# Initiation au développement Web (M1106)

**Eugen Dedu**

Maître de conférences
Univ. Bourgogne Franche-Comté, IUT de Belfort-Montbéliard
Dépt. R&T, 1ère année
Montbéliard, France
nov. 2018

http://eugen.dedu.free.fr
eugen.dedu@univ-fcomte.fr

# Intégration dans la formation R&T

- Pré-requis : rien
- Prolongement : Web dynamique (S2)

- FI : 4h30 CM, 1h30 TD, 18h TP
- FA : 4h30 CM, 1h30 TD, 15h TP
- Examen : environ 1h30 en fin de dernière séance de TP, sur ordinateur, Internet autorisé

- Objectif : savoir concevoir des pages Web simples pour des média variés (PC, tablette, smartphone) en utilisant les dernières technologies
  - c'est une condition sine qua non pour l'écriture des sites Web dynamiques
  - c'est un outil pour des fonctionnalités d'administration par exemple
- Ce n'est pas faire multimédia ou pages dynamiques ou encore sites Web complexes

# Ce que je n'apprécie pas

- Utilisation du téléphone ; les mettre dans le sac
- Rigoler, sourire aux autres, embêter les autres
- Gros mots
- Retard >10'

# Bibliographie

- Tout livre ou site Web traitant du même sujet, voir à la bibliothèque
- https://developer.mozilla.org/fr/docs/web/html
- https://www.w3schools.com

# Différence entre Web et Internet

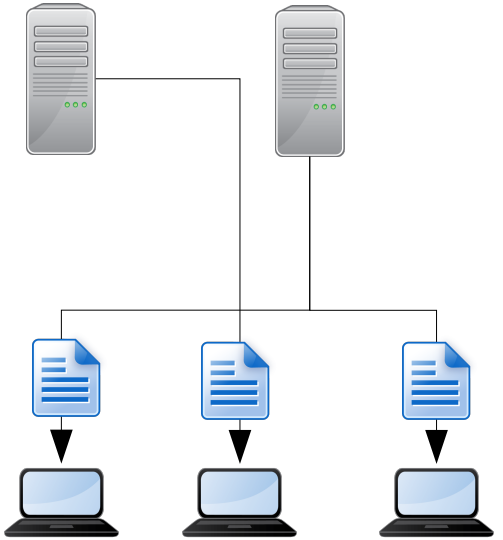
- Internet = le réseau
- Services utilisant Internet : FTP, WWW (Web), courriel, jeux, VoIP etc.
  - pour accéder à ces services, on utilise normalement un logiciel spécifique, mais parfois on peut utiliser un navigateur Web, par ex. on lit ses courriels dans le navigateur
- Le Web a été créé par un chercheur au CERN, Genève, Tim-Berners Lee, en 1991
- HTML (*HyperText Markup Language*) = « langage » de description de la page Web
- HTTP (*HyperText Transfer Protocol*) = le protocole définissant l'échange de données entre le client (navigateur) et le serveur

# Plan du cours

- Introduction
- Pages HTML
  - généralités
  - formulaires
  - multimédia
  - attributs événements
- CSS
- Responsive Web Design

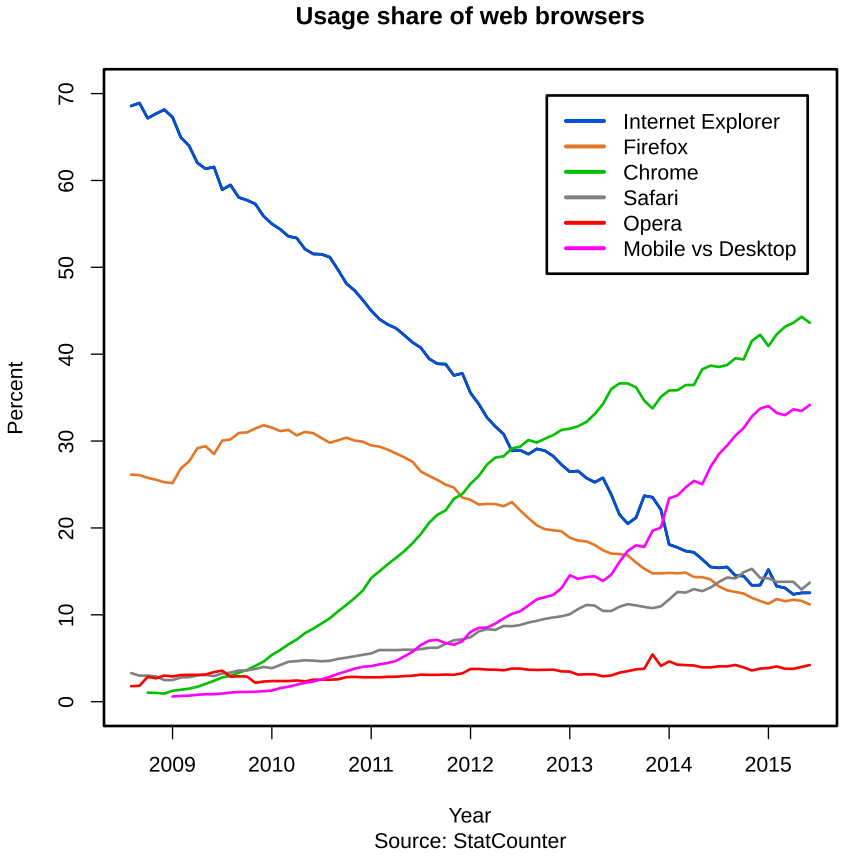
# Modèle client-serveur

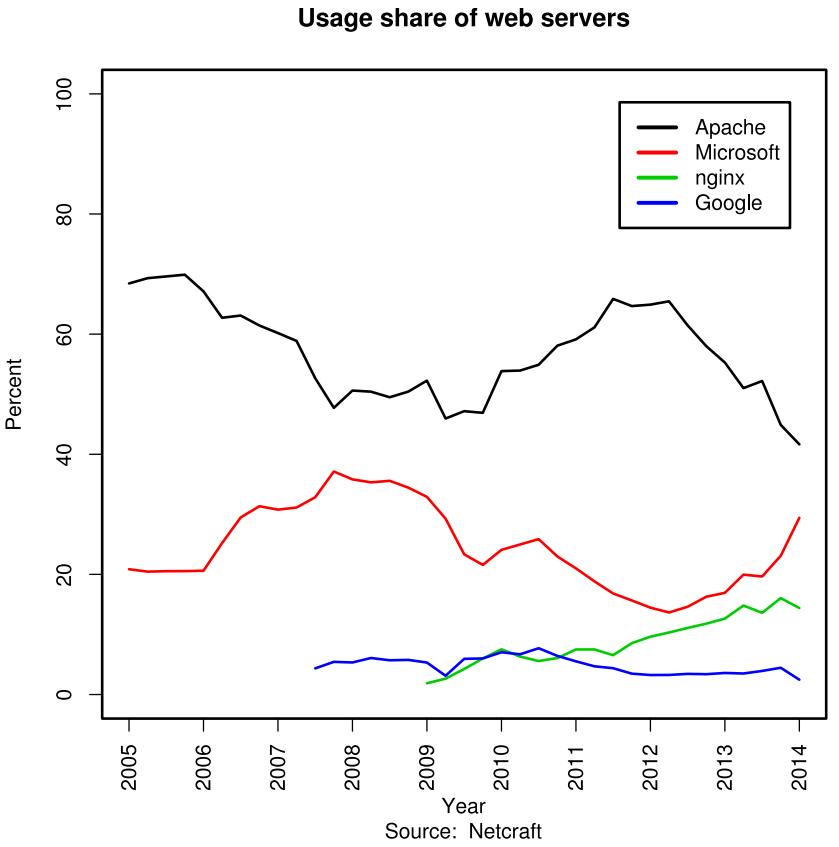
- HTTP utilise le modèle de communication client-serveur
- Dans ce modèle, les machines sont divisées en deux : serveurs et clients
  - serveur = la machine qui attend des connexions et qui a les pages Web
  - client = la machine qui initie les connexions
- D'autres modèles de communication existent :
  - le pair-à-pair (p2p), où chaque entité contient des données et est client et serveur à la fois, par ex. la visioconférence
  - graphes, arbres



# Clients et serveurs Web

- Même si les navigateurs/serveurs Web sont faits par des organismes différents, ils sont tous compatibles, car ils suivent les spécifications HTTP, PHP etc.



# Plan du cours

- Introduction
- Pages HTML
  - généralités
  - formulaires
  - multimédia
  - attributs événements
- CSS
- Responsive Web Design

# Structure d'une page HTML

- Une page HTML est un document **texte** (composé de caractères latins), comme un programme écrit en C, et contrairement aux documents Word ou aux fichiers exécutables par ex.
- Dans un navigateur, ctrl-u affiche la source de la page
- Une page a une déclaration plus deux parties :
  - déclaration de la page HTML5 : <!DOCTYPE html>
  - head : diverses informations sur la page, comme le titre, l'encodage des caractères, l'icône du site, éventuellement aussi l'auteur de la page, la date etc.
  - body : le contenu proprement dit de la page, affiché par le navigateur
- Il est préférable d'utiliser des minuscules comme nom de fichier
- Dans ce cours nous allons valider les pages écrites : http://validator.w3.org

**Page HTML minimale :**

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>

<body>
  Ma première page.
</body>
</html>
```

# Élément et balise HTML

- Élément HTML : <h1>Ceci est un titre</h1>
  - le contenu de l'élément et la balise (*tag*) HTML
- Certains éléments finissent à la balise de début (n'ont pas de balise de fin), par ex. <hr>
- Les éléments peuvent être imbriqués, par ex. Gras et italique
- Les balises sont case insensitive, mais il est recommandé d'utiliser des minuscules, par ex. écrire <body>, pas <BODY>
- Conseils pratiques :
  - bien fermer chaque balise (le cas échéant)
  - indenter le code aux balises importantes (head, body, div, tableaux, h1, ul, ol)

# Attributs des balises/éléments

- Certains éléments peuvent avoir des attributs
- Exemple : <a href=”index.html”>Lien</a>
- Les attributs se mettent toujours dans la balise initiale (ouvrante)
- Sous format nom=”valeur”
- Attributs utilisables pour tout élément HTML :
  - id – identifiant de l'élément
  - title – information supplémentaire, infobulle
  - et d'autres

# En-tête d'une page HTML

- Balises recommandées (lire « obligatoires ») :
  - <meta charset="UTF-8">
    - erreur commune : encodage non spécifié
    - sauvegarder le fichier en UTF-8 aussi !!
  - <title>Titre de la page</title>
    - affiché dans la barre de titre et l'onglet du navigateur
    - affiché par les moteurs de recherche
    - utilisé par défaut quand on ajoute la page dans les favoris
- Exemples de balises optionnelles :
  - <link rel="icon" href="favicon.png">, l'icône étant 16x16 généralement
  - <meta name="author" content=”Eugen Dedu”>
  - <meta http-equiv="refresh" content="30">
  - <link href="styles.css" rel="stylesheet">

# Quelques balises usuelles

- Éléments bloc, affichés l'un au-dessous de l'autre :
  - <h1>Titre</h1>, h2...h6
  - ...
  - <hr>
  - <ul><li>ligne</li>...</ul> et pareil pour <ol>
- Éléments en ligne, affichés l'un après l'autre :
  - <a href="...">...</a>
  - ..., ..., <small>...</small>
- Éléments inline-block, affichés l'un après l'autre mais en tant que bloc à l'intérieur :
  - <img src="..." alt="...">
- <table><tr><td>cell1<td>cell2...</tr>...</table>
  - au lieu de td on peut mettre th pour une cellule d'en-tête
- Commentaires : entre <!-- et -->

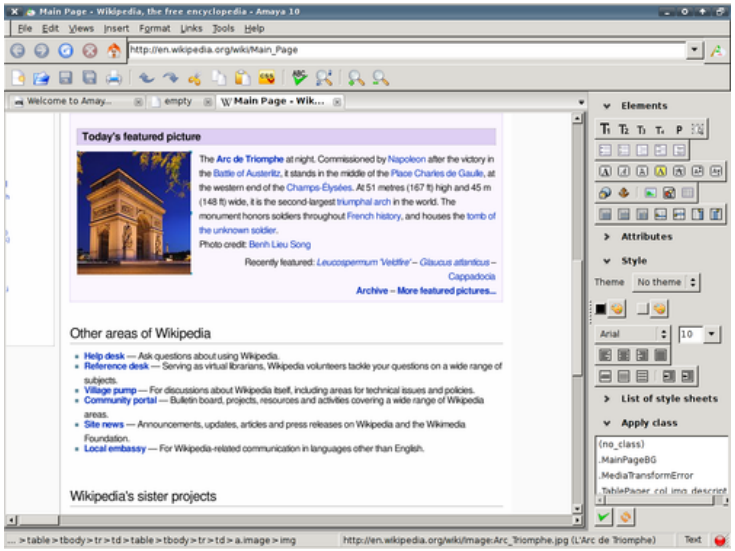
# Contenu d'une page HTML

- Exemples de balises HTML, voir balises.html
- Beaucoup d'autres balises existent, mais pas si importantes pour un technicien RT...
- Cinq caractères sont réservés :

| Caractère réservé | Écriture en HTML |
|---|---|
| < | &lt; |
| > | &gt; |
| & | &amp; |
| " | &quot; |
| ' | &apos; |

# Éditeurs HTML

- Éditeurs texte : emacs, gedit, brackets, notepad, bluefish etc.
  - la page doit être visualisée (lue) dans un navigateur
- Éditeurs WYSIWYG : libreoffice, word, dreamweaver – en plus du texte, on peut éditer sur le *rendu* de la page (comme dans le navigateur), mais l'avantage du style visuel est limité :
  - n'est pas adapté au HTML, qui utilise un style sémantique parfois (ex. h1 vs b)
  - on ne peut pas utiliser toutes les fonctionnalités HTML
  - ne suffit pas pour faire de bonnes pages (par ex. mettre dans CSS le style de tout le site)
  - ne permet pas de créer des pages dynamiques (voir S2), et de nos jours on écrit plus de pages dynamiques que statiques
  - la page est parfois rendue différemment par les navigateurs
  - les sites modernes n'utilisent plus des fichiers, mais des BD
- Facilités additionnelles : intégration avec CSS et JavaScript, autocomplétion, choix facile des couleurs etc.




Éditeur HTML visuel Amaya (arrêté)

# Création d'un site Web

- Pour héberger son site Web il faut une machine (serveur) connectée en permanence à Internet
- Soit une machine personnelle, mais il faut avoir une adresse IP publique etc.
- Soit un hébergeur :
  - gratuit (par ex. free.fr) ou payant
  - qualité de service différente : bande passante, fonctionnalités fournies (PHP, MySQL, FTP/SSH etc.), sauvegardes régulières, mises-à-jour des logiciels pour sécurité etc.

# Évolution des pages Web

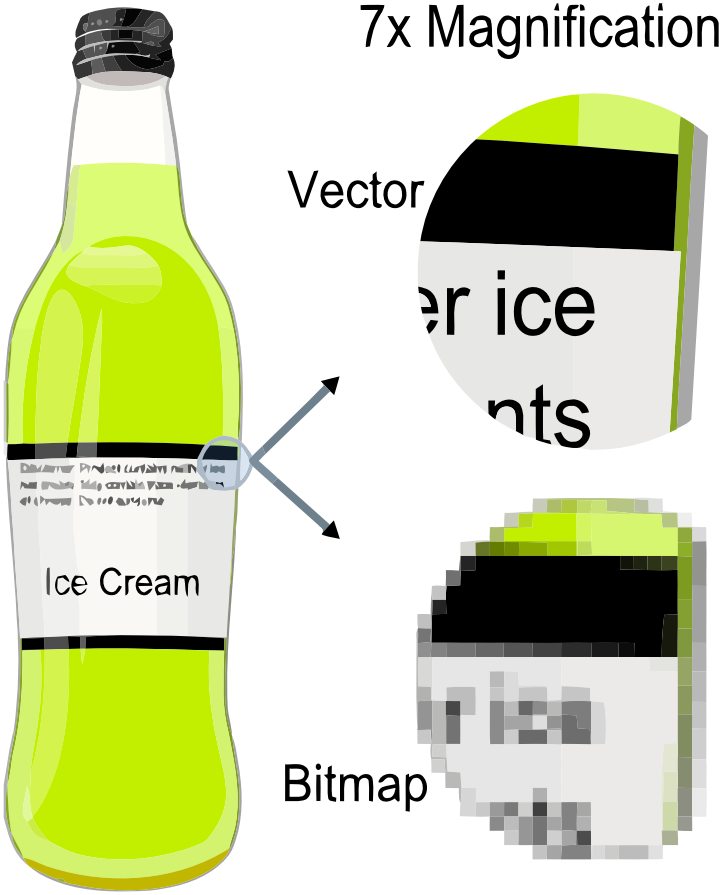
- Au début (1991), texte, liens et images (essentiellement) – on pouvait faire clic pour se balader sur d'autres pages !!
- CGI, PHP etc. – générer une page à la volée
- Formulaires – page générée en fonction du choix de l'utilisateur
- ~2000 : plugin – flash pour présenter de la vidéo
- CSS – séparation du contenu de la page et de sa présentation
- JavaScript – le navigateur anime les pages
- 2010–2016 : balises audio et vidéo standard, menu contextuel personnalisable, images responsives
- Futur ???

# La spécification HTML

- Plusieurs versions de la *spécification HTML* existent :
  - HTML version 4 (1997) – encore utilisée
  - HTML5 (2014) – JavaScript comme langage standard, introduit multimédia, améliore les formulaires, les attributs événements etc. – c'est ce que nous utiliserons dans ce cours, html5test
  - HTML 5.1 (nov. 2016) – images responsives, accès au menu contextuel etc.
  - HTML 5.2 (...)

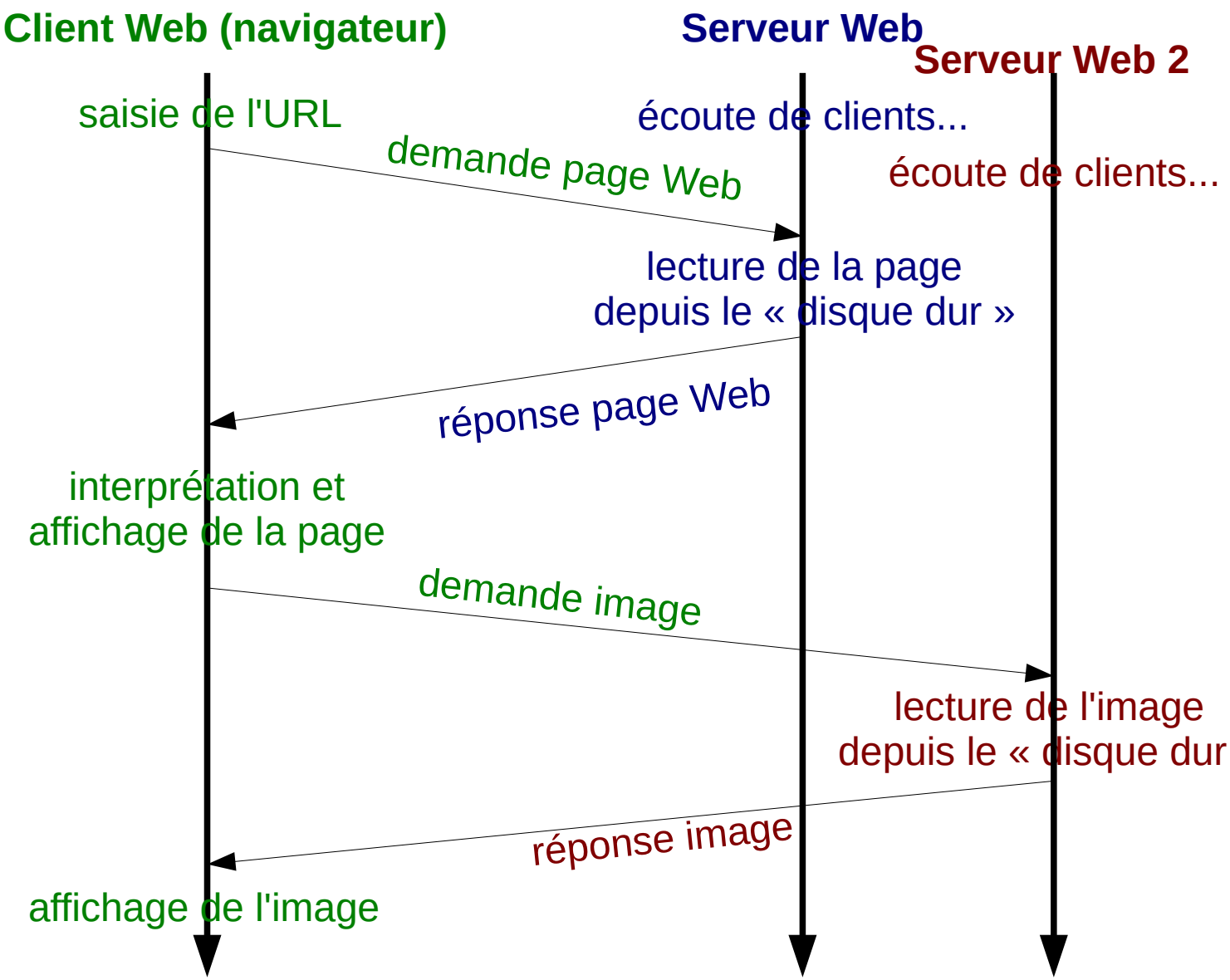
# Types d'images

- <img src="..." alt="...">
- Bitmap
  - non compressées : bmp
  - compressées :
    - sans perte d'information (*lossless*) : png – approprié pour des images avec peu de couleurs et régulières, comme les diagrammes, les graphiques et les captures d'écrans texte
    - avec perte d'information (*lossy*) : jpg – approprié pour des images avec de fortes variations de couleurs, comme les photos
- Vectorielles
  - non compressées : svg
- Éditeurs d'images : gimp, photoshop etc.
  - démo : réduire la taille des images




Source : http://upload.wikimedia.org/wikipedia/commons/a/aa/VectorBitmapExample.svg

# Protocole HTTP, transfert d'une page Web avec une image




HTTP est un protocole de niveau application, donc les échanges apparaissent comme données dans les paquets

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Page simple</title>
</head>
<body>
  Ma première page.
  <img src="..." alt="...">
</body>
</html>
```

Intérêt de mettre width et height de l'image

D'autres objets embarqués : css, éléments multimédia, ...

# En-têtes HTTP

**Demande page Web :**

```
GET /index.php HTTP/1.1
Host: rt.pu-pm.univ-fcomte.fr
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en
Accept-Encoding: gzip, deflate
Cookie: AGIMUS=...; wiki_rt_session=...; wiki_rtUserID=2; wiki_rtUserName=Eugen+Dedu; [...]
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Referer: ...
```

**Réponse page Web :**

```
HTTP/1.1 200 OK
Date: Fri, 03 Nov 2017 20:34:18 GMT
Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny16 with Suhosin-Patch
X-Powered-By: PHP/5.2.6-1+lenny16
Content-language: fr
Vary: Accept-Encoding,Cookie
X-Vary-Options: Cookie;string-contains=wiki_rtUserID;[...]
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: private, must-revalidate, max-age=0
Last-modified: Mon, 30 Oct 2017 11:43:44 GMT
Content-Encoding: gzip
Content-Length: 5451
Content-Type: text/html; charset=UTF-8
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive

<!DOCTYPE html>
<html>
...
```

# Les URLs

- protocole://host.domain:port/path/filename
  - protocole : http, https, file, ftp, ...
- Si filename n'est pas spécifié, ce sera index.html (ou index.php etc.) qui sera utilisé, s'il existe, sinon il affiche le répertoire
  - ex. avec http://eugen.dedu.free.fr et http://eugen.dedu.free.fr/teaching
- Si host.domain d'un lien n'est pas spécifié, le fichier sera pris du même serveur que la page HTML
  - ex. avec http://eugen.dedu.free.fr, partie ParSSAP
- Si file:, le navigateur lit lui-même le fichier depuis son disque dur, **sans** passer par le serveur Web

# Formulaires

- Servent à envoyer des données au serveur, en lui faisant une requête avec les données
- Champs classiques : texte (visible ou avec étoiles), cases à cocher, boutons radio, bouton d'envoi...

```
<form action="traitement.php">
Name: <input type="text" name="name"><br>
Password: <input type="password" name="pwd"><br>

<input type="checkbox" name="velo" value="oui">J'ai un vélo<br>
<label><input type="checkbox" name="voiture" value="oui">J'ai une voiture</label><br>

<input type="radio" name="chat" value="oui">Oui<br>
<input type="radio" name="chat" value="non">Non<br>

<input type="submit" value="Envoyer">
</form>
```

**traitement.php** :

```
<?php
    // affiche tous les paramètres
    foreach ($_GET as $key => $val)
        echo "$key = $val<br>";
?>
```

# Formulaires, quelques champs/attributs utiles

- Champs de saisie :
  - type="email" – champ texte qui valide la *syntaxe* de l'entrée
  - type="number" – sur smartphone affiche un clavier qu'avec des chiffres
  - type="range" – curseur entre deux valeurs
  - type="date" – affiche un calendrier pour choisir une date
- Attributs :
  - required – champ obligatoire
  - value, min, max, step
  - autofocus – reçoit le focus une fois la page chargée
  - placeholder="jj-mm-aaaa" – affiche de l'aide avant la saisie de l'utilisateur

# Éléments multimédia – audio

- La balise <audio> joue un fichier audio
- Attributs : autoplay, controls, loop, src et d'autres
- Plusieurs codecs supportés, voir [wikipedia] pour la liste complète
- Pas tous les navigateurs implémentent tous les codecs => mettre à disposition plusieurs formats de l'audio

```
<audio src="radio.ogg" controls>
Audio tag not supported.
</audio>
```

| Browser | Ogg Opus | Ogg Vorbis | MP3 and AAC |
|---|---|---|---|
| Firefox | O | O | O |
| Chrome | O | O | O |
| Edge | O | N | O |
| Safari | N | N | O |
| Opera | O | O | O |

```
<audio controls>
  <source src="radio.ogg">
  <source src="radio.mp3">
  Audio tag not supported.
</audio>
```

# Éléments multimédia – vidéo

- La balise <video> affiche une vidéo en streaming
- Attributs : autoplay, controls, height, width, loop, src et d'autres
- Conteneur = format du fichier ; codec audio/vidéo = méthode de compression audio/vidéo
- Plusieurs formats supportés, voir [wikipedia] pour la liste complète
- Pas tous les navigateurs implémentent tous les formats

```
<video src="movie.ogg" controls>
Video tag not supported.
</video>
```

| Browser | Ogg Theora | WebM VP9 | MP4 H.264 |
|---|---|---|---|
| Firefox | O | O | O |
| Chrome | O | O | O |
| Edge | N | O(hwaccl) | O |
| Safari | N | N | O |
| Opera | O | O | O |

```
<video controls>
  <source src="movie.webm" type="video/webm;
    codecs=vp9,vorbis">
  <source src="movie.mp4" type="video/mp4">
  Video tag not supported.
</video>
```

# Éléments multimédia – streaming adaptatif

- Les navigateurs sont en train d'implémenter le standard DASH (*Dynamic Adaptive Streaming over HTTP*), qui permet de faire de la vidéo adaptative (la qualité change à la volée en fonction de la qualité de la transmission)
  - chaque morceau de N secondes d'une vidéo est encodée en plusieurs bitrates dans des fichiers séparés
  - le navigateur charge la vidéo morceau par morceau
  - à chaque chargement d'un morceau, le navigateur choisit le fichier avec le bitrate approprié, en fonction de la vitesse de téléchargement du/des morceaux précédents
  - voir [mozilla] par exemple

# Les attributs événements

- But : permettre d'exécuter du code (JavaScript par ex.) lors d'événements
- Exemples d'événements :
  - sur la fenêtre :
    - onload – après chargement de la page
    - onresize – redimensionnement de la page
  - sur les formulaires :
    - onchange – la valeur d'un élément change
    - onfocus/onblur – réception/perte de focus
    - onsubmit – envoi de formulaire
  - sur le clavier : onkeypress
  - sur la souris : onclick, onmousemove
  - sur les médias : onplay, onended, onloadedmetadata, onpause
- Comme tout attribut, ils sont attachés à des éléments HTML :
  - <h1 onclick=”script”>...</h1>
  - où script peut être : alert ('Clic')

# Plan du cours

- Introduction
- Pages HTML
  - généralités
  - formulaires
  - multimédia
  - attributs événements
- CSS
- Responsive Web Design

# Feuilles de styles (CSS) – besoin

- Cascading Style Sheets
- Pour des sites avec beaucoup de pages, HTML seul a des limites
- Supposons qu'on souhaite utiliser <table border="1" bgcolor="silver" cellpadding="3" cellspacing="0"> pour les tableaux, pour uniformiser la présentation du site
- Inconvénients de HTML pur :
  - il faut réfléchir à la présentation pendant la rédaction des pages
  - il faut répéter cela pour tous les tableaux de toutes les pages du site
  - si on souhaite modifier ce style il faut remodifier tous les tableaux
  - ces styles sont identiques pour écran, imprimante, smartphone etc.
- Solution :
  - on spécifie le style des tableaux dans un fichier CSS
- Cela est similaire aux styles dans les texteur (par ex. Word), faire un exemple

# Fichier CSS

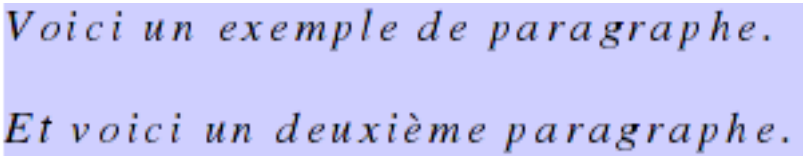
**Page HTML :**

```
...
<head>
<link href="styles.css" rel="stylesheet">
</head>

<body>
<p>Voici un exemple de paragraphe.</p>
<p>Et voici un deuxième paragraphe.</p>
</body>
...
```

**Fichier styles.css :**

```
body {
  background-color: blue;
  letter-spacing: .1em;
}
p {
  font-style: italic;
}
```



# Notions de CSS

- Sélecteur – spécifie à qui s'applique le style
- Quelques exemples :
  - p – tous les éléments p
  - .mon-style – tous les éléments avec class="mon-style"
  - #mon-style – l'élément avec id="mon-style"
  - ul li – tous les éléments li dans des éléments ul
  - [attr=valeur] – tous les éléments ayant un attribut attr=valeur
  - beaucoup plus

- Bloc de déclarations – spécifie le style
- Une ou plusieurs déclarations de type propriété:valeur, séparées par ;
  - font-weight: bold;
  - line-height: 1.3em;
  - color: red;
  - background-color: blue;
  - text-align: center;
  - margin-top: 3em;
  - text-indent: 2em;
  - border-bottom: 1px;
  - border-radius: 8px;
  - display: none;
    - https://sources.debian.org/src/ekiga/4.0.1-8, click to toggle, chercher toggle dans source
  - etc. etc.

**Fichier styles.css :**

```
p {
  font-style: italic;
  text-align: center;
}
```

# Trois méthodes d'utilisation des CSS

- <p **style="color: red"**>...
- => s'applique juste à ce 

- <head>...**<style>p {color: red}</style>**...</head>
- => s'applique à tous les  de la page

- Fichier **styles.css : p {color: red}**
- pages HTML : <head>...**<link href="styles.css" rel="stylesheet">**...</head>

  => s'applique à tous les  de toutes les pages incluant styles.css

  => les pages chargent plus vite, car le .css est téléchargé une seule fois

# Positionnement dans CSS – ne pas faire

- 3 types de positionnement
- Normal : en ligne (horizontalement) et bloc (verticalement)
  - les balises span (en ligne) et div (bloc) permettent de regrouper des éléments
- Floats : float:left ou right
  - clear:left ou right ou both : cet élément sera mis au-dessous de tous les éléments left ou right ou les deux
- Positionnement absolu, qui sort du flux normal d'affichage

**Ex. d'organisation de page :**

```
        Header (div)
-----------------------------
Menu            | Content
(div float:left)| (div)
-----------------------------
Footer (div clear:both)
```

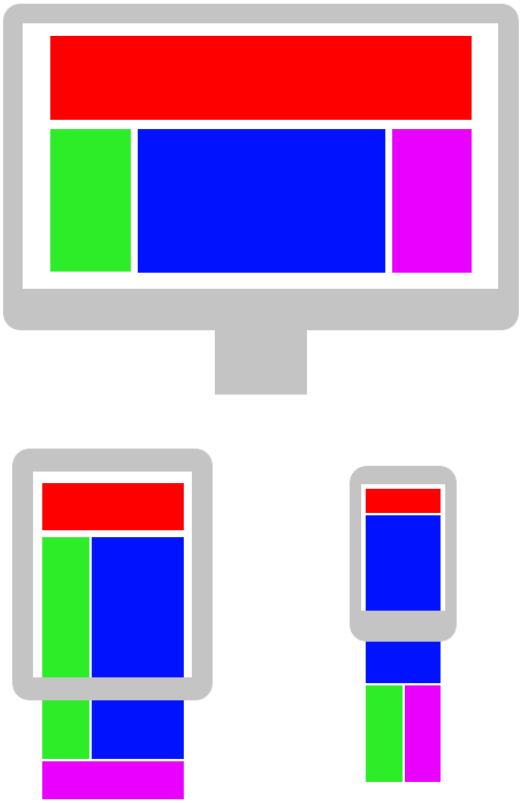
```
h2 {
  position:absolute;
  left:100px;
  top:150px;
}
```

# Plan du cours

- Introduction
- Pages HTML
  - généralités
  - formulaires
  - multimédia
  - attributs événements
- CSS
- Responsive Web Design

# Responsive Web Design

- Conception de site Web adaptatif (RWD) = ensemble de techniques pour rendre optimale la lecture et la navigation en s'adaptant aux caractéristiques du client (taille fenêtre, taille police, résolution etc.)
- Ex. : bibliographie multi-colonne dans wikipedia, rwd.html
- C'est un sujet encore en développement
  - comment afficher des tableaux sur de petits écrans ?
  - comment réorganiser le contenu pour les petits écrans ?



Source : https://en.wikipedia.org/wiki/Responsive_web_design

# Mauvais exemples de HTML

- Mauvaise utilisation des balises :
  - utilisation de ---- au lieu de <hr>
- Image de grande taille (gros fichier) affichée en petit [page perso, IPIN]
- Utilisation de checkbox au lieu de bouton radio
- Autofocus dans un champ qui n'est pas l'objet principal de la page

# RWD – mauvais exemples

- Taille fixe de la fenêtre
  - trop petite [hotnews], il y a des zones non utilisées
  - texte en dehors de la fenêtre, même la barre de défilement n'aide pas [coreboot]
- Police spécifiée en pixels
  - trop petite pour mon ordinateur qui a une grande résolution (ctrl-+ sous firefox avec mémorisation pour tout le site)
- Taille fixe des éléments
  - si j'agrandis la police, les caractères vont déborder l'élément [menu dans mitropolia, info dans les deux boîtes ns3]
- Impression difficilement lisible [agerpress]

# RWD – bons exemples

- S'adapte à la largeur de la fenêtre
  - change le nombre de colonnes de la bibliographie [wikipedia]
  - se réorganise [vtp]
  - change le nombre de colonnes, remplace plusieurs boutons en un seul, enlève de l'information si trop petit (météo), remplace text search par un bouton etc., voir haut de la page pp. et un article [ bostonglobe]
  - affiche ou non des menus, change le nombre de colonnes et leur largeur [agerpress, tweakers, erc]
  - a comme but le WRD [boursorama]
- Utilise les em (taille relative à la taille de la police courante) au lieu de pixels (taille absolue) pour changer la hauteur des caractères
- Ne précise pas en dur la taille des blocs (menus par ex.)
- S'adapte à la résolution
- S'adapte au média
  - la page imprimée n'a pas les menu en haut, à gauche et à droite [wikipedia]

# Hétérogénéité du média

- La page Web peut être vue sur plusieurs supports : écran (PC, tablette, smartphone), papier imprimé, Braille etc.
- De plus, les terminaux avec le même support peuvent avoir des caractéristiques différentes : largeur, hauteur, orientation etc.
- On souhaite s'adapter au support, par ex. ne pas afficher le menu si imprimé [wikipedia], ou bien afficher le menu à gauche si grande résolution et en haut si petite résolution
- Pour tester l'effet de la taille de la fenêtre sous firefox : menu Tools->WebDeveloper->ResponsiveDesignMode

# Solution technique : spécification du média

- Utiliser *media* (all, screen, print, braille etc.) avec différents fichiers CSS :
  - <link href="screen.css" **media="screen"** rel="stylesheet">
  - <link href="print.css" **media="print"** rel="stylesheet">
- Le navigateur choisit le média correspondant
  - les navigateurs desktop et de mobile choisissent tous les deux screen

# Spécification précise du média (taille de la fenêtre par ex.)

- Spécifier le type média avec une ou plusieurs expressions limitantes, chacune entre parenthèses (= média query)
- Expressions limitantes : width, height, color (nombre de bits par pixel), resolution (dpi), orientation etc.
  - les quatre premières acceptent max- et min- aussi

**Dans la page Web :**

```
<link rel="stylesheet" href="style.css"
    media="(max-width: 80em)">
```

**Dans le fichier css :**

```
@media (max-width: 80em) {
  ...
}
```

**Dans un fichier css à part importé par le fichier css :**

```
@import url("style2.css") (max-width: 80em);
```

# RWD – exemple

- Présenter le code source de rwd.html et rwd.css

# Conseil de RWD

- Utilisation de la taille de la police par défaut du navigateur (cf. *Préférences*) ; s'il faut absolument la changer, utiliser em (1em = hauteur de la police courante) au lieu de pixels ou points
- Multi-colonnage si largeur trop grande
- Redimensionnement éventuel d'images
- Éviter besoin de zoom, barre de défilement horizontale
- etc. etc.

- Défis actuels et futur de RWD [alsacreations] !!

# Conclusions

- Écrire des pages Web simples en HTML, y compris avec des formulaires et du multimédia, est abordable
  - mais si on veut des effets spéciaux cela devient complexe et il faut aussi avoir la fibre artistique
- Les styles sont très utiles pour un site de plus de quelques pages
- Les média sont hétérogènes (taille police par défaut, taille écran, différents médias etc.) et un bon site prend en compte cela
  - mais beaucoup de sites réels ne passeraient pas l'examen...