

Langages de l'internet

Introduction aux principales technologies existantes et proposition d'une méthodologie de génie logiciel

Bruno Mascret

CPE Lyon
Techniques et langages de l'Internet

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Introduction

Présentation du module

Objectif du module :

- ≡ découvrir les principaux langages du web ;
- ≡ savoir les utiliser et les mettre en œuvre ;
- ≡ acquérir une méthodologie de développement web.

Peu de cours, beaucoup de pratique...

Introduction

Présentation du module

Différents types de TPs vous seront proposés :

1. TPs de découverte d'application du cours : destinés à vous familiariser avec les bases des langages et à vous apprendre les bonnes pratiques liées à leur utilisation ;
2. TPs de mise en pratique concrète : ces TPs « fil rouge » seront directement liés à un problème concret pour lequel il vous sera demandé de proposer des solutions en utilisant les compétences et connaissances acquises en cours et durant la réalisation des TPs de découverte et d'application du cours ;

L'enchaînement des TPs se fera directement en lien avec le cours.

Introduction

Présentation du module

- séance 1 : cours 1 (2h) : Web statique, accessibilité, méthodologie de conception
- séance 2 : Web statique : Accessibilité, HTML et CSS (TP 1) (2h)
- séance 3 : Fil rouge : Première ébauche statique du site (TP 2) (4h)
- séance 4 : cours 2 (2h) : Web dynamique, PHP
- séance 5 : cours 3 (2h) : Web dynamique avancé, objet, formulaires
- séance 6 : Web dynamique avec PHP (TP 3) (2h) + Fil rouge : Dynamisation de site avec PHP (TP 4) (2h)

Introduction

Présentation du module

- séance 7 : cours 3bis1 : bases de données et SQL 1
- séance 8 : TP 4 bis (4h)
- séance 9 : cours 3bis : PHP, BD, sécurité (2h)
- séance 10 : PHP avancé : gestion de formulaires, variables web globales, objet (TP 5) (30min) + Fil rouge : Formulaires HTML et PHP Objet (TP 6) (1h30) + SQL, PHP, PDO, et injections SQL (TP 7) (1h) + Fil rouge : Bases de données et PHP (TP 8) (1h)
- séance 11 : cours 4 (2h)
- séance 12 : XML par l'exemple (TP 9) (2h)
- séance 13 : Fil rouge : XML (TP 10) (2h), fin des travaux + Évaluation (2h)

Introduction

Ressources

Pour ce module :

- Un support de cours (celui là)
- Un livret interactif des TPs détaillé
- Un guide détaillé
- Une webographie

Vos connaissances web/langages, expériences ?

Me contacter : bruno.mascret@cpe.fr

Introduction

Auteurs

Cours : **Bruno Mascret, Oscar Figueiredo, David Odin, Geoffroy Charollais**

TPs : **Bruno Mascret**

Intervenants : **Bruno Mascret, Françoise Perrin, Mohamed Sallami, Charles Perrin, Vivien Guillet, Timothé Bordiga, Rémi Marengo, Jonas Pauthier**

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Architecture web

Rappels client serveur

- QUI fait quoi ?
- à quel endroit ?

Structure générale du World Wide Web

- Structure de base : client-serveur classique.



Client



Serveur
HTTP + HTML

- Utilisation fréquente de serveurs proxy ou reverse-proxy
- Extension en architecture n-tiers



Client



Serveur
HTTP



Serveur
d'applications



Base de
données

URL : Uniform Resource Locator

Protocole Hôte Port Chemin Ressource Ancre variables GET

http :// www.cpe.fr 80 :/contact/ adresses.html #direction bla=1&foo=2?

Protocole	http, ftp, news, file, etc. suivi de ://
Hôte	DNS ou IP
Port	Facultatif, 80 par défaut pour http
Chemin d'accès	Séquence d'accès séparée par des / et terminée par un /
Ressource	Document html ou script
Ancre	Voir et <XX id="...">
Variables GET	commencent par un ?, puis des couples variable=valeur séparés par des &

HTTP : HyperText Transfert Protocol

☰ Protocole simplifié

- Transactions en 4 phases
 - Connexion
 - Requête (GET, POST, PUT, ...)
 - Réponse
 - Fermeture
- Sans état
 - Ne maintient aucune information entre 2 requêtes (différent de FTP)
- Sans authentification obligatoire
 - Pas de réponse de login à la connexion
- Optimisé pour la gestion de requêtes simples et nombreuses

☰ HTTP/1.1

- Connexions persistantes
- Négociations de types de documents, compression de fichiers
- Meilleure gestion des caches et des proxys.

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

HTML : HyperText Markup Language

HTML

≡ SGML : Standard Generalized Markup Language

- Langage de description de **structure de document**
- Description d'après une DTD (*Document Type Definition*)

```
<!DOCTYPE report PUBLIC "-//CPE//DTD Report//EN">
<report>
  <author>John Doe</author>
  <date>Feb 1st, 2007</date>
  <title>Report on WWW</title>
  <h>Introduction
  <p>In this report, we show that...
```

≡ Pourquoi SGML comme base pour HTML ?

≡ A l'origine, outil pour chercheurs (publications)

≡ Une normalisation chaotique mais rapide

- Une version normalisées : HTML 2.0
- Des tentatives d'extension avortées : HTML+, HTML 3.0
- Recommandations du W3C : HTML3.2, 4.0, 4.01, XHTML 1.0, 1.1, HTML5...

XHTML

- Ré-écriture de HTML en XML => beaucoup plus de rigueur
 - Le document doit être bien formé (cf. Règles XML correspondantes)
 - Toutes les balises de fermeture sont obligatoires
 - Tous les attributs doivent être valués et entre guillemets
 - Sensibilités à la case : balises et attributs HTML en **minuscules**

- Prologue

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- Éléments vides

- Utilisation de la notation XML pour les éléments vides
- Un espace permet de rester "compatible" avec les navigateurs plus anciens : `
``<hr />`

- Autres...

- Utilisation de l'attribut `id` plutôt que `name`

Vocabulaire HTML

- ≡ Document HTML = flux text + balisage (markup)
- ≡ **Entité** : unité textuelle atomique SGML
 - caractères simples
 - caractères spéciaux : & ; < ; &ouelig; . . .
- ≡ **Étiquette ou balise** : séquence marquant le début ou la fin d'un élément
- ≡ **Élément** : une unité textuelle en tant que composant structurel d'un document
- ≡ **Attribut** : modificateur agissant sur une instance d'un élément.

<p class="warn">

Balise d'ouverture

Vocabulaire HTML

- ≡ Document HTML = flux text + balisage (markup)
- ≡ **Entité** : unité textuelle atomique SGML
 - caractères simples
 - caractères spéciaux : & ; < ; &ouelig; . . .
- ≡ **Étiquette ou balise** : séquence marquant le début ou la fin d'un élément
- ≡ **Élément** : une unité textuelle en tant que composant structurel d'un document
- ≡ **Attribut** : modificateur agissant sur une instance d'un élément.

<p class="warn">

Attribut

Vocabulaire HTML

- ≡ Document HTML = flux text + balisage (markup)
- ≡ **Entité** : unité textuelle atomique SGML
 - caractères simples
 - caractères spéciaux : & ; < ; &ouelig;...
- ≡ **Étiquette ou balise** : séquence marquant le début ou la fin d'un élément
- ≡ **Élément** : une unité textuelle en tant que composant structurel d'un document
- ≡ **Attribut** : modificateur agissant sur une instance d'un élément.

<p class="warn">Exemple d'élément de paragraphe

Contenu de l'élément

Vocabulaire HTML

- ≡ Document HTML = flux text + balisage (markup)
- ≡ **Entité** : unité textuelle atomique SGML
 - caractères simples
 - caractères spéciaux : & ; < ; &ouelig; . . .
- ≡ **Étiquette ou balise** : séquence marquant le début ou la fin d'un élément
- ≡ **Élément** : une unité textuelle en tant que composant structurel d'un document
- ≡ **Attribut** : modificateur agissant sur une instance d'un élément.

<p class="warn">Exemple d'élément de paragraphe </p>

Balise de clôture

HTML évolue !

Quelques exemples de grandeurs et décadence...

- la balise blink, la balise marquee
- les frames
- les tableaux
- les formulaires

Il faut se tenir informé des évolutions du langage, des travaux en cours, etc.

Problématiques liées au format !

Un document HTML peut contenir à la fois :

- du contenu
- des structures
- des informations de présentation

Quel(s) problème(s) cela pose-t-il ?

Ce qu'il faut retenir !

- On ne peut pas utiliser n'importe quelle balise dans n'importe quel contexte ¹
- Les attributs autorisés dépendent de l'élément
- Certains attributs sont obligatoires
- Les balises véhiculent un sens : il faut les utiliser pour cela et pas pour autre chose (cf. sémantique)
- Les vieilles balises de mise en forme ne doivent plus être utilisées

1. On parle de « sémantique » des balises

Structure générale d'un document

html
head
body

```
<?xml version='1.0' encoding='ISO-8859-1'>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml1-strict.dtd">

<!-- Un commentaire -->

<html>
  <head>
    <title>Titre du document</title>
  </head>
  <body>
    ... Le corps du document avec le balisage ...
  </body>
</html>
```

En-tête de document (head)

- ≡ Type de contenu, protocole, encodage : **vivement conseillé !**
- ≡ Titre du document (`title`) : **obligatoire**
- ≡ Méta-information, aide à l'indexation (`meta`)
- ≡ Feuille de style incluse (`style`) : **souvent une erreur !**
- ≡ Liens externes (javascript, CSS, etc.) (`link`)

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>HTML en une le&ccedilla;on</title>
<meta name="description" content="un cours sur le langage HTML" />
<meta name="keywords" content="html, initiation , cours" />
<style type="text/css">
  <!--
    p { text-align: justified }
  -->
</style>
</head>
```

Corps de document HTML

≡ Balisage de niveau bloc (structure en général le document)

- Titres (`h1`, `h2`, ..., `h6`)
- Paragraphes (`p`)
- Blocs préformatés (`pre`)
- Listes (`ul`, `ol`, `dl`)
- Formulaires à remplir (`form`)
- Tableaux (`table`)

≡ Balisage de niveau ligne (*inline*)

- Structuration légère : `span`
- Sémantique (`em`, `strong`, `code`, `var`, `kbd`, **acronym**, **cite**, `samp`, `dfn`)
- Physique (`sup`, `sub`)
- Ancres et liens hypertexte (`a`)
- Images (`img`)
- Contenus exécutables (`object`, `script`)

≡ Séparateurs

- `br`, `hr`

Un document XHTML complet

```
<?xml version='1.0' encoding='ISO-8859-1'>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Cours HTML</title>
    <meta name="description"
      content="un cours sur le langage HTML" />
    <meta name="keywords"
      content="html, initiation, cours" />
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <h1>HTML en une leçon</h1>

    <h2>Introduction</h2>

    <p>Il convient tout d'abord de...</p>

    <h2>Les éléments HTML</h2>

    <p>Il en existe <em>diverses sortes</em></p>

  </body>
</html>
```

Ancres et hyperliens

Hyperlien

Ancre

```
<a href "...">...</ a>
```

```
<XX id ="...">...</ XX>
```

```
http :// monsite.net/doc.html
```

```
...
```

```
...
```

```
<h2 id="p3">Partie 3</h2>
```

```
...
```

```
...
```



```
http :// monsite.net/doc.html#p3
```

```
<a href="http :// www.cpe.fr/">CPE</a>
```



Listes

Type	Exemple	Affichage
Simple	<pre> Pommes Poires </pre>	<ul style="list-style-type: none"> • Pommes • Poire
Ordonnée	<pre> Intro Contexte </pre>	<ol style="list-style-type: none"> 1 Intro 2 Contexte
Définitions	<pre><dl> <dt>HTML</dt> <dd>HyperText...</dd> <dt>SGML</dt> <dd>Standard...</dd> </dl></pre>	<p>HTML HyperText Markup. . .</p> <p>SGML Standard Generalized. . .</p>

Formulaires

- ≡ **form** : englobe tous les éléments d'un formulaire
 - **action** : URL d'un programme de traitement
 - **method** : méthode HTTP de transmission des paramètres (get ou post)
- ≡ **input** : un champ de saisie (au sens large)
 - **type** : type du champ (text, password, checkbox, radio, image, hidden, submit, reset)
 - **name** : nom du champ, passé avec la valeur correspondante au programme de traitement
 - des attributs supplémentaires existent et dépendent du type.

HTML5 apporte de nombreuses nouvelles fonctionnalités aux formulaires (sélecteurs de couleurs, champs de recherche, etc.).

Tableaux

```
<table>
  <caption>Titre du Tableau</caption>
  <tr>
    <th>Titre col 1</th>
    <th>Titre col 2</th>
    <th>Titre col 3</th>
  </tr>
  <tr>
    <td rowspan="2">objet 1</td>
    <td>objet 2</td>
    <td>objet 3</td>
  </tr>
  <tr>
    <td>objet 4</td>
    <td>objet 5</td>
  </tr>
</table>
```

Titre col 1	Titre col 2	Titre col 3
objet 1	objet 2	objet 3
	objet 4	objet 5

Un tableau ne doit jamais être utilisé pour autre chose que de la présentation de données tabulaires !

Vous n'utiliserez pas un tableur pour écrire un rapport de stage, non ?

Cascading Style Sheet

■ Objectif : découplage du fond et de la forme :

- HTML décrit le contenu
- CSS prend en charge la mise en page

■ Standardisation

- CSS 1 est une recommandation W3C depuis 1996
- CSS 2.1 est candidat à recommandation depuis février 2004
- Firefox et IE implémentent une large partie de CSS 2

■ Avantages de CSS

- Contrôle beaucoup plus fin de la présentation que HTML
- Possibilité de distinction des médias (écran, impression, vocal, etc.)
- S'applique à HTML et XML

CSS : Principes de fonctionnement

≡ Héritage

- La plupart des propriétés affectées à un élément HTML/XML s'appliquent aux éléments imbriqués
- Par défaut, certaines propriétés ne sont pas héritées. On peut le forcer

≡ Spécificité

- Si plusieurs déclarations peuvent s'appliquer, la déclaration la plus spécifique l'emporte

≡ Cascade

- Les déclarations portant sur des propriétés différentes se combinent
- Déclarations de spécificité équivalente ? La dernière apparue l'emporte.

Feuilles de style CSS

≡ Syntaxe générale

```
selecteur
{
  propriete : valeur;
  propriete : valeur;
  ...
}
```

≡ Directive @

- @page : propriétés de page (impression)
- @media : restriction de déclarations à certains media
- @import : import de déclarations depuis un fichier externe

```
@import url(generic.css);
@media print {
  body {
    font-family: serif;
  }
}
```

CSS : Sélecteurs

☰ Détermine à quels éléments s'applique la déclaration

- Voir [http :// www.w3.org/TR/CSS21/selector.html](http://www.w3.org/TR/CSS21/selector.html)

<code>e1</code>	Les éléments <code>e1</code>
<code>e11, e12</code>	Les éléments <code>e11</code> et <code>e12</code>
<code>e11 e12</code>	Les éléments <code>e12</code> contenus dans un élément <code>e11</code>
<code>e11.warn</code>	Les éléments <code>e11</code> ayant un attribut <code>class</code> de valeur <code>warn</code>
<code>e1#pos143</code>	L'élément <code>e1</code> ayant l'attribut <code>id</code> de valeur <code>pos143</code>
<code>e1:hover</code>	Un élément <code>e1</code> lorsqu'il est survolé par la souris

```
body {  
    font-family: serif;  
}  
div.resume span.motcle {  
    font-weight: bolder;  
}
```

CSS : Propriétés et directives

■ Mesure des longueurs

- Mesures relatives : `em`, `ex`, `%`, `px`
- Mesures absolues : `in`, `pt`, `pc`, `mm`, `cm`

■ Couleurs

- Prédéfinies : `aqua`, `black`, `blue`, `green`, `maroon`, `navy`, `yellow`, ...
- Numériques : `#rrvvbb`, `rgb(n, n, n)`

■ URLs

- Notation spécifique : `url(http://my.server.com/img.png)`

Propriétés usuelles (1/2)

■ Couleurs et fonds

Propriété	Valeurs possibles
color	une couleur...
background-color	<i>couleur</i> transparent
background-image	<i>url</i>
background-repeat	repeat repeat-x repeat-y

■ Polices

Propriété	Valeurs possibles
font-family	<i>police</i> sans-serif cursive...
font-size	<i>taille</i> x-small small medium large...
font-weight	normal bold bolder lighter
font-style	normal oblique italic

■ Texte

Propriété	Valeurs possibles
text-indent	<i>longueur</i> %
text-align	left right center justify
text-decoration	underline overline blink

Propriétés usuelles (2/2)

≡ Espacement

Propriété	Valeurs possibles
margin	<i>longueur %</i>
padding	longueur %

≡ Bordures

Propriété	Valeurs possibles
border	<i>width style color</i>
border-style	solid dotted dashed double...

≡ Listes

Propriété	Valeurs possibles
list-style-type	disc circle square decimal...
list-style-image	<i>url</i>
list-style-position	inside outside

CSS : Contenu bloc ou inline

- Contenu bloc : empilement vertical de contenus inline ou bloc (ex. : liste à puce, div)
- Contenu inline : contenu à l'intérieur d'un bloc d'une ligne
 - Principe : formatage sur une seule ligne, puis découpé

`<p>`Un element inline est formate
d'abord sur ``une`` ligne.`</p>`

Un élément inline est formaté d'abord sur *une* ligne.

Un élément inline est
formaté d'abord sur
une ligne.

Page

- Conséquences
 - Les marges verticales ne s'appliquent pas aux éléments inline
 - Le padding vertical ne change pas l'interligne
 - Pour contrôler l'interligne, utiliser `line-height`

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

CSS : display

Les éléments HTML ont tous un type d'affichage par défaut

- `display: block`
 - `p`, `div`, `pre`, `h1-h6`, `ul`, `form`,...
 - Un bloc de texte isolé avec saut de ligne avant et après, bordure optionnelle
- `display: inline`
 - `span`, `em`, `strong`, `cite`,...
 - Du texte (découpé en ligne) à l'intérieur d'un bloc
- `display: inline-block`
 - `img`, `textarea`, `input`,...
 - Un bloc de texte sans saut de ligne avant/après, inséré dans une ligne
- `display: table`
 - `table`
 - Un tableau
- `display: list-item`
 - `li`, `dt`, `dd`
 - Un bloc avec une puce et un retrait

Autres types de `display` : `inline-table`, `table-row-group`, `table-cell`,...

Mise en page des blocs

≡ Flux normal : empilement vertical des blocs

- Espacement / décalages : jouer sur `margin` et/ou `padding`

≡ Blocs flottants

- Propriété `float` (valeurs `left`, `right` ou `none`)
- Éviter de cotoyer un flottant : propriété `clear` (valeur `left`, `right`, `both`, `none`)

≡ Blocs positionnés

- Propriété `position`
 - `static` : flux normal
 - `relative` : décalé par rapport à sa position par défaut
 - `absolute` : en dehors du flux, Position absolue par rapport au premier élément ancêtre de position non static
 - `fixed` : en dehors du flux. Position absolue par rapport à la fenêtre (*viewport*)
- Propriétés `top`, `right`, `bottom`, `left`
 - Spécifient la position pour des blocs `relative`, `absolute` ou `fixed`

Mise en page des blocs

≡ Flux normal : empilement vertical des blocs

- Espacement / décalages : jouer sur `margin` et/ou `padding`

≡ Blocs flottants

- Propriété `float` (valeurs `left`, `right` ou `none`)
- Éviter de cotoyer un flottant : propriété `clear` (valeur `left`, `right`, `both`, `none`)

≡ Blocs positionnés

- Propriété `position`
 - `static` : flux normal
 - `relative` : décalé par rapport à sa position par défaut
 - `absolute` : en dehors du flux, Position absolue par rapport au premier élément ancêtre de position non static
 - `fixed` : en dehors du flux. Position absolue par rapport à la fenêtre (*viewport*)
- Propriétés `top`, `right`, `bottom`, `left`
 - Spécifient la position pour des blocs `relative`, `absolute` ou `fixed`

Mise en page des blocs

≡ Flux normal : empilement vertical des blocs

- Espacement / décalages : jouer sur `margin` et/ou `padding`

≡ Blocs flottants

- Propriété `float` (valeurs `left`, `right` ou `none`)
- Éviter de cotoyer un flottant : propriété `clear` (valeur `left`, `right`, `both`, `none`)

≡ Blocs positionnés

- Propriété `position`
 - `static` : flux normal
 - `relative` : décalé par rapport à sa position par défaut
 - `absolute` : en dehors du flux, Position absolue par rapport au premier élément ancêtre de position non static
 - `fixed` : en dehors du flux. Position absolue par rapport à la fenêtre (*viewport*)
- Propriétés `top`, `right`, `bottom`, `left`
 - Spécifient la position pour des blocs `relative`, `absolute` ou `fixed`

Débordement de bloc

☰ Si un bloc contient trop de texte, il déborde

- Propriété `overflow`

- `visible` : le texte peut déborder du bloc (valeur par défaut)
- `hidden` : le texte est tronqué pour tenir à l'intérieur
- `scroll` : des ascenseurs sont ajoutés au bloc
- `auto` : le navigateur décide d'ajouter ou pas des ascenseurs

<http://www.csszengarden.com>

<http://bmascret.free.fr>

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Langages pseudo–dynamiques

Le dynamisme, c'est quoi ?

Le dynamisme, c'est quoi ?

- ≡ Y a des trucs qui bougent ?
- ≡ Y a des trucs qui font du bruit ?
- ≡ Y a des jeux ?

**Le dynamisme consiste en l'adaptation du contenu d'une page à un
CONTEXTE**

Langages pseudo–dynamiques

Le dynamisme, c'est quoi ?

Le dynamisme, c'est quoi ?

- ≡ Y a des trucs qui bougent ?
- ≡ Y a des trucs qui font du bruit ?
- ≡ Y a des jeux ?

Le dynamisme consiste en l'adaptation du contenu d'une page à un
CONTEXTE

Langages pseudo–dynamiques

Le dynamisme, c'est quoi ?

Le dynamisme, c'est quoi ?

- Y a des trucs qui bougent ?
- Y a des trucs qui font du bruit ?
- Y a des jeux ?

**Le dynamisme consiste en l'adaptation du contenu d'une page à un
CONTEXTE**

Langages pseudo–dynamiques

Javascript

Possibilité d'inclure de petits programmes sur une page web.

≡ Sous forme d'objets incorporés :

- applettes (java, python) ;
- object (vidéo, son, lecteurs divers, etc.) ;

≡ Sous forme de scripts

- Javascripts
- autres scripts...

Ces programmes peuvent eux-mêmes faire appel à des programmes distants (principe d'ajax).

Ils posent de grandes difficultés en terme d'accessibilité ! (accessibilité de javascript : ARIA).

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 **Accessibilité**
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Accessibilité

C'est quoi ?

- la notion d'accessibilité d'un site internet n'est pas facile à définir
- elle dépend beaucoup des intentions du concepteur
- Un site internet est dit accessible si tout utilisateur, quel que soit sa configuration matérielle et sa configuration logicielle, est capable d'apprécier le contenu de ce site de la même manière que tout autre utilisateur pourvu de configurations différentes.*
- contenu : toute information et tout document proposé
- WAI : Mettre le Web et ses services à la disposition de tous les individus, quel que soit leur matériel ou logiciel, leur infrastructure réseau, leur langue maternelle, leur culture, leur localisation géographique, ou leurs aptitudes physiques ou mentales.*

Accessibilité

Pourquoi ?

- Obligation : loi du 11 février 2005 ;
- Image de marque, politique ;
- Une situation de handicap n'est pas nécessairement associée à une déficience : nous pouvons tous en vivre une !
- Un site accessible doit nécessairement être conçu "proprement"
- Evolution facilitée
- Adaptation naturelle aux différents matériels et à des configurations de plus en plus hétérogènes (OS, navigateur, écran, etc.)

L'accessibilité n'est pas une problématique liée à la déficience, mais à la situation de handicap au sens très large

Accessibilité

Comment ?

- en respectant les normes (W3C) et en suivant les recommandations (WAI, section 508, accessiweb) ;
- en utilisant des technologies ouvertes indépendantes des OS ;
- en utilisant des formats ouverts ;
- en suivant les évolutions des normes et des technologies !

Interlude illustratif : navigation en mode texte sur un site web.

Navigation sans souris sur un site graphique

Accessibilité

Avantages/inconvénients de cette approche

Inconvénients

- Il faut souvent convaincre les décideurs...
- ... et se montrer diplomate avec les webdesigners !
- L'accessibilité est à envisager dès le début du projet, pas comme une réparation !

- Phase de conception/modélisation plus longue

Avantages

- Modélisation robuste ;
- Garantie de portabilité et d'adaptation du site ;
- Facilité d'évolution/ de relookage ;
- Aspect politique et communication.

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - HTML
 - CSS
- 4 Vers le dynamisme...
 - Javascript/Applet/Flash
- 5 Accessibilité
- 6 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement

Méthodologie de conception

Phase 1 : analyse

☰ Identifier le contexte :

- acteurs du projet, dont le public visé
- existence d'une charte graphique
- accessibilité des autres pages web si poursuite d'un projet existant.

☰ Identifier les objectifs

- ajouter aux objectifs du site la notion d'accessibilité ;
- définition de ses contraintes d'accessibilité ;
- définition du niveau d'accessibilité.

☰ Identifier les moyens

- qui fait quoi ?
- en combien de temps ?
- qui teste ?

Méthodologie de conception

Phase 2 : préparation

Collecter, préparer et organiser les ressources

- percevoir l'organisation physique du site (hiérarchie, répertoires dédiés).
- Une structure claire (et logique !) est facilement maintenable.
- alternative aux formats propriétaire (conversion si besoin)
- attention aux ressources non accessibles !

Réaliser des patrons

- c'est là que nos amis du design peuvent s'exprimer !
- penser aux ergonomes
- valider !

Identifier le contenu et la forme

Méthodologie de conception

Phase 3 : contenu

≡ STRUCTURER

- trouver les blocs logiques
- utiliser les bonnes structures
- structurer logiquement le document

≡ SAISIR

≡ ADAPTER

Un exemple de réalisation d'une page web :

[http://amerinsa.insa-](http://amerinsa.insa-lyon.fr/portail/doku.php?id=formation:disciplines:1a:informatique:c2i:htmlcss)

[lyon.fr/portail/doku.php?id=formation:disciplines:1a:informatique:c2i:htmlcss](http://amerinsa.insa-lyon.fr/portail/doku.php?id=formation:disciplines:1a:informatique:c2i:htmlcss)

Méthodologie de conception

Phase 4 : mise en forme

- Une feuille de style est directement liée à la structure du document
- Elle est codée indépendamment : il ne devrait PAS y avoir de style dans des balises de contenu
- Il peut y en avoir plusieurs pour un même site.

Interlude : <http://www.csszengarden.com> et <http://bmascret.free.fr>

Méthodologie de conception

Phase 5 : tests et déploiement

- tester est indispensable
- valider le code HTML ET l'accessibilité
- il existe de nombreux outils de validation...
- ... mais ce ne sont pas des hommes !

Illustration : quelques outils de validation en ligne

QUESTIONS ?