CSS

130 Base en CSS

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision
2023-11-18	Ajout des exemples (en particulier, les animations)
2024-11-19	Ajout info sur link ref versus @import

Sommaire

A Introduction

Le HTML qui permet de décrire la structure d'un document reste un langage austère : pas de couleurs, pas d'effets et une certaine rigidité.

Pour changer une police dans une page, il faut plusieurs balises, rendant la lecture du document toujours plus compliquée.

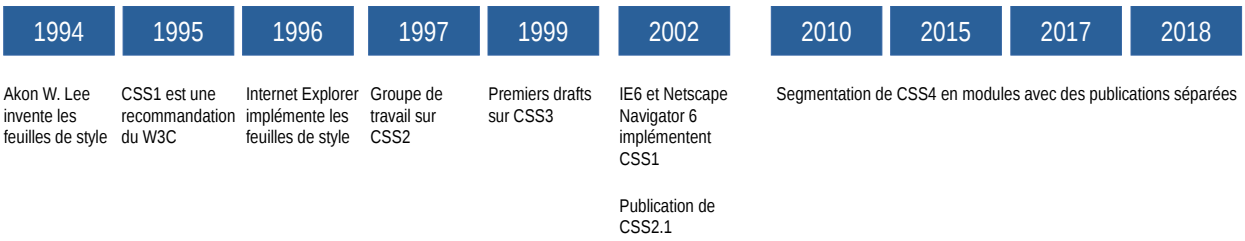
```
<font size="2"><font face="arial"><font color="red"><b>Bonjour !</b></font></font></font>
```

exemple de code HTML avant CSS

De nombreux sites empilent les balises pour obtenir un affichage plus joli mais cela génère des lenteurs (plus de code), des erreurs (plus de risques d'oubli de fermeture de balises) et une lecture difficile pour le développeur.

A.1 Historique

En 1994, le langage CSS sort de l'imagination de Håkon W. Lie : ce dernier tente d'adapter les fonctionnalités disponibles dans les traitements de texte, sur HTML. Son langage devient rapidement populaire, permettant d'effectuer toutes ces modifications esthétiques dans un fichier séparé, ou bien dans une zone du fichier HTML.



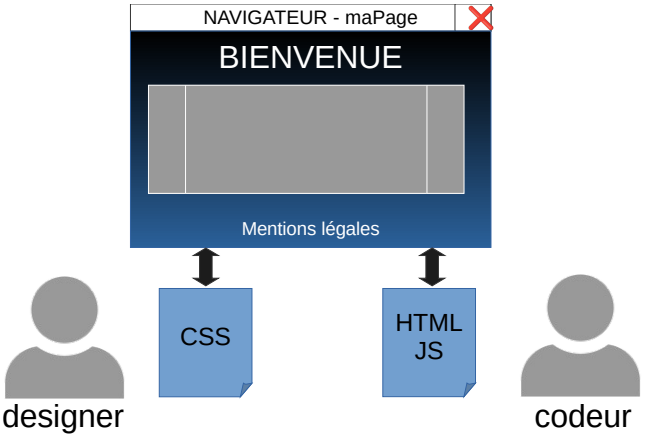
Ce qui est remarquable dans l'histoire de CSS, c'est qu'il y a eu deux camps, dans les années 1995 : ceux qui pensaient qu'il fallait améliorer les balises HTML, et ceux qui préféraient une séparation avec l'usage de feuilles de styles.

Microsoft a notamment soutenu fortement les feuilles de styles, grâce à son navigateur Internet Explorer.

En 2020, il n'existe pas de version CSS4 et ce sont plutôt des spécifications sur des modules, avec un manque de versionning qui fait encore débat au sein du W3C.

A.2 Présentation

CSS est un langage développé pour faciliter la vie des développeurs et des designers



Le fort intérêt de CSS est de pouvoir maîtriser la mise en page des fichiers HTML, sans devoir modifier ceux-ci.

En séparant le fond et la forme, il est plus facile de moderniser un site existant, ou bien de l'adapter à la charte graphique de l'entreprise.

CSS signifie **Cascaded Style Sheet,** ce que nous pouvons traduire par Fichier de style en cascade.

Ce cours aborde les bases en CSS :

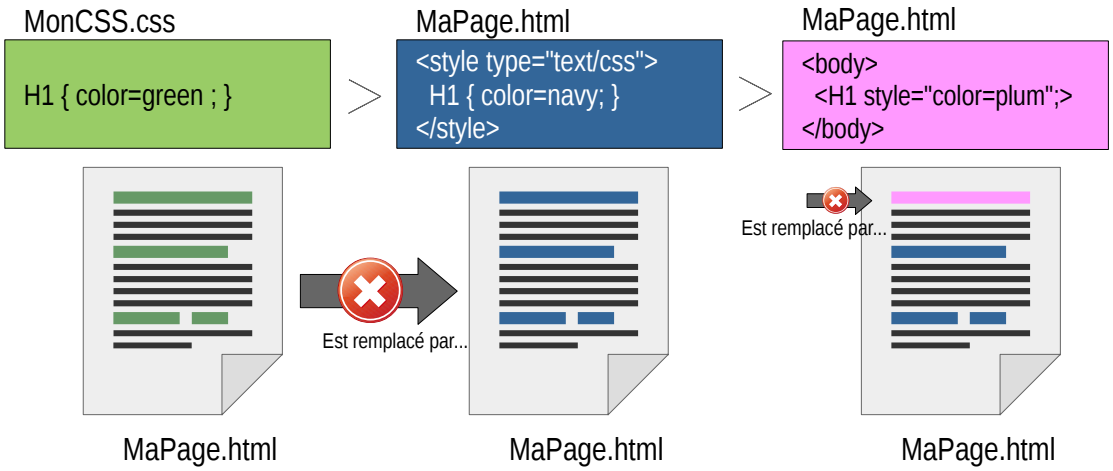
- Emplacements et hiérarchie des instructions CSS (inline, entête, fichier)
- Syntaxe de CSS
- Usage des conteneurs HTML
- Les classes et les ID
- Les unités de mesures
- Les propriétés CSS

B Fonctionnement CSS

B.1 Emplacement et hiérarchie des instructions CSS

Les instructions CSS n'auront pas la même priorité, selon leur emplacement dans le code, d'où l'idée de "cascade" dans l'abréviation CSS.

L'exemple ci-dessous montre que les instructions les plus "proches" des balises écrasent les directives (les ordres) des instructions CSS précédentes.



Ainsi, l'ordre de coloriser les balises `<H1>` en vert dans le fichier monCSS.css sera écrasé par les instructions CSS présentes dans le bloc `<style>` de la page HTML, qui sera à son tour écrasé par les attributs de style dans la balise `<H1 style="">` elle-même.

Dans un fichier HTML, la balise qui appelle le fichier CSS doit se trouver dans l'entête `head` :

Méthode 1

```
<head>
    <link rel = "stylesheet" type = "text/css" href = "/css/style.css">
</head>
```

Cette méthode est la plus rapide, car elle permet le téléchargement en parallèle.

Elle a également un autre avantage, celui de permettre au navigateur de choisir le style de la feuille s'il s'agit d'un écran ou d'une impression :

```
<link rel="stylesheet" href="habillage.css" type="text/css" media="screen" />
<link rel="stylesheet" href="impression.css" type="text/css" media="print" />
```

La seconde méthode existe depuis CSS2. Théoriquement moins performante, elle permet de gérer l'ordre de chargement (pas de parallélisation). Elle peut aussi optimiser le chargement de multiples styles et dans ce cas aboutir à de meilleures performances.

Méthode 2

```
<head>
     <style>
          @import url('/css/style.css')
     </style>
</head>
```

En conclusion, les deux méthodes ne posent aucun problème pour des styles CSS simples et légers. La question se pose pour des structures plus complexes et dans ce cas, il sera nécessaire de faire des "benchmarks".
Pour aller plus loin : https://medium.com/@theaminul/the-pros-and-cons-of-using-import-in-css-a-comprehensive-guide-1820e05e6218

B.2 Syntaxe CSS

CSS se rapproche davantage des langages de programmation que de HTML. Pour cela, on choisit sur quel élément le code CSS doit s'appliquer et on ouvre un bloc avec les symboles `{` et `}` (accolades).

Chaque instruction doit se terminer avec un symbole `;` (point-virgule).

On peut ajouter un bloc de commentaire, avec les symboles `/*` et `*/` pour terminer le bloc.

À l'intérieur des blocs, on indente le contenu, pour clarifier la lecture.

B.2.1 Exemple CSS de base

Voici un premier exemple de code CSS, intégré à l'intérieur d'une page HTML :

premier exemple CSS.css

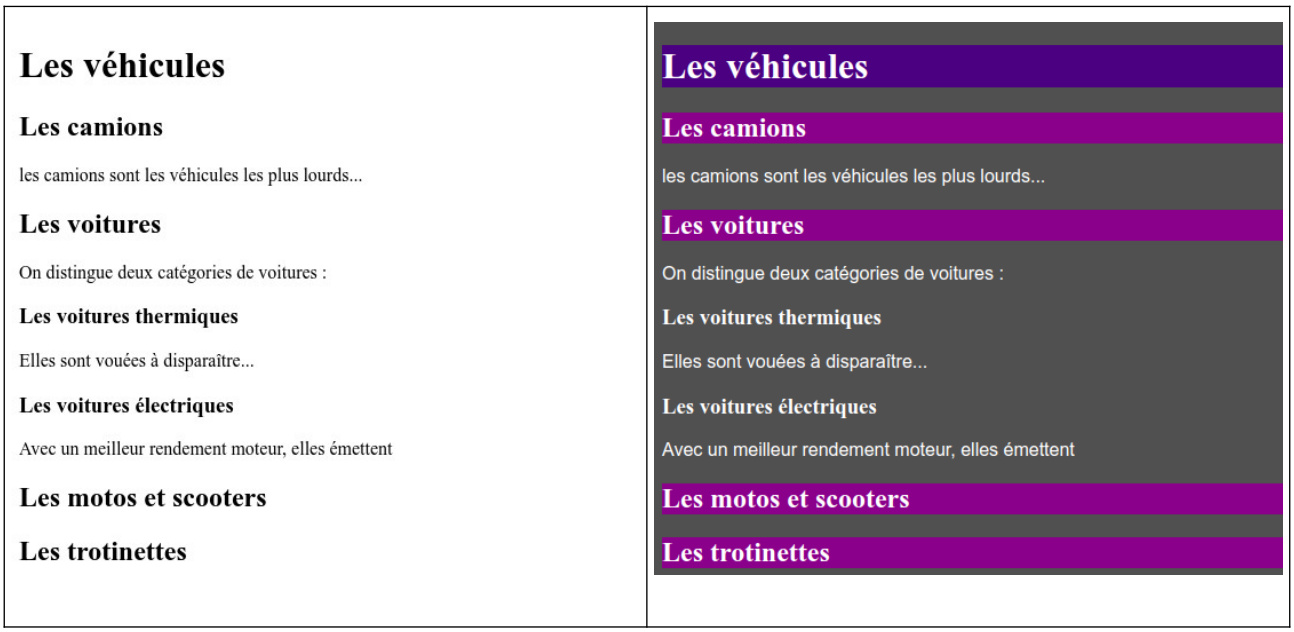
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Premier exemple CSS</title>

    <style>
        body {
            background-color: #505050;
            color: white;
        }
        h1 { background-color: indigo; }
        h2 { background-color:darkmagenta;  }
        p  { font-family: Arial, Helvetica, sans-serif; }
    </style>

</head>
<body>
    <h1>Les véhicules</h1>
    <h2>Les camions</h2>
    <p> les camions sont les véhicules les plus lourds...</p>
    <h2>Les voitures</h2>
    <p>On distingue deux catégories de voitures :</p>
```

```
    <h3>Les voitures thermiques</h3>
    <p>Elles sont vouées à disparaître...</p>
    <h3>Les voitures électriques</h3>
    <p>Avec un meilleur rendement moteur, elles émettent</p>
    <h2>Les motos et scooters</h2>
    <h2>Les trotinettes</h2>
</body>
</html>
```

La page est ainsi modifiée :

- la balise `body` verra sa couleur d'arrière-plan passer au gris et la couleur d'écriture en blanc
- les balises `h1` et `h2` auront une couleur d'arrière-plan indigo et magenta foncé
- la balise `p` utilisera une police de type sans-serif (d'abord `Arial`, si inexistant, `Helvetica` et sinon n'importe quelle police du système, en `sans-serif`)

Voici le résultat avec la page HTML sans puis avec la partie CSS :



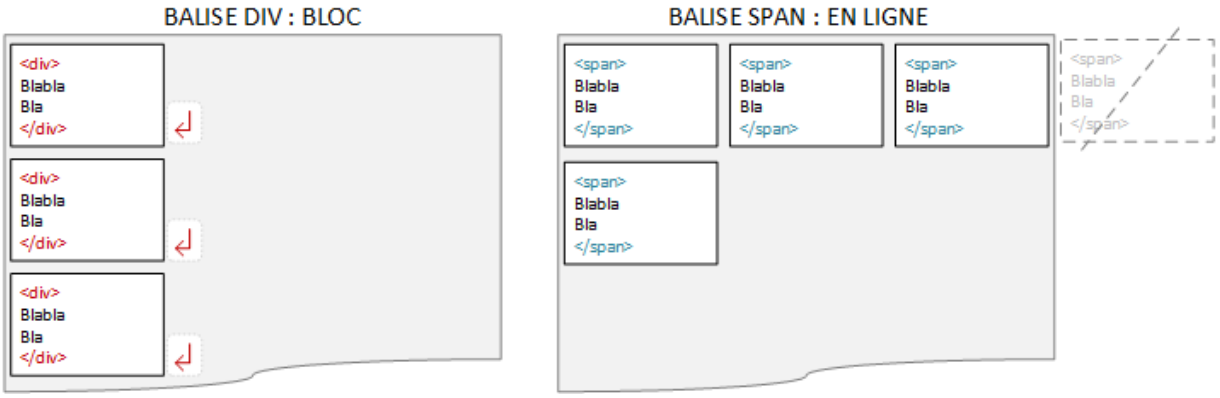
On peut agir sur les différentes **balises** existantes de HTML, mais également sur les **sélecteurs** présents dans le DOM.

B.3 Usage des conteneurs HTML

Le cours HTML explique le fonctionnement des deux balises `<span>` et `<div>`. Ces balises sont des conteneurs et sont encore plus importantes dans les pages modernes, grâce à CSS : en modifiant les styles et attributs de ces balises, il devient possible de placer du code HTML de manière précise sur la page, ou inversement, de laisser le navigateur adapter le contenu à la page.

B.3.1 Conteneurs

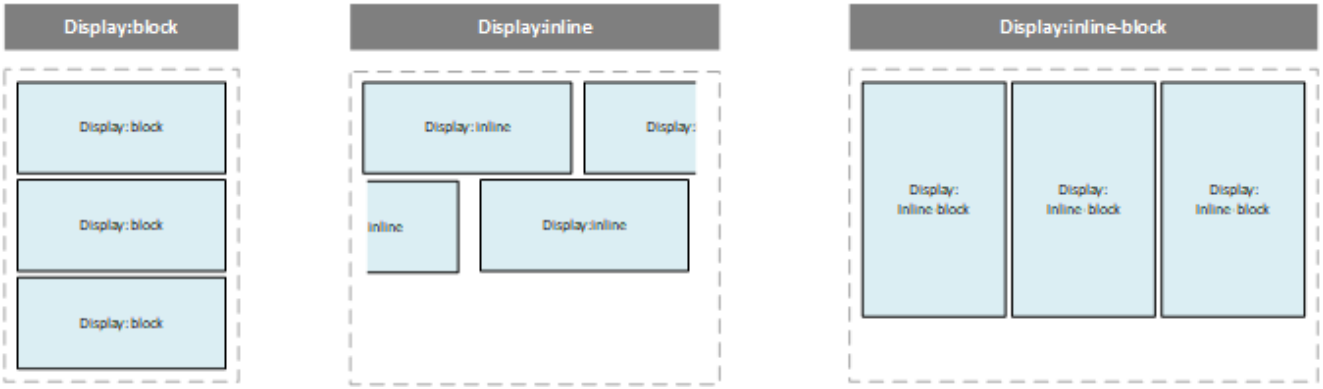
CSS va permettre de gérer le comportement des blocs. Typiquement, la plupart des frameworks CSS proposent des **flashcards**, basés sur l'utilisation de la balise `<div>`. Le comportement de `<div>` ne sera plus en bloc mais en ligne, comme la balise `<span>`.



Exemple de code :

```
div { display: inline-block ; }
```

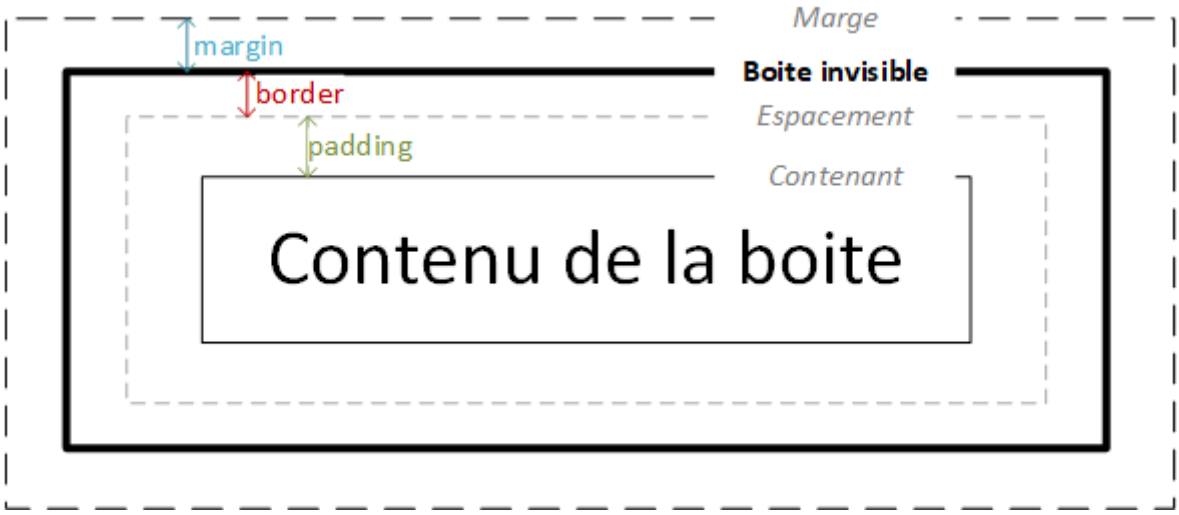
On peut ainsi définir des blocs ayant une hauteur précise, alignés entre eux et capables de combler les vides automatiquement.



B.3.2 Attributs de conteneurs

Les conteneurs (et de manière générale, les balises HTML) ont des propriétés modifiables, comme la couleur d'écriture, la couleur de fond, la taille et le type de police… mais aussi des espacements et des bordures.

On trouve donc fréquemment ce schéma indiquant les propriétés d'espacement entre le texte et la bordure, puis entre la bordure et un prochain élément.



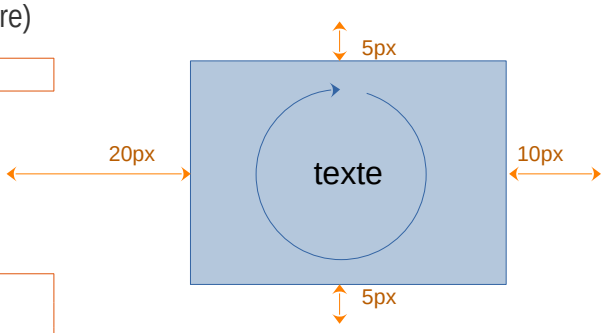
Le **padding** (rembourrage) concerne l'intérieur de la bordure.

Le **marging** (marge) définit l'espacement extérieur à la bordure.

Il y a deux manières de créer les rembourrages et les marges :

Méthode N°1 (débute en haut, tourne dans le sens horaire)

```
p { margin: 5px 10px 5px 20px; }
```



Méthode N°2 (déclaration de chaque marge)

```
p { margin-top: 5px;
    margin-right: 10px;
    margin-bottom: 5px;
    margin-left: 20px;
}
```

B.3.3 Exemple de padding/margin

L'exemple précédent est modifié comme suit :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Premier exemple CSS</title>

    <style>
        body {
            background-color: #505050;
            color: white;
        }
        h1 {
            background-color: indigo;
            padding: 30px;
        }
        h2 {
            background-color:darkmagenta;
            border-radius: 50px;
            padding-left: 10px;
          }
        p  {
            font-family: Arial, Helvetica, sans-serif;
            margin-left: 40px;
        }
    </style>

</head>
<body>
    <h1>Les véhicules</h1>
    <h2>Les camions</h2>
    <p> les camions sont les véhicules les plus lourds...</p>
    <h2>Les voitures</h2>
    <p>On distingue deux catégories de voitures :</p>
    <h3>Les voitures thermiques</h3>
    <p>Elles sont vouées à disparaître...</p>
    <h3>Les voitures électriques</h3>
    <p>Avec un meilleur rendement moteur, elles émettent</p>
    <h2>Les motos et scooters</h2>
    <h2>Les trotinettes</h2>
</body>
</html>
```

La balise `h1` bénéficie d'une marge entre l'écriture et le bord de son cadre de 30 pixels.

La balise `h2` est décalée de 10 pixels uniquement à sa gauche, dans son cadre.

Le cadre de la balise `p` est décalé de 40 pixels du bord gauche.

Remarque : un effet d'arrondi sur le cadre `h2` est ajouté mais sera vu plus tard.


Les véhicules
Les camions
les camions sont les véhicules les plus lourds...
Les voitures
On distingue deux catégories de voitures :
Les voitures thermiques
Elles sont vouées à disparaître...
Les voitures électriques
Avec un meilleur rendement moteur, elles émettent
Les motos et scooters
Les trotinettes

B.4 Les sélecteurs

B.4.1 Sélecteurs par nom (body {}, div {}, ...)

Les **noms de balises** sont utilisés, le problème étant que toutes les balises identiques prendront les mêmes propriétés CSS. Exemple de balises : body, p, div, h1, span, strong…

code CSS

```
body { background-color: #101010; }
```

B.4.2 Sélecteur par classe (.nom {})

Les **classes** sont comme des **étiquettes** qui donnent aux éléments, un comportement commun. Il est possible qu'une balise appartienne à plusieurs classes. CSS y accède grâce au symbole `.` (point).

Par exemple, nous pouvons définir 3 classes :

- une classe **msg** qui définit un cadre noir, avec une bordure de 3 pixels et des coins arrondis
- une classe **alert** dont le fond sera rouge avec une écriture blanche

Le code HTML serait le suivant :

```
<div class="msg alert">Ceci est un message d'alerte</div>
```

code CSS

```
.msg { border-style: solid; border-width: 3px; }
.alert {  color: white; background-color: #FF0000; }
```

Ici, les deux classes se cumulent, affichant un cadre avec un remplissage rouge.

B.4.3 Sélecteur par identifiant (#mot {})

Les **ID sont uniques** : il ne doit pas y avoir d'éléments identiques dans la même page. Les ID (comme identifiants) servent à identifier des éléments d'une page, pour pouvoir interagir de manière spécifique (notamment avec JavaScript). On y accède en CSS avec le symbole `#` (mnémotechnique : ID et DIèse).

Voici un exemple de code :

```
<div id="compteur">0000</div>
```

code CSS

```
#compteur { border-style: solid; border-width: 3px; }
```

B.4.4 Sélecteur par pseudo-classe (:hover {})

Les pseudo-classes représentent des états sur des éléments existants (tel que h1, p, div, table, etc.) :

- `:hover` indique un état survolé pour un élément
- `:link`, `:active` et `:visited` représente l'état d'un lien (respectivement, affichage du lien, affichage lors du survol et affichage après visite)
- `:first-child` indique le premier sous-élément
- `:lang` permet de modifier un comportement en fonction de la langue
- `:focus`, `:read-only`, `:checked`, `:required`, etc. permettent d'appliquer des styles sur des éléments de saisie (formulaires)
- ...

Exemple de code :

```
h1 { background-color: blue; }
h1:hover { background-color: orange; }
```

B.4.5 Sélecteur par pseudo-élément (::before {})

Cette famille est plus rarement utilisée, mais permet de définir des styles en fonctions de règles. Par exemple, le pseudo-élément::first-letter permet de créer une lettrine pour un paragraphe.

```
#demo-first-letter::first-letter {
  color:red; font-weight:bold;
}
```

B.5 Unité de mesure

Pour changer les tailles des bordures ou des polices, CSS permet l'usage de plusieurs unités de mesures. Les unités absolues sont constantes et ne varient pas en fonction du périphérique ou de la taille de l'écran. À l'inverse, les unités relatives expriment un rapport entre un élément et un autre (ratio).

- **Unités absolues : px** (pixel) et **pt** (point) sont des valeurs de taille absolue. Le pixel est l'unité sur les écrans alors que le point est l'unité des imprimantes. Si la maîtrise de l'affichage sur un ordinateur fixe reste un objectif intéressant, l'adaptation sur les terminaux mobiles sera plus difficile.
- **Unités relatives : %, vw** et **em** (et **rem** pour 'root em') sont des unités relatives.
  100 % signifie généralement 12pt mais dépend des paramètres du navigateur. Pour une personne visuellement déficiente, la taille par défaut sera plus élevée et 100 % signifiera alors peut-être 18pt !
  Em signifie "ième", dont 1em = 12pt, 2em = 24pt et .5em sera 6pt.
  Enfin, vw est une unité en centième de la largeur de la page.

px	pt	em	rem	%	vw	vh
Unité absolue	Unité absolue	Unité relative	Unité relative	Unité relative	Unité relative	Unité relative
pixel	Point (écran)	Relative à l'élément conteneur parent	Relative à l'élément conteneur root	Pourcentage de l'élément parent	Relative à la largeur du viewport	Relative à la hauteur du viewport

Il est fortement recommandé d'utiliser les unités relatives, qui facilitent la mise à l'échelle de la page et limitent les problèmes de changement d'appareil (passer d'un téléphone à un écran d'ordinateur).

CSS n'acceptera pas une commande sans unité.

C Propriétés CSS

Il existe des propriétés statiques (qui ne changent que lorsqu'on modifie leurs valeurs) et des propriétés dynamiques (animations).

C.1 Propriétés statiques

C.1.1 Couleurs et traits

Le mot clé pour déterminer la couleur d'un texte ou d'un trait est `color`, tandis que le mot-clé pour la couleur d'arrière-plan est `background-color`.

CSS accepte 3 formats pour affecter une couleur : couleur nommée (`red`, `blue`... `chartreuse`), code hexadécimal (`#RRGGBB`, chaque valeur allant de 00 à FF) et avec la fonction `rgb(rouge, vert, bleu)`. Il existe également une fonction `rgba(rouge, vert, bleu, transparence)` qui ajoute la gestion de la transparence.

Vous trouverez une liste complète des noms de couleurs sur https://www.quackit.com/css/css_color_codes.cfm

Color Name	Hex Code RGB	Decimal Code RGB
Reds		
IndianRed	CD5C5C	205,92,92
LightCoral	F08080	240,128,128
Salmon	FA8072	250,128,114
DarkSalmon	E9967A	233,150,122
LightSalmon	FFA07A	255,160,122
Crimson	DC143C	220,20,60
Red	FF0000	255,0,0
FireBrick	B22222	178,34,34

Color Name	Hex Code RGB	Decimal Code RGB
Greens		
GreenYellow	ADFF2F	173,255,47
Chartreuse	7FFF00	127,255,0
LawnGreen	7CFC00	124,252,0
Lime	00FF00	0,255,0
LimeGreen	32CD32	50,205,50
PaleGreen	98FB98	152,251,152

Color Name	Hex Code RGB	Decimal Code RGB
Browns		
Cornsilk	FFF8DC	255,248,220
BlanchedAlmond	FFEBCD	255,235,205
Bisque	FFE4C4	255,228,196
NavajoWhite	FFDEAD	255,222,173
Wheat	F5DEB3	245,222,179
BurlyWood	DEB887	222,184,135
Tan	D2B48C	210,180,140

extrait des couleurs

Enfin, il existe une propriété `opacity` à laquelle on affecte une valeur décimale allant de 0 (transparent) à 1 (totalement opaque). Une valeur de 0.5 signifie donc une transparence de 50 %. La propriété opacity s'applique à tous les éléments que la balise contient (texte et fond, images, etc.).

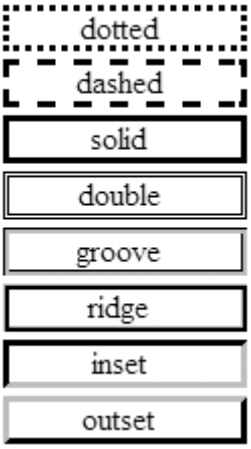
Exemple d'utilisation :

```
h1 { color: white ; background-color: #FF0000; opacity: 0.3; }
p { color:  chartreuse ; }
div {background-color: rgb(127, 127, 0); }
h2 { color: white; background-color: rgba(255, 127,0, 0.3); }
```

Concernant les traits, on utilise les commandes `border-color`, `border-style`, `border-width`, `border-radius`, etc.

Elles agissent sur la couleur, le type de trait, l'épaisseur du trait et l'arrondissement des angles.

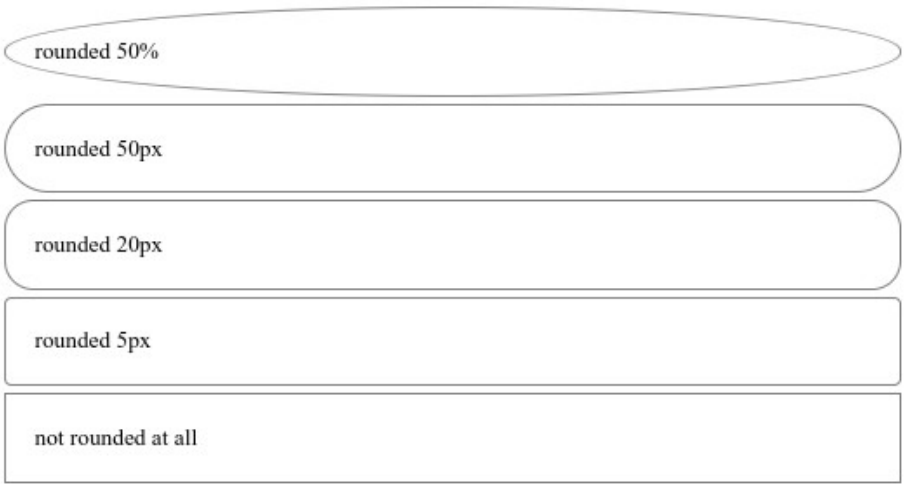
Les paramètres pour border-style sont : `none`, `dotted`, `dashed`, `solid`, `double`, `groove` et `inset`, `outset`, `ridge`. Comme pour les marges et bourrages, il est possible de définir chaque bord différemment.



Exemple d'utilisation :

```
h1 { border-color: crimson; border-style: none none none double; border-radius: 10px}
```

Enfin, l'aspect des cadres peut-être modernisé (adouci) par l'usage de la propriété `border-radius` ou `radius` si la propriété commence par `border :` !



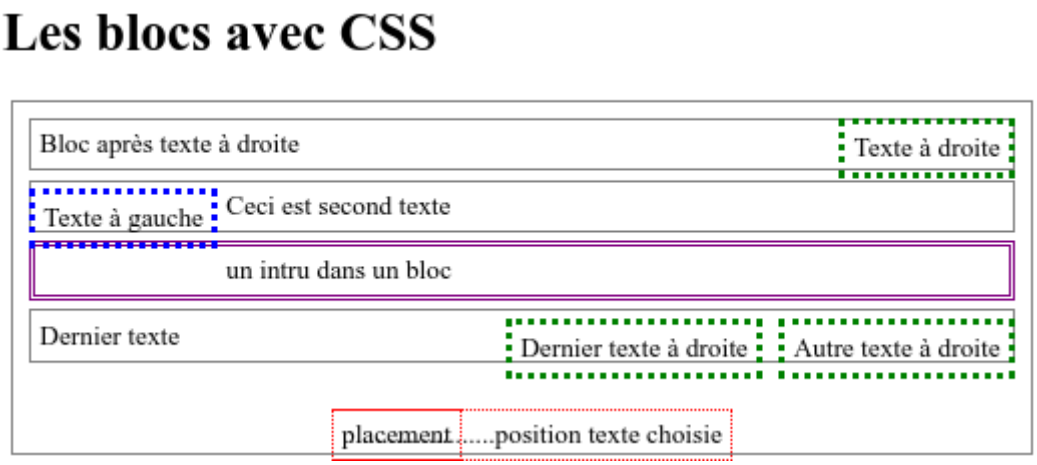
Comme pour border-style, il est possible de n'arrondir qu'un, deux ou trois bords.

```
h1 { border-color: crimson; border-radius: 20px 0px 20px 5px;}
```

C.1.2 Les positions et tailles

C'est la partie la plus étrange de CSS : il existe de nombreuses propriétés, mais elles ne sont pas toutes compatibles, ou plutôt, le comportement peut sembler erratique lorsqu'il y a trop de définitions.

Voici un exemple ne contenant que des blocs :



Les codes CSS et HTML sont les suivants.

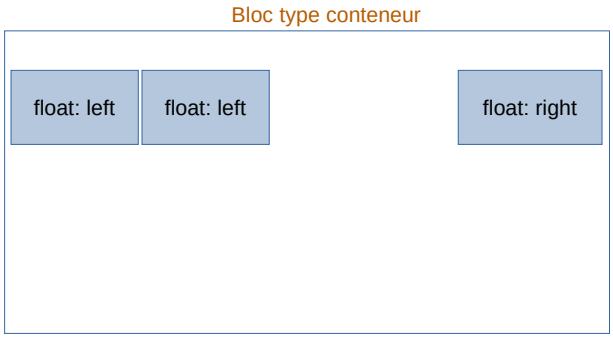
blocs_positions.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Blocs et positions</title>
    <style>
        div { border: solid 1px grey; margin: 5px; padding: 5px;}
        .container { width: 600px; height: 200px; }
        .right { float:right; border: dotted green;}
        .left { float:left; border: dotted blue;}
        .none { float: none; border: double purple;}
        .position { position: absolute; top: 190px; left: 200px; border: dotted 1px red;}
    </style>
</head>
<body>
    <h1>Les blocs avec CSS</h1>
    <div class="position">placement</div>
    <div class="container">
        <div class="right"> Texte à droite </div>
        <div> Bloc après texte à droite </div>
        <div class="left"> Texte à gauche </div>
        <div> Ceci est second texte </div>
        <div class="none"> un intru dans un bloc </div>
        <div class="right"> Autre texte à droite </div>
        <div class="right"> Dernier texte à droite </div>
        <div> Dernier texte </div>
        <div class="position">.....position texte choisie</div>
    </div>
</body>
</html>
```

.

C.1.2.a propriété float

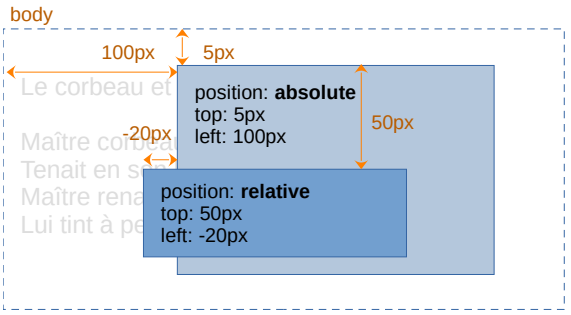
La définition d'un **conteneur** permet de placer les différents blocs à l'intérieur de celui-ci : c'est une étape importante pour la propriété `float` qui s'applique sur les enfants d'un objet ("à droite de..." ou "à gauche de..."). S'il y a plusieurs balises `float` à la suite, elles restent sur la même ligne.

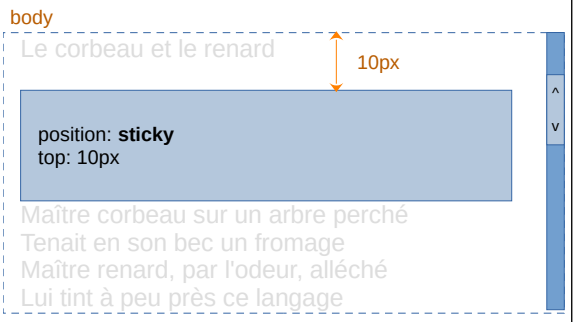


C.1.2.b Propriété position

À l'inverse, la propriété `position` (avec la valeur `absolute`) s'applique depuis le haut de la page (c'est la raison pour laquelle elles se chevauchent dans l'exemple précédent). Pour utiliser la propriété position, on utilise les balises `top`, `left`, `bottom` et `right` pour le placement.

Les différentes valeurs pour la propriété `position` sont :

Valeur	fonctionnement	exemple
static	Fonctionnement par défaut	
fixed	Ne change pas de place, même en cas de scrolling.	
absolute	Position absolue par rapport à la balise <body> Le bord de l'écran en haut à gauche est la référence.	
relative	Position relative au conteneur (une division, un bloc, etc.). Il est possible de fournir des valeurs positives ou négatives.	

sticky	S'affiche à l'endroit de sa déclaration (en prenant sa propre place) mais en cas de scrolling, ne peut sortir de l'écran : exemple, s'approcher du bord supérieur (distance 'top')	

C.1.2.c Propriétés de taille (width et height)

pour déterminer la taille fixe d'un élément, il suffit d'utiliser les propriétés `width` (largeur) et `height` (hauteur).

C.1.2.d Propriété de couche (z-index)

Les éléments peuvent se recouvrir (notamment avec l'utilisation des propriétés `float` et `position`. Il est possible d'ordonner ces calques ou couches (en anglais, 'layer') avec la propriété `z-index` suivi d'un nombre : plus le nombre est grand, plus il est devant.

```
<html>
   <head>
   </head>
   <body>
      <div style = "background-color:red;
         width:300px;
         height:100px;
         position:relative;
         top:10px;
         left:80px;
         z-index:2;">
      </div>

      <div style = "background-color:yellow;
         width:300px;
         height:100px;
         position:relative;
         top:-60px;
         left:35px;
         z-index:1;">
      </div>

      <div style = "background-color:green;
         width:300px;
         height:100px;
         position:relative;
         top:-220px;
         left:120px;
         z-index:3;">
      </div>
   </body>
</html>
```

C.1.2.e Centrer un bloc

Il existe plusieurs solutions, mais la difficulté est de trouver le milieu du bloc de référence (body ou div).

```
.container { position:relative; }
#blocCentre {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%)
}
```

Nous verrons dans le cours CSS avancé, qu'il existe d'autres moyens plus pratiques.

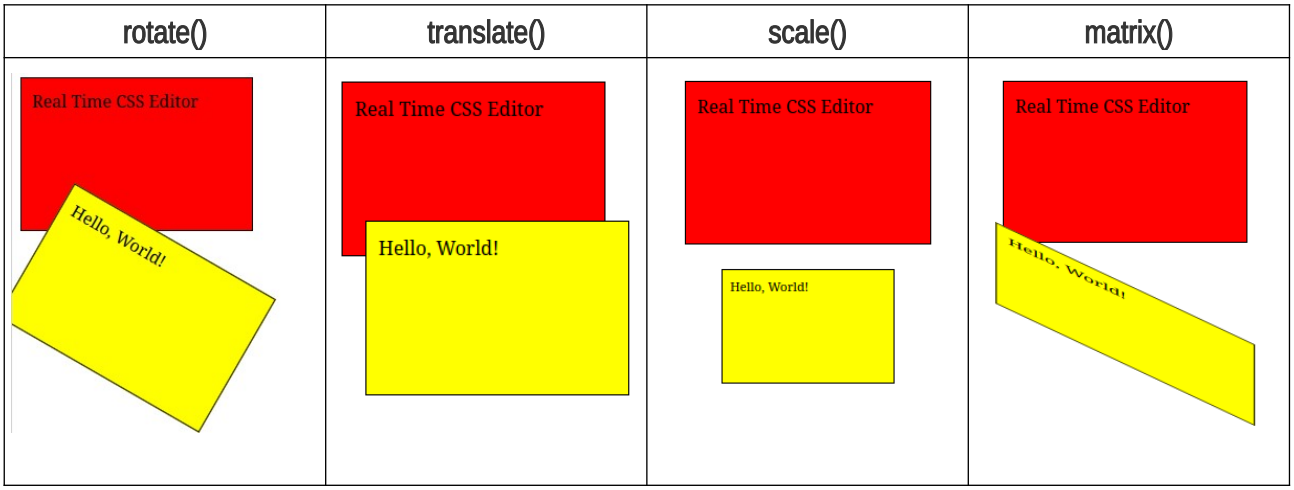
C.2 Propriétés de transformation

Cette section est probablement la plus amusante en CSS : parmi ses propriétés, CSS dispose d'un arsenal de possibilités pour déplacer, tourner et modifier la taille des éléments d'une page. Pour cela, il faut invoquer la propriété `transform` et indiquer l'action à exécuter. À l'intérieur, on appelle une fonction avec des paramètres.

fonction	description
`rotate(angle)`	Tourne l'élément de X degré ou X turn
`translate(x, y)`	Déplace l'élément de X à gauche, Y en bas
`scale(ratioX, ratioY)`	Redimensionne l'élément
`matrix(a, b, c, d, tx, ty)`	
`Matrix3d(...)`	

La propriété transform accepte plusieurs actions sur la même ligne, la commande suivante fonctionne :

```
transform: rotate(30deg) translate(0px, -110px) scale(0.5);
```



Dans les anciens codes on trouvera parfois une version personnalisée : *ms-transform* pour Microsoft, *-moz-transform* pour Mozilla (Firefox), *-webkit-transform* pour Apple (Safari) et Chrome (Google) et *-o-transform* pour Opera et Vivaldi.

C.3 Propriété d'animation

Cette section est la dernière du cours CSS de base et pourtant, elle apporte un dynamisme particulier à CSS. La propriété est `animation` et accepte plusieurs paramètres.

- `Animation-delay:250ms` ; détermine le début de l'animation (0s signifie un lancement immédiat). Une valeur négative interagit comme si l'animation était lancée depuis le temps choisi. Seule l'unité en seconde (s) ou milliseconde (ms) est acceptée.
- `Animation-direction:normal` ; détermine le sens de l'animation. Les mots clés sont `normal`, `reverse`, `alternate` et `alternate-reverse`.

C.3.1 Exemple d'animation

Dans le code d'exemple initial, remplacez toute la partie CSS par celle-ci :

```
<style>
    body {
        background-color: #505050;
        color: white;
    }
    h1 {
        background-color: indigo;
        padding: 30px;
    }
    h2 {
        background-color:darkmagenta;
        border-radius: 50px;
        padding-left: 10px;
        width: 300px;
        animation: rotation 2s linear;
        transform: rotate(-3deg); /* la rotation reste active à la fin de l'animation */
    }
    @keyframes rotation {
        from {
            transform: rotate(0deg);
        }
        to {
            transform: rotate(-3deg);
        }
    }
    p  {
        font-family: Arial, Helvetica, sans-serif;
        margin-left: 40px;
    }
</style>
```

La balise h2 se voit affecter l'animation qui porte le nom `rotation`. À la fin de l'animation, on spécifie la position finale. Commentez successivement les lignes dans la section h2 pour voir la différence.


Les véhicules
Les camions
les camions sont les véhicules les plus lourds...
Les voitures
On distingue deux catégories de voitures :
Les voitures thermiques
Elles sont vouées à disparaître...
Les voitures électriques
Avec un meilleur rendement moteur, elles émettent
Les motos et scooters
Les trotinettes

⚠️ EN CONSTRUCTION

D Annexes

D.1 Sources

Historique CSS

http://cerig.pagora.grenoble-inp.fr/dossier/feuilles-de-style/page01.htm

Transformation

https://developer.mozilla.org/fr/docs/Web/CSS/transform

Matrix

https://dev.opera.com/articles/understanding-the-css-transforms-matrix/

Exemples réalisés avec

https://www.tutorialspoint.com/online_css_editor.php

Animation

https://developer.mozilla.org/fr/docs/Web/CSS/animation

CheatSheet

https://quickref.me/css3

Un encart de base

Attention, il est important de vérifier blabla que blabla. Cela peut arriver lorsque blabla ou bien que truc active le machin et dépasse la limite du bidule. Il est donc important de ne pas laisser les choses dériver sans un contrôle des valeurs sinon blabla.

Il reste cependant la possibilité d'écrire en dessous du nombre de lignes.

Un encart d'attention + puce attention

Attention, il est important de vérifier blabla que blabla. Cela peut arriver lorsque blabla ou bien que truc active le machin et dépasse la limite du bidule. Il est donc important de ne pas laisser les choses dériver sans un contrôle des valeurs sinon blabla.
Il reste cependant la possibilité d'écrire en dessous du nombre de lignes.

Un encart stop + puce stop

Attention, il est important de vérifier blabla que blabla. Cela peut arriver lorsque blabla ou bien que truc active le machin et dépasse la limite du bidule. Il est donc important de ne pas laisser les choses dériver sans un contrôle des valeurs sinon blabla.
Il reste cependant la possibilité d'écrire en dessous du nombre de lignes.

Un encart de réflexion

Attention, il est important de vérifier blabla que blabla. Cela peut arriver lorsque blabla ou bien que truc active le machin et dépasse la limite du bidule. Il est donc important de ne pas laisser les choses dériver sans un contrôle des valeurs sinon blabla.
Il reste cependant la possibilité d'écrire en dessous du nombre de lignes.

Un encart de note

Attention, il est important de vérifier blabla que blabla. Cela peut arriver lorsque blabla ou bien que truc active le machin et dépasse la limite du bidule. Il est donc important de ne pas laisser les choses dériver sans un contrôle des valeurs sinon blabla.

Il reste cependant la possibilité d'écrire en dessous du nombre de lignes.