ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Introduzione a Javascript III parte

Fabio Vitali

Corsi di laurea in Informatica e
Informatica per il Management

Alma Mater – Università di Bologna

Oggi parleremo di…

Javascript

- Sintassi base (parte I)
- Modello oggetti del browser (parte II)

AJAX (parte III):

- ***Architettura di riferimento***
- ***XMLHttpRequest***

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Ajax

AJAX: Introduzione

AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) è una tecnica per la creazione di applicazioni Web interattive.

Permette l'aggiornamento **asincrono** di **porzioni** di pagine HTML

Utilizzato per incrementare:

- l'**interattività**
- la **velocità**
- l'**usabilità**

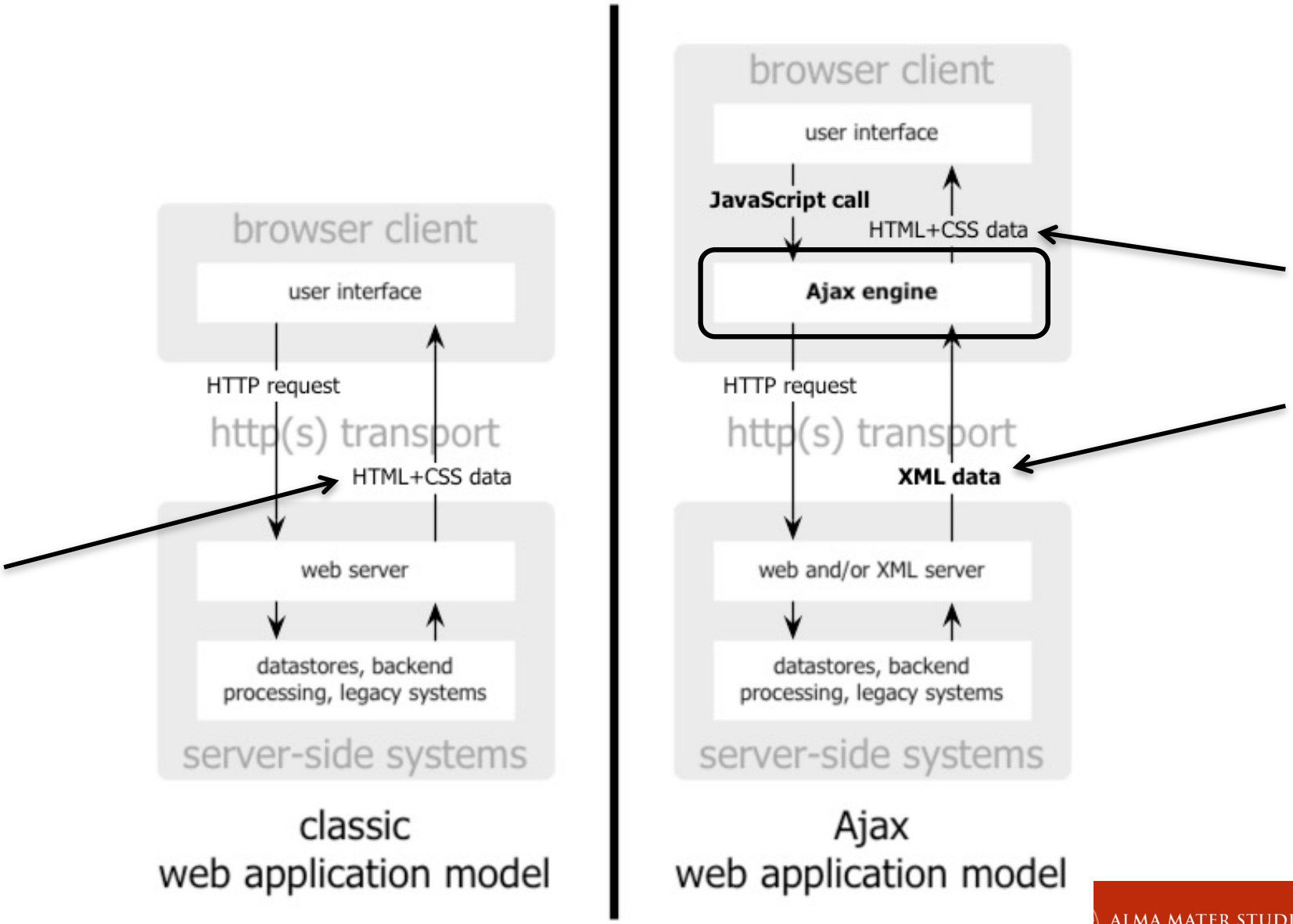
AJAX: Discussione

Non è un linguaggio di programmazione o una tecnologia specifica

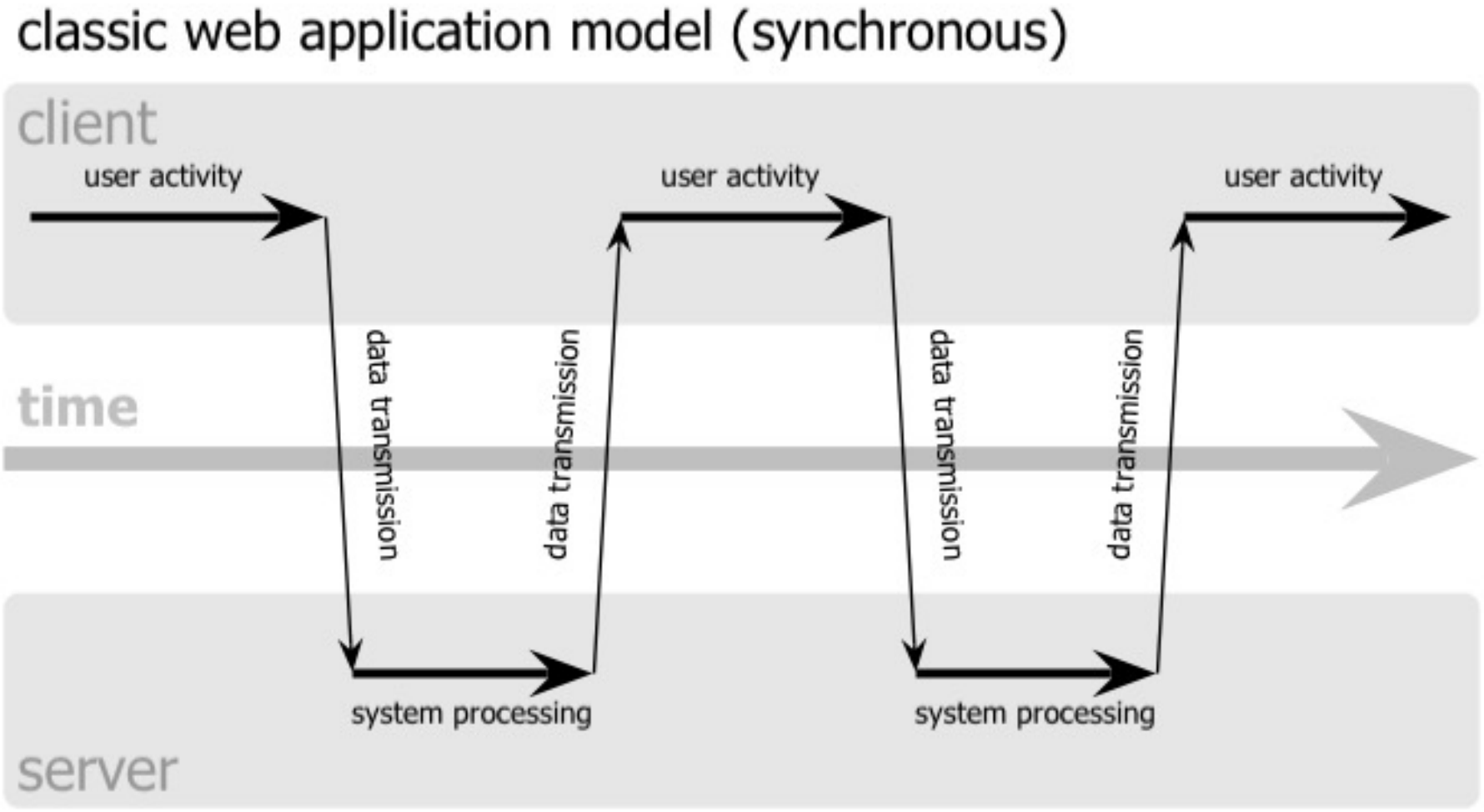
E' un termine che indica l'utilizzo di una combinazione di tecnologie comunemente utilizzate sul Web:

- HTML e CSS
- DOM (modificabile attraverso JavaScript per la manipolazione dinamica dei contenuti e dell'aspetto)
- La libreria XMLHttpRequest (XHR) per lo scambio di messaggi asincroni fra browser e web server
- XML o JSON come meta-linguaggi dei dati scambiati

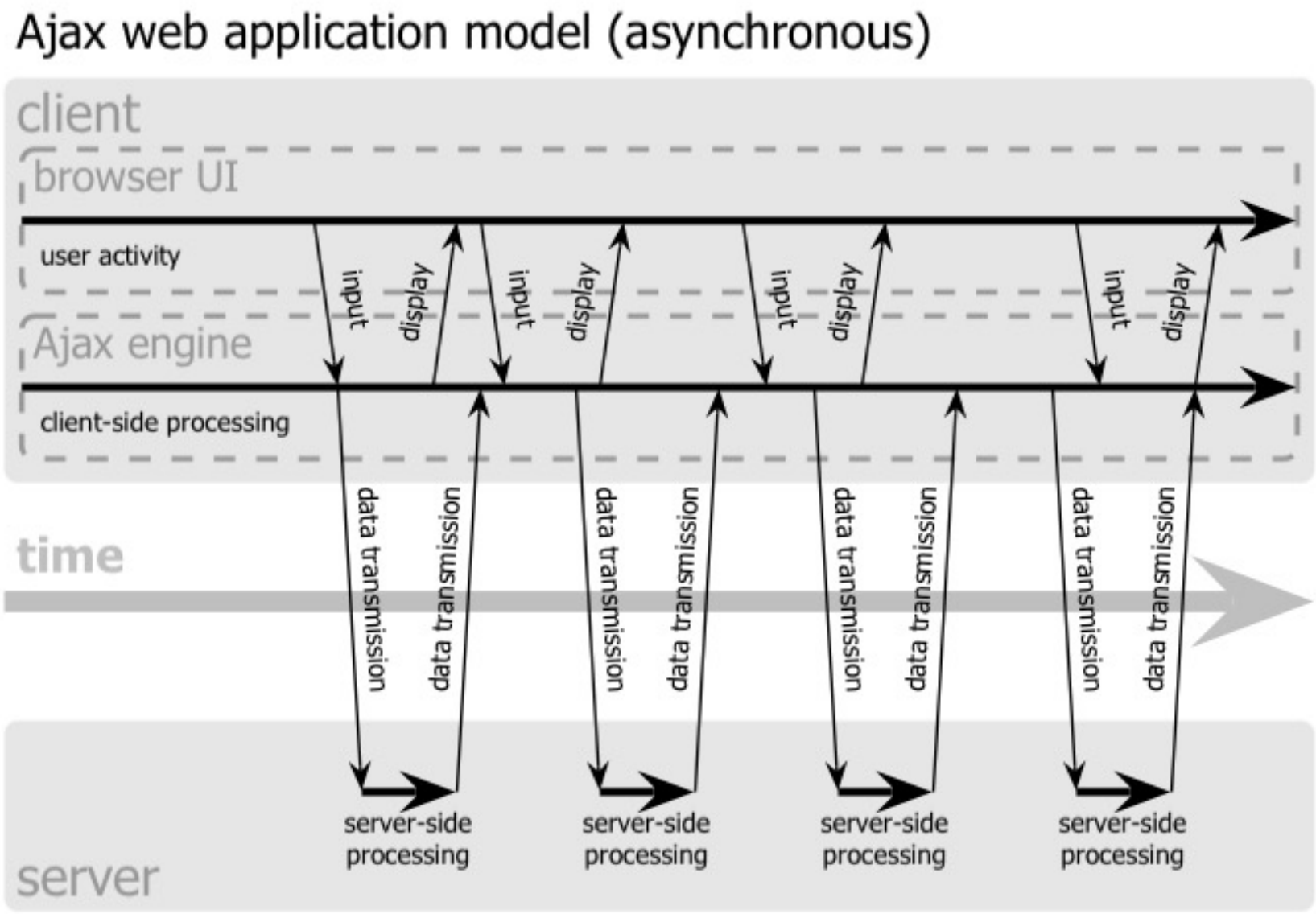
AJAX: Architettura (1)



AJAX: Architettura (2)



AJAX: Architettura (3)



AJAX: Un po’ di storia

Inizialmente sviluppatto da Microsoft (*XMLHttpRequest*) come oggetto ActiveX

In seguito implementato in tutti i principali browser ad iniziare da Mozillla 1.0 sebbene con alcune differenze

Il termine **Ajax** è comparso per la prima volta nel 2005 in un articolo di *Jesse James Garrett*

Ajax: Pregi

- **Usabilità**
  - Interattività (Improve user experience)
  - Non costringe l'utente all'attesa di fronte ad una pagina bianca durante la richiesta e l'elaborazione delle pagine (non più click-and-wait)
- **Velocità**
  - Minore quantità di dati scambiati (non è necessario richiedere intere pagine)
  - Una parte della computazione è spostata sul client
- **Portabilità**
  - Supportato dai maggiori browser
  - Se correttamente utilizzato è platform-independent
  - Non richiede plug-in

Ajax: Difetti

- **Usabilità**
  - Non c'è navigazione: il pulsante “back” non funziona
  - Non c'è navigazione: l'inserimento di segnalibri non funziona
  - Poiché i contenuti sono dinamici non sono correttamente indicizzati dai motori di ricerca
- **Accessibilità**
  - Non supportato da browser non-visuali
  - Richiede meccanismi di accesso alternativi
- **Configurazione**:
  - È necessario aver abilitato Javascript
  - in Internet Explorer è necessario anche aver abilitato gli oggetti ActiveX
- **Compatibilità**:
  - È necessario un test sistematico sui diversi browser per evitare problemi dovuti alle differenze fra i vari browser
  - Richiede funzionalità alternative per i browser che non supportano Javascript

Creare un’applicazione AJAX

Un’applicazione AJAX è divisa in alcuni momenti chiave:

1. Creazione e configurazione delle richieste per il server
   - Usando XMLHTTPRequest
   - *Usando funzioni di libreria che nascondono XMLHTTPRequest*
   - *Usando fetch()*
2. Attivazione della richiesta HTTP…
3. ***… passa del tempo…***
4. … ricezione della risposta HTTP e analisi dei dati (o errore)
5. Modifiche al DOM della pagina

Creazione dell'oggetto XMLHttpRequest

```
if (window.XMLHttpRequest) {    // Mozilla, Safari,...
 http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // Internet Explorer
 try {
  http_request = new ActiveXObject("Msxml2.XMLHTTP");
 } catch (e) {
  try {
   http_request = new ActiveXObject("Microsoft.XMLHTTP");
  } catch (e) {
   …
  }
 }
}
```

Inizializzazione della richiesta (sincrona, asincrona)

La funzione open prepara la connessione HTTP (non viene ancora attivata). Due tipi di connessioni: sincrona e bloccante, oppure asincrona.

```
http_request.open('GET','http://www.example.org/file.json',false);
```

Nel caso di funzione asincrona, prima di attivare la richiesta è necessario specificare la funzione che si occuperà di gestire la risposta e aprire la connessione con il server

```
http_request.onreadystatechange = myfunction;
http_request.open('GET','http://www.example.org/file.json',true);
```

I parametri della 'open' specificano:

- il metodo HTTP della richiesta,
- l'URL a cui inviare la richiesta,
- un booleano che indica se la richiesta è asincrona,
- due parametri opzionali che specificano nome utente e password

Invio della richiesta

La richiesta viene inviata per mezzo di una ‘send’:

```
http_request.send(null);
```

Il parametro della ‘send’ contiene il body della risorsa da inviare al server:

- per una POST ha la forma di una query-string

  ```
  name=value&anothername=othervalue&so=on
  ```

- per un GET ha valore “`null`” (in questo caso i parametri sono passati tramite l’URL indicato della precedente “`open`”)
- può anche essere un qualsiasi altro tipo di dati; in questo caso è necessario specificare il tipo MIME dei dati inviati:

```
http_request.setRequestHeader('Content-Type', 'mime/type');
```

Gestione della risposta (1)

La funzione asincrona incaricata di gestire la risposta deve controllare lo stato della richiesta:

```
function myfunction() {
    if (http_request.readyState == 4) {
        // risposta ricevuta
    } else {
        // risposta non ricevuta ancora
    }
    ...
}
```

I valori per 'readyState' possono essere:

- 0 = uninitialize
- 1 = loading
- 2 = loaded
- 3 = interactive
- 4 = complete

Gestione della risposta (2)

E’ poi necessario controllare lo status code della risposta HTTP:

```
if (http_request.status == 200) {
    // perfetto!
}
else {
    // c'è stato un problema con la richiesta,
    // per esempio un 404 (File Not Found)
    // oppure 500 (Internal Server Error)
}
```

Infine è possibile leggere la risposta inviata dal server utilizzando:

- http_request.responseText che restituisce la risposta come testo semplice
- http_request.responseXML che restituisce la risposta come XMLDocument

Leggere e visualizzare dati (modalità sincrona)

```
function getData(){

  // load the Ajax data
  myXMLHTTPRequest = new XMLHttpRequest();
  myXMLHTTPRequest.open("GET", "file.json", false);
  myXMLHTTPRequest.send(null);

  //legge la risposta
  let d = JSON.parse(myXMLHTTPRequest.responseText);

  // prepara i dati
  var fragment = prepareData(d)

  // modifica il documento corrente
  document.getElementById("area1").appendChild(fragment);
}
```

SINCRONO!

Browser bloccato e non accetta interazione con l'utente

Leggere e visualizzare dati (modalità asincrona)

```
function getData(){
    // load the Ajax data
    myXMLHTTPRequest = new XMLHttpRequest();
    myXMLHTTPRequest.onreadystatechange = myfunction;
    myXMLHTTPRequest.open("GET", "file.json", true);
    myXMLHTTPRequest.send(null);
```

ASINCRONO!

Browser immediatamente libero e accetta interazioni con l'utente

```
  }

  function myfunction() {
    if (myXMLHTTPRequest.readyState == 4) {
      if (myXMLHTTPRequest.status == 200) {
          //legge la risposta
          let d = JSON.parse(myXMLHTTPRequest.responseText);
          // prepara i dati
          var fragment = prepareData(d)
          // modifica il documento corrente
          document.getElementById("area1").appendChild(fragment);
        }
     }
  }
```

Semplificando...

- La complessità di XMLHttpRequest e le differenze di implementazione tra browser e browser hanno portato a suggerire molte alternative:
  - jQuery è stato introdotto ed è diventato famoso anche perché forniva un meccanismo per fare connessioni Ajax molto più semplice anche se ancora basato su callback.
  - Sia React sia Angular introducono librerie interne per fare connessioni Ajax totalmente integrate nel loro framework
  - Con il passaggio da W3C a WhatWG, e con l'introduzione delle promesse, è stata proposta e standardizzata una specifica API nativa, chiamata Fetch, per realizzare connessioni Ajax con una libreria nuova NON basata su XMLHttpRequest.
- Ne parleremo via via...

I framework Ajax

Sono librerie Javascript che semplificano la vita nella creazione di applicazioni Ajax anche complesse.

Hanno tre scopi fondamentali

- ***Astrazione***: gestiscono le differenze tra un browser e l'altro e forniscono un modello di programmazione unico (o quasi) che funziona MOLTO PROBABILMENTE su tutti o molti browser.
- ***Struttura dell'applicazione***: forniscono un modello di progetto dell'applicazione omogeneo, indicando con esattezza come e dove fornire le caratteristiche individuali dell'applicazione
- ***Libreria di widget***: forniscono una (più o meno) ricca collezione di elementi di interfaccia liberamente assemblabili per creare velocemente interfacce sofisticate e modulari

Categorie di framework

- Modello applicativo
  - Framework interni
    - Sono frameworks che vengono usati direttamente dentro alla pagina HTML con programmi scritti in Javascript
  - Framework esterni
    - Sono frameworks usati all'interno di un processo di sviluppo client e server e sono disponibili in un linguaggio di programmazione indipendente da Javascript (ad esempio in Java).
- Ricchezza funzionale
  - Librerie di supporto JavaScript
    - Prototype, jQuery e MooTools sono semplicemente livelli di astrazione cross-browser per task di basso livello DOM-oriented, ad esempio per arricchire graficamente un sito web tradizionale.
  - Framework RIA (Rich Internet Application)
    - Ext, GWT, YUI, Dojo e qooxdoo sono framework ricchi per la creazione di applicazioni complete, e includono una ricca collezione di widget, modelli di comunicazione client e server, funzionalità grafiche e interattive, e spesso anche strumenti di sviluppo.

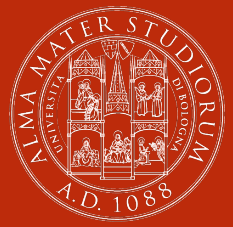
Alcuni framework rilevanti

Librerie Javascript

- Prototype (http://www.prototypejs.org/)
- jQuery (http://jquery.com/)

Rich Internet Application Framework

- Angular (http://angular.io/)
- React (http://reactjs.org/)
- Vue (http://vuejs.org/)



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it

www.unibo.it