Université Sidi Mohamed Ben Abdllah

Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

WEB DYNAMIQUE: AJAX

Réalisé par: Pr. Mahraz Med Adnane

Année universitaire:2016/2017

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

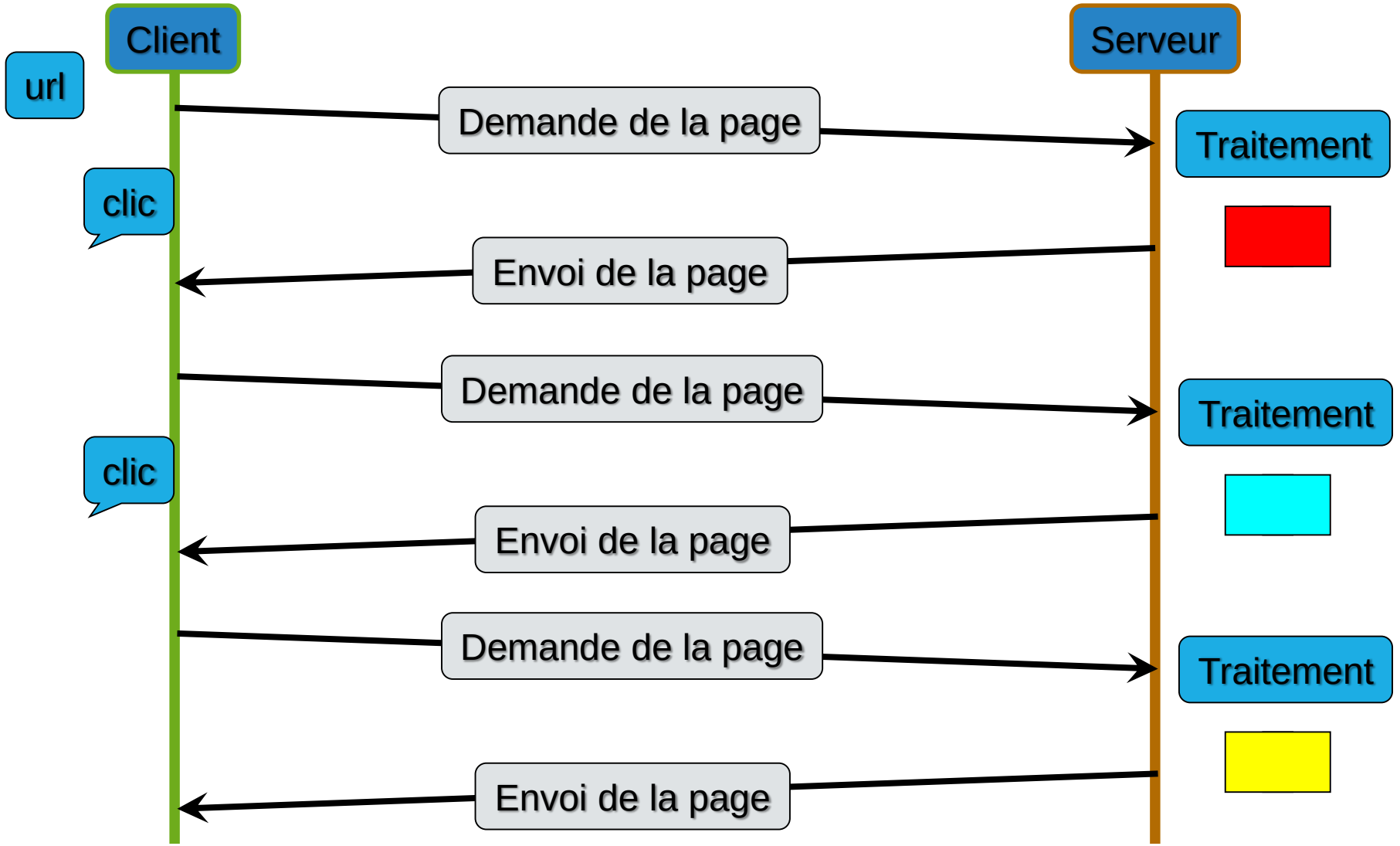
AJAX ?

- 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)

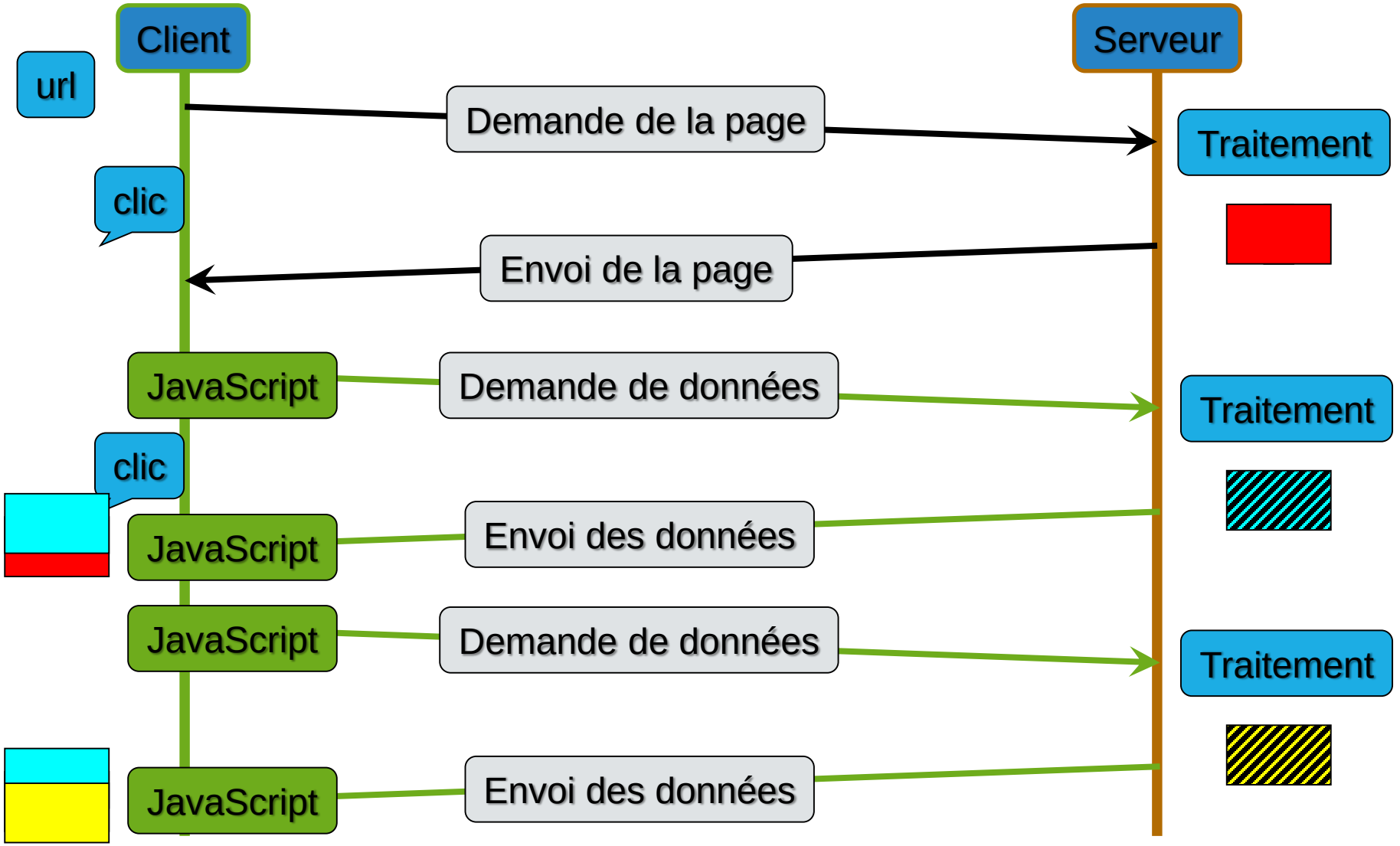
ASYNCHRONE ? (UNE VISION)

- Fonctionnement classique du Web :
  - Chargement d'une page
  - Interaction de l'utilisateur (clic, sélection, formulaire, …)
  - Chargement d'une autre page
  - Interaction de l'utilisateur (clic, sélection, formulaire, …)
  - Chargement d'une autre page
  - ...
- Fonctionnement "AJAX" du Web :
  - Chargement d'une page
  - Interaction de l'utilisateur (clic, sélection, formulaire, …)
  - Chargement d'une partie de page
  - Interaction de l'utilisateur (clic, sélection, formulaire, …)
  - Chargement d'une autre partie de page
  - ...

ÉCHANGES CLIENT/SERVEUR CLASSIQUES



ÉCHANGES CLIENT/SERVEUR EN AJAX



XMLHTTPREQUEST

- Cœur d'AJAX
- Objet JavaScript
- En fait, objets JavaScript
  - Microsoft :
    - ActiveXObject("Microsoft.XMLHTTP")
  - Les autres :
    - XMLHttpRequest()
- Permet d'effectuer des requêtes HTTP et d'en récupérer le résultat

XMLHTTPREQUEST : PROPRIÉTÉS

`onreadystatechange`	Gestionnaire d'événements pour les changements d'état. Il faut assigner une fonction à cette propriété pour traiter sur les données renvoyées.
`readyState`	Statut de l'objet.
`responseText`	Réponse sous forme de chaîne de caractères.
`responseXML`	Réponse sous forme d'objet DOM.

XMLHTTPREQUEST : PROPRIÉTÉS ET MÉTHODES

`status`	Code numérique de réponse du serveur HTTP.
`statusText`	Message accompagnant le code de réponse.
`open`	Prépare une requête HTTP en indiquant ses paramètres.
`send`	Effectue la requête HTTP, en envoyant les données.
`setRequestHeader`	Assigne une valeur à un champ d'entête HTTP.

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

INSTANCIER **XMLHTTPREQUEST**

```
function GetXmlHttpObject() {
  var objXMLHttp = null ;
 if(window.XMLHttpRequest){
 objXMLHttp=new XMLHttpRequest();//autres hormis IE
}else{
 objXMLHttp=new ActiveXObject("Microsoft.XMLHTTP");
// pour IE
}
return objXMLHttp ;
}
var xmlHttp = GetXmlHttpObject();
```

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

XMLHTTPREQUEST::OPEN()

- Initialiser une requête HTTP
- **open (*method*, *URL* [, *asyncFlag*[, *userName* [, *password*]]])**
- Paramètres :
  - ***method*** : **"GET"** ou **"POST"** (ou **"HEAD"**)
  - ***URL*** : relative ou absolue
  - ***asyncFlag*** : mode asynchrone ? **true** ou **false**
  - ***userName*** : nom d'utilisateur
  - ***password*** : mot de passe
- Remarque : l'URL peut contenir des paramètres, **test.php?id=12&a=salut**

PARAMÈTRES D'URL EN JAVASCRIPT

- Comment *échapper* une chaîne en JavaScript ?
  - Écrire une table de transcription (bof…)
  - Utiliser ***string*** **escape(*string*)**
  - **escape("J'ai faim !")**
    ➔ J%27ai%20faim%20%21
- Comment *déséchapper* une chaîne ?
  - Écrire une table de transcription inverse (bof…)
  - Utiliser ***string*** **unescape(*string*)**
  - **unescape("J%27ai%20faim%20%21")**
    ➔ J'ai faim !
- **var url="test.php?v="+escape(valeur_v)**

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

XMLHTTPREQUEST::SEND()

- Effectue la requête HTTP initialisée avec **`XMLHTTPRequest::open()`**
- Requête de type **`"GET"`** ou **`"HEAD"`**
  - Aucune donnée à envoyer
  - **`xmlHttp.send(null) ;`**
- Requête de type **`"POST"`**
  - données à envoyer
  -
```
xmlHttp.setRequestHeader(
   'Content-Type',
   'application/x-www-form-urlencoded') ;
xmlHttp.send("v=valeur&x=12");
```

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

CHANGEMENTS D'ÉTAT DE LA REQUÊTE

- 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

FONCTION DE TRAITEMENT DU RÉSULTAT

- Désigner la fonction qui sera lancée lors des changements d'état de la requête
- Désigner la fonction qui sera lancée quand la requête sera terminée et donc que les données seront disponibles
- **XMLHTTPRequest.onreadystatechange**
-
```
function traiter() {
  if (xmlHttp.readyState == 4)
    window.alert('Données dispo !');
}
xmlHttp.onreadystatechange=traiter;
```

Requête terminée

UTILISATION DE **XMLHTTPREQUEST**

1. Instancier l'objet
2. Initialiser une requête
   - Méthode, URL
   - Asynchrone ?
3. Effectuer la requête
   - Envoyer des données ?
4. Associer une fonction au traitement du résultat de la requête
5. Traiter le résultat
   - Texte ?
   - XML / JSON ?

TRAITER LE RÉSULTAT DE LA REQUÊTE

- Traitement effectué dans la fonction associée au changement d'état de la requête quand l'état est 4 (requête terminée)
- Le résultat peut se présenter sous 3 formes :
  - texte qui peut aussi être du code HTML
  - XML
  - JSON
- Le traitement consiste généralement en une modification de la page Web courante en utilisant JavaScript, le DOM et les CSS

RÉCUPÉRER LE RÉSULTAT DE LA REQUÊTE

- 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)`**

EXPLOITER DES DONNÉES AU FORMAT TEXTE

- Texte brut
  - `document.mon_formulaire.saisie.value = xmlHttp.responseText`

- Texte contenant du code HTML
  - `document.getElementById('txt').innerHTML = xmlHttp.responseText`

AJAX EN JQUERY

- jQuery offre plusieurs méthodes pour effectuer des appels asynchrones en AJAX:
- $.get() et $.post() : ces méthodes (autonomes) permettent de charger des données d'un serveur web via une requête GET ou POST.

  La syntaxe est celle-ci (c'est la même chose pour $.post) :

  $.get(URL, [données sous forme de query string], [callback]) ;

AJAX EN JQUERY

- Cette méthode est la plus complète et flexible de toutes. $.get, $.post sont des « raccourcis » de $.ajax(). C'est donc également celle qui demande le plus de configuration.
- Pour la configurer, il suffit de lui passer un objet littéral. Les paramètres de cet objet les plus couramment utilisés sont :
- $.ajax({
  - url : l'adresse serveur vers laquelle faire la requête
  - type : le type de requête, « GET » ou « POST »
  - data : les données à envoyer au serveur
  - dataType : le format de données renvoyées par le serveur (« xml », « html », « json », « text »)
  - async : true ou false, défini si la requête doit être synchrone ou asynchrone
  - success : une fonction appelée si la requête réussit. Cette fonction reçoit en premier argument les données renvoyées par le serveur
  - error : une fonction appelée si la requête échoue })

PETIT BILAN AUTOUR D'AJAX

- Pour :
  - Basé sur des standards ouverts
  - Minimise la bande passante
  - Interfaces réactives, attente réduite
  - Interfaces proches des clients lourds
- Contre :
  - Maximise le nombre de requêtes
  - Coût de développement
  - Perte de suivant / précédent, favoris
  - A la mode, il faut en coller partout…
  - JavaScript, accessibilité des anciens navigateurs