Licence informatique, 2020-2021

Programmation de sites Web

Gilles Subrenat[1]
Rita Zrour[1]

[1]Laboratoire SIC
Gilles.Subrenat@univ-poitiers.fr
Rita.Zrour@univ-poitiers.fr

Licence I troisième année, 2020-2021

Plan → A - Généralités

- Web
  - $\neq$ Internet
  - $=$ ensemble d'applications s'appuyant sur Internet
  - Autres applications sur Internet
    - courrier électronique, chat
    - jeux
    - ...
  - Définition
    - "un système hypertexte public fonctionnant sur Internet et qui permet de consulter, avec un navigateur, des pages mises en ligne dans des sites"
- Système hypertexte
  - lie des documents entre eux par des hyperliens
- Hyperlien
  - permet de passer automatiquement d'un document à un autre
  - est unidirectionnel (la cible n'est pas consciente d'être visée)
  - peut être incohérent (erreur 404)

Source : *http://fr.wikipedia.org/wiki/World_Wide_Web*

- 13 mars 1989
  - Invention du Web par Tim Berners-Lee
  - Tim Berners-Lee est actuellement président du W3C
    - World Wide Web Consortium
    - gère les standards/normes du Web
  - NB : notion d'hyperlien date de 1945
    - autres utilisations d'hyperliens : PDF, encyclopédies, ...
- 1990
  - Technologies URL, HTML, HTTP
  - Premier site
- 1992
  - 26 sites "fiables"
- 1993
  - Web passe dans le domaine public
  - Navigateurs NCSA Mosaic et Lynx
  - $\approx$ 600 sites

- 1994
  - Site Yahoo
  - Navigateur Netscape Navigator 1.0
  - $\approx$ 10 000 sites
- 1995
  - Serveur Apache, navigateur Internet Explorer (IE 1.0 et IE 2.0)
  - Moteur de recherche Altavista
  - Technologies Java et JavaScript
  - $\approx$ 25 000 sites
- 1996
  - Navigateurs Netscape 2.0 et 3.0, IE 3.0, Opera 2.1
  - Technologie CSS level 1
- 1997
  - Navigateurs Netscape 4.0 et 3.0, IE 4.0
  - Standards HTML 3.2, HTML 4.0
  - $\approx$ 1 000 000 sites

- 1998
  - Site Google
  - Technologies CSS level 2, DOM level 1
- 1999
  - Navigateur IE 5.0
  - Standard HTML 4.01
- 2000
  - Standard XHTML 1.0
  - $\approx$ 20 000 000 sites
- 2001
  - Navigateur IE 6
  - Site Wikipedia
- 2002
  - Navigateur Mozilla 1.0
- 2003
  - Navigateur Safari

- 2004
  - Concept du Web 2.0
  - Navigateur Mozilla Firefox 1.5
  - $\approx$ 60 000 000 sites
- 2006
  - Navigateurs Firefox 2.0, IE 7
  - $\approx$ 100 000 000 sites
- 2008
  - Navigateurs Firefox 3.0, Google Chrome 0.2
  - $\approx$ 180 000 000 sites
- 2010
  - Navigateurs Firefox 3.6, Safari 5, Google Chrome 8
- 2012
  - Navigateurs Firefox 16, Safari 6, Google Chrome 19, IE 9
  - $\approx$ 630 000 000 sites

- Evolution des sites
  - Sites statiques
    - contenu figé, écrit en dur par les concepteurs
  - Sites dynamiques
    - contenu construit à la volée
    - dépend des actions de l'internaute
    - sites couplés avec des bases de données
    - → site commercial comme exemple typique
- Conséquences
  - vrais langages de programmation pour gérer le coté dynamique
  - ⇒ la création est devenue un vrai travail de programmation
    - Existence des CMS (*Content Management System*)
    - → Joomla, SPIP, Wordpress, ...
    - → gestion de sites dynamiques sans (beaucoup de) programmation
  - ⇒ la gestion de la sécurité est cruciale

Plan → A - Généralités

- Web terminologie exacte :
  - World Wide Web
- Web noms usuels :
  - Web, WWW, toile, W3, ...
- *en ligne* :
  - connecté au réseau
- *hôte* :
  - ordinateur en ligne référencé par une *adresse IP*
  - un hôte peut avoir plusieurs noms (notion de DNS).
- *ressource* :
  - "document" (texte, image, son, ...) accessible (localement ou à distance)
  - nécessite le respect d'un *protocole de communication*.
- *HTTP* (HyperText Transfer Protocol)
  - protocole classique d'accès aux ressources du Web
  - *HTTPS* est la version sécurisée (cryptée) de ce protocole

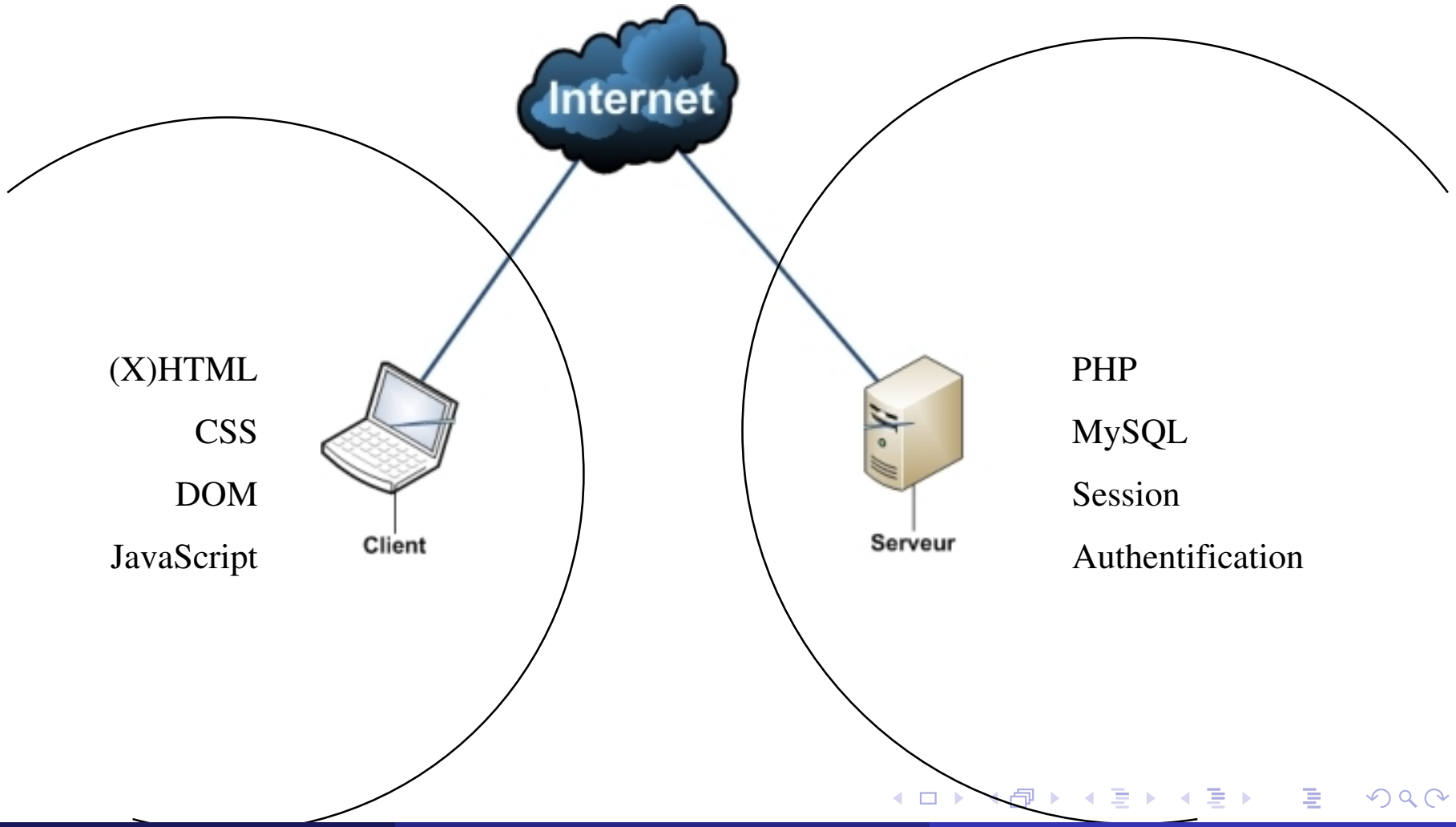
- *URL* (Uniform Ressource Locator)
  - chaîne indiquant l'emplacement d'une ressource
- *hyperlien*
  - élément associé à une ressource.
- *(X)HTML* (eXtensible HyperText Markup Language)
  - langage décrivant un document (contenu et mise en forme).
- *CSS* (Cascading Style Sheet)
  - langage décrivant la mise en forme d'un document (X)HTML
- *clients-serveurs*
  - un serveur est un hôte sur lequel tourne un logiciel serveur
  - les logiciels clients se connectent au logiciel serveur
- *serveur web*
  - hôte sur lequel tourne un serveur HTTP
- *navigateur web*
  - logiciel client HTTP récupérant et affichant des ressources
  - d'autres protocoles peuvent être supportés.

- *page web*
  - document, généralement architecturé autour du langage HTML
  - affichable par un navigateur.
- *site web*
  - ensembles de ressources cohérentes entre elles
- *côté client*
  - ce qui s'exécute sur la machine de l'internaute (navigateur)
  - affichage (interprétation du HTML), JavaScript
- *côté serveur*
  - ce qui s'exécute sur la machine hébergeant le site
  - bases de données, PHP/ASP/...

- Technologies à foison
  - HTML, XHTML, XML, CSS, XSLT, JavaScript, PHP, ASP, JSP, JSF, SQL, MySQL, PostgreSQL, Oracle, Hibernate, ODBC, JDBC, Ajax, J2EE, Servlets, Applets, Apache, IIS, ...
- Bibliothèques à foison
  - Prototype, Overlib, JQuery, ZendFramework, Symfony, CakePHP, ...
- Concepts/problématiques à foison
  - architecture n-tiers, architecture clients-serveurs, Web 2.0, sécurité/piratage (SQL Injection, session hijacking, cross-site scripting, ...), authentification, ...
- Gestionnaires de contenu à foison
  - SPIP, Wiki, PhpBB, Joomla, Wordpress, Moodle, ...
- ...
- Cadre du cours
  - étude partielle de quelques technologies et concepts

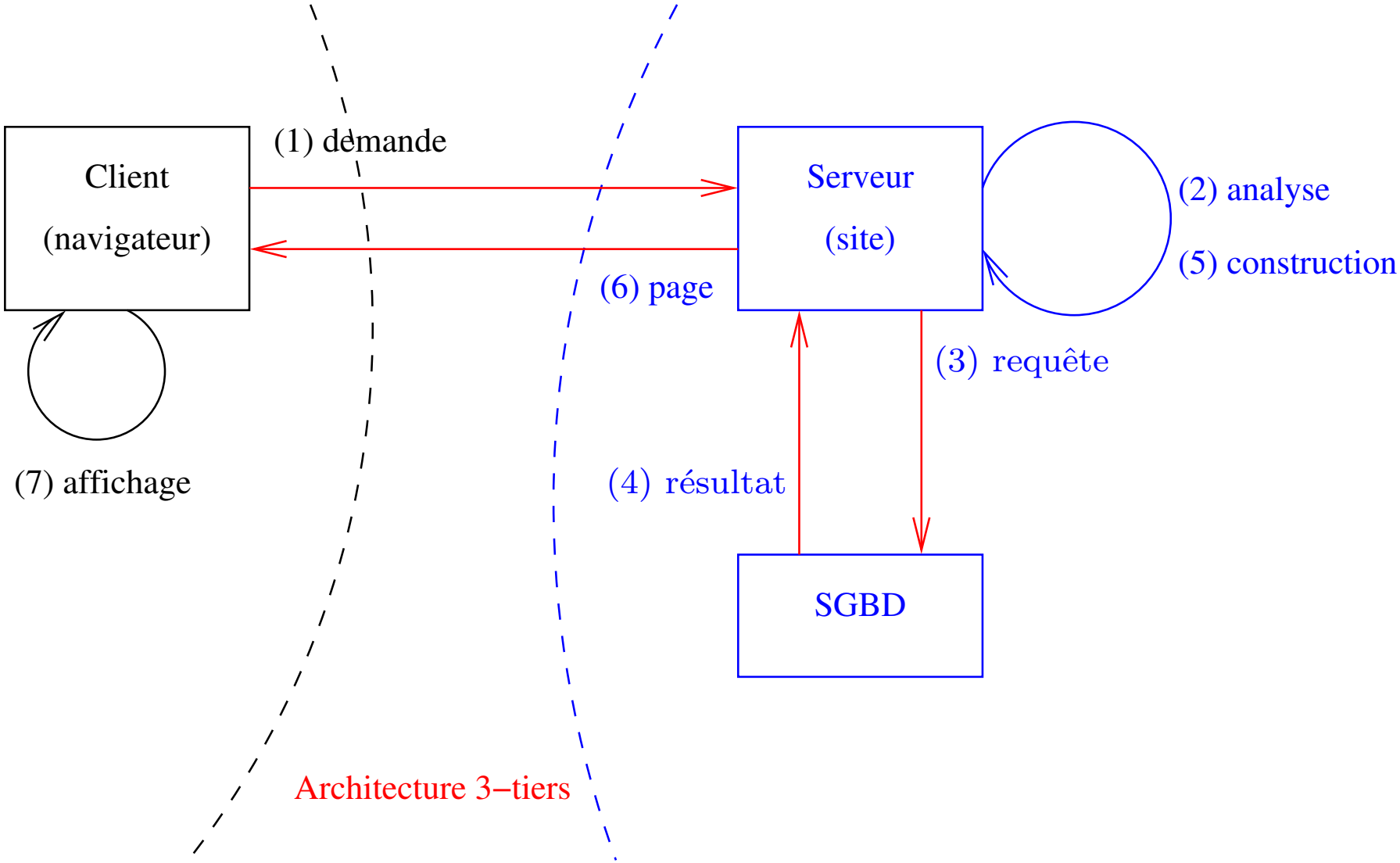
- But
  - faire un site un peu évolué
  - faire un site dynamique (en relation avec une base de données)
  - gérer les authentifications
  - appréhender les problèmes liés à la sécurité
- Technologies
  - XHTML
  - CSS
  - DOM
  - JavaScript
  - PHP
  - MySQL
  - Sessions utilisateurs, variables de session
  - Authentification
  - Sécurité (sensibilisation)

- Vue schématique



Plan → B - Survol

- Deux acteurs
  - *client* : navigateur (utilisateur)
  - *serveur* : hébergeur des ressources Web
- Cheminement d'une requête
  1. *demande* : le client demande une page au serveur
     - exemple on tape : *http://monsite.com/articles/voir.php?id=4*
  2. *analyse* : le serveur traite la demande
     - langages *PHP*, *ASP*, *JSP*, ...
  3. *interrogation BD* : le serveur interroge la base de données
     - systèmes *MySQL*, *Oracle*, *SQLite*, ...
  4. *résultat BD* : et récupère le résultat
  5. *construction page* : le serveur construit le code XHTML/JavaScript
  6. *envoi résultat* : et l'envoie au client
  7. *affichage* : le client interprète le code et l'affiche
  - site statique : seules les étapes 1, 6 et 7 sont effectuées


Client
(navigateur)
(1) demande
Serveur
(site)
(2) analyse
(5) construction
(6) page
(3) requête
(4) résultat
(7) affichage
SGBD
Architecture 3–tiers

Plan → B - Survol

- W3C
  - World Wide Web Consortium
  - *http://www.w3.org*
  - organisme de normalisation
  - rappel : un grand nombre de technologies gravitent autour du Web
    - langages de description
    - langages de programmation
    - $\Rightarrow$ organisme de standardisation
- (X)HTML (eXtensible HyperText Markup Language)
  - Langage de description de document à base de balises
    - exemple : * ... *
  - Objectifs
    - structurer sémantiquement une page Web
    - mettre en forme : à éviter (cf. CSS)
    - inclure des ressources multimédia (images, ...)
    - inclure des formulaires de saisie
    - inclure des éléments programmables (applets)
  - Associé à des feuillles de style (CSS)
  - Associé à du code JavaScript

- CSS (Cascading Style Sheets)
  - Feuille de style décrivant la présentation d'un document XHTML
  - Externaliser les aspects de présentation
    - $\Rightarrow$ uniformiser la présentation d'un site
    - $\Rightarrow$ modifications aisées
  - Principe
    - un fichier (X)HTML : le contenu et la sémantique
    - un fichier CSS : la présentation
    - chaque page du site inclut le même fichier CSS
- DOM (Document Object Model)
  - Description arborescente d'une interface (page HTML)
  - Modifiable dynamiquement
    - par une action de l'utilisateur
    - via un langage de programmation (JavaScript)
  - Exemple
    - l'utilisateur clique sur un mot
    - une partie du texte disparait (ou apparait)

- JavaScript
  - Ce n'est pas Java !
  - Langage orienté objet principalement utilisé pour les pages Web interactives
  - Exécuté dans le navigateur
  - Utilisation
    - vérifier des formulaires
    - piloter le DOM
- PHP (PHP : Hypertext Preprocessor)
  - Langage de scripts orienté objet
  - Exécuté sur le serveur
  - Génère du code HTML à la volée
  - Accède à une base de données
- SQL (Structured Query Language)
  - Langage d'interrogation de bases de données relationnelles

Plan → B - Survol

1. Schéma de communication
2. Technologies utilisées
3. Outils de gestion/conception
   - Plateforme d'exploitation (serveur)
   - Environnements de développement
   - Navigateurs
   - Plugins Firefox

- WAMP (Windows)
  - Windows Apache Mysql PHP
  - Produits
    - WampServer : *http://www.wampserver.com/*
    - EasyPHP : *http://www.easyphp.org*
  - Installation simple (en une fois)
  - Intègrent PhpMyAdmin (gestion de bases de données)
  - Destinés à un développement local (i.e. non production)
  - IIS : outil de production
- LAMP (Linux)
  - Linux Apache Mysql PHP
  - Ensemble de de paquetages indépendants et en interaction
    - installation/configuration plus délicates
    - environnement de production
    - *http://doc.ubuntu-fr.org/apache*
  - XAMPP : installation tout en un

- Outils graphiques
  - The Gimp
  - Photoshop
- Editeurs
  - Notepad++ : site *http://notepad-plus.sourceforge.net*
  - Kate : sous Linux
  - Kompozer : site *http://www.kompozer.net*
    - mode WYSIWYG disponible
  - Sublimetext, ...
- IDE (Environnement de Développement Intégré)
  - bluefish : site *http://bluefish.openoffice.nl*
    - dédié au développement Web
  - Netbeans : site *http://www.netbeans.org*
  - Eclipse : site *http://www.eclipse.org*
  - PHPStorm, ...

- Il en existe un grand nombre

Nomination	12/2015	12/2013	12/2011	12/2010	
Internet Explorer	48.6 %	58.2 %	51.9 %	59.3 %	Windov
Firefox	12.1 %	18.2 %	21.8 %	23.7 %	“tous”
Chrome	32.3 %	16.4 %	19.1 %	10.4 %	“tous”
Safari	4.5 %	5.8 %	5.0 %	4.0 %	MacOS
Opera	1.6 %	1.3 %	1.7 %	2.3 %	“tous”
autres	0.9 %	0.4 %	0.6 %	0.4 %	

- source : *http://marketshare.hitslink.com* (ordinateurs fixes)
- navigateur récent ⇒ meilleur respect des normes
  - Internet Explorer respecte de plus en plus la norme

- Beaucoup de plugins facilitent le développement Web
  - source : http://www.framasoft.net/article4783.html
- FireBug
  - édition de code XHTML, CSS, code JavaScript, DOM
  - site : https://addons.mozilla.org/fr/firefox/addon/1843
- WebDeveloppper
  - outil similaire, manipulation des paramètres de la page
  - site : https://addons.mozilla.org/fr/firefox/addon/60
- ProfileSwitcher
  - ouvrir plusieurs profils simultanément (sans déconnexion/reconnexion)
- ColorZilla
  - récupérer une couleur sur une page Web
  - site : https://addons.mozilla.org/fr/firefox/addon/271
- FirePHP, YSlow, Tamper Data, Selenium et iMacros, Search Status, Stylish et GreaseMonkey, FireFTP, IETab, HTML Validator, ...

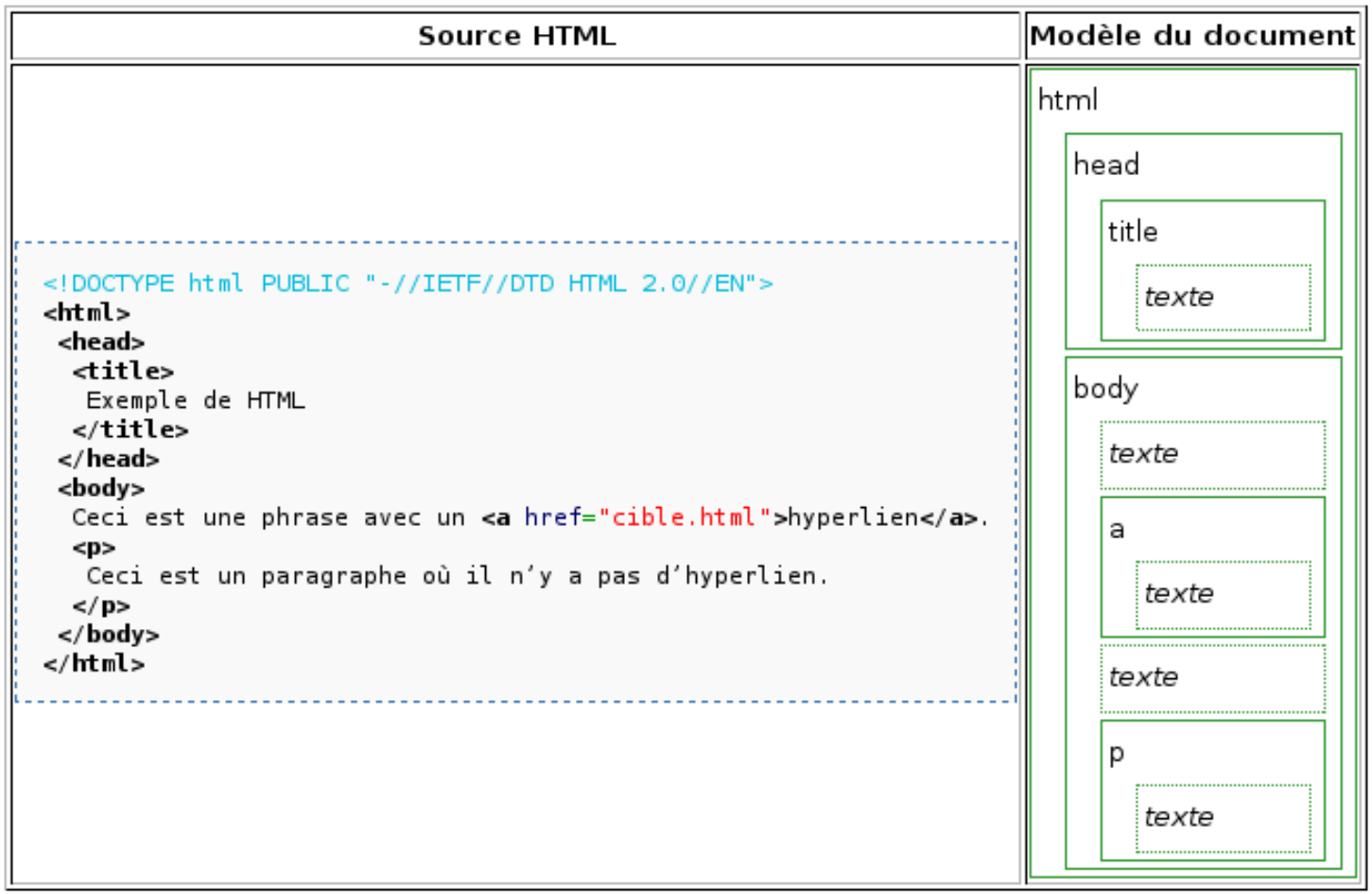
Troisième partie III

XHTML

Plan - C - XHTML

Plan → C - XHTML

- XHTML : structure arborescente
  - utilisée par CSS et JavaScript



  - style appliqué à un noeud : hérité par les fils

- XHTML : squelette d'une page
  - doctype
  - document
    - entête
    - corps

```
1  <!DOCTYPE ...>
2  <html>
3     <head>
4        ...
5     </head>
6
7     <body>
8        ...
9     </body>
10 </html>
```

- Tout se situe dans ou entre des balises
  - $\Rightarrow$ structuration du document
- Balises par paire
  - balise ouvrante et balise fermante
  - syntaxe : <nom_balise> < /nom_balise>
  - exemple :

```
<strong>texte important</strong>
```

- Balises seules ou auto-fermantes
  - syntaxe : <nom_balise />
  - exemple :

```
<br />
```

- Une balise peut contenir des attributs
  - certains sont obligatoires
  - syntaxe : <nom_balise nom_attr1="val1" nom_attr2="val2" ...>
  - exemple :

```
<img src="soleil.png" alt="un soleil" />
```

- Doctype : indique la norme utilisée
  - → le langage et sa version : HTML 4.01, XML 1.0, ...
  - → le dialecte : transitional, strict, ...
- Placé en tête de fichier
- Choix du cours : XHTML 1.0 Strict
  - cf. *http ://www.w3.org/TR/xhtml1/*

```
1 <!DOCTYPE html
2     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- Validité d'un document
  - le choix d'un doctype impose un langage et sa syntaxe
  - il faut respecter scrupuleusement cette syntaxe

  ⇒ affichage correct dans un navigateur

  ⇒ bonne prise en compte par les systèmes pour mal-voyants
  - site de validation : *http ://validator.w3.org/*

- Modèle d'une page minimale

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html
3      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
4      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
5  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
6    <head>
7      <title>Mon titre</title>
8      <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
9    </head>
10   <body>
11     <p>Mon paragraphe</p>
12   </body>
13 </html>
```

- balise *html* englobe tout le document
  - attributs *xmlns* (obligatoire), *xml :lang* et *lang* (valeurs : *en*, *fr*, ...)
- balise *head*
  - balise *title* : titre de la fenêtre du navigateur
  - balise *meta* : jeu de caractères utilisé (UTF-8, iso-8859-1, ...)
  - note : grand nombre de balises *meta* : référencement par les robots, ...
- balise *body*
  - le contenu proprement dit

- Commentaires
  - syntaxe : entre <!-- et -->
  - exemple : `<!-- un commentaire -->`
  - note : commentaires visibles par l'internaute
- Caractères particuliers
  - Certains caractères ne doivent ou ne devraient pas s'écrire tels quels
  - & : `&` (obligatoire)
  - < : `<` (obligatoire)
  - > : `>`
  - " : `&quot;`
  - ' : `&apos;`
  - espace insécable : ` `

Plan → C - XHTML

- Paragraphe
  - (presque) tout texte est dans un paragraphe
  - syntaxe : et
- Saut de ligne
  - solution 1 : créer un nouveau paragraphe
  - à l'intérieur d'un paragraphe :
  - avec un trait horizontal : <hr />

```
1  <body>
2    <p> Mon premier paragraphe </p>
3    <p>
4      Mon second paragraphe. <br />
5      La suite.
6    </p>
7    <hr />
8    <p> Et un dernier. </p>
9  </body>
```

  - les espaces/retours-chariot multiples n'ont aucun effet

- Titres
  - 6 niveaux (par importance décroissante)
    - <h1> et </h1>
    - <h2> et </h2>
    - <h3> et </h3>
    - <h4> et </h4>
    - <h5> et </h5>
    - <h6> et </h6>
  - Signification sémantique
    - indiquer un titre de premier ordre, secondaire, ...
    - NE PAS utiliser pour grossir les caractères

    → utilisation des fichiers CSS pour la mise en forme

    - "incorrect" : `<h1 style="color: red;">Mon titre</h1>`
    - "correct" : `<h1>Mon titre</h1>` + fichier CSS

- Mise en valeur
  - Mise en valeur faible
    - syntaxe : et
    - exemple :
```
... les invités, dont le <em>maire</em> de la commune, vont ...
```
  - Mise en valeur forte
    - syntaxe : et
    - exemple :
```
un point crucial est la <strong>sécurité</strong> des transactions
```
- Divers
  - Exposant
    - syntaxe : ^{et}, ...
  - Indice
    - syntaxe : _{et}, ...
  - Citation courte (dans un paragraphe)
    - syntaxe : <q> et </q>, ...
  - Citation longue (dans un bloc à part)
    - syntaxe : <blockquote> et </blockquote>, ...

- Divers
  - Code (programmation) dans un bloc
    - syntaxe : <code> et </code>, ...
  - Acronyme
    - syntaxe : <acronym> et </acronym>, ...
    - exemple : `<acronym title="Fausse Bonne Idée">FBI</acronym>`
  - Affichage tel quel
    - syntaxe : <pre> et </pre>, ...
  - Attribut title : infobulle "courte"
    - applicable sur presque toutes les balises
- Balises *inline* et *bloc*
  - Balise *inline*
    - pour désigner un élément dans un paragraphe
    - exemple : , <img>, ...
  - Balise *bloc*
    - pour avoir un bloc (≈ paragraphe) séparé
    - exemple : <p>, <h1>, ...

- Structuration
  - Balise span (mode inline)
    - pour encadrer du texte dans un paragraphe
    - a priori aucun effet visuel (on donne un style via CSS)
    - syntaxe : et
  - Balise div (mode bloc)
    - pour encadrer un bloc sémantiquement cohérent

    → ajoute un noeud dans le DOM
    - a priori aucun effet visuel (on donne un style via CSS)
    - syntaxe : et

```
1  <body>
2     <div id="menu">
3        <p> ... </p>
4     </div>
5     <div id="corps">
6        <div id="intro">
7           <p> ... le <span class="technique">mail</span> est un outil ... </p>
8        </div>
9        <div id="argumentation">
10          <p> ... </p>
11       </div>
12    </div>
13 </body>
```

Plan → C - XHTML

- Liste non ordonnée (à puces)
  - et pour délimiter la liste
  - et pour délimiter un item
- Liste ordonnée (numérotée)
  - et pour délimiter la liste
  - et pour délimiter un item

```
1  <body>
2     <ul>
3        <li> hanball</li>
4        <li> football</li>
5        <li> curling</li>
6     </ul>
7
8     <ol>
9        <li> kilogramme</li>
10       <li> quintal</li>
11       <li> tonne</li>
12    </ol>
13
```

```
1     <ul>
2        <li>
3           handball
4           <ul>
5              <li> arbitre </li>
6              <li>
7                 joueur
8                 <ul>
9                    <li> actif </li>
10                   <li> remplaçant </li>
11                </ul>
12             </li>
13             <li> spectateur </li>
14          </ul>
15       </li>
16       <li> curling </li>
17    </ul>
18 </body>
```

- Liste de termes avec définitions
  - <dl> et </dl> pour délimiter la liste
  - <dt> et </dt> pour délimiter un terme
  - <dd> et </dd> pour délimiter une définition

```
1   <body>
2     <dl>
3       <dt> Donnée </dt>
4         <dd> cédée définitivement </dd>
5         <dd> information(s) en informatique </dd>
6       <dt> Vendu </dt>
7         <dd> cédé contre argent </dd>
8         <dd> traître </dd>
9     </dl>
10  </body>
```

- Plus loin avec les listes
  - choix du type de puce (carré, cercle, ...)
  - choix de la numérotation (romain, décimal, ...)
  - choix du début de la numérotation
  - choix du retrait
  - ...

  → encore une fois en CSS

Plan → C - XHTML

- Balises élémentaires
  - <table> et </table> pour délimiter le tableau
  - <tr> et </tr> pour délimiter une ligne
  - <td> et </td> pour délimiter une case
  - <th> et </th> pour délimiter une case d'entête
  - <caption> et </caption> pour donner une légende au tableau
- Structuration
  1. <thead> et </thead> pour encadrer les lignes d'entête
  2. <tfoot> et </tfoot> pour encadrer les lignes de "pied" de tableau
  3. <tbody> et </tbody> pour encadrer les lignes du corps du tableau

  → attention : l'ordre de déclaration est impératif
- Fusion de cellules
  - attribut <rowspan> : fusion verticale
  - attribut <colspan> : fusion horizontale

  → cf. exemple
- Mise en forme
  - bordures, espacement inter-cases, ...

  → fichiers de configuration CSS

- Exemple

```
<body>
  <table class="normal">
    <caption> Titre tableau </caption>

    <thead>
      <tr>
        <th> colonne 1 </th>
        <th> colonne 2 </th>
        <th> colonne 3 </th>
      </tr>
    </thead>

    <tfoot>
      <tr>
        <th> bas 1 </th>
        <th> bas 2 </th>
        <th> bas 3 </th>
      </tr>
    </tfoot>

    <tbody>
      <tr>
        <td>case 11 </td>
        <td>case 12 </td>
        <td>case 13 </td>
      </tr>
      <tr>
        <td>case 21 </td>
        <td>case 22 </td>
        <td>case 23 </td>
      </tr>
      <tr>
        <td rowspan="2">fusion v </td>
        <td>case 32 </td>
        <td>case 33 </td>
      </tr>
      <tr>
        <!-- donc pas de case ici -->
        <td>case 42 </td>
        <td>case 43 </td>
      </tr>
      <tr>
        <td colspan="2">fusion h </td>
        <!-- donc pas de case ici -->
        <td>case 53 </td>
      </tr>
      <tr>
        <td>case 61 </td>
        <td colspan="2">fusion h </td>
        <!-- donc pas de case ici -->
      </tr>
    </tbody>
  </table>
</body>
```

Plan → C - XHTML

- 3 types d'images supportés par les navigateurs
  - .jpg (ou .jpeg) : pour les photos
  - .png : gère la transparence
  - .gif : gère la transparence et les animations
- Syntaxe :
  - balise auto-fermante
  - attribut src obligatoire : image à afficher
    - image locale via le chemin : *img/photo/soleil.png*
    - image distante via une URL : *http ://photos.com/soleil.png*
  - attribut alt obligatoire (en XHTML)
    - texte à afficher si l'image ne peut être affichée
    - primordial pour les mal-voyants par exemple
  - attribut title facultatif : infobulle
  - attributs width et height : préindique les dimensions de l'image
    - ne déforme pas la page si l'image est inaccessible

```
<img src="img/photos/soleil.jpg" alt="un soleil" title="vacances !" width="300" height="200" />
```

Plan → C - XHTML

- Hyperlien relatif
  - désigne une page du même site : *"inscription/formulaire.html"*
  - à utiliser obligatoirement pour cibler une page du même site
- Hyperlien absolu
  - vers une page d'un autre site : *"http ://mon.site.com/accueil"*
- Syntaxe
  - balises <a> et
  - attribut href pour désigner la cible
  - entre les balises : texte cliquable
  - attribut target="_blank" interdit en XHTML
    - force l'ouverture dans une nouvelle fenêtre
    - explicable, mais discutable aussi

```
<a href="informatique/intro.html" title="pour commencer">allez voir une intro</a>
<a href="http://mon.site.com/toto.html" title="ne mène nulle part">chez moi</a>
```

- Lien pour envoyer un mail
  - on remplace *http* par *mailto* dans l'attribut *href*

```
<a href="mailto:toto@titi.com" title="envoyer un mail">écrivez-moi</a>
```

- Ancre
  - définition : point de repère dans une page
  - intérêt : sert de cible à un lien pour arriver directement en un endroit précis d'une page.
  - création
    - ajout de l'attribut id dans une balise (souvent un titre)
    - ```
      <h2 id="nom_de_l_ancre">titre</h2>
      ```
    - note : l'attribut id est utilisé également par CSS et Javascript
    - note : la valeur d'un identifiant doit être unique dans une page
  - lien
    - syntaxe pour un lien à l'intérieur de la page courante
    - ```
      <a href="#nom_de_l_ancre">cliquez ici</a>
      ```
    - syntaxe pour un lien vers une ancre d'une page extérieure
    - ```
      <a href="http://mon.site.com/toto.html#nom_de_l_ancre">cliquez ici</a>
      ```
      ```

Plan → C - XHTML

- But
  - saisie d'informations par l'internaute
  - (facultatif) analyse par JavaScript
  - récupérées par le serveur du site (code PHP par exemple)
  - ou envoyées par mail
- Syntaxe
  - balises <form> et </form>
  - attributs
    - id : pour que JavaScript fasse des vérifications et pour CSS
    - méthod : méthode d'envoi des données get ou post
    - action : destinataire du formulaire (mail ou autre page)
  - méthode get
    - données transmises dans la barre d'adresse
    - taille des données limitée
  - méthode post
    - données transmises en arrière-plan
    - taille (presque) non limitée
    - possibilité d'envoyer des fichiers

- Exemple d'entête

```
1   <form id="form1" method="post" action="traitement.php">
2     <!-- items du formulaire ici -->
3   </form>
4
5   <form id="form2" method="get" action="traitement.php">
6     <!-- items du formulaire ici -->
7   </form>
8
9   <!-- on note l attribut enctype pour l envoi par mail -->
10  <form id="form3" method="post" action="mailto:moi@france.fr" enctype="text/plain">
11    <!-- items du formulaire ici -->
12  </form>
```

- Éléments d'un formulaire
  - différents items de saisie
    - zones de texte, listes déroulantes, cases à cocher, ...
    - bouton de validation et bouton de réinitialisation
    - directive de structuration (blocs d'items)
  - attribut name
    - indispensable pour l'exploitation des données (sur le serveur)
  - les items sont entre les balises et

- Zone de saisie texte
  - Saisie au clavier sur une seule ligne
  - Balise + attribut : <input type="text" />
  - Attributs
    - value : pour préremplir le champ
    - readonly : pour interdire les modifications
    - size : largeur de la zone
    - maxlength : nombre maximum de caractères autorisés
    - name : pour le serveur (nom de variable)

```
1  <form id="form1" method="post" action="traitement.php">
2    <p>
3      <input type="text" name="nom" />
4    </p>
5    <p>
6      <input type="text" name="login" value="Batman" readonly="readonly" />
7    </p>
8    <p>
9      <input type="text" name="prenom" value="Paul" size="10" maxlength="30" />
10   </p>
11 </form>
```

- Label
  - Texte illustrant/expliquant un élément de saisie
  - Balises : <label> et </label>
  - Un item de saisie DOIT être relié à un label
    - il faut un attribut id : pour l'item relié
    - attribut for dans la balise label

```
1  <form id="f1" method="post" action="traitement.php">
2    <p>
3      <label for="f1_nom">Votre nom :</label>
4      <input type="text" id ="f1_nom" name="nom" />
5    </p>
6  </form>
```

  - Intérêts
    - clic sur le label ⇒ sélection de l'item
      → prise de focus pour une zone texte
      → sélection pour une case à cocher ou un radio-bouton
    - navigation pour mal-voyants facilitée

- Mot de passe
  - même syntaxe qu'une zone de saisie de texte
  - sauf pour l'attribut type="password"
  - saisie clavier masquée
- Zone de texte multi-lignes
  - Balises : <textarea> et </textarea> (et non *input*)
  - Attributs
    - rows et cols (obligatoires) : taille de la zone
    - attributs id, name, readonly : cf. zone de texte
    - texte entre les balises : valeur par défaut

```
1  <form id="form1" method="post" action="traitement.php">
2    <p>
3      <label for="f1_cv">CV</label><br />
4      <textarea id="f1_cv" name="cv" rows="4" cols="40">
5        votre cursus ici
6      </textarea>
7    </p>
8  </form>
```

- Cases à cocher
  - Choix *oui/non* pour une proposition
  - 0, 1 ou plusieurs sélections parmi un ensemble de propositions
  - Balise + attribut : <input type="checkbox" />
    - balise à répéter pour chaque case
  - Attribut checked="checked" : pour précocher une case
  - Attribut name : pour le serveur (nom de variable)
    - précision technique : pour toutes les cases d'un même groupe (i.e. cases sémantiquement liées), on donne le même nom de variable suivi d'une paire de crochets (variable de type tableau)
  - Attribut value : valeur à envoyer au serveur
  - on n'oublie pas les labels : primordiaux ici

```
1   <form id="f1" method="post" action="traitement.php">
2     <p>
3       Vos plats préférés <br />
4       <input type="checkbox" name="plats[]" value="pizza" id="f1_piz" />
5       <label for="f1_piz">Pizza</label><br />
6       <input type="checkbox" name="plats[]" value="hamburger" id="f1_ham" checked="checked" />
7       <label for="f1_ham">Hamburger</label><br />
8       <input type="checkbox" name="plats[]" value="frite" id="f1_fri" />
9       <label for="f1_fri">Frites</label><br />
10    </p>
11  </form>
```

- Boutons radio
  - 1 (ou 0) sélection parmi un ensemble de propositions
  - Balise + attribut : <input type="radio">
    - balise à répéter pour chaque bouton
  - Attribut checked="checked" : pour précocher un bouton
  - Attribut name : le même pour tous les boutons d'un même groupe
  - Attribut value : valeur à envoyer au serveur
  - on n'oublie pas les labels : primordiaux ici

```
1  <form id="f1" method="post" action="traitement.php">
2    <p>
3      Votre état civil <br />
4      <input type="radio" name="civil" value="Mme" id="f1_civil_mme" />
5      <label for="f1_civil_mme">Madame</label><br />
6      <input type="radio" name="civil" value="Mlle" id="f1_civil_mlle" />
7      <label for="f1_civil_mlle">Mademoiselle</label><br />
8      <input type="radio" name="civil" value="M" id="f1_civil_m" />
9      <label for="f1_civil_m">Monsieur</label><br />
10     Votre âge <br />
11     <input type="radio" name="age" value="17-" id="f1_age_mineur" />
12     <label for="f1_age_mineur">17 ans ou moins</label><br />
13     <input type="radio" name="age" value="18+" id="f1_age_majeur" checked="checked" />
14     <label for="f1_age_majeur">18 ans ou plus</label><br />
15   </p>
16 </form>
```

- Liste déroulante ($\Leftrightarrow$ boutons radio)
  - 1 sélection parmi un ensemble de propositions
  - Balises de la liste : <select> et </select>
    - attribut <name> pour le serveur
    - sans oublier l'association avec un label
  - Balises pour une entrée : <option> et </option>
    - attribut <value> pour préciser l'entrée sélectionnée (pour le serveur)
    - texte entre les balises : texte affiché dans la page
    - attribut selected="selected" pour un choix par défaut
  - Balises <optgroup> et </optgroup> :
    - regrouper les entrées par groupes, avec titres
    - attribut <label> pour préciser le titre
  - Liste avec ascenseur
    - Dans la balise <select>, on ajoute l'attribut size (entier)

    $\rightarrow$ pour listes longues (ou courtes)

    $\rightarrow$ pour un affichage en permanence de plusieurs entrées
  - Liste avec ascenseur à choix multiples ($\Leftrightarrow$ cases à cocher)
    - Dans la balise <select>, on ajoute l'attribut multiple="multiple"

```
<form id="f1" method="post" action="traitement.php">
  <p>
    <label for="f1_choix">Votre choix</label><br />
    <select name="choix" id="f1_choix">
      <optgroup label="Sciences">
        <option value="maths"> Mathématiques </option>
        <option value="info" selected="selected"> Informatique </option>
        <option value="phys"> Physique </option>
      </optgroup>
      <optgroup label="Culture">
        <option value="jeux"> Jeux vidéo </option>
        <option value="lecture"> Lecture </option>
        <option value="cinema"> Cinéma </option>
        <option value="theatre"> Théâtre </option>
      </optgroup>
    </select>
  </p>
  <p>
    <label for="f2_choix">Votre choix</label><br />
    <select name="choix[]" id="f2_choix" size="3" multiple="multiple">
      <optgroup label="Sciences">
        <option value="maths"> Mathématiques </option>
        <option value="info" selected="selected"> Informatique </option>
        <option value="phys"> Physique </option>
      </optgroup>
      <optgroup label="Culture">
        <option value="jeux" selected="selected"> Jeux vidéo </option>
        <option value="lecture"> Lecture </option>
        <option value="cinema"> Cinéma </option>
        <option value="theatre"> Théâtre </option>
      </optgroup>
    </select>
  </p>
```

- Hiérarchisation d'un formulaire
  - Balises : <fieldset> et </fieldset>
  - But : regrouper plusieurs items de saisie dans des blocs
  - Balises <legend> et </legend> pour donner un titre à un bloc
  - Imbrication des blocs autorisée

```
1   <form id="f1" method="post" action="traitement.php">
2     <fieldset>
3       <legend>cadre 1</legend>
4        <input type="text" />
5        <input type="text" />
6        <input type="text" />
7     </fieldset>
8     <fieldset>
9        <legend>cadre 2</legend>
10       <fieldset>
11          <legend>cadre 2 interne</legend>
12          <input type="text" />
13          <input type="text" />
14          <input type="text" />
15       </fieldset>
16       <input type="text" />
17       <input type="text" />
18       <input type="text" />
19    </fieldset>
20  </form>
```

- Champ caché
  - balise : <input type="hidden" value="..."/>
  - attribut value obligatoire (sinon aucun intérêt)
  - non visible dans la page web
    - Attention : l'internaute peut voir le code de la page et les champs cachés
    - L'internaute peut modifier la valeur d'un champ caché avant envoi

```
<input type="hidden" name="id" value="1028" />
```

  - envoyé avec le reste du formulaire
- Mise en forme
  - couleur, image de fond, taille des caractères, ...

  $\Rightarrow$ fichier CSS

- Navigation au clavier
  - touche Tab pour passer à l'item suivant
  - touche Shift+Tab pour passer à l'item précédent
  - Attribut tabindex="[numéro]"
    - par défaut : ordre de déclaration dans le code XHTML
    - sinon Tab passe au numéro suivant
    - les numéros ne sont pas (obligatoirement) consécutifs

```
1   <!-- à ne pas faire -->
2   <form id="f1" method="post" action="traitement.php">
3     <p>
4       <input type="text" id="a" tabindex="20" />
5       <label for="a">: prénom (champ 2)</label><br />
6       <input type="text" id="b" tabindex="10" />
7       <label for="b">: nom (champ 1)</label><br />
8       <input type="text" id="c" tabindex="40" />
9       <label for="c">: âge (champ 4)</label><br />
10      <input type="text" id="d" tabindex="30" />
11      <label for="d">: ville (champ 3)</label>
12    </p>
13  </form>
```

- Boutons
  - Envoi/validation du formulaire
    - syntaxe : <input type="submit" />
  - Réinitialisation du formulaire
    - syntaxe : <input type="reset" />
  - Sans action (pour brancher une fonction JavaScript)
    - syntaxe : <input type="button" />
  - Attribut value : change le texte du bouton

```
1   <!-- <form id="f1" method="post" action="traitement.php"> -->
2   <form id="f1" method="post" action="mailto:toto@toto.fr" enctype="text/plain">
3     <p>
4       <input type="hidden" name="id" value="1028" />
5       <input type="text" name="login" id="f1_login" value="Batman" />
6       <label for="f1_login">: pseudo</label><br />
7       <input type="text" name="nom" id="f1_nom" />
8       <label for="f1_nom">: nom et prénom</label><br />
9       <input type="password" name ="passwd" id="f1_passwd" />
10      <label for="f1_passwd">: mot de passe</label>
11    </p>
12    <p>
13      <input type="submit" value="Envoyer" />
14      <input type="reset" value="Réinitialiser" />
15      <input type="button" value="Rien" />
16    </p>
17  </form>
```

- Envoi d'un fichier
  - Modifier la balise form
    - ajout de l'attribut : enctype="multipart/form-data"
  - Syntaxe : <input type="file" />
    - attributs name, id, tabindex
    - penser au label associé
  - Aspect visuel : champ de saisie avec bouton de navigation
  - Limiter la taille des fichiers envoyés
    - ajout d'un champ caché
    - <input type="hidden" name="MAX_FILE_SIZE" value="taille" />

```
1   <form id="f1" method="post" action="traitement.php" enctype="multipart/form-data">
2     <p>
3       <input type="hidden" name="MAX_FILE_SIZE" value="2000" />
4       <label for="f1_fich">Fichier :</label><br />
5       <input type="file" id="f1_fich" name="fichier" />
6     </p>
7     <p>
8       <input type="submit" value="Envoyer" />
9     </p>
10  </form>
```

Plan → D - CSS

- CSS (Cascading Style Sheet)
  - décrit la mise en forme d'un document XHTML
  - grand nombre de propriétés
  - syntaxe d'une propriété
    - nom_propriété : valeur ;
    - exemple : *font-style : italic*
- Gestion interne du navigateur
  - rencontre d'une balise XHTML
  - correspondance avec la(les) propriété(s)
  - affichage graphique
- Où insérer le code CSS
  - directement dans les balises (bof)
  - en tête du fichier XHTML (mieux)
  - dans un fichier à part (bien)

- Propriétés dans les balises
  - Syntaxe : <balise style=" /* liste de propriétés */ ">
  - Exemple :
```
<h1 style="color: yellow; font-size: 150%;">
```
  - Non recommandé
    - lecture difficile
    - modifications difficiles
    - style alternatif : impossible
    - style différent selon les médias : impossible
- Propriétés hors des balises
  - Comment différencier une balise parmi sa famille (*h1* par exemple) ?
    - utilisation de l'attribut id et /ou class
    - exemple :
```
<h1 id="titre_intro">
```
    - exemple :
```
<h1 class="titre_intro">
```
    - puis attribution d'un style à l'identifiant ou la classe

- Propriétés en tête du fichier XHTML
  - Dans le header
  - Syntaxe :

```
<head>
   <title>Mon titre</title>
   <meta http-equiv="content-type" content="text/html; charset=iso-8859-1"/>
   <style type="text/css">
      /* cf. section sur la syntaxe */
   </style>
</head>
```

  - Exemple :

```
<style type="text/css">
   h1
   {
      color: red;
   }
</style>
```

  - Inconvénients
    - à répéter dans plusieurs fichiers
    - style alternatif : impossible
    - style différent selon les médias : impossible

- Propriétés dans un fichier séparé
  - Déclaration dans le header :

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

  - Exemple :

```
<link rel="stylesheet" media="screen" type="text/css" title="amoi" href="mon_style.css" />
```

    - href : nom ou url du fichier de style
    - media : destination de l'affichage (*all*, *screen*, *print*, ...)
  - Avantages
    - un seul fichier pour plusieurs pages

    $\Rightarrow$ modifications aisées
    - style alternatif possible
    - style différent selon les médias possibles
    - fichier chargé une seule fois par le navigateur
  - Styles alternatifs
    - plusieurs styles au choix pour l'utilisateur
    - accès : menu *"Affichage/Style de la page"* du navigateur
    - syntaxe : alternate stylesheet au lieu de stylesheet

- Un peu plus loin avec les médias
  - Source : *http://www.yoyodesign.org/doc/w3c/css2/media.html*
  - Exemples de médias :
    - all : tout support d'"affichage"
    - aural : synthétiseur de parole
    - braille : pour appareils braille à retour tactile
    - print : pour l'impression
    - screen : pour les écrans
    - et aussi embossed, handheld, projection, tty, tv, ...
  - Catégories de médias
    - continu ou paginé
    - visuel, auditif ou tactile
    - grille de caractères ou bitmap
    - interactif ou statique
    - all pour tout type de médias
- Directive @import
  - inclure une feuille de style à l'intérieur d'une autre feuille
  - syntaxe : @import url("fichier css") media1, meda2, ... ;
  - exemple :

```
@import url("fineprint.css") print, embossed;
```

- Syntaxe d'une feuille CSS
  - une feuille est constituée de
    - une liste de règles-at (@import par exemple)
    - une liste de jeux de règles
  - un jeu de règles est constitué de
    - un sélecteur
    - un bloc de déclaration
  - un bloc de déclaration est constitué d'une liste de déclarations
  - une déclaration est constituée de
    - une propriété
    - une valeur
  - illustration :

```
1  /* règles-at */
2  @import ...;
3  ...
4
5  /* un jeu de règles */
6  selecteur1
7  {
8     propriété1: valeur1;    /* une déclaration */
9     propriété2: valeur2;    /* une autre */
10    ...
11 }
12 ...
```

- Syntaxe d'une feuille CSS
  - Commentaires
    - entre /* et */
    - exemple /* ceci est un commentaire */
  - Exemple récapitulatif

```
1  @import url("css_general.css") all;
2
3  h1
4  {
5     background-color: orange;          /* arrière-plan      */
6     border: 1px solid black;           /* encadrement       */
7     color: yellow;                     /* avant-plan        */
8     font-size: 150%;                   /* taille de la police */
9     padding: 1em;                      /* marge intérieure  */
10 }
```

  - Équivalence d'écriture

```
selecteur1
{
   propriété1: valeur1;
   propriété2: valeur2;
   propriété3: valeur3;
   propriété4: valeur4;
}
```

$\Leftrightarrow$

```
selecteur1 { propriété1: valeur1; }
selecteur1 { propriété2: valeur2; }
selecteur1 { propriété3: valeur3; }
selecteur1 { propriété4: valeur4; }
```

Plan → D - CSS

- Syntaxe générale :

```
nom-balise
{
   propriété1: valeur1;    /* une déclaration */
   propriété2: valeur2;    /* une autre */
   ...
}
```

- Partage de code entre plusieurs balises

```
nom-balise1, nom-balise2, ...
{
   ...
}
```

  - on note la virgule comme séparateur
- Sélecteur * : signifie toutes les balises
- Exemple :

```
1  h1, h2
2  {
3     text-align: center;
4     border: 2px solid black;
5  }
6
7  h1 { color: red; }
8
9  h2 { color: blue; }
```

Plan → D - CSS

- Classes
  - Donner un nom à un ensemble de propriétés
  - Syntaxe CSS :
    - sélecteur : .nom-classe
    - bloc de déclaration : même syntaxe
  - Exemple :

exemple.css

```
1 .resume
2 {
3    text-align: center;
4    font-style: italic;
5 }
```

exemple.html

```
1     <p class="resume">
2        Lorem ipsum dolor sit amet,
3        consectetuer adipiscing elit. Aliquam
4        lacinia leo ac nunc. Nulla aliquet.
5     </p>
```

  - Intérêts
    - même style pour des balises différentes
    - appliquer un style hors balise classique (i.e. *span* et *div*)
    - différencier une balise particulière de la règle générale
  - Appliquer plusieurs classes
    - syntaxe : `<p class="classe1 classe2 ...">` ...

- Identifiants
  - Toute balise peut comporter l'attribut id
    - exemple : `<h1 id="un-id">`
    - attention : un id est unique dans une page XHTML
    - on associe des propriétés CSS à un id
  - Syntaxe CSS :
    - sélecteur : #nom-id
    - bloc de déclaration : même syntaxe
  - Exemple :

exemple.css

```
1 #conclusion
2 {
3    text-align: center;
4    font-style: italic;
5 }
```

exemple.html

```
1     <p id="conclusion">
2        Lorem ipsum dolor sit amet,
3        consectetuer adipiscing elit. Aliquam
4        lacinia leo ac nunc. Nulla aliquet.
5     </p>
```

  - Intérêts
    - surtout utilisés pour placer les blocs sur la page
    - beaucoup moins pour les effets visuels

Plan → D - CSS

- Propriété héritée par une balise
  1. définie explicitement dans une balise parente
  2. automatiquement appliquée aux les balises filles (imbriquées)
  - exemple : <h1>La sécurité est primordiale</h1>
    - si h1 { color : blue ; }
    - si aucune couleur n'est définie pour em
    - ⇒ le mot en *em*phase est également bleu
  - une telle propriété (ici pour *em*) a la valeur par défaut inherited
- Propriété par défaut pour une balise
  - la propriété par défaut est prioritaire sur l'héritage
  - exemple : <h1>La sécurité est primordiale</h1>
    - le mot "sécurité" est en gras par défaut
    - quelle que soit la propriété font-weight de la balise englobante, le mot reste en gras
    - il faut mettre explicitement, pour la balise strong, la valeur de la propriété font-weight à inherited pour hériter du parent

Plan → D - CSS

- Élément simple
  - déjà vu
  - exemple : h1 {color : blue ; }
  - les propriétés s'appliquent à toutes les balises de ce type
- Sélecteur universel
  - syntaxe : *
  - les propriétés s'appliquent à n'importe quel élément
  - facultatif s'il est couplé avec un autre élément
    - *.nom-classe ⇔ .nom-classe
    - *#nom-id ⇔ #nom-id
- Sélecteur de classe
  - syntaxe : balise.nom-classe
  - exemple : h1.premier {color : blue ; }
  - pour les titres h1 dont l'attribut class contient la valeur premier.
  - équivalent à h1[class~="nom-classe"]
  - attention, ne pas confondre avec balise .nom-classe (i.e séparés par un espace)

- Sélecteur d'id
  - syntaxe : balise#identifiant
  - même principe que le sélecteur de classe
- Sélecteurs d'attribut
  - Version 1
    - syntaxe : balise[nom-attr]
    - les propriétés s'appliquent à la balise qui contient l'attribut nommé, quelle que soit sa valeur
    - exemple :
```
h1[title] {...
```
    - reconnait :
```
<h1 title="blabla">...
```
    - ignore :
```
<h1 class="blabla">...
```

- Sélecteurs d'attribut (suite)
  - Version 2
    - syntaxe : balise[nom-attr="valeur"]
    - les propriétés s'appliquent à la balise qui contient l'attribut nommé, s'il contient exactement la valeur
    - exemple :

```
h1[title="maison"] {...
```

    - reconnait :

```
<h1 title="maison">...
```

    - ignore :

```
<h1 title="blabla">...
<h1 title="maison jardin">...
<h1 title="jardin maison">...
<h1 title="maison-jardin">...
```

- Sélecteurs d'attribut (suite)
  - Version 3
    - syntaxe : balise[nom-attr~="valeur"]
    - les propriétés s'appliquent à la balise qui contient l'attribut nommé s'il contient le mot *valeur*, à une position quelconque, dans une liste de valeurs séparées par des espaces
    - exemple :

```
h1[title~="maison"] {...
```

    - reconnait :

```
<h1 title="maison">...
<h1 title="maison jardin">...
<h1 title="jardin maison">...
```

    - ignore :

```
<h1 title="blabla">...
<h1 title="maison-jardin">...
```

- Sélecteurs d'attribut (suite)
  - Version 4
    - syntaxe : balise[nom-attr|="valeur"]
    - les propriétés s'appliquent à la balise qui contient l'attribut nommé s'il débute par *valeur* dans une liste de valeurs séparées par des tirets
    - exemple :

```
h1[title|="maison"] {...
```

    - reconnait :

```
<h1 title="maison">...
<h1 title="maison-jardin">...
<h1 title="maison-jardin piscine">... /* ??? */
```

    - ignore :

```
<h1 title="blabla">...
<h1 title="jardin-maison">...
<h1 title="maison jardin">...
```

- Sélecteurs descendants
  - syntaxe : élément1 élément2
  - sélectionne l'élément2 s'il est un descendant de élément1

```css
/*
 * pour les balises strong à condition qu'elles soient dans un titre de niveau h1
 */
h1 strong
{
   background-color: rgb(255, 100, 100);
}

/*
 * style pour les balises strong à condition qu'elles soient dans un titre de niveau h1,
 * et à condition que ce dernier inclut la classe "particulier" et qu'en plus
 * il (le titre) soit dans un div d'identifiant "intro"
 */
div#intro h1.particulier strong
{
  background-color: rgb(255, 100, 255);
}
```

```html
<h1> Titre <strong>important</strong> </h1>
<div id="intro">
   <div id="abstract">
      <h1 class="particulier"> Titre <strong>important</strong> </h1>
   </div>
</div>
```

- Sélecteurs d'enfants
  - syntaxe : élément1 > élément2
  - sélectionne l'élément2 s'il est un fils de élément1

```css
/*
 * pour les balises strong à condition qu'elles soient DIRECTEMENT dans
 * un paragraphe qui lui-même est DIRECTEMENT dans un div
 */
div > p > strong
{ background-color: rgb(255, 100, 100); }

/*
 * style pour les balises strong à condition qu'elles soient DIRECTEMENT dans un titre de
 * niveau h1, et à condition que ce dernier inclut la classe "particulier" et qu'en plus
 * il (le titre) soit DIRECTEMENT dans un div d'identifiant "intro"
 */
div#intro > h1.particulier > strong
{ background-color: rgb(255, 100, 255); }
```

```html
<div id="intro">
  <p> Bonjour <strong>le</strong> monde </p>
  <blockquote>
    <p> Là, ça ne marche <strong>plus</strong> </p>
  </blockquote>
  <h1 class="particulier">Titre avec un <strong>un mot</strong> en strong </h1>
</div>
```

- Sélecteur premier fils
  - syntaxe : élément :first-child
  - sélectionne l'élément s'il est le premier fils de son parent

```
1  /*
2   * pour les balises strong à condition qu'elles soient le
3   * premier fils de leurs parents directs, quels qu'ils soient
4   */
5  strong:first-child
6  { background-color: rgb(255, 100, 100); }
7
8  /*
9   * style pour les balises em à condition qu'elles soient dans un div
10  * de classe "particulier" et en plus qu'elles soient le premier fils de
11  * leurs parents directs (qui n'est pas obligatoirement le div)
12  */
13 div.particulier em:first-child
14 { background-color: rgb(100, 255, 100); }
```

```
1      <p> Bonjour <strong>le</strong> monde </p>
2      <div class="particulier">
3         <p> un premier paragraphe </p>
4         <p> et un <em>autre</em> </p>
5      </div>
```

- Sélecteurs adjacents
  - syntaxe : élément1 + élément2
  - sélectionne l'élément2 s'il est immédiatement précédé de élément1

```
1 /*
2  * pour un un paragraphe suivant IMMEDIATEMENT un titre h1
3  */
4 h1+p
5 { background-color: rgb(255, 100, 100); }
```

```
1       <div>
2          <h1>un titre</h1>
3          <p> Bonjour le monde </p>
4       </div>
```

- Pseudo-classes
  - lien non encore visité
    - syntaxe : balise :link
  - lien visité
    - syntaxe : balise :visited
  - élément désigné, non activé (survolé)
    - syntaxe : élément :hover
  - élément activé
    - syntaxe : élément :active
  - élément ayant le focus
    - syntaxe : élément :focus

```
1  a:link
2  { color: red; }
3  a:visited
4  { color: blue; }
5  a:hover
6  { color: green; }
7
8  strong:active
9  { text-decoration:blink; }
10
11 input:focus
12 { background-color: rgb(255,255,220); }
```

exemple.html

```
1      <p>
2         <a href="http://w3.org">W3C</a>
3         gère les <strong>standards</strong><br />
4         <input type="text" value="votre choix" />
5      </p>
6
7   </body>
8  </html>
```

- Pseudo-éléments
  - première ligne d'un paragraphe
    - syntaxe : élément :first-line
  - première lettre d'un élément
    - syntaxe : élément :first-letter
  - insertion de texte avant et après un élément
    - syntaxe : élément :before { content : "..." ; }
    - syntaxe : élément :after { content : "..." ; }

```
1  p.ligne:first-line
2  { text-transform: uppercase; }
3
4  p.lettre:first-letter
5  {
6     font-size: 400%;
7     color: green;
8     font-style: italic;
9     margin: 3px;
10    float: left;
11 }
12
13 strong:before
14 { content: "==>"; }
15
16 strong:after
17 { content: "<=="; }
```

exemple.html

```
1       <p class="ligne">
2          Lorem ipsum dolor sit amet, ...
3       </p>
4       <p class="lettre">
5          Lorem ipsum dolor sit amet, ...
6       </p>
7       <p>
8          Lorem ipsum <strong>dolor</strong>
9          sit amet, ...
10      </p>
```

Plan → D - CSS

- Propriété *position* sur une boîte
  - valeur *static*
    - s'insère normalement dans le flux
  - valeur *relative*
    - s'insère normalement dans le flux, puis est décalée relativement à cette position
    - cf. propriétés *left, top, right et bottom*
    - les autres éléments sont placés sans tenir compte du déplacement
  - propriétés *left, top, right et bottom*
    - décalage par rapport à un bord
    - longueur ou pourcentage

```
1 /* déplacement vers le haut et la droite */
2 em
3 {
4    position: relative;
5    bottom: 10px;  /* longueur fixe */
6    left: 5%;      /* proportionnel au bloc englobant */
7 }
```

- Propriété *position* sur une boîte
  - valeur *absolute*
    - ne fait plus partie du flux
    - positionné par rapport au bloc conteneur
    - cf. propriétés left, top, right et bottom
  - valeur *fixed*
    - ne fait plus partie du flux
    - comme absolute
    - mais de défile pas (média screen)

```
1  img.logo
2  {
3     position: absolute;
4     top: 5px;
5     right: 5%;
6  }
7
8  img.titre
9  {
10    position: fixed;
11    top: 0px;
12    left: 0%;
13 }
```

- Propriété *float* sur une boîte
  - boîte positionnée à gauche (valeur *left*) ou à droite (valeur *right*)
  - une largeur doit être spécifiée (propriété *width*)
  - le texte (d'une autre boîte) s'écoule le long du flanc gauche (valeur *right*) ou droit (valeur *left*)
  - propriété *clear*
    - stoppe la propriété d'écoulement

```
1  #menu
2  {
3     margin-top: 30px;
4     float: left;
5     width: 20%;
6  }
7
8  #corps
9  {
10    margin-left: 22%;
11 }
12
13 #piedpage
14 {
15    clear: both;
16 }
```

Plan → D - CSS

- Grand nombre de propriétés
- Mécanique d'application des styles
  - complète
  - complexe
  - très précise
- À étudier
  - valeurs spécifiées vs. valeurs calculées vs. valeurs réelles
  - attribut !important
  - provenance des feuilles de style
    - auteur (≈ concepteur du site)
    - utilisateur (≈ internaute)
    - agent utilisateur (≈ navigateur)
- Les valeurs
  - entiers
  - longueurs : unités relatives ou absolues, pourcentage
  - ...
- ...

Plan → E - JavaScript

- Exécution sur le poste du client
  - peut être désactivé
  - un site devrait fonctionner même si JavaScript est désactivé
- Exécution au moment du chargement de la page
- Réaction à des événements (utilisateurs ou non)
  - chargement/déchargement d'une page,
  - validation d'un formulaire,
  - clic sur un élément de la page,
  - passage de la souris sur un élément,
  - ...
- Rappel : CSS réagit aussi à quelques événements
- Formulaires
  - vérification des informations entrées
    - ⇒ blocage de l'envoi vers le serveur
    - ⇒ limite le traffic réseau
  - désactivation possible de JavaScript
    - ⇒ vérification supplémentaire par le serveur

- Place de JavaScript dans une page Web
  - dans les attributs de certaines balises
  - au milieu du code XHTML,
  - dans un fichier à part (inclus au milieu du HTML)
- Code directement dans une page
  - entre les balises <script type="text/javascript"> et </script>
  - problème du validateur avec <, > et &
    - ⇒ balisage complété :

```
<script type="text/javascript">
  //<![CDATA[
  /* le code JS ici */
  //]]>
</script>
```

- Code dans un fichier externe

```
<script src="source.js" type="application/javascript"></script>
```

  - ⇒ code XHTML plus lisible
  - ⇒ plus de CDATA à gérer
  - ⇒ à privilégier

Plan → E - JavaScript

- Langage
  - ce n'est pas Java
  - il est orienté objets
- Variables simples
  - non typées
    - i.e. elles peuvent changer de type au cours de l'exécution
    - à éviter : on privilégiera un type constant
  - déclaration
    - non obligatoire
    - portée globale par défaut : à utiliser avec parcimonie
    - portée locale : déclaration avec le mot-clef var
  - opérateur +
    - addition
    - concaténation de chaînes de caractères (si un des deux opérandes est une chaîne)

    $\Rightarrow$ attention danger !

- Variables simples (suite)
  - exemples illustratifs

```
1   <script type="text/javascript">
2     //<![CDATA[
3     var a;          // déclaration explicite, type "undefined", portée locale
4     var b = 3;      // déclaration explicite, type number, portée locale
5     var d = true;   // déclaration explicite, type booléen, portée locale
6     c = "123";      // déclaration implicite, type string, portée globalse
7     a = 17;         // a devient un number
8     b = c + a;      // concaténation : a est casté en chaine, b devient une string
9     b = c / a;      // c transformé en number, b redevient un number
10    //]]>
11  </script>
```

  - conversions implicites
    - attention !
- Commentaires
  - une ligne : débute par // jusqu'à la fin de la ligne
  - multi-lignes : débute par /* jusqu'à */
  - ils sont indispensables
  - mettre de bons commentaires est difficile

- Opérateurs
  - arithmétiques

++ --	post (et pré) incrémentation/décrémentation
-	moins unaire
* / %	multiplication, division, modulo
+ -	addition, soustraction

```
a = 3 * 4;      // a vaut 12
a = 3*4 + 6/2;  // a vaut 15
a++ ;           // a vaut 16
```

  - affectations

=	affectation
*= /= %=	opération et affectation
+= -=	opération et affectation

```
a = 15;       // affection classique
a *= 3;       // a vaut 45, équivalent à a = a * 3
a = 15;
a = a*3 + 5;  // a vaut 50
a = 15;
a *= 3 + 5;   // a vaut 120, équivalent à a = a * (3+5)
```

- Opérateurs
  - comparaisons

!= ==	(non) égalité de valeurs
!== ===	(non) égalité de valeurs et de types
< <= > >=	plus petit/grand (ou égal)

```
a = (15 == 15);     // true
a = (15 === 15);    // true
a = (15 == "15");   // true
a = (15 === "15");  // false (types différents)
```

  - logiques

!	*NOT* (négation)
&& ||	AND (ET), OR (OU)

```
a = !(15 > 15);                          // true
a = (14 == 14) && (2 < 3) || (2 > 3); // true;
```

  - autre

typeof	donne le type d'une variable

```
a = typeof 15;        // number
a = typeof("15");     // string
```

- Tests
  - if then else
    - syntaxe :
      if (*expr booléenne*)
      bloc
      else if (*expr booléenne*)
      bloc
      ...
      else
      bloc
      un bloc de plusieurs instructions doit être entre accolades
    - exemple

```
if (a < 0)
{ // accolades obligatoires
   a = -a;
   document.write("a était négatif");
}
else if (a > 0)
   document.write("a est positif");
else
{ // accolades facultatives
   document.write("a est nul");
}
```

- Tests
  - switch case
    - syntaxe :

```
switch (expr)
{
    case val1 :
        instructions
        break ;
    case val2 :
        instructions
        break ;
    ...
    default :
        instructions
        break ;
}
```

la clause *default* est facultative

exemple :

```
switch (a)
{
   case 2:
      // instructions pour a == 2
      a = 2*a;
      break;
   case 3:
      // instructions pour a == 3
      a = 3*a - 2;
      break;
   default:
      // instructions si a différent de 2 et 3
      a ++;
      break; // inutile
}
```

- Boucles
  - while et do while
    - syntaxe :

while (*expr booléenne*)
  bloc
	do
	  bloc
	while (*expr booléenne*)

un bloc de plusieurs instructions doit être entre accolades

- break : interrompt et sort de la boucle
- continue : passe immédiatement à l'itération suivante
- exemple

```
1   while (a < 1024)
2      a = 2*a;
3
4   do
5   {
6      a = 2*a;
7   } while (a < 2048);
8
9   var i = 1;
10  while (a < 1024)
11  {
12     a += i;
13     i ++;
14     document.write(i + " " + a + "<br />");
15  }
```

- Boucles
  - for
    - syntaxe :
      for (var *indice* = *val* ; test terminaison ; opération sur indice)
      bloc
      un bloc de plusieurs instructions doit être entre accolades
    - break : interrompt et sort de la boucle
    - continue : passe immédiatement à l'itération suivante
    - exemple

```
1        a = 7;
2        for (i = 0; i < 10; i++)     // de 0 à 9
3        {
4           a += i;
5           document.write(i + " " + a + "<br />");
6        }
7        for (i = 0; i < 30; i += 5)  // de 0 à 25 par pas de 5
8        {
9           a += i;
10          document.write(i + " " + a + "<br />");
11       }
12       for (i = 9; i >= 0; i--)     // de 9 à 0
13       {
14          a += i;
15          document.write(i + " " + a + "<br />");
16       }
```

- Fonctions
  - syntaxe :
    function *nom-fonction*(*liste paramètres*)
    {
        instructions
        return *expression* ;
    }
    - une fonction sans *return* est une procédure
    - les paramètres ne sont pas typés
  - exemple

```
1   function afficheBonjour(n, prefix)
2   {
3      for (var i = 0; i < n; i++)
4         document.write(prefix + "Bonjour !<br />");
5   }
6   afficheBonjour(7, "== ");
7
8   function oppose(n)
9   {
10     return -n;
11  }
12  document.write(oppose(15) + "<br />");
```

Plan → E - JavaScript

- Généralités
  - un tableau est un objet composé : il contient plusieurs valeurs
  - la numérotation des cases commence à partir de 0 (zéro)
    - tableau de 5 cases ⇒ numérotées de 0 à 4
  - on accède à une case avec l'opérateur [ ]
    - tableau *tab* ⇒ *tab[0]*, *tab[1]*, *tab[2]*, ..., *tab[n-1]*
  - toutes les cases n'ont pas (forcément) le même type
    - tableau *tab* de 3 cases
      ⇒ *tab[0] = 127 ; tab[1] = "bonjour" ; tab[2] = false ;*
  - l'attribut *length* permet d'obtenir la taille d'un tableau
    - tableau *tab* ⇒ *tab.length*
  - la taille d'un tableau est dynamique
    - on peut ajouter ou supprimer des cases

- Création
  - `var tab1 = new Array();`
    - tableau vide (taille 0)
  - `var tab1 = new Array("chat", 3.14, true, 288);`
    - tableau de 4 cases pré-remplies
  - `var tab1 = ["chat", 3.14, true, 288];`
    - même chose
  - `var tab1 = new Array("chat");`
    - tableau de 1 case pré-remplie
  - `var tab1 = [10];`
    - tableau de 1 case pré-remplie
  - `var tab1 = new Array(10);`
    - tableau de 10 cases (indéfinies)
    - et non pas tableau de 1 case qui contiendrait 10

- Parcours

```
function aff(t, titre)
{
   document.write("<strong>" + titre + "</strong>"+ "<br />");
   document.write("longueur : " + t.length + "<br />");
   for (var i = 0; i < t.length; i++)
      document.write(i + " : " + t[i] + "<br />");
   document.write("<br />");
}
```

- Modification d'une case
  - affectation classique : *tab[3] = qqch ;*
- Ajout d'une case
  - comme pour l'affectation, mais avec un numéro inexistant
    - attention : pas de trous dans le tableau
    - cases intermédiaires initialisées à *undefined*

```
var tab = ["chat", 12, true]; // 3 cases
tab[6] = 3.14;                // 7 cases !!!!!!
```

- Tableaux multidimensionnels
  - une case d'un tableau peut être un tableau
  - boucles imbiquées pour un parcours

```
var tab = [[11, 12, 13],
           [21, 22],      // pas obligatoirement la même taille !
           [31, 32, 33]]
```

- Méthodes sur les tableaux
  - syntaxe : tab.methode(paramètres) ;
  - join : transforme le tableau en chaîne de caractères
  - concat : ajoute des éléments en fin
  - reverse : inverse l'ordre des cases
  - sort : tri un tableau
  - slice : extrait un sous-tableau
  - shift/unshit : supprime/ajoute une case en tête
  - pop/push : supprime/ajoute une case en fin

Plan → E - JavaScript

- Généralités
  - même principe que les tableau à indices numériques, sauf que :
    - les indices sont des chaînes (entre guillemets) ou des nombres
    - le parcours avec un for classique ne marche plus
    - accès par notation "crochets" : tab["capitale"]
    - accès par notation pointée : tab.capitale
- Parcours
  - syntaxe : for (var *indice* in *tableau*) { ... }
    où l'"indice" permet de désigner les cases du tableau
- Exemple

```
var tab = new Array;
tab["France"] = 33;
tab[18] = true;
tab["chat"] = "miaou";
for (var key in tab)
{
   document.write("indice : " + key + "<br />");
   document.write("valeur = " + tab[key] + "<br />");
```

Plan → E - JavaScript

- Côté serveur
  - JavaScript tourne/tournait essentiellement sur le poste client. C'est aussi une technologie côté serveur avec NodeJS.
- Bibliothèques, frameworks
  - JavaScript est enrichi de nombreuses extensions qui facilitent la vie des programmeurs. On peut citer JQuery, Angular JS , ...
- AJAX
  - AJAX est du JavaScript qui permet de faire une requête vers le serveur afin de mettre à jour uniquement une partie de la page web courante.
  - L'utilisation d'une bibliothèque telle JQuery facilite grandement la programmation AJAX.

Plan → F - PHP

- *http://php.net*
- Exécution sur le serveur
  - demande du client
  - génération dynamique de la page *HTML*
  - envoi au client
  - (le client ne voit pas le code PHP)
- PHP permet de :
  - générer dynamiquement du code HTML
  - interagir avec une base de données
  - authentifier un client
  - gérer la réception d'un formulaire
  - ...
- Code directement dans la page HTML
  - balise <?php et ?>
- Code dans un fichier externe (exemple *source.php*)
  - <?php include("source.php"); ?>
- Note : le serveur doit être capable d'interpréter le PHP

Plan → F - PHP

- PHP est orienté objets
- Variables
  - déclarations inutiles (elles sont implicites)
  - le nom est précédé du signe $
  - initialement une variable est vide (valeur NULL)
  - fonction isset pour tester l'existence d'une variable

```
<?php
   // code incorrect même s'il ne plante pas
   echo 'la valeur de $v (non créée) est --&gt;' . $v . '&lt;--<br />';
   if (isset($v))
      echo '$v existe'. '<br />';
   else
      echo '$v n\'existe pas'. '<br />';

   // déclaration implicite avec type défini automatiquement
   $v = 123;
   echo 'la valeur de $v (créée) est --&gt;' . $v . '&lt;--<br />';
   if (isset($v))
      echo '$v existe'. '<br />';
   else
      echo '$v n\'existe pas'. '<br />';
?>
```

- Structures de contrôle comme en C
  - if ... elseif ... else, switch, while, for

```
<?php
   $v = 123;
   if ($v < 0)
      echo $v . ' est négatif<br />';
   elseif ($v == 0)
      echo $v . ' est nul<br />';
   else
      echo $v . ' est positif<br />';
?>
```

  - opérateurs booléens && et || (ne pas utiliser *AND* et *OR*)
- Tableaux classiques
  - ajout (suppression) dynamique de cases

```
$villes = array("Poitiers", "Paris", "Tours");
echo 'Chez nous : ' . $villes[0] . '<br />';
$villes[1] = "Lutèce";
$villes[] = "Beyrouth";
$villes[] = "Canberra";
for ($i = 0; $i < count($villes); $i ++)
   echo $i . ' = ' . $villes[$i] . '<br />';
```

- Tableaux associatifs
  - indicés par des nombres ou des chaînes de caractères : clé/valeur
  - quelques fonctions :
    count : nombre d'éléments
    foreach : parcours d'un tableau
    array_key_exist : test d'existence d'une clé
    in_array : test d'existence d'une valeur
    array_search : clé associée à une valeur
    explode : transformer une chaîne en tableau
    implode : tranformer un tableau en chaîne

```php
$villes = array(
   "chez nous" => "Poitiers",
   "capitale"  => "Paris",
   "pas loin" =>"Tours");
echo 'Chez nous : ' . $villes["chez nous"] . '<br />';
$villes["capitale"] = "Lutèce";
$villes["liban"] = "Beyrouth";
foreach ($villes as $cle => $valeur)
   echo $cle . ' = ' . $villes[$cle] . '<br />';
```

- Chaînes de caractères
  - très utilisées en PHP ⇒ "pléthore" de fonctions
  - concaténation : opérateur point "."
  - quelques fonctions :
    addslashes : backslashe apostrophes, guillemets et backslashes
    stripslashes : fonction inverse
    htmlspecialchars : transforme certains caractères en entités HTML
    htmlentities : transforme tous les caractères en entités HTML
    urlencode, rawurlencode : encode une chaîne en URL
    nl2br : transforme les retour chariot en *
*
    st_replace, ...
  - certaines fonctions utilisent les expressions régulières

```php
$v = '<h1>' . 'la "solution" de l\'élève est ambiguë' . '</h1>';
echo $v . '<br />';
$x = addslashes($v);
echo $x . '<br />';
$x = htmlspecialchars($v);
echo $x . '<br />';
$x = htmlentities($v);
echo $x . '<br />';
$v = 'http://site.com/index.php?titre=' . urlencode('un livre')
   . '&auteur=' . urlencode('noémie');
```

- Fonctions
  - paramètres non typés
  - paramètres optionnels

```
function afficheBonjour($n)
{
   echo 'Je vais afficher ' . $n . ' fois la phrase :<br />';
   $s = 'Bonjour d\'un code <strong>encore simple</strong> écrit en PHP.';
   for ($i = 0; $i < $n; $i++)
      echo $s . " (" . $i . " fois) <br />";
}
afficheBonjour(3);

function decalageCarre($n, $origine = 0) // paramètre optionnel
{
   return $n*$n + $origine;
}
echo decalageCarre(5) . '<br />';
echo decalageCarre(5, 3) . '<br />';

function modif($a, &$b) // passage par référence
{
   $a ++; // inutile
   $b ++; // oui
}
$v = 1;
$x = 1;
modif($v, $x);
echo '$v vaut ' . $v . ' et $x vaut ' . $x . '<br />';
```

- Variables de variable
  - une variable contient le nom d'une autre variable

```php
$ma_maison = 'poitiers';
$ma_voiture = 'tricycle';
$ma_profession = 'prof';
$quoi = 'ma_voiture';
echo ${$quoi};
$quoi = 'voiture';
echo ${'ma_' . $quoi};
$quoi = 'ma_voiture';
$quoiquoi = 'quoi';
echo ${${$quoiquoi}};   // beurk celui-là
```

  - attention à la lisibilité
  - utiliser plutôt les tableaux associatifs
- PHP langage objets
  - vrai langage objet
  - membres, méthodes, héritage, ...
  - à étudier

Plan → F - PHP

- Envoi direct (dans le lien)
  - succession de couples variable/valeur directement dans l'adresse
  - syntaxe :
    `<a href="toto.php?nom1=val1&nom2=val2&...&nomn=valn" ...`
  - Récupérés, dans la page cible, dans le tableau associatif : $_GET
    $_GET["nom1"] ... $_GET["nomn"]

```
Lien initial : <a href="envoi_get.php?id=10&titre=Un%20test%20avec%20GET">
lien avec 2 pararamètres</a> <br />
Contenu de $_GET <br />
<?php
   if ((! isset($_GET["id"])) && (! isset($_GET["titre"])))
      echo 'aucune des entrées <q>id</q> et <q>titre</q> n\'est présente';
   else
   {
      if (isset($_GET["id"]))
      {
         echo '<q>id</q> (non protégé) vaut <q>' . $_GET["id"] . '</q><br />';
         echo '<q>id</q> (protégé) vaut <q>' . htmlentities($_GET["id"]) . '</q><br />';
      }
      if (isset($_GET["titre"]))
      {
         echo '<q>titre</q> (non protégé) vaut <q>' . $_GET["titre"] . '</q><br />';
         echo '<q>titre</q> (protégé) vaut <q>' . htmlentities($_GET["titre"]) . '</q><br />';
      }
   }
?>
```

- Envoi d'un formulaire
  - méthode *get* ⇒ données récupérées dans $_GET
  - méthode *post* ⇒ données récupérées dans $_POST
  - noms des entrées : attributs name du formulaire

```
<form method="post" action="envoi_form.php">
   <p>
      <input type="text" value="10" name="id" /> : id<br />
      <input type="text" value="tartempion" name="nom" /> : nom<br />
      <input type="submit" value="go" />
   </p>
</form>
<p>
   <?php
      if ((! isset($_POST["id"])) && (! isset($_POST["nom"])))
         echo 'aucune des entrées <q>id</q> et <q>titre</q> n\'est présente';
      else
      {
         if (isset($_POST["id"]))
         {
            echo '<q>id</q> (non protégé) vaut <q>' . $_POST["id"] . '</q><br />';
            echo '<q>id</q> (protégé) vaut <q>' . htmlentities($_POST["id"]) . '</q><br />';
         }
         if (isset($_POST["nom"]))
         {
            echo '<q>nom</q> (non protégé) vaut <q>' . $_POST["nom"] . '</q><br />';
            echo '<q>nom</q> (protégé) vaut <q>' . htmlentities($_POST["nom"]) . '</q><br />';
         }
      }
```

- Envoi de fichiers (formulaire suite)
  - données récupérées dans $_FILES
  - une entrée par fichier transmis qui est un tableau à 5 cases :
    - name : nom du fichier chez le client
    - type : type (Mime) du contenu du fichier
    - tmp_name : nom du fichier sur le serveur
    - error : code d'erreur
    - size : taille du fichiers reçu
  - fonctions utiles :
    - is_uploaded_file : vérifie la provenance d'un fichier
    - move_uploaded_file : déplacement d'un fichier téléchargé
    - getimagesize : quelques informations sur une image
    - ...

```
<form method="post" action="envoi_form.php" enctype="multipart/form-data">
   <p>
      <input type="file" name="avatar" /> : format réduit<br />
      <input type="file" name="portrait" /> : plus grand<br />
      <input type="submit" value="go" />
   </p>
</form>
<?php
   echo '<pre>' . print_r($_FILES, true) . '</pre>';
?>
```

Plan → F - PHP

- Variables super-globales
  - quelques exemples
    - $_SERVER['REMOTE_ADDR'] : adresse IP du client
    - $_SERVER['PHP_SELF] : *URL* de la page courante
    - $_ENV : variables d'environnement du système
    - $_GET, $_POST et $_FILES
    - $_SESSION : variables de session (cf. ci-dessous)
    - $_COOKIE : cookies de l'utilisateurs
    - ...
  - les plus utilisées
    - récupération d'informations (cf. pages précédentes)
    - gestion des cookies (cf. pages suivantes)
    - gestion des sessions (cf. pages suivantes)

```php
echo '<pre> $_SESSION = ' . print_r($_SESSION, true) . '</pre>';
echo '<pre> $_ENV = ' . print_r($_ENV, true) . '</pre>';
echo '<pre> $_COOKIE = ' . print_r($_COOKIE, true) . '</pre>';
echo '<pre> $_SERVER = ' . print_r($_SERVER, true) . '</pre>';
phpinfo();
```

- Sessions
  - but : variables transmises d'une page à l'autre (stockées sur le serveur)
  - fonction session_start
    - à appeler sur chaque page
    - crée/restaure les variables de session (tableau $_SESSION)
    - doit ABSOLUMENT être appelée en tout premier

```
<?php // même pas un espace avant l'ouverture de php
   session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

  - fonction session_destroy
    - détruit définitivement une session

```
if (! isset($_SESSION['compteur']))
{
   echo '<code>compteur</code> n\'existe pas, mise à 1<br />';
   $_SESSION['compteur'] = 1;
}
else
{
   echo '<code>compteur</code> existe, incrémentation<br />';
   $_SESSION['compteur'] ++;
}
echo 'valeur de <code>compteur</code> :' . $_SESSION['compteur'];
```

- Sessions (suite)
  - utilisation classique : ne pas demander un mot de passe à l'utilisateur sur chaque page

```php
if (  (! isset($_POST["login"]))      // première connexion
    && (! isset($_SESSION['user'])))
{
   echo '<form method="post" action="sessions_auth.php">';
   echo '  <p>';
   echo '     <input type="text" name="login"/> : votre login<br />';
   echo '     <input type="submit" />';
   echo '  </p>';
   echo '</form>';
}
else
{
   if (isset($_POST["login"]))  // réception du formulaire
   {
      $_SESSION['user'] = htmlentities($_POST["login"]);
      $_SESSION['cpt'] = 1;
      echo '<h1>' . $_SESSION['user'] . ', bienvenue </h1>';
   }
   else   // suite de la visite
   {
      $_SESSION['cpt'] ++;
      echo '<h1>' . $_SESSION['user'] . ', merci de rester avec nous </h1>';
   }
   echo '<p> passage numéro : ' . $_SESSION['cpt'] . ' </p>';
   echo '<p> continuez donc à <a href="sessions_auth.php">naviguer sur notre site </p>';
}
```

- Cookies
  - petit fichier, à durée "limitée", déposé sur la machine cliente
    ⇒ informations persistantes (au-delà de la session)
  - fonction setcookie
    - création/modification/suppression d'un cookie
    - doit ABSOLUMENT être appelée en tout premier (cf. sessions).
  - remarques
    - possibilité de mettre des tableaux dans des cookies
    - cookies accessibles via la variable $_COOKIE
    - après setcookie, recharger la page pour y accéder via $_COOKIE

```
1  <?php // même pas un espace avant l'ouverture de php
2     if (isset($_COOKIE['cpt']))
3        $valeur = $_COOKIE['cpt'] + 1;
4     else
5        $valeur = 1;
6     setcookie('cpt', $valeur, time() + 60); // expiration = 60 secondes
7  ?>
8  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
1          echo '<p> On a mis la valeur ' . $valeur . ' dans $_COOKIE[\'cpt\'] </p>';
2          if (isset($_COOKIE['cpt']))
3             echo '<p> $_COOKIE[\'cpt\'] vaut ' . $_COOKIE['cpt'] . '</p>';
4          else
5             echo '<p> $_COOKIE[\'cpt\'] n\'est pas positionné </p>';
```

Plan → F - PHP

- Introduction
  - grand nombre de bibliothèques (internes ou non à PHP)
  - illustration avec MySQL
    ⇒ fonctionnement similaire avec d'autres SGBD (PostgreSQL, ...)
  - site d'administration : PhpMyAdmin (PhpPgAdmin)
- Connexion/déconnexion
  - connexion : new PDO(connexion_info, login, password)

```php
try
{
    $login = 'cours_web';
    $passwd = 'toto';
    $bdd = new PDO('mysql:host=localhost;dbname=banque;charset=utf8', $login, $passwd);
}
catch (PDOException $e)
{
    echo "Erreur : " . $e->getMessage() . "\n";
    exit();
}
```

  - déconnexion :

```php
$bdd = null;
```

- Requête
  - interrogation : query(requête)
    - retour false en cas d'erreur
    - la réponse (PDOStatement) sinon

```php
$sql = 'SELECT * FROM mouvements';
$reponse = $bdd->query($sql);
if (! $reponse)
   exit("Erreur query !");
```

  - exploitation : fetch(...)
    - récupère une ligne du résultat
    - appels en boucle

```php
while ($ligne = $reponse->fetch(PDO::FETCH_ASSOC))
    print_r($ligne);
$reponse->closeCursor();
```

  - exploitation : fetchAll(...)

```php
$lignes = $reponse->fetchAll(PDO::FETCH_ASSOC);
print_r($lignes);
$reponse->closeCursor();
```

  - cf. manuel pour d'autres fonctions (rowCount, columnCount, ...)

Plan → F - PHP

- Données sensibles
  - ne pas stoker les mots de passe en clair, mais en crypté (md5)
  - ne pas stoker les numéros de cartes bancaires

  $\Rightarrow$ dégâts limités en cas de piratage de la base
- Filtrer les données utilisateur
  - une donnée extérieure (par un <input type="text" /> par exemple), peut contenir du code malveillant
    - du code HTML
    - du code JavaScript
    - du SQL (injection SQL)
  - il faut systématiquement filtrer et analyser
    - htmlentities, htmlspecialchars
    - addslashes
    - ...
- Partie importante (et pénible) de la programmation web
  - prise en charge par les frameworks tels CakePHP ou Symfony.

Plan → F - PHP

- Définition d'un framework
  - ensemble de composants (gestion de formulaires, authentification, ...)
  - une architecture logicielle pour structurer/organiser le code
  - ce N'est PAS un CMS, le programmeur a encore du travail à fournir
- Avantages
  - gain de productivité
  - incitation à produire un code propre
  - l'architecture (MVC dans notre cas) favorise le travail à plusieurs développeurs
  - communauté active
  - mise à jour automatisées
- Inconvénients
  - temps d'apprentissage conséquent
  - être au fait de certaines technologies récentes
  - plus gourmand en ressources serveur
- Symfony
  - objet de ce cours
  - cf. TP