NFA016 : les cascading style sheets (CSS)

O. Pons, S. Rosmorduc, M. Simonot

Conservatoire National des Arts & Métiers

CSS = Cascading Style Sheets

- écrits dans un fichier .css

  Exemple : `style1.css`

- un même fichier peut être partagé par plusieurs pages ⇒ Uniformité de présentation)
- ajouter dans l'en-tête (`head`) de la page HTML qui utilise le CSS :

```
<link rel="stylesheet" type="text/css"
      href="style1.css">
```

- la css définit des règles pour appliquer des *styles* (couleur, police, etc.) à des parties d'une page HTML.
- On peut définir *plusieurs* feuilles, dont certaines ne sont utilisées qu'avec des afficheurs spécifiques :

```
<link rel="stylesheet" href="css/presentation.css"/>
<link rel="stylesheet" href="css/presentation-screen.css" media="sc
<link rel="stylesheet" href="css/presentation-print.css" media="pri
```

- quelque media possibles : all, braille, handheld, print, projection, screen.

Webographie

- standard CSS2 : http://www.w3.org/TR/CSS2
- proposition CSS3 : http://www.w3.org/TR/CSS
- W3Schools http://www.w3schools.com/cssref
- Compatibilité des navigateurs : http://www.quirksmode.org/css/contents.html

Structure d'une feuille de style

```
body {color:rgb(183,0,183); margin: 1.5cm;}
h1, h2 {
   color: aqua;
   text-align: center;
}
p {
   background-color: yellow;
}
```

- Suite de règles ;
- Une règle =

```
Sélecteur {
   propriété : valeur ;
   ...
   propriété : valeur ;
}
```

Les sélecteurs

Définissent à quels éléments un style est appliqué.

```
1  body {color:rgb(183,0,183); margin: 1.5cm;}
2  h1, h2 {
3     color: aqua;
4     text-align: center;
5  }
6  p {
7     background-color: yellow;
8  }
```

- les éléments du `body` sont en violet avec une marge de 1,5cm
- les paragraphes ont *en plus* une couleur de fond jaune
- les titres `h1` et `h2` sont *en plus* centrés et leur texte est en bleu clair

Certaines propriétés comme les fontes ou la couleur sont héritées par les éléments descendant.

On peut sélectionner...

un type de balise

```
h2 {text-align: center;}
```

une classe

```
.important {color: red;}
```

un id

```
#entete {background-color: blue;}
```

une balise d'une certaine classe

```
p.important {color: blue;}
```

un sélecteur descendant d'un autre sélecteur

```
p em {color:gray;}
```

tous les `em` dans un paragraphe seront en gris

`p .important {color:green;}` Tous les élements de classe important à l'intérieur d'un paragraphe seront verts.

On peut sélectionner ...

`.resume em {color:yellow;}` Tous les `em` descendants d'un élement de classe `resume` seront jaunes.

un sélecteur enfant d'un autre sélecteur

```
.resume>em {color:yellow;}
```

un sélecteur frère droit d'un autre sélecteur

```
h5+p {color:pink;}
```

un sélecteur frère (non forcément immédiat) d'un autre (CSS3)

```
h5 ~ p {color:pink;}
```

(le signe ~ est un "tilde")

Attribut id et feuille de style

```
1 p#resume {color :green;}
2 #toto {color :blue;}
```

et dans la page HTML :

```
1 <h1 id="toto">Celui-ci est bleu </h1>
2 <p id="resume"> celui-là est vert </p>
```

Rappel : deux éléments dans la même page ne peuvent avoir le même id.

Sélecteur d'attributs général

Pour sélectionner un élément sur la valeur d'un attribut quelconque :

```
[ ATTR=VALEUR]
```

Exemple:

```
1 * [class="important"] {color :red;}
2 p [class="remarque"] {color :red;}
3 *[lang="fr"] {color :red;}
```

- "*" : n'importe quelle balise
- fonctionne sur n'importe quel attribut
- pas de valeur précisée ⇒ sélectionne n'importe quelle valeur précisée

Remarque

`.toto` et `*[class="toto"]` sont similaires mais...

```
<p class="titi toto"> un paragraphe;...</p>
```

est reconnu par :

```
.toto {color: red;}
*[class="titi toto"] {color: red;}
```

mais pas par :

```
*[class="toto"] {color: red;}
```

Pseudo-classes

Sélecteurs additionnels, ajoutant typiquement des informations d'état ou de position.

S'écrivent ":" suivi de la pseudo-classe

Exemples

- `:first-child` : l'élément est le premier fils de son élément père (par exemple le premier *li* d'une liste

```
li:first-child {color: red;}
```

- pour les liens : `:visited` : liens visités, `:link` autres liens
- position de la souris `:hover` actif quand la souris est sur l'élément.

pseudo-classe nth-child : sélectionner un élément sur n (CSS3)

typiquement, permet d'alterner des présentations différentes dans des éléments successifs (tables, listes...)

Exemple :

```
li:nth-child(even) {background-color: #A0A0FF;}
li:nth-child(odd) {background-color: white;}
li:first-child {background-color: #FF0000;}
```

- un
- deux
- trois
- quatre
- cinq

- forme : nth-child(odd) (impair), nth-child(even) (pair) ou nth-child(**a**n+**b**)
- on commence à compter à 1
- dans la forme générale, on donne la période et l'indice du premier élément. `nth-child(3n+2)` : sélectionne 1

élément sur 3, à partir du numéro 2.

Remarque : on ne compte que les *éléments*. Le texte isolé ne compte pas

un *exemple*

```
<style type="text/css">
  #dfc :first-child {background-color: red;}
</style>
<div id="dfc">
  un <em>exemple</em>
</div>
```

Exercice

Lire la page Html et la feuille de style qui suivent et répondre aux questions suivantes :

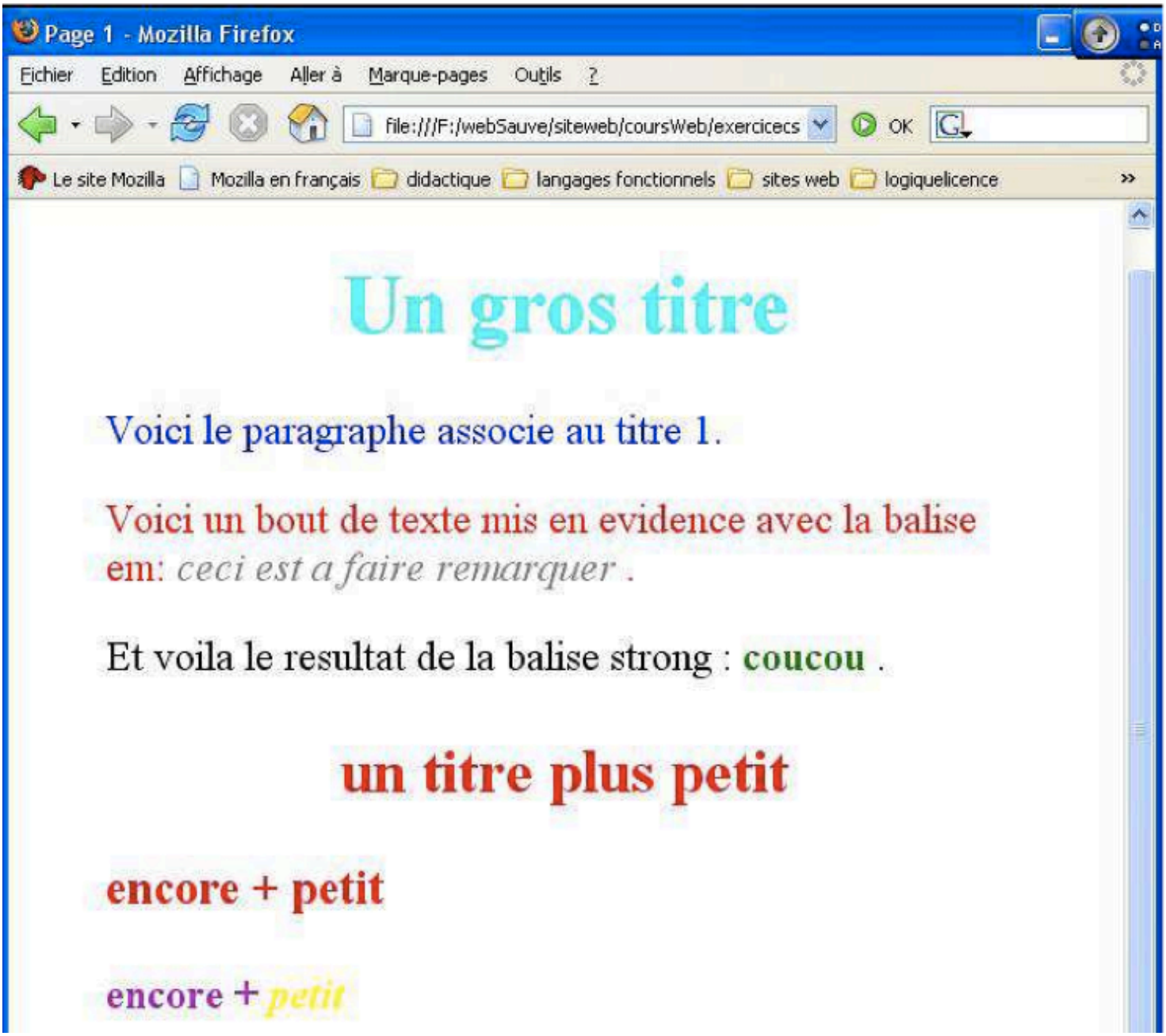
- Dessiner la structure de la page HTML sous forme d'arbre.
- Sur chaque noeud de l'arbre, ajouter les numéros de règles css qui s'appliquent.
- en déduire l'affichage de la page.

Exercice

```
<body>
  <h1 >Un gros titre</h1>
  <p class="bleu">Voici le paragraphe
    associe au titre 1.
  </p>
  <p class="important" >Voici
    un bout de texte
    mis en evidence avec la
    balise em:
    <em> ceci est a faire remarquer </em>.
  </p>
  <p> Et voila le resultat
      de la balise strong :
   <strong class="important"> coucou </stro
  </p>
  <h2 class="important">un titre plus petit
  <h3 class="important">encore + petit</h3>
  <h4 class="resume">encore + <em>petit</em
  <h5>encore + petit</h5>
  <p>toto</p>
```

```
body {
 color:rgb(183,0,183); /* violet */
 margin:1.5cm;
}
h1,h2{
 color:aqua;
 text-align:center;
}
p{color:black;}
.important {color:red;}
p.bleu {color:blue;}
p em  {color:gray;}
p .important {color:green;}
.resume em {color:yellow;}
h5 + p {color:lime;}
```

Exercice



div, span et feuilles de style

```css
body { color:rgb(183,0,183);}
h1,h2{ color:aqua;}
.important {background-color:grey;}
.important h1 {color:yellow;}
.important p {color:aqua;}
```

```html
<h1 >chapitre 2: les feuilles de style</h1>
<div class="important">
  <h1> Les selecteurs </h1>
  <p>Differents types de selecteurs</p>
</div>
<h1> Les propriétés </h1>
<p>bla bla bla</p>
```

Autres moyens d'introduire un style

Il existe deux autres manières d'introduire le style. On préfère normalement des feuilles de styles indépendantes.

- dans le document HTML lui-même :

```
<head>
  <title> un beau titre</title>
  <style>
     h1 {text-align: centered;}
  </style>
...
```

- comme attribut d'une balise ouvrante ; s'applique à cette balise-là uniquement.

```
<p style="text-align: centered; background-color: red;">....</p>
```

On ne met que les propriétés dans ce cas, pas de sélecteur

Contenu d'un style (les propriétés)

Très nombreuses propriétés. Nous allons en voir quelques-unes. webographie :

- http://www.yoyodesign.org/doc/w3c/css1/index.html
- http://www.yoyodesign.org/doc/w3c/css2/cover.html
- http://pompage.net/
- http://fr.selfhtml.org

  et surtout leur chapitre sur les propriétés css: http://fr.selfhtml.org/css/proprietes/index.htm

- Validateur css: http://jigsaw.w3.org/css-validator/

Couleur

Trois façons de les désigner :

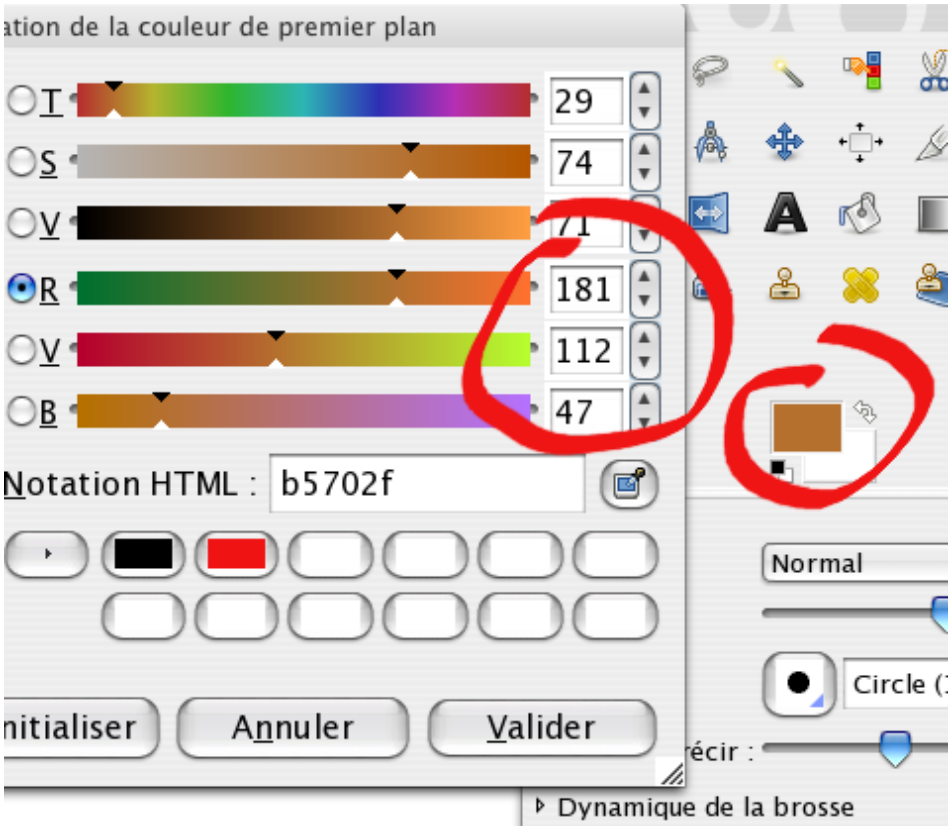
- par nom : 16 noms prédéfinis `white, silver, gray, black, red, maroon, lime, green, yellow, blue, navy, fushia, purple, aqua, teal`
- schéma RGB : trois valeurs, entre 0 et 255 pour le rouge, le vert, le bleu :

  ```
  rgb(181, 112, 47)
  ```

  ou trois pourcentages :

  ```
  rgb(80%, 10%, 20%)
  ```

  voir les logiciels de dessin (gimp, photoshop, etc...)



- en hexadécimal (six chiffres hexadécimaux, deux par composante). De `#000000` (noir) à `#FFFFFF` (blanc).

Propriétés de couleur

color

fixe la couleur du *texte*

```
1 | p {color: rgb(0,10,200);}
```

background-color

fixe la couleur du fond.

```
1 | p {color: red; background-color: yellow;}
```

(texte rouge sur fond jaune).

Polices de caractères

- groupées en *familles* : même auteur, aspect compatible entre les variations de la police (exemple : Garamond, Times)
- dans une même famille, plusieurs *styles* disponibles : normal, *italique*, *oblique*
- plusieurs graisses disponibles : normal, fin et **gras**

Polices de caractères

Problèmes divers, en particulier de droits.

Le navigateur client n'a pas forcément la police demandée

- utiliser des polices courantes
- proposer plusieurs polices
- proposer une famille comme dernier recours

font-family

```
p {font-family: helvetica, verdana,sans-serif;}
```

- On donne une liste de noms de polices. La première à être disponible est utilisée
- Si espaces dans un nom : mettre des guillemets :

  ```
  p {font-family: Georgia, "Times New Roman", Times, serif;}
  ```

- trois familles de base : Serif, Sans-Serif et `Monospace`.

Normallement, la dernière famille est l'une des trois familles de base.

Téléchargement de polices

Permet de définir ses propres "familles", à partir de polices qui seront téléchargées.

```
1 @font-face
2 {
3 font-family: maPolice;
4 src: url('gentium.ttf'),
5 url('gentium.eot'); /* IE */
6 }
```

pb: droits sur les polices ; formats différents (IE : Embedded OpenType, les autres : ttf)

convertisseurs ttf-> eot disponibles

Autres caractéristiques des polices et des textes

font-style
: valeur : `italic, oblique` ou `normal`

font-weight
: valeur : `bold, bolder, lighter, normal`

font-size

- valeur numérique absolue: 12pt, 10mm, 1cm, 10px
- taille prédéfinie : `xx-small, x-small, small, medium, large, x-large, xx-large`
- taille relative au parent : smaller, larger (agrandit ou rétrécit la police par rapport à celle du parent)

- valeur relative numérique : 150%, par rapport à la taille dans le parent.
- taille relative à la police actuelle: 1em, 1ex (largeur d'un "m", hauteur d'un "x")

text-decoration
: valeur `underline, overline, line-through` (barré) `none`

text-align
: alignement horizontal du texte : `left, right, center, justify`

text-indent
: indentation du premier paragraphe (décalage du début du texte)

Les boîtes

L'affichage est composé de *boîtes* contenant le texte. Chaque boîte est composée de plusieurs parties : le contenu (le texte lui-même), l'espace intérieur (*spanning*), la bordure et la marge.

(margin)

(border)

(padding)

le contenu le contenu

le contenu le contenu

propriétés des boîtes : les dimensions

Normalement, une boîte de type "block" occupe tout l'espace horizontal disponible, et sa hauteur est adaptée au texte à afficher.

démonstration

Inversement, des éléments de type span n'occupent que l'espace strictement nécessaire, pas forcément rectangulaire. (démonstration,)

Il est possible de fixer les dimensions des boîtes.

Les unités

- cm, mm, pt : dimensions classiques. 1pt= 1/72 pouce
- px: un pixel
- em : largeur d'un « m »
- ex : hauteur d'un « x »
- % : pourcentage de la boîte englobante ou de la page.

  démo démo

```
<div style="width: 10em; background-color: red; padding: 1px;">
  <div style="width: 50%; background-color: yellow; margin: 1ex;">
    démo démo
  </div>
</div>
```

- (note hors programme: sans padding sur le père, on aura *fusion des marges*, phénomène sur lequel nous ne nous étendrons pas, mais qui donnera ceci :

  démo démo

propriétés des boîtes : les dimensions

height, width
min-height, max-height, min-width, max-width
: valeur : % ou px, em, cm... Attention: width et height ne sont les dimensions internes des boîtes (hors padding, bordure, margin).

```
#contact {height:4cm; width:30% ;}
```

margin
: espace minimal entre la bordure de l'élément et les autres éléments. *la marge est **transparente** ; elle ne prend donc pas la couleur de fond de l'élément. Il faut y penser véritablement comme à un espace.*

padding
: espace entre le texte et la bordure de l'élément

on a aussi `margin-top, margin-right, margin-left, margin-bottom, padding-top,...padding-left`

Propriétés des boîtes : les bordures (`border`)

Trois informations: largeur, style et couleur.

```
p {border:medium solid black;}
h1 {border:4px dotted green;}
```

Le style peut être (entre autres)

none

dotted

dashed

solid

double

groove

ridge

inset

outset

CSS3 introduit border-radius et border-image pour des bordures plus complexes.

solid, border-radius: 20px;

a aussi `border-top, border-bottom, border-left, border-right`

Bordures et tableaux

Les bordures s'utilisent assez naturellement avec les tableaux.

En règle générale, on va placer une bordure sur les cases.

```
1 th,td: {border: solid 1pt black;}
```

a	**b**
a	b

pour coller les cases, `border-collapse` (valeurs : `separate` ou `collapse`)

exemple

```
1 table {border-collapse: collapse;}
2 th,td: {border: solid 1pt black;}
```

a	**b**
a	b

Exercice

On veut mettre en ligne un livre sur la programmation.

Chaque chapitre a la même structure :

- résumé ;
- texte du chapitre ;

- éventuellement un encadré exposant les principaux résultats exposés.

Écrire la page HTML correspondant au chapitre : *héritage en java* et la feuille de style qui s'appliquera à tous les chapitres de ce livre.

On veut obtenir la présentation suivante :

Héritage en Java

Résumé

Ce chapitre du cours traite de concepts relatifs à la programmation objet (hiérarchie de classe, héritage, extension, masquage) et sera illustré par un exemple de représentation de comptes bancaires et d'opérations liées à la manipulation de ceux-ci (retrait, dépôt, consultation).

Une classe simple

Décrivons maintenant la classe permettant de représenter un compte bancaire. Il faut, au minimum, connaître le propriétaire (que l'on représentera simplement par son nom), le numéro du compte .....

> En java, dire qu'une classe en étend une autre (A extends B) signifie que tout élément de A est aussi un élément de B.

Note : la sobriété est normalement une vertu en typographie. Nous avons choisi à dessein des couleurs criardes pour simplifier l'exercice.

Propriétés de listes

`list-style-type` : apparence des signes d'une liste.

propriété de la *liste* (ul ou ol)

decimal
: 1.,2.,3.,4. etc...

lower-roman
: i.,ii.,iii.,iv. etc.

upper-roman
: I.,II.,III.,IV. etc...

lower-alpha
: a.,b.,c.,d. etc.

upper-alpha
: A.,B.,C.,D. etc...

disc
: rond plein comme puce

circle
: puce ronde

square
: puce rectangulaire

none

pas de puce, pas de numérotation

`list-style-image`

```
list-style-image:url([nom de fichier])}
```

Vous permet de mentionner un graphique pour vos propres puces.

Exercice

Dans le livre précédent, tous les 3 ou 4 chapitres, il y a des chapitres particuliers: les « mise en pratique». Ils contiennent des exercices et des projets, et ont une présentation différente.

Écrire la page HTML correspondant et modifier la feuille de style en conséquence.

Héritage en Java

Exercice

Ajouter une classe `CompteRemunereAvecDecouvert`.

Projet

Écrire un logiciel de jeu d'échec (on ne demande pas que le logiciel joue, seulement qu'il arbitre).

Quizz

1. Si la classe B redéfinit la méthode f() de la classe A, et que j'écris `A a= new B(); a.f();` J'appelle :
   i. la méthode f de la classe A
   ii. la méthode f de la classe B
   iii. les deux
   iv. ça ne compile pas.
2. Dire qu'une classe est abstraite signifie :
   qu'elle est très compliquée à comprendre
   que seules des sous-classes pourront être instanciées
   qu'elle ne désigne pas un objet de la vie courante

Règles de résolutions de conflits

supposons que nous ayons :

```
h1 {color: blue; font-size: large;}
.titre {color:red;}
```

Comment se présente donc

```
<h1 class="titre">un titre</h1>
```

- Les deux sélecteurs s'appliquent ;

- pour la taille, pas de conflit : large
- pour la couleur, intuitivement : le sélecteur le *plus spécifique* s'applique. C'est `.titre`.⇒ texte en rouge

Règles de résolutions de conflits

- Pour des CSS de même origine, la priorité se fait selon la spécificité des règles.
- Pour une même spécificité, on choisit la dernière règle spécifiée.

Calcul de la spécificité

On construit un triplet d'entiers :

1. n1= le nombre d'ids d'attributs dans le sélecteur
2. n2= le nombre d'attributs et de classes dans le sélecteur
3. n3= le nombre de noms d'éléments et de pseudo-éléments(`h1, h2, em, p, :first-line ...`) dans le sélecteur.

Règles de résolutions de conflits

On compare les triplets par ordre *lexicographique* et le plus grand est le plus spécifique.

(a1,b1,c1) > (a2,b2,c2) si :

- a1 > a2
- ou a1 = a2 et b1 > b2
- ou a2 = a2 et b1 = b2 et c1 > c2

Exemples :

- p em : spécificité 0,0,2
- .important p: spécificité 0,1,1

En cas d'égalité, la dernière règle spécifiée l'emporte.

Positionnement et feuilles de styles

Problème : obtenir une mise en page complexe

- au départ, les CSS sont assez pauvres dans ce domaine.
- plusieurs mécanismes en concurrence
- pas mal de bricolage

à la base, mise en page de base : composition de boîtes de la largeur de l'écran. Empilement des boites en question

Titre

Paragraphe 1 Paragraphe
1 Paragraphe 1
Paragraphe 1

Paragraphe 2 Paragraphe
12 Paragraphe 2
Paragraphe 2 Paragraphe
2

- liste ...
- liste...

Positionnement

Réalisé par plusieurs propriétés :

display
: comportement général de l'élément : bloc, en-ligne, etc....

position
: manière dont l'élément se positionne par rapport aux autres ou à la page

top, bottom, left, right
: position de l'élément par rapport à son point de repère (fixé par `position`)

display (block, inline, none, etc.)

position (static | relative | absolute | fixed)

top, bottom, left, right

display

- Permet essentiellement de fixer si l'élément se conduit comme un bloc ou s'insère au fil du texte.
- nombreuses autres valeurs (éléments de tableaux, etc).

Valeurs possibles

inline

l'élément ne crée pas de nouvelle zone. Il s'insère dans un paragraphe existant. Pas de passage à la ligne.

un paragraphe normal (en block)

un paragraphe en inline un second paragraphe en inline bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

```
<p style="background-color: red;">un paragraphe normal (en block)<
<p style="display: inline; background-color: red;">un paragraphe
  en  inline</p>
<p style="display: inline; background-color: red;">un
second paragraphe
  en inline </p>
```

Notez qu'il n'est pas possible de fixer la taille des éléments `inline` (il faut utiliser block, inline-block, ou float)

block

l'élément est disposé dans un bloc rectangulaire dont on peut fixer la taille.

none
l'élément n'est pas affiché du tout

hidden
l'élément n'est pas affiché, mais la mise en page tient compte de sa taille.

inline-block
extérieurement inline, peut contenir des blocs.

Position

La propriété `position` a différentes valeurs : `static,relative,absolute,fixed`

Emplacement

- L'emplacement peut être modifié grâce aux propriétés `top, bottom, left, right`
- positions par rapport à un point de référence d'un des côtés du cadre

Positionnement relatif

valeurs : static et relative.

- static : valeur par défaut. Position dépend de la position d'un élément précédent
- relative : position dépend de la position d'un élément précédent, mais on peut la modifier avec top, bottom, left et right.

un paragraphe statique

un paragraphe relatif au précédent...

un paragraphe relatif au précédent...

```
<p> un paragraphe statique</p>
<p style="position: relative; left: 4em;"> un paragraphe relatif
au précédent...</p>
<p style="position: relative; left: 4em;"> un paragraphe relatif au pré
```

La position de référence est la position « normale », pas la position modifiée

Positionnement absolu

Positionnement absolu par rapport à un élément ancêtre (qui *contient* l'élément à placer) et non à un élément qui précède (comme pour relatif).

Élément de référence: le premier parent en remontant dans l'arbre de la page **qui ne soit pas en statique**

Les élément absolus ne sont pas pris en compte dans la mise en page des autres éléments.

un exemple un div absolu

```
<div style="position: relative;">un exemple
```

```
    <div style="position: absolute; top: 0em; left: 3em;">un div absolu
  </div>
```

Les flottants

Au départ: positionnement des illustrations dans le texte... comme ça :

bla bla

Rapidement détourné pour faire de la mise en page plus complexe

Valeurs

- left : flotte à gauche. Le texte contourne l'élément.
- right: flotte à droite. Le texte contourne l'élément.

Comportement par rapport au parent

Les flottants ne sont pas pris en compte dans le calcul de la taille du parent

un essai. Vous voyez, ça ne tient pas.

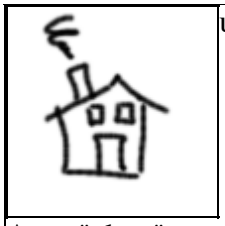
```
<div style="border: 1px solid;"> un essai.
    <img src="coursCSSdata/dessin.png" style="border: 1px solid
  black; padding: 3px; float:left;"> Vous voyez, ça ne
  tient pas.
  </div>
```

clear

Propriété qui permet d'interdire la présence d'un flottant d'un côté ou de l'autre d'un élément. On déplace l'élément après les flottants concernés.

valeurs:

- right : pas de flottant à droite
- left: pas de flottant à gauche
- both: pas de flottant du tout

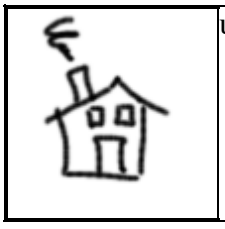

un essai.

Avec "clear", on saute après le flottant

```
<div style="border: 1px solid;"> un essai.
   <img src="coursCSSdata/dessin.png" style="border: 1px solid
 black; padding: 3px; float:left;">
   <p  style="clear:both;">Avec "clear", on saute après le flottant</p>
</div>
```

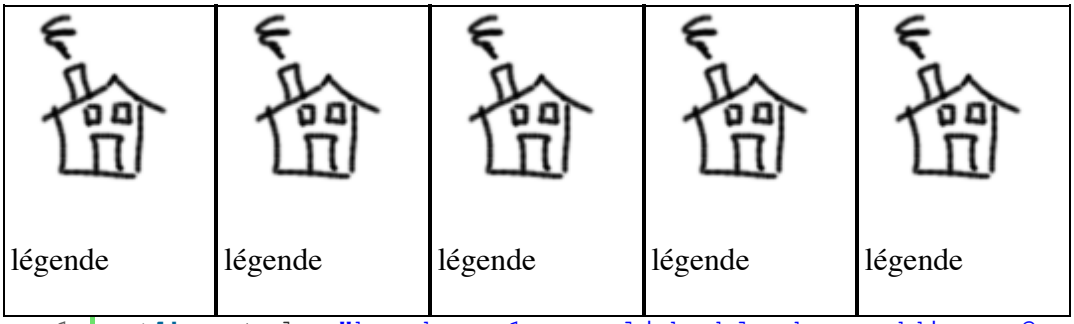
Utilisation de clear pour inclure les éléments dans un cadre

On met parfois un div vide avec "clear: both" pour forcer l'élément à s'agrandir pour contenir les flottants.


un essai.

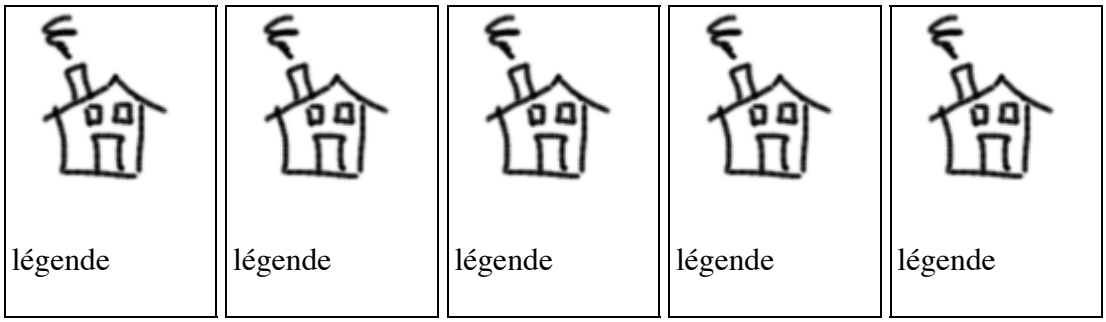
Astuces de mise en page

Float permet de créer une liste d'éléments complexes côte à côte.


légende légende légende légende légende

```
<div style="border: 1px solid  black; padding: 3px; float:left;">
 <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
<div style="border: 1px solid  black; padding: 3px; float:left;">
 <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
<div style="border: 1px solid  black; padding: 3px; float:left;">
 <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
<div style="border: 1px solid  black; padding: 3px; float:left;">
 <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
<div style="border: 1px solid  black; padding: 3px; float:left;">
  <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
```

autre solution pour le même effet : inline-block (mais "certains" navigateurs le gèrent mal)




```
<div style="border: 1px solid  black; padding: 3px; display:inline-blo
  <img src="coursCSSdata/dessin.png"/> <p> légende</p></div>
```

Combinaison de plusieurs propriétés

Généralement, utilisation simultanée de float, position, margin, etc...

Exemple

Menu en float à gauche. Largeur fixeLe contenu

a

a

a

a

```
  <div>
  <div style="background-color: red; width: 15 em; height: 12ex;float:
    Menu en float à gauche. Largeur fixe</div>
  <div style="background-color: lightblue;"> Le contenu</div>
  <div style="clear: both"></div>
</div>
```

Menu en absolu. Largeur fixe (15em)Le contenu. Marge de 16 em.

du texte

du texte

```
<div style="margin-top: 1em; position: relative;">
  <div style="background-color: red; width: 15em; height:
  100%; position: absolute; margin: 0em; padding: 0em;"> Menu en absolu
  <div style="background-color: lightblue; margin-left: 15em;"> <p>Le co
  16 em. <p> du texte <p> du texte</div>
</div>
```

Utilisation de display:inline-block;

- Remplace certains usages de float ;
- support IE : >= 8
- Se conduit comme un élément *inline* par rapport à son parent, et comme un bloc par rapport à ses enfants.
- positionnement vertical : combiner avec `vertical-align` ; éventuellement utiliser min-height/min-width

Un premier essai, avec une largeur fixe, juste pour voir comment ça marche;

Un second bloc, sans largeur donnée.

du texte tout simple

dernier bloc.

```
<style type="text/css">
  #inlineblocdemo div {
      display: inline-block;
      border: solid 2pt;
      vertical-align: top; padding:
      0.5em;
  }
</style>
<div id="inlineblocdemo">
  <div style="width: 15em;">
    Un premier essai, avec une largeur fixe, juste pour voir comment ça
  </div>
      <div >
    Un second bloc, sans largeur donnée.
  </div>
  du texte tout simple
      <div >
    dernier bloc.
  </div>
</div>
```