# What this class is about:

An overview of how the Internet works
What is a website and how to create one from scratch
...with semantic markup and design/layout
...with interactivity through client- and server-side programs
...with up to four different languages
...by searching through online documentation
...and by following detailed specifications
...and overall, building design/development strategies across the "full-stack"

The end result? A better understanding of the web, important technologies, and a portfolio for you to show!

# What this class is *not* about:

While this course is very practical and hands on (you ***will*** write lots of code and be able to show it off), it is a foundational survey course.

This means:

… you will learn the foundations of what powers the Web, but not the latest framework everyone might be talking about.
… you will learn a lot of concepts and a lot of terminology, but there will *always* be more; we don't (and can't) cover it all.
… as such, while people have gotten Web programming/development jobs after this course, our goal is to give you the basis with which to learn more -- it's an intro course, after all.

# Modules

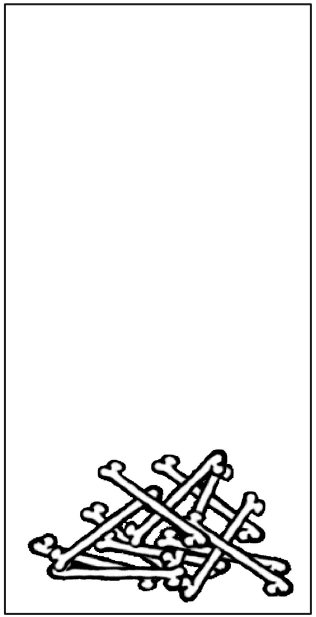
1. Web page structure and appearance with HTML5 and CSS.
2. Client-side interactivity with JavaScript.
3. Using web services (APIs) as a client with JavaScript.
4. Writing JSON-based web services (APIs) with Node.js.
5. Storing and retrieving information in a database with SQLite and server-side programs.

# Web Development Tools

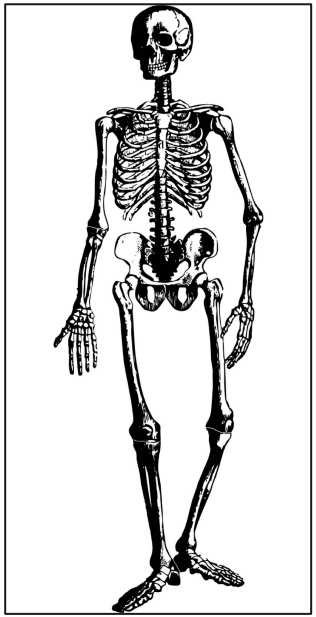
Throughout the quarter, we will be using the following web development tools:

- Chrome: a browser to view and debug web pages
- VSCode: a text editor to write HTML/CSS/JS/SQL (with various helpful packages available)
- Github to clone/push CP/HW repositories (built into VSCode)

# So What is a Web Page Really?



CONTENT



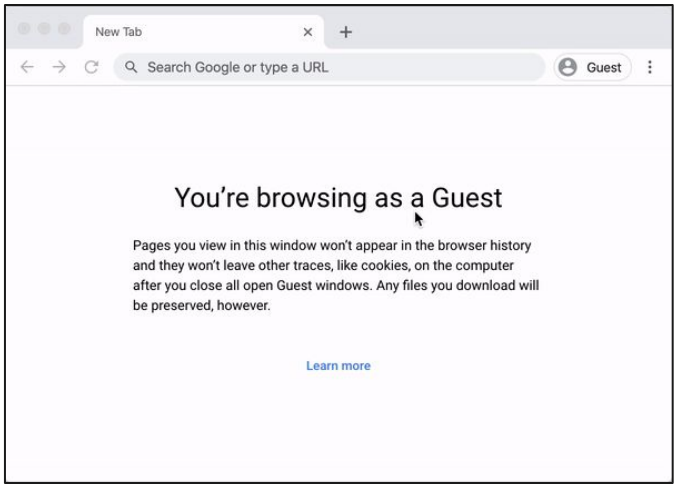
STRUCTURE


STYLE


BEHAVIOR

# Ok, but what is it really?

What's everything involved here?




It's just this, right?

1. Decide on URL...
2. Type it in...
3. Hit enter...
4. Website loads!

But what happens between 3 and 4?

# What happens in the second half?

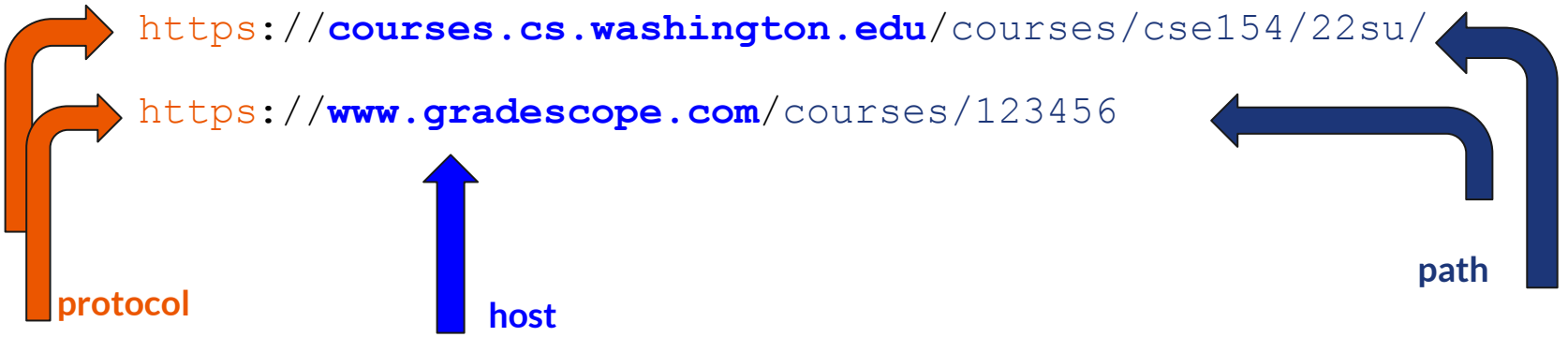
You don't have Google.com on your computer. So, where does it come from?

1. Figure out where it is
2. Ask for it to be sent to us
3. Check and verify what we get
4. Show it

# The thing in the address bar. Where is the website?

**Uniform Resource Locator** (URL): An identifier for the location of a document

A couple of basic URLs:



**Protocol**: is “an established set of rules that determine how data is transmitted between different devices in the same network” (source)

# The Internet

- The internet: the way computers connect to share information with each other
  - Communication through internet protocol suite (TCP - transmission control protocol/IP - internet protocol), set of communication protocols
- Global network
- “Network of networks”
- Enables us to do and be able to access an immense amount of services/information
- Most common use of the internet: accessing the world wide web
  - Other uses: email, file transfer, etc.

# The World Wide Web

- Application that runs on the internet
- Collection of information we have access to via the internet
- We need the internet to access the world wide web

# Internet VS. The Web

## INTERNET

Computers (servers) connected to each other via a series of networks
Powered by layers upon layers:

- Physical: The cables between them
- Data & Network: The [small] packets of information
- Transport (TCP/IP): Providing connections and reliability
- Application: Tying everything together to be useful

## THE WEB

- Collection of pages of information
- Text... but with some "Hyper" around it
- Pages can link to each other
- Pages have style and interactivity

# Remember that URL? (https://google.com/)

Need to go out to the internet to get the webpage.
Internet is low-level: based on numbers (IP addresses), not names.

**Domain Name System (DNS)**
A Domain Name System translates human-readable names to IP addresses

- Example: cs.washington.edu → 34.215.139.216
  - Hostname of cs.washington.edu (which we might put into the browser's address bar)
  - ... has IP address of 34.215.139.216 (which will be used to contact the server via the internet)

# More of the URL than the host

**`https`**`://`**`courses.cs.washington.edu`**`/courses/cse154/22su/`

We've handled the host to IP address (so we know who to ask for the web page)
The "**protocol**" tells us *how*:

- HTTP: HyperText Transfer Protocol
- Gives us the instructions (protocols) for how to share (transfer) web content ("hypertext")

And the rest tells us *what*:

- From the `courses.cs.washington.edu`server (aka **host**)...
- I'd like the thing called `courses/cse154/22su/`... (aka **path** or resource)

# HTTP codes

With all responses that a  web server sends, there is a response code which signifies the status of the response

**Common Codes:**

| Number | Meaning |
| --- | --- |
| 200 | ok |
| 301-303 | Page has moved (permanently or temporarily) |
| 403 | You are forbidden to access this page |
| 404 | Page not found |
| 418 | I'm a teapot (fun fact, example) |
| 500 | Internal server error |

Complete list (as dogs, as cats)

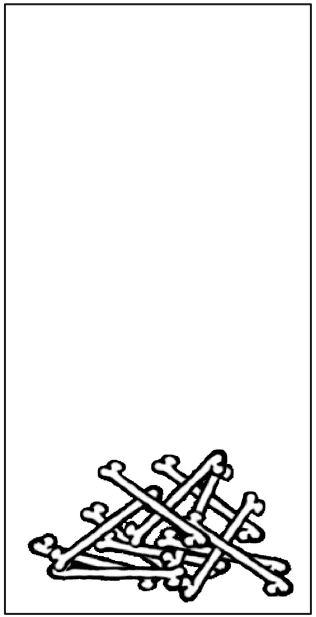
# The real innovation

HTTP built resilience into the internet by creating the 404.
A website will always give a response, even if what a user wants isn't found.

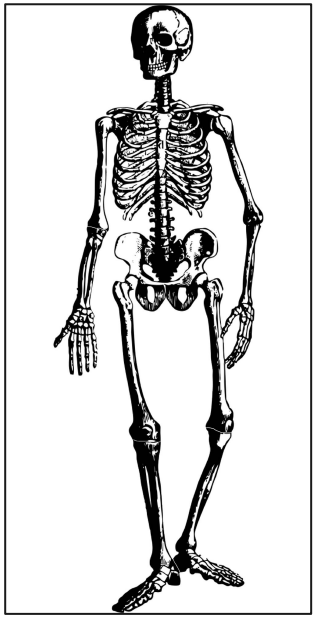
Examples:

- ACM's 404
- WSDOT
- Imgur
- FT Labs
- Discord
- Android
- GitHub

# Then... we have the web page right?



WORDS + IMAGES



HTML


CSS


JAVASCRIPT

# What's in a web page

**Hypertext Markup Language** (HTML): semantic markup for web page content

**Cascading Style Sheets** (CSS): styling web pages

**Client-side Javascript**: adding programmable interactivity to web pages

**Asynchronous Javascript and XML**: fetching data from web services using JavaScript fetch API

**JavaScript Object Notation** (JSON): file format for organizing human readable data

# A brief introduction to HTML