HTTP et HTML

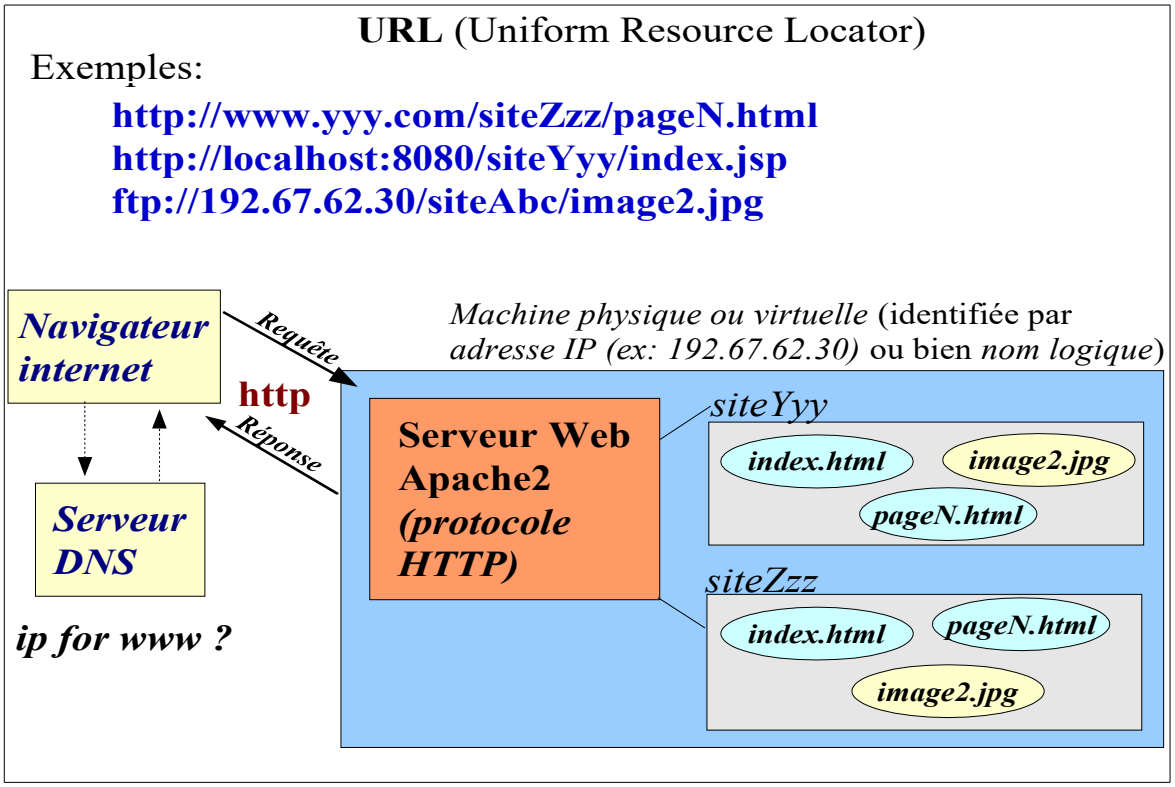
(avec CSS , Ajax)

Table des matières

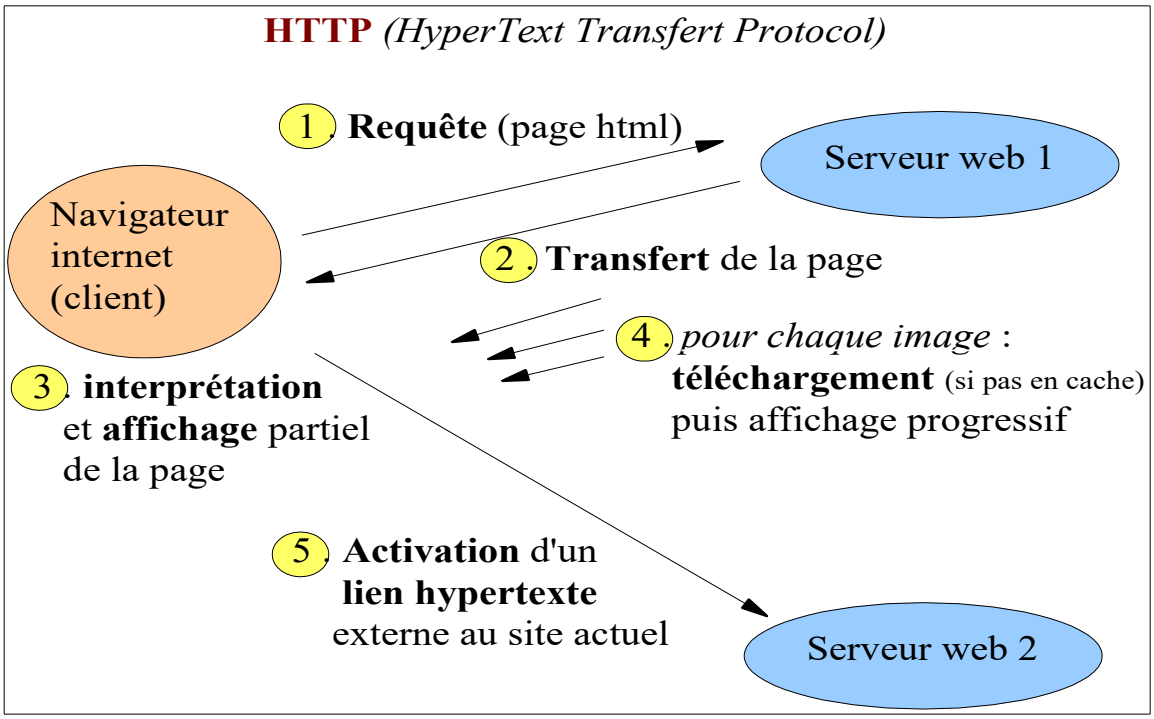
I - URL , HTTP, HTML (présentation)

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

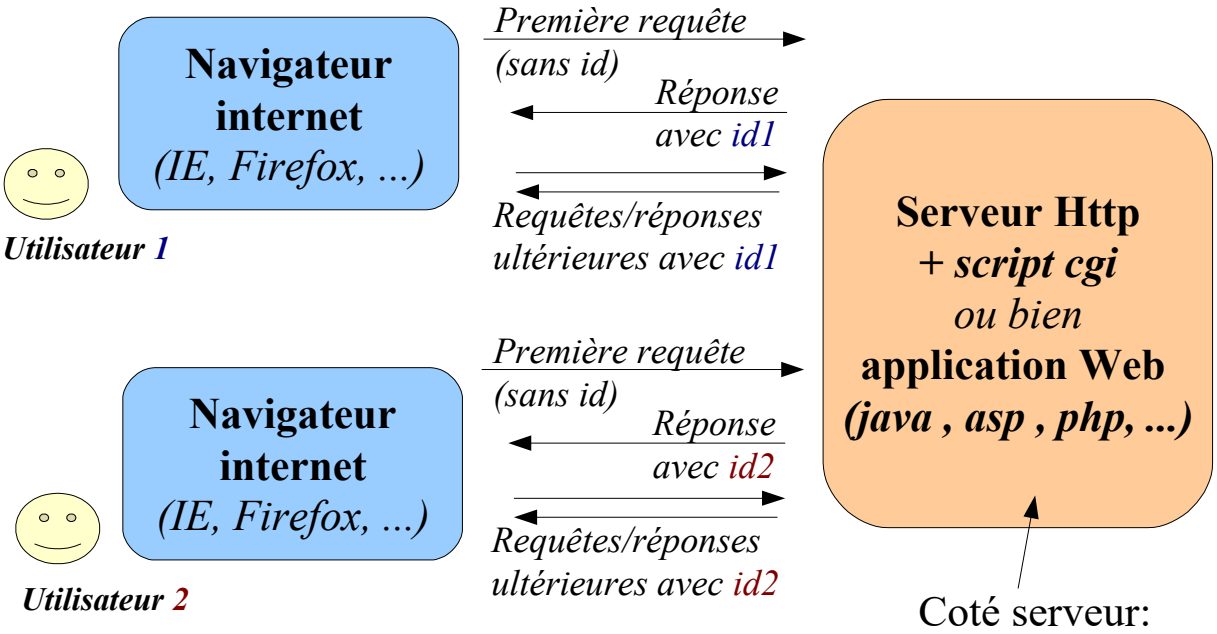
Réponse http

```
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)*

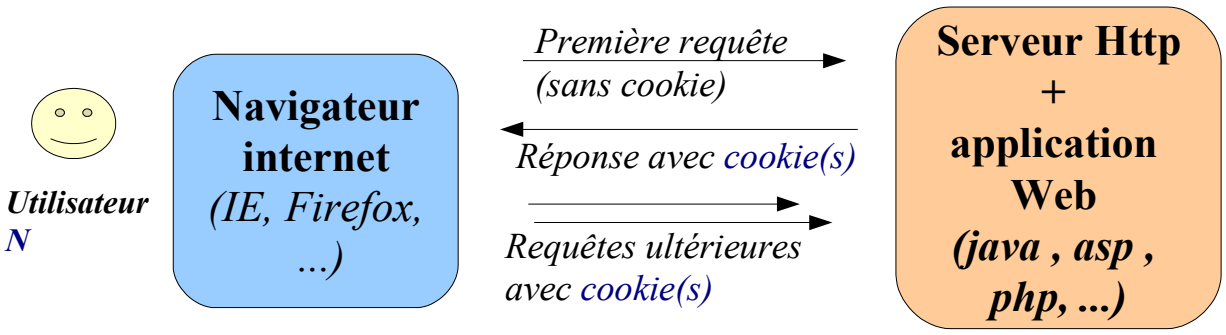
HTTP *(lien à moyen/long terme avec l'utilisateur)*



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 : alain
    Nom:     Therieur
    Age :    40
  </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 : rouge
     Nom: Therieur
     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:

```
<html>                                  --Marque début document
  <head>                                --début de l'entête
    <title>Titre de ma page</title>  --Titre du document (onglet)
  </head>                               --Fin de l'entête

  <body>                                -Début corps du document


  Description de ma page                --Description page


  </body>                               --Fin du corps
</html>                                   --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. Eléments de bases

Niveaux de titre => H1 (le plus gros) vers H6 (le plus petit)
exemple: **<H3>** Titre du paragraphe **</H3>**
** ** suite du paragraphe ** **

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) ****

**** Italique ****
**** 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 **<img>** 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 coté). 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 :

```
<img src="images/fp_logo.gif" width="114" height="43">
```

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> semetre1 </th> <th> semestre2 </th>
       <th> semetre1 </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	
semetre1	**semestre2**	**semetre1**	**semestre2**
1000	1600	1200	1800
1080	1700	1300	1900

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.htm"> 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.htm"> siteA </a>
```

Lien hypertexte sous forme d'image:

```
<a href="mapage.htm"> <img src="image1.gif"/> </a>
```

8. Les puces (listes d'élements)

li --> **L**iste **I**tem

ul --> **U**nordered **L**ist
ol --> **O**rdered **L**ist

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 :	**Paragraphe à puces rondes:**
1. ligne1 2. ligne2 3. ligne3	• ligne1 • ligne2 • ligne3
Paragraphe à puces numérotées I :	**Paragraphe à puces carrées:**
I. ligne1 II. ligne2 III. ligne3	• 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 le nom du script (cgi, servlet, page asp, ...) qui sera invoqué 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:

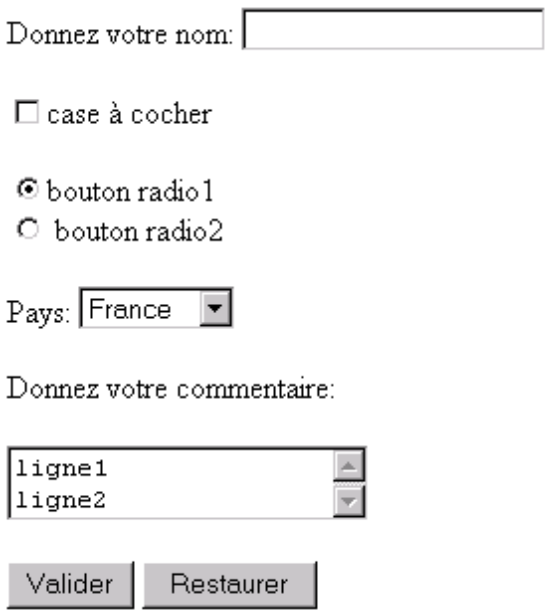
nom=therieur&prenom=alex

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 radio1    <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:



IV - Principaux apports de HTML5

V - Styles CSS

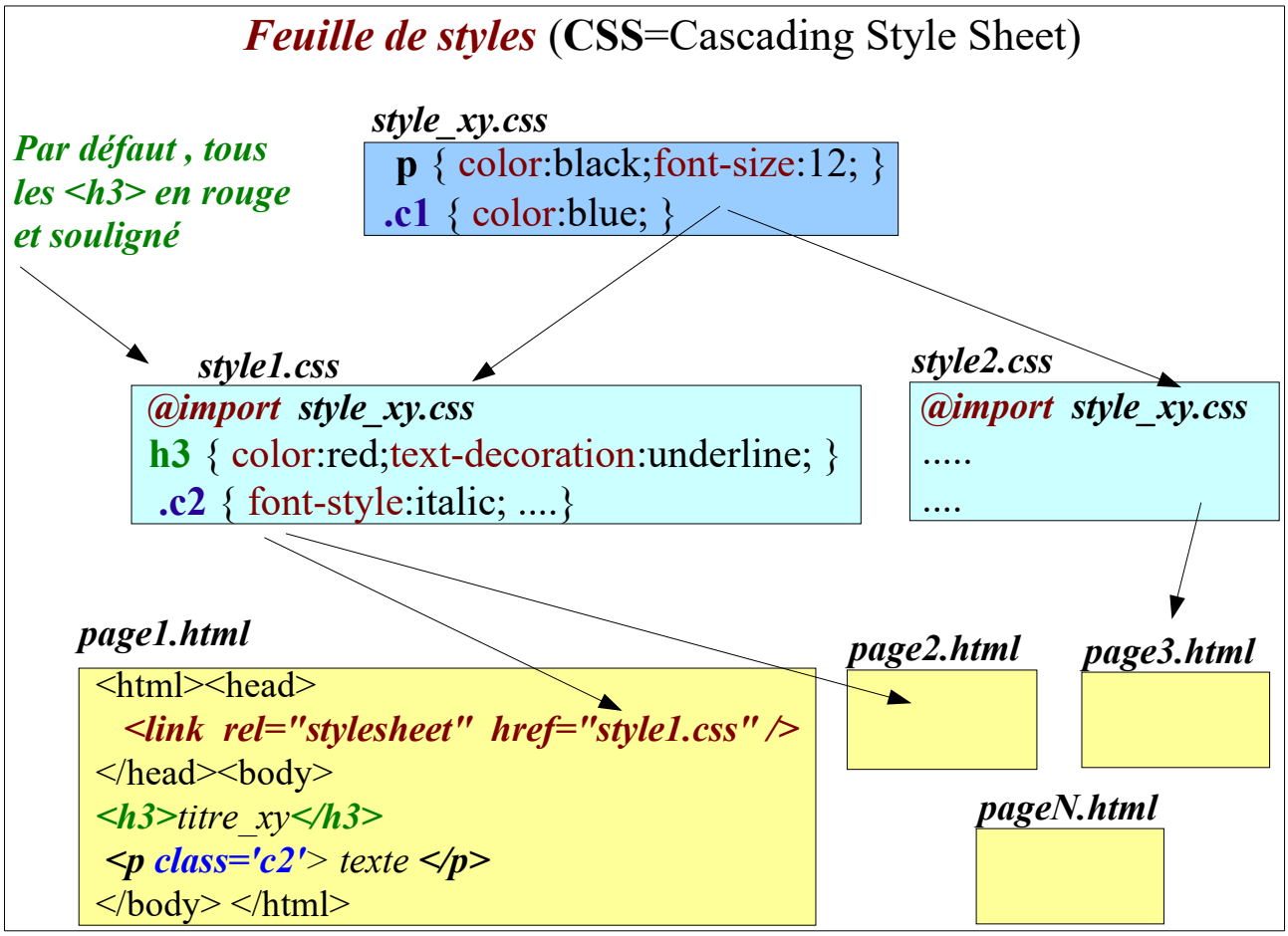
1. Feuilles de styles (CSS, ...)

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 (Cascading Style Sheet)

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 :

- Pour les **règles normales**, la **feuille de style de l'auteur est prioritaire** sur celle de l'utilisateur, elle même prioritaire sur la règle par défaut de l'application.
- Pour les **règles importantes**, **la feuille de style de l'utilisateur est prioritaire sur celle de l'auteur** *(différence avec CSS-1)* , elle même prioritaire sur la règle par défaut de l'application.
- 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 appliquer 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

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

Avec CSS niveau 2 , 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}

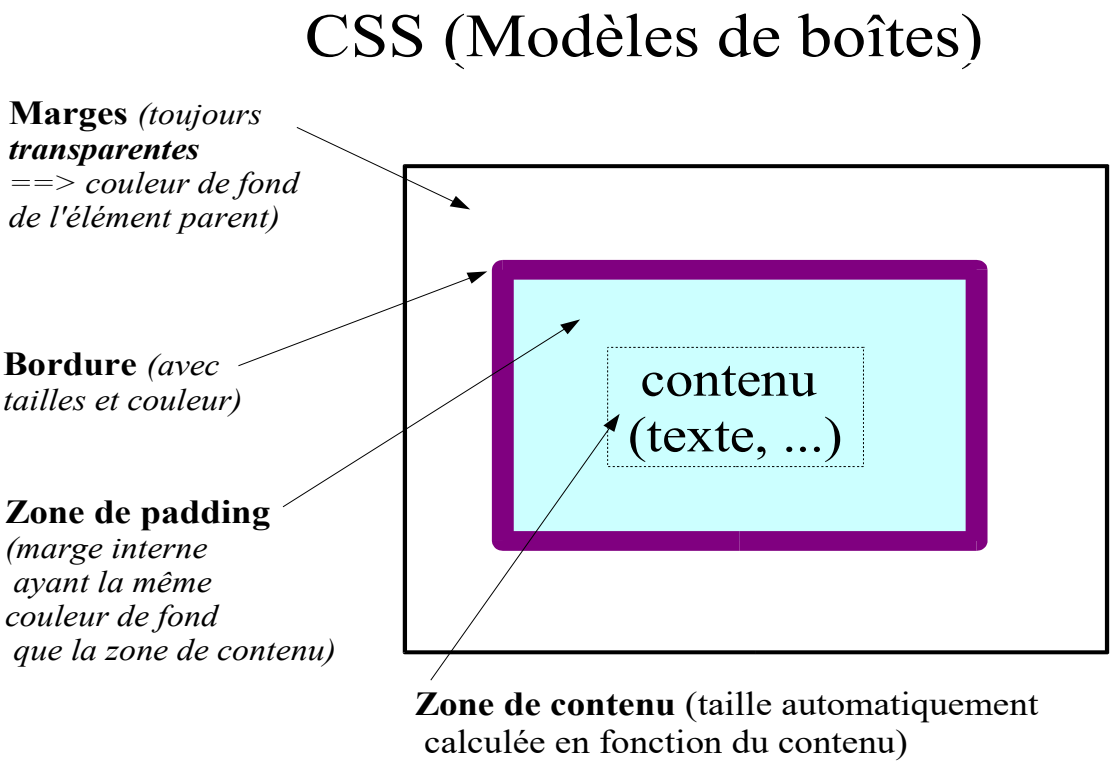
p:**first-letter** { propriété: valeur}

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

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

4. Principaux attributs de CSS

4.1. Modèle 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
%	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).

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 (*repeat*, *no-repeat*)
background-attachement	image de fond fixe ou défilante % contenu ? (*fixed*, *scroll*)

4.2. Propriétés d'affichage / positionnement

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

4.3. Mise en forme du texte

Principales propriétés:

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

VI - Javascript (essentiel)

1. Javascript (js)

Javascript est un ***langage interprété*** qui est essentiellement utilisé coté client de façon à ajouter des ***effets dynamiques*** au sein d'une page HTML .

JavaScript (js)

(langage interprété – essentiellement coté navigateur)

Page html interprétée (avec javascript)

```
<html>
  <head>
    <script>
       function afficherTexte(texte)
       { alert(texte);
       }
    </script>
   </head>
  <body>
     <form>
       <input type='button'
          onClick="afficherTexte('ok')" />
     </form>
  </body>
</html>
```

*Script incorporé dans la page html (peu conseillé) ou mieux encore placé dans un fichier annexe xxxx.**js***

***Déclenchement** de la fonction javascript "afficherTexte()" sur l'**événement "click"** .*

Référence vers un fichier de script séparé:

```
...
< SCRIPT LANGUAGE="JavaScript" SRC="util.js">
</SCRIPT>
...
```

Le ***fichier texte*** **util.js** pourra ainsi comporter un **ensemble de fonctions JavaScipt prêtes à être réutilisées dans différentes pages HTML**.

Principales fonctionnalités de javascript (js)

* **contrôles de saisie de premier niveau** (*numérique? entre mini et maxi ? valeur renseignée (non vide) ?*) directement pris en charge par le navigateur internet.

* **effets dynamiques** (affichage temporaire de *menus déroulants* , *boîtes de messages* , *développement [+] / contraction [-]* d'une partie d'une *arborescence* , ...)

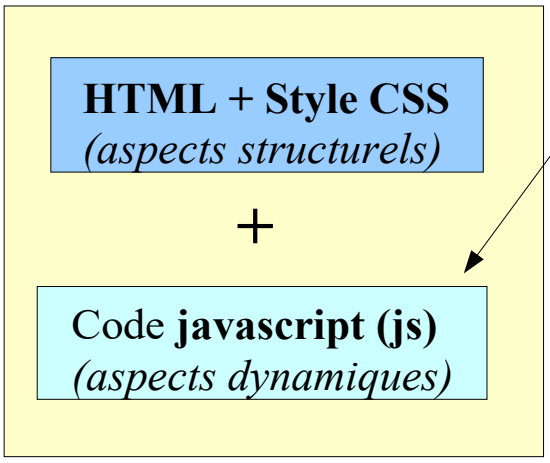
* **changement** dynamique d'un **style css** attaché à un élément html (*display="none" ou "true" , ...*)

* **ré-actualisation d'une partie du contenu html suite à une requête ajax** .

* **autres aspects divers** (plus ou moins gadgets).

Positionnement (et écueils) de javascript

Ensemble (DHTML) interprété coté navigateur *après avoir été généralement généré dynamiquement coté serveur (par code java ou php ou ...)*



Programmation délicate:
(langage très faiblement typé, *scripts devant idéalement être interprétables par tous les navigateurs*).
--> en pratique : code javascript à beaucoup tester ou bien à indirectement générer depuis un framework .

Convergence récente entre IE et FireFox sur le modèle objet (DOM) mais encore quelques différences sur la gestion des événements.

VII - DHTML & Ajax

DHTML
...

1. Ajax

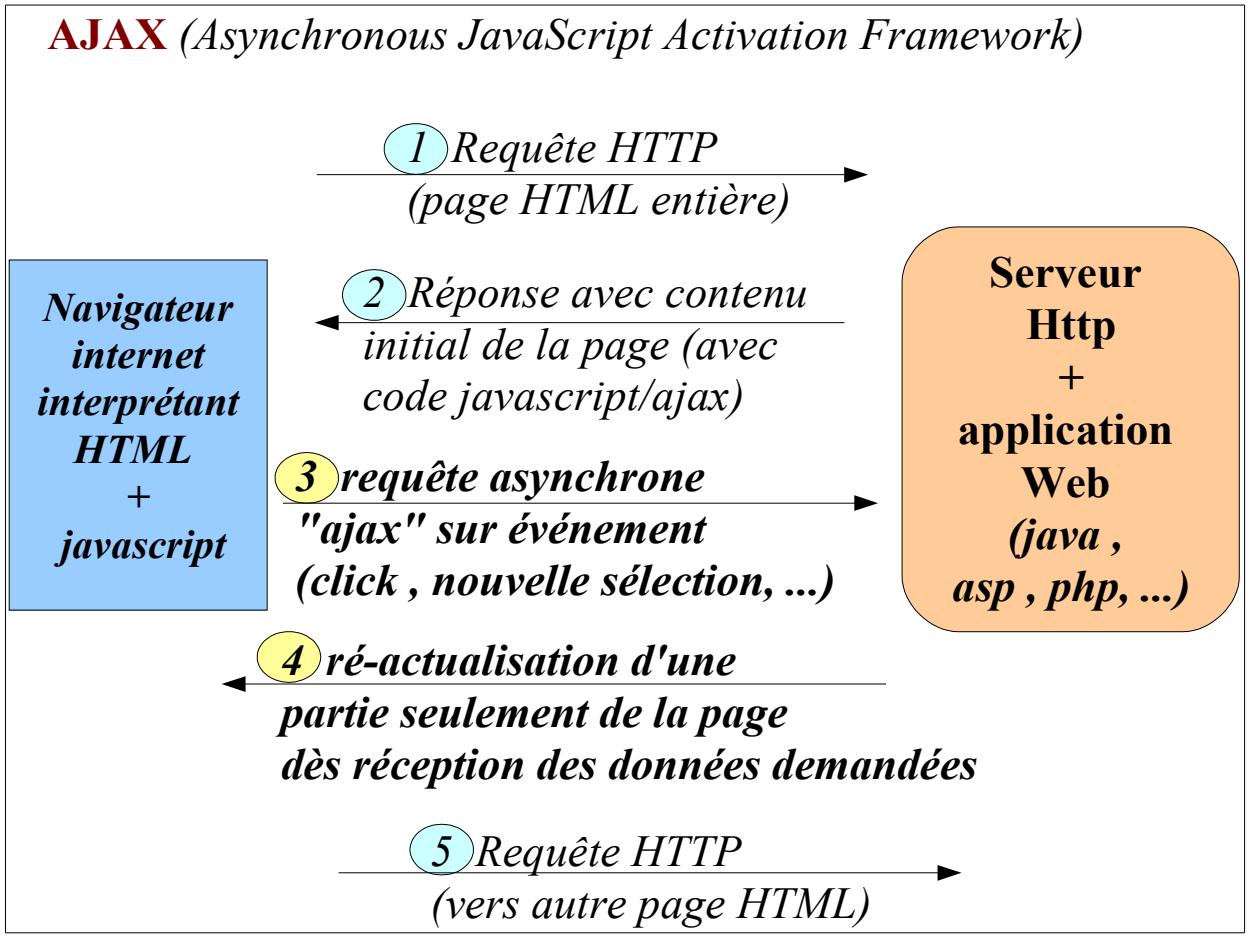
1.1. Principe

Le sigle ***AJAX*** correspond à ***Asynchronous Javascript Activation framework***

Cette technologie permet d'ajouter au sein d'une page HTML du code javascript (qui sera interprété coté client par un navigateur et) qui va:

- déclencher un appel HTTP invisible vers une entité coté serveur (ex: Servlet , ...)
- récupérer via une "***callback asynchrone***" les données de la réponse (Chaîne de caractères, données XML , ...)
- mettre en forme (afficher) ces données en modifiant (via l'api HTML-DOM) le contenu et/ou la structure de la page HTML courante.

Grand avantage: ces micros requêtes déclenchées via AJAX ne déclenchent pas une ré-actualisation complète de la page (pas de submit) ==> IHM globalement plus fluide et plus réactive ; optimisation du trafic réseau et des traitements coté serveur (on ne demande et véhicule que ce qui est nécessaire sans systématiquement régénérer et recharger des pages entières).



1.2. Principales utilisations d'ajax

Principales utilisations d'Ajax

* ***aide contextuelle / auto-complétion***
  (ex: récupérer la liste des villes commençant par "P")
* ***actualisation d'une liste "détails" dès nouvelle sélection dans une liste "principale"***
  (ex: récupérer tous les communes du département "92")
* *Toute autre ré-actualisation partielle nécessitant des données coté serveur (ex: libellé selon code , validation partielle, ...).*

Ajax et SPA (Single Page Application)

*Tous les accès aux données sont effectués par des requêtes ajax

* La réactualisation d'une partie de la page s'effectue souvent via jQuery ou un framework plus évolué (ex : Angular)

Ajax est essentiellement utilisé pour effectuer des appels vers des web services REST (http).

La réponse à un appel ajax peut être très variée (partie HTML , document/données XML , données JSON, …)

Un appel Ajax est par défaut soumis à des restrictions au niveau des noms de domaines.
Pour pouvoir effectuer un appel ajax vers un autre domaine, il faut que celui-ci ait paramétré une autorisation "CORS" .

1.3. XMLHttpRequest (xhr)

Tous les navigateurs internet (pas trop anciens) offrent un objet prédéfini et standardisé "XMLHttpRequest" permettant d'effectuer des requêtes "ajax".

Requête ajax en pur javascript (sans framework) :

```
function makeAjaxRequest(callback) {
        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function() {
                if (xhr.readyState == 4 && (xhr.status == 200 || xhr.status == 0)) {
                        callback(xhr.responseText);
                }
        };
        var ajaxUrl="handlingData.php" ; //ou ...
        xhr.open("GET",ajaxUrl , true);
        xhr.send(null);
}

function readDataCallBack(sData) {
        //var myP=document.getElementById('myP');
          var myP=document.querySelector('#myP');
          myP.innerHTML = sData;
}

makeAjaxRequest(readDataCallBack);
```

XHR (XMLHttpRequest) est une api de bas niveau.

Pour plus d'efficacité , un appel ajax sera souvent déclenché via une surcouche , bibliothèque ou framework javascript (ex : jQuery , Angular , ….)

VIII - Compléments pour HTML (Object, SVG, ...)

1. Add on / plugins , objets (applet, activeX) , flash,...

Il est possible d'insérer dans une page HTML des objets qui seront gérés par des technologies annexes (ex: activeX Microsoft , appet Java , animation flash, ...).
Ces "objets" seront visualisés comme des sous fenêtre avec:

- un contenu dynamique (animation, vidéo , ...)
- des interactions possibles avec l'utilisateur

C'est un peu comme si on incorporait une "mini application" dans une partie d'une page HTML.

1.1. Insertion d'un applet Java (ancienne syntaxe)

1.2. Insertion d'un objet (applet ou activeX ou ...)

1.3. Insertion d'une animation "flash"

2. SVG

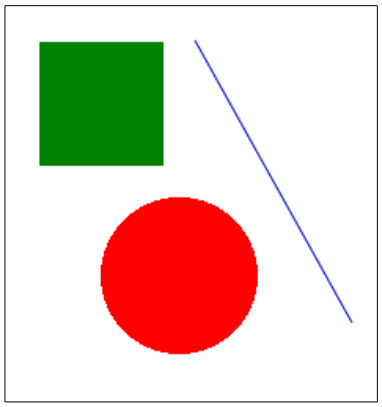
Signifiant ***Scalable Vector Graphics*** , **SVG** est un langage dérivé d'XML qui permet d'encoder des **images vectorielles en 2D** .

Correspondant au type MIME "***image/svg+xml***" , un fichier SVG ne peut s'afficher qu'au sein d'un navigateur adéquat (ex: IE + plugin ***SVG-Viewer*** d'adobe) .

Principales fonctionnalités de SVG:

- Coder des figures, schémas et diagrammes de façon assez compacte sans pour autant sacrifier la qualité (l'image reste nette quelque soit le facteur d'agrandissement).
- Simple à encoder par programmation (la génération d'un fichier SVG est quasiment aussi simple que celle d'un fichier texte).
- Possibilité d'animation simple en rajoutant du code JavaScript

Exemple:



peut être encodé de la façon suivante:

scene.svg

```
<svg  width='500'  height='350' >
        <rect  x='50'  y='50'  width='80'  height='80'  style='fill:green' />
        <circle  cx='140'  cy='200'  r='50'  style='fill:red' />
        <line  x1='150'  y1='50'  x2='250'  y2='230'  style='fill:blue;stroke: mediumblue;' />
</svg>
```

NB:

Une image SVG peut éventuellement être intégrée au sein d'une page HTML de la façon suivante:

```
<object align="middle" data="scene.svg"
      border="0" name="SVGEmbed3"
      width="300" height="300"
      type="image/svg-xml">
</object>
```

ANNEXES

IX - Annexe – HTML (points avancés, divers)

X - Annexe – Bibliographie, Liens WEB + TP

1. Bibliographie et liens vers sites "internet"

2. TP