AJAX

Nathalie Hernandez / Fabien Amarger

PLAN

- **AJAX - PRINCIPE**
- **L'objet XMLHttpRequest**
  - Envoyer une requête asynchrone
  - Envoyer et Recevoir une réponse du serveur
    - En texte
    - En JSON

WEB 2.0 – web d'aujourd'hui

- **Une version du Web aux mains des internautes**
  - Le contenu n’est plus défini uniquement par les experts du Web, mais aussi par l’internaute qui le veut (blog, réseaux sociaux par exemple)
  - La mise en forme s'adapte au média (téléphone, lecteur d’écran) et aux préférences de l’utilisateur (options du navigateur)
  - **Le Web n’a plus vocation à être un entrepôt d’information mais un espace collaboratif favorisant le partage et l’échange (texte, image, document)**

Ce qu’il faut pour y arriver

- **Des technologies pour rattraper la convivialité et l'interactivité des applications « classiques »**
  - rafraîchissement de parties de page,
  - personnalisation à la volée de l’interface,
  - glisser déposer,
  - complétion automatique, …
- **Des standards pour mettre un peu d’ordre et avancer dans une direction commune**

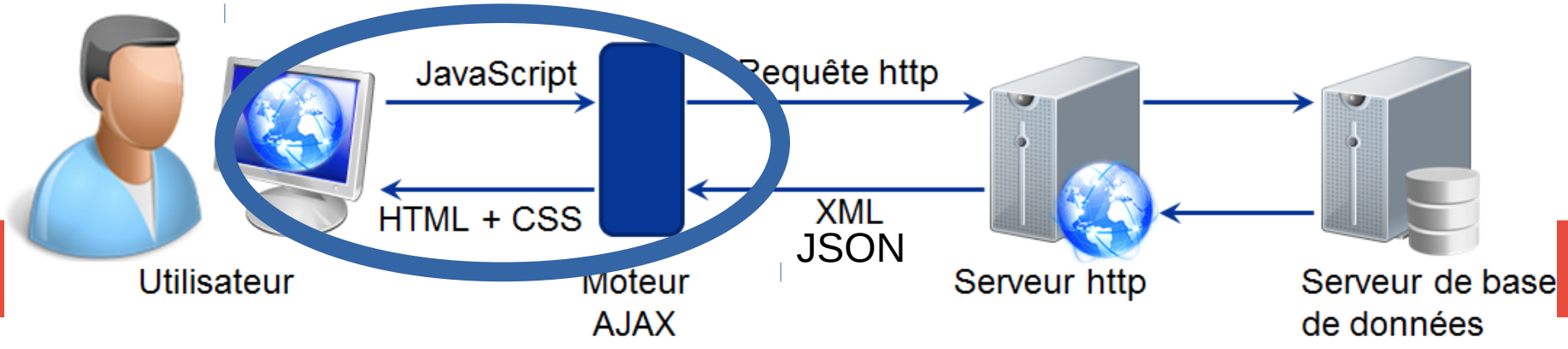
Les technologies

- **Coté client : front-end**
  - HTML 5 pour le contenu et la structuration de l'information
  - CSS 3 pour la présentation
  - DOM pour la représentation en mémoire et la manipulation d'une page
  - JavaScript pour le comportement de l'application côté client

- **Coté serveur : back-end**
  - Un langage dédié (PhP, node.js, JEE)
  - Un SGBD

ET AJAX dans tout ça !!

- **AJAX (Asynchronous JavaScript and XML) pour des interactions asynchrones entre le client et le serveur**
- **AJAX est une conjonction de technologies**
  - HTML
  - CSS
  - DOM
  - Echange et manipulation de données (XML/JSON) via des requêtes HTTP
  - Récupération asynchrone de données
  - JavaScript reliant le tout



Le Web AVEC AJAX

Points importants de l'architecture :

- **Les composants d'une application Web (coté client) peuvent**
  - EFFECTUER DES REQUÊTES SUR LE SERVEUR
  - OBTENIR DES INFORMATIONS
  - METTRE À JOUR LA PAGE AFFICHÉE EN MODIFIANT LE DOM

    => PLUS BESOIN DE RAFRAÎCHIR COMPLÈTEMENT LA PAGE

- **Asynchronisme**

  =>LES REQUÊTES ENVOYÉES AU SERVEUR NE BLOQUENT PAS LE NAVIGATEUR

- **Généralisation : tous les événements interceptés par le navigateur (clics, survol, …) peuvent déclencher une requête asynchrone**

Application WEB AVANT AJAX

navigateur

serveur

connexion

Page HTML

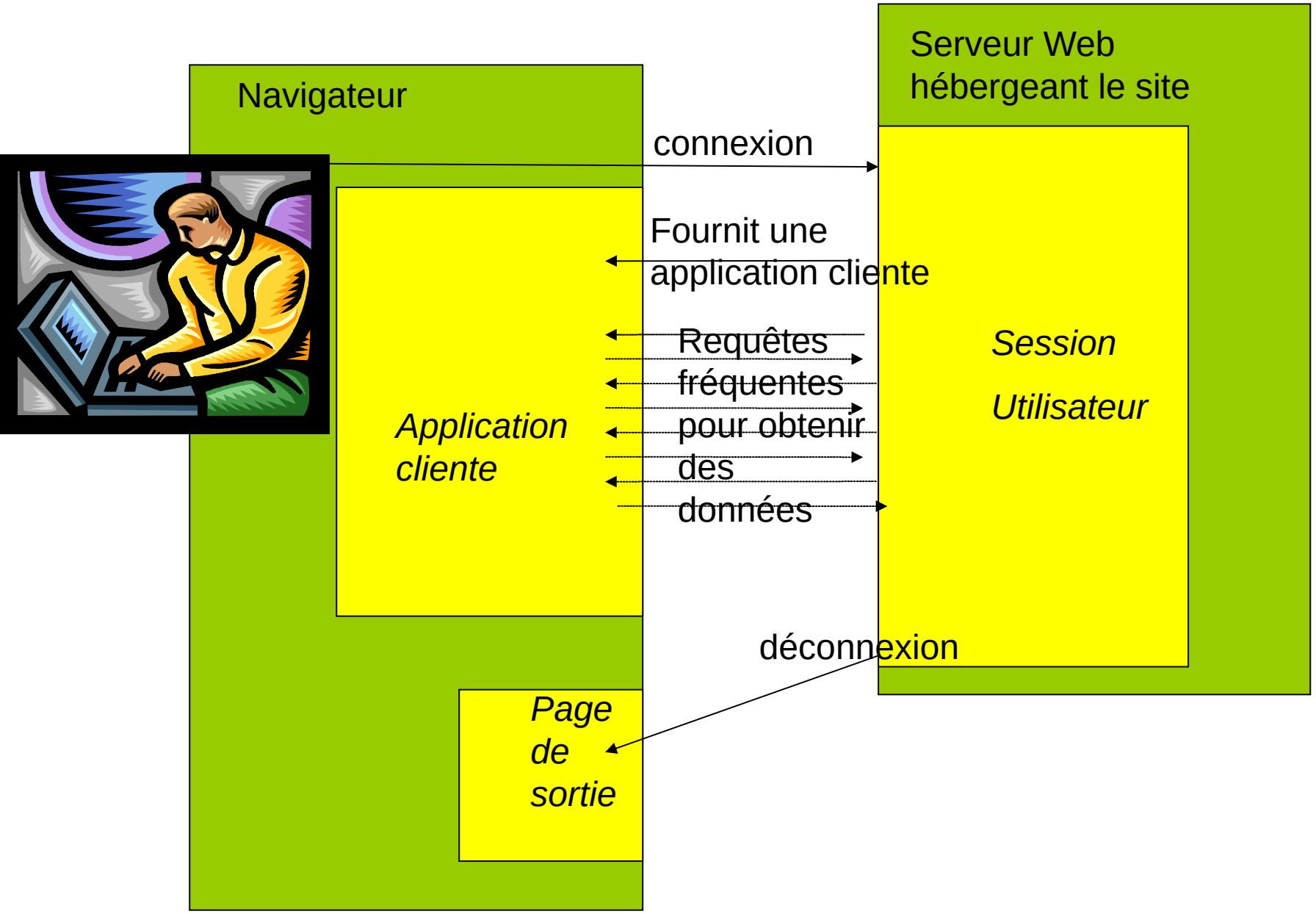
Page HTML

Session Utilisateur

déconnexion

Page de sortie

Application WEB AVEC AJAX



Application WEB AVEC AJAX

Navigateur

Serveur Web hébergeant le site

connexion

Fournit une application cliente

Requêtes fréquentes pour obtenir des données

Application cliente

Session Utilisateur

déconnexion

Page de sortie

Web service REST

Requêtes pour obtenir des données

Premiers Pas avec AJAX

- **Les requêtes asynchrones sont gérées avec l'objet JS XMLHttpRequest**

- **Principe de cet objet :**
  - Faire transiter des données (par forcément en XML…) entre le client et le serveur via **une requête HTTP**

- **Historique :**
  - Objet javascript créé par Microsoft adopté par Mozilla
  - Extension non standard du DOM supportée par la majorité des navigateurs (W3C Working Draft 26 May 2014)

Créer une instance de l'objet XMLHttpRequest

Pour créer le requeteur :

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

Transmettre une requête

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

/*ouvre la liaison avec le serveur*/

```
xhr.open(HttpMethod, url, type_connex);
```

/* mafonction est exécutée à chaque changement d'état de la réponse*/

```
xhr.onreadystatechange=mafonction;
```

/*envoie la requête avec d'éventuels parametres*/

```
xhr.send(params);
```

Ouverture de la liaison avec le serveur

```
xhr.open(HttpMethod, url, type_connex);
```

- **HttpMethod :**
  - GET : DEMANDE DE DONNÉES AU SERVEUR (EVENTUELS PARAM. APRÈS L'URL)
  - POST : TRANSMETTRE DES DONNÉES AU SERVEUR (DONNÉES DÉFINIES DANS MÉTHODE SEND)
  - HEAD (EN-TÊTE DES FICHIERS, EX : DERNIÈRE MODIFICATION DE FICHIERS),
  - DELETE / PUT
- **url : url du script effectuant le traitement**
- **type_connex :**
  - TRUE : ASYNCHRONE
  - FALSE : SYNCHRONE // DÉPRÉCIÉ

Traiter la réponse

```
xhr.onreadystatechange=mafonction;
```

- **Méthodes utilisables dans la fonction associée aux changements de statut de la réponse**
  - Récupérer l'entête de la réponse

```
xhr.getResponseHeader()
```

  - Récupérer les données renvoyées par le serveur au format texte

```
xhr.responseText;
```

  - Récupérer les données renvoyées par le serveur au format XML

```
xhr.responseXML;
```

(renvoie un document XML manipulable à partir des interfaces du DOM tels que getElementById(), childNodes,…)

Informations sur la réponse

- **Dans la fonction associée à la réception de la réponse, on peut surveiller l'état de la requête :** `xhr.readyState;`

Plusieurs valeurs :

- 0 : UNINITIALIZED (LIAISON PAS ENCORE FAITE)
- 1 : LOADIND (OBJET INITIALISÉ (OPEN) MAIS PAS ENVOYÉ)
- 2 : LOADED (REQUÊTE ENVOYÉE, MAIS DE RÉPONSE)
- 3 : INTERACTIVE (RÉPONSE EN COURS DE RÉCEPTION)
- 4 : COMPLETED (REQUÊTE ACHEVÉE)

- **Le résultat de la requête contient également un statut HTTP :** `xhr.status`

le code 200 indique que tout s'est bien passé

Envoi de la requête

```
xhr.send(params);
```

params

null : généralement si la requête est de type GET/ HEADER

Sinon : avec la méthode POST n'importe quel type de données dont on peut préciser le type avec la méthode

```
xhr.setRequestHeader("Content-Type", type MIME du contenu);
```

Un exemple pour récapituler

```
var req = XMLHttpRequest();



req.open("GET","quickstart.php?name=Hernandez", "true");

req.onreadystatechange  = function (){
    if (req.readyState == 4 && req.status == 200) {
        alert(req.responseText);
    }
}

req.send(null); // pas de parametre
```

Exemple de requête HEAD

Recupérer la date de dernière modification d'une page

```
var req = XMLHttpRequest() ;
 req.open("HEAD", "/faq/index.html",true);
  req.onreadystatechange=function() {
   if (req.readyState==4) {
    alert("File was last modified on - "+
     req.getResponseHeader("Last-Modified"))
   }
  }
  req.send(null)
```

Exemple d'entête HTTP associées à une ressource :

HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Thu, 04 Apr 2002 11:34:01 GMT
Content-Type: text/html
Last-Modified: Thu, 14 Mar 2002…
Content-Length: 32282

Le format JSON

- **JSON est un format léger d'échange de données(RFC 4627 (2006))**
- **JSON se base sur deux structures :**
  - Une **collection de couple** clé : valeur caractérisant un **objet**

    { clé1 : valeur1**,** clé2 : valeur2**, ...}**
  - **Une valeur pouvant être un objet ou une liste ordonnée d'objet**

    { clé : **[**

    { clé1a : valeur1a**,** clé2a : valeur2a**, ...}**

    { clé1b : valeur1b**,** clé2b : valeur2b**, ...}**

    ]

    }

JSON / example

```
{books:[
 {title:"AJAX et PHP : Comment construire des applications web réactives »,
  isbn:"978-2100506842"},
 {title:"Pratique de MySQL et PHP »,
  isbn:"978-2100523368"},
 {title:"Tout sur le web 2.0 et 3.0, 2e édition »,
  isbn:"978-2100543427 »}
]}
```

JSON et JS

- JSON est
  - un format texte complètement indépendant de tout langage
  - JSON plus concis que XML (utilisation de moins de bande passante)
- **JSON est directement interprétable en JS**

Les objets initialisés

Évite de faire un appel au constructeur d'Object

```
var obj = {
    property_1:   value_1,   // value_# may be an number...
    property_2:   value_2, // or a string...
                           //   or a Object,
                          // or an array

    property_n: value_n }; //

var myHonda = {color: "red", wheels: 4, engine: {cylinders: 4, size: 2.2}, owners : [{name:"Nathalie", gender:"f" },{name:"Ollivier", gender:"m" } ]};
```

JSON et JS

- **JSON est directement interprétable en JS**

```
var myJSON={"books":[
 {"title":"AJAX et PHP : Comment construire des applications web
   réactives »,
  isbn:"978-2100506842"},
 {"title":"Pratique de MySQL et PHP »,
  "isbn:"978-2100523368"},
 {"title":"Tout sur le web 2.0 et 3.0, 2e édition »,
 "isbn":"978-2100543427 »}
]}

for (var i=0; i<myJSON.books.length; i++)
      console.log (myJSON.books[i].title +  ", " + myJSON.books[i].isbn );
```

JSON Object – objet natif de JS

- **Intégré dans la spécification ECMAScript 5.1 (juin 2011) supportés par la majorité des navigateurs actuels**

```
JSON.parse(montexte)
```

=>Interprète une chaîne de caractères « montexte » comme du JSON, et produisant l'objet javascript correspondant => plus sûre car n'interprète pas les scripts si y en a

```
JSON.stringify(monObj)
```

=> Retourne une chaîne de caractères JSON correspondant à l'Objet spécifié

JSON et XMLHTTPRequest – reception de données

```
var req = XMLHttpRequest() ;
...
req.onreadystatechange = function () {
var jsonResponse = JSON.parse(req.responseText);
  // Parcourir la table de hachage.
  for (var i=0; i<jsonResponse.books.length; i++)
    console.log(jsonResponse.books[i].title +  ", " +
jsonResponse.books[i].isbn + "<br />");
}
```

→ **avec json accès direct aux couples, plus direct que le parcour du document xml à partir du Dom**

→ **Attention à l'utilisation de la fonction eval en JS => elle exécute le code**

JSON et XMLHTTPRequest – envoi de données

```
var myJSON={"books":[
 {"title":"AJAX et PHP : Comment construire des applications web
réactives »,
  isbn:"978-2100506842"},
 {"title":"Pratique de MySQL et PHP ",
  "isbn:"978-2100523368"},
 {"title":"Tout sur le web 2.0 et 3.0, 2e édition ",
 "isbn":"978-2100543427"}
]}
//Que l'on peut envoyer par HTTPRequest en post
var req=XMLHttpRequest() ;
req.open("POST", "/scripts/SaveBooks.php", true);
req.setRequestHeader("Content-Type", "application/json");
req.send("param="+JSON.stringify(myJSON));
```

Exemple complet

Réaliser une application qui affiche l’heure donnée par le serveur et qui en attendant le chargement affiche une image

Page.html

```
<!doctype html>
<html lang="fr">
<head>
<title> Un vrai exemple </title>
<link href="style.css" rel="stylesheet" type="text/css" />
<script src="ajax_functions.js" type="text/javascript"></script>
</head>
<body>
<div>
  <h1>Demo</h1>
   <p>Placer la souris dans la zone ci-dessous pour obtenir l'heure</p>
   <div id="showtime" class="displaybox"></div>
</div>
</body>
</html>
```

Version en JSON

serveur.php

```
<?php
header('Content-Type: application/json'); // Indiquer que le contenu
de la sortie est de type JSON.
// Créer le tableau de la réponse.
$chaineheure="It is ".date('H:i:s')." on ".date('M d, Y').".
$response = array(
 "clock" => array(
"timestring" => $chaineheure));
echo json_encode($response);  // Encoder le tableau au format JSON.
?>
```

ajax_function.js

```
function getServerTime() {
  var myReq=getRequester();
  myReq.open("GET", "servertime.php", true);
  myReq.onreadystatechange = traiter_reponse
  myReq.send(null);
}

function initEvenements() {
document.getElementById('showtime').addEventListerner('mouseover',getSer
verTime);
}
Window.addEventListener('load',initEvenements);
```

La même version en JSON

```
function traiter_reponse() {
   if (myReq.readyState == 4) {
    if(myReq.status == 200) {
responseJSON = JSON.parse(xmlHttp.responseText);
var texte = document.createTextNode(responseJSON.clock.timestring) ;
var noeudPres=document.getElementById('showtime').firstChild;
document.getElementById('showtime').replaceChild(texte,noeudPres);
    }
  } else { // la réponse n'est pas encore arrivée
 if (document.getElementById('showtime').firstChild == null) {
         var img = document.createElement('img');
         img.setAttribute('src','ajax-loader.gif');
         document.getElementById('showtime').appendChild(img);
     }  }
  }
```

AJAX pour intérogger des API WEB

- **AJAX permet d'interroger des API WEB (web service rest)**
- **Le site web peut être enrichi avec des données / traitements effectués par des tiers**
- **Exemple open data Toulouse : Accès à l'agenda des manifestations culturelles de la ville mis à jour quotidennement**

  https://data.toulouse-metropole.fr/explore/dataset/agenda-des-manifestations-culturelles-so-toulouse/api/

  => analyser le format JSON fourni dans le corps de la réponse HTTP

Ajax_fonction.js

```
function getManif(){
 var req = new XMLHttpRequest();
 req.open("GET", "https://data.toulouse-metropole.fr/api/records/1.0/search/?dataset=agenda-des-
manifestations-culturelles-so-toulouse");
 req.onreadystatechange = function (){
  if (req.readyState == 4 && req.status == 200) {
   var Data = JSON.parse(req.responseText);
   var lieu ="";
   Data.records.forEach(function(y) {
         lieu+=y.fields.nom_de_la_manifestation + " ";
         });
    var texte = document.createTextNode(lieu);
  if ( !document.getElementById('showliste').firstChild)
        document.getElementById("showliste").appendChild(texte);
  else
document.getElementById("showliste").replaceChild(texte,document.getElementById('showliste').firstChild)
;
   }
  }
  req.send(null);
}
```