

CSS3

Premiers pas

Support de cours Version 1.3

Sommaire

Sommaire	2
1 Introduction.....	5
1.1 Définition	5
1.2 Historique	6
1.3 Compatibilité des navigateurs.....	7
1.4 Pour les lecteurs pressés	9
2 Introduction à CSS3	10
2.1 Concrètement, CSS, c'est quoi ?.....	10
2.2 Quels outils pour créer du CSS ?	12
2.2.1 Editeurs de code et IDE	12
2.2.1 Editeurs en ligne	12
2.3 Quelques principes de base	13
2.3.1 Généralités	13
2.3.2 Modes d'inclusion du CSS.....	14
2.3.3 Théorie et pratique.....	18
2.3.4 Une petite coloration ? ☺	19
2.3.5 Introduction aux sélecteurs.....	20
2.4 Le coin du webdesigner	23
2.4.1 Styler du texte	23
2.4.2 Disposer, afficher, cacher	26
2.4.3 Styler une liste et en faire un menu	34
2.4.4 Sélecteurs CSS	36
2.4.5 Pseudo-classes CSS	43
2.4.6 Pseudo-classes CSS3	46
2.4.7 Mediaqueries et Responsive design.....	50
2.4.8 Effets de dégradés	55
2.4.9 Positionnement et Z-index	58
2.5 Techniques CSS3 avancées	63
2.5.1 Inclusion d'animation Javascript en fond de page	63

2.5.2 Effets de transition	67
2.5.3 Effets de transformation	70
2.5.4 Effets d'animation	76
2.5.5 Flexbox.....	80
2.5.5 Grid	82
2.5.6 La fonction calc()	85
3 Conclusion	86
4 Sites	87
5 Annexe.....	88
5.1 Vidage de cache dans Chrome	88
5.2 Simulateur de smartphone dans Chrome	90
5.3 Raccourcis clavier pour Google Chrome	91
5.4 Astuces de webdesigners	92
5.4.1 Ne pas confondre source initial et source « live »	92
5.4.2 CSS réel et calculé.....	93
5.4.3 Qui est le plus fort ?	94
6 Changelog.....	95

Notes de l'auteur :

Je m'appelle Grégory Jarrige.

Je suis développeur professionnel depuis 1991. Après avoir longtemps travaillé sur des gros systèmes et des langages et technos propriétaires, j'ai fait le pari de me former aux technos et langage open source vers 2005-2006. J'ai commencé à développer des applications webs professionnelles à partir de 2007, avant d'en faire mon activité principale à partir de 2010. L'arrivée du HTML5 dans la même période a été pour moi une véritable bénédiction, et surtout un formidable terrain d'expérimentation (avec des API comme Canvas, WebAudio, etc...).

En plus de mon activité de développeur freelance, je suis également formateur - sur des sujets tels que PHP, HTML5, Javascript, SQL – tantôt en entreprise, tantôt dans le cadre de programmes de reconversion (GRETA notamment).

Ce support est une création réalisée en mai 2017 en vue de proposer une introduction rapide à la norme CSS3 pour des personnes débutant en programmation. On trouvera quelques redites dans ce support par rapport au support qui s'intitule « HTML5 et Javascript » (ce dernier support couvre certains des changements intervenus avec CSS3 sans revenir aux bases du CSS). Le présent support se situe dans la continuité du support qui s'intitule « HTML5 – Premiers pas ».

Ce document est disponible en téléchargement libre sur mon compte Github :

<https://github.com/gregja/JSCorner>

Il est publié sous Licence Creative Commons n° 6. Je souhaite que tout le monde puisse en bénéficier, et particulièrement les membres du meetup « Creative Coding Paris », que j'anime avec quelques amis :

<https://www.meetup.com/fr-FR/CreativeCodeParis>

1 Introduction

1.1 Définition

La norme CSS est une norme assez ancienne puisque que ses origines remontent au milieu des années 90. Elaborée par le W3C, la norme CSS a pour objectif de pallier les carences de HTML en matière de rendu des pages webs.

Définition de HTML empruntée à Wikipédia :

Les feuilles de style en cascade, généralement appelées CSS de l'anglais « Cascading Style Sheets », forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

Source : https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade

Pour un historique complet de CSS, je vous invite à lire l'intégralité de la présentation proposée par Wikipédia.



<http://www.w3.org/TR/html5/>

Le W3C (World Wide Web Consortium), est un organisme de normalisation à but non-lucratif, fondé en octobre 1994, chargé de promouvoir la compatibilité des technologies du World Wide Web telles que : HTML5, HTML, XHTML, XML, RDF, SPARQL, CSS, XSL, PNG, SVG et SOAP.

Un grand nombre d'entreprises partenaires participent plus ou moins directement aux activités du W3C. En règle générale, elles mettent à la disposition du W3C – pour une durée limitée - des experts de différents domaines, qui participent à des groupes de travail. On dénombrait 383 entreprises partenaires en 2013, elles sont 461 en avril 2017.

Pour une présentation détaillée du W3C et de ses activités, on recommandera la lecture de la page Wikipédia en anglais (plus à jour) :

https://en.wikipedia.org/wiki/World_Wide_Web_Consortium

1.2 Historique

Si le développement de la norme CSS3 a débuté en 1999, ce n'est qu'à partir de 2007 que les premiers éléments de cette norme ont commencé à être diffusés, et adoptés progressivement par les éditeurs de navigateurs.

En 2017, la situation s'est nettement améliorée, un grand nombre des éléments de la norme CSS3 sont intégrés nativement dans les principaux navigateurs, et le travail des webdesigners en est grandement facilité.

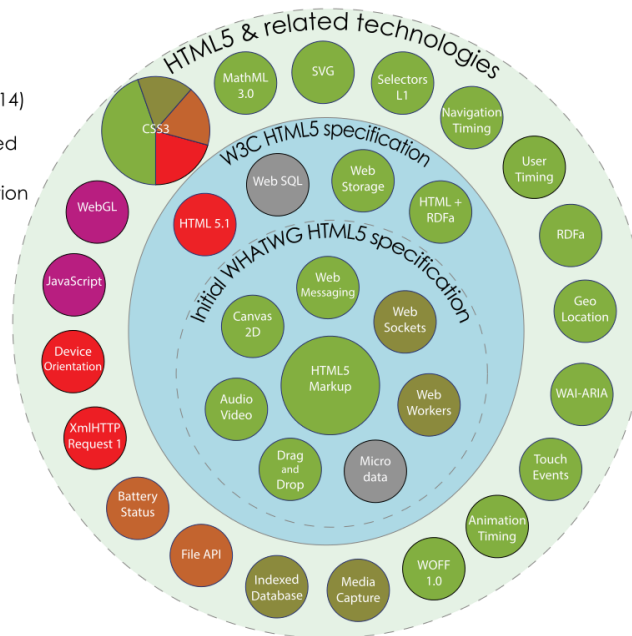
Un autre événement a accéléré l'adoption de CSS3, c'est l'arrivée vers 2010 du framework CSS Bootstrap édité par Twitter. Utilisant massivement les caractéristiques de CSS3, ce framework a rencontré un succès indéniable auprès de la large communauté des développeurs webs, car il propose un code CSS prêt à l'emploi, avec un large panel de composants graphiques au rendu très professionnel. Nous aurons l'occasion d'en reparler.

Ce graphique emprunté à Wikipédia (<https://fr.wikipedia.org/wiki/HTML5>) montre la couverture fonctionnelle que propose la norme HTML5, soit au travers de sa spécification, soit au travers de projets parallèles venus se greffer dans un second temps (comme WebGL par exemple).

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Nous allons nous concentrer dans ce support sur l'étude exclusive du rond central qui s'intitule « HTML5 Markup ». C'est la partie qui définit la syntaxe et le fonctionnement des balises HTML5 de base qui composent l'essentiel des pages sur lesquelles nous « surfons » jour après jour.

1.3 Compatibilité des navigateurs

En ce qui concerne le support de « CSS3 », les navigateurs récents assurent une compatibilité presque parfaite, même si nous verrons qu'il y a quelques « loupés » sur certaines nouvelles balises ou attributs.

Tous les navigateurs ne sont pas égaux face aux nouvelles fonctionnalités proposées par la norme HTML5.

Pour savoir si un navigateur - ou une version de navigateur - supporte une fonctionnalité du HTML5, le site de référence est :

<http://caniuse.com>

Can I use ? [Settings](#)

2 results found

CSS Gradients - CR

Method of defining a linear or radial color gradient as a CSS image.

France 97.29% + 0.11% = 97.4%
 unprefixed: 96.75%
 Global 94.69% + 0.1% = 94.8%
 unprefixed: 93.74%

[Current aligned](#) [Usage relative](#) [Date relative](#) [Show all](#)

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
		45	55						
		48	56	9.1		9.3			

Autre site intéressant proposant des informations complémentaires :

<http://html5please.com>

HTML5 PLEASE

Use the new and shiny responsibly.

Look up HTML5, CSS3, etc features, know if they are ready for use, and if so find out how you should use them – with polyfills, fallbacks or as they are. [tell me more](#) ▶

CSS

1.4 Pour les lecteurs pressés

Si vous lisez ce document, c'est que vous souhaitez apprendre le CSS, que ce soit par envie ou par nécessité.

J'ai essayé en rédigeant ce document, de laisser de côté le superflu, et de me concentrer sur les éléments qui me semblent essentiels, dans cette norme CSS3. Et je l'ai fait avec mon point de vue de développeur d'application « métier ».

Si vous êtes pressé, vous n'avez pas besoin de maîtriser tout dans le détail, en tout cas pas tout de suite. Concentrez-vous en premier lieu sur les propriétés CSS3 usuelles (décrites dans le chapitre 2.3, et 2.4.1 à 2.4.4). Les chapitres 2.4.2 et 2.4.4 me semblent être les plus importants, passez-y du temps, faites beaucoup d'essais pour bien comprendre les principes.

J'insiste beaucoup sur l'importance de la pratique. Vous êtes peut être pressé, mais dites-vous bien que vous ne comprendrez et ne maîtriserez le CSS que par la pratique. Si vous comptez lire ce document sans faire le moindre test, passez votre chemin, allez lire autre chose, ça vous évitera de perdre votre temps. Le CSS n'est pas si compliqué que cela, rassurez-vous, mais beaucoup de notions propres au CSS se comprennent très bien avec un petit peu de pratique, alors qu'elles demeurent complètement abstraites si on ne les a pas mises en œuvre soi-même au moins une fois. C'est particulièrement vrai des chapitres 2.4.2 (avec les problématiques de positionnement) et 2.4.4 (avec les problématiques de sélecteurs CSS, qui sont passionnantes).

Quand vous commencerez à vous sentir à l'aise avec le CSS, vous pourrez lire les chapitres laissés de côté lors de votre première lecture. Les mediaqueries, les effets de dégradé, les effets de transformation (dans le chapitre 2.5), sont quelques unes des nombreuses pépites que recèle le CSS. Pensez aussi à jeter un coup d'œil à l'annexe, qui contient beaucoup d'astuces que j'ai collectées au fil de l'eau.

Je sais par avance que vous allez faire de belles découvertes, car le CSS3, c'est « cool et fun » 😊

2 Introduction à CSS3

2.1 Concrètement, CSS, c'est quoi ?

CSS = Cascading Style Sheet

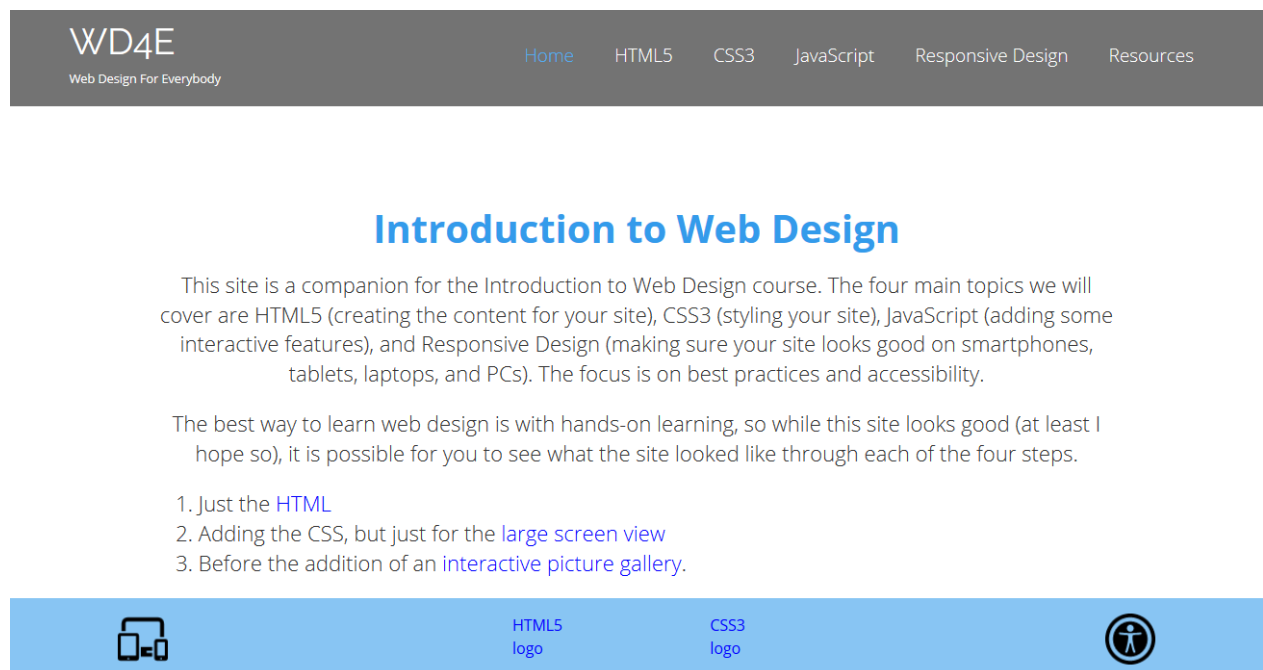
... que l'on peut traduire en « Feuilles de style en cascade. »

Les feuilles de style CSS permettent d'améliorer le rendu des pages HTML de manière très significative.

Si l'on prend l'exemple du site suivant :

<http://www.intro-webdesign.com/>

Le rendu de la page avec son CSS est le suivant :



Si l'on retire à cette même page son code CSS, on obtient le rendu suivant :



... C'est très moche, pas vrai ?

Regardons d'un peu plus près un des éléments de la page, par exemple le menu :

Avec CSS	Sans CSS
<p>The menu is styled with a dark background and white text. The links are: Home, HTML5, CSS3, JavaScript, Responsive Design, and Resources.</p>	<p>The menu is a simple list of links: Home, HTML5, CSS3, JavaScript, Responsive Design, and Resources.</p>

Sans le CSS, on constate que le menu est une simple liste « ul » comme celle que vous avez appris à réaliser pendant le cours HTML. Grâce à un peu de code CSS, cette liste assez moche est devenue un beau menu, avec une disposition des informations très différente.

2.2 Quels outils pour créer du CSS ?

2.2.1 Editeurs de code et IDE

La plupart des éditeurs de code présentés dans le cours HTML sont excellents pour la manipulation du code HTML. Si vous avez une préférence pour l'un de ces outils pour la manipulation du HTML, il y a de grandes chances pour qu'il vous convienne aussi pour la manipulation du CSS. Certains de ces éditeurs proposent des assistants, de l'auto-complétion, et même des modèles (templates) prêts à l'emploi.

2.2.1 Editeurs en ligne

Mais pour débiter en CSS, on n'est pas obligés d'utiliser l'un ou l'autre de ces logiciels. Nous allons utiliser les mêmes outils que ceux utilisés pendant le cours HTML, soit :

- Codepen : <https://codepen.io/>
- Codecircle : <https://live.codecircle.com/>

Comme ils ont déjà été présentés dans le support de cours HTML5, je n'y reviens pas ici.

Nous utiliserons alternativement Codepen et Codecircle, dans la suite de ce cours, histoire de nous continuer à nous familiariser avec ces deux solutions, et avec leurs avantages et inconvénients.

2.3 Quelques principes de base

2.3.1 Généralités

Le même fichier HTML, sur lequel aucun CSS n'est appliqué, va présenter un look différent sur les navigateurs concurrents, tels que Firefox, Chrome, Safari, Opera, Internet Explorer, etc...

Cela est dû au fait que chacun des navigateurs applique à chaque balise HTML un rendu par défaut. Ce rendu par défaut est spécifique à chaque éditeur de navigateur, et on constate des disparités plus ou moins fortes à l'usage. Par exemple une balise « th » n'apparaîtra pas de la même façon dans Firefox et dans Chrome.

CSS permet de « lisser » ces disparités, en forçant les navigateurs à utiliser un style qui sera le même pour tout le monde. Comme les navigateurs ne sont pas tous homogènes en termes de prise en charge du CSS, et en particulier du CSS3, on peut quelquefois constater des disparités au niveau du rendu. Mais ces disparités sont en général moins importantes que si on laisse les pages s'afficher sans CSS.

2.3.2 Modes d'inclusion du CSS

Il existe plusieurs manières d'insérer du CSS dans une page.

La première manière consiste à insérer ce CSS directement dans le code HTML, au moyen de l'attribut « style » :

Voici un exemple :

- Le code

```
<p style="color:lawngreen">Ceci est un paragraphe en couleur  
"lawngreen"</p>
```

```
<p>Ceci est un paragraphe de couleur neutre avec un <span  
style="color:violet">bout de texte en violet</span>.</p>
```

- Le rendu

Ceci est un paragraphe en couleur "lawngreen"

Ceci est un paragraphe de couleur neutre avec un bout de texte en violet.

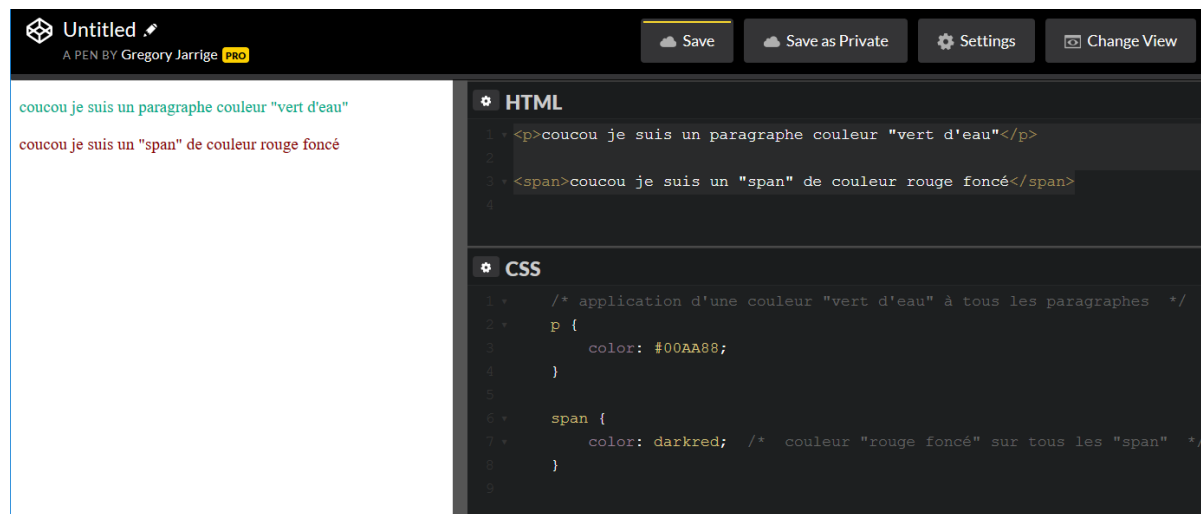
Testez cet exemple dans Codepen ou CodeCircle, amusez-vous à mixer différents « span » et paragraphes, avec des couleurs différentes.

La seconde manière d'insérer du code CSS dans une page, consiste à placer ce code dans une partie délimitée par les balises « style » et « /style », comme dans l'exemple suivant :

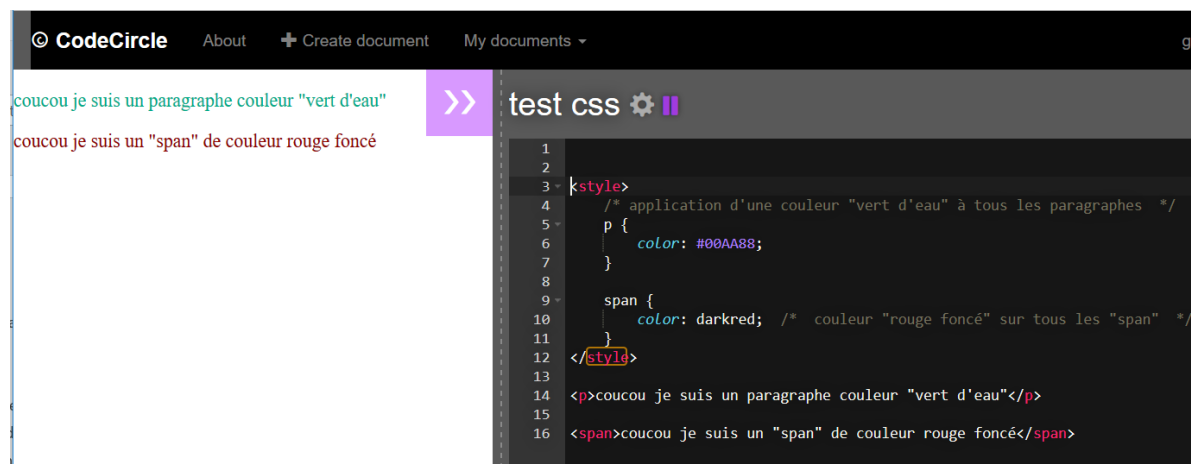
```
<style>
  /* application d'une couleur "vert d'eau" à tous les paragraphes par défaut */
  p {
    color: #00AA88; /* couleur verte d'eau */
  }

  span {
    color: darkred; /* couleur "rouge foncé" sur tous les "span" par défaut */
  }
</style>
```

Dans Codepen, cela donne ceci :



.. et dans CodeCircle :



Dans une page web classique, le CSS se place traditionnellement dans la partie « head » de la page :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    p {
      color: #00AA88;
      background-color: black;
      height: 24px;
      border: double;
    }
    span {
      color: darkred;
    }
  </style>
</head>
<body>

</body>
</html>
```

Lorsque l'on souhaite qu'un même code CSS soit réutilisable sur plusieurs pages, on le place dans un fichier CSS indépendant, et on insère dans la page un lien vers ce fichier :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link href="css/monstyle.css" rel="stylesheet" type="text/css">
</head>
<body>

</body>
</html>
```

Le contenu du fichier CSS « monstyle.css » sera le suivant :

```
p {
  color: #00AA88;
  background-color: black;
```



```
height: 24px;  
border: double;  
}  
span {  
color: darkred;  
}
```

Dans quel ordre le navigateur prend-il en compte le CSS qui est défini dans une page :

1. Rendu par défaut du navigateur
2. CSS externe (en anglais « external style », chargé à partir de fichier CSS indépendant)
3. CSS interne (en anglais « internal style », défini dans la section « head » de la page)
4. CSS en ligne (en anglais « inline style », c'est le CSS placé sur chaque balise HTML via l'attribut « style »)

En CSS, c'est « le dernier qui a parlé qui a raison », donc si nous avons un même sélecteur définit dans 2 endroits différents, le dernier sélecteur défini « écrase » les valeurs définies précédemment. Mais ce comportement peut quelquefois être embêtant. On peut le court-circuiter en ajoutant la valeur « !important » à droite de la valeur d'un attribut.

```
h1{  
color: blue;  
font-family: Arial !important;  
}  
  
h1{  
font-family: Times;  
}
```

2.3.3 Théorie et pratique

Maintenant que nous avons vu le principe d'inclusion, faisons un peu de théorie (pas trop, juste ce qu'il faut 😊).

A gauche, nous avons la théorie, à droite la pratique 😊 :

<pre>selector { property: value; }</pre>	<pre>p { color: green; }</pre>
--	--

Le principe du CSS est le suivant :

- On commence par définir un sélecteur, ici une balise « p »
- Entre les accolades, nous allons définir les différentes propriétés que l'on souhaite appliquer à notre sélecteur (ici on applique la couleur « bleue » à notre sélecteur)

On retrouvera toujours ce principe, tout au long de notre étude de CSS.

A l'intérieur d'un sélecteur, nous pouvons placer plusieurs propriétés séparées par des points virgules. Je vous invite à tester le code suivant. Que fait-il ? Quel rendu obtenez-vous ?

```
p {  
    color: #00AA88; /* couleur verte d'eau */  
    height: 24px;  
    border: dashed;  
}
```

Amusez-vous à modifier la valeur de l'attribut « height ».

Pour l'attribut « border », essayez d'autres valeurs comme « dotted » ou « double ».

Sur la page de W3Schools, vous allez trouver d'autres valeurs possibles pour cet attribut « border », je vous invite à les tester :

https://www.w3schools.com/cssref/pr_border-style.asp

Je vous invite aussi à essayer l'attribut « background-color » avec différentes couleurs :

```
p {  
  color: #00AA88; /* couleur verte d'eau */  
  background-color: black;  
  height: 24px;  
  border: dashed;  
}
```

2.3.4 Une petite coloration ? 😊

Nous disposons de plusieurs manières de définir les couleurs en CSS

- Nom de couleurs en anglais, tels que : « blue », « red », « yellow », etc...
- Valeur hexadécimales : #0000FF, #FF0000, #FFFF00
- Valeurs RGB (Red Green Blue) : rgb(0, 0, 1), rgb(1, 0, 0), rgb(1, 1, 0)
- Valeurs RGBA (RGB + niveau d'opacité allant de 0 pour transparent à 1 pour complètement opaque).

Voici un exemple emprunté à W3Schools :

https://www.w3schools.com/css/css3_colors.asp

The screenshot displays a web page titled 'RGBA colors:'. It features six horizontal bars, each with a color label and a corresponding color swatch: Red (red), Green (green), Blue (blue), Grey (grey), Yellow (yellow), and Cerise (pink). To the right of the color swatches, there are two code blocks. The first block, labeled 'HTML', shows the HTML code for the color swatches:

```
1 <p>RGBA colors:</p>  
2 <p id="p1">Red</p>  
3 <p id="p2">Green</p>  
4 <p id="p3">Blue</p>  
5 <p id="p4">Grey</p>  
6 <p id="p5">Yellow</p>  
7 <p id="p6">Cerise</p>  
8
```

 The second block, labeled 'CSS', shows the CSS code for the color swatches:

```
1 #p1 {background-color:rgba(255,0,0,1);}   
2 #p2 {background-color:rgba(0,255,0,0.3);}   
3 #p3 {background-color:rgba(0,0,255,0.3);}   
4 #p4 {background-color:rgba(192,192,192,0.3);}   
5 #p5 {background-color:rgba(255,255,0,0.3);}   
6 #p6 {background-color:rgba(255,0,255,0.3);}   
7
```

Vous remarquerez ici que notre sélecteur est ici préfixé par un dièse #, mais si vous comparez avec le HTML, vous constatez que la valeur qui suit le dièse correspond à l'une ou l'autre des valeurs définies dans les attributs « id » du code HTML.

2.3.5 Introduction aux sélecteurs

Dans le chapitre d'introduction aux sélecteurs CSS, nous avons étudié l'exemple suivant :

A gauche, nous avons la théorie, à droite la pratique 😊 :

```
selector {  
    property: value;  
}
```

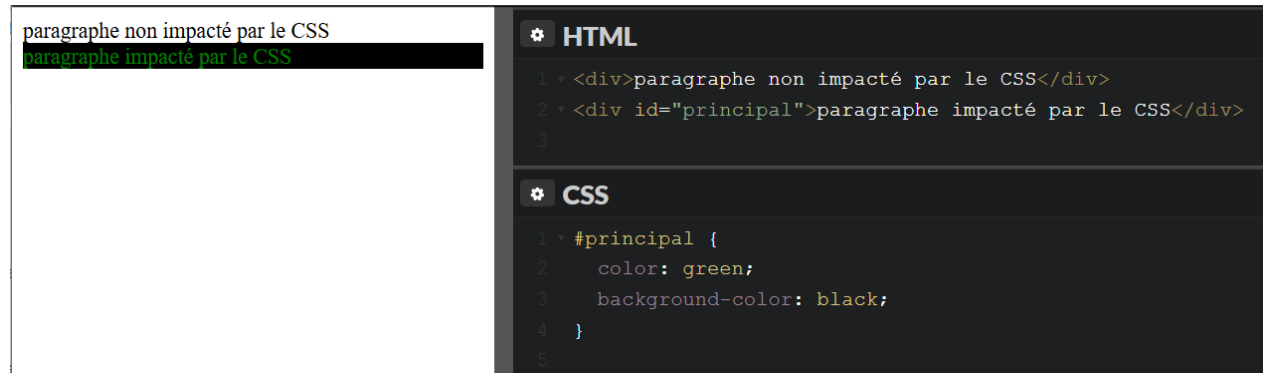
```
p {  
    color: green;  
}
```

L'exemple de sélecteur est très simple, il revient à dire au navigateur que l'on souhaite qu'il sélectionne toutes les balises « p » de la page, et qu'il leur applique une couleur « verte ».

Mais dans la « vraie vie », on n'écrit jamais de choses aussi simplistes.

En réalité, Nous avons besoin de sélecteurs plus... sélectifs.

Exemple 1 : j'ai une balise HTML qui a l'attribut « id » fixé à la valeur « principal ». Je souhaite appliquer à cette seule balise une couleur verte avec un fond noir, je vais dans ce cas utiliser le sélecteur suivant : #principal



Exemple 2 : j'ai plusieurs balises HTML auxquelles j'aimerais appliquer le même style CSS. Mais j'aimerais pouvoir exclure certaines balises de la sélection. Je vais dans ce cas définir une classe CSS. Comment définit-on une classe ?

- Au niveau du HTML, on utilise l'attribut « class » et on lui donne une valeur (par exemple : « monstyle »)
- Au niveau du CSS, on utilise la valeur définie pour la classe concernée, préfixée d'un point (dans notre exemple, ce sera : « .monstyle »)

Le CSS s'écrit de la façon suivante :

```

.monstyle {
  color: green;
  background-color: black;
}

```

... et le HTML de la façon suivante :

```

<div>paragraphe non impacté par la classe "monstyle"</div>
<div class="monstyle">paragraphe impacté par la classe "monstyle"</div>
<div>paragraphe non impacté par la classe "monstyle"</div>
<div class="monstyle">paragraphe impacté par la classe "monstyle"</div>

```

Dans Codepen, cela donne ceci :



En résumé, nous connaissons pour le moment 3 types de sélecteurs :

- Sur les balises HTML elles-même (par exemple la balise « p »)
- Sur l'identifiant des balises (soit l'attribut HTML « id »), dans ce cas le sélecteur commence par # suivi de l'identifiant de l'élément cible
- Sur la classe des balises (soit l'attribut HTML « class »), dans ce cas le sélecteur comment un point (.) suivi de la classe

Il existe d'autres techniques liées aux sélecteurs mais nous les étudierons plus tard.

2.4 Le coin du webdesigner

2.4.1 Styler du texte

Comment le webdesigner s'y prend-il pour donner un style particulier à du texte ?

Eh bien, il dispose de plusieurs options que nous allons explorer, telles que :

- font (family, style, variant, size)
- color et background
- alignment
- line-height

Voici quelques unes des polices parmi les plus couramment utilisées :

Helvetica, Courier, "Courier New",
"Comic Sans MS", *cursive*, Verdana

Attribuer une police à une balise se fait très simplement :

```
h1 {  
  font-family: arial;  
}
```

Mais certaines balises ne sont pas supportées par certains navigateurs, aussi on définit généralement une liste de polices de style relativement proches, à charge pour le navigateur de prendre la première qui lui convient dans cette liste :

```
h1 {  
  font-family: courier, impact, arial;  
}
```

En plus de la police, on peut définir un style, avec l'attribut « font-style » associé à l'une des valeurs suivantes :

- normal
- italic
- oblique

La couleur du texte et la couleur de fond se définissent respectivement avec les attributs « color » et « background-color », nous en avons déjà vu quelques exemples.

L'alignement du texte par rapport à la balise parente (qui peut être une « div », une « td », une « p », etc...) se fait au moyen de l'attribut « text-align », qui peut prendre l'une des valeurs suivantes :

- left
- right
- center
- justify

Il est possible d'ajuster l'espace entre chaque ligne de texte avec l'attribut « line-height ». Sur l'un des exemples précédents, définissez par exemple une hauteur de 100%, 200%, 300%, comparez les résultats :

```
div {
  line-height: 300%
}
```

En termes de taille de police, l'attribut « font-size » nous offre un large panel de possibilités :

- xx-small, x-small, small, smaller
- medium
- larger, large, x-large, xx-large
- taille définie en pixelx (exemple : 12px)
- taille définie en % (exemple : 15%)

La taille des polices de caractères peut être définie dans différentes unités, telles que :

Unité	Description	Type d'unité
cm	centimeters	absolue
mm	millimeters	absolue
in	inches (1in = 96px = 2.54cm)	absolue
px *	pixels (1px = 1/96th of 1in)	absolue
pt	points (1pt = 1/72 of 1in)	absolue
pc	picas (1pc = 12 pt)	absolue
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	relative
ex	Relative to the x-height of the current font (rarely used)	relative

ch	Relative to width of the "0" (zero)	relative
rem	Relative to font-size of the root element	relative
vw	Relative to 1% of the width of the viewport*	relative
vh	Relative to 1% of the height of the viewport*	relative
vmin	Relative to 1% of viewport's* smaller dimension	relative
vmax	Relative to 1% of viewport's* larger dimension	relative
%	Pourcentages	relative

Pour de plus amples précisions sur ce sujet :

<https://developer.mozilla.org/fr/docs/Web/CSS/font-size>

https://www.w3schools.com/cssref/css_units.asp

Petite astuce : on peut jouer sur de nombreuses autres propriétés CSS liées au texte, comme par exemple la couleur et l'ombre (text-shadow) :

```
<p style="font-size:60px;color:lightblue; text-shadow:2px 2px 4px #000000;">
  Aenean nisl libero, venenatis id consequat at,...
</p>
```

Aenean nisl libero, venenatis id consequat
at, interdum eu lorem. Praesent lectus
neque, rhoncus sit amet massa euismod,
aliquet bibendum elit. Vestibulum

On notera l'existence d'un ancien attribut « nowrap », déprécié en HTML5, et qu'il convient de remplacer par une classe de ce type :

```
.nowrap {
  white-space:nowrap;
}
```

Concrètement, « nowrap » signifie qu'aucun saut de ligne ne sera effectué sur des espaces blancs. Si le texte concerné se trouve dans une cellule de tableau, et si ce texte est long, par défaut le navigateur va répartir le texte sur plusieurs lignes à l'intérieur de la cellule, en utilisant

les caractères blancs comme indicateurs de saut de ligne. Avec l'attribut « nowrap », le navigateur ne pourra pas effectuer de saut de ligne, l'intégralité du texte sera affiché sur une seule ligne, et ce quelle que soit sa longueur.

2.4.2 Disposer, afficher, cacher

Tout élément de la page est considéré comme une boîte (« box ») d'un point de vue CSS. L'affichage d'une boîte a un impact direct sur la disposition des boîtes voisines.

On distingue les éléments (ou boîtes) de niveau « bloc » et les éléments « en ligne ».

Un élément de niveau « bloc » commence toujours sur une nouvelle ligne et occupe toute la largeur disponible (en résumé, il s'étend à gauche et à droite aussi loin que possible). Les balises « div », « p », « form », et « h1 » à « h6 » entrent dans cette catégorie.

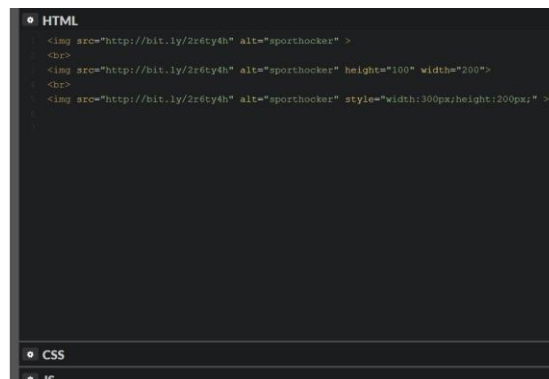
On oppose aux éléments de niveau « bloc » les éléments « en ligne » (inline elements), comme par exemple la balise « span », qui n'occupe que la largeur nécessaire à l'affichage de son contenu et pas plus.

Les balises « img » entrent également dans la catégorie des « inline elements ». L'exemple que nous avons utilisé dans le support de cours HTML va nous permettre d'illustrer ce principe.

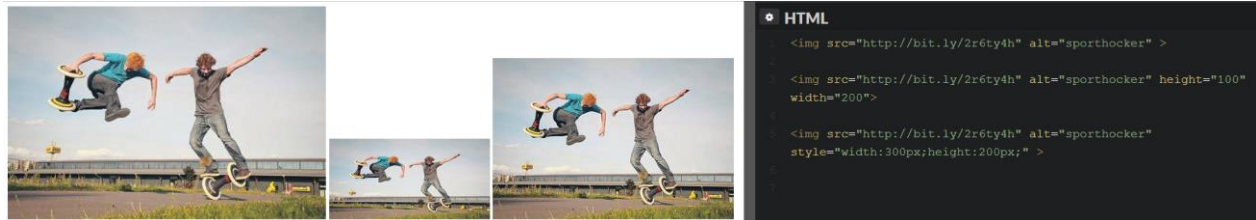
Pour rappel, le code source était le suivant :

```
  
<br>  
  
<br>  

```

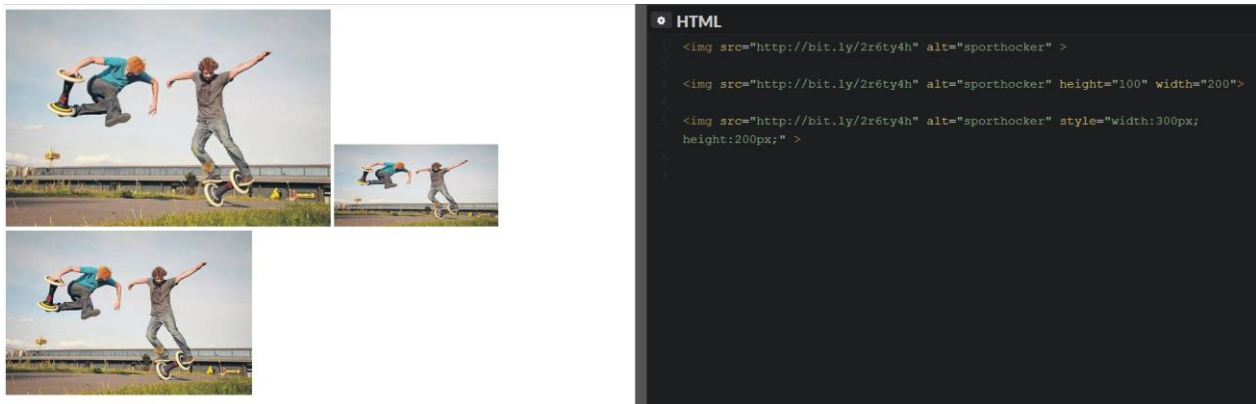


Vous aviez probablement remarqué qu'entre chaque image nous avons placé une balise « br » pour forcer un saut de ligne entre chacune des images. Que se passerait-il si nous supprimions cette balise « br » ? Réponse ci-dessous :



Explication : la balise « img » étant une balise de type « inline », elle occupe la place qui lui est indispensable, et pas plus. Si elle a la place suffisante pour s'insérer dans la ligne en cours, elle s'y place, sinon, elle se place sur la ligne suivante.

Regardons ce qui se passe si je réduis la taille de la fenêtre de gauche :

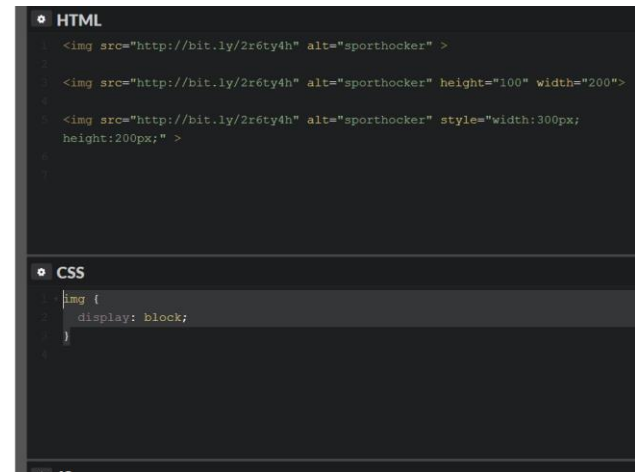


On voit que la 3^{ème} image n'a pas trouvé la place nécessaire pour s'afficher sur la ligne occupée par les 2 images précédentes, aussi elle est allée se placer automatiquement sur la ligne suivante.

Quand je dis que c'est l'image qui est allée se placer à cet endroit, en réalité c'est une vue de l'esprit, c'est même un peu trop simpliste. En réalité, le navigateur fonctionne comme un programme de dessin, et il dessine littéralement chaque élément HTML sur la page. Ce programme de dessin est équipé d'un algorithme très sophistiqué qui calcule la place de chaque élément et détermine la disposition optimale, en fonction des caractéristiques de chaque élément. La notion d'élément de niveau « block » ou de niveau « inline » est primordiale pour cet algorithme de dessin.

Le navigateur considère que les balises « `img` » sont de niveau « `inline` », donc elles sont sensées se placer les unes à la suite des autres, sur une même ligne tant qu'il y a de la place. Mais rien ne nous empêche de modifier cela, et de dire au navigateur que désormais les balises « `img` » sont de niveau « `bloc` ». Cela se fait très simplement en ajoutant le code CSS ci-dessous :

```
img {  
    display: block;  
}
```

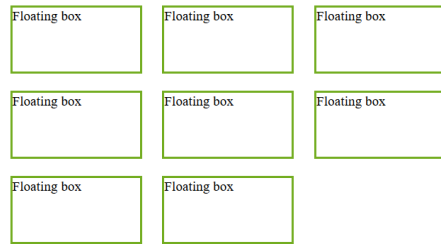


Voilà, plus besoin de balise « `br` », chaque image se trouve placée sous la précédente.

Nous avons vu les niveaux « `inline` » et « `block` », nous devons maintenant étudier le niveau « `inline-block` ». Ce dernier permet à un élément d'occuper une taille (en largeur et hauteur) différente de sa taille réelle, tout en conservant certains avantages du niveau « `inline` ». Un exemple emprunté à W3Schools va nous aider à illustrer notre propos :

https://www.w3schools.com/css/css_inline-block.asp

The New Way - using inline-block



Another box, after the floating boxes...

```

• HTML
1 <h2>The New Way - using inline-block</h2>
2 <div class="floating-box">Floating box</div>
3 <div class="floating-box">Floating box</div>
4 <div class="floating-box">Floating box</div>
5 <div class="floating-box">Floating box</div>
6 <div class="floating-box">Floating box</div>
7 <div class="floating-box">Floating box</div>
8 <div class="floating-box">Floating box</div>
9 <div class="floating-box">Floating box</div>
10 <div class="after-box">Another box, after the floating boxes...</div>

• CSS
1 .floating-box {
2   display: inline-block;
3   width: 150px;
4   height: 75px;
5   margin: 10px;
6   border: 3px solid #73AD21;
7 }
8 .after-box {
9   border: 3px solid red;
10 }

```

Le code CSS est le suivant :

```

.floating-box {
  display: inline-block;
  width: 150px;
  height: 75px;
  margin: 10px;
  border: 3px solid #73AD21;
}
.after-box {
  border: 3px solid red;
}

```

Et le code HTML est le suivant :

```

<h2>The New Way - using inline-block</h2>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="after-box">Another box, after the floating boxes...</div>

```

Amusez-vous à élargir ou rétrécir la zone d’affichage, vous allez constater que ce système est très flexible :



Cette disposition de « div » est relativement courante dans les applications webs. On retrouve souvent ce principe dans des affichages de type « tableaux de bord » pour la gestion, les statistiques, etc...

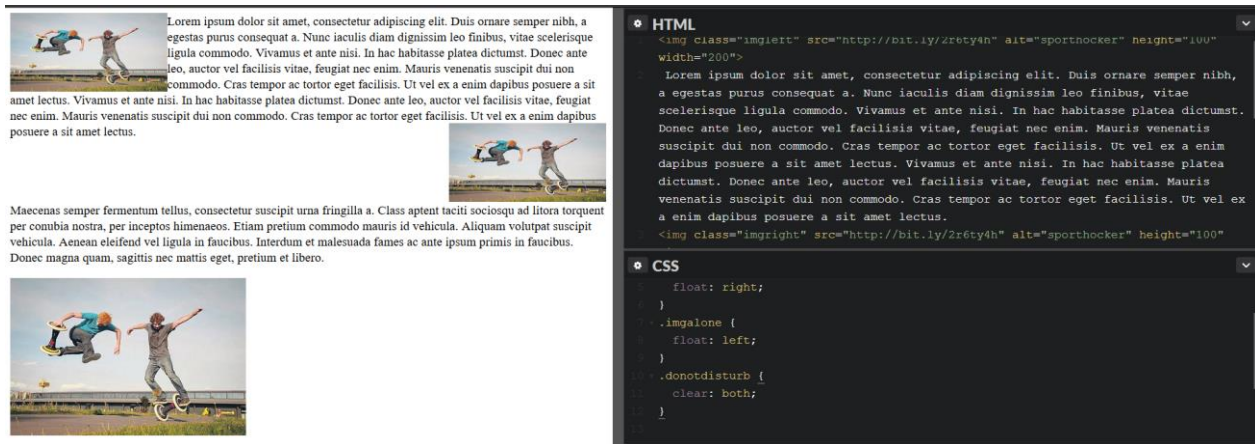
On voit que le positionnement d’éléments sur une page est une opération délicate qui nécessite pas mal de pratique. On peut contourner la difficulté en s’appuyant sur un système de grille (ou « grid » en anglais).

Des projets comme [Bootstrap](#) ou [960.gs](#) proposent leur propre système de grille, mais on peut aussi s’appuyer sur les nouvelles techniques du CSS3 que sont Grid et Flexbox. Ces deux techniques sont présentées dans les chapitres 2.5.5 et 2.5.6.

Nous ne pouvons clore ce chapitre sans parler des attributs « float » et « clear » :

- « float » : permet de définir un élément comme étant flottant à l’intérieur de la page. Cet attribut peut prendre les valeurs « left » et « right »
- « clear » : aucun élément flottant n'est autorisé sur le côté gauche ou droit d'un élément spécifié. Valeurs possibles : « left », « right », « both ».

Voici un nouvel exemple dans lequel nous avons combiné des images et du texte. La première est définie en « float : left », la seconde est en mode « float : right ». La dernière image est en mode « float : left », mais elle ne peut se combiner avec le paragraphe qui précède car celui-ci est en mode « clear : both ».



Voici le code CSS correspondant à l'exemple ci-dessus :

```

.imgleft {
  float: left;
}
.imgright {
  float: right;
}
.imgalone {
  float: left;
}
.donotdisturb {
  clear: both;
}
  
```

Le code HTML de l'exemple (les « lorem ipsum » ont été raccourcis) :

```


  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ornare semper nibh, a egestas purus consequat a. Nunc iaculis diam dignissim leo finibus, vitae scelerisque ligula commodo. Vivamus et ante nisi. In hac habitasse platea dictumst. 
  <p class="donotdisturb">Maecenas semper fermentum tellus, consectetur suscipit urna fringilla a. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. </p>
  
  
```

Avant de clore ce chapitre, je rappelle un point évoqué dans le cours « HTML5 – Premiers pas », à savoir que certaines balises et propriétés sont dépréciées en HTML5 :

<https://www.w3.org/TR/html5-diff/#obsolete-elements>

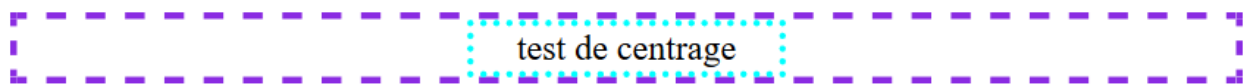
<https://www.w3.org/TR/html5-diff/#obsolete-attributes>

Sachant que certains éléments HTML dépréciés servaient à positionner des éléments dans la page, comme la balise « center » ou l'attribut « align », il faut trouver les solutions de substitution permettant de faire la même chose via CSS. Le site Openclassrooms propose un excellent tuto sur le sujet :

<https://openclassrooms.com/courses/centrer-en-css>

Je recommande une lecture attentive de ce tuto, mais on retiendra notamment qu'il est possible de centrer un élément dans son conteneur parent en définissant une marge « auto » sur l'élément HTML considéré.

Voici un exemple dans lequel on centre une « div » à l'intérieur d'une autre « div », et on en profite pour centrer le texte contenu par la « div » :



Voici le code CSS :

```
div {  
    border-style: dashed;  
    border-color: blueviolet;  
}  
.center-div {  
    text-align: center; /* centrage du texte dans la div */  
    /* display: block => valeur par défaut */  
    width: 25%; /* occupe 25 % de l'espace du conteneur */  
    margin:auto; /* centre la div dans le conteneur parent */  
    border-style: dotted;  
    border-color: aqua;  
}
```

Voici le code HTML :

```
<div>  
    <div class="center-div">test de centrage</div>  
</div>
```

Dans l'exemple ci-dessus, j'ai utilisé les propriétés « border-style » et « border-color » sur les 2 « divs » afin de mieux distinguer chaque « div », et ainsi de mieux comprendre comment chaque élément se positionne.

Avant de clore ce chapitre, parlons brièvement des techniques permettant de cacher des éléments HTML.

En règle générale, on utilisera la propriété CSS « display » associée à la valeur « none ».

Cette valeur a pour effet de faire disparaître l'élément considéré, comme dans l'exemple suivant :

```
img {  
    display: none;  
}
```

ATTENTION : il ne faut pas confondre « display: none » et « visibility: hidden » :

- « display: none » : masque totalement l'élément et annule des propriétés telles que margin, padding, width, height...
- « visibility: hidden » : masque seulement l'élément, laissant un espace vide sur l'emplacement de l'élément concerné.

En conclusion :

Ce chapitre est dense, il contient beaucoup d'informations à « digérer ». Je vous recommande de prendre du temps pour étudier les différents exemples, les modifier, les triturer en tous sens. C'est la meilleure manière de comprendre les concepts que nous venons d'étudier.

2.4.3 Styler une liste et en faire un menu

Vous avez très certainement navigué sur des sites internet affichant des menus horizontaux, sans vous douter qu'il s'agissait de simples listes « ul » comme celles que vous avez étudié durant le cours d'introduction à HTML.

Code HTML	<pre><ul id="menu_horizontal"> Accueil News Forum Contact Admin </pre>
Rendu sans CSS	<ul style="list-style-type: none"> • Accueil • News • Forum • Contact • Admin
Rendu avec CSS	Accueil News Forum Contact Admin

Le rendu avec CSS a été obtenu grâce au code suivant :

```
ul#menu_horizontal li {
  display : inline;
  padding : 0 0.5em; /* Pour espacer les boutons entre eux */
}
```

On constate qu'il suffit parfois de très peu de code CSS, pour modifier de manière importante le rendu de certains éléments.

Le site Alsacrédations propose un excellent tutorial sur la création de menus en CSS :
<https://www.alsacreations.com/tuto/lire/574-Creer-des-menus-simples-en-CSS.html>

Il est intéressant de noter qu'une balise « a » peut prendre différents états :

- a:link état standard, lien non visité
- a:visited lien déjà visité

- `a:hover` état temporaire déclenché par le fait de survoler l'élément avec la souris
- `a:focus` état temporaire déclenché par le fait de se positionner sur le lien avec le clavier (via la touche de tabulation)
- `a:active` lien cliqué

Chacun de ces états peut faire l'objet d'un style particulier.

```
a:hover {  
    color: grey;  
    background-color: purple;  
}  
a:link {  
    color: green;  
}  
a:visited {  
    color: brown;  
}  
a:focus {  
    color: yellow;  
    background-color: black;  
}  
a:active {  
    color: red;  
}
```

IMPORTANT : Ces états distincts sont appelés des pseudo-classes CSS. Nous verrons dans un prochain chapitre que les pseudo-classes sont plus étoffées avec CSS3.

Pour que le navigateur prenne correctement en compte le style de ces différents états, il faut respecter les règles suivantes :

- « `a:hover` » doit être défini avant « `a:link` »
- « `a:visited` » et « `a:active` » doivent être définis après « `a:hover` »

Quelques propriétés CSS associées aux balises « `a` » qu'il est intéressant de connaître :

- `list-style-type`
- `list-style-image`
- `list-style-position`
- `list-style`

Pour de plus amples informations sur ces propriétés :

https://www.w3schools.com/css/css_list.asp

2.4.4 Sélecteurs CSS

Nous avons vu dans le chapitre d'introduction aux sélecteurs comment sélectionner des éléments d'une page sur :

- le type d'élément (sélection de tous les éléments de même type)
- l'identifiant d'un élément (son attribut « id »)
- la classe (sélectionne un ou plusieurs éléments rattachés à la même classe)

Mais nous pouvons écrire des sélecteurs plus sophistiqués, tels que :

Sélecteurs descendants (Descendant selectors)	Syntaxe : (p a)	Applique un style spécifique à toutes les "ancres" qui se trouvent à l'intérieur d'un "paragraphe"
Sélecteurs d'enfants (child selectors)	Syntaxe : (div > a)	Ce sélecteur est plus contraignant, il ne s'applique qu'aux "ancres" qui sont des enfants directs d'une "div". Si des éléments intermédiaires se trouvent entre la "div" et une balise "a", cette dernière n'est pas retenue par le sélecteur
Frère ou Soeur adjacent (Adjacent sibling)	Syntaxe : (li:first-of-type + li)	les éléments sélectionnés sont strictement au même niveau. Dans l'exemple ci-contre, on veut sélectionner le second élément « li » d'une liste (*)
Sélection d'une liste d'éléments	h1, p, div, span { ... }	les éléments sélectionnés peuvent se situer à différents niveaux de la page
Limiter le périmètre d'une sélection	p.main {} div p.special {}	Paragraphes utilisant la classe « main » ; Paragraphes utilisant la classe « special » et se trouvant dans une « div »

(*) ce type de sélecteur n'est pas évident à comprendre. L'exemple ci-dessous, emprunté à MDN, vous semblera peut être plus clair :

CSS

```
1  li:first-of-type + li {  
2      color: red;  
3  }
```

HTML

```
1  <ul>  
2      <li>Un</li>  
3      <li>Deux</li>  
4      <li>Trois</li>  
5  </ul>
```

Résultat

- Un
- Deux
- Trois

Source : https://developer.mozilla.org/fr/docs/Web/CSS/S%C3%A9lecteur_de_voisin_direct

Pour illustrer les principes énoncés page précédente, reprenons notre liste d'albums de bande dessinée (vue dans le cours sur HTML5), et voyons comment nous pouvons la « styler » :

1. Garulfo
 - Auteurs : Ayroles, Maïorana, Leprévost
 - Editeur : Delcourt
2. Horologiom
 - Auteurs : Fabrice Lebeault
 - Editeur : Delcourt
3. Le château des étoiles
 - Auteurs : Alex Alice
 - Editeur : Rue De Sèvres
4. Le voyage extraordinaire
 - Auteurs : Camboni, Filippi
 - Editeur : Vents d'Ouest
5. Sillage
 - Auteurs : Buchet, Morvan
 - Editeur : Delcourt
6. Bone
 - Auteurs : Jeff Smith
 - Editeur : Delcourt
7. Travis
 - Auteurs : Christophe Quet, Fred Duval
 - Editeur : Delcourt

```

HTML
-----
39 +         <ul>
40 +             <li>Auteurs : Christophe Quet, Fred Duval</li>
41 +             <li>Editeur : Delcourt</li>
42 +         </ul>
43     </li>
44 </ol>
45

CSS
1 + ol li {
2     color: green;
3 }
4 + ol li li {
5     color: blue;
6 }
7

```

Vous obtiendrez un résultat similaire en appliquant cette petite variante du code CSS précédent :

```

ol li {
    color: green;
}
ol li > ul li {
    color: blue;
}

```

Le CSS3 permet d'effectuer des sélections avec des critères de filtrage très précis. Nous allons en étudier un exemple ci-dessous.

Soit la liste HTML suivante :

```
<ul class="liste" id="id-liste">
  <li id="item_1" class="good" data-info="item-1">Elément 1</li>
  <li id="item_2" class="good" data-info="">Elément 2</li>
  <li id="item_3" class="good" >Elément 3</li>
  <li id="item_4" class="bad" data-info="item-4">Elément 4</li>
  <li id="item_5" class="good" data-info="item-5">Elément 5</li>
  <li id="item_6" class="bad" data-info="item-6">Elément 6</li>
  <li id="item_7" class="good" data-info="item-7">Elément 7</li>
</ul>
```

Voici le CSS :

```
/* tout élément ayant l'attribut "data-info" est de couleur marron */
[data-info] {
  color:maroon;
}
/* tout élément dont l'attribut "data-info" commence par "item" */
/* est de couleur "aqua" et encadré de points */
[data-info|=item] {
  color:aqua;
  border-style: dotted;
}
```

Et voici le résultat :

- Elément 1
- Elément 2
- Elément 3
- Elément 4
- Elément 5
- Elément 6
- Elément 7

Observez les éléments 2 et 3 : êtes-vous sûr d'avoir compris pourquoi ils ne sont pas de la même couleur que les autres éléments ? Faites des essais, modifiez les sélecteurs, c'est le meilleur moyen de comprendre et d'apprendre.

Toujours à partir de la même liste HTML, voici une autre série de sélecteurs CSS. Vous n'allez pas forcément comprendre leur fonctionnement tout de suite, testez-les :

```
/* les lignes paires ont le texte en gris */
.liste>li:nth-child(even) {
    color:silver;
}
/* le premier "li" est en rouge */
.liste>li:first-of-type {
    color:red;
}
/* le dernier "li" est en maron */
.liste>li:last-of-type {
    color: maroon;
}
/* les lignes impaires ont un fond violet */
.liste>li:nth-child(2n+1) {
    background-color: blueviolet;
}
/* strictement équivalent au sélecteur précédent */
.liste>li:nth-child(odd) {
    background-color: blueviolet;
}
```

Voici le résultat :

- **Elément 1**
- Élément 2
- **Elément 3**
- Élément 4
- **Elément 5**
- Élément 6
- **Elément 7**

Les sélecteurs de cet exemple sont assez sophistiqués, il s'appuient notamment sur des pseudo-classes (comme par exemple :last-of-type, :nth-child, etc...), qui sont décrites dans les chapitres suivants.

Vous voyez que les sélecteurs CSS sont des outils très puissants, on peut les utiliser aussi pour styler des champs de formulaires, comme dans l'exemple suivant :

Le CSS :

```
input[type=text]:required {  
    border-color: red;  
    border-style: solid;  
}
```

Le HTML :

```
<input type="text" id="nomid" name="nom" value="" required  
placeholder="votre nom SVP" />
```

Le résultat :

Nom

Si l'on prend un peu de recul par rapport au sélecteur que l'on vient d'écrire, on pourrait dire que cela ressemble étrangement à une requête SQL.

Eh oui, écrire ce sélecteur CSS :

```
input[type=text]:required
```

... revient à écrire une requête demandant au navigateur de rechercher dans le DOM s'il existe des éléments « input » de type « text », dont la liste des attributs contient la valeur « required ». En SQL, cela pourrait donner ceci :

```
UPDATE dom  
SET *.style.border-color = "red",  
    *.style.border-style = "solid"  
WHERE dom.element = "input"  
    and dom.element.type = "text"  
    and dom.element.attributes like "%required%"
```

Ce parallèle semble audacieux au premier abord, mais en réalité il est très juste. Si vous connaissez, et surtout si vous aimez le SQL, alors vous allez peut être aimer les sélecteurs CSS 😊. Si vous n'aimez pas SQL, il est peut être temps de revoir votre position.

Vous l'aurez peut être compris, ce chapitre sur les sélecteurs CSS est sans doute l'un des chapitres les plus importants de ce cours.

Je dois souligner que nous allons retrouver ces sélecteurs CSS aussi en Javascript, car on verra qu'il est possible avec les API « `querySelector` » et « `querySelectorAll` », d'utiliser les mêmes techniques qu'en CSS, pour sélectionner des éléments du DOM. Exemple :

```
var test = document.querySelectorAll('input[type=text]:required');
console.log(test);
```

Résultat dans la console du navigateur, obtenu à partir du champ de formulaire de l'exemple précédent :

```
▼ [input#nomid] ⓘ
  length: 1
  ► 0: input#nomid
  ► __proto__: NodeList
```

On voit que l'on est en mesure de sélectionner, par programmation, la liste des éléments du DOM qui sont des « input » de types « text » contenant un attribut « required ». On utilise ici strictement la même syntaxe que dans l'exemple de la page précédente.

Bref, si vous maîtrisez les sélecteurs CSS, vous êtes les « rois du pétrole ». J'ai l'air de plaisanter, mais en réalité je n'ai jamais été aussi sérieux.

Si vous souhaitez maîtriser pleinement les sélecteurs CSS, je vous recommande la lecture du support « HTML5 et Javascript » qui se trouve dans le même dépôt Github que le présent document. Ce support de cours contient plusieurs chapitres relatifs aux sélecteurs CSS, notamment le chapitre 2.7.2, le chapitre 9.1, et surtout le chapitre 9.2 qui contient de nombreux exemples et exercices.

2.4.5 Pseudo-classes CSS

Définition des pseudos-classes empruntée à Mozilla :

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

Les pseudo-classes et les pseudo-éléments permettent d'appliquer un style à un élément non seulement en fonction de son contenu dans l'arbre du document mais aussi en fonction de facteurs externes (l'historique de navigation par exemple avec :visited ; le statut du contenu avec :checked ; la position de la souris :hover).

Quelques exemples d'utilisation :

Les champs de formulaires obligatoires (attribut "required") peuvent être "stylés" au moyen de la pseudo-classe :required.

Et les champs optionnels peuvent être "stylés" au moyen de la pseudo-classe :optional.

```
<style>
  input:required { border-color: red; }
  input:optional { border-color: silver; }
</style>
```

Les pseudo-classes :valid et :invalid sont utiles pour appliquer un style particulier aux champs dont le contenu est valide, et à ceux dont le contenu est invalide.

```
<style>
  input:valid { color: green; }
  input:invalid { color: red; }
</style>
```

On peut "styler" les éléments de formulaire selon un attribut particulier. Par exemple, les champs ayant l'attribut "disabled" peuvent être "stylés" avec la pseudo-classe :disabled. L'inverse est vrai pour l'attribut "enabled". Pour un champ de formulaire "readonly", vous disposez de la pseudo-classe :read-only (l'inverse de cette pseudo-classe étant :read-write).

Les boutons radios et cases à cocher bénéficient de pseudo-classes dédiées, comme :indeterminate quand aucune case n'est cochée, et surtout :checked pour attribuer un style particulier à des case cochées.

Curieusement, il n'existe pas de pseudo-classes pour "styler" les cases non cochées, mais on peut contourner la difficulté en utilisant le sélecteur suivant :

```
var notchecked = document.querySelectorAll("input[type='checkbox']:not(:checked)");
```

Pour les champs de type "number" et "range", on dispose des pseudo-classes :in-range et :out-of-range, qui se passent de commentaires.

Liste des pseudo-classes CSS empruntée à la documentation de Mozilla :

- [:active](#)
- [:any](#)
- [:checked](#)
- [:default](#)
- [:dir\(\)](#)
- [:disabled](#)
- [:empty](#)
- [:enabled](#)
- [:first](#)
- [:first-child](#)
- [:first-of-type](#)
- [:fullscreen](#)
- [:focus](#)
- [:hover](#)
- [:indeterminate](#)
- [:in-range](#)
- [:invalid](#)
- [:lang\(\)](#)
- [:last-child](#)
- [:last-of-type](#)
- [:left](#)
- [:link](#)
- [:not\(\)](#)
- [:nth-child\(\)](#)
- [:nth-last-child\(\)](#)
- [:nth-last-of-type\(\)](#)
- [:nth-of-type\(\)](#)
- [:only-child](#)
- [:only-of-type](#)
- [:optional](#)
- [:out-of-range](#)
- [:read-only](#)
- [:read-write](#)
- [:required](#)
- [:right](#)
- [:root](#)

- [:scope](#)
- [:target](#)
- [:valid](#)
- [:visited](#)

On trouve également les pseudo-classes `:after` et `:before`, que l'on peut utiliser pour ajouter un contenu avant ou après un élément.

Exemple :

```
.defile:after {  
    content: attr(data-text);  
}
```

Explication : dans l'exemple ci-dessus, le sélecteur identifie les éléments liés à la classe « `defile` ». Pour chaque élément sélectionné, la fonction `attr()` récupère le contenu de l'attribut « `data-text` » (s'il existe), et elle insère ce contenu juste après le contenu HTML de l'élément considéré.

A noter : on retrouvera ces mêmes sélecteurs CSS en Javascript, car il est possible de les utiliser avec les API `querySelector` et `querySelectorAll` (cf. cours sur « Javascript et HTML5 »). Dans le chapitre suivant, nous allons approfondir l'utilisation de certains sélecteurs CSS3, au travers de différents exemples.

2.4.6 Pseudo-classes CSS3

Ce chapitre approfondit le chapitre précédent, en particulier sur l'utilisation de certaines pseudo-classes apparues avec la norme CSS3.

La pseudo-class :nth-child()

La notation :nth-child(an+b) désigne un élément de la page qui a "an+b-1" frères et soeurs (en anglais : "siblings") **devant lui** dans l'arborescence du DOM. Plutôt que de s'apesantir sur une définition inutilement compliquée, étudions quelques exemples :

```
tr:nth-child(2n+1) /* sélectionne toutes les lignes impaires d'un tableau HTML */
tr:nth-child(odd)  /* idem */
tr:nth-child(2n+0) /* sélectionne toutes les lignes paires d'un tableau HTML */
tr:nth-child(even) /* idem */

<style>
/* Exemple : sur un groupe de 4 paragraphes, appliquer des couleurs différentes sur
chacun des paragraphes */
p:nth-child(4n+1) { color: navy; }
p:nth-child(4n+2) { color: green; }
p:nth-child(4n+3) { color: maroon; }
p:nth-child(4n+4) { color: purple; }
</style>
```

Autres exemples avec le signe moins :

```
:nth-child(10n-1) /* sélectionne les 9ème, 19ème, 29ème, etc, éléments */
:nth-child(10n+9) /* idem */
:nth-child(10n+-1) /* syntaxe invalide, sera ignorée par le navigateur */
```

Cas particulier avec :nth-child(b).

Exemples:

```
foo:nth-child(0n+5) /* sélectionne un élément foo qui est 5ème enfant du parent
auquel il appartient */
foo:nth-child(5)    /* idem */
```

Si $a=1$, ou $a=-1$, alors il peut être omis.

```
bar:nth-child(1n+0) /* sélectionne tous les éléments bar */
bar:nth-child(n+0)  /* idem */
bar:nth-child(n)    /* idem */
bar                 /* idem */
```

Attention à la gestion des espaces dans les critères de sélection :

```
/* Exemples valides: */
:nth-child( 3n + 1 )
:nth-child( +3n - 2 )
:nth-child( -n+ 6)
:nth-child( +6 )
/* Exemples invalides: */
:nth-child(3 n)
:nth-child(+ 2n)
:nth-child(+ 2)
```

Autre exemple :

```
tr:nth-child(-n+6) /* sélectionne la 6ème ligne d'un tableau HTML */
```

La pseudo-classe :nth-last-child() :

La notation :nth-last-child(an+b) désigne un élément de la page qui a "an+b-1" frères (siblings) **derrière lui** dans l'arborescence du DOM.

Exemples :

```
tr:nth-last-child(-n+2) /* sélectionne les 2 dernière lignes d'un tableau HTML */
li:nth-last-child(odd) /* sélectionne toutes les lignes impaires d'une liste "li"
en partant de la dernière */
```

La pseudo-classe :nth-of-type() :

La notation de pseudo-classe :nth-of-type(an+b) sélectionne un élément qui a an+b-1 éléments "frères et soeurs" (siblings) **de même type** devant lui dans l'arborescence du DOM.

Exemple :

```
/* Alternance d'image paires et impaires, à droite et à gauche de la page */
img:nth-of-type(2n+1) { float: right; }
img:nth-of-type(2n) { float: left; }
```

La pseudo-classe :nth-last-of-type()

La notation :nth-last-of-type(an+b) représente un element qui a an+b-1 frères et soeurs de même type **après** lui dans l'arborescence du DOM.

Exemple : Pour représenter tous les enfants de type "h2" dans l'arborescence de l'élément "body", excepté le premier et le dernier, on peut utiliser la syntaxe suivante

```
body > h2:nth-of-type(n+2):nth-last-of-type(n+2)
body > h2:not(:first-of-type):not(:last-of-type) /* idem */
```

La pseudo-classe :first-child

Equivalut à :nth-child(1).

La pseudo-classe :first-child représente un élément qui est le premier enfant d'un groupe sélectionné.

Par exemple, le sélecteur suivant sélectionne un élément "p" qui est le premier enfant d'un élément "div".

```
div > p:first-child
```

Sur le bloc de code HTML ci-dessous, le sélecteur ci-dessus sélectionnera l'élément "p" qui se trouve à l'intérieur de la "div" :

```
<p> The last P before the note.</p>
<div class="note">
  <p> The first P inside the note.</p>
</div>
```

Mais ce même sélecteur ne sélectionnera pas l'élément "p" à l'intérieur de l'élément "div" dans l'exemple de code HTML ci-dessous :

```
<p> The last P before the note.</p>
<div class="note">
  <h2> Note </h2>
  <p> The first P inside the note.</p>
</div>
```

La pseudo-classe :last-child

Equivalut à :nth-last-child(1).

La pseudo-classe :last-child fait référence à un élément qui est le dernier enfant d'un élément sélectionné précédemment.

Par exemple, le sélecteur suivant sélectionne le dernier élément "li" d'une liste non ordonnée ("ol").

```
ol > li:last-child
```

La pseudo-classe :first-of-type

Equivalut à :nth-of-type(1).

La pseudo-classe :first-of-type fait référence à un élément qui est le premier frère (ou soeur) de son type, dans la liste des enfants d'un élément parent sélectionné.

Par exemple, le sélecteur suivant représente le premier élément "dt" d'une liste de définition

"dl", cet élément "dt" étant le premier élément de son type dans la liste des enfants de l'élément "dl".

dl dt:first-of-type

Ce sélecteur permettra de récupérer les éléments "dt gigogne" et "dt fusée", mais pas l'élément "dt table".

```
<dl>
  <dt>gigogne</dt>
  <dd>
    <dl>
      <dt>fusée</dt>
      <dd>multistage rocket</dd>
      <dt>table</dt>
      <dd>nest of tables</dd>
    </dl>
  </dd>
</dl>
```

La pseudo-classe :last-of-type

Equivaut à :nth-last-of-type(1).

La pseudo-classe :last-of-type représente un élément qui est le dernier frère (ou soeur) de son type, dans la liste des enfants d'un élément parent sélectionné.

Par exemple, le sélecteur suivant sélectionne la dernière cellule "td" de la ligne "tr" :

tr > td:last-of-type

A noter : on retrouvera ces notions de sélecteurs CSS3 en Javascript, car on peut appliquer strictement les mêmes sélecteurs avec les API `querySelector` et `querySelectorAll` (cf. cours sur « Javascript et HTML5 »).

2.4.7 Mediaqueries et Responsive design

Les "media queries" constituent une des avancées les plus importantes du HTML5, et répondent à un des besoins essentiels des webdesigners, désigné par le terme de "responsive design".

Un "media query" est en fait un état logique pouvant prendre 2 positions :

- true : les règles de style définies sur ce media query s'appliquent
- false : elles ne s'appliquent pas

Petit rappel : en CSS 2.1 et HTML 4.01, la syntaxe pour affecter un groupe de directives CSS à un média était la suivante :

```
<link rel="stylesheet" href="foo.css" media="screen">
```

En HTML5, l'attribut "media" prend se voir associer un "media query", ce qui revient schématiquement à écrire ceci :

```
<link rel="stylesheet" href="foo.css" media="screen and (query)">
```

Nous verrons dans un instant ce que l'on peut mettre à la place de (query).

Une autre solution pour déclarer du CSS avec chargement conditionné par "media query" consiste à le déclarer directement dans le code CSS, de la façon suivante :

```
<style>
  @media screen and (query) { ... }
</style>
```

On notera qu'il est possible d'inclure des feuilles de style externes via la directive @import, de la façon suivante :

```
<style>
@import url('foo.css') screen and (query);
</style>
```

Mais des problèmes de performances ont été relevés avec cette technique, donc il convient de l'utiliser avec prudence. Pour de plus amples informations à ce sujet, on recommandera la lecture de l'article de Steve Souders

Chargement du CSS conditionné par la taille de l'écran

Les "media queries" sont généralement utilisés pour charger dynamiquement du code CSS sous condition que les caractéristiques de l'écran répondent aux critères définis dans la requête

(query).

Premier exemple :

```
<style>
  @media screen and (width: 480px) {
    body { background-color: white; }
  }
  @media screen and (width: 768px) {
    body { background-color: silver; }
  }
</style>
```

Second exemple :

```
<link rel="stylesheet" href="foo_480.css" media="screen and (width: 480px)">
<link rel="stylesheet" href="foo_768.css" media="screen and (width: 768px)">
```

On peut donc utiliser cette technique pour adapter la présentation de la page aux caractéristiques d'affichage de différents types de terminaux. Par exemple, un texte apparaissant sur plusieurs colonnes dans une page de 768 pixels, pourra être affiché sur une seule colonne dans une page de résolution inférieure, avec éventuellement une autre police, voire une autre taille de caractères.

Vous devez tenir compte de deux ensembles de dimensions: d'abord :

- d'une part, les dimensions du dispositif lui-même,
- et d'autre part, les dimensions de la zone d'affichage de l'agent sur cet appareil (qui est pour la plupart des utilisateurs un navigateur web, mais la fenêtre peut aussi être une fenêtre d'application). Une personne peut visiter votre site en utilisant une énorme télévision à écran large, mais c'est de peu d'intérêt pour vous si l'application que la personne utilise pour afficher votre site occupe seulement un quart de l'écran. Mais en règle générale, ces deux ensembles de dimensions sont les mêmes sur la plupart des smartphones et tablettes, par exemple, la largeur du navigateur est la même que la largeur de l'appareil.

Cette seconde dimension de la zone d'affichage est communément désignée par le terme de "viewport". Pour préciser le viewport, on indique généralement la dimension horizontale (la largeur en pixels) du viewport, à charge pour le périphérique d'ajuster la résolution horizontale en proportion. C'est la raison pour laquelle on précise rarement - mais il est tout à fait possible de le faire si besoin - le paramètre "height" (hauteur) dans les déclarations telles que celle ci-dessous :

```
<link rel="stylesheet" href="foo.css" media="screen and (width: 480px)">
```

On notera que les dimensions du viewport sont généralement exprimées en pixels, mais qu'il est possible d'utiliser d'autres unités, à condition que les périphériques "cibles" les acceptent.

Les attributs "width" et "height" sont un peu trop restrictifs, on aura tout intérêt à les coupler à,

ou à les remplacer par, les attributs "max-width", "min-height", "max-height" et "min-height", comme dans l'exemple suivant :

```
<link rel="stylesheet" href="foo_480.css" media="screen and (max-width: 480px)">
<link rel="stylesheet" href="foo_768.css" media="screen and (min-width: 481px) and
(max-width: 768px)">
<link rel="stylesheet" href="foo_1024.css" media="screen and (min-width: 769px)">
```

On pourra dès lors ajuster, dans chacun des fichiers CSS ci-dessus les dimensions des "div" et autres blocs de données, aux dimensions du viewport considéré.

On peut aussi déclarer des "media queries" directement sur des éléments HTML (ou des classes), à l'intérieur même des feuilles de styles :

```
<style>
div1 @media screen and (min-width: 769px) {
  h1 {
    border-style: solid;
    font-size: 2em;
  }
}
div1 @media screen and (max-width: 768px) {
  h1 {
    border-style: dotted;
    font-size: 3.6em;
  }
}
</style>
```

Nous avons vu les critères de sélection les plus simples, mais il en existe d'autres, éventuellement combinables entre eux :

```
<style>
@media screen and (device-aspect-ratio: 4/3) { ... }
@media screen and (min-aspect-ratio: 8/5) { ... }
@media screen and (orientation: portrait) { ... }
@media all and (orientation: landscape) and (min-width: 800em) { ... }
@media (orientation: landscape) and (min-width: 800em) { ... }
@media (orientation: landscape), (min-width: 800em) {}
@media not all and (device-aspect-ratio: 8/5) {}
</style>
```

Dans le cas des images notamment, il peut être intéressant de prévoir plusieurs résolutions pour chaque image, de manière à pouvoir charger les images dans la résolution la plus adaptée au périphérique considéré :

```
<style>
@media screen and (resolution: 96dpi) {
  E { background-image: url('foo.png'); }
}
@media screen and (min-resolution: 192dpi) {
  E { background-image: url('foo-hires.png'); }
}
@media screen and (min-resolution: 2dppx) {
  E { background-image: url('foo-hires.png'); }
}
</style>
```

On peut aussi adapter le CSS au type de périphérique d'entrée utilisé par l'utilisateur. Par exemple :

```
<style>
@media screen and (pointer: coarse) {
  a { padding: 1em; }
}
</style>
```

Le pointeur "coarse" (en anglais "grossier", au sens "graphique" du terme) utilisé dans l'exemple ci-dessus désigne les périphériques tactiles. Les périphériques plus précis, tels que la souris ou le stilet, sont caractérisés par le mot clé "fine".

Une caractéristique essentielle des périphériques tels que la souris est de permettre le survol (hover) des éléments de la page, ce que ne permettent pas les périphériques tactiles. Media query prend ce problème en considération, avec le mot clé "hover", comme dans l'exemple suivant :

```
<style>
@media screen and (pointer: coarse) {
  a { padding: 1em; }
}
@media screen and (hover:0) {}
</style>
```

Manipulation des Media Queries en JavaScript

Le langage Javascript a été enrichi de méthodes permettant de manipuler les "media queries".

La méthode `matchMedia()` attachée à l'objet "window" permet d'obtenir des informations intéressantes. Testez la dans la console de votre navigateur :

```
window.matchMedia('screen and (min-width: 800px)');
```

```
/*  
Le résultat obtenu est un objet de type MediaQueryList :  
- Résultat obtenu avec une fenêtre d'affichage de taille supérieure ou égale à 800  
pixels :  
    MediaQueryList {matches: true, media: "screen and (min-width: 800px)",  
addListener: function, removeListener: function}  
- Résultat obtenu avec une fenêtre d'affichage de taille supérieure ou égale à 800  
pixels :  
    MediaQueryList {matches: false, media: "screen and (min-width: 800px)",  
addListener: function, removeListener: function}  
*/
```

On peut dès lors exploiter l'information pour tout un tas d'usages :

```
var mq = window.matchMedia('screen and (min-width: 800px)');  
if (mq.matches) {  
    // faire quelque chose  
} else {  
    // faire autre chose  
}
```

Recommandations :

Les "media queries" rencontrent un grand succès dans le monde du webdesign, pour leur souplesse, mais aussi parce qu'ils sont reconnus par la grande majorité des terminaux équipés de navigateurs récents. Si vous souhaitez les utiliser pour des applications "métier", il convient de faire des tests poussés, pour s'assurer que les terminaux cibles de vos applications les supportent.

Une des recommandations faites par certains gourous du webdesign, qu'il convient de prendre en considération, est la suivante : il est préférable de concevoir l'interface utilisateur en premier lieu pour les périphériques d'affichage cibles ayant le plus petit viewport (donc les smartphones). Une fois que l'interface utilisateur est au point, on peut compléter le design en ajoutant des caractéristiques propres aux périphériques offrant des dimensions supérieures. Il semblerait que dans la pratique, la méthode inverse - consistant à concevoir l'application pour le viewport le plus élevé, puis à la "dépouiller" au fur et à mesure que l'on diminue le viewport - donne des résultats moins satisfaisants.

2.4.8 Effets de dégradés

La norme CSS3 apporte la possibilité de pouvoir générer des effets de dégradés à la place des banales couleurs de remplissage disponibles en CSS2. Le présent chapitre est une introduction à l'utilisation de ces effets.

Avec CSS3, on peut désormais obtenir des effets de dégradés linaires :

```
background-image: linear-gradient(to top left, #C24704, #FFEB79);
```

On peut aussi obtenir des effets de dégradés radials :

```
background-image: radial-gradient(circle farthest-corner at 100px 50px,
    #C24704, #FFEB79 35%, #00ADA7);
```

Ou encore des effets de motifs assez sophistiqués :

```
background-image: repeating-linear-gradient(-45deg, #426A77, #FFF 6px);
```

On peut également définir des effets d'arrondi, exprimés en pourcentages ou en pixels :

```
border-radius: 20%;
border-radius: 3px;
```

Petite astuce : on peut utiliser la propriété « border-radius » pour dessiner des cercles. Exemple (à tester dans Codepen par exemple) :

```
<div style="width:100px; height:100px; border-radius:50%; border: 1px solid
black"></div>
```

Les éditeurs de certains navigateurs avaient fait le pari de commencer à intégrer ces nouveaux filtres, avant que la spécification CSS3 ne soit complètement finalisée. N'étant pas certains de l'implémentation finale de certains effets, ils décidèrent d'adopter une convention consistant à préfixer les nouveaux filtres CSS comme dans l'exemple suivant :

```
background-image: -moz-linear-gradient(315deg, #C24704, #FFEB79);
/* Mozilla (Firefox) */
background-image: -o-linear-gradient(315deg, #C24704, #FFEB79); /* Opera */
background-image: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#C24704),
to(#FFEB79)); /* Webkit (Chrome et Safari) */
background-image: linear-gradient(135deg, #C24704, #FFEB79);
/* Spécification CSS3 standard */
```

A partir de Firefox version 27 et Google Chrome version 32, les préfixes indiqués ci-dessus ne sont plus nécessaires.

En revanche, sur IE9, ces effets sont inopérants, alors qu'ils fonctionnent à partir de IE 10.

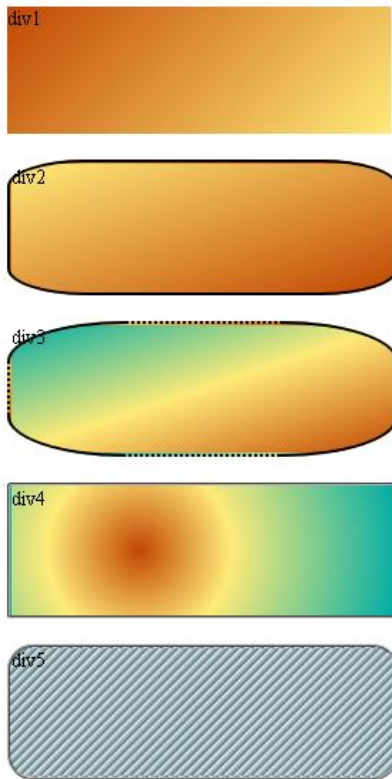
Exemple d'utilisation de ces effets sur plusieurs « div » :

```
<!DOCTYPE html>
<html lang="fr">
```

```
<meta charset="utf-8">
<head>
<title>Insert title here</title>
</head>
<style>
div {
    width: 300px;
    height: 100px;
    margin: 20px;
}
.div1 {
    background-image: linear-gradient(#C24704, #FFEB79);
    border-radius: 10%;
    border-style: dashed;
}
.div2 {
    background-image: linear-gradient(to top left, #C24704, #FFEB79);
    border-radius: 20%;
    border-style: solid;
}
.div3 {
    background-image: linear-gradient(to top left, #C24704, #FFEB79, #00ADA7);
    border-radius: 30%;
    border-style: dotted;
}
.div4 {
    background-image: radial-gradient(circle farthest-corner at 100px 50px,
    #C24704, #FFEB79 35%, #00ADA7);
    border-radius: 3px;
    border-style: groove;
}
.div5 {
    background-image: repeating-linear-gradient(-45deg, #426A77, #FFF 6px);
    border-radius: 20px;
    border-style: ridge;
}

</style>
<body>
<div class="div1">div1</div>
<div class="div2">div2</div>
<div class="div3">div3</div>
<div class="div4">div4</div>
<div class="div5">div5</div>
</body>
</html>
```


Les résultats obtenus dans Firefox et Google Chrome sont assez similaires :



On le voit, ces nouveaux effets permettent à des développeurs webs peu à l'aise avec le webdesign, d'obtenir des effets esthétiques, avec seulement quelques lignes de code CSS.

Pour approfondir le sujet, les développeurs gagneront à étudier le code CSS du Bootstrap de Twitter, qui est riche en effets de ce genre.

Avant de décider de les utiliser de manière intensive, il conviendra de vérifier la compatibilité de ces effets avec les navigateurs cibles.

2.4.9 Positionnement et Z-index

Le positionnement des éléments d'une page est une opération délicate, qui nécessite une bonne connaissance des attributs CSS dédiés, et un peu de pratique.

C'est l'attribut CSS « position » qui nous permet de placer précisément un élément sur une page.

Nous disposons de plusieurs types de positionnement (sont indiqués en gras ceux qui sont les plus couramment utilisés) :

- **static** : valeur par défaut, les éléments sont affichés dans l'ordre dans lequel ils sont définis dans le document
- **absolute** : l'élément est positionné relativement au premier élément ancêtre positionné de manière non statique
- **fixed** : l'élément est positionné relativement à la fenêtre du navigateur
- **relative** : l'élément est positionné relativement à sa position normale, donc "left:20px" ajoute 20 pixels sur la gauche de l'élément
- **initial** : redéfinit la valeur d'une propriété en lui redonnant sa valeur par défaut.
- **inherit** : hérite de la propriété de l'élément parent (assez peu utilisé)

Le site Alsacrérations propose un bon tutorial sur les problèmes de positionnement :

<https://www.alsacreations.com/tuto/lire/608-initiation-positionnement-css.html>

Nous allons nous baser sur certaines des définitions proposées dans l'article d'Alsacrérations, et les combiner avec les exemples proposés par W3Schools (que je vous invite à tester au fil de l'eau sur Codepen ou CodeCircle).

https://www.w3schools.com/css/css_positioning.asp

La position statique (position:static)

Elle correspond simplement à la valeur par défaut d'un élément du flux. Il n'y a que peu d'intérêt à la préciser, si ce n'est dans le but de rétablir dans le flux un élément en particulier parmi une série qui serait positionnée hors du flux.

Voici un exemple emprunté à W3Schools :

```
div.static {  
  position: static;  
  border: 3px solid #73AD21;  
}
```

La position absolue (position:absolute)

Elle permet de ne pas dépendre de l'ordre dans lequel les éléments HTML sont déclarés dans le code, contrairement aux flottants que nous verrons plus tard.

La position absolue s'affranchit définitivement du cordon liant jusqu'alors intimement contenu et présentation. L'élément étant totalement extrait du flux, il ne dépend plus du tout des éléments qui le côtoient. Il faut voir le positionnement absolu comme étant une méthode radicale (mais puissante) qui retire tout à fait un élément du flux : il n'existe pour ainsi dire plus aux yeux des éléments qui, eux, restent dans le flux.

Rappelons un point important concernant ce mode de positionnement : un élément positionné en absolu se réfère non pas à son parent direct, mais au premier ancêtre positionné qu'il rencontre.

L'élément, n'étant plus dans le flux naturel, perd une de ses caractéristiques majeures qui est celle de recouvrir la totalité de la largeur disponible de l'élément parent.

Il est capital de noter qu'un élément bénéficiant d'une position absolue ne bougera pas de sa position initiale tant que l'une des propriétés top, bottom, left ou right n'a pas été précisée ; il s'agit d'ailleurs là d'un comportement applicable à toutes les positions.

Voici un exemple emprunté à W3Schools (ici appliqué à une balise « h2 » mais vous pouvez l'appliquer à une « div » si ça vous tente) :

```
h2 {  
  position: absolute;  
  left: 100px;  
  top: 150px;  
}
```

La position fixe (position:fixed)

Elle s'apparente au positionnement absolu, à l'exception des points suivants :

- Lorsque le positionnement est précisé (top, right, ...), l'élément est toujours positionné par rapport à la fenêtre du navigateur
- L'élément est fixé à un endroit et ne pourra se mouvoir, même lors de la présence d'une barre de défilement. En d'autres termes, la position initiale est fixée au chargement de la page, le fait qu'une éventuelle scrollbar puisse être utilisée n'a aucune influence sur le positionnement de l'élément : il ne bouge plus de la position initialement définie.

Exemple emprunté à W3Schools :

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```

La propriété z-index

La propriété z-index est très intéressante car elle va nous permettre de superposer des éléments HTML. On va par exemple pouvoir placer une image sous un paragraphe, ce dernier apparaissant en surimpression par-dessus l'image. On utilise aussi cette technique pour faire apparaître temporairement un bloc d'information, sous une forme généralement désignée par le terme de « fenêtre modale ».

Le site Alsacrérations l'explique ainsi :

La propriété z-index permet de préciser l'empilement de certains éléments d'une page, c'est-à-dire sur l'axe vertical. Elle permet par exemple d'indiquer que pour deux éléments A et B partiellement ou totalement superposés, A sera placé au dessus de B ou inversement.

Mais l'utilisation de cette propriété comporte quelques pièges. Il y a deux informations principales à retenir :

1. Seuls les éléments positionnés peuvent avoir un z-index. Un élément positionné est un élément dont la propriété CSS position a pour valeur relative, absolute ou fixed. Par défaut, les éléments d'une page ne sont pas positionnés (ils sont en position:static).

2. Les valeurs les plus élevées sont au premier plan, et les plus faibles sont au second plan. Un z-index de 2 sera placé au dessus d'un z-index de 1, et un z-index de -1 sera placé au dessus d'un z-index de -2.

La documentation de Mozilla relative à la propriété « position » :

<https://developer.mozilla.org/fr/docs/Web/CSS/position>

Le site documentaire de Mozilla propose un dossier très complet z-index :

https://developer.mozilla.org/fr/docs/Web/CSS/Comprendre_z-index

Un article complémentaire qui présente certains des pièges que vous pouvez rencontrer lors de l'utilisation de z-index :

<https://iamvdo.me/blog/comprendre-z-index-et-les-contextes-dempilement>

Concrètement, si l'on reprend notre exemple de photos sur le sporthocker, comment pourrait-on faire pour superposer les 3 photos, avec en prime un petit texte en surimpression ? Voici l'effet que l'on souhaiterait obtenir :



Avant de lire la solution page suivante, essayez d'imaginer comment vous pourriez vous y prendre pour superposer les images. Prenez le temps de faire quelques tests.

Voici le code HTML :

```
<div>The Sporthocker is really fun</div>



```

Et voici le code CSS :

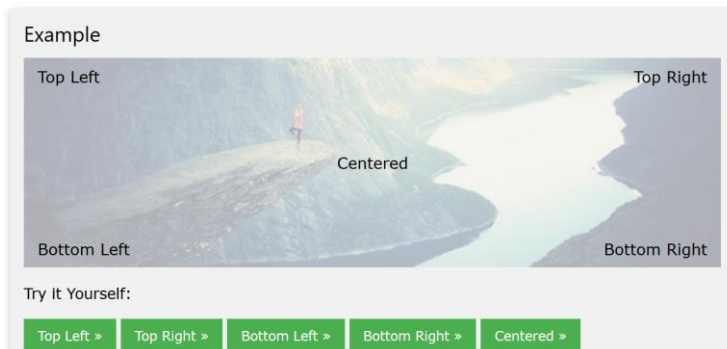
```
.img1, .img2, .img3 {
  position: absolute; /* fonctionne aussi avec "fixed" */
  top:10px;
  left:10px;
}
.img1 {
  z-index: -1;
}
.img2 {
  z-index: 1;
}
.img3 {
  z-index: 2;
}
```

Je vous invite à modifier la valeur de z-index pour certaines classes, afin de voir l'effet produit.

Pour approfondir le sujet, je vous invite également à étudier les différents exemples proposés par W3Schools sur cette page :

https://www.w3schools.com/css/css_positioning.asp

How to position text over an image:



... et à réaliser les exercices qui sont proposés à la fin de cette même page.

Vous avez sans doute remarqué que le positionnement d'éléments sur une page HTML n'est pas toujours évident.

Il faut effectuer beaucoup de tests avant de véritablement maîtriser le sujet, et ne pas hésiter à

croiser les articles et les approches, pour obtenir une bonne vue d'ensemble.

<http://fr.learnlayout.com/position.html>

2.5 Techniques CSS3 avancées

2.5.1 Inclusion d'animation Javascript en fond de page

De plus en plus de sites utilisent des fonds de page animés. Il existe sans aucun doute plusieurs manières d'effectuer des animations, certaines s'appuient exclusivement sur du CSS, avec des techniques de transition et de transformation, comme on le verra dans les chapitres suivants. La technique que nous allons étudier dans ce chapitre utilise un peu de code JS et un peu de code CSS.

Pour créer une animation pilotée par Javascript, nous allons nous appuyer sur le framework P5.js, qui simplifie l'utilisation de l'API HTML5 Canvas. Pour rappel, cette API Canvas permet d'interagir en Javascript avec les balises de même nom, et de générer des graphiques et animations à l'intérieur d'une page web.

Pour une introduction à P5.js, je recommande la lecture du tutoriel d'initiation à P5 qui se trouve dans mon dépôt Github suivant (cf. document « Tuto_P5_Premiers_pas.pdf ») :

<https://github.com/gregja/JSCorner>

Pour voir comment cela fonctionne, allez sur la page ci-dessous. Il s'agit d'un projet Codepen :

<https://codepen.io/gregja/project/full/APYbGR/>

Codepen permet, depuis mars 2017, de créer des projets en ligne. Cela permet de réaliser des prototypes d'un niveau plus avancé que ce que l'on pourrait faire avec un simple Pen. En l'état actuel des choses, CodeCircle n'offre pas la même souplesse, aussi nous nous concentrerons sur ce projet Codepen pour nos explications.

La page « index.html » du projet se présente de la façon suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
```

```

<meta charset="UTF-8">
<title>Superball in background with P5.js</title>
<style>
  #home-sketch-frame {
    position: fixed;
    width: 100%;
    height: 100%;
    left: 0;
    top: 0;
    z-index: -2;
    overflow: hidden;
    pointer-events: all;
    border: 0;
  }
</style>
</head>
<body>
<iframe tabindex='-1' id='home-sketch-frame'
  src='p5_featured/sketch_p5.html'></iframe>
<div id="lipsum">
  <p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis
    elementum et odio non varius. Morbi semper luctus libero sed ornare.
    Pellentesque id metus urna. Etiam aliquet ut tellus egestas bibendum.
  </p>
</div>
</body>
</html>

```

Pour ne pas alourdir inutilement le source ci-dessus, je n'ai laissé qu'un seul paragraphe « lorem ipsum » abrégé. Si vous ne savez pas (ou plus) ce que signifie « lorem ipsum », faites une recherche sur internet, ou reportez-vous au support du cours d'introduction au HTML5.

Il y a deux éléments importants dans le code ci-dessus :

- L'iframe :

```

<iframe tabindex='-1' id='home-sketch-frame'
  src='p5_featured/sketch_p5.html'></iframe>

```

- l'attribut CSS z-index associé à l'élément ayant pour identifiant « home-sketch-frame » :

```

#home-sketch-frame {
  z-index: -2;
}

```

L'attribut z-index fixé à -2 indique au navigateur que l'élément HTML correspondant doit se situer en arrière-plan, sous les éléments ayant une valeur de z-index supérieure (c'est-à-dire dans notre cas, tout le reste de la page, y compris les paragraphes « lorem ipsum »). Si l'on fixe cet attribut z-index à 1, le « canvas » va se trouver en premier plan, et les paragraphes « lorem ipsum » ne seront plus visibles.

L'iframe a trois attributs importants :

- « tabindex » : l'attribut étant fixé à -1, cela rend impossible toute sélection via la touche de tabulation
- « src » : définir le chemin d'accès vers la page HTML dans laquelle sera générée l'animation
- « id » : c'est l'identifiant qui permet de cibler facilement l'iframe en CSS, pour lui affecter le fameux attribut z-index.

Voyons maintenant le code source de « sketch_p5.html » :

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <script language="javascript" type="text/javascript"
    src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.8/p5.min.js"
  ></script>
  <script language="javascript" type="text/javascript"
    src="superball.js"></script>
  <style>
    body {
      padding: 0;
      margin: 0;
    }
    canvas {
      opacity: 0.5;
    }
  </style>
</head>
<body></body>
</html>
```

On voit qu'il s'agit d'une page de structure très classique, dont les éléments importants sont :

- Le chargement du framework P5.js à partir du site CDNJS (qui est une sorte de gigantesque répertoire de frameworks Javascript)
- Le chargement du script « superball.js » qui se trouve dans le même répertoire que la page en cours
- La définition de l'attribut CSS « opacity », qui définit une opacité de 0.5 pour la balise « canvas » générée par P5.js. Ceci a pour effet de créer un fond légèrement opaque. Plus proche de 1, le fond sera plus foncé, plus proche de 0, il sera plus clair.

Voici enfin le sketch P5.js à l'origine de la balle (re)bondissante :

```
// rebonds
var y = 50.0;
var x = 50.0;
```

```
var speed = 2.0;
var radius = 30;
var ydirection = 1;
var xdirection = 5;

function setup() {
  createCanvas(windowWidth, windowHeight);
  background(0); // fond noir
  smooth(); // rendu amélioré
  noStroke(); // pas de contour
  ellipseMode(RADIUS); // le centre est la référence de l'ellipse
}

function draw() {
  fill(0, 20); // couleur de remplissage pour le rectangle
                // (noir avec transparence de 20%)
  rect(0, 0, width, height);
  fill(255); // couleur de remplissage pour l'ellipse (blanc)
  ellipse(x, y, radius, radius);
  y += speed * ydirection;
  x += speed * xdirection;
  if (y > height-radius || y < radius) {
    ydirection = -ydirection;
  }
  if (x > width-radius || x < radius) {
    xdirection = -xdirection;
  }
}
```

Le site officiel du framework P5.js est le suivant :

<https://p5js.org/>

Pour rappel, le tutoriel sur P5.js se trouve ici :

<https://github.com/gregja/JSCorner>

2.5.2 Effets de transition

Quand un élément HTML passe d'un état à un autre, il est possible de changer son apparence via un effet de transition.

On peut par exemple changer progressivement la couleur d'un élément dès que l'on clique dessus, ou encore accroître légèrement la taille d'un élément que le curseur survole cet élément.

Voici les propriétés CSS3 qui sont à notre disposition :

- transition-property
 - propriété(s) sur lesquelles la transition doit s'appliquer ? (size, color, position, etc... si plusieurs propriétés, bien les séparer par une virgule)
- transition-timing-function
 - une transition lente, ou quelque chose de plus rapide ?
 - valeurs possibles : linear, ease, etc... (voir liste ci-après)
- transition-delay
 - délai (facultatif) avant de démarrage de l'effet de transition, exprimé en secondes (s) ou millisecondes (ms)
- transition-duration
 - durée de l'effet de transition, exprimé en secondes (s) ou millisecondes (ms)

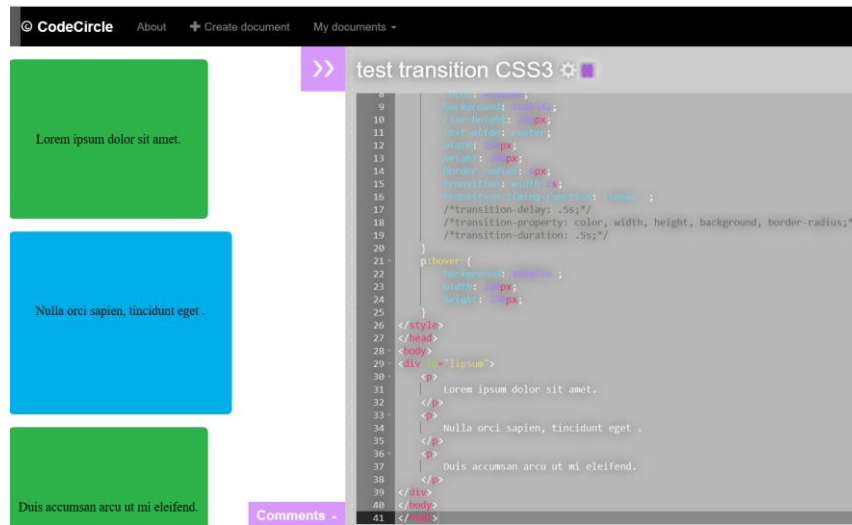
Les valeurs possibles pour la propriété « transition-timing-function » sont des paramètres que l'on retrouve dans de nombreux logiciels d'animation :

- linear : vitesse constante tout au long de la transition
- ease : début rapide et fin plus lente
- ease-in : lent au début, puis accélération jusqu'à la fin
- ease-out : rapide au début, puis ralentissement jusqu'à la fin
- ease-in-out : lent au début, accélération au milieu, puis lent à la fin

Dans l'exemple de code CSS ci-dessous, on attribue un style à plusieurs paragraphes, et on définit un effet de transition qui va se déclencher au bout de 100 millisecondes et durer 2 secondes. La transition sera de type « ease », elle va s'appliquer uniquement aux propriétés définies dans la propriété « transition-property », et elle se déclenchera dès que la souris survolera (p:hover) l'un ou l'autres des paragraphes :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Effet de transition en CSS3</title>
  <style>
    p {
      color: #000000;
      background: #2db34a;
      line-height: 200px;
      text-align: center;
      width: 250px;
      height: 200px;
      border-radius: 6px;
      transition-timing-function: ease;
      transition-duration: 2s;
      transition-delay: 100ms;
      transition-property: background, width, height;
    }
    p:hover {
      background: #00afe9 ;
      width: 280px;
      height: 230px;
    }
  </style>
</head>
<body>
<div id="lipsum">
  <p>
    Lorem ipsum dolor sit amet.
  </p>
  <p>
    Nulla orci sapien, tincidunt eget .
  </p>
  <p>
    Duis accumsan arcu ut mi eleifend.
  </p>
</div>
</body>
</html>
```

Dans CodeCircle, si on survole avec la souris le second paragraphe, il grossit progressivement et devient bleu :



Ce même paragraphe reprend sa couleur et sa forme initiale dès que l'on cesse de le survoler. Nous avons utilisé dans notre exemple le type de transition « ease », je vous invite à tester les autres types (à déguster sans modération ☺).

Pour de plus amples précisions sur les différents types de transition :

https://www.w3schools.com/cssref/css3_pr_transition-timing-function.asp

2.5.3 Effets de transformation

La norme CSS3 apporte un certain nombre d'effets de transformation 2D, et 3D, que l'on peut appliquer aux éléments d'une page.

Les effets de transformation 2D disponibles sont les suivants :

- translate
- rotate
- scale (changement d'échelle)
- skew (obliquer)
- matrix (permet de combiner tous les effets de transformation 2D en une seule instruction)

transform:translate(x,y);

déplace l'élément de X pixels vers la gauche (valeur négative) ou la droite (valeur positive), et de Y pixels vers le haut (valeur négative) ou le bas (valeur positive). On rappelle que le point de coordonnées 0,0 correspond au coin supérieur gauche de l'écran. Ne pas oublier d'indiquer « px » à droite des deux valeurs, sinon l'effet ne fonctionne pas.

Exemple :

```
transform:translate(100px, 75px);
```

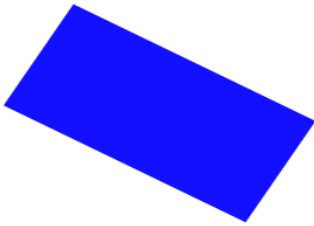


transform:rotate(deg);

Fait pivoter l'élément d'un certain nombre de degrés (valeur positive ou négative suivie de « deg »)

Exemple :

```
transform: rotate (30deg) ;
```



transform:scale(width, height);

Modifier l'échelle de l'élément en largeur et en hauteur (2 = taille doublée, 0.5 = taille / 2, etc...)

Exemple :

```
transform: scale (2, 3) ;
```



transform:skew(x-angle, y-angle);

Fait pivoter l'élément d'un certain nombre de degrés sur l'axe X et sur l'axe Y

Exemple :

```
transform:skew(30deg, 15deg);
```



On peut combiner les effets de transformation avec les effets de transition vus au chapitre précédent, mais il ne faut pas oublier dans ce cas d'ajouter la valeur « transform » à l'attribut « transition-property », pour indiquer au navigateur que la transition s'applique à la propriété « transform ».

Exemple :

```
p {
  color: #000000;
  background: #2db34a;
  line-height: 200px;
  text-align: center;
  width: 250px;
  height: 200px;
  border-radius: 6px;
  transition: 2s;
  transition-timing-function: linear ;
  transition-property: background, width, height, transform;
}
p:hover {
  background: #00afe9 ;
  width: 280px;
  height: 230px;
  transform:skew(30deg, 15deg);
}
```

Point important : on peut combiner plusieurs effets de transformation en une seule opération. Il faut simplement les séparer par un blanc.

Exemple :

```
transform: translate(300px, 200px) rotate(30deg) scale(2,3)
          skew(30deg, 15deg);
```


Matrix

La méthode `matrix()` combine toutes les transformations 2D en une seule.

Cette méthode utilise 6 paramètres, qui sont les suivants :

```
matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())
```

L'usage de cette méthode est peu commode, et pour bien comprendre son fonctionnement, on recommandera la lecture de l'article suivant :

<http://www.useragentman.com/blog/2011/01/07/css3-matrix-transform-for-the-mathematically-challenged/>

Heureusement il existe un éditeur graphique en ligne qui simplifie beaucoup les choses :

<http://www.useragentman.com/matrix/>

Voici un extrait de page HTML dans lequel on a testé la plupart des effets disponibles (sauf « matrix ») :

```
<style>
p {
    color: #000000;
    background: #2db34a;
    line-height: 200px;
    text-align: center;
    width: 250px;
    height: 200px;
    border-radius: 6px;
}
p.parag1 {
    transition-timing-function: ease;
    transition-duration: 2s;
    transition-delay: 100ms;
    transition-property: background, width, height, transform;
}
p.parag2 {
    transition-timing-function: ease;
    transition-duration: 2s;
    transition-delay: 100ms;
    transition-property: background, width, height, transform;
}
p.parag3 {
    transition-timing-function: ease;
    transition-duration: 2s;
    transition-delay: 100ms;
    transition-property: background, width, height, transform;
}
p.parag4 {
    transition-timing-function: ease;
    transition-duration: 2s;
    transition-delay: 100ms;
    transition-property: background, width, height, transform;
}
```

```

    }
    p.parag5 {
        transition-timing-function: ease;
        transition-duration: 2s;
        transition-delay: 100ms;
        transition-property: background, width, height, transform;
    }

    p.parag1:hover {
        background: greenyellow ;
        transform: translate(300px, 200px) ;
    }
    p.parag2:hover {
        background: darkseagreen ;
        transform: rotate(30deg);
    }
    p.parag3:hover {
        background: mediumvioletred ;
        transform: scale(2,3);
    }
    p.parag4:hover {
        background: darkblue ;
        transform: skew(30deg, 15deg);
    }
    p.parag5:hover {
        background: darkblue ;
        transform: translate(300px, 200px) rotate(30deg) scale(2,3)
                        skew(30deg, 15deg);
    }
</style>

<div id="lipsum">
    <p class="parag1">
        translate(300px, 200px)
    </p>
    <p class="parag2">
        rotate(30deg);
    </p>
    <p class="parag3">
        scale(2,3);
    </p>
    <p class="parag4">
        skew(30deg, 15deg);
    </p>
    <p class="parag5">
        les 4 effets combinés
    </p>
</div>

```

Rotation 3D

On peut appliquer des effets de rotation selon les 3 axes X, Y et Z :

```
transform: rotateY(deg) ;  
transform: rotateX(deg) ;  
transform: rotateZ(deg) ;  
transform: rotate3d(x, y, z) ;
```

Nous ne nous attarderons pas sur le fonctionnement de la rotation3D, qui ne présente pas de difficulté particulière, mais je vous invite à la tester. Pour de plus amples informations sur ce sujet :

https://www.w3schools.com/css/css3_3dtransforms.asp

Pour clore ce chapitre sur les effets de transformation, il faut souligner qu'il existe des disparités entre les navigateurs, au niveau du rythme d'adoption des spécifications. A la décharge des éditeurs de navigateurs, on peut ajouter que certaines spécifications du W3C sont demeurées à l'état de brouillon (draft) relativement longtemps, laissant planer un doute sur le fonctionnement de certaines fonctions CSS, jusqu'à publication de la spécification définitive (certaines d'entre elles ont été publiées en 2014, ou plus tard). Pendant une période transitoire, les éditeurs de navigateurs ont implémenté certaines fonctionnalités en les préfixant par précaution.

Voici les préfixes utilisés par chaque famille de navigateur :

- Pour Chrome et Safari : -webkit-
- pour Firefox : -moz-
- pour Opera : -o-
- pour IE : -ms-

Exemple d'utilisation :

```
div {  
  -ms-transform: translate(50px, 100px); /* IE 9 */  
  -webkit-transform: translate(50px, 100px); /* Safari */  
  transform: translate(50px, 100px); /* Fonction standard */  
}
```

Nous n'avons fait que survoler les possibilités offertes par CSS3 en matière d'effets de transition et de transformation, voici quelques conseils de lecture pour approfondir ces sujets :

https://www.w3schools.com/css/css3_2dtransforms.asp

<https://developer.mozilla.org/fr/docs/Web/CSS/transform-function>

https://www.w3schools.com/css/css3_3dtransforms.asp

<https://www.alsacreations.com/article/lire/1418-css3-transformations-2d.html>

2.5.4 Effets d'animation

La norme CSS3 apporte la possibilité d'effectuer des effets d'animation en s'appuyant sur la notion de « keyframes ». Parfaitement compatible avec les effets de transition et de transformation vus précédemment, cette technique de « keyframes » permet d'obtenir des effets plus fluides et mieux contrôlés dans le temps. Cette technique fait le bonheur des webdesigners, mais comme elle est très simple à utiliser, on ne va pas s'en priver ☺.

L'exemple que nous allons étudier a été emprunté à l'excellent magazine Webdesign (numéro de juin-juillet 2017).

Le principe est le suivant : nous souhaitons obtenir un effet de tremblement sur les éléments d'une liste, quand nous les survolons avec la souris. Nous allons pour notre test, réutiliser notre fameuse liste de langages utilisée précédemment, nous allons simplement ajouter à cette liste une classe CSS que nous appellerons « tremblement » :

```
<ol class="tremblement">
  <li>java</li>
  <li>javascript</li>
  <li>scala</li>
  <li>haskell</li>
  <li>go</li>
  <li>rust</li>
  <li>julia</li>
  <li>basic</li>
  <li>pascal</li>
  <li>fortran</li>
  <li>c</li>
  <li>c++</li>
  <li>c#</li>
  <li>ruby</li>
  <li>python</li>
</ol>
```

Au niveau du code CSS, nous avons besoin de créer une classe « tremblement », et d'indiquer ce qui va se passer quand nous survolerons chacun des enfants de la liste rattachée à cette classe.

```
.tremblement > * {  
    position: relative;  
    color: blueviolet;  
}  
.tremblement > *:hover {  
    animation: xtremblement .15s infinite;  
    color: red;  
}  
@keyframes xtremblement {  
    0% {left: 0;}  
    25% {left: -5px;}  
    75% {left: 5px;}  
    100% {left: 0;}  
}
```

L'effet qui a pour nom « xtremblement » dure 150 millisecondes : à 0 % du temps, l'élément est à sa position initiale, à 25% du temps, l'élément se décale de 5 pixels vers la gauche, etc...

Dans l'exemple ci-dessus, nous avons utilisé le raccourci consistant à définir l'ensemble des paramètres de l'animation sur une seule ligne :

```
animation: xtremblement .15s infinite;
```

... mais on peut aussi détailler chacun des paramètres, ligne à ligne, avec les propriétés suivantes :

- animation-timing-function
 - une transition lente, ou quelque chose de plus rapide ?
 - valeurs possibles : linear, ease... soit les mêmes que pour les effets de transition (cf. chapitre spécifique).
- animation-delay
 - délai (facultatif) avant de démarrage de l'animation, exprimé en secondes (s) ou millisecondes (ms)
- animation-duration
 - durée de l'animation, exprimé en secondes (s) ou millisecondes (ms)
- animation-iteration-count
 - nombre de répétitions de l'animation, par défaut « jouée » qu'une seule fois, peut être paramétré à « infinite »
- animation-direction :
 - définir à « reverse » pour rejouer l'animation en sens inverse
- animation-fill-mode :
 - avec la valeur « forwards », l'objet conserve l'état final produit par l'animation, avec « backwards », il revient à son état initial

Voici un autre exemple que je vous invite à tester. Il consiste à enchaîner plusieurs effets d'animation sur une « div », dès que l'on clique sur cette « div ». Ne lâchez pas le bouton lors du clic, sinon l'effet s'arrête avant la fin :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Effet de animation en CSS3</title>
  <style>
    p {
      color: #000000;
      background: #2db34a;
      line-height: 200px;
      text-align: center;
      width: 250px;
      height: 200px;
      border-radius: 6px;
    }
    p.parag1 {
      animation-timing-function: ease;
      animation-duration: 2s;
      animation-delay: 100ms;
      animation-property: background, width, height, transform;
    }
    p.parag1:active {
      background: greenyellow ;
      animation: multieffet 10s ;
    }

    @keyframes multieffet {
      0% {transform: translate(0px, 0px) ;}
      15% {transform: translate(300px, 200px) ;
          background: dimgrey;}
      25% {transform: rotate(30deg);}
      50% {transform: scale(2,3)}
      75% {transform: skew(30deg, 15deg)}
      100% {transform: translate(0px, 0px) ;}
    }
  </style>
</head>
<body>
<div id="lipsum">
  <p class="parag1">
    Cliquez-moi et ne me lâchez pas
  </p>
</div>
</body>
</html>
```

On peut utiliser cette technique d'animation CSS pour créer un effet de type « bandeau défilant », comme celui présenté dans le tutoriel suivant :

<http://labs.viaxoft.com/texte-defilant-en-css/>

Quelques liens pour approfondir le sujet :

https://www.w3schools.com/css/css3_animations.asp

[https://developer.mozilla.org/fr/docs/Web/CSS/Animations CSS/Utiliser les animations CSS](https://developer.mozilla.org/fr/docs/Web/CSS/Animations_CSS/Utiliser_les_animations_CSS)

<https://www.alsacreations.com/article/lire/1418-css3-transformations-2d.html>

<https://openclassrooms.com/courses/utilisez-les-effets-avances-de-css-sur-votre-site/les-animations-css-1>

2.5.5 Flexbox

La norme CSS évolue, et de nouvelles fonctionnalités apparaissent régulièrement.

Parmi les nouveautés récentes qui sont particulièrement attractives, il y a Flexbox.

Hormis IE qui ne sait gérer cette fonctionnalité qu'à partir de la version 11, Flexbox bénéficie d'un support assez large par les navigateurs (cf. caniuse.com).

Extrait de la documentation MDN :

Flexbox (pour flexible box) est un [mode de mise en page](#) prévoyant une disposition des éléments d'une page de telle sorte que ces éléments possèdent un comportement prévisible lorsqu'ils doivent s'accommoder de différentes tailles d'écrans/appareils. Dans de nombreux cas, le modèle de boîte Flexbox offre une amélioration du modèle block dans lequel les flottements (float) ne sont pas utilisés, pas plus que la fusion des marges du conteneur flex avec ses éléments.

The screenshot shows a code editor with two panels. The left panel, labeled 'HTML', contains the following code:

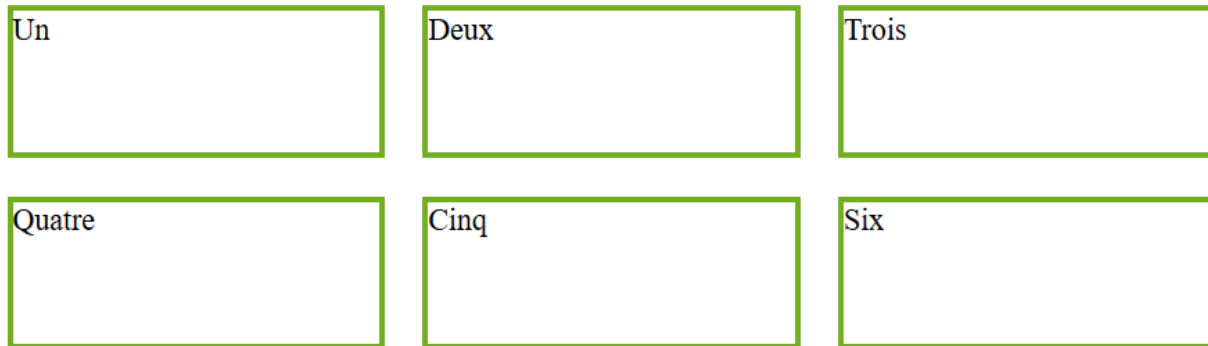
```
3 <div class="block flex">
4   <div class="one">Un</div>
5   <div class="two">Deux</div>
6   <div class="three">Trois</div>
7   <div class="four">Quatre</div>
8   <div class="five">Cinq</div>
9   <div class="six">Six</div>
10 </div>
```

The right panel, labeled 'CSS', contains the following code:

```
1 .block > * {
2   width: 150px;
3   height: 75px;
4   margin: 10px;
5   border: 3px solid #73AD21;
6 }
7
8 .flex {
9   display: flex;
10  flex-wrap: wrap;
```

Below the code editor, a visual representation of the layout is shown. It consists of six rectangular boxes with green borders, arranged in two rows. The top row contains four boxes labeled 'Un', 'Deux', 'Trois', and 'Quatre'. The bottom row contains two boxes labeled 'Cinq' and 'Six'. The boxes are arranged in a grid that demonstrates the Flexbox layout, where the first row wraps onto the second row.

Si on réduit la taille de l'écran, cela donne :



On voit que le système est très flexible, voici le code source utilisé dans l'exemple ci-dessus :

Le CSS :

```
.block > * {  
    width: 150px;  
    height: 75px;  
    margin: 10px;  
    border: 3px solid #73AD21;  
}  
.flex {  
    display: flex;  
    flex-wrap: wrap;  
    flex-direction: row;  
}  
.flex > * {  
    flex: 1 1 auto;  
}
```

Le HTML :

```
<div class="block flex">  
    <div class="one">Un</div>  
    <div class="two">Deux</div>  
    <div class="three">Trois</div>  
    <div class="four">Quatre</div>  
    <div class="five">Cinq</div>  
    <div class="six">Six</div>  
</div>
```

Pour approfondir le sujet :

https://developer.mozilla.org/fr/docs/Web/CSS/Disposition_des_bo%C3%AEtes_flexibles_CSS/Utilisation_des_flexbox_en_CSS

Un livre de référence sur le sujet, de Raphaël Goetter (paru en 2016) :

<https://www.eyrolles.com/Informatique/Livre/css-3-flexbox-9782212143638>

2.5.5 Grid

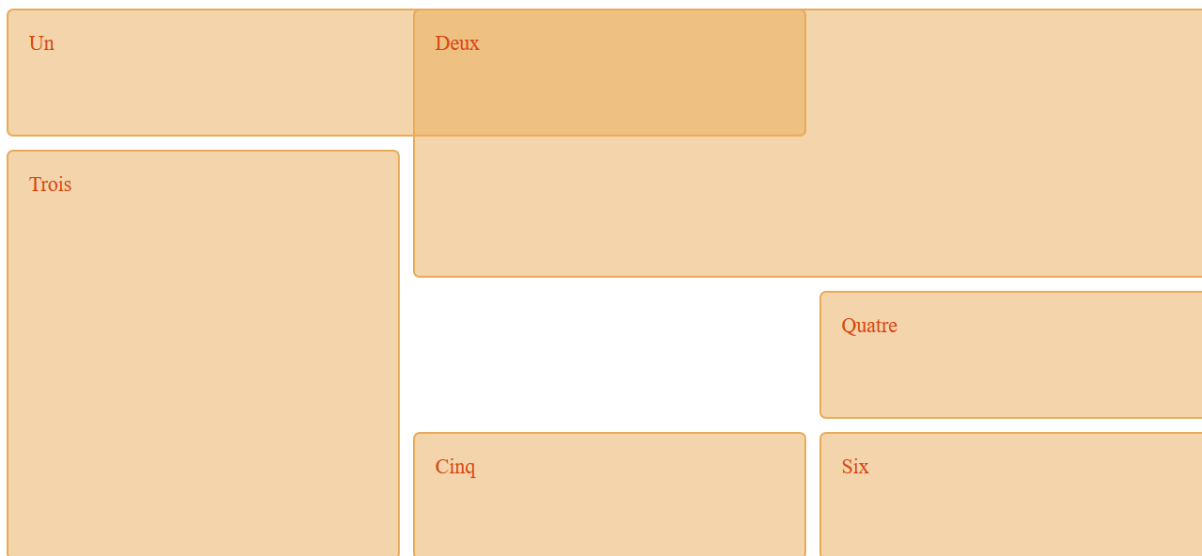
Comme Flexbox, Grid constitue une avancée majeure de CSS3. Mais il n'est pas supporté par l'ensemble des navigateurs, et il conviendra de vérifier sur Caniuse.com, la liste des navigateurs compatibles.

Extraits de la documentation MDN :

Le module **CSS Grid layout** (modèle de disposition en grille) est un module de la spécification CSS qui permet de créer des mises en page en divisant l'espace d'affichage en régions utilisables par une application ou en définissant des relations de taille, position et d'empilement entre les éléments HTML.

L'exemple présenté sur la page de MDN est révélateur de la souplesse procurée par Grid :

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



Le HTML :

```
<div class="wrapper">
  <div class="one">Un</div>
  <div class="two">Deux</div>
  <div class="three">Trois</div>
  <div class="four">Quatre</div>
  <div class="five">Cinq</div>
  <div class="six">Six</div>
</div>
```

Le CSS :

```
* {box-sizing: border-box;}
.wrapper {
    max-width: 940px;
    margin: 0 auto;
}

.wrapper > div {
    border: 2px solid rgb(233,171,88);
    border-radius: 5px;
    background-color: rgba(233,171,88,.5);
    padding: 1em;
    color: #d9480f;
}

.wrapper {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 10px;
    grid-auto-rows: minmax(100px, auto);
}

.one {
    grid-column: 1 / 3;
    grid-row: 1;
}

.two {
    grid-column: 2 / 4;
    grid-row: 1 / 3;
}

.three {
    grid-row: 2 / 5;
    grid-column: 1;
}

.four {
    grid-column: 3;
    grid-row: 3;
}

.five {
    grid-column: 2;
    grid-row: 4;
}

.six {
    grid-column: 3;
    grid-row: 4;
}
```

La mise au point de Grid peut être délicate, mais l'équipe Mozilla est en train de doter Firefox d'un outil de mise au point des Grids particulièrement intéressant. Disponible (en juillet 2017) dans la version « nightly », on devrait le voir arriver bientôt dans la version « stable » :

<https://hacks.mozilla.org/2017/06/new-css-grid-layout-panel-in-firefox-nightly/>

Plusieurs très bons tutos sont parus sur ce sujet :

<https://hacks.mozilla.org/2017/10/an-introduction-to-css-grid-layout-part-1/>

<https://hacks.mozilla.org/2017/10/an-introduction-to-css-grid-layout-part-2/>

<https://mozilladevelopers.github.io/playground/>

2.5.6 La fonction calc()

calc() est une fonction CSS3 offrant la possibilité de réaliser des opérations mathématiques en CSS (addition, soustraction, division et multiplication).

Exemple 1 :

```
div {  
    width: calc(100% - 20px);  
    padding: 10px;  
}
```

Exemple 2 :

```
.content {  
    position: relative;  
    top: calc(100px - 5rem);  
}
```

Pour approfondir le sujet :

<https://www.alsacreations.com/article/lire/1630-la-fonction-calc-en-css.html>

<https://speakerdeck.com/hteumeuleu/la-technique-des-fab-four>

3 Conclusion

On l'a vu au travers de ce cours, CSS est puissant mais parfois complexe.

Cette norme est parfois décriée pour la complexité de son modèle d'héritage, qui peut aboutir à des fichiers CSS difficilement maintenables, s'ils ne sont pas élaborés avec soin et une bonne maîtrise de la norme.

L'impossibilité de pouvoir gérer nativement des variables conduit parfois à des redondances de code qui sont problématiques en termes de maintenance. Pour contourner cette limitation, des pré-processeurs CSS tels que SASS, LESS et PostCSS ont été élaborés par des développeurs talentueux. Ces outils concurrents ont été adoptés par une partie de la communauté des webdesigners.

<http://lesscss.org/>
<http://sass-lang.com/>
<http://postcss.org/>

Si le sujet vous intéresse, vous trouverez une bonne introduction à SASS dans le hors série n° 44 du magazine Web Design.

La vidéo de Jonathan Verecchia

<http://verekia.com/slides/css-preprocessors/>

Réussir un beau design de site nécessite des compétences à la fois techniques et artistiques, qui nécessitent un long apprentissage. Ce n'est pas un hasard si des frameworks CSS comme Bootstrap ont rencontré un si grand succès auprès de la grande communauté des développeurs web. Nous verrons dans le cours consacré à Bootstrap, que ce dernier fournit des solutions et élégantes pour obtenir des pages au rendu professionnel.

4 Sites

Conseils et astuces :

<http://www.csszengarden.com/>

Outils permettant d'améliorer l'accessibilité des pages :

<http://www.intro-webdesign.com/index.html>

Assistant pour la génération de code CSS3 :

<http://css3generator.com/>

Documentations diverses :

<https://developer.mozilla.org/fr/docs/Web/css>

<https://www.w3schools.com/css/>

<http://fr.learnlayout.com/>

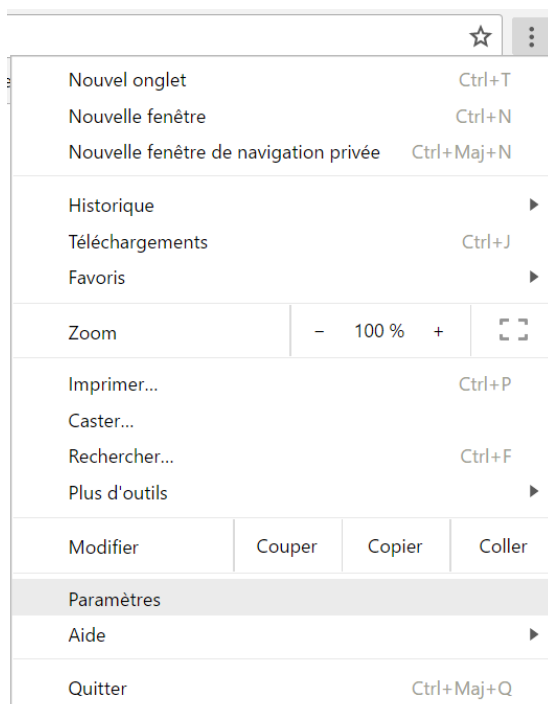
<https://www.alsacreations.com/astuce/>

5 Annexe

5.1 Vidage de cache dans Chrome

Pour obliger le navigateur de prendre en compte des modifications de code HTML, CSS ou JS, il faut vider le cache. Sous Firefox, ce vidage se fait facilement avec les touches « Control » et « F5 », mais sous Chrome il faut passer par la procédure ci-dessous :

Etape 1 : cliquer sur les 3 petits points à droite de la barre de saisie d'URL, puis sélectionner l'option « paramètres »



Etape 2 : saisir « cache » dans le filtre

Résultats de la recherche

Confidentialité

[Paramètres de contenu...](#) [Effacer les données de navigation...](#)

Google Chrome utilise parfois des services Web pour améliorer votre confort de navigation. Vous avez la possibilité de désactiver ces services. [En savoir plus](#)

- ☒ Utiliser un service Web pour résoudre les erreurs de navigation
- ☒ Utiliser un service de prédiction afin de compléter les recherches et les URL saisies dans la barre d'adresse
- ☒ Utiliser un service de prédiction pour charger les pages plus rapidement
- ☐ Signaler automatiquement les incidents de sécurité potentiels à Google
- ☒ Assurer votre protection et celle de votre appareil contre les sites dangereux
- ☐ Utiliser un service Web pour corriger les erreurs d'orthographe
- ☒ Envoyer automatiquement les statistiques d'utilisation et les rapports d'erreur à Google
- ☐ Envoyer une demande "Interdire le suivi" pendant la navigation

Etape 3 : cocher l'option « images et fichiers en cache », puis cliquer sur le bouton « effacer... »

Effacer les données de navigation ×

Conseil : Le mode navigation privée (Ctrl+Maj+N) pourra vous être utile la prochaine fois.

Effacer les éléments :

- ☐ Historique de navigation
- ☐ Historique des téléchargements
- ☐ Cookies et autres données de site et de plug-in
- ☒ Images et fichiers en cache – moins de 129 Mo
- ☐ Mots de passe
- ☐ Données de saisie automatique
- ☐ Données d'application hébergée
- ☐ Licences multimédias


[Effacer les données de navigation](#) [Annuler](#)

i Les données synchronisées seront effacées de tous les appareils. Certains paramètres qui peuvent refléter vos habitudes de navigation ne seront pas effacés. [En savoir plus](#)

5.2 Simulateur de smartphone dans Chrome

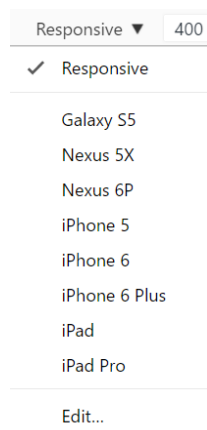
Google Chrome offre un outil permettant de simuler l’affichage de différents modèles de smartphone.

Pour y accéder, il faut passer par les outils de développement de Chrome, soit la touche F12 sur Windows et Linux (pour Mac, voir annexe 4.3).

L’icône  permet d’activer le simulateur qui apparaît dans la partie gauche du navigateur, avec le mode d’affichage « responsive » par défaut :



On peut sélectionner différents modèles de smartphones et tablettes et comparer les rendus :



5.3 Raccourcis clavier pour Google Chrome

To access the developer tools, open a web page or web app in Google Chrome. Then take one of the following actions:

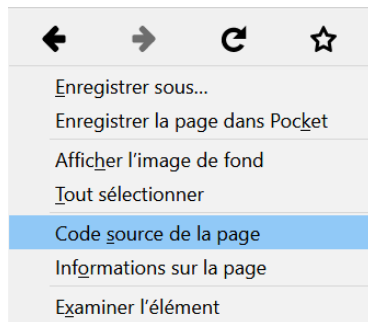
- Select the Wrench menu at the top-right of your browser window, then select Tools -> Developer tools.
- Right-click on any page element and select Inspect element.
- On Windows and Linux, press
 - Ctrl+Shift+I to open Developer Tools
 - Ctrl+Shift+J to open Developer Tools and bring focus to the Console.
 - Ctrl+Shift+C to toggle Inspect Element mode.
- On OS X, press
 - ⌘+⌘+I (Option+Command+I) to open Developer Tools
 - ⌘+⌘+J (Option+Command+J) to open Developer Tools and bring focus to the Console.
 - ⌘+⌘+C (Control+Command+C) to toggle Inspect Element mode.

<https://superuser.com/questions/119275/how-can-i-get-chrome-to-launch-and-close-developer-tools-on-mac-like-firebug>

5.4 Astuces de webdesigners

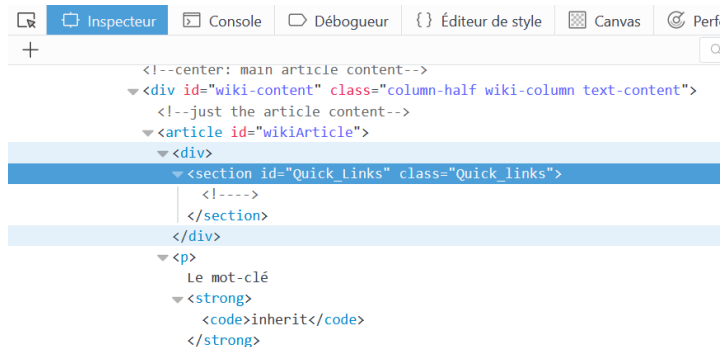
5.4.1 Ne pas confondre source initial et source « live »

Il ne faut confondre le code source initial d'une page, avec le code source « live » de cette même page. L'option « affichage du code source » d'une page permet de consulter le code source d'une page tel qu'il était au moment où le navigateur l'a reçu.



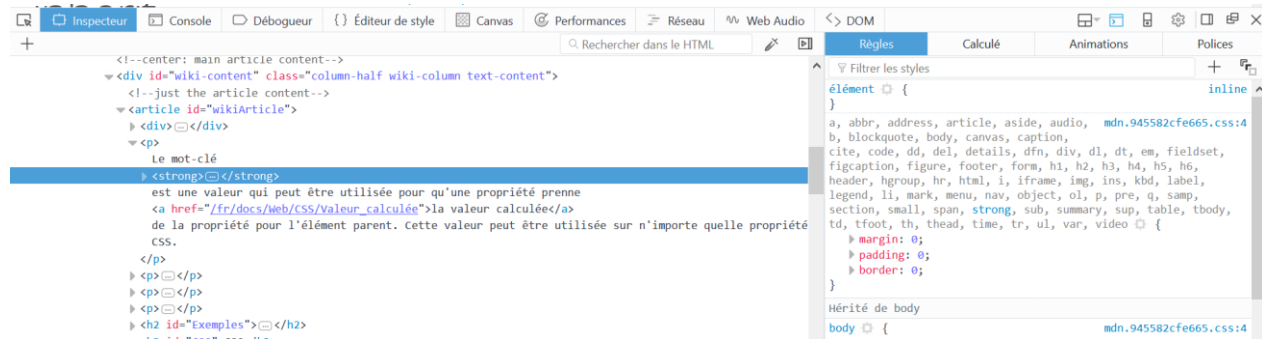
Cette option affiche le code source initial de la page.

Le code source que l'on consulte via les outils de développement, et en particulier l'option « examen d'élément », permet d'afficher une représentation du DOM (Document Object Model), soit une version « live » du code source, qui a pu être largement transformé par du code Javascript :



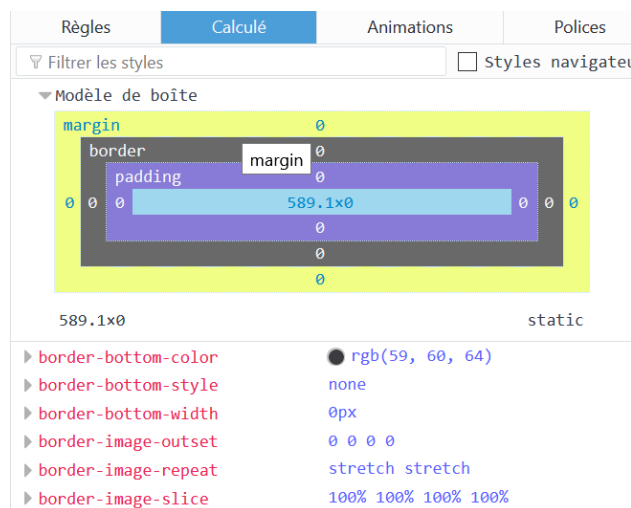
5.4.2 CSS réel et calculé

Quand on sélectionne un élément dans une page via l'inspecteur d'élément, on voit apparaître dans le cadre de droite le code CSS relatif à l'élément sélectionné :



Il est possible de modifier dynamiquement le code CSS dans le cadre de droite, pour des besoins de test et de prototypage.

Le cadre de droite met à disposition un onglet qui s'intitule « Calculé » et qui affiche une synthèse du CSS appliqué à l'élément sélectionné. Il est possible de copier-coller le contenu de ce cadre pour réutiliser ce code CSS dans une autre page si nécessaire :



5.4.3 Qui est le plus fort ?

Lors de sa présentation à Best Of Web 2017, Rémi Parmentier présentait un certain nombre de techniques CSS très intéressantes :

<https://speakerdeck.com/hteumeuleu/la-technique-des-fab-four>

Question posée par Rémi : qui gagne entre les 3 propriétés CSS ci-dessous quand elles sont utilisées conjointement et dans cet ordre ?

```
min-width :160px ;  
width :320px ; /* c'est lui qui gagne */  
max-width :480px ;
```

En cas de conflit (si max-width est inférieur à width) :

```
min-width :160px ;  
width :480px ;  
max-width :320px ; /* c'est lui qui gagne */
```

En cas d'incohérence dans le CSS :

```
min-width :480px ; /* c'est lui qui gagne (conforme spec W3C) */  
width :320px ;  
max-width :160px ;
```

Rémi Parmentier sur Twitter : @HTeuMeuLeu

6 Changelog

Version 1.2 publiée le 01/07/2017 :

- ajout de précisions dans le chapitre 2.4.2 sur les balises dépréciées, ainsi que sur les problématiques de positionnement et de centrage
- ajout de gros compléments sur le chapitre 2.4.4, lecture vivement recommandée, surtout la fin du chapitre !!!

Version 1.3 publiée le 19/10/2017 :

- Ajout de liens vers des tutos (pour le chapitre « grid ») et de références bibliographique (pour le chapitre sur Flexbox)
- Ajout dans le chapitre 2.4.2 de précisions sur les problématiques de grille (en anglais « grid »).
- Ajout au chapitre 2.4.8 d'une astuce permettant de dessiner un cercle en pur CSS (via « border-radius »)