

# Positionnement CSS

# Intérêt des feuilles de style

- Meilleur contrôle sur le résultat final
- Meilleure maîtrise de la présentation de la page
- Conception facilitée
- Unification des styles des pages facilitée sans modification du contenu de la page
- Simplification du code HTML

# Les C.S.S

- Permettent de séparer le fond de la forme
- Assurent une compatibilité plus grande
- Permettent une maintenance facilitée

# Comment ajouter des feuilles de styles ?

- Les CSS peuvent être ajoutées de 3 façons :
  - Directement dans le document (en ligne)
  - Dans un bloc de l'entête du document
  - Dans un fichier lié au document
- On peut utiliser les 3 simultanément mais un style sera prédominant (hiérarchie des styles)

# CSS en ligne

- Il suffit d'ajouter l'attribut STYLE à l'élément que l'on veut modifier

```
<ELEMENT STYLE="property:value;  
property:value; ...">...</ELEMENT>
```
- Un style en ligne peut être appliqué à n'importe quel élément HTML et cela modifie seulement l'apparence de l'élément concerné
- Méthode très peu utilisée sauf pour surcharger un style déjà défini pour un type d'élément

# Exemple de CSS en ligne

```
<BODY>  
<H2 STYLE="font-size:26pt;color:red;  
  text-align: center;">  
  titre avec un style</H2>  
<H2>Titre sans style</H2>  
</BODY>
```

**titre avec un style**

**Titre sans style**

# Ecriture des Styles :

Le vocabulaire :

- Style est un attribut universel.

Attention : les balises HTML ont des attributs, les styles ~~ont des propriétés.~~

```
<p align="center">
```

p est le nom de la balise, align un attribut de la balise et "center" sa valeur.

- Ecrit en style, cela donne :

```
<p style="text-align:left;">
```

p est le nom de la balise, style un attribut de la balise et " text-align:left;" sa valeur.

text-align est une propriété du style, et left sa valeur.

# CSS en entête

- On définit un bloc de style dans l'entête délimité par

```
<STYLE TYPE="text/css">...</STYLE>
```

- Le style ainsi défini sera appliqué à tous les éléments du document qui possèdent un style
- Dans le bloc de style on trouve des règles de style portant sur un élément et le style présenté entre accolades "{...}" avec des propriétés énoncés sous la forme

```
propriété : valeur ;
```



# La balise <style>

```
<style
```

```
  type="text/css"
```

```
  title="mes_styles"
```

```
  media="all" >
```

```
p
```

```
{
```

```
  text-align : right;
```

```
  color : black
```

```
}
```

```
</style>
```

• Les informations de la balise sont de type texte (optionnel)

# La balise <style>

```
<style
    type="text/css"
    title="mes_styles"
    media="all" >

p
{
    text-align : right;
    color : black
}

</style>
```

● Nom donné (choisi)  
aux informations de  
style  
(optionnel)

# La balise <style>

```
<style
  type="text/css"
  title="mes_styles"
  media="all" >

p
{
  text-align : right;
  color : black
}
</style>
```

Indique à quel média  
s'applique la feuille  
de style.  
(optionnel)

- all
- screen
- print
- projection
- tv
- braille

# La balise <style>

```
<style
    type="text/css"
    title="mes_styles"
    media="all" >
```

p

élément pour lequel on  
défini le style

{

```
text-align : right;
color : black
```

}

délimiteurs de début et  
de fin de style

```
</style>
```

# La balise <style>

```
<style
    type="text/css"
    title="mes_styles"
    media="all" >
```

```
P
```

```
{
```

```
text-align : right ;
```

```
color : black
```

```
}
```

```
</style>
```

● propriété

● valeur pour  
cette propriété

# La balise <style>

```
<style
    type="text/css"
    title="mes_styles"
    media="all" >
```

P

```
{
  text-align : right ;
  color : black
}
```

h1

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

```
</style>
```

Il est possible de définir  
le style de plusieurs  
éléments

# Exemple de CSS en entête

```
<HTML> <HEAD> <TITLE>CSS en entête</TITLE>  
<STYLE TYPE="text/css">  
h2 {font-size : 26pt ;color : red ;text-align : center;}  
</STYLE>  
</HEAD>  
<BODY>  
<H2>titre avec un style</H2>  
<H2>Titre tjrs avec un style</H2>  
</BODY>  
</HTML>
```

**titre avec un style**

**Titre tjrs avec un style**

# feuille de style externe

- et si le site possède plusieurs pages ?

On crée une feuille de style externe,  
liée à chaque page Web qui l'utilise

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



# feuille de style externe

- Les styles sont dans un fichier séparé
- On précise dans l'entête quel fichier lié utiliser pour le style avec :

```
<LINK REL="STYLESHEET" HREF="style.css">
```

- Dans le fichier de style les règles des styles sont définies selon le modèle donné précédemment
- On utilise l'extension ".css" pour les fichiers de style

# Lier une feuille de style externe

```
<html>
```

```
<head>
```

```
< link rel="stylesheet"
```

```
  href="style.css"
```

```
  type="text/css"
```

```
  media="screen" >
```

```
</head>
```

c'est une  
feuille de style  
qui est liée

# Lier une feuille de style externe

```
<html>

<head>

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

</head>
```

Ou se trouve la feuille de style ?

Ici, dans le fichier `style.css`  
du répertoire courant

# Lier une feuille de style externe

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

Une feuille de style, c'est  
un fichier texte

# Lier une feuille de style externe

```
<html>

<head>

  < link rel="stylesheet"

    href="style.css"

    type="text/css"

    media="screen" >

</head>
```

Pour quel média la feuille de style est-elle valable ?

(optionnel, par défaut : tous)

# L'écriture des feuilles de styles

- `/* */` sert à placer des commentaires dans la feuille de style.
- sélecteur { propriété1:valeur1;  
propriété2:valeur2; }
  - sélecteur : c'est à quelles balise le style s'applique.
  - Entre accolades, l'ensemble des propriétés et de leurs valeurs respectives.

# le fichier `style.css`

- Il comporte les informations de style sur les différents éléments

```
Fichier « style.css »
```

```
P
{
  text-align : right ;
  color : black
}

h1
{ text-align : center }
```

# Exemple de CSS en fichier lié

```
<HTML><HEAD>
```

```
<LINK REL="STYLESHEET" HREF="style.css">
```

```
</HEAD>
```

```
<BODY>
```

```
<H2>titre avec un style</H2>
```

```
<H2>Titre tjrs avec un style</H2>
```

```
</BODY>
```

```
</HTML>
```

```
h2 {
```

```
    font-size:25pt;
```

```
    font-family:arial;
```

```
    color:red;
```

```
    text-align:center;
```

```
}
```

**titre avec un style**

**Titre tjrs avec un style**



# des styles en cascade

Mais au fait, pourquoi  
« en cascade »  
?

# 1<sup>ère</sup> Cascade

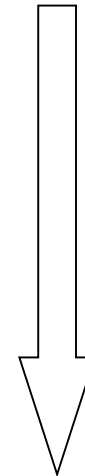
styles définis en cascade  
du plus éloigné au plus proche

Que se passe-t-il en cas de redéfinition en cascade ?

feuille de style externe

balise <style> s'une page

attribut style dans une balise



du – prioritaire  
au + prioritaire

# Quelques propriétés pour commencer ?

Après cette présentation générale, voyons quelques exemples...

- body
- p
- h1, h2, h3, h4, h5, h6

# body

- Couleur du texte & du fond
- Placer une image de fond
- Rajouter des marges

# Un peu de couleur...

- `color`

la propriété `color` permet de fixer la couleur du texte.

- Comment manipuler les couleurs ?
  - liste de couleur standard
  - Un code hexadécimal `#00FF45`
  - Un code rgb, de 0 à 255 `rgb(0,0,100)`
  - un code rgb, en Pourcentage `rgb(10%,10%,30%)`

# Un peu de couleur (suite)

- Liste des couleurs standard :

aqua, black, blue, fuchsia, gray, green,  
lime, maroon, navy, olive, purple, red,  
silver, teal, white, yellow

# Un peu de couleur (de fond)

- `background-color`

la propriété `background-color` permet de fixer la couleur de fond de la page.

```
body
{
  background-color : rgb(0,0,0) ;
  color : white
}
```

# Une image de fond

- `background-image`

permet de spécifier une image de fond

- `background-repeat`

l'image de fond doit-elle être répétée ?

- `background-position`

Où placer l'image de fond ?

- `background-attachment`

L'image de fond bouge-t-elle en même temps que la page ?



background-image

- none
- url()

background-repeat

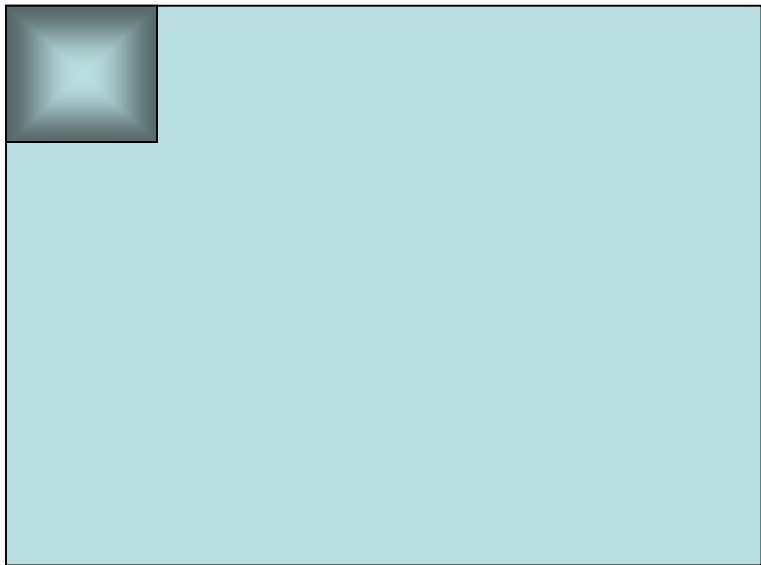
- repeat
- no-repeat
- repeat-x
- repeat-y

background-position

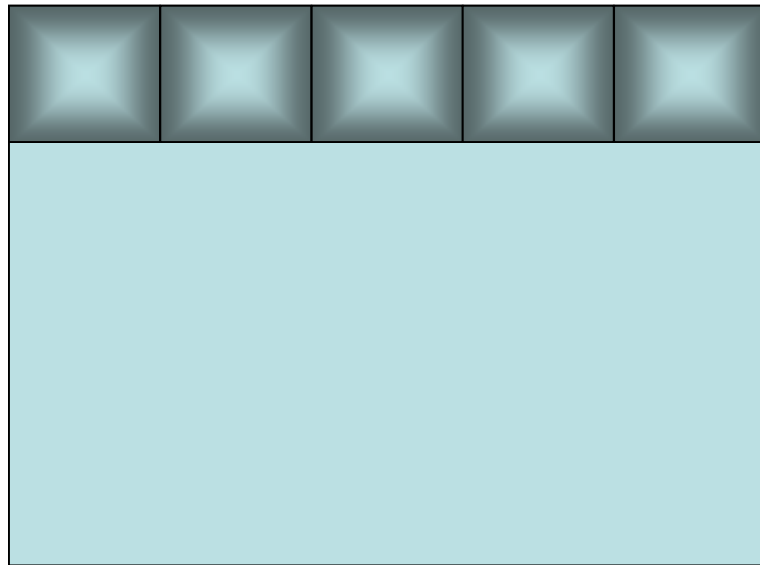
background-attachment

# background-repeat

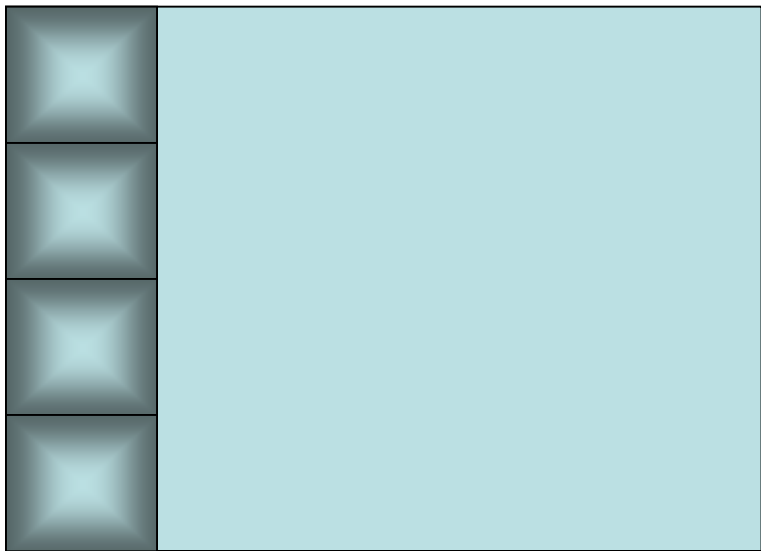
no-repeat



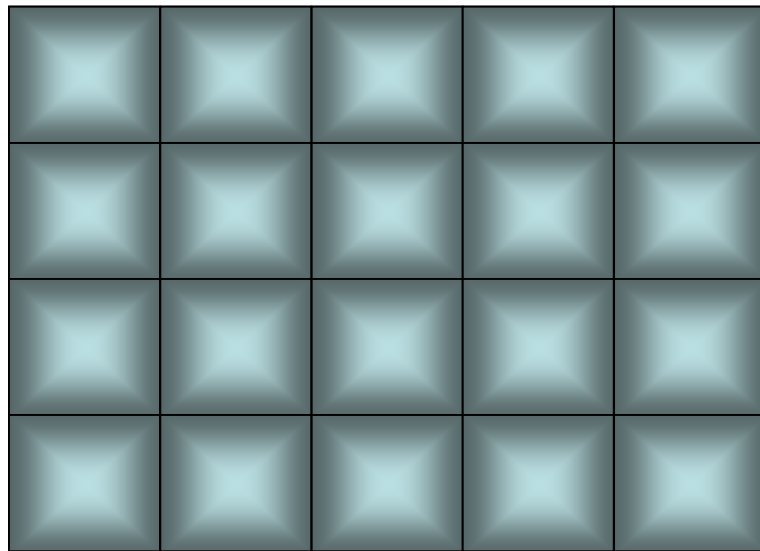
repeat-x



repeat-y



repeat



# Mesures en C.S.S

- % (pourcentage)
  - la taille / position de l'élément est calculée de façon relative
- px (pixel)
  - Le pixel est la plus petit unité de l'écran
- em
  - 1em = 100%, 1.2em = 120%, ...
- pt
  - correspond à une unité d'imprimerie

# Exemple

- HTML :

```
<body>
  <div id="page">
    <div id="en-tete">
      ...
    </div>
    <div id="contenu">
      ...
    </div>
  </div>
</body>
```

CSS :

```
html {font-size: 100%;}
```

```
body {font-size: 0.8em;}
```

```
div#en-tete {font-size: 1.2em;}
```

```
/* On augmente la taille du texte pour l'en-tête */
```

```
div#contenu {}
```

```
/* On ne modifie pas la taille du texte pour le contenu principal.
```

```
  Du coup, le texte du contenu principal garde la taille du texte définie pour BODY */
```

# Quelques propriétés sur les fontes

- `font-size`
  - permet de fixer la taille de la police
  - les unités de mesures vues précédemment sont utilisables
  - il existe aussi des tailles relatives prédéfinies :
    - `xx-small`
    - `x-small`
    - `small`
    - `medium,`
    - `large,`
    - `x-large,`
    - `xx-large,`
    - `larger,`
    - `smaller`

# Quelques propriétés sur les fontes

- `font-family`
  - permet de spécifier le type de police que l'on souhaite afficher
  - ! Toutes les polices ne sont pas disponibles sur tous les ordinateurs !
  - Il est possible de spécifier plusieurs polices, dans l'ordre où l'on souhaite les utiliser

# Quelques propriétés sur les fontes

- `font-family`
  - familles de polices génériques :
    - `serif`,
    - `sans-serif`,
    - `monospace`,
    - `cursive`,
    - `fantasy`.

*exemple :*

```
font-family : "times new roman", serif, sans-serif
```

# Quelques propriétés sur les fontes

- `font-weight`
  - précise le niveau de gras de la police :
    - `normal`,
    - `bold`,
    - `100`, `200`, ...
- `font-style`
  - précise le style de fonte :
    - `normal`
    - `italic`
    - `oblic`

*exemple : afficher du texte en italique et en gras*

```
font-weight : bold ;  
font-style  : italic ;
```



# Quelques propriétés sur les fontes

- `font-variant`
  - utilise une fonte normale ou en petite capitale :
    - `normal`,
    - `small-caps`.
- `font`
  - raccourci permettant de tout spécifier

*exemple : utilisation du raccourci* `font`

```
font : italic bold 1em cursive ;
```

# Propriétés du texte

- word-spacing : normal | <length>
- letter-spacing : normal | <length>
- text-decoration : none | [ underline || overline ||  
line-through || blink ]
- vertical-align : baseline | sub | super | top | text-top |  
middle | bottom | text-bottom | <percentage>
- text-transform : capitalize | uppercase | lowercase | none
- text-align : left | right | center | justify
- text-indent : <length> | <percentage>
- line-height : <number> | <length> | <percentage>

# Quelques propriétés sur les textes

- `text-align`

- permet de gérer l'alignement du texte

- `right,`
    - `left,`
    - `center,`
    - `justify.`

- `text-decoration`

- comment doit apparaître le texte

- `none` (aucun)
    - `underline` (souligné)
    - `overline` (surligné)
    - `line-through` (barré)
    - `blink` (clignotant)

# Quelques propriétés sur les textes

- `text-transformation`
  - on peut appliquer des transformations au texte
    - `capitalize` (1ere lettre de chaque mot en majuscule)
    - `uppercase` (transformer en majuscule)
    - `lowercase` (transformer en minuscule)
    - `none` (aucune)
- `text-indent`
  - Indentation de début de paragraphe
  - Définit l'indentation du texte, c'est à dire le retrait de la première ligne des paragraphes.
  - Une valeur d'indentation positive provoque le retrait habituel. Au contraire, une valeur d'indentation négative "tire" la première ligne dans la marge.

*exemple :*

```
P {  
  text-transformation : lowercase ;  
  text-indent          : 1cm;  
}
```

# Ah oui, au fait ...

## Valeurs pour Text-Decoration

<b>None</b>	Pas de décoration (valeur par défaut).
Underline	<u>Soulignement.</u>
Overline	Surlignement (trait tracé au dessus du texte).
Line-Through	Caractères barrés.
Blink	Clignotement (ne fonctionne pas avec Internet Explorer).
Inherit	Même valeur que celle de l'élément parent.

*peut s'écrire ...*

```
h1,h2,h3,h4,h5,h6 {text-decoration : underline }
```

# Changer le style

*Comme dit précédemment ...*

```
p {text-decoration : underline }  
strong {text-decoration : underline }
```

*peut s'écrire ...*

```
p,strong {text-decoration : underline }
```

*Mais attention ...*

```
p strong {text-decoration : underline }
```

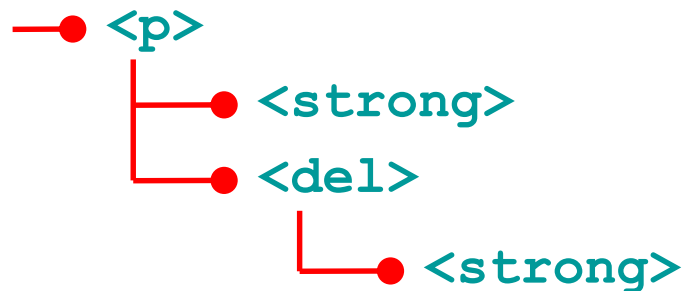
*possède un autre sens :*

*Le style n'est appliqué à strong que s'il est imbriqué dans une balise p*

*Exemple ...*

```
P strong {color : red}
```

```
<p>La couleur<strong>rouge</strong>  
est appliquée ici,  
mais aussi <del><strong>la</strong></del></p>  
<strong>Mais pas la</strong>
```



*De même ...*

```
p+strong {text-decoration : underline }
```

*Le style n'est appliqué à `strong` que s'il suit immédiatement une balise `p`*

*Exemple ...*

```
p {text-indent : 1em}  
p+p {text-indent : 0}
```

*Le premier paragraphe d'un texte sera indenté,  
mais pas les suivants*



Des classes et des ID

# Les classes

(Où certains éléments veulent se la jouer différent...)

- Et si certains types d'éléments devaient être affichés différemment ?
- *exemple* : Le chapeau d'un article

*Ceci est le premier paragraphe de l'article, qui nous sert de chapeau. C'est un paragraphe, on ne peut lui ôter ça, mais on aimerait bien pouvoir l'afficher différemment, histoire que ça soit plus drôle.*

Et puis ceci est un paragraphe tout ce qu'il y a de plus banal, rien de bien exaltant

Et celui-ci aussi est un autre paragraphe plutôt ennuyeux

# Et bien, c'est possible !

(Et c'est la classe !)

- il existe un attribut class
- Applicable à tous les éléments
- permettant de spécifier un style particulier

`<p class="chapeau">`Ceci est le premier paragraphe de l'article, qui nous sert de chapeau. C'est un paragraphe, on ne peut lui ôter ça, mais on aimerait bien pouvoir l'afficher différemment, histoire que ça soit plus drôle .`</p>`

`<p>` Et puis ceci est un paragraphe tout ce qu'il y a de plus banal, rien de bien exaltant`</p>`

`<p>` Et celui-ci aussi est un autre paragraphe plutôt ennuyeux`</p>`

# et du côté du style...

```
P {  
    color : black;  
    background-color : white;  
}  
  
P.chapeau {  
    font-size    : smaller ;  
    text-align   : justify;  
}  
  
P.chapeau:first-letter {  
    font-size    : 1.2em ;  
    font-weight  : bold ;  
}  
  
.petit { font-size : 0.7em }
```

# et du côté du style...

```
P {  
  color : black;  
  background-color : white;  
}
```

```
P.chapeau {  
  font-size : smaller ;  
  text-align : justify;  
}
```

```
P.chapeau:first-letter {  
  font-size : 1.2em ;  
  font-weight : bold ;  
}
```

```
.petit { font-size : 0.7em }
```

Encore une fois, le style cascade de la classe la plus générale vers une classe particulière

Il est possible de cumuler classe et pseudo-classe

on peut définir une classe sans préciser à quel élément elle s'applique

# Les identifiants

(Appelez moi par mon nom !)

- On peut définir un identifiant unique pour un élément
- On peut associer un style à cet identifiant

`<p id="chapeau1">`Ceci est le premier paragraphe de l'article, qui nous sert de chapeau. C'est un paragraphe, on ne peut lui ôter ça, mais on aimerait bien pouvoir l'afficher différemment, histoire que ça soit plus drôle. `</p>`

`<p>` Et puis ceci est un paragraphe tout ce qu'il y a de plus banal, rien de bien exaltant `</p>`

`<p>` Et celui-ci aussi est un autre paragraphe plutôt ennuyeux `</p>`

# et du côté du style...

```
P {  
    color : black;  
    background-color : white;  
}  
  
P#chapeau1 {  
    font-size    : smaller ;  
    text-align   : justify;  
}
```

# et du coté du style...

```
P {  
  color : black;  
  background-color : white;  
}
```

```
P#chapeau1 {  
  font-size : smaller ;  
  text-align : justify;  
}
```



. pour classe

# pour identifiant



# Mais alors, classe ou identifiant ?

- Un identifiant doit être unique
  - Utilisé pour se référer à un élément particulier
  - informations de position
- Une classe peut servir plusieurs fois
  - on peut y mettre les propriétés de style
- Il est possible de cumuler `class` et `id`

# Pseudo-éléments & Pseudo-classes

# Les Pseudo-éléments

- Un pseudo-élément est une caractéristique qui permet de rajouter du détail de style.
- On reconnaît un pseudo-élément à l'utilisation de « : » dans la définition du style correspondant
- Quelques exemples ?

# Quelques pseudo-éléments

- `:first-letter`
  - ne s'applique que sur la première lettre de l'ensemble
- `:first-line`
  - ne s'applique que sur la première ligne de l'ensemble

On peut préciser (ou pas) à quel élément appliquer le pseudo-élément

```
P:first-letter { color : red; font-size : large ;}
```

```
:first-letter { color : red; font-size : large; }
```

# Les Pseudo-classe

- Une pseudo-classe est très similaire à un pseudo-élément.
- On parle de pseudo-classe quand un objet peut avoir différents états
- Un exemple ?

# Les Pseudo-classe liens

- Un lien peut avoir plusieurs états
  - Visité
  - Actif
  - Pointé par la souris
- Les pseudo-classes permettent de donner des styles différents à ces différentes classes de liens.

- `:link`
  - s'applique à un lien actif (avant visite)
- `:visited`
  - s'applique à un lien visité
- `:active`
  - au moment où le lien est activé
- `:hover`
  - lorsque la souris est sur le lien
- `:focus`
  - lorsque le focus est sur le lien (tabulation par exemple)

# Modèles d'affichage en C.S.S

Mise en boîte...



# Les types de blocs

```
<h3>Titre</h3>  
<P>ceci est le premier paragraphe. <em>Cette phrase  
peut être très importante</em>, mais <strong>celle-ci  
l'est plus encore</strong></P><h3>Autre  
titre</h3><h4>titre (encore)</h4><h4>titre  
(toujours)</h4>
```

## Titre

Ceci est le premier paragraphe

*Cette phrase peut être très importante*

Mais,

**Celle-ci l'est plus encore**

## Autre titre

Titre (encore)

Titre (toujours)

# Deux types d'éléments

- Les éléments « bloc »

- Les éléments blocs s'empilent les uns sur les autres
- Un élément bloc peut contenir d'autres éléments blocs
- Un élément bloc peut contenir des éléments en ligne
- *exemples* : p, h1, h2, h3, h4, h5, h6, div

- Les éléments « en ligne »

- Les éléments en ligne se placent les uns à côté des autres
- Un élément en ligne peut contenir d'autres éléments en ligne
- Un élément en ligne ne peut pas contenir d'éléments bloc.
- *exemples* : em, strong, b, i, span

# Deux types d'éléments

- Les éléments « bloc »

- Les éléments blocs s'empilent les uns sur les autres
- Un élément bloc peut contenir d'autres éléments blocs
- Un élément bloc peut contenir des éléments en ligne
- *exemples* : p, h1, h2, h3, h4, h5, h6, div

- Les éléments « en ligne »

- Les éléments en ligne se placent les uns à côté des autres
- Un élément en ligne peut contenir d'autres éléments en ligne
- Un élément en ligne ne peut pas contenir d'éléments bloc.
- *exemples* : em, strong, b, i, span

# DIV & SPAN

- Containers génériques.
- `<div>` permet de structurer le document
- `<span>` permet de changer la mise en page de quelques mots dans un texte
- `<div>` est très utilisé pour la mise en page

# exemple d'utilisation de DIV

```
<div id="menu" >
```

Ici, on peut mettre tout ce qui concerne les menus de navigation.

```
</div>
```

```
<div id="principal" >
```

Ici, c'est le contenu principal de la page (par exemple)

```
</div>
```

```
<div id="pied_page" >
```

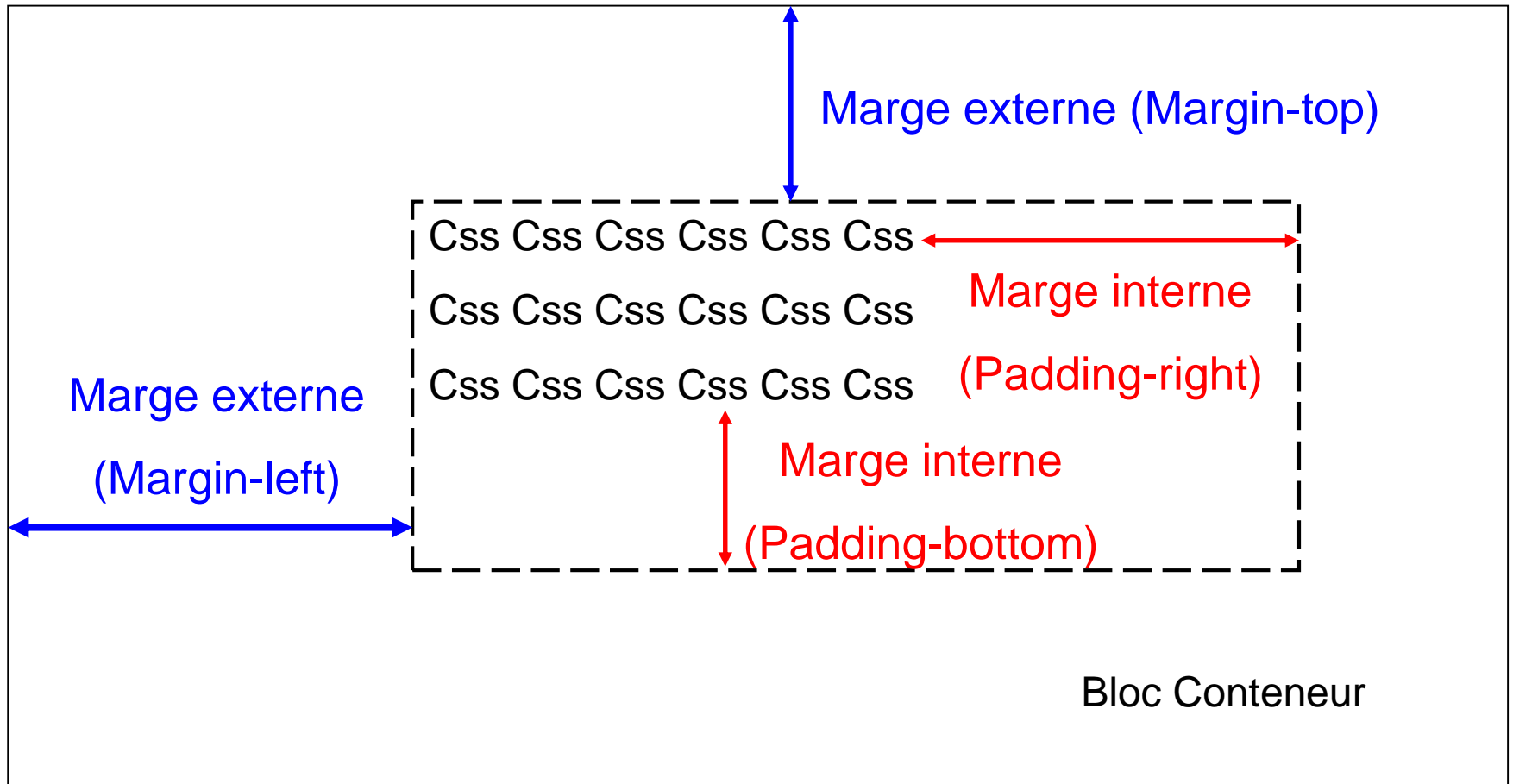
Et là, c'est le pied de page, ou on peut mettre les informations légales...

```
</div>
```

# les propriétés des boites

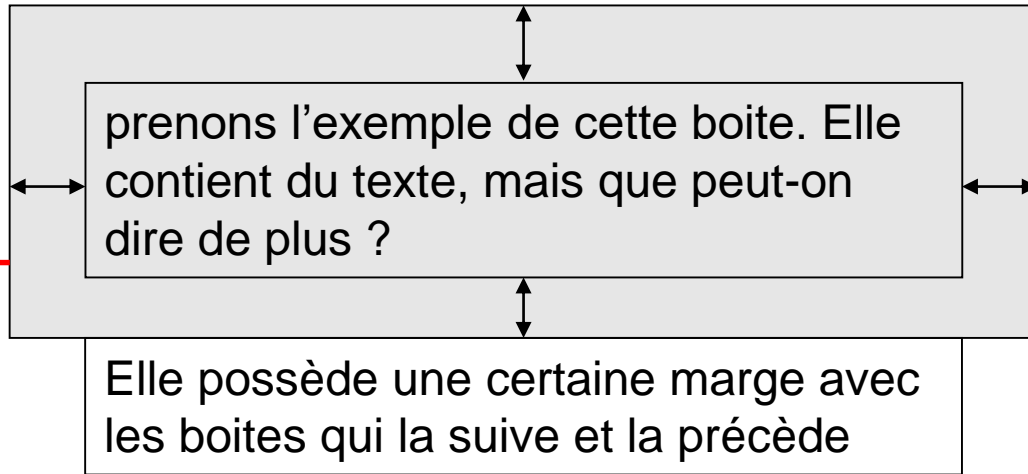
(voyons ce qu'il y a en marge...)

# Rappels



prenons l'exemple de cette boîte. Elle contient du texte, mais que peut-on dire de plus ?





propriété  
margin

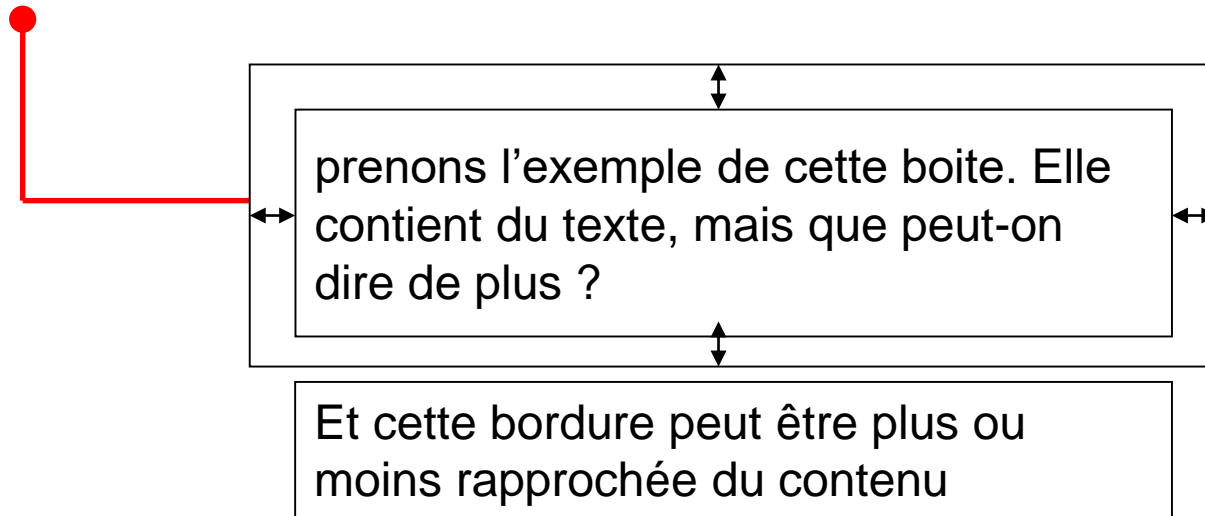
# propriété `border`



prenons l'exemple de cette boîte. Elle contient du texte, mais que peut-on dire de plus ?

Elle possède une bordure blanche, mais qui aurait aussi bien pu être verte, en pointillé ou invisible

propriété  
margin

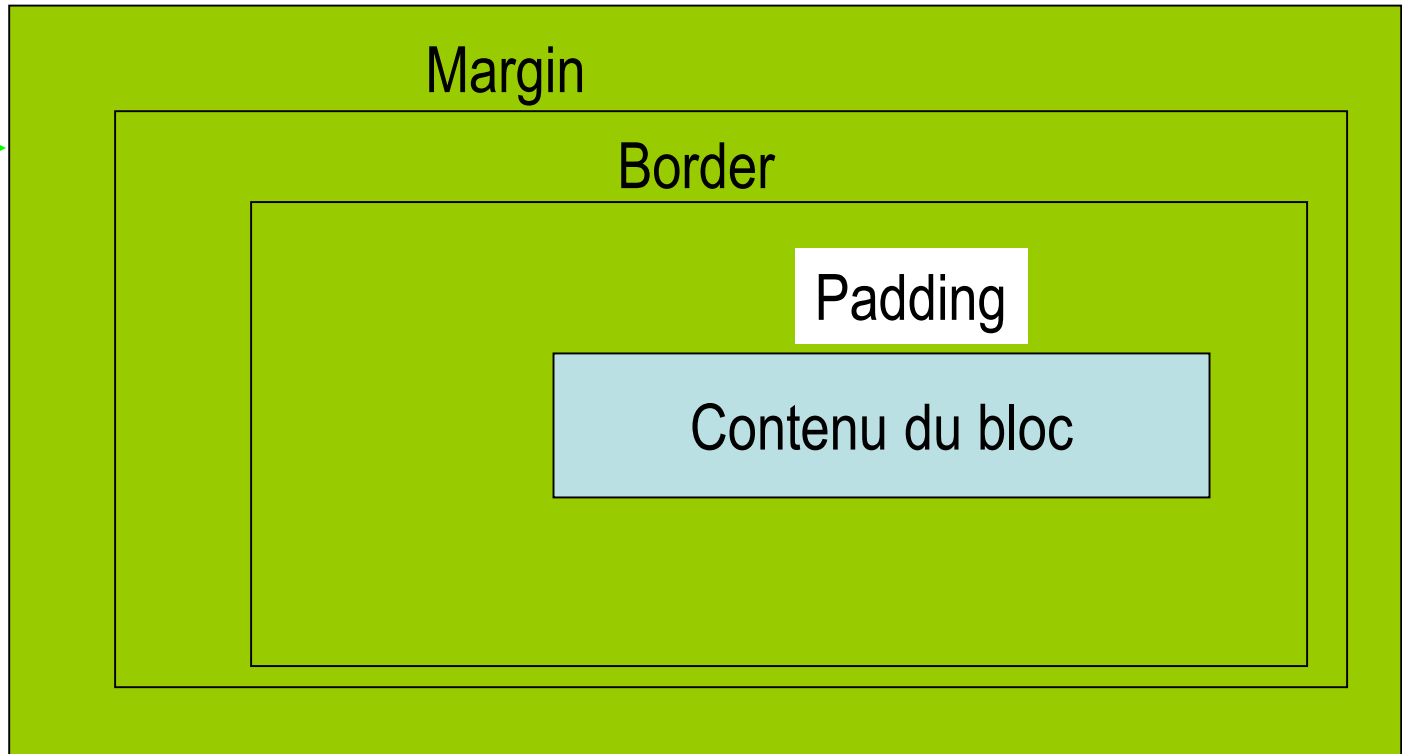


# Les boîtes dans les CSS

- Une page HTML est construite sur un "canevas" (ensemble du texte) qui est plus grand que la fenêtre (partie visible)
- Un élément est considéré comme une boîte rectangulaire placé sur le canevas, pour lequel on peut définir la marge (*margin*), la bordure (*border*) et la distance entre le bord et le contenu (*padding*)
- Chaque partie de la boîte est divisée en quatre (*top*, *left*, *right*, *bottom*)

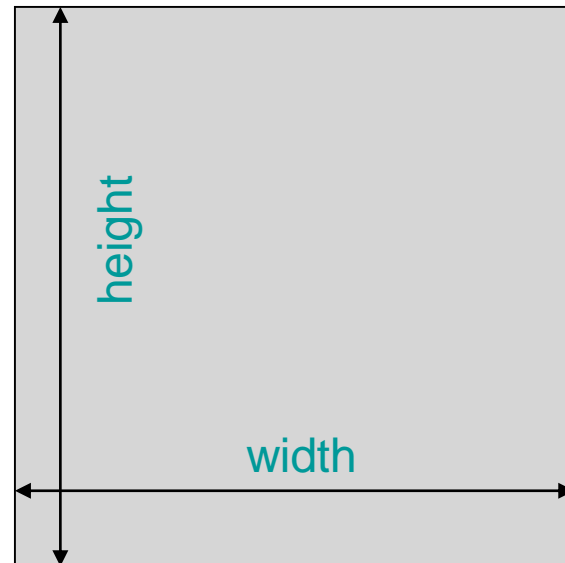
# Propriétés des boîtes

Place  
occupée  
par le  
bloc



# Les propriétés de dimension

- **width**
  - Permet de spécifier la largeur d'une boîte
- **height**
  - Permet de spécifier la hauteur d'une boîte



# propriétés de bordure

- `border-color`
  - gère la couleur de la bordure
- `border-width`
  - **numérique** suivie de px ou % ou pt ou em,
  - **thin**, épaisseur de bordure mince,
  - medium, épaisseur de bordure moyenne (valeur par défaut),
  - **thick**, épaisseur de bordure épaisse,

# propriétés de bordure

- **border-style**

- none, aucune bordure,
- dotted, bordure en pointillé (sauf internet explorer),
- dashed, bordure en tirets (sauf internet explorer),
- solid, bordure continue,
- double, bordure en double trait,
- groove, bordure en creux,
- ridge, bordure en saillie,
- inset, bordure en 3d lumière basse,
- outset, bordure en 3d lumière haute,
- inherit, hérite de son parent (css 2).



# propriétés de bordure (suite)

- on peut aussi régler individuellement les différentes bordures
- `border-top-width`
- `border-bottom-width`
- `border-left-width`
- `border-right-width`
- Comme pour `font`, il existe un raccourci `border`

# taille des marges

- **margin**

- permet de régler la taille de la marge

- **margin-top**
    - **margin-bottom**
    - **margin-left**
    - **margin-right**

- **padding**

- permet de régler la taille du padding

- **padding-top**
    - **padding-bottom**
    - **padding-left**
    - **padding-right**

# Des marges automatiques

- **margin : auto**

- permet normalement de régler les marges au mieux pour centrer l'élément.
- Malheureusement, Internet Explorer n'est pas de cet avis

# Positionnement C.S.S 2

(Qu'est ce qu'on en fait de toutes ces boites ?)

# Le flux

- On oubliera les tableaux !
- On oubliera les Frames !

# Positionnement en flux normal

- Les éléments de type bloc sont affichés par défaut dans une succession verticale

`<p class="jaune">Une boîte jaune</p>`

`<p class="verte">Une boîte verte</p>`

Une boîte jaune

Une boîte verte

- Principaux éléments de type bloc : `div` , `h1` , `h2` , `h3` , `h4` , `h5` , `h6` , `p` , `ul` , `ol` , `li` , `dl` , `dd` , `blockquote` , `pre` , `address`.

# Positionnement en flux normal

- Les éléments de type ligne sont alignés par défaut dans une succession horizontale

<p>

<span class="jaune">Une boîte jaune</span>

<span class="verte">Une boîte verte</span>

</p>

Une boîte jauneUne boîte verte

- Principaux éléments de type ligne : span, a, img, em, strong, q, cite, code, kbd, samp, var, abbr, acronym, ins, del.

# Positionnement en flux normal

- Passer d'un type à l'autre : la propriété `display`
  - Valeurs : *display : inline; ou display : block;*
- Positionnement relatif : permet d'inscrire un contenu dans le flux normal puis de le décaler verticalement ou horizontalement
  - Possibilité de chevauchement
  - Ex : `<p> Lorem <span style="position: relative; bottom: 5px; background-color: #ffff00; "> boîte en position relative </span> ipsum dolor. </p>`

Lorem **boîte en position relative** ipsum dolor

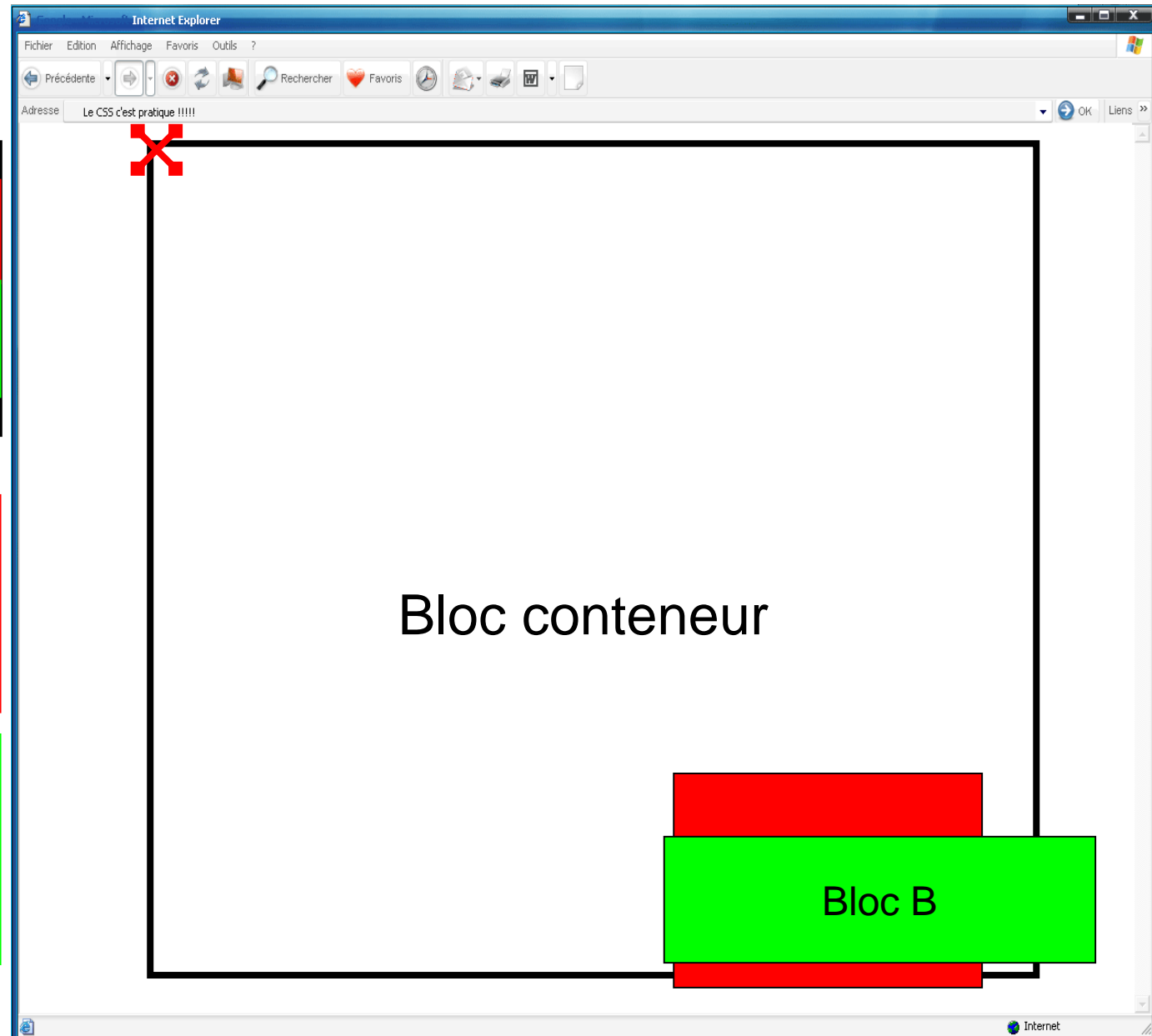


# Le Flux normal

```
<BODY>  
<div class="conteneur">  
  <div class="normalA">  
    Bloc A  
  </div>  
  <div class="normalB">  
    Bloc B  
  </div>  
</div>  
</BODY>
```

```
.normalA {  
width:150px;  
height:150px;  
background-color:red;  
border:1px solid black;  
}
```

```
.normalB {  
width:250px ;  
height:100px ;  
background-color:green;  
border:1px solid black;  
}
```

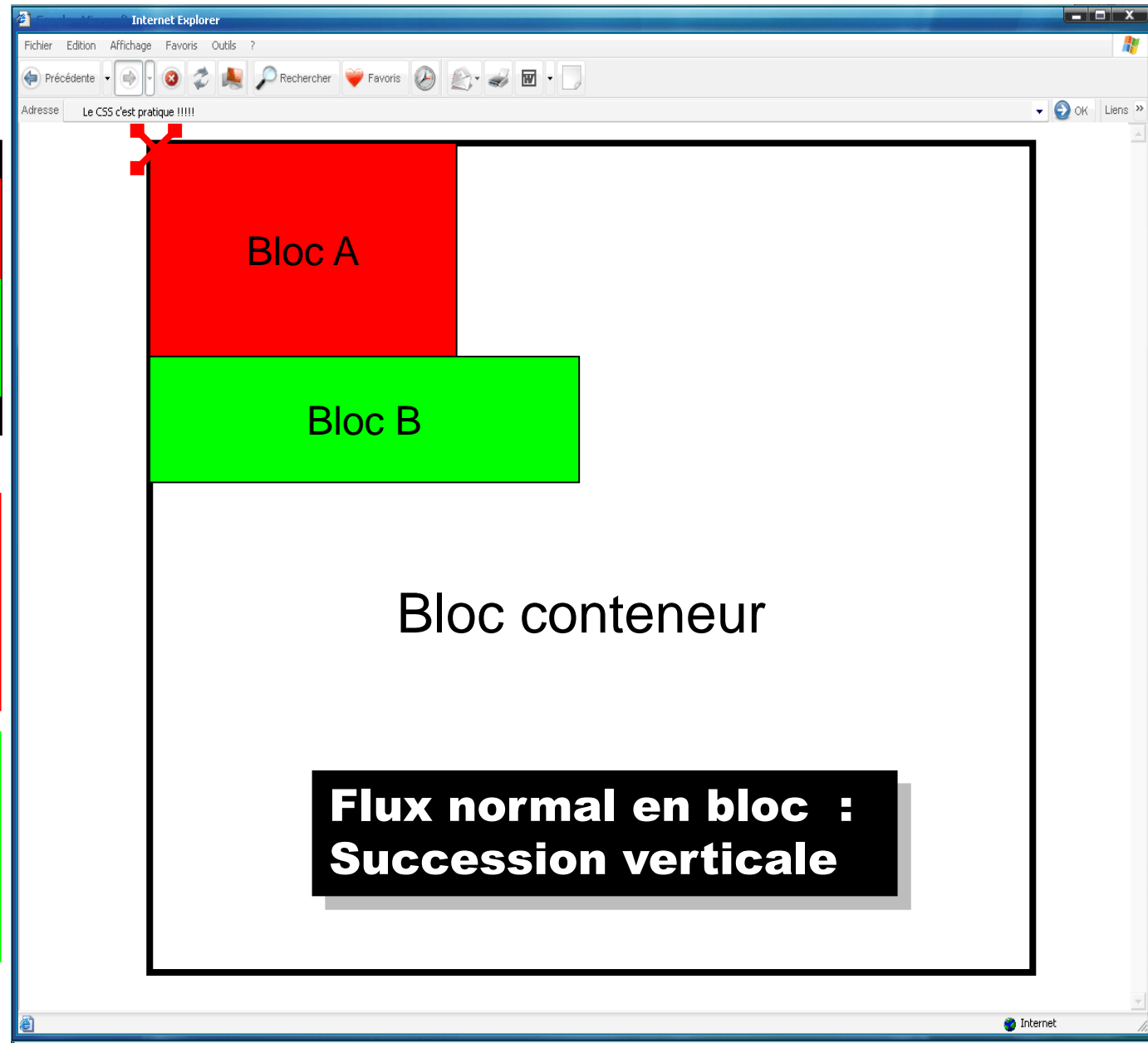


# Le Flux normal : en bloc

```
<BODY>  
<div class="conteneur">  
  <div class="normalA">  
    Bloc A  
  </div>  
  <div class="normalB">  
    Bloc B  
  </div>  
</div>  
</BODY>
```

```
.normalA {  
width:150px;  
height:150px;  
background-color:red;  
border:1px solid black;  
}
```

```
.normalB {  
width:250px ;  
height:100px ;  
background-color:green;  
border:1px solid black;  
}
```



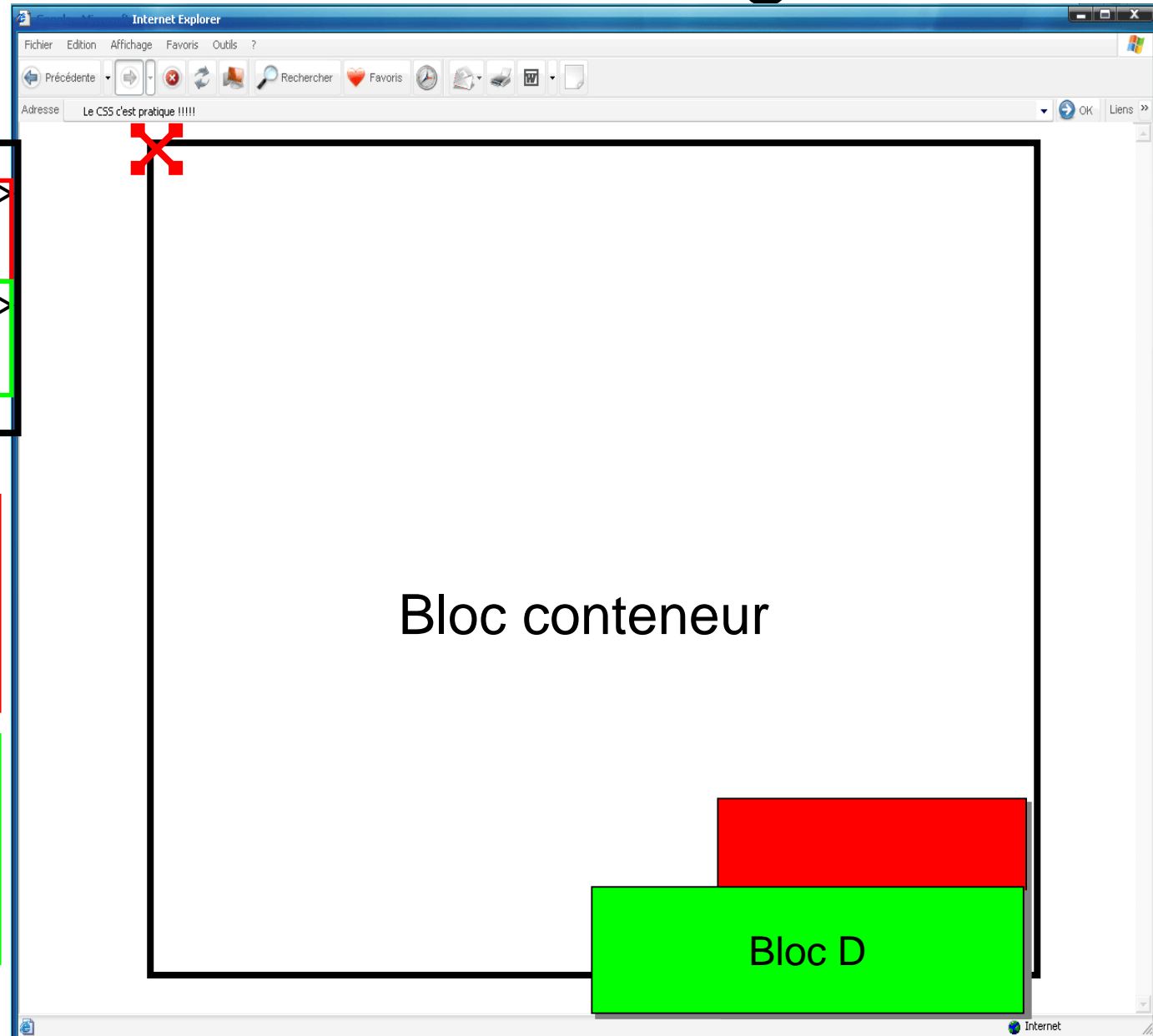
**Flux normal en bloc :  
Succession verticale**

# Le Flux normal : en ligne

```
<BODY>  
<div class="conteneur">  
  <span class="normalC">  
    Bloc C  
  </span>  
  <span class="normalD">  
    Bloc D  
  </span>  
</div>  
</BODY>
```

```
.normalC {  
width:150px;  
height:150px;  
background-color:red;  
border:1px solid black;  
}
```

```
.normalD {  
width:250px ;  
height:100px ;  
background-color:green;  
border:1px solid black;  
}
```

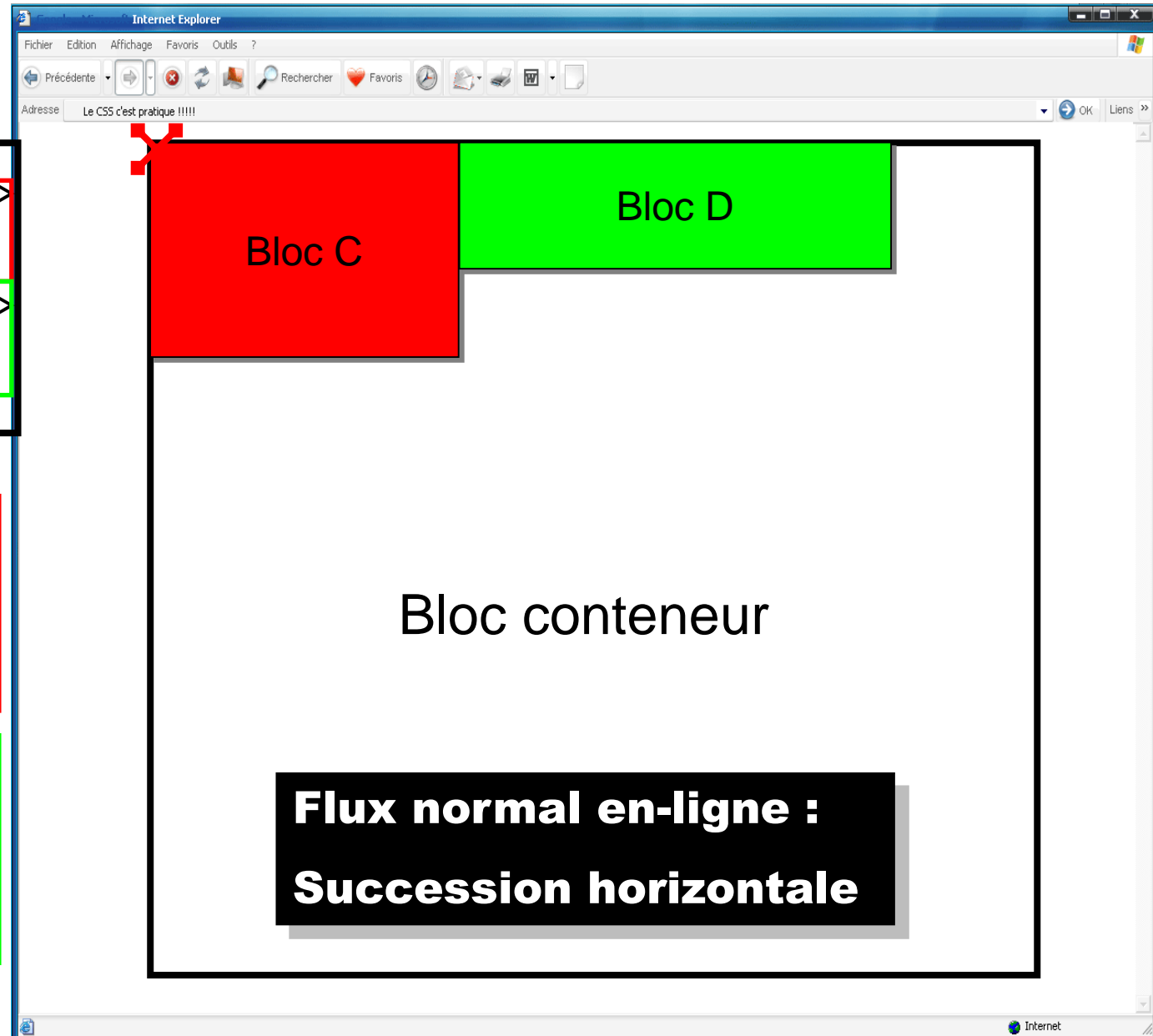


# Le Flux normal : en ligne

```
<BODY>  
<div class="conteneur">  
  <span class="normalC">  
    Bloc C  
  </span>  
  <span class="normalD">  
    Bloc D  
  </span>  
</div>  
</BODY>
```

```
.normalC {  
width:150px;  
height:150px;  
background-color:red;  
border:1px solid black;  
}
```

```
.normalD {  
width:250px ;  
height:100px ;  
background-color:green;  
border:1px solid black;  
}
```



# Les propriétés de placement

- position
  - **absolute** Le bloc est placé de façon absolue
  - **relative** Le bloc est placé relativement à la position qu'il aurait du occuper
  - **fixed** le bloc est épinglé à l'écran (ne fonctionne pas sous Internet Explorer)

# Les propriétés de placement (2)

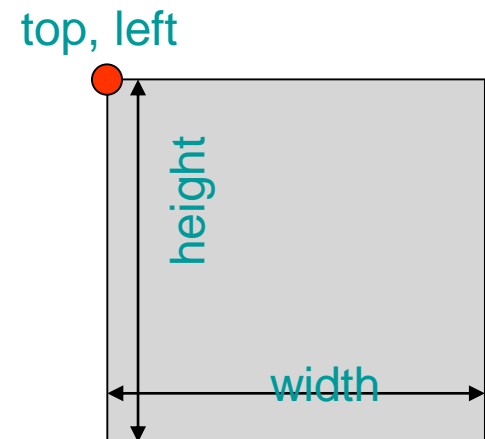
- position
  - **absolute** Le bloc est placé de façon absolue

pour les positions **absolute** et **relative**, il existe deux propriétés

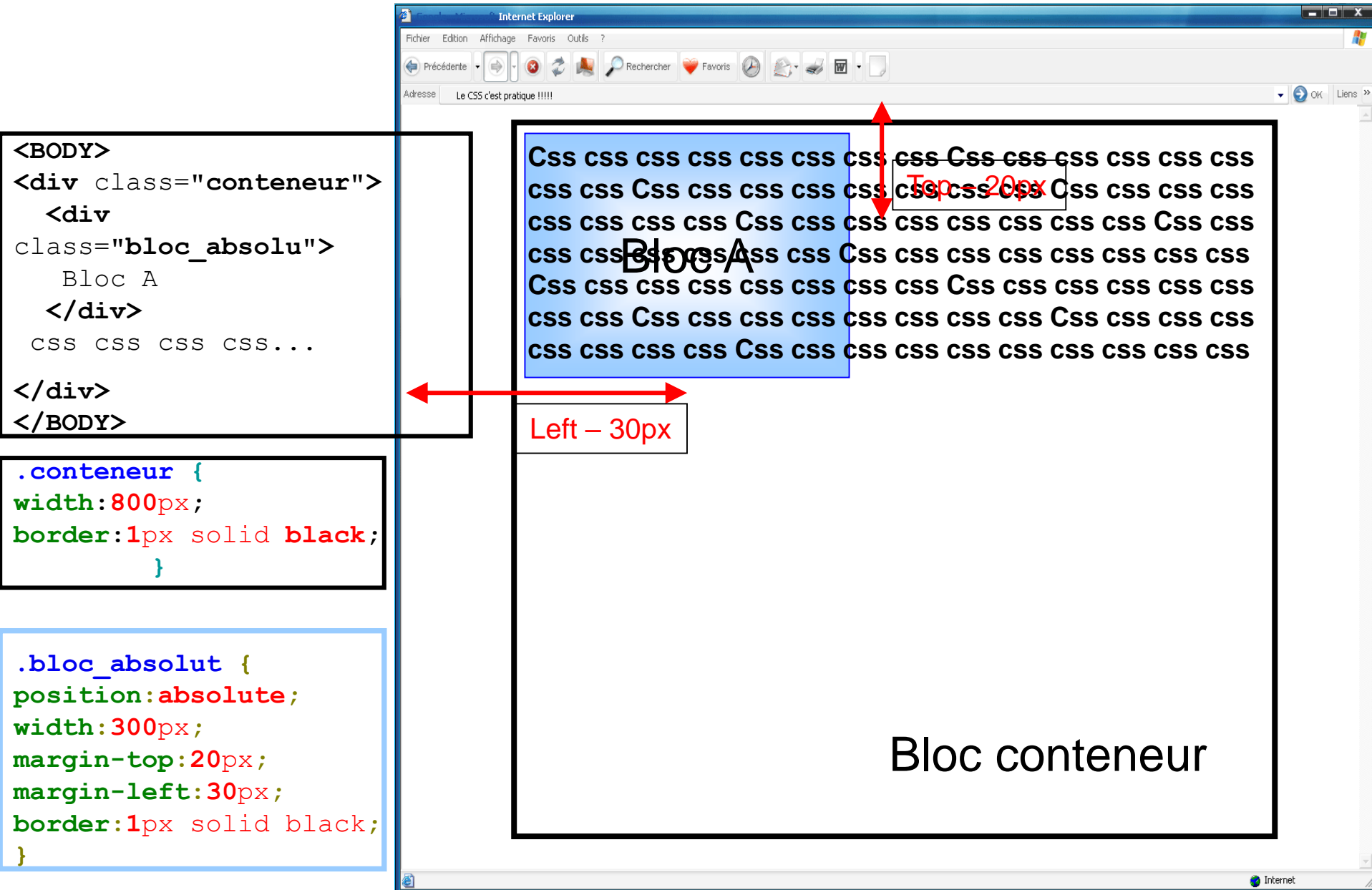
- **top**
- **left**

qui permettent de spécifier où doit se placer le coin haut gauche du bloc

```
div#toto {  
  position : absolute;  
  top      : 50% ;  
  left     : 100px;  
}
```



# Le Flux absolu



```

<BODY>
<div class="conteneur">
  <div
class="bloc_relative">
    Bloc A
  </div>

  CSS CSS CSS CSS CSS...
</div>
</BODY>

```

```

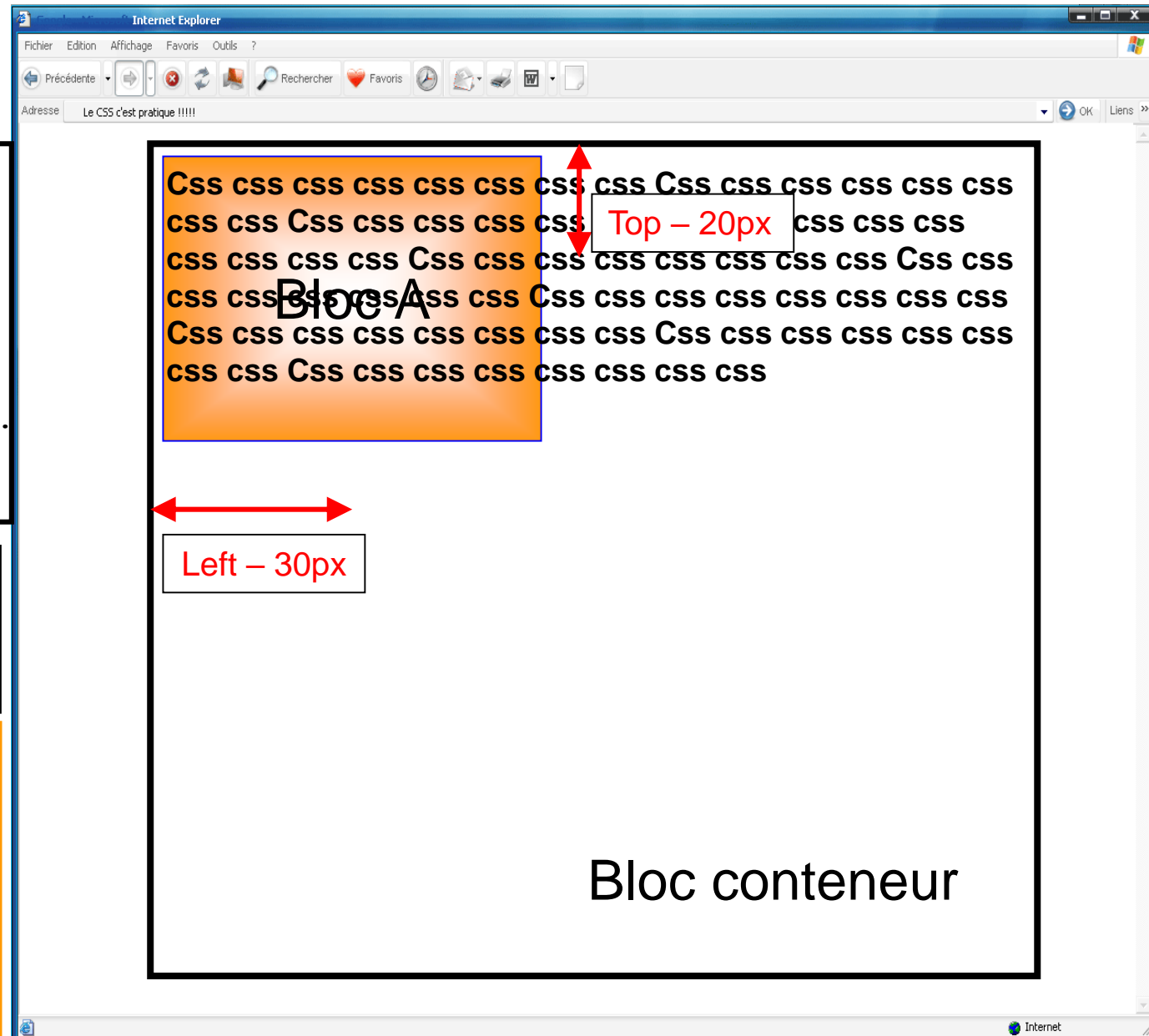
.conteneur {
width:800px;
border:1px solid black;
}

```

```

.bloc_absolu {
position:relative;
width:300px;
margin-top:20px;
margin-left:30px;
border:1px solid black;
}

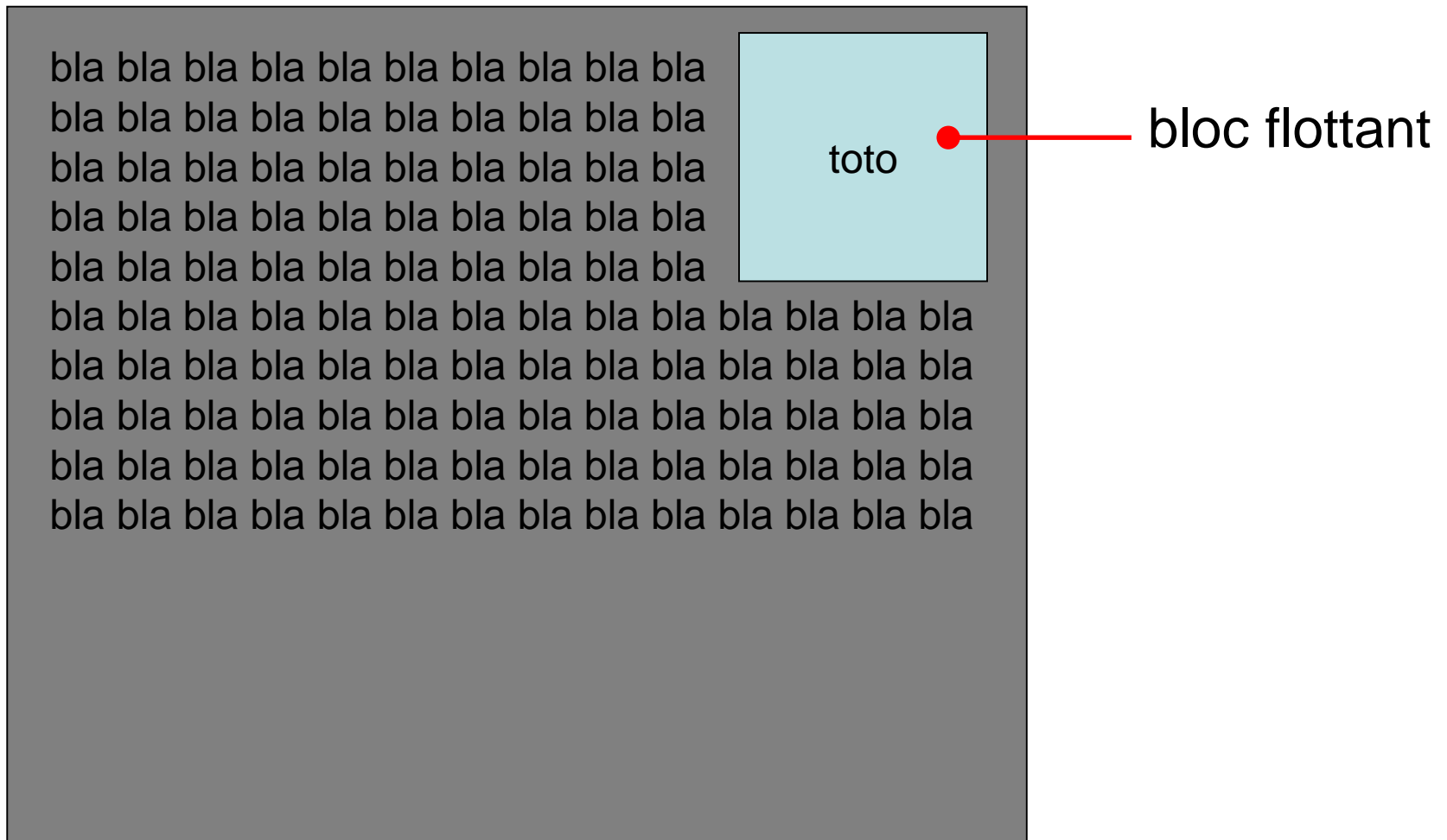
```





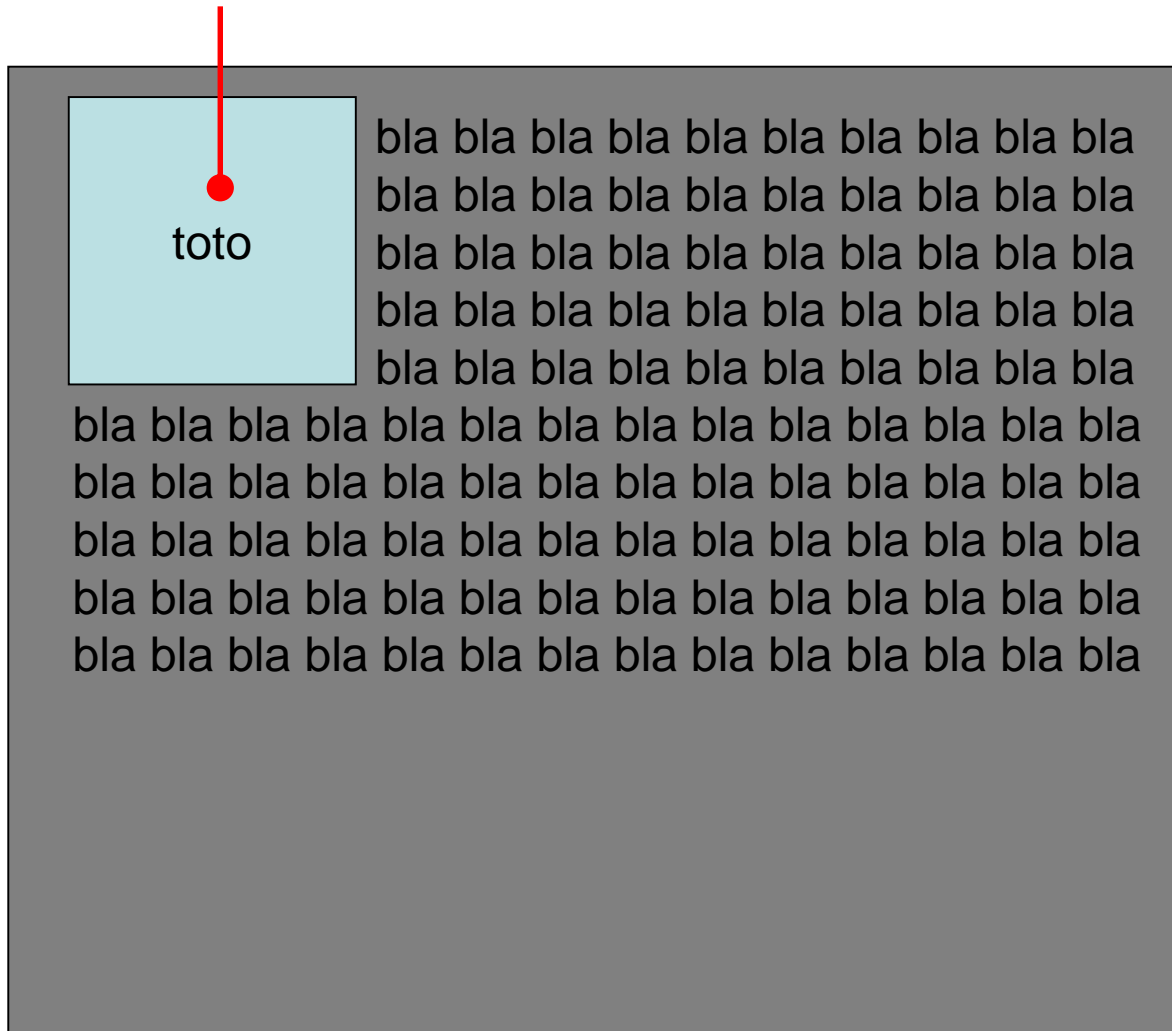
# Le placement flottant

- la propriété `float` peut prendre trois valeurs
  - `none`
  - `left`
  - `right`
- cette propriété indique comment le bloc flotte par rapport aux suivants.



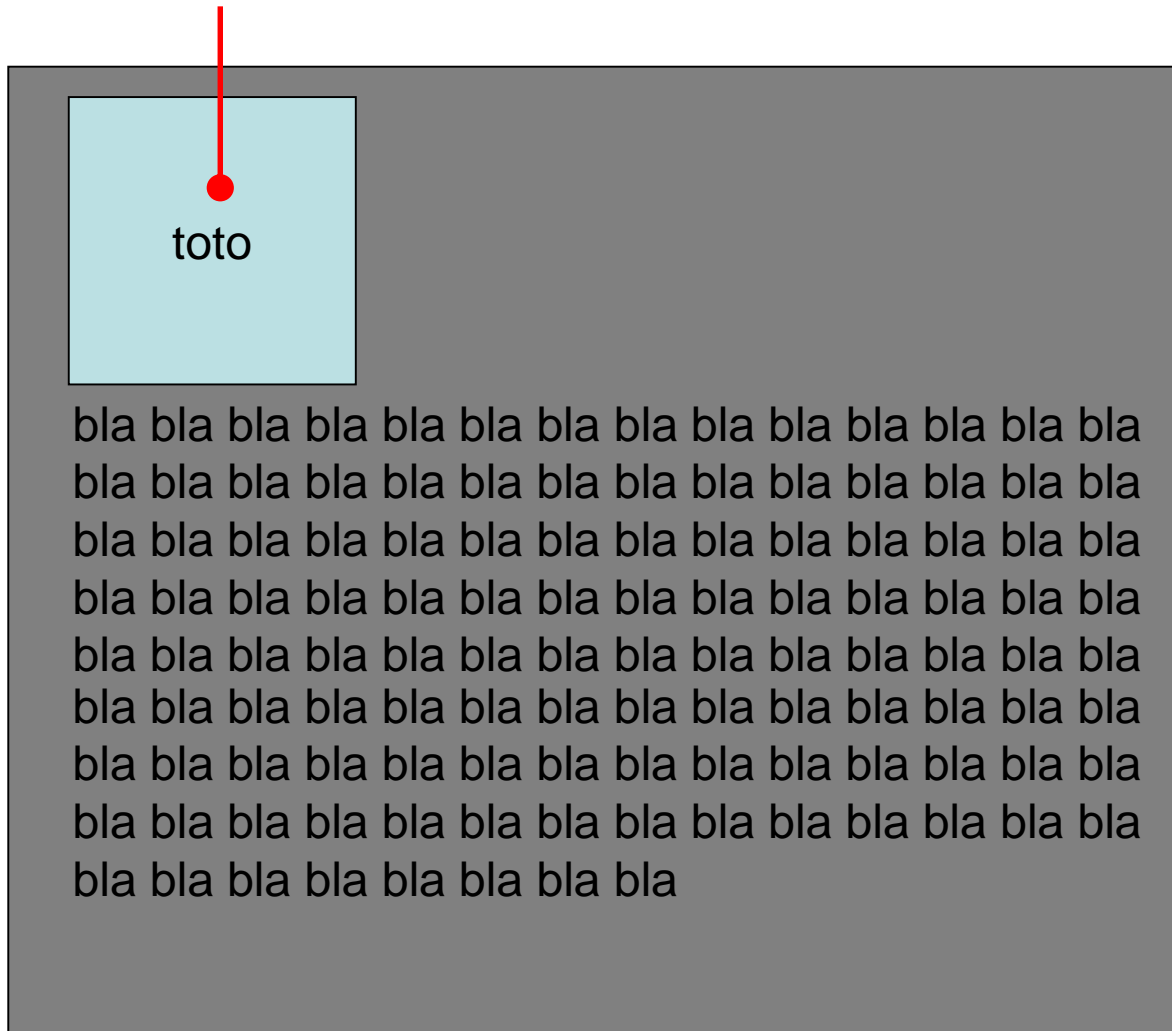
```
div#toto { float : right }
```

# bloc flottant



```
div#toto { float : left }
```

# bloc non flottant

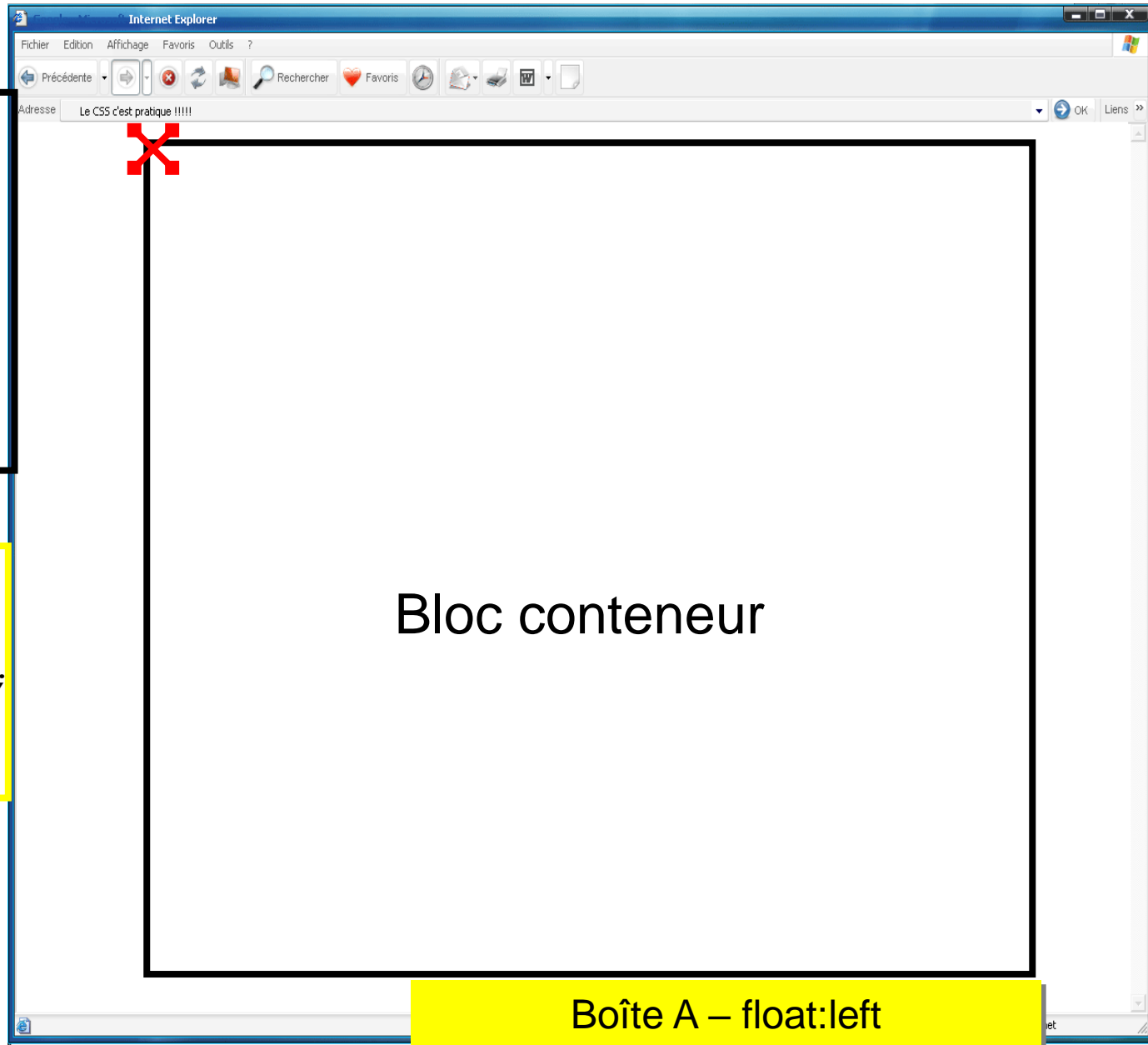


```
div#toto { float : none }
```

# Le Flux flottant

```
<BODY>  
<div class="conteneur">  
  <div class="flotteA">  
    Boîte A  
  </div>  
<p>  
  Texte...blabla ...  
</p>  
</div>  
</BODY>
```

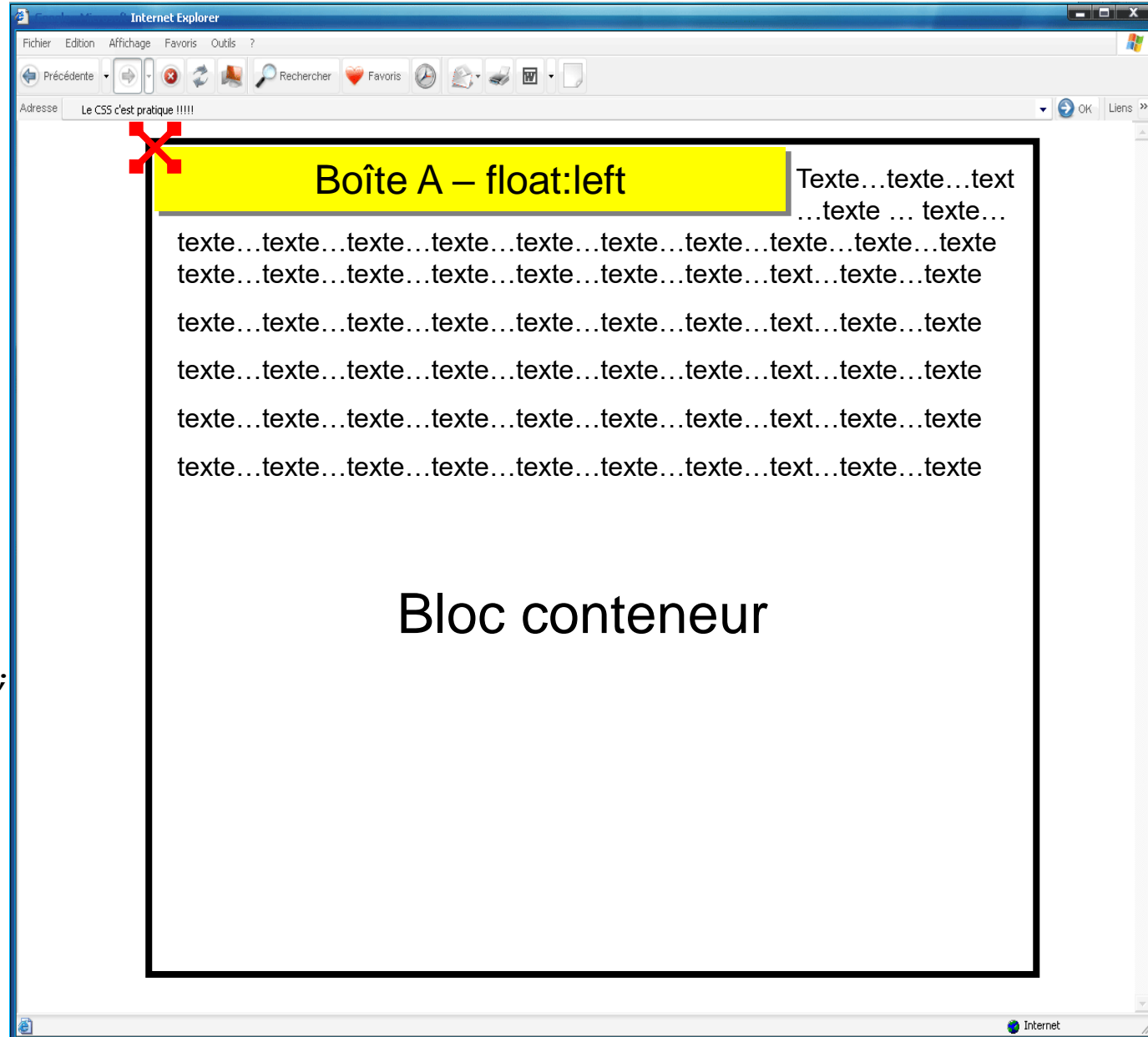
```
.flotteA {  
float:left;  
width:500px;  
background-color:yellow;  
border:1px solid black;  
}
```



# Le Flux flottant

```
<BODY>
<div class="conteneur">
  <div class="flotteA">
    Boîte A
  </div>
  <p>
    Texte...texte ...
  </p>
</div>
</BODY>
```

```
.flotteA {
float:left;
width:650px;
background-color:yellow;
border:1px solid black;
}
```



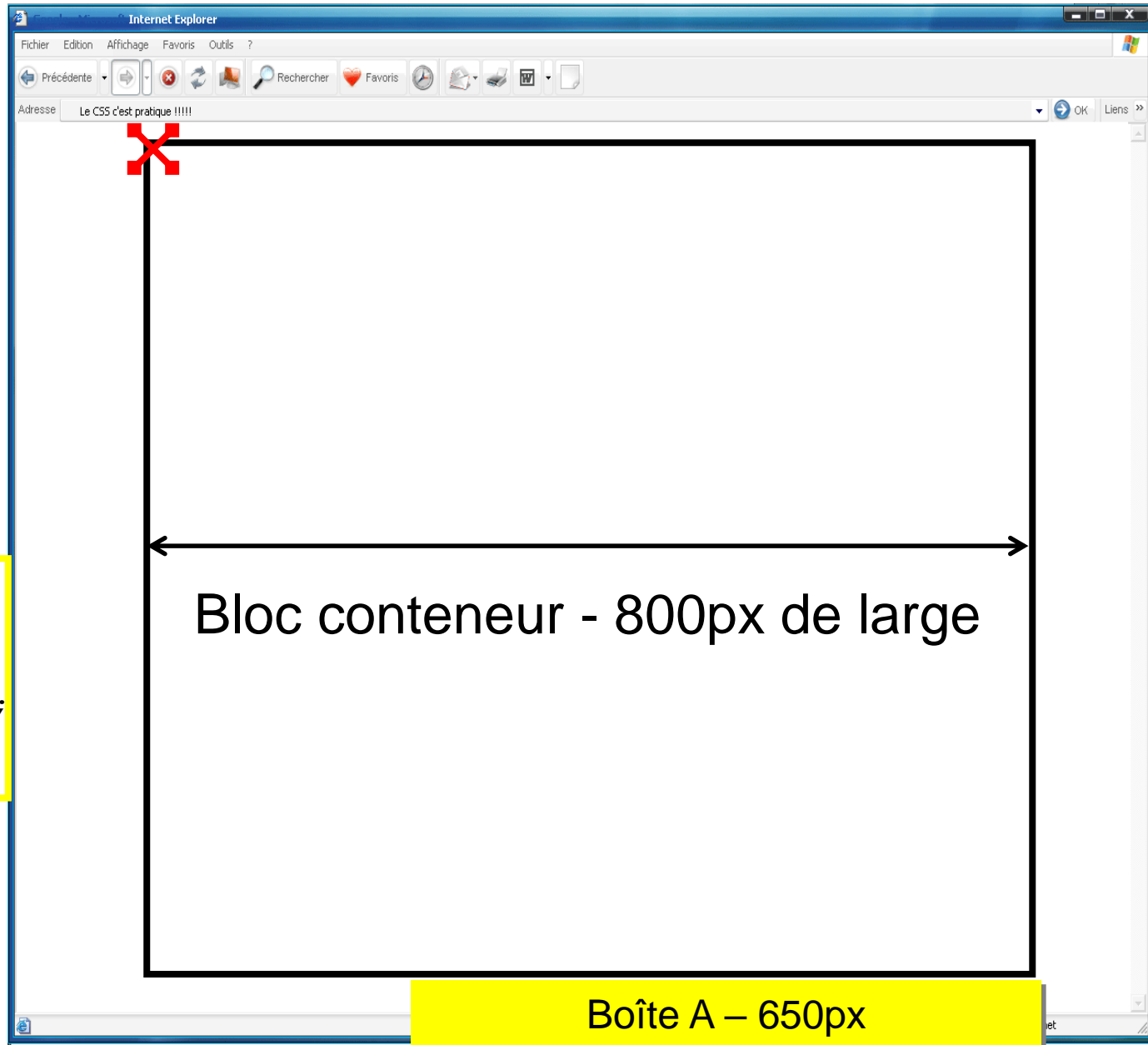
# Le Flux flottant

```
<BODY>
<div class="conteneur">
  <div class="flotteA">
    Boîte A
  </div>
  <div class="flotteB">
    Boîte B
  </div>
</div>
</BODY>
```

```
.conteneur {
width:800px;
border:1px solid black;
}
```

```
.flotteA {
float:left;
width:650px;
background-color:yellow;
border:1px solid black;
}
```

```
.flotteB {
float:left;
width:100px;
background-color:blue;
border:1px solid black;
}
```









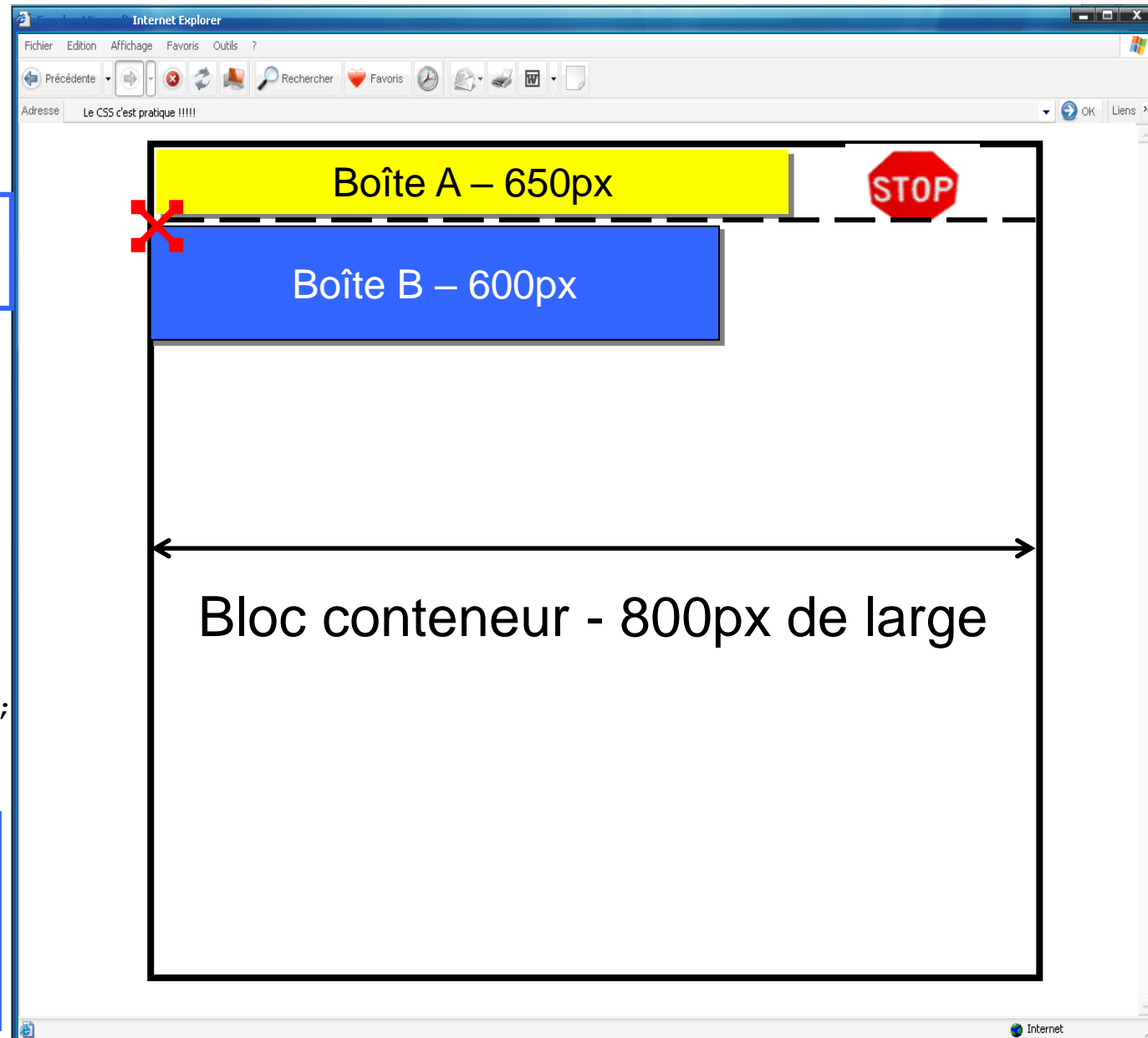
# Le Flux flottant

```
<BODY>
<div class="conteneur">
  <div class="flotteA">
    Boîte A
  </div>
  <div class="flotteB">
    Boîte B
  </div>
</div>
</BODY>
```

```
.conteneur {
width:800px;
border:1px solid black;
}

.flotteA {
float:left;
width:650px;
background-color:yellow;
border:1px solid black;
}
```

```
.flotteB {
float:left;
width:600px;
background-color:blue;
border:1px solid black;
}
```



# Position float : exple

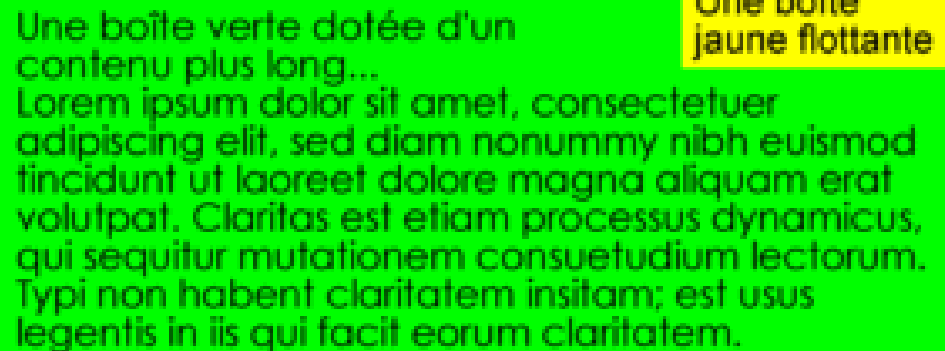
En HTML:

```
<p class="jaune">Une boîte jaune flottant</p>
```

```
<p class="verte">Une boîte verte doté d'un contenu plus long...</p>
```

En CSS:

```
.jaune {  
    background-color: #ffff00;  
    float: right;  
    width: 100px;  
    margin: 0;  
}  
.verte {  
    background-color: #00ff00;  
}
```



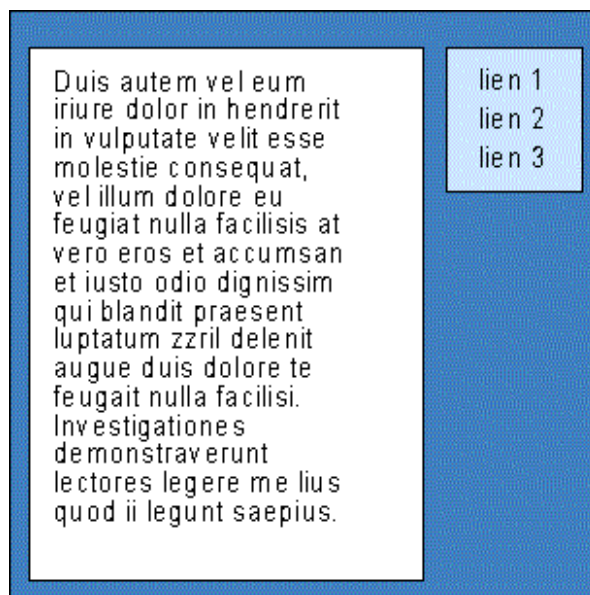
Une boîte verte dotée d'un contenu plus long...  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.

Une boîte jaune flottante

# Position float : exple2

En CSS :

```
.content {  
    float: left;  
    width: 70%;  
}  
  
.menu {  
    margin-left: 80%;  
    border: 1px solid #000000;  
    padding: 1em;  
}
```



HTML :

```
... <body>  
<div class="content"> ...  
</div>  
<div class="menu">  
<ul>  
    <li>lien_1</li>  
    <li>lien_2</li>  
    <li>lien_3</li>  
</ul>  
</div> </body>
```

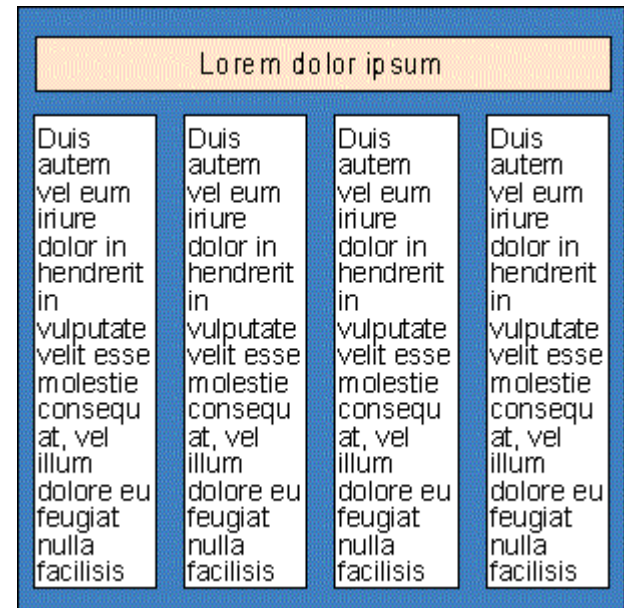
# Position float : exple3

En CSS :

```
body {  
    margin: 0;  
    padding: 0;  
}  
.float {  
    float: left;  
    width: 20%;  
    margin: 1em 0;  
}  
p,h1,h2 {  
    margin: 1em;  
}
```

HTML :

```
... <body> ...  
<div class="float"> ... </div>  
<div class="float"> ... </div>  
<div class="float"> ... </div>  
<div class="float"> ... </div>  
</body>
```



**Notion:**

**margin:** 'margin-top' 'margin-right' 'margin-bottom' 'margin-left';


# Position float : problème du débordement

- Si boîtes de tailles différentes, il peut y avoir des décalages disgracieux
- Solution : ajouter une classe spacer : élément bloc au contenu fictif (l'espace insécable &nbsp;), doté de la propriété clear qui lui interdit d'être adjacent à une boîte flottante.


CSS : `.spacer { clear: both; }`

HTML: `<hr class="spacer« />`

# Création d'un site



```
div#head {  
  background: url("tiles.png");  
  height: 30px;  
  padding: 0.5em 0.5em 2px 120px;  
  text-align: left;  
  font: bold 1.5em Verdana, Arial, Helvetica, sans-serif;  
}
```



# Le CSS 2

Accueil :: Contact :: Liens

Menu Accueil	Presentation
<ul style="list-style-type: none"><li>Introduction</li><li>Syntaxe</li><li>Flux</li><li>Propriétés</li><li>Menu déroulant</li><li>Exemples</li><li>Divers</li></ul>	<h2>Le CSS, c'est quoi ?</h2> <p>Littéralement, css veut dire Cascading Style Sheets (en français Feuille de Style en cascade).</p> <p>Le design d'un site évolue toujours au fil du temps. Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond...</p> <p>Avec les "Cascading Style Sheet" (CSS), ce lourd handicap est résolu, car en utilisant un fichier .css totalement indépendant, il suffira uniquement d'y modifier les tailles de police ou d'y mettre une nouvelle image de fond pour que celle-ci soit automatiquement changée sur toutes les pages où l'on fait appel à la feuille de style .</p> <p>Aujourd'hui ce formidable outil est indispensable à l'heure où sur internet tout change vite...</p> <p>A partir du menu de gauche (c'est un menu simple sans déroulement), nous allons faire évoluer ce menu, tout en étant confronté aux problèmes de compatibilité des navigateurs.</p>

# Exemple complet

## Bloc Header – flux normal

### Menu Accueil

- ::: Introduction
- ::: Syntaxe
- ::: Flux
- ::: Propriétés
- ::: Menu déroulant
- ::: Exemples
- ::: Divers

### Presentation

#### Le CSS, c'est quoi ?

Littéralement, css veut dire Cascading Style Sheets (en français Feuille de Style en cascade).

**Bloc conteneur**  
Le design d'une page évolue toujours et tout le temps. Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond...

Avec les "Cascading Style Sheet" (CSS), ce lourd handicap est résolu, car en utilisant un fichier .css totalement indépendant, il suffira uniquement d'y modifier les tailles de police ou d'y mettre une nouvelle image de fond pour que celle-ci soit automatiquement changée sur toutes les pages où l'on fait appel à la feuille de style .

Aujourd'hui ce formidable outil est indispensable à l'heure où sur internet tout change vite...

A partir du menu de gauche (c'est un menu simple sans déroulement), nous allons faire évoluer ce menu, tout en étant confronté aux problèmes de compatibilité des navigateurs.



# Exemple complet

Header\_gauche - float left

Header\_droit – float left

## Menu Accueil

- ::: Introduction
- ::: Syntaxe
- ::: Flux
- ::: Propriétés
- ::: Menu déroulant
- ::: Exemples
- ::: Divers

## Presentation

### Le CSS, c'est quoi ?

Littéralement, css veut dire Cascading Style Sheets (en français Feuille de Style en cascade).

Le design d'un site évolue toujours au fil du temps. Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond...

Avec les "Cascading Style Sheet" (CSS), ce lourd handicap est résolu, car en utilisant un fichier .css totalement indépendant, il suffira uniquement d'y modifier les tailles de police ou d'y mettre une nouvelle image de fond pour que celle-ci soit automatiquement changée sur toutes les pages où l'on fait appel à la feuille de style .

Aujourd'hui ce formidable outil est indispensable à l'heure où sur internet tout change vite...

A partir du menu de gauche (c'est un menu simple sans déroulement), nous allons faire évoluer ce menu, tout en étant confronté aux problèmes de compatibilité des navigateurs.

# Exemple complet

Header\_gauche - float left

Header\_droit – float left

Bloc Menu Header – flux normal

## Menu Accueil

- ::: Introduction
- ::: Syntaxe
- ::: Flux
- ::: Propriétés
- ::: Menu déroulant
- ::: Exemples
- ::: Divers

## Presentation

### Le CSS, c'est quoi ?

Littéralement, css veut dire Cascading Style Sheets (en français Feuille de Style en cascade).

Le design d'un site évolue toujours au fil du temps. Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond...

Avec les "Cascading Style Sheet" (CSS), ce lourd handicap est résolu, car en utilisant un fichier .css totalement indépendant, il suffira uniquement d'y modifier les tailles de police ou d'y mettre une nouvelle image de fond pour que celle-ci soit automatiquement changée sur toutes les pages où l'on fait appel à la feuille de style .

Aujourd'hui ce formidable outil est indispensable à l'heure où sur internet tout change vite...

A partir du menu de gauche (c'est un menu simple sans déroulement), nous allons faire évoluer ce menu, tout en étant confronté aux problèmes de compatibilité des navigateurs.

# Exemple complet

Header\_gauche - float left

Header\_droit – float left

Bloc Menu Header – flux normal

## Presentation

### Le CSS, c'est quoi ?

Littéralement, css veut dire Cascading Style Sheets (en français Feuille de Style en cascade).

Le design d'un site évolue toujours au fil du temps. Le problème, lorsqu'on n'utilise pas de feuilles de style, c'est qu'il faut reprendre toutes les pages une à une pour modifier une police de caractère ou une couleur de fond...

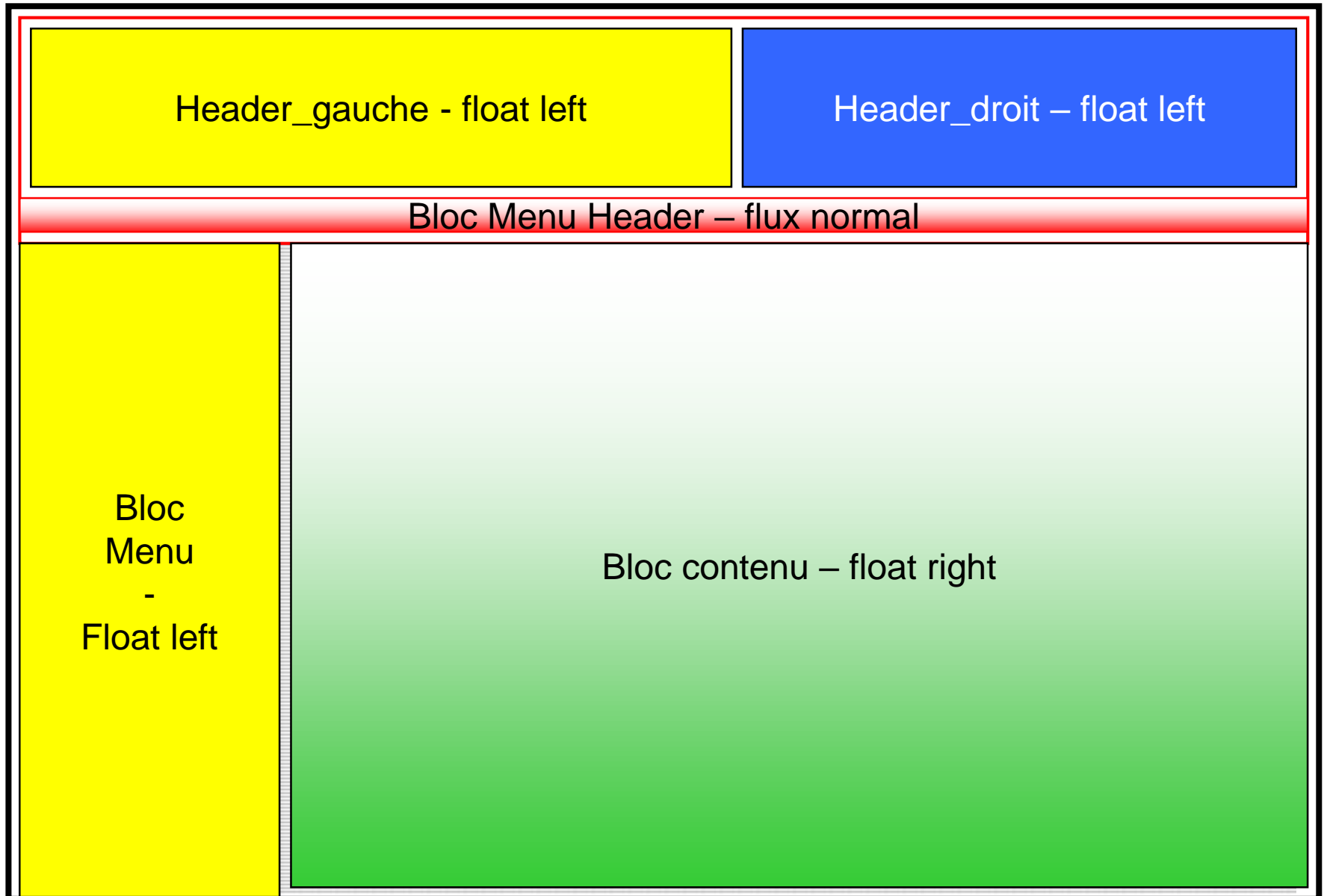
Avec les "Cascading Style Sheet" (CSS), ce lourd handicap est résolu, car en utilisant un fichier .css totalement indépendant, il suffira uniquement d'y modifier les tailles de police ou d'y mettre une nouvelle image de fond pour que celle-ci soit automatiquement changée sur toutes les pages où l'on fait appel à la feuille de style .

Aujourd'hui ce formidable outil est indispensable à l'heure où sur internet tout change vite...

A partir du menu de gauche (c'est un menu simple sans déroulement), nous allons faire évoluer ce menu, tout en étant confronté aux problèmes de compatibilité des navigateurs.

Bloc  
Menu  
-  
Float left

# Exemple complet



# Visualisation C.S.S

(Un autre point de vue ?)

# La propriété `visibility`

- `visibility` peut prendre deux valeurs :
  - `visible`
  - `hidden`
- un élément `hidden` devient invisible, mais la place qu'il occupe à l'écran est tout de même réservée.

# propriété visibility : **visible**

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 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 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

# propriété visibility : **hidden**





# La propriété `display`

- `Display` indique le comportement que doit adopter l'élément
  - `none`
  - `inline`
  - `block`
  - `list-item`
  - ...
- `none` indique qu'il ne faut pas réserver de la place pour cet élément

**XHTML**

# Le XHTML

- Le XHTML 1.0 (eXtensible HTML) est une recommandation du W3C
- Version de HTML respectant la syntaxe de XML afin de permettre une structure plus stricte et garantir la qualité des documents
- Il intègre toutefois les caractéristiques les plus intéressantes du XML : données structurées, extension des fonctions. Il est codé comme du HTML sauf le prologue, qui doit obligatoirement être du type :
- **`<!doctype html public " "-//w3c//dtd XHTML 1.0 //EN">`**

# Écrire en XHTML (1)

1. Le document doit commencer par la déclaration suivante :

```
<?xml version="1.0"?>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
    Transitional//EN" "DTD/xhtml11-  
    transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">
```

2. Les balises `<head>` et `<title>` sont obligatoires
3. Les balises et leurs attributs doivent être en minuscules
4. Un attribut doit être entre guillemets
5. Tous les attributs doivent avoir une valeur

```
<div align="center">du Xhtml </div>
```

# Écrire en XHTML (2)

6. Toutes les balises ouvertes doivent être refermées.
7. Les balises doivent être correctement imbriquées :  
`<strong><em>texte</em></strong>`
8. Les éléments en ligne (pas de bloc) ne peuvent pas contenir de bloc
9. L'attribut `alt` est obligatoire pour la balise `img` et `type` pour `script`

# Écrire en XHTML (3)

## 10. Les commentaires ne sont plus spécifiés de la même façon :

```
<script> <!-- contenu -->
</script>
```

```
<script> <![CDATA contenu ]]>
</script>
```

Pour valider le code XHTML, on utilisera <http://validator.w3.org>

**DHTML**

# **Le HTML Dynamique (DHTML) : introduction**

**Le HTML dynamique permet de réaliser des  
présentation animées et interactives**

- ✱ Un bloc peut se déplacer sur la page  
(animation ou interaction avec l'utilisateur)
- ✱ Le style des blocs peut être modifié
- ✱ La capture d'un événement souris permet de déclencher des actions
- ✱ Des zones de l'écran peuvent être réservées pour le défilement de textes
- ✱ Des menus déroulants peuvent être créés



# Intérêt du DHTML

- Meilleure maîtrise de l'interaction lecteur document
- La personnalisation des pages peut se faire par
  - Script CGI : pour des applications complètes
  - Applets Java : interaction au sein de l'applet
  - Plug-ins : Flash, acrobat reader etc ...
  - Par le rafraichissement de pages
  - Par des programmes au sein même des pages HTML qui sont interprétés par le serveur : Php pour Apache par exemple.

# DHTML : la coopération

- HTML 4.0 : associe aux éléments des identifiants ou des classes et définit la capture d'événements
- Les feuilles de style : langage normalisé de définition de positionnement des éléments sur le canevas
- DOM (Document Object Model) : Le DOM interprète chaque élément qui constitue votre page Web. Ces éléments sont considérés comme des objets auxquels les scripts peuvent accéder. Ce procédé a été mis au point par le W3C, afin d'établir une norme pour assurer une compatibilité (lors de la lecture d'une page HTML Dynamique) parfaite entre les deux principaux navigateurs du marché (I.Explorer et Netscape).

# Le Document Object Model

- Quand une page est chargée par un navigateur, il y a construction de graphe des tous les objets qui sont affichés sur le canevas. Ce graphe sert à désigner les éléments pour que les scripts puissent y avoir accès.
- Le DOM dépend du navigateur utilisé
  - Dans IE, chaque élément peut devenir scriptable (modification du style, de la position du contenu)
  - Dans Netscape, seuls les DIV et les images peuvent être modifiées ou déplacées
- Conséquence : le code Javascript est différent pour obtenir le même résultat
- Nouvelles normes : Unification du DOM

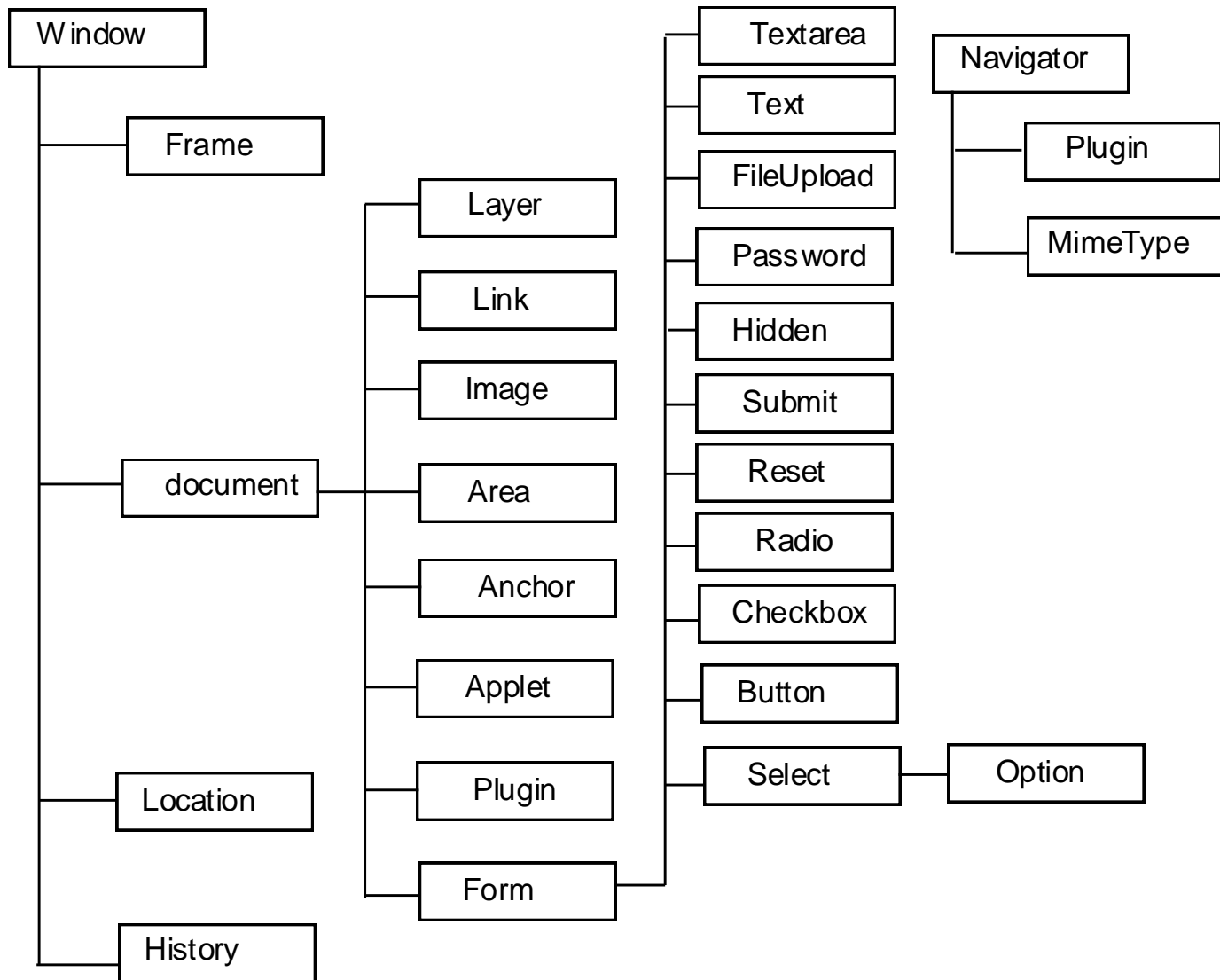
# Stratégie de développement cross-browser

- L'écriture de pages dynamiques va nécessiter de satisfaire les deux navigateurs
- Pour cela on peut
  - Développer des pages spécifiques à chaque navigateur
  - Faire des branchements internes (les scripts sont spécifiques à chaque navigateur)
  - Réaliser une API qui permet aux développeurs de sites de s'appuyer sur des fonctions qui vont traduire pour chaque navigateur

# Exemple de Détection du navigateur

```
var IEDyn; //vrai si IE > 4
var NSDyn; //vrai si NS > 4
var noDyn=true; //autre navigateur
If (parseInt(Navigator.appVersion) >=4) {
    if (navigator.appName=="Netscape") {
        NSDyn=true;
        IEDyn=false;
    }
    else {
        if (!navigator.appVersion.indexOf("Win")==-1) {
            IEDyn=true;
            NSDyn=false;
        }
    }
}
```

# Le DOM de Netscape



# Placer un script dans une page

- Les scripts peuvent être placés dans différents endroits d'une page HTML
- En général les fonctions sont dans un script de l'en-tête
- Pour éviter que le script ne soit interprété par des navigateurs qui ne reconnaissent pas le Javascript, on place des commentaires

```
<script type="text/JavaScript">
```

```
<!--
```

```
    écriture du script (les fonctions)
```

```
// -->
```

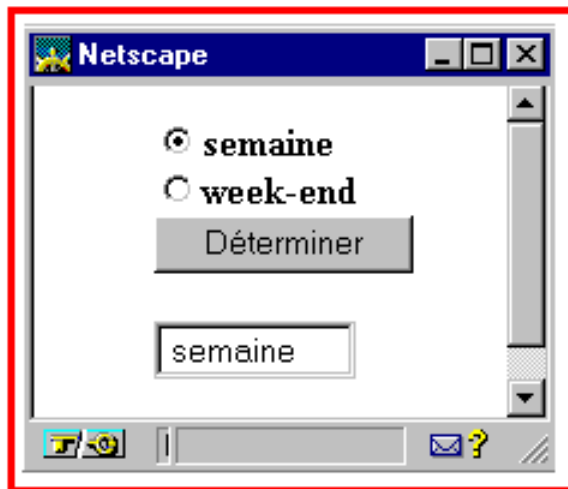
```
</script>
```

# Les objets en Javascript

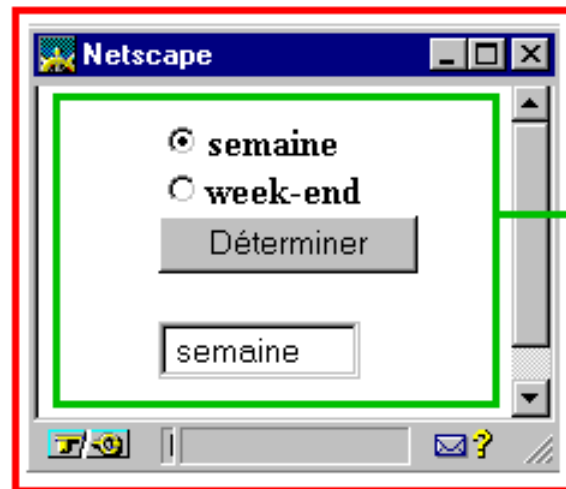
- De nombreux éléments HTML sont vus comme des "objets" dont le programme Javascript peut lire les caractéristiques, avant de les transformer.
- Un objet possède :
  - Des propriétés qui définissent les caractéristiques de l'objet
  - Des méthodes qui représentent les calculs sur l'objet



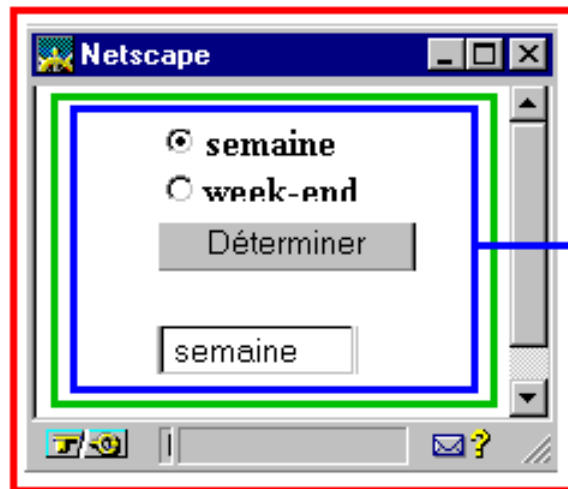
# Exemple de hiérarchie objet



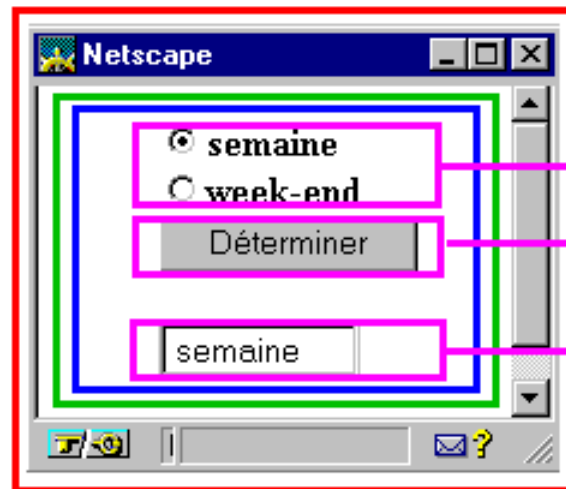
objet fenêtre



objet document



objet formulaire



objet radio

objet bouton

objet texte

# Les propriétés des objets

- Représentées par une notation hiérarchique à point : `nom-objet . nom-propriété`
- Objet Image :
  - `<IMG name="im1" src="icomes/arbre.gif" width="20" height="20" alt="un arbre">`
  - Cet élément est vu comme l'objet de nom `im1`
    - `im1.src` correspond à `"icomes/arbre.gif"`
    - `im1.width` correspond à `20`
    - `im1.height` correspond à `20`
- On peut changer certaines propriétés grâce à l'opérateur d'affectation =
  - `im1.src="icomes/voiture.gif"`

# Les méthodes des objets

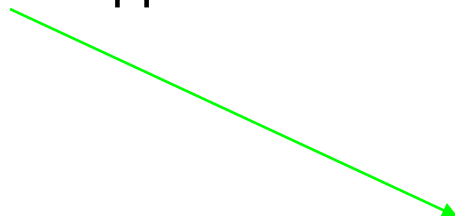
- Une méthode (ou fonction) permet d'effectuer des calculs ou de modifier les propriétés d'un objet
- Méthode `write()` de l'objet `document` :
  - `document.write("<p>Bonjour</p>");`
  - L'effet de la méthode `write` dépend de l'endroit où elle est placée dans le fichier HTML
- La chaîne de caractères donnée en paramètre peut-être :
  - Une simple chaîne de caractères : "Bonjour"
  - Une variable de type `String`
  - Du code HTML

# Exemple d'utilisation

```
<!DOCTYPE HTML public "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<TITLE>la méthode write</TITLE>
<STYLE type="text/css">
<!--
#p1 {
    color: red; font-size: 24pt; text-align: center;
}
-->
</STYLE>
<script type="text/JavaScript">
<!--
document.write("<p id='p1'>Bonjour</p>")
// -->
</script>
</HEAD>
<BODY>
<P align=center>le monde</P>
</BODY>
</HTML>
```

# Exemple d'utilisation : visualisation

Texte inséré par la méthode write qui se trouve dans l'entête du document. La mention d'un identifiant entraîne l'application de la feuille de style associée



Bonjour

le monde

# L'objet chaîne de caractères String

- L'objet String représente une chaîne de caractère (un morceau de texte)
- On crée cet objet par l'opérateur new
  - `chaine1=new String("Bonjour, mes amis");`
  - `chaine2=new String("Belle journée");`
- Cet objet possède plusieurs méthodes dont `length()` etc.

# Les événements intrinsèques

- Les éléments HTML peuvent capturer des événements qui correspondent aux actions de l'utilisateur
- On peut associer un handler qui capture l'événement et le redirige vers une fonction JavaScript
- Handlers associés aux mouvements de la souris ou du clavier :

**onClick** : clic sur la boîte de l'élément concerné

**ondblclick** : double clic (ne fonctionne pas avec tous les navigateurs)

**onmousedown** : le bouton de la souris est enfoncé

**onmouseup** : le bouton de la souris est relâché

**onmouseover** : le pointeur passe sur la surface de la boîte concernée

**onmouseout** : le pointeur quitte la boîte

**onmousemove** : le pointeur se déplace

**onkeypress** : une touche clavier est pressée

**onkeydown** : touche clavier maintenue enfoncée

**onkeyup** : touche clavier relâchée

# Exemple de handler

- Les handlers sont placés comme attributs de la balise de l'élément
- Exemple : un lien capte l'événement d'entrée et de sortie du pointeur de la zone représentant le lien. Ceci déclenche l'action Javascript associée.

```
<A href="destination.htm"
    onMouseOver="im1.src='../images/avionrouge.gif'"
    onMouseOut="im1.src='../images/avionbleu.gif'">
    <IMG src="../images/avionvert.gif"
        name= "im1"
        width = "50"
        height = " 50 "
        alt = "prends l'avion pour là bas"
        border=0>
</A>
```



# Les fonctions JavaScript

- Les interactions se font par le biais des handlers qui transmettent la capture de l'événement à une fonction qui exécutera une action
- Une fonction se déclare de la façon suivante :

```
function nom(parametres) {  
    //commentaire concernant la fonction  
    instruction1;  
    instruction2;  
    .....  
}
```

- Les déclarations de fonctions sont généralement placées dans l'entête. L'appel d'une fonction se fait par son nom et ses paramètres

```
function Exemple(Texte) {  
    alert(texte);  
}
```

Cette méthode affiche du texte dans une boite de dialogue bloquante

# Exemple de réalisation

- La fonction `changefond(couleur)` va changer la couleur de fond d'une division du document. L'événement déclenchant est le clic sur un bouton d'un formulaire grâce à `onClick`
- ```
function changefond(couleur) {  
    var p=document.getElementById("d1");  
    p.style.backgroundColor=couleur;  
}
```
- ```
<INPUT onClick="changefond('yellow');" type="submit" name="bouton1" value="Jaune">
```
- Voir un site de specifications du DOM :
  - <http://www.mozilla.org/docs/dom/>

# Conclusion et Ouverture

- Liberté dans la mise en page des sites web.
- La présentation est beaucoup simple.