INF2005 – Programmation web

Ajax

Jacques Berger

Objectifs

Introduire le paradigme Ajax

Prérequis

HTML

Javascript

Javascript

Avec Javascript, le concept de fichier est différent d'un langage habituel

Le code est exécuté par le fureteur du client, nous n'accèderons pas à des fichiers locaux par le fureteur

Il faut extraire le fichier du serveur

Javascript

Pour charger un fichier, on doit donc faire une requête HTTP

Nécessite l'objet XMLHttpRequest

```
var requestObject = new XMLHttpRequest();
requestObject.open("GET", "document.xml", false);
requestObject.send();
var contenuTexte = requestObject.responseText;
```

XMLHttpRequest

Objet servant à

- Envoyer une requête HTTP
- Traiter la réponse HTTP
- Envoyer et recevoir des données du serveur

XMLHttpRequest

Méthode open

Paramètre #1 : méthode HTTP

La méthode HTTP à utiliser pour envoyer la requête, habituellement GET ou POST

XMLHttpRequest

Méthode open

Paramètre #2 : URL

L'URL de la requête (relative ou absolue)

Paramètre #3 : asynchrone

true pour envoyer la requête de façon asynchrone, false pour l'envoyer de façon synchrone

XMLHttpRequest

Méthode send
- Envoie la requête HTTP

Paramètre optionnel : Les données à envoyer au serveur si POST
- Données brutes
- XML
- autre

XMLHttpRequest

Propriété responseXML

Le résultat de la requête sous forme d'arbre DOM si le résultat est du XML

Propriété responseText

Le résultat textuel de la requête

Ajax

Ce n'est pas un langage, c'est un modèle de communication entre le fureteur et le serveur web

Asynchronous Javascript And Xml

Maintenant un nom commun

Ajax

Modèles

- Application web traditionnelle
- Application Ajax

Ajax

Application traditionnelle

Un appel au serveur pour chaque modification de la page
À chaque fois, le serveur génère la page au complet et le fureteur l'interprète au complet
Full Page Refresh

Ajax

Application Ajax

Lors du chargement initial, le serveur envoie la page web et l'application Javascript

Ensuite, le Javascript s'occupe de communiquer avec le serveur de façon asynchrone

Le Javascript met à jour la page dynamiquement lorsque le serveur répond

Ajax

Idées de base

- Éviter le full page refresh
- Améliorer la convivialité (fluidité) de la page web
- Diminuer la charge de travail du serveur web

Ajax

Les communications subséquentes entre le fureteur et le serveur peuvent être sérialisées de plusieurs façons :

- XML
- HTML
- JSON
- Données brutes

Ajax

Données brutes
- Peu de bande passante
- Peu de travail pour le serveur
- Facile à manipuler avec le Javascript
- À éviter pour les chaînes de caractères
  - Encodage non spécifié

Ajax

HTML
- Le serveur ne génère qu'un fragment HTML
- Ce fragment sera inséré au bon endroit
- Facile à manipuler avec Javascript
- Plus volumineux que les données brutes
- Méthode très populaire

Ajax

XML
- Volumineux
- Plus complexe à générer
- Facile à manipuler
- Méthode favorisée dans les débuts d'Ajax

Ajax

JSON
- Moins volumineux que XML
- Tend à remplacer XML

Ajax

Fonctionnement

On doit concevoir notre script du côté de serveur de façon à recevoir une requête HTTP et générer du contenu sérialisé (HTML, XML, JSON, brut)

Ajax

Fonctionnement

Le client doit communiquer avec la fonctionnalité Ajax du serveur

Le Javascript utilisera l'objet XMLHttpRequest

Ajax

La requête doit être envoyée au serveur de façon asynchrone pour éviter de bloquer l'utilisateur dans ses actions

Au préalable, on a enregistré une méthode à appeler lorsque le XMLHttpRequest recevra une réponse

Ajax

Enregistrement de la méthode et envoie de la requête

Propriété onreadystatechange

```
request.open("GET", url, true);
request.onreadystatechange = updatePage;
request.send();
```

Ajax

La méthode va être invoquée à chaque changement d'état de l'objet XMLHttpRequest

On doit obtenir le numéro de l'état qui nous intéresse

Ajax

0 : Requête non initialisée
1 : Connexion établie avec le serveur
2 : Requête reçue par le serveur
3 : Le serveur traite la requête
4 : La réponse est prête

Ajax

On doit vérifier l'état dans la fonction appelée

Propriété readyState

```
function updatePage() {
    if (request.readyState == 4) {
        var container = document.getElementById("time");
        var content = request.responseText;
        container.innerHTML = content;
    }
}
```

Fetch

Nouvelle fonctionnalité expérimentale dans les fureteurs : fetch

Appel Ajax simplifié utilisant les nouveautés du Javascript

- Promesses
- async/await

Fetch

Voir exemple

Un peu plus loin...

XMLHttpRequest

https://www.w3.org/TR/XMLHttpRequest/