Tutorial on AJAX

Lakshman M N

Object inet Consulting

Overview

- Introduction
- Motivation
- Remote Scripting
- Example
- Issues
- Applications

Introduction

- Stands for “Asynchronous JavaScript and XML “
- Development technique for creating interactive web applications
- Not a new *Technology* but more of a *Pattern*

Motivation for AJAX

- WebPages always RELOAD and never get UPDATED
- Users wait for the entire page to load even if a single piece of data is needed
- Single request/response restrictions

Components

- HTML (or XHTML) and CSS
  - Presenting information
- Document Object Model
  - Dynamic display and interaction with the information
- XMLHttpRequest object
  - Retrieving data **ASYNCHRONOUSLY** from the web server.
- Javascript
  - Binding everything together

Uses of AJAX Paradigm

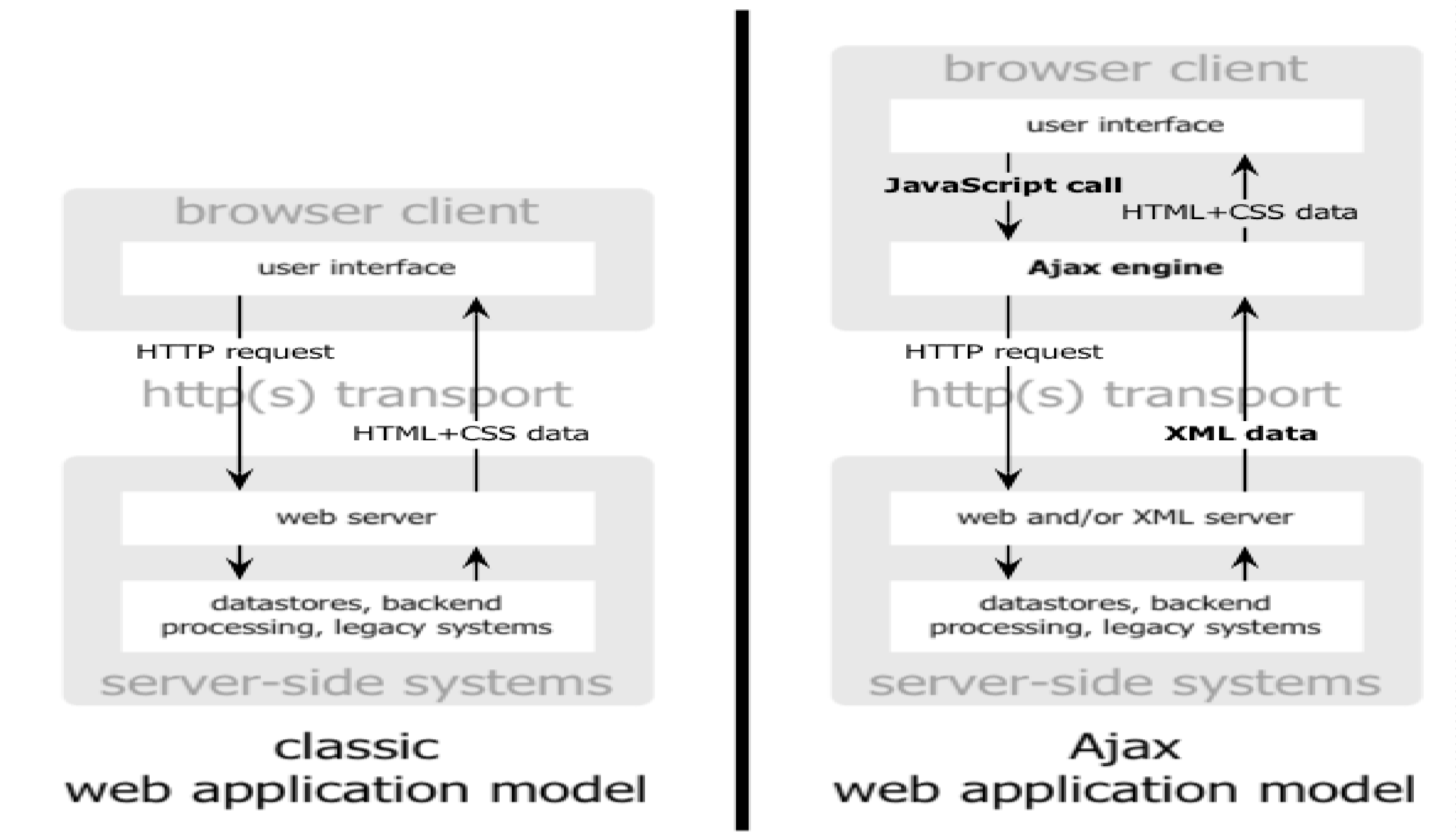
- Real-Time Form Data Validation
  - Form data that require server-side validation can be validated in a form “before” the user submits it.

- Auto completion
  - A specific portion of form data may be auto-completed as the user types.

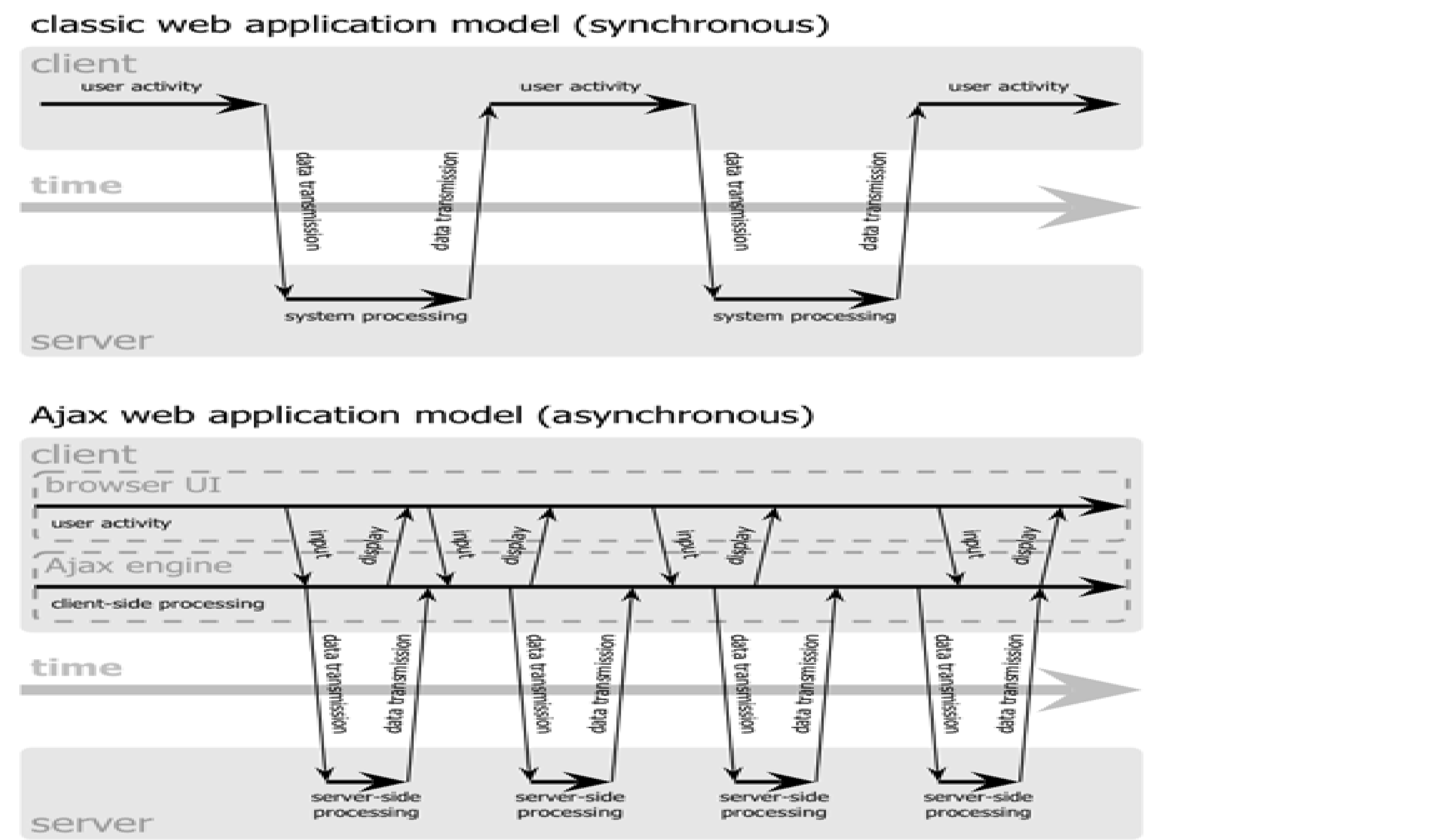
- Master Details Operations
  - Based on a client event, an HTML page can fetch more detailed information on data without refreshing the page.

- Sophisticated UI Controls
  - Controls such as tree controls, menus, and progress bars may be provided without page refreshes.

Web Application and AJAX



Request Processing



Asynchronous processing - XMLHttpRequest

- Allows to kick off an HTTP request in background
- Callbacks kick back into Javascript Code
- Supported in all standard browsers
- Similar to “image” object
  - Dynamically change the URL of the image source without using a page refresh

Example using XMLHttpRequest – Step 1

- Create Object
  - Worry about Browser Specific Creation !
- Example
  - `var requester = new XMLHttpRequest();`
  - `var requester = new ActiveXObject("Microsoft.XMLHTTP");`

Example using XMLHttpRequest – Step 1

- Some browsers expect the response from the server to contain an XML mime-type header.
- To satisfy this add:

```
requester.
       overrideMimeType(‘text/xml’)
```

Using XMLHttpRequest – Step 2

- Transferring data to Server
  - `Open()` to initialize connection to Server
  - `Send()` to send the actual Data
- Example
  - `requester.open(“GET", url, true)`
    - The first parameter is the HTTP request method.
    - The second parameter is the url of the page being requested.
    - The third parameter indicates whether the request is asynchronous.

Using XMLHttpRequest – Step 2

- The parameter to the **`send()`** method is the data to be sent to the server is POST is employed.
- The data should be in the form of a querystring.

```
name=value&name1=value1&name2=value2
```

- In order to POST data the MIME type of the request needs to be changed

```
requester.setRequestHeader(‘Content-Type’,’application/x-www-form-urlencoded’)
```

What happens after sending data ?

- **`XMLHttpRequest`** contacts the server and retrieves the data
  - Can take indeterminate amount of time

- Event Listener to determine when the object has finished retrieving data
  - Specifically listen for changes in "**`readyState`**" variable

Using XMLHttpRequest – Step 3

- Set up a function to handle the event when the **`readyState`** is changed to 4
  - 0 – Uninitialised
  - 1 – Loading
  - 2 – Loaded
  - 3 – Interactive
  - 4 – Completed
- Example

```
requester.onreadystatechange = stateHandler;
```

Using XMLHttpRequest – Step 3 Contd

- Check whether the **`XMLHttpRequest`** object successfully retrieved the data, or was given an error code

- Example

  ```
  if (requester.readyState == 4)
   {
           if (requester.status == 200)
           {
                success();
           }
   }
  ```

Using XMLHttpRequest – Step 4

- Parse and display data
  - **responseXML**
    - DOM-structured object
  - **responseText**
    - One complete string

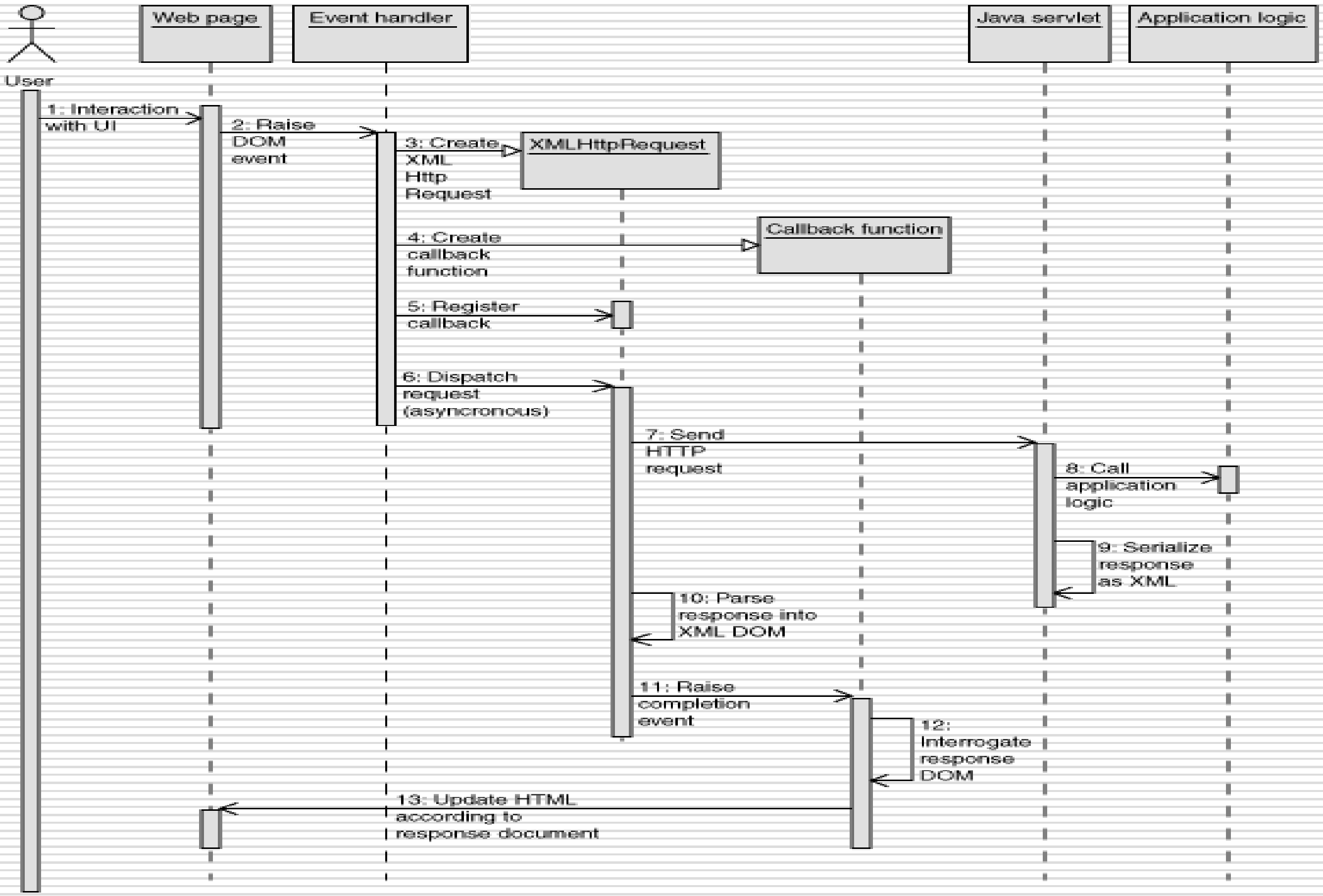
- Example

```
var nameNode = requester.responseXML.
               getElementsByTagName("name“)[0];

var nameTextNode = nameNode.childNodes[0];

var name = nameTextNode.nodeValue;
```

Interaction between Components


User
Web page
Event handler
Java servlet
Application logic
1: Interaction with UI
2: Raise DOM event
3: Create XML Http Request
XMLHttpRequest
4: Create callback function
Callback function
5: Register callback
6: Dispatch request (asyncronous)
7: Send HTTP request
8: Call application logic
9: Serialize response as XML
10: Parse response into XML DOM
11: Raise completion event
12: Interrogate response DOM
13: Update HTML according to response document

Problems

- Breaking the “back” button
  - Browsers record static page visits
  - Invisible *IFrames* can invoke changes that populate the history
- Changing parts of the page unexpectedly
  - Should only ever occur in narrowly defined places
- Bookmarking a particular “State” becomes difficult
  - Javascript generates the page NOT server

Problems Contd

- Increase in the code size on browser
  - Response time affected
- Difficult to debug
  - Processing logic both in client and server
- Viewable Source
  - Open to hackers or plagiarism
- Server Load
  - Asynchronous request is a “heavy” operation

Role of AJAX in Web 2.0

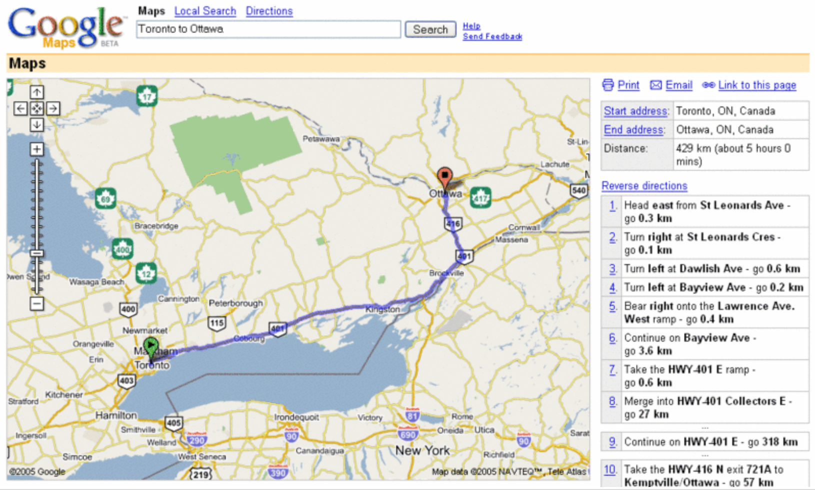
- Core features of Web 2.0
  - Web as a Platform
  - Collective Intelligence
  - Above the level of Single Device
  - Services , not packaged software
  - Rich User experiences

- AJAX
  - Assists in User Interfaces
  - Less machine readable / linkable webpages

AJAX as of today

- Frameworks
  - Client-side
    - DOGO
    - BackBase
    - AJForm
  - Server-side
    - JSON (Javascript Object Notation)
    - Struts – Layout
    - ComfortASP.NET
  - Many more at
    - http://www.ajaxpatterns.org/AJAXFramework
- Browsers
  - IE , Mozilla, Galeon, Firefox

Who is using AJAX ?



Toronto to Ottawa

Start address:	Toronto, ON, Canada
End address:	Ottawa, ON, Canada
Distance:	429 km (about 5 hours 0 mins)

1. Head **east** from **St Leonards Ave** - go **0.3 km**
2. Turn **right** at **St Leonards Cres** - go **0.1 km**
3. Turn **left** at **Dawlish Ave** - go **0.6 km**
4. Turn **left** at **Bayview Ave** - go **0.2 km**
5. Bear **right** onto the **Lawrence Ave. West** ramp - go **0.4 km**
6. Continue on **Bayview Ave** - go **3.6 km**
7. Take the **HWY-401 E** ramp - go **0.6 km**
8. Merge into **HWY-401 Collectors E** - go **27 km**
9. Continue on **HWY-401 E** - go **318 km**
10. Take the **HWY-416 N** exit **721A** to **Kemptville/Ottawa** - go **57 km**

Reading Material

- **Overview**
  - http://en.wikipedia.org/wiki/AJAX
  - http://java.sun.com/developer/technicalArticles/J2EE/AJAX/index.html?cid=59754
- **Original Article from Adaptive path**
  - http://www.adaptivepath.com/publications/essays/archives/000385.php
- **Examples**
  - http://www.sitepoint.com/article/remote-scripting-ajax.html
  - http://www.mousewhisperer.co.uk/ajax_page.html
  - http://www.clearnova.com/ajax/
  - http://www.webpasties.com/xmlHttpRequest/
- **AJAX based Applications**
  - http://www.ajaxreview.com/
  - http://ajaxblog.com/
- **Issues/Problems**
  - http://sourcelabs.com/ajb/archives/2005/05/ajax_mistakes.html

Thank You

- Questions ?