WIM 1.1

Cascading Style Sheets 2 et 3

Denis Monnerat

monnerat@u-pec.fr

Updated: 2017/11/22

IUT de Fontainebleau

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

IUT
SENART
FONTAINEBLEAU

Sommaire

1. Introduction

2. Css

3. Quelques éléments de présentations

Introduction

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de propriétés d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de propriétés d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

```
<body>
  boîte body
  <div class="c1">
    boîte div
    <p>boîte p<span>un texte
        dans sa
        boîte</span>
    </p>
    <ul>
      <li>boîte li 1</li>
      <li>boîte li 2</li>
    </ul>
  </div>
</body>
```

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de propriétés d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de propriétés d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Règle de sélection et propriétés : le langage CSS permet de :

- sélectionner un/des éléments,

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

```
<body>
  <p>un paragraphe</p>
  <p class="exemple">un autre
    paragraphe selectionné
    par la règle
  </p>
</body>
```

Principes généraux

Boîtes et flux

- Chaque élément du document html génére une zone (boîte) munie de propriétés d'affichage : marges, bordure, arrière-plan, largeur, etc.
- Ces boîtes peuvent être déplacées par rapport à leur position par défaut dans le flux.

Règle de sélection et propriétés : le langage CSS permet de :

- sélectionner un/des éléments,
- modifier les valeurs de certaines propiétés pour les éléments selectionnés.

Séparer le fond de la forme

html avec des éléments de présentation.

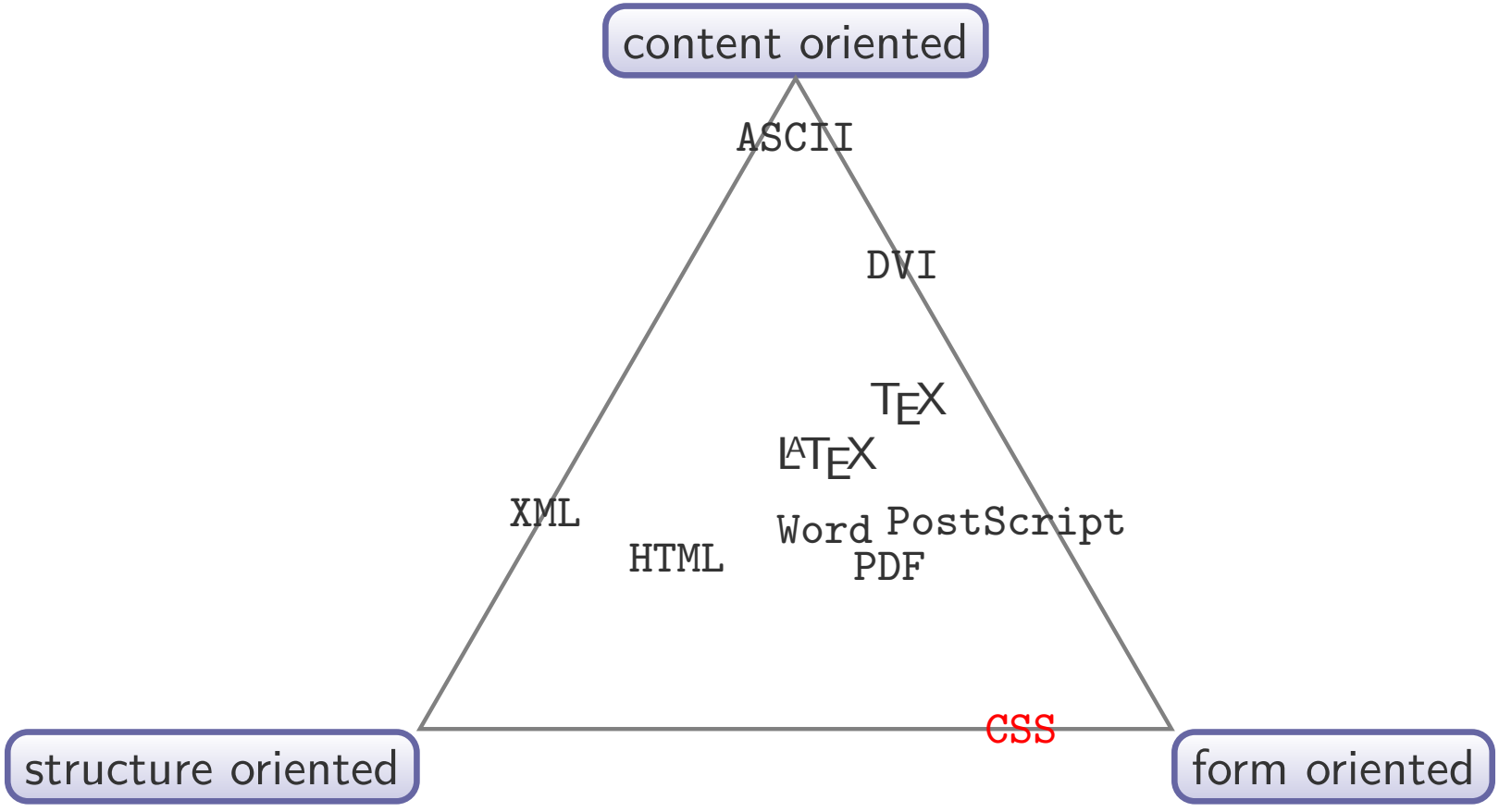
```
<br/><br/>
<b><font size="+2">
    Notre famille
</font></b>
<br/><br/>
<font size="+1">
  Michele et Jean-Luc
</font>
```

html sans éléments de présentation. La mise en page est contrôlée ailleurs.

```
<div id="ma_famille">
  <h3>Notre famille</h3>
  <p>Michele et Jean-Luc</p>
</div>
```

La seconde version est bien meilleure.

CSS s'occupe de la mise en forme.



Mise en cascade et héritage

Mise en cascade des styles.

- Imbrication de plusieurs styles avec des priorités différentes.
- Les propriétés non définies pour la balise courante **sont héritées** des balises parentes (quand cela est possible).
- un style défini directement dans la page est prioritaire sur un style défini dans un fichier.

Intérêts

Principes des Cascading Style Sheets.

- Distinction entre structuration et mise en forme.
- Appliquer des règles de styles (règles de mise en forme) aux balises html (sans en changer le contenu).

Avantages

- Réutilisation et partage.
- Homogénéisation.
- Amélioration de l'accéssibilité.
- Adaptation aux média.
- Modification du contenu et/ou de la présentation indépendante.

Où trouver de l'information

Plusieurs niveaux de recommandations pour CSS : 1,2 et 3.

W3C

`http://www.w3.org/TR/CSS/`

Le niveau 3 est en discussion. La suite concerne surtout CSS2.

Réferences sur le web

- `http://www.w3schools.com/`,
- `http://fr.selfhtml.org/`

Css

Dans l'html

```
<h1 style="color:green">...</h1>
<p style="color:red">...</p>
```

```
<head>
  <style type="text/css">
    p{
      font-size:16pt;
      color:blue;
    }
  </style>
</head>
```

À éviter !

A l'extérieur

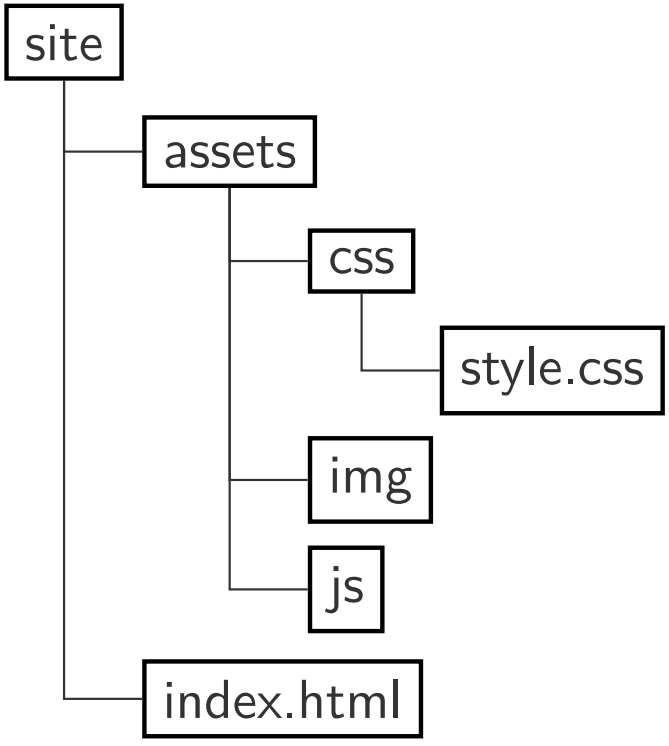
Stockage dans un/des fichiers à part, chargé(s) avec la page.

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

Règle CSS2 `@import` (peuvent se trouver dans un fichier css)

```
<style type="text/css">
  @import "generales.css";
  @import url("avancees.css");
  @import url("impression.css") print, embossed;
  @import url("portable.css") handheld;
  @import url("normal.css") screen;
</style>
```

Organisation classique du code d'un site



CSS resets

Chaque navigateur a ses propres styles par défaut pour rendre les titres, paragraphes, listes, etc.

Pour assurer la compatibilité d'un site avec les différents navigateurs, les CSS resets sont largement utilisés. On peut citer

Le reset css d'Eric Meyer :

```
http://meyerweb.com/eric/tools/css/reset/
```

Normalize.css

```
https://github.com/necolas/normalize.css
```

Le css reset de YUI (yaohoo)

```
http://yuilibrary.com/yui/docs/cssreset/
```

Il faut les placer en tout premier, de manière à ce qu'ils s'appliquent d'abord.

Notion de règle de sélection

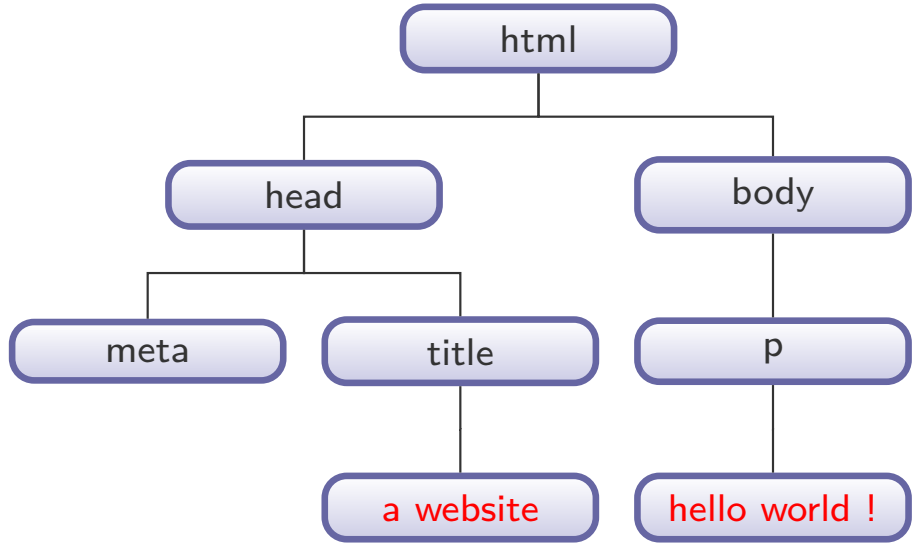
Une déclaration (**Règle**) en css regroupe 3 parties :

- un selecteur,
- une propriété,
- et une valeur.

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

```
body {color:black;}
p {font-family : "helvetica";}
p {text-align :center;color:red;}
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>un site</title>
  </head>
  <body>
    <p>hello world !</p>
  </body>
</html>
```



Les règles de sélection utilisent la représentation arborescente d'un document html avec les notions de descendants, de fils, de frères, etc.

Sélection

Appliquer un style à toutes les balises **selecteur universel** * :

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

Appliquer un style à une balise particulière : **selecteur de type** :

```
h1,h2,h3{
  width:100%;
}
```

```
li p{
  text-align:center;
}
```

Sélection : classes de style

Classe particulière pour un élément particulier :

```
p.right{text-align:right}
p.left{text-align:left}

<p class="right">droite</p>
<p class="left">gauche</p>
```

Classe générale pour un élément particulier.

```
p{font-size:16pt;}
```

Classe particulière pour n'importe quel élément.

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

On peut cumuler les classes !

Sélection id

On peut selectionner un élément unique grâce à son identifiant dans une règle.
Des mécanismes plus avancées permettent d'affiner la sélection ...

```
<style>
#vert{color:green;}
p#para1{color:red;}
</style>

<h1 id="vert">
  titre en vert
</h1>

<p id="para1">
  paragraphe en rouge
</p>
```

Sélecteur de descendants

Descendant (imbrication) :

```
li p{
  text-align:center;
}
```

```
div * p {text-align:center}
```

Expliquez ?

```
.menu li .sel {
font-size : 1em;
}
```

Sélecteur d'enfants

Descendants directs (les fils) :

```
#menu > p {color : green;}
```

S'applique à tous les p fils de l'élement `#menu`.

Enfants adjacents : de la forme E1 + E2. E2 est le sujet du selecteur ; vérifié quand E1 et E2 sont adjacents dans l'arbre.

```
h1 + h2 { margin-top: -5mm }
h1.opener + h2 { margin-top: -5mm }
h2 + p { margin-top:0.8em }
```

Sélecteur d'attributs

Avec CSS2, on peut selectionner des éléments par leurs attributs. Les sélecteurs d'attributs peuvent trouver une correspondance de quatre façons :

- `att` : quand un élément a un attribut "att", quelle que soit sa valeur ;
- `att=val` : quand un élément a un attribut "att" dont la valeur est exactement "val" ;
- `att~=val` : quand un élément avec un attribut "att" qui contient val
- `att|=val` : quand un élément avec un attribut "att" qui commence par val .

```
h1[title] { color: blue; }
```

```
a[rel~="copyright"]
a[href="http://www.w3.org/"]
```

```
*[lang|="en"] { color : red }
```

Remarque : pour la selection de l'attribut class, pour les documents html, on utilise aussi la notation pointée.

Pseudo-éléments et pseudo-classes

CSS introduit les concepts de pseudo-éléments et de pseudo-classes qui permettent une mise en forme à partir d'informations absentes de l'arbre du document.

- Les pseudo-éléments créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document.
- Les pseudo-classes classent les éléments selon des caractéristiques autres que leur nom, attribut ou contenu, celles-ci ne pouvant pas en principe être déduites de l'arbre du document. Les pseudo-classes peuvent être **dynamiques**.

Pseudo-classes

- `:first-child`,`:last-child`,`:nth-child(n)` : correspond au premier, dernier, nièeme élément enfant d'un autre élément :

```
div > p:first-child {text-indent : 0;}

* > a:first-child    /* a est le premier enfant pour
                        tout element */
a:first-child        /* Idem */
```

- Pseudo-classes d'ancre : `:link` et `:visited`
  - La pseudo-classe :link s'applique aux liens qui n'ont pas été visités ;
  - La pseudo-classe :visited s'applique lorsque le lien a été visité par l'utilisateur.
  - Etats exclusifs !

```
a:link { color: red }
:link  { color: red }   /* idem */
```

- Pseudo-classes dynamiques : `:hover`, `:active` et `:focus`
  - `:hover` : qui est appliquée quand l'utilisateur désigne un élément (au moyen d'un appareil de pointage) sans l'activer.
  - `:active`, qui est appliquée quand l'utilisateur active un élément. Par exemple, entre le moment où l'utilisateur presse le bouton de la souris et le relâche.
  - `:focus` , qui s'applique quand un élément reçoit l'attention (celui-ci acceptant les événements du clavier ou d'autres formes d'entrées de texte).
- Ces pseudo-classes ne s'excluent pas mutuellement
- Les éléments ne sont pas nécessairement des liens.

pseudo-éléments

- `:first-letter`, `:first-line`.

```css
p:first-letter {
  font-size: 200%;
  font-style: italic;
  font-weight: bold;
  float: left
}
```

- `:before`, `:after`

```css
h1:before {
  content: "Chapitre " counter(chapitre) ". ";
  counter-increment: chapitre;  /* Ajoute 1 au chapitre */
  counter-reset: section;       /* Remet la section a zero *
}
```

Calcul de la priorité d'un selecteur

c'est un nombre à 4 chiffres *abcd*.

- *a* : compte 1 si la déclaration provient d'un attribut style, 0 sinon.
- *b* : compte le nombre d'attributs `id` dans le selcteur.
- *c* : compte le nombre d'autres attributs et de pseudo-classes dans le selecteur.
- *d* : compte le nombre d'éléments et de pseudo-éléments dans le selecteur.

```
*               {}  /* a=0 b=0 c=0 d=0 -> specificity = 0,0,0,0 */
li              {}  /* a=0 b=0 c=0 d=1 -> specificity = 0,0,0,1 */
li:first-line {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
ul li           {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
ul ol+li        {}  /* a=0 b=0 c=0 d=3 -> specificity = 0,0,0,3 */
h1 + *[rel=up]{}  /* a=0 b=0 c=1 d=1 -> specificity = 0,0,1,1 */
ul ol li.red  {}  /* a=0 b=0 c=1 d=3 -> specificity = 0,0,1,3 */
li.red.level  {}  /* a=0 b=0 c=2 d=1 -> specificity = 0,0,2,1 */
#x34y           {}  /* a=0 b=1 c=0 d=0 -> specificity = 0,1,0,0 */
style=""            /* a=1 b=0 c=0 d=0 -> specificity = 1,0,0,0 */
```

Liste des propriétes CSS

- Couleurs
- Arrière-plan, bordures
- Boîtes standards
- Boîtes flexibles
- Texte
- Decoration (Texte)
- Polices
- Modes d'écriture
- Table
- Listes et compteurs
- Animation
- Transformation
- Transition
- Interface utilisateur
- Multi-colonne
- Paged Media
- Generated Content
- Filter Effects
- Image/Replaced Content
- Masking
- Speech
- Marquee

`http://www.w3schools.com/cssref/`

- 3 nouveaux selecteurs d'attributs :

att^ =val : quand un élément a un attribut "att" qui commence par val.
att$=val : quand un élément a un attribut "att" qui finit par val.
att*=val : quand un élément a un attribut "att" qui contient par val.

- Adjacence indirecte : ~ : tous les frères suivants.
- Pseudo classes :
  `:nth-child(),:nth-last-child(),:last-child,:checked`
  `:empty, :not`
- Pseudo-éléments : `::selection` représente la selection de l'utilisateur.

```
body :not(p){
color:red;
}
p:nth-child(2) {
    background: #ff0000;
}
p:nth-of-type(2) {
    background: #ff0000;
}
p ~ div {
padding : 10px;
}
```

CSS3

- Ombrages sur le texte et les boîtes : `[box|text]-shadow`.
- Coins arrondis : `border-radius`
- Transparence : `rgba(...)`.
- Positionnement `inline-block`.
- Transformation géométrique : `transform`
- Media Queries.

un lien `http://www.goetter.fr/`

Responsive web design et Media Queries

Application de feuilles de styles en fonction des périphériques clients : **Responsive Web Design**.

En CSS2 :

```
<!doctype html>
<head>
  <meta charset="utf-8">
  <title>Media Queries !</title>
  <link rel="stylesheet" media="screen" href="screen.css"
    type="text/css" />
    <link rel="stylesheet" media="print" href="print.css"
    type="text/css" />
</head>
```

On peut intégrer dans une feuille de style la selection du media avec la règle

```
@media print {
  #menu, h1 {
    display:none;
  }
}
```

En CSS3, on étend cette possibilité en intégrant les capacités du matériel : taille, orientation, etc.

```
@media screen and (min-width: 200px) and (max-width: 640px) {
  .bloc {
    display:block;
    clear:both;
  }
}
```

Media Queries

```
<!-- Media Query CSS dans un élément link -->
<link rel="stylesheet" media="(max-width: 800px)"
                      href="example.css" />

<!-- Media Query CSS dans une feuille de style -->
<style>
  @media (max-width: 600px) {
    .facet_sidebar {
      display: none;
    }
  }
</style>
```

Syntaxe de `@media`

- Utilisation d'expressions booléennes :
  - `and` : et
  - `not` : non
  - `,` : ou
- Types de media :

Type	Description
braille	Pour aveugle
embossed	Imprimantes en braille
handheld	Tenu à la main (ex : téléphone)
print	Impression
projection	Projecteurs
screen	Écran
speech	Synthèse vocale
tty	Terminal à chasse fixe
tv	Téléviseur
all	Tous

Syntaxe : types de fonctionnalités

Propriété	Valeur	Description
color	Entier	Nombre de couleurs (bits)
color-index	Entier	Nombre d'entrés dans la palette
device-aspect-ratio	Entier/Entier	Aspect ratio
device-height	Booléen	Hauteur (appareil)
device-width	Entier	Largeur (appareil)
grid	Entier	Vrai pour un appareil à grille
height	Entier	Hauteur (surface)
max-device-width	Entier	Largeur (appareil)
monochrome	Entier	Nombre de bits
orientation	landscape/portrait	Orientation de l'écran
resolution	Résolution	Résolution
scan	progressive/interlaced	Pour tv
width	Entier	Longueur

Bonnes pratiques

Quelques idées à prendre en considération :

- Menus non affichés par défaut sur terminaux de petite taille (utilisation de js ou de pages séparées).
- Éviter l'utilisation du zoom ou du défilement, peu pratique pour la navigation.
- Privilégier les tailles relatives, en % ou em ou rem, plutôt qu'absolues (pt, px).
- Privilégier des images vectorielles (format svg), sinon, aspect pixelisé en cas d'agrandissement.

Quelques éléments de présentations

Le modèle des boîtes

- Tous les éléments de la page (caractérisés par des balises) sont lus dans leur ordre d'apparition dans le flux HTML et sont positionnés dans des boîtes les unes à la suite des autres sur l'écran.
- Toutes ces boîtes sont par défaut placées relativement les unes par rapport aux autres, c'est à dire à la suite. En CSS, il est tout à fait possible de sortir une boîte du flux normal pour la placer n'importe où sur la page (position :fixed ; par exemple).
- En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :
  - Les balises inline : c'est le cas par exemple des liens <a>.
  - Les balises block : c'est le cas par exemple des paragraphes .

Propriété `display` : block vs inline

`block`	Un élément de type block va prendre tout l'espace disponible horizontalement et l'élément suivant commencera obligatoirement à la ligne (un peu comme s'il y avait un retour chariot). Les balises block sont disposées les unes en dessous des autres, quelque soient leurs dimensions
`inline`	Les éléments en ligne se placent au fil du texte. Ils servent à donner du sens à certaines parties du texte. La taille d'un élément en ligne est celle de son contenu, par exemple une phrase en italique au milieu d'un paragraphe est un élément en ligne
`inline-block`	mixe des deux
`none`	masque totalement l'élément

Il y a d'autres valeurs possible (liste exhaustive) :

`https://developer.mozilla.org/en-US/docs/Web/CSS/display`

- La largeur et la hauteur ne peuvent être fixées que pour des éléments de type block (à l'exception des éléments en ligne ne contenant pas de texte <img /> <input />...) ;
- Une boîte block peut-être positionnée sur l'axe horizontal et vertical, alors qu'une boîte inline ne peut l'être que sur l'axe horizontal ;
- Les propriétés CSS position :absolute ; et position :fixed ; ne fonctionnement pas en inline ;
- Un élément block (sauf pour les paragraphes <p> et titres <h1>...<h6>) peut contenir d'autres éléments blocks ou en ligne ;
- Un élément en ligne ne peut contenir que d'autres éléments en ligne ;
- Les éléments de type block possèdent des valeurs de marges internes (padding) et externes (margin) non nulles, contrairement aux éléments en ligne.

Un exemple

```
<body>
  boîte body
  <div class="c1">
    boîte div
    <p>boîte p<span>un texte
        dans sa
        boîte</span>
    </p>
    <ul>
      <li>boîte li 1</li>
      <li>boîte li 2</li>
    </ul>
  </div>
</body>
```

Unités et valeurs

Unités de mesure

- unités absolues :

inch (in)	un pouce (2,54 centimètres) ;
millimétre (mm)	un millimètre (0,0394 pouce)
centimétre (cm)	un centimètre (0.394 pouce)
point (pt)	unité liée aux fontes (18pt=0.25in)
pica (pc)	unité liée aux fontes (12pc=18pt)

- unités relatives :

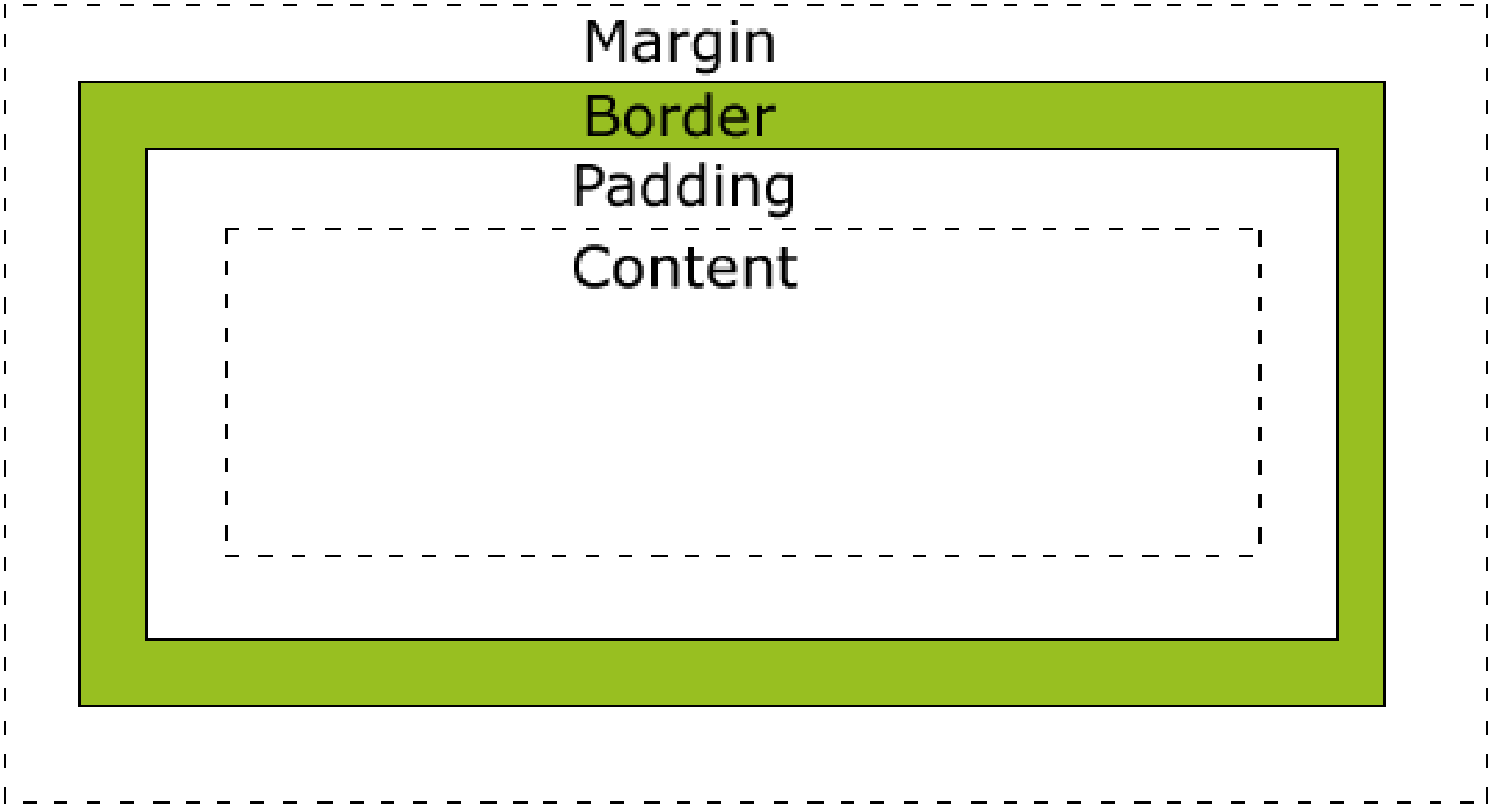
pixel (px)	un pixel (unité physique à l´écran)
em (em)	la valeur de font-size pour le container englobant
ex (ex)	la hauteur d'un x minuscule pour la fonte utilisée pour le container englobant

- autres unités :

pourcentage (%)	pourcentage des dimensions du container englobant
auto (auto)	calcul automatique par le navigateur (margesessentiellement)

Il existe également depuis css3 l'unité rem (root em). On donne une fraction de la taille par rapport à la racine, et non l'élément parent.

Dimensions d'une boîte



Attention : la dimension réelle d'une boîte est la somme de toutes ces grandeurs !

Dimensions d'une boîte

Largeur et hauteur

`height`	hauteur ;
`min-height`	hauteur minimale ;
`max-height`	hauteur maximale ;
`width`	largeur ;
`min-width`	largeur minimale ;
`max-width`	largeur maximale ;
`margin`	espace entre la bordure et le container englobant ;
`padding`	espace de remplissage entre la bordure et le texte ;

Ces informations sont utilisées par le navigateur : en cas de redimensionnement, il affichera une barre d'ascenseur horizontale ou verticale dès que la fenêtre est trop petite.

Marges

Individualisation

- `margin-left: 4px;`
- `margin: 10px 20px 30 px 40px;`

Calcul automatique

```
#contenu
{
width : 600px;
margin : auto;
}
```

Remarque le calcul automatique des marges permet de positionner horizontalement un élément block.

Propiété `box-sizing`

Propriété css3 expérimentale : avec la valeur `border-box`, la taille de l'élément comprend également la padding et le bord.

Positionnement des containers

Types de positionnement : valeur de l'attribut `position`

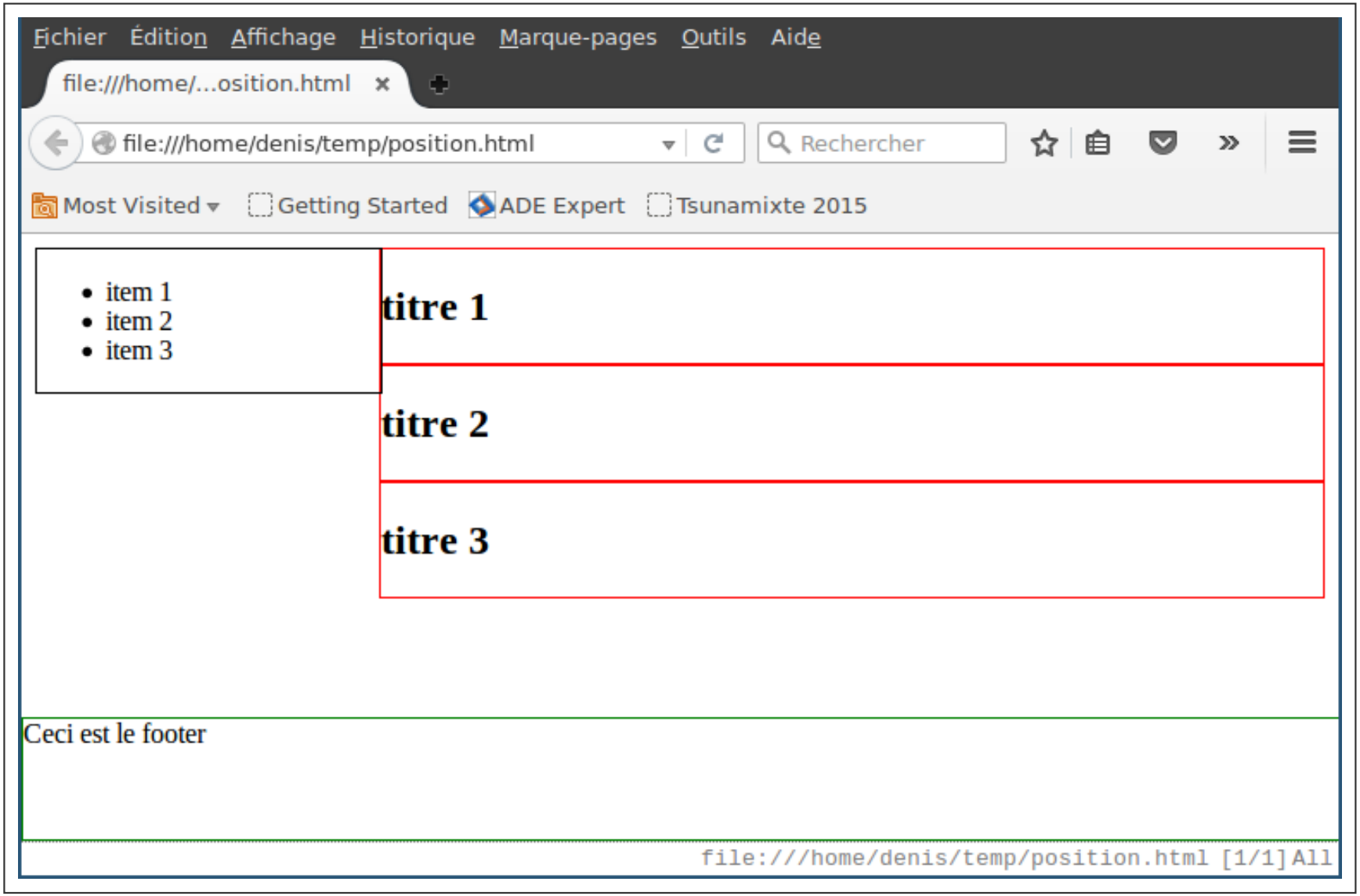
`static`	positionnement par défaut par rapport au container englobant ;
`relative`	positionnement relativement à la position attendue ; l'espace normal reservé dans le flux pour l'élément est maintenu.
`absolute`	positionnement absolu, relatif (!!!) au premier container englobant non static, sera déplacé en cas de déroulement vertical ou horizontal (scrolling) ;
`fixed`	positionnement immuable par rapport à la zone de visualisation, ne sera pas déplacé en cas de déroulement vertical ou horizontal ;

Remarque `float` est une propriété à part entière, indiquant que le positionnement doit se faire en dehors du flux normal d'affichage, nous reverrons cela par la suite...

Un exemple :

```css
body{
  position : relative;
}
nav{
  position : absolute;
  left : 0px;
  width:200px;
  border : 1px solid black;
}
section{
  margin-left : 200px;
  border : 1px solid red;
}
footer {
  position: fixed;
  bottom: 0;
  left: 0;
  height: 70px;
  width: 100%;
  border : 1px solid green;
}
```

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <nav>
      <ul>
        <li>item 1</li>
        <li>item 2</li>
        <li>item 3</li>
      </ul>
    </nav>
    <section>
      <h1>titre 1</h1>
    </section>
    <section>
      <h1>titre 2</h1>
    </section>
    <section>
      <h1>titre 3</h1>
    </section>
    <footer>
      Ceci est le footer
    </footer>
  </body>
</html>
```


Fichier Édition Affichage Historique Marque-pages Outils Aide
file:///home/...osition.html
file:///home/denis/temp/position.html
Rechercher
Most Visited
Getting Started
ADE Expert
Tsunamixte 2015
item 1
item 2
item 3
titre 1
titre 2
titre 3
Ceci est le footer
file:///home/denis/temp/position.html [1/1] All

Positionner un bloc en CSS

Un bloc se positionne par rapport à son conteneur. Le conteneur initial est `body`.

- Mode `inline` ou `block` ou `inline-block`.
- En précisant des marges
- En utilisant la propriété `float` : permet de positionner un bloc à droite ou à gauche de son conteneur, le reste du conteneur occupera l'espace restant.
- en utilisant des positions absolues/relatives
  - `position:relative` : par rapport à l'élément lui-même !
  - `position:absolute` : par rapport au coin supérieur gauche du conteneur. Utiliser `top,left,right,bottom` pour les coordonnées.
- Rendre le bloc visible ou pas : `visibility`.
- Superposer des blocs : `z-index`.
- On peut tronquer (clipper) un bloc avec la propriété `clip`

Postionnement flottant des containers

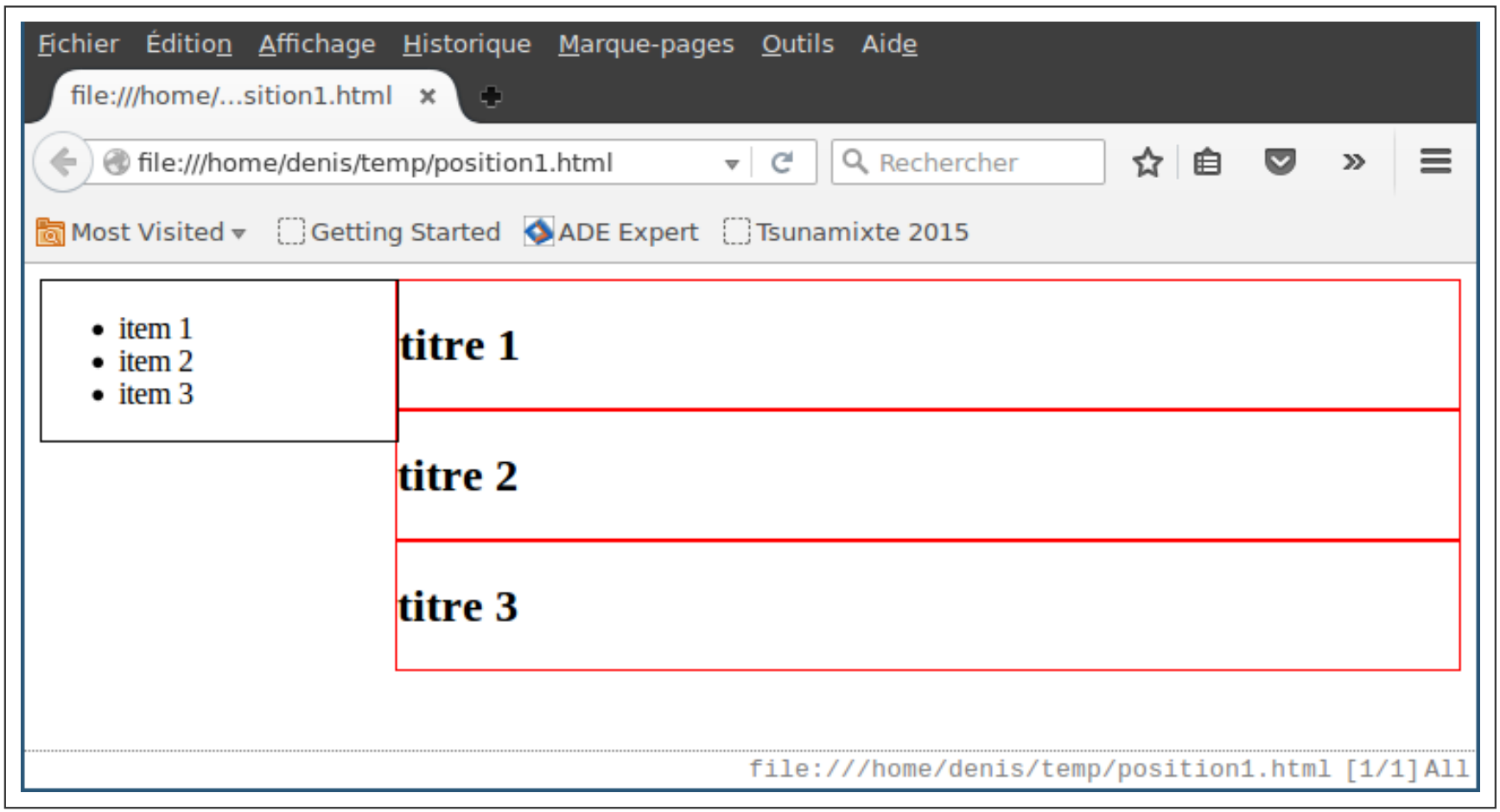
`float: left|right|none`

- le container ayant la propriété `float` sort du flux, il est placé lorsqu'il est rencontré dans le document HTML ;
- il est placé relativement au container englobant ;
- la suite du document se place "autour" du container flottant.

```
<style type="text/css">
  .img{float:left;}
</style>
<body>
  <img class="img" src="linux.jpg"/>
  <p>Vive Linux</p>
</body>
```

Exemple en utilisant des marges en pourcentage :

```
nav {
  float: left;
  width: 25%;
}
section {
  margin-left: 25%;
}
```



Choisir la taille des polices d'affichages

Préférer l'unité `em` par rapport à des tailles unités fixes (`px`,`pt`,...).