Intro to Web App Development

CS 4640
Programming Languages for Web Applications

[Robert W. Sebesta, “Programming the World Wide Web
Jon Duckett, Interactive Frontend Web Development]
What is the Web?

- A set of standards or infrastructures for distributing information to the world
  - TCP/IP
  - HTTP
  - URL
  - HTML
  - CSS
  - ...

Hypertext and the WWW

1945: Vannevar Bush proposes hypertext
1965: Ted Nelson coins the term “Hypertext”
1969: ARPANET comes online
1980: Tim Berners-Lee writes ENQUIRE
1989: Tim Berners-Lee’s Information Management proposal became WWW
1990: HTML defined
1992: CERN (Switzerland) releases WWW
1993: First browser: NCSA Mosaic
1994: First widely used commercial browser: Netscape
1997: More than 31,000,000 pages
2000: More than 100,000,000 hosts, more back-end programming than front-end hypertext
2004: 3,307,998,701 pages (Google)
2017: More than 4,6200,000,000 pages (Google) and large number of web app failures
How do Web Apps fit in with the World Around Them?

Object type: **Hotel**

Object type: **Car**

Object type: **Car**

Object type: **Car**
Objects and Properties

Object type: Hotel

Properties
Name: Awesome  
Rating: 5  
Rooms: 70  
Bookings: 56  
Pool: true  
Gym: true

Object type: Car  
Make: UVA1  
currentSpeed: 30mph  
Color: yellow  
Fuel: gasoline

Object type: Car  
Make: UVA2  
currentSpeed: 20mph  
Color: red  
Fuel: gasoline

Object type: Car  
Make: UVA2  
currentSpeed: 35mph  
Color: blue  
Fuel: gasoline
Objects and Events

Object type: **Hotel**

<table>
<thead>
<tr>
<th>Event</th>
<th>Happens when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserve</td>
<td>reservation is made</td>
</tr>
<tr>
<td>Cancel</td>
<td>reservation is cancelled</td>
</tr>
</tbody>
</table>

Object type: **Car**

<table>
<thead>
<tr>
<th>Event</th>
<th>Happens when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break:</td>
<td>driver slows down</td>
</tr>
<tr>
<td>Accelerate:</td>
<td>driver speeds up</td>
</tr>
</tbody>
</table>

Object type: **Car**

<table>
<thead>
<tr>
<th>Event</th>
<th>Happens when</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break:</td>
<td>driver slows down</td>
</tr>
<tr>
<td>Accelerate:</td>
<td>driver speeds up</td>
</tr>
</tbody>
</table>
Objects and Methods

Object type: Hotel

**Method**
- `makeReservation()` increases value of `bookings` property
- `cancelReservation()` decreases value of `bookings` property
- `checkAvailability()` subtracts value of `bookings` property from value of `rooms` property and returns number of rooms available

Object type: Car

**Method**
- `changeSpeed()` increases or decreases value of `currentSpeed` property
## All Together

### Object type: Hotel

<table>
<thead>
<tr>
<th>Event</th>
<th>Happens when</th>
<th>Method called</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserve</td>
<td>reservation is made</td>
<td>makeReservation()</td>
<td>Name: Awesome</td>
</tr>
<tr>
<td></td>
<td>reservation is cancelled</td>
<td>cancelReservation()</td>
<td>Rating: 5</td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td></td>
<td>Rooms: 70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bookings: 56</td>
</tr>
</tbody>
</table>

**Method**
- makeReservation()
- cancelReservation()
- checkAVailability()

**What it does**
- makeReservation() increases value of `bookings` property
- cancelReservation() decreases value of `bookings` property
- checkAVailability() subtracts value of `bookings` property from value of `rooms` property and returns number of rooms available

### Object type: Car

<table>
<thead>
<tr>
<th>Event</th>
<th>Happens when</th>
<th>Method called</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Break</td>
<td>driver slows down</td>
<td>changeSpeed()</td>
<td>Make: UVA1</td>
</tr>
<tr>
<td>Accelerate</td>
<td>driver speeds up</td>
<td>changeSpeed()</td>
<td>currentSpeed: 30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Color: yellow</td>
</tr>
</tbody>
</table>

**Method**
- changeSpeed()

**What it does**
- changeSpeed() increases or decreases value of `currentSpeed` property

**Properties**
- Fuel: gasoline
Web Browsers and Objects

Object type: **Window**

Properties
Location: http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html

Object type: **Document**

Properties
URL: http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html
lastModified: 01/02/2017 10:19:23
Title: CS 4640 (Programming Languages for Web Apps)
BOM: Browser Object Model

BOM – collection of objects that the browser makes available to us for use with JavaScript

window Object

location Object, history Object, document Object, navigator Object, screen Object

form Object, link Object, image Object

DOM
## DOM: Document Object Model

### Object type: Document

#### Properties
- **URL:** [http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html](http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html)
- **lastModified:** 01/02/2017 10:19:23
- **Title:** CS 4640 (Programming Languages for Web Apps)

#### Event
- **Load**: page and content have finished loading
- **Click**: user clicks the mouse over the page
- **Keypress**: user presses down on a key

#### Method
- **write()**: adds content to the document
- **getElementById()**: accesses an element of a given id attribute
How A Browser See A Web Page

The browser receives an HTML page

It creates a model of the page and stores it in memory

It shows the page on screen using a rendering engine

The browser receives an HTML page

Do users look at web apps the way they are? Or do users look at web apps the way they think?

CS 4640: Programming Languages for Web Applications
Syllabus — Spring 2018

Instructor
contact information

Upsorn Praphamontripong
Office: Rice Hall 206
Email: upsorn
Office Hours: TBD

TA contact information

TBD

Class hours

Tuesday, Thursday 2:00pm - 3:15pm, Olsson Hall 009

Prerequisite

CS 2150
Programming skills are required and students are expected to know HTML and Java or learn on their own.

Overview

This course presents the fundamentals of software development of web applications. The front end development, back end development, user interface design, and models for web-based information retrieval and processing are

http-equiv
content-type
Content
text/html
...
HTTP: HyperText Transfer Protocol

**HTTP Request**

```
GET /~up3f/cs4640/syllabus.html HTTP/1.1
Host: cs.virginia.edu
Accept: text/html
...
```

**HTML returned content**

```
<html>
<head><title>CS 4640</title></head>
<body>...
```

**Web server**

**HTTP Response**

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
...
```

**Request line**

```
GET /~up3f/cs4640/syllabus.html HTTP/1.1
Host: cs.virginia.edu
Accept: text/html
...
```

**Response status codes:**

1xx Informational
2xx Success
3xx Redirection
4xx Client error
5xx Server error

**Common MIME types:**

application/json
application/pdf
image/png
URI: Uniform Resource Identifier

URI: `<scheme>://<domain><path>?<query>`

http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html

Use HTTP scheme
(Other popular schemes: ftp, mailto, file)

Connect to cs.virginia.edu
May be host name or an IP address
Optional port number (e.g., :8080 for port 8080)
e.g., http://localhost:8080/myproject/register.php

Request resource
Stored in up3f/cs4640 folder

URI, URL, and URN

URI: Uniform Resource Identifier
Specify resource either by location or by name, or both

URL: Uniform Resource Locator
Specify resource by location

URN: Uniform Resource Name
Specify resource by name

http://www.domain.com:1234/path/to/resource?p1=v1&p2&v2

- protocol
- domain (or host)
- port
- resource path
- query
General Web Terminology

- **Web page**: Data that fits in one browser screen
  - **Static**: HTML exists as a file on a computer
  - **Dynamic**: Created as needed

- **Web site**: A collection of connected web pages

- **Web application**: A program that is deployed on the web
  - User interface (UI) is in HTML
  - User interacts through HTTP's request / response cycle
Static Web Pages

- URL corresponds to directory location on server
- Server responds to HTTP request by returning requested files or documents

- Advantages
  - Simple

- Disadvantages
  - No interactivity
Dynamic Web Pages

• Server responds to HTTP request by running a program that processes the request and produces the response

• Different content is displayed each time the web page is viewed

• Two types of dynamic web page
  • Client-side scripting
  • Server-side scripting
Architectural Styles

• How to partition a system
• How components identify and communicate with each other
• How information is communicated
• How elements of a system can evolve

Web architectural styles constantly change
The Web Today

• Increasingly reliance

• Modern web applications are
  • Distributed (world-wide)
  • Heterogeneous (hardware and software)
  • Highly user interactive
  • Built on new technology
  • Evolve from one architecture style to another, combine multiple styles
    • Newer architectural styles are not always better – more complex and may be overkill for simple sites

• The software is
  • Very loosely coupled
  • Written in multiple languages
  • Often generated dynamically
Important Quality Attributes

1. Reliability
2. Usability
3. Security

Customers have little “site loyalty” and will switch quickly, thus time to market is much less important than in other application areas.

4. Availability
5. Scalability
6. Maintainability
7. Performance & Time to market
8. ...

(but still important!)
Summary

• Web sites and web apps are now too complicated for individuals to manage

• They need to be engineered by teams of people with diverse talents:
  • Programming skills
  • Graphics design
  • Usability
  • Information layout and engineering
  • Data communications
  • Database

We need web site engineering
Extra slides

(in case you are curious or don’t want to wait until we talk about “Web software model”)

Static Web Pages

• URL corresponds to directory location on server

• Server responds to HTTP request by returning requested files or documents

• Advantages
  • Simple

• Disadvantages
  • No interactivity
Dynamic Web Pages

- Server responds to HTTP request by running a program that processes the request and produces the response

- Different content is displayed each time the web page is viewed

- Two types of dynamic web page
  - Client-side scripting
  - Server-side scripting
Client-Side Scripting

- Generate HTML on the client through scripts

Example: JavaScript

```html
<html>
<head>
  <title>Example: JavaScript to create a table of factorials</title>
</head>
<body>
  <script type="text/javascript">
    document.write("<h2>Table of Factorials</h2>"endencies);
    for (i = 1, fact = 1; i < 10; i++, fact *= i) {
      document.write(i + "! = " + fact);
      document.write("<br>");
    }
  </script>
</body>
</html>
```

- Advantages
  - Interactivity, input validation, customization, improving usability

- Disadvantages
  - Browser compatibility
Server-Side Scripting

• Generate HTML on the server through scripts

• Early approaches emphasized embedding server code inside HTML pages

• Examples: PHP, JSP

```
<html>
<head>
    <title>Login example</title>
</head>
<body>
    You logged in as <b><font color="green">$name</font></b> with password <b><font color="green">$pwd</font></b>
</body>
</html>
```

```
<%@ page language="java" %>  
<!-- Set global information for the page -->
<!-- Declare the variable -->
%! int count = 0; %>
<!-- Scriptlet - Java code -->  
%
for (int i = 0; i < 10; i++)
{
    count = count+1;
%
<br />
The counter value is: %= count %>
</body>
</html>
```
Advantages
Hidden script, no plug-ins download

Disadvantages
Mixing logic and presentation
Server-Side Framework

• Structure server into tiers, organizes logic into classes
• Execution on the server
• Can be single-page or multiple pages with page-centric design, dispatcher design, model-view-control design, or combination of any architectures
• Example: JSP (with separation of concerns)

<table>
<thead>
<tr>
<th>Presentation</th>
<th>HTML and JSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic</td>
<td>Java classes (non-servlet)</td>
</tr>
<tr>
<td>Data</td>
<td>Servlets and beans</td>
</tr>
<tr>
<td>• Data content</td>
<td>Data structure, beans, Java classes</td>
</tr>
<tr>
<td>• Data representation</td>
<td>Data storage</td>
</tr>
<tr>
<td>• Data storage</td>
<td>Database and files, Oracle, SQL</td>
</tr>
</tbody>
</table>
Server-Side Framework Site

Advantages
- Separation of concerns, maintain and reuse

Disadvantages
- Need to load an entire page to get new data
Single Page Application (SPA)

- Client-side logic sends messages to server, receives response
- Logic is associated with a single HTML page, written in JavaScript
- HTML elements dynamically added and removed through DOM manipulation
- Enabling technologies
  - AJAX
  - DOM Manipulation
  - JSON
  - Jquery
  - AngularJS (include and routing are useful)
- Example: Gmail
### Single Page Application Site

<table>
<thead>
<tr>
<th>Browser</th>
<th>HTML elements</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTML</strong></td>
<td>event</td>
<td><strong>HTML elements</strong></td>
</tr>
<tr>
<td><strong>Web server</strong></td>
<td><strong>HTTP request</strong></td>
<td><strong>HTTP response</strong></td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td><strong>Presentation</strong></td>
<td><strong>Logic</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Data</strong></td>
</tr>
</tbody>
</table>

#### Advantages
- Fast (load most resources once; only data are transmitted)

#### Disadvantages
- Hard to maintain and reuse, cross-site scripting, memory leak
Front End Frameworks

- Client is organized into separate components, capturing model of web application data
- Components separate logic from presentation
- Components dynamically generate corresponding code based on component state
- Example: Angular
Front End Framework Site

Advantages
- Code organization, reuse, quick and easy to develop

Disadvantages
- Duplicate logic in client and server