



# Introduction to AJAX

Bringing Interactivity & Intuitiveness Into Web Applications

By : Bhanwar Gupta  
SD-Team-Member  
Jsoft Solutions



# Applications today

- **You have two basic choices: Desktop applications and Web applications**

## Desktop applications

- ✓ It usually come on a CD and install completely on your computer.
- ✓ The code that runs these applications resides on your desktop.
- ✓ It is usually pretty fast, has great user interfaces and is incredibly dynamic.

## Web applications

- ✓ It runs on a Web server and you access the application with your Web browser
- ✓ It provides services you could never get on your desktop  
(think about Amazon.com and eBay).
- ✓ waiting for a server to respond, waiting for a screen to refresh,  
waiting for a request to come back and generate a new page.

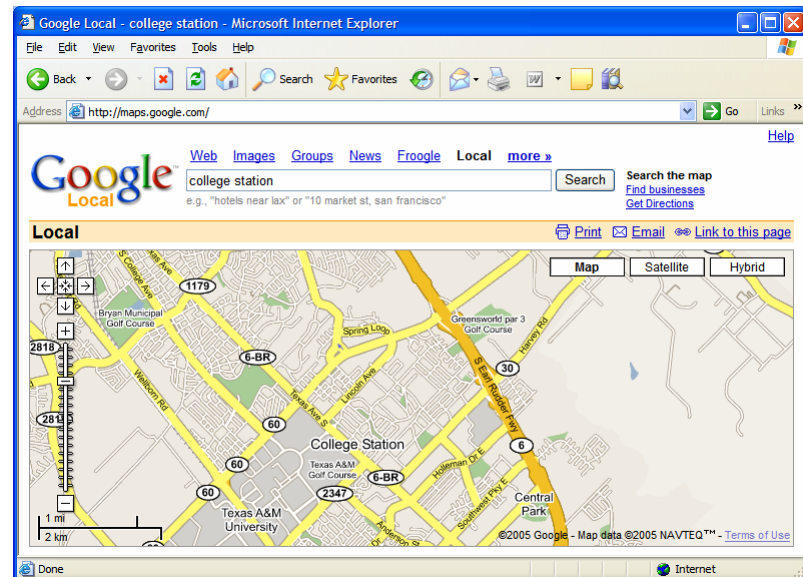
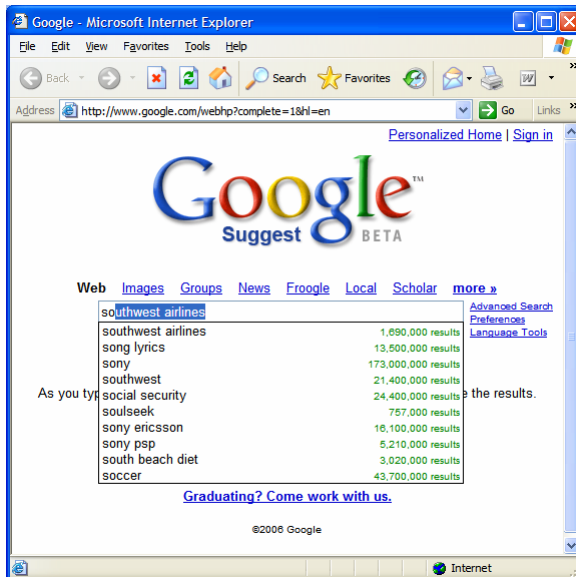
☺ Ajax attempts to bridge the gap between the functionality and interactivity

# What is AJAX ?

- A Web development technique for creating interactive web applications

## Intension

- ✓ Shift a great deal of interaction to the Web surfer's computer
- ✓ Used to retrieve data and update selected parts of the page without refreshing the entire page
- ✓ Example: Google Suggest, Google Maps



☺ Increase the Web page's interactivity, speed, and usability; better user experience

# The "birth" of Ajax

## THE WALL STREET JOURNAL.

© Dow Jones & Company, Inc. All Rights Reserved.

VOL. CCXXXIII NO. 116 E/P/R \*\*\*

THURSDAY, APRIL 1, 2004

Internet Address: <http://wsj.com>

*3/31/05: Google Pioneers Use of Old Microsoft Tools in New Web Programs*

# “Ajax”

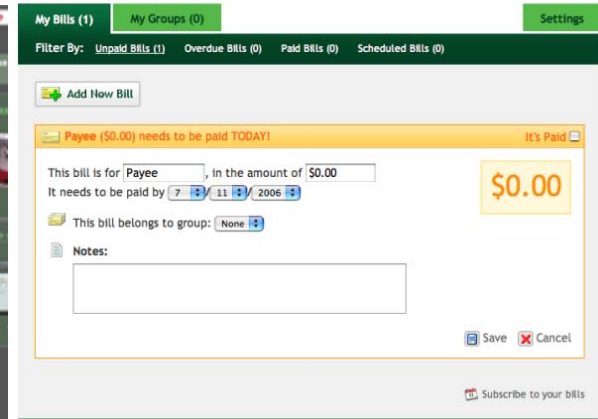
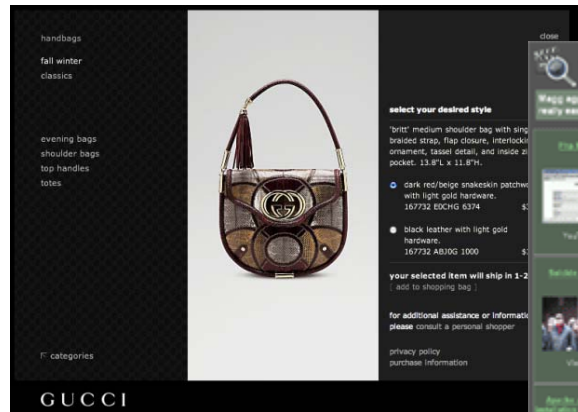
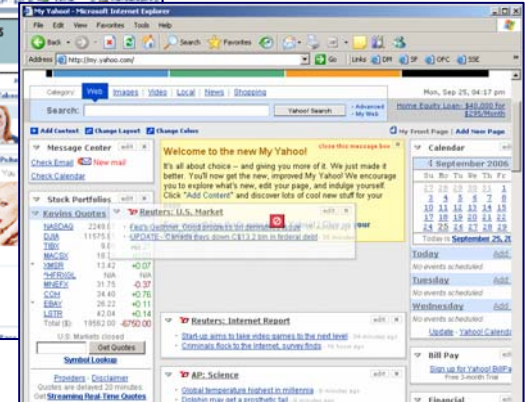
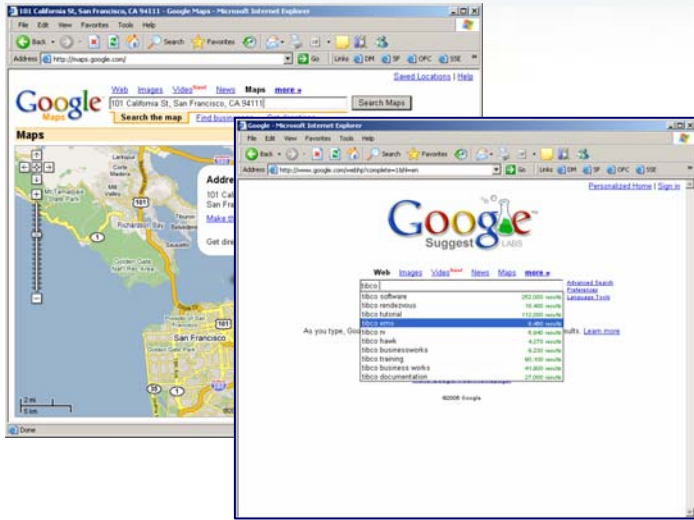
“a big step toward  
of having the kind  
responsiveness  
programs that's  
only with desktop

5%. And according to a press release, Adam Smith

The screenshot shows the Google Maps interface from 2004. The search bar contains 'starbucks' and the location 'alexandria, va'. The search results show 'Starbucks Coffee: Alexandria' at 532 King St. A map on the right displays the location with a red pin. The interface includes navigation controls, a search bar, and a list of results.

The screenshot shows the Flickr website interface. It features a calendar view for April 2006, with a grid of photo thumbnails for each day. A pop-up window displays a selection of photos from April 22nd, 2006, including a cow, a panda, and a jellyfish. The interface includes navigation links, a search bar, and a list of results.

# Ajax Enriched HTML Pages



# Google™

Suggest BETA

[Web](#) [Images](#) [Groups](#) [News](#) [Froogle](#) [Local](#) [Desktop](#) [more »](#)

ajax	
<b>ajax</b>	<b>3,840,000 results</b>
ajax amsterdam	502,000 results
ajax fc	710,000 results
ajax ontario	275,000 results
ajax grips	8,860 results
ajax football club	573,000 results
ajax public library	40,500 results
ajax football	454,000 results
ajax soccer	27,000 results
ajax pickering transit	1,700 results

[Advanced Search](#)  
[Preferences](#)  
[Language Tools](#)

As you type, Google

results. [Learn more](#)

Wow..I didn't know soccer teams did web programming..

# Technologies

- **You might already know the components of Ajax**

XHTML (or HTML) and CSS

- ✓ Marking up and styling information

DOM accessed with a client-side scripting language

- ✓ Dynamically display and interact with the information presented

XMLHttpRequest object

- ✓ Exchange data asynchronously with the web server

XML

- ✓ Format for transferring data

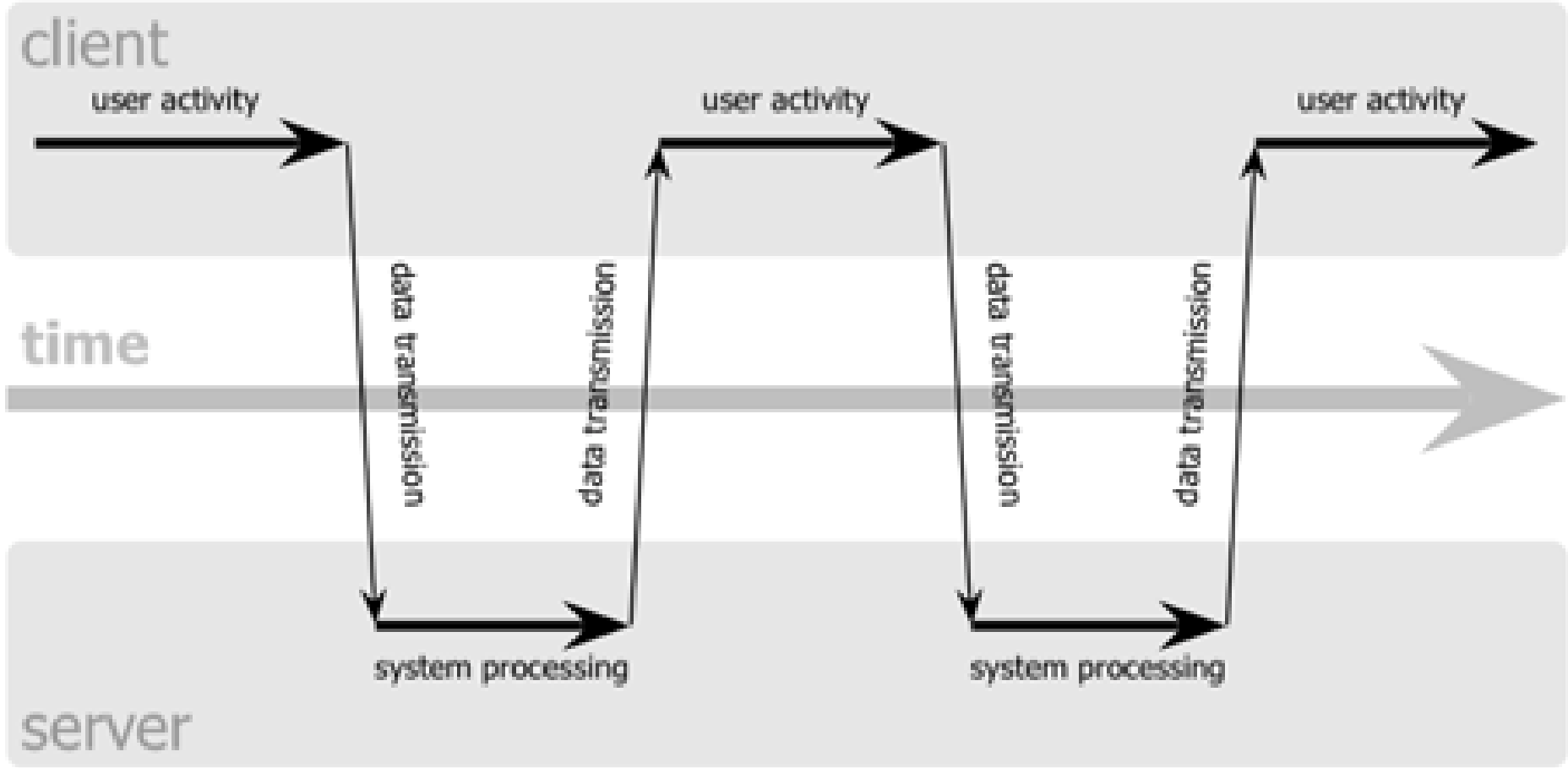
# Why Do I Care About AJAX?

- **Enables building Rich Internet Applications (RIA)**
- **Allows dynamic interaction on the Web**
- **Improves performance**
- **Real-time updates**
- **No plug-ins required**



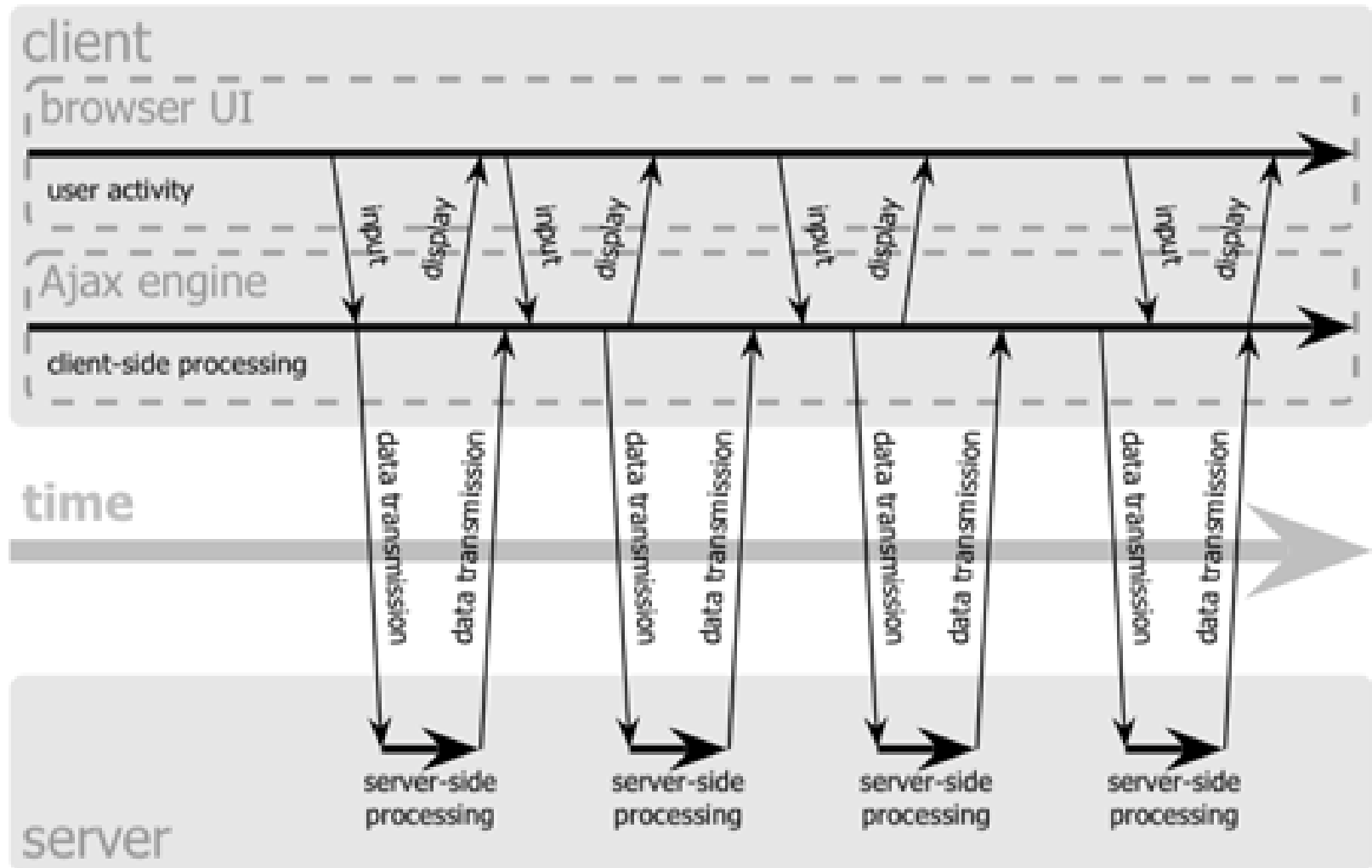
# Classic Web Application Model

classic web application model (synchronous)

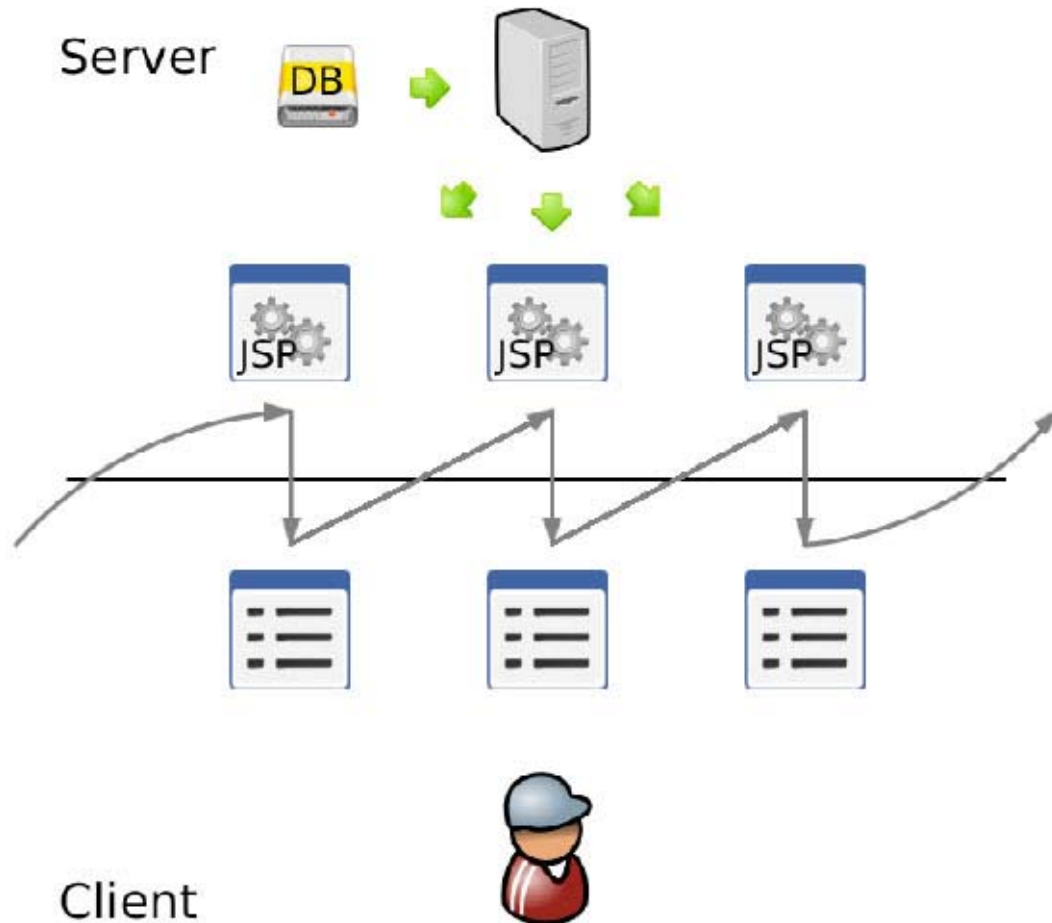


# Ajax web application model

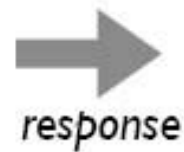
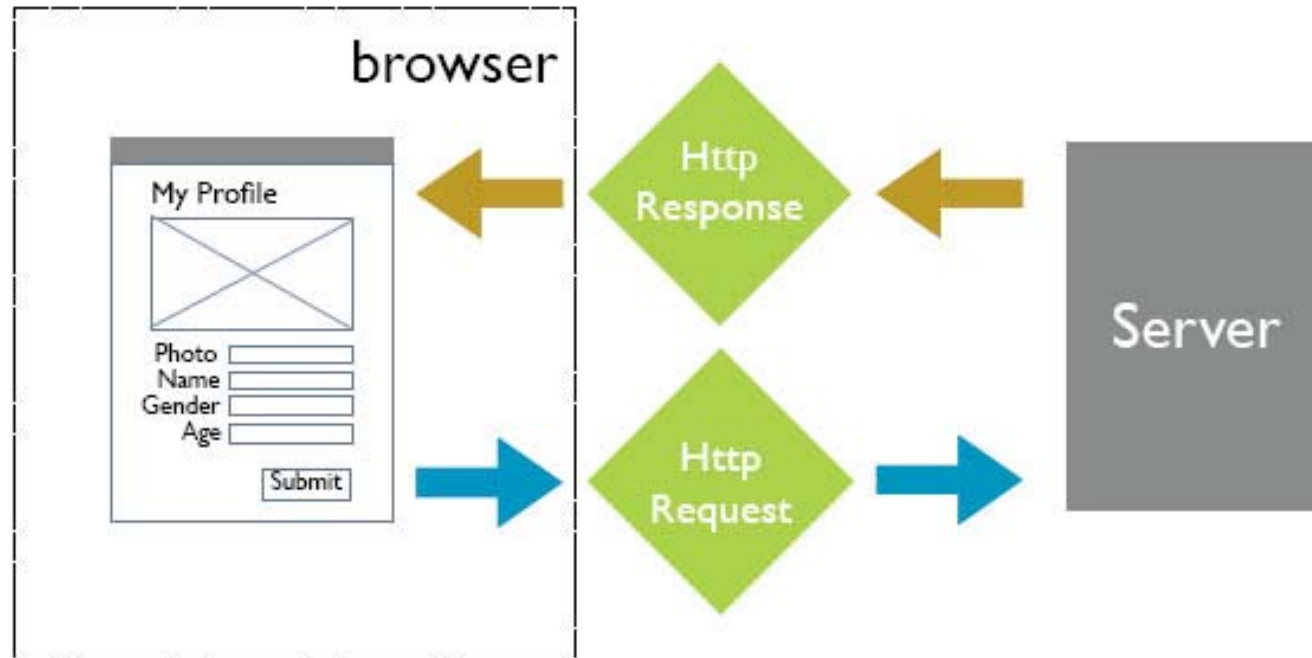
## Ajax web application model (asynchronous)



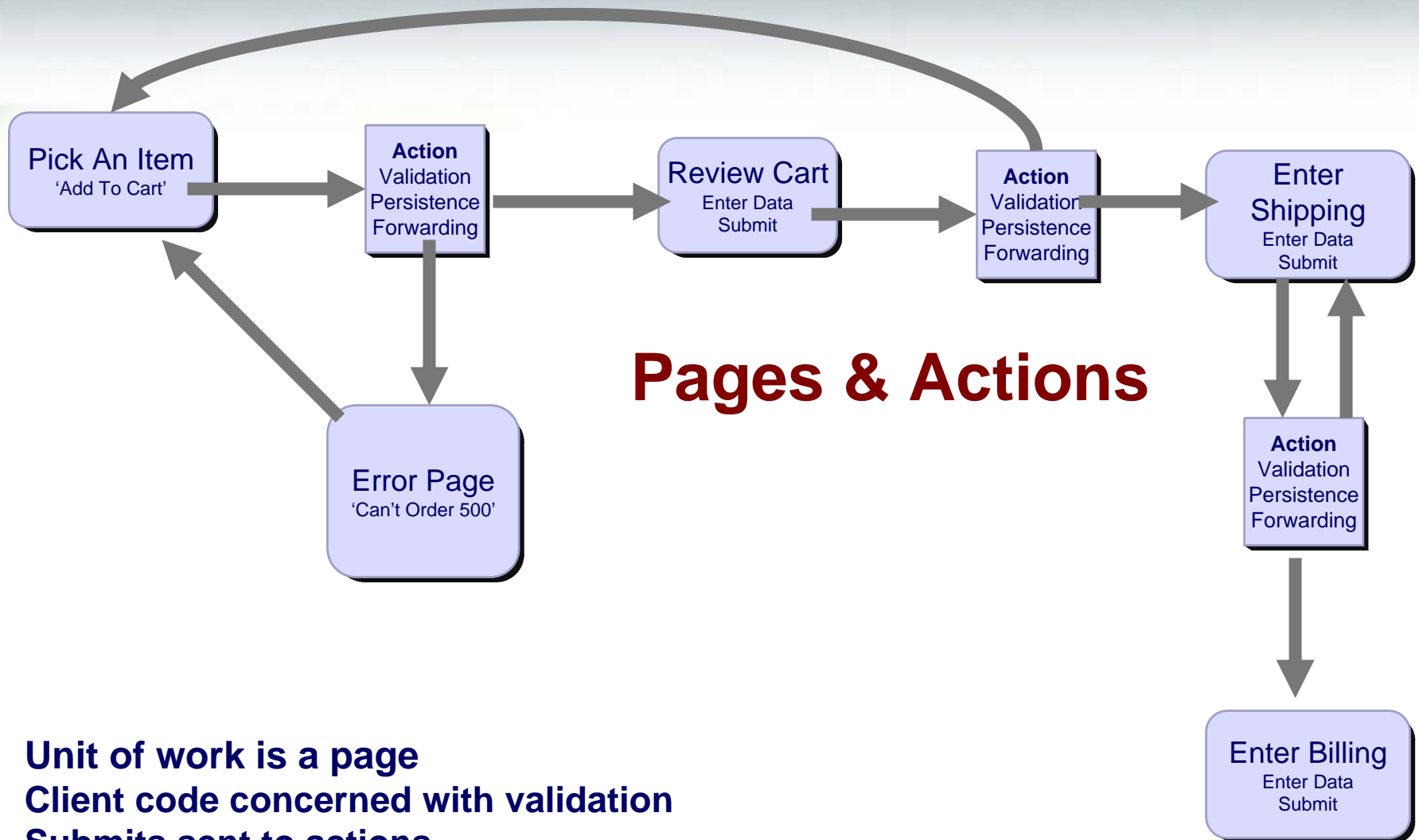
# Traditional Server-Client Architecture



# Traditional server response



# Traditional Web Applications...



## Pages & Actions

**Unit of work is a page**

**Client code concerned with validation**

**Submits sent to actions**

**Actions perform work and then forward to next page**

**Page refresh for each submit**

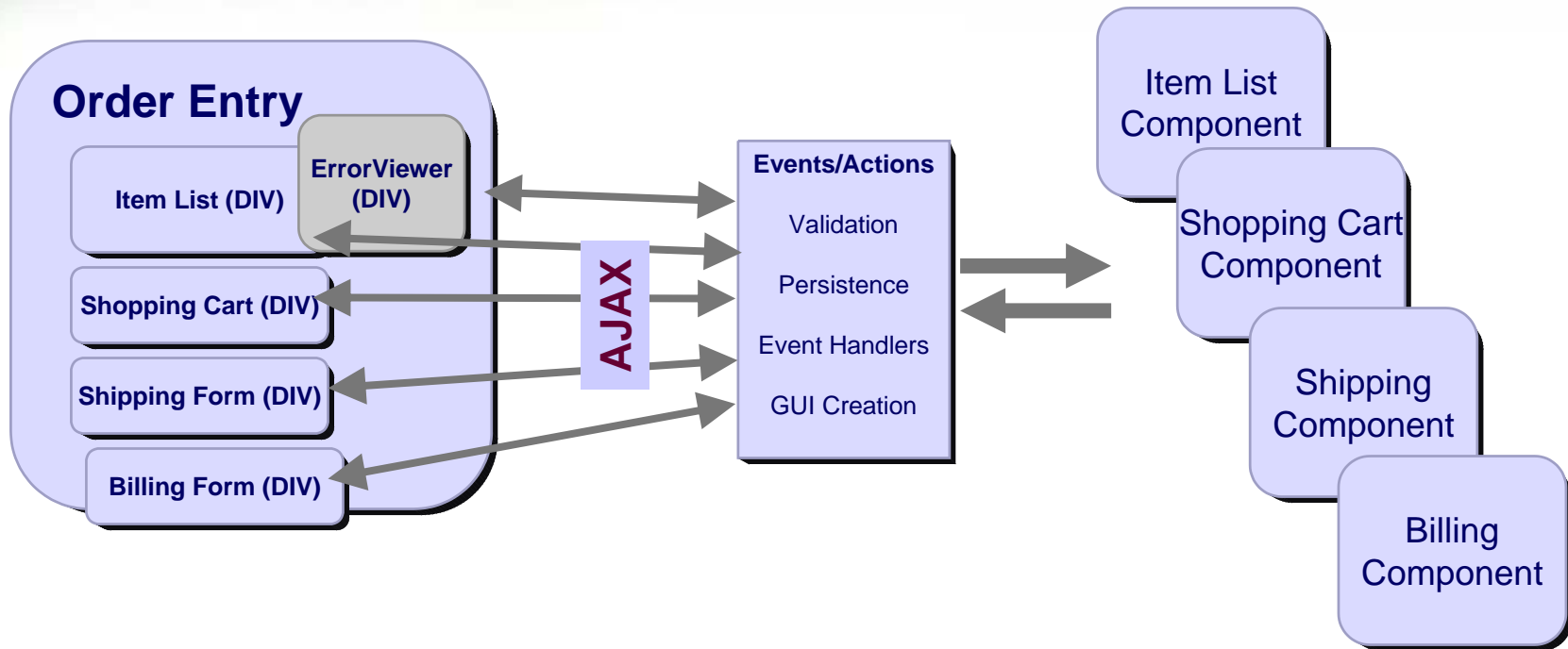
## Ajax Model - New & Improved!

- Now with *Deeper Interaction!*
- Now with *Just-in-Time Data & Just-in-Time Logic!*
  - ★ XMLHttpRequest (XHR) is the secret sauce!
- Now with *Richer Interface!*
- *All dimensions are closer*



# AJAX Changes How Web Apps are Built

## Components & Events



Unit of work is a component

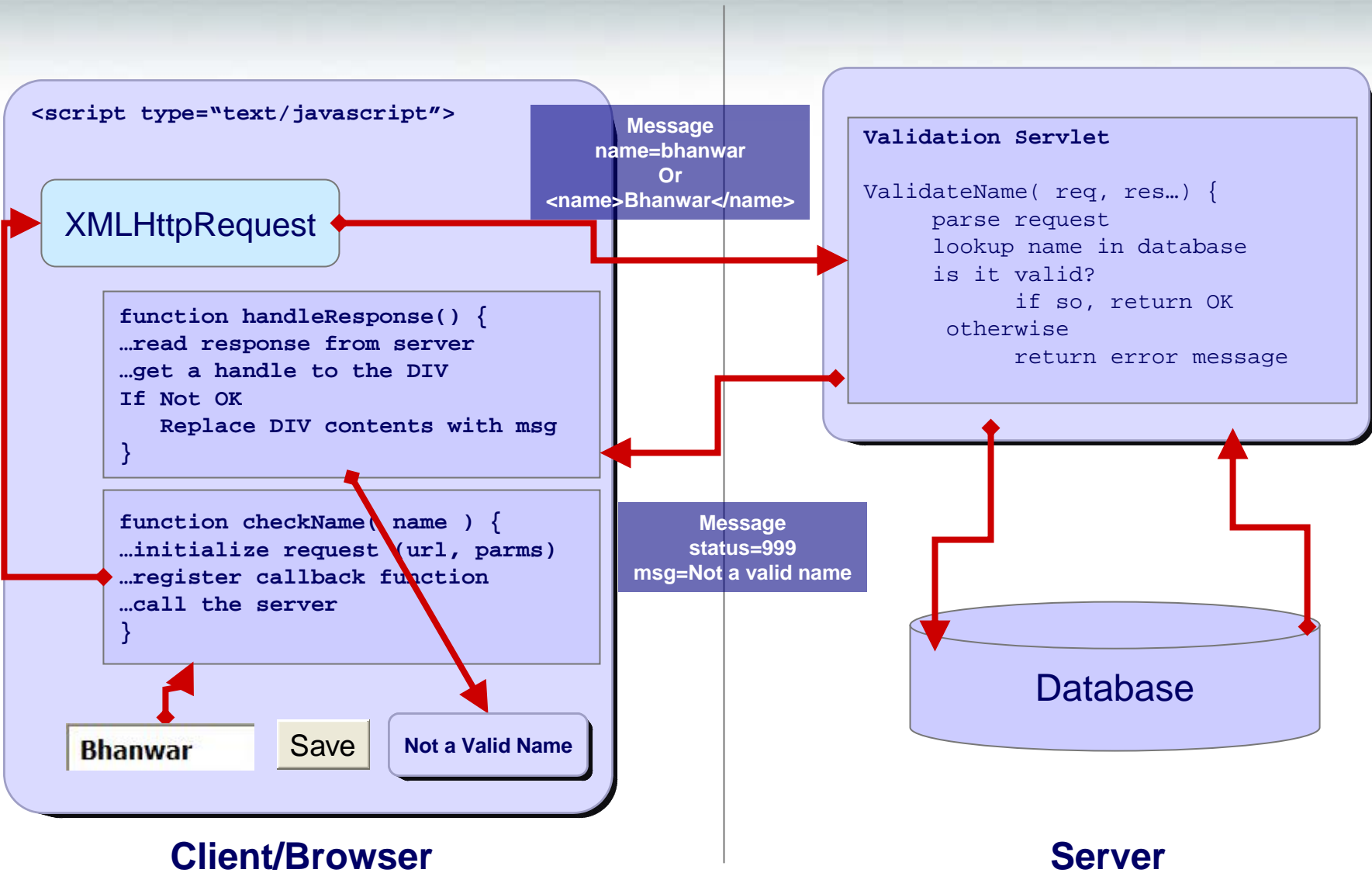
Three-Tier Client/Server Model

Client code has validation, flow, layout, data interchange

No submit buttons—save buttons

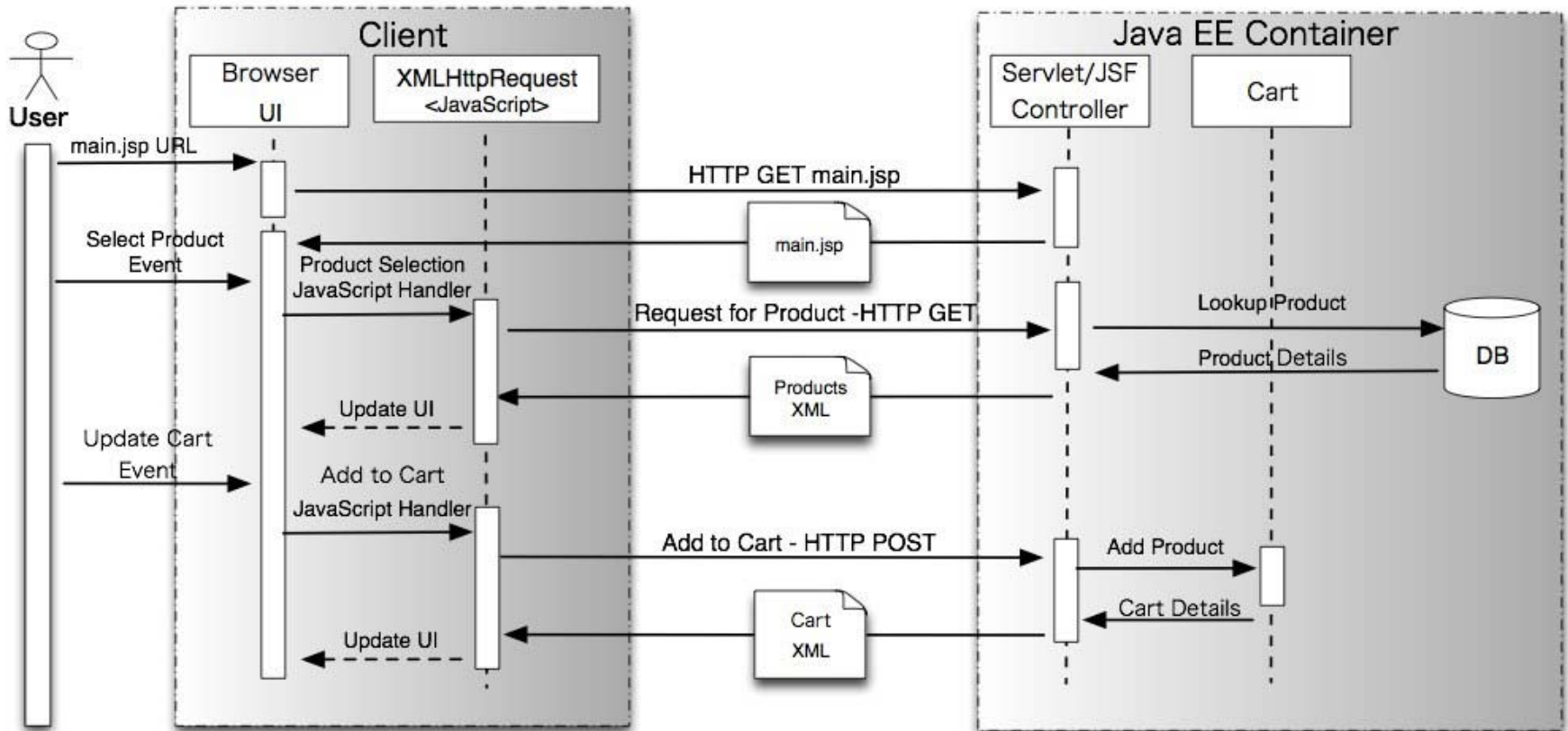
Only parts of pages are updated at a time

# AJAX -- Message Flow



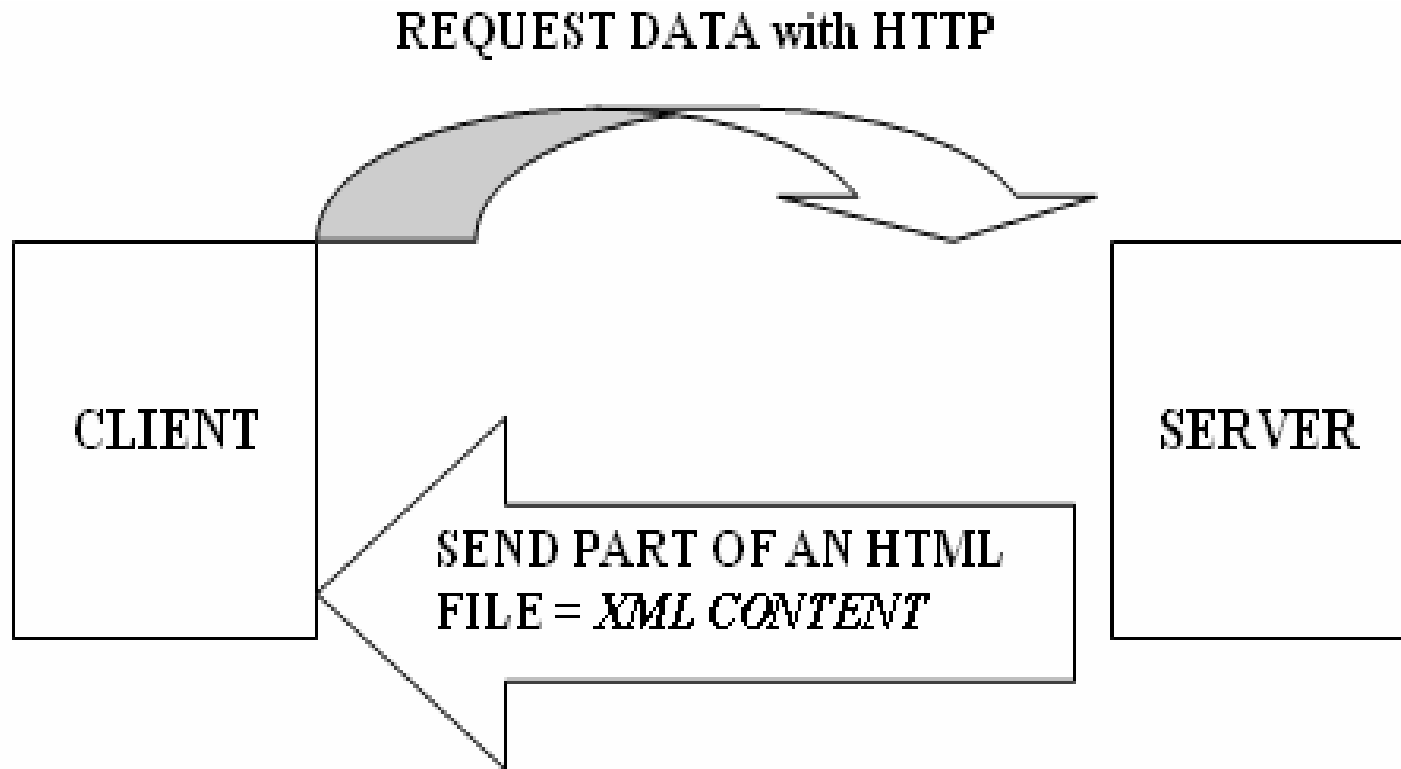


# AJAX -- Client-Server Communication



# Data Exchange in AJAX

- In AJAX:



*Data Exchange in AJAX*

# Ajax Alternatives

	XUL	XAML	SVG	Flash	Applets	Ajax
Desktop-like UI						
Platform Independence	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Vendor Independence				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skill Set Transferrance						<input checked="" type="checkbox"/>



# The Technologies



- **DHTML – HTML + JavaScript + CSS**
- **AJAX – DHTML + XML**
- **Flash**
- **Browser Extensions/Plug-ins**
- **Java**
- **ActiveX**
- **Others not to be discussed**
  - XUL, CURL, etc.
- **Backend Frameworks**

# Pure Browser: DHMTL & AJAX

- **Nothin' but browser**
- **Uses very open technologies**
- **Allows for simple richness**

**AJAX – new info from server without refresh**

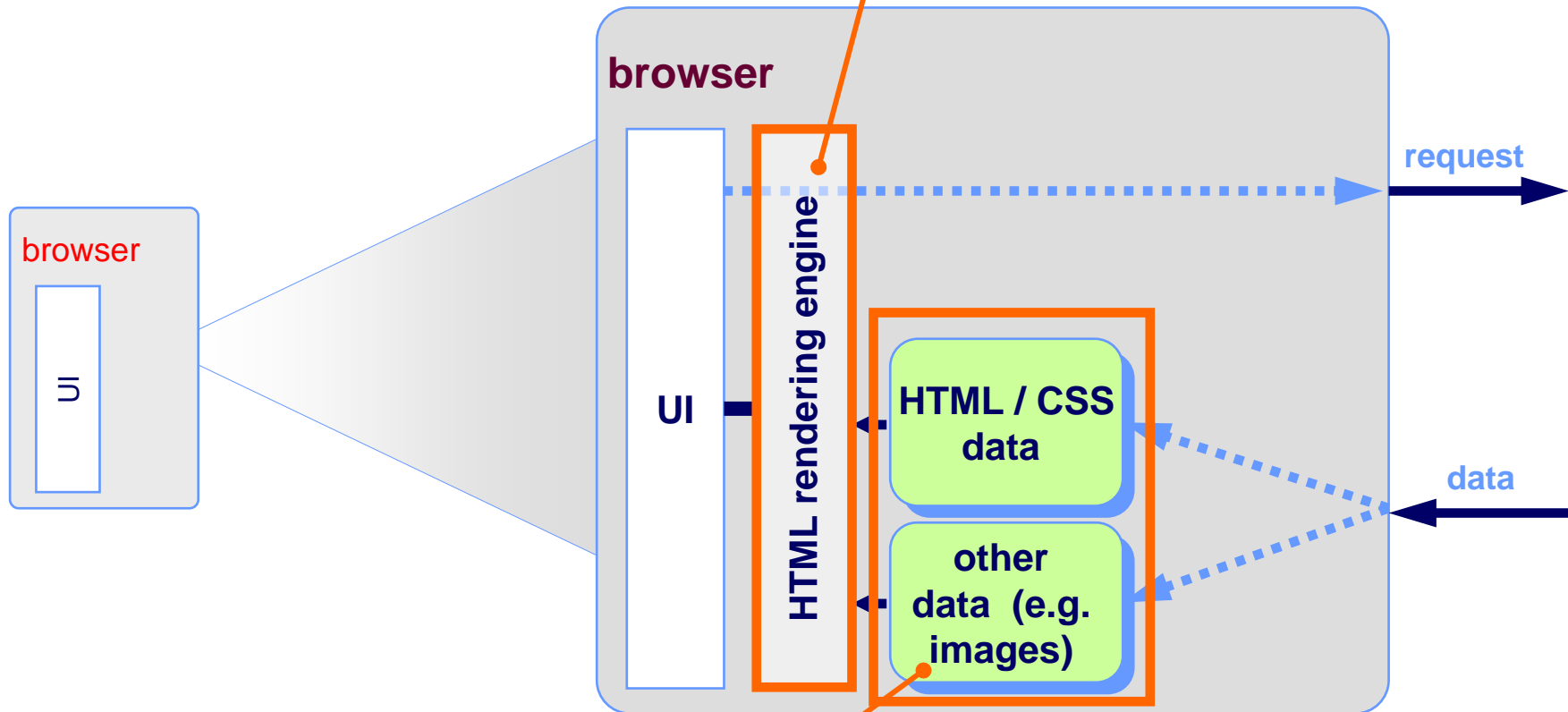
- **A JavaScript call makes a query to the server**
- **Server returns XML**
- **JavaScript manipulates CSS to reformat XML in place on existing screen**



# what's inside a browser so this works?

What really happens “under the hood” of a “classic” browser.

**rendering engine** -- takes HTML/CSS data and images, and ‘knows’ how to format and display it.



**data stores.** Browser knows how to save and manage data it downloads.

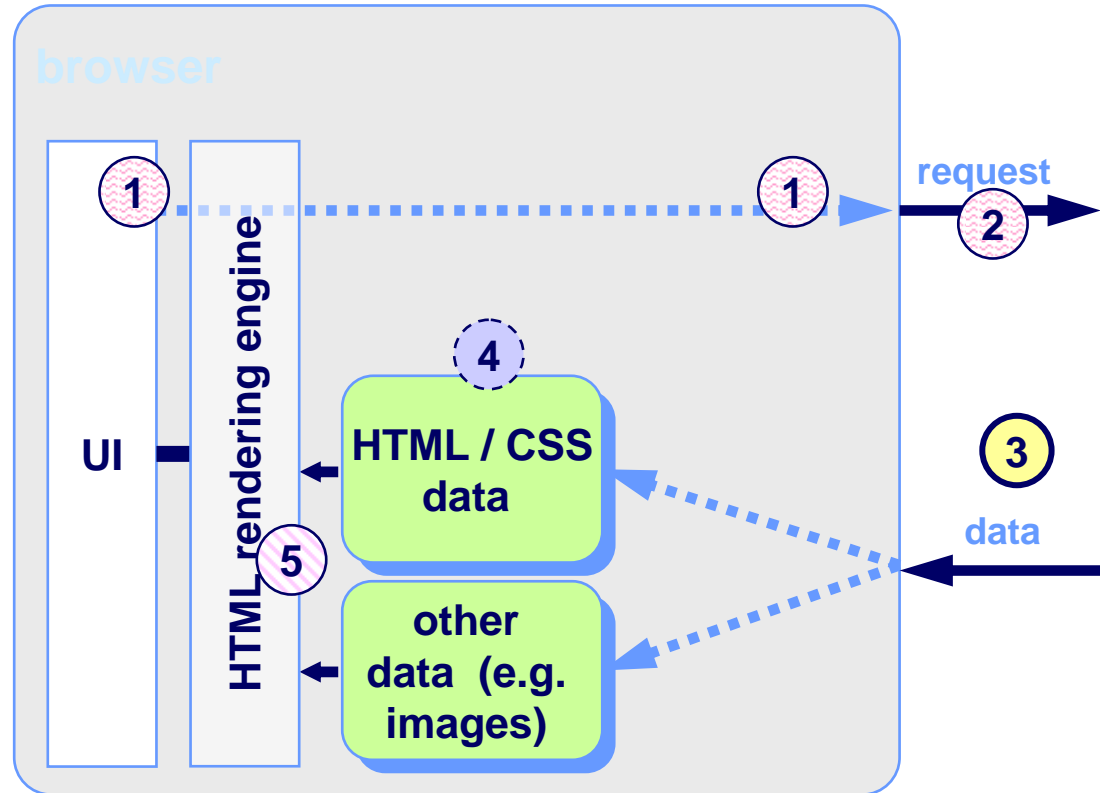
# how it all works --

1. User clicks on a link: rendering engine / browser sends of a request  
for a document. 2

2. Data comes back 3

3. Browser knows what to do with it and where to put it 4

4. HTML and CSS go to the rendering engine, which starts painting the screen. This engine also knows to send out more requests for images, needed in the page. 5

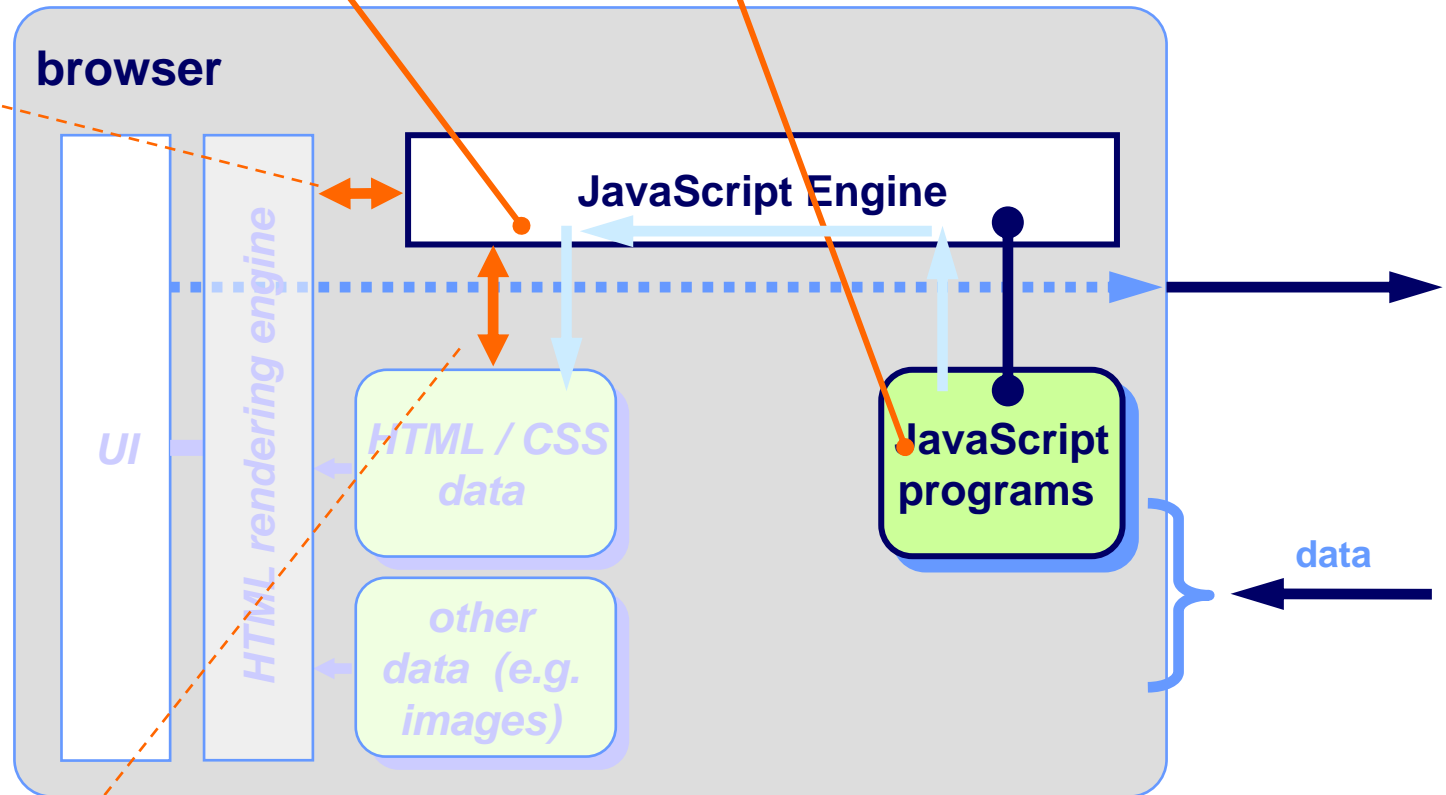


# enter JavaScript (1996-8)

**Javascript engine** – can run programs downloaded alongside the HTML, CSS and images.

## JavaScript

programs can detect UI events (clicks, etc.) and run code when the user does something: interactivity is *programmable*.



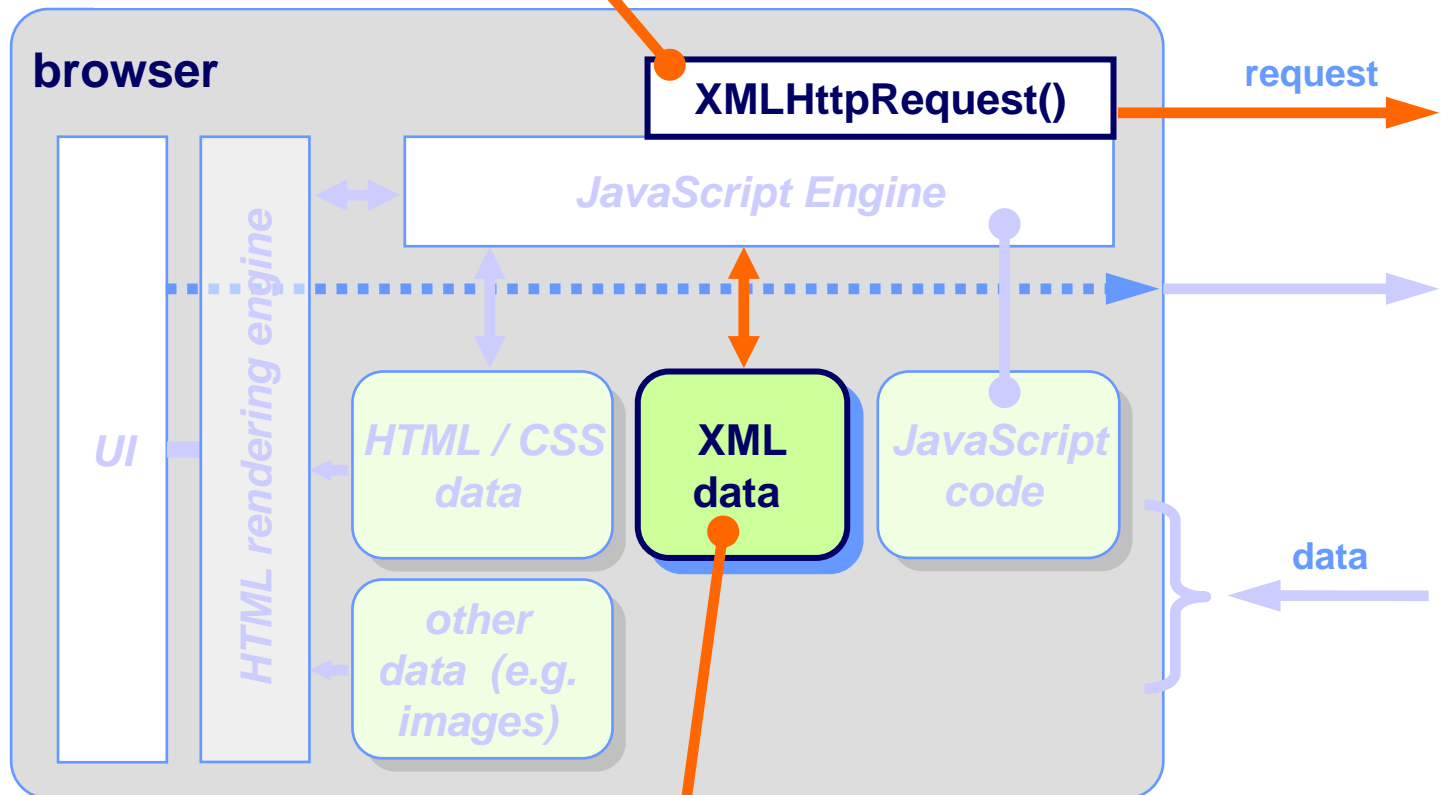
**JavaScript** programs, via the engine, can access and modify the HTML / CSS data, dynamically changing the UI that's displayed.



# Ajax: XML data, and a new tool

a new JavaScript function.. This lets JavaScript programs send out requests for data (images, XML, whatever) on their own, without waiting for a user click.

JavaScript programs can now go off and “do their own thing,” including getting data from elsewhere, without waiting for the user to do something!



**XML data support.** Browsers can now store XML data, and access / manipulate from JavaScript programs via the JavaScript engine.

# which brings us to Ajax

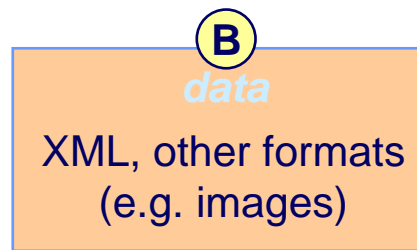
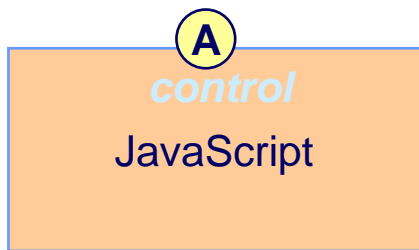
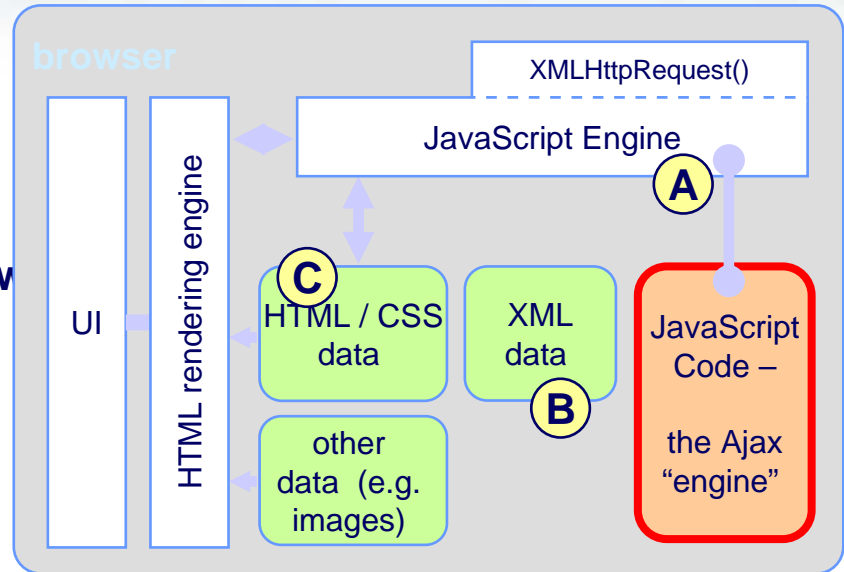
- Ajax is a *pattern* for building applications on the browser.

- The pattern is:

**A** Use JavaScript to control the show  
Use it to *asynchronously* request and retrieve data from remote servers, whenever the program thinks this is a good idea (i.e. not just when the user does something),

**B** Use *XML* to send numerical or text-style data to the browser. Then use JavaScript to extract data from the XML,

**C** Use HTML and CSS for display: manipulate this using JavaScript.



## Anatomy of a Pattern

- Ajax design patterns contain three steps
    - **Trigger** (event or timer)
    - **Operation** (Ajax, remote scripting)
    - **Update** (presentation)
- 



## Trigger

- Every pattern starts with
  - a user event
  - a timer event

mouseout hover keypress keydown mousedown  
drop filter choices mouseup drag click mousedown select  
focus blur resize move timeout



## Operation

Ajax patterns open the door to immediacy

- **Lookup** I can get information when I need it
- **Persist** I can save in real-time
- **Validate** I can prevent errors early
- **Invoke** I can make things happen now
- **Message** I can communicate instantly



## Update

- Finally, patterns reflect a visual change

**Indication. Busy Indication. Cursor Busy. In Context Busy. In Context Progress. Inline Status. Transition. Brighten Transition. Cross Fade Transition. Dim Transition. Expand Transition. Fade In Transition. Fade Out Transition. Flip Transition. Move Transition. Self-Healing Transition. Collapse Transition. Slide Transition. Rich Internet Object. Available. Selected. Identifiable Object. Slide-out. Popup Balloon.**



## Pattern-O-Matic



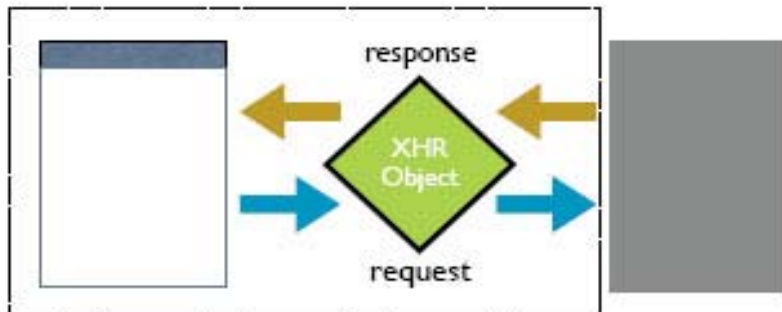
# Ajax – XMLHttpRequest object

- An XMLHttpRequest object can be created on the client by javascript
- Depending on the web browser (Firefox, various versions of Internet Explorer), the code to create it is different



## But What Can XHR Do?

- Make a request
- Return a response
- And do it asynchronously



# Ajax – XMLHttpRequest object

- `// javascript code for all browsers`
- `function getXMLHttpRequest()`
- `{`
- `var request = false;`
- `try`
- `{`
- `request = new XMLHttpRequest( ); // Firefox`
- `}`
- `catch( err1 )`
- `{`
- `try`
- `{`
- `request = new ActiveXObject( "Msxml2.XMLHTTP" );`
- `// some IE versions`
- `}`

# Ajax – XMLHttpRequest object

- `catch( err2 )`
- `{`
- `try`
- `{`
- `request = new XMLHttpRequest( "Microsoft.XMLHTTP" );`
- `// some other IE versions`
- `}`
- `catch( err3 )`
- `{`
- `request = false;`
- `}`
- `}`
- `return request;`
- `}`

# XMLHttpRequest Properties

- **onreadystatechange**
  - Event handler that fires at each state change
  - You implement your own function that handles this
- **readyState** – current status of request
  - 0 = uninitialized
  - 1 = loading
  - 2 = loaded
  - 3 = interactive (some data has been returned)
    - This is broken in IE right now
  - 4 = complete
- **status**
  - HTTP Status returned from server: 200 = OK
- **responseText**
  - String version of data returned from server
- **responseXML**
  - XML DOM document of data returned
- **statusText**
  - Status text returned from server

# Ajax – Sending a request to the server

- **// get an XMLHttpRequest object**
- **// build a url variable, call it url**
- **// open a connection with the server**
- **/\* prepare to receive response, i.e. give it to a function \*/**
- **// send the request**

# Ajax – Sending a request to the server

- `var myRequest = getXMLHttpRequest( );`
- `function callAjax( ) // javascript code`
- `{`
- `var url = “test.php?mike”;`
- `myRequest.open( “GET”, url, true );`
- `myRequest.onreadystatechange = responseAjax();`
- `myRequest.send( null );`
- `}`

# Ajax – waiting for a response

- The `readystatechange` property of our `XMLHttpRequest` object can have the following values:
- 0 → uninitialized
- 1 → loading
- 2 → loaded
- 3 → interactive
- 4 → completed

# Ajax – coding the response function

- The function is called when there is a change in the readystate property
- .. But we only want the function to execute when the readystate has value 4



# Ajax – coding the response

- `function responseAjax( ) // javascript code`
- `{`
- `if (myRequest.readyState == 4 )`
- `{`
- `// code here`
- `} // else do nothing`
- `}`

# Ajax – coding the response

```
• function responseAjax( ) // javascript code
• {
•   if (myRequest.readyState == 4 )
•   {
•     if ( myRequest.status == 200 ) // OK
•     {
•       // code goes here (success)
•     }
•     else
•     {
•       // code an error message here
•     }
•   } // else do nothing
• }
```

# AJAX -- Message Flow

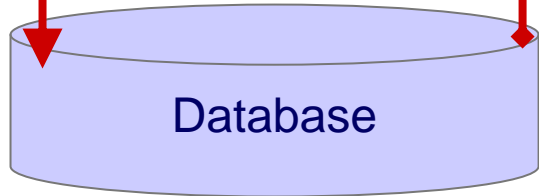
```
<script type="text/javascript">
XMLHttpRequest
function doResp() {
  if _ajax.readyState == 4 and
  _ajax.status != 200 {
    div=document.getElementById('status')
    div.innerHTML = _ajax.responseText;
  }
}
Save field onchange event:
_ajax = new XMLHttpRequest();
_ajax.onreadystatechange = doResp;
url = './validate?field='
      +this.name+'&value='+this.value;
_ajax.open('GET', url, true )

```

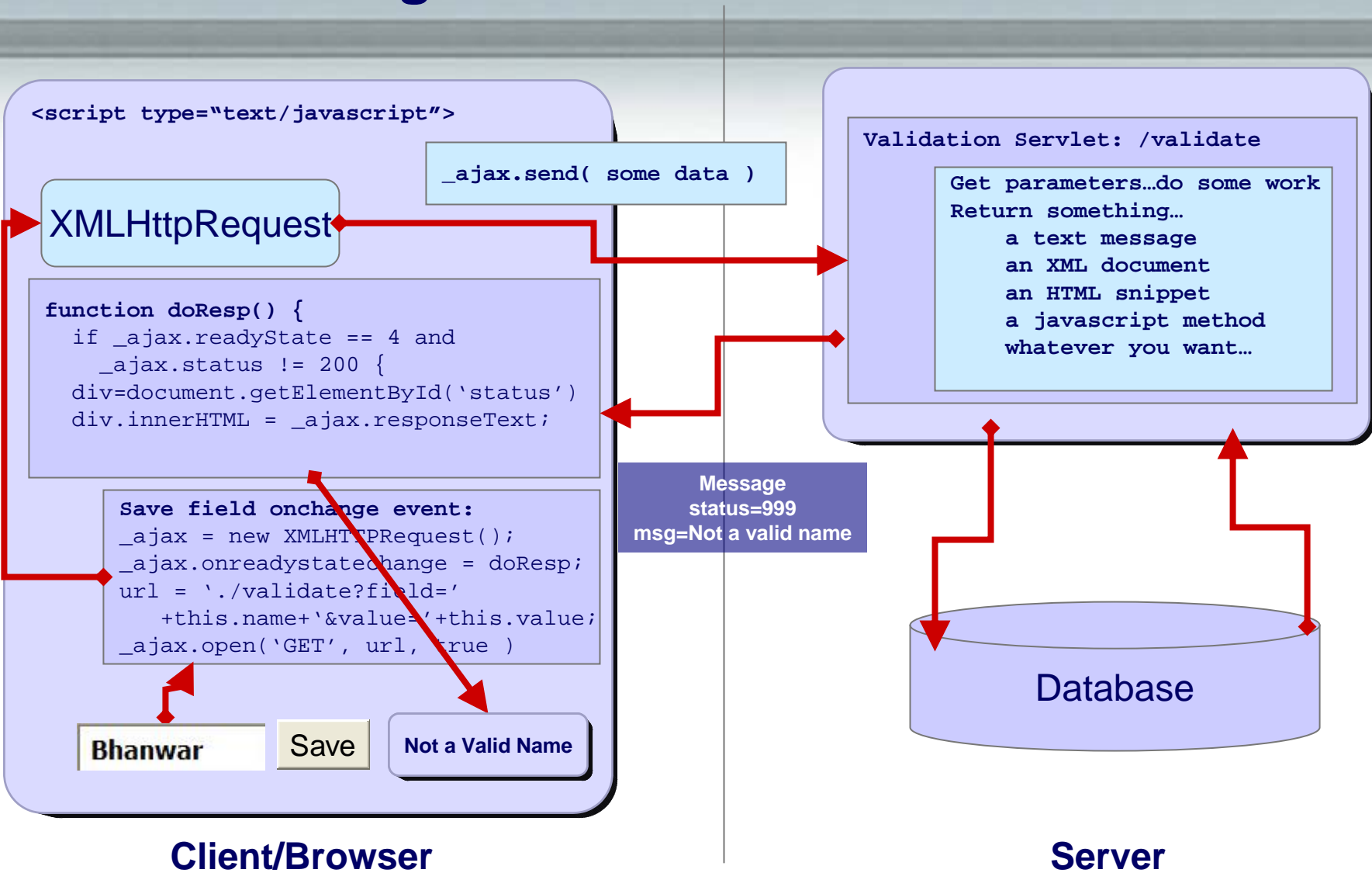
Client/Browser

```
Validation Servlet: /validate
Get parameters...do some work
Return something...
  a text message
  an XML document
  an HTML snippet
  a javascript method
  whatever you want...

```



Server



# Structure of System

- **Client/Server architecture**
- **XMLHTTP object is used to make request and get response in Client side**
- **Request can be done via “GET” or “POST” methods**
  - **“GET”**: parameters are attached to the url used to connect.
  - **“POST”**: parameters are sent as parameters to a function
- **Not many changes in Server side**
  - **Response is a combination of xml tags**

# What are the Issues with AJAX?

- **User does not know updates will occur.**
- **User does not notice an update.**
- **User can not find the updated information.**
- **Unexpected changes in focus.**
- **Loss of Back button functionality\*.**
- **URIs can not be bookmarked\*.**

\*These issues will not be discussed as they are not specific to accessibility.

# Specific Accessibility Issues

- **Assistive Technology users are not aware of updates**
  - Updates occur on a different section of the page than the user is currently interacting with.
  - Clicking a link updates a different section of the page.
  - Auto-complete fields or generated options not available to assistive technology.
  - User has no idea how to find new or updated content
  - Changes in focus prohibit complete review of the page
  - Changes in focus cause disorientation and additional navigation.

# Informing the User

- **Explain the interaction to the user**
  - Before accessing the AJAX enabled page
  - Within the AJAX enabled page
- **Where possible, provide a manual update option**
  - Necessary for WCAG 2.0 Guideline 2.2
- **Save the user's update preference**

# Make updates Noticeable

- **Change the background color of updated data**
  - Use a subtle color
  - Change only for a few seconds
  - Best for small areas of the page
- **Briefly blink the updated data**
  - Blink for 3 seconds or less
  - Avoid the flash threshold



# Help Users Find Updated Information

- **Provide option for updates via an Alert**
- **Provide option to set focus to new data.**
- **Use HTML header tags to mark sections with updated content.**
- **Use DHTML Accessibility Alert role in conjunction with a floating pop-up box.**
- **Use DHTML Accessibility Description role to describe new content.**

# Back & Refresh Buttons

- **Back Button**
  - Make it work as an undo?
- **Refresh Button**
  - Save state in `<body onload>` method?

# Ajax Disadvantages

- ***Client Side***
  - Poor compatibility with very old or obscure browsers, and many mobile devices.
  - Limited Capabilities like multimedia, interaction with web-cams and printers, local data storage and real time graphics.
  - The first-time long wait for Ajax sites.
  - Problem with back/forward buttons and bookmarks.
- ***Developer Side***
  - Easily Abused by “bad” programmers.
  - Not everyone have JavaScript enabled.
  - Too much code makes the browser slow.

# Trends in Market

## IBM

- ✓ IBM AJAX Toolkit Framework
- ✓ Eclipse framework support for AJAX Toolkits
- ✓ Collaboration with Open Source

## Mircosoft

- ✓ AJAX.net(code-named Atlas)
- ✓ It will work across any Web browser that supports AJAX technologies
- ✓ But it is not compatible with others(AJAX-style)

## Others

- ✓ Yahoo : give the user a feeling that they are no longer in a browser
- ✓ Oracle : Application Server 10g release 3, Support AJAX
- ✓ Google : trogdor, an ajax webpage editor

# Sources

- <http://ajaxpatterns.org>
- <http://www.telerik.com/products/ajax/history-of-ajax.aspx>
- <http://alexbosworth.backpackit.com/pub/67688>
- <http://ajaxian.com/by/topic/usability/>
- [http://en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))

# Thank You

Contact Details:



**Bhanwar Gupta**

**[bhanwar.gupta@jsw.in](mailto:bhanwar.gupta@jsw.in)**