Université de Msila

Faculté des mathématiques et de l'informatique

Département d'informatique

3ième année Licence ISIL

Introduction à AJAX

(Asynchronous JavaScript And XML)

Présenté par : Meliouh.A

2018-2019

AJAX ?

- N'est PAS une technologie
- N'est PAS un logiciel
- N'est PAS un greffon (plug-in)
- C'est l'utilisation conjointe de :
  - HTML
  - CSS
  - DOM / JavaScript
  - XMLHttpRequest (JavaScript)
  - XML (ou JSON)

AJAX ?

- AJAX : **A**synchronous **J**avaScript **A**nd **X**ML

  (JavaScript asynchrone et XML)

  - JavaScript : langage de script côté client (navigateur)
  - Asynchrone : par rapport au chargement de la page Web et des portions de page Web
  - XML : langage à balises permettant, entre autre, de structurer des données

- Permet, grâce à JavaScript, la récupération de données XML (mais aussi texte ou JSON) disponibles sur un serveur Web

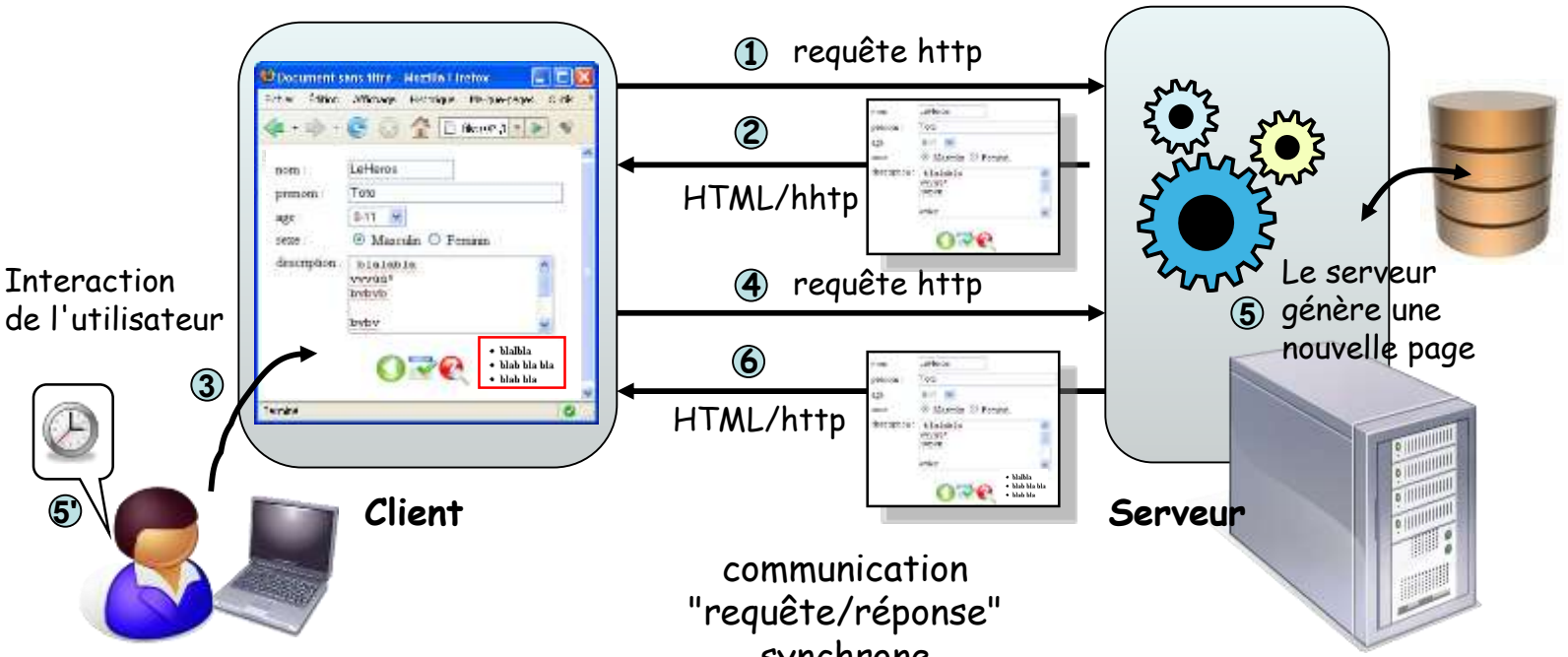
AJAX ?

Asynchronous JavaScript And XML

- Un système de communication asynchrone remplace le classique modèle requête/résponse HTTP synchrone.
  - L'utilisateur continue à utiliser l'application pendant que le programme client requête des informations au serveur en tâche de fond !
  - Séparation de l'affichage et de la récupération des données.
- Ajax permet de modifier partiellement la page affichée par le navigateur pour la mettre à jour sans avoir à recharger la page entière.

Fonctionnement classique du Web :

- Navigateur outil générique d'affichage : il n'a aucune intelligence de l'application
- Logique de navigation sous forme d'enchaînement de pages est déterminée par le serveur.



Limites des applications Web "Classiques"

- Contrainte par HTML
  - Ensemble limité de widgets
- Pas de retour immédiat aux activités de l'utilisateur
  - L'utilisateur doit attendre la page suivante générée par le serveur
- Interruption des activités de l'utilisateur
  - L'utilisateur ne peut effectuer d'autres opérations pendant qu'il attend une réponse
- Perte du contexte opérationnel suite au rafraîchissement
  - Perte de la position de scrolling dans la page
  - Le cerveau doit réanalyser entièrement toute nouvelle page

Remédier aux limites du Web "Classique"

- Animation des écrans prise en charge du côté client
  - Assurée par du code exécuté sur le navigateur
    - → limite les échanges navigateur/serveur web
    - → possibilité d'augmenter l'interactivité et de réaliser des comportements ergonomiques plus évolués
- Optimisation des échanges navigateur/serveur
  - **Communication asynchrone**
    - Lorsqu'une requête est émise par le client, celui-ci reprend la main immédiatement
  - Echange des données plutôt que de la présentation **(une fois la page initiale chargée)**

→Technologies RIA (Rich Internet Application)

→Web 2.0 versus Web 1.0

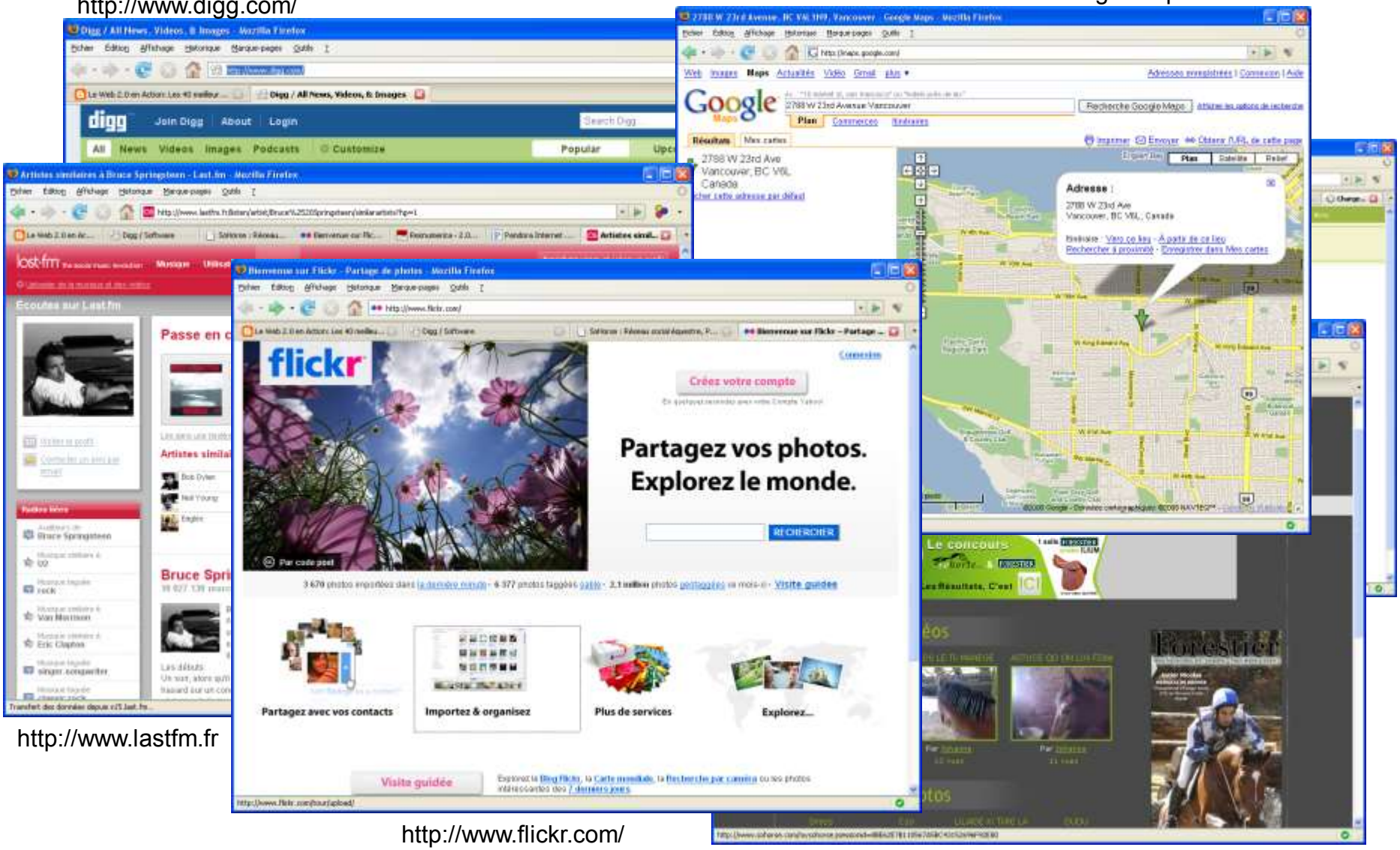
RIA c'est quoi ?

- C'est reproduire ou du moins s'approcher de l'expérience utilisateur des applications Desktop, dans une application web.
- Le programme répond rapidement et intuitivement.
- Feedback quasi instantané.
  - Une cellule dans un tableur change de couleur quand on passe la souris dessus.
  - Un password est validé à chaque touche tapée.
- Les choses se passent naturellement.
  - Pas besoin de cliquer sur un bouton pour déclencher un événement.

Exemples de sites web 2.0

http://www.digg.com/

Google maps



http://www.lastfm.fr

http://www.flickr.com/

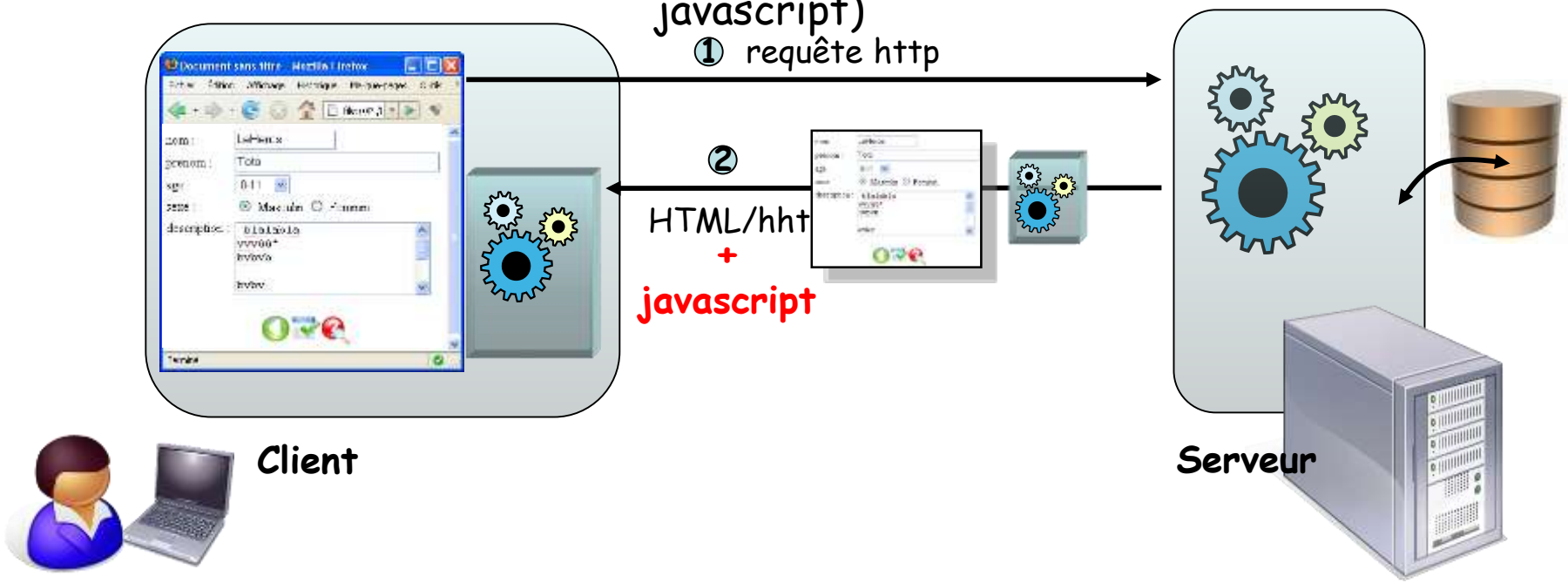
AJAX sur le web

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

Application Web AJAX

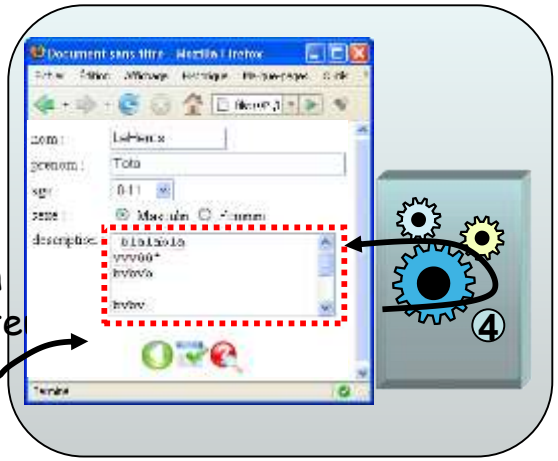
- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur

1er échange similaire au web "classique" : le serveur envoie une page au client mais en y embarquant de l'intelligence (code javascript)



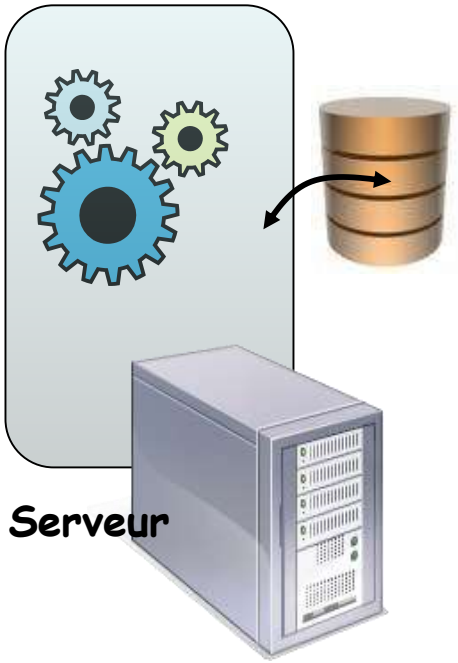
Application Web AJAX

- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur



Client

Certaines requêtes de l'utilisateur sont traitées localement par le navigateur grâce à la couche d'intelligence qui accompagne la présentation

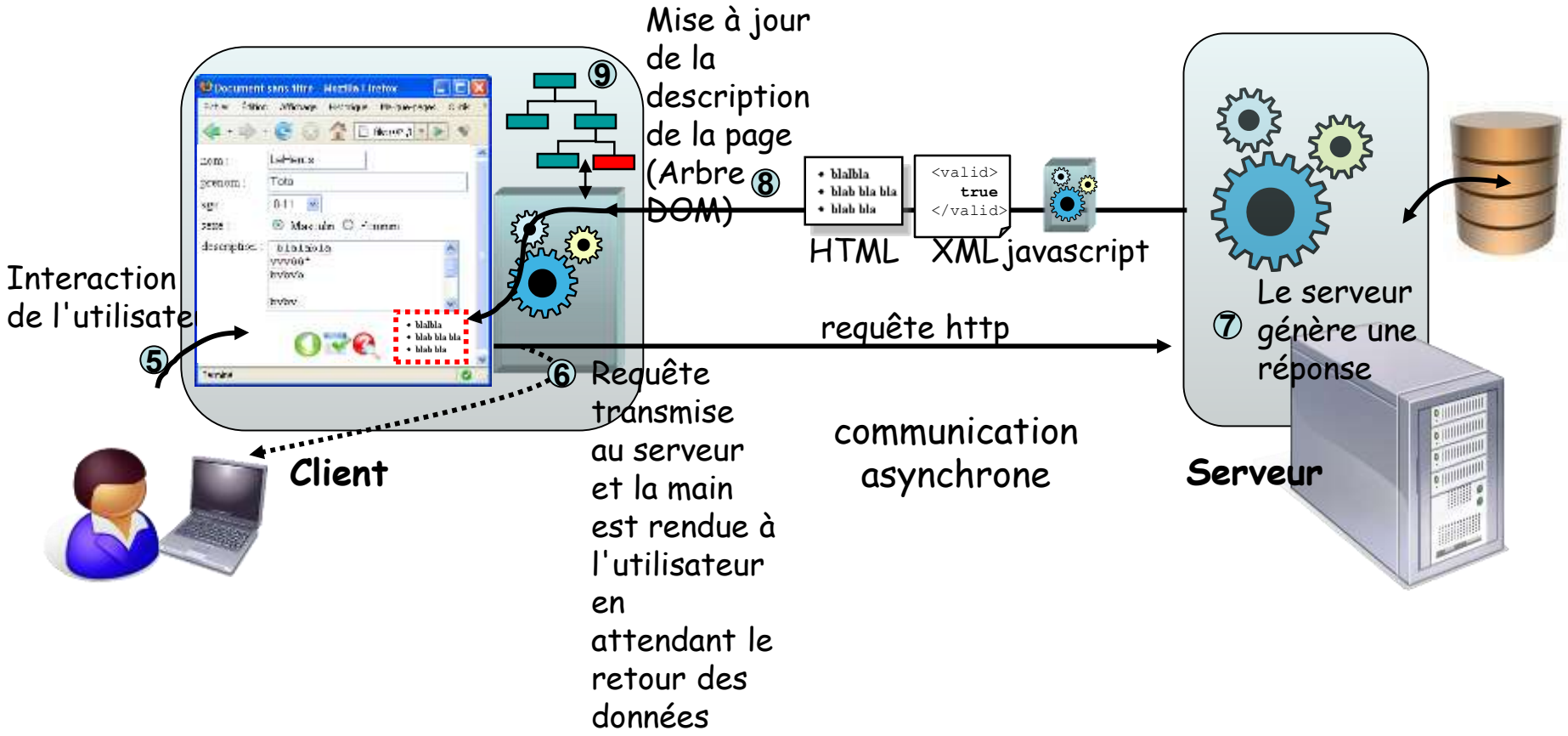


Serveur

Application Web AJAX

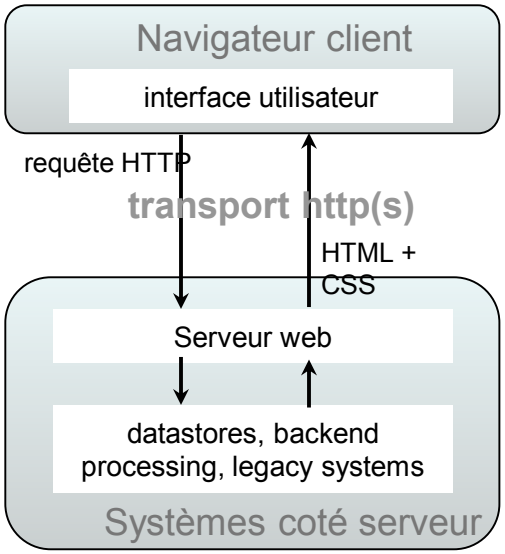
- Une partie de l'intelligence fonctionnelle de l'application est déportée vers le navigateur

D'autres requêtes nécessitent l'interrogation du serveur

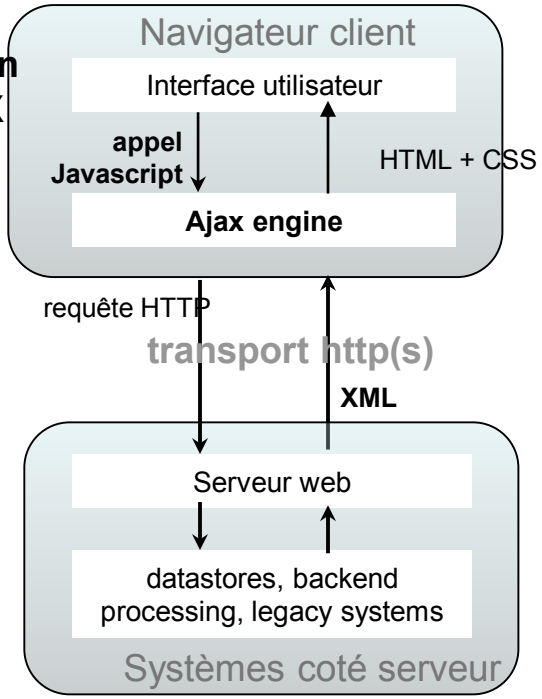


Modèles des applications WEB

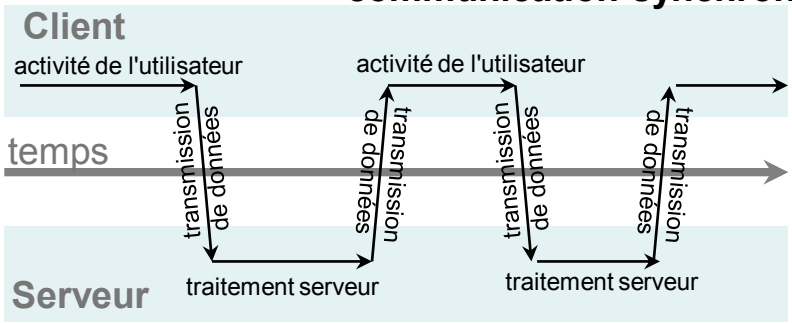
Application Web "classique"



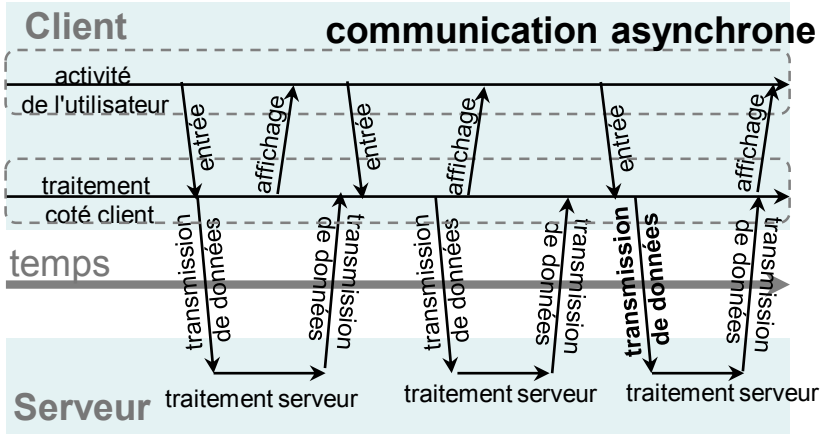
Application Web AJAX



communication synchrone



communication asynchrone



Technologies utilisées en AJAX

AJAX = DHTML (DOM + CSS + Javascript) + XmlhttpRequest

- **Javascript**
  - Langage de script orienté objet et faiblement typé
  - Fonctions javascript invoquées lorsque intervient un événement sur la page
  - "Glue" pour tout le fonctionnement d'AJAX
- **DOM (Document Object Model)**
  - API pour accéder à des documents structurés
  - Représente la structure de documents XML et HTML
- **CSS (Cascading Style Sheets)**
  - Permet un séparation claire du contenu et de la forme de la présentation
  - Peut être modifié par le code Javascript
- **XmlhttpRequest**
  - Objet Javascript qui assure une interaction asynchrone avec le serveur

XMLHttpRequest

- Objet JavaScript
  - En fait, objets JavaScript
  - Microsoft :
    - ActiveXObject("Msxml2.XMLHTTP")
    - ActiveXObject("Microsoft.XMLHTTP")
  - Les autres :
    - XMLHttpRequest()
- Supporté par tous les navigateurs modernes
  - Mozilla™, Firefox, Safari, et Opera, y compri sur Smartphones,
- Communique avec le serveur via des GET/POST HTTP standards,
- L'objet XMLHttpRequest est utilisé en tâche de fond pour s'occuper des communications asynchrones
  - Pas d'interruption des interactions de l'utilisateur avec l'application

Attributs de l'objet XMLHttpRequest

Attributs	Valeurs possibles
readyState	0: non initialisé. 1: connexion établie. 2: requête reçue. 3: réponse en cours. 4: terminé.
Status	200 est ok 404 si la page n'est pas trouvée.
responseText	contient les données chargées dans une chaîne de caractères.
responseXml	contient les données chargées sous forme xml, les méthodes de DOM servent à les extraire.
Onreadystatechange	propriété activée par un évènement de changement d'état. On lui assigne une fonction.

Méthodes de l'objet XMLHttpRequest

Méthodes	Paramètres
open(*method, url, async*)	Indique le type de demande *method* : le type de demande: GET ou POST *url* : le serveur (fichier) emplacement *async* : true (asynchrone) ou false (synchrone)
send()	Envoie la requête au serveur (utilisé pour GET)
send(*string*)	Envoie la requête au serveur (utilisé pour POST)
abort	Abandonne la requête HTTP.

Utilisation de `XMLHTTPRequest`

1. Créer un l'objet
2. Envoyez une demande à un serveur
   - Méthode, URL
   - Asynchrone ?
3. Associer une fonction au traitement du résultat de la requête
4. Traiter le résultat
   - Texte ?
   - XML / JSON ?

Utilisation de `XMLHTTPRequest`

1. Créer un objet XMLHttpRequest

- Tous les navigateurs modernes (Chrome, IE7 +, Firefox, Safari, et Opera) ont un objet intégré **XMLHttpRequest**.
- Syntaxe pour créer un objet XMLHttpRequest:
  - *variable* = new XMLHttpRequest();
- Les anciennes versions d'Internet Explorer (IE5 et IE6) utilisent un objet ActiveX:
  - variable = new ActiveXObject("Microsoft.XMLHTTP");

Utilisation de XMLHTTPRequest

1. Créer un objet XMLHttpRequest

- Pour gérer tous les navigateurs, y compris IE5 et IE6, vérifier si le navigateur prend en charge l'objet XMLHttpRequest. Si elle le fait, créer un objet **XMLHttpRequest**, sinon, créer un **ActiveXObject**:

- Exemple

```
var xhttp;
if (window.XMLHttpRequest) {
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Utilisation de `XMLHTTPRequest`

2- Envoyez une demande à un serveur

Pour envoyer une demande à un serveur, nous utilisons les méthodes **open**() et **send**() de l'objet **XMLHttpRequest**::

- Exemple

```
xhttp.open("GET", "ajax_info.txt", true);

xhttp.send();
```

Utilisation de XMLHTTPRequest

2- Envoyez une demande à un serveur

- Dans l'exemple ci-dessus, vous pouvez obtenir un résultat en cache. Pour éviter cela, ajouter un ID unique à URL:

```
xhttp.open("GET","teste_get.php?t="+Math.random(), true);
```

- Si vous voulez envoyer des informations à la méthode GET, ajouter les informations à l'URL:

```
xhttp.open("GET",
"teste_get2.php?fname=Mohammed&lname=YAHIA" , true);
```

- Exemple avec POST

```
xhttp.open("POST", "ajax_test.php", true);
xhttp.send("fname=Mohammed&lname=YAHIA");
```

Utilisation de `XMLHTTPRequest`

2- Envoyez une demande à un serveur

GET ou POSTE ?

- GET est plus simple et plus rapide que POST , et peut être utilisé dans la plupart des cas.
- Cependant, toujours utiliser une requête POST lorsque:
  - Un fichier mis en cache n'est pas une option (mise à jour d'un fichier ou base de données sur le serveur).
  - Envoi d' une grande quantité de données sur le serveur (POST n'a aucune limitation de taille).
  - Envoi d' entrée de l' utilisateur (qui peut contenir des caractères inconnus), POST est plus robuste et plus sûre que GET .

Utilisation de `XMLHTTPRequest`

2- Envoyez une demande à un serveur

Asynchronous - True ou False?

- Les requêtes au serveur doivent être envoyées de manière **asynchrone**. (la valeur true pour le troisième paramètre dans la méthode open() de l'objet `XMLHTTPRequest`).
- Avec l'envoi de requêtes **asynchrones**, JavaScript n'a pas à attendre la réponse du serveur, mais peut à la place: :
  - Exécuter d'autres scripts en attendant la réponse du serveur.
  - Faire face à la réponse lorsque la réponse prête.
- L'utilisation de requêtes **synchrones n'est pas recommandée**, JavaScript ne va pas continuer à exécuter, jusqu'à ce que la réponse du serveur est prêt. Si le serveur est occupé ou lent, l'application se bloque ou s'arrête.

Utilisation de `XMLHTTPRequest`

3- Associer une fonction au traitement du résultat de la requête

- Quand une requête est envoyée à un serveur, nous voulons effectuer certaines actions en fonction de la réponse..
- Chaque changement d'état de la requête engendre un événement
- L'état de la requête est reflété par l'état de l'objet **`XMLHTTPRequest`** : propriété **`readyState`**
  - 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

Utilisation de XMLHTTPRequest

3- Associer une fonction au traitement du résultat de la requête

- L'événement onreadystatechange est déclenché cinq fois (0-4), une fois pour chaque changement de readyState.
- Onreadystatechange enregistre une fonction (ou le nom d'une fonction) à appeler automatiquement.
- Lorsque readyState est 4 et Status est 200, la réponse est prête:

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
  if(xhttp.readyState == 4 && xhttp.status == 200) {
    document.getElementById("demo").innerHTML =
xhttp.responseText;
  }
};
```

Utilisation de XMLHTTPRequest

3- Associer une fonction au traitement du résultat de la requête

```
function loadDoc() {
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = traiter;
}

function traiter () {
  if(xhttp.readyState == 4 && xhttp.status == 200) {
    document.getElementById("demo").innerHTML =
xhttp.responseText;
  }
};
```

Utilisation de `XMLHTTPRequest`

4- Traiter le résultat de la requête

- Le résultat peut se présenter sous 3 formes :
  - Sous forme de texte
    - texte brut
    - texte contenant du code HTML
    - **`XMLHTTPRequest.responseText`**
  - Sous forme d'un objet DOM XML
    - **`XMLHTTPRequest.responseXML`**
    - Serveur : **`Content-Type: text/xml`**
  - Sous forme de données JSON
    - texte contenant du code JSON
    - **`eval(XMLHTTPRequest.responseText)`**
- Le traitement consiste généralement en une modification de la page Web en utilisant JavaScript, le DOM et les CSS.

Utilisation de XMLHTTPRequest

4- Traiter le résultat de la requête

- Exploiter des données au format texte
- Exemples
- Texte brut

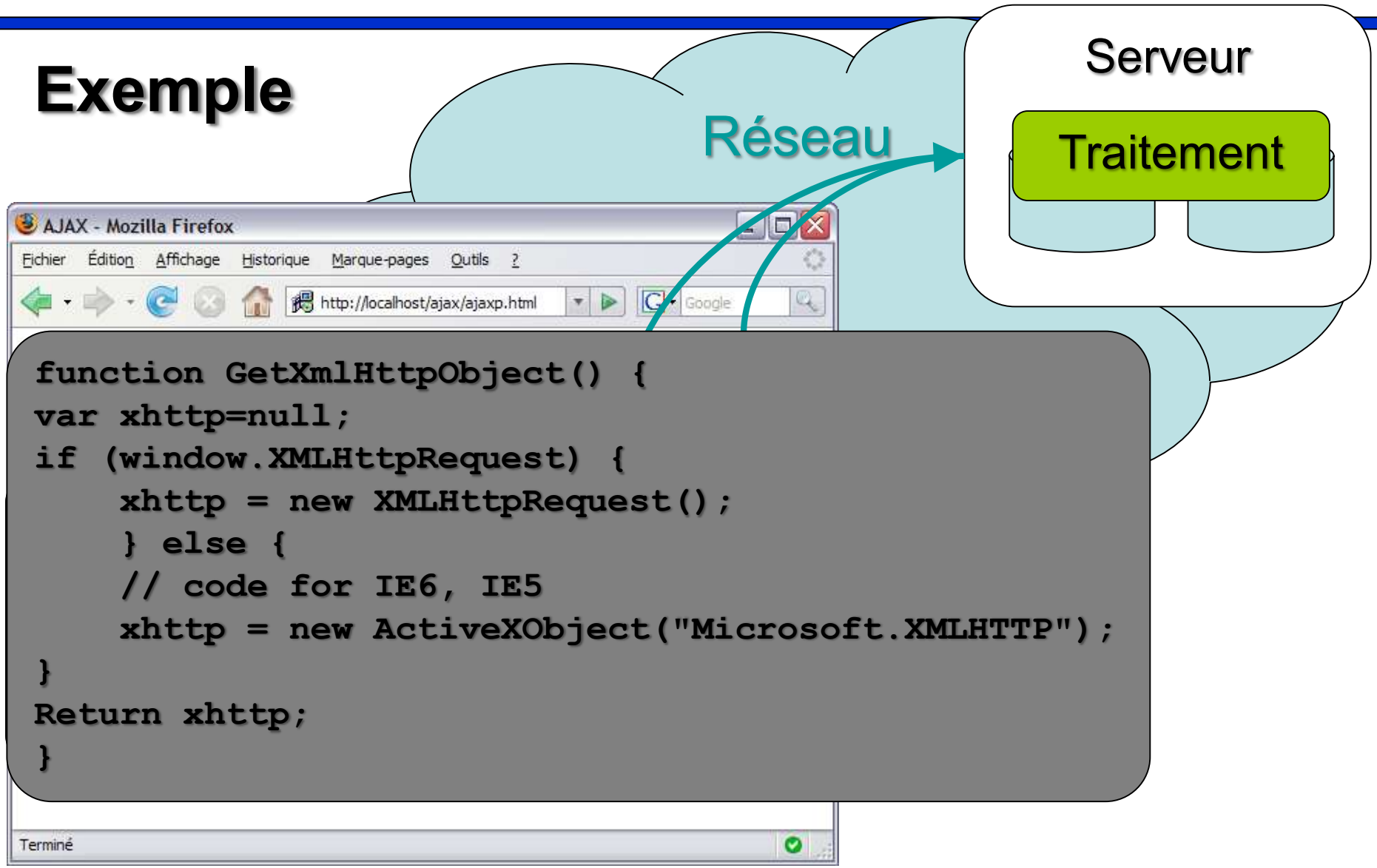
```
document.mon_formulaire.saisie.value
= xmlHttp.responseText
```

- Texte contenant du code HTML

```
document.getElementById('txt').innerHTML
= xmlHttp.responseText
```

Exploiter des données au format Texte

Exemple




```
function GetXmlHttpObject() {
var xhttp=null;
if (window.XMLHttpRequest) {
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
Return xhttp;
}
```

Exploiter des données au format XML

Exploiter des données au format XML

Rappel XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Recette SYSTEM "Recette.dtd">
<Recette>
  <Titre>Tarte aux pommes</Titre>
  <Auteur> <Nom>Mohammed</Nom>
           <Prenom>Yahia</Prenom>
  </Auteur>
</Recette>
```

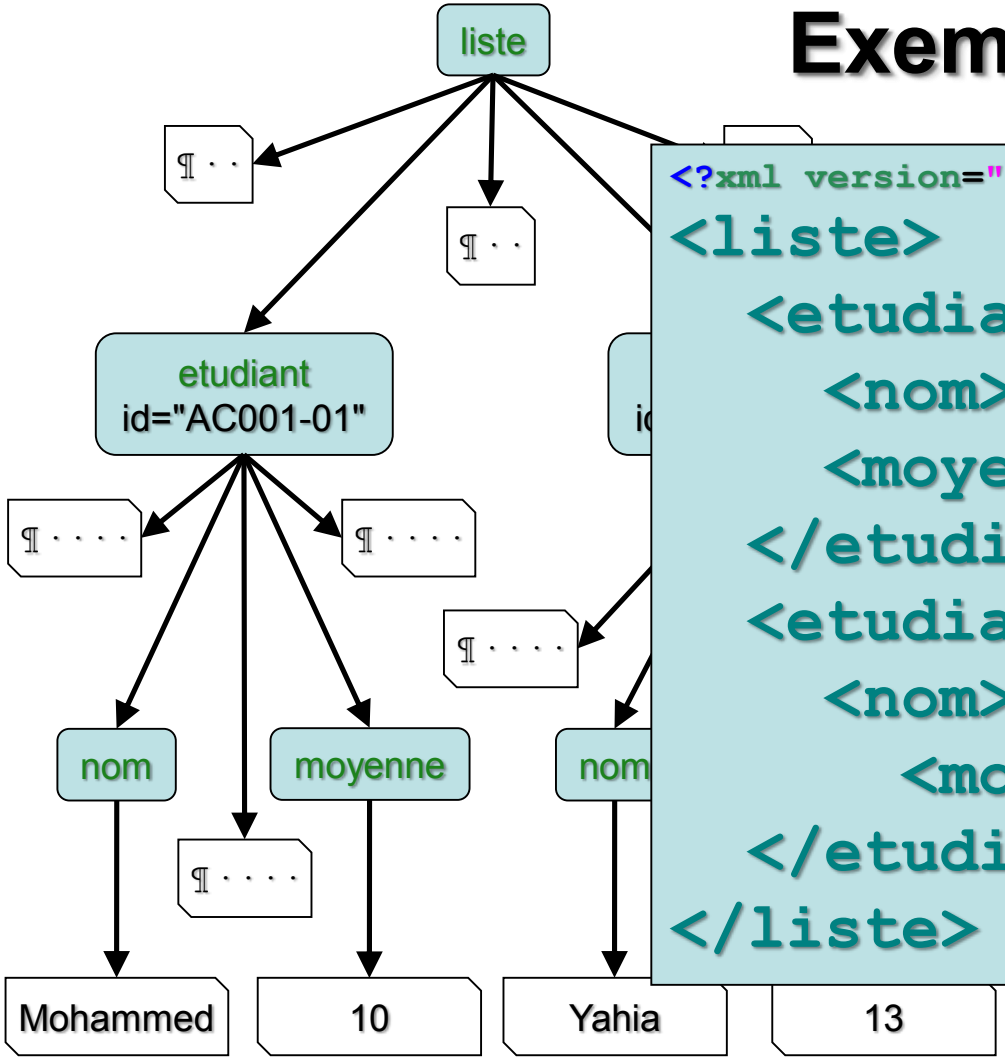
- Ça ressemble à du HTML où :
  - on utilise ses propres balises
  - la syntaxe est rigoureuse
- Ce document ne peut pas s'afficher directement
- On utilise des d'outils pour manipuler l'information de ce document

Exploiter des données au format XML

- **Objet XML** :
  - `objXML= XMLHTTPRequest.responseXML`
  - `objXML.getElementsByTagName(n)`
    → Collection de nœuds
- **Collection de nœuds :**
  - `length` → nombre d'éléments
  - `[x]` → accès au x^éme^ élément
- **Nœud** :
  - `firstChild` → Premier fils
  - `childNodes` → Collection de fils
  - `hasChildNodes()` → Possède des fils ?
  - `nodeValue` → Valeur du nœud
  - `nodeName` → Nom du nœud
  - `getAttribute(a)` → Valeur de l'attribut

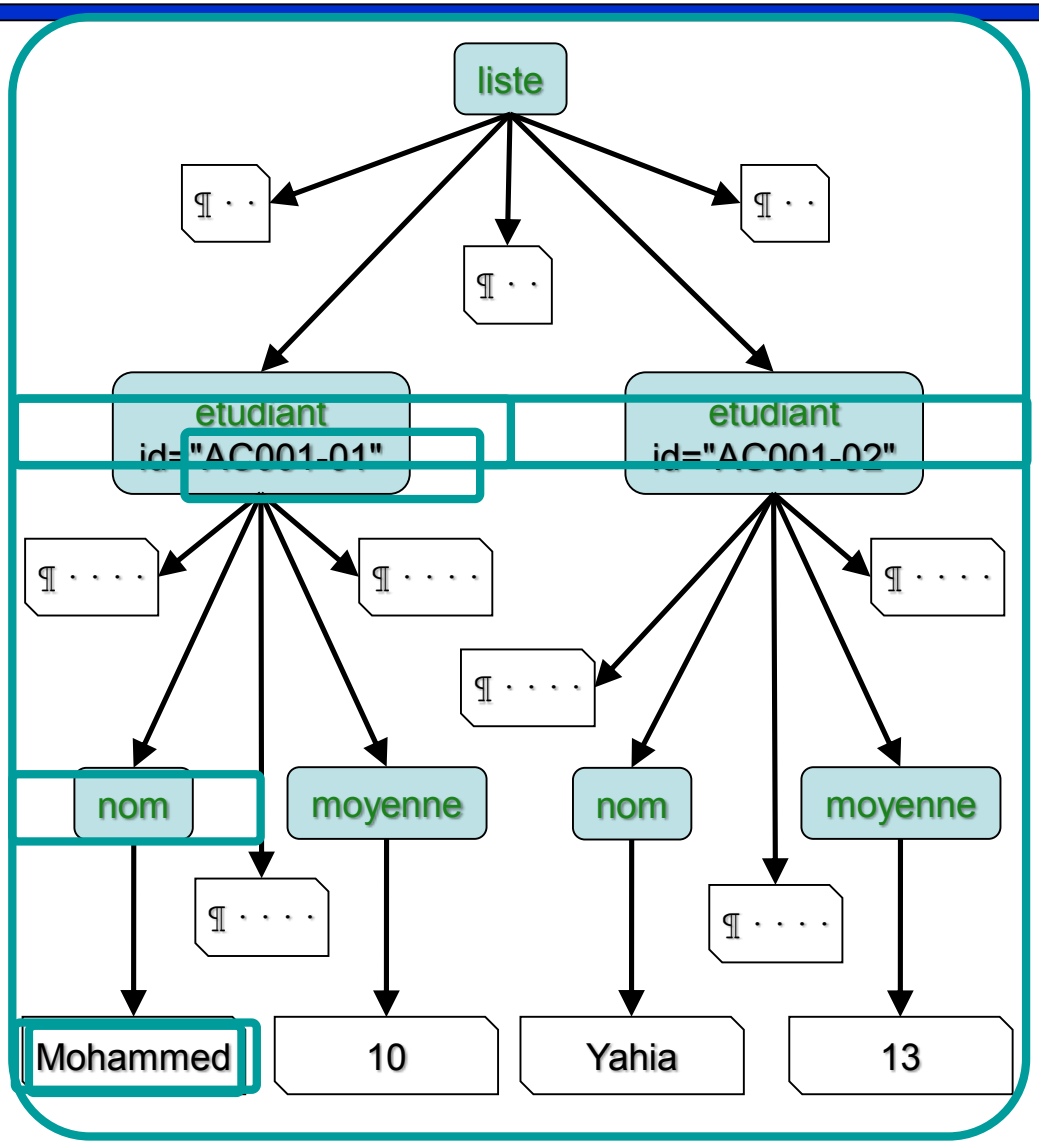
Exploiter des données au format XML

Exemple XML ➔ DOM




```
<?xml version="1.0" encoding="UTF-8" ?>
<liste>
   <etudiant id="AC001-01">
      <nom>Mohammed</nom>
      <moyenne>10</moyenne>
   </etudiant>
   <etudiant id="AC001-02">
      <nom>Yahia</nom>
         <moyenne>13</moyenne>
   </etudiant>
</liste>
```

Exemple XML ➔ DOM




```
xmlHttp.responseXML
```

```
.getElementsByTagName(
        'etudiant')
```

```
[0]
```

```
.getAttribute('id')
```

```
.getElementsByTagName(
    'nom      ')[0]
```

```
.firstChild
```

```
.nodeValue
```

Exploiter des données au format JSON

JSON (JavaScript Object Notation)

- JSON (JavaScript Object Notation) est un format de données générique.
- permet de transmettre de l'information structurée entre différentes technologies.
- Utilise la syntaxe des objets JavaScript
- C'est une très bonne technique à utiliser pour le développement AJAX
- PHP supporte JSON depuis 5.2.
- JSON est utilisé Dans le domaines des smartphones du fait de sa légèreté...
- Site Web : http://www.json.org/
- Il est fortement conseillé de travailler en UTF-8 avec JSON (pour le transport des accents).

JSON (JavaScript Object Notation)

http://www.json.org/

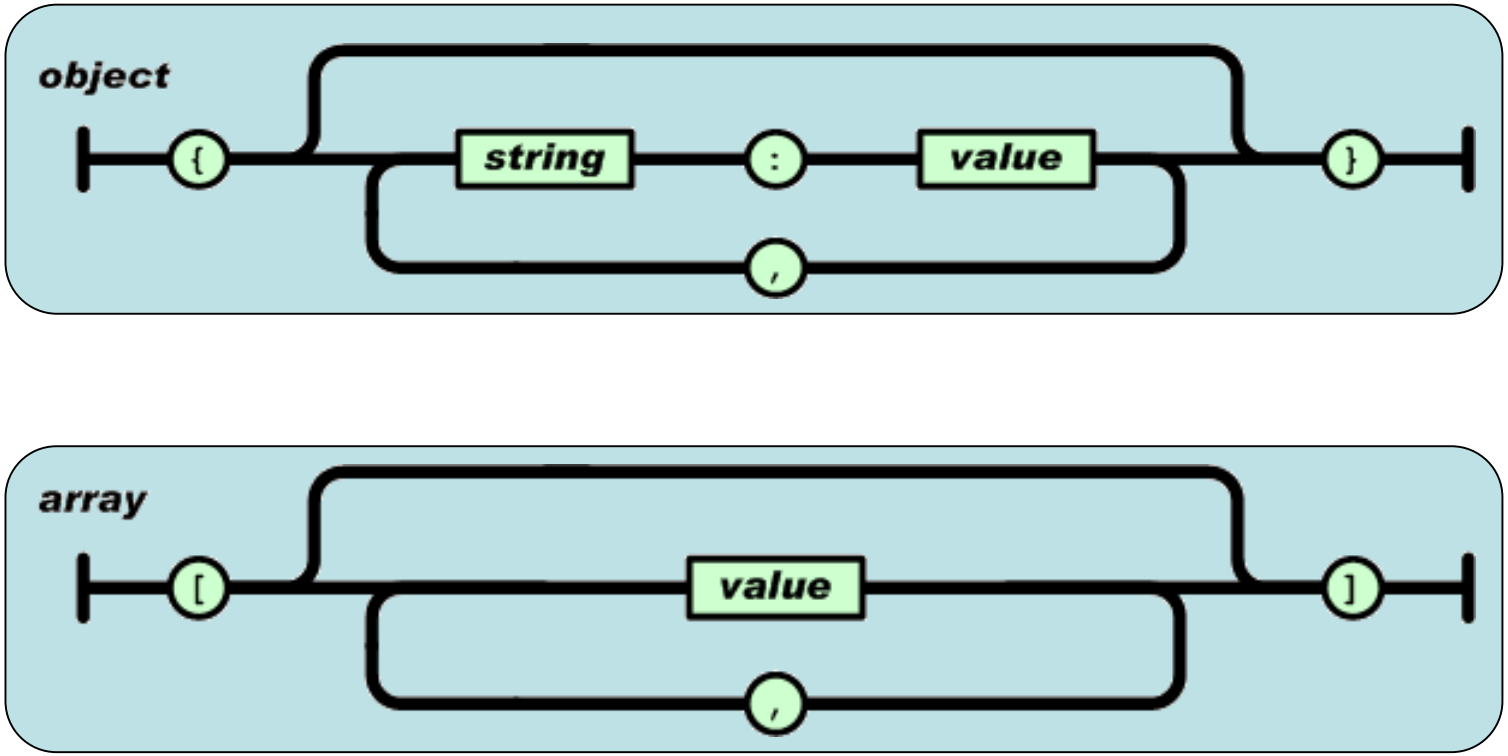
- Deux structures :

1. **Objet**
   - **{}**
   - **{chaîne : valeur}**
   - **{chaîne : valeur, chaîne : valeur, …}**
2. **Tableau**
   - **[]**
   - **[valeur]**
   - **[valeur, valeur, …]**

- **Valeur :**
  - **chaîne|nombre|objet|tableau|true|false|null**

JSON (JavaScript Object Notation)

http://www.json.org/



JSON (JavaScript Object Notation)

Exemple

```
{
"firstName":  "Mohammed",
"lastName" :  "Yahia",
"age" : 25,
"address" :{
    "streetAddress": "21 2nd Street",
    "city" : "Msila",
    "state" : "Msila",
    "postalCode" : "10021"
    },
"phoneNumber":[  {  "type" : "home",
                    "number": "213 662-1234"},
                 {   "type" : "fax",
                     "number": « 213 355-4567"}
    ]
"married":true
}
```

JSON vs XML

```
{
  "menu": "Fichier",
  "commandes": [
      {
          "title": "Nouveau",
          "action":"CreateDoc"
      },
      {
          "title": "Ouvrir",
          "action": "OpenDoc"
      },
      {
          "title": "Fermer",
          "action": "CloseDoc"
      }
   ]
}
```

```
<?xml version="1.0" ?>
<root>
  <menu>Fichier</menu>
  <commands>
     <item>
         <title>Nouveau</value>
         <action>CreateDoc</action>
     </item>
     <item>
         <title>Ouvrir</value>
         <action>OpenDoc</action>
     </item>
     <item>
         <title>Fermer</value>
         <action>CloseDoc</action>
     </item>
  </commands>
</root>
```

JSON vs XML

- **Les avantages de JSON:**
  - ✓ La vitesse de traitement.
  - ✓ La simplicité de mise en œuvre.
  - ✓ On n'a pas besoin de parser un fichier XML pour extraire des informations à travers le net,
  - ✓ car JSON est reconnu nativement par JavaScript.
- **Les avantages de XML:**
  - ✓ XML est extensible quand au langage, on peut créer des formats (comme RSS).
  - ✓ Il est largement utilisé et reconnu par tous les langages de programmation.
  - ✓ Il est plus facile à lire.

Lecture de données JSON en JavaScript

- Idée initiale :
  - JavaScript dispose d'une fonction **eval()** qui exécute du code JavaScript contenu dans une chaine
  - Utiliser eval() pour analyser du JSON :

```
donnee = ‘{ "nom": "Brahimi", "prenoms": [ "Mohamed", "Yahia"] }';
eval("a = " + donnee);
```

- Problème : énorme faille de sécurité !
  - Et si le soi-disant JSON contient du code malicieux ?
- ⇒ Ne jamais utiliser eval().
- ⇒ utiliser la bibliothèque JSON

Bibliothèque JSON

- **JSON.parse** : chaine JSON → valeur JavaScript

```
donnee = '{"nom": "Brahimi", "prenoms": ["Mohammed", "Yahia"]}';
d = JSON.parse(donnee);
```

→ Object {nom: "Brahimi", prenoms: Array[2]}

- **JSON.stringify** : valeur JavaScript → chaine JSON

```
liste = [3.14, {nom: "Pi"}];
j = JSON.stringify(liste);
```

→ "[3.14,{"nom":"Pi"}]"

Exploiter des données au format JSON

Exemple

Fichier,json

```
{
  "menu": "Fichier",
  "commandes": [
     {
        "title": "Nouveau",
        "action":"CreateDoc"
     },
     {
        "title": "Ouvrir",
        "action": "OpenDoc"
     },
     {
        "title": "Fermer",
        "action": "CloseDoc"
     }
   ]
}
```

Exploiter des données au format JSON

Exemple

- **Le code XMLHttpRequest:**

```
var req = new XMLHttpRequest();
req.open("GET","fichier.json", true);
req.onreadystatechange = monCode;
// la fonction de prise en charge (callback)
req.send();
```

- **Le code JavaScript:**

```
function monCode() {
   if (req.readyState == 4)    {
    var doc = JSON.parse(  req.responseText );
   }
}
```

Exploiter des données au format JSON

Exemple

- **Utilisation des données:**

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

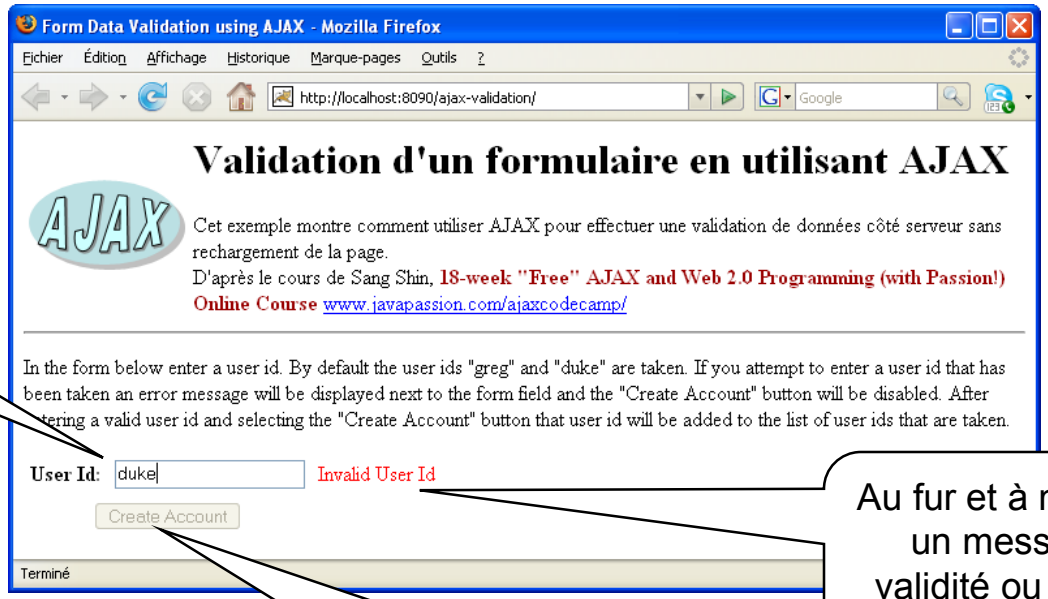
- **Comment on accède aux données:**

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

Exemple d'une interaction AJAX

Anatomie d'une interaction AJAX

- Exemple d'après : *AJAX Basics and Development Tools* de Sang Shin (sang.shin@sun.com, Sun Microsystems) www.javapassion.com/ajaxcodecamp

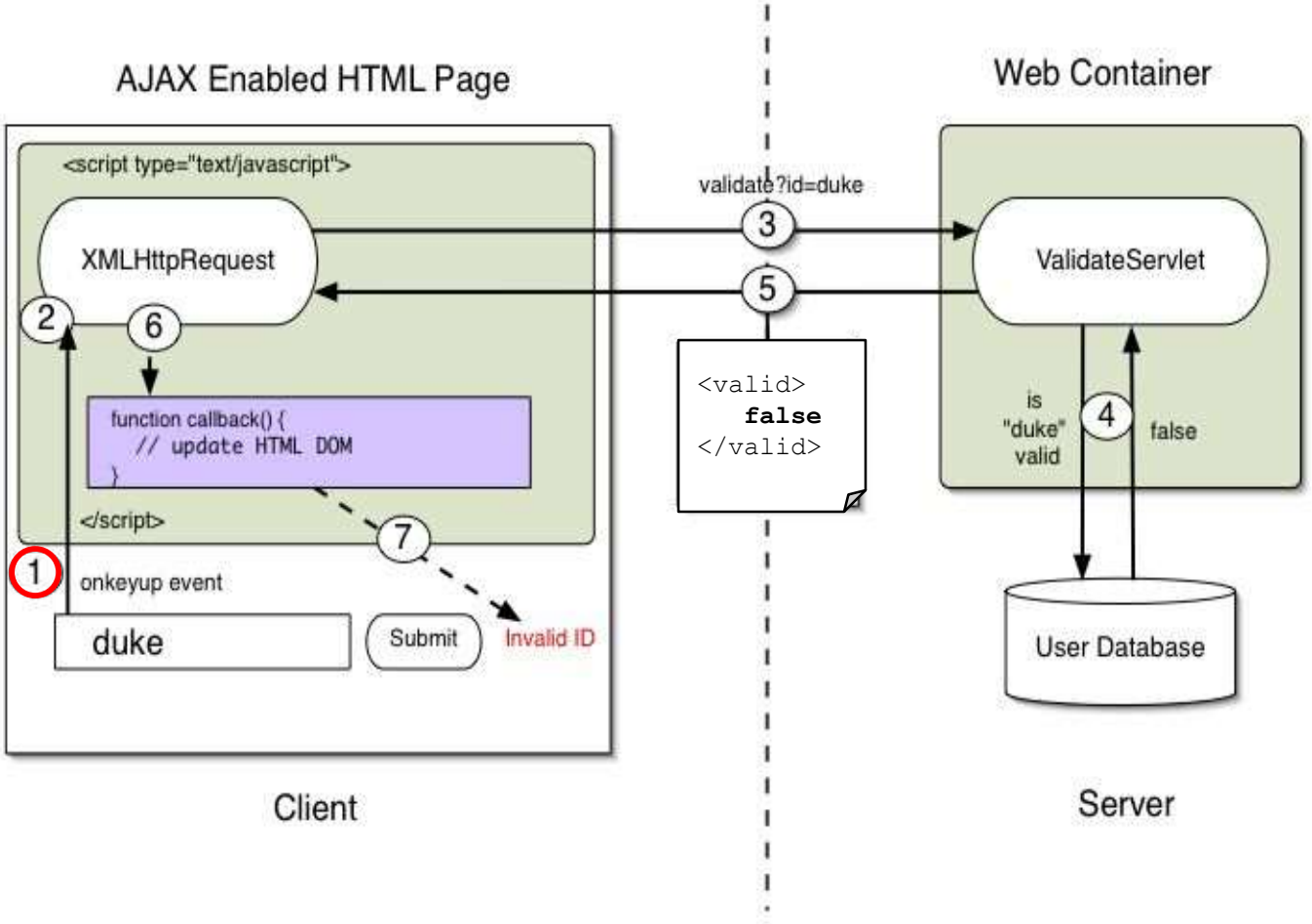


L'utilisateur saisit un identifiant

Au fur et à mesure de la frappe un message indiquant la validité ou non de l'identifiant est affiché

Le bouton de création n'est activé que si l'identifiant est valide (n'est pas déjà utilisé)

Anatomie d'une interaction AJAX



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() et traite ce résultat

⑦ Mise à jour de la page HTML (DOM)

Anatomie d'une interaction AJAX

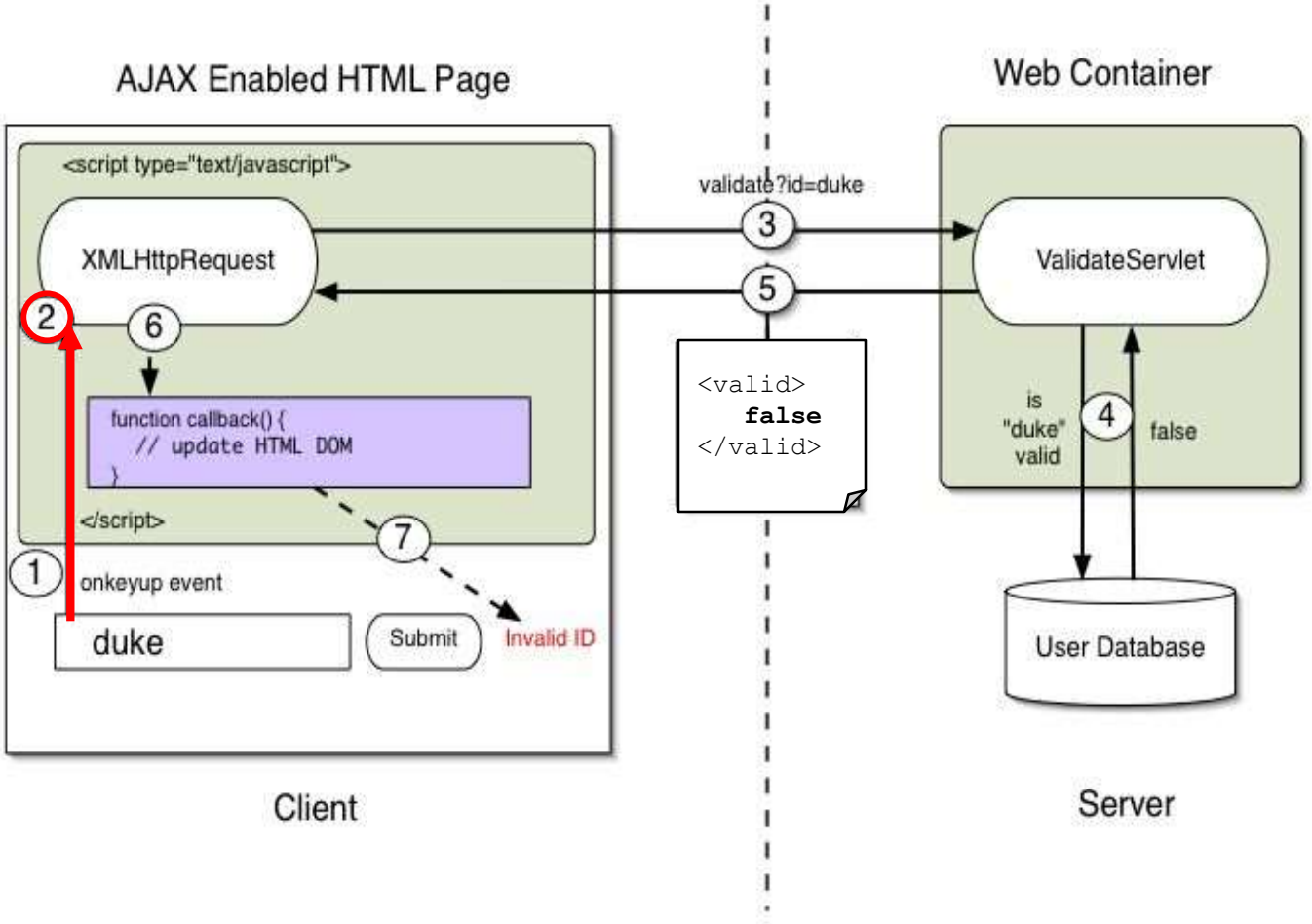
1 Gestion des événements dans le formulaire HTML

La fonction Javascript validateUserId est associée au champ de saisie de texte "userid" pour la gestion des événements de type onkeyup : validateUserId est appelée chaque fois que l'utilisateur tape une lettre dans le champ de saisie.


```
<form name="updateAccount" action="validate" method="post">
    <input type="hidden" name="action" value="create"/>
    <table border="0" cellpadding="5" cellspacing="0">
        <tr>
            <td><b>User Id:</b></td>
            <td>
                <input type="text" size="20" id="userid" name="id" onkeyup="validateUserId()">
            </td>
            <td>
                <div id="userIdMessage"></div>
            </td>
        </tr>
        <tr>
            <td align="right" colspan="2">
                <div align="center">
                  <input id="submit_btn" type="Submit" value="Create Account">
                </div></td>
            <td></td>
        </tr>
    </table>
</form>
```

L'élément div d'id userIdMessage spécifie la position où sera affiché le message de validation de l'entrée

Anatomie d'une interaction AJAX



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() et traite ce résultat

⑦ Mise à jour de la page HTML (DOM)

Anatomie d'une interaction AJAX

Coté client :

la fonction JavaScript invoqué à chaque événement "keyup"sur le champ de saisie

2

```
var req;

function validateUserId() {

    if (window.XMLHttpRequest) {
          req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
          isIE = true;
          req = new ActiveXObject("Microsoft.XMLHTTP");
   }

   req.onreadystatechange = processRequest;

   if (!target)
       target = document.getElementById("userid");
   var url = "validate?id=" + encodeURI(target.value);

   req.open("GET", url, true);


}
```

Création et configuration d'un objet `XMLHttpRequest`

Selon le navigateur l'objet XMLHttpRequest est crée différemment

fonction callback : fonction Javascript (voir plus loin) qui sera invoquée lorsque le serveur aura fini de traiter la requête :

En Javascript les fonctions sont des objets et peuvent être manipulées en tant que tels

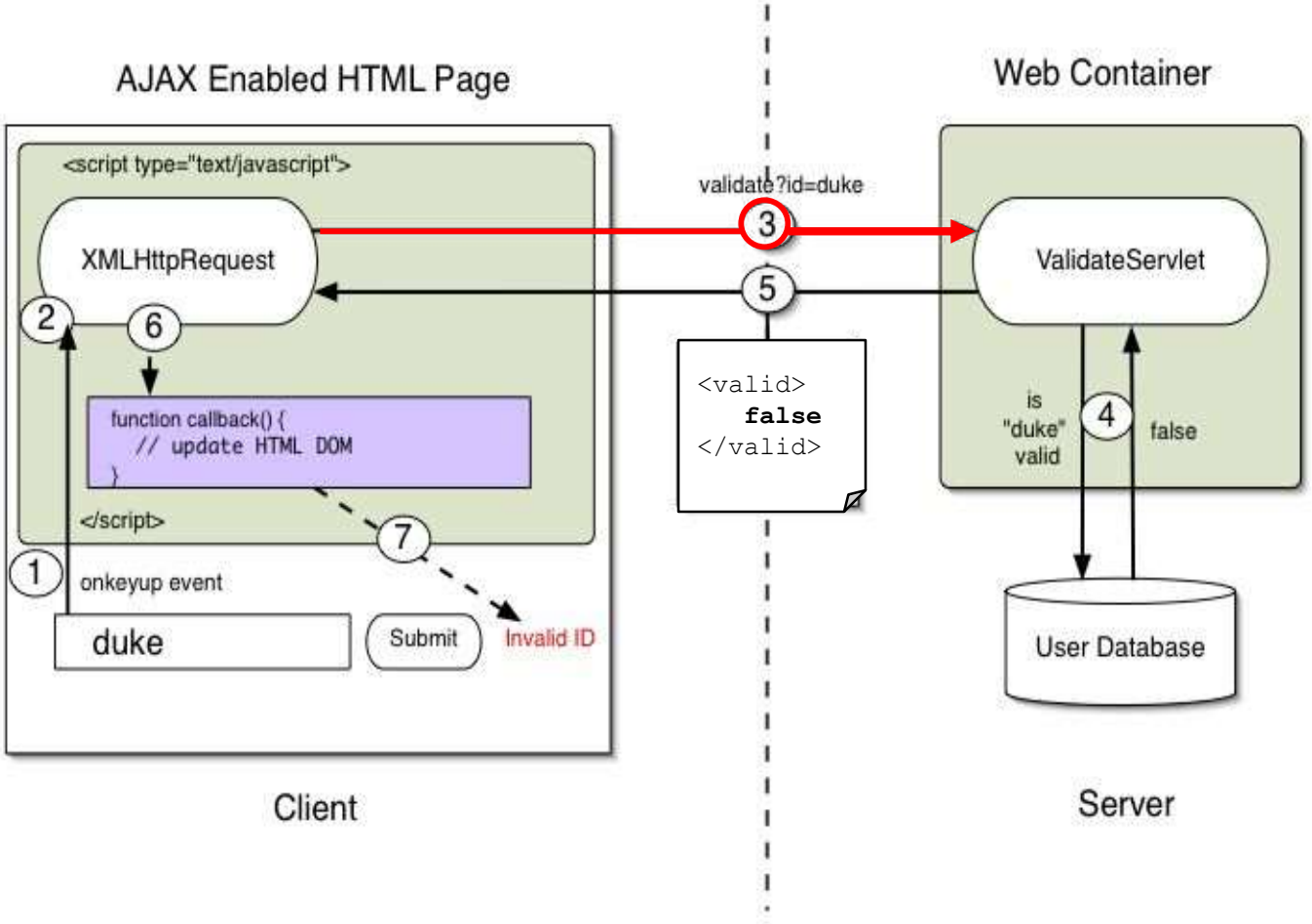
Récupération de la valeur userid tapée par l'utilisateur (via API DOM)

```
<input type="text" size="20" id="userid"
    name="id" onkeyup="validateUserId()">
```

et construction de l'url du composant serveur qui sera invoqué

L'appel sera asynchrone

Anatomie d'une interaction AJAX



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() et traite ce résultat

⑦ Mise à jour de la page HTML (DOM)

Anatomie d'une interaction AJAX

Coté client :

la fonction JavaScript invoqué à chaque événement "keyup"sur le champ de saisie

```
var req;

function validateUserId() {

    if (window.XMLHttpRequest) {
          req = new XMLHttpRequest();
   } else if (window.ActiveXObject) {
          isIE = true;
          req = new ActiveXObject("Microsoft.XMLHTTP");
   }

   req.onreadystatechange = processRequest;

   if (!target)
       target = document.getElementById("userid");
   var url = "validate?id=" + escape(target.value);

   req.open("GET", url, true);

    req.send(null);
}
```

2

Création et configuration d'un objet `XMLHttpRequest`

Selon le navigateur l'objet XMLHttpRequest est crée différemment

fonction callback : fonction Javascript (voir plus loin) qui sera invoquée lorsque le serveur aura fini de traiter la requête :

En Javascript les fonctions sont des objets et peuvent être manipulées en tant que tels

Récupération de la valeur userid tapée par l'utilisateur (via API DOM)

```
<input type="text" size="20" id="userid"
    name="id" onkeyup="validateUserId()">
```

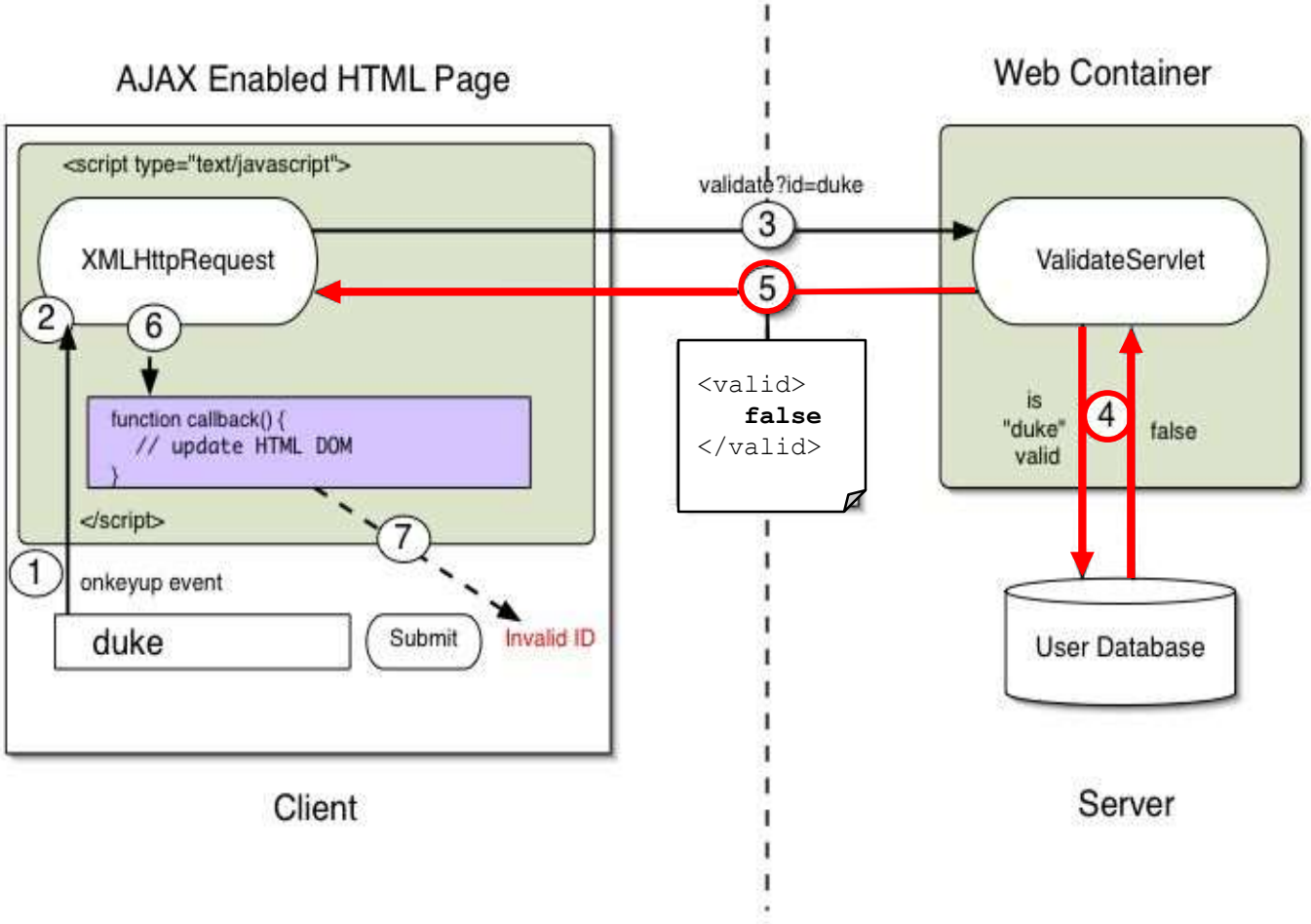
et construction de l'url du composant serveur qui sera invoqué

L'appel sera asynchrone

3

L' objet `XMLHttpRequest` effectue une requête asynchrone

Anatomie d'une interaction AJAX



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() et traite ce résultat

⑦ Mise à jour de la page HTML (DOM)

Anatomie d'une interaction AJAX

Coté Serveur :

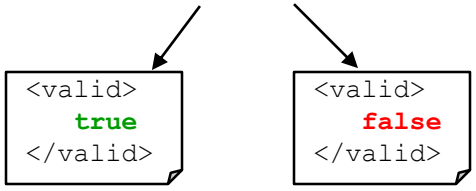
la servlet traitant la requête GET émise par la fonction JavaScript `validateUuserId`

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse  response)
        throws IOException, ServletException {

    String targetId = request.getParameter("id");

    if ((targetId != null) &&
          LoginManager.validateUserId(targetId.trim())) {

        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
    } else {

        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>false</valid>");
    }
}
```

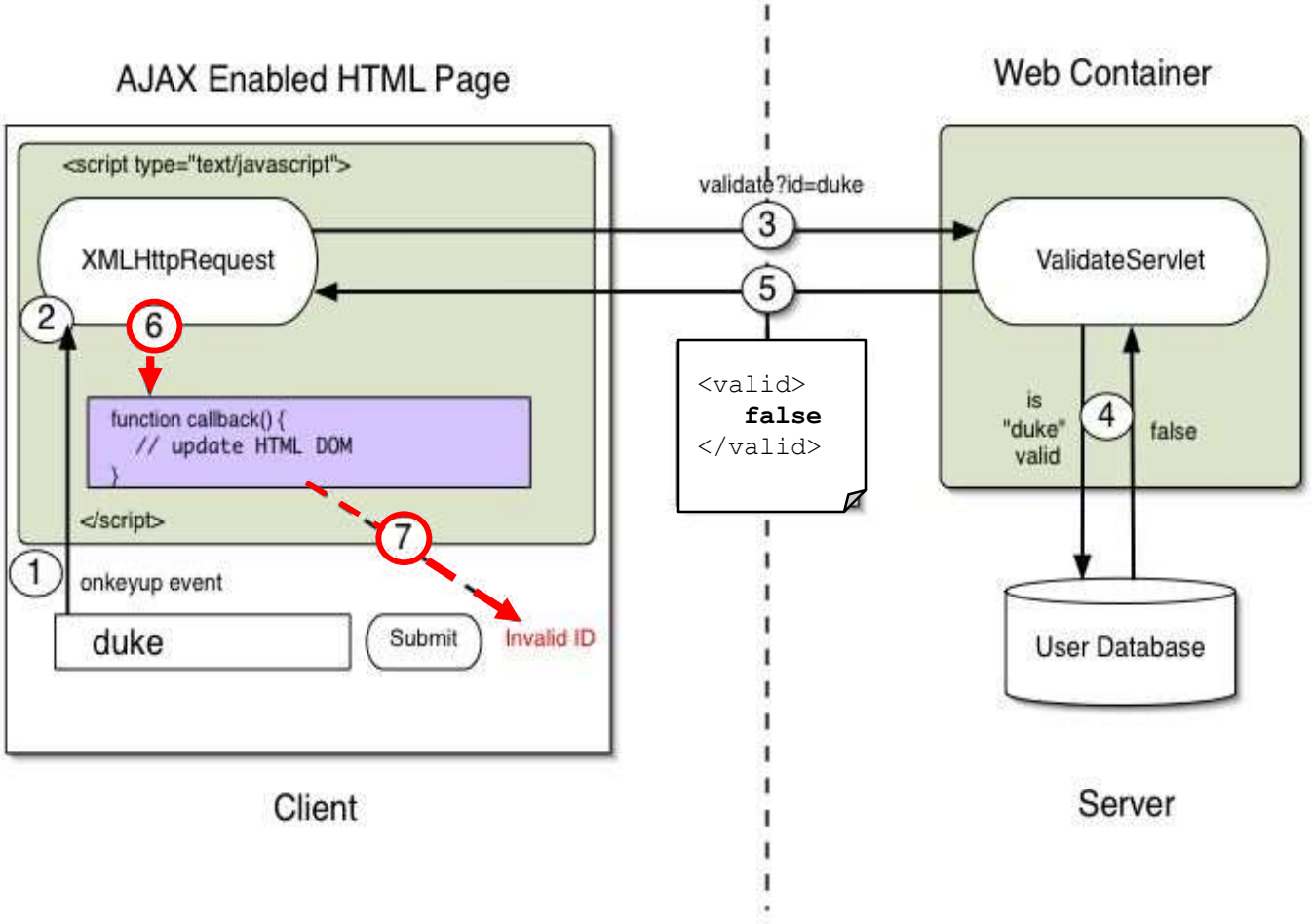
4 **Le servlet valide l'identifiant soumis**

S'agit-il d'un identifiant déjà utilisé ?

5 **Le servlet retourne un document XML contenant le résultat de la validation**



Anatomie d'une interaction AJAX



① Événement sur le client → appel d'une fonction javascript

② Création et configuration d'un objet XMLHttpRequest

③ L' objet XMLHttpRequest fait une requête asynchrone

④ Le servlet valide l'identifiant soumis

⑤ Le servlet retourne un document XML contenant le résultat de la validation

⑥ L'objet XMLHttpRequest appelle la fonction callback() qui traite ce résultat

⑦ Mise à jour de la page HTML (DOM)

Anatomie d'une interaction AJAX

Coté client :

6 L'objet `XMLHttpRequest` appelle la fonction callback() et traite ce résultat

```
function processRequest() {

   if (req.readyState == 4) {
      if (req.status == 200) {


         var message = req.responseXML.
               getElementsByTagName("valid")[0].
               childNodes[0].nodeValue;


         setMessageUsingDOM(message);


          var submitBtn = document.getElementById("submit_btn");
          if (message == "false") {
               submitBtn.disabled = true;
          } else {
               submitBtn.disabled = false;
          }
      }
   }
}
```

```
<valid>
   false
</valid>
```

Cette fonction est invoquée chaque fois que le champ readyState de l'objet XMLHttpRequest est modifié

readyState == 4 et status = 200 indiquent que la réponse a été correctement reçue par le client

Extraction de la valeur true ou false des données retournées par le serveur

7 Mise à jour de la page HTML (DOM)

Affiche dans la zone prévue à cet effet la validité ou non de l'identificateur fourni

Active ou désactive le bouton de soumission du formulaire selon validité de l'identificateur fourni

Anatomie d'une interaction AJAX

7 Mise à jour de la page HTML (DOM)

```
<td>
   <div id="userIdMessage"></div>
</td>
```

Invalid User Id

```
function setMessageUsingDOM(message) {

   var userMessageElement = document.getElementById("userIdMessage");

   var messageText;
   if (message == "false") {
      userMessageElement.style.color = "red";
      messageText = "Invalid User Id";
   } else {
      userMessageElement.style.color = "green";
      messageText = "Valid User Id";
   }

   var messageBody = document.createTextNode(messageText);

   if (userMessageElement.childNodes[0]) {
       userMessageElement.replaceChild(
              messageBody, userMessageElement.childNodes[0]);
   } else {
       userMessageElement.appendChild(messageBody);
   }
}
```

Récupération de l'objet DOM correspondant à la zone du message grâce à l'id inséré dans le code HTML

Préparation du message

Création du message
Si il existe déjà un message le remplace par le nouveau, sinon le rajoute

Alors, c'est comme ça qu'on fait de l'AJAX ?

Ajax Frameworks et Toolkits

- Non, aujourd'hui on ne fait quasiment plus d'Ajax de si bas niveau, ce cours est fait pour "**comprendre les bases**".
- Les toolkits comme jQuery, Dojo, Symphony, Prototype, etc cachent la complexité de XMLHttpRequest et simplifient l'usage des APIs du DOM.

Les bibliothèques AJAX

- Prototype JavaScript Framework.
  http://www.prototypejs.org/
- The Yahoo! User Interface Library (YUI).
  http://developer.yahoo.com/yui/
- Microsoft ASP .NET AJAX (supporté dans VS).
  http://www.asp.net/ajax/
- Script Aculos
  http://script.aculo.us/
- Jquery
  http://jquery.com

Sécurité et AJAX

AJAX Sécurité: Côté Serveur

- Les moteurs Ajax des navigateurs n'autorisent que des requêtes Ajax vers le serveur qui a servi la page
  - Mais de nombreux frameworks utilisent des astuces à base de iFrame HTML pour arriver à requêter en ajax des serveurs externes.
  - Souvent c'est transparent pour l'utilisateur.

AJAX Sécurité: Côté Client

- Le code JavaScript est visible pour un hacker.
  - Des techniques de compression de code peuvent être utilisées.
- Attention quand le serveur envoie du javascript qui est évalué sur le client (eval(…) de js)
  - Trou de sécurité possible.

Avantages et inconvénients d'AJAX

Avantages :

- Plus interactivité au niveau du client.
- Réponse plus rapide.
- Réduction des transactions client/serveur (récupération des scripts et des feuilles de style une fois pour toute).
- Séparation des méthodes pour la transmission de l'information et des formats utilisés pour représenter les informations.

Inconvénients :

- Pas d'enregistrement dans l'historique du navigateur des pages modifiées dynamiquement.

⇨ Solution en modifiant la partie ancre (#) de l'URL.

- Pas d'indexation possible des pages par les moteurs de Recherche.
- Si un navigateur ne supporte pas Javascript et AJAX, la page est inutilisable.