisé pour la mise en forme visuelle. Grâce à cette propriété, il est possible de translater les éléments, de les tourner, d'appliquer des homothéties, de les distordre pour en changer la perspective.

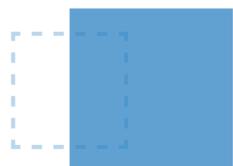
- `translate()`
- `rotate()`
- `scale()`
- `skew()`
- `matrix()`

Exemple:

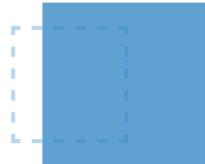
Attention à l'ordre des transformations !!

```
<div></div>
```

```
div {
    border: solid red;
    transform: translate(50%, 0) scale(1.5);
    width: 140px;
    height: 60px;
}
```



```
.square {
    transform: scale(1.5) translate(50%, 0);
```



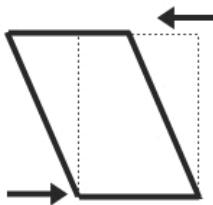
```
.square {
    transform: translate(50%, 0) scale(1.5);
```

20.1. Transformations 2D possibles

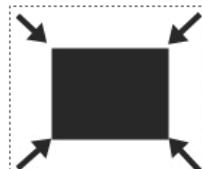
Rotate



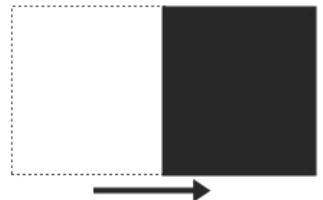
Skew



Scale



Translate



20.2. Transformations 3D possibles

Les transformations 3D utilisent la même propriété que les transformations 2D, à savoir la propriété « transform ».

En revanche, on a de nouvelles fonctions de transformations :

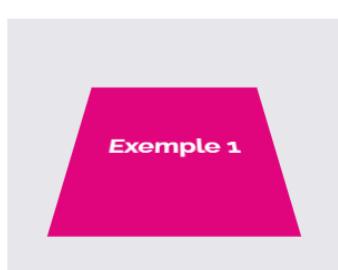
- **rotate3d(x,y,z,angle)** : pour des rotations en 3D.
- **translate3d(x,y,z)** : pour des déplacements en 3D.
- **scale3d(x,y,z)** : pour des changements d'échelle en 3D.
- **matrix3d()** : spécifie une matrice de transformation 4×4 avec 16 valeurs entre les parenthèses.

Ce sont les versions courtes mais on peut toujours utiliser les versions longues avec `scaleX()`, `scaleY()`,... en les complétant avec `scaleZ()`, `translateZ()` et `rotateZ()` qui précisent le 3ème axe. Vous noterez qu'on a perdu la fonction « skew ».

De nouvelles propriétés sont également apparues pour pouvoir gérer cet affichage en 3D :

- **perspective** : sans perspective, pas de 3D. Elle représente la notion de profondeur.
- **perspective-origin** : définit l'origine de la perspective (Là où les points de fuite des éléments se rejoignent).
- **transform-style** : définit si les éléments inclus subissent les transformations en 3D. 2 valeurs sont possibles : flat et preserve-3d.
- **backface-visibility** : précise si les faces arrières des éléments sont visibles ou non. 2 valeurs sont possibles : visible et hidden.

Rotation x



Rotation y



Rotation z



```
<div class="rotation3d">
  <div class="run-rotation">
    <figure class="box box-1">Exemple 1</figure>
  </div>
</div>

<div class="rotation3d">
  <div class="run-rotation">
    <figure class="box box-2">Exemple 2</figure>
  </div>
</div>

<div class="rotation3d">
  <div class="run-rotation">
    <figure class="box box-3">Exemple 3</figure>
  </div>
</div>
```

```
.rotation3d {
  background: #eaeaea;
  float: left;
}

.box {
  background: #e4087e;
  height: 120px;
  line-height: 120px;
  text-align: center;
  width: 120px;
}
```

```

color: #fff;
font-weight: 700;
}

.run-rotation {
  cursor: pointer;
  transform-style: preserve-3d;
}

.run-rotation:hover {
  animation: run-rotation 5s linear infinite;
}

@keyframes run-rotation {
  0% {
    transform: rotateY(0deg);
  }
  100% {
    transform: rotateY(360deg);
  }
}

.box-1 {
  transform: perspective(200px) rotateX(45deg);
}

.box-2 {
  transform: perspective(200px) rotateY(45deg);
}

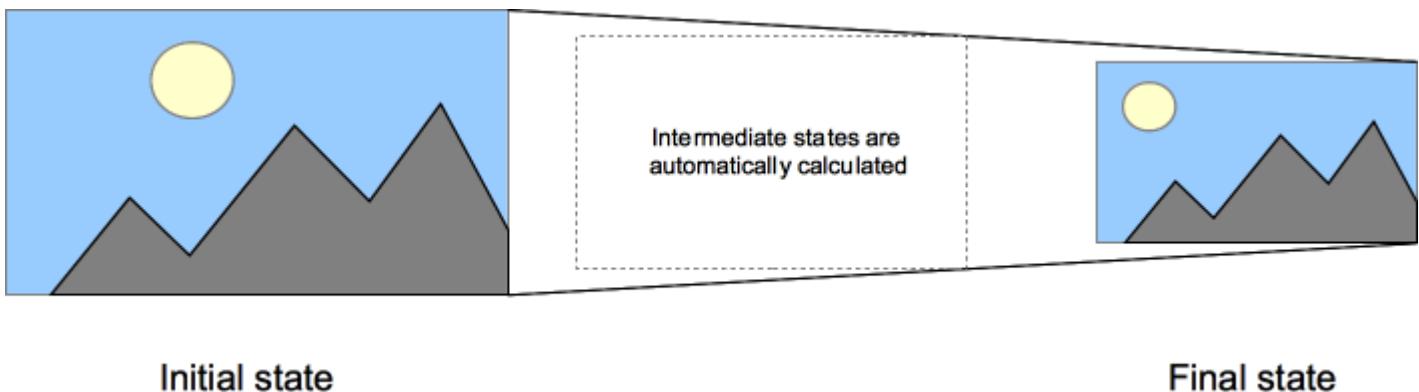
.box-3 {
  transform: perspective(200px) rotateZ(45deg);
}

```

21. TRANSITIONS CSS

Les **transitions CSS** permettent de contrôler la vitesse d'animation lorsque les propriétés CSS sont modifiées. Plutôt que le changement soit immédiat, on peut l'étaler sur une certaine période. Ainsi, si on souhaite passer un élément de blanc à noir, on pourra utiliser les transitions CSS afin que cette modification soit effectuée progressivement, selon une courbe d'accélération donnée.

Les animations qui utilisent des transitions entre deux états sont souvent appelées *transitions implicites* car l'état initial et l'état final sont définis implicitement par le navigateur.



Les transitions CSS vous permettent de choisir :

- les propriétés à animer en les listant explicitement
- le début de l'animation
- la durée de l'animation
- la façon dont la transition s'exécutera
- [transition-property](#)

Cette propriété définit le nom des propriétés CSS pour lesquelles on veut appliquer des transitions. Seules les propriétés listées ici seront sujettes aux transitions. Les modifications appliquées aux autres propriétés seront instantanées.

- [transition-duration](#)

Cette propriété définit la durée de la transition. On peut définir une durée pour toutes les transitions ou une durée pour chacune des propriétés.

- [transition-timing-function](#)

Cette propriété définit une fonction qui décrit la façon dont les valeurs intermédiaires sont calculées. On utilise pour cela des [fonctions de temporisation](#).

- [transition-delay](#)

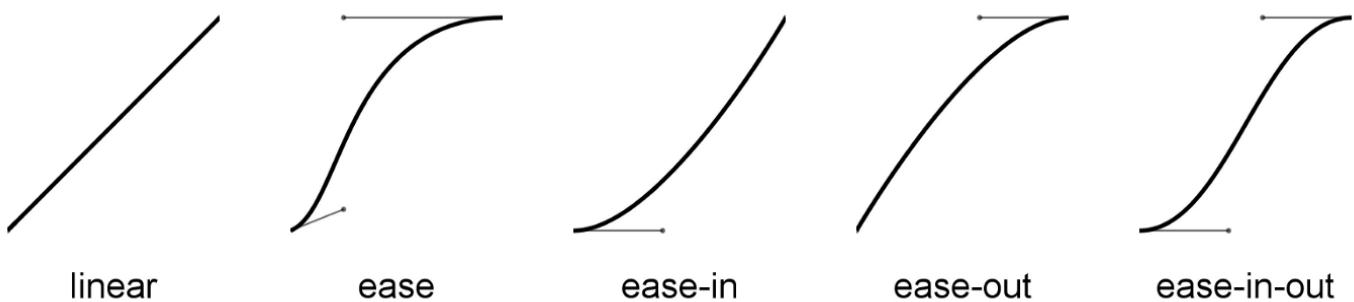
Cette propriété indique le temps à attendre entre le moment où la propriété est modifiée et le début de la transition.

La syntaxe de la propriété raccourcie [transition](#) est :

```
div {
    transition: <property> <duration> <timing-function> <delay>;
}
```

21.1. Timing-function

La propriété [transition-timing-function](#) décrit la façon dont les valeurs intermédiaires des propriétés CSS affectées par un [effet de transition](#) sont calculées. Ceci permet donc de définir une



- **linear** : démarre et se termine à la même vitesse. Mieux pour les propriétés comme la couleur et l'opacité plutôt que le mouvement.
- **ease** : Accélère rapidement et ralentit progressivement. C'est le CSS par défaut, une belle fonction polyvalente.
- **ease-in** : Accélère progressivement et se termine rapidement. Idéal pour animer des éléments hors de vue.
- **ease-out** : Démarre à grande vitesse et ralentit. Idéal pour animer des éléments en vue.
- **ease-in-out** : Accélère progressivement puis ralentit. Comme la facilité, mais avec un démarrage plus lent. Peut être utile pour les animations qui commencent et se terminent à l'écran.

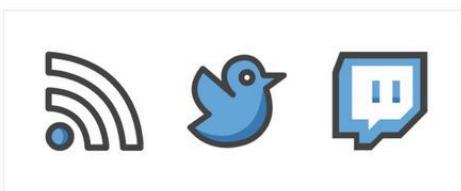
22. IMAGES

22.1. Filtres

La propriété CSS **filter** permet d'appliquer des filtres et d'obtenir des effets graphiques de flou, de saturation, etc. Les filtres sont généralement utilisés pour ajuster le rendu d'une image, d'un arrière-plan ou des bordures.

Exemple :

```
img {
  filter: grayscale(100%);
}
```



```

```



```
img { filter: blur(7px); }
```



```
img { filter: brightness(0.4); }
```



```
img { filter: contrast(200%); }
```



```
img { filter: drop-shadow(16px 16px 20px blue); }
```



```
img { filter: grayscale(50%); }
```



```
img { filter: hue-rotate(90deg); }
```



```
img { filter: invert(75%); }
```



```
img { filter: opacity(25%); }
```



```
img { filter: saturate(400%); }
```



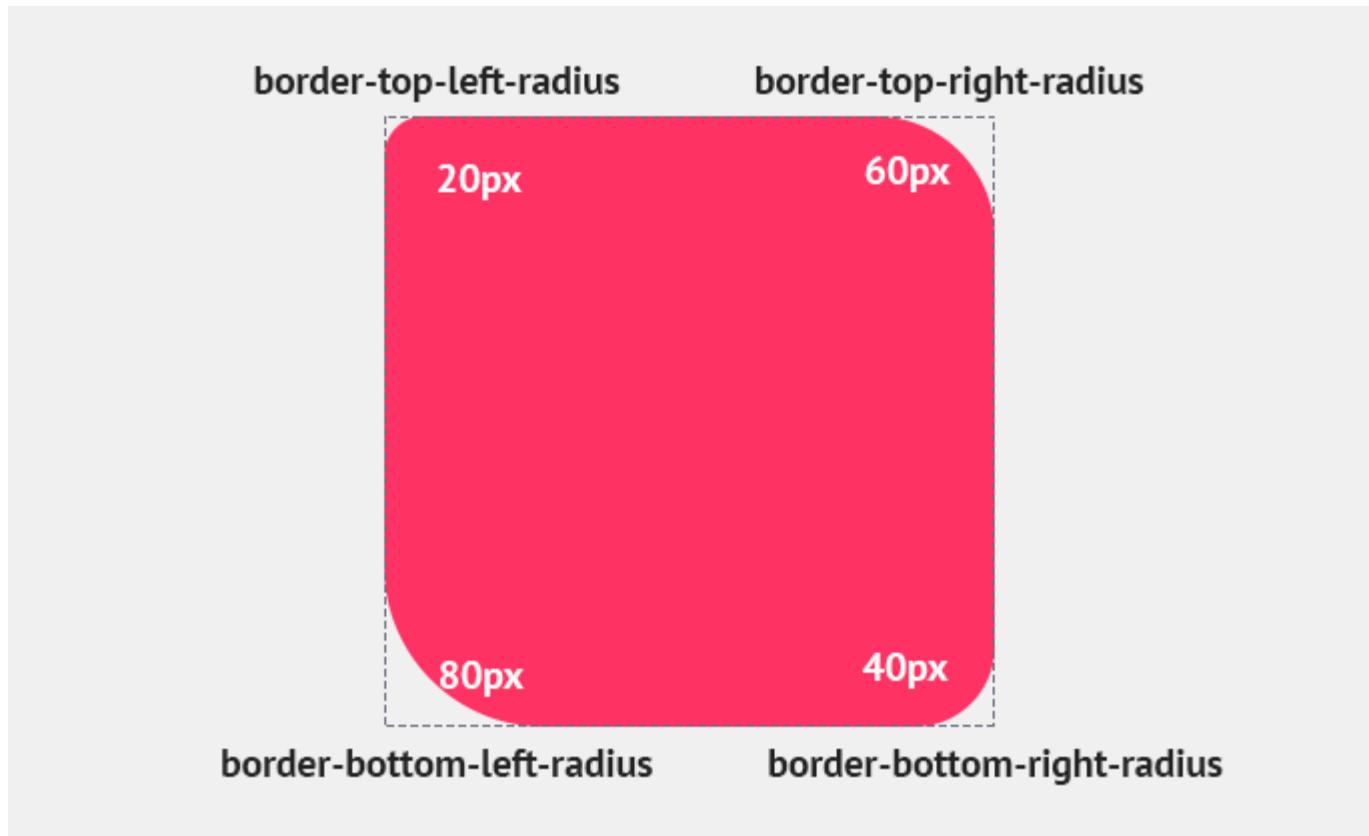
```
img { filter: sepia(60%); }
```

22.2. border-radius

La propriété **CSS border-radius** permet de définir des coins arrondis pour la bordure d'un élément. La courbure de chaque coin est définie avec un ou deux rayons de courbures qui permettent de définir un arc de cercle ou un arc d'ellipse.

Exemple :

```
img{
    border-radius: 20px 60px 40px 80px;
}
```



22.3. La propriété background

La propriété `CSS background` est une [propriété raccourcie](#) qui permet de définir les différentes valeurs des propriétés liées à la gestion des arrière-plans d'un élément (couleur, image, origine, taille, répétition, etc.).

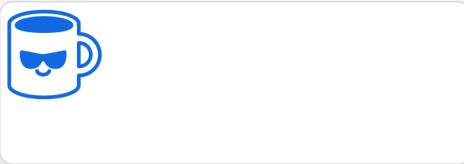
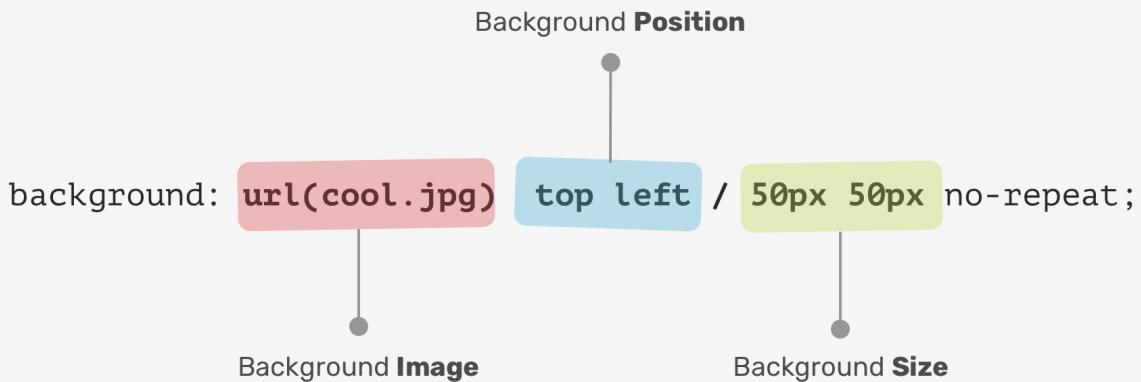
Valeur initiale	pour chaque propriété individuelle de la propriété raccourcie : <code>background-image</code> : none <code>background-position</code> : 0% 0% <code>background-size</code> : auto auto <code>background-repeat</code> : repeat <code>background-origin</code> : padding-box <code>background-clip</code> : border-box <code>background-attachment</code> : scroll <code>background-color</code> : transparent
------------------------	---

22.4. La propriété background-image

La propriété `background-image` permet de définir une ou plusieurs images comme arrière(s)-plan(s) pour un élément.

Exemple :

```
.element {  
    background: url(cool.jpg) top left/50px 50px no-repeat;  
}
```



Background **Position**

background: **url(cool.jpg)** **top 20px left 20px** / **50px 50px** no-repeat;

background: **url(cool.jpg)** **left 20px top 20px** / **50px 50px** no-repeat;



```
.container {
    background-color: #000;
    background-image: url(images/bg.gif);
    background-repeat: no-repeat;
    background-position: left top;
}
```



```
.container {
    background: #000 url(images/bg.gif) no-repeat left top;
```

22.5. La propriété background-size

La propriété [CSS](#) **background-size** définit la taille des images d'arrière-plan pour l'élément. La taille de l'image peut être contrainte, complètement ou partiellement afin de conserver ses proportions.

Exemple:

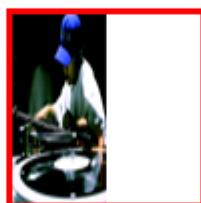
```
#exemple {
    background: url(mountain.jpg);
    background-repeat: no-repeat;
    background-size: contain;
}
```

background-size:

auto;



50% 100%;

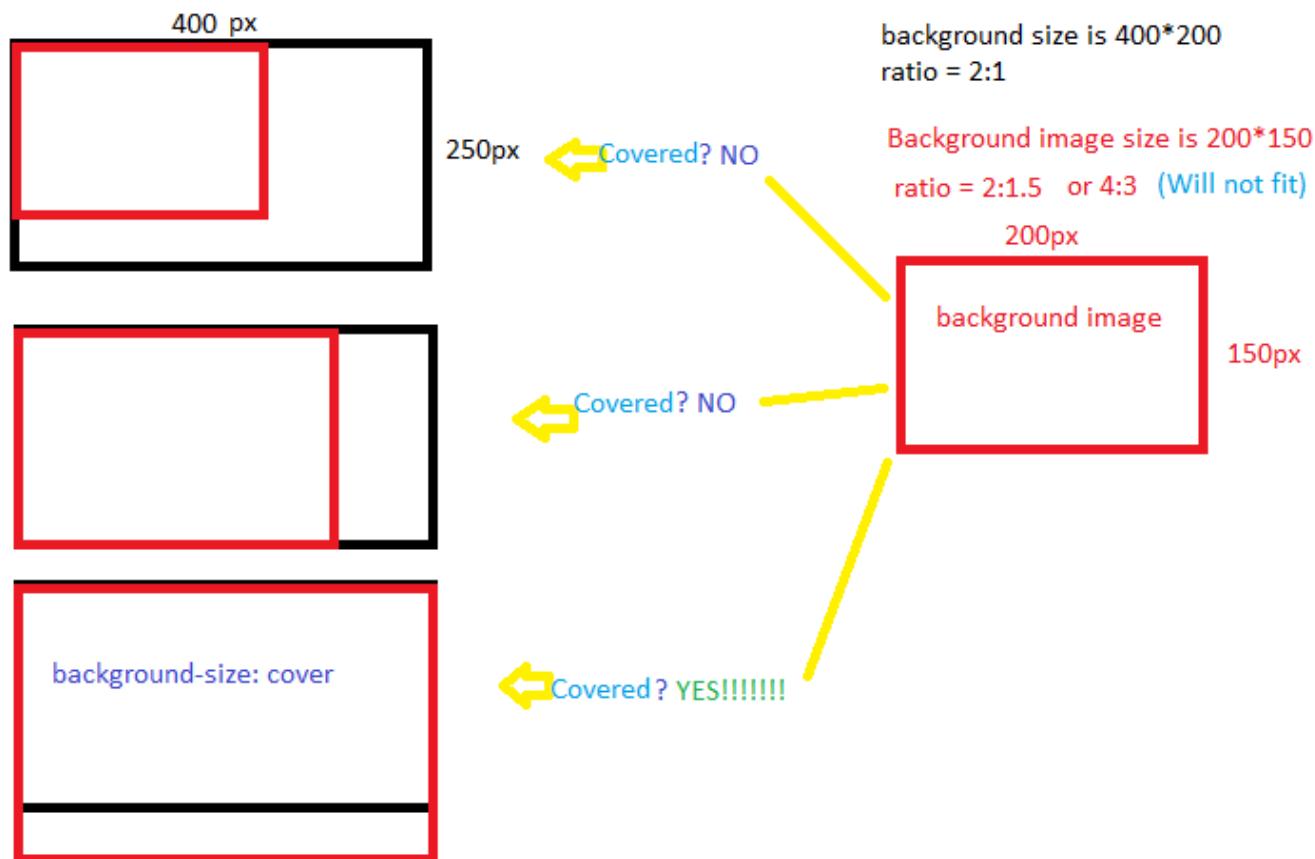


cover;



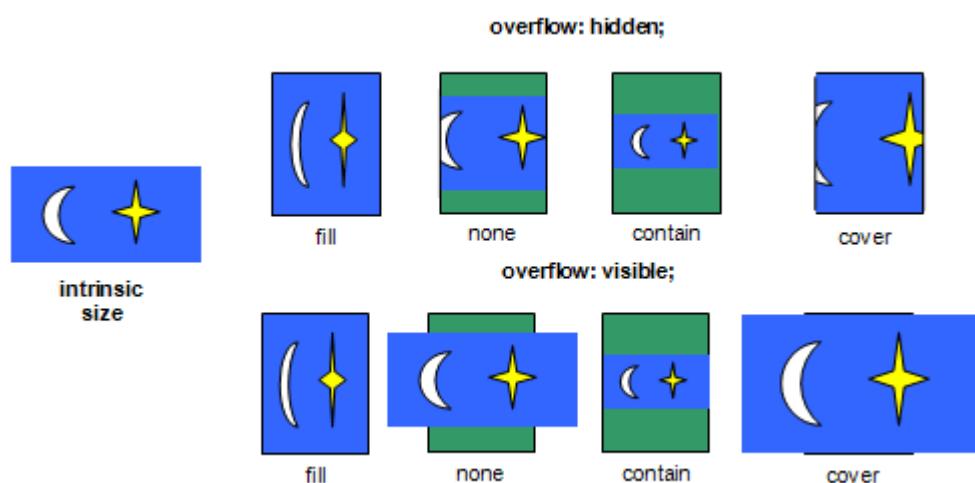
contain;

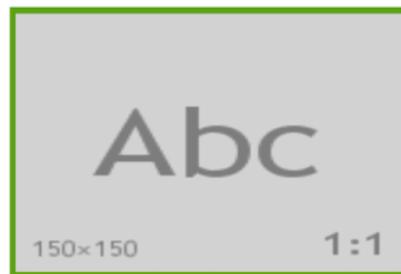
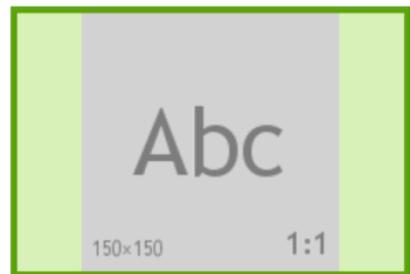
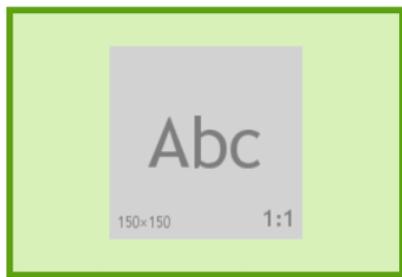
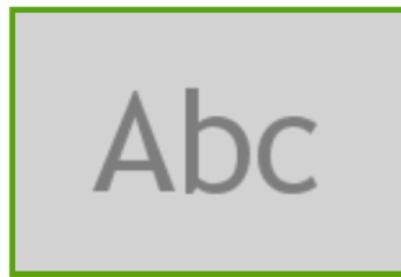
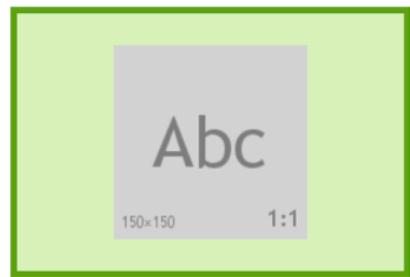




22.6. La propriété `object-fit`

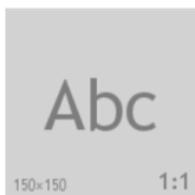
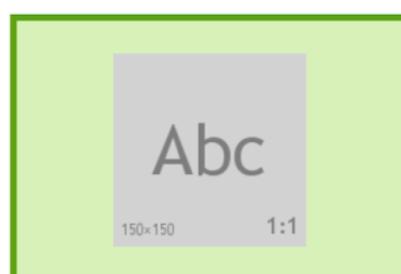
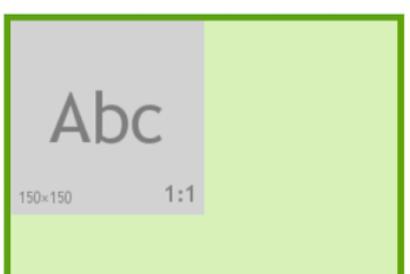
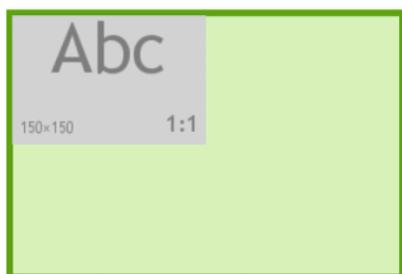
La propriété CSS `object-fit` définit la façon dont le contenu d'un élément remplacé (<https://developer.mozilla.org/fr/docs/Web/HTML/Element/Img>) ou [] par exemple) doit s'adapter à son conteneur en utilisant sa largeur et sa hauteur.



Original image**object-fit:fill****object-fit:contain****object-fit:scale-down****object-fit:cover****object-fit:none**

22.7. La propriété `object-position`

La propriété **object-position** détermine l'alignement d'un [élément remplacé](#) au sein de sa boîte. Les zones de la boîte qui ne sont pas recouvertes par le contenu de l'élément remplacé montreront l'arrière-plan de l'élément.

Original image**object-position:50% 50%****object-position:0 0****object-position:0 -50px****object-position:top right****object-position:bottom**