Welcome to CS 80: Internet Programming

Instructor: Mark Edmonds

edmonds_mark@smc.edu

Using These Slides

`->`	next slide
`<-`	prev slide
`Esc`	global `ctrl+f`

Expectations and Success

- Turn in assignments, tests, projects on time
- Complete the project
  - Minimum of HTML5, CSS, and JavaScript. See Project online for more info
- Attend class

Project Information

- You will be required to submit a project worth 20% of your grade the last week of class
- You may work in groups of up to 4. The expected quality/effort will be based on how many group members you have
- You are only required to use HTML5 and CSS, but it is strongly encouraged to use other technologies you learn during the semester
- You are expected to work a minimum of 20 hours on this project (it will show in the final result)

Motivation

- Computer science is a huge, blossoming field
- Learning about computer science will make you more employable
  - Regardless of your field
- Applications of computer science:
  - Cloud computing, GPS, robotics, email, the web, video games, cars (self-driving or not), cell phones, etc.

Chapter 1: Introduction to Computers and the Internet

Client-side Programming

- Web applications that run on the client
  - *Client* refers to the user's device (e.g. a computer or cell phone)
- Example: you make your own webpage using HTML5
- Technologies: HTML5 (hyper-text markup language), CSS (cascading stylesheets), Javascript

Server-side Programming

- Applications that respond to client-side requests
  - *Server* refers to a machine that responds to *clients*
  - These applications respond to information coming from the client
- Example: user submits a registration form that must be saved in the server's database
- Technologies: Apache (web server), MySQL (database), PHP

HTML5

- Chapters 2 and 3
- HyperText Markup Language
- Specifies the content and the structure of a webpage

CSS

- Chapter 4
- CSS = "Cascading Style Sheet"
- Controls the presentation of the webpage
- Separates the presentation from the content and structure of the webpage
- Enables changing the view of the website

Javascript

- Chapters 6 and 7
- Javascript helps make web pages dynamic; e.g. they change with user interaction
- These interactions are referred to as *events*
  - Examples of events: losing focus on a form, hitting a button, changing the mouse location

jQuery

- Simplifies Javascript by providing expansive library
- Provides portable behavior
- Why does portable matter?
  - Not ever web browser behaves the same way
  - You've probably seen a web browser complain about compatibility issues before

Demos

https://developer.mozilla.org/en-US/docs/Web/Demos_of_open_web_technologies

A brief history of the Internet

- ARPANET
  - The first version of the Internet
  - ARPA = "Advanced Research Projects Agency"
  - The first message was sent from UCLA to Stanford
  - More universities and networks were added, slowly creating the modern Internet

A brief history of the Internet

TCP/IP

- TCP = "Transmission Control Protocol"
  - Ensures transmitted data integrity and verifies destination address
- IP = "Internet Protocol"
- Different types and configurations of networks appeared over time
  - Each one was a sub-network
- IP creates a network of networks

A brief history of the Internet

- Each computer on sub-networks and the Internet is assigned its own IP address
  - We used to use IPv4, which is 32-bit (meaning approximately four billion address)
  - We ran out of IPv4 address; we now use IPv6, which is 128-bit (containing approximately 340 follow by 36 zeros worth of address
    * We should be good for a while!

World Wide Web, HTML, HTTP

- World Wide Web is a section of the internet
- Specifically, this 'section' of the internet servers HTML (hypertext markup language) documents
- The hypertext transfer protocol (HTTP) is the communication protocol used to serve an HTML document
- A URL (uniform resource locator) specifies the location of the html document on the web.
  - Start with `http://`

Web Basics

Hyperlinks

- Link to another document
- Documents are typically HTML pages
  - Can also link to an email address via `mailto:`
  - Can also link to a file, even a local file (the local file would need to exist)

Web Basics

URIs and URLs

- URI = uniform resource identifiers
  - Can be files (`file://`), file transfers (`ftp://`),
- URL = URIs that start with `http://` or `https://`
  - Remember, `http://` = hypertext transfer protocol

Web Basics

Anatomy of a URL

- Ex: `http://www.deitel.com/books/downloads.html`
- `www.deitel.com` is the **hostname** (which is hosted on a **host** computer)
- This hostname is translated into an IP address (remember IP stands for Internet Protocol)
- `/books/downloads.html` is the resource's location (`/books`) and name (downloads.html) on the webserver

Web Basics

Anatomy of a URL

- This IP address uniquely identifies the host on the Internet
- This translation occurs by looking up the hostname on a DNS server (DNS = domain name system)

Web Basics

Anatomy of a URL

- DNS is complex, but think of it like a network of computers whose sole purpose is to facilitate this lookup
- The complicated part: these servers need to sync updates without messing the entire system up
- Hackers perform DNS attacks to trick people into giving up personal information to the wrong website (even though the site appears legitimate)

Web Basics

The request/receive process

- Web browser sends
  - `GET /books/downloads.html HTTP/1.1`
- `GET` is an HTTP command asked for a particular resource, in this case requesting
  - `/books/downloads/html`
- `HTTP/1.1` specifies the protocol and version
  - Any server that can understand HTTP version 1.1 can successfully interpret this request

Web Basics

The request/receive process

- The server responds with another message, indicating what happened with the request
  - HTTP/1.1 200 OK -> indicates success
  - HTTP/1.1 404 Not found -> indicates the resource could not be found

Web Basics

The request/receive process

- The server responds with another message, indicating what happened with the request
  - 100s = informational (continue, switching protocols)
  - 200s = success (ok, accepted, no content)
  - 300s = redirects, more action needed (multiple choices, use proxy, permanent redirect)
  - 400s = client error (bad request, unauthorized, forbidden)
  - 500s = server error (internal server error, bad gateway, loop detected)

HTTP Header

- Part of the HTTP response
- Provides additional information about the response, such as the type of information being sent (e.g. jpeg, html, etc.)
- Ends with a blank line to indicate that the content is about to start
- Then the content of the message is sent
- Two most common HTTP requests: `GET` (request information) and `POST` (send information)

Client-side scripting

- Script runs on the client
- Example: Javascript
- Problems: script execution is limited by client's browser's capabilities, security concerns
- Benefits: doesn't use bandwidth to process information (e.g. form validation can happen entirely on the client

Server-side scripting

- Scripts that run on the server
- Example: PHP, database access
- Problems: time (client may wait for server), extra bandwidth (potentially)
- Benefits: not limited by client's computer; means web designer can control what's on the page

Ajax

- Covered in detail in Chapter 16
- Allows webpages to behave like desktop applications
- Facilitated by sending requests to the server without loading an entirely new HTML document

Types of Programming Languages

- Computers only understand 0's and 1's
- When you write code, the code is eventually translated down to a language that is comprised of 0's and 1's
- Languages that are understood directly by computers (e.g. they are made of 0's and 1's) are known as *machine langauges*
- Writing code this way is very cumbersome
- People started writing English abbreviations of these 0's and 1's
  - These formed a higher abstraction of programming, known as assembly languages

Types of Programming Languages

- Machine languages still quite cumbersome, so *high-level languages* were created to provide more abstraction
  - High level languages, such as C++, Python, Java, Javascript, etc, allow programmers to write statements that are almost english

Types of Programming Languages

- Two types of high-level languages: *interpreted* and *compiled*
- *Interpreted* - program statements are translated to machine code as the program executes (as each line of code is running)
- E.g. Python, Javascript
  - Good for the web, code can start running as soon as it downloads to the client's computer

Types of Programming Languages

- *Compiled* - all program statements are converted into machine code, then the program can run
  - E.g. C++, Java
  - Generally significantly faster than interpreted languages
  - The translation to machine language happens before the program is run, instead of as the program is running

Object Oriented Programming

- This is a topic for an entire course

- How many people have heard of this before?
- How many people have used OOP before?
- Basic concept: group data and operations together logically and allow hierarchies of relationships

Object Oriented Programming

- Operations (methods): steer, accelerate, decelerate, fill up with gas, etc
- Data (attributes): speed, heading, gas tank status, odometer, etc.

Object Oriented Programming

- You don't have to know how the car precisely works to use one. You simply know that the pedal accelerates the car, brake decelerates, etc
- This is known as *information hiding*
- Another engineer has designed how the care operates, you just use the public (available) operations
  - But there are many more operations happening inside the car than you are aware

Object Oriented Programming

- We refer to the design of the car as the *class* (think of this like a blueprint)
- Once you have the design for a car, you can reuse it
  - Each "usage" is an *instantiation* of a car
- Similarly, once you have a design of a car, you can extend it to add new features
  - E.g. add upgrades, wings, etc
  - This is known as inheritance