Ajax ML

Asynchronous JavaScript

fabien.hermenier@unice.fr

? fonction synchrone

l'appel de la fonction est bloquant

1. appel de la fonction f()
2. on attend sagement la fin de l'exécution
3. on récupère le résultat

? fonction asynchrone

l'appel de la fonction est non-bloquant

1. appel de la fonction f(g)
   g est la fonction de **callback**
   (fonction à exécuter quand f() aura fini)
2. on continue l'exécution

? fonction asynchrone

l'appel de la fonction est non-bloquant

+

- pas d'attente inutile,
- très performant avec des entrées/sorties

−

- flot d'exécution non linéaire
- difficile à programmer /debugger

« asynchronous spaghetti code »

use-case @ facebook

1. chargement du mur (600 ko de HTML)
2. nouveau statut: *«une fin terrible pour HIMYM»*

synchrone VS asynchrone

3. Requête depuis un formulaire
4. On patiente durant le re-téléchargement du mur (600 ko de HTML).
5. re-affichage complet de la page

3. requête depuis du JavaScript
4. on continue de glander

@réponse du serveur (< 1ko)
-> mise à jour du mur

& prog. asynchrone Web

moins de latence
des applications plus réactives
moins de trafic réseau
moins de charge sur le serveur

XMLHttpRequest

buzz word en 2005

objet JavaScript permettant l'envoi asynchrone de requêtes HTTP sur le serveur

on peut envoyer n'importe quoi

```
var xhr = new XMLHttpRequest()

// La requête a envoyer
 // method: ‘POST’,’GET’,…
// url: l’URL du serveur destination
xhr.open(method,url)

// la méthode à exécuter à chaque fois
// que le statut de la requête change
xhr.onreadystatechange = function() {

   if (xhr.readyState == 4) { //La réponse a été reçu
      if (xhr.status == 200) {
         /* le serveur a répondu par un code 200,
            on affiche le contenu */
         console.log(xhr.responseText)
      } else {
         /*…*/
      }
   }
}

//envoie la requête (ici, sans contenu)
xhr.send()
```

Exemple de requête GET

(récupérer le sujet de projet ‘eswap’ sur le site du cours)

```
  var xhr = new XMLHttpRequest()

  xhr.open(‘GET’, ‘http://users.polytech.unice.fr/
~hermenie/adw/rest.php?query=subject&id=démineur')
xhr.onreadystatechange = function() {
   if (xhr.readyState == 4) {
      if (xhr.status == 200) {
         console.log(xhr.responseText)
      } else {
         console.log(‘Erreur. code=‘ + xhr.status)
      }
   }
}
xhr.send()
```

Exemple de requête POST

(s'authentifier sur le site du cours)

```
  var xhr = new XMLHttpRequest()

  xhr.open('POST', 'http://users.polytech.unice.fr/
~hermenie/adw/rest.php?query=auth')

//On envoie les données d'un formulaire
 xhr.setRequestHeader("Content-type", "application/x-
www-form-urlencoded");
xhr.onreadystatechange = function() {/*…*/}

//Le contenu du formulaire est dans le corps de la
// requête (principe de fonctionnement de POST)
xhr.send('username=' + l + '&password=' + p)
```

échanger des données avec du JavaScript avec JSON

une simple structure de données JavaScript dans une chaine de caractère

```
{
‘nom’:’hermenier’,
‘prenom’:’fabien’,
mails:[
   ‘fabien.hermenier@unice.fr’,
   ’fabien.hermenier@inria.fr’
   ]
}
```

JSON

`JSON.parse()` pour décoder
`JSON.stringify()` pour encoder

```
var message = "{‘type’ : ‘foo’,‘values’: [1,2,3]}"
var x = JSON.parse(message)
console.log(x.type + ‘ ‘ + x.values[0]);
//affiche « foo 1 »

var msg2 = JSON.stringify(x)
//msg2 == message
```

Debugger l'envoi de requête

`console.log()`

onglet '`network`' dans la console du navigateur pour afficher les requêtes envoyées

`postman`: un des plugins pour chrome pour executer des requêtes HTTP

> Ca vaut encore le coup de générer du contenu affichable côté serveur ? Non parce que JavaScript ca tabasse quand même et en plus on allège la charge du serveur !

Une tendance pour les (très très) gros sites

côté serveur

- envoie de pages statique
- interaction par une approche Ajax

côté client

- génération de toute l'interface
- interaction avec le serveur