Rendu et positionnement

Sylvain Tenier, Romain Vallée, Vincent Derrien

Département TIC - Esigelec

2016 - Semestre 7

Plan

Plan

Page web : ensemble de zones rectangulaires

FIGURE : Structuration de page avec les balises sémantiques HTML 5

Le modèle de boîte du w3c

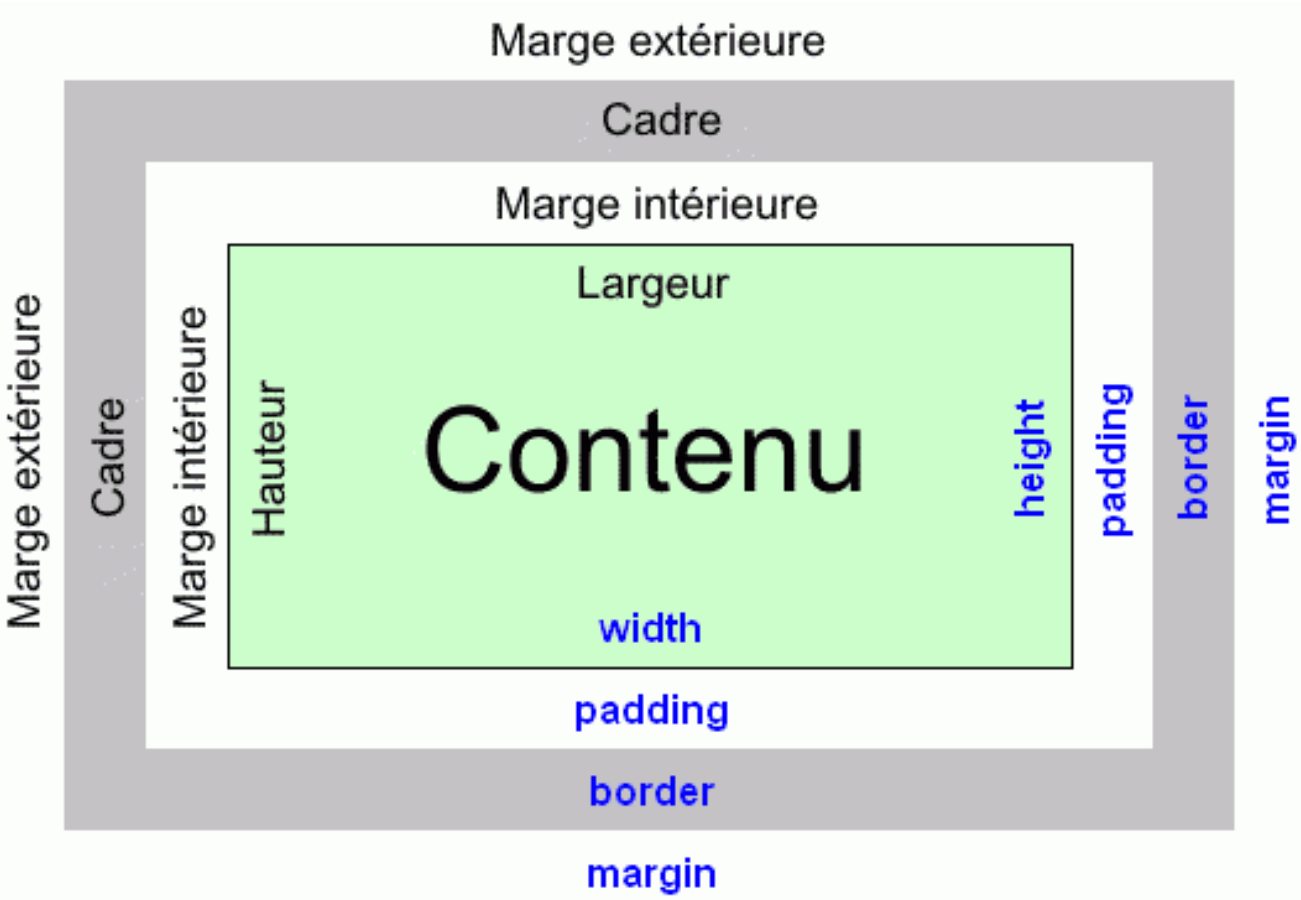


FIGURE : Rendu d'un élément HTML

Analyse d'une boîte

- 4 composantes
  1. La zone de contenu (`width`, `height`)
  2. Une bordure (`border`)
  3. Une marge interne (`padding`)
  4. Une marge externe (`margin`)
- Norme w3c : espace occupé = `width` + `padding` + `border`
  - Attention au mode *quirks* d'Internet Explorer

Application 1 : calcul de l'espace occupé

Règle CSS applicable à tous les paragraphes

```
1 p{
2   width: 100px;
3   padding: 10px;
4   margin: 5px;
5   border: 1px solid #000;
6   background-color: yellow;
7 }
8
```

- Quelle largeur occupe un paragraphe ?

Largeur totale : `width` + `padding` + `border`

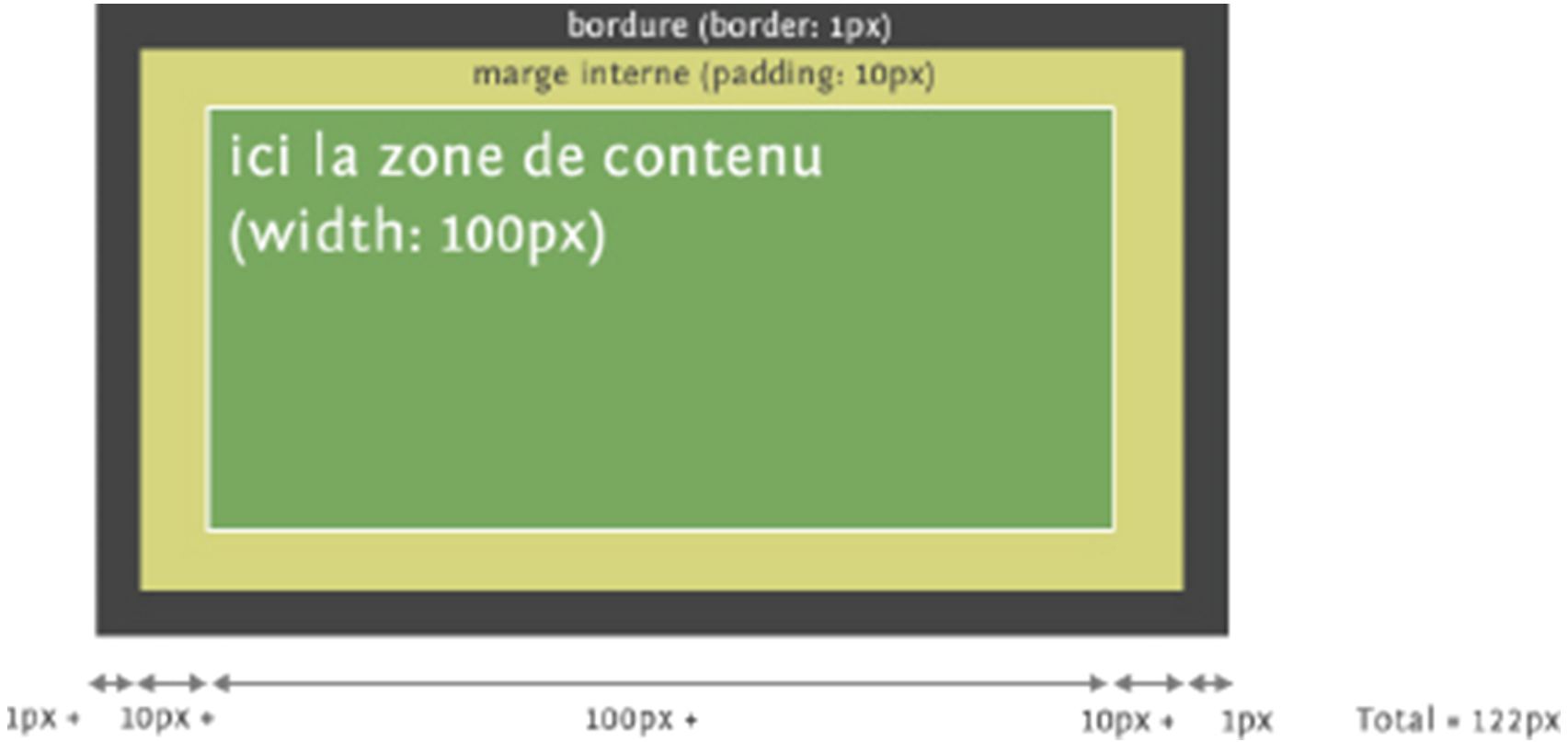


FIGURE : La marge interne et la bordure *s'ajoutent* au contenu

Application 2 : fluidité

- Testez le code suivant :

```
1  <!DOCTYPE html>
2  <html><head><meta charset="utf-8"><title>Appli2</title>
3    <style>
4      body{width: 60%; padding: 0 10px; margin: 0 auto;}
5      #p1{border: 1px solid red;
6      width: 100%;
7      height: 50px;
8      font-size: 32px;
9      background-color: yellow;}
10   </style></head><body>
11   <p id="p1"> Le contenu de ce paragraphe restera-t-il dans sa
       boite ?</p>
12   </body>
13 </html>
14
```

- Agrandissez la taille de la police dans le navigateur (`Ctrl +`). Que constatez-vous ?

Bonnes pratiques

- Privilégier les conceptions fluides
  - Éviter de fixer une hauteur si le contenu peut être redimensionné par l'utilisateur
- Fixer des minima/maxima
  - Largeur : `min-width` / `max-width`
  - Hauteur : `min-height` / `max-height`
- Tester !

Plan

Rappel : sélecteurs CSS

Sélecteur d'élément `x`

sélectionne tous les éléments `x` de la page

Sélecteur d'identifiant `#ix`

sélectionne l'élément *unique* possedant l'attribut `id` correspondant

- Page non valide si attribut id dupliqué
- Code HTML : `<p id="ix">`

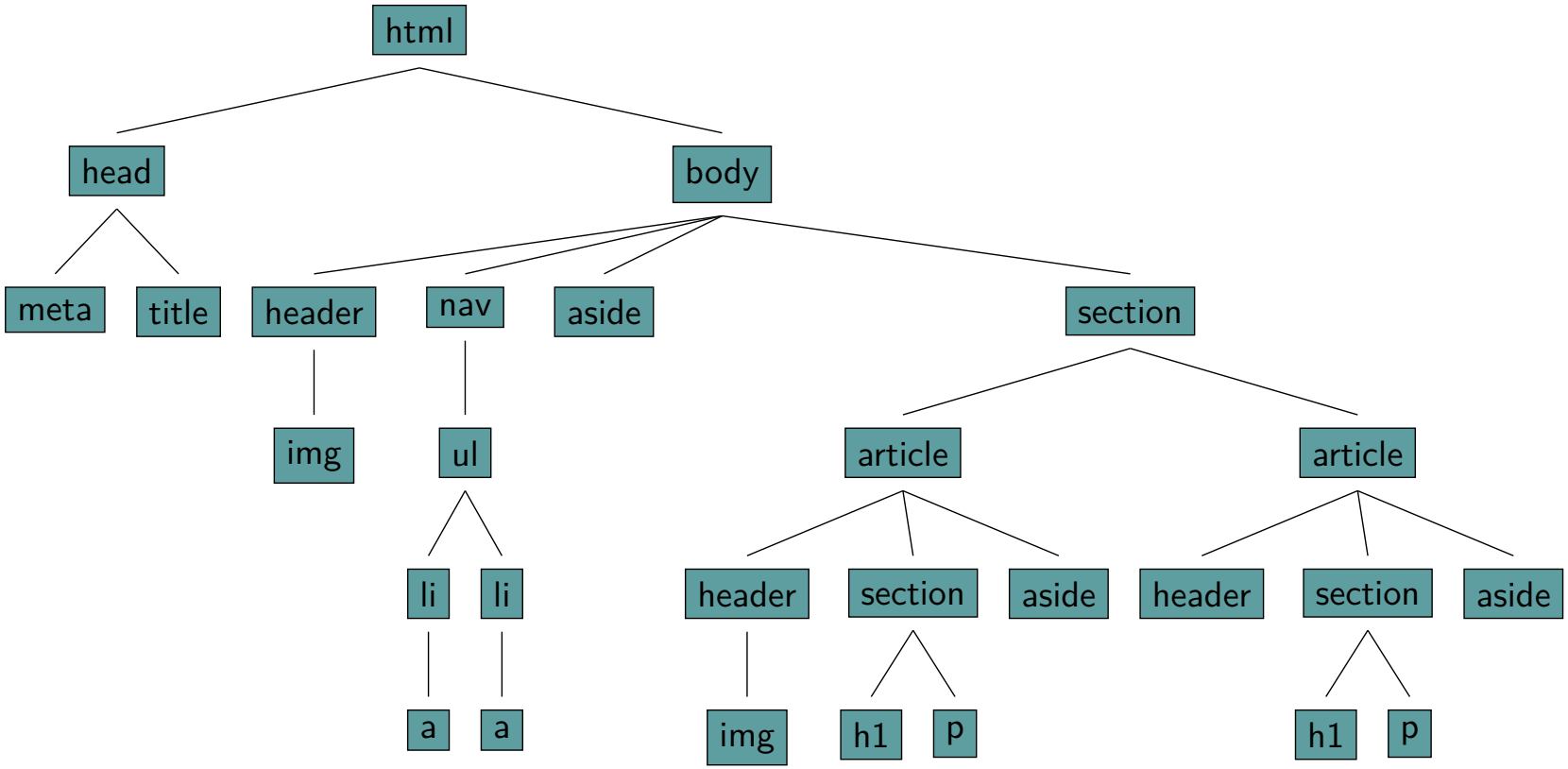
Sélecteur de classe `.cx`

sélectionne les éléments possédant l'attribut `class` correspondant

- Code HTML : `<p class="cx">`

Structure arborescente d'une page Web

DOM : Document Object Model



Sélecteurs CSS avancés

1. Pseudo-classe `:first-child`
   - Ex : `article:first-child` sélectionne le premier article d'une série d'articles
2. Pseudo-éléments `:first-letter, :first-line`
3. Sélection des fils directs `>`
   - Ex : `body > header` sélectionne le ou les éléments `header` dont le parent *direct* est un `body`
4. Sélection du frère `+`
   - Ex : `nav + aside` ne sélectionne que les `aside` précédés d'un élément `nav`
5. Sélection des descendants (espace)
   - Ex : `section header` sélectionne les `header` ayant un `section` comme ancêtre

Application 3 : priorités des sélecteurs

À partir du fichier "structureHtml5.html" récupéré sur ENT

1. Affichez la page dans Chrome
2. Cliquez sur "inspecter l'élément"
3. Ajoutez les règles CSS suivantes directement dans Chrome

```
1 article{ color: pink;}
2 #articles article{ color: red;}
3 #art1 { color: blue;}
4 article{ color: yellow;}
5
```

Pour ajouter une règle, cliquez sur le bouton + situé à droite dans l'onglet "Styles" de l'inspecteur Chrome

- Qu'en déduisez-vous sur les priorités ?

Que dit l'inspecteur ?

Priorités et résolution de conflits

- En cas de conflit, la règle qui prévaut est fonction du sélecteur
- Un poids est donné :
  1. à la présence ou non de la règle dans un attribut `style`
  2. au nombre d'*identifiants* présents dans le sélecteur
  3. au nombre de classes (ou d'attributs ou pseudo-classes)
  4. au nombre d'éléments (ou pseudo-éléments)
- En cas d'égalité, l'ordre d'apparition de la règle prévaut (*Cascading* StyleSheet)
- Mode de calcul complet :
  - `http://openweb.eu.org/articles/cascade_css/`

Plan

Type de rendu des éléments HTML

La propriété `display`

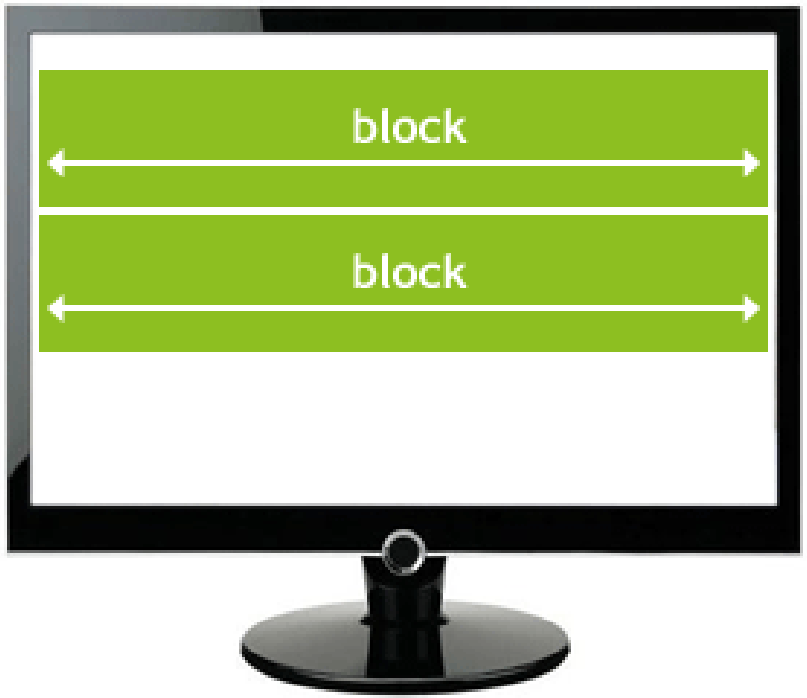
- La propriété `display` définit la manière dont un élément se positionne dans le document
- Ses valeurs possibles sont :
  1. `block`
  2. `inline`
  3. `list-item`
  4. `inline-block`
  5. `table, table-cell, table-row`
  6. `none`
- Référence :`http://www.alsacreations.com/tuto/lire/530-La-structure-des-balises-bloc-et-en-ligne.html`

Type de rendu “block”

L’élément

- se place en dessous du bloc précédent
- occupe toute la largeur disponible dans le conteneur
  - le conteneur est l’élément parent dans l’arbre DOM
- peut être redimensionné (`width`,`height`)

- Exemples : `<p>`, `<div>`



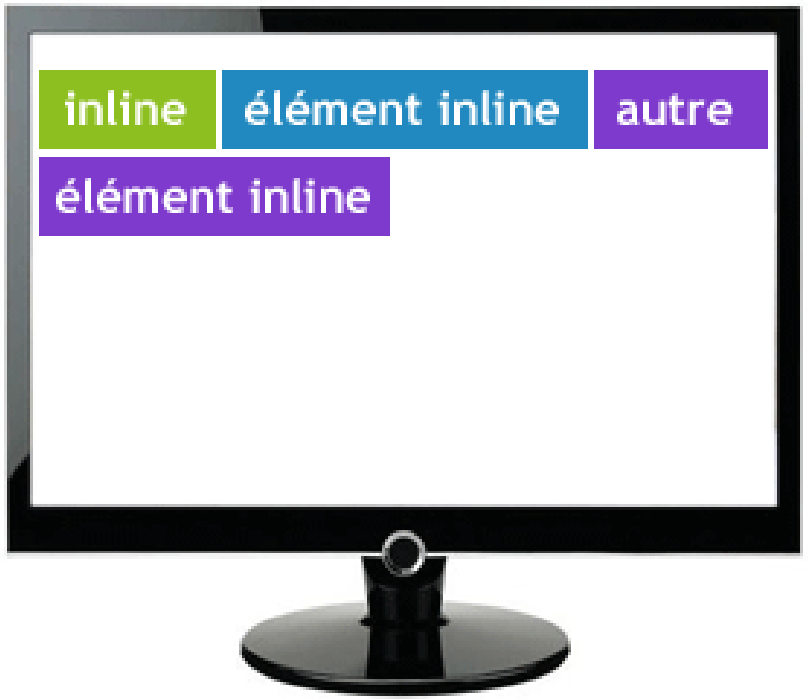
Type de rendu “inline”

L’élément

- se place à la suite de l’élément qui le précède
- reste dans la même ligne
- est non positionnable
- est non redimensionnable

- Exemples :

  `<span>`, `<a>`, `<button>`

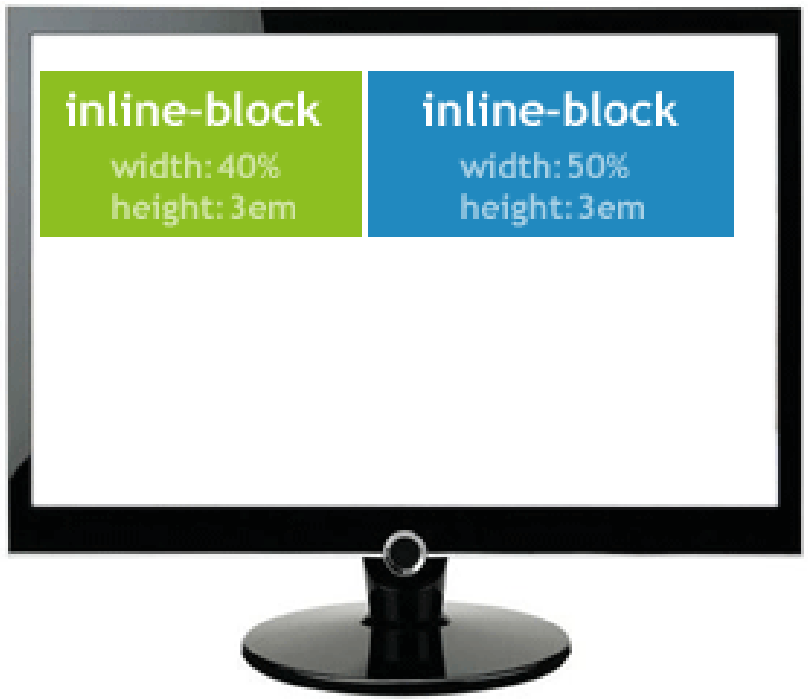


Type de rendu “inline-block”

L’élément

- se comporte comme un élément `inline`
- mais peut être dimensionné

- Aucun élément n’est concerné par défaut



Type de rendu “list-item”

L’élément

- se comporte comme un élément `block`
- possède des propriétés supplémentaires liées aux puces
  - `list-style`
  - `list-style-type`
    `list-type-image`
    `list-type-position`

- Exemple : `<li>`

Type de rendu “none”

L’élément

- disparaît entièrement de l’affichage

- Aucun élément n’est concerné par défaut
- La manipulation en javascript de la déclaration `display:none;` permet de créer des animations

Plan

Plan

Positionnement dans le flux courant

Chaque élément est affiché

- dans l'ordre du document HTML
- sur le même plan que les autres éléments
- le plus en haut à gauche au sein de son conteneur

Deux éléments frères

- s'empilent verticalement si la déclaration `display:block` leur est associée
- se positionnent sur la même ligne (si l'espace est suffisant) si la déclaration `display:inline` leur est associée

Exemple de positionnement dans le flux courant

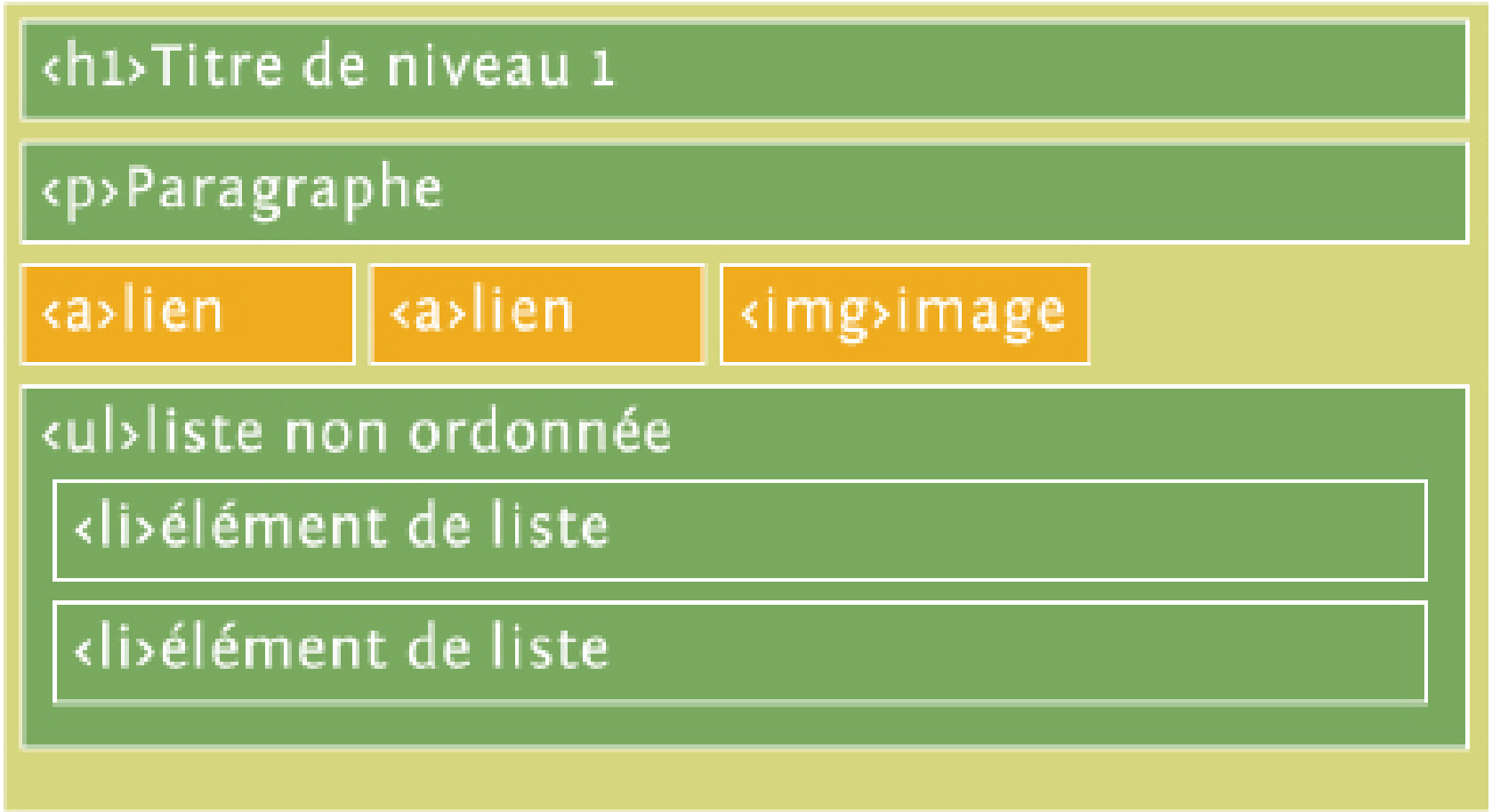


FIGURE : Positionnement par défaut

Plan

Hors flux : positionnement absolu

Fonctionnement du positionnement absolu

Déclaration : `position:absolute;`

- Fonctionne comme un calque
  1. L'élément sort du flux
  2. Les autres l'oublient et prennent sa place
- Propriétés :

  `top, right, bottom, left`
- Exemple :

```
1  #p2{
2    position: absolute;
3    right: 0; top: 0;
4  }
5
```

FIGURE : Avant

FIGURE : Après

Hors flux : positionnement absolu

Propriétés particulières

L'élément positionné en absolu

- se positionne en référence au premier ancêtre lui-même positionné
- devient dimensionnable, même avec la déclaration `display:inline`
- occupe exactement la largeur de son contenu
- peut être étiré

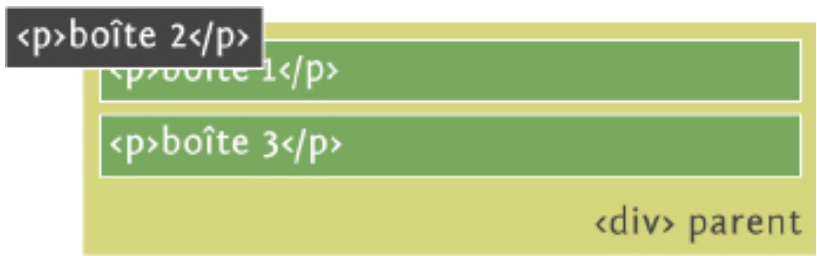


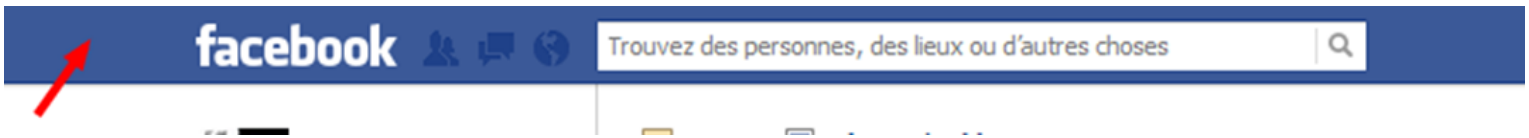
FIGURE : Parent non positionné

FIGURE : Parent positionné

Hors flux : positionnement fixé

La déclaration `position:fixed;`

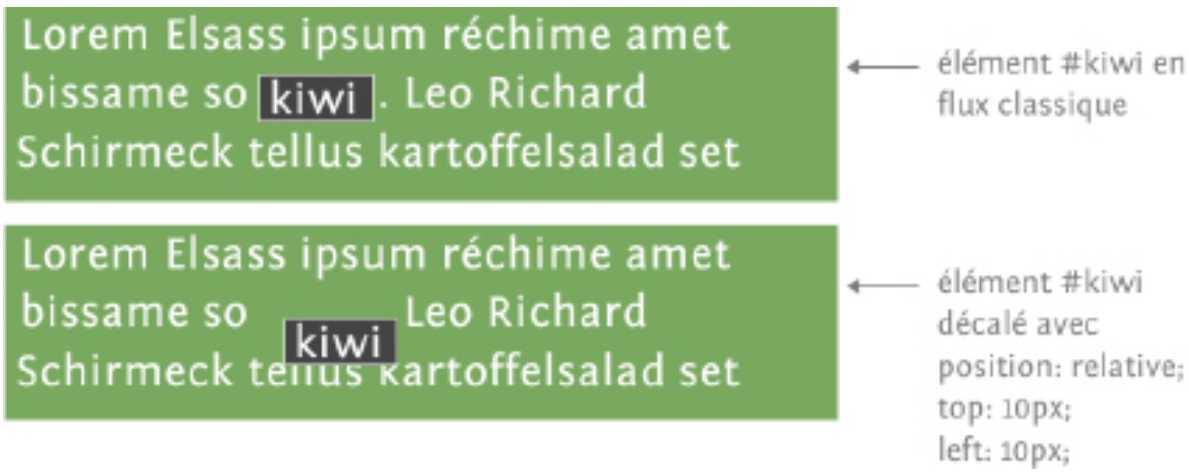
- L'élément auquel la déclaration `position:fixed` est appliquée :
  1. sort du flux
  2. se positionne par rapport à la partie visible de la page et reste fixé au défilement
  3. devient dimensionnable, même avec une déclaration `display:inline`
- Exemple (à tester après la séance)



Hors flux : positionnement relatif

La déclaration `position:relative;`

- L'élément auquel la déclaration `position:relative` est appliquée :
  1. est décalé à partir de sa position initiale dans le flux
  2. laisse un *fantôme* dans le flux
     - L'espace initial est laissé vide

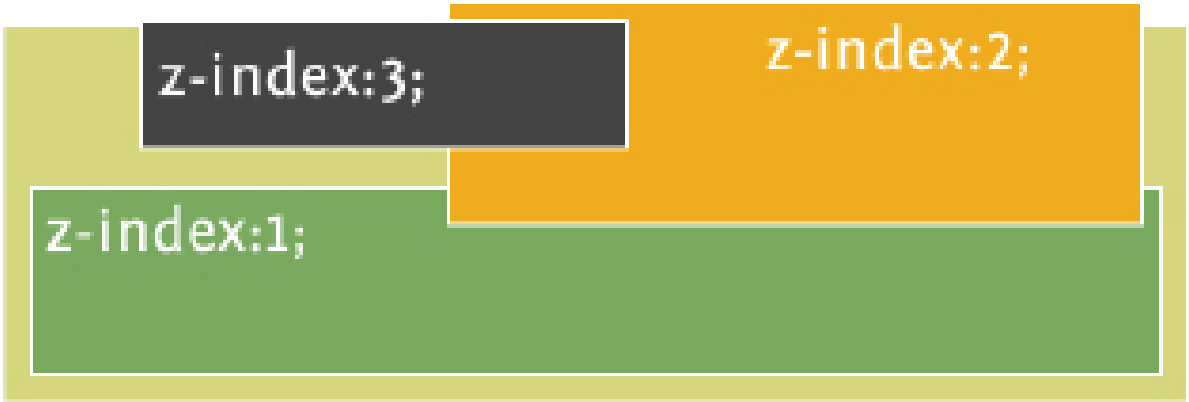


Hors flux : gestion de la profondeur

La propriété `z-index`

Une profondeur peut être spécifiée avec `z-index`

- à tout élément *positionné* (fixe, relatif ou absolu)
- par une valeur numérique qui donne un poids
  - Le poids le plus fort est au dessus



Hors flux : positionnement flottant

La propriété `float`

- Déclaration : `float: left;` ou `float: right;`
- Conçu au départ pour gérer des images et des textes
- Usage détourné pour positionner des blocs
  - Très utilisé (trop !)
  - Très complexe et générateur d'erreurs
  - Prise en charge différente selon les navigateurs

Hors flux : positionnement flottant

Usage *normal* des flottants

Un élément flottant

- est ôté du flux
- est placé à l'extrémité gauche ou droite de son conteneur
- reste à sa hauteur de ligne initiale

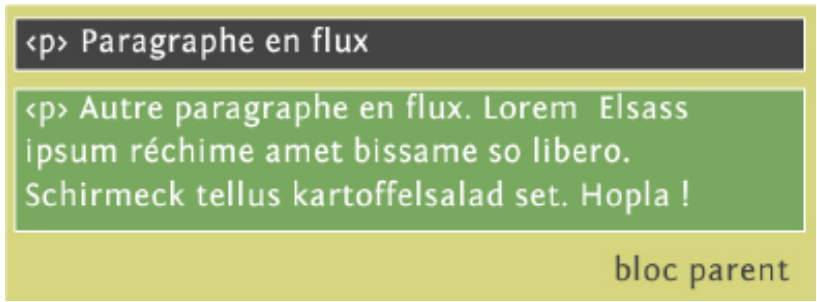


FIGURE : Sans `float`

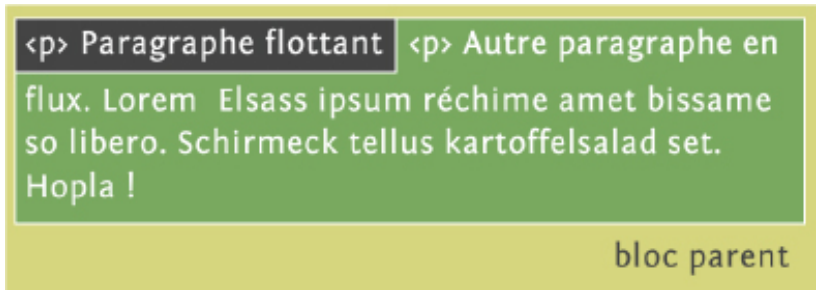


FIGURE : Avec `float`

Exemple : positionnement de blocs côte-à-côte

Objectif : rester dans le flux

Historiquement

- seuls les éléments de type block sont dimensionnables
- seuls les éléments de type inline se placent les uns à côté des autres
- obligation de sortir du flux pour cumuler les deux avantages

Depuis le 19 mars 2009 (IE 8)

- le support de CSS 2.1 par tous les navigateurs permet de rester dans le flux

Exemple à proscrire en 2016

`float` : une technique “Hors flux” complexe

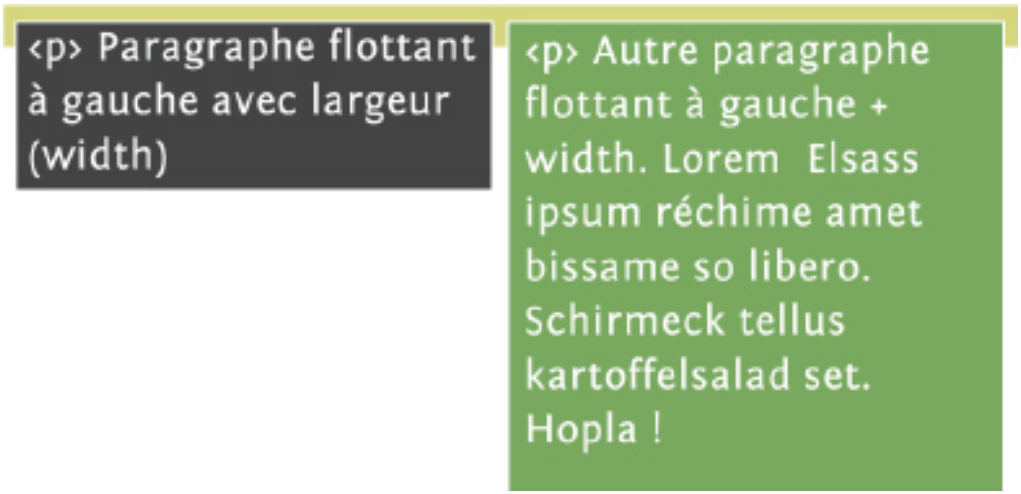


FIGURE : Mauvaise utilisation de `float` pour le positionnement

- Attention : les éléments sont sortis du flux, le conteneur n’occupe plus aucun espace
- Utilisation des propriétés `width` et `clear` pour éviter les débordements

Positionnement moderne de blocs côte-à-côte

`inline-block` : une technique simple en CSS 2.1

Pour positionner 2 élements ou plus :

- côte à côte (comme des élements en ligne)
- dimensionnables (comme des blocs)
- tout en restant dans le flux

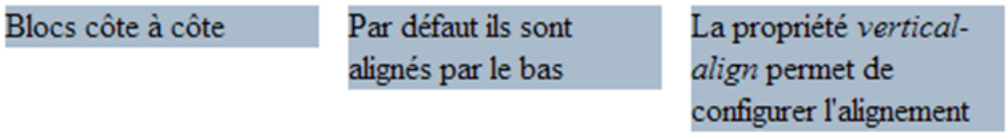
3 propriétés doivent être définies pour chaque élément

display doit prendre la valeur `inline-block`

vertical-align doit prendre la valeur `text-top`

width doit prendre une valeur de sorte que l'espace occupé ne dépasse pas 100% du parent

Exemple d'application




```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>positionnement horizontal</title>
                <style>
                p{display: inline-block;
                width: 150px;
                margin: 0 10px 0 0;
                background-color: #abc;
                vertical-align:text-top;}
                </style>
        </head>
        <body>
                <p> Blocs côte à côte</p>
                <p>Par défaut ils sont alignés par le bas</p>
                <p>La propriété <em>vertical-align</em> permet de configurer l'alignement</em> </p>
        </body>
</html>
```

FIGURE : Positionnement moderne d'éléments côte-à-côte

Blocs côte-à-côte : à retenir

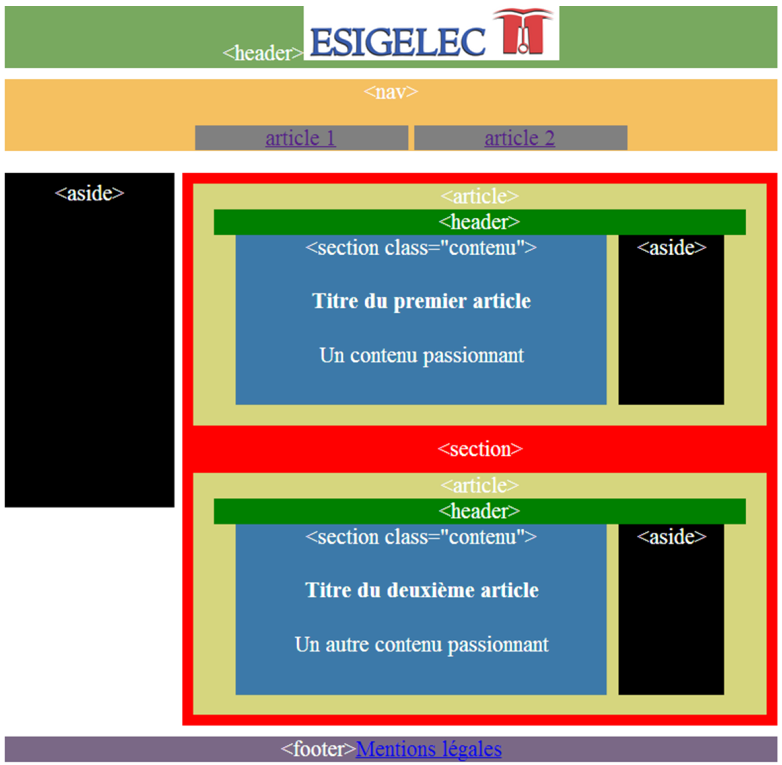
- Le positionnement CSS est la partie la plus compliquée d'un projet
- Les navigateurs récents supportent CSS 2.1
  - Utiliser les techniques modernes permet de s'affranchir des bidouilles communément utilisées
- Attention à vos références ! !
  - Utilisez les validateurs

Plan

Exercice d'application

Consignes

- Reprendre le fichier "Exemple de structure HTML 5"
- Ajouter les styles pour reproduire le résultat suivant sans toucher au HTML



Liens utiles

- W3C CSS Validation Service
  - jigsaw.w3.org/css-validator
- CSS Developper guide
  - developer.mozilla.org/en-US/docs/Web/Guide/CSS
- Communauté francophone d'apprentissage des standards du Web
  - `www.alsacreations.com`
- La beauté du design en CSS
  - `www.csszengarden.com`
- Bootstrap Framework
  - getbootstrap.com