Une page avec le style par défaut

css Zen Garden: The Beauty in ...

css Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the CSS Resource Guide for advanced tutorials and tips on working with CSS.

You may modify the style sheet in any way you wish, but not the HTML. This may seem daunting at first if you've never worked this way before, but follow the listed links to learn more, and use the sample files as a guide.

Download the sample html file and css file to work on a copy locally. Once you have completed your masterpiece (and please, don't submit half-finished work) upload your .css file to a web server under your control. Send us a link to the file and if we choose to use it, we will spider the associated images. Final submissions will be placed on our server.

Un premier style

css Zen Garden: The Beauty in ...

元 完全な技

Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished visually through CSS-based design. Select any style sheet from the list to load it into this page.

Download the sample html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.

Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.

The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The code remains the same, the only thing that has changed is the external .css file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. To date, most examples of neat tricks and hacks have been demonstrated by structurists and coders. Designers have yet to make their mark. This needs to change.

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the CSS Resource Guide for advanced tutorials and tips on working with CSS.

You may modify the style sheet in any way you wish, but not the HTML. This may seem daunting at first if you've never worked this way before, but follow the listed links to learn more, and use the sample files as a guide.

Download the sample html file and css file to work on a copy locally. Once you have completed your masterpiece (and please, don't submit half-finished work) upload your .css file to a web server under your control. Send us a link to the file and if we choose to use it, we will spider the associated images. Final submissions will be placed on our server.

Benefits

Why participate? For recognition, inspiration, and a resource we can all refer to when making the case for CSS-based design. This is sorely needed, even today. More and more major sites are taking the leap, but not enough have. One day this gallery will be a historical curiosity; that day is not today.

select a design:

- Under the Sea! by Eric Steltz
- Make 'em Proud by Michael McAghon and Scotty Reifsnyder
- Orchid Beauty by Kevin Addison
- Oceanscape by Justin Gray
- CSS Co., Ltd. by Benjamin Klemm
- Sakura by Tatsuya Uchida
- Kyoto Forest by John Politowski
- A Walk in the Garden by Simon Van Hauwermeiren

archives:

- next designs »
- View All Designs

resources:

- View This Design's CSS
- CSS Resources
- FAQ
- Submit a Design
- Translations

La feuille de style

http://www.cssz...den-sample.css

```
/* css Zen Garden default style v1.02 */
/* css released under Creative Commons License - http://creativecommons.org/licenses/by-nc-sa/1.0/  */

/* This file based on 'Tranquille' by Dave Shea */
/* You may use this file as a foundation for any new work, but you may find it easier to start from scratch. */
/* Not all elements are defined in this file, so you'll most likely want to refer to the xhtml as well. */

/* Your images should be linked as if the CSS file sits in the same folder as the images. ie. no paths. */


/* basic elements */
html {
        margin: 0;
        padding: 0;
        }
body {
        font: 75% georgia, sans-serif;
        line-height: 1.88889;
        color: #555753;
        background: #fff url(blossoms.jpg) no-repeat bottom right;
        margin: 0;
        padding: 0;
        }
p {
        margin-top: 0;
        text-align: justify;
        }
h3 {
        font: italic normal 1.4em georgia, sans-serif;
        letter-spacing: 1px;
        margin-bottom: 0;
        color: #7D775C;
        }
a:link {
        font-weight: bold;
        text-decoration: none;
        color: #B7A5DF;
        }
a:visited {
        font-weight: bold;
        text-decoration: none;
        color: #D4CDDC;
        }
a:hover, a:active {
        text-decoration: underline;
        color: #9685BA;
        }
acronym {
        border-bottom: none;
        }
```

Un autre style

Encore un autre style

Un dernier style

css Zen Garden: The Beauty in ...

Download the sample html file and css file

CSS Zen Garden

A demonstration of what can be accomplished visually through CSS-based design. Select any stylesheet from the list to load it into this page.

Select a Design

- **Under the Sea!** by Eric Stoltz
- **Make 'em Proud** by Michael McAghon and Scotty Reifsnyder
- **Orchid Beauty** by Kevin Addison
- **Oceanscape** by Justin Gray
- **CSS Co., Ltd.** by Benjamin Klemm
- **Sakura** by Tatsuya Uchida
- **Kyoto Forest** by John Politowski
- **A Walk in the Garden** by Simon Van Hauwermeiren

Archives

- **next designs »**
- **View All Designs**

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, and broken CSS support.
Today, we must clear the mind of past practices. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP and the major browser creators.
The css Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the (yet to be) time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About

There is clearly a need for CSS to be taken seriously by graphic artists. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page.
The code remains the same, the only thing that has changed is the external

Participation

Graphic artists only please. You are modifying this page, so strong CSS skills are necessary, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the CSS **Resource Guide** for advanced tutorials and tips on working with CSS.

Les feuilles de style

Introduction

Fonctionnement

- Format d'une règle
- Emplacement et priorité des règles
- Quelques propriétés

Concepts avancés

- Classes et ID
- Pseudo-classes
- Héritage
- Pseudo-éléments
- Boîtes, position et position

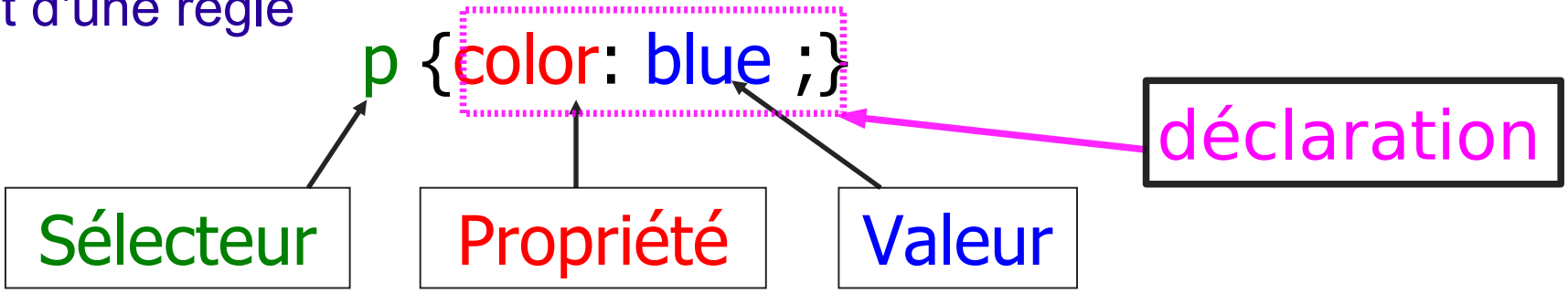
Cascading Style Sheets

- version 1.0 (1996)
- version 2.0 (1998)
- version 3.0 (divisé en modules : en 2011, couleurs, sélecteurs)

Buts

- dissocier le fond (HTML) de la forme (CSS)
  - rendre le code lisible + accélérer les transferts
  - homogénéiser les sites
  - facilité de maintenance/déploiement (changer le style des 98 pages d'un site en modifiant une ligne d'un seul fichier)
- adapter la forme au média
  - écran
  - impression
  - accessibilité

Format d'une règle



Plusieurs déclarations

```
p {color: red; font-size: 12px; font-weight: bold;}
```

Plusieurs sélecteurs

```
h1,h2,h3 {color: blue; font-style: italic;}
```

Plusieurs règles et 1 même sélecteur

```
h1,h2 {color: blue; font-weight: bold;}
h2 {font-style: italic;}
```

Où mettre les règles ?

À l'intérieur des balises HTML (style incorporé)

attribut *style*

```
<h2 style="color:green">
```

Dans l'en-tête du fichier HTML (style intégré)

balise `<style>`

```
<head>
   <style>
      h1 , h2 {color:green;}
      h1 {font-size:36pt ; padding:24pt;}
      p {color:blue;}
   </style>
   ...
</head>
```

Où mettre les règles ? (2)

Dans un fichier séparé (style lié)

monstyle.css

```
/* le fichier css */
h1,h2 {
   color:blue ;
   font-size:20pt;
}
h2 {font-style:italic;}
p {color:green;}
ul {color:red;}
```

Où mettre les règles ? (3)

Dans un fichier séparé (style lié)

balise `<link>`

fichier.html

```
<!DOCTYPE html>
<html lang="fr">
   <head>
        <link href="monstyle.css"
           rel="stylesheet" />
   </head>
   <body>
     …
   </body>
</html>
```

La cascade

Ordre de priorité d'application

+++++	style incorporé
++++	style intégré
+++	style lié
++	style défini du navigateur
+	style par défaut du navigateur

Exemple des priorités

monstyle.css

```
h1,h2 {color:blue; font-size:20pt;}
h2 {font-style:italic;}
p {color:green;}
ul {color:red;}
```

ex-prop.html (sans en-têtes)

```
<head>
  <link href="monstyle.css" rel="stylesheet" />
  <style type="text/css">
     h1 {color:green}
     p {color:blue}
  </style>
</head>
```

Exemple des priorités (suite)

```
    <body>
        <h1 style="color:red"> Les CSS </h1>
        <h2> Placement des règles </h2>
        <p> dans le fichier html : </p>
        <ul>
            <li> en attribut </li>
            <li> dans l'en-tête </li>
        </ul>
        <p style="font-size:20pt"> dans un fichier CSS </p>
    </body>
</html>
```

Les CSS

Placement des règles

dans le fichier html :

- en attribut
- dans l'en-tête

dans un fichier CSS

Les couleurs

Synthèse soustractive

- combinaison de matières de couleurs différentes
- couleurs primaires : cyan (bleu primaire), magenta (rouge primaire), jaune
- absence de couleur : blanc

Synthèse additive

- combinaison de sources de lumière de couleurs différentes
- couleurs primaires : rouge, vert, bleu
- absence de couleur : noir

Quelques exemples de styles

Définir les couleurs

par leur nom (`red, yellow, blue, gray`,..., il y a 16 couleurs standards)

en utilisant rgb (*red, green, blue*) pour coder les composantes (de 0 – rien à 255 – maximal)

idem mais avec les valeurs hexadécimales

La couleur du texte (**`color`**)

```
p  {color:red}
h1 {color:rgb(12,254,64)}
h2 {color:#0CFE40}
```

Changer la couleur du fond (**`background-color`**)

```
body {background-color:yellow}
p {background-color:red}
```

Quelques exemples de styles (suite)

Image de fond (**background-image**)

```
background-image:url("./images/nom.jpg")
background-image:url("http://www.image.com/img.jpg")
```

Répétition de l'image (**background-repeat**)

```
repeat/no-repeat/repeat-x/repeat-y
```

répétition/pas de répétition/répétition horizontale/répétition verticale

Défilement de l'image avec le texte (**background-attachment**)

```
scroll/fixed
```

Alignement de l'image (**background-position**)

```
top/center/left/right/bottom
```

Quelques exemples de styles (suite)

Les styles de texte

définir l'indentation (**`text-indent`**)

```
ex: p {text-indent:10px}
```

alignement du text (**`text-align`**)

```
left, right, center, justify
ex: p {text-align:center}
```

"décoration" du texte (**`text-decoration`**)

```
none, underline, overline, line-through
ex: cite {text-decoration:overline}
```

Quelques exemples de styles (suite)

Les fontes

utiliser une fonte (**`font-family`**)

noms par défaut sur les systèmes :

- `serif` : avec empattement
- `sans-serif` : sans empattement
- `cursive` : aspect manuscrit
- `fantasy` : polices décoratives
- `monospace` : chaque caractère a la même largeur

ex :

```
body {font-family: "times new roman", "times", "serif"}
h1   {font-family: "arial","verdana","sans-serif"}
```

Quelques exemples de styles (suite)

Ce texte est dans une fonte « serif »

Ce texte est dans une fonte « sans-serif »

Ce texte est dans une fonte « cursive »

Ce texte est dans une fonte « fantasy »

Ce texte est dans une fonte « monospace »

Quelques exemples de styles (suite)

Les fontes (suite)

taille d'une fonte (**font-size**)

la taille se définit :

en pixel (`1 px = 0,75pt`), en point (`pt`), en cm (`cm`)

proportionnellement à la valeur courante (`%`)

proportionnellement à la valeur courante (`em`) (1em = 100%)

ex :
```
body {font-size: 12pt}
h1{font-size: 200%}
```

"épaisseur" de la fonte (**font-weight**)

valeurs : `normal, bold, bolder,…`

ex : `strong {font-weight: bolder}`

style de la fonte (**font-style**)

valeurs : `normal, italic, oblic,…`

ex : `p {font-style: italic}`

Quelques exemples de styles (fin)

Les listes

style de (la puce de) liste (**`list-style-type`**)

valeurs : `none, disc, circle, square, decimal, lower-alpha,…`

ex : `ul {list-style-type:disc}`

etc.

CSS – concepts avancés

- id et class
- Pseudo-classes
- Héritage et sélecteurs
- Pseudo-éléments
- Les boîtes
- Le positionnement

id et class

But

permettre à une règle de ne s'appliquer qu'à une partie des objets contenus dans un même type de balise

Exemple

```
<h2> Titre normal </h2>
<h2 class="special"> Titre spécial </h2>
<h2> Un autre titre normal </h2>
<h2 class="special"> Encore un titre spécial </h2>
<h3 class="special"> un sous-titre spécial </h3>
```

```
h2 {font-family:Helvetica; color:black}
h2.special {color:red}
```

Titre normal

Titre spécial

Un autre titre normal

Encore un titre spécial

un sous-titre spécial

id et class

Mode d'application

en cas d'ambiguïté, c'est toujours la règle la plus *spécifique* qui s'applique (ici `h2.special`)

s'appliquent à toutes les balises

Question

comment appliquer les règles à tous les éléments de classe *special* ?

Réponse

le sélecteur *****

```
*.special {color:red}
.normal {color:black} /* abbréviation */
```

id et class

```
h2 {font-family:Helvetica; color:black}
*.special {color:red}
```

Titre normal

Titre spécial

Un autre titre normal

Encore un titre spécial

un sous-titre spécial

id et class (suite)

Les id

les id sont des classes particulières dont la valeur n'apparaît qu'une seule fois dans le document HTML

Exemple

```
<h1 id="premierTitre"> Un titre unique </h1>
<h1> Un titre comme les autres </h1>
<h1> Un autre titre normal </h1>
```

Sélecteur

```
h1#premierTitre {text-align:center}
```

id et class (suite)

Un titre unique

Un titre comme les autres

Un autre titre normal

Sélecteurs d'attributs

Principe

Sélection d'éléments en fonction de leurs attributs

Syntaxe

e[att] : élément *e* ayant l'attribut *att*

e[att="val"] : élément *e* dont l'attribut *att* a pour valeur *val*

Exemple

```
img[title]   /* les images avec un attribut title */
a[href="http://www.univ-artois.fr"] /* les liens vers
 le site http://www.univ-artois.fr */
```

pseudo-classes dynamiques

Principe

Classes qui ne sont pas attachées à des balises mais qui sont activées quand certains **événements** surviennent (activation, survol du curseur, etc.).

Cas des liens `<a>`

`:link` : le lien dans son état normal (pas encore visité)

`:visited` : si le lien a déjà été visité

un lien pas encore visité

un lien déjà visité

Toutes les balises (pseudo-classes dynamiques)

`:hover` : la souris pointe sur (survole) l'élément

`:active` : l'utilisateur est en train d'activer l'élément
par exemple : cliquer sur un lien

pseudo-classes dynamiques

Remarque importante

Attention à l'ordre des déclarations :
La dernière déclaration gagne

Exemple

```
a:link {color:black}
a:visited {color:gray}
a:hover {text-decoration:none}
a:active {color:navy}
```

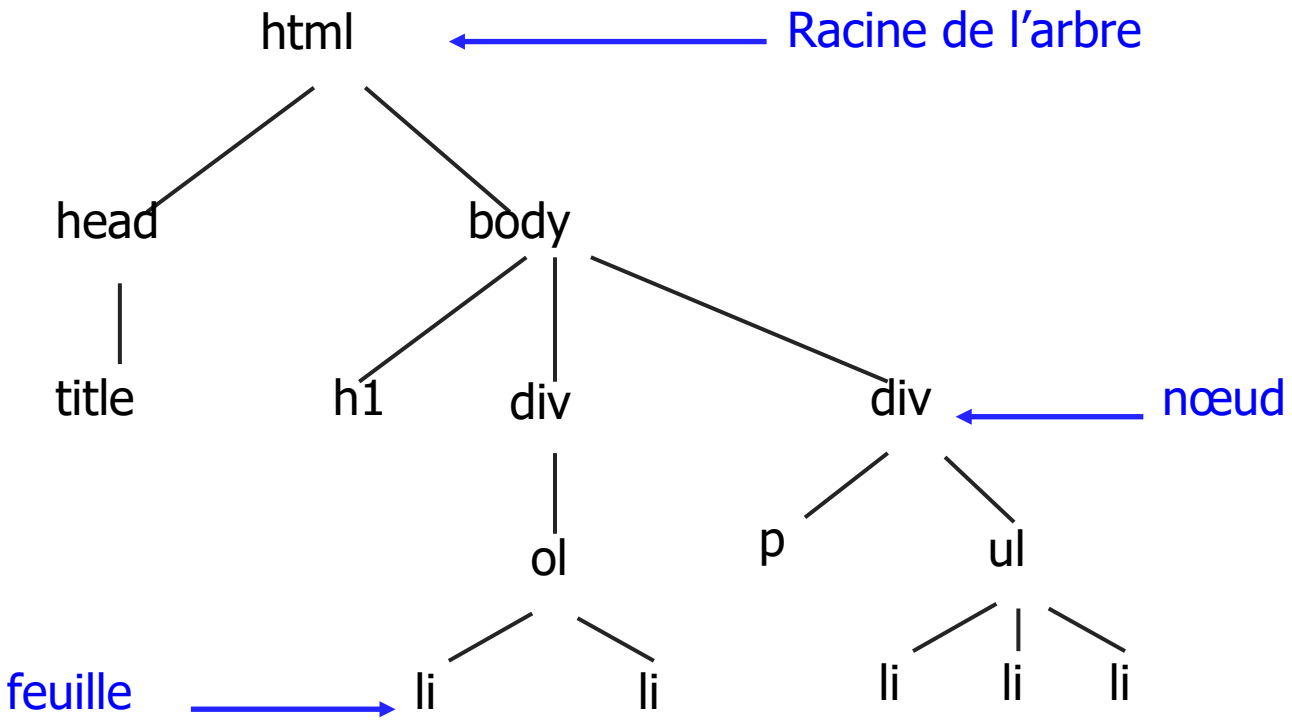
Héritage dans les CSS

Une page est un arbre !

Exemple (sans les en-têtes) :

```
<html>
    <head>
        <title> TITRE </title>
    </head>
    <body>
        <h1> titre 1 </h1>
        <div>
            <ol>
                <li> ligne 1</li>
                <li> ligne 2</li>
            </ol>
        </div>
        <div>
            <p> blabla </p>
            <ul>
                <li> ligne 1</li>
                <li> ligne 2</li>
                <li> ligne 3</li>
            </ul>
        </div>
    </body>
</html>
```

Arbre correspondant



div est le père de *ul*
h1 et *div* sont des frères
ol est fils direct de *div* et fils indirect de *html*

Remarque : les feuilles sont *ordonnées* de la gauche vers la droite

Héritage des propriétés

Principe dans les CSS

Les styles appliqués à un nœud père sont appliqués à tous les nœuds fils : on dit que les fils *héritent* des propriétés des parents.

Exemple

```
body {
    font-family:verdana,helvetica,sans-serif;
    font-size:14pt;
} /* définit une police et une taille par défaut */
h1 {font-size:80%} /* 80% de 14 pt */
h1 span {font-size:40%} /* 40% de 80% de 14 pt ! */
```

Exceptions

les propriétés concernant les marges, les bordures, le remplissage, le placement, etc.

pseudo-classes structurelles

Principe

permettent de sélectionner un élément en fonction de sa position par rapport à son élément parent

Pseudo-classes structurelles

```
p :first-child {color:blue} /* le premier paragraphe
 de l'élément père */

ul li:nth-child (5) /* le 5ième élément d'une liste
 non numérotée */

ul li:nth-child (2n+1) /* tous les éléments impairs
 d'une liste non numérotée */

ul li:last-child /* le dernier élément d'une liste
 non numérotée */
```

Pseudo-classe de négation

Principe

s'applique à un sélecteur simple

ne peut pas s'appliquer à elle-même

Pseudo-classe de négation

```
li:not(:first-child) {color:blue} /* tous les
 éléments de liste sauf le premier */
```

```
:not (:not(...))
```
invalide

Sélecteur simple

Sélecteur simple

sélecteur de type : `h1, p, …`

sélecteur universel : `*`

sélecteur d'attribut : `[att], [att="valeur"], ...`

sélecteur de classe : `.classe`

sélecteur d'identificateur : `#ident`

pseudo-classe : `:pseudo-classe`

Séquence de sélecteurs simples

commence par le sélecteur universel ou un sélecteur de type

suivi de zéro ou plusieurs sélecteurs simples

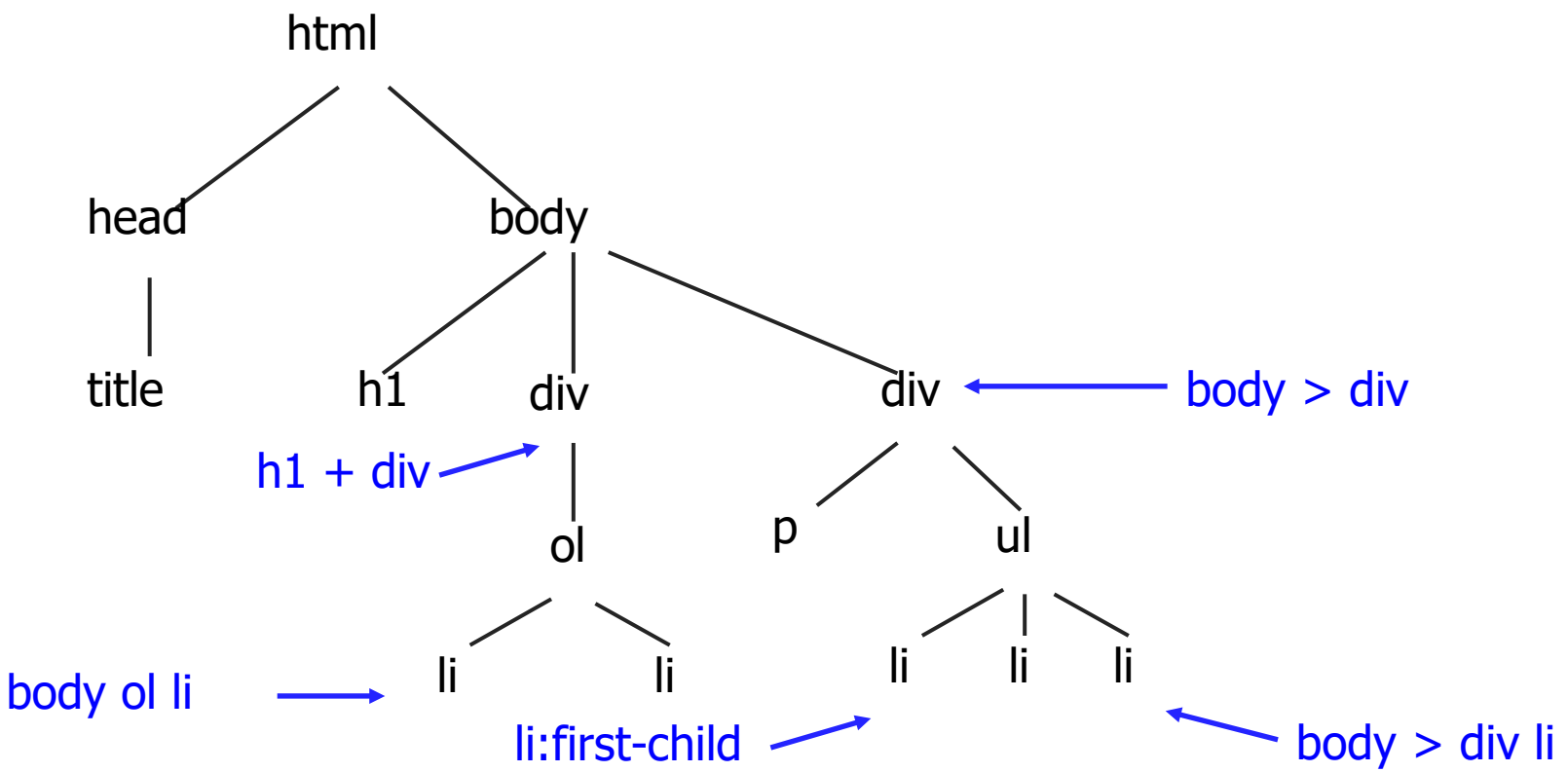
Principe

permettent de sélectionner n'importe quel élément de l'arbre

Combinateurs

```
div␣p {color:blue}   /* les p qui ont div comme ancêtre */
div > p {color:blue} /* les p qui ont un div
                        directement comme parent */
div + p {color:blue} /* p petit frère adjacent d'un div */
div ~ p {color:blue} /* p petit frère d'un div */
```

Exemples


html
head
body
title
h1
div
div
body > div
h1 + div
ol
p
ul
li
li
li
li
li
body ol li
li:first-child
body > div li

pseudo-éléments

Principe

permettent de récupérer des parties particulières d'un élément

Quelques pseudo-éléments

`::first-letter` permet de récupérer la première lettre

Exemple :

```
p::first-letter {
    font-size:300%;
    float:left
} /* création d'une lettrine */
```

Ceci est un paragraphe. La première lettre de ce paragraphe a un style particulier grâce au pseudo-élément *::first-letter* : elle est agrandie et elle flotte à gauche (de façon à être répartie sur plusieurs lignes). Nous verrons les flottants au prochain cours.

Quelques pseudo-éléments (suite)

`::first-line` permet d'accéder à la première ligne d'un élément

Exemple :

```
p::first-line {font-variant:small-caps}
/* les lettres de la première ligne (de longueur
  variable) sont en petites majuscules */
```

CECI EST UN PARAGRAPHE. LA PREMIÈRE LIGNE DE CE paragraphe a un style particulier grâce au pseudo-élément *::first-line* : elle est en petites majuscules.

Quelques pseudo-éléments (suite)

`::before` `et` **`::after`** permettent d'ajouter un texte avant et après

Exemple :

```
<span class="age"> 20 </span>
```

```
span.age::before {content:"Age: "}
span.age::after {content:"ans"}
/* affichera Age: 20 ans */
```

Sélecteur

chaîne de séquences de sélecteurs simples séparées par des combinateurs

un pseudo-élément peut être ajouté à la fin

s'applique sur la dernière séquence de sélecteurs simples

Exemples

```
p:not(:first-child) > a[href="#haut"]:hover

section#intro p:first-child::first-line
```

CSS – concepts avancés

id et class

Pseudo-classes

Héritage et sélecteurs

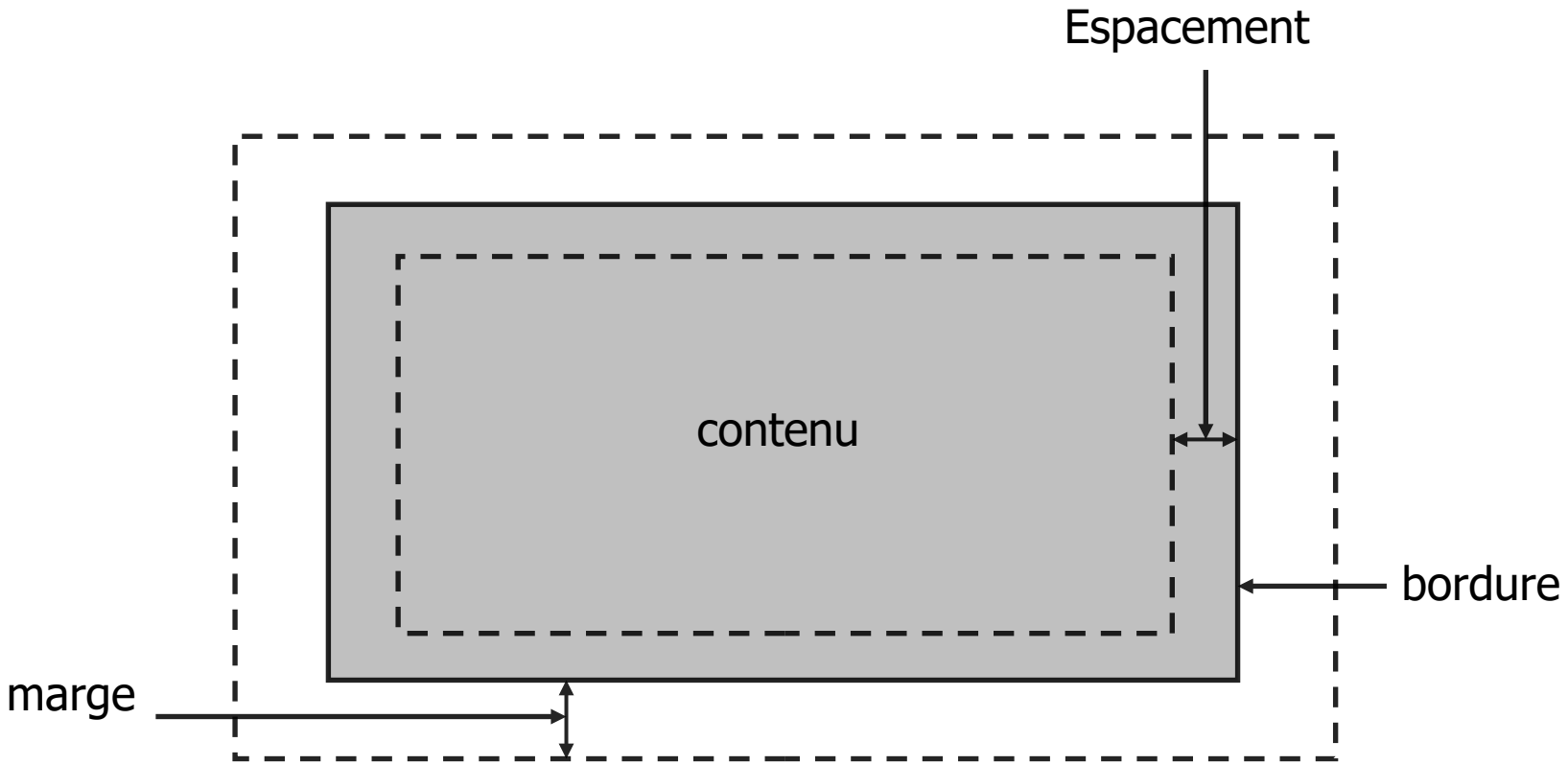
Pseudo-éléments

Les boîtes

Le positionnement

Principe

une boîte est associée à tout élément (nœud de l'arbre du document HTML)



Parties d'une boîte

De l'intérieur vers l'extérieur

Contenu

propriétés `height` et `width`

Espacement (`padding`)

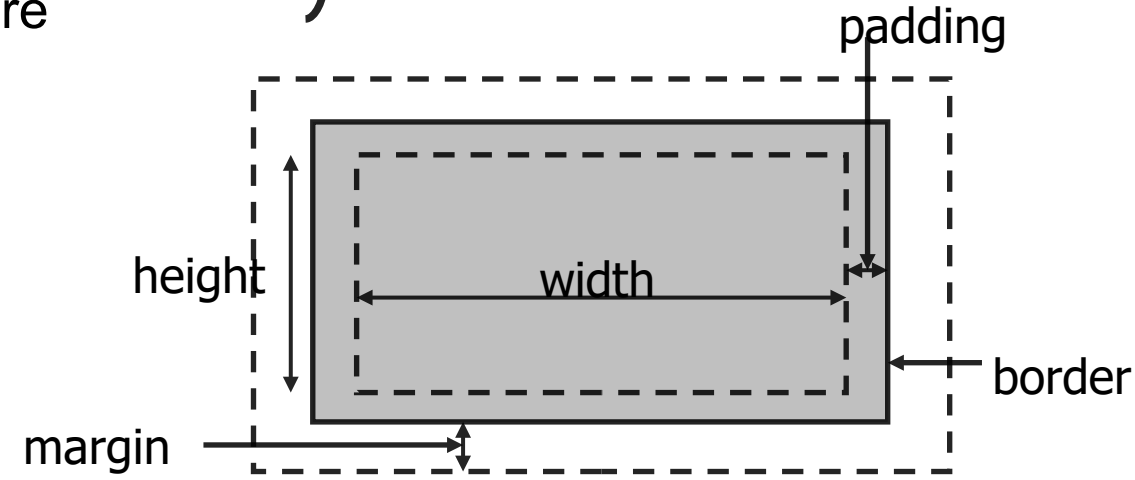
espace entre le contenu et la bordure

Bordure (`border`)

Marge (`margin`)

espace au-delà de la bordure

Segments
haut (`top`),
droit (`right`),
bas (`bottom`),
gauche (`left`)



en gris la partie visible

la marge est transparente

Propriétés des espacements

La distance du contenu à la bordure (`padding`)

`padding-top, padding-right, padding-bottom, padding-left`

`padding`

```
padding: 5cm; /* top right bottom left*/
padding: 5cm 4cm; /* top bottom, right left*/
padding: 5cm 4cm 3cm; /* top, right left, bottom*/
padding: 5cm 4cm 3cm 2cm; /* top, right, bottom, left*/
```

La distance au-delà de la bordure (`margin`)

`margin-top, margin-right, margin-bottom, margin-left`

`margin`

NB : sauf exception, les marges verticales (`top, bottom`) fusionnent (max des valeurs)

Valeur des propriétés

longueur (nombre + unité de mesure)

pourcentage de la largeur du contenu (`width`)

Propriétés des bordures

Les bordures

`border-style`

`border-top-style, border-right-style, border-bottom-style, border-left-style`

`border-width`

`border-top-width, border-right-width, border-bottom-width,border-left-width`

`border-color`

`border-top-color, border-right-color, border-bottom-color, border-left-color`

la couleur par défaut de la bordure est celle du contenu (`color`)

Utilisation des raccourcis `border-style, border-width, border-color` avec 1 à 4 valeurs, voir exemple précédent avec `padding`

Propriétés des bordures

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

largeur, style et couleur de l'une des bordures

exemple :

```
h2 {border-bottom : thin double navy}
h2 {border-bottom : thin double}
```

`border`

largeur, style et couleur des 4 bordures

Exemple

```
...
p  {
   background-color:pink;
   border-width:medium;
   border-style:solid;
   border-color:blue black green yellow;
   padding:30pt;
   margin:1cm;
   width:6cm;
}
...
...
<p> Que c'est beau ! </p>
...
```

CSS – concepts avancés

id et class

Pseudo-classes

Héritage et sélecteurs

Pseudo-éléments

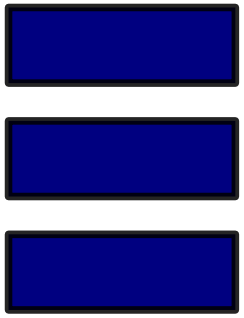
Les boîtes

Le positionnement

Placement des éléments

Éléments affichés en bloc (`block`)

`h1, p, div, ...`



Éléments affichés en ligne (`inline`)

`a, img, em, cite, span, …`

Placement des éléments

Éléments affichés en ligne avec des propriétés de bloc, telles que largeur et hauteur (`inline-block`)

```
img, …
```

Pour les modifier : `display`

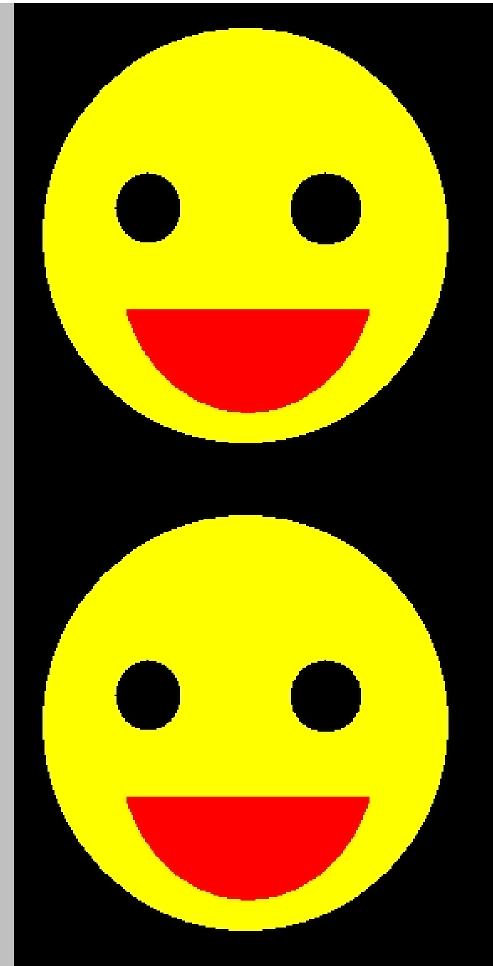
```
img {
        display : block;
        margin : auto;
}
```

Placement des éléments : display

Affichage par défaut


bla bla bla bla bla
blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla
bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla
blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla
blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla
blabla bla blabla bla blabla bla bla

Placement des éléments : display

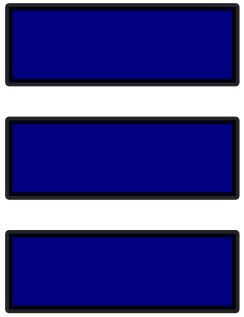


bla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla bla bla blabla bla blabla bla blabla bla blabla bla bla

Éléments affichés en bloc (`block`)

Affichés les uns après les autres verticalement

Notion de **flux**



Placement des éléments

Définir la position (`position`)

`static`

`relative`

`absolute`

`fixed`

Exemple

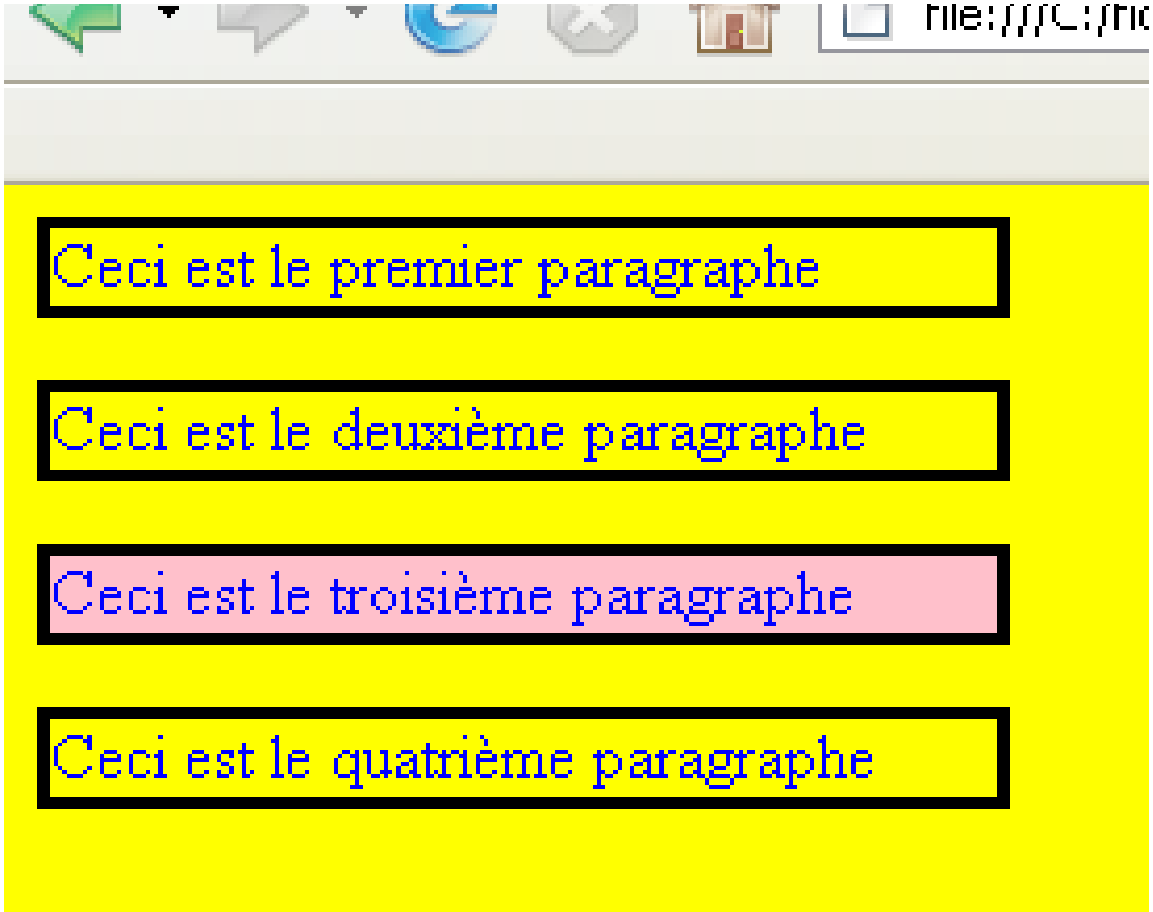
```
/* css */

body {
  background-color:yellow;
}
p {
  color:blue;
  border-style:solid;
  border-color:black;
  width:6cm;
}
p#specialpara {
  background-color:pink;
}
```

```
<!-- html -->
<p>
   Ceci est le premier paragraphe
</p>
<p>
   Ceci est le deuxième paragraphe
</p>
<p id="specialpara">
   Ceci est le troisième paragraphe
</p>
<p>
   Ceci est le quatrième paragraphe
</p>
```

Static

chaque boîte est placée après la précédente (position par défaut)

boîte dans le flux normal



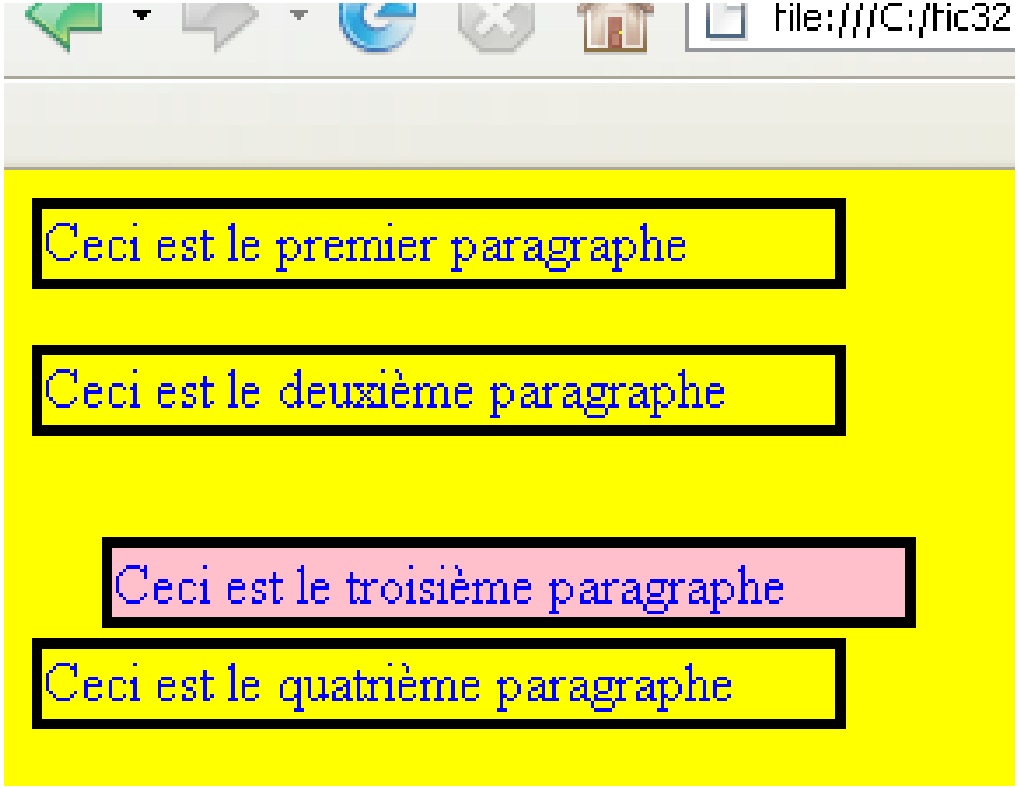
Exemple (suite)

relative

la boîte est placée par rapport à sa position statique (la place statique est conservée).

boîte dans le flux normal → boîte suivante placée normalement

```
p#specialpara {
  background-color:pink;
  position:relative;
  top:10pt;
  left:20px;
}
```



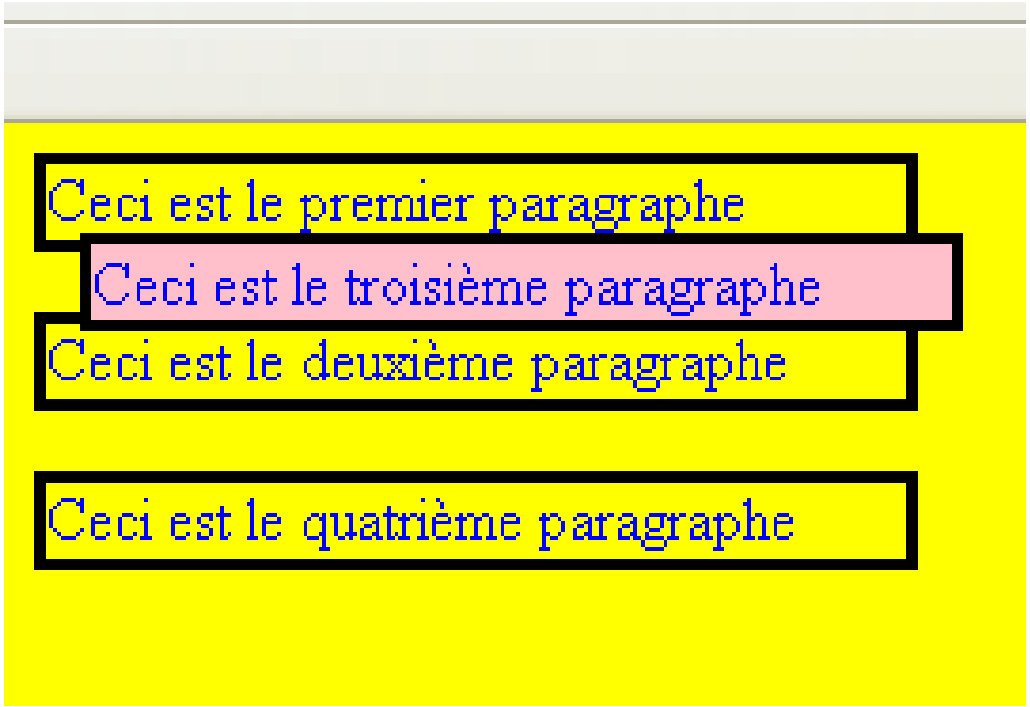
Exemple (suite)

absolute

la boîte est placée par rapport à son conteneur (se déplace en même temps)

boîte hors du flux normal → aucun effet sur le placement de la boîte suivante

```
p#specialpara {
  background-color:pink;
  position:absolute;
  top:10pt;
  left:20px;
}
```



Exemple (suite)

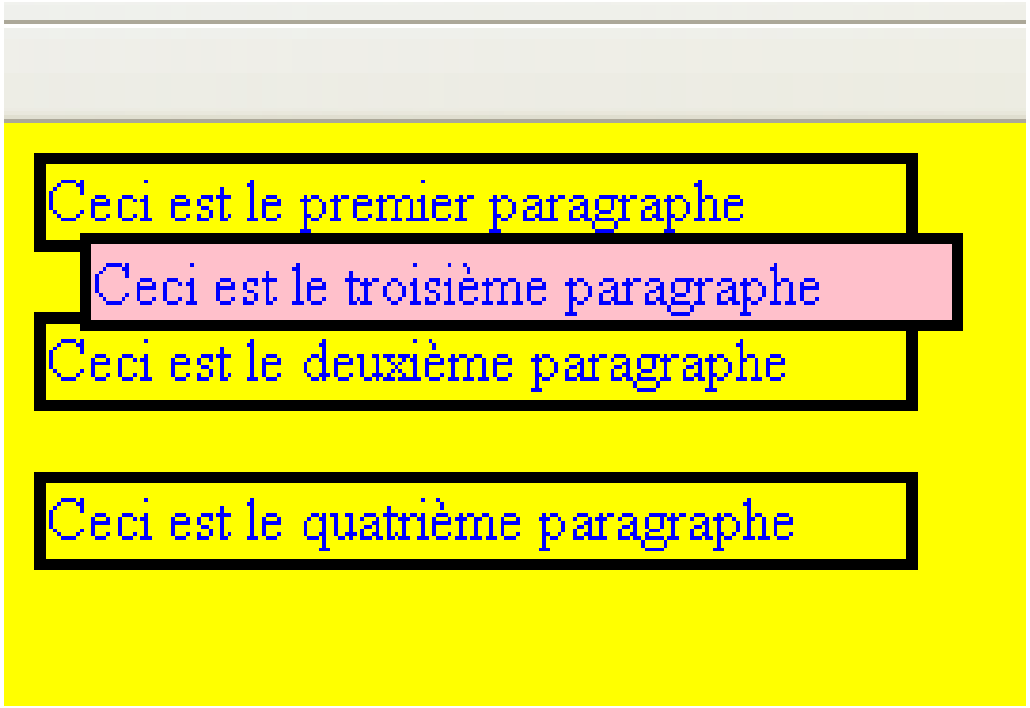
fixed

la boîte est placée par rapport à la fenêtre du navigateur

elle ne se déplace pas lorsque la page défile.

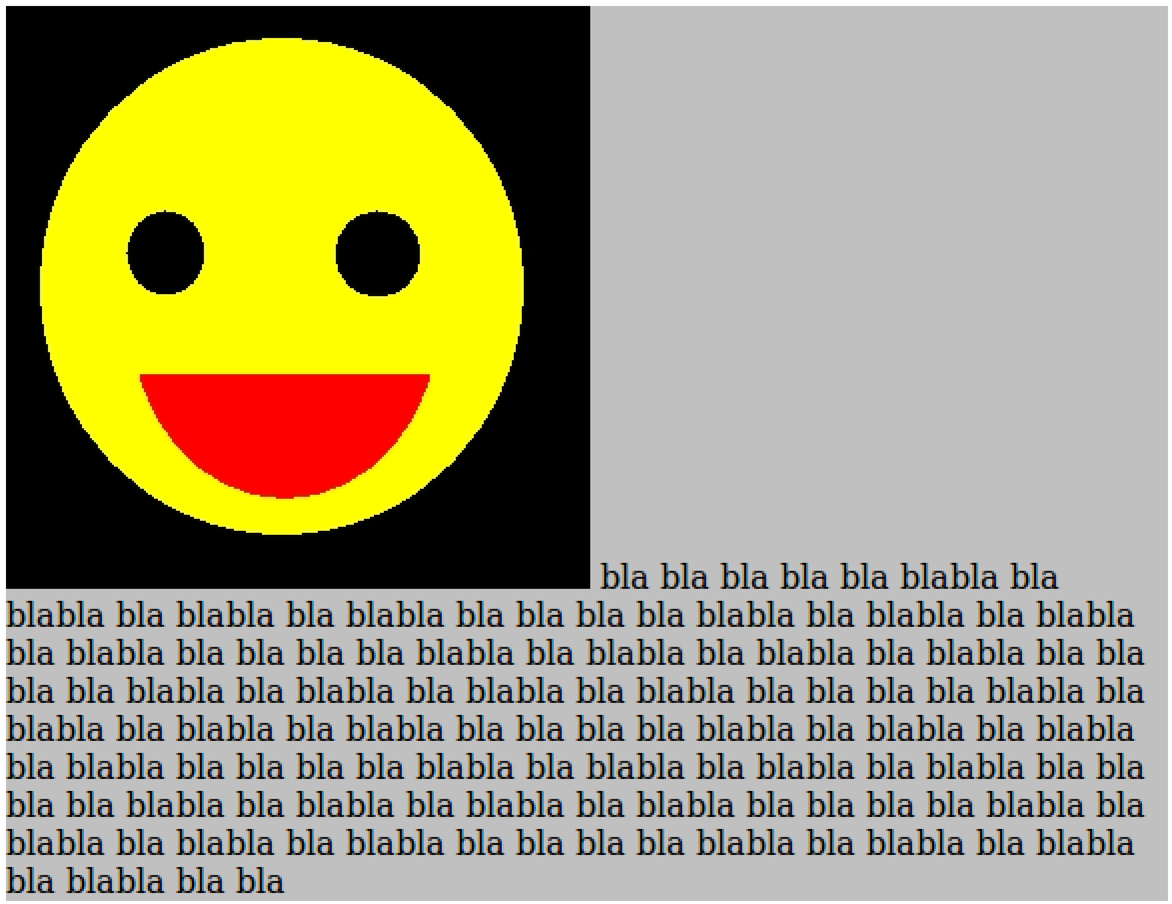
boîte hors du flux normal → aucun effet sur le placement de la boîte suivante

```
p#specialpara {
  background-color:pink;
  position:fixed;
  top:10pt;
  left:20px;
}
```



Sans rien

<p> bla bla bla … </p>



Les flottants

Avec flottant

```
<p> <img .../> bla bla bla … </p>
```

```
img {float:left}
```

But

permettre à un contenu de se répartir le long d'un autre élément

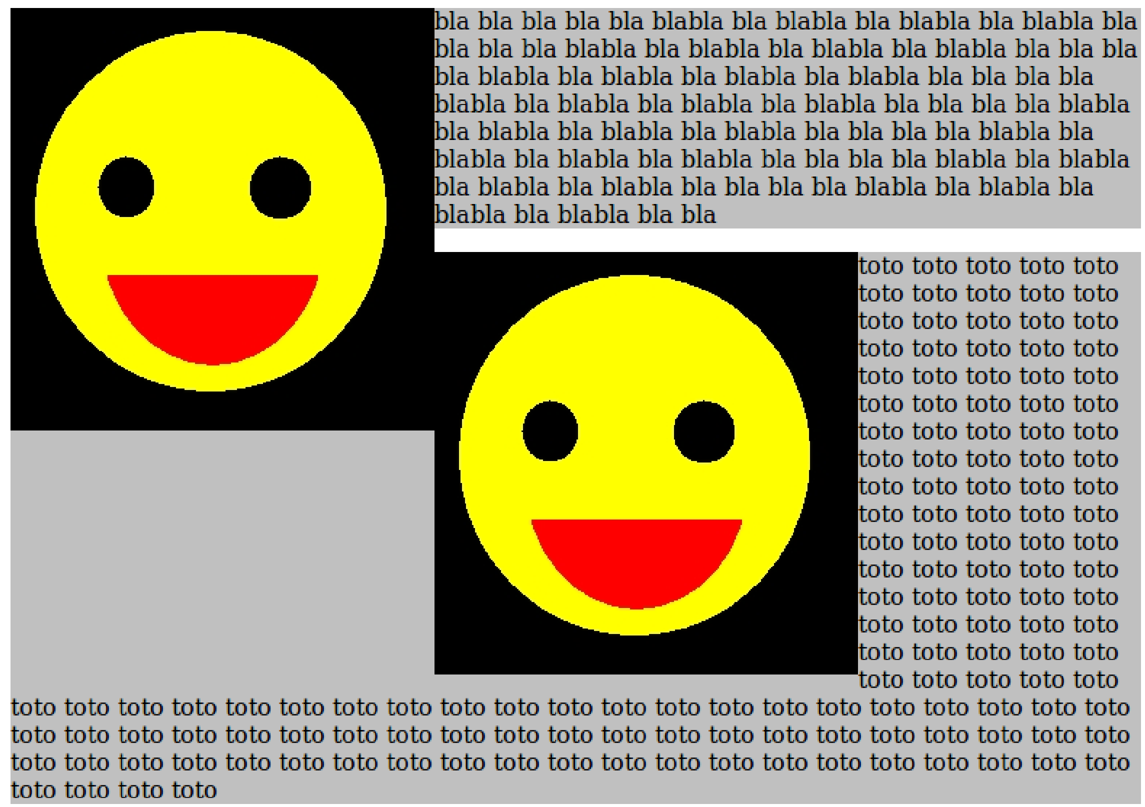
un flottant est hors du flux normal

Les flottants

Les flottants sont placés le plus haut possible

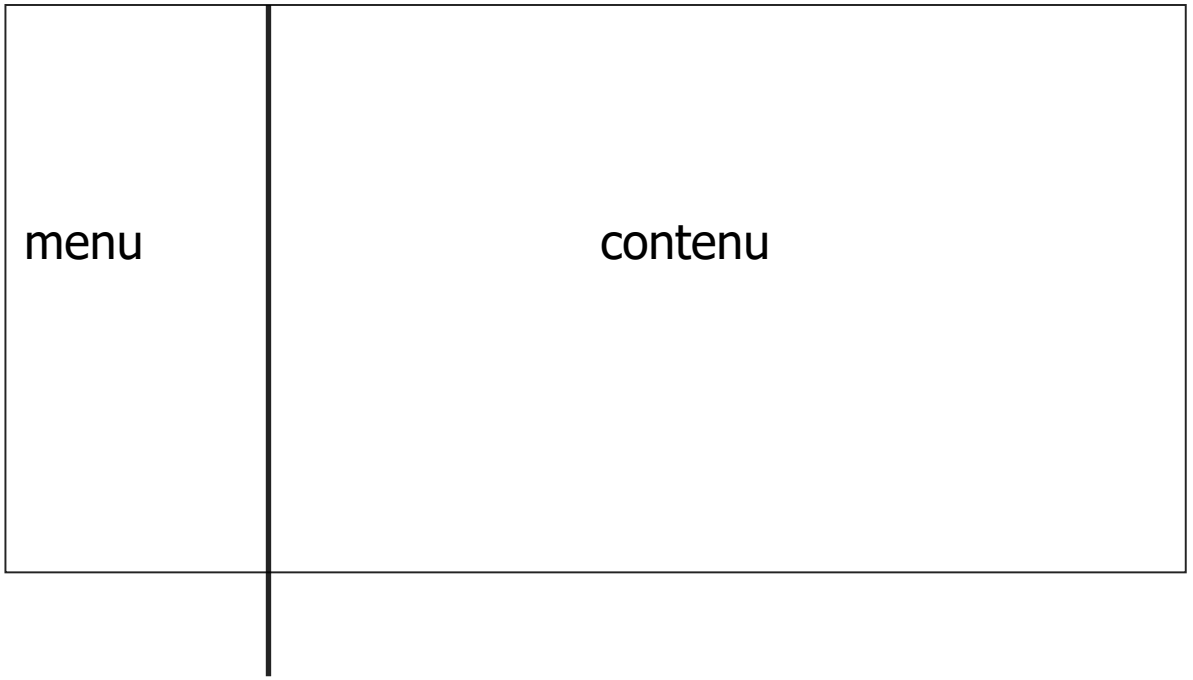
```
<p> <img .../> bla bla bla … </p>
<p> <img .../> toto toto toto … </p>
```

```
img {float:left}
```



Mise en page en 2 colonnes

pour cela, on utilise deux flottants !



Exemple

Fichier html

```
...
<div id="menu">
   ...
</div>
<div id="contenu">
   ...
</div>
...
```

CSS

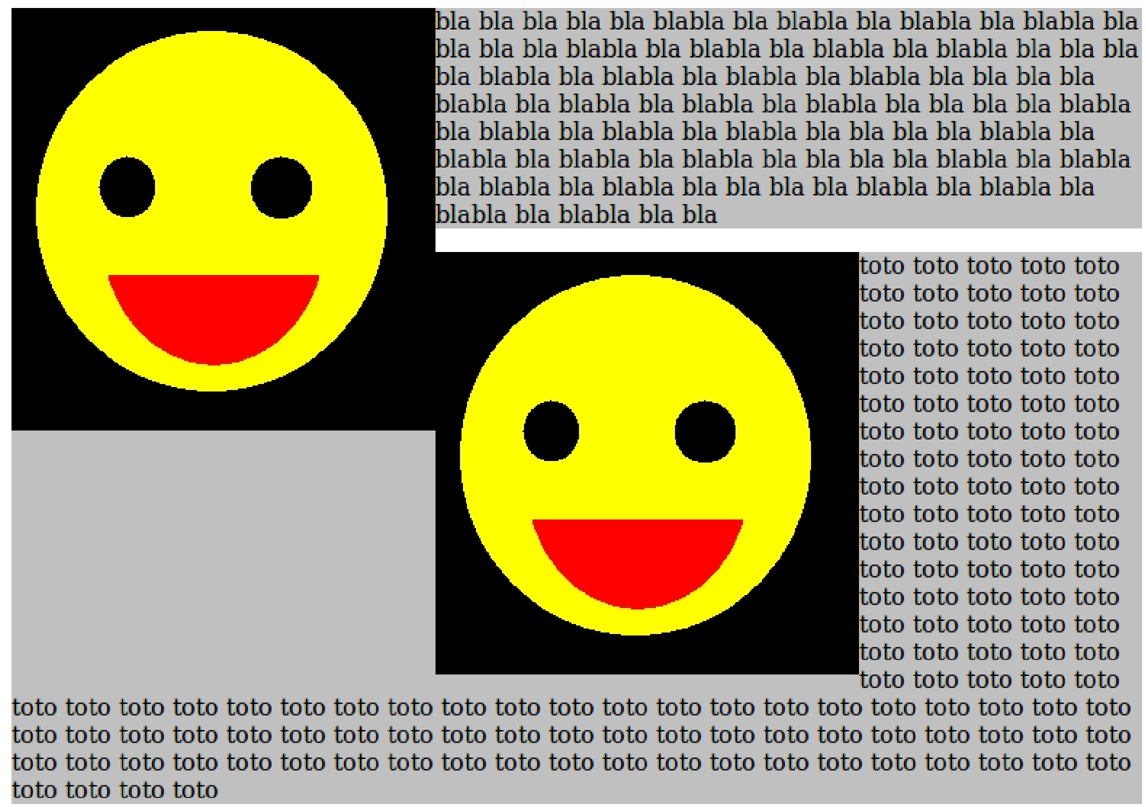
```
div#menu {
   width:10%;
   float:left
}
div#contenu {
   width:85%;
   float:left
}
```

clear

Le problème des flottants trop courts

```
<p> <img .../> bla bla bla … </p>
<p> <img .../> toto toto toto … </p>
```

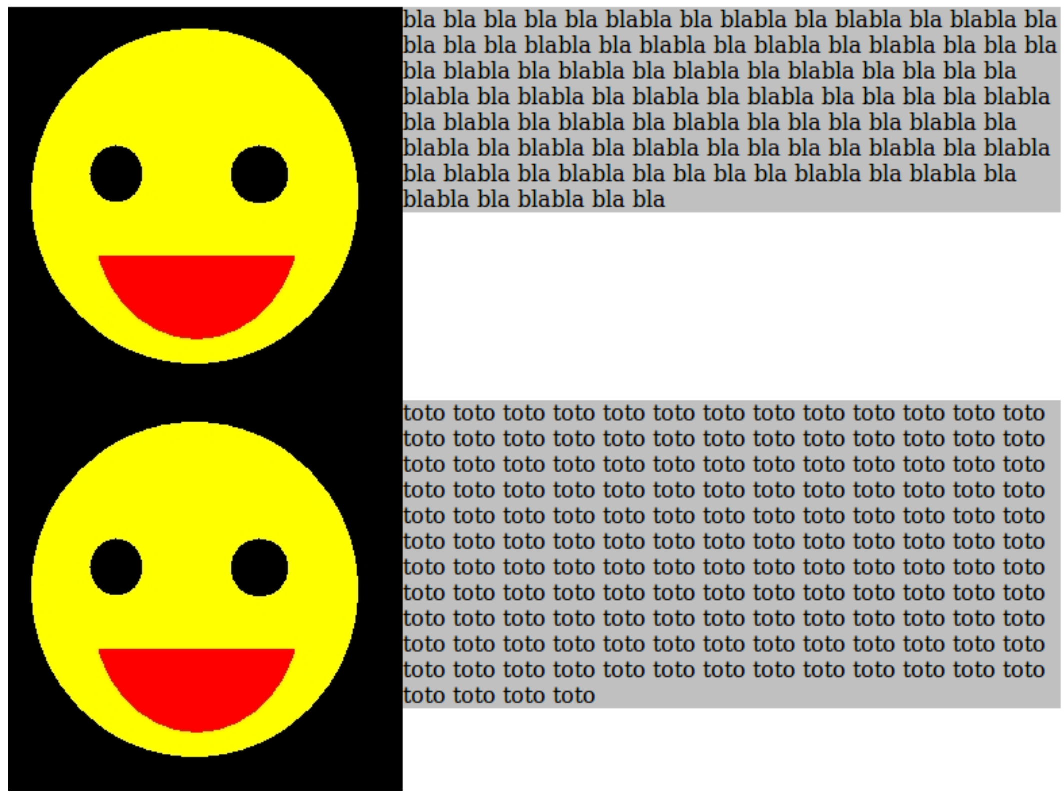
```
img {float:left}
```



But

Repousser l'élément en-dessous des flottants précédents situés à droite et/ou à gauche

```
p {clear:both}
```



Mise en page : Grid Layout vs Flexbox

Les boîtes flexibles (flexbox) permettent d'organiser du contenu sur une dimension (sur une ligne ou sur une colonne)

Grid Layout est un module spécifique au CSS permettant de faire une mise en page bidirectionnelle.

On peut mixer les 2 principes, mais on ne peut pas faire de Grid dans Flex.

Flexbox

Le module des boîtes flexibles, aussi appelé « flexbox », a été conçu comme un modèle de disposition unidimensionnel et comme une méthode permettant de distribuer l'espace entre des objets d'une interface ainsi que de les aligner.

Nouvelle valeur pour display : flex (sur le container)

les flexbox gèrent une seule dimension à la fois : une ligne ou une colonne.

Lié au sens de lecture du document (de gauche à droite en français, et de droite à gauche dans une langue arabe)

L'axe principal est défini par la propriété **flex-direction** et l'axe secondaire est l'axe qui lui est perpendiculaire.

- row : gestion en ligne (valeur par défaut)

- column : gestion en colonne

Propriété pour les composants : flex

Raccourci pour flex-grow, flex-shrink et flex-basis

Flex-grow : facteur de grossissement utilisé. Plus la valeur est élevée, plus l'élément sera étendu si nécessaire. La valeur par défaut est 0

Flex-shrink : facteur de rétrécissement utilisé. Plus la valeur est élevée, plus l'élément sera compressé si nécessaire. La valeur par défaut est 1

Flex-basis : taille initiale principale. longueur absolue ou un pourcentage relatif à la taille principale du conteneur flexible ou encore le mot-clé auto (valeur par défaut). **Prioritaire sur width**

Propriété pour les composants : flex

Par défaut, les éléments flexibles ne se rétrécissent pas en dessous de la taille minimale du contenu. Pour modifier ce comportement, il faudra paramétrer min-width ou min-height

Flex-wrap

La propriété flex-wrap indique si les éléments flexibles sont contraints à être disposés sur une seule ligne ou s'ils peuvent être affichés sur plusieurs lignes avec un retour automatique.

Valeurs admises :

nowrap : affichage sur une seule ligne avec débordement de capacité du container

wrap : création de nouvelles lignes de haut en bas

wrap-reverse : création de nouvelles lignes de bas en haut

Justify-content : façon d'aligner les éléments sur l'axe primaire

- normal / stretch : espace disponible équitablement réparti entre tous les boîtes (valeur par défaut)

- flex-start : boites alignées sur la ligne de début

- flex-end : boites alignées sur la ligne de fin

- center : boites centrées

Align-items : façon d'aligner les éléments sur l'axe secondaire

- stretch : toutes les boîtes ont la taille de la plus grande (valeur par défaut)

- flex-start : boites alignées sur la ligne de début

- flex-end : boites alignées sur la ligne de fin

- center : boites centrées

Grid Layout

Méthode calquée sur le même fonctionnement que les tableaux, le CSS Grid utilise donc des colonnes et des lignes. Cependant sa structure n'est pas fixée, ce qui lui permet de faire des mises en page plus variées

Nouvelle valeur pour display : grid (sur le container)

Grid Layout

Méthode calquée sur le même fonctionnement que les tableaux, le CSS Grid utilise donc des colonnes et des lignes. Cependant sa structure n'est pas fixée, ce qui lui permet de faire des mises en page plus variées

Nouvelle valeur pour display : grid (sur le container)

L'unité fr

Les pistes peuvent être définies à l'aide de n'importe quelle unité de mesure. Les grilles proposent aussi une nouvelle unité de mesure pour aider à la création de pistes flexibles. Cette unité, fr, représente une fraction de l'espace disponible dans le conteneur de la grille.

Grid-template-columns

Permet de définir la taille des colonnes

Exemples :

grid-template-columns: 500px 1fr 2fr ;

grid-template-columns: repeat(3, 1fr);

/* équivalent à 1fr 1fr 1fr */

Grid-auto-rows:

Permet de définir la taille des lignes

Exemple :

grid-auto-rows:100px;