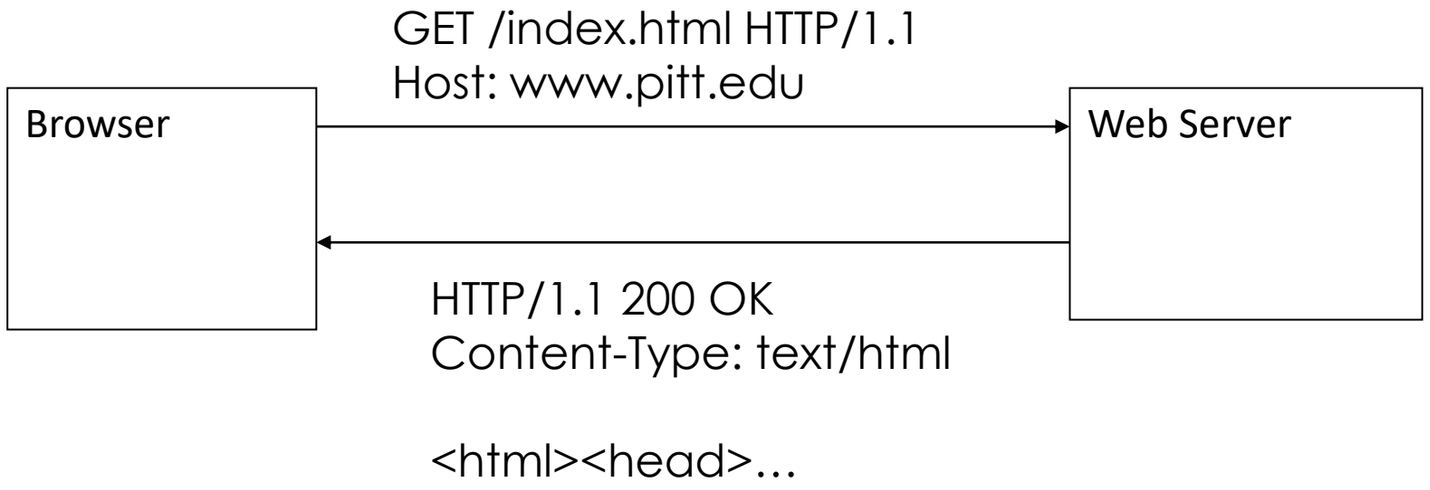
REST

Representational State Transfer

Hypertext Transfer Protocol (HTTP)

- A communications protocol
- Allows retrieving inter-linked text documents (hypertext)
  - World Wide Web.
- HTTP Verbs
  - HEAD
  - **GET**
  - **POST**
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT



Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web.
- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification.
- A collection of network architecture principles which outline how resources are defined and addressed

REST and HTTP

- The motivation for REST was to capture the characteristics of the Web which made the Web successful.
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET
  - HTTP PUT, HTTP DELETE

REST - not a Standard

- REST is not a standard
  - JSR 311: JAX-RS: The Java™ API for RESTful Web Services
- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc (Resource Representations)
  - text/xml, text/html, image/gif, image/jpeg, etc (Resource Types, MIME Types)

Resources

- The key abstraction of information in REST is a resource.
- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on
- Represented with a global identifier (URI in HTTP)
  - http://www.boeing.com/aircraft/747

Naming Resources

- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data

Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE

HTTP GET

- How clients ask for the information they seek.
- Issuing a GET request transfers the data from the server to the client in some representation

- GET http://localhost/books
  - Retrieve all books
- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021
- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP PUT, HTTP POST

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

HTTP DELETE

- Removes the resource identified by the URI

- DELETE http://localhost/books/ISBN-0011
  - Delete book identified by ISBN-0011

Representations

- How data is represented or returned to the client for presentation.
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data

Representations

- XML

```
<COURSE>
        <ID>CS2650</ID>
        <NAME>Distributed Multimedia Software</NAME>
</COURSE>
```

- JSON

```
{course : {
            {id: CS2650}
           {name: Distributed Multimedia Sofware}
        }
}
```

JSON

JSON is the acronym for JavaScript Object Notation.

- **It is a data interchange format, considered much more convenient than XML**
  - It is lightweight in terms of the amount of data exchanged
  - Very easy to process for a programming language, especially JavaScript
  - It is reasonably simple to read for a human operator
- **It is widely supported by major programming languages**
- **It is based on the notation used for object literals and array literals in JavaScript.**

JSON Syntax

- The JSON syntax is based on that of JavaScript object and array literals
- A "JSON object" is a string that is equivalent to a JavaScript object literal, where the keys are enclosed in double quotes "«

```
{
 "Country" : "England", "Year" : 1959, "TypeOfMusic" : "Rock'n'Roll",
 "Components" : ["Paul","John","George","Ringo"]
}
```

JSON Object

```
'{"Country " : "England", "Year" : 1959,
 "TypeOfMusic" : "Rock'n'Roll", "Components" :
  ["Paul","John","George","Ringo"] }'
```

Real Life Examples

- Google Maps
- Google AJAX Search API
- Yahoo Search API
- Amazon WebServices

REST and the Web

- The Web is an example of a REST system!

- All of those Web services that you have been using all these many years - book ordering services, search services, online dictionary services, etc - are REST-based Web services.

References

- Representational State Transfer http://en.wikipedia.org/wiki/Representational_State_Transfer

- Roy Fieldings Thesis http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

AJAX

Asynchronous Javascript And XML

Limiti del modello a ricaricamento di pagina

- Le applicazioni Web tradizionali espongono invece un modello di interazione rigido
  - Modello "**Click, wait, and refresh**"
  - È necessario *refresh della pagina da parte del server* per la gestione di qualunque evento
    (sottomissione di dati tramite form, visita di link per ottenere informazioni di interesse, …)
- È ancora **modello sincrono**: l'utente effettua una richiesta e deve attendere la risposta da parte del server

Un nuovo modello

- L'utilizzo di **DHTML** (JavaScript/Eventi + DOM + CSS) delinea un nuovo modello per applicazioni Web
- => *Modello a eventi simile a quello delle applicazioni tradizionali*
- Abbiamo però due livelli di eventi:
  - **Eventi locali** che portano ad una modifica diretta DOM da parte di Javascript e quindi a cambiamento locale della pagina
  - **Eventi remoti** ottenuti tramite ricaricamento della pagina che viene modificata lato server in base ai parametri passati in GET o POST
- Il ricaricamento di pagina per rispondere a interazione con l'utente prende il nome di **postback**

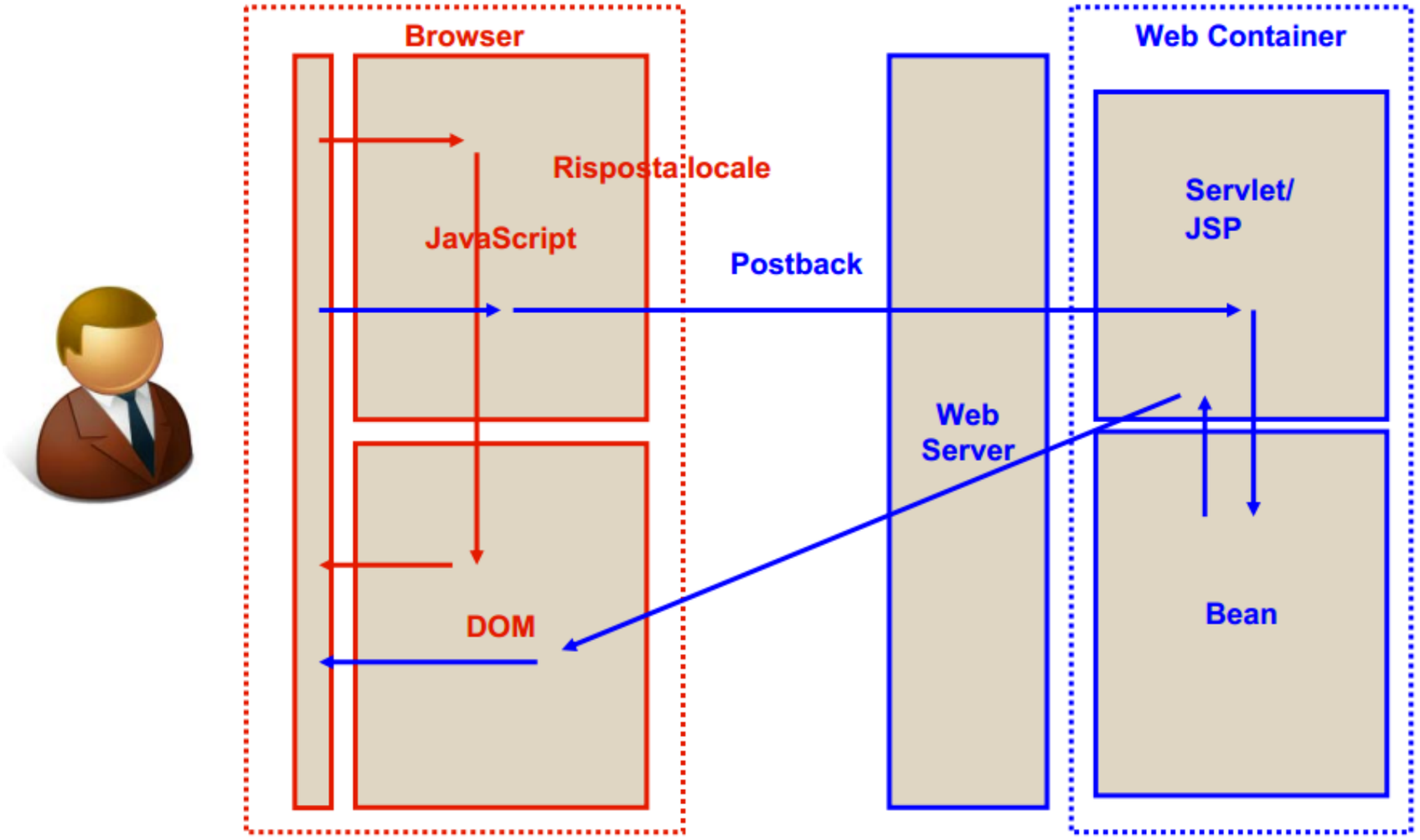
AJAX e asincronicità

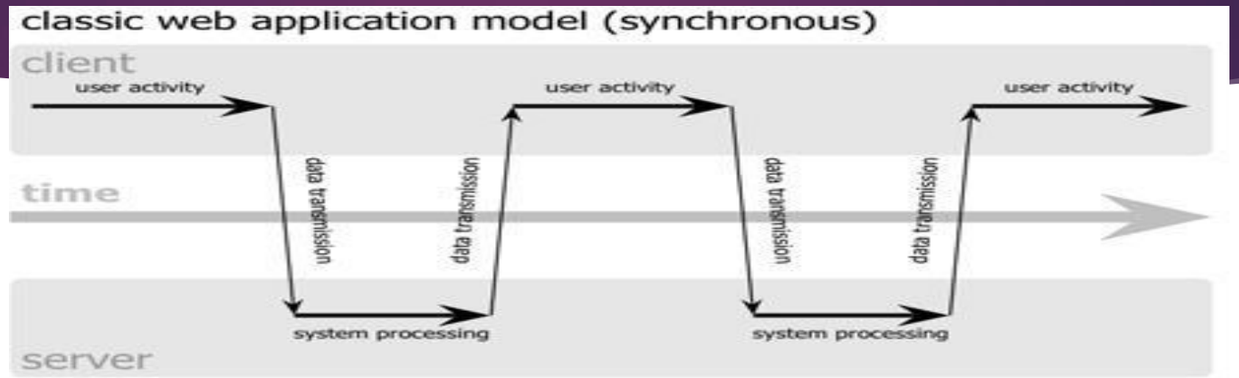
- Il modello **AJAX** è nato per superare queste limitazioni
- AJAX non è un acronimo ma spesso viene interpretato come **A**synchronous **J**avascript **A**nd **X**ml
- È basato su tecnologie standard:
  - JavaScript
  - DOM
  - XML
  - HTML
  - CSS

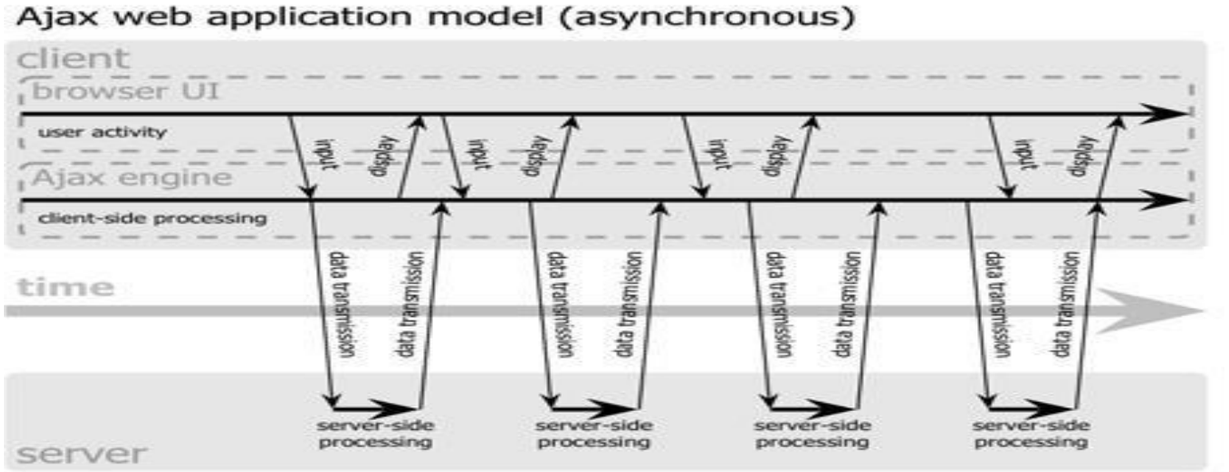
AJAX and asynchrony

- **AJAX aims to support user-friendly applications with high interactivity (the term RIA - Rich Interface Application is often used)**
- **The underlying idea of AJAX is to enable JavaScript scripts to interact directly with the server.**
- **The JavaScript XMLHttpRequest object is used.**
  - It allows obtaining data from the server without the need to reload the entire page
  - It achieves asynchronous *communication* between the client and server: the client doesn't interrupt the interaction with the user even when it's waiting for responses from the server.

Modello a eventi a due livelli




classic web application model (synchronous)
client
user activity
user activity
user activity
data transmission
data transmission
data transmission
data transmission
time
system processing
system processing
server


Ajax web application model (asynchronous)
client
browser UI
user activity
input
display
input
display
input
display
input
display
Ajax engine
client-side processing
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
data transmission
time
server-side processing
server-side processing
server-side processing
server-side processing
server

AJAX Technologies

- HTML
  - Used to build web forms and identify fields
- Javascript
  - Facilitates asynchronous communication and modification of HTML in-place
- DOM - Document Object Model
  - Used via Javascript to work with both the structure of your HTML and also XML from the server

A typical sequence of AJAX

- **An event is triggered by the interaction between the user and the web page**
- **The event involves the execution of a JavaScript function in which:**
  - An **XMLHttpRequest** object is instantiated
  - The **XMLHttpRequest** is configured, including associating a callback function and making configurations, …
  - An asynchronous call to the server is made
- **The server processes the request and responds to the client**
- **The browser invokes the callback function, which:**
  - processes the result
  - Updates the page's Document Object Model (DOM) to display the processing results

XMLHttpRequest

- L'oggetto `XMLHttpRequest` effettua la richiesta di una risorsa via HTTP a server Web
  - Non sostituisce l'URI della propria richiesta all'URI corrente
  - Non provoca un cambio di pagina
  - Può inviare eventuali informazioni (parametri) sotto forma di variabili (come una form)
- Può effettuare sia richieste GET che POST
- Le richieste possono essere di tipo
  - **Sincrono:** blocca flusso di esecuzione del codice Javascript (ci interessa poco)
  - **Asincrono:** non interrompe il flusso di esecuzione del codice Javascript né le operazioni dell'utente sulla pagina

AJAX Libraries

- Prototype
  - http://www.prototypejs.org/
- Scriptaculous
  - http://script.aculo.us/
- JQuery
  - http://jquery.com/
- Mochikit
  - http://mochikit.com/

Prototype Sample

```
new Ajax.Request('/some_url',   {  method:'get',
onSuccess: function(transport)
{
var response = transport.responseText || "no response text";  alert("Success!
            \n\n" + response);
},
onFailure: function()
{
alert('Something went wrong...')
}
});
```

Prototype Example

```
<html>
<head>
	<title>Testing Prototype</title>
	<script src="http://www.prototypejs.org/assets/2008/9/29/prototype-1.6.0.3.js"></script>

	<script>
	function getProducts()
	{
		new Ajax.Updater('products', 'products.html', { method: 'get' });
	}
	</script>
</head>
<body>
<h2>Our fantastic products</h2>
<div id="products"><a href = "#" onClick="getProducts();">(fetch product list ...)</a></div>
</body>
</html>
```

AJAX in JQuery

- Simplified
- $.ajax(url, [settings])
  - url : a string containing the url - optional
  - settings : key-value pairs
  - Returns jqXHR object (superset of XMLHTTPRequest object)
- Example:

```
$.ajax({
  url: "some.php",
  type: "POST",
  data: { name: "John", location: "Boston" }
});
```

The jqXHR Object

- Superset of the XMLHTTPRequest
- Contains responseText and responseXML properties and getResponseHeader() method
- Other functions
  - jqXHR.done(function(data,textStatus, jqXHR){})
  - jqXHR.fail(function(jqXHR, textStatus, errorThrown){})
  - jqXHR.always(function(data, textStatus, error){})

AJAX & the jqXHR Object

```
var jqxhr = $.ajax( "example.php" )
        .done(function() {  alert( "success"    );})
        .fail(function() {  alert( "error" );})
        .always(function() {  alert( "complete" );});
```

AJAX in JQuery

- $.get(url [, data] [, success(data,textStatus, jqXHR){})

```
$.get( "ajax/test.html", function( data ) {
   document.getElementByClassName("result").innerHTML = data;
   alert( "Load was performed." );
});
```

- $.post(url [, data] [, success(data,textStatus, jqXHR){})

```
$.post( "ajax/test.html", postdata, function( data ) {
   document.getElementByClassName("result").innerHTML = data;
});
```

- $.getJSON(url [, data] [, success(data,textStatus, jqXHR){})
  - Use an AJAX get request to get JSON data

From JSON String to an Object

- **One of the most widely used parsers is the one provided by the website www.json.org**
  - http://www.json.org/json.js

- **JavaScript exposes the JSON object with two methods:**
  - **`JSON.parse(strJSON)`**: converts a JSON string into a JavaScript object.
  - **`JSON.stringify(objJSON)`**: converts a JavaScript object into a JSON string.

JSON and AJAX - 1

On the client side:

- Create a JavaScript object and populate its properties with the necessary information
- Use **`JSON.stringify()`** to convert the object into a JSON String.

JSON and AJAX - 2

On the server side:

- The JSON string is decoded and transformed into a Java object using a specific parser (**GSON, Jackson, or others**)
- You process the object
- You create a new Java object containing the response data
- You transform the Java object into a JSON string using the parser
- You transmit the JSON string to the client in the body of the HTTP response:

```
response.out.write(strJSON);
```

- **This process is automated when using Spring Boot's Rest Controllers.**

Example with Gson

```
public class Car {
    public String brand = null;
    public int    doors = 0;
}
```

```
Car car = new Car();
car.brand = "Rover";
car.doors = 5;

Gson gson = new Gson();

String json = gson.toJson(car);
```

```
String json = "{\"brand\":\"Jeep\", \"doors\": 3}";

Gson gson = new Gson();

Car car = gson.fromJson(json, Car.class);
```

JSON and AJAX - 3

On the client side, upon receiving:

- You convert the JSON string into a JavaScript object using `JSON.parse()`
- You freely use the object for the intended purposes.