INTRODUCTION TO WEB APPLICATION DEVELOPMENT

KARIM ALIBHAI

TODO:

- Get to know each other
- A welcome to the course
- Your cumulative projects
- **Lesson 1:** The Insides of a Computer
- **Lesson 2:** What on Earth is a web app?

INTRODUCTIONS

- Your name
- Your current/future career path
- Something you are passionate about
- An interesting fact about your passion that the average person wouldn't know

THE CATCH

- Everyone will be assigned a random partner.
- You must introduce your partner, not yourself.

BEGIN:
10:00:00.

TIME'S UP!

TEACHING ASSISTANTS

- Farhan Kanjiyani
- Jeff Peng

FARHAN KANJIYANI (@FARHAN5498)

- Studying Business & Computer Science.
- I <3 Video Games & Technology.

JEFF PENG (@JEFFTHEHAUTBOIS)

- Studying Electrical Engineering.
- I <3 technology, Ultimate Frisbee, & the Oboe.

TEACHER

KARIM ALIBHAI (@KARIMSA)

WHO AM I?

- My name is Karim. (?!)
- I'm a Developer.
- I <3 JavaScript, W3, & teaching.

WELCOME!

PACE

- This is not a class about programming, it's about concepts.
- In-class teachings will keep up with class average.
- TAs are present for the purpose of helping students who would like extra help.
- TAs as well as the teacher will be available via email outside of class time to answer questions.

ASSIGNMENTS

- There will be regular assignments to help you learn to properly apply the lessons.
- There are supplemental exercises for people that would like extra practice.
- There are advanced exercises for people that would like to challenge themselves.

READINGS & QUIZZES

- There will be regular readings **required** to keep up with the course.
- There will be **no quizzes**.
- Whenever possible, we will live code projects together.
  - Every person will be responsible for creating a separate part of the project.
  - Due to this, *if you do not keep up with readings, your **entire** group will suffer.*

CUMULATIVE PROJECT

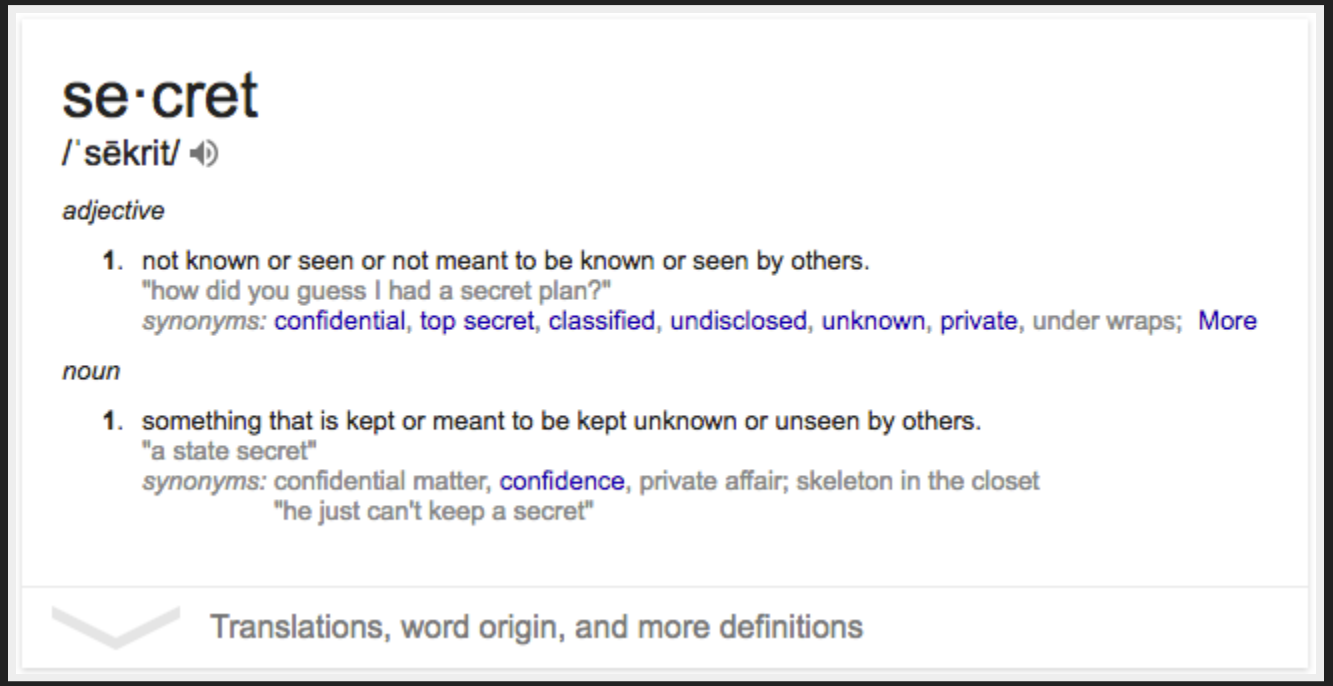
- The best practice for a working environment is a working environment.
- Think of a family member or friend that would benefit from a web application.
- Either contact this person and ask if you can create it for them...
- ... or create it just for fun.

REQUIREMENTS & TIME SPENT

- You may work in pairs of two or alone.
- There will be time after some classes to work on your projects.
- You must also work on your projects at home.
- Try to incorporate as many of the lessons you learn as possible.

THE GRAND PRIZE

- **The top 3 projects will receive a *secret grand prize*.**



se·cret

/ˈsēkrit/

adjective

1. not known or seen or not meant to be known or seen by others.
   "how did you guess I had a secret plan?"
   synonyms: confidential, top secret, classified, undisclosed, unknown, private, under wraps; More

noun

1. something that is kept or meant to be kept unknown or unseen by others.
   "a state secret"
   synonyms: confidential matter, confidence, private affair; skeleton in the closet
   "he just can't keep a secret"

Translations, word origin, and more definitions

LESSON 1: THE INSIDES OF A COMPUTER

THE LAYERS OF A COMPUTER

- Basic Input and Output System (BIOS) is responsible for communication with all hardware.
- Operating System (OS) is the software responsible for acting as a host.
  - Provides software known as `drivers` for application to communicate with hardware.
  - Controls the remaining layers.

THE FILESYSTEM LAYER

- Filesystem is responsible for long-term data storage and retreival.
- The hardware behind the filesystem is the hard drive(s).
- There are two means of data organization:
  - A packet of data is referred to as a `file`.
  - A group of these packets is known as a `directory` (often referred to as a folder).
  - A file *cannot* be within another file, only within a directory.
  - A directory *can* be within another directory.

NOTES ON THE FILESYSTEM

- The location of a file or directory is referred to as the `path`.
- Within a path, names of files and directories are separated using a `path separator`.
  - On Unix-based systems, the path separator is `/`. (i.e. `/path/to/my/file.txt`)
  - On Windows, the path separator is `\`. (i.e. `\path\to\my\file.txt`)

QUESTION TIME: WHICH IS WHICH?

- Given the principles of the filesystem, how would you tell what names belong to files and which ones are directories?
- For instance: `/a/b/c/d` (name the type for each of the letters).

MULTIPLE HARD DRIVES

- Every HDD has its own filesystem.
- On Windows, all drives are assigned a letter which preceeds every path (followed by a colon).
  - The default drive is always the `C` drive.
  - For example: `C:\Users\Administrator`
- On Unix, all drives are `mounted` onto a directory.
  - For example: `/mnt/My-USB` or `/Volumes/My-USB`

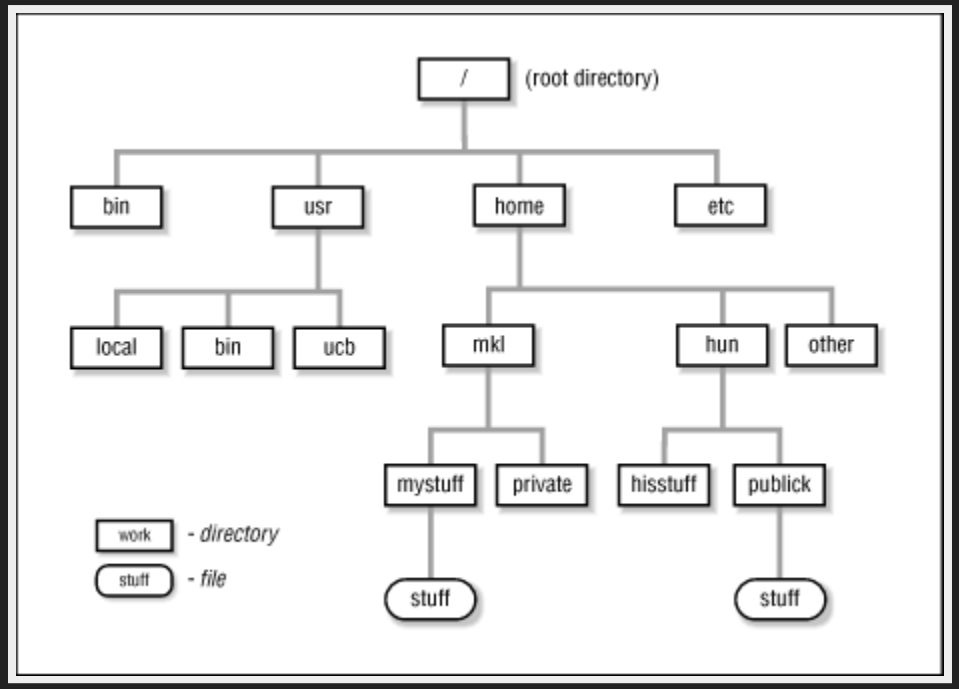
SPECIAL CASES

- Windows filesystems are NOT case sensitive.
  - Therefore, `\users` is the same thing as `\Users`.
- *nix filesystems are case sensitive.
  - Therefore, `/Users` and `/users` are `NOT` the same thing.
- You should try to avoid using uppercase characters or whitespaces in paths
  - Whitespaces in paths must be escaped using a backslash on *nix.
  - For example: `/my directory` is an invalid path; `/my\ directory` is a valid path.
- Valid path characters change based on the filesystem implementation.

THE FILESYSTEM TREE

- The top-most directory in any OS is referred to as the `root` directory.
- The name of the directory is not `root`, that is simply what it is referred to as.
- The name of the root directory is the same as the path separator.
- When a file or directory is nested in another directory, we can say that it is the child of the other directory.
- Therefore, the other directory is the parent of this file or directory.

THE FILESYSTEM TREE



RELATIVE & ABSOLUTE PATHS

- Paths can be either relative or absolute.
- Within any directory, there are two special directories:
  - `.` : represents the current directory.
  - `..` : represents the parent directory.
- Absolute paths define the location of a file/directory from the root of the filesystem.
- Relative paths define the location of a file/directory relative to the current directory or a specific file.

PRACTICE

FILE EXTENSIONS

- Within a filename, the actual name of the file and its extension are separated by a period.
- The file extension tells the OS and the user what type of file it is.
- Popular file extensions:
  - Image files: jpeg, jpg, png, gif.
  - Video files: avi, mp4, mov.
  - Text files: .txt
  - Executables: .exe (windows-only)
- On Linux, file extensions are not necessary.

THE USER & GROUPS LAYER

- The OS mananges a list of users & groups.
- All users and groups have specific permissions within the OS.
- All files & directories have owners that set read, write, and execute permissions.
- The users with the highest privileges are root users (on Unix systems) and administrators (on Windows).
- Some admins must provide their password before executing any admin-level functions (i.e. deleting protected files).

THE NETWORKING LAYER

- Responsible for inter-device communication.
- Underlying *internal* hardware is known as the Network Interface Card (NIC).
- Networking involves much *external* hardware as well.

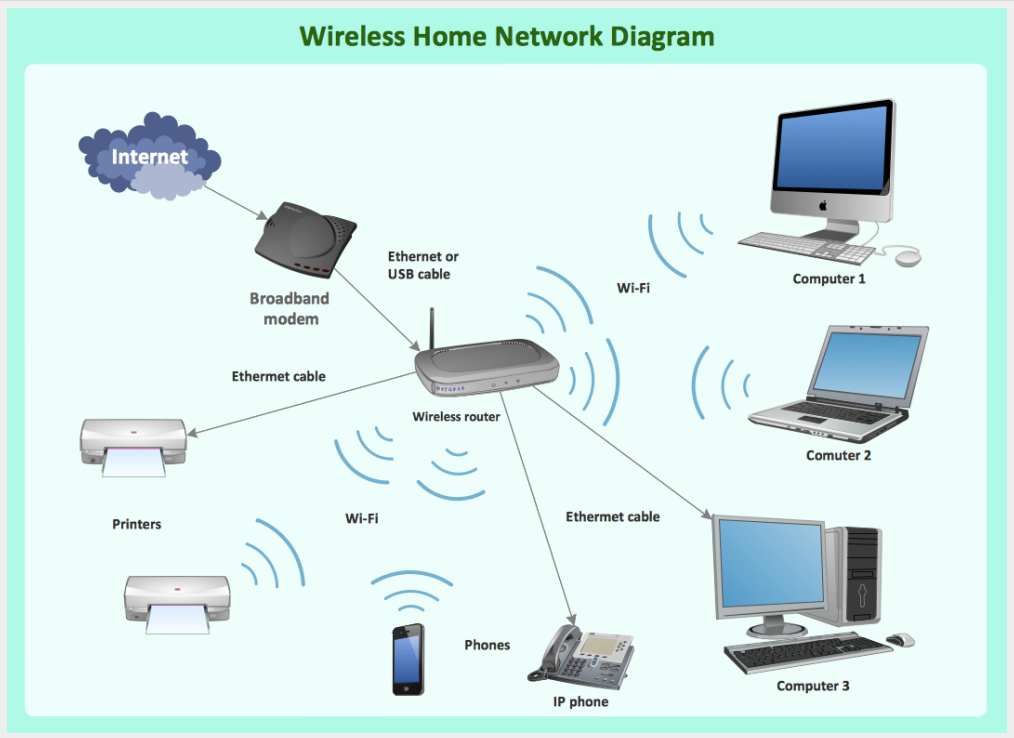
INTERNET PROTOCOL ADDRESS

- An IP address is an address of a device on a network.
- The format of an IP address is predefined.
- IPv4 addresses (most common) format: four groups of decimal digits (8 bit groups).
  - For example: 192.168.1.101
- IPv6 address format: eight groups of four hexadecimal digits (16 bit groups).
  - For example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

THE ROUTER

- Creates a local network for inter-device communication for all connected devices.
- Manages communication between local network and the internet.
- For devices, communication on the local network and devices on the internet is almost identical.
- Assigns a local IP address to all devices, including itself.
- Routers have a local IP **and** a public IP.

THE LOCAL AREA NETWORK (LAN + WLAN)

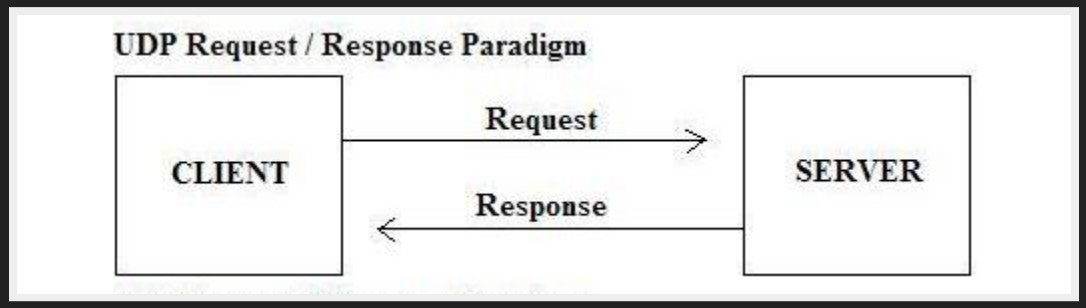


PORTS

- There are 65536 (or 2^16) `ports` on any given device.
- Ports are where all data between devices are sent from and to.
- Ports are specified by integers. (i.e. the first port is 1, the last is 65536)
- Ports < 1024 can only be opened by applications with administrative access.
- There are two main protocols which can be used to network: TCP & UDP.

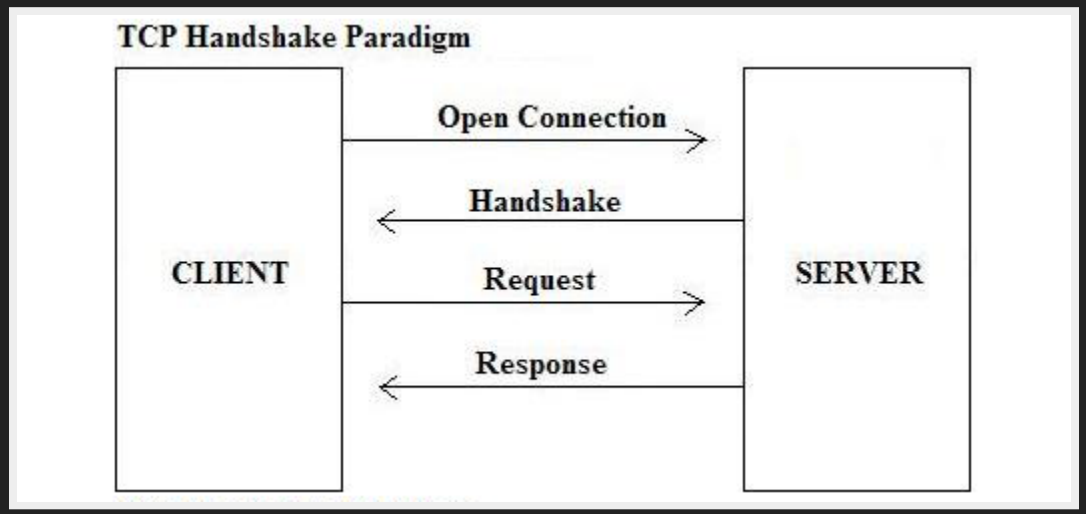
USER DATAGRAM PROTOCOL (UDP)

- Known as a `best-effort` protocol.



TRANSMISSION CONTROL PROTOCOL (TCP)

- Requires an on-going connection between hosts.
- Each host must continuously acknowledge received and sent data.
- This keeps both hosts up-to-date on each other's presence.



QUESTION TIME: WHAT PROTOCOL SHOULD BE USED FOR EACH?

- A chatting application.
- Device discovery.
- Video streaming.
- Internet audio/video calling.

NOTES ON NETWORKING

- For a device to receive data over a network, it must have an application that binds onto a port.
- For a device to send data over a network, it needs the IP address and port of the receiver.

PUBLIC ROUTING

- When communicating with devices over the internet, you specify the external device's router's public IP.
- The router then forwards (or *routes*) the connection through to the device on the LAN.
- But what about when IP addresses change? And aren't IPs too complicated for users to specify?

DOMAIN NAME SYSTEM (DNS)

- DNS is responsible for resolving domain names to IP addresses.
- Domain names are alpha numeric names that can be bought from Domain Registrars.
- There are only a set number of top-level domains. (i.e. `com`, `ca`, `org`, `co`)
- Your domain name must have a top-level domain attached to it.
- Your domain may have as many sub-domains as you would like (i.e. `images`.google.com, `docs`.google.com)
- Your sub-domains can have sub-domains. (i.e. `scontent`.xx.fbcdn.com)

DESIGNS OF A NETWORKING APP

- Server-Client: a server awaits connections, a client connects to the server.
- Peer-to-Peer (P2P): peer discovers other peers. Everyone is a client and a server.
- Both are protocol-independent.
- The trade-off:
  - P2P offers a better connection as the number of clients increases.
  - Server-client offers the security of a central authority.

THE ENDS OF AN APP

- `Front-end` is the part of the application that the users sees and interacts with.
- `Back-end` is the part of the application that runs on the server.

THE APPLICATION LAYER

- Recall: the OS serves as a host.
- The OS allows code to run on the machine at a higher level than the OS.
  - Heirgo, the OS *hosts* applications.
  - The OS provides programmable interfaces through which applications can access low-level functions.
- Applications that run directly on top of the OS are known as `native` applications.

PROGRAMMING

- In reality, computers can only execute machine code.
- The process of compiling involves a program known as a compiler.
  - A compiler reads some code and writes the corresponding machine code.
- All programming languages compile down to machine code.
- During compilation, an appropriate executable is generated.
- These languages are also known as `compile-time languages`.

NATIVE APPS

- Have direct access to all interfaces that the OS allows.
- Are written in a *programming* language that `compiles` into machine code.
  - Therefore, the code runs **directly** on the machine.
- Must be compiled for every architecture (i.e. windows, linux, and darwin - possibly with 32bit and 64bit versions)
- Require the user to provide administrative privileges for particular functions to be performed.

LANGUAGES ON LANGUAGES

- Some programming languages are built to compile to other languages.
- These languages are rewritten into their lower-level cousins during compilation.
- The outputted code is then compiled into machine code.
- For example:
  - C and C++ are translated to Assembly on compilation.
  - CoffeeScript and TypeScript are translated to JavaScript on compilation.

LANGUAGES THAT DON'T COMPILE

- Some languages are not compiled, they are interpreted.
- There is a `run-time` responsible for reading the language and dispatching the appropriate actions.
- These languages are known as:
  - interpreted
  - scripting languages
  - run-time languages
- JavaScript is interpreted top to bottom.
- "*Java* is to *Java*Script as *Car* is to *Car*pet."

LANGUAGES THAT AREN'T LANGUAGES

- Markup languages are not programming languages.
- Markup does not dispatch any actions, it defines things.
- For example:
  - Markdown: defines the style of sections of text.
  - YAML: like a big tree of data.
  - HTML: defines the elements of a web page.

LESSON 2: WHAT ON EARTH IS A WEB APP?

THE WEB BROWSER

- Always a native application
- Flow of displaying a web app:
  - Render HTML top to bottom.
  - If there are any asset imports, fetch assets and process them.
  - If CSS is loaded, render using CSS engine and re-render whenever the HTML updates.
  - If JS is loaded, execute the code using the JS engine.

THE WEB SERVER

- Websites are stored as a group of resources on a web server.
- The server is essentially responsible for content delivery.
- Web browsers request a web page (written in HTML) from the server.
- During loading of the page, the browser will request all needed assets from the server.

SANDBOXING

- Websites live only within the confines of the web browser.
- The web browser restricts websites into a sandbox with no OS access.
- Websites can only access limited resources as allowed by the browser.
- For this reason, web apps are often thought to be limited and useless.

THE STRENGTHS OF A WEB APP

- All OS-related access (networking, filesystems, etc.) can be handled on the backend.
- The front-end is very capable of adjusting to every platform that supports a web browser.
- Nothing is ever compiled and so you can get rid of the concepts of:
  - Software updates
  - Prerequisites
  - System requirements
  - Administrator access

THE NETWORKING SPECIFICS

- All web traffic happens over TCP.
- Port 80 is for regular web traffic.
- Port 443 is for secure web traffic.
- These are defaults but other ports are allowed.
- For example, entering `google.com:8080` in your browser will use port 8080.