

2023/2024(1)
EF234301 Web Programming

Lecture #7

ASP.NET & RIA

Misbakhul Munir **IRFAN SUBAKTI**

司馬伊凡

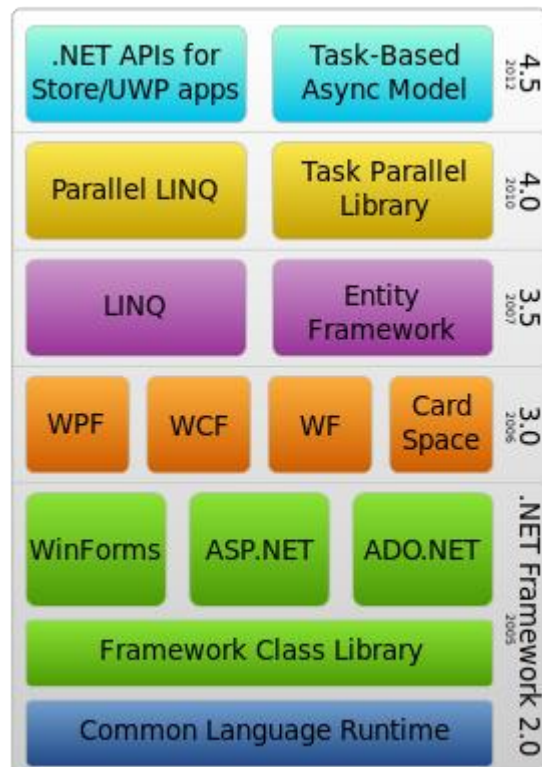
Мисбакхул Мунир **Ирфан Субакти**

ASP.NET

- Introduction



.NET Framework



.NET – A unified platform



Web development world

- CGI (Common Gateway Interface)
 - Requires a lot of server resources
- ASP Classic
 - Interpreted and loosely-typed code
 - Mixes layout and logic code
 - Limited development and debugging tools
 - No real state management
- ASP.NET
 - Compiled languages support
 - .NET framework support
 - Code separation
 - Visual web development
 - State management
 - XML based configuration files

Advantages of ASP.NET

- *Object oriented programming* model
 - Event driven
 - Control based architecture
- Code in any supported languages
 - C#, VB.NET, and others
- *Faster* than ASP classic
- Rapid development style
- Multidevice and multibrowser
- Easy to deploy and configure

ASP.NET 2.0 features

- Rich controls
- Master pages
- Themes
- Security and membership
- Data source controls
- Web parts
- Profiles

ASP.NET 3.5 features

- LINQ (Language Integrated Query)
- ASP.NET AJAX (Asynchronous JavaScript And XML)
- ASP.NET MVC (different application architecture)

ASP.NET 4.0 features

- Focus:
 - improving the performance and Search-Engine Optimization (SEO)

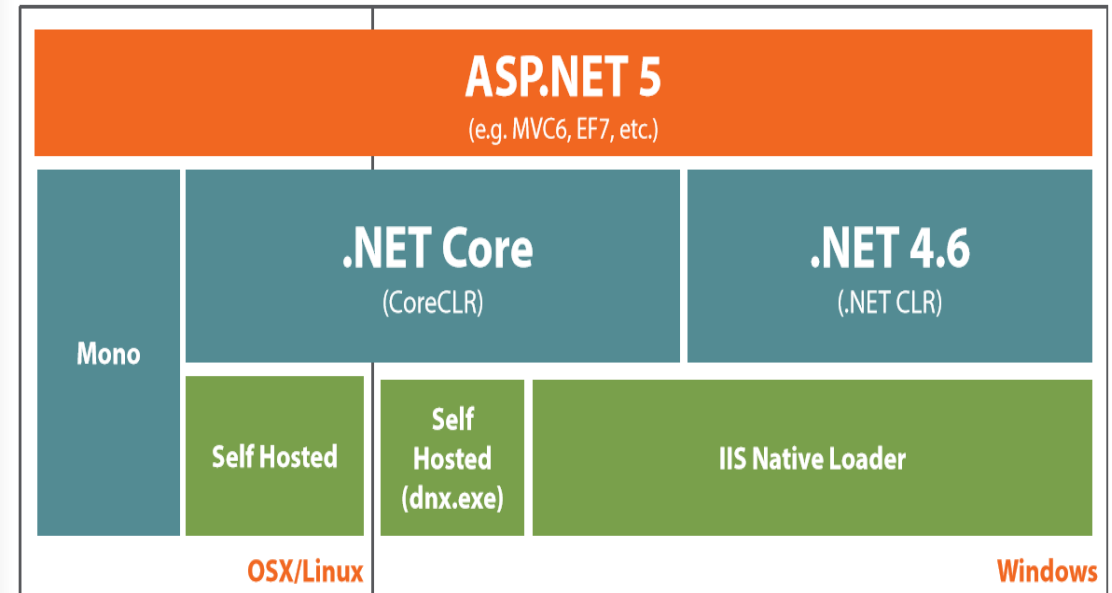
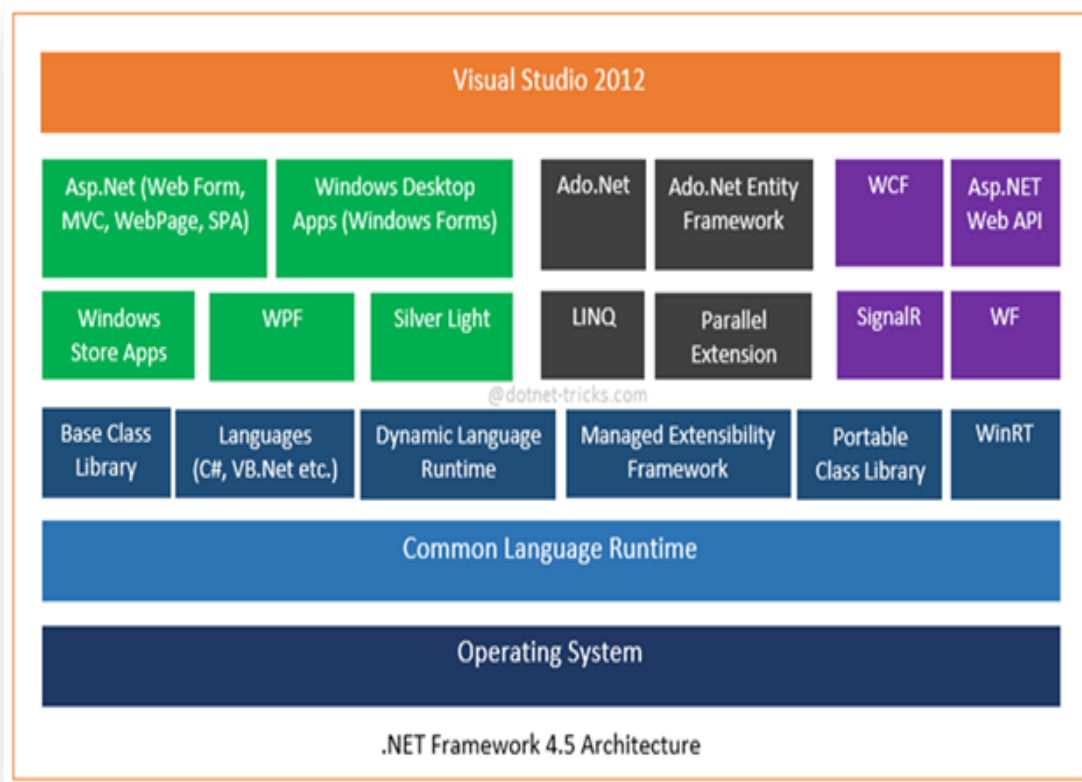
ASP.NET 4.0 features (continued)

- Output cache extensibility
- Session state compression
- View state mode for individual control
- *Page.MetaKeyword* and *Page.MetaDescription* properties
- *Response.RedirectPermanent* method
- Routing in ASP.NET
- Increase the URL character length
- New syntax for HTML Encode
- Predictable Client IDs
- *Web.config* file refactoring
- Auto-Start ASP.NET applications
- Improvements on Microsoft Ajax Library

ASP.NET 5.0 features

- 2 huge changes
 - It's much leaner – e.g. no longer requires System.Web.dll.
 - It's modular – almost all features are implemented as NuGet packages
- NuGet → a free and open-source package manager designed for the Microsoft development platform

ASP.NET framework



ASP.NET page syntax

- * .aspx file extension
- Directive syntax
- Code declaration blocks
- Code render blocks
- Server side comments
- Custom server control
- Data-binding expression syntax
- Server-Side Object Tag Syntax
- Server-Side Include Directive Syntax

Directive syntax

`<%@ %>`

- @ Page
- @ Control
- @ Import
- @ Implements
- @ Register
- @ Assembly
- @ Master
- @ WebHandler
- ...

```
<%@ Page Language="C#" MasterPageFile="~/master"
ValidateRequest="false" %>
```

```
<%@ Register Src="~/Workflow/Navigation.ascx"
TagName="Navigation" TagPrefix="spg" %>
```

Code declaration block

<script></script>

```
<script runat="server" language="codelanguage" Src="pathname">  
    Code goes here.  
</script>
```

```
<html>  
<script language="C#" runat="server">  
    void EnterBtn_Click(Object Src, EventArgs E) {  
        Message.Text = "Hello " + Name.Text + ", welcome to ASP.NET!";  
    }  
</script>  
</html>
```

Code render blocks

`<%= %>`

```
<% inline code %>  
<%=inline expression %>
```

```
<% for (int i=0; i<10; i++) { %>  
    <font size="<%=i %>"> Hello World! </font>  
<% } %>
```

Server side comments

```
<%-- --%>
```

```
<%-- this is block of comment --%>
```

Note: do not use comment inside <% %> block.

```
<%--
```

```
  <asp:button runat="server" id="MyButton" OnClick="MyButton_Click" />
```

```
--%>
```


Custom server control

```
<tagprefix:tagname id="OptionalID" attributename="value"  
eventname="eventhandlermethod" runat="server" />  
OR  
<tagprefix:tagname id="OptionalID" runat="server" />
```

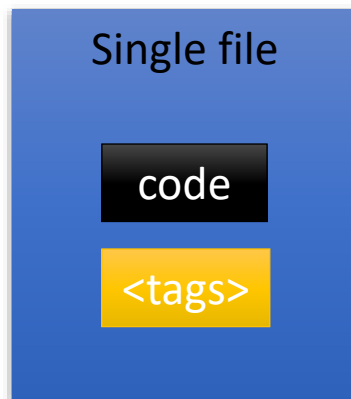
```
<%@ Register Src="~/Workflow/Navigation.ascx" TagName="Navigation"  
TagPrefix="spg" %>  
  
<spg:Navigation ID="controlNavigation" runat="server" />
```

Hello World

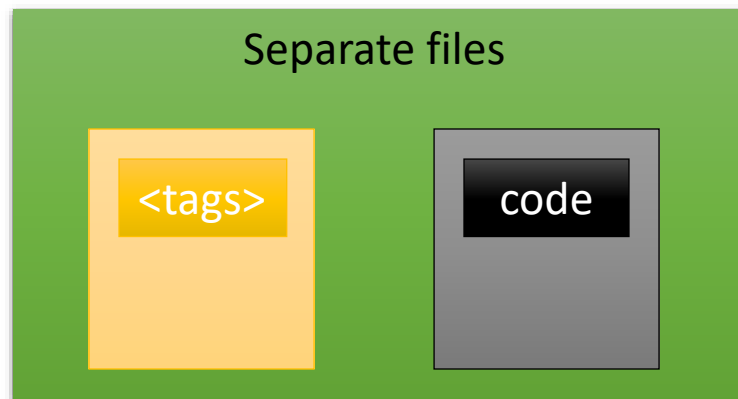
- The steps

Web forms

- Web form combines declarative tags (HTML, ASP.NET directives, server controls, static text) with code.
- Code separation is encouraged.



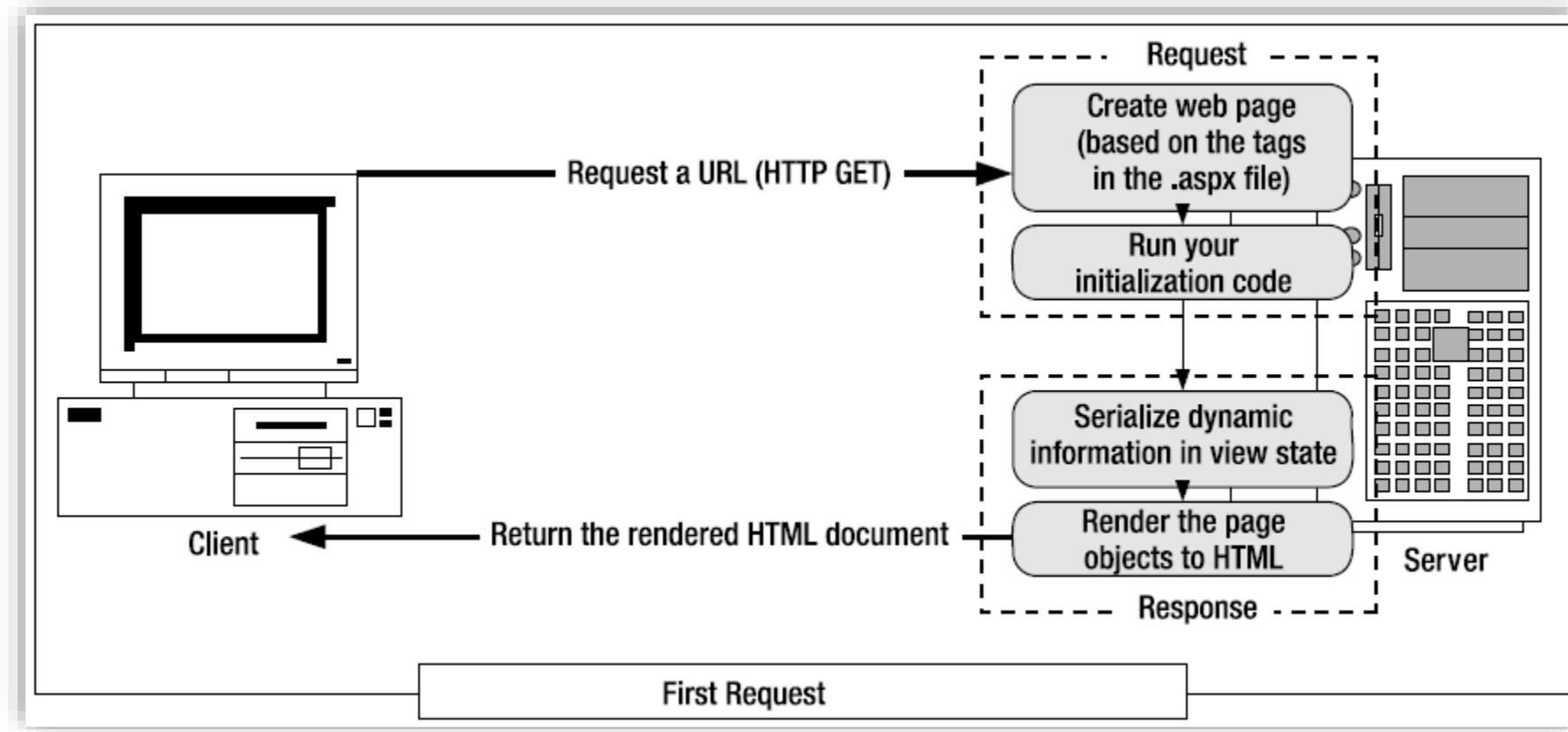
Form.aspx



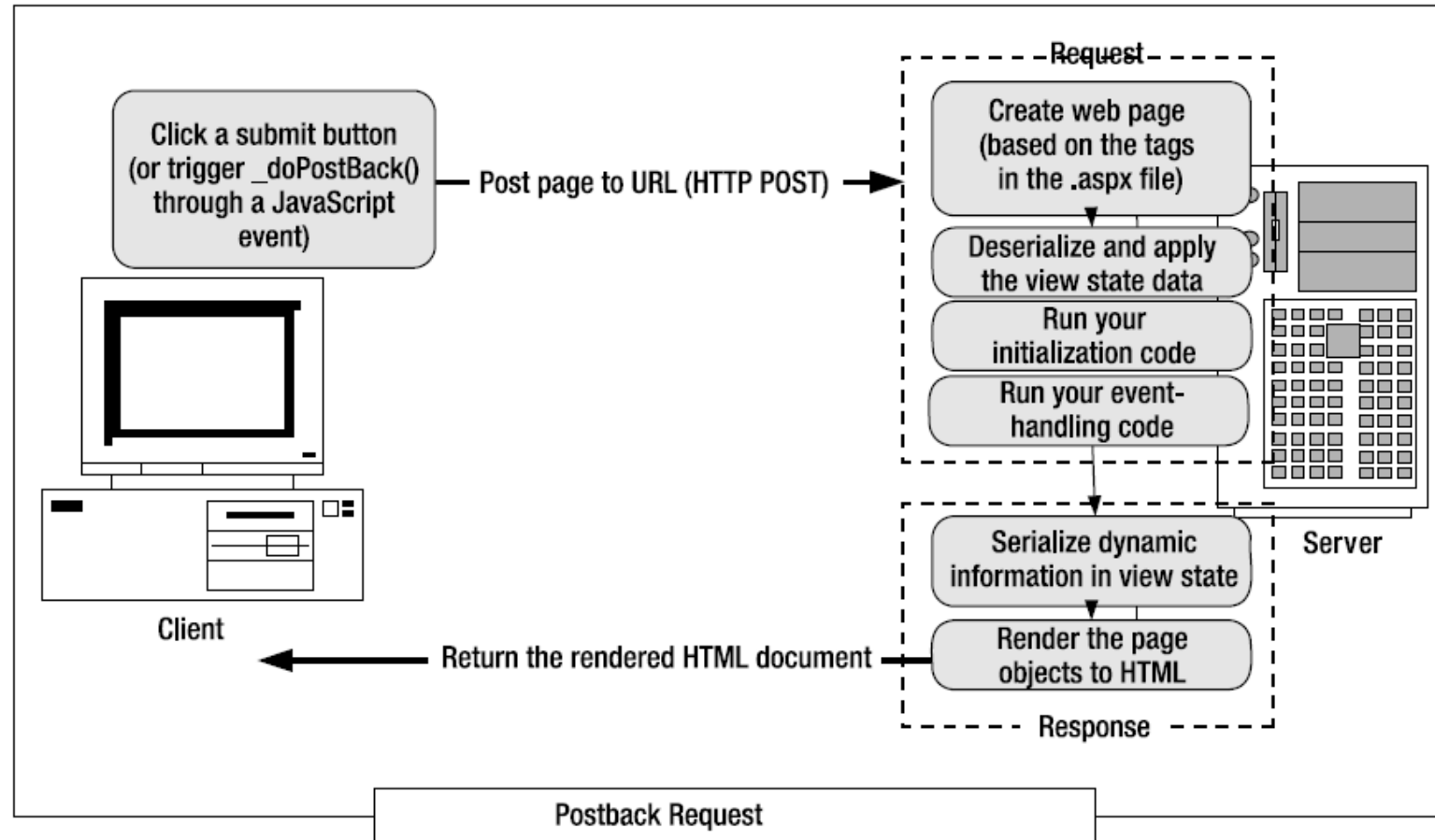
Form.aspx

Form.aspx.cs

Page request



Page request (continued)



Page class

- Session
- Application
- Cache
- Request
- Response
- Server
- User
- Trace

Validation control

- RequiredFieldValidator
- RangeValidator
- RegularExpressionValidator
- CompareValidator
- CustomValidator
- ValidationSummary

ASP.NET learning site

- <https://dotnet.microsoft.com/apps/aspnet>
- <https://docs.microsoft.com/>

Rich Internet Application (RIA) → Rich Web Application (RWA)

- Web-based applications → characteristics of graphical desktop applications.
- Built with powerful development tools → run faster and be more engaging.



RIA/RWA: the definition

- A web application that has many of the characteristics of desktop application software.
- The concept is closely related to a single-page application, and may allow the user interactive features such as drag and drop, background menu, WYSIWYG editing, etc.
- **HTML5** is a current standard for delivering rich web applications, supported by all major browsers.



RIA/RWA: the example



- Revenue: US\$15 billion (2019), US\$19.8 billion (2020), US\$28.8 billion (2021), US\$29.24 billion (2022)
- Parent: Google LLC (2006–present)
- Written in Python (core/API), C (through CPython), C++, Java (through Guice platform), Go, JavaScript (UI)

RIA: case study of YouTube

- 2010, the mobile version of the site was relaunched based on HTML5, avoiding the need to use Adobe **Flash** Player and optimized for use with touch screen controls → an app for the Android platform
- 2011, an HTML5 version of the YouTube player began supporting side-by-side 3D footage that is compatible with NVidia 3D Vision.
- 2015, YouTube announced that **HTML5** would be the **default playback** method on supported browsers.
 - YouTube used to employ Adobe Dynamic Streaming for Flash, but with the switch to HTML5 video now streams video using Dynamic Adaptive Streaming over HTTP (MPEG-DASH), an adaptive bit-rate HTTP-based streaming solution optimizing the bitrate and quality for the available network.

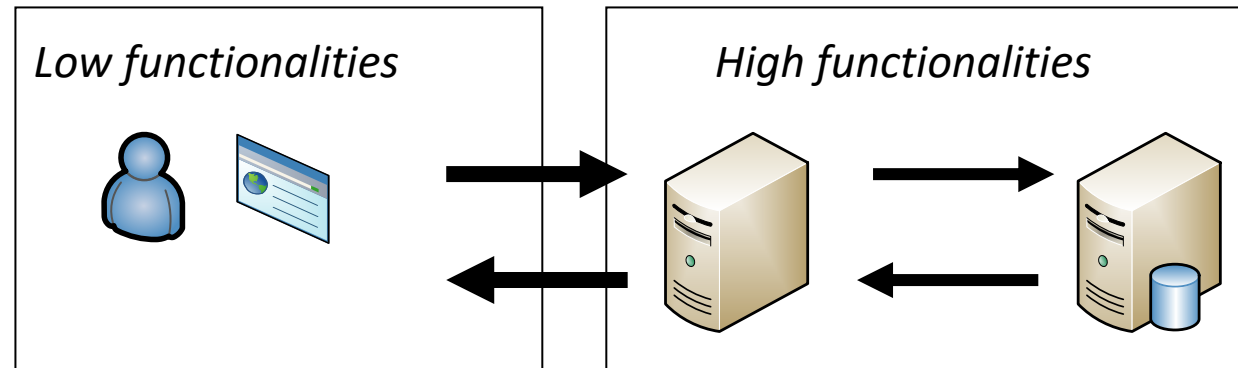
RIA: the characteristic



- *Dynamic download*: RIA's are typically downloaded using an on-demand model.
- *Sandboxed* : RIA's typically 'live' within a sealed sandbox, with no access to the host O/S.
- *Asynchronous communications*: event or user-based transactions with the remote server, breaking the page-based reload paradigm.
- *Highly dynamic and flexible user interfaces*: menus, tabs, multimedia, rich editing, generally funky goodies.
- *Client-side functionality*: many functions can now live at the client, even have the ability to host mini client-side databases.
- *Created using bespoke scripting languages*: in many cases, often with a good dose of AJAX thrown in.

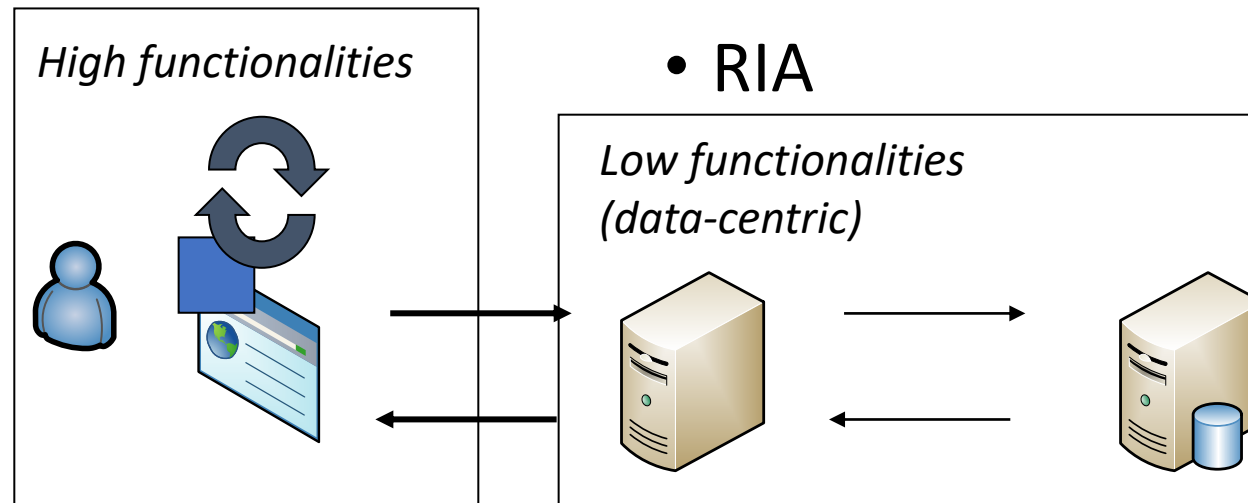
RIA: the comparison

- Traditional web



- A lot of E2E transactions
- Server's workload high

- RIA



- A few E2E transactions
- Server's workload low

RIA: the technologies (before HTML5)

- Adobe Flash → Apache Flex
 - Flash → closed source | Flex → open source
 - Client runtime on Mac, Windows, Linux
 - Flex used “ActionScript”, MXML, Flex Class Library
 - Interface development uses Flash IDE
 - Flex: plugin for Eclipse
 - Companies used:
 - BBC iPlayer, ChessCube, HBO Go, CareCloud, VMware



RIA: the technologies (before HTML5)

- AIR (Adobe Integrated Runtime) → by using Adobe Animate, ActionScript & Flex
 - eBay Desktop, Pandora One desktop, TweetDeck, Angry Birds, Machinarum



RIA: the technologies (before HTML5) – cont.

- Java applet → faster than JavaScript
 - Deprecated



RIA: the technologies (before HTML5) – cont.

- JavaFX → part of OpenJDK: OpenJFX project
 - Version 1.0 by using JRE (Java Runtime Environment)
 - Available for Windows, Linux, Mac
 - Open source, GPL
 - IDE: NetBeans
 - Companies used:



doubleSlash



Biting Bit



Keylord



Technologies



CaseFleet



Open
Lowcode



Full Stack



Endeeper



HyperSoft

RIA: the technologies (before HTML5) – cont.

- Microsoft Silverlight

- .NET based
- Browser plugin on WPF (Windows Presentation Foundation)
- Available for Mac and Windows
- IDE: Visual Studio
- Terminate on Internet Explorer 11 (12 October, 2021)
- Companies used:
 - Magnet Media Inc.
 - Creative Mettle
 - Treliant Risk Adviors LLC
 - Zeta Interactive
 - Kaseya Limited



RIA: the technologies (before HTML5) – cont.

- XForms → next generation HTML/XHTML forms
 - Declarative XML-based programming language.
 - W3C standard, and in use in companies around the world.

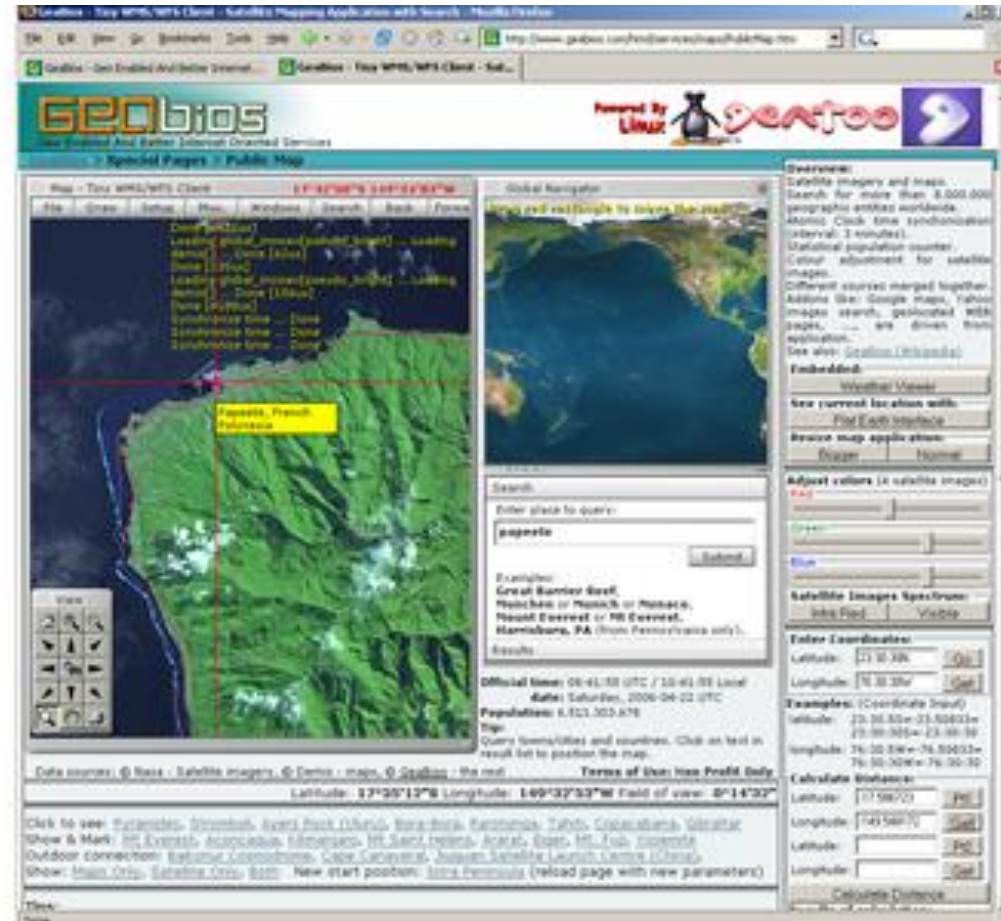


RIA: the technologies (before HTML5) – cont.

- Open Laszlo
 - Discontinued open-source platform for the development and delivery of RIA
 - Released under the Open Source Initiative certified Common Public License (CPL).
 - Consists of the LZX programming language and the OpenLaszlo Server.
 - LZX is an XML and JavaScript description language similar in spirit to XUL, MXML, and Extensible Application Markup Language (XAML).
 - LZX enables a declarative, text-based development process that supports rapid prototyping and software development best practices → to be familiar to traditional web application developers who are familiar with HTML and JavaScript.

RIA: the technologies (before HTML5) – cont.

- Open Laszlo (continued)



RIA: HTML5 technology

- 5th and latest major version of HTML that is a World Wide Web Consortium (W3C) recommendation.
- The current specification is known as the HTML Living Standard and is maintained by a consortium of the major browser vendors (Apple, Google, Mozilla, and Microsoft), the Web Hypertext Application Technology Working Group (WHATWG).
- Intended to subsume not only HTML 4 but also XHTML 1 and DOM Level 2 HTML.
- HTML5.3 (2017)

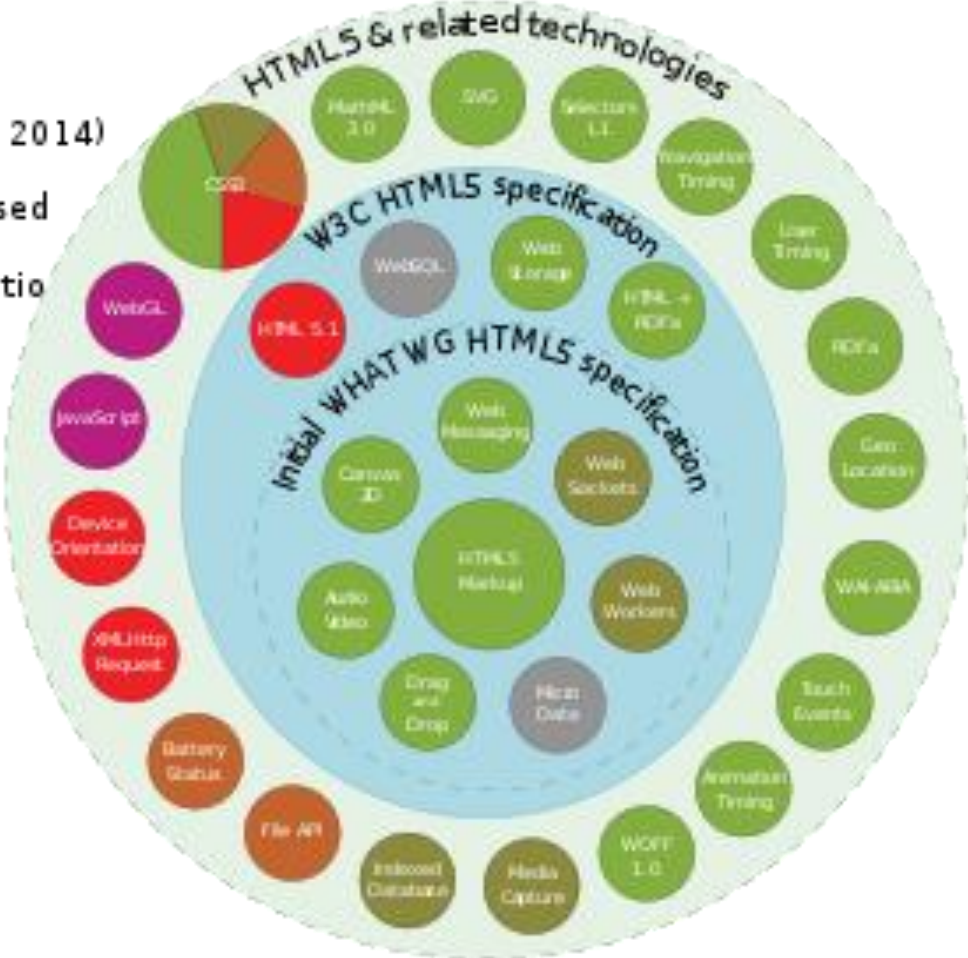


RIA: HTML5 technology (continued)

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Silverlight: Hello World

- Needs 3 things
 - HTML file
 - Silverlight.js provided in SDK
 - XAML file in Silverlight XAML for defining canvas
- Hello World in Silverlight



Hello World

Silverlight: Hello World (continued)

JavaScript include

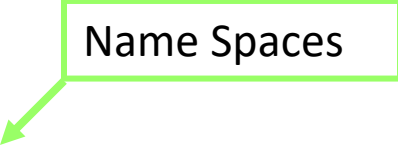


```
<html><head> <title>Hello World</title>
  <script src="js/Silverlight.js" type="text/javascript"></script>
</head><body>
  <div id="Ag1Host" style="background:#FFFFFF">
    <script type="text/javascript">
      var pe1 = document.getElementById("Ag1Host");
    </script></div>
<script>
Silverlight.createObjectEx({source: 'xaml/MyFirstSilverlightPage.xml',
  parentElement:pe1, id:'Ag1', properties:{width:'300', height:'100',
  background:'#00FFFFFF', isWindowless:'true', framerate:'24', version:'0.90.0'},
  events:{onError:null, onLoad:null}, context:null});
</script></body></html>
```

Silverlight: Hello World (continued)

- XAML file:

```
<canvas  
  xmlns="http://schemas.microsoft.com/client/2007"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  
  <TextBlock FontSize="36" Canvas.Left="17" Canvas.Top="10"  
    Foreground="Blue">Hello World</TextBlock>  
</canvas>
```



JavaFX: Hello World

```
import javafx.ui.*;
Frame {
    title: "Hello World JavaFX"
    width: 200
    height: 50
    content: Label {
        text: "Hello World"
    }
    visible: true
}
```




Running on IDE

An advantage: declared object instantiation → throw away the impractical thing about swing

- For command line development see http://jfx.wikia.com/wiki/A_command_line_development_of_HelloWorld.fx

AJAX: RIA without browser plugin

[Ads by Google](#) [Peter Answers](#) [Ask and Get Answers](#) [Tarot Cards](#) [Psychic Tarot](#) [Studio Peter](#)

 Select a language ▼

Peter Answers Virtual Tarot

Do you need to ask a question? Are you looking for answers? Peter offers you a space to ask anything you want. However, before each question you must write a petition. If the answer is not what you expected, at least you make catharsis and ask again.

Petition:

Question:

Peter answers: Asynchronous JavaScript and XML.


How can I make a petition?
You can make a petition by either writing: **Peter, please answer the following question:** or by using the short sentence: **Peter, please answer:**
Press "Enter" to go to the following textbox.

How can I ask a question?
Please keep a friendly language. End the question by pressing "Enter" or using a question mark (?).

Why is it a Virtual Tarot?
The people who have used the game have called it "Virtual Tarot" because, even if it not played with cards, the "guessing character" of it is related to the name.

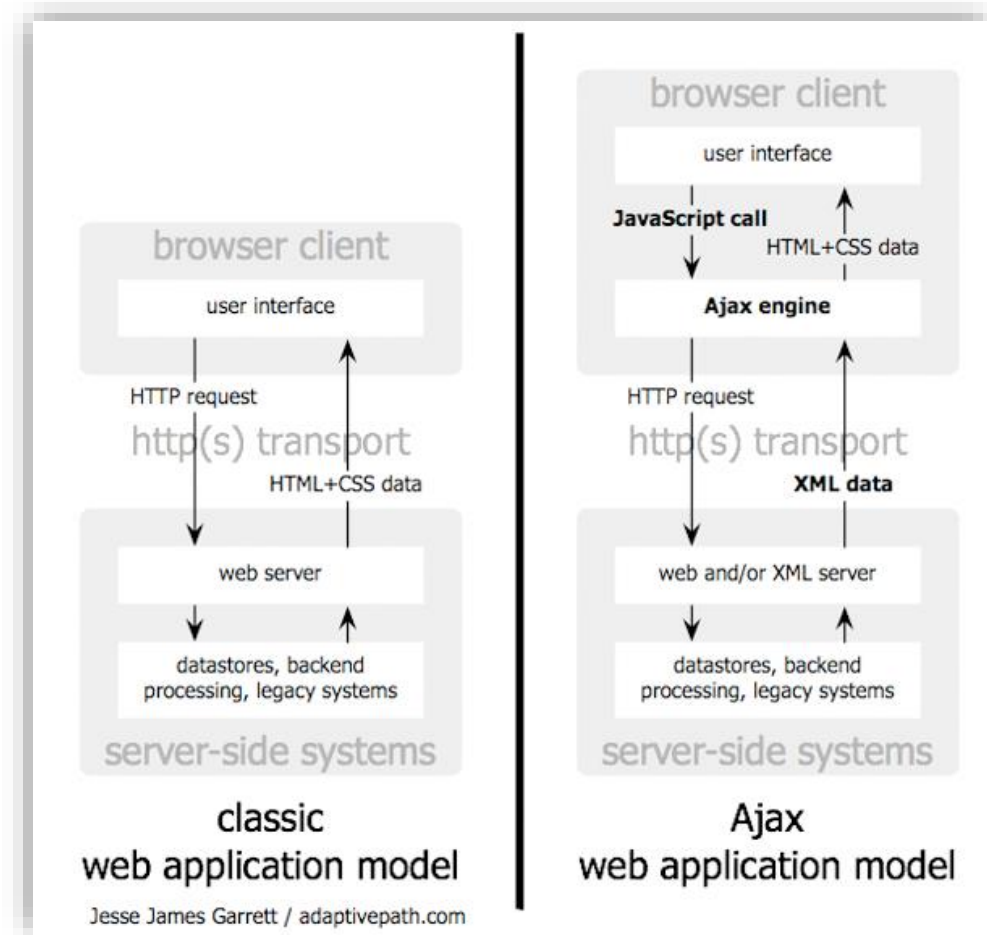
[2009 Tarot Prediction](#)
Your year read in a Professional's Tarot Cards, now completely Free!
www.AboutAstro.com

[What will happen in 2009](#)
With your birthdate, I will tell you what will it be 2009. Free
www.sara-freder.com/

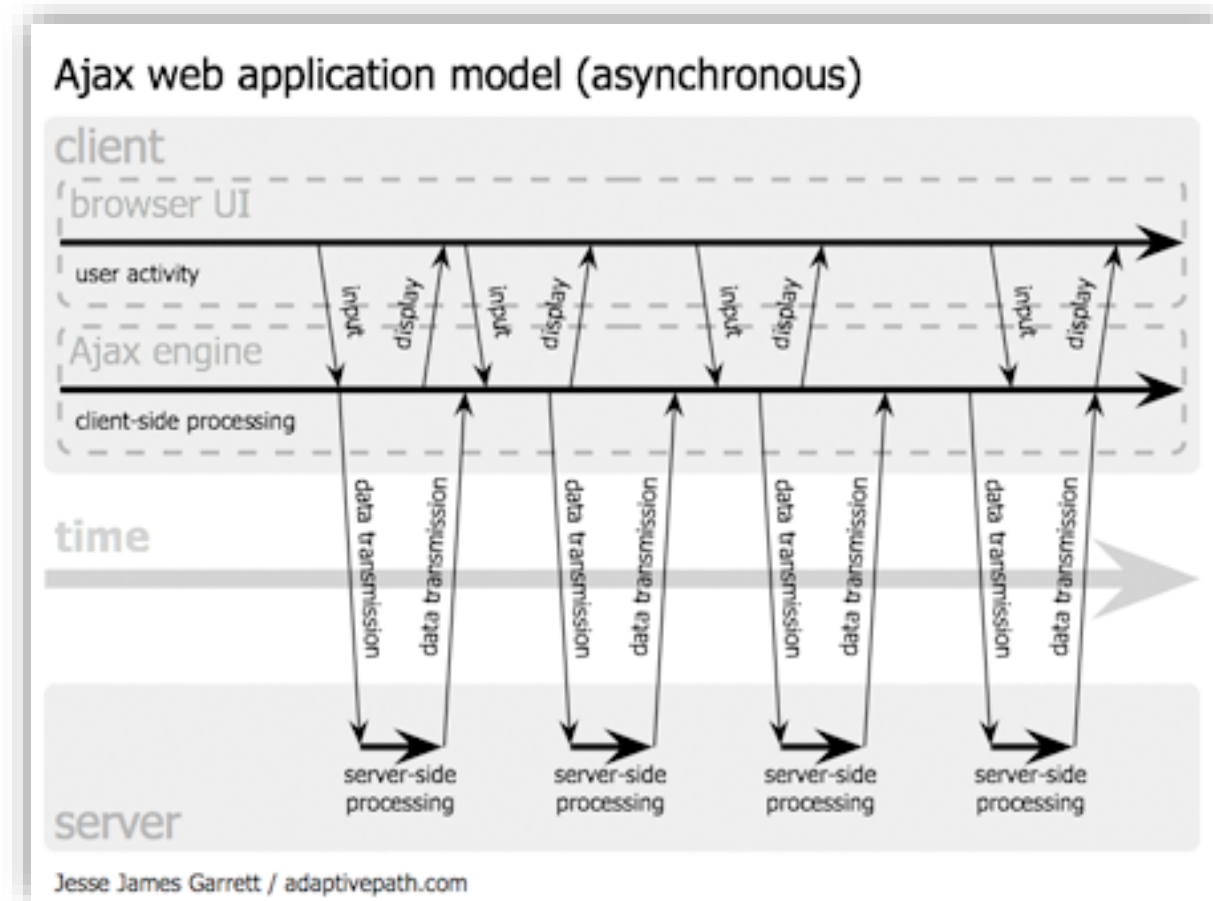
 Ads by Google

Peter Answers 3.5 - Virtual Tarot - Original idea: Wizard 666 - [Privacy Policy](#)

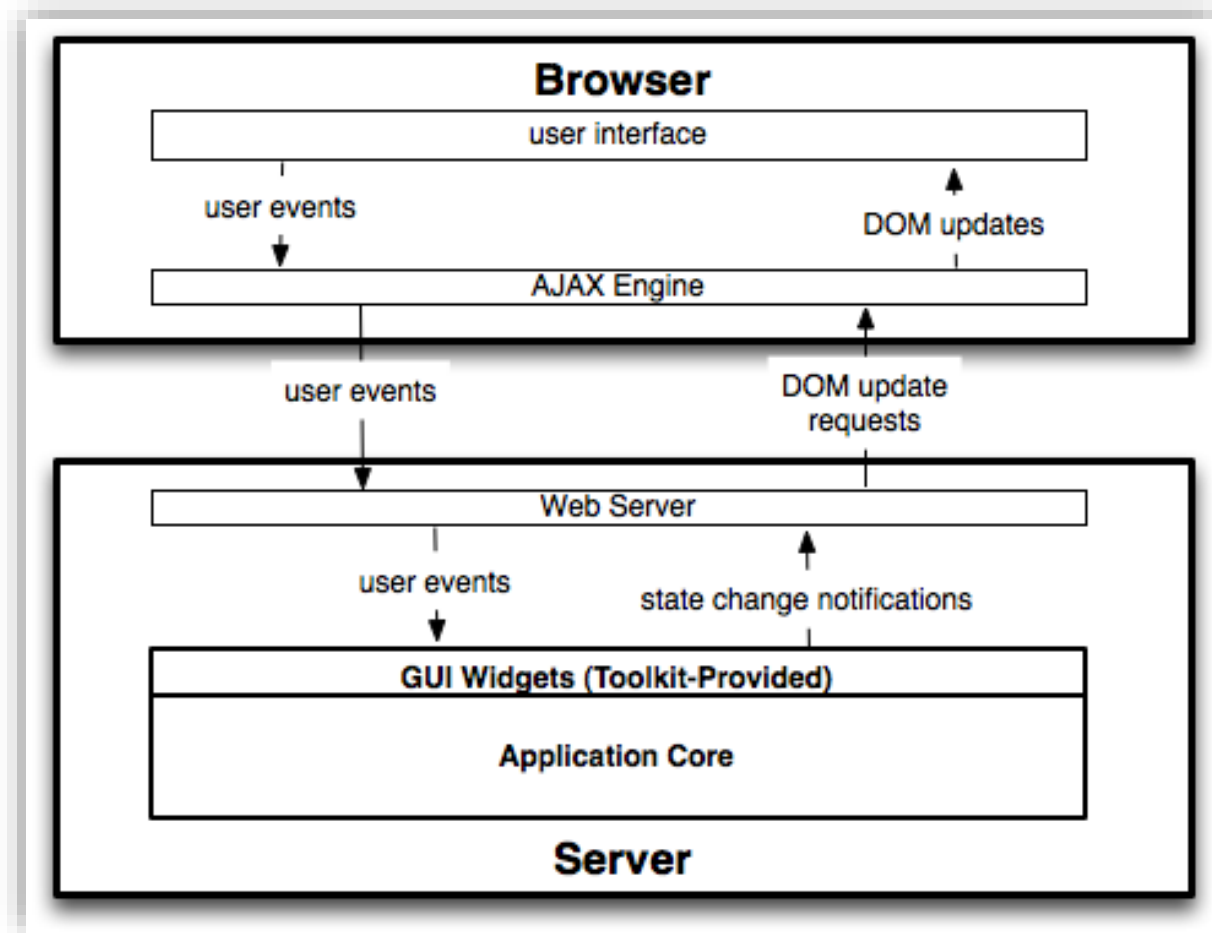
AJAX: what is that?



AJAX: event handling



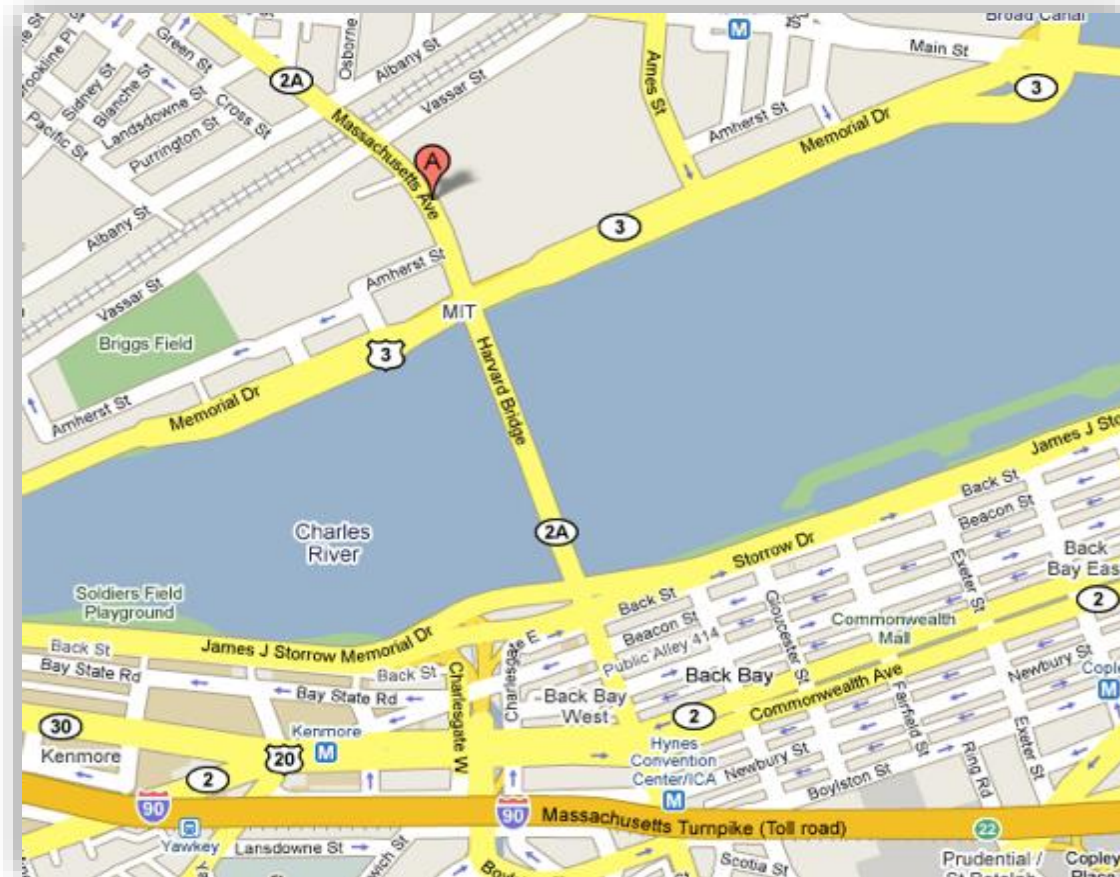
AJAX: thin client approach



- You write your application in the application core. Network-agnostic server application
- You manipulate your widgets on the server side.
- Widgets translate state changes into DOM updates
- Stateful server: All the interesting stuff in the “Application Core”
- You’re basically writing something that runs entirely in the server.

AJAX: thin client approach – the example

- Google maps → pretending that it's a thin client app



- Google Maps is NOT implemented with a thin client approach, but let's pretend that it was.

AJAX: thin client approach – the example (continued)

- A grid of images

77 Massachusetts Ave
Cambridge, MA 02139

Selected businesses at this address:

- [Laboratory For Nuclear Science](#) Place removed
- [Massachusetts Institute of Technology \(MIT\) - 2 reviews](#)
- [Massachusetts Institute of Technology: Independent Activities Period](#)
- [Mit Libraries](#)
- [Mit Medical - 2 reviews](#)
- [Rotch Library](#)
- [Science Library](#)
- [Sloan Management Review](#)
- [Spectroscopy Laboratory](#)
- [Zelas Laurie E](#)

[All businesses at this address >](#)

```
Inspect Edit  img < div < div < div < div#tileContainer < div#map < div#main_map < div#page < body.leaf < html
Console HTML CSS Script DOM Net Options Sty
z=16&s=Galileo"/>




```

AJAX: thin client approach – the example (continued)

- Google maps



- Let's suppose something fairly simplistic – a 3x3 grid of images.
- What happens if you drag the map up and to the left?

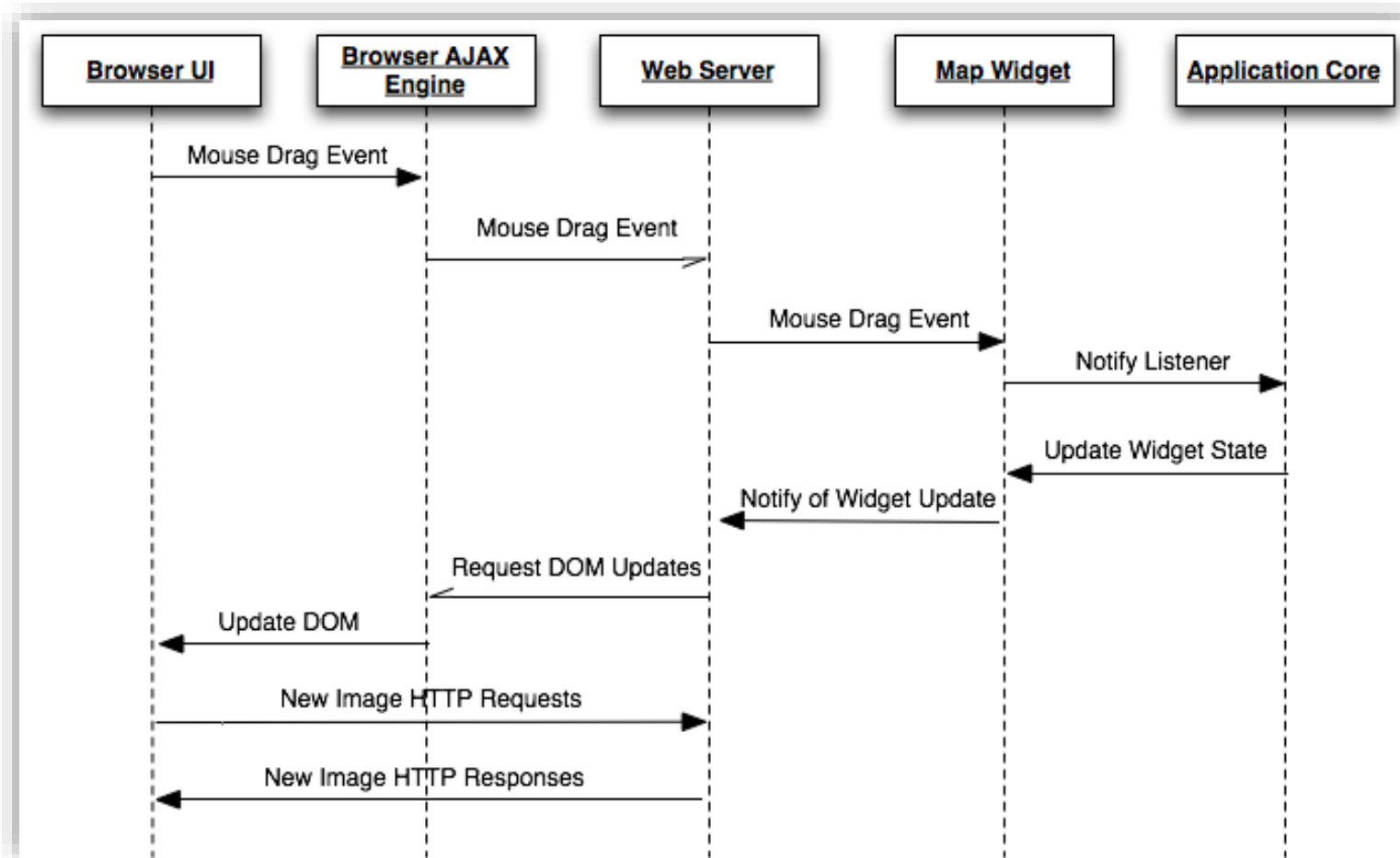
AJAX: thin client approach – the example (continued)

- Google maps



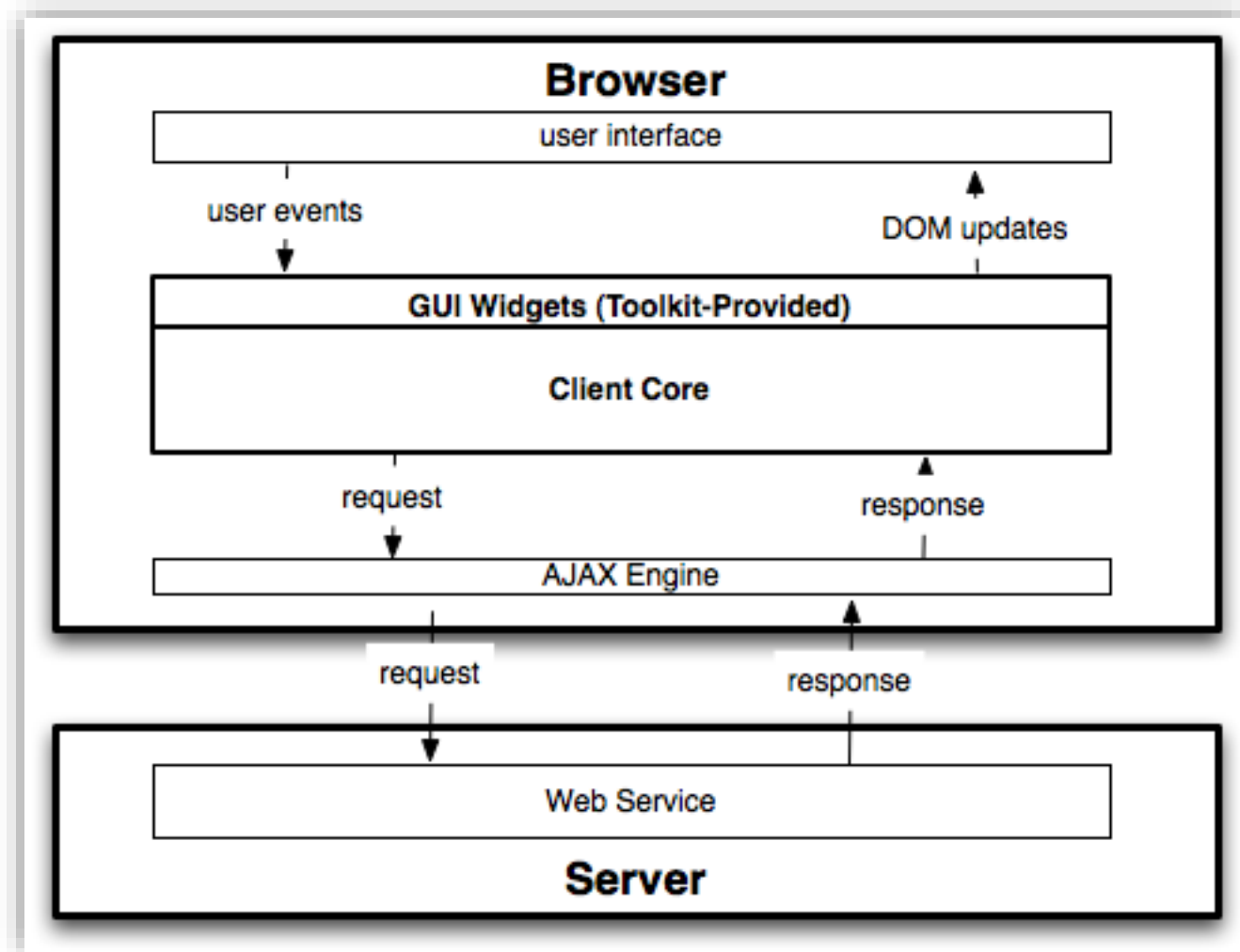
- 1) Top left of your original map goes off screen.
- 2) Shift existing images up and left.
- 3) Place images 10-16 on the screen.
- 4) Download those images. (Separate from previous step)

AJAX: thin client approach – the sequence



- “Update widget state” to include new images, new positions
- Note async and sync calls
- The first several requests just have to do with repositioning the DOM.
- Note that the images themselves will be fetched in a separate HTTP request

AJAX: fat client approach



- You write your application in the client core.
- You manipulate your widgets on the client side.
- Network-aware client application: If you need something from the server, you make a service call.
- Stateless server. It might just be a web service (or several web services)

AJAX: fat client approach (continued)



- Shift existing images.
- Download images 10-16 and place them on the screen.

AJAX: fat client approach – the sequence

