

# 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]

# Web and Internet

## Web

**HTML, CSS, Browser**

## Internet

Computer network

Application layer

Specifies the shared communication protocols → over network: DNS, FTP, **HTTP**, POP, SSH, Telnet, ...

Transport layer

Provides host-to-host communication services: TCP, ...

Internet layer

Transports packets from host across network boundaries: IP, ICMP, ...

Link layer

Operates on the link that a host is physically connected to: MAC (Ethernet, DSL, ...)

A set of standards or infrastructures for distributing information to the world

# Hypertext and the WWW

---

- 1945: Vannevar Bush proposes **hypertext**
- 1965: Ted Nelson coins the term "**Hypertext**" – beyond the linear constraints of text
- 1969: **ARPANET** comes online
- 1980: Tim Berners-Lee writes **ENQUIRE**, a notebook program allowing links to be made between nodes with titles
- 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)
- May-2022: More than **1,950,000,000 websites** (www.internetlivestats.com), and large number of **web app failures**

# Aspects to Consider

---

## Software engineering aspect

- How can we design for change and reuse?
- Many developers may involve
  - What happens when a new developer joins the team?
  - How can a developer successfully maintain / change / refactor the code without understanding the whole system?

## Usability aspect

- How can we design web apps that are **usable** for their intended purpose?
- A web app may serve millions of users with different needs
  - What happens when a new user interacts with the app?
  - How can we make a web app less frustrating to use?

# 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



(More information:  
[https://en.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier))

# URI, URL, and URN

**URI:** Uniform Resource **I**dentifier

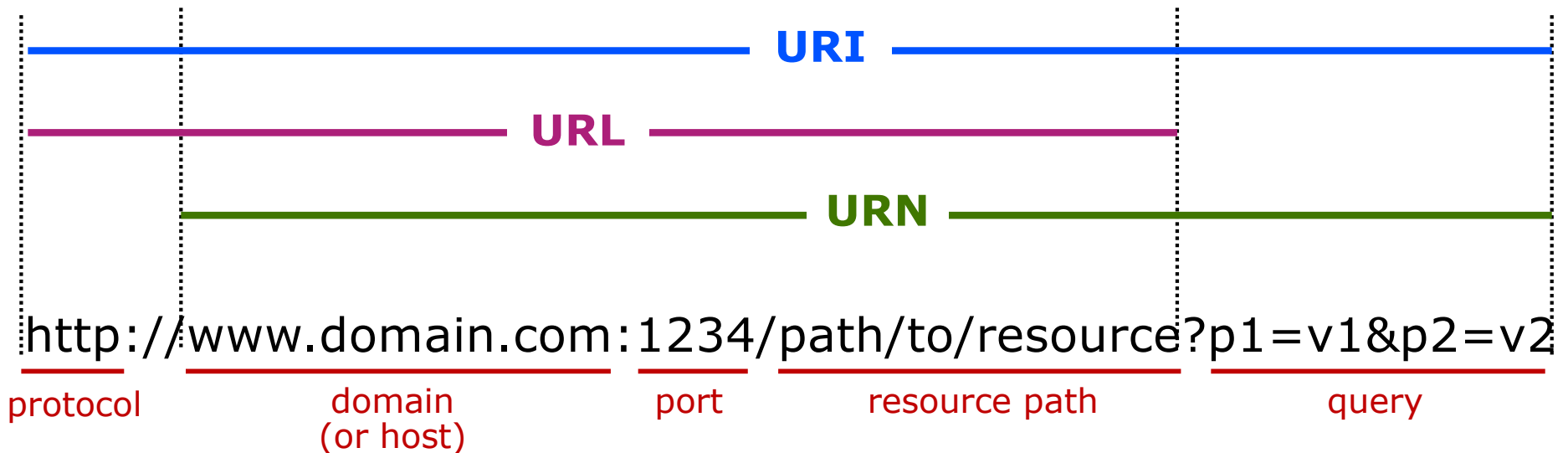
Specify resource either by location or by name, or both

→ **URL:** Uniform Resource **L**ocator

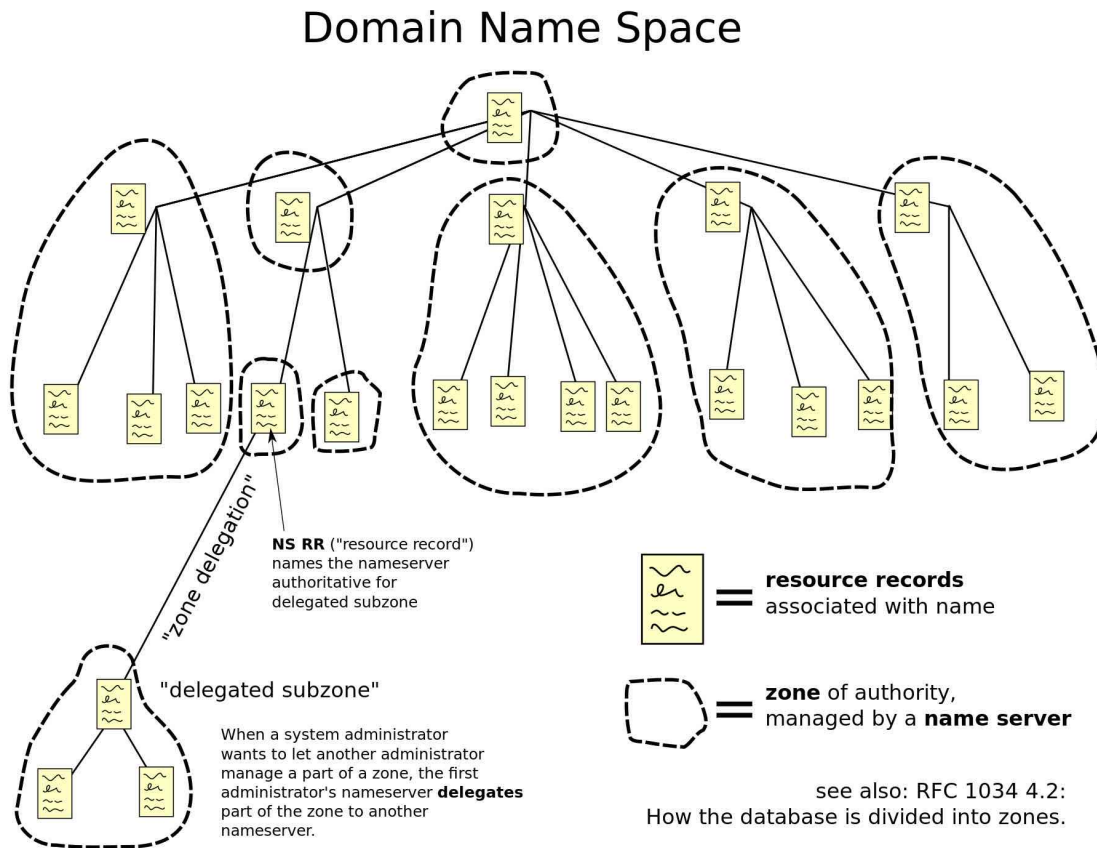
Specify resource by location

→ **URN:** Uniform Resource **N**ame

Specify resource by name



# DNS: Domain Name System



## Domain Name System

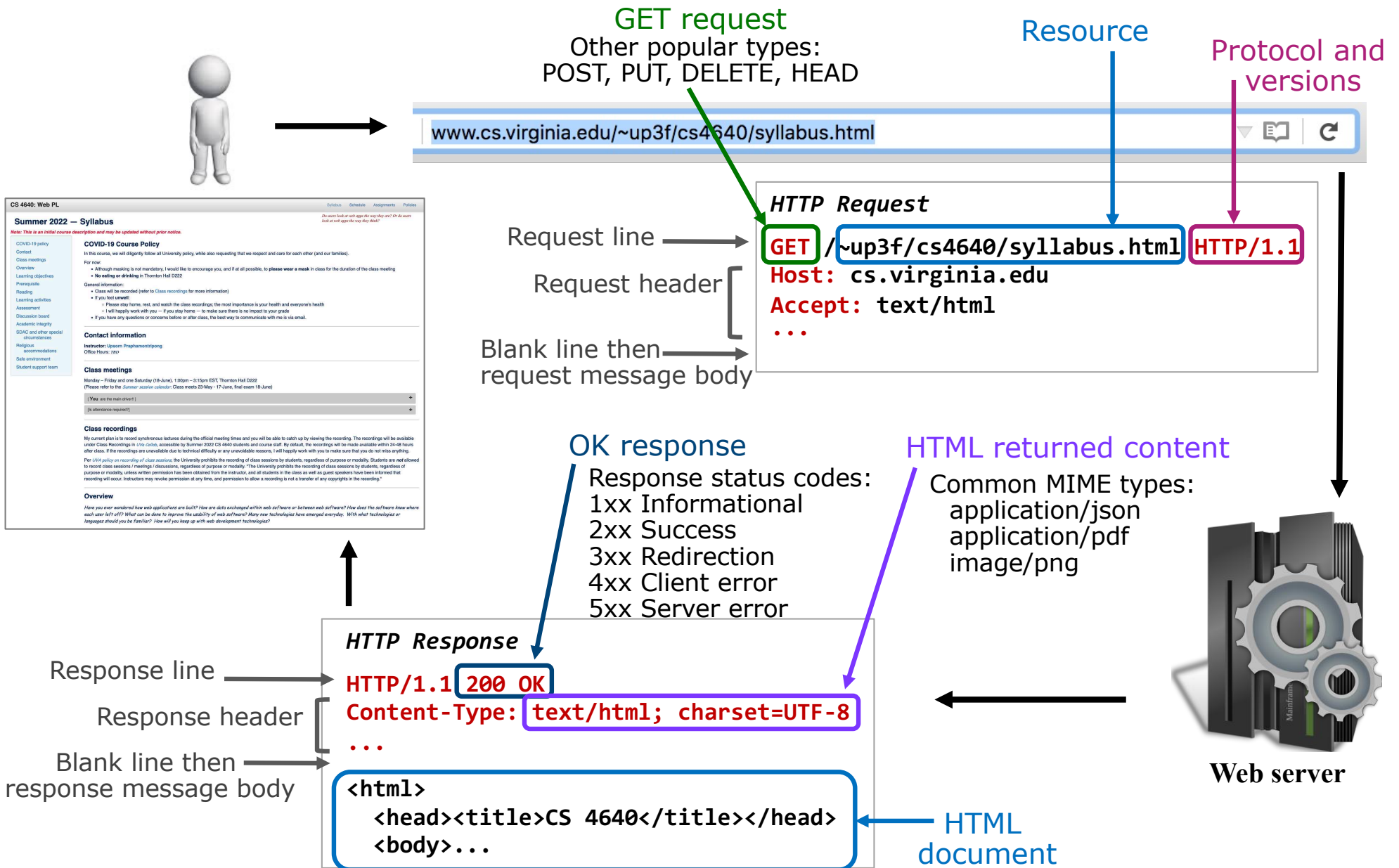
~ phonebook of the Internet

Map domain names to IP addresses

Hierarchical Domain Name System for class Internet, organized into zones, each served by a name server

[ref: [https://en.wikipedia.org/wiki/Domain\\_Name\\_System](https://en.wikipedia.org/wiki/Domain_Name_System)]

# HTTP: HyperText Transfer Protocol





# Properties of HTTP

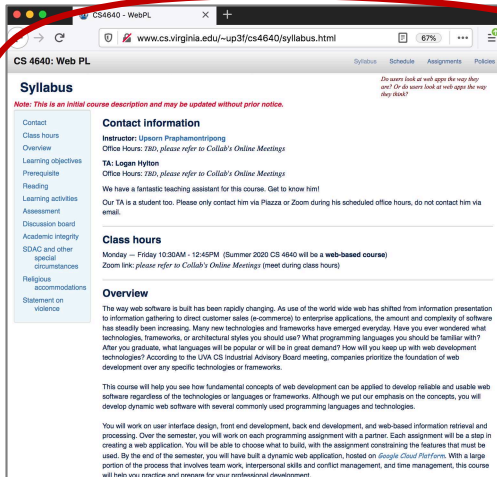
---

- **Request-response**
  - Interactions always initiated by client request to server
  - Server responds with results
- **Stateless**
  - Each request-response cycle independent from other
  - Any state information (login credentials, shopping carts, exam scores, ...) needs to be maintained somehow

# Client-side vs. Server-side



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



## HTTP Request

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



Web server



## HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
...
<html>
<head><title>CS 4640</title></head>
<body>...
```



# 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

# Client-Side Scripting

- Generate HTML on the **client** through scripts
- Example: JavaScript

```
1 <html>
2 <head>
3   <title>Example: JavaScript to create a table of factorials</title>
4 </head>
5 <body>
6   <script type="text/javascript">
7     document.write("<h2>Table of Factorials</h2>");
8     for (i = 1, fact = 1; i < 10; i++, fact *= i) {
9       document.write(i + "! = " + fact);
10      document.write("<br>");
11    }
12  </script>
13 </body>
14 </html>
```

## Table of Factorials

1! = 1  
2! = 2  
3! = 6  
4! = 24  
5! = 120  
6! = 720  
7! = 5040  
8! = 40320  
9! = 362880

## 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

```
1 <!doctype html>
2 <html>
3 <head>
4   <title>Login example</title>
5 </head>
6 <body>
7   You logged in as <font color="green"><b><?php echo $_POST["name"]; ?></b></font><br>
8   with password <font color="green"><b><?php echo $_POST["pwd"]; ?></b></font>
9 </body>
10 </html>
```

PHP

```
1 <html>
2 <head>
3   <title>Counting with a JSP</title>
4 </head>
5 <body>
6   <!-- Set global information for the page -->
7   <%@ page language="java" %>
8
9   <!-- Declare the variable -->
10  <%! int count = 0; %>
11
12  <!-- Scriptlet - Java code -->
13  <%
14    for (int i = 0; i < 10; i++)
15    {
16      count = count+1;
17    }
18  <br />
19    The counter value is: <%= count %>
20  <% } %>
21 </body>
22 </html>
```

JSP

## Advantages

- Server produces HTML, potentially increase security, improve browser compatibility

## Disadvantages

- Less interactivity

# 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 architectural 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.

(but still important!)

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

# Summary

---

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