Feuilles de style CSS

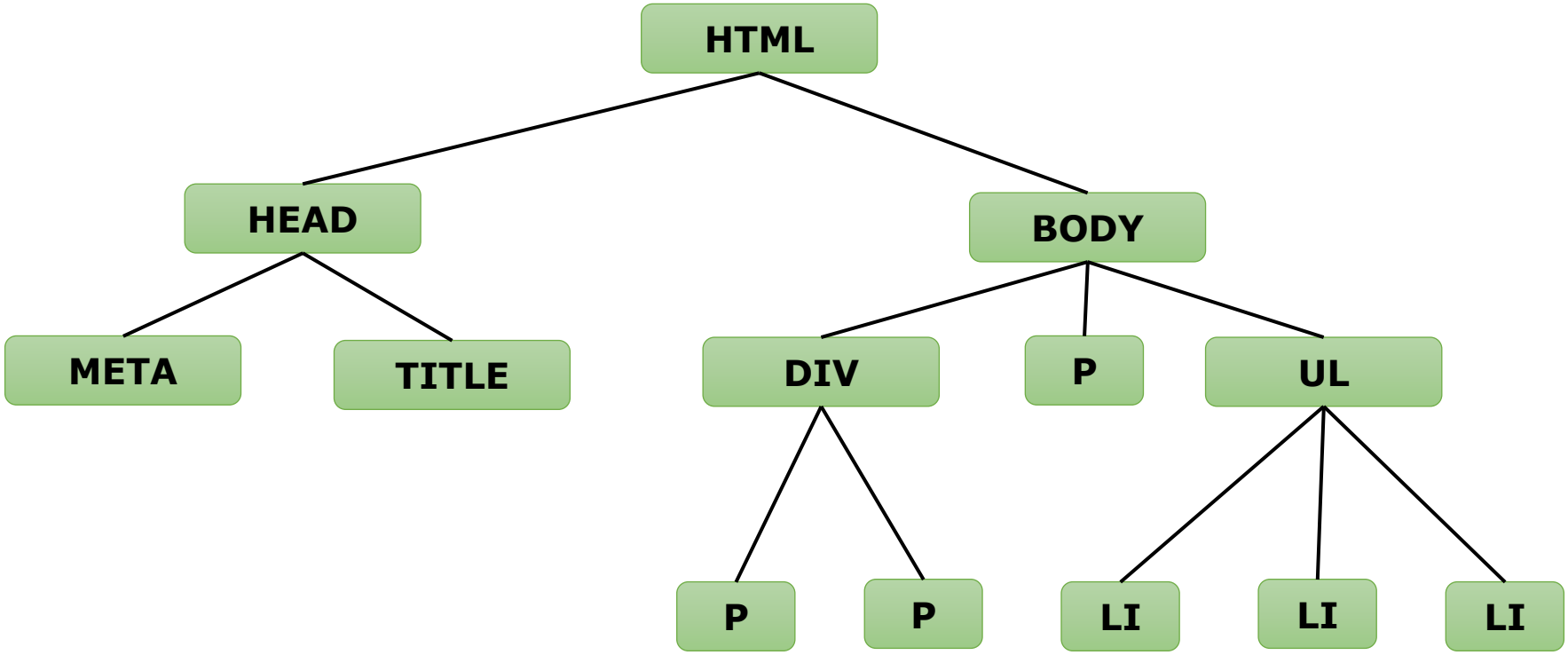
CSS (Cascade Style Sheets)

Prof. M. LAROUSSI

Généalogie des éléments HTML

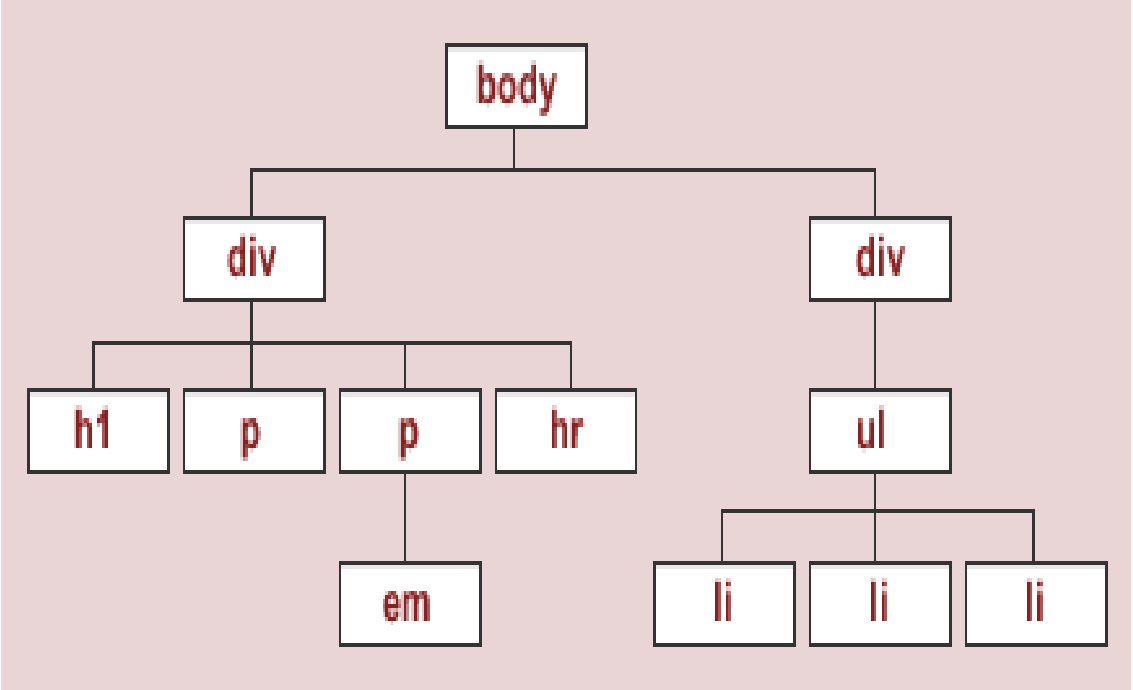
- Les différents éléments composant un fichier HTML (titres, paragraphes, listes, liens, images…) s'organisent sous la forme d'un « arbre généalogique », que l'on nomme également "arbre du document" **ou DOM pour *Document Object Model***.

- Cet arbre généalogique est composé de ***fratries*** et de degrés de parenté qui sont exploités par les sélecteurs CSS pour cibler les différents éléments du document.

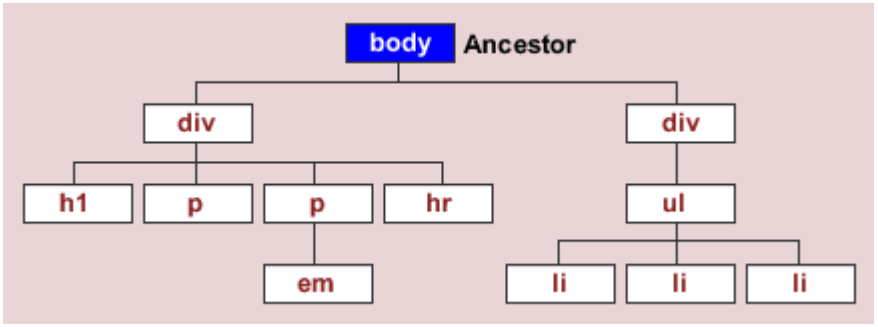


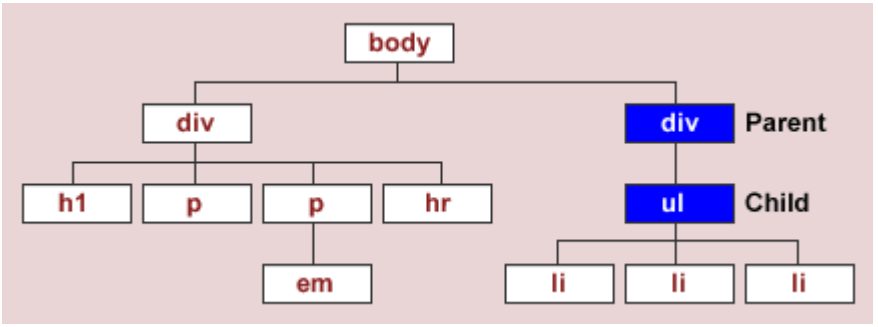
Ancêtre, parent, frère

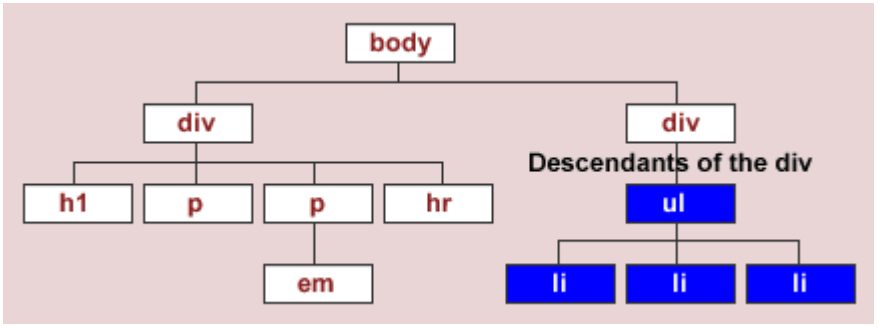
```
<body>
<div id="contenu">
<h1>Premier titre</h1>
<p>Actualités du jour</p>
<p>Echos politiques<p>
<hr>
</div>
<div id="nav">
<ul>
        <li>item 1</li>
        <li>item 2</li>
        <li>item 3</li>
</ul>
</div>
</body>
```

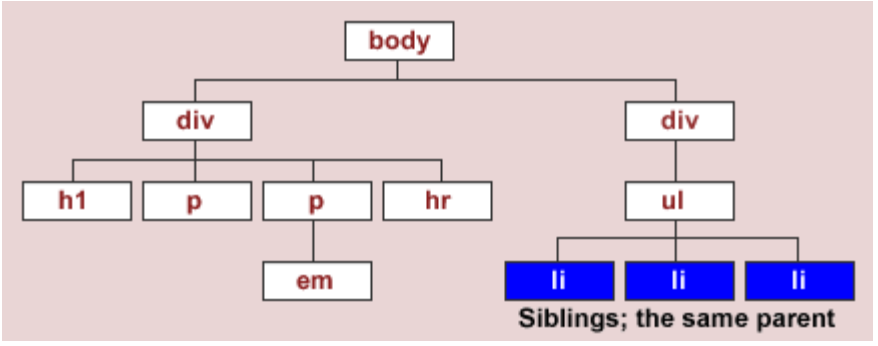


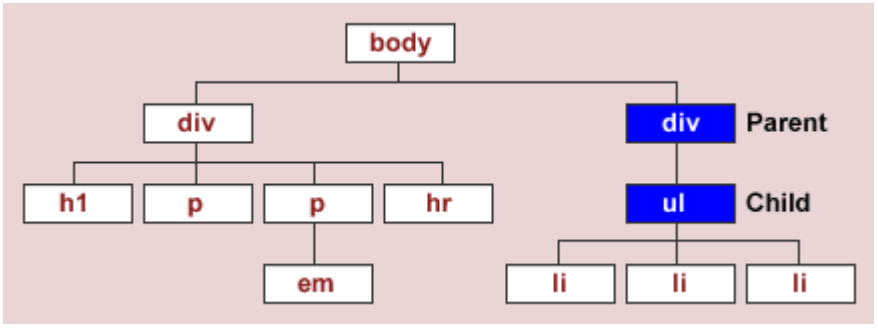
Généalogie des éléments HTML











Pseudo-classe et pseudo-élément

Dans l'optique de mettre en forme des éléments selon un état contextuel (survol, premier enfant, première ligne…), qui ne serait pas défini dans l'arbre du document, ***CSS introduit les concepts de pseudo-classe et de pseudo-élément***.

Exemples Pseudo-classes et pseudo-éléments

Parmi ***les*** **pseudo-classes** les plus couramment usitées :

- Différents états des liens hypertextes :
  (:link, :visited, :hover, :focus, :active)

- Le premier enfant d'un élément **(:first-child)**

Du côté des **pseudo-éléments** :

- La première lettre d'un bloc **(::first-letter)**
- La première ligne **(::first-line)**
- Le contenu créé avant **(::before)**
- Le contenu créé après **(::after)**

N.B - La liste des pseudo-classes : https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes

Pseudo-classe et pseudo-élément

Le sélecteur ***:nth-child*** ("n-ième enfant") permet de sélectionner un ou plusieurs enfants d'un parent

```
:nth-child(number) {
  css declarations;
}
```

tr:nth-child(3n+1)
Permettra de cibler les lignes 1, 4, 7, …

tr:nth-child(odd)
Permettra de cibler les lignes impaires : 1, 3, 5, …

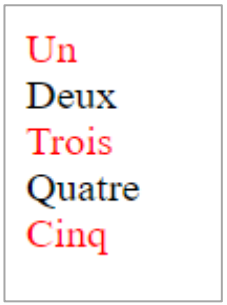
tr:nth-child(4n)
Permettra de cibler les lignes : 4, 8, …

tr:nth-child(even)
Permettra de cibler les lignes paires d'un tableau : 2, 4, 6, …

Pseudo-classe et pseudo-élément

```
//HTML
<section class="grid">
  <article class="module">Un</article>
  <article class="module">Deux</article>
  <article class="module">Trois</article>
  <article class="module">Quatre</article>
  <article class="module">Cinq</article>
</section>
```

```
//CSS
.module:nth-child(2n+1) {
  color: red;
}
Cela impactera toutes les lignes
impaires
```




```
/*cible le deuxième li de chaque ul*/
    ul li:nth-child(2){ ... }

    /*cible le 2e, le 4e, le 6e... li de chaque ul*/
    ul li:nth-child(2n+2){ ... }
```

Pseudo-classe et pseudo-élément

```
a:link { color: blue; }                 /* liens non visités */

a:visited { color: purple; }            /* liens visités */

a:hover { background: yellow; }         /* survol du pointeur */

a:active { color: green; }              /* liens actifs */
```

```
/* Cible n'importe quel élément <p> qui est */
/* le premier fils de son élément parent */

    p:first-child {
      color: white;
      background-color: black;
    }
```

Mode de déclaration

Plusieurs méthodes existent pour utiliser les CSS.

Style en ligne (inline)

on applique le style directement au sein de la balise HTML, via l'attribut : **style=**
→ ***Ce mode est prioritaire sur tous les autres.***

```
<p style="text-align:center; color:red"> Lycée LDV </p>
```

Entête du document (interne)

Balise HTML **<style>** que l'on place au sein de l'élément **<head>** de la page HTML.
→ Ce mode est prioritaire sur les feuilles de styles externes.

```
<head>
    <style type="text/css" media="screen">
    ....
    </style>
</head>
```

Mode de déclaration

Feuille de styles externe

La mise en forme est complètement externalisée dans un fichier de styles CSS lié à l'aide de :

- `<link>`
- `@import`

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

```
<link rel="stylesheet" href="habillage.css" type="text/css" media="screen" />
<link rel="stylesheet" href="texte.css" type="text/css" media="screen" />
<link rel="stylesheet" href="impression.css" type="text/css" media="print" />
```

```
<style type="text/css">

        @import url(styles/habillage.css);
        @import url(styles/texte.css);

</style>
```

```
@import "common.css" screen;

@import url('landscape.css') screen and (orientation:landscape);
```

`@import` est une propriété CSS2 qui doit être suivie de l'URL d'un fichier qui contiendra des styles à appliquer en plus de la feuille de style en cours.

On peut utiliser @import :

- entre les balises <style> et <style> dans la section <head> d'une page HTML;
- au début d'un fichier CSS, pour inclure une ou plusieurs autres feuilles de styles;

Cette deuxième possibilité est intéressante car elle permet de créer des feuilles de styles plus évolutives (on lie un seul fichier depuis la page HTML, et on gère l'import des feuilles de styles directement depuis ce fichier CSS racine).

Problème : ce fonctionnement peut légèrement ralentir le chargement des styles.

Les combinateurs et règles CSS

Sélecteur	*Signification*	*Exemples*
*	Tout élément.	`* { margin: auto; }`
E	Tout élément E (un élément de type E).	`a { color: red; }` `ul { margin-left: 0; }`

E F	Tout élément F enfant d'un parent E (Diret ou non)	`<div id="container">`
		`  <ul>`
		`  <li> List Item`
		`  <ul>`
		`  <li> Child </li>`
		`  </ul>`
		`  </li>`
		`  <li> List Item </li>`
		`  <li> List Item </li>`
		`  <li> List Item </li>`
		`  </ul>`
		`</div>`
		
		`//css`
		`ul li {`
		`  color: red;`
		`}`

Sélecteur	*Signification*	*Exemples*

E > F	Tout élément F enfant direct d'un élément E	`<div id="container">`
		`  <ul>`
		`  <li> List Item`
		`  <ul>`
		`  <li> Child </li>`
		`  </ul>`
		`  </li>`
		`  <li> List Item </li>`
		`  <li> List Item </li>`
		`  <li> List Item </li>`
		`  </ul>`
		`</div>`
		
		`//CSS`
		`div#container > ul {`
		`border: 1px solid black;`
		`}`

Les combinateurs et règles CSS

Sélecteur	Signification	Exemples

`E + F`	Elément F immédiatement précédé par un élément E.
	E et F ont le même parent.
	Permet de cibler le premier frère d'un élément E.
		`ul + p {`
		`  color: blue;`
		`}`
		
		`//HTML`
		`<div>`
		`  <ul>`
		`  <li>Test1</li>`
		`  <li>Test2</li>`
		`  </ul>`
		`  <p>Test3</p> //seul concerné`
		`  <p>Test4</p>`
		`</div>`

`E ~ F`	Tous les frères de E seront concernés	`li:hover ~ li { opacity: 0.4; }`
`E[test]`	Tout élément E avec l'attribut *test* (peu importe la valeur).	`<p align="???">… </p>`
`E[test="val"]`	Tout élément E dont l'attribut *test* à la valeur "*val*".	`<p align="center">… </p>`
`E[lang|="en"]`	Tout élément E dont l'attribut *lang* est une liste de valeurs séparées par des tirets, celle-ci commençant par *val*.	`<div lang="en-fr-ch">… </div>`

Combinateurs et règles CSS – Pseudo-Classe

Sélecteur	*Signification*	*Exemples*
`E:first-child`	Premier enfant de d'un parent E	`ul li:first-child { border-top: none; }`
`E:last-child`	Dernier élément d'un parent	`ul > li:last-child { color: green; }`
`E:link` `E:visited`	L'élément E, ancre source d'un hyperlien, lorsque la cible n'est pas visitée (:link) ou déjà visitée (:visited)	`a:link { color: red; }` `a:visted { color: purple; }`

`E:active`
`E:hover`
`E:focus`
	L'élément E durant certaines actions de l'utilisateur :
	est actif, est survolé, a le focus.
		Style sur les liens hypertextes

`E.`*`val`*	Cible tous les éléments "val" de la classe E	`li.spacious { margin: 2em; }`
`E#`*`mon_id`*	Cible tout élément E dont l'ID est *mon_id*.	

Combinateurs et règles CSS – Pseudo-éléments

Sélecteur	Signification	Exemples

`::first-line`	Première ligne d'un paragraphe	`p:first-line {`
		` text-transform: uppercase;`
		`}`

`::first-letter`	Première lettre d'un élément	`<p class="intro">`
		`  Premier Paragraphe`
		`</p>`
		
		`//CSS`
		`p.intro:first-letter {`
		`  font-size:4em;`
		`  font-weight:bold;`
		`  color: #f00;`
		`  float:left;`
		`}`

`::after`
`::before`
	Cible un contenu généré avant ou après un élément donné	`.ruban::after {`
		`content: "Voyez cette boîte orange.";`
		`background-color: #FFBA10;`
		`border-color: black;`
		`border-style: dotted; }`

`::selection`	Applique la règle de style à la sélection du texte de l'élément faite par l'utilisateur.	`p::selection {`
		` background:#006644`
		`}`

Poids des sélecteurs

!important;

La déclaration ***!important*** est un mot-clé que vous pouvez ajouter à la fin d'une déclaration CSS pour donner plus de poids à cette valeur.

Quand la déclaration ***!important*** est utilisée sur une paire spécifique de propriété/valeur, cette valeur-là échappera à la cascade et deviendra, comme son nom le suggère, la valeur la plus importante pour cette propriété, surclassant toutes les autres valeurs.

```
 .exemple1{
    color: red  !important;
}
   #exemple2{
    color: blue;
}

<p id="exemple2"
class="exemple1">
   Premier paragraphe </p>
```

Résultat obtenu

Premier paragraphe

Priorité des sélecteurs – Notion d'héritage

Il est reconnu que lorsque deux règles CSS différentes ciblent le même élément, c'est la dernière déclarée (la plus basse dans le code source) qui s'applique et qui écrase la précédente.

En cas de conflit entre les règles, une notion du poids de la règle sera appliquée :

1. **Poids "a"** :
   Règle CSS déclarée à l'aide de l'option !important.

2. **Poids "b"** :
   Règle CSS déclarée à l'aide de l'attribut HTML **style=** au sein de l'élément.

3. **Poids "c"** :
   Sélecteur d'identifiant (**id**).

4. **Poids "d"** :
   Sélecteur de classe, d'attribut (**[]**) ou de pseudo-classe **(:hover, :focus, :first-child).**

5. **Poids "e"** : sélecteur d'élément **(p, div, a,…)**
   ou de pseudo-élément **(:first-letter, :before, :after, :first-line).**

Priorité des sélecteurs – Poids d'un sélecteur

```
Body#contact .title h1 {
  font-size: 1.5em;
  font-weight: bold;
  text-aign:right;
}
```

!important	Attribut Style	id	class	élément
0	0	1	1	2

Poids des sélecteurs

Un exemple d'utilisation des poids pour sélectionner la bonne règle CSS :

Déclaration CSS	*!important*	*nombre id (#)*	*nombre de classes (.)*	*nombre de balises*	*poids*
`#menu`	**0**	**1**	**0**	**0**	**0100**
`p.interligne`	**0**	**0**	**1**	**1**	**0011**
`.interligne.espace`	**0**	**0**	**2**	**1**	**0021**
`.partie1 .bloc.info .class-txt span.class-span`	**0**	**0**	**5**	**1**	**0051**
`div (avec !important)`	**1**	**0**	**0**	**1**	**1001**

Poids des sélecteurs

Partie HTML

```
<div id="warning">
<a class="error" href="url">source de l'erreur</a>
</div>
```

Supposons que les règles CSS appliquées soient les suivantes.

Partie CSS

```
a {color: green;}          /* 1 sélecteur d'élément (poids « d ») */
#warning a {color: blue;}  /* sélecteur d'identifiant (poids « b ») + 1 sélecteur de ➥ poids « d » */
a.error {color: red;}      /* 1 sélecteur de classe (poids « c ») + 1 sélecteur de poids ➥ « d » */
```

La couleur du lien indiquant une erreur sera bleue car la règle appliquée est :
`#warning a {color:blue;}`

Bien que la règle `a.error {color: red;}` lui succède dans l'ordre du code CSS, le sélecteur d'identifiant demeure prioritaire.

Si je souhaite que la couleur du lien soit rouge, je dois ajouter un poids au moins équivalent à celui du sélecteur `a.error`.

Concrètement, il faut utiliser : `#warning a.error`.

```
/* Le fichier exemple.css */
h2 + p {
   margin-top: 0.8em
}
h2 + h3 {
   margin-top: 0.2em
}
body {
  background-image: url('w3aspnet.gif');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: center;
}
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<link rel="stylesheet" href="exemple.css" type="text/css">
</head>
<body>

<h2> La propriété background-position </h2>
<p>L'image de background sera centrée</p>
<h3>Utilisation des images en background</h3>

</body>
</html>
```

Applications

```
.interligne {
    line-height: 2em;
    width: 10em;
    background-color: green;
    text-align: justify;
}


/* Notation détaillée */

p {
    border-width: 3px;
    /* border-style: outset; */
    border-color: #134d00;
    border-radius: 3px;
    border-image: url(bordure1.jpg)
30 round;
}
```

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}


#borderimg1 {
  border: 15px solid transparent;
  padding: 15px;
  border-image: url(bordure1.jpg) 30
round;
}
```