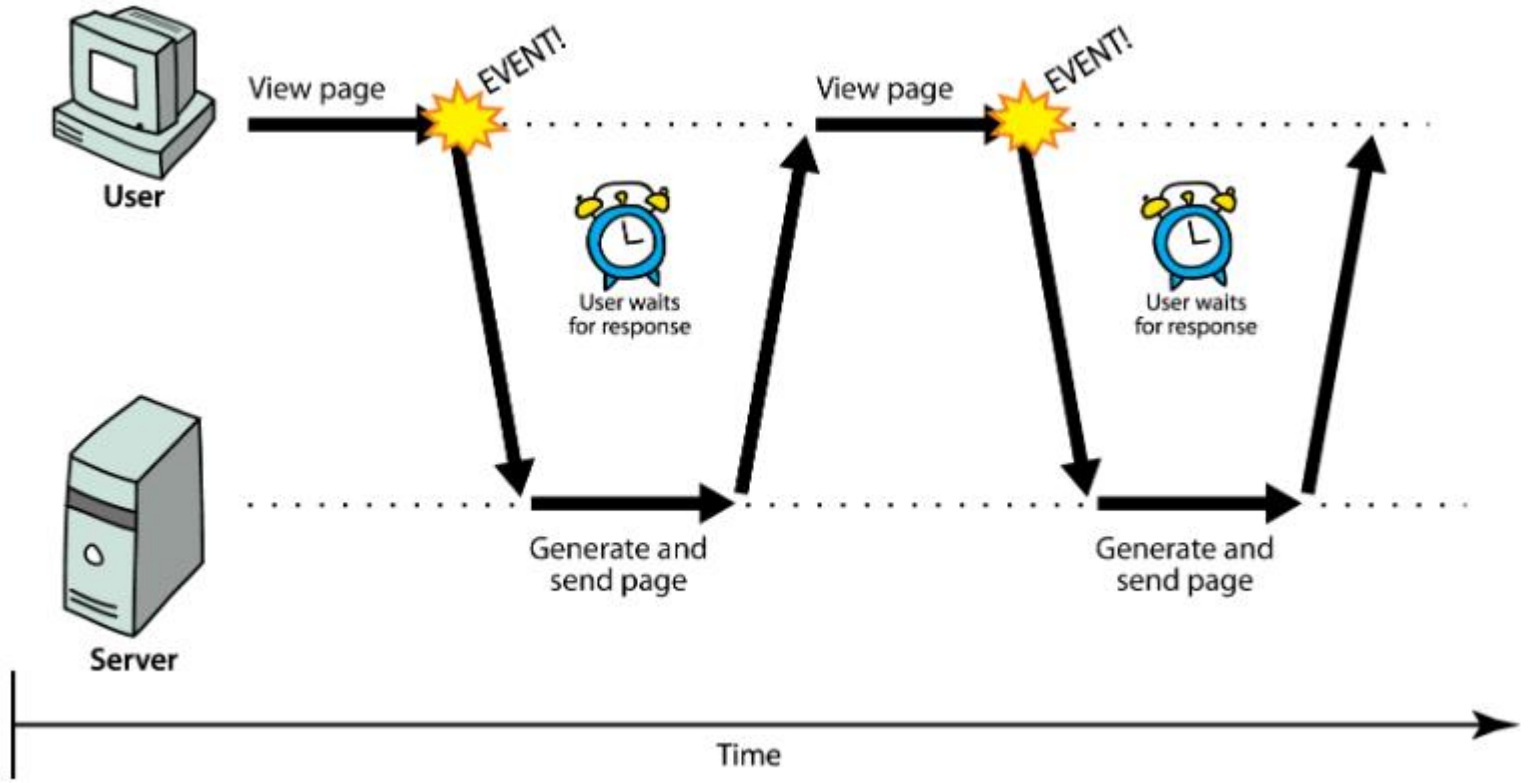
AJAX

Prof. Amina ADADI

2ème Année Génie Logiciel

Année universitaire 2019-2020

Problème de la communication synchrone



Synchronisée : l'utilisateur doit attendre pendant que la nouvelle page se charge. La communication suit alors le modèle **(click, wait, refresh)**

AJAX

Ajax => Asynchronous JavaScript and XML

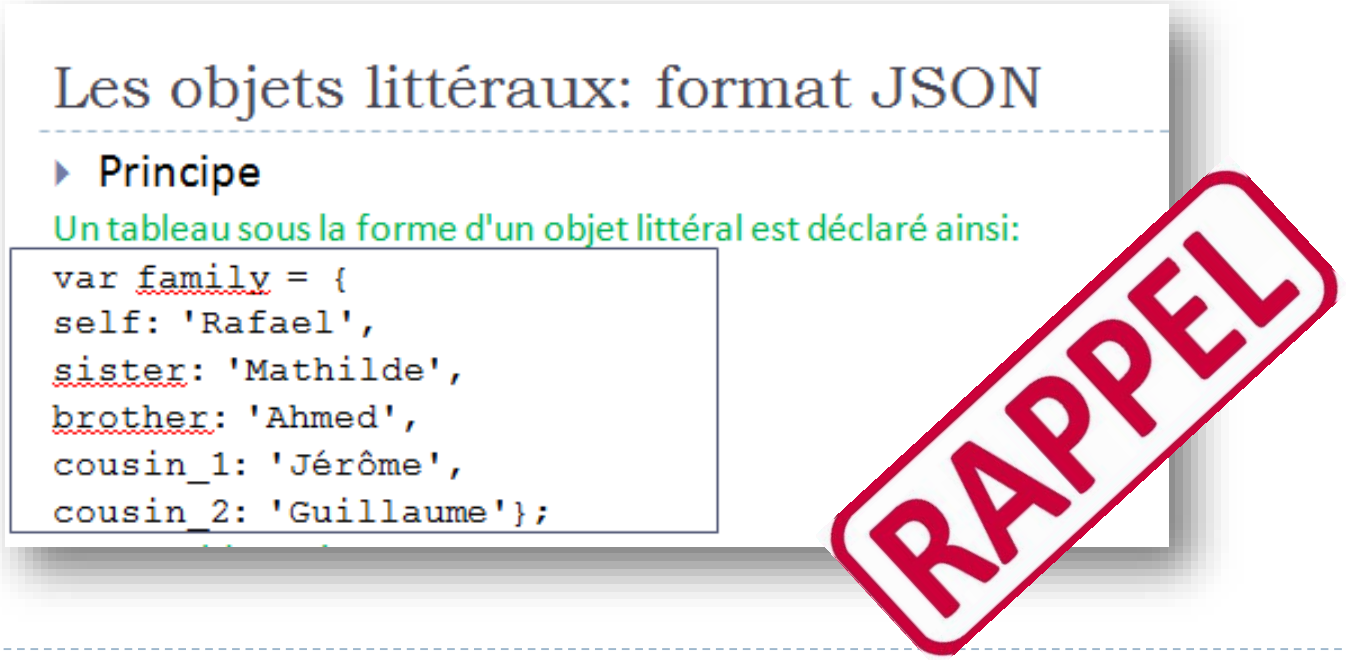
- n'est pas un nouveau langage.
- il s'agit de fonctions en JavaScript.
- permet d'interroger le serveur sans recharger intégralement la page.
- la gestion des événements se fait de façon asynchrone.

En pratique, le XML a été remplacé par du **JSON**.
mais "AJAJ", c'est ridicule, donc on a gardé **AJAX**

JSON

JSON=> JavaScript Object Notation

- format texte permettant d'échanger des objets entre le client et le serveur.



JSON Vs XML

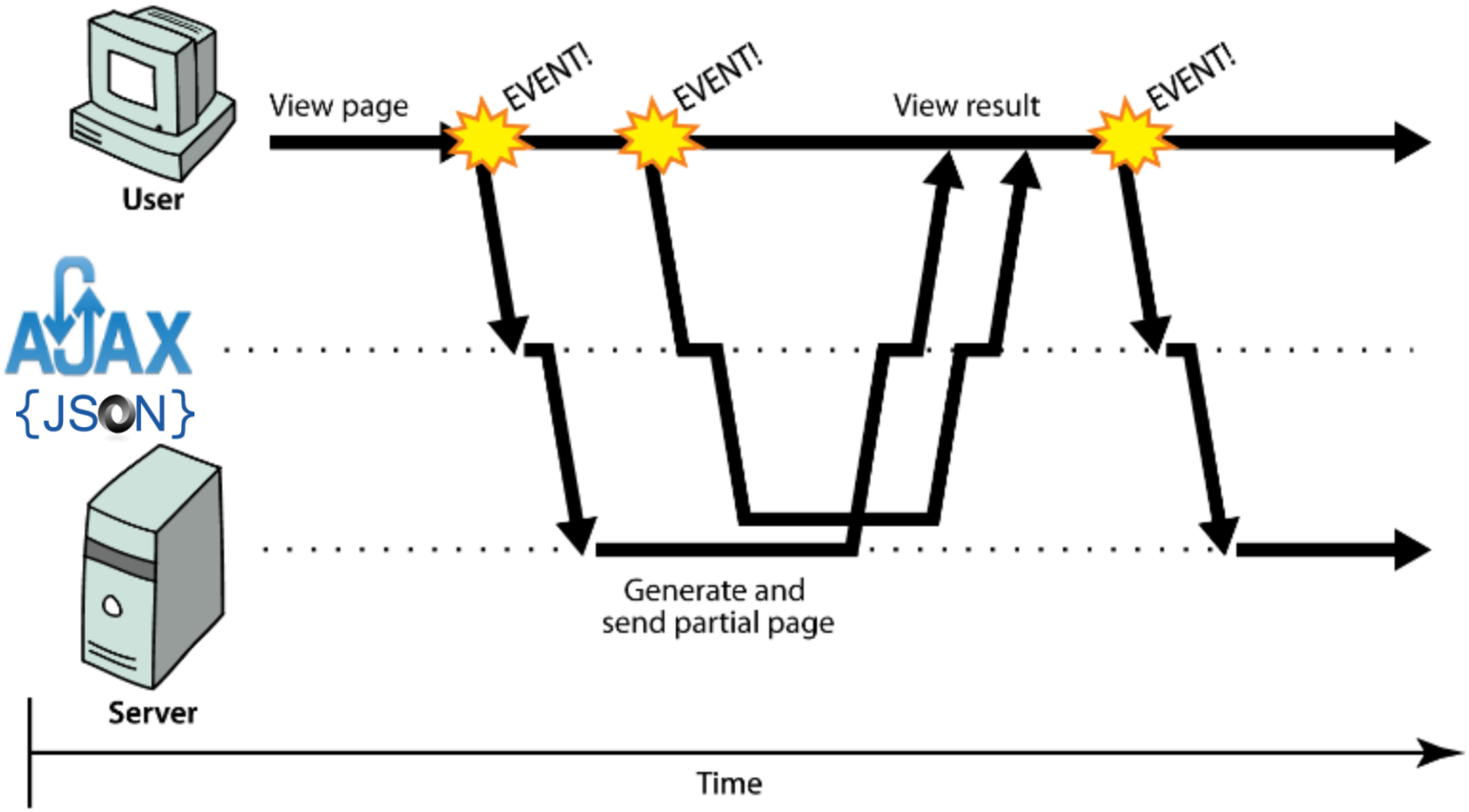
```
{
 ” etudiant ” {
  ”nom” : ” Ahmadi” ,
  ”prenom” : ” Ahmed” ,
   ”age” : ” 20 ” ,
 }
}
```

```
<etudiant>
<nom>  Ahmadi </nom>
<prenom>  Ahmed </
prenom>
<age>20</ age>
</ etudiant
```

JSON a la réputation:

- d’ être moins verbeux, donc moins volumineux,
- d’ être plus facile à parser.
- Pour le serveur, comme pour le client, c’est donc un gain en efficacité.

Communication Asynchrone



Asynchrone: L'utilisateur peut continuer à interagir avec la page pendant que les données sont en cours de chargement

Comment ça fonctionne ?

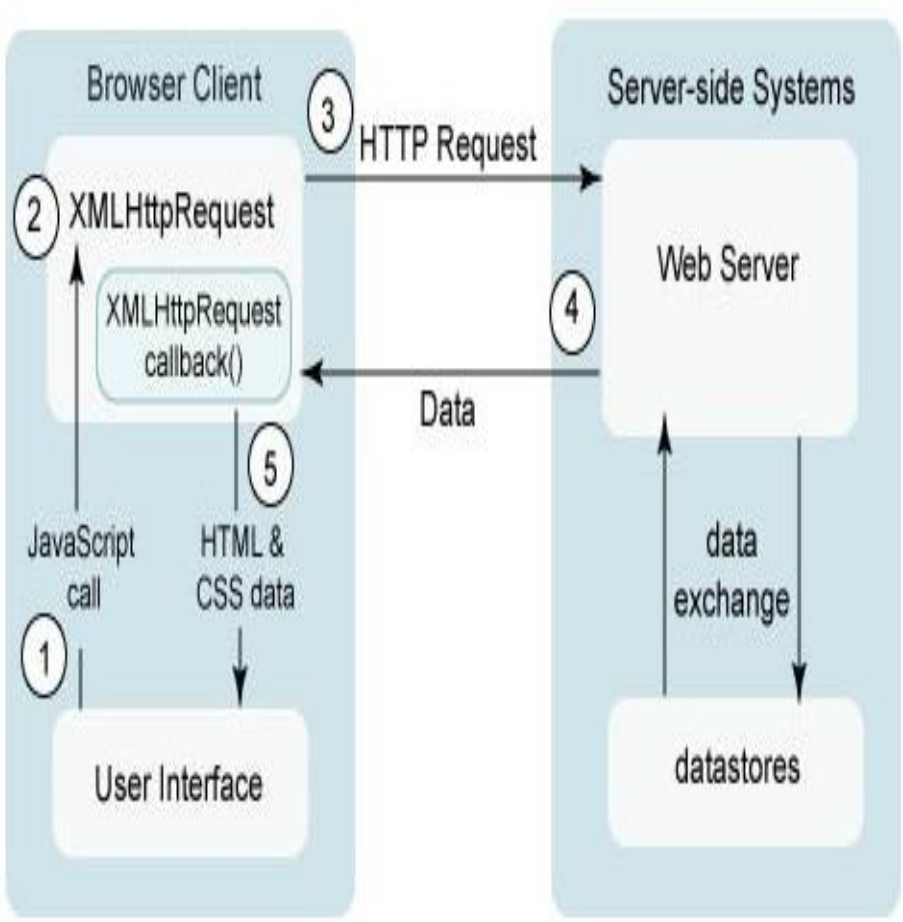
- **L'idée:** utiliser conjointement les technologies Web: HTML, CSS, DOM, Javascript, XMLHttpRequest, JSON/XML

1. JavaScript inclut un objet XMLHttpRequest qui peut récupérer des fichier du serveur d'une manière asynchrone
2. Ces fichiers se basent sur JSON comme format d'échange
3. Les données des fichiers récupérés sont ajoutés dans le HTML avec les styles appropriés en utilisant DOM.

Comment ça fonctionne ?

- Requête typique AJAX:

1. L'utilisateur clique, déclenche un gestionnaire d'événement.
2. Le code du gestionnaire crée un objet XMLHttpRequest .
3. L'objet XMLHttpRequest demande les données du serveur.
4. Le serveur récupère les données appropriées (base de données) et les renvoient.
5. XMLHttpRequest déclenche un événement lorsque les données sont reçues.
    1. Souvent la fonction est appelée **callback.**
    2. on peut gérer l'événement associé au **callback.**
6. Le gestionnaire de l'événement **callback** traite les données reçu et l'affiche sur le document HTML.



L'objet XMLHttpRequest

- Création de l'objet

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

- Méthodes de l'objet

Méthode	Description
new XMLHttpRequest()	Crée un nouveau objet XMLHttpRequest
abort()	Annule la requête en cours
open(method, url, async, user, psw)	Spécifie la requête method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Envoi la requête au seveur utilisée pour les requêtes GET
send(string)	Envoi la requête au seveur utilisée pour les requêtes POST

L'objet XMLHttpRequest

- Propriétés de l'objet

Propriété	Description
onreadystatechange	Définit la fonction qui doit être appelé une fois la propriété readyState change
readyState	Contient le statut de la requête XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Retourne les données de réponse comme un string
responseXML	Retourne les données de réponse comme données XML
status	Retourne le nombre du status-number de la requête: 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Retourne le status-text (e.g. "OK" or "Not Found")

Exemple

```
<!DOCTYPE html>
<html> <body>
<div id=« ajax">
  <h2>AJAX va changer ce text</h2>
  <button type="button" onclick="loadDoc()"> Cliquer pour Changer contenu</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("ajax").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
 </script>

</body>
```

Exemple (avec fichier JSON)

```
<!DOCTYPE html>
<html> <body>
<div id=« ajax">
  <h2>AJAX va changer ce text</h2>
  <button type="button" onclick="loadDoc()"> Cliquer pour Changer contenu</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      var myObj = JSON.parse(this.responseText);
       document.getElementById("ajax").innerHTML = myObj.name;
    }
  };
  xhttp.open("GET", « json_demo.json", true);
  xhttp.send();
}
 </script>

</body>
```

Tant que les données sont écrites en JSON avec perse() on peut gérer le string retourné par la requête sous forme d'objets

Exercice 9

Reprenez l'exercice 5 du cours HTML relatif aux tableaux et charger dynamiquement les notes des étudiants à partir d'un fichier JSON et ce en réponse à un clique sur un bouton

Nom	Prénom	Matière						Moyenne
		POO		PHP		BD		
		Note	Note X Coeff	Note	Note X Coeff	Note	Note X Coeff	
ALLALI	Ali	12	48	15.5	93	4.5	9	12.5
IDRISSI	Meriem	13	52	6	36	13	26	9.5
FATHI	Imane	7	28	10	60	7	14	8.5
ALAMI	Amine	8	32	16	96	14	28	13
Moyenne de la classe								10.875

Tableau HTML