

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
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

Plan

1 Introduction

- Présentation du module
- Ressources

2 Architecture web

- Généralités

3 Langage Statiques

- Problématique
- HTML
- CSS

4 Accessibilité

5 Méthodologie de conception

- Analyse
- Préparation
- Gestion du contenu
- Mise en forme
- Tests et déploiement

6 Mettre en forme un document HTML

7 Vers le dynamisme...

- Le dynamisme, c'est quoi ?
- Javascript/Applet/Flash
- DOM et javascript
- Événements HTML
- Quelques exemples d'utilisation de javascript
- Bases de javascript
- Les formulaires

8 ...Au vrai dynamisme !!!

- Architectures dynamiques
- CGI
- PHP

9 Généralisation : XML

- Présentation des architectures
- Validation et modèles

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) : Problématique, historique du Web, accessibilité, méthodologie de conception
- ☰ séance 2 : cours 2 (2h) : Mise en forme de document, web dynamique local avec javascript
- ☰ séance 3 : cours 3 (2h) : Le DOM (Document Object Model) et javascript

Introduction

Présentation du module

TLI 1 : bases des langages et technologies de l'Internet

TLI 1 : bases des langages et technologies de l'Internet																			
		S1	S2	S3	Séance 4		S5	Séance 6		Séance 7		S8	Séance 9		S10	Séance 11		Séance 12	
		C1	C2	C3	TP1		C4	TP2		TP3		C5	TP4		C6	TP5		TP6	
		Intro HTML WAI	CSS JS 1	JS 2 DOM	HTML Accessib. Bonne Prat.		PHP Base	CSS JS		JS DOM		XML	Formulaires PHP1		BD	HttpRequest Intro AJAX		Stockage données et connaissances (SQL et XML)	
Cours		2	2	2			2					2			2				
TP						4		4		4		4		4		4		4	
4ETI	A	Mercredi 8:00-12:15	BM	BM	BM	MS DC	BM	MS DC	MS DC	BM	MS DC	BM	MS DC	FP	MS DC	MS DC	MS DC	MS BM	
		Vendredi 13:30-17:45				25 nov.		2 déc.	9 déc.		16 déc.		6 janv.		13 janv.				
	B	13:30-17:45	5/11	12/11	16/11	DC BM	26/11	DC MS	DC BM	10/12	MS DC	10/12	MS DC	8 janv.	15 janv.				
		Mardi 13:30-17:45	10-12h	10-12h	10-12h	TB CP	8h-10h	TB CP	TB CP	8h-10h	TB CP	8h-10h	TB CP	8 janv.	15 janv.				
	C	13:30-17:45	10-12h	10-12h	10-12h	24 nov.	1 déc.	8 déc.	15 déc.	5 janv.	12 janv.								
		MS BM				BM MS	MS BM	BM MS	BM MS	BM MS									
	D	Lundi 8:00-12:15	10-12h	10-12h	10-12h	23 nov.	30 nov.	7 déc.	14 déc.	4 janv.	11 janv.								
						MS BM	BM MS	MS BM	BM MS	BM MS	BM MS								

SEANCE TP	SEANCE COURS
-----------	--------------

BM	Bruno Mascret
FP	Françoise Perrin
DC	Dino Cosmas
MS	Mohammed Sallami
CP	Charles Perrin
TB	Timothé Bordiga

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, Dino Cosmas, Mohamed Sallami, Charles Perrin, Timothé Bordiga**

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

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

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

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

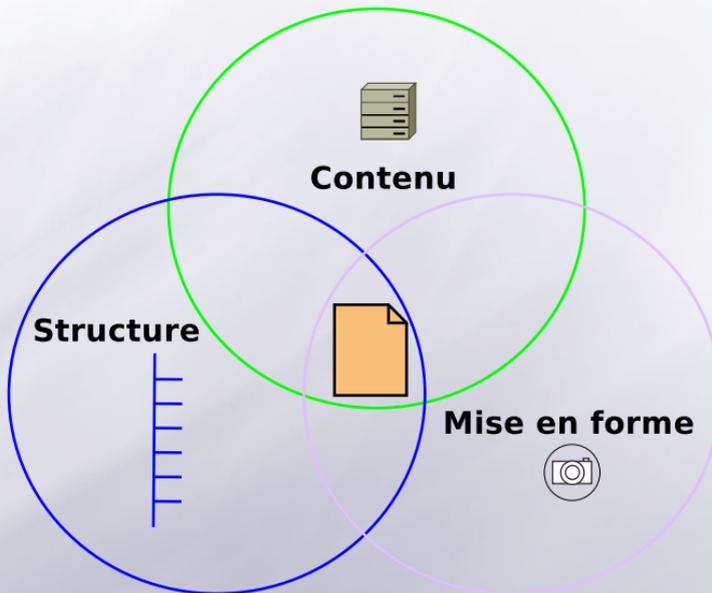
- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

Problématique

Qu'est-ce qu'un document ?

Comment représenter des données et des contenus sur le web ?

Problématique



Quelques exemples....

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*)

```
1 <!DOCTYPE report PUBLIC "-//CPE//DTD Report//EN">
2 <report >
3 <author>John Doe</author>
4 <date>Feb 1st, 2007</date>
5 <title >Report on WWW</title >
6 <h>Introduction
7 <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

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "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 : `&`; `<`; `œ`...
- ≡ **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 : `&`; `<`; `œ`...
- ≡ **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

```
1 <?xml version='1.0' encoding='ISO-8859-1'>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1
  Strict//EN"
3 "http://www.w3.org/TR/xhtml11/DTD/xhtml-strict.
  dtd">
4
5 <!-- Un commentaire -->
6
7 <html>
8 <head>
9 <title>Titre du document</title>
10 </head>
11 <body>
12 ... Le corps du document avec le balisage ...
13 </body>
14 </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`)

```
1 <head>
2 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
3 <title >HTML en une le&ccedilla;on</title >
4 <meta name="description" content="un cours sur le langage HTML" />
5 <meta name="keywords" content="html,initiation,cours" />
6 <style type="text/css">
7 <!--
8 p { text-align: justified }
9 -->
10 </style >
11 </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

```
1 <?xml version='1.0' encoding='ISO-8859-1'>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <html>
5 <head>
6 <title>Cours HTML</title>
7 <meta name="description"
8 content="un cours sur le langage HTML" />
9 <meta name="keywords"
10 content="html, initiation, cours" />
11 <link rel="stylesheet" type="text/css" href="style.css" />
12 </head>
13 <body>
14 <h1>HTML en une leçon</h1>
15
16 <h2>Introduction</h2>
17
18 <p>Il convient tout d'abord de...</p>
19
20 <h2>Les éléments HTML</h2>
21
22 <p>Il en existe <em>diverses sortes</em></p>
23
24 </body>
25 </html>
```

Ancre et hyperliens

Ancre

Hyperlien

```
<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

```

1 <table>
2 <caption>Titre du Tableau</caption>
3 <tr>
4 <th>Titre col 1</th>
5 <th>Titre col 2</th>
6 <th>Titre col 3</th>
7 </tr>
8 <tr>
9 <td rowspan="2">objet 1</td>
10 <td>objet 2</td>
11 <td>objet 3</td>
12 </tr>
13 <tr>
14 <td>objet 4</td>
15 <td>objet 5</td>
16 </tr>
17 </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

Nous l'étudierons en détail lors du prochain cours.

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 **Accessibilité**
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

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
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 **Méthodologie de conception**
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

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

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

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

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

```
1 selecteur
2 {
3   propriete : valeur ;
4   propriete : valeur ;
5   ...
6 }
```

☰ Directive @

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

```
1 @import url(generic.css);
2 @media print {
3   body {
4     font-family: serif;
5   }
6 }
```

CSS : Sélecteurs

☰ Détermine à quels éléments s'applique la déclaration

- Voir <http://www.w3.org/TR/CSS21/selector.html>

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

```

1  body {
2  font-family: serif;
3  }
4  div.resume span.motcle {
5  font-weight: bolder;
6  }
```

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

```
1 <p>Un element inline est formate
2 d'abord sur <em>une</em> 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

Feuilles responsive

La technique de design responsive (à la mode...) permet d'adapter le comportement d'une mise en page en fonction :

- ☰ de dimensions (taille de fenêtre, d'écran...)
- ☰ de media (impression, devices)
- ☰ de contextes (présentation, lecture rapide...).

Elle s'appuie sur :

- ☰ des directives
- ☰ des unités de mesures contextualisées (pourcentage de taille de fenêtre, d'écran, etc.) : vw, vh, vmin (minimum entre vw et vh), vmax..., rem (root em, constant tout au long du document)

Lire <http://www.alsacreations.com/article/lire/1615-cest-quoi-le-responsive-web-design.html>

Quelques exemples de CSS :

- ☰ <http://www.csszengarden.com> : un des meilleurs sites démontrant la puissance de CSS ;
- ☰ <http://bmascret.free.fr> : un exemple basique de CV accessible, avec feuilles CSS alternatives ;
- ☰ les exemples et tutoriels donnés sur le support des TPs.

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

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).

Langages pseudo–dynamiques

Javascript

DOM est une API représentant un document dans un navigateur web.

L'objet représentant l'instance en mémoire de la page est représenté par la variable *document*.

Langages pseudo-dynamiques

Javascript

```
1  /*Repris et adapté par Bruno Mascret de http://dmouronval.developpez.com/tutoriels/html/formulaires-html5/
2  * Les trois auteurs et traducteur
3
4  * Jan Kleinert *
5  * Traducteur : Didier Mouronval
6  * Bovino
7
8  * Licence creative commons By
9  * http://creativecommons.org/licenses/by/3.0/deed.fr
10 */
11 function check(input) {
12     if (input.value != document.getElementById('email_addr').value) {
13         input.setCustomValidity('Les deux adresses e-mail ne correspondent pas.');
```

```
14     } else {
15         // le champ est valide : on réinitialise le message d'erreur
16         input.setCustomValidity('');
17     }
18 }
19
20 function getTotal(form){
21     form.total.value = (form.nights.valueAsNumber * 15) +
22     ((form.guests.valueAsNumber - 1) * 10)
23 }
```

Browser Object Model

☰ Extension non standardisée de DOM

☰ Permet d'accéder

- aux fenêtres affichées, au navigateur
- à des objets utilitaires, comme le XMLHttpRequest (nous l'étudierons plus tard).

window

location
history
frames
navigator

appName
appVersion

Quelques méthodes d'exploitation de DOM en javascript

☰ Voir le cours XML

☰ Récupérer éléments et attributs

```
1 paragraphs = document.getElementsByTagName ("p")
2 element = paragraphs[0].childNodes[2]
3 field = document.getElementById ("creditcard")
4 field_type = field.getAttribute ("type")
```

☰ Modifier le texte d'un élément

```
1 score = document.getElementById ("score")
2 score.firstChild.nodeValue = 12
```

☰ Ajouter un élément

```
1 heading = document.createElement ("h1");
2 text = document.createTextNode ("Conclusion");
3 heading.appendChild(text);
4 document.body.appendChild(heading);
```

Attributs HTML d'événements

HTML utilise des attributs de certaines balises pour indiquer comment réagir à des événements. Ce sont ces événements auxquels il sera possible d'associer un script javascript.

onabort	img	Interruption de chargement
onblur, onfocus	form, frame, window	Gain et perte de focus
onchange	select, text, textarea	Modification de contenu
onclick	élément de formulaire, lien	Clic sur l'élément
ondblclick	élément de formulaire, lien	Double-clic sur l'élément
onerror		Erreur lors du chargement
onkeydown	document, bouton, lien	Appui sur une touche
onkeyup	document, bouton, lien	Touche relevée
onkeypress	document, bouton, lien	onkeydown suivi de onkeyup
onload	img, window	Fin de chargement de page
onunload	window	Changement de page
onmousedown, onmouseup	document, bouton, lien	Bouton de la souris
onmouseover, onmouseout	la plupart des éléments	Survol d'un élément

Javascript : modification d'image au survol

```
1 <a href="target.html"  
2 onmouseout="document.getElementById('butc').src='contact.png' "  
3 onmouseover="document.getElementById('butc').src='contactp.png' ">  
4   
8 </a>
```

Javascript : manipulation de styles CSS (1/3)

☰ CSS

```
1  .hinted {
2  color: blue;
3  }
4  .hintbox {
5  position: relative;
6  }
7  .hint {
8  display: none;
9  width: 10em;
10 position: absolute;
11 left: 0; top: -2em;
12 background-color: #ffffcc;
13 font-size: 75%;
14 padding: 0.5em;
15 }
```

Javascript : manipulation de styles CSS (2/3)

☰ Javascript

```
1 function toggle(id,display) {  
2   elem = document.getElementById(id);  
3   if (display)  
4     elem.style.display = 'block';  
5   else  
6     elem.style.display = 'none';  
7 }
```

Javascript : manipulation de styles CSS (3/3)

☰ XHTML

```
1 <p>Petit exemple
2 <span class="hinted"
3 onmouseover="toggle('xhtml',true)"
4 onmouseout="toggle('xhtml',false)">XHTML</span>
5 <span class="hintbox">
6 <span class="hint" id='xhtml'>eXtensible Hypertext Markup
7 Language</span>
8 </span>
9 et Javascript : une infobulle d'aide contextuelle.</p>
```

Javascript : caractéristiques générales

☰ Syntaxe dérivée de Java et de C++

☰ Variables non typées

☰ Seules les variables locales **doivent** être déclarées (`var`)

☰ Types de données

- scalaire

- nombre (12, 13.34, 6.02e23)
- booléen (true, false)
- chaîne ("test", "un 'a' ")
- regexp (/foo/i)

- composé

- tableau (a[0]=12; b=[1, true])
- objet (o.x=12; o.name="poisson")

☰ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++

- ≡ Variables non typées

- ≡ Seules les variables locales **doivent** être déclarées (`var`)

- ≡ Types de données

- scalaire

- nombre (12, 13.34, 6.02e23)
- booléen (true, false)
- chaîne ("test", "un 'a' ")
- regexp (/foo/i)

- composé

- tableau (a[0]=12; b=[1, true])
- objet (o.x=12; o.name="poisson")

- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++
- ≡ Variables non typées
- ≡ Seules les variables locales **doivent** être déclarées (`var`)
- ≡ Types de données
 - scalaire
 - nombre (12, 13.34, 6.02e23)
 - booléen (true, false)
 - chaîne ("test", "un 'a' ")
 - regexp (/foo/i)
 - composé
 - tableau (a[0]=12; b=[1, true])
 - objet (o.x=12; o.name="po")
- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++
- ≡ Variables non typées
- ≡ Seules les variables locales **doivent** être déclarées (`var`)
- ≡ Types de données
 - scalaire
 - nombre (12, 13.34, 6.02e23)
 - booléen (true, false)
 - chaîne ("test", "un 'a' ")
 - regexp (/foo/i)
 - composé
 - tableau (a[0]=12; b=[1, true])
 - objet (o.x=12; o.name="point")
- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++
- ≡ Variables non typées
- ≡ Seules les variables locales **doivent** être déclarées (`var`)
- ≡ Types de données
 - scalaire
 - nombre (12, 13.34, 6.02e23)
 - booléen (true, false)
 - chaîne ("test", "un 'a' ")
 - regexp (/foo/i)
 - composé
 - tableau (a[0]=12; b=[1, true])
 - objet (o.x=12; o.name="point")
- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++
- ≡ Variables non typées
- ≡ Seules les variables locales **doivent** être déclarées (`var`)
- ≡ Types de données
 - scalaire
 - nombre (12, 13.34, 6.02e23)
 - booléen (true, false)
 - chaîne ("test", "un 'a' ")
 - regexp (/foo/i)
 - composé
 - tableau (a[0]=12; b=[1, true])
 - objet (o.x=12; o.name="point")
- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : caractéristiques générales

- ≡ Syntaxe dérivée de Java et de C++
- ≡ Variables non typées
- ≡ Seules les variables locales **doivent** être déclarées (`var`)
- ≡ Types de données
 - scalaire
 - nombre (12, 13.34, 6.02e23)
 - booléen (true, false)
 - chaîne ("test", "un 'a' ")
 - regexp (/foo/i)
 - composé
 - tableau (a[0]=12;b=[1,true])
 - objet (o.x=12; o.name="point")
- ≡ Tous les types de données sont associés à un type objet fondamental : Number, Boolean, String, Array, Object, RegExp, Dates, Function, Math

Javascript : objets et tableaux

Objet : collection de données et fonctions (propriétés) indexée par une clef symbolique

```
1 var o = new Object();
2 var now = new Date();
3 var pt = {x:212, y:38}
4 function pt_translate (dx, dy)
5 {
6   this.x += dx
7   this.y += dy;
8 }
9 pt.translate = pt_translate
```

Tableau : collection de données indexée par des entiers 0..n

```
1 var a=new Array()
2 var b=new Array(10)
3 var c=new Array(1,2,3)
4 var d=[1,2,3]
5 var e=[1,true,[1,2],[x:1,y:2],"Hello"]
```

Javascript : chaînes de caractères

☰ Opérateurs

```
1 var str1 = "Javascript"
2 var str2 = "en 10 lecons"
3 var str = str1 + " " + str2
4
5 str += " faciles !"
6
7 if (reponse == "oui")
8   ...
9
10 if (mot1 < mot2)
11   alert (mot1 + " est avant " + mot2 + " dans le dictionnaire");
12
13 if (reponse != pass)
14   alert ("Mauvais mot de passe")
```

☰ Toute chaîne est un objet String

```
1 if (pass.length < 8)
2   alert("Mot de passe trop court")
3
4 debut = reponse.substring(0,3)
```

Javascript : constructions originales

☰ for...in : listage des propriétés d'un objet

```
1 function dump_props (obj, objName)
2 {
3   var result = ""
4   for (var i in obj)
5   {
6     result += objName + "." + i + " = " + obj[i] + "<br />"
7   }
8   retult += "<br />"
9   return result
10 }
```

☰ var : déclaration de variables

```
1 var num_hits = 0, cust_no = 0
```

☰ with : sélection de portée sur un objet

```
1 var a, x, y, r = 10
2 with (Math)
3 {
4   a = PI * r * r
5   x = r * cos (PI)
6   y = r * sin (PI / 2)
7 }
```

Intégration Javascript / HTML

☰ L'élément `script` : intégration du code Javascript

```
1
2 /*Une BONNE manière de faire*/
3 <script type="text/javascript" src="http://host/myscript.js" />
4
5 /*Une MAUVAISE manière de procéder*/
6 <script type="text/javascript">
7 <!--
8 function alert(msg)
9 {
10 alert("ALERTE : " + msg)
11 }
12 /-->
13 </script>
```

Pourquoi faut-il éviter la deuxième manière de procéder ?

☰ Exécution du code Javascript

- Lors de l'affichage du document pour les balises `script`
- Sur événement grace aux attributs HTML d'événements

Les formulaires

- Transmettent des informations dans des variables POST ou GET
- Les variables sont encapsulées dans la requête HTTP
- La **méthode**, POST ou GET, doit être précisée ;
- il est possible de vérifier les données côté client (javascript, html 5) ;
- il est **OBLIGATOIRE de vérifier les données côté serveur !**

Les formulaires

```

1 </li>
2 <label for="guests">Nombre d'invités (10 E par invité supplémentaire) :</label>
3 <input type="number" id="guests" name="guests" value="1" min="1" max="4" required>
4 </li>
5 </li>
6 <label for="total">Total :</label>
7 <output id="total" name="total">99</output>.00 E
8 </li>
9 </li>
10 <label for="promo">Code de promotion :</label>
11 <input type="text" id="promo" name="promo" pattern="[A-Za-z0-9]{6}"
12 <title="Le code de promotion contient six caractères alphanumériques.">
13 </li>
14 </li>
15 <input type="submit" value="Effectuer la réservation" />
16 </li>
17 </ul>
18 </form>

```

<http://localhost/tp5/formulaire.html>

Les formulaires

```
1  /*Repris et adapté par Bruno Mascret de http://dmouronval.developpez.com/tutoriels/html/formulaires-html5/
2  * Les trois auteurs et traducteur
3  *
4  * Jan Kleinert *
5  * Traducteur : Didier Mouronval
6  * Bovino
7  *
8  * Licence creative commons By
9  * http://creativecommons.org/licenses/by/3.0/deed.fr
10 */
11 function check(input) {
12     if (input.value != document.getElementById('email_addr').value) {
13         input.setCustomValidity('Les deux adresses e-mail ne correspondent pas.');
```

```
14     } else {
15         // le champ est valide : on réinitialise le message d'erreur
16         input.setCustomValidity('');
17     }
18 }
19
20 function getTotal(form){
21     form.total.value = (form.nights.valueAsNumber * 15) +
22     ((form.guests.valueAsNumber - 1) * 10)
23 }
```

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

Architecture dynamiques

Il y aura exécution de code **côté serveur**

- CGI
- Server APIs (Apache modules, ISAPI)
- ASP, PHP, JSP
- Serveurs d'application (Zope, *groupware)
- Requêtes en arrière-plan : Javascript et AJAX

Modes d'exécution côté serveur

☰ CGI : Common Gateway Interface

Machine Serveur



☰ Active Server Pages (ASP), PHP

Machine serveur



Modes d'exécution côté serveur

CGI : Common Gateway Interface

Machine Serveur



Active Server Pages (ASP), PHP

Machine serveur



Modes d'exécutions côté serveur (2)

Server API (Apache modules, ISAPI, NSAPI)

Machine Serveur

Client	Serveur HTTP	Module / DLL Module / DLL	HTTP
--------	-----------------	------------------------------	------

Java servlets et serveurs d'applications

Machine Serveur

Client	Serveur HTTP	JVM	servlet servlet	HTTP
--------	-----------------	-----	--------------------	------

Modes d'exécutions côté serveur (2)

Server API (Apache modules, ISAPI, NSAPI)

Machine Serveur

Client	Serveur HTTP	Module / DLL Module / DLL	HTTP
--------	-----------------	------------------------------	------

Java servlets et serveurs d'applications

Machine Serveur

Client	Serveur HTTP	JVM	servlet servlet	HTTP
--------	-----------------	-----	--------------------	------

Modes d'exécutions côté serveur (3)

Javascript avec Ajax (Objet XMLHttpRequest)

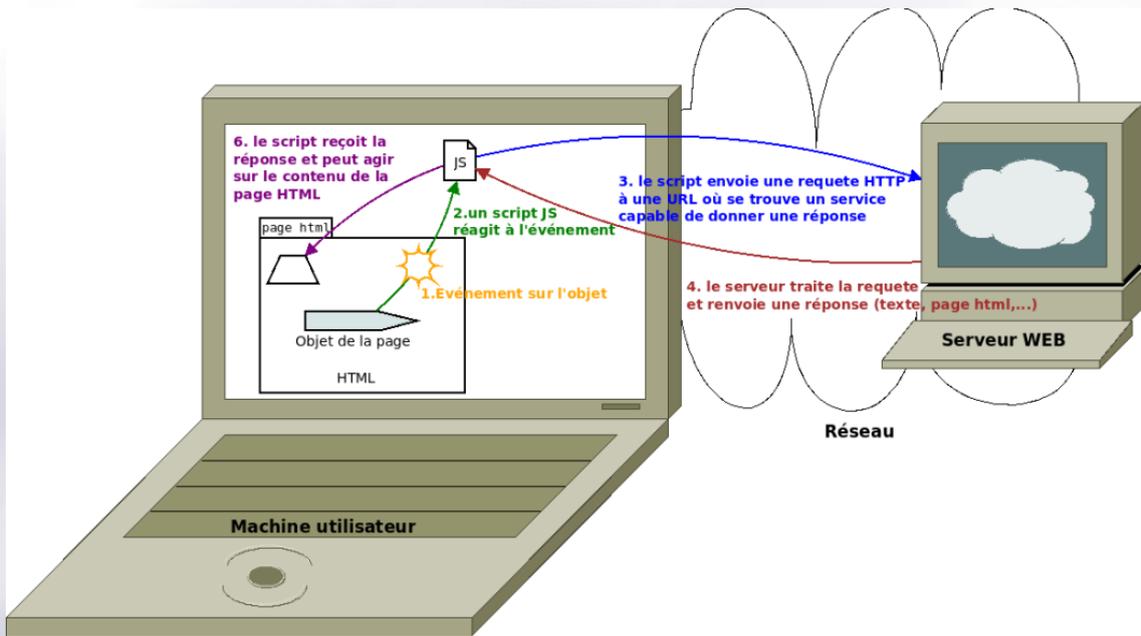


FIGURE : Principe du fonctionnement d'AJAX

CGI : Common Gateway Interface

Machine serveur

Client

Serveur
HTTP

Script ou
exécutable

Perl, C/C+, Python,...

CGI sur HTTP

CGI spécifie :

- Que le serveur HTTP et le processus de traitement communiquent via `stdin/stdout`
- Un certain nombre de variables d'environnement passées par le serveur HTTP au processus de traitement

Variables d'environnement CGI

SERVER_SOFTWARE	Nom et version du logiciel serveur HTTP
SERVER_NAME	Nom d'hôte ou adresse IP du serveur
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.0
SERVER_PORT	Port TCP du serveur Web
REQUEST_METHOD	GET, HEAD ou POST
SCRIPT_NAME	Nom virtuel du script invoqué
QUERY_STRING	La ligne d'arguments de la requête (sans décodage)
REMOTE_HOST	Le nom d'hôte du client
REMOTE_ADDR	L'adresse IP du client
AUTH_TYPE	Méthode d'authentification (si applicable)
REMOTE_USER	Utilisateur distant (si disponible)
REMOTE_IDENT	Identification de l'utilisateur distant par le serveur
CONTENT_TYPE	Type de données reçues (requête POST)
CONTENT_LENGTH	Longueur des données reçues (requête POST)
HTTP_*	Lignes d'entête HTTP reçues du client

CGI : document renvoyé

- Par défaut le serveur renvoie le document généré tel quel.
- Il est donc indispensable d'inclure au moins l'entête de type de contenu

```
1 content-type: text/html
```

Ligne vide indispensable

```
1 <html>
2 <head>
3 <title>Ok</title>
4 </head>
5 <body>
6 <p>Merci pour tout</p>
7 </body>
8 </html>
```

CGI : document renvoyé

- Par défaut le serveur renvoie le document généré tel quel.
- Il est donc indispensable d'inclure au moins l'entête de type de contenu

```
1 Content-type: text/html
```

Ligne vide indispensable

```
1 <html>
2 <head>
3 <title>Ok</title>
4 </head>
5 <body>
6 <p>Merci pour tout</p>
7 </body>
8 </html>
```

Critique de CGI

- ≡ Mécanisme simple et universel
- ≡ Indépendance du langage du processus de traitement
- ≡ Isolation du processus de traitement du serveur web
- ≡ Le lancement d'un processus externe à chaque requête est très lourd (ce qui peut se contourner en utilisant des modules du serveur web comme `mod_perl` ou FastCGI).

PHP

- ≡ Langage de scriptage côté serveur
- ≡ Fonctionnellement analogue à ASP (Microsoft)
- ≡ Multi plates-formes (même si plutôt lié au départ à Linux et Apache)
- ≡ Pilier du quatuor Linux-Apache-MySQL-PHP (LAMP)
- ≡ Syntaxe mêlant C et Perl
- ≡ Documentation très fournie (et en français !) sur <http://www.php.net/manual/fr/>
- ≡ Très nombreuses bibliothèques
 - Connexion bases de données
 - Paiement “sécurisé”
 - XML
 - Réseau
 - Accès OS
 - etc. (plus de 200 bibliothèques aujourd’hui)

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

Différentes versions

Vraiment utilisable depuis la version 3, il s'agissait auparavant d'un petit langage d'automatisation de certaines tâches.

PHP5 est la version 5 actuellement la plus diffusée (en attendant version 6...)

Les apports de PHP 5 :

- ≡ plus rapide
- ≡ nouvelles fonctionnalités (tableau, base de données InterBase, internationalisation, flux, date, etc.)
- ≡ intégration de SQLite
- ≡ nouveau modèle objet (utilisation de références)
- ≡ quelques (rares) incompatibilités

PHP : généralités sur le langage

☰ Syntaxe

- Sensible à la case
- instructions terminées par des ;
- Commentaires C (`/**`), C++ (`//`) ou shell (`#`)
- Variables préfixées par \$
- Variables non typées
- Déclaration automatique à l'affectation

☰ Types de données

Scalaire			composé		
booléen	entier	flottant	chaîne	objet	tableau
TRUE	16	0.017	"test"	\$inst->web=12;	
False	020	17.0e-3	'un *a*'		
	0x10		\$variable		
			\$valeur		
				indexé	associatif
				\$a[0]=12;	\$j["jan"]=31;

PHP : généralités sur le langage

☰ Syntaxe

- Sensible à la case
- instructions terminées par des ;
- Commentaires C (`/**`), C++ (`//`) ou shell (`#`)
- Variables préfixées par \$
- Variables non typées
- Déclaration automatique à l'affectation

☰ Types de données

Scalaire			composé		
booléen	entier	flottant	chaîne	objet	tableau
TRUE	16	0.017	"test"	\$inst = new Inst;	
False	020	17.0e-3	'un *s*'		
	0x10		\$variable		
			\$valeur		
				indexé	associatif
				\$a[0]=12;	\$j["jan"]=31;

PHP : généralités sur le langage

☰ Syntaxe

- Sensible à la case
- instructions terminées par des ;
- Commentaires C (`/**`), C++ (`//`) ou shell (`#`)
- Variables préfixées par \$
- Variables non typées
- Déclaration automatique à l'affectation

☰ Types de données

Scalaire			composé		
booléen	entier	flottant	chaîne	objet	tableau
TRUE	16	0.017	"test"	\$inst = new Inst;	
False	020	17.0e-3	'un *s*'		
	0x10		\$variable		
			\$valeur		
				indexé	associatif
				\$a[0]=12;	\$j["jan"]=31;

PHP : généralités sur le langage

☰ Syntaxe

- Sensible à la case
- instructions terminées par des ;
- Commentaires C (`/**`), C++ (`//`) ou shell (`#`)
- Variables préfixées par \$
- Variables non typées
- Déclaration automatique à l'affectation

☰ Types de données

Scalaire			composé	
booléen	entier	flottant	chaîne	objet
TRUE	16	0.017	"test"	objet
FALSE	020	17.0e-3	'un "a"'	tableau
	0x10		'\$variable'	
			"\$valeur"	
				indexé
				associatif
				\$a[0]=12;
				\$j["jan"]=31;

PHP : généralités sur le langage

☰ Syntaxe

- Sensible à la case
- instructions terminées par des ;
- Commentaires C (`/**`), C++ (`//`) ou shell (`#`)
- Variables préfixées par \$
- Variables non typées
- Déclaration automatique à l'affectation

☰ Types de données

Scalaire			composé		
booléen	entier	flottant	chaîne	objet	tableau
TRUE	16	0.017	"test"	\$inst->mb=12;	
False	020	17.0e-3	'un "a"'		
	0x10		'\$variable'		
			"\$valeur"		
				indexé	associatif
				\$a[0]=12;	\$j["jan"]=31

Opérations de base

- ≡ Opérations classiques sur les entiers et les flottants,
- ≡ Les opérateurs sont en général les mêmes qu'en C,
- ≡ Opérations simples sur les chaînes de caractères (concaténation par `.`), transformées en un entier en cas d'opérations arithmétiques,
- ≡ Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- ≡ Opérateur de référence : `& : $a = &$b;`

Opérations de base

- ≡ Opérations classiques sur les entiers et les flottants,
- ≡ Les opérateurs sont en général les mêmes qu'en C,
- ≡ Opérations simples sur les chaînes de caractères (concaténation par `.`), transformées en un entier en cas d'opérations arithmétiques,
- ≡ Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- ≡ Opérateur de référence : `& : $a = &$b;`

Opérations de base

- ≡ Opérations classiques sur les entiers et les flottants,
- ≡ Les opérateurs sont en général les mêmes qu'en C,
- ≡ Opérations simples sur les chaînes de caractères (concaténation par `.`), transformées en un entier en cas d'opérations arithmétiques,
- ≡ Déclaration de tableaux : `$arr = array("foo"=> "bar", 12 => true);` ou `$arr[2] = 53;`
- ≡ Opérateur de référence : `& : $a = &$b;`

Opérations de base

- ≡ Opérations classiques sur les entiers et les flottants,
- ≡ Les opérateurs sont en général les mêmes qu'en C,
- ≡ Opérations simples sur les chaînes de caractères (concaténation par `.`), transformées en un entier en cas d'opérations arithmétiques,
- ≡ Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- ≡ Opérateur de référence : `&` : `$a = &$b;`

Opérations de base

- ≡ Opérations classiques sur les entiers et les flottants,
- ≡ Les opérateurs sont en général les mêmes qu'en C,
- ≡ Opérations simples sur les chaînes de caractères (concaténation par `.`), transformées en un entier en cas d'opérations arithmétiques,
- ≡ Déclaration de tableaux : `$arr = array("foo" => "bar", 12 => true);` ou `$arr[2] = 53;`
- ≡ Opérateur de référence : `&` : `$a = &$b;`

PHP : classes

☰ Exemple

```
1 class test
2 {
3     var $str = "Hello world!";
4     function init($str)
5     {
6         $this->str = $str;
7     }
8 }
9 $class = new test;
10 print $class->str;
11
12 $class->init("Coucou");
13 print $class->str;
```

PHP : classes

☰ Héritage : par le mot clef `extends`

```
1 class A { var $str = "Hello world!"; }
2
3 class B extends A { var $bla = 'foo'; }
4
5 $objet = new B;
6 print $objet->str;
```

☰ Pas d'héritage multiple, ni de visibilité, ni de destructeurs, mais opérateur de classes :

PHP : classes

☰ Héritage : par le mot clef `extends`

```
1 class A { var $str = "Hello world!"; }
2
3 class B extends A { var $bla = 'foo'; }
4
5 $objet = new B;
6 print $objet->str;
```

☰ Pas d'héritage multiple, ni de visibilité, ni de destructeurs, mais opérateur de classes ::

Constructeurs

Un constructeur est simplement une fonction (méthode) qui a le même nom que la classe.

```
1 <?php
2 class A {
3     function A() {
4         echo "Je suis le constructeur de A.<br />\n";
5     }
6     function B() {
7         echo "Je suis une fonction standard appelee B dans la classe A.<br />\n";
8         echo "Je ne suis pas le constructeur de A.<br />\n";
9     }
10 }
11
12 class B extends A {
13 }
14 // Cette syntaxe va appeler B() comme constructeur.
15 $b = new B;
16 ?>
```

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (public, private, protected)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

Les classes en PHP 5

Le système d'objet a totalement été réécrit pour PHP 5, qui propose beaucoup de nouvelles fonctionnalités.

- ≡ Constructeurs et destructeurs (`__construct` et `__destruct`)
- ≡ Visibilité (`public`, `private`, `protected`)
- ≡ Méthodes statiques
- ≡ Constantes de classes
- ≡ Classes abstraites
- ≡ Interfaces
- ≡ Méthodes finales
- ≡ Clonage
- ≡ Introspection
- ≡ ...

PHP : structures de contrôles

1	<code>if (\$hue==1) {</code>	1	<code>if (\$hue==1):</code>
2	<code>\$color = "blue";</code>	2	<code>\$color = "blue";</code>
3	<code>} elseif (\$hue==2) {</code>	3	<code>elseif (\$hue==2):</code>
4	<code>\$color = "green";</code>	4	<code>\$color = "green";</code>
5	<code>} else {</code>	5	<code>else:</code>
6	<code>\$color = "unknown";</code>	6	<code>\$color = "unknown";</code>
7	<code>}</code>	7	<code>endif;</code>
1	<code>while (\$i-->0) {</code>	1	<code>while (\$i--):</code>
2	<code>print \$i;</code>	2	<code>print \$i;</code>
3	<code>}</code>	3	<code>endwhile;</code>
1	<code>do {</code>	1	<code>do {</code>
2	<code>print \$i;</code>	2	<code>print \$i;</code>
3	<code>} while (\$i < 12);</code>	3	<code>} while (\$i < 12);</code>
1	<code>for (\$i=0; \$i<10; \$i++) {</code>	1	<code>for (\$i=0; \$i<10; \$i++):</code>
2	<code>print \$i;</code>	2	<code>print \$i;</code>
3	<code>}</code>	3	<code>endfor;</code>
1	<code>\$tableau = array('05' => "abc", '35' => "def");</code>		
2	<code>foreach (\$tableau as \$clef => \$valeur)</code>		
3	<code>{</code>		
4	<code>print \$clef . ' ' . \$valeur . '
';</code>		
5	<code>}</code>		

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

- Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2     global $var;
3     echo $var;           => Hello world
4
5     $var = "Hello world";
6     test();

```

- Certaines variables sont *superglobales*

- Mot clef `static` comme en C.

- Passage de paramètres par valeur ou par référence

```

1 function sqr(x) 1 $x = 2;
2 $x = $x * $x; 2   sqr($x);
3 return $x;    3   print "sqr(x)";
4              4   sqr(&$x);
5              5   print $x;

```

=>

- Variables dynamiques

```

1 $foo = "hello!";
2 $bar = "foo";    => hello!
3 print $$bar;

```

- Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE; // affiche "Bonjour le monde."
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

≡ Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

≡ Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

≡ Certaines variables sont *superglobales*

≡ Mot clef `static` comme en C.

≡ Passage de paramètres par valeur ou par référence

```

1   1 $x = 2;
2   2   function sqr(x) 2   sqr($x);           =>   2
3   3   $x = $x + $x; 3   print "$x\n";
4   4   return $x; 4   sqr(&$x);           4
5   5   print $x; 5   print $x;

```

≡ Variables dynamiques

```

1 $foo = "hello!";
2 $bar = "foo";           =>   hello!
3 print $$bar;

```

≡ Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE;
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

- Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

- Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

- Certaines variables sont *superglobales*

- Mot clef `static` comme en C.

- Passage de paramètres par valeur ou par référence

```

1   1 $x = 2;
2   2   function sqr(x) 2   sqr($x);           =>   2
3   3   $x = $x + $x;    3   print "$x\n";
4   4   return $x;      4   sqr(&$x);           4
5   5   print $x;

```

- Variables dynamiques

```

1 $foo = "hello!";
2 $bar = "foo";           =>   hello!
3 print $$bar;

```

- Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE;
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

≡ Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

≡ Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

≡ Certaines variables sont *superglobales*

≡ Mot clef `static` comme en C.

≡ Passage de paramètres par valeur ou par référence

```

1   1 $x = 2;
function sqrt(x) 2   sqrt($x);
2   2 $x = $x + $x; 3   print "$x\n";           =>   2
3   3 return $x;    4   sqrt(&$x);
4   4               5   print $x;
5   5

```

≡ Variables dynamiques

```

1 $foo = "hello!";
2 $bar = "foo";           =>   hello!
3 print $$bar;

```

≡ Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE;
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

≡ Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

≡ Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

≡ Certaines variables sont *superglobales*

≡ Mot clef `static` comme en C.

≡ Passage de paramètres par valeur ou par référence

```

1 function sqr(x) { 1 $x = 2;
2   $x = $x * $x; 2   sqr($x);
3   return $x;    3   print "$x<br />";   =>   2
4 }               4   sqr(&$x);
5               5   print $x;           4

```

≡ Variables dynamiques

```

1 $foo = "hello!";
2 $bar = "hello!";   =>   hello!
3 print $bar;

```

≡ Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE;
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

≡ Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

≡ Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

≡ Certaines variables sont *superglobales*

≡ Mot clef `static` comme en C.

≡ Passage de paramètres par valeur ou par référence

```

1 function sqr(x) { 1 $x = 2;
2   $x = $x * $x; 2   sqr($x);
3   return $x;    3   print "$x<br />";   =>
4 }              4   sqr(&$x);
5               5   print $x;           4

```

≡ Variables dynamiques

```

1 $foo = "hello !";
2 $bar = "foo";    =>   hello !
3 print $$bar;

```

≡ Constantes

```

1 define('CONSTANTE', "Bonjour le monde.");
2 echo CONSTANTE;
3 echo $CONSTANTE;

```

PHP : particularités (1/2)

≡ Inclusion d'autres fichiers par `include()`, `require`, `include_once` et `require_once`

≡ Variables globales non visibles des fonctions, mot clé `global`

```

1 function test() {
2   global $var;
3   echo $var;           =>   Hello world
4 }
5 $var = "Hello world";
6 test();

```

≡ Certaines variables sont *superglobales*

≡ Mot clef `static` comme en C.

≡ Passage de paramètres par valeur ou par référence

```

1 function sqr(x) { 1 $x = 2;
2   $x = $x * $x; 2   sqr($x);
3   return $x;    3   print "$x<br />";   =>
4 }               4   sqr(&$x);
5                 5   print $x;           4

```

≡ Variables dynamiques

```

1 $foo = "hello !";
2 $bar = "foo";    =>   hello !
3 print $$bar;

```

≡ Constantes

```

1 define("CONSTANTE", "Bonjour le monde.");
2 echo CONSTANTE; // affiche "Bonjour le monde."
3 echo Constante; // affiche "Constante" et une note

```

PHP : particularités (2/2)

- Les fonctions peuvent être appelées par leur nom :

```
1 function salut ()
2 {
3 echo 'bonjour';
4 }
5 $a = 'salut';
6 $a(); // affiche bonjour
```

- Les arguments de fonctions peuvent être par valeur, par référence et/ou avec une valeur par défaut (comme en C++)

PHP : particularités (2/2)

- Les fonctions peuvent être appelées par leur nom :

```
1 function salut ()
2 {
3 echo 'bonjour';
4 }
5 $a = 'salut';
6 $a(); // affiche bonjour
```

- Les arguments de fonctions peuvent être par valeur, par référence et/ou avec une valeur par défaut (comme en C++)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- ☰ Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- ☰ Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- ☰ Initialisez **toutes** vos variables !
- ☰ Vérifiez le contenu des variables fournies par l'utilisateur (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- ≡ Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- ≡ Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- ≡ Initialisez **toutes** vos variables !
- ≡ Vérifiez le contenu des variables fournies par l'utilisateur (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- ≡ Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- ≡ Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- ≡ Initialisez **toutes** vos variables !
- ≡ Vérifiez le contenu des variables fournies par l'utilisateur (chemin, injections sql, etc.)

Problèmes de sécurités

L'utilisation de PHP est en soi un problème de sécurité. Il faut alors faire tout particulièrement attention à certains points :

- ≡ Les scripts PHP ont accès à (au moins une partie, au moins en lecture) du système de fichier du serveur.
- ≡ Vos scripts PHP peuvent être inclus par des scripts en provenance d'autres serveurs. . .
- ≡ Initialisez **toutes** vos variables !
- ≡ Vérifiez le contenu des variables fournies par l'utilisateur (chemin, injections sql, etc.)

Intégration PHP/HTML

≡ Insertion de code PHP dans HTML

Technique la plus simple, mais à **proscrire dès que le volume de code augmente !**

1	<html>	1	<html>
2	<head>	2	<head>
3	<title>Configuration PHP</title>	3	<title>Configuration PHP</title>
4	</head>	4	</head>
5		5	
6	<body>	6	<body>
7	<h1>Configuration PHP</h1>	7	<h1>Configuration PHP</h1>
8		8	
9	<?php phpinfo(); ?>	9	<? phpinfo(); ?>
10	</body>	10	</body>
11	</html>	11	</body>
		12	</html>

≡ Variables prédéfinies

- Variables génériques (similaires à CGI : \$HTTP_USER_AGENT, \$REMOTE_ADDR, ...)
- Arguments de requête (POST, GET et COOKIES) dans des tableaux globaux :

```

1 $_GET['mavARIABLE'],
2 $_POST['mavARIABLE'],
3 $_COOKIE['moncookie']

```

PHP avancé

Les variables super globales

- ☰ Accessibles PARTOUT...

- ☰ ...ce qui ne veut pas dire qu'il soit recommandé de les utiliser partout !!!

Principales variables superglobales :

- ☰ `$_SERVER` : valeurs données par le serveur (très nombreuses !).

- ☰ `$_ENV` : variables d'environnement données par le serveur. ex : "USER", nom de l'utilisateur

- ☰ `$_SESSION` : variables pour un client qui sont stockées sur le serveur le temps de sa présence sur le site.

- ☰ `$_COOKIE` : variables enregistrés sur l'ordinateur du client. Pas de durée de fin de vie.

- ☰ `$_GET` : données envoyées en paramètres dans l'URL.

- ☰ `$_POST` : données de la page encapsulées dans la requête (mais visible avec les bons outils quand même !)

- ☰ `$_FILES` : liste des fichiers envoyés par la page (majoritairement via formulaire).

PHP avancé

Les variables super globales

Utilisation

```
1 <?php
2
3 $_COOKIE[preference] = array();
4 $_COOKIE['preference'][couleur] = 'vert';
5
6 $_SESSION['nom'] = 'Mascret';
7 $_SESSION['mail'] = $_POST['mail'];
8
9 if(isset($_GET['action'])){
10     if($_GET['action']=="menu"){
11         affiche_Menu();
12     }
13     elseif($_GET['action']=="cmd"){
14         afficheCommande();
15     }
16 }
17 else{
18
19     echo "rien a faire";
20 }
21 ?>
```

PHP avancé

Les variables super globales

Utilisation

```
1 <?php
2 echo "<h1>Serveur</h1>";
3 print_r($_SERVER);
4 displayTab($_SERVER);
5 echo "<h1>Env</h1>";
6 print_r($_ENV);
7 displayTab($_ENV);
8
9 echo "<h1>POST</h1>";
10 print_r($_POST);
11 displayTab($_POST);
12
13 echo "<h1>GET</h1>";
14 print_r($_GET);
15 displayTab($_GET);
16
17 function displayTab($t){
18     echo "<ul>";
19     foreach($t as $k=>$v){
20         echo "<li>". $k .": ". $v. "</li>";
21     }
22     echo "</ul>";
23
24 }
25
26 ?>
```

<http://localhost/tp5/lib/main.php>

PHP avancé

Les sessions

Hors programme, étudié en détail en spécialité....

- Permettent de maintenir une pseudo-connexion
- Optimisation
- Calculs facilités
- Gains en temps... si utilisées !
- Utilisent explicitement la superglobale `$_SESSION` et implicitement `$_COOKIE`

PHP avancé

Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : `session_start()` demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
 - le serveur dispose chez lui du numéro de session et des données associées ;
 - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui-ci transfère son identifiant. **Attention : Il est indispensable d'appeler `session_start()` à chaque nouvelle requête !** Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (`session_destroy()`), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé

Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : `session_start()` demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
 - le serveur dispose chez lui du numéro de session et des données associées ;
 - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui-ci transfère son identifiant. **Attention : Il est indispensable d'appeler `session_start()` à chaque nouvelle requête !** Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (`session_destroy()`), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé

Les sessions

Exemple : Déroulement d'une session

1. Ouverture de session : `session_start()` demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
 - le serveur dispose chez lui du numéro de session et des données associées ;
 - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui-ci transfère son identifiant. **Attention : Il est indispensable d'appeler `session_start()` à chaque nouvelle requête !** Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (`session_destroy()`), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

Exemple : Déroulement d'une session

1. Ouverture de session : `session_start()` demande à PHP de générer un numéro unique (PHPSESSID) en hexadécimal (bf4c57deff89006ca44347dfe9egf899 par exemple) qui est transmis au client lors de la réponse par un cookie. A ce stade :
 - le serveur dispose chez lui du numéro de session et des données associées ;
 - le client dispose d'un cookie avec son identifiant.
2. Les scripts PHP sont alors en mesure de créer et stocker des informations sous forme de variables dans l'espace session **côté serveur** (chaînes, nombres, tableaux, etc.) ;
3. A chaque nouvelle requête client, celui-ci transfère son identifiant. **Attention : Il est indispensable d'appeler `session_start()` à chaque nouvelle requête !** Le programme PHP est en mesure grâce à cet identifiant de retrouver les données chez lui.
4. Après un certain temps d'inactivité, ou lorsque l'utilisateur est déconnecté (`session_destroy()`), les données sont supprimées sur le serveur. Il faut alors commencer une nouvelle session.

PHP avancé

Les sessions

Exemple simple

```
1 <?php
2 session_start();
3
4 $_SESSION['prenom'] = 'Bruno';
5 $_SESSION['nom'] = 'Mascret';
6 $_SESSION['mail'] = $_POST['mail'];
7 ?>
```

PHP avancé

Les sessions

Exemple complet

```
1 <?php
2 session_start();
3
4 if(isset($_SESSION['nom'])){
5     echo "Session ouverte:".$_REQUEST['SESSION_NAME'];
6 }
7 else{
8     if(verifLoginPass($_POST['login'], $_POST['pass'])){
9         //identifiant et pass ok
10
11         $_SESSION['prenom'] = $_POST['prenom'];
12         $_SESSION['nom'] = $_POST['nom'];
13         $_SESSION['mail'] = $_POST['mail'];
14     }
15     else{
16         echo "Erreur";
17     }
18 }
19
20 ?>
```

Plan

- 1 Introduction
 - Présentation du module
 - Ressources
- 2 Architecture web
 - Généralités
- 3 Langage Statiques
 - Problématique
 - HTML
 - CSS
- 4 Accessibilité
- 5 Méthodologie de conception
 - Analyse
 - Préparation
 - Gestion du contenu
 - Mise en forme
 - Tests et déploiement
- 6 Mettre en forme un document HTML
- 7 Vers le dynamisme...
 - Le dynamisme, c'est quoi ?
 - Javascript/Applet/Flash
 - DOM et javascript
 - Événements HTML
 - Quelques exemples d'utilisation de javascript
 - Bases de javascript
 - Les formulaires
- 8 ...Au vrai dynamisme !!!
 - Architectures dynamiques
 - CGI
 - PHP
- 9 Généralisation : XML
 - Présentation des architectures
 - Validation et modèles

Introduction

XML : modèles et instances de modèles
Quels intérêts, quelles applications ?

- ☰ représentation de données
- ☰ organisation arborescente des données
- ☰ spécification des formats
- ☰ ...

Des API existent pour pratiquement chaque langage (php, java, c++, etc.)

Introduction

XML : modèles et instances de modèles
Quels intérêts, quelles applications ?

- ☰ représentation de données
- ☰ organisation arborescente des données
- ☰ spécification des formats
- ☰ ...

Des API existent pour pratiquement chaque langage (php, java, c++, etc.)

Présentation des architectures

XML

Notions de balises, attributs, commentaire

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <?xml-stylesheet type="text/xsl" href="nombres.xsl" ?>
3 <liste_nombres>
4   <nombre valeur="10">dix</nombre>
5   <nombre valeur="0">zéro</nombre>
6   <nombre valeur="33">trente trois</nombre>
7   <nombre valeur="6">le premier nombre parfait &#233;</nombre>
8   <secret cache="ytjtfghkhu">pas montrer
9     <attention>le mot de passe &#133;</attention>
10    ça
11  </secret>
12 </liste_nombres>
```

Modèles, validations

XML

Problème : qu'est-ce qu'un document XML valide ?

Deux réponses :

- ☰ un document VALIDE DANS LA FORME (qui respecte la syntaxe XML)
- ☰ un document VALIDE DANS SUR LE FOND (qui respecte un MODÈLE)

Modèles, validations

XML

Exemple de document non valide sur la forme

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <liste_nombres>
3   <nombre valeur="10">dix
4   <nombre valeur="0">zéro</nombre>
5   <nombre valeur="33">trente trois</nombre>
6   <nombre valeur="6">le premier nombre parfait &#233;<nombre>
7   <secret cache="ytjtfghkkuj">pas montrer
8     <attention>le mot de passe &#133;</attention>
9     ç
10  </secret>
11 </liste_nombres>
12 <liste_nombres>
13   <nombre valeur="10">dix</nombre>
14 </liste_nombres>
```

Modèles, validations

XML

Exemple de document non valide sur le fond

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <?xml-stylesheet type="text/xsl" href="nombres.xsl" ?>
3
4 <nombre valeur="10">
5   <liste_nombres>
6     <nombre valeur="0">zÃ©ro</nombre>
7     <nombre valeur="33">trente trois</nombre>
8     <nombre value="6">le premier nombre parfait &#233;</nombre>
9     <secret truc="ytjtfghkuj">pas montrer
10    <attention>le mot de passe &#133;</attention>
11    Ã§a
12  </secret>
13  </liste_nombres>
14 </nombre>
```

Modèles, validations

XML

Comment donner un modèle à un document XML ?

Deux solutions :

- ☰ fournir une DTD
- ☰ fournir un schema

Modèles, validations

XML

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN"
3 "http://www.w3.org/Math/DTD/mathml2/xhtml1-math11-f.dtd" >
4 <html xml:lang="fr-FR" xmlns="http://www.w3.org/1999/xhtml">
5   <head>
6     <title>TEST Mathml</title>
7   </head>
8   <body dir="ltr">
9     <h1 id="toc0">Exemple de formule mathématique en mathML</h1>
10    <p>
11      <math xmlns="http://www.w3.org/1998/Math/MathML">
12        <mrow>
13          <mrow>
14            <mfrac>
15              <mn>2</mn>
16              <msqrt>
17                <mrow>
18                  <mn>5</mn>
19                <mo stretchy="false">&plus;</mo>
20                <mi>x</mi>
21              </mrow>
22            </msqrt>
23          </mfrac>
24          <mo stretchy="false">&equals;</mo>
25          <mo stretchy="false">&alpha;</mo>
26        </mrow>
27        <mrow>
28          <mo stretchy="false">&pi;</mo>
29          <mo stretchy="false">&plus;</mo>
30          <mo stretchy="false">&mu;</mo>
31        </mrow>
32      </math>
33    </p>

```

Modèles, validations

XML

Cas concret : réalisation de DTD et de schema xml

`exemples/document.xml`

`exemples/windob.dtd`

`exemples/newWindob2.xsd`