Introduction to Modern Web Development

CST 365 – Data-Driven Web Applications
Michael Ruth, Ph.D.
Associate Professor
Computer Science & I.T.
mruth@roosevelt.edu

Objectives

- Discuss background concepts and concerns important to learning Web development including networking, important protocols, HTTP, & browsers
- Describe the role of the modern Web developer
- Explain the differences between static & dynamic web and client-side & server-side development
- Describe the importance of using frameworks
- Discuss important concepts relative to server-side Web development including n-tiered architecture
- Explain important concepts relative to client-side Web development including Web frameworks
- Describe full-stack web development & IoC

Background

- There are some things we need to know before we begin, specifically:
  - Network protocols
  - Networking
  - What is the Web?
  - Web protocols (HTTP)
  - Client/Server?

The Internet Protocols

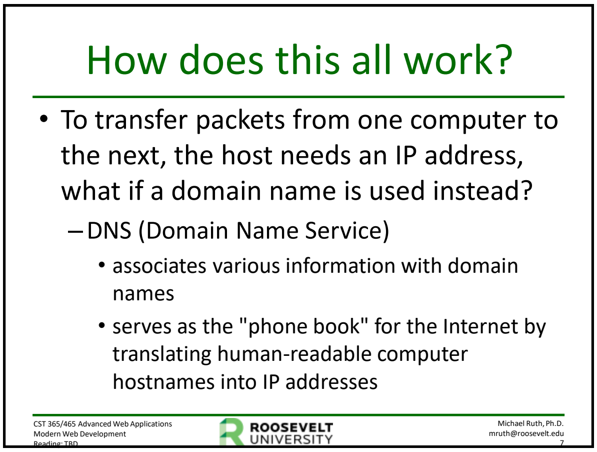
- Internet Protocol (IP)
  - Responsible for delivering packets from the source host to the destination host solely based on its address
    - Every host is identified using a unique IP address
      - 32-bits using dotted decimal notation
      - Ex: 198.168.1.1
      - Each of the four parts is called an “octet”
    - Connectionless & unreliable

More on IP hosts

- So, each host is identified by a single IP address
  - What constitutes a host?

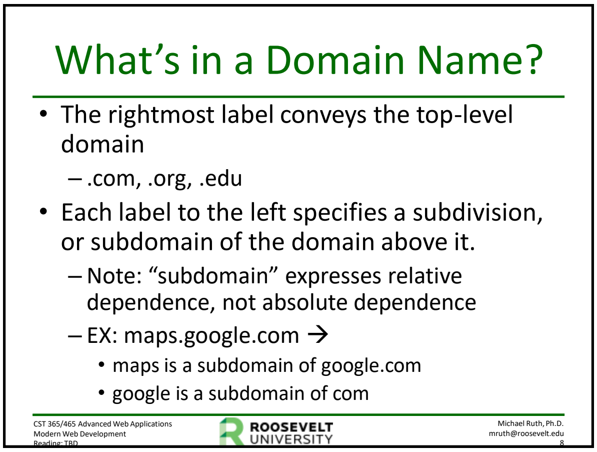
IP Addresses & People

- IP addresses are not easy, so host names are used
  - unique name by which a network-attached device is known – generally human readable
    - Ie: google.com
  - usually a combination of the host's local name with its parent domain's name
    - Ie: maps.google.com
- Also known as a “Domain Name”
  - a name given to a collection of network devices that belong to a domain which is managed according to some common property of the members or within a common administrative boundary



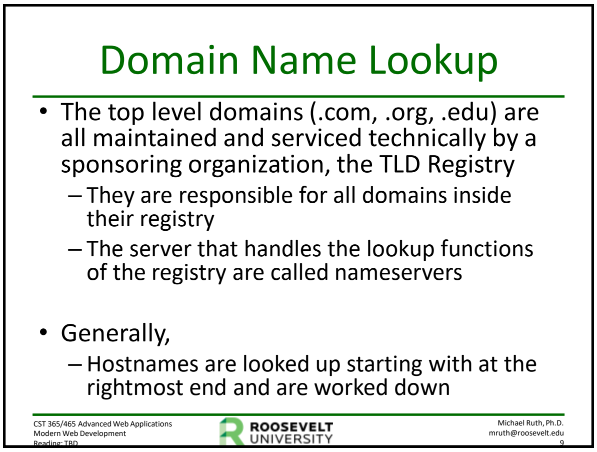
How does this all work?

- To transfer packets from one computer to the next, the host needs an IP address, what if a domain name is used instead?
  - DNS (Domain Name Service)
    - associates various information with domain names
    - serves as the "phone book" for the Internet by translating human-readable computer hostnames into IP addresses



What’s in a Domain Name?

- The rightmost label conveys the top-level domain
  - .com, .org, .edu
- Each label to the left specifies a subdivision, or subdomain of the domain above it.
  - Note: “subdomain” expresses relative dependence, not absolute dependence
  - EX: maps.google.com →
    - maps is a subdomain of google.com
    - google is a subdomain of com



Domain Name Lookup

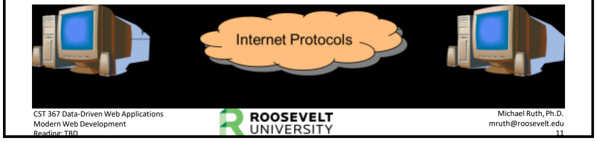
- The top level domains (.com, .org, .edu) are all maintained and serviced technically by a sponsoring organization, the TLD Registry
  - They are responsible for all domains inside their registry
  - The server that handles the lookup functions of the registry are called nameservers
- Generally,
  - Hostnames are looked up starting with at the rightmost end and are worked down

TCP & UDP

- They provide a communication service at an intermediate level between an application program and the Internet Protocol (IP)
- The difference between the two is in how they work:
  - TCP
    - Connection-oriented
    - Reliable
    - Ordered
  - UDP
    - Message-oriented
    - Best effort delivery

So what about applications?

- Suppose multiple applications are running on the same host. How do I connect to one application and not the other?
  - Each application on the host has a "port number"
    - Some are well defined: 20, 21, 25, 80, etc
    - Some are application dependent.



WWW – Some Terms

- **World-Wide Web (WWW)** – The group of worldwide connected websites
- A **Web site** is simply a collection of web pages
- A **Web page** is simply a document on the WWW which is identified by a unique address, Uniform Resource Locator or "URL"
- **URL** – The specific address of a Web page
- A **hyperlink** (or link) – the mechanism by which one navigates from one web page to another using the URL

So, What makes up the Web?

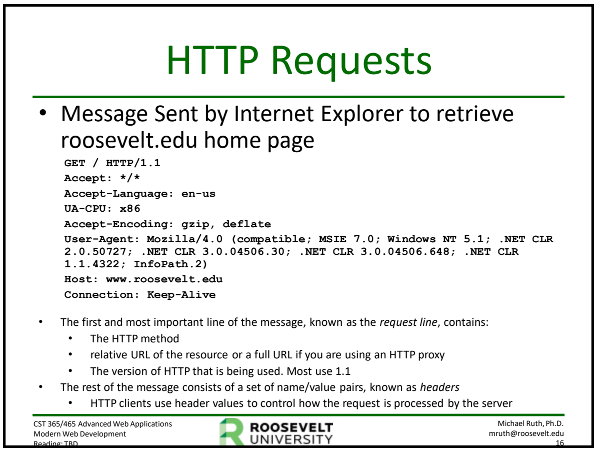
- The Web is really just a another way of thinking about the Internet and it's resources
- The Web introduced three important things that made the Internet more flexible and easy to use:
  - **Hypertext Transfer Protocol (HTTP)**
    - The protocol used to communicate on the Web
  - **URLS**
    - Addressing scheme for resources on the Web
  - **Applications (AKA Web Browsers)**
    - Hypertext Markup Language (HTML)
      - provides a means to describe the structure of web pages

Hypertext Transfer Protocol (HTTP)

- Hypertext Transfer Protocol (HTTP) is a another communications protocol for the transfer of information on the Internet.
  - Its use for retrieving linked text documents led to the establishment of the Web (or World-Wide Web)
    - Documents are linked via URLs

HTTP Protocol

- Client-server protocol
  - Client is end-user
    - Terminal end-user is also called User-agent
- Reliable data transfer (TCP on port 80)
- Request/response interaction
  - Requests: ASCII text
  - Response: three-digit status code and phrase (w/resource)
- Synchronous
  - User agent awaits response … before issuing the next request

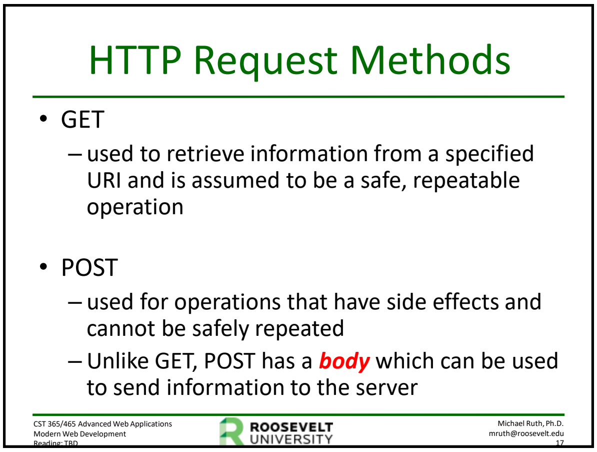


HTTP Requests

- Message Sent by Internet Explorer to retrieve roosevelt.edu home page

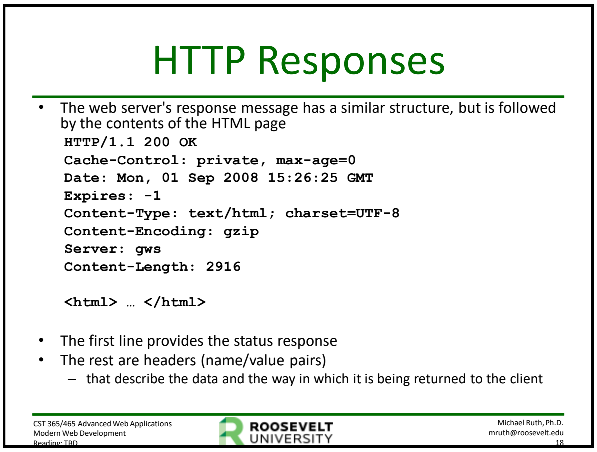
```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR
1.1.4322; InfoPath.2)
Host: www.roosevelt.edu
Connection: Keep-Alive
```

- The first and most important line of the message, known as the *request line*, contains:
  - The HTTP method
  - relative URL of the resource or a full URL if you are using an HTTP proxy
  - The version of HTTP that is being used. Most use 1.1
- The rest of the message consists of a set of name/value pairs, known as *headers*
  - HTTP clients use header values to control how the request is processed by the server



HTTP Request Methods

- GET
  - used to retrieve information from a specified URI and is assumed to be a safe, repeatable operation
- POST
  - used for operations that have side effects and cannot be safely repeated
  - Unlike GET, POST has a ***body*** which can be used to send information to the server

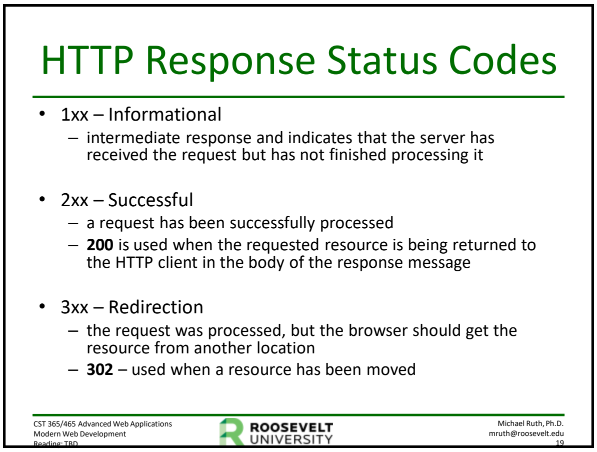


HTTP Responses

- The web server's response message has a similar structure, but is followed by the contents of the HTML page

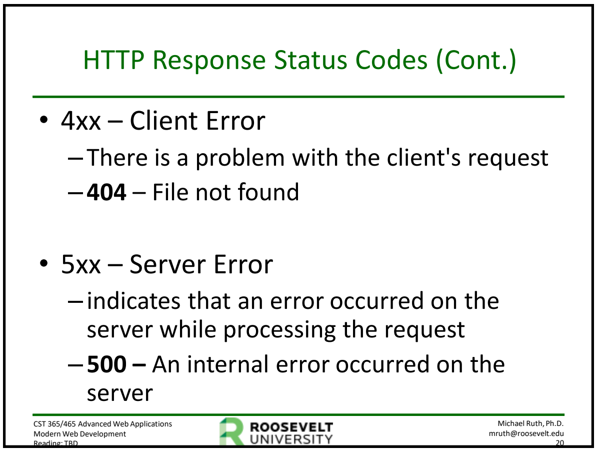
```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Date: Mon, 01 Sep 2008 15:26:25 GMT
Expires: -1
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 2916

<html> … </html>
```

- The first line provides the status response
- The rest are headers (name/value pairs)
  - that describe the data and the way in which it is being returned to the client



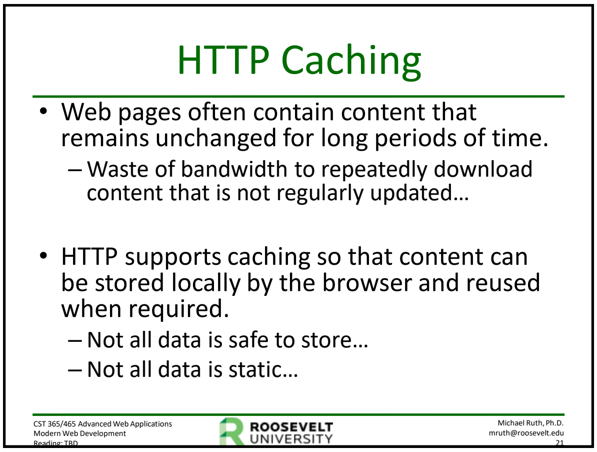
HTTP Response Status Codes

- 1xx – Informational
  - intermediate response and indicates that the server has received the request but has not finished processing it
- 2xx – Successful
  - a request has been successfully processed
  - **200** is used when the requested resource is being returned to the HTTP client in the body of the response message
- 3xx – Redirection
  - the request was processed, but the browser should get the resource from another location
  - **302** – used when a resource has been moved



HTTP Response Status Codes (Cont.)

- 4xx – Client Error
  - There is a problem with the client's request
  - **404** – File not found
- 5xx – Server Error
  - indicates that an error occurred on the server while processing the request
  - **500 –** An internal error occurred on the server



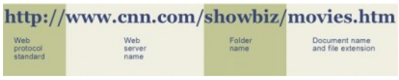
HTTP Caching

- Web pages often contain content that remains unchanged for long periods of time.
  - Waste of bandwidth to repeatedly download content that is not regularly updated…
- HTTP supports caching so that content can be stored locally by the browser and reused when required.
  - Not all data is safe to store…
  - Not all data is static…

HTTP Request/Response

Addressing URLs

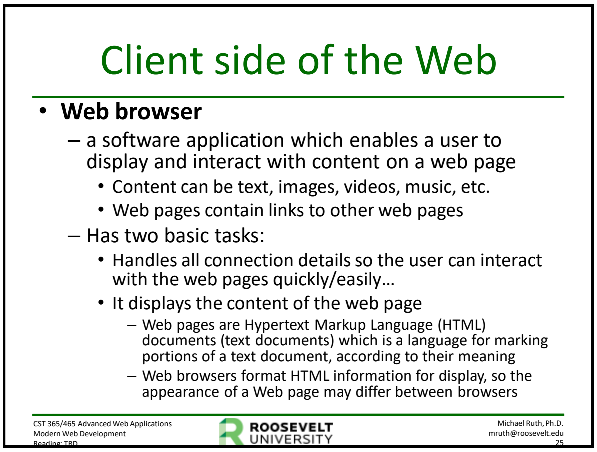
- The typical structure of a URL is:
  - **Protocol** (http://)
    - How to get the resource
  - **Web server name** (www.cnn.com)
    - Where to get the resource
  - **Folder name/Document name & Extension** (/showbiz/movies.htm)
    - Where on the host is the resource located



http://www.cnn.com/showbiz/movies.htm

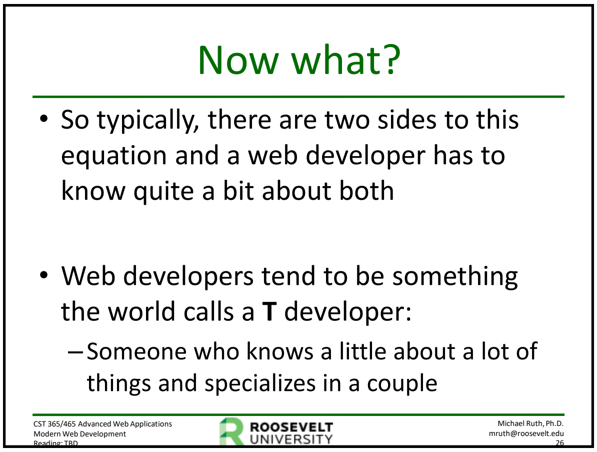
Rest of the Picture

- Remember this diagram? What happens at the client end?



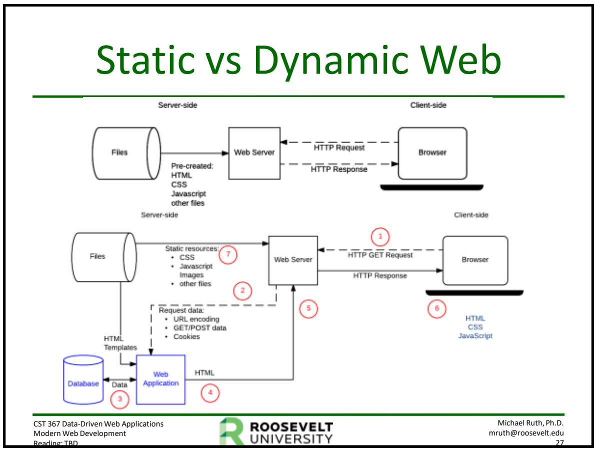
Client side of the Web

- **Web browser**
  - a software application which enables a user to display and interact with content on a web page
    - Content can be text, images, videos, music, etc.
    - Web pages contain links to other web pages
  - Has two basic tasks:
    - Handles all connection details so the user can interact with the web pages quickly/easily…
    - It displays the content of the web page
      - Web pages are Hypertext Markup Language (HTML) documents (text documents) which is a language for marking portions of a text document, according to their meaning
      - Web browsers format HTML information for display, so the appearance of a Web page may differ between browsers

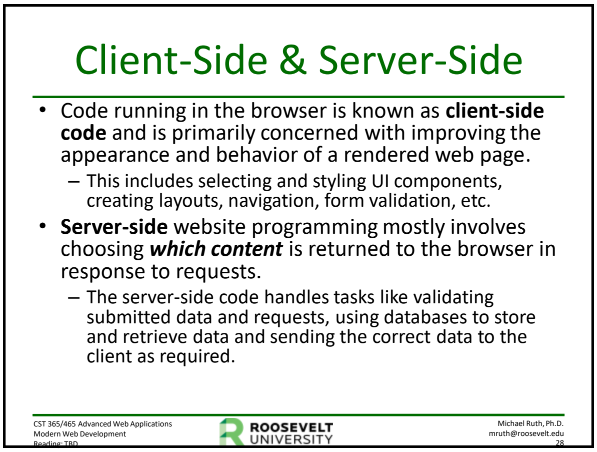


Now what?

- So typically, there are two sides to this equation and a web developer has to know quite a bit about both
- Web developers tend to be something the world calls a **T** developer:
  - Someone who knows a little about a lot of things and specializes in a couple

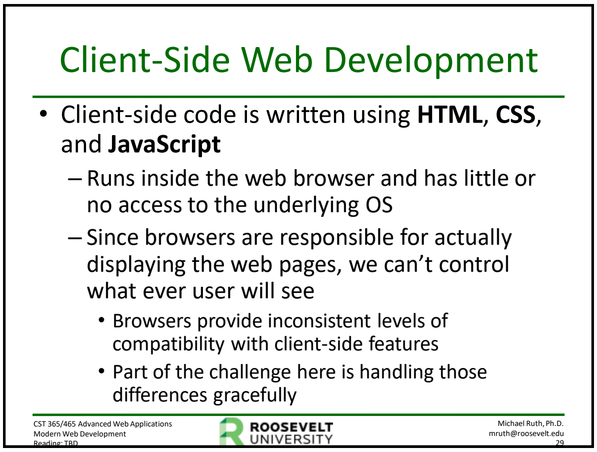


Static vs Dynamic Web



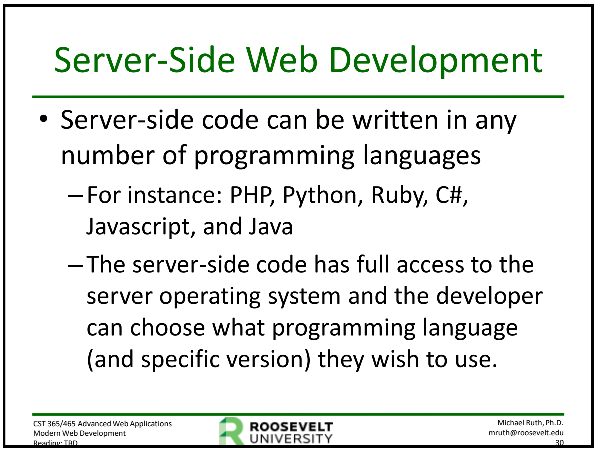
Client-Side & Server-Side

- Code running in the browser is known as **client-side code** and is primarily concerned with improving the appearance and behavior of a rendered web page.
  - This includes selecting and styling UI components, creating layouts, navigation, form validation, etc.
- **Server-side** website programming mostly involves choosing ***which content*** is returned to the browser in response to requests.
  - The server-side code handles tasks like validating submitted data and requests, using databases to store and retrieve data and sending the correct data to the client as required.



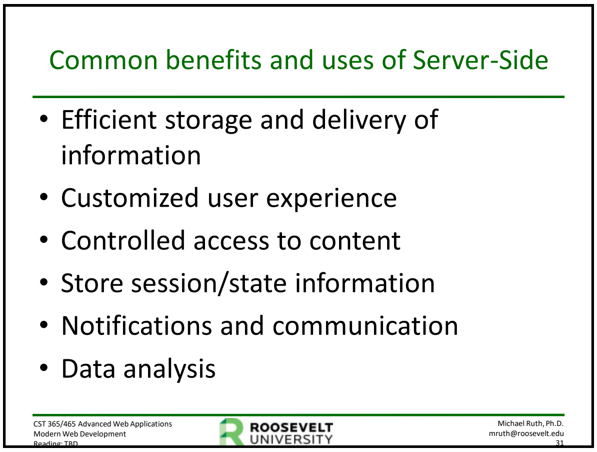
Client-Side Web Development

- Client-side code is written using **HTML**, **CSS**, and **JavaScript**
  - Runs inside the web browser and has little or no access to the underlying OS
  - Since browsers are responsible for actually displaying the web pages, we can't control what ever user will see
    - Browsers provide inconsistent levels of compatibility with client-side features
    - Part of the challenge here is handling those differences gracefully



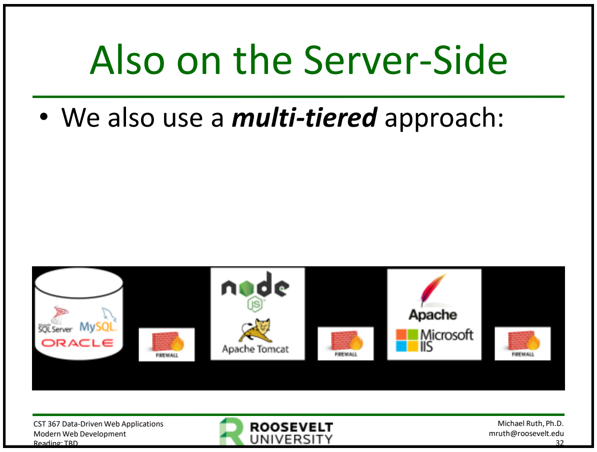
Server-Side Web Development

- Server-side code can be written in any number of programming languages
  - For instance: PHP, Python, Ruby, C#, Javascript, and Java
  - The server-side code has full access to the server operating system and the developer can choose what programming language (and specific version) they wish to use.



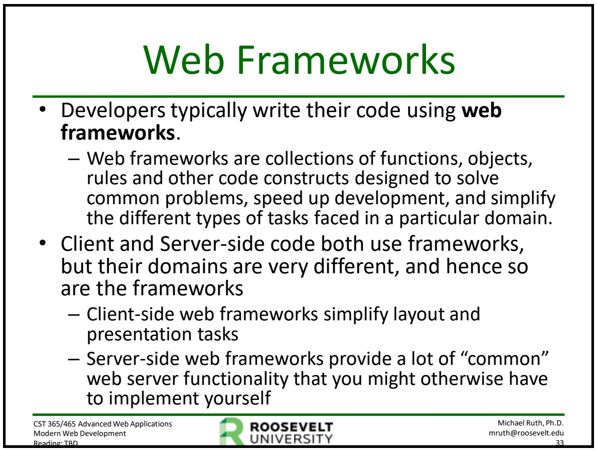
Common benefits and uses of Server-Side

- Efficient storage and delivery of information
- Customized user experience
- Controlled access to content
- Store session/state information
- Notifications and communication
- Data analysis



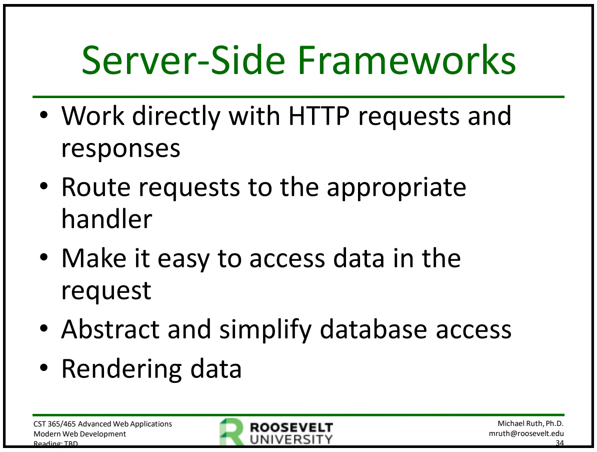
Also on the Server-Side

- We also use a ***multi-tiered*** approach:



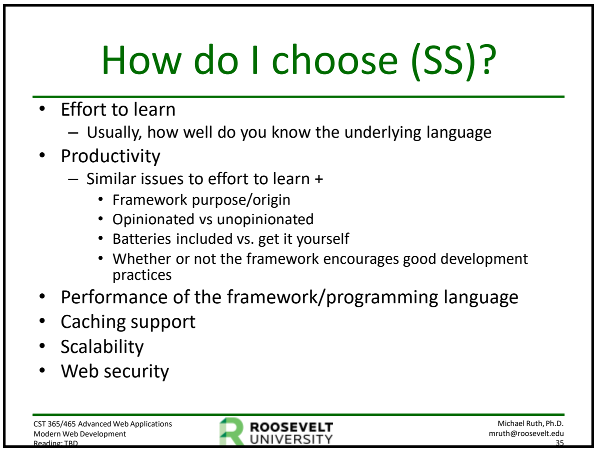
Web Frameworks

- Developers typically write their code using **web frameworks**.
  - Web frameworks are collections of functions, objects, rules and other code constructs designed to solve common problems, speed up development, and simplify the different types of tasks faced in a particular domain.
- Client and Server-side code both use frameworks, but their domains are very different, and hence so are the frameworks
  - Client-side web frameworks simplify layout and presentation tasks
  - Server-side web frameworks provide a lot of "common" web server functionality that you might otherwise have to implement yourself



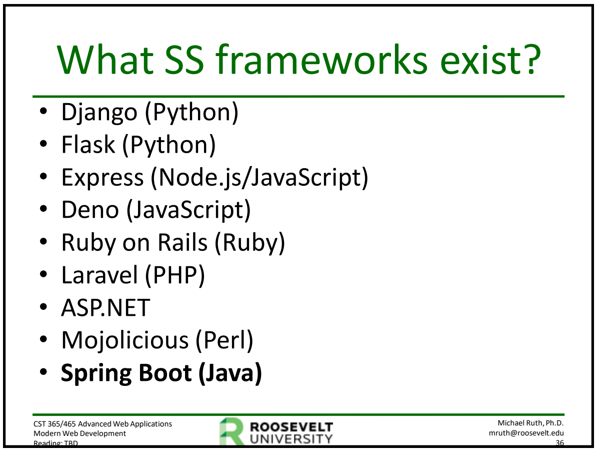
Server-Side Frameworks

- Work directly with HTTP requests and responses
- Route requests to the appropriate handler
- Make it easy to access data in the request
- Abstract and simplify database access
- Rendering data



How do I choose (SS)?

- Effort to learn
  - Usually, how well do you know the underlying language
- Productivity
  - Similar issues to effort to learn +
    - Framework purpose/origin
    - Opinionated vs unopinionated
    - Batteries included vs. get it yourself
    - Whether or not the framework encourages good development practices
- Performance of the framework/programming language
- Caching support
- Scalability
- Web security



What SS frameworks exist?

- Django (Python)
- Flask (Python)
- Express (Node.js/JavaScript)
- Deno (JavaScript)
- Ruby on Rails (Ruby)
- Laravel (PHP)
- ASP.NET
- Mojolicious (Perl)
- **Spring Boot (Java)**

Client-Side Frameworks

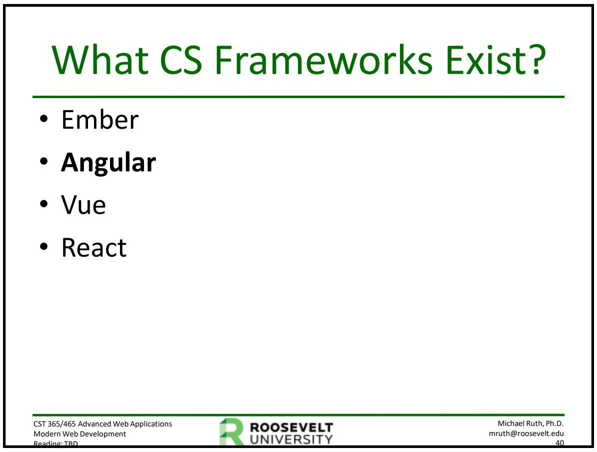
- JavaScript is an essential part of the web, used on 95% of all websites and client-side frameworks are built on JS
- The advent of modern JavaScript frameworks has made it much easier to build highly dynamic, interactive applications.
- A **framework** is a library that offers opinions about how software gets built.
  - These opinions allow for predictability and homogeneity in an application;
  - predictability allows software to scale to an enormous size and still be maintainable;
  - predictability and maintainability are essential for the health and longevity of software.

JavaScript Frameworks

- JavaScript frameworks offers a way to write user interfaces more *declaratively*
- Tooling
- Compartmentalization
- Routing

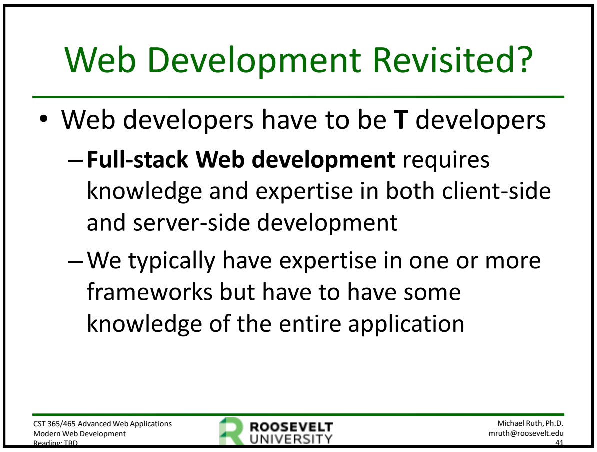
How do I choose?

- Familiarity with the tool
- Overengineering
- Larger code base and abstraction
- Questions?
  - What browsers does the framework support?
  - What domain-specific languages does the framework utilize?
  - Does the framework have a strong community and good docs (and other support) available?



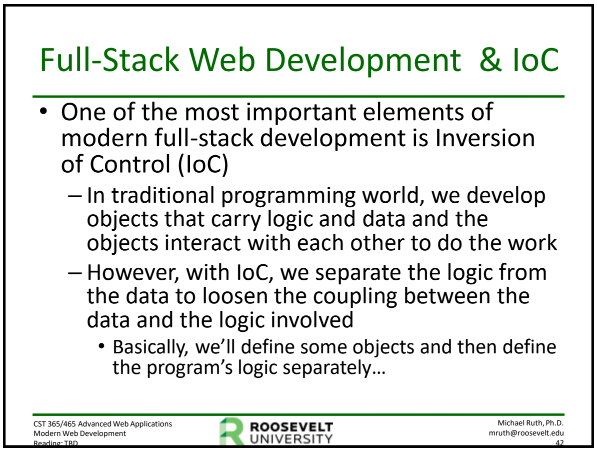
What CS Frameworks Exist?

- Ember
- **Angular**
- Vue
- React



Web Development Revisited?

- Web developers have to be **T** developers
  - **Full-stack Web development** requires knowledge and expertise in both client-side and server-side development
  - We typically have expertise in one or more frameworks but have to have some knowledge of the entire application

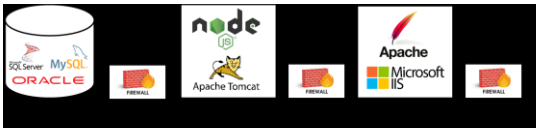


Full-Stack Web Development & IoC

- One of the most important elements of modern full-stack development is Inversion of Control (IoC)
  - In traditional programming world, we develop objects that carry logic and data and the objects interact with each other to do the work
  - However, with IoC, we separate the logic from the data to loosen the coupling between the data and the logic involved
    - Basically, we'll define some objects and then define the program's logic separately…

Full-Stack Web Development

- Let’s look at this diagram in more detail:

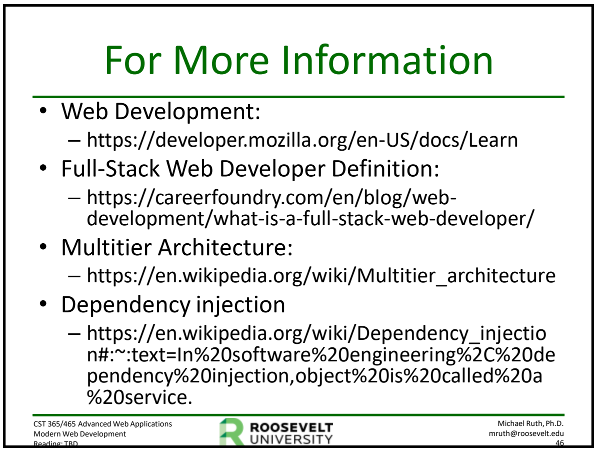


So…

- Client-side:
  - We need to learn HTML, CSS, and Javascript so we can learn to use the Angular framework
- Server-side:
  - We’ll need to learn a bit about Tomcat application servers, Servlets, JSP pages on the way to learning how to use Spring Boot

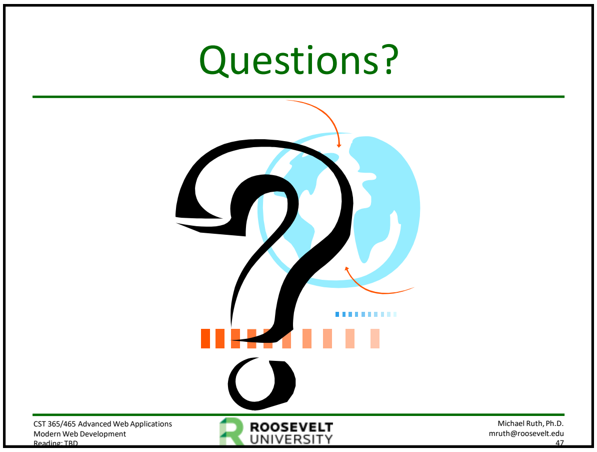
Summary

- Discussed background concepts and concerns important to learning Web development including networking, important protocols, HTTP, & browsers
- Described the role of the modern Web developer
- Explained the differences between static & dynamic web and client-side & server-side development
- Described the importance of using frameworks
- Discussed important concepts relative to server-side Web development including n-tiered architecture
- Explained important concepts relative to client-side Web development including Web frameworks
- Described full-stack web development & IoC



For More Information

- Web Development:
  - https://developer.mozilla.org/en-US/docs/Learn
- Full-Stack Web Developer Definition:
  - https://careerfoundry.com/en/blog/web-development/what-is-a-full-stack-web-developer/
- Multitier Architecture:
  - https://en.wikipedia.org/wiki/Multitier_architecture
- Dependency injection
  - https://en.wikipedia.org/wiki/Dependency_injection#:~:text=In%20software%20engineering%2C%20dependency%20injection,object%20is%20called%20a%20service.



Questions?