INTRODUCTION AU WEB, HTTP, HTML ET CSS

SR10

Ahmed.lounis@hds.utc.fr

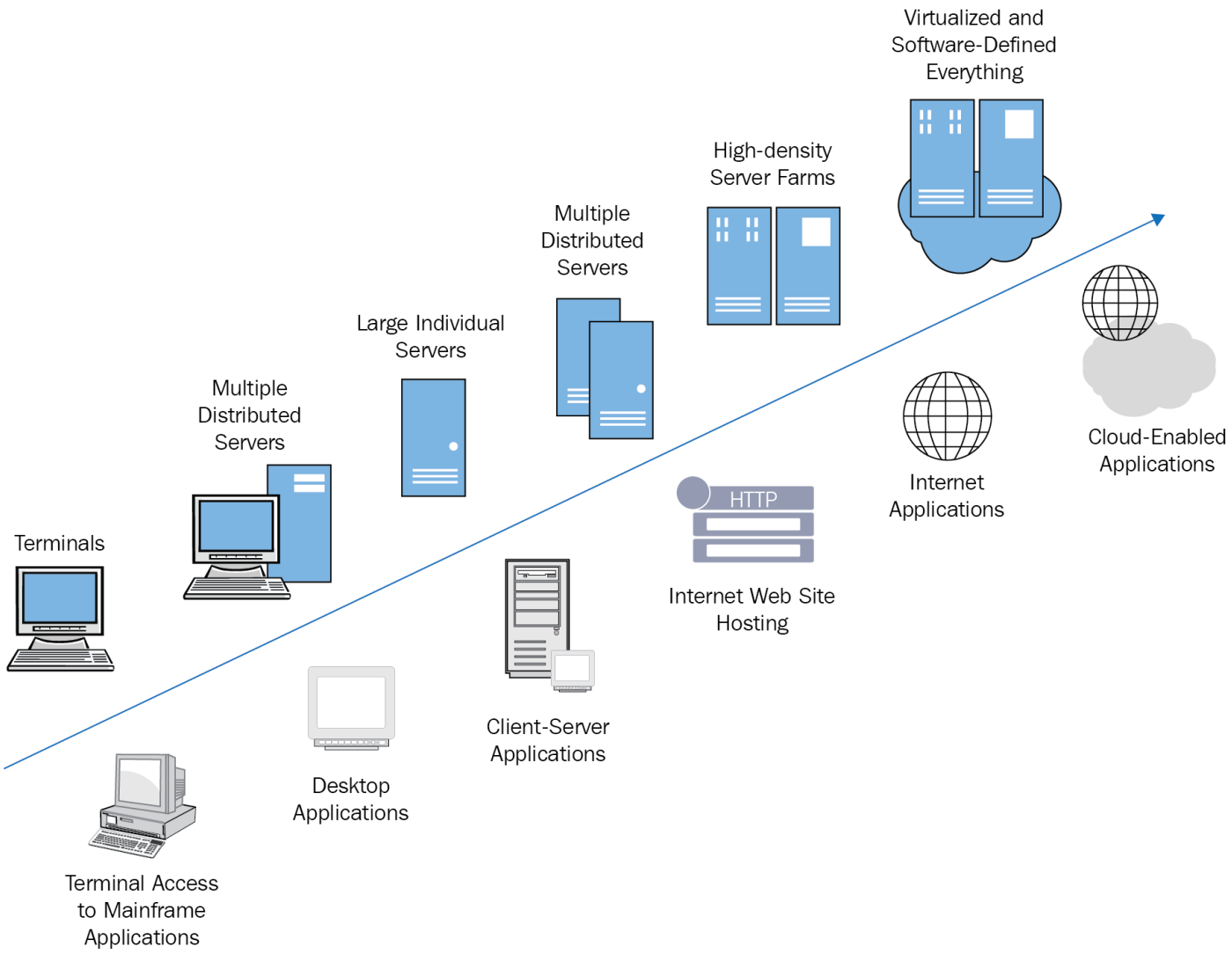
Web, HTTP et HTML

- Le Web : présentation générale
- L'URL et le protocole HTTP
- HTML, exemples , Feuille de style CSS
- Consommation énergétique

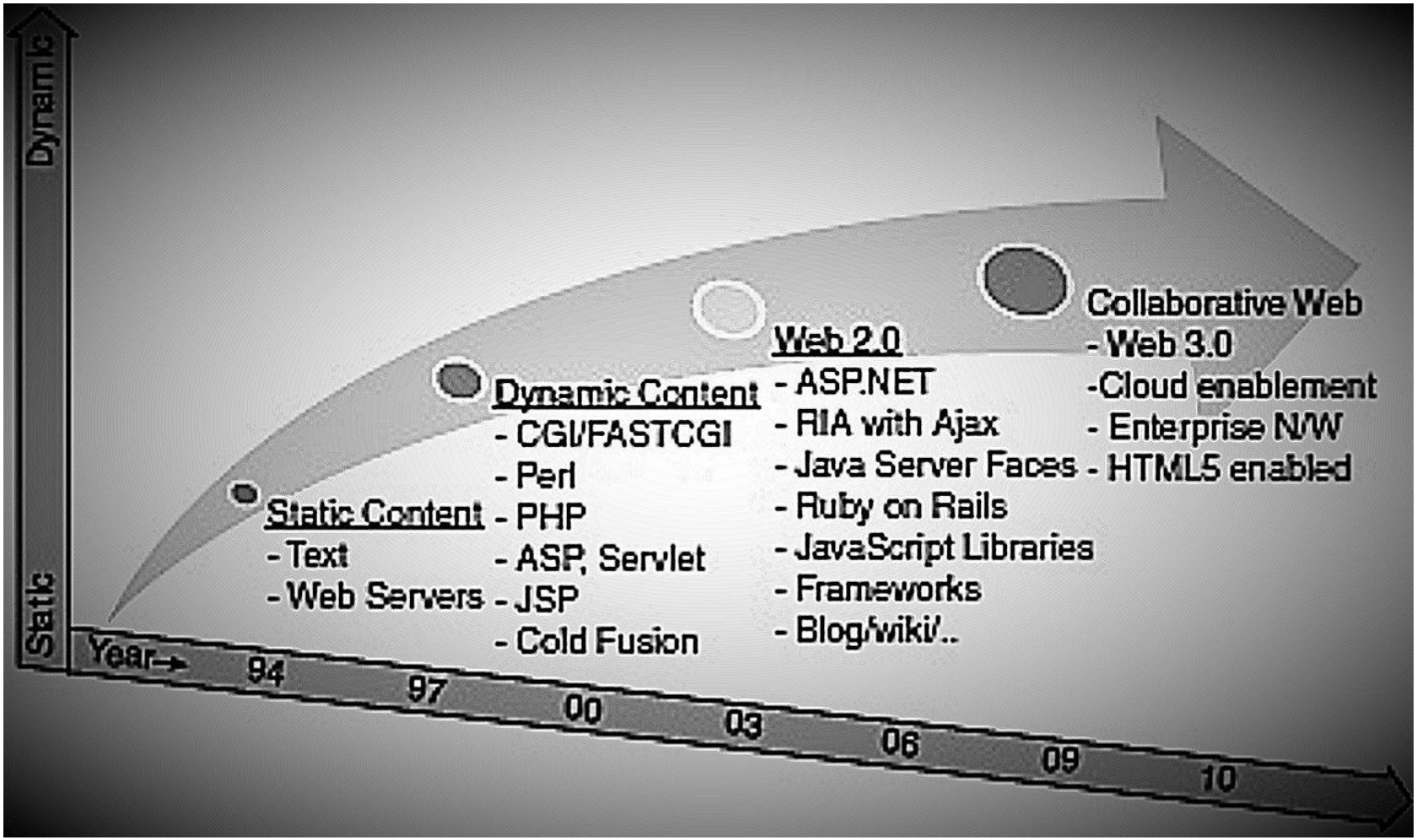
Introduction générale Web

- **Web = système d'information hyper-média réparti :**
  - **texte, images, images animées, son …**
- **Les informations sont stockées sur des serveurs :**
  - **www.utc.fr**
  - **www.google.fr**
  - **…**
- **Pour interroger les serveurs on utilise des clients (navigateur web ou browser) tels que Mozilla Firefox, Google Chrome, Safari, Opera…**
- **Ce sont ces clients qui traduisent nos demandes en requêtes HTTP pour dialoguer avec les serveurs**

Evolution des architectures des applications



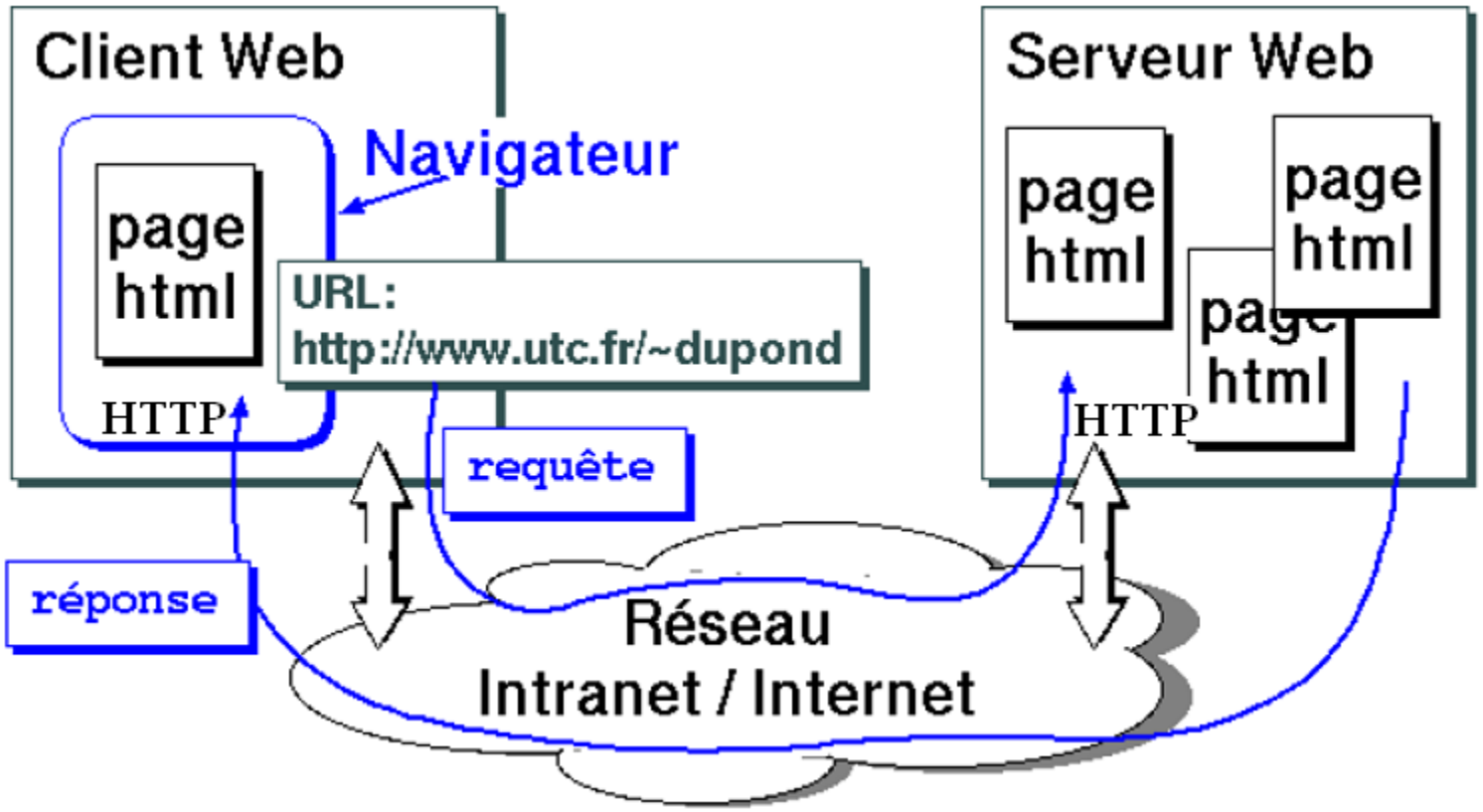
Tendances de l'évolution du web



Introduction générale Web

- Web est composé de trois grandes briques:
  - HTML (**HyperText Markup Language**)
  - URL (**Uniform Resource Locator)**
  - HTTP (**HyperText Transfer Protocol**)

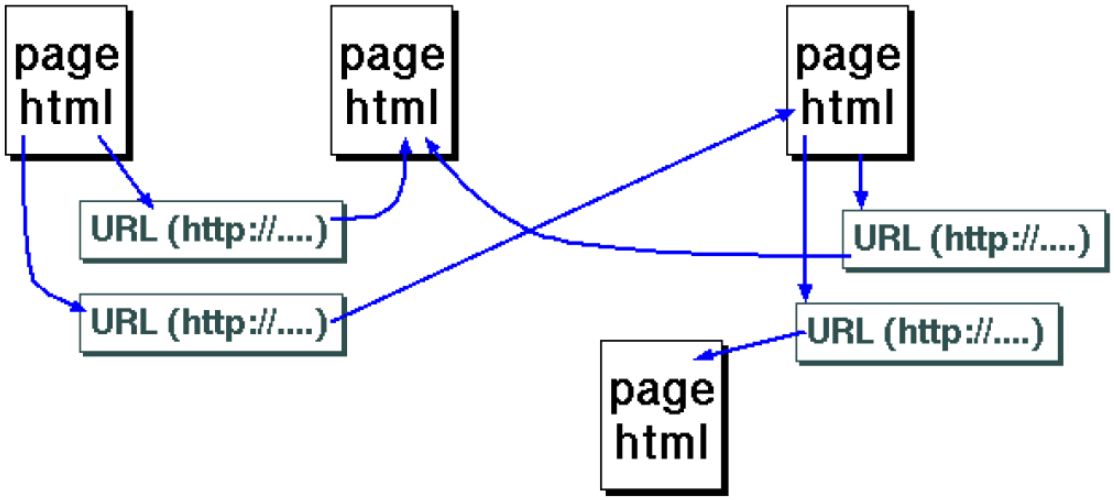
Cheminement d'une connexion (I)



requête passe par socket TCP/IP (communication des données dans le réseau avec une fiabilité max)

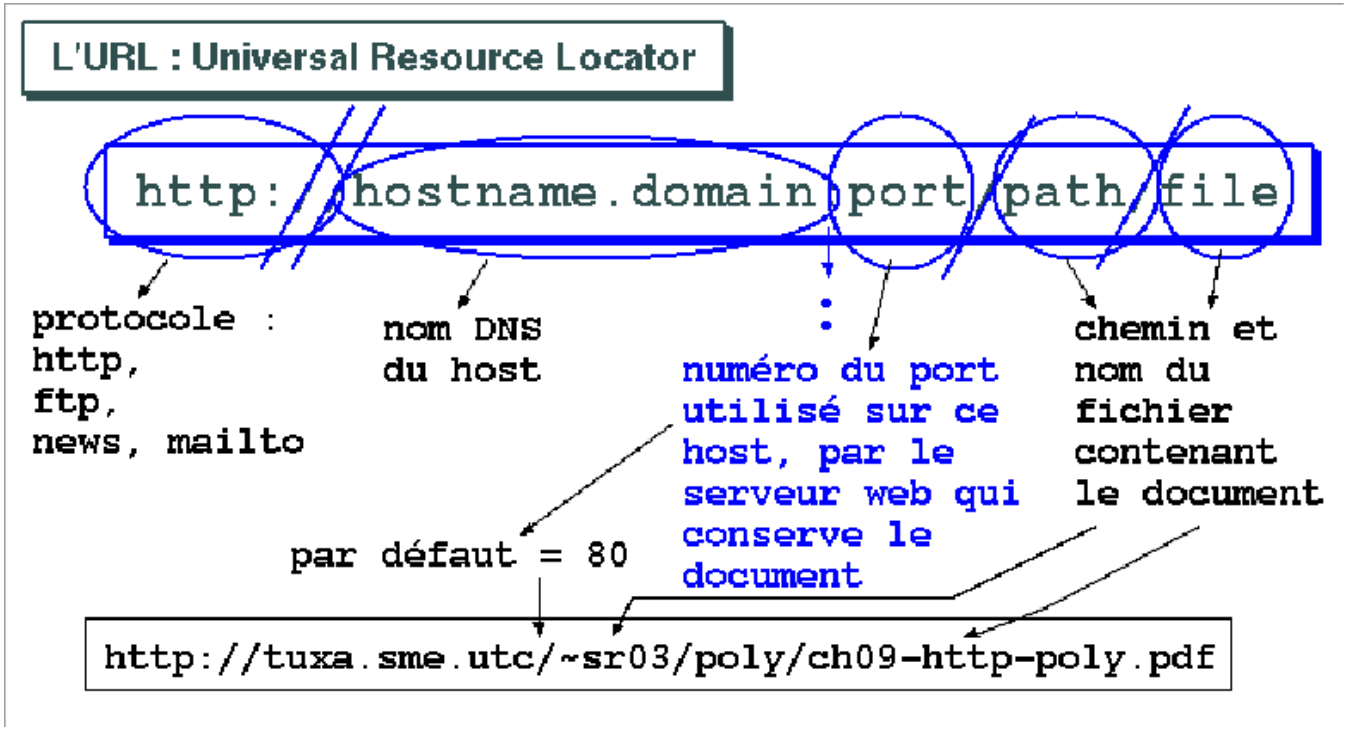
Web: aspects généraux
navigation entre les pages web

- WEB : relier les documents à travers un réseau mondial (1989)
- Les pages web sont écrites par HTML = *Hypertext Markup Language*
- Hypertext : ensemble de documents reliés par des « hyperliens »



Partager les documents en les reliant entre eux et en cachant la méthode d'accès et la localisation.

URL, URI, URN…



URL : Uniform Ressource Locator

URN : Uniform Ressource Name

URI : Uniform Ressource Identifier (URL+URN)

URI : identifiant de ressource (URL ou URN ou combinaison des deux)
- URL : identifier et localiser une ressource sur le réseau
- URN : nommer ressource de manière unique sans indiquer son emplacement

Web et HTTP

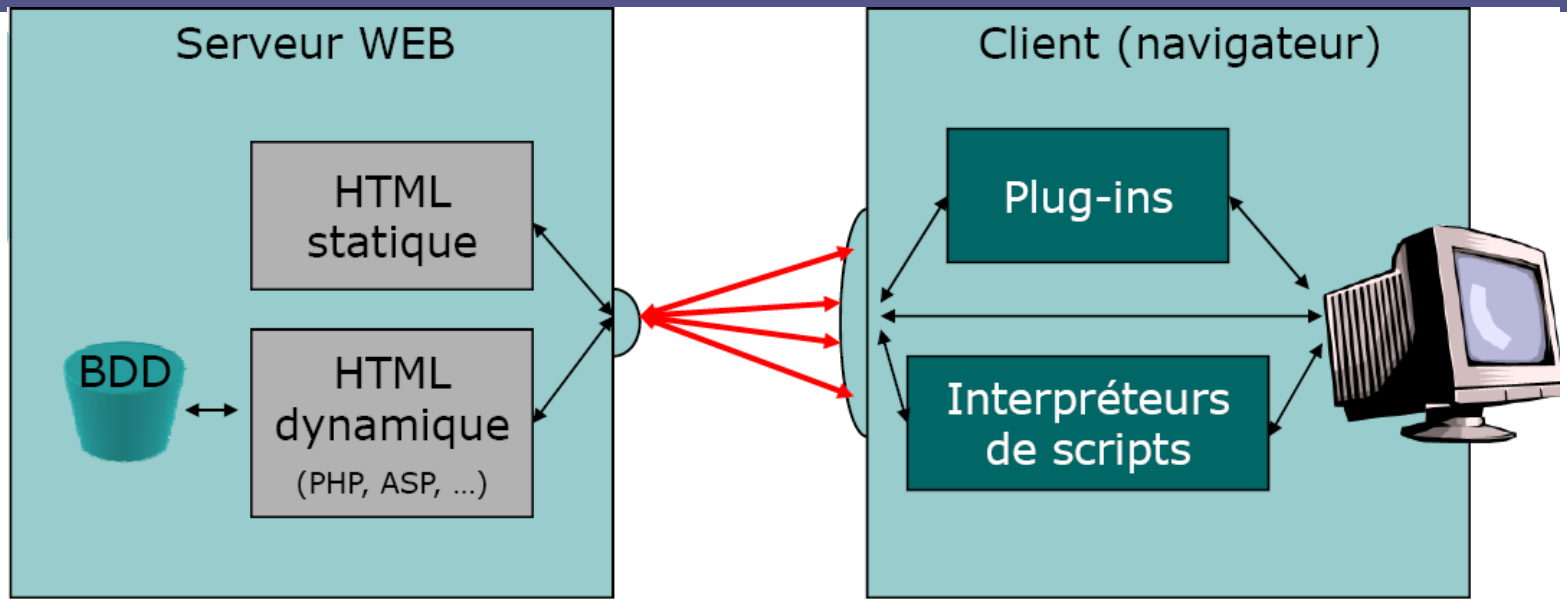
- **Web fonctionne selon le modèle client-serveur**
  - **Le client envoie des requêtes au serveur :**
  - **demande de transfert de fichiers**
  - **exécution de programmes sur le serveur**
  - **mise à jour de fichier**
  - **…**
- **Utilisation du protocole HTTP**
  - **définit le syntaxe utilisé pour les échanges entre client et serveur Web**
- **Les objets manipulés sont repérés par leur URL**

HTTP

- HTTP est un protocole « sans mémoire »
  - pas de notion de session avec HTTP
  - chaque communication est indépendante.

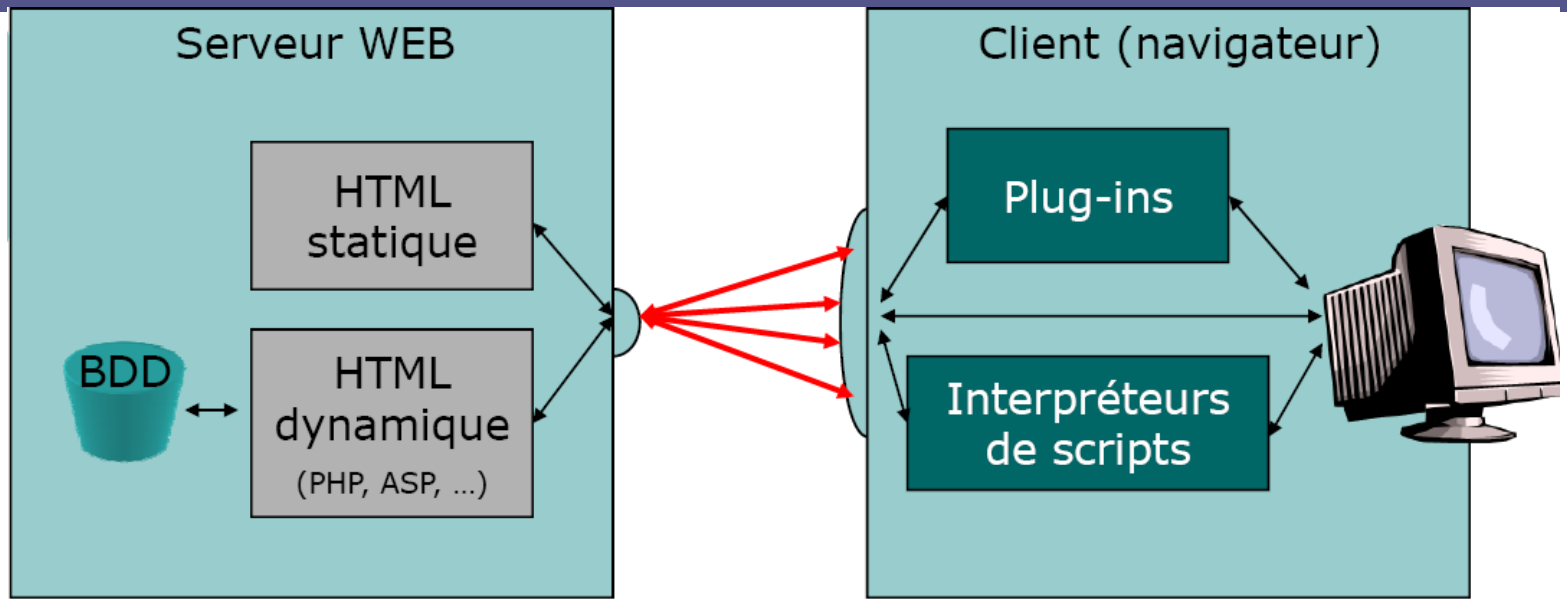
- Cookies
  - informations stockées au niveau du client, concernant la navigation.
  - exemple :
    - Identification et authentification : création d'un cookie
    - revenir sur la page : lecture du cookie
    - déconnexion ou fin de la durée de vie du cookie : destruction

HTTP



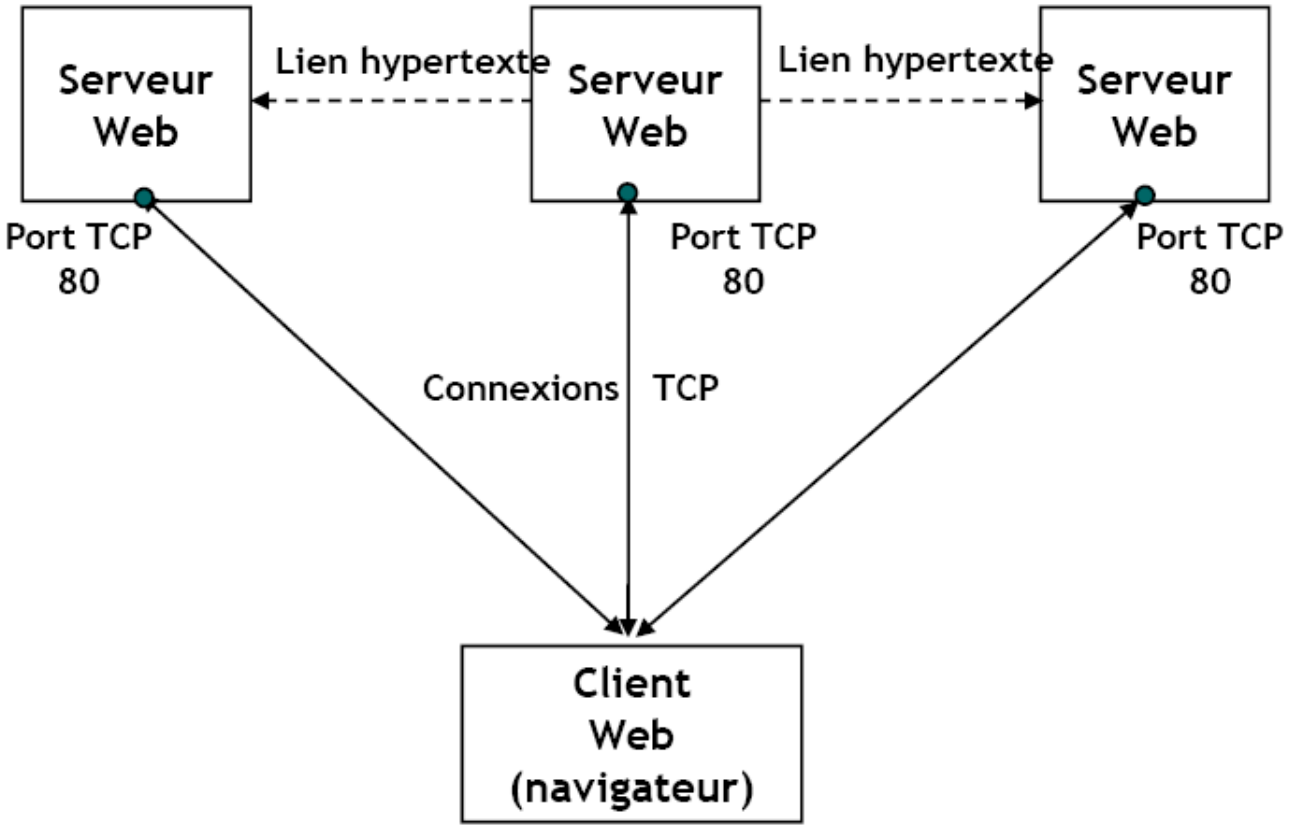
- Côté serveur
  - HTML statique
  - HTML dynamique
  - calcul à partir d'éléments que le client a transmis au serveur dans sa requête.
  - informations issues d'une base de données mise à jour par un moyen quelconque.
    - création du code HTML à envoyer au client!

HTTP



- Côté client
  - contrôle de la validité des informations saisies dans un formulaire, avant de les envoyer au serveur
  - traitement local de certaines informations pour afficher un résultat.
    - animations…

HTTP : mécanismes



HTTP : historique

- Ce protocole est normalisé (RFC 1945, RFC 2616, RFC 2068 et RFC 75401) qui en définissent quatre versions :
  - 0.9 qui est le protocole définit à l'origine par Tim Berners-Lee,
  - 1.0 qui apporte de très nombreuses fonctionnalités (typage des documents, codage du contenu, identification, ...)
  - 1.1 maintenant la plus avancée qui permet de réaliser différentes négociations, d'obtenir des **canaux persistants**, pipeliner les requetes etc.
  - 2.0 (pourcentage d'adoption >=45%) compatible avec les applications HTTP 1.1 et conserve la majorité de la syntaxe de HTTP 1.1. Le changement touche la manière dont la donnée est segmentée et transportée entre le client et les serveurs afin d'optimiser les performances (compression des entêtes, Preload API, Serveur Push, etc.)

En cours : http 3.0 basé sur UDP (internet-Draft chez IETF)

canal persistant : fonctionnalité qui permet d'utiliser une seule connexion TCP pour envoyer et recevoir plusieurs requêtes et réponses HTTP, au lieu d'ouvrir une nouvelle connexion pour chaque paire de requête-réponse

HTTP

MIME : standard de l'Internet qui indique la nature et le format d'un document, fichier, ou ensemble de données (type de données prédéfini)

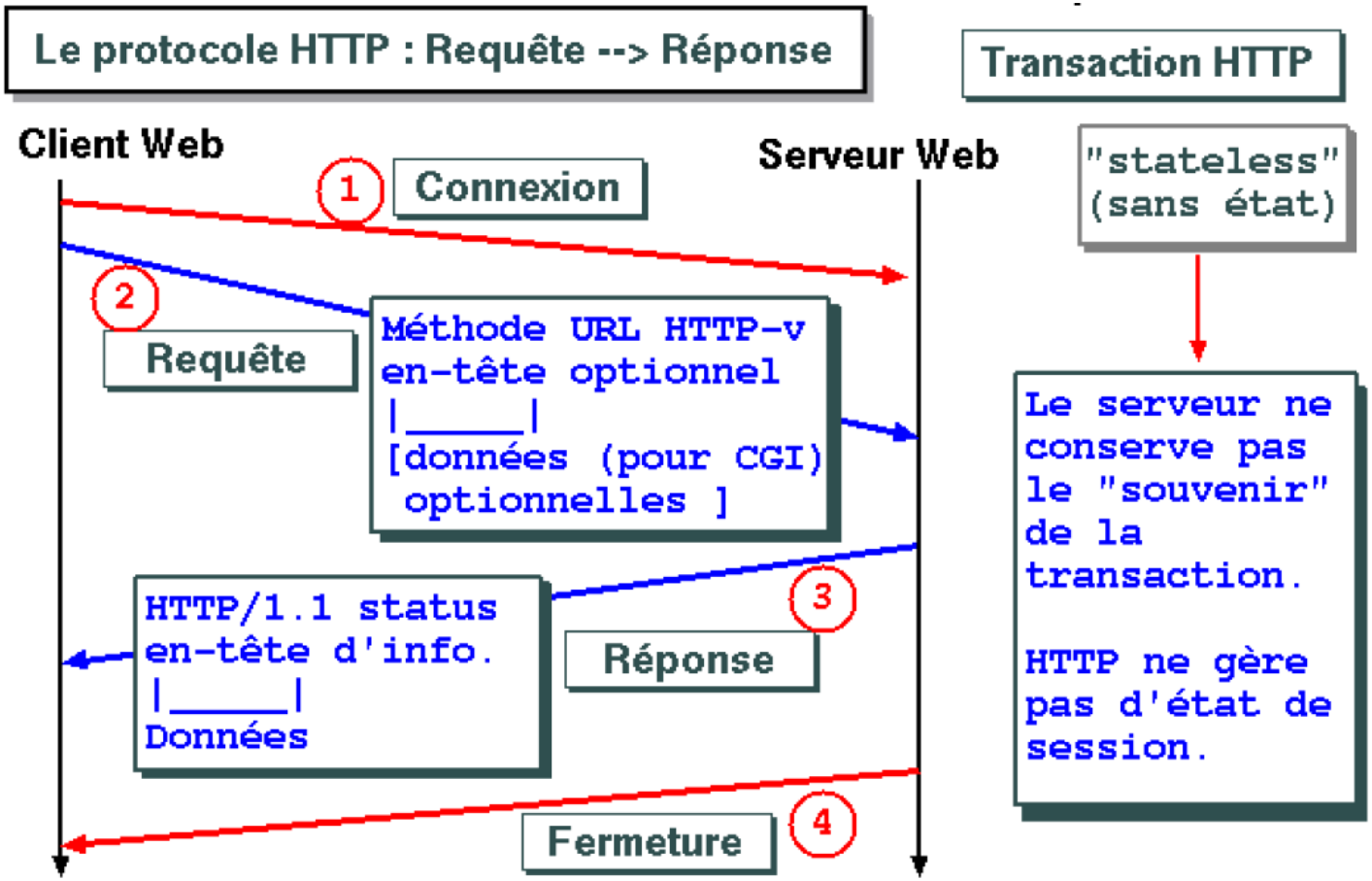
- HTTP transporte des données de type MIME.
  - codage données binaires → texte, images, documents, etc…

- Le navigateur (le client) communique avec le serveur Web à travers une ou plusieurs connexions TCP.

Préciser n° port si différent : http://192.182.1.1:3000/

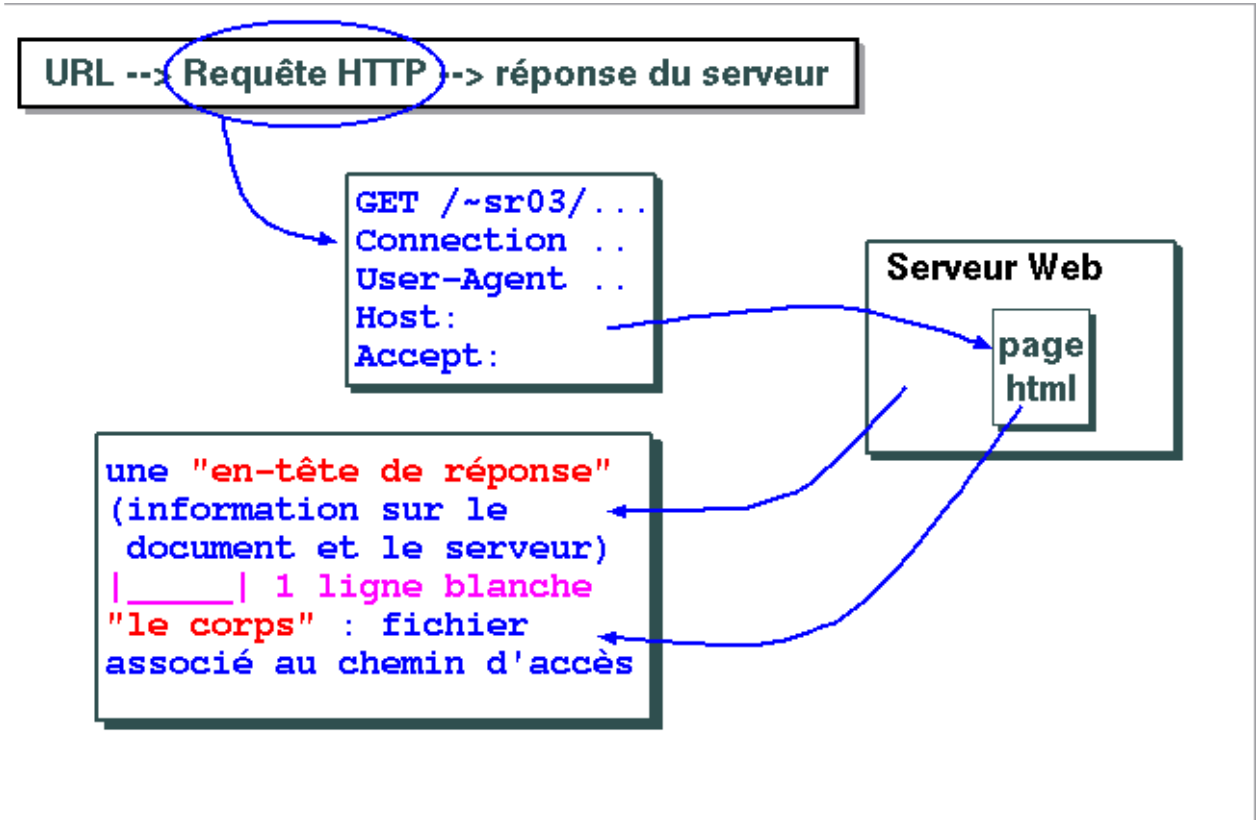
- Le port de notoriété publique d'un serveur Web est 80
  - HTTP est un protocole très simple:
    - le client établit une connexion TCP avec le serveur
    - il envoie une requête et récupère la réponse du serveur
    - le serveur marque la fin de la réponse en fermant la connexion TCP

HTTP : comment ça marche?



Cycle de vie d'une connexion HTTP

HTTP



Il existe deux types de messages HTTP :
les requêtes et les réponses

HTTP

- L'URL est fourni au navigateur,
- Le navigateur analyse l'URL et envoi une requête au serveur web décrit par cet URL.
- Par exemple : http ://tuxa.utc.fr/sr10/td1.html provoque l'envoi au serveur de :

```
GET /sr10/td1.html HTTP/1.1
Connection : Keep-Alive
User-Agent : Firefox
Host : tuxa.utc.fr
Accept-Charset : iso-8859-5, unicode-1-1
Accept : image/gif, image/x-xbitmap, image/jpeg, */*
```

Requêtes HTTP 1.0

- La requête HTTP :
  - *ligne de requête*
  - *en-têtes (0 ou plus)*
  - *<ligne blanche>*
  - *corps de la requête (présent uniquement pour un POST)*

- Le format de la ligne de requête est :
  - *requête URL-voulue HTTP-version*

GET /sr10/td1.html HTTP/1.0

Requêtes HTTP 1.0

- Il existe trois requêtes distinctes :
  - la requête **GET**, qui renvoie le contenu de l'URL demandée (si elle existe)
  - la requête **HEAD**, où le serveur ne renvoie que l'en-tête (permet notamment de vérifier l'accessibilité)
  - la requête **POST**, qui permet d'envoyer des données au serveur (formulaires…)

Les requêtes simples

HTTP : Requêtes simples

GET

Le client veut juste
demander un document:
GET /chemin/doc.htm HTTP/1.1

HEAD

Le client veut juste
de l'information sur
un document, mais pas
le document lui-même :
HEAD /chemin/doc.htm HTTP/1.1

Le client
veut
seulement
récupérer
de
l'information

Les requêtes avec envois d'information

HTTP : Requêtes avec envoi d'informations

POST

On fournit de l'information au serveur (par ex. issue d'un formulaire) par:

```
POST /chemin/cgi-bin/form.php HTTP/1.1
en-tête
|_____| ligne blanche
var1=data1&var2=data2&....
```

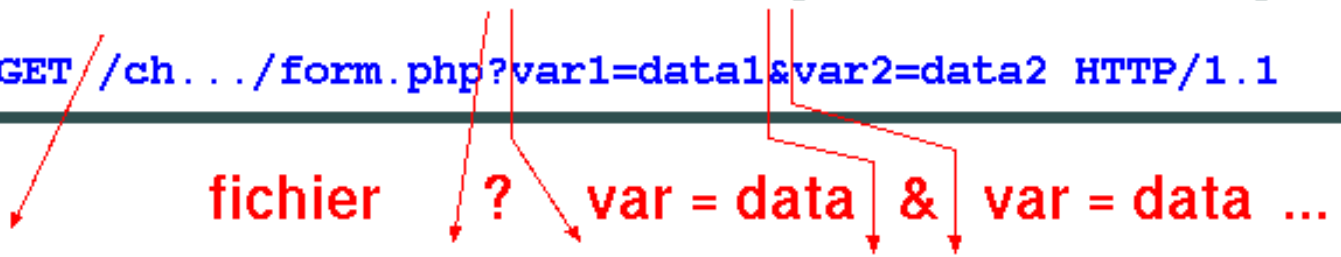
GET + infos

GET + Format "URL encodé"

<FORM>...method=GET>

l'info. saisie dans la forme est passée au serveur par:

```
GET /ch.../form.php?var1=data1&var2=data2 HTTP/1.1
```



Réponses

- Une réponse HTTP est de la forme :
  - ligne de statut
  - en-têtes (0 ou plus)
  - <ligne blanche>
  - corps de la réponse (il contient le document demandé)

- La ligne de statut est de la forme :
  - HTTP-version code-réponse phrase-réponse

Les en-têtes

- Les requêtes et les réponses peuvent contenir un certain nombre de champs d'en-tête
- Une ligne blanche permet de séparer ces derniers du document
- Un en-tête se divise en un nom de champ, un caractère ':', un espace et une valeur de champ
  - Les en-têtes sont classés en quatre catégories
    - Les en-têtes généraux HTTP;
    - ceux qui s'appliquent aux requêtes;
    - ceux qui s'appliquent aux réponses;
    - ceux qui décrivent le corps du message.

Les en-têtes

En-têtes Généraux HTTP

- Connection = listes d'option (*close pour terminer une connexion*).
- Date = date actuelle.
- Content-type = type MIME des données envoyées
- Content-length = longueur des données après les en-têtes

En-têtes de requêtes client

- En-têtes de requêtes client HTTP
  - Accept = type MIME visualisable par l'agent
  - Accept-Encoding = méthodes de codage acceptées
    - compress, x-gzip, x-zip
  - Accept-Charset = jeu de caractères préféré du client
  - Accept-Language = liste de langues
    - fr, en, …
  - Authorization = Identification du browser auprès du serveur
  - Cookie = cookie retourné
  - Content-Length = Longueur du corps de la requête

En-têtes de requêtes client (suite)

- From = adresse email de l'utilisateur
  - rarement envoyé pour conserver l'anonymat de l'utilisateur
- Host = spécifie la machine et le port du serveur
  - un serveur peut héberger plusieurs serveurs
- If-Modified-Since = condition de retrait
  - la page n'est transférée que si elle a été modifiée depuis la date précisée. Utilisé par les caches
- Referer = URL d'origine
  - page contenant l'ancre à partir de laquelle a été trouvé l'URL.
- User-Agent = donner des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation.

Exemple de requête

Exemple de demande client. Soit l'URL: http://tuxa.sme.utc/sr03/td1.html
Le message envoyé au serveur est:

GET /sr03/td1.html HTTP/1.1 (1)
*Accept: image/gif, image/x-xbitmap, image/jpeg, */* (2)*
Accept-Language: fr (3)
Accept-Encoding: gzip, deflate (4)
User-Agent: MSIE 7.01 (5)
Host: tuxa.utc.fr (6)
Connection: Keep-Alive (7)

***1-** Le client demande au serveur (GET) le document dont la localisation est /sr03/td1.html . Le client indique sa version de protocole (HTTP1.1)*
***2-** Le client indique au serveur quels sont les types de documents qu'il accepte.*
***3-** Le client indique sa langue préférée (français.)*
***4-** Le client indique qu'il sait traiter une réponse compressée par gzip et/ou deflate*
***5-** Le client s'identifie comme étant Microsoft Internet Explorer 7.01,*
***6-** Le client fournit le nom du serveur vis-à-vis de lui-même.*
***7-** Le client demande au serveur de conserver la connexion ouverte. Il s'agit d'une connexion TCP et non d'une connexion applicative.*

Entêtes des réponses

- Set-Cookie = créer ou modifie un cookie sur le client
- Allow = méthodes autorisées pour l'URI
- Last-Modified = date de dernière modification du doc.
  - utilisé par les caches
- Content-Length = taille du document en octet
  - utilisé par le client pour suivre la progression des chargements
- Content-Location : URI de l'entité
- Age = ancienneté du document en secondes
- Proxy-Authenticate = système d'authentification du proxy
- Retry-After = date ou nombre de secondes pour un nouvel essai en cas de code 503 (service unavailable)
- …

Les codes de réponses

- La première ligne de la réponse d'un serveur Web s'appelle la ligne de statut
- Elle commence par la version du protocole HTTP; elle est suivie d'un code numérique de réponse de trois chiffres

  - 100-199 Informationnel
    - 100 : Continue (le client peut envoyer la suite de la requête), ...
  - 200-299 Succès de la requête client
    - 200: OK, 201: Created, 204 : No Content, ...
  - 300-399 Redirection de la Requête client
    - 301: Redirection, 302: Found, 304: Not Modified, 305 : Use Proxy, ...
  - 400-499 Requête client incomplète vérifier que URL existe, droits, sécurité, etc)
    - 400: Bad Request, 401: Unauthorized, 403: Forbidden, 404: Not Found
  - 500-599 Erreur Serveur liée au langage de programmation (pointeurs, classes, etc)
    - 500: Server Error, 501: Not Implemented, 502: Bad Gateway, 503: Out Of Resources (Service Unavailable)

Un exemple de réponse


```
HTTP/1.1 200 OK
Date: Mon, 24 Mar 2003 15:04:59 GMT
Server: Apache/1.3.26 (Unix) PHP/4.1.1
Last-Modified: Thu, 27 Sep 2001 14:48:40 GMT
ETag: "93d24-114-3bb33c48"
Accept-Ranges: bytes
Content-Length: 276
Connection: Keep-Alive
Content-Type: text/html
Data (276 bytes)
```

HTTP : les cookies

- Principe: Les cookies font partie des spécifications de HTTP, lequel les utilise pour remédier à la « perte de connexion » en faisant une suivie de sessions.
  - Un **cookie** est en réalité un fichier stocké sur le disque dur de l'utilisateur;
  - Ils permettent au serveur web de reconnaître les clients d'une page web à l'autre.
- Le serveur envoie un en-tête :

  Set-Cookie : NOM=VALEUR; domain=NOM_DE_DOMAINE; expires=DATE;
- Le navigateur envoie au serveur :

  Cookie : NOM1=VALEUR1; NOM2=VALEUR2;

mécanisme de session non supporté par HTTP => à programmer

HTTP 1.1

- connexions TCP persistantes
- nouveaux en-têtes
- nouvelles requêtes
  - **PUT** : permet d'envoyer au serveur un document à enregistrer à l'URI spécifiée.
    - des en-têtes permettent de contrôler que tout s'est bien passé
  - **DELETE** : efface la ressource spécifiée.
  - **OPTIONS** : permet au client de savoir les options de communication utilisables pour obtenir la ressource.
    - savoir ce qu'on peut faire sans pour autant devoir demander une ressource.
  - **TRACE** : méthode de contrôle. Demande au serveur de renvoyer la requête telle qu'elle a été reçue.
  - …

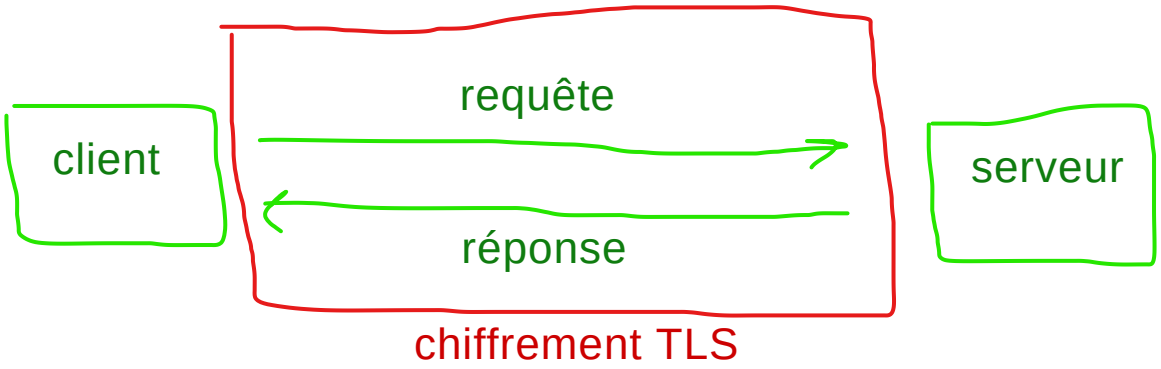
L'essentiel de HTTP

1. HTTP se situe au niveau de la couche Application.
2. HTTP, protocole léger, ne transportant que du texte;
3. HTTP repose sur le système de requêtes/réponse ;
4. HTTP est un protocole sans état; si sauvegarde de session, faite par le programme
5. HTTP assure le transfert bidirectionnel ;
6. Négociation de fonctionnalités ;
7. Support de caches et du proxy serveur;
8. Gestion des sessions grâce aux cookies.

création de cookie avec login (par exemple)
- cookie : espace mémoire navigateur
- existe par défaut dans HTTP à travers entête

HTTPS

- **HTTPS = HTTP Sécurisé**
  - HTTP + une couche de chiffrement (SSl ou TLS).
  - Utilise le port 443 (au lieu de 80).
  - Tout visiteur peut vérifier l'identité du destinataire par un certificat d'authentification (le vice-versa est aussi possible).



Limitation de protocole HTTP

- Limites de protocole HTTP :
  - half duplex : le protocole repose sur le modèle requête/réponse.
  - verbeux : chaque requête et réponse HTTP doit avoir un en-tête (header) ce qui augmente le trafic sur le réseau.
  - il n'est pas possible d'utiliser un mode push de la part du serveur (le serveur envoie à son initiative des données au client).

- Plusieurs solutions ont été proposées pour contourner cette limitation :
  - long pooling : Le serveur n'envoie la réponse que si un évènement le force le d'envoyer les données (données disponibles).
  - Streaming : une technique de transfert de données de type push qui permet à un serveur Web d'envoyer en continu des données à un client via une seule connexion.
  - HTML SSE API : la page Web reçoit automatiquement des mises à jour d'un serveur.
- Cependant, il était nécessaire de définir un standard qui permette la communication entre les clients et le serveur de manière bidirectionnel (canal full-duplex)
- Le mode full-duplex : envoyer des messages du côté client et serveur indépendamment l'un de l'autre.

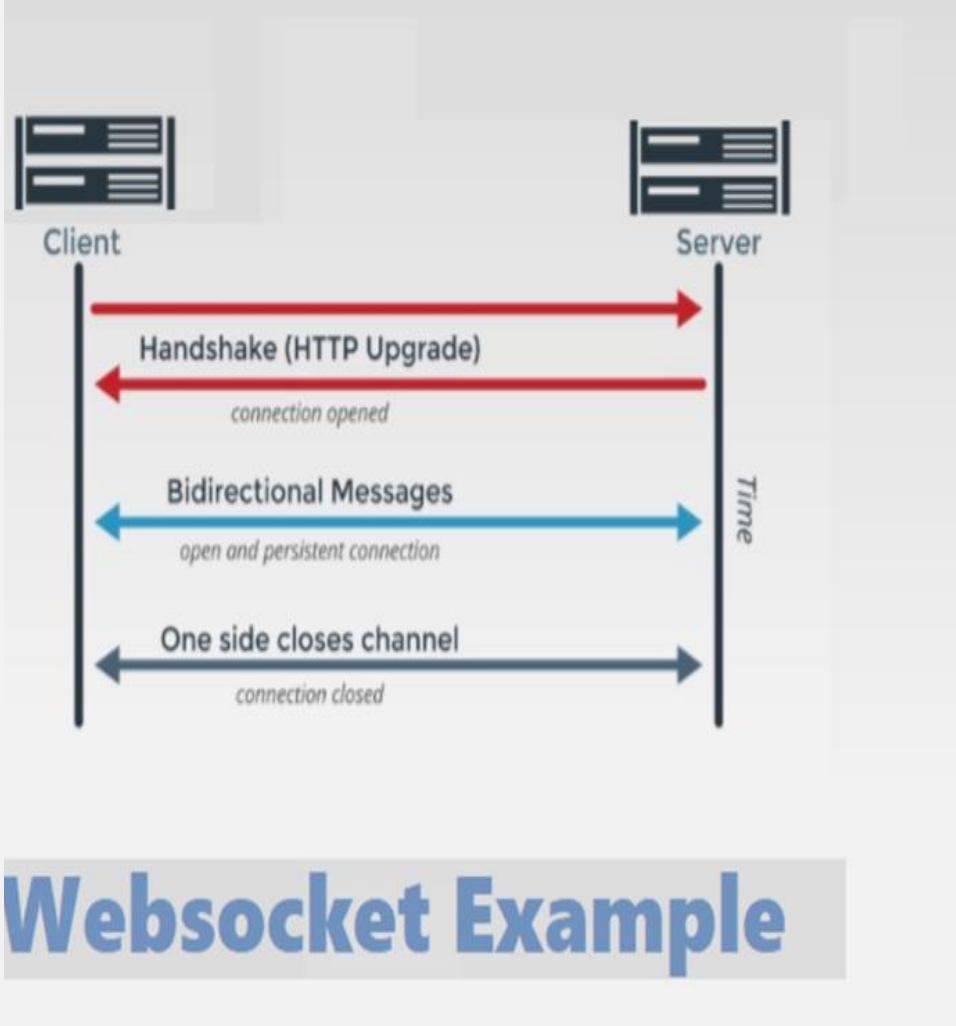
WebSocket

- Est un standard du Web désignant un protocole réseau de la couche application et une interface de programmation du World Wide Web visant à créer des canaux de communication full-duplex par-dessus une connexion TCP pour les navigateurs web.
- Le protocole a été normalisé par l'IETF dans la RFC 64552 en 2011 et l'interface de programmation par le W3C{Wikipédia}.
- C'est à partir du protocole HTTP 1.1 que le changement de protocole est supporté (passer vers WebSocket).
- Si le changement de protocole est réussi alors il n'est plus possible d'utiliser le protocole HTTP et tous les échanges suivants doivent utiliser le protocole WebSocket.

But websocket : garder connexion ouverte => connexion bidirectionnelle (utile pour application en temps réel)

Exemple WebSocket

1. Le protocole HTTP est utilisé (uniquement) pour établir la connexion d'une WebSocket entre le client (initiateur) et le serveur.
2. Une WebSocket est identifiée par une URI particulière définie dans la RFC dont la syntaxe générale est : ws(s)://host[:port]path[?param
3. La fermeture de la connexion permet de passer la WebSocket à l'état déconnecté (peut être à l'initiative du client ou serveur).



Requête/réponse HTTP pour l'ouverture d'une connexion WebSocket

Requête

GET /MaWebApp/echo HTTP/1.1
Cache-Control: no-cache
Connection: Upgrade
Host: localhost:8080
Origin: http://localhost:8080
Pragma: no-cache
Sec-WebSocket-Extensions: x-webkit-deflate-frame
Sec-WebSocket-Key: LwsTSMPv4TKzQscBprG1Iw==
Sec-WebSocket-Version: 13
Upgrade: websocket
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.101 Safari/537.36

Réponse

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Sec-WebSocket-Accept: 9JUSDZQDUFa0yLScZ26xQdyzFy4=
Server: GlassFish Server Open Source Edition 4.0
Upgrade: websocket
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition 4.0 Java/Oracle Corporation/1.7)

La mise en œuvre des WebSockets

- Plusieurs implémentations sont disponibles :
  - JAVA
    - GNU WebSocket4J, une implémentation du protocole WebSocket en Java ;
    - jWebSocket, implémentation Java côté serveur et JavaScript/HTML5 côté client5 ;
  - Javascript
    - ScaleDrone, implémentation javascript du protocole pour REST, Node.js, PHP, Ruby ;
    - Socket.io, implémentation javascript du protocole pour Node.js ;
- Implémenter un serveur WebSocket
  - C#, JAVA, Python, etc
  - https://developer.mozilla.org/fr/docs/Web/API/WebSockets_API/Writing_WebSocket_servers
  - https://developer.mozilla.org/fr/docs/Web/API/WebSockets_API/Writing_a_WebSocket_server_in_Java

HTML: principes de base

- Les principaux documents du Web sont des fichiers textes. On ajoute à ces textes une **structure** et à l'aide de **balises** ("tags") tels que <title>, <h2>,<head>, <p>, <ol>, <li>, etc ...

- HTML est un langage à balise issu d'un langage antérieur et bien plus complexe SGML défini par la communauté des documentalistes et gestionnaires de bibliothèques pour décrire complètement des documents de tout type.

- Après la définition et le succès d'usage de XML, on a fait évoluer HTML en **XHTML** qui reprend et étend HTML en XML

HTML : principes de base

- Html est simple et « léger »
  - Helloworld.doc = 23.5 Ko
  - Helloworld.pdf = 7.02 Ko
  - Helloworld.html = 88 octets

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Ma page test</title>
    </head>
    <body>
        <p>Voici ma page web</p>
    </body>
</html>
```

HTML : principes de base

- HTML est construit sur le concept des balises.

- Les types de balises :
  - Balises contenant des méta-informations
  - Balises de mises en forme de la page web
  - Balises de liens
  - Balises d'insertion multimédia (images, son, vidéos..)
  - ..

- Les balises ne sont pas sensibles à la casse (*préférer en minuscule*).

- Comment écrire les balises :
  - <nom-balise> … </nom-balise>
  - <nom-balise /> fermer balise si sur une seule ligne

Balises contenant des méta-informations

dans l'entête

- Les balises méta servent à :
  - Renseigner sur la langue utilisée du document web, le type de document, le codage utilisé, l'auteur,...
  - Rediriger automatiquement vers une autre page,
  - Interdire la mise en cache ou l'indexation par les moteurs de recherche…

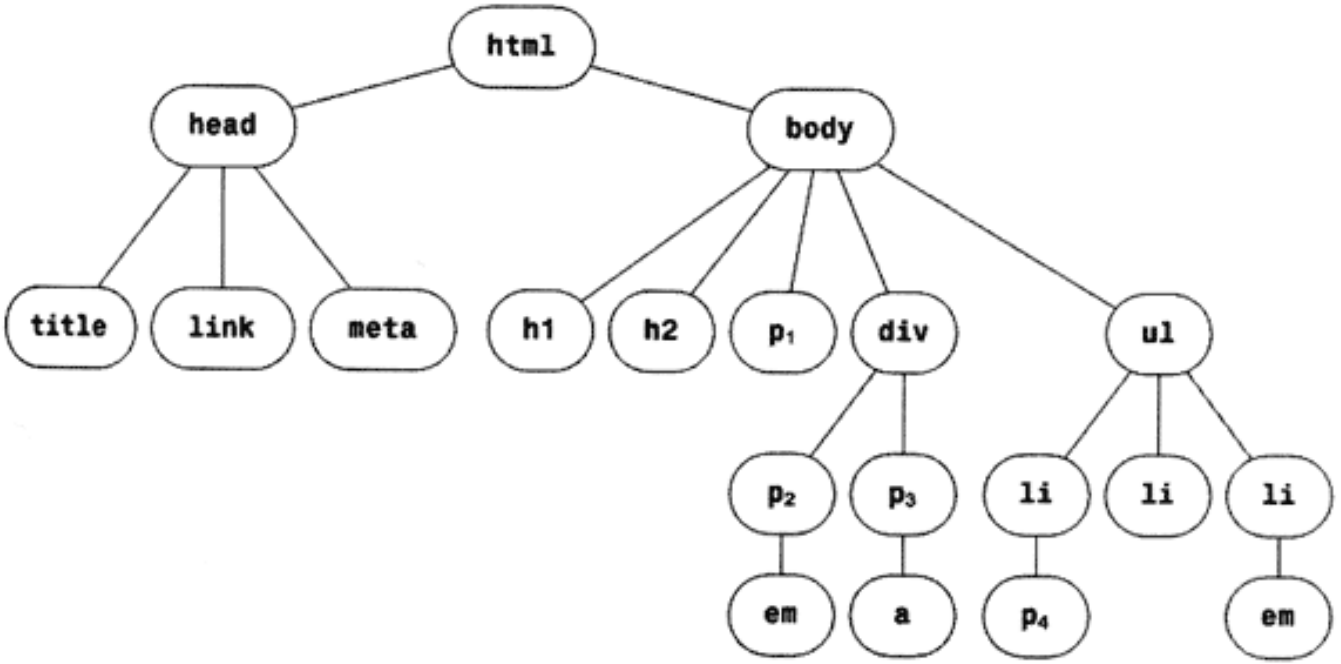
- Exemples :

  Les métas **NAME**, permettant de décrire la page HTML
  - <META NAME="Author" CONTENT= "Ahmed LOUNIS">
  - <META NAME="Copyright" CONTENT= "UTC">
  - <META NAME="Keywords" CONTENT= "SR10, Développement web">

  - **Les balises <META HTTP-EQUIV, permettent d'envoyer des informations supplémentaires au navigateur via le protocole HTTP**
    - <META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    - <META http-equiv="Content-Language"content= "en, fr">
    - <META http-equiv="Refresh" content="10; URL=http://www.hds.utc.fr">
    - <META http-equiv="Pragma" CONTENT="no-cache">
    - <META http-equiv="Content-Style-Type" CONTENT="text/css">

Balises de mises en forme de la page web

- Exemples de balises :
  - <Title>
  - <body>
  - <h1> h2, h3…h6.
  - <p>
  - , <i> ..
  - <font>
  - <hr>
  - <table> <col>…
  - …
  -
  - ….



D'autres balises

- Un document HTML devrait de plus commencer par une déclaration de type de document (<!DOCTYPE HTML PUBLIC "*type_de_HTML*" "*adresse_de_DTD*"> :
  - HTML5 : <!DOCTYPE html>
  - HTM 4.0.1 **Strict** : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">
  - HTM 4.0.1 Transitional : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
  - HTM 4.0.1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

- Balises d'insertion multimédia (images, son, vidéos..) et graphisme (2D)
  - <img>, <object>
  - <video>, <audio>
  - <canvas>

D'autres balises (suite)

- <! ... --> Commentaire
- Balises utilisées pour implémenter une feuille de style par bloc entier
  - ,
- Intégrer des scripts, styles, PHP,…
  - <script>, <style>, <?php … ?>

Balises de liens


```
<HTML>
   <HEAD> <TITLE>titre</TITLE> </HEAD>
   <BODY>
    <H3>Titre de niveau 3 </H3>
    <p><b>Un premier paragraphe :</b> Exemple de liens HTTP </p>
    <br>
       <a HREF="http://www.utc.fr/interne">Aller page interne UTC</a>

        <a HREF="./test3.html">voir test3.html</a>

       <balise id "lienInterne"> ... </balise>

       <a href="#lienInterne"> Lien interne:  </a>

       <a href="url/nom_du_fichier.html#lienInterne"> lien vers une section particulière d'une
       autre page
       </a>
  </Body>
</html>
```

Un lien "absolu"

Un lien "relatif"

Lien interne : Il est possible de marquer un endroit précis d'une page et d'y accéder par un lien hypertexte grâce à l'attribut NAME ou ID :

Pas de lien complets, chemins relatifs plutôt (sinon dépendant de la machine)
+ éviter les majuscules pour les noms de page

HTML : premiers exemples

- Les balises donnent une **structure** aux documents HTML :

```
<!DOCTYPE HTML >
<HTML>
   <HEAD>
      <TITLE>Titre du document</TITLE>
   </HEAD>
   <BODY>
      <P>Insérer du texte ou des images ici</P>
      <P>Un paragraphe</p>
   </BODY>
</HTML>..
```

HTML : premiers exemples (suite)

Le programme **tidy** (https://www.html-tidy.org/) permet de vérifier la correction d'un document html.

Autre exemple :

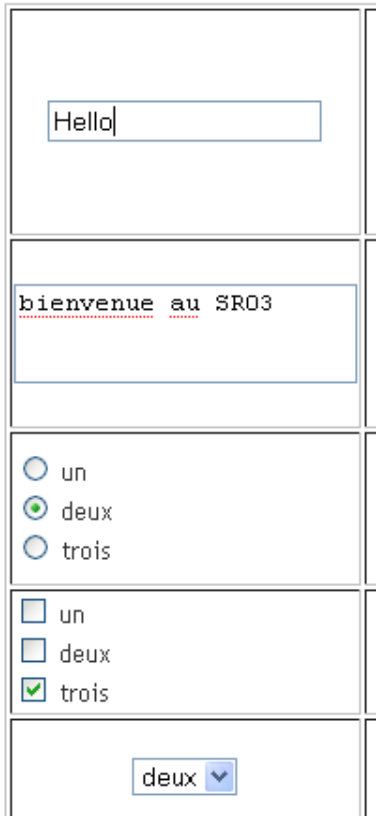
```
<!DOCTYPE HTML >
<HTML>
  <HEAD>
     <!-- commentaire, (en fait du SGML ignoré par HTML) -->
     <TITLE>Titre affiché dans la barre de titre du browser</TITLE>
   </HEAD>
   <BODY>
      <H1>Ceci est un titre de niveau 1 (le plus gros)</H1>
      <p><b>Un premier paragraphe :</b> HTTP est basé sur des fichiers
        textes.</p>
      <p><i>Un deuxième paragraphe :</i> Le web : bla bla ...</p> …
    </BODY>
</HTML>
```

Les formulaires

- **Les balises du formulaire :**

  balise d'ouverture **<FORM>** et fermeture **</FORM>** :

  La spécification de la méthode employée pour l'envoi d'informations au serveur + l'adresse de destination des informations (**<FORM** METHOD=GET ACTION=cgi-bin/test8.cgi>.. **</FORM>)**



Les Balises :

- **La balise <input>**
- **La balise <select>**
- **La balise <option>**
- **La balise <textarea>**

Les champs :

- **Champ de texte**
- **Zone de texte**
- **Boutons radios**
- **Cases à cocher**
- **Liste déroulante**
- **Liste ouverte**

Les formulaires


```
<HTML>
   <HEAD> <TITLE> Exemple de page avec une forme </TITLE> </HEAD>
    <BODY>
          <H3>Exemple de page avec une forme de saisie.</H3>
          <FORM METHOD=GET ACTION=cgi-bin/test8.php>
                    <P>Entrez vos Noms et Adresse :<BR>
                    <P>Nom : <INPUT NAME=nom SIZE=40>
                    <P>Adresse :<INPUT NAME=adresse SIZE=60>
                    <P><INPUT TYPE=submit VALUE=Envoyer> .
                    <INPUT TYPE=reset VALUE=Effacer>
          </FORM>
   </BODY>
 </HTML>
```

ACTION=cgi-bin/test8.cgi : les serveurs web sont souvent configurés pour n'accepter les "cgi" que dans des répertoires particuliers que l'on peut protéger (**drwx–x–x**). Ces fichier ".cgi" peuvent être écrits dans **n'importe quel** langage (PHP, perl, python, C, C++, bash, ...).

Les formulaires


```
<FORM METHOD=GET ACTION=cgi-bin/test9.cgi> .
    <P>Entrez votre nom et vos choix :<BR> </P>
    <P>Nom : <INPUT TYPE=TEXT NAME=nom SIZE=40>
    <P>Choisir 1 parmi 3 :
    <INPUT TYPE=radio NAME=choix1 value="un" />numéro un
    <INPUT TYPE=radio NAME=choix1 value="deux" />numéro deux
    <INPUT TYPE=radio NAME=choix1 value="trois" />numéro trois </P>
    <P>Cocher si urgent : <INPUT TYPE=checkbox Name=urg /> </P>
    <p>choisir 1 dans menu : </P>
    <SELECT NAME=fichier SIZE=3>
    <OPTION>fichier1.txt <OPTION>fichier2.txt
    <OPTION>fichier3.txt <OPTION>fichier4.txt <OPTION>fichier5.txt
    </SELECT>
    <P><INPUT TYPE=submit VALUE=Envoyer><INPUT TYPE=reset VALUE=Effacer> </P>
</FORM>
```

Remarque :

- <SELECT .. SIZE=3> => menu avec "ascenseur" ;
- Si SIZE=1 => menu "déroulant".

Les formulaires

.

Entrez votre nom et vos choix :

Nom :

Choisir 1 parmi 3 : numéro un numéro deux numéro trois

Cocher si urgent :

choisir 1 dans menu :

fichier1.txt
fichier2.txt
fichier3.txt

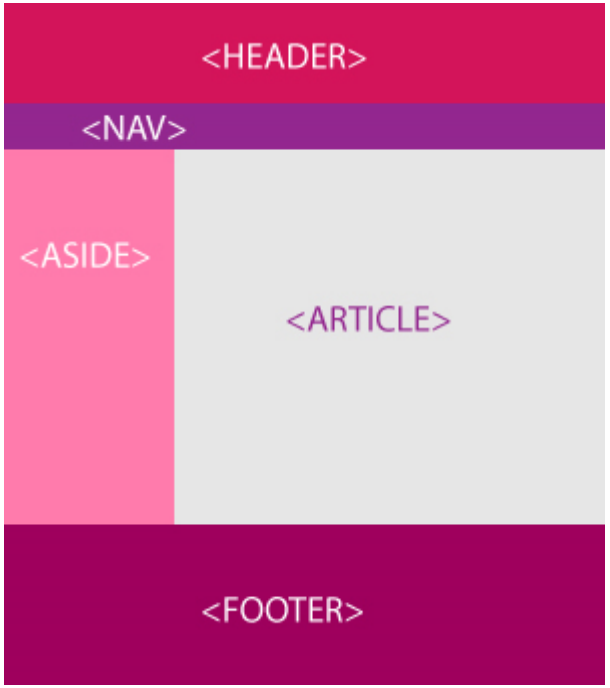
Envoyer Effacer

Événements associés à un formulaire

- Événements associés à la balise <form> :
  - onSubmit
  - onReset
- Événements associés aux champs :
  - onChange
  - onBlur
  - onFocus
  - onSelect
  - ..

HTML5 apporte le support de nouveaux éléments pour aider le navigateur à déterminer la sémantique d'une page.
Il apporte aussi les micro-données :

- utilisées par Google pour référencement ;
- faciliter le « screen scraping » des pages internet.



- un en-tête de page : **HEADER**
- une barre de navigation : **NAV**
- une colonne de gauche : **ASIDE**
- une zone principale pour le contenu : **ARTICLE**
- un pied de page : **FOOTER**

HTML

- HTML décrit la structure du document, ainsi (hélas) que des indications sur la façon d'afficher certains éléments. Mais cette façon dépend de la configuration du browser sur laquelle l'auteur du document n'a pas d'action.

- **Ce mélange** entre structure et "aspect" est mauvais et il est corrigé par l'utilisation des **CSS** et de **XHTML**.

Feuilles de style CSS (Cascading Style Sheets)

Objectif : simplifier la présentation/design des pages web

Les avantages de CSS :

- La **structure et la présentation** sont gérées séparément
- Présenter de façon homogène,
- Positionner rigoureusement les éléments,
- Le code HTML est **allégé** et gagne en lisibilité.

CSS : balise style

- Balise <style>
  - attributs :
    - type="..." : type de contenu Internet
    - media="..." : définit le média de destination (screen, print, projection, braille, speech, all)
    - title="..." : titre de la feuille de styles

```
<STYLE type="text/css">
<!--
.....
-->
</STYLE>
```

CSS : les propriétés

CSS - les propriétés :

- Propriétés de mise en forme des polices
- Propriétés de mise en forme de texte
- Propriétés des couleurs de texte et de fonds de pages
- Propriétés de mise en forme des paragraphes
- Propriétés des bordures de la boîte BORDER
- Propriétés de marge externe MARGIN
- Propriétés de marge interne PADDING
- Propriétés de LIST-STYLE

Les détails se trouvent dans :https://www.w3.org/TR/CSS/#properties

Feuilles de style

- Voici un exemple de règle CSS permettant d'afficher les entêtes principaux (H1) en bleu :

```
H1 { color : blue }
```

- Une règle CSS est composée de 2 parties : un sélecteur (ici H1) et une déclaration (color:blue).

- Une déclaration à elle-même deux parties : une propriété (color) et une valeur (blue).

```
H1, P { color : red }
P { margin-left : 1cm ; text-style : italic }
```

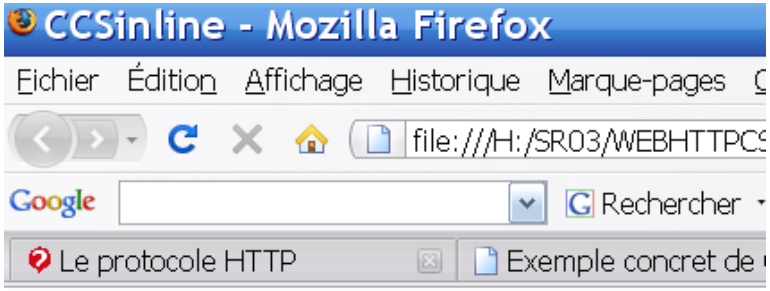
Placer une feuille de style

- Le code CSS peut être placé à 3 positions différentes dans la page.
  - Dans l'élément HTML lui-même : attribut style
  - Dans la page HTML : balise style
  - Dans un fichier indépendant

Placer une feuille de style directement dans la balise HTML

cf cours 3


```
<HTML>
  <HEAD>
    <TITLE>
      CCSinline
    </TITLE>
  </HEAD>
  <BODY>
<p style="font-style: italic; size: 1.5em;"> CSS
directement dans la balise HTML </p>
  </BODY>
</HTML>
```



CSS directement dans la balise HTML

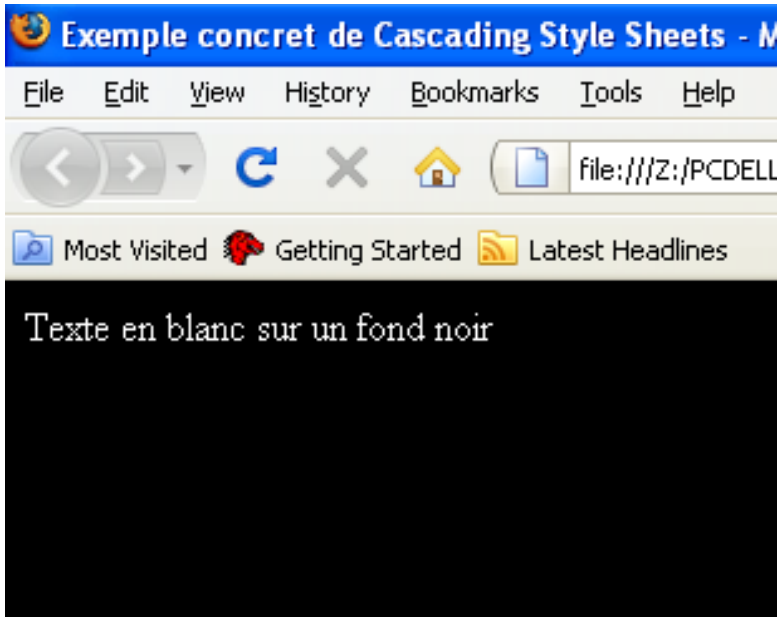
Cette méthode est à éviter

Placer une feuille de style dans l'entête de la page HTML

cf cours 3

- Dans le code dans la page HTML, entre les deux balises <head> et </head>.

```
<HTML>
 <HEAD>
  <TITLE>Exemple concret de Cascading Style Sheets</TITLE>
  <STYLE TYPE="text/css">
   <!--
    BODY {
     color: white;
     background: black;
    }
   //-->
  </STYLE>
 </HEAD>
 <BODY>
  Texte en blanc sur un fond noir
 </BODY>
</HTML>
```



Placer une feuille de style dans un fichier séparé

- Syntaxe: <link rel="stylesheet" type="text/css" href="style.css" />
- La méthode "**<link href=...**" permet aussi de mettre en place des feuilles de styles destinées aux différents medias (imprimante, navigateurs de PDA, *etc.*).
  - screen, projection, print, all…

- Comment ça marche : Il s'agit de placer la feuille de style dans un fichier séparé, et à y faire référence dans l'entête du document.

```
…<head>
<title><title>
<link rel="stylesheet" type="text/css" href="style.css" media="screen, projection"
   />
</head>
<body> …
```

Fichier CSS (style.css) :

```
H1, H2,{ color : red; font : bold}   /* Les titres principaux sont en rouge   */
H3, H4, H5 { color : green} /* Les titres secondaires sont en vert */
```

Placer une feuille de style par importation

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

Syntaxe:

```
<style type="text/css">
@import url(styles/affichage.css) media;
</style>
```

Cela peut être utile pour importer des feuilles de style dans d'autres feuilles de style.

DHTML

- **HTML dynamique**, (***Dynamic HTML***, ou **DHTML)**, est un nom générique donné à l'ensemble des techniques utilisées par l'auteur d'une page web pour que celle-ci soit capable de se modifier elle-même en cours de consultation dans le navigateur web.

- Comment ça marche :
  - la représentation interne est initialement déterminée par le document HTML et les informations de style CSS constituant la page web.
  - Les modifications sont effectuées via JavaScript, qui accède à la représentation interne à travers l'interface de programmation Document Object Model (DOM).
    - Attention : Les propriétés CSS du DOM ne correspondent pas nécessairement aux noms "CSS ".

DOM : représentation objet du document HTML => manipulation possible via Javascript (attributs, méthodes)

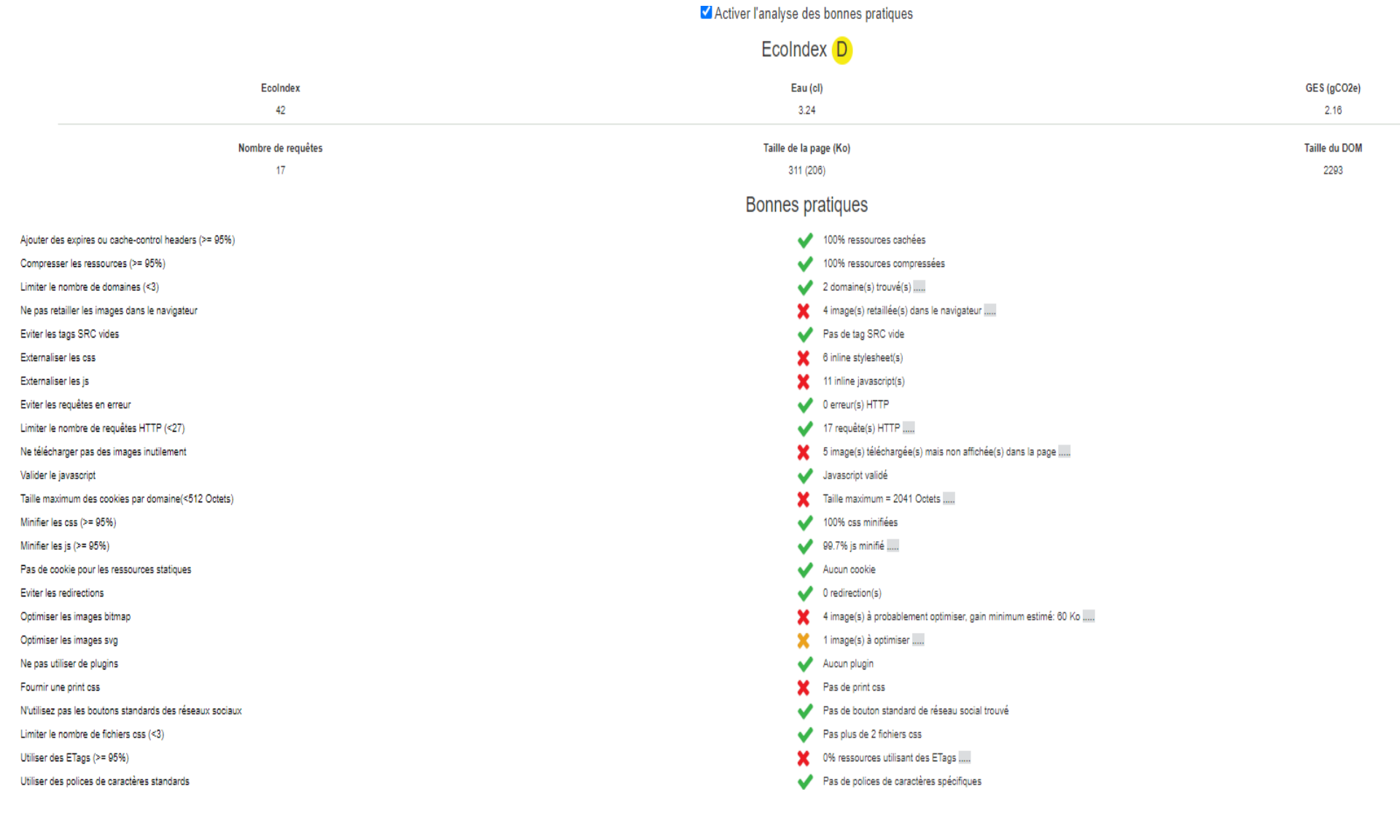
Consommation énergétique des standards web

- L'empreinte carbone des TIC serait équivalente à celle de l'aviation civile sur une année
- Quelques bonnes pratiques (Guide des 115 bonnes pratiques)
  - Pages web moins complexes (DOM simple)
  - Favoriser un design simplifié, épuré et adapté au Web
  - Préférer l'approche mobile quand c'est possible
  - Limiter le nombre de requêtes HTTP
  - Stocker localement les données statiques
  - Utiliser un framework vs développer sur mesure
  - Choisir un format de données adaptées

Evaluer l'emprunte écologique de votre page web

- GreenIT
  - Site web : **Eco Index**
  - Plugin chrome ou firefoxe
- Quelques éléments de résultat d'évaluation :
  - Performance environnementale absolue à l'aide d'un score sur 100 (higher is better) ;
  - Performance environnementale relative à l'aide d'une note de A à G ;
  - L'empreinte technique de la page :
    - Poids de la page qui se mesure avec la bande passante en Ko et témoigne des efforts à faire pour transporter la page jusqu'au navigateur.
    - Complexité de la page qui se mesure en nombre d'éléments du DOM
    - Charge serveur qui se mesure en nombre de requêtes HTTP (aller retours serveurs)
  - L'empreinte environnementale associée (gaz à effet de serre et eau).

Plugin Green IT (en mode développeur)



☑ Activer l'analyse des bonnes pratiques

EcoIndex D

EcoIndex	Eau (cl)	GES (gCO2e)
42	3.24	2.16

Nombre de requêtes	Taille de la page (Ko)	Taille du DOM
17	311 (206)	2293

Bonnes pratiques

Ajouter des expires ou cache-control headers (>= 95%)	✔ 100% ressources cachées
Compresser les ressources (>= 95%)	✔ 100% ressources compressées
Limiter le nombre de domaines (<3)	✔ 2 domaine(s) trouvé(s) ....
Ne pas retailler les images dans le navigateur	✘ 4 image(s) retaillée(s) dans le navigateur ....
Eviter les tags SRC vides	✔ Pas de tag SRC vide
Externaliser les css	✘ 6 inline stylesheet(s)
Externaliser les js	✘ 11 inline javascript(s)
Eviter les requêtes en erreur	✔ 0 erreur(s) HTTP
Limiter le nombre de requêtes HTTP (<27)	✔ 17 requête(s) HTTP ....
Ne télécharger pas des images inutilement	✘ 5 image(s) téléchargée(s) mais non affichée(s) dans la page ....
Valider le javascript	✔ Javascript validé
Taille maximum des cookies par domaine(<512 Octets)	✘ Taille maximum = 2041 Octets ....
Minifier les css (>= 95%)	✔ 100% css minifiées
Minifier les js (>= 95%)	✔ 99.7% js minifié ....
Pas de cookie pour les ressources statiques	✔ Aucun cookie
Eviter les redirections	✔ 0 redirection(s)
Optimiser les images bitmap	✘ 4 image(s) à probablement optimiser, gain minimum estimé: 60 Ko ....
Optimiser les images svg	✘ 1 image(s) à optimiser ....
Ne pas utiliser de plugins	✔ Aucun plugin
Fournir une print css	✘ Pas de print css
N'utilisez pas les boutons standards des réseaux sociaux	✔ Pas de bouton standard de réseau social trouvé
Limiter le nombre de fichiers css (<3)	✔ Pas plus de 2 fichiers css
Utiliser des ETags (>= 95%)	✘ 0% ressources utilisant des ETags ....
Utiliser des polices de caractères standards	✔ Pas de polices de caractères spécifiques

Exemples d'évaluation

Orange 91,7 A
Youtube : 87,9 A
Google : 80,2 A
Le bon coin 79,8 A
Twitter : 74 B
Facebook : 70,5 B
Netflix : 69,1 B
Moodle UTC : 65, 6 B
Heudiasyc : 54,1 C -> page trop lourde
Wikipédia : 39,9 D
Ent UTC : 35,6 D -> page trop complexe et trop d'aller retours serveurs
FranceTv info : 21,4 E
Amazon 3,4 G

Sources bibliographiques

- M. Vayssade « SR03 : ARCHITECTURES INTERNET » ;
- S. Cateloin, Cours *HTTP*, l'Université Louis Pasteur de Strasbourg;
- "HTML et javascript", S. Maccari et S. Martin, ed. MicroApplication, 2004.
- Douglas Comer, TCP/IP Architectures, protocoles et applications, Pearson Education, 5eme édition, 2009.
- A. Ploix, Cours *Réseaux IP : Internet Réseaux d'entreprise*, Université de Technologie de Troyes.
- Sources Internet :

  http://www.developpez.com
  http://fr.wikipedia.org
  https://developer.mozilla.org/fr/
  https://www.w3schools.com/
  http://www.alistapart.com/articles/cssatten
  https://www.jmdoudoux.fr/java/dej/chap-websockets.htm#websockets-1

- Guide des 115 bonnes pratiques : https://univ-scholarvox-com.ezproxy.utc.fr/reader/docid/88868673/page/52

Les classes d'éléments

- permettent de choisir entre plusieurs types de présentations pour un même élément HTML.
  - Exemple : on peut avoir les paragraphes d'entête, des remarques et des paragraphes normaux.

- l'attribut class, qui peut être appliqué à tous les éléments HTML, permet de préciser des classes d'éléments. On peut alors préciser les styles qui doivent s'appliquer à chaque classe.

```
.rouge { color: red }
P.entete { font-style: italic }
```

Un exemple


```
<HTML>
 <HEAD>
  <TITLE> Les classes </TITLE>
  <STYLE type="text/css">
   <!--
.rouge { color: red }
P.entete { color: green; font-style: italic }
    //-->
  </STYLE>
 </HEAD>
 <BODY>
<H2 class=rouge> Un titre rouge </H2>
<P class=entete> Un titre italic en vert </P>
 </BODY>
</HTML>
```

Sélecteurs CSS class et id

Exemple :

```
#menu { background-color:silver;
width:100px; float:left; }
#contenu { margin-left:110px; }
```

```
<h1 id="haut">Exemple des sélecteurs "class" et
"id"</h1>

<div id="menu">
<ul>
        <li>item 1</li>
        <li>item 2</li>
        <li>item 3</li>
</ul>
</div>

<div id="contenu">
        <p>
        text.
        </p>

        <p class="haut">
        <a href="#haut">Haut de page</a>
        </p>
</div>
```

- *un id s'applique à un objet unique : il ne peut pas y avoir deux mêmes id dans une page*
- *une classe peut caractériser plusieurs objets (identiques ou non)*

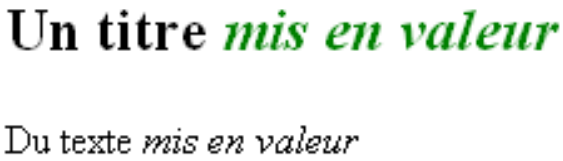
Contextes

- Il est possible de préciser les propriétés qui doivent s'appliquer à un élément dans le cas où celui est utilisé à l'intérieur d'un autre élément (et pas ailleurs).
  - H2 EM {color : green}

- Dans le code HTML, on peut utiliser l'élément EM dans un entête H2 ou pas :
  - <H2>Un titre <EM>mis en valeur</EM></H2>
    <P>Du texte <EM>mis en valeur</EM>

Un exemple


```
<HTML>
 <HEAD>
  <TITLE> Contextes </TITLE>
  <STYLE type="text/css">
   <!--
   H2 EM {color : green}
   //-->
  </STYLE>
 </HEAD>
 <BODY>
<H2>Un titre <EM>mis en valeur</EM></H2>
<P>Du texte <EM>mis en valeur</EM>
  </P>
 </BODY>
</HTML>
```



Pseudo-classes

Les pseudo-classes d'ancre. :link et :visited

- CSS permet de représenter différemment les liens qui n'ont pas été visités de ceux qui l'ont déjà été au travers des pseudo-classes ':link' et ':visited'.
  - Ex. A:link { color : fushia; } A:visited { color : olive; } A:active { color : green; }
  - La pseudo-classe :link s'applique aux liens qui n'ont pas été visités ;
  - La pseudo-classe :visited s'applique lorsque le lien a été visité par l'utilisateur.
  - La pseudo-classe :active permet de cibler un élément lorsque celui-ci est activé par l'utilisateur.

Pseudo-classes

exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
  <head>
    <link rel="stylesheet" href="P.css" type="text/css">
    <title>CSS Pseudo Classes</title>
  </head>
  <body>
        <td>
          <a href="http://www.utc.fr">site UTC</a>
        </td>
</body>
</html>
```

```
a:link
{
background-color: YellowGreen;
  color: white;
  font-family: Arial;
  font-size: .8em;
}
a:visited
{
  background-color: red;
  color: white;
  font-family: Arial;
  font-size: .8em;
}
a:hover
{
  background-color: cyan;
  color: blue;
  font-family: Arial;
  font-size: .8em;
}
```

Pseudo-éléments

- Les pseudo-éléments créent des abstractions dans l'arbre en plus des éléments déjà spécifiés par le langage du document.
  - Ainsi, certains langages n'offrent pas de mécanismes de correspondance avec la première lettre ou la première ligne du contenu d'un élément. Les pseudo-éléments de CSS permettent aux auteurs d'y accéder.

Pseudo-éléments

- P:first-letter { font-size: 200%;}
- P:first-line { font-variant:small-caps; }

```
<HTML>
  <HEAD>
    <TITLE>Lettrine et première ligne</TITLE>
    <STYLE type="text/css">
      <!--
        P              { color: red; font-size: 12pt }
        P:first-letter { color: green; font-size: 200%; font-weight: 900 }
        P:first-line   { color: blue }
      //-->
    </STYLE>
  </HEAD>
  <BODY>
    <P>
      Voici paragraphe dont la première lettre est verte et grande,
      dont le texte de la première ligne est bleu et les autres lignes
      sont rouges.
    </P>
 </BODY>
</HTML>
```

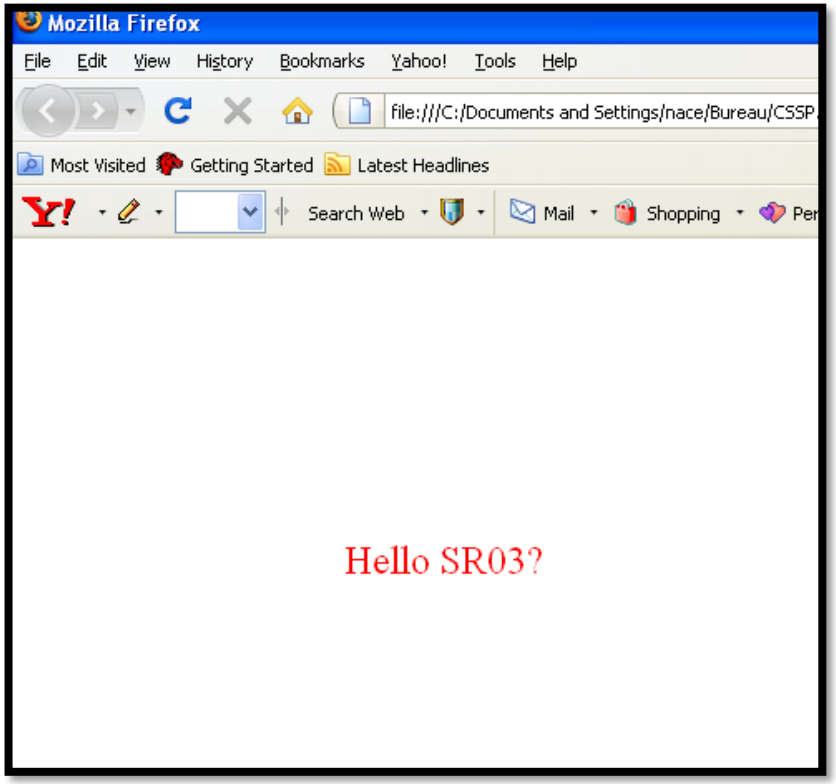
Positionner des éléments grâce aux CSS

- Il est possible grâce aux feuilles de style de positionner au pixel près du texte ou des images grâce aux balises <SPAN> (de type inline) et <DIV> (de type block).

- **Positionnement relatif et absolu**
  - Le positionnement absolu {position: absolute} se détermine par rapport au coin supérieur gauche de la fenêtre du navigateur. Les coordonnées d'un point s'expriment alors de haut en bas (top) et de gauche à droite (left).
  - La position relative se fait par rapport à d'autres éléments, c'est-à-dire que les éléments contenus dans la balises *DIV* ou *SPAN* seront positionnés à la suite des éléments HTML après lesquels ils se trouvent.

Positionner des éléments grâce aux CSS


```
<HTML>
<HEAD>
<STYLE>
<!--
.pos{position: absolute; top: 180px; left: 200px; color: red; font-size: x-large}
-->
</STYLE>
</HEAD>
<BODY>
<DIV class=pos>
Hello SR03?
</DIV>
</BODY>
</HTML>
```



Cascade

- Les propriétés des CSS peuvent être définies plusieurs fois. C'est toujours la dernière définition qui compte. Cela permet d'importer plusieurs feuilles de styles, et de redéfinir certains styles dans le document.

```
<body>
    <div>
        <p>
            bla bla bla
        </p>
    </div>
</body>
```

- Supposons qu'on dispose d'une première feuille de style, que nous appellerons style1.css qui contienne les propriétés suivantes :
  - H1 { color : red; font-size : 48pt }
    H2 { color : blue; font-size : 12pt }

- Nous utilisons aussi une autre feuille de style, nommée style2.css et contenant les propriétés suivantes :
  - H2 {color : green; }
    H3 {color : pink; font-size : 12pt }

Cascade

- Dans une page donnée, nous incluons dans l'entête l'appel de ces deux feuilles, ainsi que la définition d'autres propriétés.

```
<HEAD>
<TITLE>...</TITLE>
<LINK rel=STYLESHEET href="style1.css" type="text/css">
<LINK rel=STYLESHEET href="style2.css" type="text/css">
<STYLE type="text/css">
<!–
H1 { color : fushia; } H2 { font-size : 16pt; } H3 { font-size : 14pt; } -->
</STYLE>
</HEAD>
```

- Déterminons les valeurs utilisées pour ce document :
  - H1 : fushia, 48 points
  - H2 : vert, 16 points
  - H3 : rose, 14 points

Cascade/Héritage

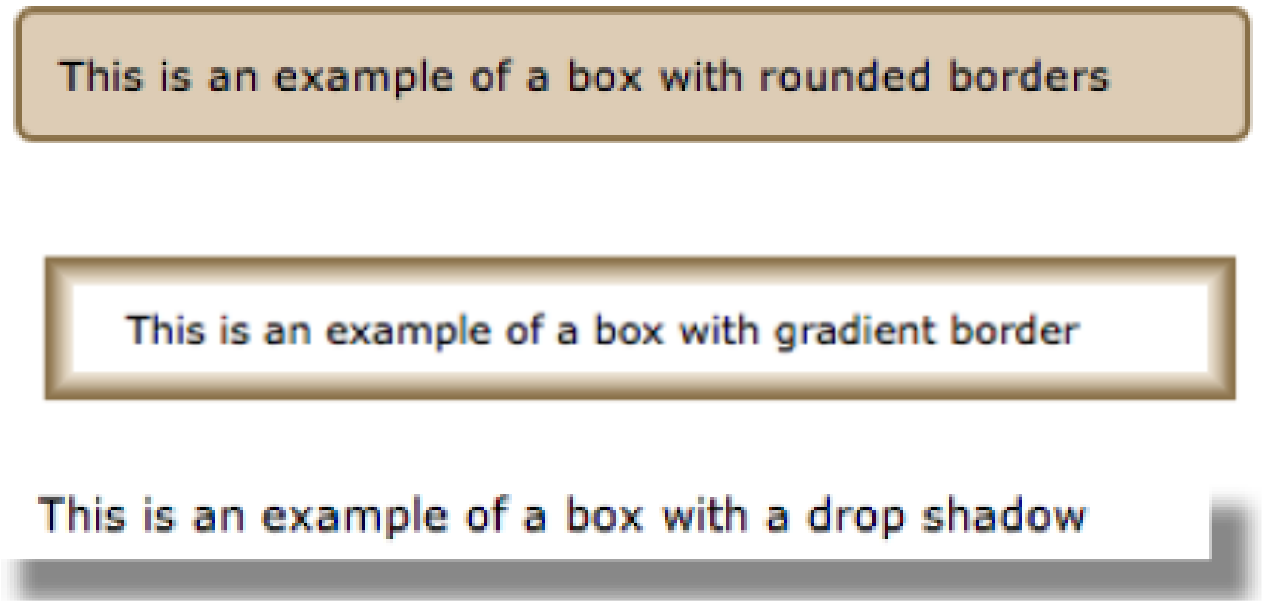
- Pour déterminer la valeur d'une propriété, on dispose donc de la notion de cascade. Dans les cas où la propriété n'a pas été définie, deux possibilités se présentent :
  - Ou bien la propriété est dite "héritée". Dans ce cas, c'est la valeur de l'élément "parent", c'est à dire de l'élément dans lequel est utilisé l'élément courant.
  - Dans l'autre cas, on prend la valeur par default.

A retenir : la dernière règle lue est prioritaire!!

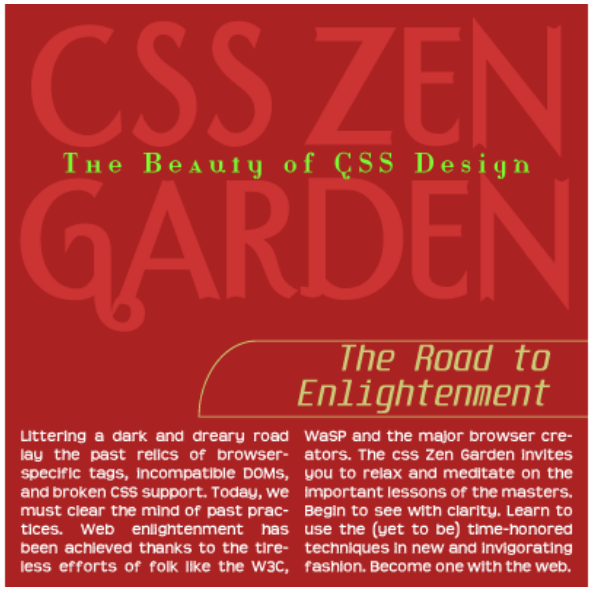
CSS3… CSS4

- CSS3 est modulaire. Le degré d'avancement de CSS3 varie selon les modules et le degré de priorité qui leur a été donné par le groupe de travail CSS
  - Il offre de nouvelles fonctionnalités pour améliorer le design des sites web, en particulier:
    - Les arrières plans et les bordures
    - Les effets de texte
    - La mise en page multi-colonnes…
- CSS4 est en cours…

CSS3


This is an example of a box with rounded borders
This is an example of a box with gradient border
This is an example of a box with a drop shadow
This is an example of text with a shadow applied

Web design via CSS



http://www.alistapart.com/articles/cssatten