Programmation Web Avancée

A J A X

(Asynchronous Javascript And XML)

Bibliographie

- Cours en ligne :
  - **Cours PWA, Virginie Sans, Univ de Rennes 1**
  - **Cours PWA AJAX, Thierry Hamon, Univ Paris 13**
  - **Cours en ligne programmation web, de Jérôme CUTRONA**
- Livres :
  - Premières applications Web 2.0 avec Ajax et PHP, Jean-Marie Defrance, 2006
  - jQuery, J. Chaffer et K. Swedberg, 2009
- Web
  - ….

AJAX
Introduction

- N’est pas une technologie spécifique et Innovante :
  - un regroupement de plusieurs technologies déjà existantes
- A pour objectif majeur l’optimisation des ressources sur Internet
  - les pages dynamiques classiques reposent entièrement sur des scripts coté serveur
    - nécessitent un rafraichissement après chaque requête passée au serveur Web
  - Le but d’AJAX est de soulager les serveurs en impliquant le client au processus du traitement (en utilisant JavaScript)
- Exemple d'application Web AJAX :
  - Google Mail, Maps, Earth, suggest …
  - Liste de suggestions automatiques
  - E-commerce
  - News
  - …

AJAX
Introduction

- AJAX utilise :
  - **Javascript:** code nécessaire pour **gérer les événements** coté client (navigateur), **envoyer les requêtes au serveur** et la **mise à jour de la page** si besoins
  - Les CSS : créer des effets spéciaux lors des mises à jour de données
  - (X)HTML
  - Le DOM : JS utilise le DOM pour avoir accès à des éléments de la page web
  - L'objet **XMLHttpRequest** de JavaScript : permet d' effectuer des requêtes sur le serveur sans interrompre les autres tâches du navigateur
  - Les formats de données : **XML**, des fichiers **textes** ou **JSON** utilisés pour les transferts entre le serveur et le client
- L'intérêt pour Ajax d'utiliser ces différentes technologies est qu'elles sont déjà intégrées dans la plupart des navigateurs actuels

AJAX
Introduction

- AJAX utilise un modèle de programmation comprenant :
  - les **évènements** : actions qui provoquent l'appel des fonctions associées aux éléments de la page(formulaires, boutons ou toute autre objet)
  - la **présentation** : les fonctions JavaScript identifient les éléments de la page grâce au DOM et communiquent avec le serveur par l'objet XMLHttpRequest

AJAX
Introduction

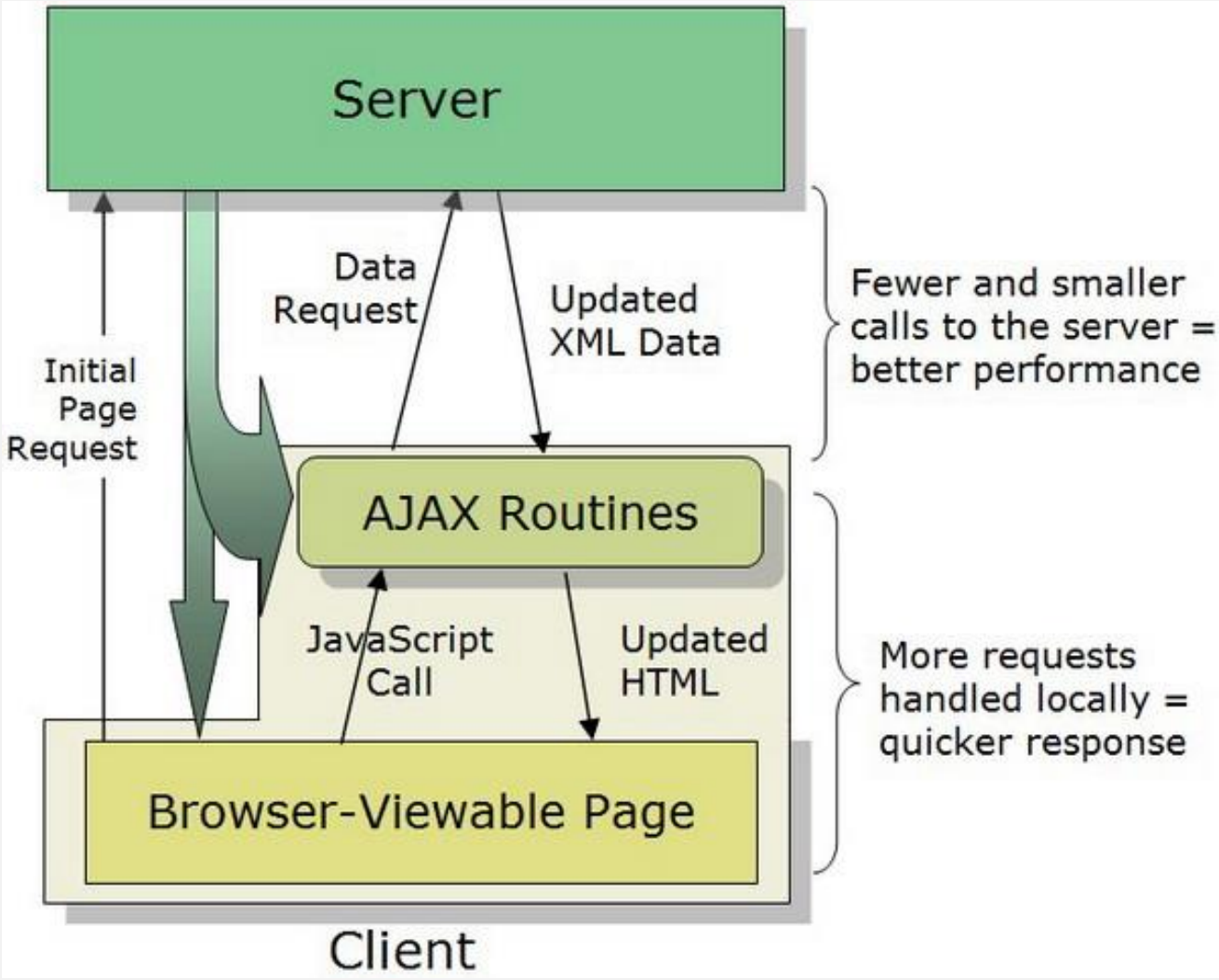
- Deux composants (Application Web Classique) :
  - Serveur (implémentation JAVA, PHP, CGI, …)
    - Contrôle général de l'application
    - Propose des ressources statiques : Modèle du document, bibliothèque de scripts, feuilles de style
    - Traitement dynamique des données
    - Composition dynamique de l'interface
  - Client (implémentation Javascript par exemple)
    - Gestion des événements utilisateur
    - Composition dynamique de l'interface
  - Dialogue :
    - HTTP, (X)HTML

AJAX
Introduction

- Deux composants (Application Web AJAX) :
  - Serveur (implémentation JAVA ou PHP par exemple)
    - Contrôle général de l'application
    - Propose des ressources statiques : Modèle du document, bibliothèque de scripts, feuilles de style
    - Traitement dynamique des données
    - Composition dynamique de l'interface
  - Client (implémentation Javascript par exemple)
    - **Contrôle délégués en fonction des événements**
    - Gestion des événements utilisateur
    - Composition dynamique de l'interface
    - **Traitement des données reçues**
  - Dialogue :
    - **HTTP, XML, JSON**

AJAX
Principe de fonctionnement

- Schéma



XMLHttpRequest : permet d'effectuer des requêtes sur le serveur

AJAX
Méthode de l'objet XMLHttpRequest

- Utilisation de 2 méthodes pour l'envoi
  - **open() :** permet d'ouvrir le fichier désiré sur le serveur. Il possède trois paramètres obligatoires
    - Méthode utilisée pour l'envoi de la requête : GET ou POST
    - URL: Chemin relatif ou absolu du fichier sur le serveur
    - Valeur booléenne (True ou False) : indiquant si la requête doit être envoyee de manière asynchrone ou non
  - **send()** : permet d'envoyer les données au serveur sous forme d'une chaîne de caractères qui respecte la structure suivante:
    - Méthode POST : parametre1=valeur1&parametre=valeur2&...
    - Méthode GET : chaîne « null »
  - **setRequestHeader():** permet de définir le codage des données passées au serveur. Elle est utilisée avec la méthode POST.
    - La syntaxe complète est la suivante:
      - setRequestHeader("Content-Type","application/x-www-form-urlencoded")
  - **abort():** permet d'annule la requête en cours

AJAX
Propriétés de l'objet XMLHttpRequest

- XMLHttpRequest permet d'interagir avec le serveur, grâce à ses méthodes et ses attributs
  - **readyState** : renvoie un entier compris entre 0 et 4 indiquant l'état de transfert entre client et serveur:
    - A chaque changement d'état, la fonction associée à **onreadystatechange** est exécutée
    - Valeurs possibles :

ETAT	Description
0	Requête non initialisée
1	Connexion établie(requête en cours d'envoie)
2	Requête reçue par le serveur
3	Réponse en cours de transfert vers le client
4	**Les données sont disponibles chez le client**

L'état qui nous intéresse est l'état **'4'** : les données qui seront traités par le navigateur (scripts JavaScript) sont disponible chez le client.
Tout autre état (de 0 à 3) sont traduits par une attente.

AJAX
Propriétés de l'objet XMLHttpRequest

- XMLHttpRequest permet d'interagir avec le serveur, grâce à ses méthodes et ses attributs
  - **Status**: renvoie un entier qui permet de savoir si l'accès au document se passe comme il faut
    - Quelques valeurs spéciales
      - 200 : OK, page trouvée sur le serveur
      - 404 : page non trouvée sur le serveur
      - 500 : problème de fonctionnement interne du serveur
  - **Onreadystatechange:** attribut ou événement activée à chaque fois qu'un changement d'état (changement de la valeur de l'attribut **readyState**) est détecté
  - **responseXML:** retourne un objet DOM du XML renvoyée par le serveur
  - **responseText:** retourne une chaîne de caractères contenant les données chargées que l'on peut traiter avec JavaScript sur le navigateur (client)

AJAX
Séquence…

- Séquence des événements qui se produisent lors de l'utilisation d'AJAX
  1. Un événement se produit au niveau navigateur
  2. JavaScript crée et configure un nouvel objet XMLHttpRequest
  3. JavaScript envoie la requête au serveur
  4. Le serveur reçoit et traite l'objet XMLHttpRequest
  5. Le serveur renvoie la réponse au navigateur
  6. La fonction appelante reçoit et traite la réponse du serveur
  7. Mise à jour des informations sur la page web

Il faut instancier un objet XMLHttpRequest pour chaque fichier que vous voulez charger

AJAX
Demo : mode synchrone

```
function RecupText() {
/*------------------Config et envoi de la requête SYNCHRONE : */
//création d'une requête uniquement pour Firefox
objetXHR = new XMLHttpRequest();
//Config. requête GET et Synchrone
objetXHR.open("get","Message.txt", false);
//envoi de la requête
objetXHR.send(null);
/*------------------------------Attente du retour SYNCHRONE : */
//récupération et affectation du texte renvoyé par le serveur à la zone résultat
if( objetXHR.readyState==4) {
var MessageTXT = objetXHR.responseText;
document.getElementById("resultat").innerHTML=MessageTXT;
}
}
```

AJAX
Demo : mode Asynchrone TXT

```
function RecupText() {
/*------------------Config et envoi de la requête SYNCHRONE : */
//création d'une requête uniquement pour Firefox
objetXHR = new XMLHttpRequest();
//Config. requête GET et Asynchrone
objetXHR.open("get","Message.txt", True);



//envoi de la requête
objetXHR.send(null);}
```

AJAX
Demo : mode Asynchrone TXT

```
function RecupText() {
/*------------------Config et envoi de la requête SYNCHRONE : */
//création d'une requête uniquement pour Firefox
objetXHR = new XMLHttpRequest();
//Config. requête GET et Asynchrone
objetXHR.open("get","Message.txt", True);
// Désignation de la fonction de rappel
objetXHR.onreadystatechange = actualiserPage();
```

```
function actualiserPage() {
if(objetXHR.readyState==4){
//récupération et Affecte du résultat renvoyé par le serveur à la zone résultat
document.getElementById("resultat").innerHTML= objetXHR.responseText;
}
```

AJAX

Création d'objet XMLHttpRequest

- Fonction permettant de vérifier si le navigateur supporte l'objet XMLHTTPRequest

```
function creationXHR() {
var resultat=null;
//test pour les navigateurs : Mozilla, Opera...
try {      resultat= new XMLHttpRequest(); }
```

```
catch (Error) {
//test pour les navigateurs Internet Explorer > 5.0
try {resultat= new ActiveXObject("Msxml2.XMLHTTP"); }
```

```
catch (Error) {
//test pour le navigateur Internet Explorer 5.0
try {resultat= new ActiveXObject("Microsoft.XMLHTTP"); }
```

```
catch (Error) {
resultat= null;
}}}
return resultat;
}
```

AJAX
Chargement asynchrone avec un paramètre **GET**

```
//instanciation de l'objet XMLHttpRequest
var objetXHR = creationXHR();
//Désignation de la fonction de rappel
objetXHR.onreadystatechange = actualiserPage();
var numero=2; //simulation du choix d'un numéro d'identifiant
objetXHR.open("get","script.php?id="+numero,true);
//envoi de la requête
objetXHR.send(null);
```

```
//Déclaration de la fonction de rappel
function actualiserPage() {
if(objetXHR.readyState==4) {
if(objetXHR.status==200) {
var resultat = objetXHR.responseText ;
//####Le résultat peut maintenant être inséré dans la page HTML
}else {
alert("Erreur HTTP N°"+ objetXHR.status); }}}
```

AJAX
Chargement asynchrone avec un paramètre **POST**

```
//instanciation de l'objet XMLHttpRequest
var objetXHR = creationXHR();
//Désignation de la fonction de rappel
objetXHR.onreadystatechange = actualiserPage;
objetXHR.open("post","script.php",true);
//Affectation du type d'encodage de la requête envoyée
objetXHR.setRequestHeader("Content-Type","application/x-www-form-urlencoded")
var numero=2; //simulation du choix d'un numéro d'identifiant
//envoi de la requête
objetXHR.send(id=numero);
```

```
//Déclaration de la fonction de rappel
function actualiserPage() {
if(objetXHR.readyState==4) {
if(objetXHR.status==200) {
var resultat = objetXHR.responseText ;
//####Le résultat peut maintenant être inséré dans la page HTML
}else {
alert("Erreur HTTP N°"+ objetXHR.status); }}}
```

TP 1 :

- Réaliser une application web permettant de créer et de signer une pétition
- Le formulaire pour la signature d'une pétition se compose de :
  - ID : identifiant de la pétition: Nous considérons que le id = 1.
  - Nom
  - Prénom
  - Email
  - Pays
  - Bouton (envoyer)
- Partie 1 : (mode synchrone)
  - Réaliser le formulaire *index.html* qui permet de signer une pétition
  - Créer une base de données permettant de stocker ces signatures
  - Créer une page PHP qui, lorsqu'elle est appelée par la page index.html :
    - ajoute la nouvelle signature dans la BD
    - affiche OK ou NotOK en fonction du succès de l'opération
- Partie 2 : Avec AJAX
  - Ajouter une zone de texte dans le formulaire permettant d'afficher les 10 dernières signatures
  - Utiliser l'objet XMLHttpRequest pour que la mise à jour de la liste se fasse de façon asynchrone

Programmation Web Avancée

A J A X

(principe de fonctionnement

la suite)

Echange de données : client / serveur

Coté client
Mise à jour des données (JS – DOM)

- **Fonctions DOM:**
  - document.getElementById("resultat")**.innerHTML**=resultat;
    - ne respecte pas la structuration des spécifications du DOM
  - manipulation des nœuds de l'arbre DOM
    - appendChild(N) : fonction js permettant d'ajouter le nœud N à la fin d'un élément DOM
    - removeChild(N) : fonction js permettant de retirer le nœud N de l'arbre DOM

Coté client
Mise à jour des données (JS – DOM)

```
<body>
<ul id="ListePetition">
  <li>Petition1</li>
  <li>Petition2</li>
</ul>
```

```
<script>
function ajouterElement(resultat) {// resultat = objetXHR.responseText
      var node = document.createElement("LI");
      var textnode = document.createTextNode("resultat");
      node.appendChild(textnode);
      document.getElementById("ListePetition").appendChild(node);
}
</script>
```

Coté client
Mise à jour des données (JS – DOM)

Création et l'ajout d'un paragraphe <p> au document html:

```
// Create a <p> element
var para = document.createElement("P");
// Create a text node
var t = document.createTextNode("This is a paragraph");
 // Append the text to <p>
para.appendChild(t);
 // Append <p> to <body>
document.body.appendChild(para);
```

Coté client

removeChild()

```
<body>
<ul id="ListePetition">
  <li>Petition1</li>
  <li>Petition2</li>
</ul>
```

Supprimer un élément

```
<script>
function supprimerElement() {
    var list = document.getElementById("ListePetition");
    list.removeChild(list.childNodes[0]); //supprime le 1er élément
} </script>
```

Supprimer tous les élémen

```
<script>
function supprimerContenu(ListePetition) {
if (ListePetition!= null) {
while(ListePetition.firstChild)
element.removeChild(ListePetition.firstChild);
}}</script>
```

Coté client
Envoie des données : méthode GET

```
var objetXHR = creationXHR();

var numero=2; //numero=document.getElementById("ID")

objetXHR.open("GET","script.php?id="+numero,true);

objetXHR.onreadystatechange = actualiserPage();

objetXHR.send(null);
```

Récupération des données : coté serveur
Nous utilisons les variables :
$_GET ou $_REQUEST

Coté client
Envoie des données : méthode POST

```
var objetXHR = creationXHR();

var numero=2; //numero=document.getElementById("ID")

objetXHR.open(« POST","script.php",true);

objetXHR.onreadystatechange = actualiserPage;

objetXHR.send(id=numero);
```

Récupération des données : coté serveur

Nous utilisons les variables :

$_POST ou $_REQUEST

Coté client
Envoie des données : format XML

- Client vers le serveur
  - La méthode post:

```
// création du document XML
var parametresXml =
"<signataire>"
        "<nom>" + document.getElementById("nom").value+ "</nom>" +
        "<prenom>" + document.getElementById("prenom ").value+
"</prenom>" //+
//….
"</signataire>";
```

```
objetXHR.setRequestHeader("Content-Type","text/xml");
```

```
objetXHR.send(parametresXml); //envoi de la requête
```

Coté client
Envoie des données : format XML

```
function FonctionJS() {
objetXHR = creationXHR();
var parametresXml =
"<signataire>" +
        "<nom>" + document.getElementById("nom").value + "</nom>" +
        "<prenom>" + document.getElementById("prenom ").value + "</prenom>"
"</signataire>";
objetXHR.open("post","monProgram.php", true);
objetXHR.onreadystatechange = actualiserPage();
objetXHR.setRequestHeader("Content-Type", "text/xml");
//envoi de la requête
objetXHR.send(parametresXml);
}
```

```
function actualiserPage() {
        // à définir ……
}
```

Coté serveur
Récupération des données : format XML

- Récupération du document XML
  - $parametresXml = file_get_contents('php://input');

- Création d'un objet XML
  - $objetSimpleXML = simplexml_load_string($parametresXml);

- Récupération des éléments du document XML
  - $nom = $objetSimpleXML->nom;
  - $prenom = $objetSimpleXML->prenom;

Définition de plusieurs niveaux hiérarchique est possible

http://www.w3schools.com/php

Coté serveur
Récupération des données : format XML

```
//récupération des paramètres au format XML
$parametresXml = file_get_contents('php://input');

//création d'un objet Simple XML à partir des paramètres récupérés
$objetSimpleXML=simplexml_load_string($parametresXml);

//récupération du nom
$nom=$objetSimpleXML->nom;

//récupération du prénom
$prenom=$objetSimpleXML->prenom;

//… insertion des données dans la BD
//….

$resultat=' OK ou NOT OK '
//envoi de la réponse au navigateur
echo $resultat
```

AJAX et XML n’est pas recommandé

AJAX JSON
(AJAJ)

JavaScript Object Notation (JSON)

- JSON est un format d'échange de données basé sur du texte
  - Natif en Javascript (faisant partie de la norme JavaScript)
  - Format alternatif à XML

Exemple d'un tableau en JSON :

```
{"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]}
```

Même exemple en XML:

```
<employees>
   <employee>
      <firstName>John</firstName> <lastName>Doe</lastName>
   </employee>
   <employee>
      <firstName>Anna</firstName> <lastName>Smith</lastName>
   </employee>
   <employee>
      <firstName>Peter</firstName> <lastName>Jones</lastName>
   </employee>
</employees>
```

http://www.w3schools.com/json/default.asp

JavaScript Object Notation (JSON)

- Accès aux données :
  - employees[0].firstName
  - employees[0].lastName
  - employees[0]["firstName"]
- Changer les données :
  - employees[0].firstName = "Gilbert";

Exemple d'un tableau en JSON :

```
{"employees":[
   {"firstName":"John", "lastName":"Doe"}
   {"firstName":"Anna", "lastName":"Smith
   {"firstName":"Peter", "lastName":"Jones
]}
```

Un nom et une valeur

```
"firstName":"John"
```

Une liste des noms/valeur

```
{"firstName":"John", "lastName":"Doe"}
```

```
var employees = [
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter","lastName": "Jones"}
];
```

Coté client
Envoie des données : JSON

```
function FonctionJS() {
    objetXHR = creationXHR();

    // création un objet JavaScript pour : (1) ajouter et transférer les données sur le serveur
    var objetJS = new Object();

    // récupération des valeurs des champs
    objetJS.nom = document.getElementById("nom");
    objetJS.prenom = document.getElementById ("prenom");

    //encodage en JSON
    var parametres =objetJS.toJSONString();

    objetXHR.open("post","monProg.php", true);
    objetXHR.onreadystatechange = actualiserPage();
    objetXHR.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

    objetXHR.send(parametres);//envoi de la requête
}
```

Ajouter sur la page index,html

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

Or json2.js

Coté serveur
Récupération des données : JSON

```
//récupération des paramètres au format XML
$parametresJSON = file_get_contents('php://input');
//bibliothèque externe JSON-PHP
require_once('JSON.php');
//
$objetJSON = new Services_JSON();
//décodage de données
$Signataire = $objetJSON->decode($parametresJSON);
//récupération des données
$nom=$Signataire->nom;
$prenom=$Signataire->prenom;

// $resultat = exécution de la requete
// $resultat : doit respecter la syntaxe de JSON

echo $resultat ;
```

```
header("Content-Type: application/json; charset=UTF-8");
```

Fonction actualiserPage() : cas d'une réponse html

```
//mise en forme HTML du résultat
$resultat='<span id="nom">'.$nom.'</span>    
<span id="prenom">'.$prenom.' </span>

//envoi de la réponse au navigateur
echo $resultat;
```

```
function actualiserPage() {
if (objetXHR.readyState == 4) {//test si le résultat est disponible
if (objetXHR.status == 200) {

var nouveauResultat = objetXHR.responseText;//récup du résultat

document.getElementById("info").innerHTML=nouveauResultat;

}
}
}
```

Fonction actualiserPage() : cas XML

```
//mise en forme XML du résultat
$resultat='<?xml version="1.0" encoding="utf-8"?>';
$resultat.= "<resultats><nom>".$nom. "</nom><prenom>".$prenom. "</prenom></resultats>";
//envoi de la réponse au navigateur
echo $resultat;
```

```
function actualiserPage() {
if (objetXHR.readyState == 4) {//teste si le résultat est disponible
if (objetXHR.status == 200) {
//récup du résultat dans un arbre XML
var ResultatXML = objetXHR.responseXML;
//création d'un pointeur racine du résultat
var racineResultats = ResultatXML.firstChild;
//récup des valeurs des éléments nom et prenom dans l'arbre
var nom=racineResultats.childNodes[0].firstChild.nodeValue;
var prenom=racineResultats.childNodes[1].firstChild.nodeValue;
//actualisation des résultats
document.getElementById("nom").innerHTML=nom;
document.getElementById("prenom").innerHTML=prenom;
}}}
```

Fonction actualiserPage() : cas JSON sans bibliothèque externe

```
$resultat='{"resultats":{"nom":" '.$nom.'","prenom":'.$prenom.'}}';
echo $resultat;
```

Elaboration de format JSON coté serveur

Recuperation des donnees avec la methode eval()

```
function actualiserPage() {
if (objetXHR.readyState == 4) {
if (objetXHR.status == 200) {

var resultat = objetXHR.responseText;
var objetJSON=eval('('+resultat+')');

// lecture de la valeur " nom " dans le tableau
var nom=objetJSON.resultats.nom;
// lecture de la valeur " prenom " dans le tableau
var prenom=objetJSON.resultats.prenom;
//actualisation des résultats
document.getElementById("nom").innerHTML=nom;
document.getElementById("prenom").innerHTML=prenom;
}}}
```

Fonction eval() présente des problèmes de sécurité

Fonction actualiserPage() :

cas JSON avec bibliothèque externe : json.js & JSON.PHP

```
require_once('JSON.php');
$tableauSignataire=array(
        "nom"=>$nom,
        "prenom"=>$prenom);
$tableauResultat =array("resultats"=>$TableauSignataire);

$objetJSON = new Services_JSON();

$resultat = $objetJSON->encode($tableauResultat);
echo $resultat;
```

```
…
var nouveauResultat = objetXHR.responseText;
var objetJSON=nouveauResultat.parseJSON();
var nom=objetJSON.resultats.nom;
var prenom=objetJSON.resultats.prenom;

document.getElementById("nom").innerHTML=nom;
document.getElementById("prenom").innerHTML=prenom;
}}}
```

TP1 : la suite

- Ajouter les fonctionnalités suivantes :
  - Interface pour l'ajout d'une nouvelle pétition
  - Notification de l'ajout d'une nouvelle pétition aux navigateurs connectés
  - Afficher en temps réel la pétition qui a le plus de signature

Travail à rendre pour le lundi 15 mai