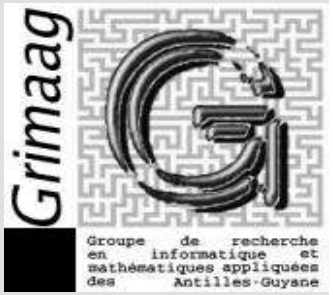
Master Info



PROGRAMMATION WEB AVANCEE

Plis fòs ba pengwen là !

Vincent Pagé vpage@univ-ag.fr Université Antilles Guyane

Présentation de l'EC

- Module Complet : 12h C / 12h TD / 12h TP => 12 / 0 /24
- Cette partie : 6h C / 0h TD / 12h TP
- Objectif du cours : Conception d'applications Web
- Plan du cours :
  - Rappels sur les sites web
  - Notion d'architecture Web
  - Client riches
  - AJAX

Introduction :
Application Web

De plus en plus, les applications professionnelles se construisent sur une architecture Web.

Exemple : Gestion des clients chez Orange...

(!= web service non abordé ici)

Il s'agit d'applications client/serveur, le serveur centralisant des données pour de nombreux clients. Trois modèles de développement :

- Client Lourd (le client fait tout, le serveur répond aux requêtes)
- Client Léger (le serveur fait tout, le client ne fait qu'afficher)
- Client Semi Léger / client riche

(intermédiaire entre les deux. ex : AJAX)

Introduction :
Application Web

Intérêt des modèles client légers :

Choix : client = navigateur Web

- Aucune maintenance du client.
- Inter-opérabilité totale (heu.....)

Pour les entreprises, ces opérations de maintenances représentent 80% du budget logiciel...

Inter-opérabilité possible en cas de

RESPECT DES STANDARDS

- par les constructeurs de navigateurs
- par les concepteurs de pages Web.

Introduction :
Rappels sur les sites Web

Bonnes pratiques de conception :

- Cahier des charges.
    - Fonctionnalités
    - Charte Graphique.
- Langage XHTML + CSS (mise en forme)
- Tests sur de nombreux navigateurs

- Si possible, utiliser un CMS existant
- Si possible, utiliser un framework de développement
    (dans les tp, pas de framework car long à apprendre)

Introduction :
conception d'une application Web

Une application Web est... une application !

- Maintenance du site faite par des non informaticiens
  => Prévoir une interface de maintenance (back end)
    - pas de règles particulières : programmation...
    - une seule règle : maximum d'infos dans la BDD
    - Bien distinguer ce qui doit etre possible ou non.

=> Temps de développement plus grand pour la back end (plus complexe en général)

Introduction :
Problèmes pour les application Web

Technique :

- Chaque action charge une nouvelle page
  => Beaucoup de trafic (si une seule chose change)
    => Ne charger qu'un morceau de page ???

Marketing :

- Pas forcément très beau (peu d'animations...)
  => Rendre possible le sapin de Noël (...)

Introduction :
Remarques non techniques

-10 ans : Client lourd => application perso / pb d'interoperabilite
- 5 ans : Prise en compte des recommandation W3C
- 4 ans : Client Léger => bonne interoperabilité
- 2 ans : apparition du Web 2.0 => clients riches...

Retour en arriere ? OUI (pour la technique)
mais bien orchestré par le marketing de grosses entreprises
=> inévitable.

=> Pour vous : Faire des clients riches « propres »

Introduction :
Problèmes des clients riches

En tant que site Web :

- Pas d'historique de navigation
- pas de bookmark
- Pb d'indexation.
- Pb d'accessibilité

=> On peut partiellement y arriver néanmoins.

Introduction :
quelques technologies clients riches

Ici : clients riches = belles applications...

Les anciens (et souvent bases des nouveaux)

- Flash / Applet Java / Javascript

Les récents : description d'interface xml

- Adobe Flex : Du flash, avec un framework de developpement (payant)
- Open Lazlo : du flash avec un framework open source + serveur Java
- Xul (zoul) : moteur de rendu (javascript : dans firefox ou standalone) + scripts d'actions en javascript
  respecte les standards w3c...
- AJAX (pas vraiment technologie mais reste sur une base nav. Web)
- WinPF : Courche présentation chez Microsoft .net 3.0
- Appolo : environnement d'execution / nouveau navigateur chez adobe...

Ce cours

Nous nous cantonnerons à une solution AJAX sans framework :

- Domaine hyper technologique, évolution constante, pas de visibilité a 3 ans (bcp d'argent en jeu).
- Javascript : fait partie de beaucoup de solutions.
- Client = navigateur (a peu près tous)
- Serveur sans contraintes (envoyer du xml / html)

le framework peut imposer des contraintes serveur (Google Web Toolkit

Introduction AJAX

AJAX n'est pas un langage. C'est un empilement de technologies permettant de régler le problème :

Comment ne recharger qu'une partie de la page.

AJAX signifie Asynchronous Javascript And XML

En cas d'action :

- Javascript détecte l'action
- fait une requête au serveur qui lui renvoie un fichier XML.
- Javascript modifie le document pour y intégrer le contenu reçu.
- le coté Asynchrone : la page n'est pas bloquée en l'attente du fichier xml

Introduction AJAX

- Pour le moment restreint à javascript
  (sécurisé pour le client)
- Ouvre la porte à de vraies applications web sans la lourdeur des applets Java....

Beaucoup de javascript pour :

- déclencher les requetes (xmlhttprequest)
- analyser le document xml reçu
- modifier le document affiché

Javascript

En fait, plutot ECMAscript (standard)
+ diverses implémentations : javascript, jscript...

Pourquoi je n'aime pas ECMAscript :
- Implémentations différentes et incompatibles !
(notez que ce sont les DOM qui sont incompatibles)
- Votre navigateur comprend les instructions du standard
    + d'autres (innerHTML)

=> on peut utiliser un framework pour s'abstraire
un peu des problèmes d'incompatibilité javascript

(ex : dojo / prototype, mochikit / Jquery, scriptaculous....)

Javascript

Au sein d'une page HTML : balise SCRIPT :

```
<script type="text/javascript">
        <!--
            // Mes lignes de code
        -->
```

ou

```
<script type="text/javascript" src="foo.js"></script>
```

syntaxe proche du C, C++....
variables déclarées mais pas typées :
var i,j,chaine, monTableau;

Objectif : modifier le contenu des pages Web....

Javascript

3 Objets crées lorsqu'une page est présente :

- Window : décrit la fenetre du navigateur
- Navigator : contient les propriétés du navigateur
- Document : Contient les propriétés et méthodes de la page Web.

Chacun a des propriétés et méthodes utiles...
A vous de les connaître !

ne pas utiliser les props du navigator pour la compatibilité...

Javascript et le DOM

Javascript en lui meme n'agit pas sur le document...
Il le fait au travers du « DOM »

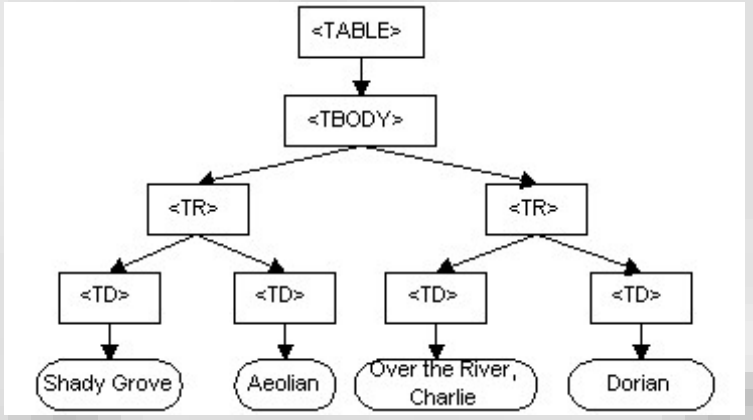
Le DOM (Document Object Model) se veut une API pour les documents html et xml...
pour le moment, DOM 1, et DOM 2...

Fournit une représentation structurelle du document et des méthodes pour accéder à son contenu et à sa présentation.

La représentation du document est celle d'un arbre.
Au sommet, l'élément html.....

Exemple d'arbre selon le DOM

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```



Propriétés des noeuds selon le DOM

ParentNode	the parent node of the current node
childNodes[n]	retrieves the n-th child of the current node
firstChild	the first child of the current node, or nothing if there's none
lastChild	the last child of the current node, or nothing if there's none
nextSibling	the next sibling of the current node, or nothing if there's none
prevSibling	the previous sibling of the current node, or nothing if there's none
nodeType	the node type, such as "TR", "IMG", "TABLE", etc.
attributes	the node's attribute objects
nodeName	the name of the node

+ nodeValue (si le noeud est de type feuille : du texte)

Javascript et le DOM

Oubliez tous les acces en document.all[""] ou document.layer....

Attrapez les elements en document.getElementById
ou document.getElementByTagName
ou toute méthode qui descend dans l'arbre !

N'utilisez pas innerHTML mais des instructions qui modifient explicitement le DOM tree...
appendChild, createTextNode, removeNode....

Javascript et le DOM

Exemple de manip d'un objet :

Mettons que notre code html contienne la ligne suivante :
 Ceci est l'ancien paragraphe

1. On repère dans le code l'element par son id (par exemple)

```
var monElement = document.getElementById("pgIntro");
```

2. On change une propriété de cet élément :

```
monElement.style.fontSize="12pt";
```

ou

```
monElement.innerHTML("Ceci est mon nouveau paragraphe");
```

(Attention innerHTML est très déconseillé)

Javascript et les evenements

Gestionnaires d'evenements :

- Liaison entre le script et les actions utilisateurs : s'inserent en tant qu'attributs d'un tag html. La valeur de l'attribut est la fonction qui gère l'évenement.

onAbort (en cas d'interruption)
onBlur (en quittant)
onChange (après modification réussie)
onClick (en cliquant)
onDblClick (en double-cliquant)
onError (en cas d'erreur)
onFocus (en activant)
onKeydown (en appuyant sur une touche)
onKeypress (en maintenant une touche appuyée)
onKeyup (en relâchant la touche)
onLoad (en chargeant un fichier)
onMousedown (en maintenant la touche de souris appuyée)
onMousemove (en bougeant la souris)
onMouseout(en quittant l'élément avec la souris)
onMouseover (en passant sur l'élément avec la souris)
onMouseUp (en relâchant la touche de souris)
onReset (en initialisant le formulaire)
onSelect (en sélectionnant du texte)
onSubmit (en envoyant le formulaire)
onUnload (en quittant le fichier)

Utilisables en fonction des tags...

Javascript et les evenements

Un exemple évènement. (pb se corsent)

voilà un code qui fonctionne (pour de très mauvaises raisons) avec IE et Firefox :

```
<html xmlns="http://www.w3c?.org/1999/xhtml" xml:lang="fr">
<head>
	<title> this is my test </title>
	<script type="text/javascript">
	<!--
	window.onload = function(){alert("fin chargement page");}

	function testEvent(ev) {
		alert('I was triggered as " type :"+ev.type);
	}

	-->
	</script>
</head>
<body onclick="testEvent(event)" onmousemove="testEvent(event)">
	<h1 onclick="testEvent(event)"> Titre </h1>
</body>
</html>
```

Javascript et les evenements

Attention

Accessibilité :

- Rien ne doit être inaccessible sans javascript
- Rien ne doit prendre en compte vos habitudes a vous
  => doublez les onclick avec des onfocus...

Compatibilité :

- les evenements ne sont pas gérés de la meme facon.
  (passage d'argument) => framework !

Surcharge de gestionnaires :

- Un seul gestionnaire window.onLoad => ecrasé...

AJAX le début

Faire une requete au serveur : XMLHttpRequest

```
var xmlHttp=new XMLHttpRequest();  // Firefox...
xmlHttp.open("GET","monfichier.xml",true);
xmlHttp.send(null);
```

Savoir que le serveur a répondu :

```
xmlHttp.onreadystatechange= function()
    {
    if(xmlHttp.readyState==4)
      {
            ....
      }
    }
```

Récupérer les donnees ;

xmlHttp.responseText ou xmlHttp.responseXML + JavaScriptObjectNotation

pour responseXML : lire les données : compliqué => sarissa

Clients riches conclusion

Faites du riche propre !

AJAX : Eventuellement si DOM...

- En codant, ayez un exemplaire du DOM sous la main :

deux DOM : le DOM xml et le DOM HTML (sur couche)
Liste des méthodes dispo et origine (
DOM 1 : xml + html obsolète.
DOM 2 : xml + html 4.01 ou xhtml 1.0 ok.
DOM 3 : en cours

Développement Web

mise en pratique

- Quels types de solutions existent ?
(site statique , site statique CMS, développement natif, développement dans un cadre de framework)

- Quels critères permettent de choisir un type de solution.
    - comment le site se maintient (et qui maintient)
    - choix de l'hébergeur.
    - budget
    - réutilisabilité / évolutionnarité.
    - besoins en design évolué.

- Quels type de solutions / infrastructure choisiriez vous pour :
dans le cas ou vous êtes
    1. Un particulier
    2. Une micro entreprise de développement web
    3. Une entreprise de développement web.

Pour
    - Le site Web de votre groupe de ragga préféré.
    - Le forum étudiant du master Info.
    - le site officiel du master Info
    - Site de vente en ligne pour une PME
    - Un site de vente en ligne pour Amazone.
    - Une application de gestion de stock
    - un site vitrine pour un artiste peintre