MCours.com

Définition

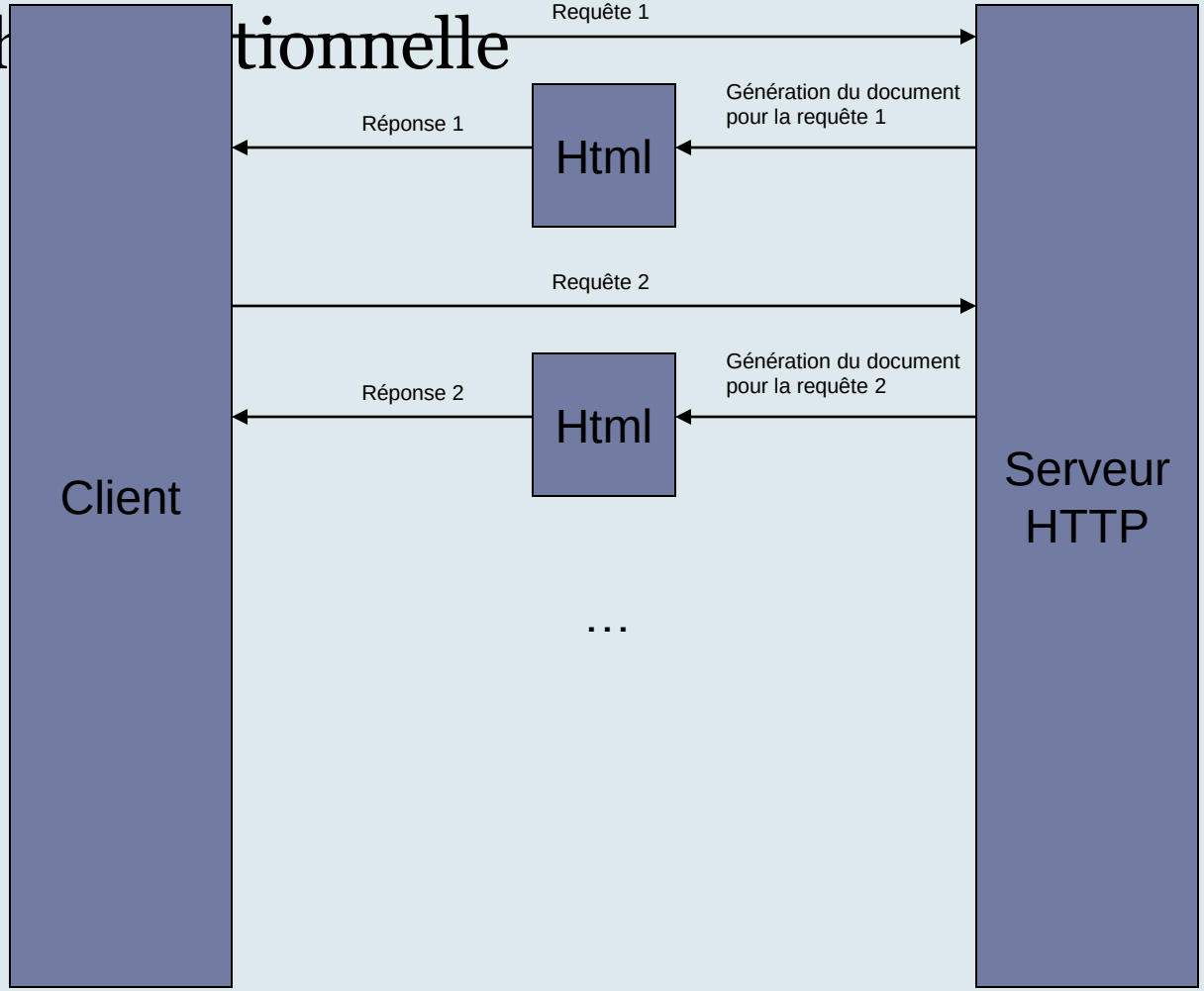
- AJAX = Asynchronous JavaScript And XML
- Méthode informatique de développement d'applications Web.
- Utilisation ***conjointe*** d'un ensemble de technologies couramment utilisées sur le Web :
  - HTML (ou XHTML) et CSS pour la mise en forme
  - DOM et JavaScript pour afficher et interagir dynamiquement avec l'information présentée
  - XML, XSLT et l'objet XMLHttpRequest pour échanger et manipuler les données de manière asynchrone avec le serveur web

Comparaison avec les applications web traditionnelles

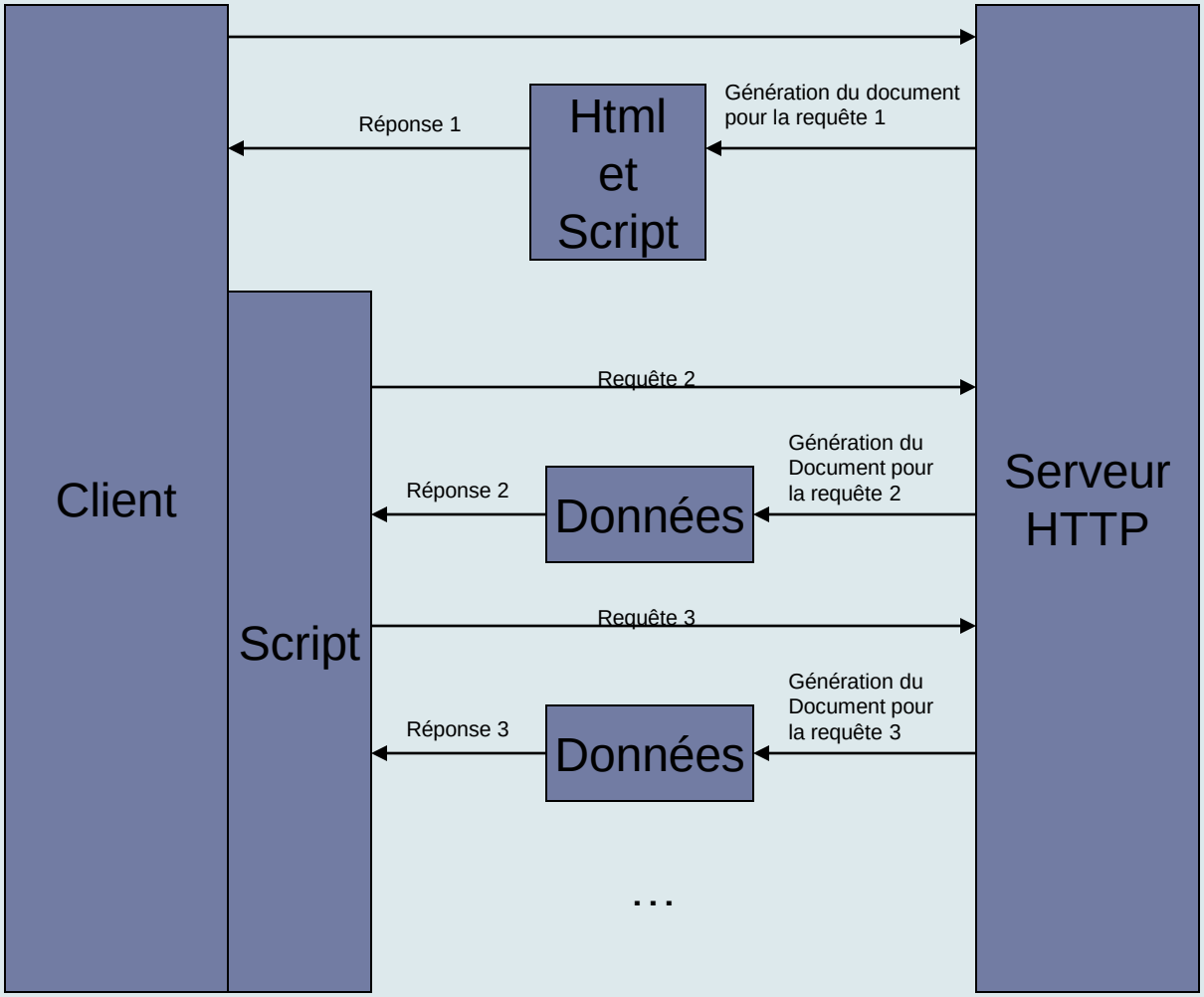
- Application WEB traditionnelle :
  - Le client envoie une requete HTTP
  - Le serveur renvoie une page
  - Cela consomme inutilement une partie de la bande passante, car une grande partie du code HTML est commun aux différentes pages de l'application
  - Le chargement d'une nouvelle page à chaque requete n'est pas ergonomique

Comparaison avec les applications web traditionnelles

Approch tionnelle



Comparaison avec les applications web traditionnelles



Comparaison avec les applications web traditionnelles

- Application AJAX :
  - Envoyer des requêtes au serveur HTTP pour ne récupérer que les données nécessaires
  - Utilisation la requête HTTP XMLHttpRequest
  - Utilisation la puissance des feuilles de style (CSS) et de Javascript côté client pour interpréter la réponse du serveur
  - Permet au navigateur de modifier partiellement la page pour la mettre à jour sans avoir à la recharger
  - Applications plus réactives, meilleure ergonomie

Comparaison avec les applications web traditionnelles

- Application AJAX :
  - quantité de données échangées fortement réduite.
  - Nécessite de charger, sur la première page, une bibliothèque AJAX volumineuse
  - Nécessite un navigateur compatible, autorisant le Javascritpt et le compasant XMLHTTP
  - Nécessite des tests minucieux car il existe de grandes différences entre les navigateurs

Qui supporte AJAX ?

- Les navigateurs Web qui supportent les technologies décrites précédemment :
  - Mozilla, Firefox, Internet Explorer 6 , Konqueror, Safari, Opera 9
- Open AJAX Initiative : groupe créé par IBM, avec des partenaires tels que
  - BEA, Borland, the Dojo Foundation, Eclipse Foundation, Google, Laszlo Systems, Mozilla Corporation, Novell, Openwave Systems, Oracle, Red Hat, Yahoo, Zend, Zimbra.

Comment cela fonctionne? (1)

- Ajax utilise un modèle de programmation comprenant d'une part la présentation, d'autre part les évènements.
- Les évènements sont les actions de l'utilisateur, qui provoquent l'appel des fonctions associées aux éléments de la page.
- L'interaction avec l'utilisateur se fait à partir des formulaires ou boutons html.
- Ces fonctions JavaScript identifient les éléments de la page grâce au DOM et communiquent avec le serveur par l'objet XmlHttpRequest.

l'objet XmlHttpRequest

- AJAX se base sur un composant embarqué dans presque tous les navigateurs récents : XmlHttpRequest
- Cet objet a d'abord été développé par Microsoft, en tant qu'objet ActiveX, pour Internet Explorer 5
- Il a ensuite été repris et implémenté sous Mozilla 1 Safari 1.2, Konqueror 3.4 et Opera 8.
- Il n'est pas supporté par les navigateurs dits de « vieille génération ».
- En avril 2006, il a été proposé pour devenir une recommandation du W3C.

Création d'un objet XMLHttpRequest

- Pour Internet Explorer (avant IE7), il faut créer un ActiveX de la manière suivante :

```
xhr = new ActiveXObject("Microsoft.XMLHTTP");
ou xhr = new ActiveXObject("Msxml2.XMLHTTP");
```

- Pour les autres navigateurs (ou à partir d'IE7), l'objet XMLHttpRequest est supporté nativement :

```
xhr = new XMLHttpRequest();
```

- Le script suivant créer l'objet selon ce que le navigateur supporte.

```
function getXMLHttpRequest()
{if (window.XMLHttpRequest) {return new XMLHttpRequest();}
else {if (window.ActiveXObject)
try {return new ActiveXObject("Msxml2.XMLHTTP");}
catch (e) {
    try {return new ActiveXObject("Microsoft.XMLHTTP");}
catch (e) {return NULL;}
}}}
```

Propriétés de l'objet XMLHttpRequest

- `onreadystatechange` : Gestionnaire d'événements pour les changements d'état. Il faut assigner une fonction à cette propriété pour effectuer des traitements sur les données renvoyées
- `readyState` : statut de l'objet.
- `responseText` : Réponse sous forme de chaîne de caractères.
- `responseXML` : Réponse sous forme d'objet *DOM*.
- `status` : code numérique de réponse du serveur *HTTP*
- `statusText` : message accompagnant le code de réponse.
  - Les code possibles pour le statut de l'objet sont :
  - 0 = non initialisé
  - 1 = ouverture. La méthode `open()` a été appelée avec succès
  - 2 = envoyé. La méthode `send()` a été appelée avec succès
  - 3 = en train de recevoir. Des données sont en train d'être transférées, mais le transfert n'est pas terminé
  - 4 = terminé. Les données sont chargées.

Méthodes de l'objet XMLHttpRequest

- `abort()` : Abandonne la requête.
- `getAllResponseHeaders()` : Renvoie l'ensemble de l'entête de la réponse sous forme de chaîne de caractères.
- `getResponseHeader("champEntete")` : Renvoie la valeur d'un champ d'entête *HTTP*.
- `open ("method","URL"[,asyncFlag[, "userName"[,"password"]]])` : Prépare une requête en indiquant la méthode, l'*URL*, la drapeau de synchronisation, le nom d'utilisateur et le mot de passe.
- `send (contenu)` : Effectue la requête, éventuellement en envoyant les données.
- `setRequestHeader("champ","valeur")` : Assigne une valeur à un champ d'entête *HTTP* qui sera envoyé lors de la requête.

Comment cela fonctionne? (2)

- Pour recueillir des informations sur le serveur, l'objet XmlHttpRequest dispose de deux méthodes:
  - - open: établit une connexion.
  - - send: envoie une requète au serveur.
- Les données fournies par le serveur seront récupérées dans les champs responseXml ou responseText de l'objet XmlHttpRequest. S'il s'agit d'un fichier xml, il sera lisible dans responseXml par les méthodes de DOM.
- Il faut créer un nouvel objet XmlHttpRequest, pour chaque fichier que vous voulez charger.

Exemple simple d'utilisation de XMLHttpRequest


```
<html><head><title>Exemple 1</title></head><body>
<script>
…
function ajax()
{ var xhr=getXMLHttpRequest();
  xhr.open("GET", "http://localhost/ajax/reponse.txt",
  false);
  xhr.send(null);
  alert(xhr.responseText);
}
</script>
<p><a href="ajax();">Cliquez-moi !</a></p>
</body></html>
```

Javascript Asynchrone

- Le choix entre synchrone et asynchrone se fait dans l'appel à XMLHttpRequest dans paramètre « flag »:
  - **true** pour asynchrone
  - **false** pour synchrone

- Dans le cas d'un appel asynchrone, le résultat est récupéré par une fonction appelée lors du déclenchement d'un événement `onreadystatechange`

```
Requete_http.onreadystatechange = function()
{ // mettre le code souhaité };
```

- Cette fonction sera appelée à chaque changement d'état de notre objet. Les états que peut prendre readyState sont:

0 non initialisée 1 en chargement
2 chargée 3 en cours de traitement
4 terminée

la partie XML de l'AJAX

- Le serveur renvoie des données XML
- La méthode responseXML de l'objet XMLHttpRequest renvoye un document XML à traiter
- La méthode javascript getElementsByTagName(nom) d'un objet (en l'occurrence XML) permet de récupérer les éléments par rapport à leur nom dans un élément.

```
var docXML= xhr.responseXML;
var items = docXML.getElementsByTagName("donnee")
for (i=0;i<items.length;i++)
{ alert (items.item(i).firstChild.data); }
```

JSON

- Le XML est une **calamité** à parser en JavaScript
- Le format JSON est beaucoup plus approprié
- JSON (JavaScript Object Notation) est un format de données générique qui utilise la notation des objets JavaScript pour transmettre de l'information structurée
- Exemple :

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

Comment utiliser le format JSON

- **Coté client**
  JSON faisant partie de la norme JavaScript. Le contenu d'un fichier JSON, ou la définition de données dans ce format sont assignés à une variable, laquelle devient un objet du programme.
- **Coté serveur**
  Les fichiers au format JSON s'utilisent dans différents langages de programmation, notamment PHP et Java grâce à des parseurs qui permettent d'accéder au contenu, et éventuellement de le convertir en classes et attributs, dans ce langage.
- **L'échange de données**
  - inclusion directe du fichier dans la page HTML au même titre qu'un fichier .js de JavaScript.
  - Ou emploi de XMLHttpRequest.

Exemple d'utilisation de JSON

- Le code XMLHttpRequest:

```
var req = new XMLHttpRequest();
req.open("GET", "fichier.json", true);
req.onreadystatechange = monCode;
req.send(null);
```

- Le code JavaScript:

```
function monCode()
{ if (req.readyState == 4)
  { var doc = eval('(' + req.responseText + ')'); } }
```

- Utilisation des données:

```
var nomMenu = document.getElementById('jsmenu');
  // trouver un champ
nomMenu.value = doc.menu.value;
// assigner une valeur au champ
```

- Accéder aux données :

```
doc.commands[0].title // lire la valeur de "title" dans le tableau
doc.commands[0].action // lire la valeur de "action" dans le tableau
```

Qui utilise Ajax

- Les clients WEB de messagerie
  Gmail, Yahoo Mail, HotMail
- Google Maps
- FlickR, Picasa
- Deezer
- Youtube, Dailymotion
- Myspace, Facebook

Sécurité Ajax

- Ajax ne permet pas de faire des requêtes *cross-domain*
  les requêtes doivente être sur le domaine courant.
  Par exemple si l'adresse de votre page est *www.exemple.com/mapage.html*, vous pouvez faire une requête sur *www.exemple.com/toto.html* mais pas sur *www.sample.com/toto.html* ni même sur *sample.exemple.com/toto.html*.

Incovénients d'Ajax

- Si JavaScript est désactivé, Ajax ne peut fonctionner. Il faut demander à l'internaute de l'activer sur son navigateur.
- Les données chargées de façon dynamique ne font pas partie de la page. Elles ne sont donc pas prises en compte par les moteurs de recherche.
- L'aspect asynchrone fait que les modifications se font avec un délai (si le traitement sur le serveur est long), ce qui peut être déconcertant.
- Le bouton « Page précédente » du navigateur ne fonctionne pas sur les requêtes AJAX

Conclusions sur Ajax

- Combinaison des langages standards du WEB (Javascript, DOM HTML, XML)
- Grâce à l'objet XMLHttpRequest
- WEB dynamique « coté client »
- Utilisé par tous les sites « WEB 2.0 »
- Un outil à utiliser en attendant le déploiement du HTML5 (prévu pour 2010)