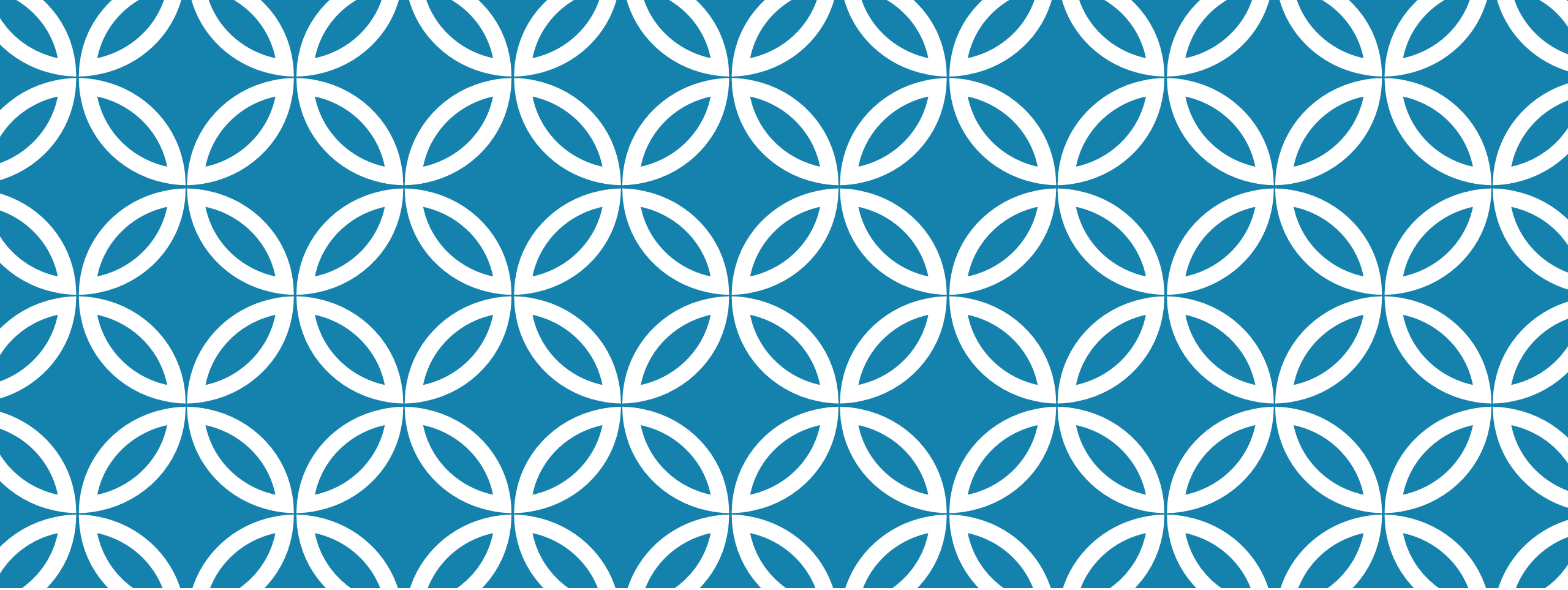


INTRODUCTION AU WEB

CSS

Par Jean-Pierre Lozi

Basé sur les cours d'Andrea Tettamanzi et Philippe Renevier

UN PROBLÈME VU EN TP : ATTENTION AUX ESPACES !

- `<p>` est valide, mais `< p >` ne l'est pas (ne va probablement pas marcher du tout) !
  - Même chose avec `< /p>` : invalide…
- De la même manière, en CSS, jamais d'espace avant les unités
  - `border-width: 10px` valide, mais `border-width: 10 px` invalide !
  - Toujours spécifier une unité, sauf pour `0` où l'unité est facultative
- Et faites bien attention aux point-virgules entre propriétés ! Par exemple :

```
a {
      color: red
      background-color: blue;
}
```

N'aura aucun effet, car le navigateur lira une seule propriété invalide :

`color: red background-color: blue;` (rouge = propriété, vert = valeur invalide)

CSS : APRÈS L'APERÇU, UN PEU D'HISTORIQUE

- Avant 1994, aucun élément de présentation dans HTML, et pas de CSS
- Vers 94-95 premiers éléments de présentation HTML comme on l'a vu (<b>, <i>, ...)
- Plusieurs propositions de feuilles de style pour séparer présentation et contenu
  - « Stylesheets for HTML » de Robert Raisch, « Stylesheet Proposal » de Pei Wei…
  - **…et « Cascading HTML Style Sheets » (CHSS) d'Håkon Wium Lie !**
- **CSS version 1.0 en 1996 !**
- Pendant de nombreuses années, compétition entre standards : en 1997, « JavaScript-Based Style Sheets » implémentées dans Netscape…
- **CSS 1.0 seulement supporté complètement par un navigateur en 2000 ! (IE 5 Mac)**

CSS : APRÈS L'APERÇU, UN PEU D'HISTORIQUE

- Chaque version de CSS inclut les précédentes (backward compatible)
- Version 1.0 de CSS en 1996 très simple : 8 sélecteurs, ~50 propriétés
- Version 2.1 de CSS en 2007… Projet CSS 2 commencé en 1998 !
  - Ajout de 11 sélecteurs et 70 propriétés… Peu supporté en 2007 (mis des années)
- CSS 3 : travail commencé sur la spécification en 1999… En parallèle à CSS 2 !
  - En cours de spécification, par module (css3-background, css3-color, css3-content…)
  - Ajout de ~20 sélecteurs, ~70 propriétés…
  - **Assez stable à l'heure actuelle, ce que nous allons utiliser !**
- Travail sur CSS 4 débuté en 2010.

CSS : APRÈS L'APERÇU, UN PEU D'HISTORIQUE

- CSS évolue très vite : rend les choses un peu compliquées
- Par exemple, les navigateurs implémentent souvent des propriétés proposées qui ne sont ensuite pas maintenues dans le standard, ou des propriétés non-standard/expérimentales qu'ils produisent eux-mêmes !
- Ce qui donne des propriétés à préfixe pour chaque moteur de rendu :
  - `-webkit-` pour Webkit (Chrome et Safari), `-moz-` pour Gecko (Firefox), `-ms-` pour Internet Explorer, `-o-`, pour Opera (anciennement), `-khtml-` pour KHTML (Konqueror)…
  - Par conséquent il faut réécrire plusieurs fois les propriétés (pour chaque préfixe)
  - Par exemple, vous verrez souvent du CSS qui ressemblera à :

```
#zoom {
    -ms-transform: scale(2);
    -moz-transform: scale(2);
    -webkit-transform: scale(2);
    transform: scale(2); // Pas valide W3C tant que la propriété non officielle…
}
```

CSS : APRÈS L'APERÇU, UN PEU D'HISTORIQUE

- L'inverse également courant : des spécification pas ou mal implémentées par certains navigateurs ! (ne soyez pas surpris)
  - C'était très courant il y a quelques années, de moins en moins
  - Tests standard pour les navigateurs pour savoir quel % de chaque spécification is supportent, par exemple, http://acidtests.org
- Etat des lieux de la spécification : http://www.w3.org/Style/CSS/current-work
- Etat des lieux des implémentations
  - https://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets)
  - http://caniuse.com (pas seulement pour CSS)
  - Sur les pages des navigateurs, par exemple : https://developer.mozilla.org/fr/docs/Web/CSS pour Firefox, http://www.opera.com/docs/specs/presto2.12/css/properties/ pour Opera…

PSEUDO-ÉLÉMÉNTS

- On a vu des pseudo-classes (`:first-child`, `:active`, `:hover`, `:focus`) : classes données automatiquement à un élément s'il répond à certaines propriétés.
- Il y a aussi des pseudo-éléménts : ce sont des « choses » auxquelles ont peut donner un style, alors qu'il ne s'agit pas vraiment d'éléments.
- `::first-line` : première ligne du texte (dépend de la taille de la fenêtre)
- `::first-letter` : première lettre du texte
- `::before` et `::after` : pour insérer un contenu généré avant ou après. Par exemple :
  `.exercice::before { content: "Exercice : "; font-weight : bold }`
  Ajoute « Exercice : » avant chaque exercice sans qu'il soit besoin de le réécrire à chaque fois.
- Bien sûr, tout utilisable ensemble (`div.exercice:hover::first-line` devrait marcher !)
- Parfois des bogues avec les pseudo-éléments… pas toujours implémentés parfaitement.

NOUVELLES PSEUDO-CLASSES CSS 3 (VOIRE 4)

- Pour une liste complète, voir, pour CSS3 : http://www.w3.org/TR/css3-selectors/
  Et pour CSS4 (support variable) : http://www.w3.org/TR/selectors4/

- Pseudo-classe : `nth-child(n)` : n^ième^ fils d'un élément.
  Utilisable avec la syntaxe : `nth-child(an+b)` : un élément qui a an+b-1 frères avant lui
  Egalement possible d'utiliser les mots-clé `odd` et `even` (impair et pair)
  Exemple :

```
tr:nth-child(2n+1) /* ligne impaire d'un tableau*/
tr:nth-child(odd) /* idem*/
```

- Pseudo-classes : `first-of-type` et `e:last-of-type` par exemple :
  `tr > td:last_of_type` : dernier `td` d'une ligne (il peut y avoir des `th` après).

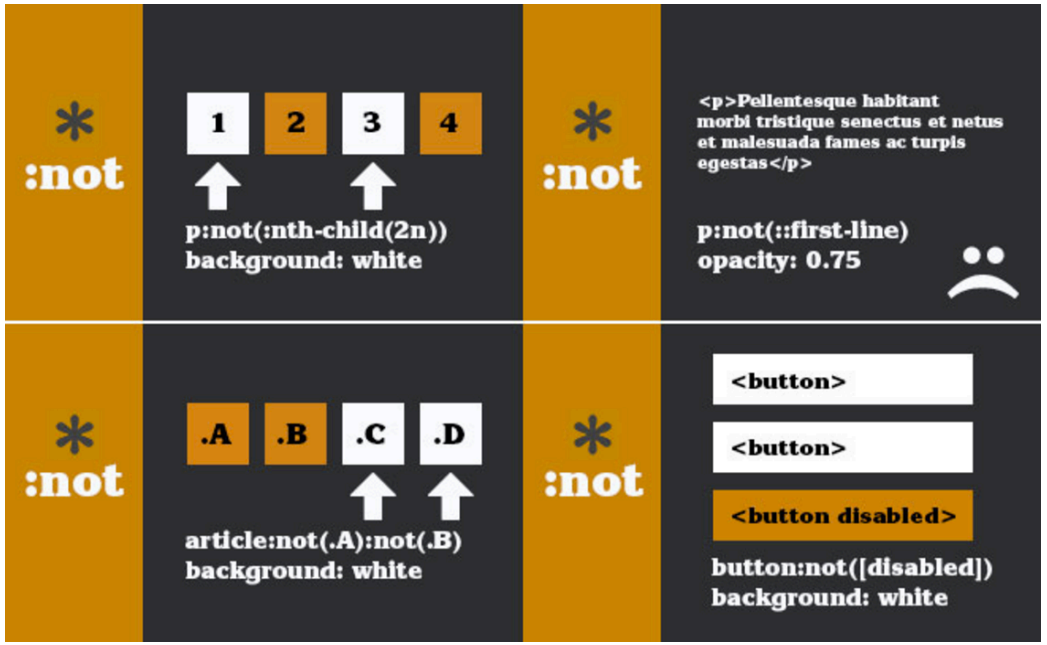
NOUVELLES PSEUDO-CLASSES CSS 3 (VOIRE 4)

- Pseudo-classe : `empty` : pour les balises vides (par exemple `<td></td>`)
- Pseudo-classe : `not(s)` : élément qui ne vérifie pas un sélecteur simple `s`

Sélecteur simple = sélecteur de type (balise HTML), sélecteur de classe (.), sélecteur #, ou une pseudo-classe

- Ne marche pas avec les pseudo-éléments !

Exemples :
(source : http://css-tricks.com)



Source : css-tricks.com

NOUVELLES PSEUDO-CLASSES CSS 3 (VOIRE 4)

- `E ~ F` : Un élément F précédé par un élément E (mais même parent)

Pas nécessairement l'un juste après l'autre !

Un exemple, quels éléments de la liste seront bleu ici ?

```
.a ~ .b {
    background-color: powderblue;
}

<ul>
    <li class="b">1st</li>
    <li class="a">2nd</li>
    <li>3rd</li>
    <li class="b">4th</li>
    <li class="b">5th</li>
</ul>
```

NOUVELLES PSEUDO-CLASSES CSS 3 (VOIRE 4)

- `E ~ F` : Un élément F précédé par un élément E (mais même parent)

Pas nécessairement l'un juste après l'autre !

Un exemple, quels éléments de la liste seront bleu ici ?

```
.a ~ .b {
    background-color: powderblue;
}

<ul>
    <li class="b">1st</li>
    <li class="a">2nd</li>
    <li>3rd</li>
    <li class="b">4th</li>
    <li class="b">5th</li>
</ul>
```

- 1st
- 2nd
- 3rd
- 4th
- 5th

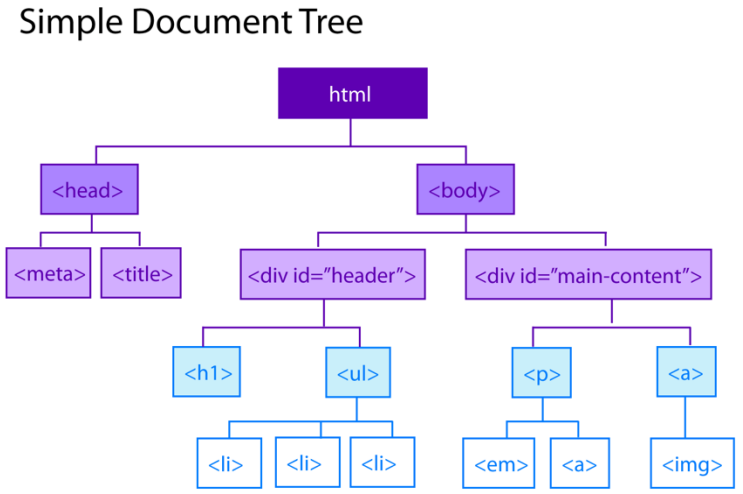
AUTRES PSEUDO-CLASSES

- `:checked` pour les cases à cocher ou les boutons radios « cochés »

Paramètre « `checked` » = coché par défaut.
Différence entre les sélecteurs `:checked` et `[checked]` ?

- `:only-child` : seul enfant
- `:only-of-type:` seul enfant de ce type
- Etc… ils deviennent assez nombreux, voir la documentation pour les autres

AFFECTATION, CASCADE ET HÉRITAGE



- Le navigateur analyse le document (arbre d'éléments HTML)
- Le navigateur calcule le style pour chaque élément
  - Pour chaque propriété (de style) possible
  - En partant du haut de l'arbre (html, puis body, … : propriétés héritées)
- Calcul en 4 étapes :
  - La valeur « spécifiée » : celle trouvée dans le CSS pour cet élément (voir slides suivants)
  - La valeur « calculée »

« Specified values are resolved to computed values during the cascade; for example URIs are made absolute and 'em' and 'ex' units are computed to pixel or absolute lengths. Computing a value never requires the user agent to render the document. »

  - La valeur « utilisée »

« Computed values are processed as far as possible without formatting the document. Some values, however, can only be determined when the document is being laid out. For example, if the width of an element is set to be a certain percentage of its containing block, the width cannot be determined until the width of the containing block has been determined. The used value is the result of taking the computed value and resolving any remaining dependencies into an absolute value. »

  - La valeur « réelle »

« A used value is in principle the value used for rendering, but a user agent may not be able to make use of the value in a given environment. For example, a user agent may only be able to render borders with integer pixel widths and may therefore have to approximate the computed width, or the user agent may be forced to use only black and white shades instead of full color. The actual value is the used value after any approximations have been applied. »

CASCADE : IDENTIFIER LES STYLES À APPLIQUER

- Trouver toutes les déclarations (i.e., toutes les propriétés pour chaque sélecteur)
- Trier selon l'origine : **auteur > utilisateur > navigateur**
  En tant qu'utilisateur, possible de spécifier une CSS dans tous les navigateur, voir:
  https://en.wikibooks.org/wiki/Cascading_Style_Sheets/User_Style_Sheets
- Trier selon les spécificités des sélecteurs :
  - Les plus spécifiques prévalent. Par exemple, avec :
    `p { color: red; } p.maclasse { color: blue; } p.maclasse:hover { color: green }`
    Un paragraphe de type `maclasse` sera vert quand on le survole avec la souris
  - Les pseudo-éléments sont considérés comme des éléments, les pseudo-classes comme des classes
- Trier par ordre d'apparition :
  - Si deux règles ont même poids, origines et spécificités, dernière arrivée l'emporte
  - Règles venant de feuilles de styles importées (avec `@import`) considérées comme étant arrivées avant celles de la feuille de style courante (directives `@import` doivent toutes être au début pour CSS valide de toute façon)

CASCADE : IDENTIFIER LES STYLES À APPLIQUER

- La spécificité est définie comme suit :
  - Est-ce un attribut style ? 0 ou 1 (=a)
  - Dans le sélecteur, compter le nombre d'attributs id (=b)
  - Puis le nombre d'attributs, classes et pseudo-classes (sauf `:not()`) (=c)
  - Et ensuite, le nombre de noms d'éléments et pseudo-éléments (=d)

- Assembler les quatre nombres a-b-c-d dans un système de nombre avec une base étendue pour obtenir la spécificité :

```
* {…}
li {…}
ol li {…}
ol li+li {…}
h1 + *[rel=prev]{…}
section ul li.liste_de_liens {…}
li.liste_de_liens[value] {…}
#boite_a_outils {…}
style="…"
```

```
/* a=0 b=0 c=0 d=0, spécificité = 0-0-0-0 */
/* a=0 b=0 c=0 d=1, spécificité = 0-0-0-1 */
/* a=0 b=0 c=0 d=2, spécificité = 0-0-0-2 */
/* a=0 b=0 c=0 d=3, spécificité = 0-0-0-3 */
/* a=0 b=0 c=1 d=1, spécificité = 0-0-1-1 */
/* a=0 b=0 c=1 d=3, spécificité = 0-0-1-3 */
/* a=0 b=0 c=2 d=1, spécificité = 0-0-2-1 */
/* a=0 b=1 c=0 d=0, spécificité = 0-1-0-0 */
/* a=1 b=0 c=0 d=0, spécificité = 1-0-0-0 */
```

CASCADE : INDENTIFIER LES STYLES À APPLIQUER

- Spécificité : voir http://specificity.keegan.st/
- Possible d'augmenter la spécificité avec `!important`

Exemple : `.message { color: blue !important; }`

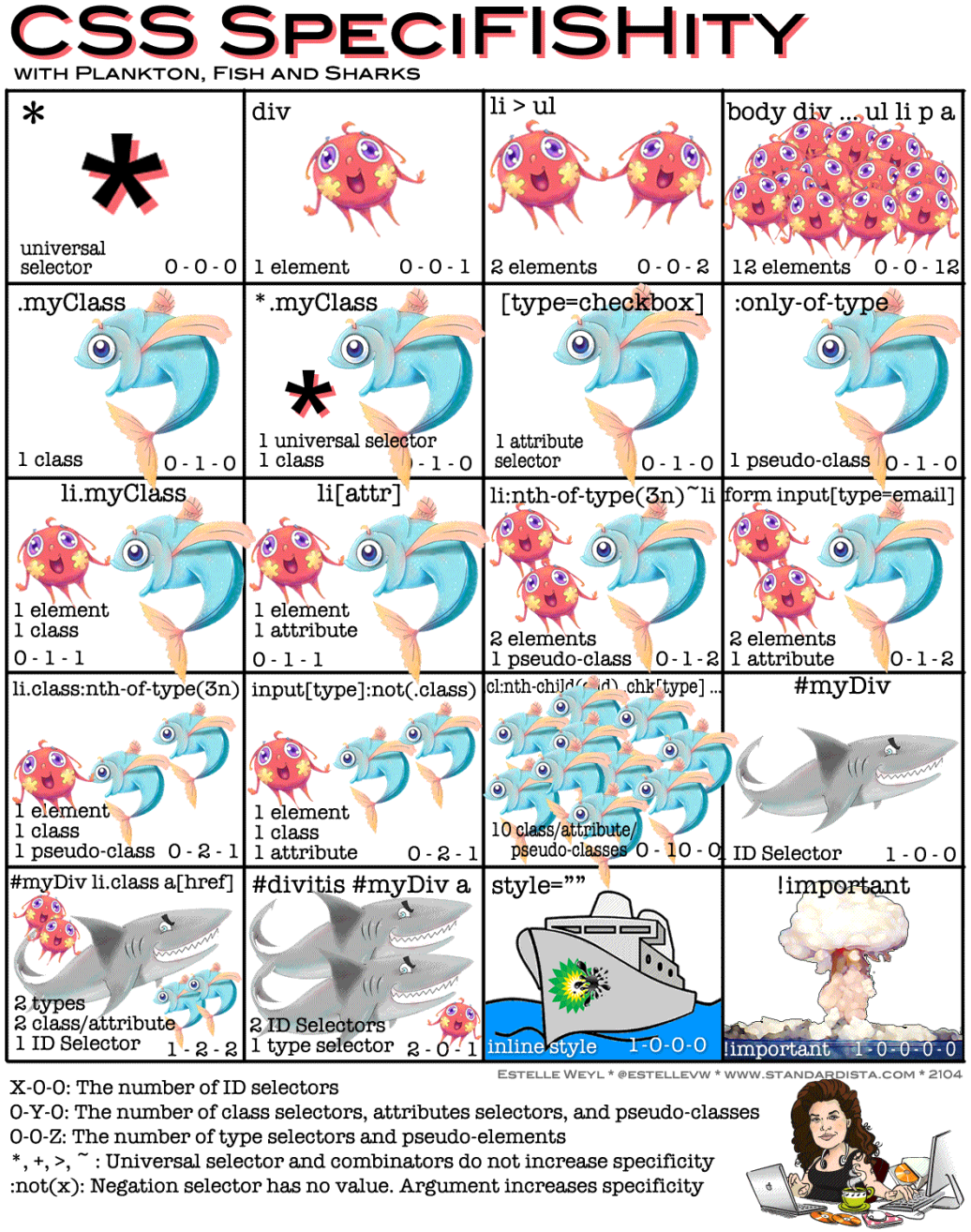
Revient à ajouter une colonne à gauche avec un 1.

CASCADE : I

- Spécificité : voir htt
- Possible d'augment

Exemple : `.message`

Revient à ajouter une



PPLIQUER

MEDIA

- Adapter la présentation au média
- Média = dispositif sur lequel le document est présenté
- Un écran (`screen`), une feuille de papier (`print`), un synthétiseur de parole (`aural`), un appareil braille (`braille`), un téléphone (`handheld`), …
- Certaines propriétés sont spécifiques à un média, par exemple :

```
@media print {
    h1 { page-break-before: always; }
}
```

- Association style – média : préciser pour quel média le style est variable

```
@import url("loudvoice.css") aural;
@media print { /* Les styles pour l'impression viennent ici. */ }
@media screen, print { body { line-height: 1.2 } }
<link rel="stylesheet" type="text/css" media="print, handheld" href="foo.css" />
```

MEDIA

- Options plus poussées : par exemple, `min-width/min-height` très utile pour faire des pages qui s'adaptent selon la taille de la fenêtre (par exemple, si la fenêtre n'est pas suffisamment large, enlever le menu avec les liens à gauche de la page et les mettre en bas…). Syntaxe :

```
@media screen and (min-width:500px) { … }
@media (min-width:500px) { … }
```

- Autres exemples :

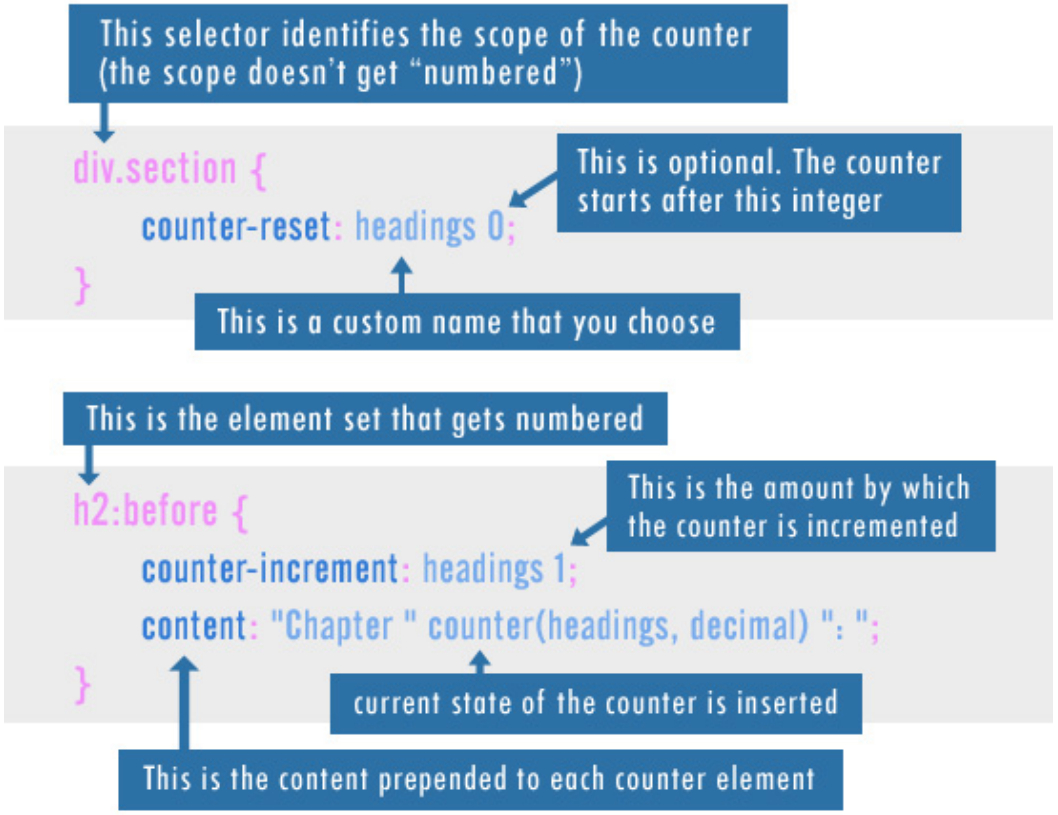
```
@media screen and (color), projection and (color) { … }
@media print and (orientation:landscape) { … }
h1, h2, h3, h4, h5, h6 {
    voice-family: paul;
    stress: 20;
    richness: 90;
    cue-before: url("ping.au")
}
```

- Voir http://www.w3.org/TR/css3-mediaqueries/

LES PROPRIÉTÉS

- Jusqu'à présent, on a vu beaucoup de propriétés « au passage », mais aucune en détail
- Maintenant : un tour d'horizon de diverses propriétés
- Liste non-exhaustive ! Il y en a beaucoup trop…
- Pour toutes les propriétés, consulter la documentation : http://www.w3.org/TR/css-2015/

LES COMPTEURS



Source : impressivewebs.com

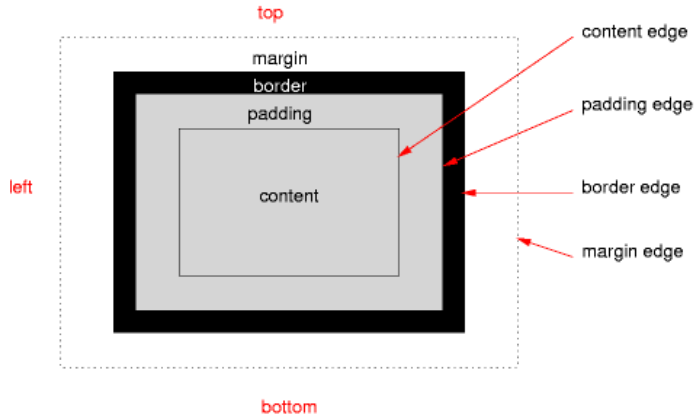
```
<div class="section">
<h2>Up to the Starting Line</h2>
<h2>A Natural Experiment of History</h2>
<h2>Collision at Cajamarca</h2>
</div>
```

Chapter 1: Up to the Starting Line

Chapter 2: A Natural Experiment of History

Chapter 3: Collision at Cajamarca

TAILLE ET MARGES



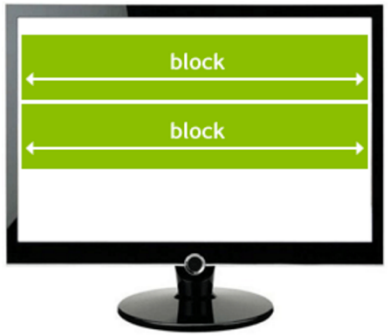
- Largeur et hauteur des éléments : `width` et `height`, taille absolue ou pourcentage. Aussi possible de spécifier `min-width` et `min-height` : largeur et hauteur minimales peuvent être étendues (si nécessaire pour le contenu, par exemple)
- Les éléments ont des marges internes et externes : `margin` représente la marge à l'extérieur de la bordure, et `padding` représente la marge à l'intérieur de la bordure
- Pas défaut, pas de bordure affichée, donc difficile de comprendre ce qui se passe parfois ! Ajouter par exemple `border: 1px solid red;` pour déboguer par exemple
- Possible de spécifier les marges extérieures avec `margin-top`, `margin-right`, `margin-left`, et `margin-bottom`… Par exemple : `margin-top: 5px;`
- Possible aussi d'utiliser juste margin avec entre une et quatre valeurs :

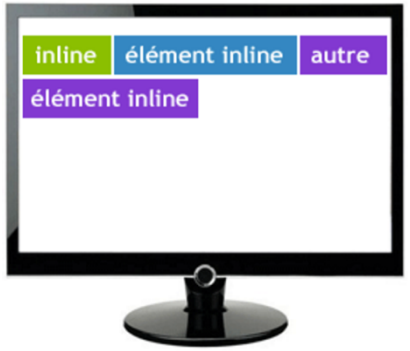
```
margin: <marge_des_quatre_côtés>;
margin: <marge_du_haut_et_du_bas> <marge_de_gauche_et_de_droite>;
margin: <marge_du_haut> <marge_de_gauche_et_de_droite> <marge_du_bas>;
margin: <marge_du_haut> <marge_de_droite> <marge_du_bas> <marge_de_gauche>;
```

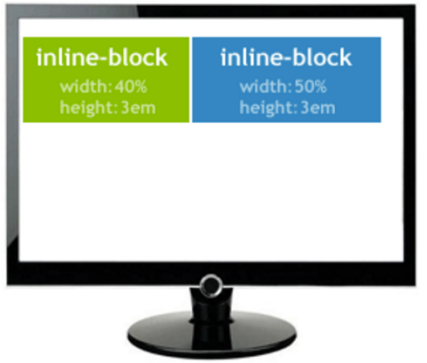
- Soit valeurs absolues, soit % de la largeur de l'élément parent… **Soit `auto`, pour les marges de gauche et de droite, ce qui permet généralement de centrer horizontalement un élément (astuce importante !)**

POSITIONNEMENT DES ÉLÉMENTS

- Trois types d'éléménts :
  - Eléments de type `inline` par défaut: `span`, `a`, `strong`, … : se comportent comme du texte : les uns à la suite des autres, de gauche à droite… N'ont pas de marges verticales, pas de largeur ou hauteur personnalisées
  - Eléments de type `block` : `div`, `h1`, `p`, `section`, … Ce sont des « briques » : prennent toute la largeur de la page, peuvent avoir des marges verticales, et peuvent avoir une largeur ou une hauteur personnalisées
  - Eléments de type `inline-block` : img, svg, math… Entre les deux : ces éléments se comportent comme du texte (les uns à la suite des autres de gauche à droite) mais peuvent avoir des dimensions !
- Possible d'utiliser `display` pour changer le type d'affichage de l'élément, par exemple `display: inline-block;`
- Aussi penser à `display: none;` pour faire disparaître un élément !
- Il y a de nombreux autres paramètres pour `display`, pour l'instant nous ne nous intéressons qu'à ces trois-là, les différences peuvent être très subtiles (lire la documentation) !







Source : alsacreations.com

POSITIONNEMENT DES ÉLÉMENTS

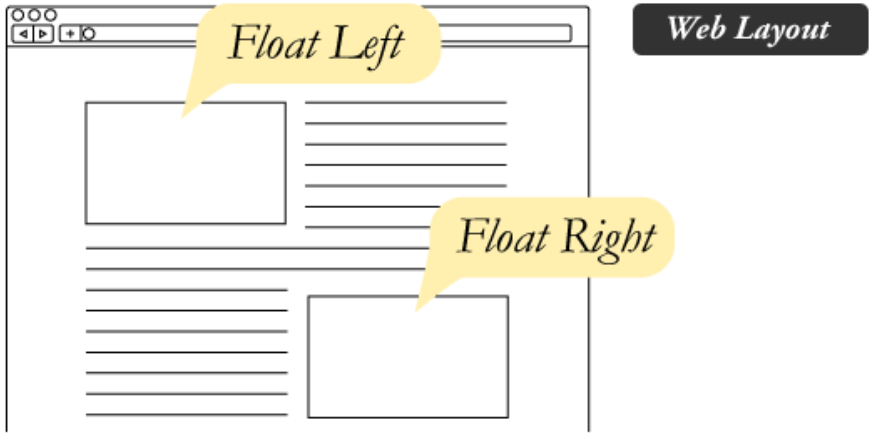
- Positionnement des éléments possible avec position, puis `top`, `left`, `bottom` et `right` :
`position: static | absolute | fixed | relative | initial | inherit;`
- Valeurs négatives possibles ! Et `fixed` = ne défile pas avec la page…

Value	Description	Play it
static	Default value. Elements render in order, as they appear in the document flow	Play it »
absolute	The element is positioned relative to its first positioned (not static) ancestor element	Play it »
fixed	The element is positioned relative to the browser window	Play it »
relative	The element is positioned relative to its normal position, so "left:20" adds 20 pixels to the element's LEFT position	Play it »
initial	Sets this property to its default value. Read about *initial*	Play it »
inherit	Inherits this property from its parent element. Read about *inherit*	

Source : w3.org

POSITIONNEMENT DES ÉLÉMENTS

- Flottement avec la propriété `float` (valeurs `none`, `left` ou `right`)
- Marche pour les `block` et `inline-block`



`none` A span element. myDIV. Another span element.

`left` myDIV. A span element. Another span element.

`right` A span element. Another span element. myDIV.

COULEURS ET FONDS

- On a déjà vu la propriété `color` : couleur du texte d'un élément

Valeur : `<couleur> | initial | inherit`
Valeur initiale : dépend du navigateur / de l'élément (souvent noire)
S'applique à : tous les éléments
Héritée : oui
Médias : visuel

- De la même manière, pour une couleur de fond : `background-color`

Valeur : `<couleur> | transparent | initial | inherit`
Valeur initiale : transparent
S'applique à : tous les éléments
Héritée : non
Médias : visuel

COULEURS ET FONDS

- Pour les fonds, en plus de `background-color`, on a `background-image`, `background-position`, `background-repeat`, `background-origin`, `background-clip`, et `background-attachment`.
- En plus de ça, on a une propriété `background` qui permet de spécifier toutes les informations des sept autres propriétés d'un seul coup !
- `background-repeat`, `background-position`, `background-size`, et `background-attachment` seulement utiles si on utilise background-image (i.e., une image pas une couleur de fond) : permettent de définir où l'image est placée, comment elle est répétée, comment elle se comporte en cas de défilement, etc.
- `background-image` permet de spécifier une image de fond, dont on passe l'url, e.g. :

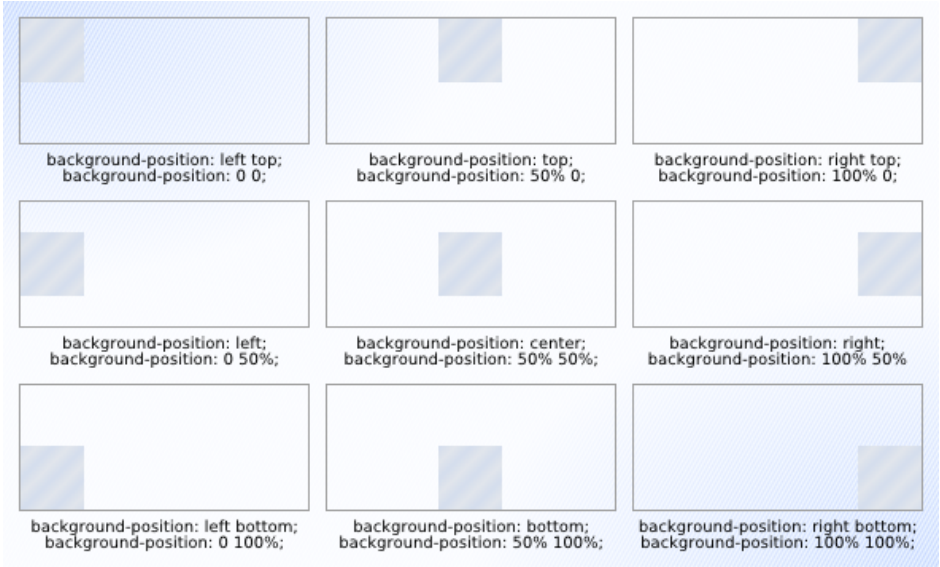
```
body { background-image: url("images/mon_fond.png") }
div.mon_menu { background-image: url("images/mon_fond.png") }
```

COULEURS ET FONDS

- `background-position` : position de l'image (qui peut ensuite être quand même répétée en x, en y, ou les deux)

Value	Description	Play it
left top left center left bottom right top right center right bottom center top center center center bottom	If you only specify one keyword, the other value will be "center"	Play it »
x% y%	The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. . Default value is: 0% 0%	Play it »
xpos ypos	The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions	Play it »
initial	Sets this property to its default value. Read about *initial*	Play it »
inherit	Inherits this property from its parent element. Read about *inherit*	

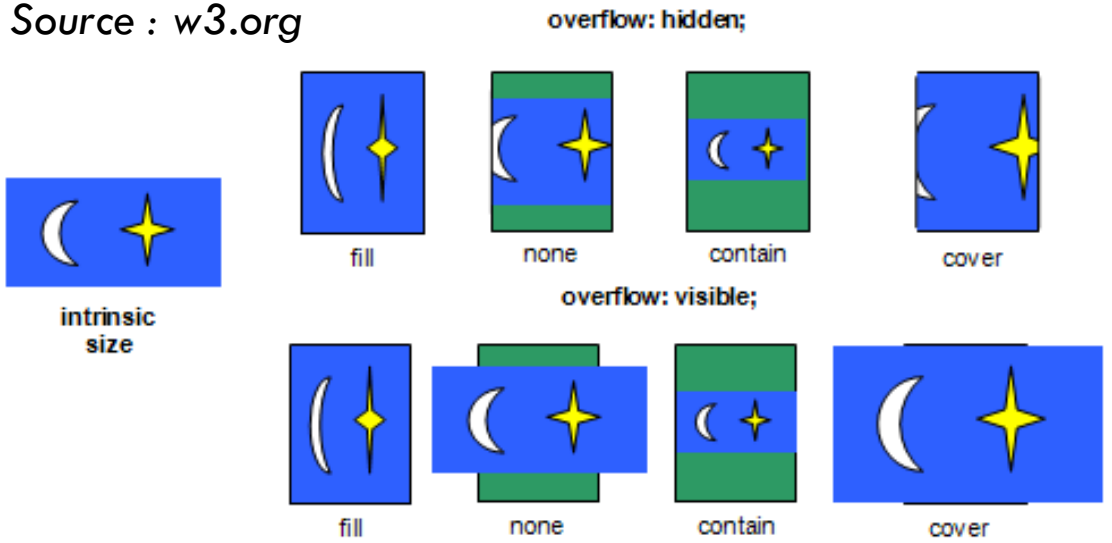
Source : w3schools



Source : slaout.linux62.org

COULEURS ET FONDS

Source : *w3.org*



- `background-size` : taille de l'image

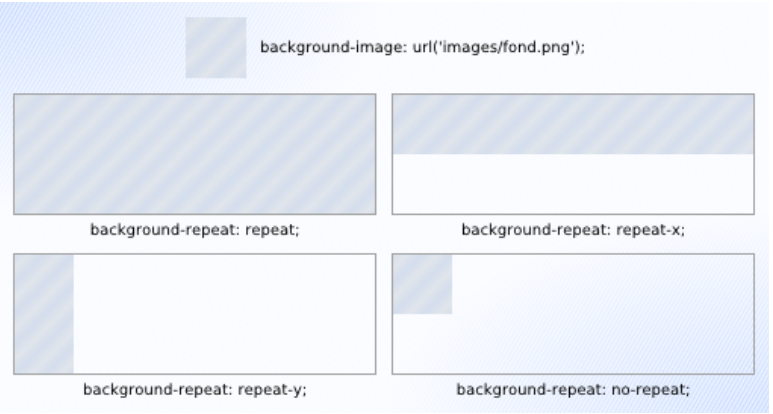
Value	Description	Play it
auto	Default value. The background-image contains its width and height	Play it »
length	Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"	Play it »
percentage	Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"	Play it »
cover	Scale the background image to be as large as possible so that the background area is completely covered by the background image. Some parts of the background image may not be in view within the background positioning area	Play it »
contain	Scale the image to the largest size such that both its width and its height can fit inside the content area	Play it »
initial	Sets this property to its default value. Read about *initial*	Play it »
inherit	Inherits this property from its parent element. Read about *inherit*	

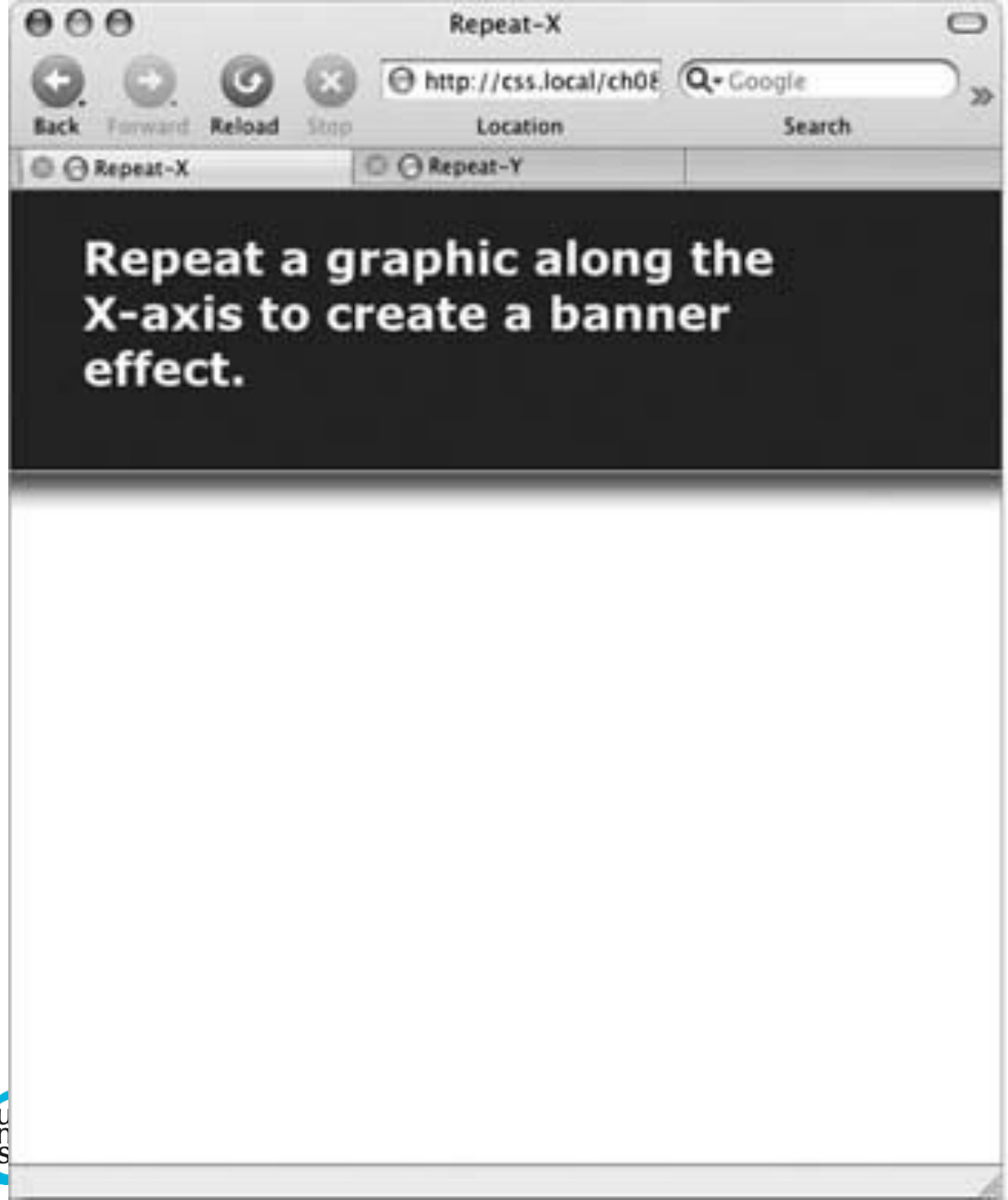
Source : *w3schools*

COULEURS ET FONDS

- `background-repeat` :
`repeat | repeat-x | repeat-y | no-repeat | initial | inherit`
- `initial` et `inherit` : toujours la même chose
- `no-repeat` : par défaut, image juste une seule fois dans le coin en haut à gauche… Peut marcher si image fondue vers la couleur de fond par exemple, sinon problématique car si quelqu'un a un très grand écran, il pourra voir les coins de l'image
- `repeat` : répète l'image en X et en Y indéfiniment, utile pour les textures, `repeat-x` : répète l'image horizontalement seulement, `repeat-y` horizontalement

Source : slaout.linux62.org



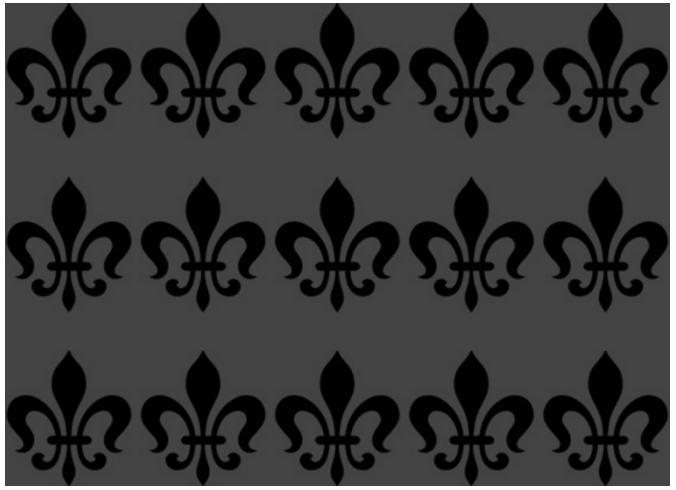

Repeat-X
Back
Forward
Reload
Stop
http://css.local/ch0E
Location
Google
Search
Repeat-X
Repeat-Y
Repeat a graphic along the X-axis to create a banner effect.


Repeat-Y
Back
Forward
Reload
Stop
http://css.local/ch0E
Location
Google
Search
Repeat-X
Repeat-Y
Instant sidebar with repeat-y

COULEURS ET FONDS

- Nouvelles options CSS 3 pour `background-repeat: space` et `round`
- Pour éviter que la dernière image ne soit coupée : `space` met des espaces entre les images, et `round` les étire…

`repeat`



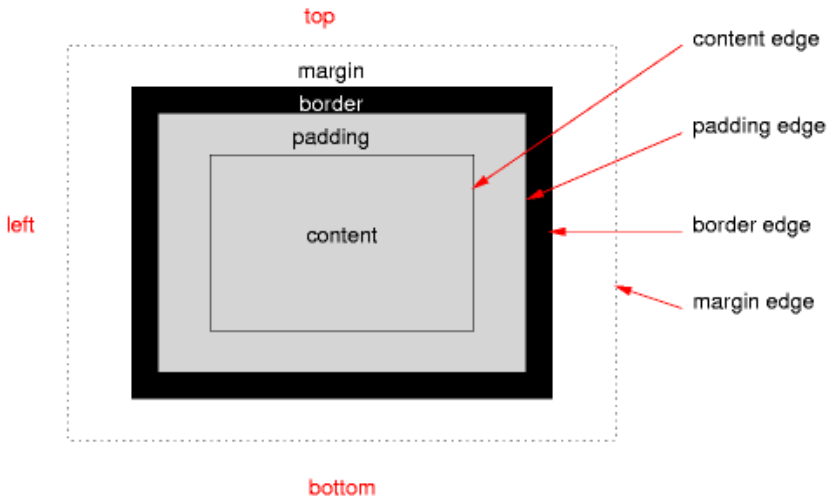
`space`

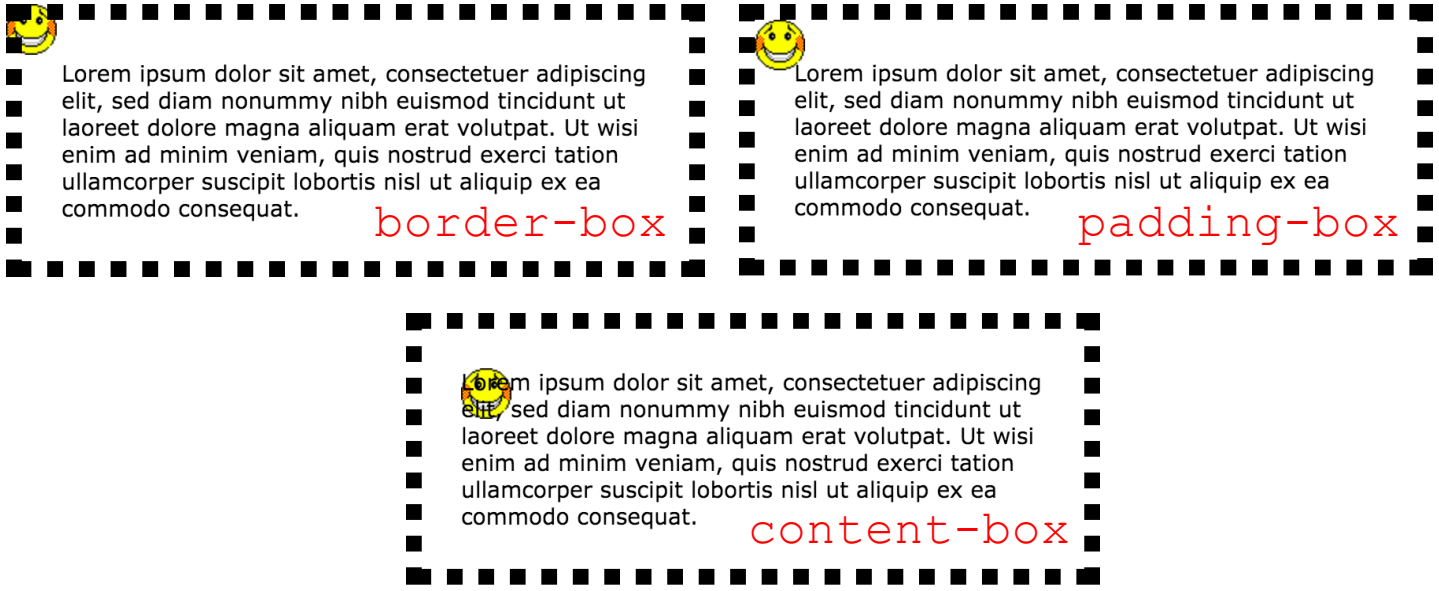
`round`

COULEURS ET FONDS

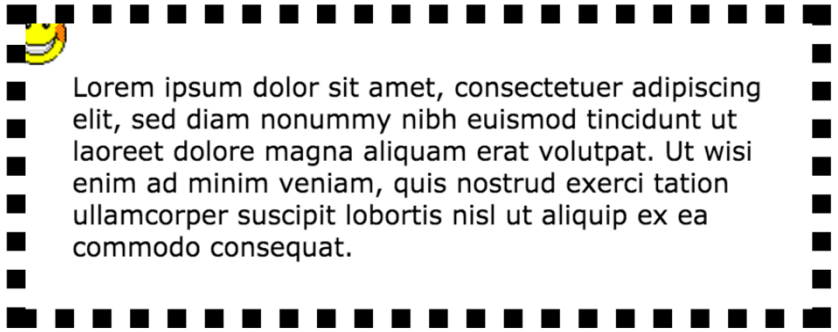
- `background-origin` : permet de définir « où commence » le fond, i.e, où est son coin en haut à gauche ? Sous la bordure, juste après la bordure, au niveau du texte ? (ou : origine du coin en haut à gauche pour background-position)

```
border-box | padding-box | content-box | initial | inherit;
```



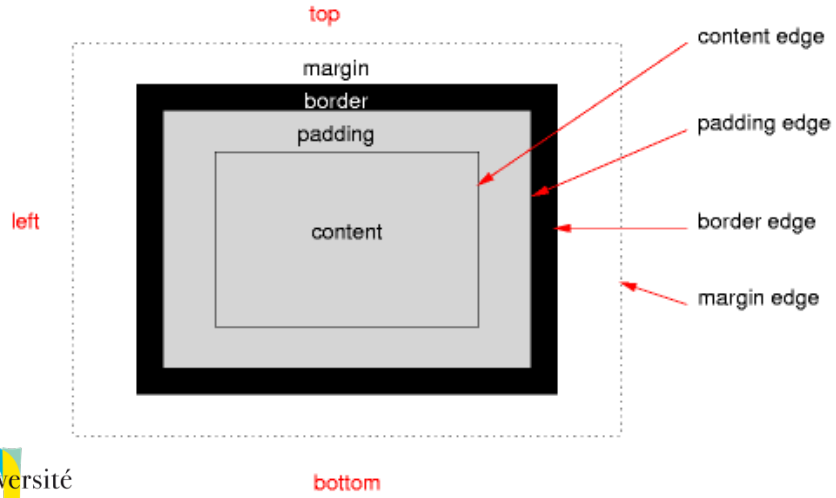


COULEURS ET FONDS



- `background-clip` : permet de définir la zone où est dessiné le fond : mêmes valeurs que pour background-position mais différent : on peut dire que l'origine est la `border-box`, mais ne dessiner le fond que dans la `padding-box` par exemple, ce qui fera que le fond sera coupé

```
border-box | padding-box | content-box | initial | inherit;
```



COULEURS ET FONDS

- `background-attachment` : permet de configurer le comportement du fond quand on fait défiler. Par défaut, `scroll`, défile avec la page. Possible d'utiliser `fixed` : le fond restera en place lorsqu'on défile. Seuls les éléments de la page (texte, etc.) défileront. Autre option `local` : utile pour un élément dans la page, fond défilera avec le contenu de l'élément, pas la page entière (regarder des démos en ligne, difficile à comprendre sans l'essayer)
- Et finalement, `background` regroupe tout (pas besoin de tout spécifier).

```
background: color image position/size repeat origin clip attachment;
```

Note: *If one of the properties in the shorthand declaration is the background-size property, you must use a / (slash) to separate it from the background-position property, e.g.* `background:url(smiley.gif) 10px 20px/50px 50px;` *will result in a background image, positioned 10 pixels from the left, 20 pixels from the top, and the size of the image will be 50 pixels wide and 50 pixels high.*

Exemple : `background: #00ff00 url("smiley.gif") no-repeat fixed center;`

COULEURS ET FONDS

- Egalement possible de d'utiliser `background-image` pour spécifier des dégradés, grâce aux fonctions `linear-gradient()` et `radial-gradient()`, qui génèrent une image contenant un dégradé dont la taille s'adapte à l'élément à laquelle elle est appliquée.
- `linear-gradient()` pour un dégradé linéaire avec un axe et des couleurs :

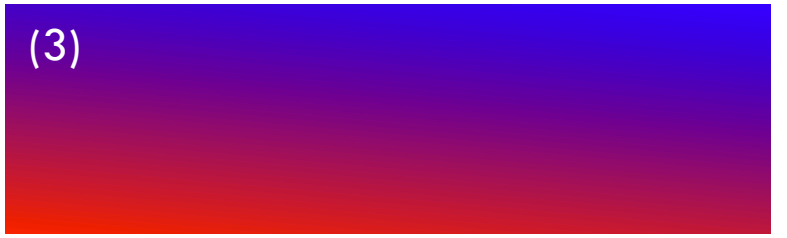
```
linear-gradient( [ [ <angle> | to <side-or-corner> ,]?
                   <color-stop> [, <color-stop>]+ )
```

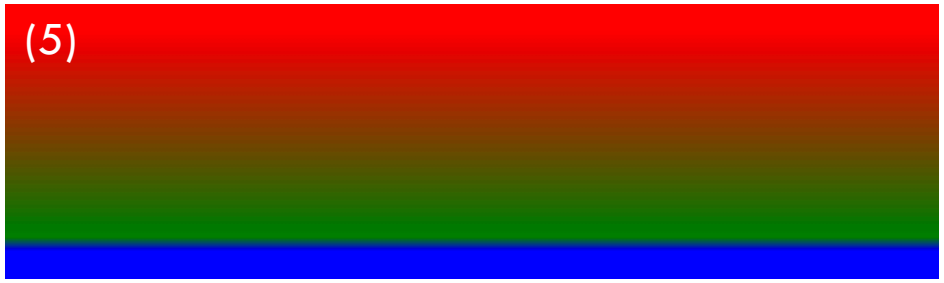
- La direction : (a) soit un côté vers où va le dégradé (`to top`, `to left`, `to bottom` ou `to right`), (b) soit un coin vers où va le dégradé (`to top left`, `to bottom right`, …), soit un angle en degrés (e.g., `25deg`). Si non spécifié : de haut en bas.
- Ensuite, une suite de couleurs, avec si on le veut un pourcentage. Dit à quel % de la page on passe à la couleur suivante…
- **Couleurs transparentes possibles !**

COULEURS ET FONDS

- Exemples de `linear-gradient()` :

```
(1) background: linear-gradient(red, blue); /* Même chose que (to bottom, red, blue) */
(2) background: linear-gradient(to bottom right, red , blue);
(3) background: linear-gradient(15deg, red, blue);
(4) background: linear-gradient(red, orange, yellow, green, blue, indigo, violet);
(5) background: linear-gradient(red 10%, green 85%, blue 90%);
```

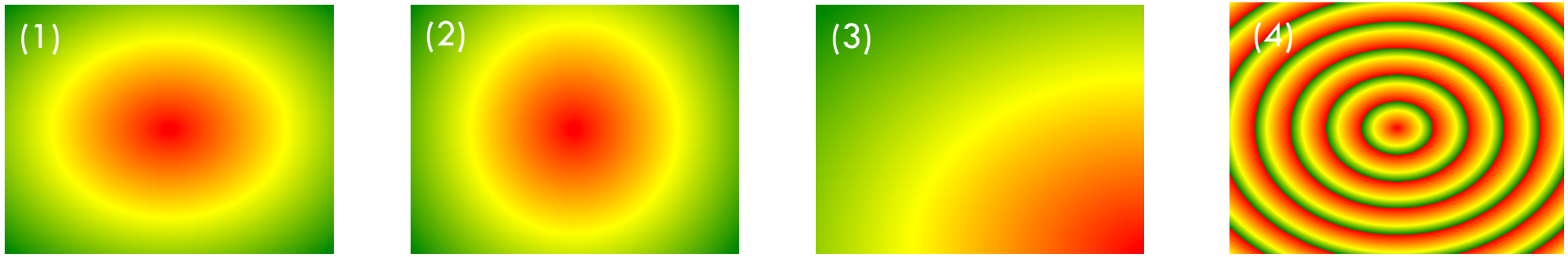




COULEURS ET FONDS

- `radial-gradient()` pour un dégradé en ellipse ou en cercle, par exemple :

```
(1) background: radial-gradient(red, yellow, green);
(2) background: radial-gradient(circle, red, yellow, green);
(3) background: radial-gradient(at bottom right, red, yellow, green);
```



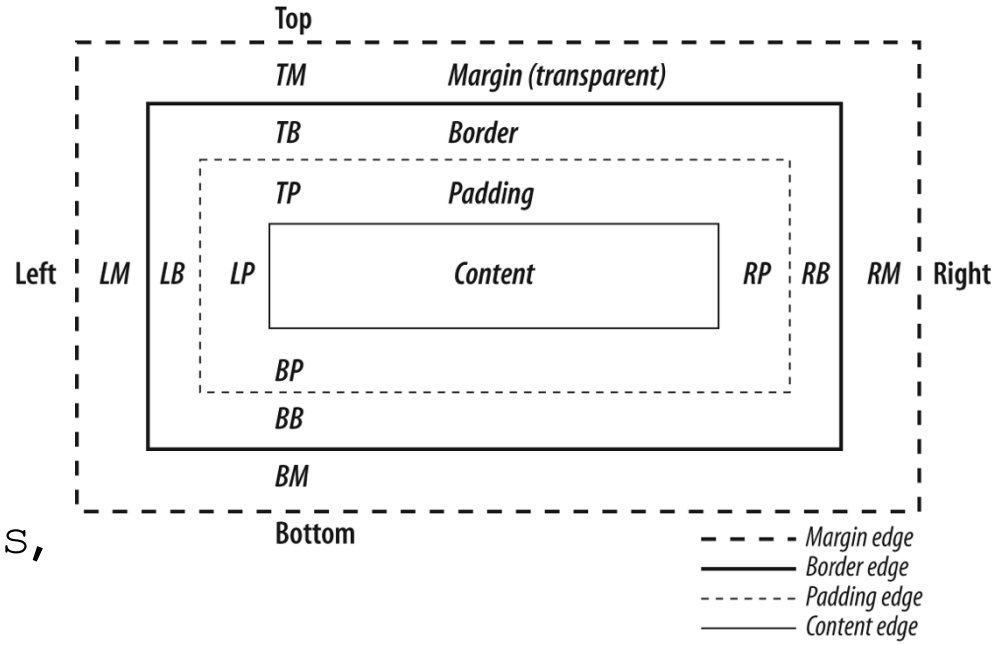
- Pas mal d'autres options, par exemple pour choisir en détail la forme et la taille de l'ellipse… Voir la documentation officielle.

- Possible de faire des dégradés qui se répètent avec `repeating-linear-gradient()` et `repeating-radial-gradient()`… Par exemple :

```
(4) background: repeating-radial-gradient(red, yellow 10%, green 15%);
```

BORDURES

- Bordures : le « contour » des éléments.
- Même idée que pour fonds. Plein de propriétés `border-*`, `border` en regroupe certaines.
- Les propriétés principales de bordure : `border-color`, `border-style`, `border-width`, `border-radius`, `border-image`
- Et à chaque fois, raccourci pour spécifier une propriété dans l'une des 4 dimensions/coins… deux valeurs pour chaque coin pour `border-radius`!
- Et pour `border-image`, cinq sous-propriétés
- Au final on se retrouve avec des propriétés du genre `border-bottom-color`, `border-top-left-radius`, `border-image-source`, `border-image-repeat`…



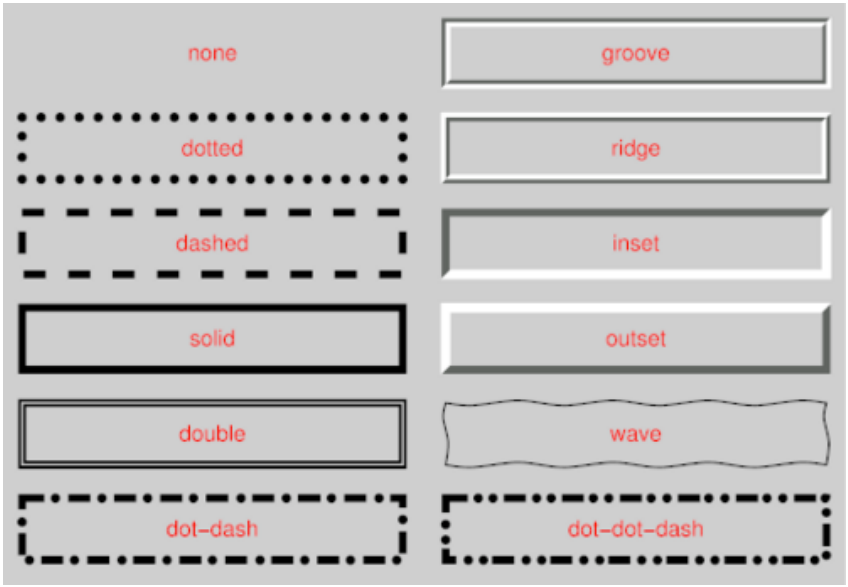
BORDURES

- Style d'une bordure : `border-style-top`, `border-style-right`, `border-style-bottom`, `border-style-left`
- Valeurs possibles :

`none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset`

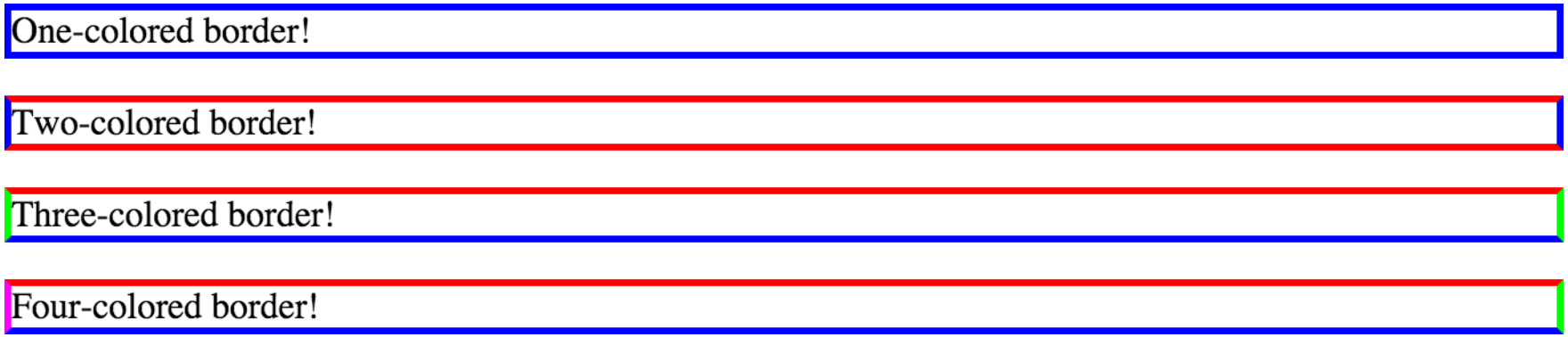
- Possible d'utiliser `border-style`, entre 1 et 4 valeurs :

```
border-color:  <style_de_toute_la_bordure>;
border-color:  <style_du_haut_et_du_bas>
               <style_de_gauche_et_de_droite>;
border-color:  <style_du_haut>  <style_de_gauche_et_de_droite>
               <style_du_bas>;
border-color:  <style_du_haut>  <style_de_droite>
               <style_du_bas>  <style_de_gauche>;
```



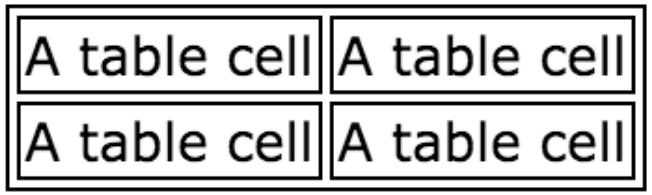
BORDURES

- Couleur d'une bordure : `border-color-top`, `border-color-right`, `border-color-bottom`, `border-color-left`, ou juste `border`
- Possible aussi d'utiliser `transparent` (même chose qu'une couleur avec canal alpha à 1.0)
- Propriété non héritée (donc si bordure, éléments internes n'en ont pas automatiquement, heureusement)
- Attention : `border-color` ne fait rien sans style spécifié, car par défaut on a la propriété `border-style: none`

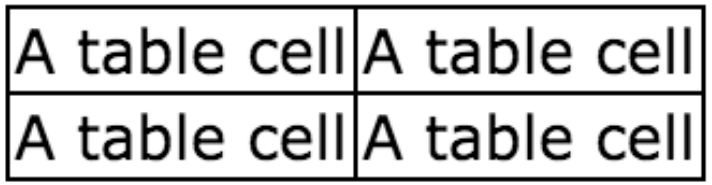


BORDURES

- Epaisseur d'une bordure : `border-width`. Valeurs : `thin`, `medium`, `thick`, ou une longueur (par exemple, `5px`). Toujours pareil : on a `border-top-width`, `border-right-width`, `border-bottom-width`, et `bottom-left-width`.
- Possible d'utiliser juste `border` pour spécifier d'un coup `border-width`, `border-style` et `border-color` (il peut en manquer)… Par exemple `border: 1px dotted red;` ou `border: dashed blue;` Et bien sûr on peut utiliser `border-top`, `border-right`, `border-bottom`, et `border-left`…
- Pour les bordures d'un tableau, on a `border-collapse` : avec `separate` (par défaut) les bordures des cellules ne sont pas fusionnées, avec `collapse`, elles sont fusionnées.



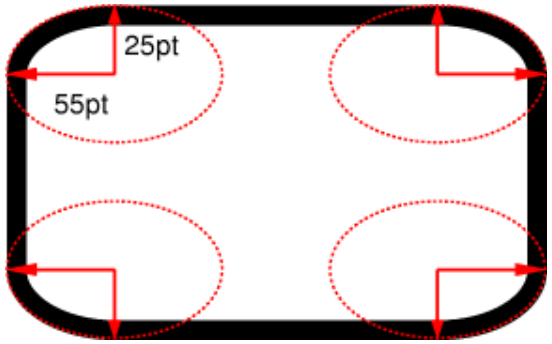
`border-collapse : separate`



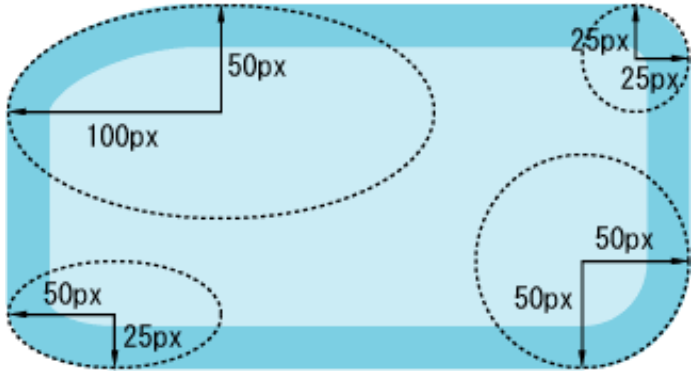
`border-collapse : collapsed`

BORDURES

- Coins arrondis avec `border-radius` : s'applique aux contours (`border`), au fond (`background`) et aux ombres (`box-shadow`)
- Réduit la taille du contenu : des choses peuvent dépasser : il faut utiliser suffisamment de padding pour éviter tout débordement



Source : w3.org



Source : htmq.com

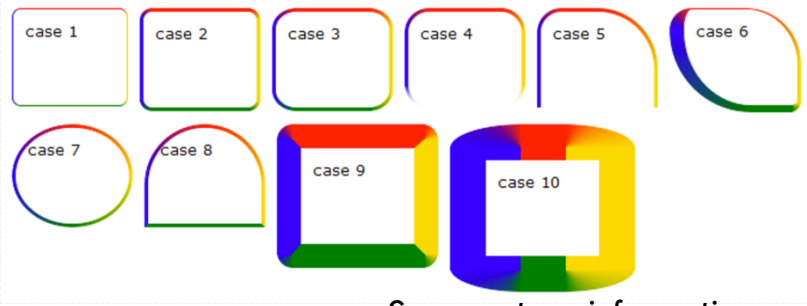
Source : w3.org

- Une propriété pour chaque coin : `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius`
- Deux longueurs ou pourcentages (de la largeur de la bordure) : horizontalement, puis verticalement

BORDURES

- Possible d'utiliser `border-radius` au lieu de chacune des propriétés `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius`, `border-bottom-left-radius`.
- On a entre une et 4 valeurs : soit une valeur globale, soit une valeur pour chacun des coins, soit s'il manque des valeurs, elles sont copiées pour les coins opposés
- Chaque valeur est soit un longueur (`3px` par exemple) si l'angle est arrondi symétriquement, soit un couple de longueurs séparés par un « / » avec l'axe horizontal puis l'axe vertical
- Par exemple :

```
border-radius: 2em 1em 4em / 0.5em 3em;
border-radius: 3px 5px 6px 8px;
border-radius: 3px / 8px;
```



Source : terrainformatica.com

BORDURES

- Possible d'utiliser une image pour une bordure avec `border-image`, elle-même décomposée en cinq sous-propriétés, `border-image-source`, `border-image-slice`, `border-image-width`, `border-image-outset`, et `border-image-repeat`. D'un seul coup, ça donne :

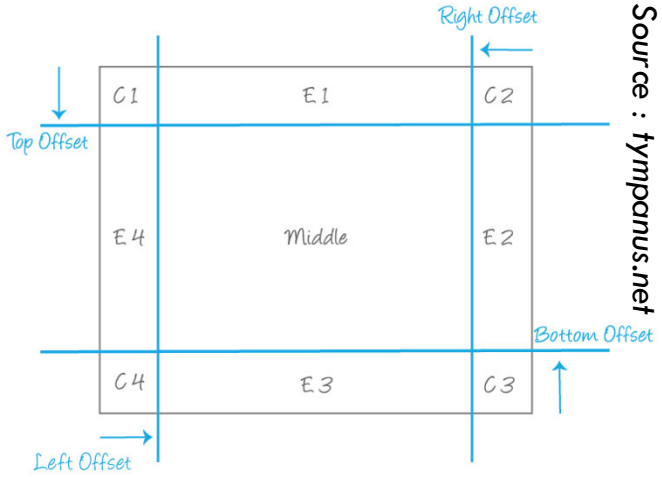
```
border-image: source slice width outset repeat | initial | inherit;
```

- `border-image-source` : la source de l'image (ou `none`), par exemple

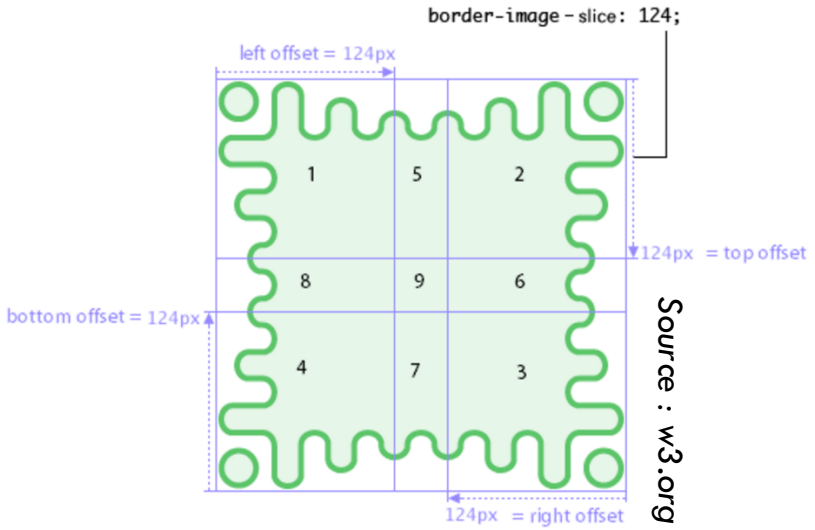
```
border-image-source: url(border.png);
```

- `border-image-slice` : 1 à 4 valeurs (absolues ou %) puis éventuellement « fill » (milieu = fond) :

This property specifies inward offsets from the top, right, bottom, and left edges of the image, dividing it into nine regions: four corners, four edges and a middle. The middle image part is discarded (treated as fully transparent) unless the 'fill' keyword is present. (It is drawn over the background; see Drawing the Border Image.)



Source : tympanus.net



Source : w3.org

BORDURES

border-image-slice: 50;

border-image-slice: 20%;

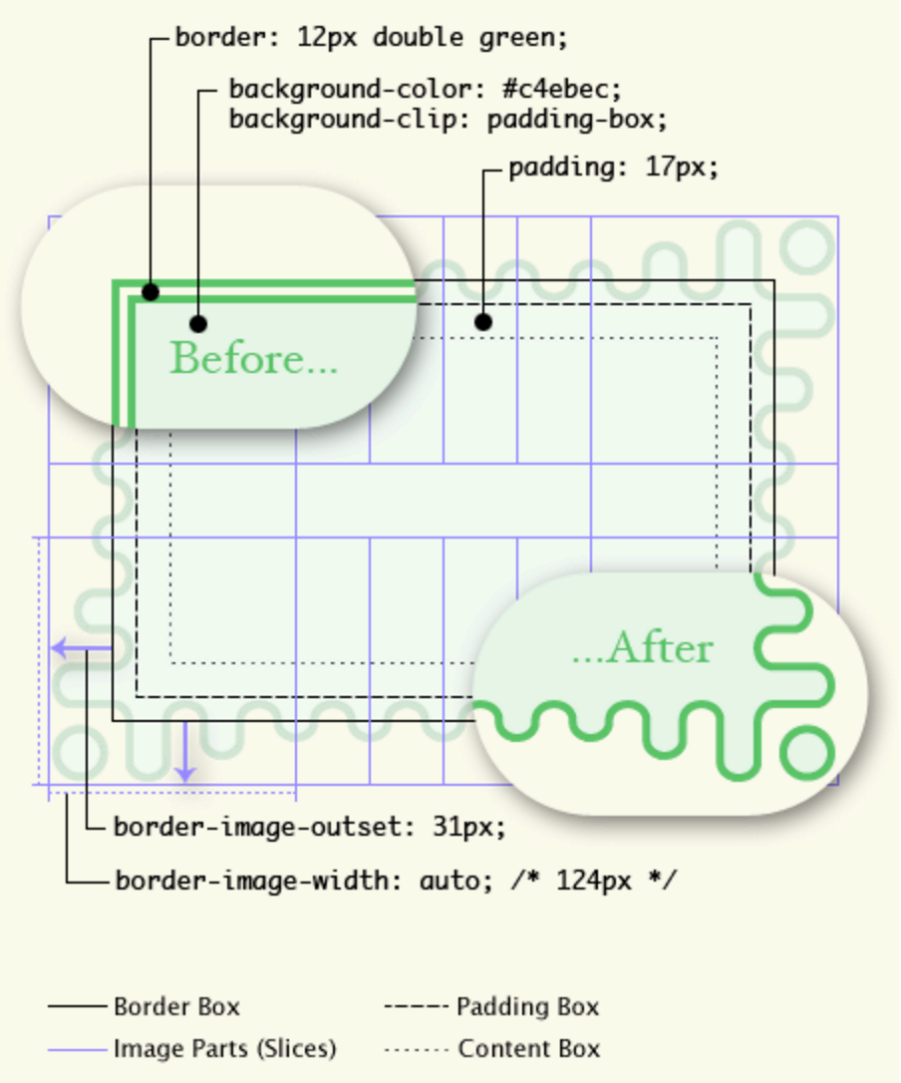
border-image-slice: 30%;

Here is the image used:

Source : w3schools.com

BORDURES

- `border-image-width` : la largeur des quatre coins (une à quatre valeurs, écart par rapport à chaque bord), permet d'étirer l'image…
- `border-image-outset` : de combien l'image dépasse-t-elle de la « border box » ? De une à quatre valeurs, comme toujours…
- Et enfin, `border-image-repeat` : comment répéter les quatre parties entre les angles pour remplir toute la longueur ?



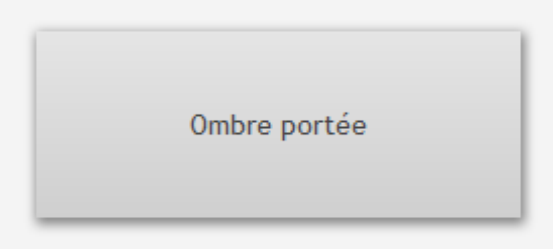
Source : w3.org

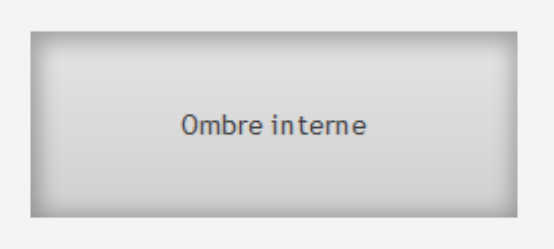
BORDURES

- `border-image-repeat: [ stretch | repeat | round | space ] {1,2};`
- Avec `stretch` : image étirée. Avec `repeat` : image répétée. Avec `round` : image répétée, puis étirée pour ne pas avoir un morceau d'image coupé au début ou à la fin. Et enfin, avec `space`, image répétée le nombre maximum de fois où elle tient, puis espace blancs entre.
- Possible de spécifier deux valeurs : horizontalement, puis verticalement.

Source : msdn.microsoft.com

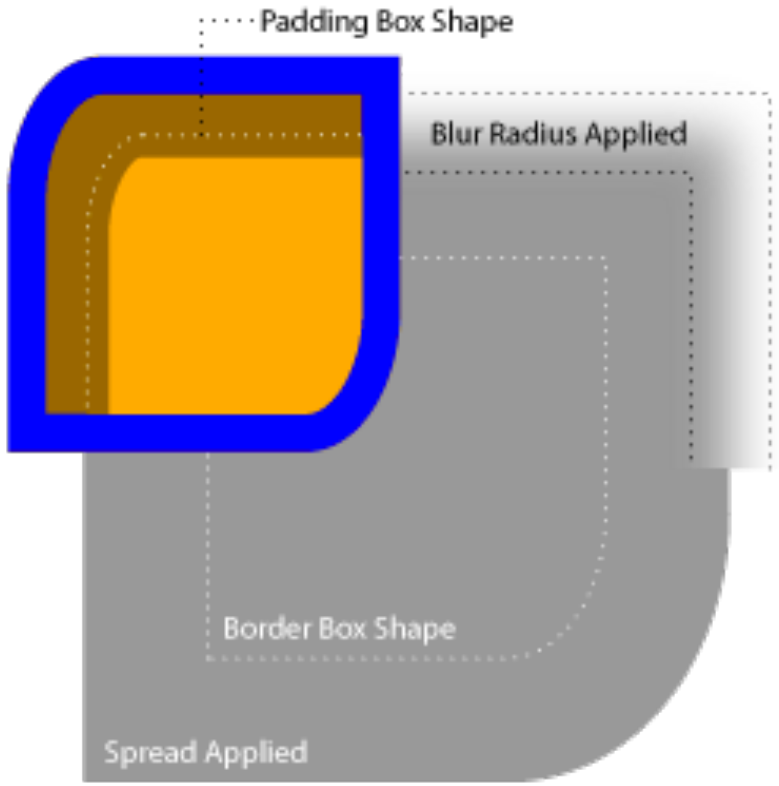
OMBRES





- La propriété `box-shadow` permet de dessiner une ou plusieurs ombres pour un élément
- Syntaxe : `box-shadow: inset? && <length>{2,4} && <color>?`
- Mettre `inset` pour une ombre interne. Couleur de l'ombre : souvent avec transparence…
- Entre 2 et 4 valeurs :
  1. Le décalage vers la droite de l'ombre (négatif pour la décaler vers la gauche)
  2. Le décalage vers le bas de l'ombre (négatif pour la décaler vers le haut)
  3. Le rayon de floutage. Zéro pour une ombre non-floue
  4. La taille de l'ombre : de combien elle s'étend au-dela de la taille de l'objet

OMBRES




```
Width:100px; Height:100px;
Border: 12px solid blue; Background-color: orange;
Border-top-left-radius: 60px 90px;
Border-bottom-right-radius: 60px 90px;
Box-shadow: 64px 64px 24px 40px rgba(0,0,0,0.4),
            12px 12px 0px 18px rgba(0,0,0,0.4) inset;
```

Source : w3.org

POLICES

- Cinq propriétés : `font-family`, `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height`… Et une propriété `font` pour tout spécifier d'un coup.
- `font-family` : une liste de polices, essayées les unes après les autres. Si le navigateur ne trouve pas la première police (pas installée, impossible de la télécharger si police spécifiée en `link`…), il essaie la suivante, etc.
  - Généralement, on met une police Windows, puis une police Mac, puis la famille générique de police pour obtenir une police similaire si aucune police n'a pu être chargée (familles génériques : `serif`, `sans-serif`, `monospace`, `cursive`, `fantasy`)
  - Utiliser soit des police installées couramment sur les machines (voir cours précédent) soit police dont on fournit l'URL (comme une Google Font).
- Exemple : `font-family: "Times New Roman", "Times", serif;`

Windows fonts / Mac fonts / Font family

Normal style	Bold style
Arial, Arial, Helvetica, *sans-serif*	**Arial, Arial, Helvetica, *sans-serif***
Arial Black, Arial Black, Gadget, *sans-serif*	**Arial Black, Arial Black, Gadget, *sans-serif***
Comic Sans MS, Comic Sans MS[5], *cursive*	**Comic Sans MS, Comic Sans MS[5], *cursive***
Courier New, Courier New, Courier[6], *monospace*	**Courier New, Courier New, Courier[6], *monospace***
Georgia[1], Georgia, *serif*	**Georgia[1], Georgia, *serif***
Impact, Impact[5], Charcoal[6], *sans-serif*	**Impact, Impact[5], Charcoal[6], *sans-serif***
Lucida Console, Monaco[5], *monospace*	**Lucida Console, Monaco[5], *monospace***
Lucida Sans Unicode, Lucida Grande, *sans-serif*	**Lucida Sans Unicode, Lucida Grande, *sans-serif***
Palatino Linotype, Book Antiqua[3], Palatino[6], *serif*	**Palatino Linotype, Book Antiqua[3], Palatino[6], *serif***
Tahoma, Geneva, *sans-serif*	**Tahoma, Geneva, *sans-serif***
Times New Roman, Times, *serif*	**Times New Roman, Times, *serif***
Trebuchet MS[1], Helvetica, *sans-serif*	**Trebuchet MS[1], Helvetica, *sans-serif***
Verdana, Verdana, Geneva, *sans-serif*	**Verdana, Verdana, Geneva, *sans-serif***
Symbol, Symbol (Symbol[2], Symbol[2])	**Symbol, Symbol (Symbol[2], Symbol[2])**
Webdings, Webdings (Webdings[2], Webdings[2])	**Webdings, Webdings (Webdings[2], Webdings[2])**
Wingdings, Zapf Dingbats (Wingdings[2], Zapf Dingbats[2])	**Wingdings, Zapf Dingbats (Wingdings[2], Zapf Dingbats[2])**
MS Sans Serif[4], Geneva, *sans-serif*	**MS Sans Serif[4], Geneva, *sans-serif***
MS Serif[4], New York[6], *serif*	**MS Serif[4], New York[6], *serif***

tout

ne

e en

ce

t on

Université Nice Sophia Antipolis

POLICES

- `font-style: normal | italic | oblique | initial | inherit`
  - Permet de choisir l'inclinaison de la police : `normal` = non inclinée, `italic` = *en italique*, `oblique` = inclinaison dans le sens opposé de l'italique (peu utilisé)
- `font-variant: normal | small-caps | initial | inherit;`
  - PERMET D'ÉCRIRE EN PETITES MAJUSCULES ou pas.
- `font-weight: normal | bold | bolder | lighter | number | initial | inherit;`
  - Permet d'écrire en gras (`bold`) ou pas (`normal`). En fait, permet de choisir assez librement l'épaisseur de la police : `bolder` = plus épaisse, `lighter` = plus fine (que `normal`)
  - Permet aussi de spécifier un multiple de 100 entre 100 et 900, du plus fin au plus épais. Sachant que `normal` = 400 et `bold` = 700

POLICES

- `font-size: medium | xx-small | x-small | small | large | x-large | xx-large | smaller | larger | length | initial | inherit;`
  - Souvent, au lieu d'une taille générique comme medium ou small, on indique la taille en points (`12pt` par exemple) ou en pixels.
  - Ne permet pas toujours aux navigateurs de redimensionner le texte pour les appareils mobiles ou pour les lecteurs pour mal voyants, par conséquent, le W3C recommande de plutôt utiliser des unités comme `em` ou `ex`.
- `line-height: normal | number | length | initial | inherit;`
  - Espacement entre les lignes. Si on spécifie un nombre sans unités : multiplication de l'espacement « normal » entre les lignes par ce nombre. Sinon, on peut spécifier un nombre absolu (en pixels, points, centimètres…) ou un pourcentage.

POLICES

- Enfin, la propriété `font` permet de spécifier tout d'un coup, dans l'ordre, il faut mettre :
```
font-style font-variant font-weight font-size/line-height font-family
```
- On ne peut pas omettre `font-size` et `font-family`. On peut omettre tout le reste.
- Si `line-height` omis, ne pas mettre le slash. Exemples :
```
font: italic bold 12pt/30px Georgia, serif;
font: oblique 300 12pt "Times New Roman", "Times", serif;
font: 12pt monospace;
```
- Possible également avec la propriété font de spécifier certaines polices du système :
  - `caption` = police utilisée dans les légendes des contrôles (par exemple: les boutons, menus déroulants etc.), `icon` = police utilisée pour les étiquettes des icônes, `menu` = police utilisée dans les menus, `message-box` = police utilisée dans les boites de dialogue, `small-caption` = la fonte utilisée pour les étiquettes de petits contrôles, `status-bar` = police utilisée dans la barre d'état

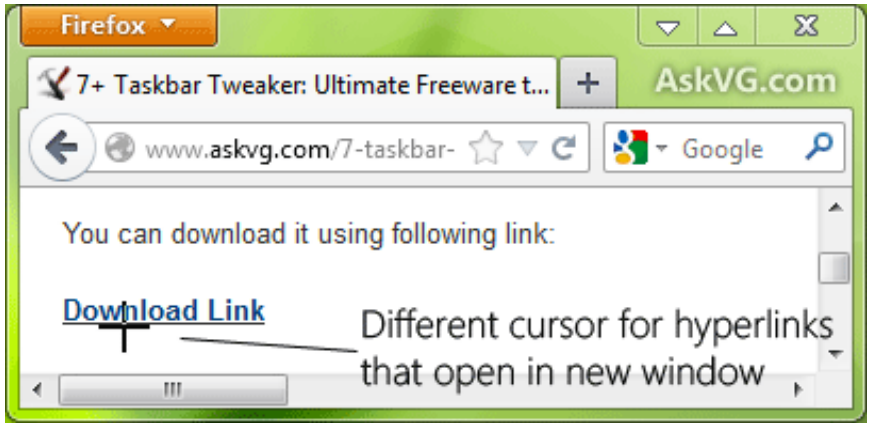
TEXTE

Quelques propriétés intéressantes pour le texte (il y en a beaucoup d'autres…)

- `text-decoration : none | underline | overline | line-through | initial…;`
  - Permet de souligner le texte mais aussi de le barrer ou de mettre une ligne au dessus…
  - Possible de choisir la couleur du soulignement (ou autre) avec `text-decoration-color`, et le style de ligne (`solid`, `dashed`, `dotted`), avec `text-decoration-style`…
- `text-align : left | right | center | justify | initial | inherit;`
  - Permet de choisir l'alignement du texte
- `letter-spacing : normal | length | initial | inherit;`
  - Permet de choisir l'espacement entre les lettres
- `text-indent: length | initial | inherit;`
  - Indentation de la première ligne d'un paragraph (alinéa)
- `text-shadow: h-shadow v-shadow blur-radius color | none | … ;`
  - Pour ajouter une ombre au texte…

CURSEURS

- Possible de spécifier un curseur qui sera affiché lorsque la souris survole un élément, par exemple : `:link[target="_blank"], :visited[target="_blank"] { cursor: crosshair; }`
- De très nombreux curseurs prédéfinis existent ! (aspect dépend de l'OS)

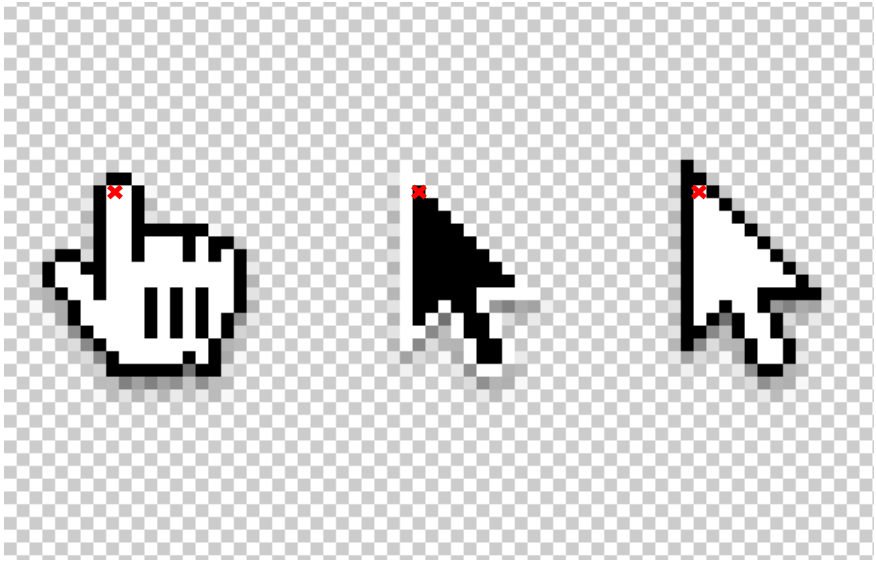


auto	move	no-drop	col-resize
all-scroll	pointer	not-allowed	row-resize
crosshair	progress	e-resize	ne-resize
default	text	n-resize	nw-resize
help	vertical-text	s-resize	se-resize
inherit	wait	w-resize	sw-resize

CURSEURS

- Curseurs personnalisés possibles : il faudra fournir une image, de préférence du PNG, car canal alpha : transparence possible pour se mélanger avec ce qu'il y a derrière le curseur (sinon on aura un carré autour du curseur…)
- Recommandé : spécifier les coordonnées du pixel qui effectue réellement le clic…
- Petite subtilité : sous Windows une image PNG ne marche pas forcément, il faut un fichier .cur (utilisez un converstisseur en ligne)
- Ne pas oublier de spécifier un curseur par défaut si le curseur n'a pas pu être chargé (même idée que les polices)
- On aura par exemple :

```
cursor : url(images/moncurseur.cur) 2 8,
         url(images/moncurseur.png) 2 8, hand;
```



TRANSFORMATIONS

- Possible de transformer un élément avec des rotations, déplacements, étirements, rotations…
- Une transformation a une origine indiquée avec transform-origin. Par exemple :

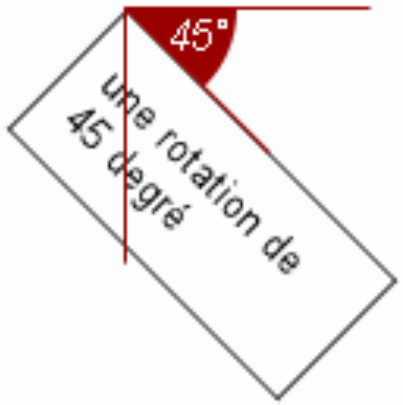
```
transform-origin: center center;
transform-origin: 60% 40%;
transform-origin: 20px 30px;
```

Par défaut, l'origine est le coin en haut à gauche (`top left`)
Possible de spécifier une troisième valeur pour les transformations 3D

- Rotation : avec `rotate()`.

Par exemple : `transform: rotate(45deg);`

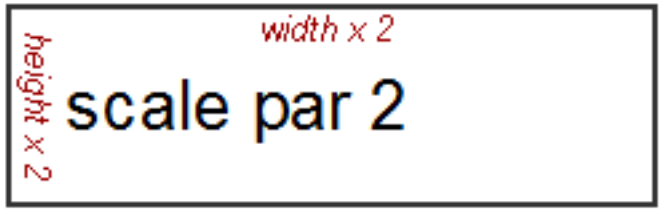
Sur Chrome et Safari, ne marche pas avec n'importe quelle balise (par exemple elements pas de type « bloc »).

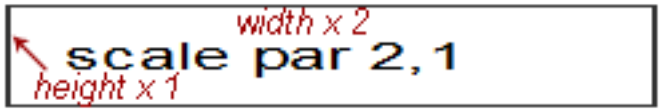


Source : developpez.com

TRANSFORMATIONS

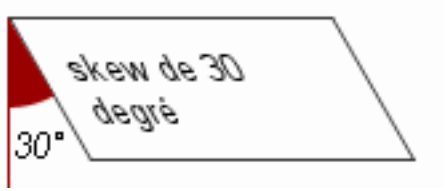
- Etirement avec `scale()`. `transform: scale(2)` multiplie par 2 la taille de l'élément ; `transform: scale(2, 1)` étire l'objet horizontalement deux fois, mais ne l'étire pas verticalement.

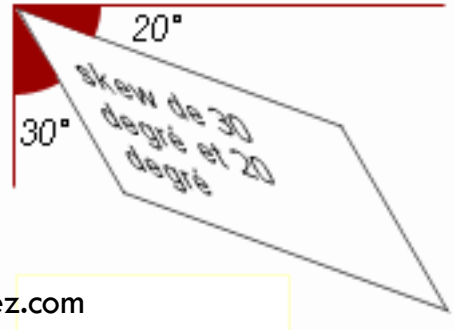


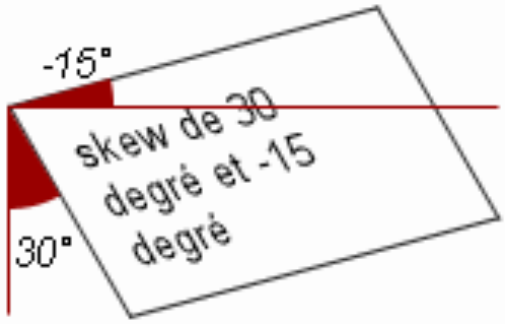


Source : developpez.com

- Glissement avec `skew()`. Si un seul paramètre, glissement horizontal, si deux paramètres, horizontal puis vertical. Les arguments peuvent être négatifs. Par exemple, `transform: skew(30deg)`, `transform: skew(30deg, 20deg)` et `transform: skew(30deg, -15deg)`.



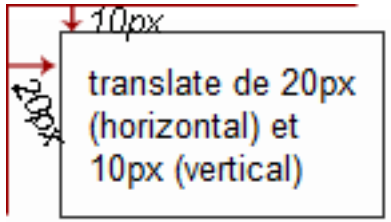




Source : developpez.com

TRANSFORMATIONS

- Translation avec `translate()`. Toujours pareil : si un seul paramètre, translation en x et en y de ce paramètre, sinon premier paramètre = décalage en x, et deuxième paramètre = décalage en y. Par exemple, `transform : translate(20px, 10px);`



Source : developpez.com

- On a aussi les fonctions `scaleX()`, `scaleY()`, `skewX()`, `skewY()`, `translateX()` et `translateY()` pour ne spécifier qu'une seule coordonnée.

- Possible d'enchainer plusieurs transformations, par exemple :

```
transform: skew(30deg, 15deg) translate(10px, 0px)
           rotate(-30deg);
```

Source : developpez.com

TRANSFORMATIONS

- Possible de spécifier une transformation 2D quelconque avec :

transform: matrix(a, b, c, d, tx, ty)

Où la matrice de transformation sera $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ à laquelle on ajoute une translation (tx, ty)

I.e., $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$

- Transformations 3D possibles !

- Pour les transformations 3D, il faut définir une perspective, avec:

```
transform: perspective(600); // Si transformation 3D sur un élément
perspective: 600; // Si transfo 3D sur un ensemble d'éléments (propriété
                     de l'élément parent)
```

La valeur donnée détermine l'intensité de l'effet 3D, plus sa valeur est élevée, plus le point de fuite est distant…

Source : developpez.com

TRANSFORMATIONS

```
HTML
<div class="perspective"></div>
<div class="perspective"></div>
<div class="perspective"></div>
CSS
.perspective{
    float:left;
    width:200px;
    height:100px;
    margin:20px;
    background-color:#333;
    -webkit-transform:perspective(600) rotateX(-45deg);
}
```

Source : developpez.com

Source : developpez.com

TRANSFORMATIONS

HTML

```
<div class="transformPerspective">
    <div></div>
    <div></div>
    <div></div>
</div>
```

CSS

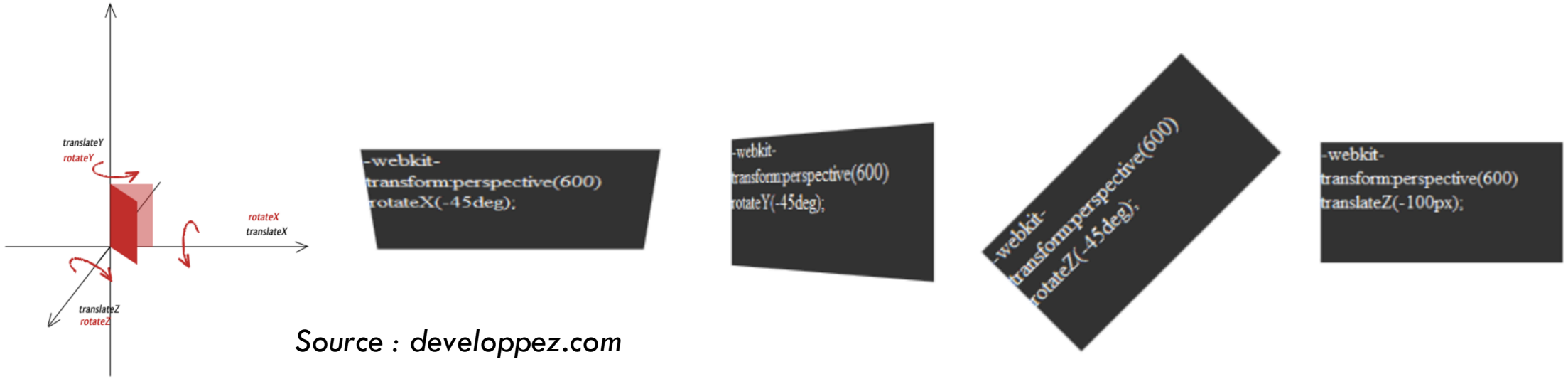
```
.transformPerspective{
    clear:left;
    -webkit-perspective:600;
    width:700px;
}
.transformPerspective div{
    float:left;
    margin:10px;
    width:200px;
    height:100px;
    background-color:#333;
    -webkit-transform: rotateX(-45deg);
}
```

Source : developpez.com

Source : developpez.com

TRANSFORMATIONS

- Transformations 3D : mêmes fonctions qu'avant, avec en plus `rotateX()`, `rotateY()`, `rotateZ()` (transformations sur les 3 axes possible au lieu d'un seul maintenant), `translateZ()`, et `scaleZ()`



Source : developpez.com

- Pas encore bien supportées sur tous les navigateurs, besoin d'utiliser des préfixes parfois

Source : developpez.com

FILTRES

- Possible d'appliquer des filtres avant qu'un élément ne soit affiché avec `filter`.
- `blur(px)` floute l'image (rayon donné en pixels), `brightness(pct)` ajuste la luminosité (0% = noir, 100% = image non modifiée), `contrast(pct)` ajuste le contraste (même chose pour les pourcentages), `saturate(pct)` ajuste la saturation, `grayscale(pct)` passe l'élément en noir et blanc, `opacity(pct)` rend l'élément plus ou moins transparent, `sepia(pct)` ajoute un effet sépia, `invert(pct)` inverse les couleurs, `hue-rotate(deg)` modifie la teinte de l'image, `drop-shadow(x, y, flou, étendue, couleur)` ajoute une ombre…


Original image


blur(5px)


Original image


contrast(200%) brightness(150%)


Original image


sepia(100%)

Source : developpez.com

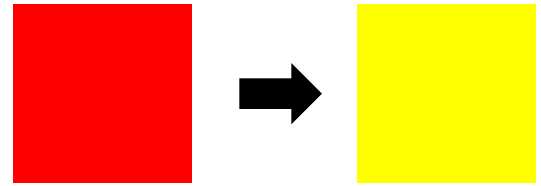
ANIMATIONS

- **Possible de faire des animations en CSS !**
- Idée : passage automatiquement d'un style vers un autre. Donc on peut faire passer un fond graduellement d'une couleur à l'autre, ou faire tourner un objet (passage d'une rotation d'un angle à un autre), etc.
- On définit une animation avec `@keyframes` pour définir plusieurs styles à plusieurs moment, par lesquels passera l'élément. Par exemple, un carré rouge qui devient un carré jaune graduellement (passage par des degrés de orange) en 4 secondes :

```
/* The animation code */
@keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}
```

Source : w3schools.com

```
/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s; /* Nécessaire */
}
```

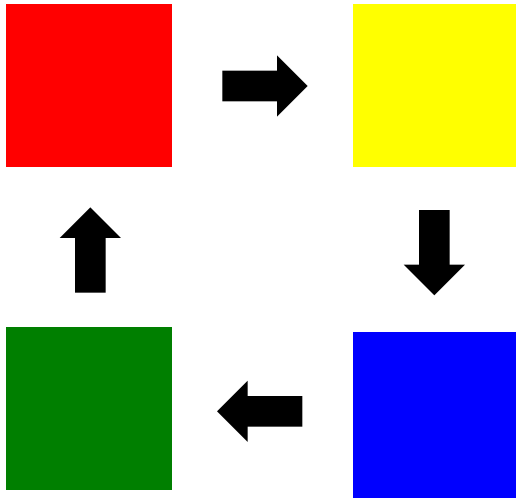


Source : developpez.com

ANIMATIONS

- Un exemple un peu plus avancé :

```
/* The animation code */
@keyframes example {
    0%   {background-color: red; left:0px; top:0px;}
    25%  {background-color: yellow; left:200px; top:0px;}
    50%  {background-color: blue; left:200px; top:200px;}
    75%  {background-color: green; left:0px; top:200px;}
    100% {background-color: red; left:0px; top:0px;}
}
```



```
/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
```

Source : w3schools.com

ANIMATIONS

Propriétés pour les animations :

- `animation-delay`: attente avant que l'animation ne commence (`2s` par exemple)
- `animation-iteration-count`: nombre de fois qu'on répète l'animation (peut être `infinite`)
- `animation-direction`: `reverse` pour inverser, `alternate` pour en avant, arrière, avant…
- `animation-timing-function`: permet un timing non linéaire. Par exemple, `ease-in` fait commencer l'animation lentement, alors que `ease-out` rend la fin plus lente. Possible de définir une fonction temporelle bien précise avec `cubic-bezier()` (voir la documentation)
- Comme toujours, possible de tout spécifier d'un coup avec `animation` :

```
div {
    animation-name: example;
    animation-duration: 5s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

⟷

```
div {
    animation: example 5s linear 2s
               infinite alternate;
}
```

POUR FINIR : PROPRIÉTÉS UTILES EN VRAC

- `vertical-align`: permet de choisir l'alignement vertical d'un élément (par rapport à son parent), par exemple, `top`, `middle`, ou `bottom`
- `overflow`, `overflow-x`, `overflow-y`: permettent de choisir ce qu'on doit faire lorsque le contenu d'un élément dépasse.
- `visibility`: pour cacher un élément (`hidden`)
- `outline-*`: pour mettre un contour à du texte (par exemple)
- z-index: pour les éléments en position `relative`, `absolute` ou `fixed`, permet de choisir lesquels sont au dessus desquels

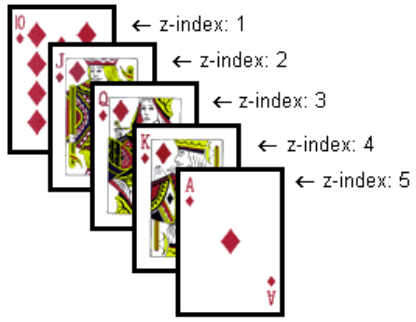
Glow outlines:

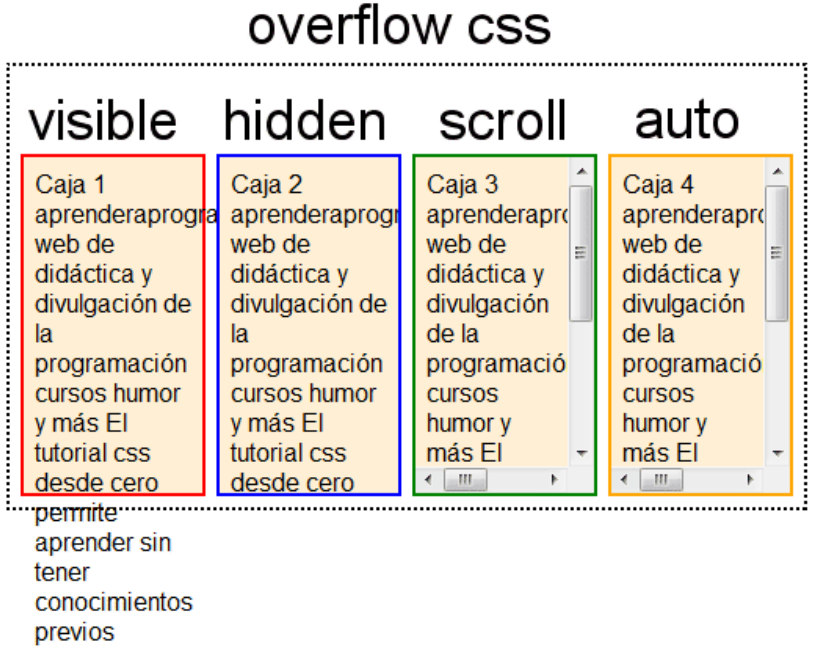
Outline width:5px, offset:0px, glow

Outline width:3px, offset:2px, glow

Outline width:1px, offset:4px glow

Outline width:1px, offset:3px, glow, no solid background





PROPRIÉTÉS CSS : À SUIVRE…

- Vous n'avez pas vu toutes les propriétés, mais vous avez déjà de quoi vous amuser ☺
- Propriétés très nombreuses, permettent de faire des choses très évoluées (des effets sur le texte, des rotations, des animations…)
- Surtout, validez toujours vos pages, et lorsque vous corrigez les erreurs, *très important de lire et de comprendre les messages d'erreurs et pas seulement de regarder la ligne !* Par exemple, que veulent dire :