RICH INTERNET APPLICATION : AJAX

JALPA MEHTA
ASSISTANT PROFESSOR
IT DEPARTMENT

Web 2.0: Evolution Towards a Read/Write Platform

Web 1.0 (1993-2003)		Web 2.0 (2003- beyond)
Pretty much HTML pages viewed through a browser		Web pages, plus a lot of other "content" shared over the web, with more interactivity; more like an application than a "page"
"Read"	**Mode**	**"Write" & Contribute**
"Page"	**Primary Unit of content**	**"Post / record"**
"static"	**State**	**"dynamic"**
Web browser	**Viewed through…**	**Browsers, RSS Readers, anything**
"Client Server"	**Architecture**	**"Web Services"**
Web Coders	**Content Created by…**	**Everyone**
	Domain of…	

Four design essentials of a Web 1.0 site include:

1. Static pages.
2. Content is served from the server's file-system.
3. Pages built using Server Side Includes or Common Gateway Interface (CGI).
4. Frames and Tables used to position and align the elements on a page.

Five major features of Web 2.0 –

1. Free sorting of information, permits users to retrieve and classify the information collectively.
2. Dynamic content that is responsive to user input.
3. Information flows between site owner and site users by means of evaluation & online commenting.
4. Developed APIs to allow self-usage, such as by a software application.
5. Web access leads to concern different, from the traditional Internet user base to a wider variety of users.

Rich Internet Application

RIA

- RIA is a Web application that has many of the characteristics of desktop application software
- It may allow the user interactive features such as drag and drop, background menu, WYSIWYG editing.
- Modern tools for complex application screens including – Several fonts,bit map and vector grahics file, animations, audio, video etc.
- Such application which can be served on web called Rich Internet Application

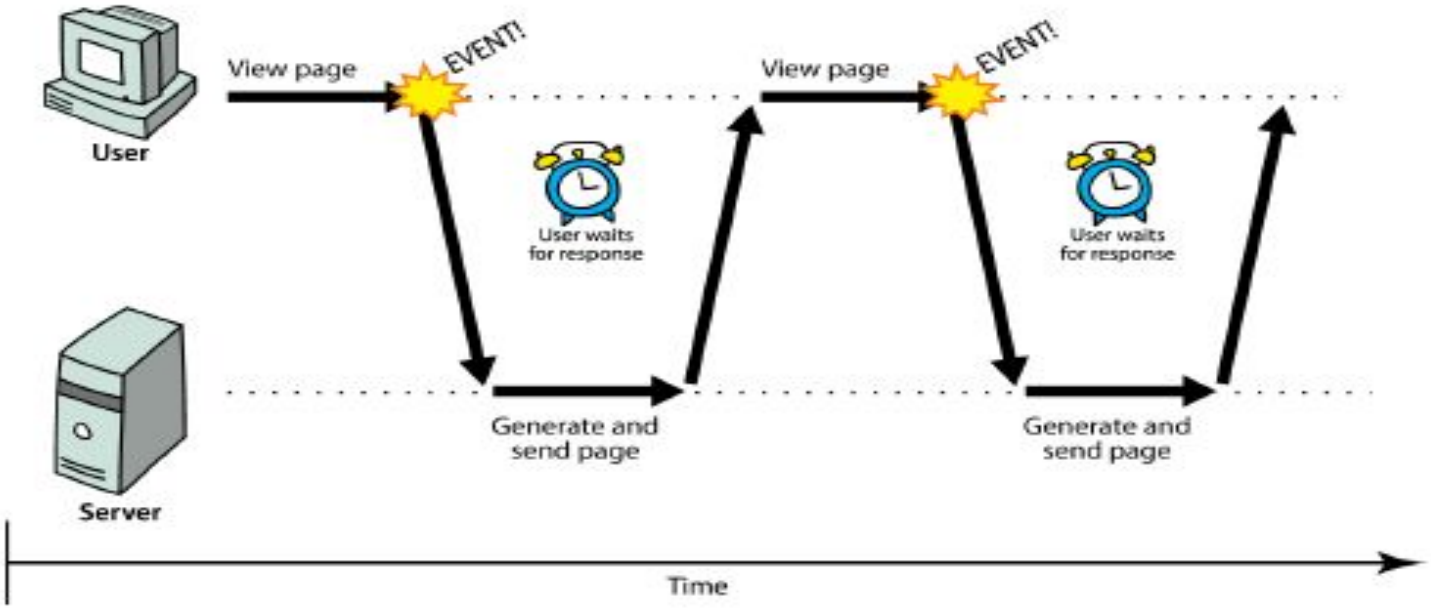
RIA

- RIAs split the processing across the Internet divided by locating the UI and related activity runs on client side and data manipulation and operation on server side.
- RIA normally runs on Web Browsers
- Browser compatibility issues
- RIA runs client portion within isolated area called a SANDBOX.
- RIA is the client engine that intermediates between the user and the application sever.

Characteristics of RIA

- Direct interaction
- Partial Page updating
- Better Feedback
- Consistency of look and feel
- Offline use
- Performance impact

Synchronous web communication



- synchronous: user must wait while new pages load
  - the typical communication pattern used in web pages (click, wait, refresh)

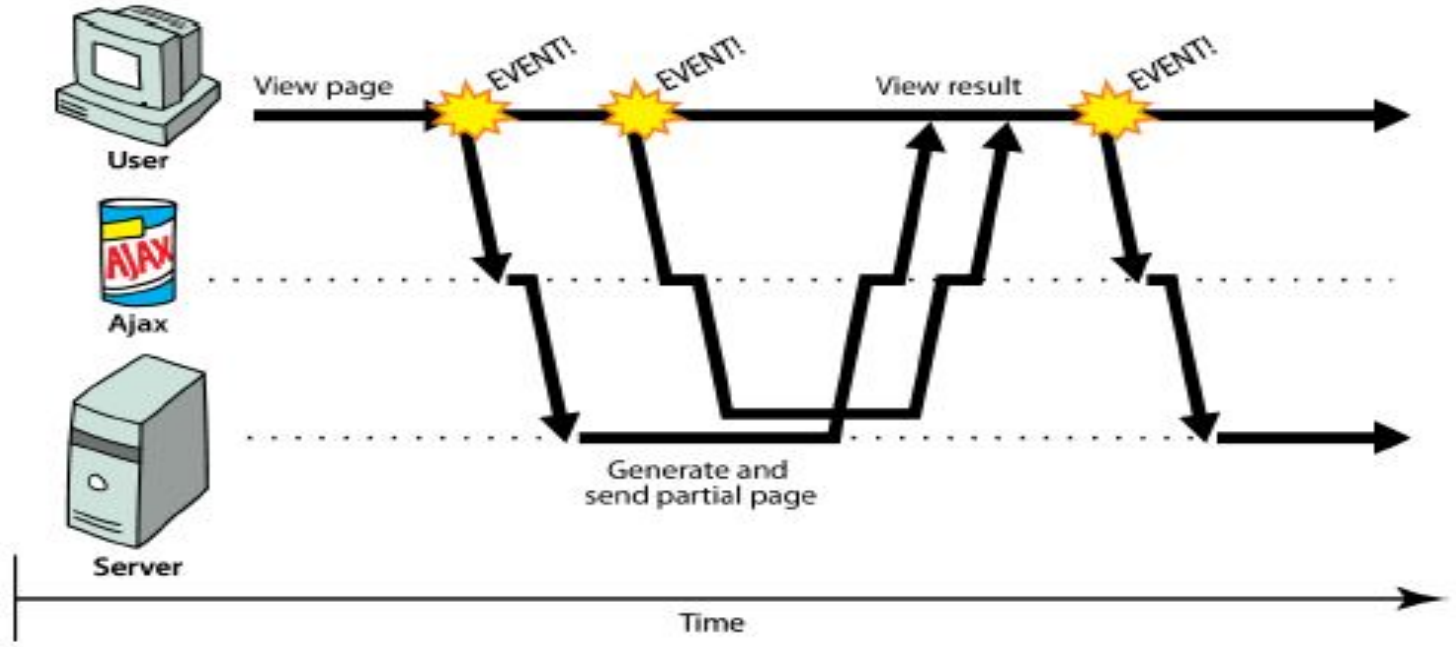
Web applications and Ajax

- **web application:** a dynamic web site that mimics the feel of a desktop app
  - presents a continuous user experience rather than disjoint pages
  - examples: Gmail, Google Maps, Google Docs and Spreadsheets, Flickr, A9

Web applications and Ajax

- **Ajax**: Asynchronous JavaScript and XML
  - not a programming language; a particular way of using JavaScript
  - downloads data from a server in the background
  - allows dynamically updating a page without making the user wait
  - avoids the "click-wait-refresh" pattern
  - Example: Google Suggest

Asynchronous web communication



- **asynchronous:** user can keep interacting with page while data loads
  - communication pattern made possible by Ajax

`XMLHttpRequest` (and why we won't use it)

- JavaScript includes an XMLHttpRequest object that can fetch files from a web server
  - supported in IE5+, Safari, Firefox, Opera, Chrome, etc. (with minor compatibilities)
- it can do this asynchronously (in the background, transparent to user)
- the contents of the fetched file can be put into current web page using the DOM

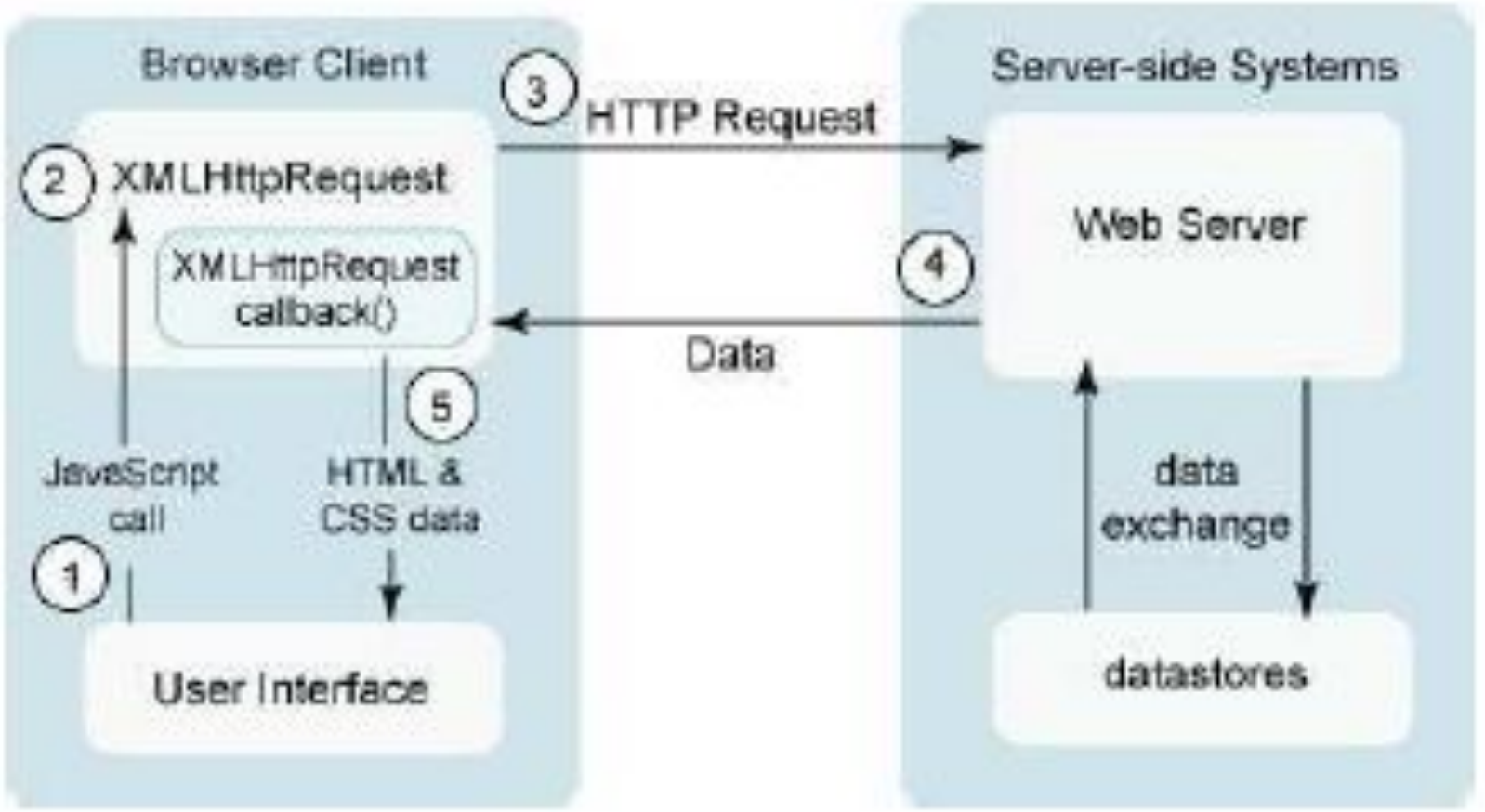
`XMLHttpRequest` (and why we won't use it)

- sounds great!...
- ... but it is clunky to use, and has various browser incompatibilities
- Prototype provides a better wrapper for Ajax, so we will use that instead

A typical Ajax request

1. user clicks, invoking an event handler
2. handler's code creates an `XMLHttpRequest` object
3. `XMLHttpRequest` object requests page from server
4. server retrieves appropriate data, sends it back
5. `XMLHttpRequest` fires an event when data arrives
   - this is often called a callback
   - you can attach a handler function to this event
6. your callback event handler processes the data and displays it

A typical Ajax request



XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *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()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

See AJAX Examples shared in teams