
HTTP et HTML

(avec CSS , scss)

Table des matières

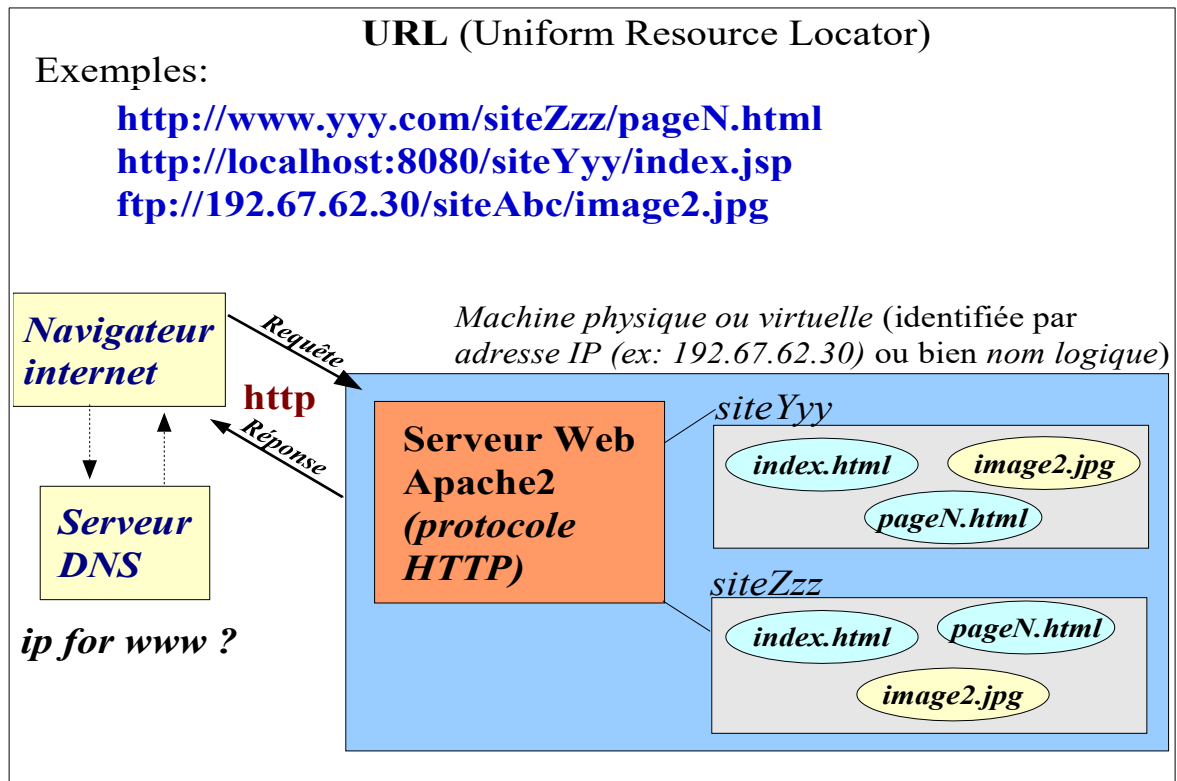
I - URL , HTTP, HTML (présentation).....	3
1. URL et Protocole HTTP.....	3
II - HTML (bases essentielles).....	6
1. Pages HTML.....	6
Description du langage HTML:.....	7
2. Architecture d'une page HTML:.....	7
3. Paramétrage de la page.....	7
4. Eléments de bases.....	8
5. Insertion d'images.....	9
6. Tableaux et bordures.....	10
7. Les liens hypertextes:.....	12
8. Les puces (listes d'éléments).....	13

III - Formulaire HTML.....	14
1. Formulaires HTML.....	14
IV - Principaux apports de HTML5.....	17
1. Versions officielles et interprétations effectives.....	17
2. Grandes lignes de l'évolution HTML4 vers HTML5.....	17
3. Placeholder , autofocus ,	17
4. Mise en page html5 (sémantique + styles css).....	18
5. Apports et précisions au niveau des formulaires.....	21
6. API "Canvas" pour les images vectorielles.....	23
7. Multimédia (audio/vidéo).....	24
V - Styles CSS.....	26
1. Feuilles de styles (CSS, ...).....	26
2. CSS (structures et principes de base).....	27
3. Sélecteurs.....	30
4. Principaux attributs de CSS.....	33
VI - Compléments CSS (fontes , transition, ...).....	38
1. Technologies css (vue d'ensemble).....	38
2. CSS (compléments).....	39
3. CSS (transitions et animations).....	45
4. CSS responsive (media-query , flex, ...).....	50
5. Sass (.scss) (pré-processeur css).....	54

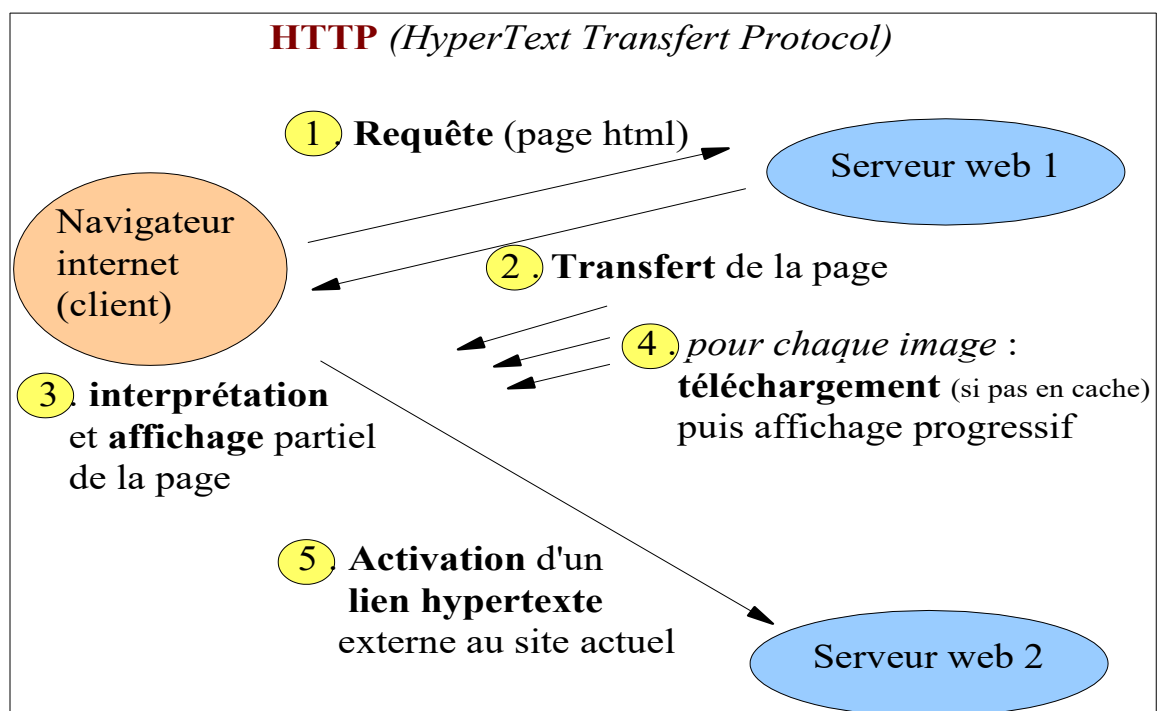
I - URL , HTTP, HTML (présentation)

1. URL et Protocole HTTP

1.1. URL (Uniform Resource Locator)



1.2. Présentation de HTTP



HTTP (*HyperText Transfert Protocol*)**HTTP est un protocole sans état**

(pas de session longue ,
connexion ,
requêtes/réponses ,
déconnexion (c'est tout))

Requête http

```
GET /page2.htm HTTP/1.0
If-Modified-Since: Monday, 29-Jan-96 15:56:20 GMT
User-Agent: .....
Accept: image/gif, image/jpeg, */*
[CRLF]
```

Les types **MIME** (Multipurpose Internet Mail Extension)
 permettent d'identifier les formats des données véhiculées :

Type MIME	extension	Contenu / interprétation
text/html	.html , .htm	page html
image/png	.png	image "png"
image/jpeg	.jpg , .jpeg	image "jpeg"
...	.zip	archive à télécharger

Réponse http

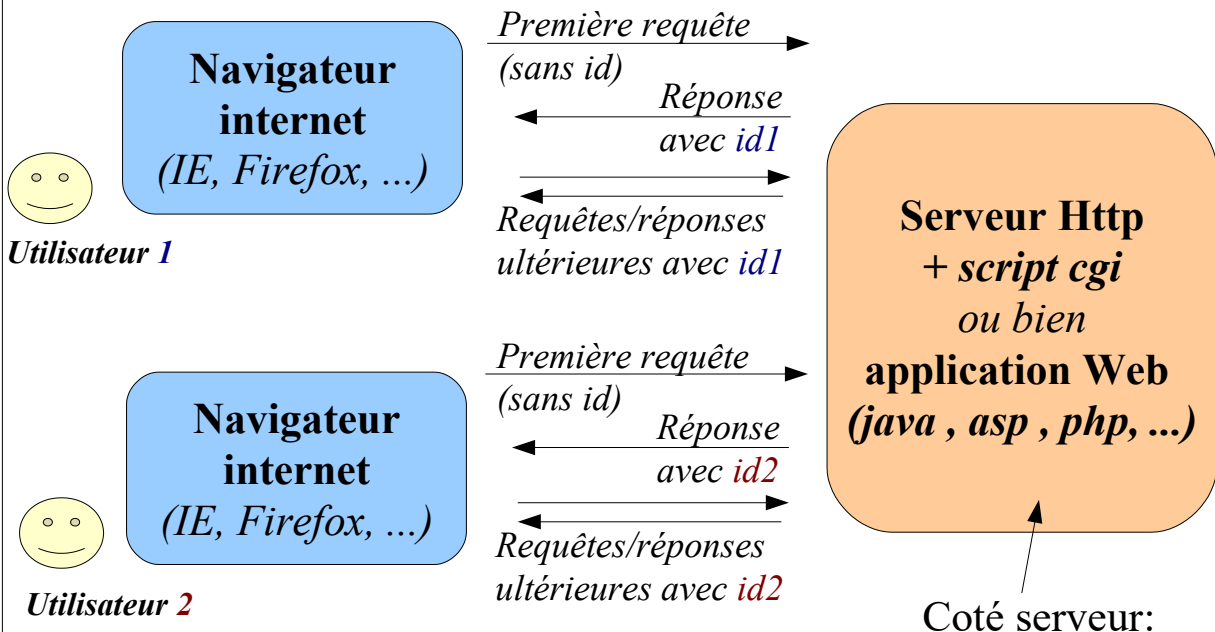
```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 226
[CRLF]
<HTML><HEAD> ... </HEAD><BODY> ... </BODY></HTML>
```

HTTP (protocole de + haut niveau)

Gère des transferts de fichiers
 avec des liens hypertextes

TCP/IP (protocoles réseaux de bas niveau)

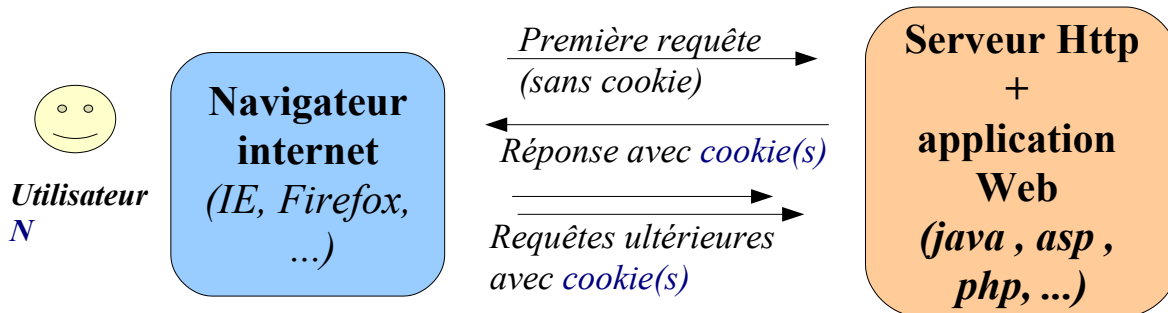
Gère les communications
 réseau à travers la toile internet
 (www=world wide web)

HTTP (*lien à moyen/long terme avec l'utilisateur*)

Techniques pour véhiculer les "id"
 des sessions:

- * *cookies http*
- * *infos en fin d'URL ou champs cachés*

Génération de nouveaux id
 et *associations*
(id_1 , données_session_utilisateur_1)
(id_2 , données_session_utilisateur_2)

Cookies HTTP (*id session ou préférences de l'utilisateur*)

Un *cookie Http* est une *information* de type

(nom, valeur, éventuelle_date_expiration, url_site_web)

qui est :

- * *générée dynamiquement coté serveur (selon code ou ...)*
- * *automatiquement stockée coté navigateur en mémoire et dans un fichier si une date d'expiration a été précisée.*
- * *systématiquement/automatiquement renvoyée au serveur à chaque émission d'une nouvelle requête http vers le site web impacté*

Exemples:

sport_préféré=football,expires=2010,...

jSessionId=A2B3C567C45677B3445A445

HTTP (modes "**GET**" et "**POST**")

Page html dans navigateur

```
<html> <head>...</head>
```

```
<body> ...
```

```
<form action="s1"
  method="GET"
  ou "POST" >
```

Prenom :

Nom:

Age :

```
</form>
```

```
</body>
```

```
</html>
```

Requête
HTTP

avec
valeurs saisies

Serveur Http
+
application Web
(java , asp , php, ...)

prenom=alain&nom=therieur&age=40

en fin d'URL (après ?)

en mode "GET" :

(ex: "http://.../s1?prenom=alain&...=...")

ou bien

Dans la partie "corps interne" de la requête HTTP en mode "POST"

Mode "GET" ---> dans historique (peu confidentiel) , utilisable dans lien hypertexte

Mode "POST" --> pas de limite de taille dans les données saisies

II - HTML (bases essentielles)

1. Pages HTML

HTML (HyperText Markup Language)

Page html interprétée (pour affichage)
dans un navigateur internet

```
<html>

<head>
  <title>titrePage</title> ...
</head>

<body>
  <b>texte_en_gras (bold)</b>
  <i>texte_en_italique</i>
  <u>texte souligné <underline></u>
  <img src='image1.jpg' />
  <a href='page2.html'> vers page 2 </a>
</body>

</html>
```

Entête invisible

Partie visible de la page

Mise en forme du texte encadré (balisé)

Insertion d'image

Lien hypertexte vers une autre page

HTML (formulaires et tableaux)

Page html dans navigateur

```
<html> <head>...</head>
<body> ...
  <form action="s1"
    method="GET" ou "POST" >
    couleur : <input type="text" value="rouge" />
    Nom: <input type="text" value="Therieur" />
    <input type="submit" value="soumettre" />
  </form>
  <table border='2' >
    <tr><th>années</th><th>valeurs</th></tr>
    <tr><td>2008</td><td>100</td></tr>
    <tr><td>2009</td><td>120</td></tr>
  </table>
</body>
</html>
```

Formulaire de saisie

Liste déroulante
 <select name="couleur">
 <option>rouge</option>
 <option>bleu</option>
 </select>

Zone de saisie
 <input type="text" .../>

Bouton poussoir
 <input type="submit" .../>

tableau

années	valeurs
2008	100
2009	120

Description du langage HTML:

Le langage **HTML** (*HyperText Markup Language*) est un langage à balise, c'est-à-dire que pour faire une action sur un groupe de mots, vous devez baliser ce groupe de mots. Par exemple, pour mettre Paris en gras, il suffit d'entourer ce mot par cette séquence d'instruction :

`Paris` donne : **Paris**

D'une façon générale, les commandes HTML sont de la forme :

- `<marqueur>Bonjour</marqueur>`
- `<marqueur attribut=valeur attribut2="valeur avec blancs">Bonjour</marqueur>`
- `<marqueurSansFin/>`

2. Architecture d'une page HTML:

Syntaxe minimale d'une page HTML:

<code><html></code>	--Marque début document
<code><head></code>	--début de l'entête
<code><title>Titre de ma page</title></code>	--Titre du document (onglet)
<code></head></code>	--Fin de l'entête
<code><body></code>	--Début corps du document
Description de ma page	--Description page
<code></body></code>	--Fin du corps
<code></html></code>	--Fin du document

3. Paramétrage de la page

Le paramètre d'une page HTML permet de déterminer le titre, une image ou une couleur de fond, un fond sonore ainsi que la couleur des liens.

Syntaxe:

`<body></body>` délimite le corps du document

`<body [bgcolor= "couleur de fond"] [link="couleur"] [vlink="couleur"] [alink="couleur"]`

`[background= "image de fond"]>` ➔ link, vlink, alink liens non visités, visités, actifs.

`<title></title>` indique le titre du document (qui apparaît dans la barre de titre ou l'onglet du navigateur)

Exemple:

```
<html>

<head>
<title>NouvellePage1</title>
<bgsound src="chord.wav" loop="-1">
</head>

<body background="WB00756_.gif" link="red" vlink="yellow" alink="blue">
....
</body>

</html>
```

Exemple2:

```
<body bgcolor="white" text="black">
texte en noir sur fond blanc
</body>
```

4. Éléments de bases

Niveaux de titre => h1 (le plus gros) vers h6 (le plus petit)

exemple: `<h3>` Titre du paragraphe `</h3>`

`<p>` suite du paragraphe `</p>`

Saut de ligne (dans un même paragraphe):

ligne1`
`

ligne2`
`

ligne3

Filet (trait de séparation) horizontal:

`<hr/>`

`<hr size="5" align="left" wight="50%" />`

Gras, Italique, Souligné:

`` gras physique ou normal (bold) ``

`` gras logique (selon capacités du navigateur) ``

`<i>` Italique `</i>`

`` Italique aussi (Enhanced Mode) ``

`<u>` Souligné (UnderLine) `</u>`

Fonte et couleur:

` texte rouge de taille 3 `

NB: code couleur = #rrggbb avec rr , gg , bb = composantes rouge, verte et bleue (00 à FF)

5. Insertion d'images

Pour insérer une image dans une page HTML, il faut utilisé le tag ou marqueur `` dont les attributs sont tous facultatifs sauf `src`:

```
<img src= "nom de l'image" [align= top | bottom | middle | left | right ]  
[ border = Epaisseur] [alt= "texte"] [width= largeur] [height= hauteur]>
```

- **src** indique le nom du fichier source de l'image.
- 1. **align** précise comment l'image doit être alignée verticalement et horizontalement par rapport à son environnement (texte à côté). Il peut prendre les valeurs : **left, right, top, bottom, texttop, middle, absmiddle, baseline, absbottom**.
- **alt** indique le texte alternatif qui sera affiché par le navigateur ne sachant pas afficher l'image ou si l'image ne peut être visualisée (également texte de l' "InfoBule" jaune).
- **boder** indique la largeur de la bordure de l'image (par défaut BORDER=0).
- **height** hauteur de l'image en pixels.
- **width** largeur de l'image en pixels.
- **hspace** et **vspace** donnent l'espace à ménager en pixels autour des images.

Exemple de code :

```

```

Actuellement, Les trois principaux formats reconnus pour les images sont JPEG(.jpg) , PNG , GIF et GIF animé.

6. Tableaux et bordures

Dans une page HTML sans tableaux, tout objet est par défaut placé sur la marge gauche de la page (comme présentation originale, il y a mieux !).

Une présentation propre et claire nécessite donc l'utilisation des tableaux pour structurer la page et positionner les objets (images, textes, animations,...) dans des cellules.

NB:

- Les cellules d'un tableau n'ont pas obligatoirement de bordure. On peut donc utiliser des "tableaux invisibles" pour simplement contrôler un alignement sur plusieurs colonnes.
- Inversement un tableau centré d'une seule cellule (1 ligne et 1 colonne) peut servir d'encadrement (Border=6 par exemple).

Syntaxe:

```
< table [border=EpaisseurBordure] [cellpadding=espace] [cellspacing=epaisseur] [width= largeurTotale]
[height=hauteurTotale]>
```

```
....
```

```
</table>
```

Pour chaque ligne du tableau (Table Row):

```
<tr [align=alignement]> ..... </tr>
```

Pour chaque cellule d'une ligne:

```
<td [colspan=nbCol] [rowspan=nbLig] [align=alignHoriz] [valign=alignVert] [width=largeur]>
élément </td>
```

NB: Une cellule ordinaire sera balisée par **td** (Table Data)
Une cellule d'entête sera balisée par **th** (Table Header)

Exemple Simple:

```
<table border="1">
  <tr> <th> FRANCE </th> <th> EXPORT </th> </tr>
  <tr> <td> 1235 </td> <td> 1238 </td> </tr>
  <tr> <td> 1525 </td> <td> 1688 </td> </tr>
</table>
```

- résultat:

FRANCE	EXPORT
1235	1238
1525	1688

Exemple plus complexe:

```

<table width=75% border="4">
  <tr> <th colspan=2 width=50%> FRANCE </th>
    <th colspan=2 width=50%> EXPORT </th>
  </tr>
  <tr> <th> semetrel </th> <th> semestre2 </th>
    <th> semetrel </th> <th> semestre2 </th>
  </tr>
  <tr> <td> 1000 </td> <td> 1600 </td>
    <td> 1200 </td> <td> 1800 </td>
  </tr>
  <tr> <td> 1080 </td> <td> 1700 </td>
    <td> 1300 </td> <td> 1900 </td>
  </tr>
</table>

```

FRANCE		EXPORT	
semetrel	semestre2	semetrel	semestre2
1000	1600	1200	1800
1080	1700	1300	1900

Tableaux rigoureux avec parties **thead** et **tbody**

```

<table border="1" id="tableDevises">
  <thead>
    <tr><th>code</th><th>nom</th><th>change</th></tr>
  </thead>
  <tbody id="bodyTableau">
    <!-- <tr id="tr_USD">
      <td>USD</td><td>Dollar</td><td>1.1</td>
    </tr> -->
  </tbody>
</table>

```

Bien que facultatives , les balises <thead> et <tbody> permettent de bien délimiter les parties "entête" et "corps" d'un tableau . Ceci est très pratique pour supprimer ou bien insérer des lignes de données au bon endroit par programmation javascript .

7. Les liens hypertextes:

Un lien hypertexte est une connexion établie depuis une page Web vers un autre fichier ou site Web. La destination du lien est le plus souvent une autre page Web, mais il peut également s'agir d'un fichier multimédia (image,...) ou même d'un programme (script **cgi** ou **servlet** java ou page **asp** , **jsp** , **php** capable de générer dynamiquement une page Html).

Syntaxe:

Création d'un lienhypertexte sur une autre page du même site:

```
<a href="monAutrePage.html"> vers page2 </a>
```

Création d'un index (en haut d'une page vers des éléments en bas de la page) :

```
<p><a href="#signet_1">aller au chapitre 1</a></p>
<p><a href="#signet_2">aller au chapitre 2</a></p>
```

```
<p> <a name="signet_1">Chapitre 1</a>
.....</p>
```

```
<p> <a name="signet_2">Chapitre 2</a>
.....</p>
```

Aller directement vers un endroit déterminé d'une autre page:

```
<a href="mapage.htm#signet_x"> chapitre x de mapage </a>
```

Lien hypertexte vers une page d'un autre site (chemin absolu):

```
<a href="http://www.societeX.fr/siteA/pageA.html"> siteA </a>
```

Lien hypertexte sous forme d'image:

```
<a href="mapage.html">  </a>
```

8. Les puces (listes d'éléments)

li --> Liste Item

ul --> Unordered List

ol --> Ordered List

Voici quelques exemples codés de listes à puces

```
<p><strong>Paragraphe à puces numérotées :</strong></p>
<ol>
  <li>ligne1</li>
  <li>ligne2</li>
  <li>ligne3</li>
</ol>
```

```
<p><strong>Paragraphe à puces numérotées I :</strong></p>
<ol type="I" start="1">
  <li>ligne1</li>
  <li>ligne2</li>
  <li>ligne3</li>
</ol>
```

```
<p><strong>Paragraphe à puces rondes:</strong></p>
<ul>
  <li>ligne1</li>
  <li>ligne2</li>
  <li>ligne3</li>
</ul>
```

```
<p><strong>Paragraphe à puces carrées:</strong></p>
<ul type="square">
  <li>ligne1</li>
  <li>ligne2</li>
  <li>ligne3</li>
</ul>
```

La copie d'écran ci-contre montre que le paragraphe à puces carrées ne s'affiche pas avec le browser Internet Explorer (versions 3 & 4).

Paragraphe à puces numérotées : 1. ligne1 2. ligne2 3. ligne3	Paragraphe à puces rondes: • ligne1 • ligne2 • ligne3
Paragraphe à puces numérotées I : I. ligne1 II. ligne2 III. ligne3	Paragraphe à puces carrées: • ligne1 • ligne2 • ligne3

III - Formulaire HTML

1. Formulaires HTML

Un formulaire HTML est délimité par les balises `<form>` et `</form>`.

Il n'y a *pas de délimitations visibles à l'écran* mais ces deux balises permettent d'encadrer un ensemble de **champs** dont les **valeurs saisies** seront **envoyées vers** un certain **programme (script)** du serveur.

L'attribut **method** permet de définir la méthode d'envoi des données au serveur (**post** ou **get**)

L'attribut **action** permet d'indiquer l'url d'un code qui sera quelquefois déclenché coté serveur (ex : .asp , .php , .jsp) pour traiter les données saisies.

Exemple simple:

```
<form method="post" action="cgi-bin/scriptA.exe">
  <p>nom: <input type="text" size="20" name="nom"/></p>
  <p>prenom: <input type="text" size="20" name="prenom"/></p>
  <p><input type="submit"/> </p>
</form>
```

- Le script invoqué va recevoir une entrée du type:

nom=therieur&prenom=alex

Le second exemple ci-dessous présente les champs formulaires les plus utilisés:

```
<form method="post" action="cgi-bin/scriptA.exe">
  <p>Donnez votre nom: <input type="text" size="20"
    name="saisie_1"/></p> <!--affiche une zone de saisie-->

  <!-- mise en place d'une case à cocher-->
  <p><input type="checkbox" name="C1"/>case à cocher </p>

  <!-- mise en place de deux boutons "radio" -->
  <p><input type="radio" checked name="R1"
    value="V1"/>bouton radiol1    <br/>

  <input type="radio" name="R1" value="V2"/> bouton radio2</p>

  <!-- mise en place d'une liste déroulante: -->
  <p>Pays: <select name="D1" size="1">
    <option value="fr"> France</option>
    <option value="us"> Etat Unis</option>
    <option value="au"> Autriche</option>
  </select> </p>

  <!--mise en place champ de saisie multiligne -->
  <p>Donnez votre commentaire: </p>
  <p><textarea name="S1" rows="2" cols="20">ligne1
ligne2</textarea> </p>

  <!-- boutons poussoirs -->
```

```
<p> <input type=submit value="Valider"/> <input type=reset/></p>
</form>
```

- résultat à l'écran:

Donnez votre nom:

☐ case à cocher

☒ bouton radio1

☐ bouton radio2

Pays:

Donnez votre commentaire:

NB : Dans le cas d'une page HTML moderne (avec un front-end javascript bien séparé du back-end), les attributs `method="..."` et `action="..."` sont très rarement utilisés et la balise `<submit>` fait souvent place à une balise `<button>` accompagnée de code javascript .

Exemple :

```
<h1>calculatrice</h1>
<form>
  <label>a: </label><input id="x" type="text" /> <br/>
  <label>b: </label><input id="b" type="text" /> <br/>
  <button onclick="calculerOperation('+')">a+b</button>
  <input type="button" onclick="calculerOperation('*')" value="a*b" />
  ...
</form>
<hr/>
<p>res = <span id="spanRes"></span> </p>
```

a: 6

b: 7

a+b a*b a-b a/b

res = 13

Encadrement visuel de certains éléments d'un formulaire

```

<fieldset class="c-fieldset">
  <legend class="c-legend">euroToFranc</legend>
  <label for="idMontantEuro">montant euro:</label> <br/>
  <input type="text" value="15" id="idMontantEuro"><br/>
  ...
</fieldset>

```

Attention : lorsque bootstrap css est utilisé , on a quelquefois besoin de changer les styles affectés par défaut par bootstrap aux éléments de type *fieldset* et *legend* .

```

.c-fieldset {
  padding: 0.5em;
  margin: 0.5em;
  border-width: 2px;
  border-style: solid;
  border-color: black;
}

.c-legend {
  width: inherit; /* Or auto */
  padding: 0 10px; /* To give a bit of padding on the left and right */
  border-bottom: none;
}

```


IV - Principaux apports de HTML5

1. Versions officielles et interprétations effectives

Principales versions officielles de HTML (W3C) :

- **HTML 4** (1997, 1999)
- **XHTML 1.0** (1999, 2000) (HTML vu comme un cas particulier du rigoureux XML)
- **HTML 5** (2014, ...)

Principaux navigateurs actuels :

- **IE 8,9,10,11,... , Edge** (*historique* : traitement particulier des événements "javascript")
- **Firefox** (moteur "gecko/moz" : mozilla)
- **Google Chrome** (moteur "webkit")

2. Grandes lignes de l'évolution HTML4 vers HTML5

- Utilisation grandement conseillée des **styles CSS** pour la mise en forme.
- Utilisation de **balises "sémantiques"** qui structurent logiquement le document (entête , section/article/paragraphe , pied de page) sans imposer un look précis .
- Plus de **précision sur les champs des formulaires** (number ,date , ...)
- API "**canvas**" pour des "images vectorielles" gérées dynamiquement par des instructions "javascript"
- Meilleure prise en charge des contenus "**multimédia**" (vidéo , audio , ...)
- Meilleure gestion du **graphisme** (rotations , coins arrondis , dégradés , ...)

3. Placeholder , autofocus , ...

L'attribut facultatif **placeholder** d'une zone input permet d'indiquer un **message de substitution** (qui s'affiche temporairement en gris dans le champ de saisie tant que l'utilisateur n'a encore rien saisi).

Exemple :

```
<input type = "email" name = "email" placeholder = "email@example.com" />
```

L'attribut facultatif **autofocus** d'une zone input permet de faire en sorte que le champ de saisie concerné récupère automatiquement le focus (curseur clignotant pour saisir des caractères) au chargement de la page.

Exemple :

```
<input type = "text" name = "search" autofocus />
```

4. Mise en page html5 (sémantique + styles css)

HTML5 apporte de nouvelles balises sémantiques permettant de mieux structurer le contenu d'une page. Ces balises ne sont par défaut associées à aucune mise en page très précise. Il faut absolument associer des styles CSS particuliers pour contrôler les dispositions et le look.

Intérêts des balises sémantiques : **interprétation du contenu d'une page HTML par un robot** qui peut ainsi mieux comprendre la signification des différentes parties d'une page web.

<i>balises</i>	<i>significations</i>
section	Sous partie d'une page ou article ou section (pouvant éventuellement comporter "header" ... , "footer")
article	(Sous-contenu) assez indépendant du reste de la page (ex : blog-post) (pouvant éventuellement comporter "header" ... , "footer")
aside	Panneau latéral ou équivalent
h1 , h2 , h3 , ... , h6	Titre d'une (sous-)section quelconque
footer	Pied de page ou de section/article (potentiellement multiple(s))
header	entête de page ou de section/article (potentiellement multiple(s))
nav	barre de navigation avec liens (menu simple ou mode plan ou ...)
address	Adresse (contact) liée à l'auteur de la page ou de l'article englobant

<section> et <article> sont très proches (à choisir en fonction du contexte)

sémantique pure (associée à aucune mise en page) :

<i>balises</i>	<i>significations</i>
<main> ou en attendant <div id="main">	Unique partie principale d'une page (dans <body> , éventuellement à coté de <header> et <footer> ou englobant <header> et <footer> mais jamais à l'intérieur de <footer>/<header>/<section>/<article>)

NB : bien que des styles CSS (2 ou 3) soient très fortement conseillés pour paramétrer la mise en page d'un site web , on peut éventuellement utiliser (à l'ancienne) des tableaux pour contrôler l'alignement des éléments mais ceci doit absolument se faire en précisant la sémantique **role="presentation"** au niveau de la balise <table> pour qu'il n'y ait pas confusion avec un tableau de données .

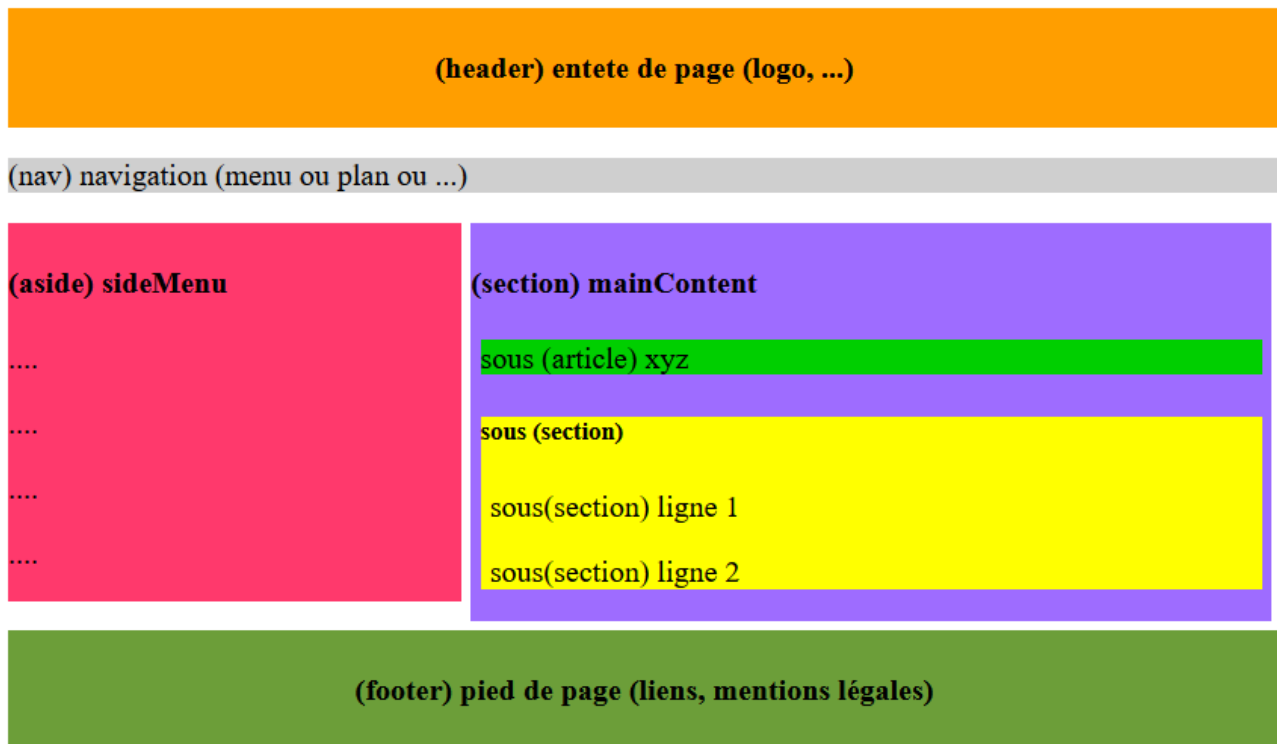
Exemple avec des balises sémantiques (parmi pleins d'autres possibilités) :

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="layout.css"/>
  <title>essai layout html5</title>
</head>
<body>
  <header id="mainHeader" role="banner">
    <h4> (header) entete de page (logo, ...)</h4>
  </header>
  <nav id="mainNav">
    <p>(nav) navigation (menu ou plan ou ...)</p>
  </nav>
  <main>
    <aside id="mainSideMenu">
      <h4> (aside) sideMenu </h4> <p> .... </p> <p> .... </p> <p> .... </p> <p> .... </p>
    </aside>
    <section id="mainContent">
      <h4> (section) mainContent</h4>
      <article>
        <p>sous (article) xyz</p>
      </article>
      <section>
        <h5>sous (section)</h5>
        <section>
          <p>sous(section) ligne 1</p>  <p>sous(section) ligne 2</p>
        </section>
      </section>
    </section>
  </main>
  <footer id="mainFooter" >
    <h4>(footer) pied de page (liens, mentions légales)</h4>
  </footer>
</body>
</html>

```

ce qui donne :



avec les styles css du fichier suivant :


layout.css

```
#mainHeader, #mainSideMenu, #mainContent, #mainFooter { padding:1px 0;}
#mainHeader { background-color:#FF9900; text-align:center; }
#mainNav    { background-color : #cccccc; }
#mainSideMenu { float:left; width:240px; background-color:#FF3366; overflow:auto; }
/* margin-left de #mainContent doit etre un peu plus grand que width de #mainSideMenu flottant */
#mainContent { margin-left:245px; background-color:#9966FF; }
/* clear:both; essentiel pour que #mainFooter soit toujours en dessous du #mainSideMenu flottant
   et jamais sur la meme ligne */
#mainFooter { background-color:#669933; text-align:center; clear:both; }
article { margin:5px; background-color : #00cc00; }
section { margin:5px; background-color : yellow; }
```

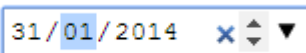
5. Apports et précisions au niveau des formulaires

5.1. Nouveaux types de "input"

Select your favorite color: `<input type="color" name="favcolor">`


Select your favorite color: 

Birthday: `<input type="date" name="bday">`

Birthday: 

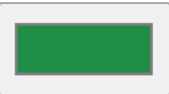
`<input type="email" name="email">` avec message d'erreur automatique s'il manque "@"


Quantity (between 1 and 5): `<input type="number" name="quantity" min="1" max="5">`

Quantity (between 1 and 5):  , plus message d'erreur automatique si non numérique ou hors intervalle , min et max sont facultatifs

Points: 0 `<input type="range" name="points" min="1" max="10">` 10


Points: 0  10

Select your favorite color: 

Birthday: 

Email :

Quantity (between 1 and 5):

Points: 0  10

navigateur favori:

Autres éventuels types de champs de saisies (selon navigateurs) :

- **type="tel"** pour numéro de téléphone
- **type="url"** pour saisir des url (par exemple avec http://)
- `<input type="search" name="googlesearch">` qui se comporte comme type=text .
- Birthday (month and year): `<input type="month" name="bdaymonth">`

5.2. Datalist

Zone combinée (liste + saisie)

```
<input list="browsers" name="browser">
```

```
<datalist id="browsers">
```

```
<option value="Internet Explorer"> <option value="Firefox">
```

```
<option value="Chrome"> <option value="Opera"> <option value="Safari">
```

```
</datalist>
```

5.3. Contrôles de saisies selon standard html5

Principales contraintes de saisies d'HTML5:

- **required** : le champ est requis (valeur obligatoire)
- **minlength="3"** : il faut saisir au moins 3 caractères
- **pattern="^[a-zA-Z].+"** : la valeur saisie doit correspondre à une expression régulière (ici ça doit commencer par un caractère alphabétique puis contenir au moins un autre caractère quelconque)
- ...

Exemple :

```
<form>
  <label>username (commençant par une lettre , au moins 2 caractères):</label>
  <input name="nomQuiVaBien"
    required pattern="^[a-zA-Z].+" /> <br/>

  <label>password (au minimum 3 caractères):</label>
  <input name="motDePasseQuiVaBien"
    type="text" required minlength="3" /> <br/>

  <label>roles (required):</label> <input name="roles" required /> <br/>
  <input type="submit" value="submit" />
</form>
```

username (commençant par une lettre , au moins 2 caractères):

password (au minimum 3 caractères):

roles (required):

! Veuillez respecter le format requis.

username (commençant par une lettre , au moins 2 caractères):

password (au minimum 3 caractères):

roles (required):

! Veuillez allonger ce texte pour qu'il comporte au moins 3 caractères. Il en compte actuellement un seul.

roles (required):

! Veuillez renseigner ce champ.

5.4. output

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
0<input type="range" id="a" value="50">100 +
<input type="number" id="b" value="50">= <output name="x" for="a b" />
</form>
```

0 100 + = 100

5.5. keygen (introduit en html5 puis devenu maintenant obsolete)

<keygen name="security"> génère une clef privée (utilisée coté client) et une clef publique envoyée au serveur .

2048 (haute sécurité) ▼

La clef publique pourra par exemple être utilisée pour générer un certificat client pour authentifier l'utilisateur dans le futur.

6. API "Canvas" pour les images vectorielles

→ voir annexe du support de cours "javascript"

7. Multimédia (audio/vidéo)

```
<h3>BWV565.mp3 (Bach):</h3>
```

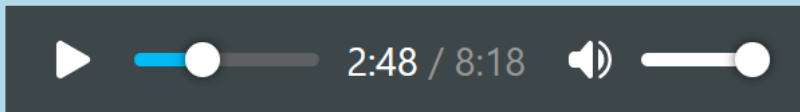
```
<audio controls >
```

```
<source src="audio/BWV565.mp3" />
```

texte affiché seulement par navigateur ne supportant pas la balise audio HTML5

```
</audio>
```

BWV565.mp3 (Bach):



```
<h3>Video via balise video/html5:<h3>
```

```
<video controls width='400' height='300'>
```

```
<source src="video/video2.mp4" />
```

<!-- attention : track (pour fichier sous-titres)

accède bien au fichier .vtt qu'en mode http -->

```
<track default kind="captions" srclang="fr" src="video/video2-captions.vtt"/>
```

texte affiché seulement par navigateur ne supportant pas la balise video HTML5

possibilité d'imbriquer ici balise object (en plan B) pour anciens navigateurs

```
<object width='400' height='400' type="video/mp4"
```

```
data="video/video2.mp4" />
```

```
</video>
```



L'attribut controls permet d'obtenir les contrôles par défaut (start / stop / pause, ...)

et une api javascript existe pour manipuler le contrôle audio/vidéo par code si besoin.

Fichier de sous-titres :

video/video2-captions.vtt

WEBVTT

00:00.200 --> 00:04.000

-premiere partie de la video.

-La voiture grimpe le début de la cote

00:04.200 --> 00:12.000

-deuxieme partie de la video.

-La voiture grimpe la seconde partie

00:12.000 --> 00:15.000

-troisieme partie de la video.

-La voiture est arrivée en haut

WEBVTT signifie **Web Video Text Tracks**

[https://fr.wikipedia.org › wiki › WebVTT](https://fr.wikipedia.org/wiki/WebVTT)
pour approfondir

V - Styles CSS

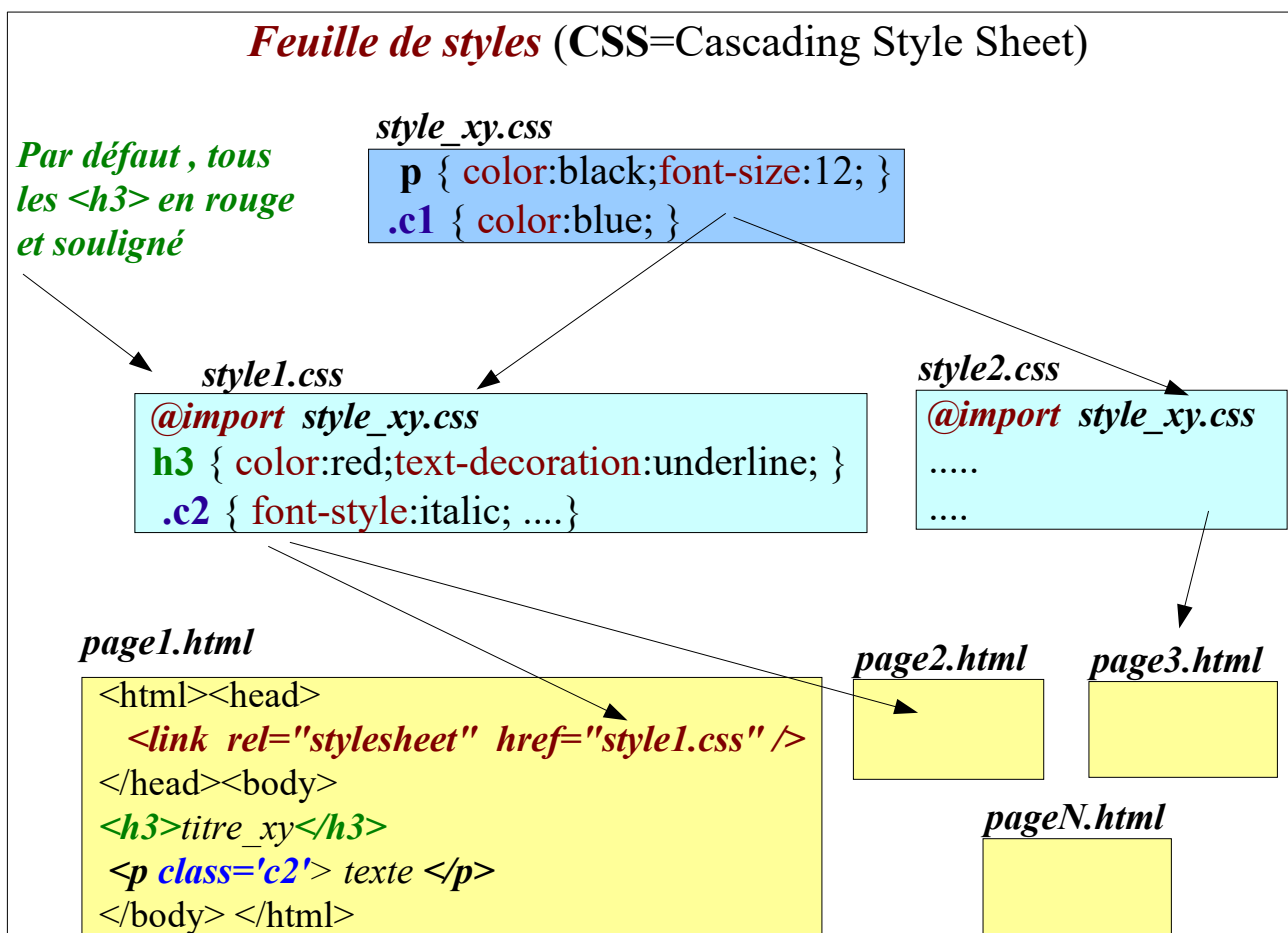
1. Feuilles de styles (CSS, ...)

1.1. Intérêts des feuilles de styles

- Clairement **découpler** (séparer) le **contenu** de la **mise en forme** .
- Permettre différentes représentations d'un même contenu (via l'application de différents styles)
- Basculer très rapidement d'un look à l'autre (en changeant les attributs d'une feuille globale).
- Maintenance du site simplifiée , évolutivité garantie .

1.2. CSS (**C**ascading **S**tyle **S**heet)

Ces **feuilles de styles** peuvent être organisées **en cascade** car on peut établir tout un tas d'inclusions entre différentes expressions complémentaires des styles :



Les **styles CSS** permettent d'appliquer automatiquement (via des règles) certains **attributs de mise en forme** (couleurs, polices , dispositions, ...) à des éléments du document source.

La **sélection d'une règle** repose principalement sur les **noms des balises** ou sur les noms des

classes de styles (attribut *class* d'une balise).

2. CSS (structures et principes de base)

Selon la version du navigateur internet utilisé (IE, NetScape, Opera , ...) , le support des styles CSS peut être assez variables (**CSS niveau 1**, **CSS niveau 2** ou **3**, intermédiaire entre le niveau 1 et 2).

CSS niveau 1	Mise en forme du texte (polices , couleurs ,alignements, ...)
CSS niveau 2	+ Média (écran , page , ...)
CSS niveau 3	Css2 + beaucoup de nouvelles fonctionnalités (rotations , ...)

style1.css

```
p {
    font-weight : bold;    font-size : 14pt;
    font-family : Arial;   font-style: italic;
}
```

Cette feuille de style peut être attachée à un *document xml* de la façon suivante:

```
<?xml version="1.0" ?>
<?xml-stylesheet href="./style1.css" type="text/css" ?>
<doc>
    <titre> titre du document </titre>
    <p> blabla </p>
</doc>
```

ou bien attachée à une *page HTML* de la façon suivante:

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="style1.css">
</head>
<body>...<p> blabla </p> </body>
</html>
```

2.1. Structure d'une feuille de style CSS

Une **feuille de style** est composée d'un **ensemble de règles**

Chaque règle est de la forme suivante:

sélecteur { propriété1: valeur1 ; prop2: val2 ; propN: valN }

héritage d'une valeur de propriété entre éléments parents:

```
livre { background : yellow }
```

```
* { color : inherit }
```

Si un élément de type livre comporte des éléments fils, alors ceux-ci auront également un fond jaune (sauf si il y a une indication explicite d'une autre couleur).

Nb: * en tant que sélecteur signifie n'importe quel élément (balise) du document.

Importation de règles provenant d'une autre feuille de style:

@import "../generic/styles_globaux.css" ;

Nb: la directive **@import** doit être placée tout en haut d'une feuille de style (avant la définition des règles).

Exemple de cascade:

Generic/styles_globaux.css

```
body { background-color:#e6e6ff; }
h3   { color:#0f0064;font-weight:bold;font-family:Verdana,Arial,Helvetica;text-align:center;width:100%; }
```

Xxx/styles.css

```
@import "../Generic/styles_globaux.css" ;
th { background-color:#feeaa5; }
h3 { background-color:#feeaa5; }
```

Xxx/pageY.html

```
<html>
<head>
<link rel="stylesheet" href="/styles.css" type="text/css" >
  <style>
    h3 { width:"60%"; }
  </style>
</head>
<body>
  <h3 style='color:blue;' > Bienvenue </h3>
</body>
</html>
```

2.2. Priorités entre règles concurrentes

Une règle marquée par "**!important**" sera prioritaire sur une règle normale.

syntaxe ==> sélecteur { propriété:valeur **!important** }

Spécifications CSS-2 :

- Dans le cas d'une **cascade** entre feuilles de styles, une **règle importée cède la place à une règle définie localement dans la feuille de style comportant la directive @import**.
- Une règle s'appliquant sur un élément précis l'emporte sur une règle ayant un sélecteur vague (ex: *).
- Si après avoir appliqué tout ce qui précède, il y a encore un conflit entre deux règles; c'est alors la dernière règle spécifiée qui l'emporte.

2.3. Types de médias (CSS niveau 2)

On peut (de façon facultative) mentionner si une règle doit s'appliquer à une sorte particulière de média (écran, imprimante , ...) .

syntaxe:

@media print {

livre { font-size: 10pt ; font-family: Courier }

/ autres règles pour l'impression */*

}

@media screen {

livre { font-size: 12pt ; font-family: Arial }

/ autres règles pour l'affichage écran */*

}

@media print , screen {

/ règles pour les 2 médias */*

}

ou

@import "style-papier.css" print;

@import "style-ecran.css" screen;

3. Sélecteurs

<i>e</i>	élément (balise) de type <i>e</i>
<i>#idXy</i>	élément dont la valeur de l'attribut id vaut <i>idXy</i>
<i>.classe1</i>	élément dont l'attribut class vaut <i>classe1</i>
<i>e1 e2</i>	élément de type <i>e2</i> qui est un descendant (direct ou indirect) d'un élément de type <i>e1</i>
<i>e1 + e2</i>	élément de type <i>e2</i> qui suit immédiatement un élément de type <i>e1</i>
<i>e[att]</i>	élément de type <i>e</i> att ='val'> quelque soit la valeur de l'attribut.
<i>e[att="foo"]</i>	élément de type <i>e</i> ayant un attribut att valant "foo"
<i>e#foo</i>	élément de type <i>e</i> ayant un attribut de type ID valant "foo"
<i>e:lang(fr)</i>	élément de type <i>e</i> ayant 'fr' comme valeur pour l'attribut xml:lang .
<i>e:link</i>	élément de type <i>e</i> qui est un lien xml:link non encore traversé
<i>e:visited</i>	élément de type <i>e</i> qui est un lien xml:link déjà traversé
<i>e:hover</i>	élément de type <i>e</i> qui est survolé à la souris (pas de click)
<i>e:active</i>	élément de type <i>e</i> qui est sélectionné (click souris)
<i>e:focus</i>	élément de type <i>e</i> qui a le focus (destinataire événements du clavier)
<i>*</i>	tout les éléments du document

Si une même règle peut s'appliquer à plusieurs type d'élément , on peut directement écrire:

e1 , e2 , e3 { font-family: Arial; font-size: 12pt }

Vocabulaire:

:hover est une **pseudo classe css**. C'est un mot clef qui peut être ajouté à un sélecteur afin d'indiquer l'état spécifique dans lequel l'élément doit être pour être ciblé par la déclaration de la règle.

::first-line est un **pseudo-élément css**. C'est un mot-clef ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle.

A partir de *CSS niveau 2* (avec ancienne syntaxe *:first-line*) , puis maintenant avec **css3** (avec nouvelle syntaxe **::first-line**) , des **pseudo-éléments** permettent d'attacher des règles particulières à la première ligne ou à la première lettre d'un texte p :

p::first-line { propriété: valeur}

p::first-letter { propriété: valeur}

from::before { **content:** "texte à insérer avant un élément de type from" }

to::after { **content:** "texte à insérer après un élément de type to" }


Principales pseudo-classes css

:hover	Survolé à la souris
:invalid	Zone <input> couramment invalide (ex : ne satisfaisant pas required)
:valid	
p:first-child	tous les éléments p d'une page qui sont les premiers enfant de leur parent
p:last-child	derniers enfant de leur parent
p:first-of-type	tous les éléments p qui sont le premier élément de type p enfant de leur parent
p:last-of-type	derniers enfant de type p de leur parent
tr:nth-child(2n)	Élément tr d'ordre pair (en position 2n vis à vis de leur parent)
tr:nth-child(2n+1)	Élément tr d'ordre impair (en position 2n+1 vis à vis de leur parent)
e:focus	Sélectionne un élément e qui a le focus (dans lequel le curseur de la souris est placé)
e:checked	Sélectionne tout élément e de type input coché au sens large (checked ou selected)
...	

Principaux pseudo-éléments css

::first-line	Première ligne
::first-letter	Première lettre
::before , ::after	
::selection	Partie sélectionnée (à la souris ou autrement)
::placeholder	Texte temporaire de substitution
...	

Quelques applications concrètes :

 <pre> <div class="acrostiche"> <p>Architecte</p> <p>Logiciel</p> </div> </pre>	<pre> .acrostiche > p::first-letter { font-family : playfair , Arial ; color : red; font-size : 2rem; } </pre>
--	---

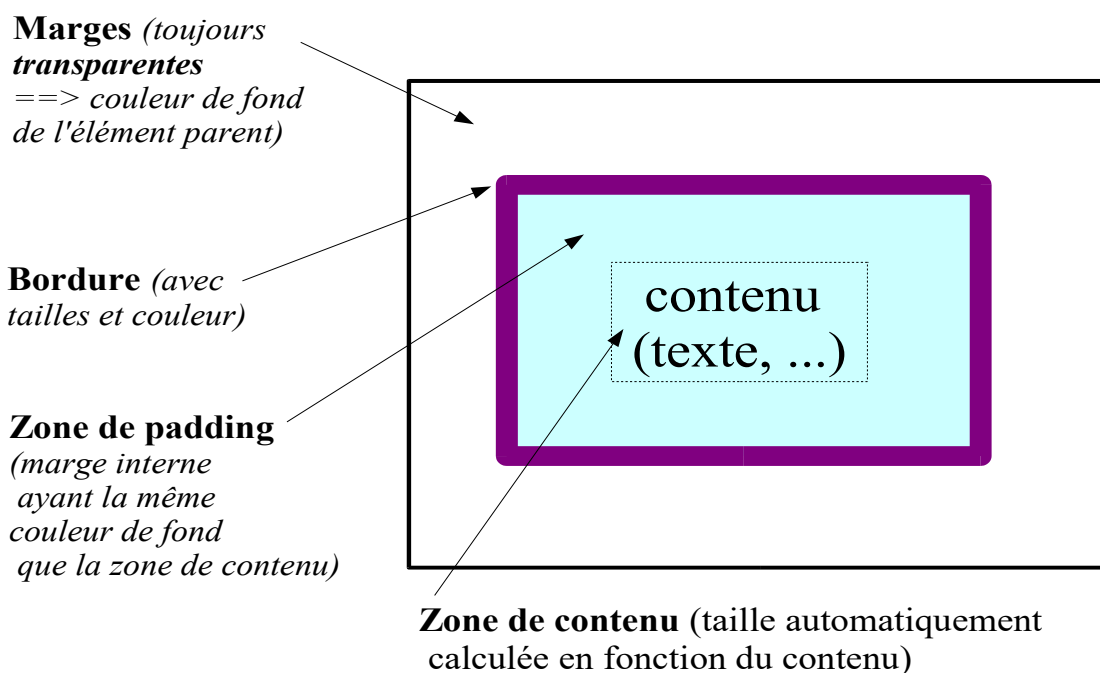
code	nom	change
EUR	Euro	1
USD	Dollar	1.1
GBP	Livre	0.9
JPY	Yen	123.45

```
tbody tr:nth-child(2n) {  
  color: black;  
  background-color: white;  
}  
  
tbody tr:nth-child(2n+1) {  
  color: white;  
  background-color: black;  
}  
  
thead th {  
  background-color: yellow;  
}
```


4. Principaux attributs de CSS

4.1. Modèle de boîtes

CSS (Modèles de boîtes)



NB:

- La **marge** est de taille réglable et est **toujours transparente**.
- La **bordure** est entièrement paramétrable (couleur, relief,...).
- La **boîte de remplissage (padding)** est de taille variable et **a toujours la même couleur de fond que la boîte de contenu**.
- La boîte de contenu est la boîte de référence qui contient les données.

Principales propriétés des boîtes:

width , height	dimension de la boîte de contenu
min-width,...max-height	dimension minimale de la boîte de contenu
margin-top, margin-right, margin-bottom, margin-left	dimensions des marges
margin	dimensions pour toutes les marges
padding , padding-top, ...	dimension de la zone de padding
border, border-top, ...	dimensions de la bordure
border-color	couleur de la bordure
border-style	Valeurs possibles: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

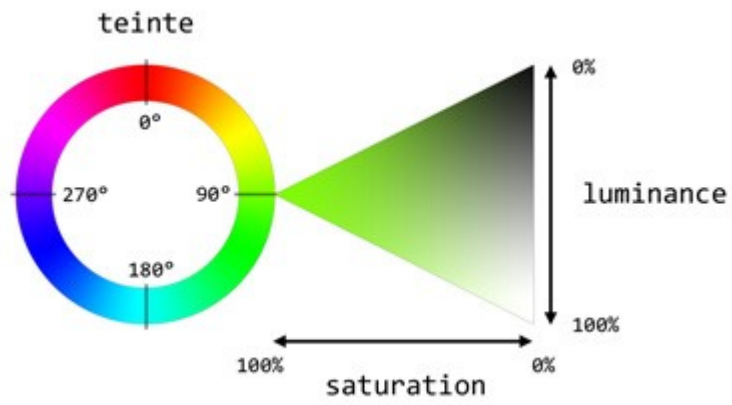
Unités pour les dimensions:

in	inch (pouce)
cm	centimètre
mm	millimètre
pt	point (1/72° de pouce)
pc	pica (12 pt)
em	taille police courante (1 em = à peu près hauteur d'un caractère selon fontSize)
rem	root em = taille de la police attachée à body ou html
%	pourcentage de la largeur ou hauteur du conteneur ou de ...
px	pixel

Codes de couleur:

#RRGGBB ou **aqua**(#00FFFF) , **black** (#000000) , **blue** (#0000FF) , **fuchsia** (#FF00FF) , **gray** (#808080) , **green** (#008000) , **lime** (#00FF00) , **maroon** (#800000) , **navy** (#000080) , **olive** (#808000) , **purple** (#800080) , **red** (#FF0000) , **silver** (#C0C0C0) , **teal** (#008080) , **white** (#FFFFFF) , **yellow** (#FFFF00).

Fonctions pour code de couleur:

<i>fonctions</i>	<i>comportements</i>
rgb(r,g,b)	mélange <u>additif</u> des composantes de bases red , green , blue (valeurs entre 0 et 255)
hsl(h,l,s)	<p>HSL signifie Hue (teinte), Saturation et Luminosité.</p> <p>La teinte est mesurée en degrés de 0 à 360 .</p> <p>Les saturation et luminosité sont exprimées en % (de 0 à 100)</p> 
rgba(r,g,b,a) et hsla(h,l,s,a)	<p>Avec gestion complémentaire de l'opacité/transparence .</p> <p>Le coefficient a (dit alpha) doit avoir une valeur entre 0 et 1 (ex : 0.5)</p> <p>Si a proche de 0 : beaucoup de transparence ,</p> <p>Si a proche de 1 : beaucoup d'opacité .</p>

NB: De très anciens navigateurs peuvent ne pas être capables de gérer hsl , rgba et hsla .

Exemples:

```
.c1 { color: lightGreen; }  
.c2 { color: rgb(255, 0, 0); }  
.c3 { color: rgba(255, 0, 0, 0.5); }  
.c4 { color: hsl(0, 100%, 50%); }  
.c5 { color: hsla(0, 100%, 50% , 0.5); }
```

color	couleur du texte
background-color	couleur de fond (boite de contenu et de remplissage)
background-image	image de fond
background-repeat	mosaïque ou pas (<i>repeat, no-repeat</i>)
background-attachement	image de fond fixe ou défilante % contenu ? (<i>fixed, scroll</i>)

4.2. Propriétés d'affichage / positionnement

display	type de boîte (<i>block</i> , <i>inline</i> , <i>none</i> ,...) si <i>none</i> → pas de boîte générée (collapse) , si différent de <i>none</i> → expand
position	mode de positionnement (<i>fixed</i> : pas de scrolling , <i>static</i> : positionnement normal , <i>absolute</i> : selon coordonnées , <i>relative</i>)
top, right, bottom, left	distance entre un des bords de la boîte courante par rapport au bord adjacent du bloc englobant.
float	Boîte flottante (si pas <i>none</i>) par rapport à la marge droite (<i>right</i>) ou gauche (<i>left</i>) du bloc englobant.
clear	Indique quel coté (<i>left</i> , <i>right</i> , <i>both</i> , <i>none</i>) ne peut pas être adjacent à la boîte flottante qui précède, Provoque un décalage vers le bas
overflow	ce qui dépasse est <i>visible</i> , <i>hidden</i> ou <i>scroll</i>
clip	<i>auto</i> ou <i>rect</i> (top,right,bottom,left) – zone de découpe
visibility	<i>visible</i> , <i>hidden</i>

Différence entre **display**(*block*, *inline*, *inline-block*, *none*) et **visibility**(*visible*, *hidden*) :

Lorsque la propriété **visibility** vaut *hidden* l'élément concerné est caché (invisible) mais occupe toujours la même surface dans la page (on voit généralement une zone blanche) .

Lorsque la propriété **display** vaut *none* l'élément concerné n'est pas rendu du tout dans la page et il occupe aucune surface (effet de contraction : les lignes plus bas remontent) .

visible et display différent de <i>none</i>	Avec visibility : <i>hidden</i>	Avec display : <i>none</i>
ligne avant <div style="border: 1px solid black; padding: 5px; text-align: center;">AAAA BBBB</div> ligne après	ligne avant ligne après	ligne avant ligne après

Liste des éléments html qui sont par défaut de type "inline" :

`<a>` `<abbr>` `<acronym>` , `<audio>` (si les contrôles sont visibles) , `` , `<bdi>` , `<bdo>` `<big>` `
` `<button>` `<canvas>` `<cite>` `<code>` `<data>` `<datalist>` `` `<dfn>` `` `<embed>` `<i>` `<iframe>` `` `<input>` `<ins>` `<kbd>` `<label>` `<map>` `<mark>` `<meter>` `<noscript>` `<object>` `<output>` `<picture>` `<progress>` `<q>` `<ruby>` `<s>` `<samp>` `<script>` `<select>` `<slot>` `<small>` `` `` `<sub>` `<sup>` `<svg>` `<template>` `<u>` `<tt>` `<var>` `<video>` `<wbr>`

Liste des éléments html qui sont par défaut de type "block" :

`<address>` `<article>` `<aside>` `<blockquote>` `<details>` `<dialog>` `<dd>` `<div>` `<dt>` `<fieldset>` `<figcaption>` `<figure>` `<footer>` `<form>` `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` `<header>`

<hgroup> <hr> <main> <nav> <p> <pre> <section> <table>

Différences entre les catégories de contenu html :

Catégories de contenu	Caractéristiques
inline	Pas de saut de ligne entre 2 éléments consécutifs de type inline . La largeur correspond automatiquement à celle du contenu (texte ou ...)
block	Saut de ligne entre 2 éléments consécutifs de type block. On peut fixer toutes les propriétés de tailles (width , height ,)

NB: Changer display de **inline** vers **inline-block** (par exemple sur <label>) permet quelquefois de pouvoir fixer librement la largeur (la propriété width est prise en compte) .

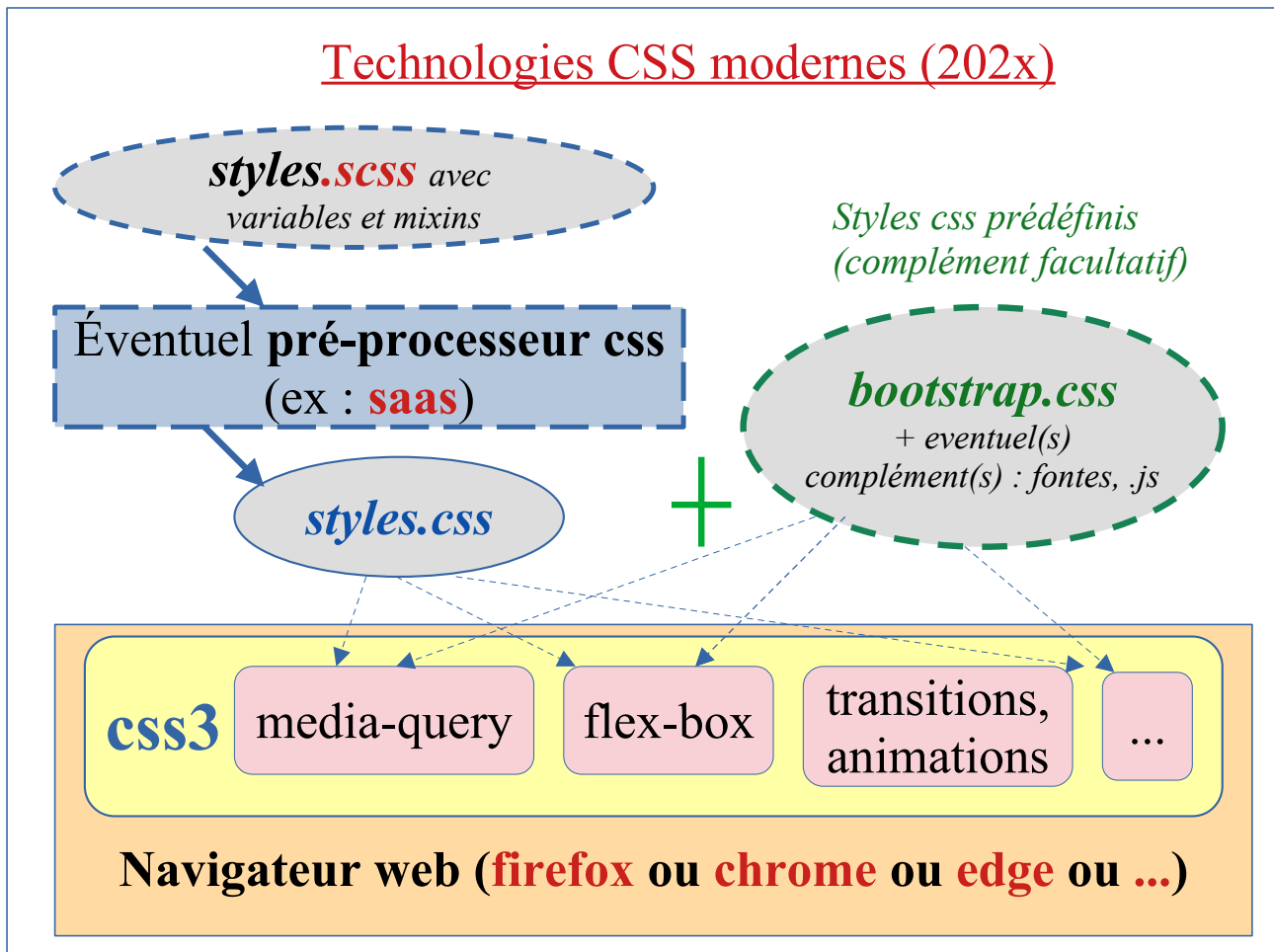
4.3. Mise en forme du texte

Principales propriétés:

font-family	nom(<i>Garamond ,Symbol, Times New Roman</i>) ou famille (<i>serif, sans-serif, cursive, fantasy, monospace</i>) de police
font-style	<i>normal , italic , oblique</i>
font-variant	<i>normal , small-caps</i>
font-weight	<i>100, normal(400) , bold (700) , bolder , lighter , 900</i>
font-size	<i>hauteur_absolue en point</i> ou (<i>larger, smaller</i>) en v-relatif ou (<i>xx-small, x-small, small, medium, large, x-large, xx-large</i>) en v-absolue.
text-indent	indentation de la première ligne de l'élément
text-align	alignement du texte (<i>left, right, center, justify</i>)
text-decoration	<i>underline, overline, line-through, blink, none</i>
text-shadow	<i>couleur_ombrage , dist_horiz, dist_vert</i>
letter-spacing	<i>normal</i> ou <i>distance supplémentaire</i>
word-spacing	<i>normal</i> ou <i>distance supplémentaire</i>
text-transform	<i>uppercase, lowercase, none</i> ou <i>capitalize</i> (1 ^{ère} lettre en Majuscule)
white-space	<i>normal, pre, nowrap</i>

VI - Compléments CSS (fontes , transition, ...)

1. Technologies css (vue d'ensemble)



Base stable supportée par tous les navigateurs modernes :

- media-query
- flex-box
- autres parties de css3 (transitions, animations, pseudo classes et pseudo éléments , ...)

Complément classique (facultatif):

- bootstrap-css (attention aux différences en v3, v4, v5 !!!! , attention aux conflits et effets de bords)
- pré-processeur saas (et fichiers .scss)
- fontes additionnelles (icônes , fontes spécifiques, ...)

2. CSS (compléments)

2.1. Polices de caractères (Fontes)

Famille de Fontes	Caractéristiques	Principales Fontes
sans-serif	Sans sérif (extrémités des caractères simples , sans décorations) F F G G T T (sans-serif)	Arial , Arial narrow , Arial Black , Helvetica , Verdana
serif	Avec serif (extrémités des caractères stylisés via des petits traits) F F G G T T (serif)	Times New Roman
monospace	Chaque caractère d'une police monospace occupe la même largeur	Courier , Lucida Console
cursive	Proche d'une écriture manuscrite	Comic Sans MS
fantasy	Plus ou moins fantaisistes	Papyrus

Sans SERIF :

F F G G T T (sans-serif)

un deux trois quatre cinq (Arial)

un deux trois quatre cinq (Arial narrow)

un deux trois quatre cinq (Arial Black)

un deux trois quatre cinq (Helvetica)

un deux trois quatre cinq (Verdana)

SERIF :

F F G G T T (serif)

un deux trois quatre cinq (Times New Roman)

Monospace :

un deux trois quatre cinq (monospace)

un deux trois quatre cinq (Courier)

un deux trois quatre cinq (Lucida Console)

Cursive :

un deux trois quatre cinq (cursive)

un deux trois quatre cinq (Comic Sans MS)

Fantasy :

un deux trois quatre cinq (fantasy)

un deux trois quatre cinq (papyrus)

Exemple :

```
<p style="font-family:cursive">un deux trois quatre cinq (cursive)</p>
```

```
<p class="c1">ligne1 en classe c1 (Arial , taille 14pt , italique)</p>
<p class="c2">ligne2 en classe c2 (monospace , taille 20pt , gras) </p>
<p class="c3">ligne3 en classe c3 (cursive,centré horizontalement , souligné)</p>
```

```
.c1 { font-family : Arial; font-size : 12pt; font-style : italic; }
.c2 { font-family : monospace; font-size : 24pt; font-weight : bold; }
.c3 { font-family : cursive; text-align : center; text-decoration : underline; }
```

```
.withAlluraFont{
    font-family : Arial narrow, Arial , sans-serif;
    font-size : 2rem;
}
```

NB : la valeur de *font-family* peut être constituée de plusieurs noms de fontes (ou familles de fontes) avec une sémantique suivante : *valeurPréférée* , *ValeurAlternativeSiPasDispo* , ...

2.2. Polices personnalisées

Une définition de classe css peut faire référence à une police de caractère personnalisée (ici "alluraregular") .

styles.css

```
@import 'font/allura-webfont.css';
.withAlluraFont{
    font-family : alluraregular, Arial ;
    font-size : 2em;
}
```

Cette police de caractère n'étant pas standard (pas déjà connu par les navigateurs web) , on a besoin de stocker dans l'application une configuration de cette police (formes des caractères,) qui sera téléchargée d'un serveur HTTP vers un navigateur avant d'être utilisée .

css/font/allura-webfont.css

```
@font-face {
    font-family: 'alluraregular';
    src: url('allura-regular-webfont.woff2') format('woff2'),
        url('allura-regular-webfont.woff') format('woff'),
        url('allura-regular-webfont.svg#alluraregular') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

les fichiers *css/font/allura-regular-webfont.woff2* ,woff , ...svg correspondent à des encodages des formes des caractères dans différents format normalisés .

Paragraphe dans une police de caractère spécifique (allura regular)

P A R A G R A P H E D A N S U N E A U T R E
P O L I C E D E C A R A C T È R E
S P É C I F I Q U E (T H E **R** O O T S)

Principaux formats des fichiers de fontes

Formats	Caractéristiques
TTF : TrueType Fonts	format ancien pas compressé
OTF : OpenType Fonts	version améliorée de TTF (adobe, Microsoft) , format pas compressé
EOT : Embedded OTF	restreint à Microsoft Internet Explorer
SVG : Scalable Vector Graphics	--> fichier très léger , adapté aux mobiles et à IOS
WOFF (Web Open Font Format)	à partir de 2010 avec IE et Firefox
WOFF2	à partir de 2013 , ok avec Edge , Fx , Chrome, Opera, ...

Formats conseillés (en //) : **.woff2** et **.svg** (avec navigateurs récents)

Eventuel complément **.woff** pour navigateurs un peu plus anciens

Sources de quelques fontes/polices spécifiques:

highlight: https://www.dafont.com/fr/search.php?q=the_root

allura: <https://www.fontsquirrel.com/fonts/download/allura>
<https://www.fontsquirrel.com/fonts/download/playfair-display>

NB: Depuis de site <https://fonts.google.com> on peut télécharger des ".zip" avec plein de fontes intéressantes .

Génération de font (multi-formats):

Le site web "**fontsquirrel**" (d'url <https://www.fontsquirrel.com/tools/webfont-generator>) est un générateur de fonte multi-formats en ligne accessible à tous.

En entrée on peut uploader une fonte dans un ancien format

En sortie on peut télécharger un zip comportant la fonte dans des formats récents avec un fichier css (comportant **@font-face**) généré automatiquement .

Mode opératoire :

1. choisir mode expert avec svg
 2. uploader la font brute de départ
 3. renommer stylesheet.css en xyz-webfont.css dans partie CSS/CSS file name
 4. cocher "agreement"
 5. Download kit
- > zip -->allura-regular-webfont...

2.3. Positionnements via css

Eléments flottants ("float" et "clear")

```
.floatLeft { float: left; }
```

```
.floatRight { float: right; }
```

```
***
float:none;
***
```

La propriété float indique qu'un élément doit être retiré du flux normal et doit être placé sur le côté droit ou sur le côté gauche de son conteneur.

Le texte et les autres éléments en ligne (inline) entoureront alors l'élément flottant. L'élément est retiré du flux normal de la page mais s'inscrit toujours dans le flux (contrairement au positionnement absolu).

```
***
float:right;
***
```

```
***
float:left;
***
```

La propriété float indique qu'un élément doit être retiré du flux normal et doit être placé sur le côté droit ou sur le côté gauche de son conteneur.

La propriété float indique qu'un élément doit être retiré du flux normal et doit être placé sur le côté droit ou sur le côté gauche de son conteneur.

```
***
float:right;
***
```

```
.clearLeft { clear : left; }
```

```
.clearRight { clear : right; }
```

```
.clearBoth { clear : both; }
```

```
***
clear:right; signifie dégagé
float:left; plus bas pour ne pas être sur la
*** même ligne que des éléments
*** flottants à droite
***
***
###
***
clear:right;
###
```

```
***
div
float:left; without
*** clear
***
```

```
***
clear:left; signifie dégagé plus
float:left; bas pour ne pas être sur la
*** même ligne que des éléments
*** flottants à gauche
***
###
clear:left;
###
```

```
***
clear:both; signifie dégagé plus
float:left; bas pour ne pas être sur la
*** même ligne que des éléments
*** flottants à gauche ou à droite
***
###
clear:both;
###
```

NB : les propriétés float et clear sont un peu complexes et résultat global obtenu est un peu au cas par cas (selon les très nombreuses combinaisons possibles) → à utiliser avec précautions et avec parcimonie .

Sticky header (v1)

pageXy.html

```
<html>...<body> <header class="stickyHeader" >....</header>
<main class="mainContentUnderStickyHeader mainContentAboveStickyFooter">
...
</main>
<footer class="stickyFooter">...</footer> </body></html>
```

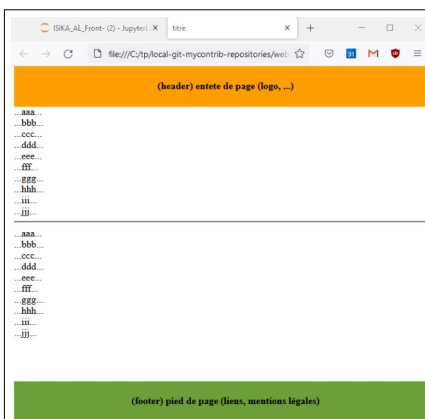
layout.css

```
.stickyHeader {
  position: fixed; /* Set to fixed position */
  top: 0; /* Position at the top of the page */
  width: 100%; /* Full width */
  height : 4rem; /* small necessary height for header content */
}

.stickyFooter {
  position: fixed; /* Set to fixed position */
  bottom: 0; /* Position at the bottom of the page */
  width: 100%; /* Full width */
  height : 4rem; /* small necessary height for footer content */
}

.mainContentUnderStickyHeader{
  margin-top: 4rem; /* Add a top margin to avoid content overlay */
}

.mainContentAboveStickyFooter{
  margin-bottom: 4rem; /* Add a bottom margin to avoid content overlay */
}
```



header toujours visible , collé au bord haut du navigateur.
partie "main" qui défile en dessous de la partie "header".

Sticky header (v2 avec flex-box)*pageXy.html*

```

<html>...
<body>
<div class="flexAllInColumnDir">
  <header id="mainHeader" class="stickyHeaderFirstInFlexCol" >
    <h4> (header) entete de page (logo, ...)</h4>...
  </header>
  <div class="remainingContentInFlexCol">
    <main>
      ...aaa...<br/>...bbb...<br/>...ccc...<br/>...ddd...<br/>...eee...<br/>  <hr/>
      ...aaa...<br/>...bbb...<br/>...ccc...<br/>...ddd...<br/>...eee...<br/>
    </main>
    <footer id="mainFooter" class="remainingContentInFlexCol" >
      <h4>(footer) pied de page (liens, mentions légales)</h4>
    </footer>
  </div>
</div>
</body></html>

```

flex-layout.css

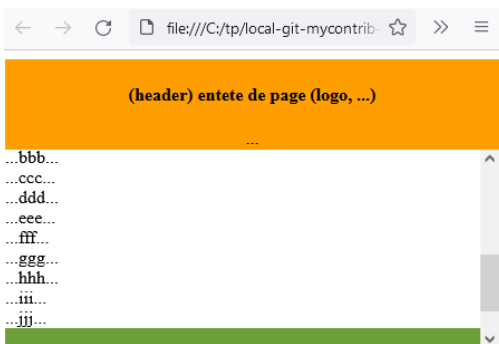
```

.flexAllInColumnDir {
  height: 100%;      width: 100%;
  display: flex;      flex-direction: column;
}

.stickyHeaderFirstInFlexCol {
  flex-shrink: 0;
}

.remainingContentInFlexCol {
  flex-grow: 1; /*ensures that the container will take up the full height of the parent container*/
  overflow-y: auto; /*adds scroll to this container*/
}

```



3. CSS (transitions et animations)

Le paramétrage d'une **transition** css permet de **passer d'un état_1/style1 à un état_2/style2 de manière très progressive** (par exemple : couleur de fond basculant progressivement du bleu au vert en 1 seconde)

3.1. transitions css

Exemple simple :

```
p {
background-color: blue;
transition-property: background-color; /* Transition sur background-color*/
transition-duration: 1s; /* Durée 1s */
}

p:hover{background-color: green;}
```

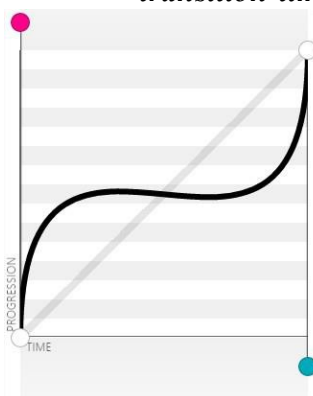
Paramétrages avancés sur une transition :

* **transition-timing-function** (accélérations éventuelles au cours du temps) :

<i>ease</i>	rapide au début, ralenti à la fin
<i>linear</i>	vitesse constante
<i>ease-in</i>	lent à l'approche , accélère à la sortie
<i>ease-out</i>	rapide à l'approche , ralenti à la sortie
<i>ease-in-out</i>	approche et sortie lente
<i>cubic-bezier</i>	courbe de bezier (selon paramétrage)

exemples : *transition-timing-function:ease;*

transition-timing-function:cubic-bezier(0,1.1,1,-0.1);



très rapide au début, légère régression, très rapide à la fin

Syntaxe synthétique : propriété "**transition**" regroupant plusieurs propriétés de transition:

```
{ transition: background-color 1s cubic-bezier(0,1.1,1,-0.1); }
```

et plusieurs transitions en même temps sont possibles :

```
{transition : background-color 2s ease , width 1s linear ;}
```

Applications concrètes (quelques effets):

```
.small-zoom { transform: scale(0.96); transition: 0.6s;}  
.small-zoom:hover { transform: scale(1.0);  
                    border-width: 4px; border-style: solid; border-color: red;}  
  
.tournerSurSoiMeme:hover { transform: rotate(360deg); transition: all 0.5s ease;}  
  
.noirEtBlanc { filter: grayscale(100%); transition: .3s ease-in-out;}  
.noirEtBlanc:hover { filter: grayscale(0); }  
  
.rond { transition: .3s ease-in-out;}  
.rond:hover { border-radius : 50% ; }
```

```

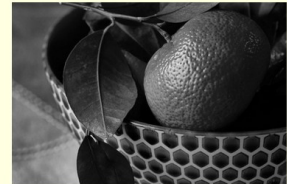
```

```

```

```

```



3.2. animations css

Au sein d'un fichier css, une animation se configure en "keyframes" (ayant un nom logique).
On y définit les étapes de l'animation (par exemples à 0 %, 50 % , 100 % du temps) :

@keyframes *monanimation*

```
{
  0% { transform: translateX(0px); }
  50% { transform: translateX(40px) rotate(-15deg) ;}
  100% { transform: translateX(80px) rotate(15deg); }
}
```

@keyframes *monrebond*

```
{
  0% { transform: translateY(0px) translateX(0px);}
  50% { transform: translateY(8px) translateX(2px);}
  100% { transform: translateY(0px) translateX(0px); }
}
```

On applique ensuite une animation via la propriété "animation" :

```
a:hover { color: blue; transition: color 1s linear; animation : monrebond 1s;}
button:hover { color: blue; animation : monrebond 1s;}

p:hover { background-color: #93db83; /* vert*/; animation: monanimation 2s ; }
```

Applications concrètes (quelques effets):

```
@keyframes flash-anim {
  0% { opacity: 1; }
  25% { opacity: 0; }
  50% { opacity: 1; }
  75% { opacity: 0; }
  100% { opacity: 1; }
}
```

```
.myflash:hover {
  animation: flash-anim 3s;
}
```

```
/* application via  */
```

```
@keyframes secouer-anim {
  0% { transform: rotate(10deg); }
  25% { transform: rotate(-10deg); }
  50% { transform: rotate(10deg); }
  75% { transform: rotate(-10deg); }
  100% { transform: rotate(0deg); }
}

.secouer:hover {
  animation: secouer-anim 3s;
}

/* application via  */
```

Effets engendrés :

- avec class="myflash", l'élément concerné apparaît , disparaît , réapparaît , redisparaît , réapparaît.
- avec class="secouer", l'élément concerné tourne plusieurs fois dans uns sens puis dans l'autre.

Autre effet à base d'animation :

Changement automatique de la couleur du texte (ou du fond) :

```
/* HSL signifie Hue, Saturation, Luminosity,
c'est à dire Teinte, Saturation, Luminosité (https://la-cascade.io/utiliser-hsl-pour-vos-couleurs/) */
@keyframes wheelHueColor {
  from, to { color: #c32283; }
  10% { color: #bd2828; }
  20% { color: #cf8517; }
  30% { color: #819c16; }
  40% { color: #36931f; }
  50% { color: #169c4c; }
  60% { color: #169c9c; }
  70% { color: #2e65b8; }
  80% { color: #4e2dd2; }
  90% { color: #8b29a3; }
}

.animTextColor {
  color: #c32283;
  animation: wheelHueColor 60s infinite;
} /* infinite pour répétition infinie de l'animation */

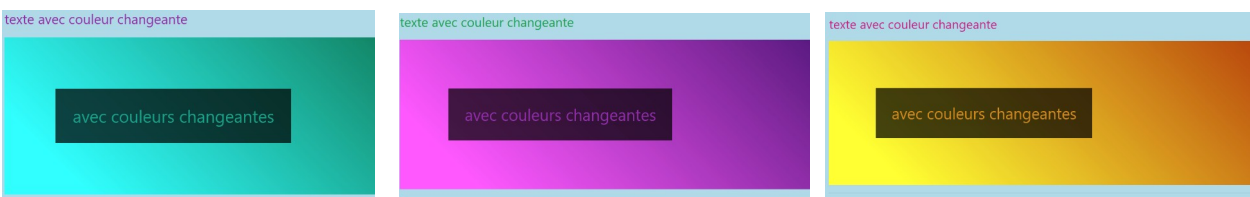
.animBackgroundColor {
  flex-flow: row wrap; align-items: center; color: #c32283;
  animation: wheelHueColor 100s infinite;
  background-color: currentColor;
  background-image: linear-gradient(45deg, white 10%, black 90%);
  background-blend-mode: overlay;
}
```



```
.textOverAnimeBackColor {
  display: inline-block; flex: 1 1 auto; margin: 3em;
  text-align: center; background: rgba(0, 0, 0, 0.75);
  padding: 1em; animation: inherit;
}
```

```
<div class="animTextColor">
  <p>texte avec couleur changeante</p>
</div>

<div class="animBackgroundColor">
  <h2 class="textOverAnimeBackColor">avec couleurs changeantes (texte et fond)</h2>
</div>
```



3.3. Utilité des animations :

Attirer l'oeil de l'utilisateur sur un détail important .

NB: le *framework angular* permet un déclenchement d'animation suite à un changement d'état de l'application (exemple: caddy modifié , utilisateur connecté, ...) via *trigger(...)* .

Ceci est particulièrement efficace pour attirer l'oeil de l'utilisateur sur un changement important .

Démarrage et contrôle d'une animation par code javascript :

```
var image1 = document.getElementById("image1");
var btnStart = document.getElementById("btnDemarrerAnimationSurImage1");
btnStart.addEventListener('click',function (){
  image1.style.animation = "none"; //stop before restart
  setTimeout( () => {
    image1.style.animation = "secouer-anim 4s";
    }, 2); //a small delay (ex: 2 ms ) is necessary
});

var btnPause = document.getElementById("btnStartStopPauseAnimationSurImage1");
btnPause.addEventListener('click',function (){
  if(image1.style.animationPlayState != 'paused')
    image1.style.animationPlayState = 'paused';
    else image1.style.animationPlayState = 'running';
});
```

4. CSS responsive (media-query , flex, ...)

4.1. media-query

```

/*d'abord les règles pour tous médias et tailles */
h1 { font-size: 72px}
h1 , h2 { color : #0ca027 }
nav { height: 50px; background-color : lightgray; text-align : center}
p { width: 200px; height: 50px; background-color: #85c4db; /*bleu ciel */ }

/* ensuite nouvelles regles pour taille > 600px de large) */
@media screen and (min-width : 600px){
  body { background-color : #f4b6a6}
}

/* ensuite nouvelles regles pour taille < 600px de large) */
@media screen and (max-width : 600px){
  body { background-color : #f0f296}
  h1 { font-size: 32px}
  h2 { color : black;}
  nav { display : none; }
}

```

Critères de sélection pour "media-query" :

height	hauteur de la fenêtre
width	largeur de la fenêtre
device-height	hauteur du périphérique (ex : tablette)
device-width	largeur du périphérique
orientation	portrait ou paysage
media screen handheld print ... all	type de média écran classique mobile (smartphone) imprimante

combinaisons de critères possibles avec **not** , **only** , **and** , **or** ...

Attention (pour une page html devant avoir un comportement "responsive" sur un smartphone , il ne faut pas oublier le paramètre "viewport" :

```
<html>
  <head> ...
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- for responsive css on mobile device -->
</head>
```

4.2. Flex box

Une boîte flexible (**flexbox**) permet de disposer les éléments (en horizontal ou en vertical) d'une manière responsive / flexible .

Il faut définir l'axe principal (**flex-direction**) ayant une des valeurs suivantes :

- **row**
- **row-reverse** (*inverse au sens d'écriture lui même associé à la langue*)
- **column**
- **column-reverse**

L'axe secondaire (cross-axis) est toujours perpendiculaire à l'axe principal .

Exemple :

```
.box {
  display: flex;
  flex-direction: row;
}
```

NB :

Si les éléments d'une boîte (en ligne par exemple) sont trop grands pour tenir sur une même ligne , ceux-ci seront :

- soit affichés sur plusieurs lignes (**flex-wrap: wrap**)
- soit éventuellement rétrécis pour tenir sur une même ligne ou vont provoquer un dépassement (**flex-wrap: nowrap** / valeur par défaut) .

Syntaxe synthétique :

flex-flow: row wrap (ou row nowrap) (ou column nowrap) ...

Exemple complet (avec images et textes qui s'ajustent) :*flex-page.html*

```

<html>
<head>
  <title>flex-page</title>
  <link rel="stylesheet" type="text/css" href="/css/styles-flex.css" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
</head>
<body>
  <h1>page responsive</h1> <h2>via flex box</h2> <hr/>
  <div class="mywrapbox">
    <div class="myitem proportionalText">-OneOneOneOne-</div>
    <div class="myitem proportionalText">*Two*Two*<br>
      *Two*Two*<br>*Two*Two*</div>
    <div class="myitem proportionalText">-ThreeThreeThree-</div>
    <div class="myitem proportionalText">*FourFourFour*</div>
  </div>
  <div class="mybox">
    <!-- les images 1 , 2 et 3 sont de différentes tailles -->
    <div class="mycell"></div>
    <div class="mycell"></div>
    <div class="mycell"></div>
  </div>
</body>
</html>

```

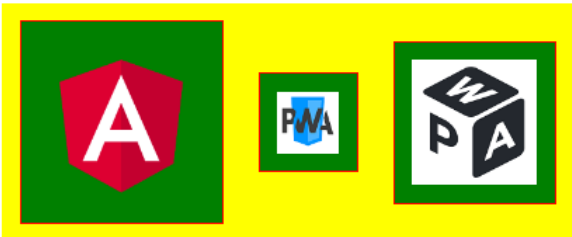
-OneOneOneOne-

*Two*Two*
 *Two*Two*
 *Two*Two*

-ThreeThreeThree-

FourFourFour



<p>-OneOneOneOne- *Two*Two* *Two*Two* *Two*Two* *FourFourFour*</p> <p>-ThreeThreeThree-</p> 	<p>4ème texte sur la ligne d'après car "wrap" sur 1ère box.</p> <p>Centrage vertical vis à vis du texte de plus grande taille '*TwoTwo
...
...Two*'</p> <p>Les images sont automatiquement réduites en taille car "nowrap" sur 2 ème box et <code>img { max-width:100%;}</code> .</p>
--	--

styles-flex.css

```

/* align-items : center ou stretch ou flex-start ou flex-end
   pour contrôler l'alignement sur l'axe secondaire si les éléments n'ont pas la même hauteur */
.mybox {
    background-color: yellow;
    display: flex;
    flex-flow: row nowrap;
    align-items : center;
}

.mywrapbox {
    display: flex;
    flex-flow: row wrap;
    align-items : center;
}

/* flex: flex-grow, flex-shrink, flex-basis
   flex-basis = taille occupée ou bien auto (automatiquement calculée)
   flex-grow et flex-shrink sont des coefficients (valeurs relatives par rapport autres éléments)*/
.myitem {
    flex: 1 1 auto;
    text-align: center; padding: 2px;
}

.mycell {
    flex: 1 1 auto;
    padding: 10px; margin: 10px; background-color: green;
    border: 1px solid red; text-align: center;
}

img { max-width:100%;} /* pour shrink sur images si trop grandes */

```

/ attention , l'ordre des min-width est important du plus petit au plus grand !!!*

*xs, sm , md , lg , xl */*

```
@media screen and (max-width : 36em){
    .proportionalText { font-size: 80%; color:red; }
}

@media screen and (min-width:36em ){
    .proportionalText {font-size: 100%;      color:black; }
}

@media screen and (min-width:48em ){
    .proportionalText {font-size: 110%;      color:green; }
}

@media screen and (min-width:64em){
    .proportionalText {font-size: 120%;      color:blue;}
}

@media screen and (min-width : 80em){
    .proportionalText {font-size: 130%;      color:red;}
}
```

5. Sass (.scss) (pré-processeur css)

5.1. Tour d'horizon des préprocesseurs css

Les préprocesseurs CSS les plus couramment employés sont aujourd'hui [Sass](#), [LESS](#) , [PostCSS](#) et [Stylus](#).

pré-processeur CSS	principales caractéristiques
LESS	préprocesseur codé en javascript (utilisation dynamique "just in time" possible mais moins rapide et rarement envisagée)
Sass	préprocesseur initialement codé en ruby , maintenant des implémentations en plein de langage (C , java, ...) . Un peu plus puissant que "less" .
PostCSS	préprocesseur modulaire à base de plugins , plus récent que Saas , moins

	connu que Saas
Stylus	basé sur nodeJs ,inspiré de saas, pour l'instant pas beaucoup utilisé



La suite de ce chapitre se focalisera sur le préprocesseur Saas (pour l'instant le plus utilisé) .

5.2. Présentation de Sass

Sass = Syntactically Awesome Stylesheet .

Les premières versions de Sass étaient très différentes de notre CSS; il n'y avait pas d'accolades, et les propriétés devaient être indentées avec un nombre précis d'espaces sous peine de recevoir un message d'erreur du compilateur.

La version 3.0 a introduit une nouvelle syntaxe, plus proche de CSS, appelée SCSS ("Sassy CSS" qu'on pourrait traduire par "CSS à la Sass" ou bien "CSS classieux"). SCSS peut maintenant être vu comme une extension de CSS, ce qui signifie que *tout ce qui est valide en CSS l'est aussi en SCSS*.

Principales fonctionnalités de sass/scss :

- utilisation de variables (couleurs , tailles, fontes, , ...)
- modularisation du "scss ==> css" (imbrication, héritage , mixin, ...)
- opérations simples pour agrandir, assombrir , éclaircir, ...
- boucles , évaluations de conditions, opérations personnalisées , ...

Limitation importante : Les navigateurs actuels analysent les fichiers .CSS mais pas directement les fichiers .SCSS. *Autrement dit, un changement de variable dans un fichier .SCSS n'a un effet qu'en régénérant une nouvelle version du fichier .CSS* (ce qui ne peut être effectué que *durant la phase de développement* et pas dynamiquement au runtime).

5.3. Installations possibles d'un processeur scss :



Télécharger et installer l'application "**Scout-app**" (ex: WIN_Scout-App_scss_2.18.16.zip) depuis le site <https://scout-app.io/> .

ou bien , en mode global avec nodeJs :

```
npm install -g sass
```

NB : l'implémentation "nodeJs" (en javascript) n'est pas la plus rapide mais elle est portable (linux, windows,)

NB : Plein d'autres possibilités sont envisageables.

5.4. Bases de scss

Fichiers avec extensions **.scss** . **Variables** préfixées par \$ (et par défaut globales).

Une variable peut éventuellement être définie et utilisée localement .

scss/*basic.scss*

```
$couleur1 : #ff00aa;
h2 { color: $couleur1 ; }
h3 { color: $couleur1 ; }
p { $couleur1 : #eebb00; color : $couleur1; }
```

```
sass scss/basic.scss out-css/basic.css
```

out-css/*basic.css*

```
h2 { color: #ff00aa; }
h3 { color: #ff00aa; }
p { color: #eebb00; }
```

Il est possible de modifier la portée d'une variable en lui attribuant un suffixe !default ou !global .

5.5. import de sous fichiers "partial"

Un sous fichier "partial" a un nom commençant par un *underscore* "_" .

Lorsqu'un tel fichier sera traité par le préprocesseur sass , sont contenu sera recopié / inclus dans un fichier ou apparaîtra l'instruction @import sans reproduire le "@import dans le css généré .

Autrement dit un sous fichier _partielXy.scss n'est pas transformé en _partielXy.css .

Exemple :

_variables.scss

```
$couleur1 : #ff00aa;
$couleur2 : #eebb00;
```

basic.scss

```
@import "_variables";
h2 { color: $couleur1 ; }
p { color : $couleur2; }
```

--> **basic.css**

```
h2 { color: #ff00aa;}
p { color: #eebb00; }
```

5.6. Imbrications de règles et pseudo sélecteur de parent &

Pour rendre le paramétrage des styles plus modulaires (avec moins de copier/coller) , on pourra imbriquer des règles les une dans les autres et au sein d'une règle imbriquée , le pseudo sélecteur & fera référence au sélecteur de la règle parente/englobante .

Exemple :

```
@import "_variables";

h2 {
  color: $couleur1 ;
  &:hover {
    color: $couleur3 ;
  }
}

p {
  color : $couleur2;
  &:hover {
    color: $couleur4 ;
  }
}
```

==>

```
h2 {
  color: #ff00aa;
}
h2:hover {
  color: #ff0000;
}

p {
  color: #eebb00;
}
p:hover {
  color: #00bb00;
}
```

5.7. Fonctions intégrées/prédéfinies

basic.scss

```
@import "_variables";
h3 { color: lighten($couleur1, 20%); }
h4 { color: darken($couleur1, 20%); }
h5 { background-color: transparentize($couleur1, 0.5);}
```

==>

basic.css

```
h3 { color: #ff66cc;}
h4 { color: #990066;}
h5 { background-color: rgba(255, 0, 170, 0.5);}
```

La liste des fonctions prédéfinies est accessible au bout de l'url suivante :

<https://sass-lang.com/documentation/Sass/Script/Functions.html>

principales fonctions intégrées :

fonctions intégrées	finalités/caractéristiques
lighten (baseColor, coeff)	éclaircir une couleur
darken (baseColor, coeff)	assombrir une couleur

5.8. @extend et %placeholder

L'instruction **@extend** (sans s) permet de récupérer (par pseudo-héritage) tous les attributs de mise en forme d'une autre règle considérée comme un simple "snippet" (morceau à réutiliser).

Lorsqu'un bloc d'attributs scss n'est utile que pour effectuer ultérieurement des héritages/copies on peut utiliser la syntaxe spéciale **%placeholderXyz { ... }** et ... **{ @extend %placeholderXyz ; ... }**

Exemple :

basic.scss

```
...
%abstract-bold-underline{
  text-decoration: underline;
  font-weight : bold;
}
h1 { @extend %abstract-bold-underline; text-transform : uppercase; }
h3 { @extend h1; color: $couleur1 ; }
h4 { @extend h1; color: $couleur2 ; }
h5 { @extend h1; color: $couleur3;}
```

==>

basic.css

```
h1, h5, h4, h3 {
  text-decoration: underline;
  font-weight: bold;
}

h1, h5, h4, h3 {
  text-transform: uppercase;
}

h3 { color: #ff66cc;}
h4 { color: #990066;}
...
```

Bien que techniquement utilisable entre éléments quelconques , le mot clef **@extend** devrait idéalement être utilisé entre éléments de sémantiques proches de manière à que la notion d'héritage soit intelligible (compréhensive , intuitive, ...) .

5.9. @mixin et @include

@mixin permet la définition d'un *snippet* réutilisable via **@include** .

un **@mixin** est assimilable à une fonction réutilisable avec la possibilité de passer des paramètres , d'effectuer des boucles , d'évaluer des conditions, ...

Exemple :

basic.scss

```
@mixin with-border($color){
  border: 2px solid $color;
}
```

==>

basic.css

```
h3 { @include with-border(blue); color: red; }
```

5.10. interpolations , évaluations sophistiquées de variables

La syntaxe **#{\$variable}** permet des juxtapositions et autres manipulations sophistiquées de variables scss .

Exemple :

```
$className: c1;
$attr: border;
p.$className {
  $attr-color: blue;
}
```

```
$className: c1;
$attr: border;
p.#{$className} {
  #{$attr}-color: blue;
}
```

==>

```
p.c1 {
  border-color: blue;
}
```

5.11. Boucles scss

```
@for $i from 1 through 3 {
  .spacer-#{ $i } {
    margin: $i * 1rem;
  }
}
```

==>

```
.spacer-1 {
  margin: 1rem;
}

.spacer-2 {
  margin: 2rem;
}

.spacer-3 {
  margin: 3rem;
}
```

5.12. Evaluation de condition scss

```
@mixin with-rotation($rotation) {
  @if $rotation == 0 {
  }
  @else {
    transform: rotate($rotation);
  }
}

.avecBordureEtRotation {
  width:300px;
  @include with-rotation(25deg);
  @include with-border(blue);
}
```

==>

```
.avecBordureEtRotation {
  width: 300px;
  transform: rotate(25deg);
  ...
}
```

5.13. Bonnes pratiques et organisations "scss"

Pour projets simples/ordinaires :

_base.scss ou bien **_variables.scss** + **_mixins.scss** (variables , mixins , reset or normalize)

_layout.scss (container , grid , layout , ...)

_component.scss (navbar , card , ...)

main.scss ou bien **styles.scss** (@import , ...)

Quelques idées (en vrac) :

- **_reset.scss** ou **_normalize.scss** (valeurs par défaut , moins de différences entre les navigateurs, ...)
- utiliser (ou bien s'inspirer de) certains "framework css" tels que "**blueprint** css" ou "**compass** css" .
- Utiliser une des librairies de fichiers .scss prédéfinis

bourbon (https://www.bourbon.io/)	très complet
https://github.com/matthieua/Sass-css3-mixins	simple
https://cssowl.owl-stars.com/	
https://davidtheclark.github.io/scut/color-swap.html	
http://breakpoint-sass.com/	
http://colindresj.github.io/saffron/	
https://sassline.com/	
http://gillesbertaux.com/andy/	

• ...

5.14. Syntaxes scss avancées

@content; utilisé au sein d'un mixin est automatiquement remplacé par le contenu d'un bloc entre accolades précisé lors de l'invocation/appel du mixin via **@include** .

.scss

```
@mixin mx-hover {
  &:hover {
    @content;
  }
}

.button {
  border: 1px solid black;
  @include mx-hover {
    border-width: 2px;
  }
}
```

⇒ .css

```
.button {
  border: 1px solid black;
}
.button:hover {
  border-width: 2px;
}
```

```
$my-media-coeffs: ( "sm": 100% , "md": 110% , "lg": 120% , "xl": 130% );
```

correspond à une **map** (*table d'association* ici entre tailles et pourcentages).

Ultérieurement , au sein d'un mixin , après `$_key = "md"` ;

l'instruction **map-get(\$my-media-coeffs, \$_key)**; permettra de récupérer la valeur de la map associée à la clef (ici 110% si `$_key` vaut "md").

5.15. Responsive scss

bases.scss

```
$my-media-breakpoints-bootstrap-css: (
  "sm": 576px,
  "md": 768px,
  "lg": 992px,
  "xl": 1200px
);

$my-media-breakpoints : $my-media-breakpoints-bootstrap-css;
//coeffs for font-size of text :
$my-media-coeffs: ( "sm": 100%, "md": 110%, "lg": 120%, "xl": 130% );

@mixin media-min($_key) {
  @if map-has-key($my-media-breakpoints, $_key) {
    @media screen and (min-width: map-get($my-media-breakpoints, $_key)) {
      &{ @content; }
    }
  } @else {
    // Log a warning.
    @warn 'Invalid breakpoint key: #{$_key}.';
  }
}

@mixin media-max($_key) {
  @media screen and (max-width: map-get($my-media-breakpoints, $_key) - 1px) {
    &{ @content; }
  }
}

@mixin media-between($_keymin, $_keymax) {
  @media screen and (min-width: map-get($my-media-breakpoints, $_keymin)) and
    (max-width: map-get($my-media-breakpoints, $_keymax) - 1px) {
    &{ @content; }
  }
}
```



```

/* alternative names for media-query mixins :
media-min or respond-below or media-breakpoint-down
, media-max or respond-above , media-between or respond-between
*/

%my-responsive-text {
  font-size:80%;
  //boucle possible sur my-media-breakpoints :
  @each $_key, $_value in $my-media-breakpoints {
    @include media-min($_key) {
      font-size: map-get($my-media-coeffs, $_key);
    }
  }
}

```

main-styles.scss

```

@import "_bases";

#contentXx { width: 100%;
  @include media-min('md') {
    font-size:100%; color:blue;
  }
  @include media-min('lg') {
    font-size:110%; color:red;
  }
}

#contentYy { width: 100%;
  @include media-max('md') {
    font-size:100%; color:blue;
  }
  @include media-between('lg','xl') {
    font-size:110%; color:red;
  }
}

#contentZz { @extend %my-responsive-text; }

```

==>

```
out-css/main-styles.css
#contentZz { font-size: 80%;}
@media screen and (min-width: 576px) {
  #contentZz { font-size: 100%; }
}
@media screen and (min-width: 768px) {
  #contentZz { font-size: 110%; }
}
@media screen and (min-width: 992px) {
  #contentZz { font-size: 120%; }
}
@media screen and (min-width: 1200px) {
  #contentZz { font-size: 130%; }
}

#contentXx { width: 100%;}
@media screen and (min-width: 768px) {
  #contentXx { font-size: 100%; color: blue; }
}

#contentYy { width: 100%;}
@media screen and (max-width: 767px) {
  #contentYy { font-size: 100%; color: blue; }
}
@media screen and (min-width: 992px) and (max-width: 1199px) {
  #contentYy { font-size: 110%; color: red; }
}
```