Welcome to CSE 330/503
Creative Programming and Rapid Prototyping

Course Information

- **Instructor**
  - Todd Sproull
  - todd@wustl.edu
  - Jolley 536
  - Office Hours by Appointment
- **Classroom**
  - Lecture free format, we will meet in lab for every class
  - Video lectures are posted on the course website calendar
  - Watch video before attempting lab
- **Time**
  - Monday and Wednesday 10 – 11:30 AM
- **Course Website**
  - http://research.engineering.wustl.edu/~todd/cse330/
- **Labs**
  - Urbauer Rooms 214, 215, 216, 218, and 222
- **Head TA**
  - Andrew McNeel
Grading

- 6 modules and a Creative Project to complete during the semester
- Most modules contain individual and group assignments
- Modules are due by the end of class on the due date
- You must “commit” the module by the end of class to receive credit
  - Otherwise it is a 0
- CSE 503S students will also complete a performance evaluation study

What is this class all about?

- A tour of Web 2.0 technologies
  - Cloud Computing
    - Amazon EC2
  - LAMP
    - Linux
    - Apache
    - MySQL
    - PHP
  - Python
  - Javascript
What is Cloud Computing?

Cloud computing is using the Internet to access someone else's software running on someone else's hardware in someone else's data center.

- Lewis Cunningham
Types of Cloud Computing

- **SaaS**
  - Software as a Service
- **PaaS**
  - Platform as a Service
- **IaaS**
  - Infrastructure as a Service

**Software as a Service (SaaS)**

- Cloud based delivery of complete software applications that run on infrastructure the SaaS vendor manages
- Accessed over the Internet and typically charged on a subscription
- Examples
  - Gmail and Yahoo Mail
  - Google Docs
  - Box.net

SaaS
Software as a Service
**Platform as a Service – (PaaS)**

- **Features**
  - Storage
  - Databases
  - Cloud Middleware
  - Scalability

- **Examples**
  - Google App Engine
  - Amazon Web Services S3
  - Heroku

**Infrastructure as a Service – (IaaS)**

- **Features**
  - Virtualization
  - Nearly instant scalability
  - Everything is a service
  - Utility style (pay for what you use)
  - Hardware, OS, Software, Storage & Network

- **Examples**
  - Amazon Web Services (AWS)
  - EMC Fortress (Storage Cloud)
  - HP Adaptive IaaS
Amazon Elastic Cloud Computing (EC2)

- This semester we are using Amazon Web Services (AWS) to run the Linux Operating System in a virtual machine
  - We avoid purchasing 100 PCs for the course
    - Instead we have virtual machines (VM)s to use
  - These machines our hosted in the cloud
  - You connect to an instance of a particular configuration of Linux

Amazon EC2 Costs

- You are only billed for the computing resources you use

- When you are done using an instance you can “stop” it from running so you do not continue to be billed

- Free Tier available for limited use
  - Sufficient for this course
  - No need to stop a Free Tier instance for the entire semester
Free Tier

As part of AWS’s Free Usage Tier, new AWS customers can get started with Amazon EC2 for free. Upon sign-up, new AWS customers receive the following EC2 services each month for one year:

- 750 hours of EC2 running Linux, RHEL, or SLES t2.micro instance usage
- 750 hours of EC2 running Microsoft Windows Server t2.micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic, plus 2 million I/Os (with Magnetic) and 1 GB of snapshot storage
- 15 GB of bandwidth out aggregated across all AWS services
- 1 GB of Regional Data Transfer

How much does this cost?

<table>
<thead>
<tr>
<th>Region: US East (N. Virginia)</th>
<th>vCPU</th>
<th>ECU</th>
<th>Memory (GB)</th>
<th>Instance Storage (GB)</th>
<th>Linux/UNIX Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Purpose - Current Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t2.micro</td>
<td>1</td>
<td>1</td>
<td>3.75 GB</td>
<td>1 x 4 SSD</td>
<td>$0.015 per hour</td>
</tr>
<tr>
<td>t2.small</td>
<td>1</td>
<td>2</td>
<td>3.75 GB</td>
<td>1 x 4 SSD</td>
<td>$0.026 per hour</td>
</tr>
<tr>
<td>t2.medium</td>
<td>2</td>
<td>4</td>
<td>3.75 GB</td>
<td>1 x 4 SSD</td>
<td>$0.049 per hour</td>
</tr>
<tr>
<td>m3.large</td>
<td>2</td>
<td>8.5</td>
<td>7.5 GB</td>
<td>1 x 32 SSD</td>
<td>$0.140 per hour</td>
</tr>
<tr>
<td>m3.large</td>
<td>2</td>
<td>13</td>
<td>15 GB</td>
<td>2 x 40 SSD</td>
<td>$0.280 per hour</td>
</tr>
<tr>
<td>m3.large</td>
<td>8</td>
<td>36</td>
<td>80 GB</td>
<td>2 x 80 SSD</td>
<td>$0.560 per hour</td>
</tr>
<tr>
<td>Compute Optimized - Current Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1.large</td>
<td>1</td>
<td>2</td>
<td>3.75 GB</td>
<td>2 x 16 SSD</td>
<td>$0.105 per hour</td>
</tr>
<tr>
<td>c1.large</td>
<td>4</td>
<td>14</td>
<td>7.5 GB</td>
<td>2 x 40 SSD</td>
<td>$0.210 per hour</td>
</tr>
<tr>
<td>c1.medium</td>
<td>8</td>
<td>29</td>
<td>15 GB</td>
<td>2 x 80 SSD</td>
<td>$0.420 per hour</td>
</tr>
<tr>
<td>c1.large</td>
<td>16</td>
<td>50</td>
<td>30 GB</td>
<td>2 x 180 SSD</td>
<td>$0.840 per hour</td>
</tr>
<tr>
<td>c1.large</td>
<td>32</td>
<td>108</td>
<td>80 GB</td>
<td>2 x 320 SSD</td>
<td>$1.680 per hour</td>
</tr>
<tr>
<td>GPU Instance - Current Generation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g2.large</td>
<td>8</td>
<td>29</td>
<td>15 GB</td>
<td>60 GB</td>
<td>$0.860 per hour</td>
</tr>
</tbody>
</table>
Module 1 – HTML and CSS

- **HyperText Markup Language (HTML)**
  - Main “markup language” for displaying web pages in a web browser

- **Cascading Style Sheets (CSS)**
  - Language for describing the “look and feel” of a markup language (such as HTML)
HTML History

• In 1989 Tim Berners-Lee introduced three technologies that allowed documents to be distributed and read
  – HTML (HyperText Markup Language)
    • A simple language to layout documents
  – HTTP (Hypertext transfer protocol)
    • Technology that transfers a page from one computer to another
  – Browser Technology
    • Software that reads the HTML pages

What is HTML?

• Initially just a text file with a few special codes (called tags)
• Clear text, case insensitive
• Ignores white space
• Comprised of tags <tag> </tag>
  – eg <p> This is some cool content inside a paragraph tag. </p>
    • The tag and contents is called an element
    • Stuff between the tags is the elements contents
• Elements have attributes
  – Allow you to create a particular class of an element
  – You can also create a unique id for an element
HTML Version Timeline

- 1992: HTML 1.0 original proposal
- 1994: HTML 2.0
- 1996: HTML 3.2, end of browser wars
- 1997: HTML 4.0, stylesheets introduced
- 1999: HTML 4.01, everyone is happy
- 2000: XHTML 1.0, an XML version of HTML
- 2001: XHTML 1.1
- 2002: XHTML 2.0
- 2008: HTML 5.0 published as working draft
- 2011: HTML 5 “Last Call” from HTML Working Group

HTML – Fundamentals

• Document Structure

< HTML >

Header

Body

< / HTML >
HTML – Fundamentals

<html>
<head>
<title>The title of your html page</title>
</head>
<body>
<!-- your web page content and markup -->
</body>
</html>

HTML – Simple Example

<html>
<head>
<title>My first webpage</title>
</head>
<body>
Hello World
<!-- This is a boring webpage... -->
</body>
</html>
HTML – Fundamentals - Example

header
<body>
  Todd Sproull
  Here is my contact info:
</body>
header
<body>
    Todd Sproull <br> <br>
    Here is my contact info: <br>
    <ol>
    <li>Office: Jolley Hall, Room 536</li>
    <li>Email: todd@wustl.edu</li>
    <li>Phone: 314-935-7140</li>
    </ol>
</body>
<header>
  Todd Sproull <br> <br>
  Here is my contact info: <br>
  <ol>
    <li>Office: Jolley Hall, Room 536</li>
    <li>Email: todd@wustl.edu</li>
    <li>Phone: 314-935-7140</li>
  </ol>
  <img src='http://www.myserver.com/images/me.jpg' />
</header>

<a href='cse436.html'>Read about my iPhone class</a>
HTML Compliance

• We want to follow best practices and adhere to standards when possible in this course.

• W3C provides an online Markup Validation Service for us to test out our web pages.
  – http://validator.w3.org/

• All web pages developed in this course must pass this validation.

HTML and CSS Tutorials

• Plenty of really good examples available online
  – http://webplatform.org

• A basic understanding of HTML is necessary for this course.

• The goal of this course is not to teach all of the amazing aspects of web design
  – But you MUST create W3C compliant web pages.

• The header <!DOCTYPE HTML> declares an HTML 5 webpage
  – Which is what we will use in this course.
Cascading Style Sheets

- A powerful way to specify styles and formatting across all documents in a web site
- Style sheets can be specified inline or as a separate document
- Helps to keep a common look and feel

Cascading Style Sheets (CSS)

- Styles enable you to define a consistent 'look' for your documents by describing once how headings, paragraphs, quotes, etc. should be displayed.
- Style sheet syntax is made up of three parts:

  selector {property: value}

  selector = element.class
CSS

- General form:

  selector {property: value} or
  
  selector {property 1: value 1; 
            property 2: value 2; 
            ...
            property n: value n }

CSS Examples

H1 {text-align: center; 
    color: blue; 
    font: Arial, Times New Roman}

P {text-align: left; 
   color: red; 
   font-family: Tahoma, Arial Narrow; 
   font-style: italics}
Using CSS - Example Page

```html
<html>
<head>
    <title>My Page Title</title>
    <style>
        element.class {
            property: value;
        }
    
        element.class {
            property: value;
        }
    
        
    </style>
</head>

<body>
    <h1>Hello</h1>
</body>
</html>
```

Using CSS - Example Page

```html
<html>
<head>
    <title>CSS Example</title>
    <style>
        h1 { color: blue; }
    </style>
</head>

<body>
    <h1>Hello</h1>
</body>
</html>
```
Using CSS - Example Page

```html
<html>
<head>
    <title>CSS Example</title>
    <link rel="stylesheet" href="mystyle.css"/>
</head>
<body>
    <h1>Hello</h1>
</body>
</html>
```

CSS Examples

```css
h1 {text-align: center; color: blue}
a {color:green; font-family:arial,courier; font-weight:bold;}
td { align:center; background-color:grey; border-color:red;}
div {position:absolute; visibility:hidden; margin:10px}
font {color:navy; font-size:2pt; font-face:trebuchet; }
```
More CSS Examples - Classes

element.class {property:value; }

h1 {color: blue}
h1.widget {color: green; }
a {color:green; font-familay:arial,courier; font-weight:bold;}
a.menu {color:cyan; font-familay:arial,courier; font-style:italics;}

<h1> Hello </h1>
<h1 class="widget"> Hello again </h1>

Using CSS Classes - Example Page

<html>
  <head>
    <title>CSS Example</title>
    <style>
      h1 { color:blue; }
      h1.widget { color:green; }
    </style>
  </head>
  <body>
    <h1>Hello</h1>
    <h1 class="widget">Hello again</h1>
  </body>
</html>
HTML Forms

- `<form>` is just another kind of HTML tag

- HTML forms are used to create (rather primitive) GUIs on Web pages
  - Usually the purpose is to ask the user for information
  - The information is then sent back to the server

- A form is an area that can contain form elements
  - Forms can be used for other things, such as a GUI for simple programs

The `<form>` tag

- The `<form arguments> ... </form> tag encloses form elements (and probably other HTML as well)
- The arguments to form tell what to do with the user input
  - `action="url"` (required)
    - Specifies where to send the data when the Submit button is clicked
  - `method="get"` (default)
    - Form data is sent as a URL with ?form_data info appended to the end
    - Can be used only if data is all ASCII and not more than 100 characters
  - `method="post"`
    - Form data is sent in the body of the URL request
    - Cannot be bookmarked by most browsers
  - `target="target"`
    - Tells where to open the page sent as a result of the request
    - `target=_blank` means open in a new window
    - `target=_top` means use the same window
HTML Form Example

formExampleGet.html

```html
<!DOCTYPE HTML>
<head> <title> My HTML Form </title></head>
<body>
<form name="input" action="http://someWebsite.com/" method="get">
    Username: <input type="text" name="user" />
    <input type="submit" value="Submit" />
</form>
</body>
```

HTML Forms

DEMO
Get vs Post

• Mantra
  – you "must not use GET requests to make changes"

• GET should never change data on the server

• Differences:
  – http://stackoverflow.com/questions/198462/is-either-get-or-post-more-secure-than-the-other
  – http://www.diffen.com/difference/Get_vs_Post

Course Wiki
Piazza

Git
Git: A Fast Version Control System

- **Git**
  - Is *distributed*
  - Has *no master* copy
  - Has fast merges
  - Scales up
  - Convenient tools still being built
  - Safeguards against corruption

What is version control?

- **Basic functionality:**
  - keep track of changes made to files (allows roll-backs)
  - merge the contributions of multiple developers

- **Benefits:**
  - facilitates backups
  - increased productivity (vs manual version control)
  - encourages experimentation
  - helps to identify/fix conflicts
  - makes source readily available – less duplicated effort
Our First Git Repository

- `mkdir first-git-repo`
- `cd first-git-repo`
- `git init`
  - Creates the basic artifacts in the .git directory
- `echo “Hello World” > hello.txt`
- `git add .`
  - Adds content to the index
  - Index reflects the working version
  - Must be run prior to a commit
- `git commit -a -m ‘Check in number one’`

- We will cover Git in more detail in later modules

Demo of Git