MAS-RAD :: Module AJAX

- Objectifs :
  - acquérir de bonnes connaissances pratiques dans le développement de client web basé sur AJAX
  - connaître et manipuler les principes et mécanismes de base de l'approche asynchrone de communication client-serveur.
  - appréhender les principes de base de la programmation événementielle.
- Contact :
  - olivier.ertz@heig-vd.ch

Organisation

- Cours :
  - introductions aux concepts et mises en pratique
  - sessions : 25.05.11 | 01.06.11 | 08.06.11 | 15.06.11 | 22.06.11

    + session d'évaluation 29.06.11
- Ressources :
  - http://ogo.heig-vd.ch/wiki/doku.php?id=ajax:start
- Pré-requis :
  - XHTML, CSS, XML et JavaScript
  - Un serveur HTTP + module PHP en localhost

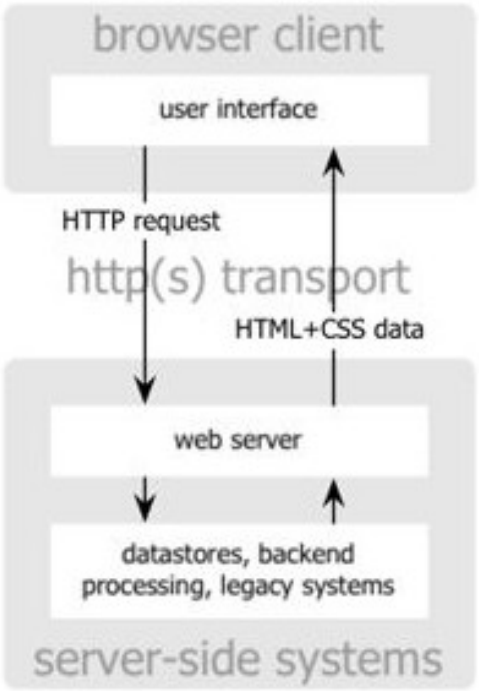
Feuille de route

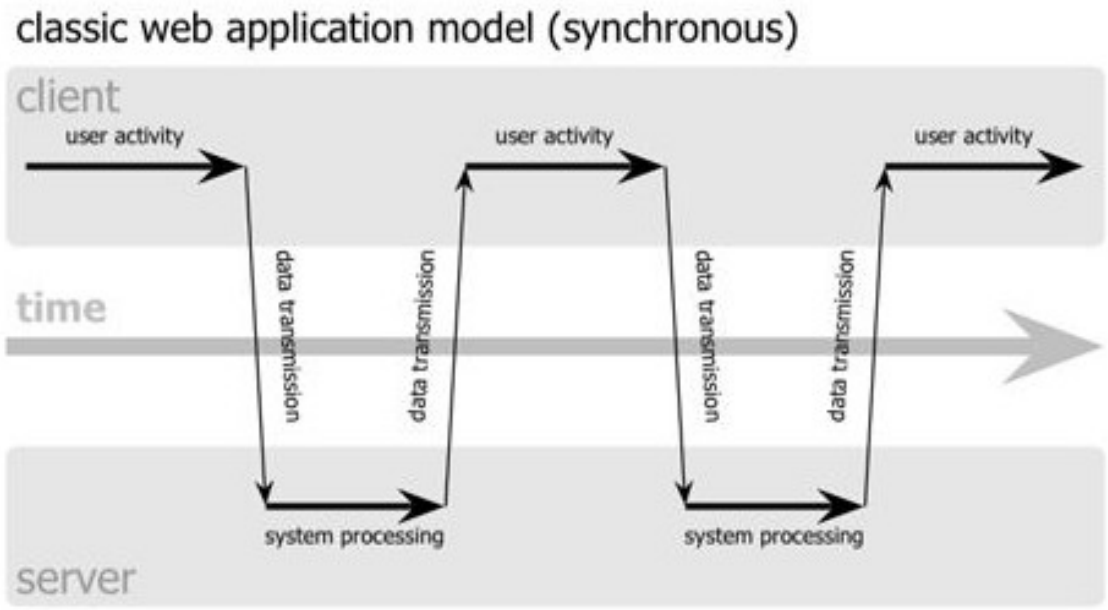
- Introduction générale
- Un premier résumé des avantages d'AJAX
- Rappels W3C : XHTML/CSS et “le DOM”
- Le format JSON (rappel ?)
- 1ère requête AJAX avec XMLHTTPRequest
- Prise en main de jQuery, "write less, do more"
- AJAX en profondeur avec jQuery
- AJAX et le dialogue "cross-domain"
- HTML5 et les "web socket"

Introduction

- AJAX (Asynchronous Javascript and XML) = combinaison de plusieurs technologies web
  - du web “statique” au web “applicatif” (Web 2.0)
  - standards W3C utilisant XHTML et CSS
  - dynamique et interaction par le DOM
  - différents formats pour l'échange de données (texte, XML, JSON, ...)
  - dialogue client-serveur asynchrone en utilisant l'objet XMLHttpRequest
  - et JavaScript pour mettre tout cela en musique

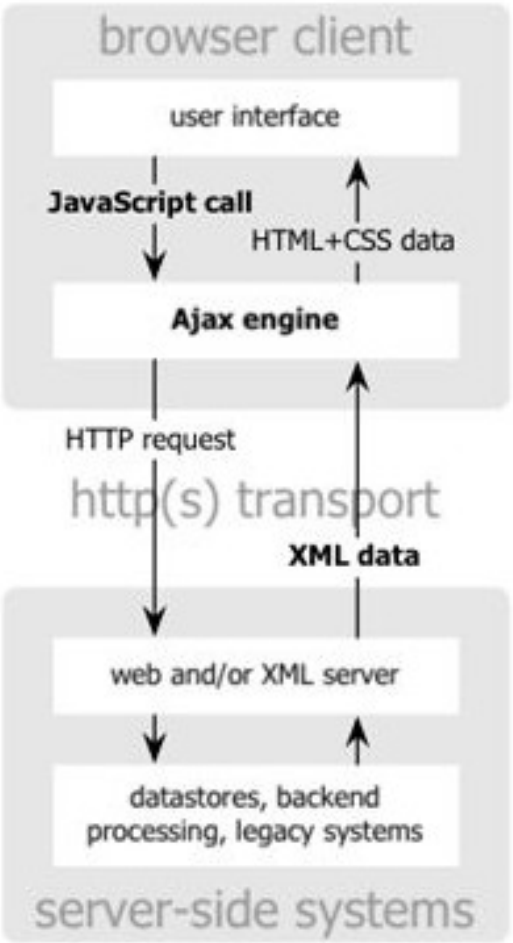
Synchronous ...

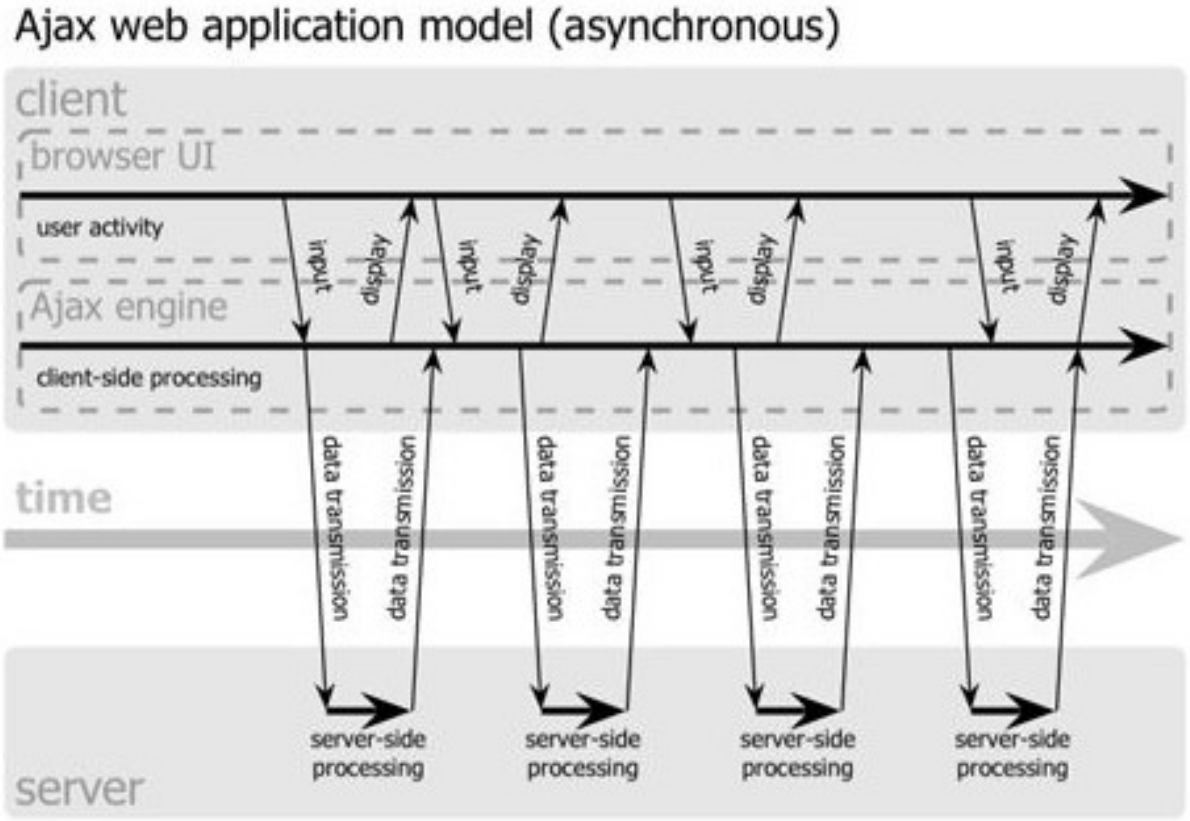




- utilisateur en attente
- page complètement rafraîchie

Asynchronous ...





De l'application web ...

- ... avec AJAX
  - la requête ne bloque pas la navigation
  - seules certaines parties peuvent être rafraîchies
  - meilleure expérience utilisateur -> RIA
  - rationnalisation du dialogue client/serveur
- Par exemples :
  - Alimenter une liste de sélection, auto-completion, galerie d'images (lightbox), webmail, Google Docs, Google Maps, ...
  - ... ???

Quelques mises au point W3C

- XHTML (eXtensible HyperText Markup Language)
  - Langage à balises pour structurer un document

    http://fr.html.net/tutorials/html/
- CSS (Cascading Style Sheets)
  - Langage de style pour présenter le contenu structuré en XHTML

    http://fr.html.net/tutorials/css/
- DOM (Document Object Model)
  - API standardisée d'accès à un document XML

    http://www.howtocreate.co.uk/tutorials/javascript/domintroduction

Format JSON

- JSON (JavaScript Object Notation) :
  - forme d'écriture de données en JavaScript
  - simple et légère au format texte
  - une arborescence de données, inspirée de XML
  - **nativement interprété** contrairement au XML
  - en JavaScript, plus aisé et plus performant
- Comment ça marche (cf. exemple) :
  - à la base, utilisation de la fonction eval() de JS
  - implémentations natives d'encodage et décodage

JSON vs. XML

```
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
     "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
     ]
   }
}}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

AJAX :: l'objet XMLHTTPRequest

- Le modèle d'application AJAX
  - des évènements associés à des éléments XHTML
  - des appels de fonctions qui manipulent le DOM
  - dialoguent avec le serveur par XMLHttpRequest

- L'objet XMLHttpRequest est essentiel
  - Apparu avec IE4
  - Méthodes GET et POST
  - Synchrone ou asynchrone

XMLHTTPRequest sous la loupe

- Deux méthodes
  - **open**: établit une connexion
  - **send**: envoie une requête au serveur
- Stockage des données en retour
  - **responseXml** pour un flux XML
  - **responseText** pour un flux brut de texte

XMLHTTPRequest sous la loupe

- Etat de la requête **readyState**
  - 0: non initialisé; 1: connexion établie;
  - 2: requête reçue; 3: réponse en cours; 4: terminé
- Propriété **onreadystatechange**
  - déclarer la fonction “callback” (changemet d'état)
- Propriété **status**
  - code status HTTP (200: ok; 404: not found)