Feuilles de style CSS

Technologies du Web 1

Jean-Christophe Routier
Licence 1 SESI
Université Lille 1

Tout cela manque quand même de style

Chaque élément HTML est affiché par le navigateur avec un style par défaut.

sansCSS.html ⊙

- les éléments <h1> apparaissent avec une police de taille *2em* et « en gras »
- les éléments <code> sont affichés avec une police à chaque fixe.
- les éléments <p> forment des blocs qui s'affichent les uns en dessous des autres avec une marge haute et basse de *1em* et leur texte est affiché avec la police par défaut du navigateur
- les éléments <q> sont encadrés par des guillemets
- les éléments sont en gras
- etc.

CSS

Cascading Style Sheets

Il est possible de modifier ce style grâce aux **CSS**

CSS = Cascading Style Sheets = Feuilles de style « en cascade »

Un concept important

Concept

La séparation de la forme et du contenu.

- la structure d'un document (et son contenu) est décrite en HTML
- sa présentation est gérée par les CSS

1. on crée le document (contenu et structure) sans se préoccuper de sa mise en forme
2. on conçoit la (les !) mise(s) en forme
   puis éventuellement on les modifie/adapte

« CSS Zen garden »
`http://www.csszengarden.com/`

Avantages

- document HTML et feuille CSS peuvent être définis dans des fichiers séparés
- construction du document (HTML) sans se préoccuper de son rendu visuel
  - création plus efficace
  - code HTML plus simple et plus lisible
  - on peut changer la feuille de style sans changer le document (évolution du « look »)
  - on peut avoir plusieurs feuilles de style pour un document
    - sélection selon le « media »
    - accessibilité
- l'homogénéité visuelle d'un site est facilitée
  - plusieurs pages peuvent partager la même feuille de style

Principe

- le langage CSS définit un ensemble de **propriétés** qui ont une influence sur l'affichage des éléments d'une page
- à chaque propriété correspond un ensemble de **valeurs** possibles
- il est possible de fixer ces propriétés pour chacun des éléments d'un document HTML
- les propriétés définissent l'apparence de la **boîte** d'un élément
- les propriétés concernent
  - l'apparence du contenu (fonte, style, couleur, ...)
  - la taille de la boîte (largeur, marges, ...)
  - le positionnement de la boîte (absolu ou relatif, visibilité)
  - ...

Règle CSS

Règle CSS

Une **règle CSS** définit pour un sélecteur une propriété CSS et sa valeur.

```
selecteur : { propriete : valeur }
```

Le **sélecteur** détermine les éléments sur lesquels s'applique la règle.

- une feuille de style CSS contient un ensemble de règles CSS

- il est possible de regrouper plusieurs règles d'un même sélecteur
  les définitions sont alors séparées par des points-virgules

```
h1 {
   color : blue;
   font-size : 12px;
}
```

« tous les éléments <h1> auront leur texte en bleu et une taille de police de 12px »

- on peut factoriser les règles partagées par des sélecteurs
  les sélecteurs sont alors séparés par des virgules

```
h1, h2 {
   color : blue;
   font-size : 12px;
}
```

*« les éléments <h1> **et** les éléments <h2> auront leur texte en bleu et une taille de police de 12px »*

Intégration des règles CSS à l'HTML

Différentes possibilités :

- sans CSS ⊙
- CSS dans le code HTML (beurk) ⊙ : à proscrire !
- règles CSS "en dur" ⊙ : bof, limitant
- feuille de style externe : la solution à adopter
  - style 1 ⊙ – la feuille css ⊙
  - style 2 ⊙ – la feuille css ⊙

Dans l'entête (<head>) du document HTML :

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

ou aussi :

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

Exemples de propriétés

- `font-family` : le type de police utilisée pour le contenu,
- `font-size` : la taille des caractères (en px, em, %, etc.)
- `font-style` : *normal*, *italic*, *oblique*
- `font-weight` : *normal*, *bold*, *lighter*, etc.
- `border` : la bordure autour du contenu de l'élément (couleur, style, ...)
- `width` : largeur du contenu (%, px, em, cm)
- `color` et `background-color` : couleurs du texte et de l'arrière-plan (*rgb(0,128,255)*, hexa *#AAAAAA*, symboles prédéfinis (*navy*, *white*, ...), hsl,)
- etc. liste ⊙ exemple ⊙

validation css : `http://jigsaw.w3.org/css-validator/`

Cascade

- plusieurs feuilles de style sont possibles pour un même document
- certaines règles s'appliquent selon les médias
- il peut y avoir des règles en conflit (portant sur les mêmes éléments)

Cascade

Le mécanisme de **cascade** détermine les règles appliquées.

3 étapes de filtre :

1. par média
2. par origine
3. par spécificité des sélecteurs

Médias

- possibilité de préciser le media dans auquel s'applique les règles définies dans la feuille style
- attribut `media` de la balise <`link`>

  ex : `media="screen"` – `media="print"`

Origine des styles

- 3 origines possibles pour les feuilles de style

  auteur définies l'auteur de la page

  utilisateur définies par celui qui consulte la page

  navigateur définies par le navigateur (*agent utilisateur*)

  En général : *auteur* > *utilisateur* > *navigateur*

nuancé par usage du mot-clé `!important`

plus de détails : `http://openweb.eu.org/articles/cascade_css`

Sélecteurs

Sélecteur

Le **sélecteur** détermine les éléments sur lesquels s'applique la règle.

nécessité de savoir comment

- définir les sélecteurs appropriés
- sont gérées les priorités entre règles en conflit

Sélecteurs simples :

E tout élément dont la balise est <E>

* tout élément

Sélecteurs et attributs

E[att] tout élément **E** dont l'attribut `att` est défini
ex : `p[lang]`, `img[alt]`, `*[title]`

E[att=val] tout élément **E** dont l'attribut `att` vaut *val*
ex : `p[lang=fr]`

E[att~=val] tout élément **E** dont l'attribut `att` est une liste de mots séparés par des espaces, l'un de ces mots vaut exactement *val*

E[att^="prefixe"] tout élément **E** dont la valeur de l'attribut `att` se termine exactement par *prefixe*
ex : `a[href^="http://fil.univ-lille1.fr"]`

E[att$="suffixe"] tout élément **E** dont la valeur de l'attribut `att` commence exactement par *suffixe*
ex : `img[src$=".png"]`, `a[href$=".pdf"]`

E[att*="val"] tout élément **E** dont la valeur de l'attribut `att` contient la sous-chaîne *val*
ex : `figure[alt*="diagramme"]`, `*[title*="timoleon"]`

Sélecteur de classe et d'id

Cas particuliers des attributs `class` et `id` :

E.c tout élément **E** appartenant à la **classe c**
équivalent à **E[class~=c]**

ex : `div.exercice`, `*.solution`, `div.rmq[title^="NB"]`

E#ident tout élément **E** dont l'**id** vaut **ident**
équivalent à **E[id=ident]**

ex : `img#joconde`, `*#joconde`, `#unique`

Sélecteurs de pseudo-classes et pseudo-éléments

E:pseudoC tout élément **E** appartenant à la **pseudo-classe** *pseudoC*

ex : `a:visited`, `a.fichier:hover`

E::pseudoE tout **pseudo-élément** *pseudoE* de l'élément **E**

ex : `h1::first-letter`, `p[lang=fr]::first-line`

pseudo-classes et pseudo-éléments présentés plus loin

Combinaison de sélecteurs

- s'appuie sur la structure arborsecente du document

si **Sel1** et **Sel2** sont des sélecteurs :

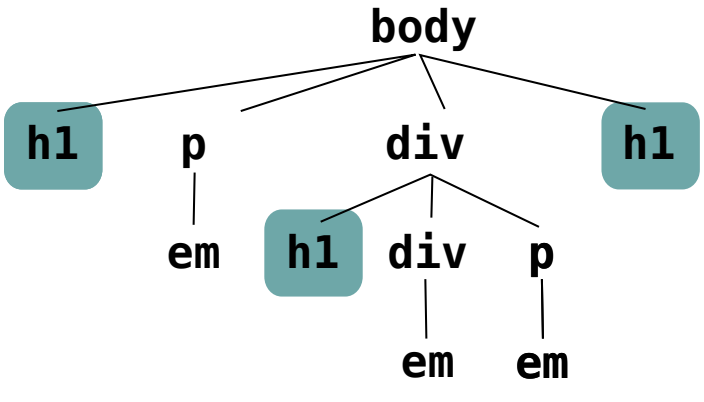
Sel1 Sel2 tout élément sélectionné par **Sel2** emboité dans un élément sélectionné par **Sel1**

Sel1 > Sel2 tout élément sélectionné par **Sel2** qui est **fils** d'un élément sélectionné par **Sel1**

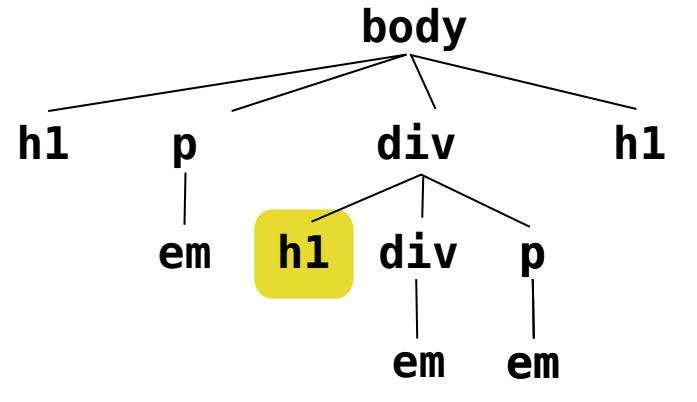
Sel1 + Sel2 tout élément sélectionné par **Sel2** qui **suit immédiatement** un élément sélectionné par **Sel1**

Sel1 ~ Sel2 tout élément sélectionné par **Sel2** qui **suit** un élément sélectionné par **Sel1**

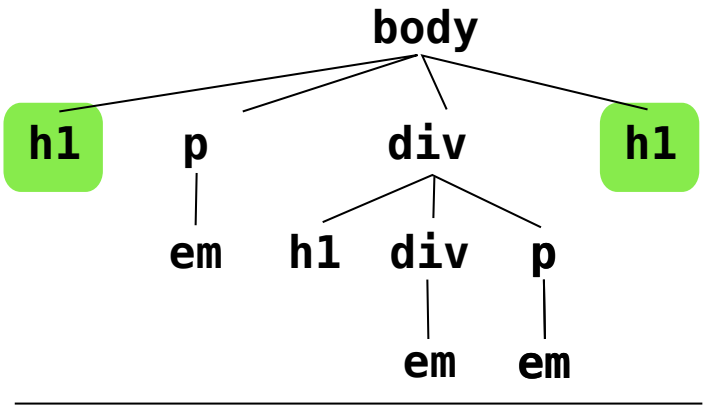
Exemples



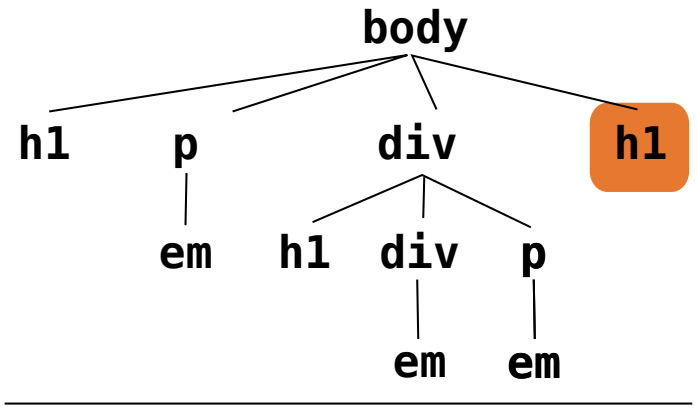
h1 *ou* body h1



div h1



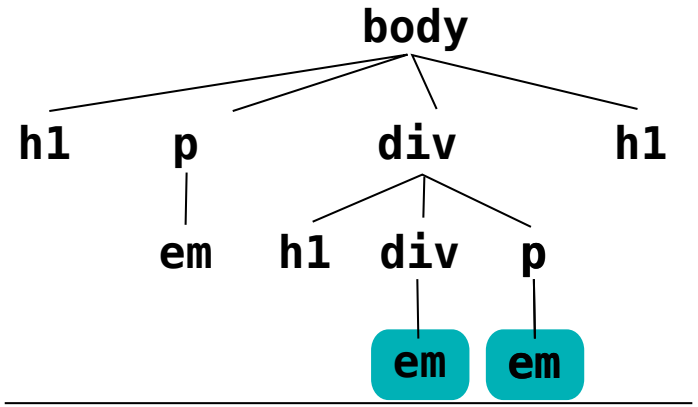
body>h1



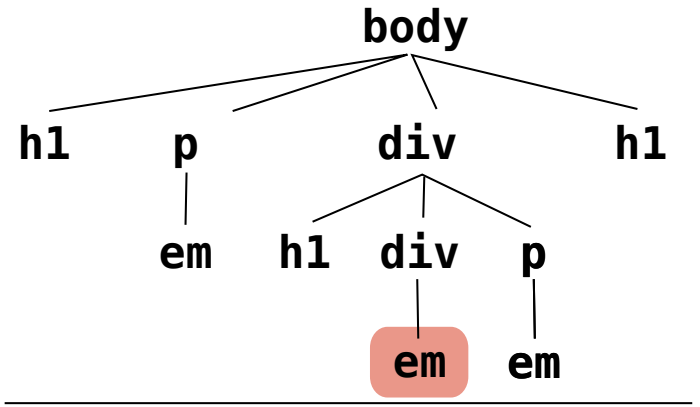
div+h1

Crédits figures Bruno Bogaert

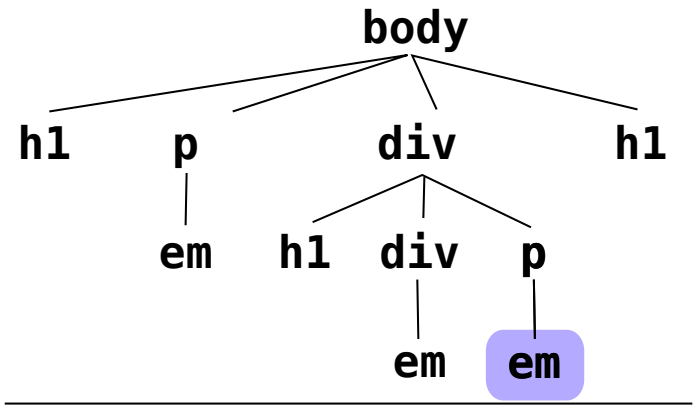
Exemples



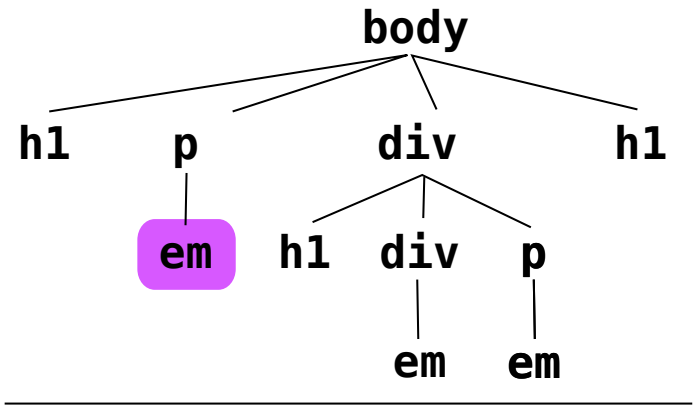
div em



div>em



div p em



body>p em

Crédits figures Bruno Bogaert

Conflit entre règles

Calcul de priorité

On compte pour chaque sélecteur :

- **a** nombre de sélecteurs d'id (= nombre de #)
- **b** nombre de classes, pseudo-classes ou d'attributs
- **c** nombre d'éléments ou de pseudo-éléments

Le sélecteur reçoit la priorité **a b c**.
Le sélecteur avec la plus grande priorité l'emporte.

En cas d'égalité, la dernière déclaration l'emporte.

Exemples

sélecteur	a	b	c
* {...}	0	0	0
h1 {...}	0	0	1
div.reponse {...}	0	1	1
#joconde {...}	1	0	0
div a {...}	0	0	2
div a:visited {...}	0	1	2
p span.fichier {...}	0	1	2
p a[href$=".pdf"] {...}	0	1	2
p.enonce a[href$=".pdf"] {...}	0	2	2
ol.exercice li.question {...}	0	2	2
div#diaporama img.gauche {...}	1	1	2
article p::first-letter {...}	0	0	3
article#special p::first-letter {...}	1	0	3

```
<p>zero</p>
<h1>Titre</h1>
<p>premier</p>
<p>second</p>
<p>troisième</p>
<p lang="en">fourth </p>
<p class="bleu">cinquième</p>
<p class="bleu">sixième</p>
<p class="bleu" id="special">septième</p>
<p>huitième</p>
<div>
  <p>neuvième</p>
  <p lang="en">tenth</p>
</div>
<p>onzième</p>
<h1>Second titre</h2>
<p>douzième</p>
```

b-g = background-color

```
p { b-g:pink}
h1+p { b-g : red; }
div>p { b-g : yellow; }
p#special { b-g: gold; }
p[lang=en] {b-g : green;}
p.bleu { b-g: lightblue; }
p+p { b-g: lightgreen; }
```

ex-selecteur.html

Héritage

Héritage

- lorsque pour un élément aucune règle ne définit de valeur pour une propriété, c'est la valeur de cette propriété pour son parent qui s'applique

- toutes les propriétés ne s'héritent pas
  ex : `margin`, `padding`, etc
- la propriété `inherit` permet d'agir sur l'héritage

Pseudo-classes

non exhaustif

Structurelles

:empty un élement **E** sans descendant (y compris nœud texte)

:first-child un élément qui est premier fils d'un autre élément
ex : `div.exercice:first-child`

:last-child élément dernier fils d'un autre élément

:nth-child($an+b$) élément $(an+b)$-ème fils d'un autre élément
ex : `div:nth-child(3)`, `div:nth-child(2n)`, `div:nth-child(even)`, `div[idx]:nth-child(3n+1)`

:nth-last-child($an+b$) $(an+b)$-ème fils en partant de la fin

:nth-of-type($an+b$) $(an+b)$nème élément du type sélectionné et qui ont le même père

Pseudo-classes (suite) et Pseudo-éléments

non exhaustif

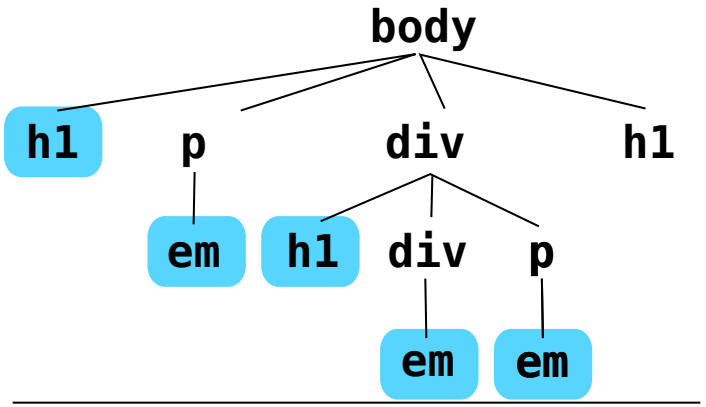
Dynamiques

:hover	est « sous » le pointeur de la souris
:visited	(<a> uniquement) lien déjà visité
:link	(<a> uniquement) lien non encore visité

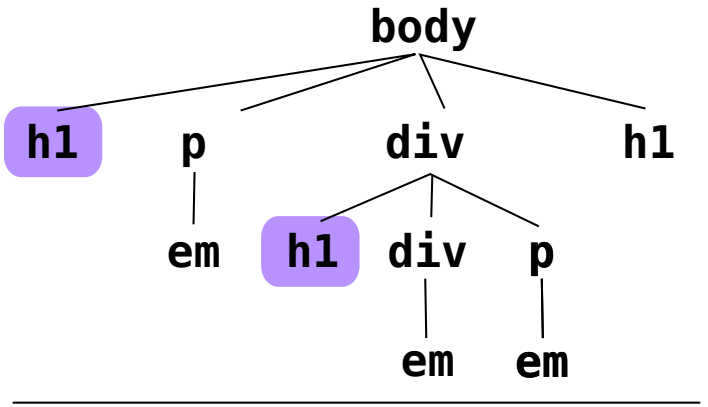
Pseudo-éléments

::first-line	la première ligne « formatée » d'un élément
::first-letter	le première lettre « formatée » d'un élément
::before	insertion de contenu avant l'élément
::after	insertion de contenu après l'élément

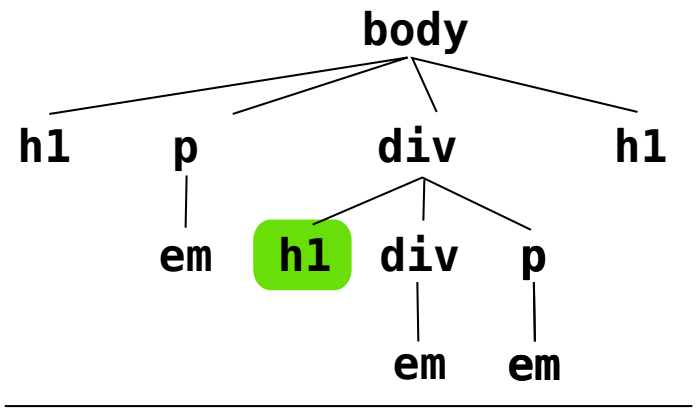
Exemples



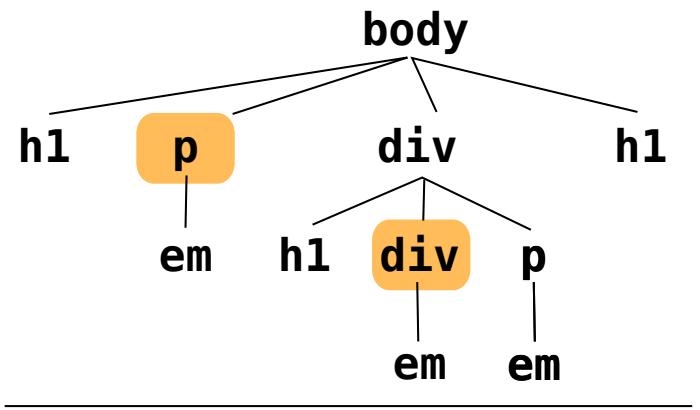
:first-child



h1:first-child



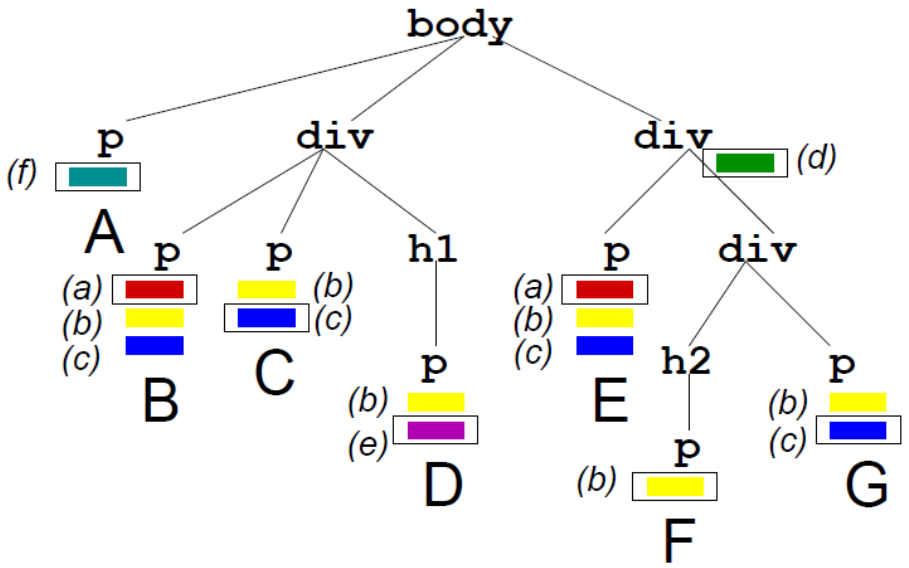
div>h1:first-child



h1:first-child+*

Crédits figures Bruno Bogaert

Exemple



(a)	`div>p:first-child`		0 1 2
(b)	`div p`		0 0 2
(c)	`div>p`		0 0 2
(d)	`div+div`		0 0 2
(e)	`div h1 *`		0 0 2
(f)	`body>p`		0 0 2

Crédits figures Bruno Bogaert