GENERAL ASSEMBLY

Hello World!

Programming for Non-Programmers

GENERAL ASSEMBLY IS A GLOBAL COMMUNITY OF INDIVIDUALS EMPOWERED TO PURSUE THE WORK WE LOVE.

Our 4-hour sprint

You will get a high level understanding of:

- How the Web works

- Get a personal profile website on the webCreate a working web application

Goals

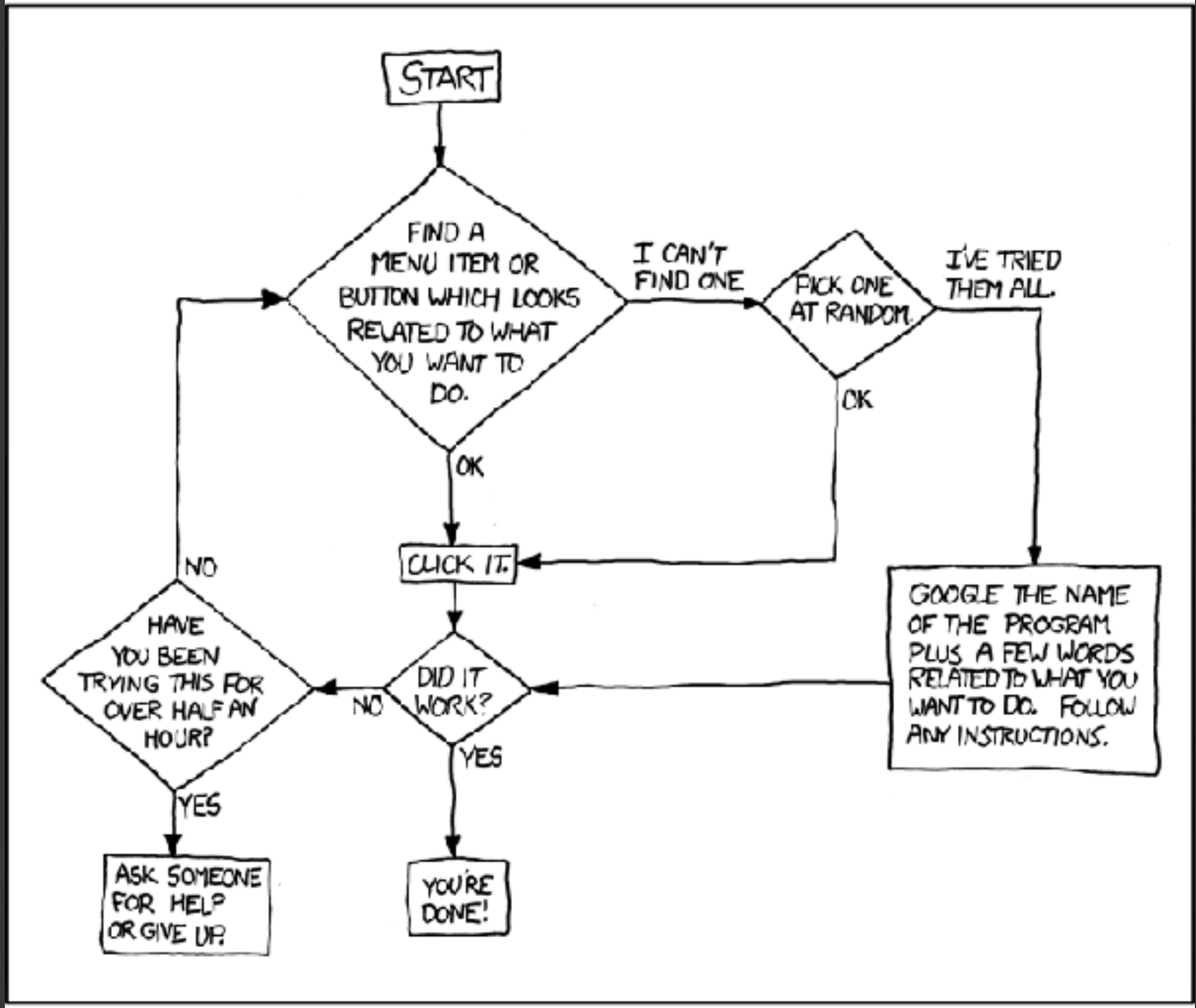
My biggest hope for this class

- You understand the web on a non-muggle level
- You understand what developers are up to
- Learn the essential words and concepts that are used on a daily basis by engineers and project/product managers on the job.
- Demystify programming
- Know how to keep learning if you are into it.

Empower yourself to figure things out for yourself

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Hi, I'm Vincent

- Made a Geocities website at the library to host bad jokes (the header tag was "GET OUT OF MY SITE!") and ska fandom.
- Former English teacher and Journalist
- Started learning Ruby in like 2012, Took a night class at GA
- In early 2014 I decided to take WDI in this very building
- 3 weeks later, started working at 2U, an education technology company, then moved on to be Lead (read: only) developer at a startup.
- Currently a contractor doing full stack web development at Google, on the Local Guides product
- I really really like teaching people about technology and minting future coders. This is my favorite class to teach!

Your turn

- What's your story? What are you doing now?
- What's your programming background if any?
- What first got you interested in web development?
- What you hope to achieve in this class?
- What are your goals beyond this class?

First, let's just install two things

Text Editor & Browser

- ☐ If you haven't already, please download Google's Chrome browser.

- ☐ Also, you will need a "text editor," which lets you edit plain text and highlights syntax to help you. MS Word doesn't cut it for coding. For this class, I recommend Visual Studio Code from Microsoft.

Some essential terminology

What is Programming?
(...and why do I care?)

PROGRAMMING

- a set of instructions
- used to solve a problem

Terms that apply to any programming language

Declarations

- We can store and retrieve data in our programs by associating them with intermediary values that we call `variables`
- **For example,** `x = 5`
- We declare that x is just a placeholder for 5 this way. The computer's *memory* holds onto this variable until it is no longer relevant.
- Think of variables like pronouns, or the x in algebra

Expressions and evaluation

- We use expressions to evaluate stuff.
- For example, `2 + 2` is an example of an expression
- That evaluates to 4
- The computer reads an expression and returns what it evaluates to
- We can use expressions and declarations in tandem to perform complex tasks. For instance, we can reference a variable we declared in an expression to help us evaluate new values which can then be stored.

Statements

- [ ] **Statements are often used to determine control flow.**
- [ ] **Control flow is essentially the order in which declarations, expressions, and other statements are executed.**
- [ ] **Here is a statement:** `The temperature is lower than 80 degrees.`
- [ ] Perhaps your program will tell you to turn the AC off *if* the temperature is lower than 80.

Algorithms

Algorithms

- [ ] People tend to talk about algorithms like they are mysterious forces of nature, some kind of post-modern secular religion or something with rarefied priests in the Bay Area. But I guarantee that you know a few algorithms yourself.

- [ ] All an algorithm is is a series of declarations, expressions, and statements that can be used over and over again to solve well defined problems of a certain type.

Let's pseudo-code an algorithm!

Tell a robot to convert temperatures

- [ ] In plain English (pseudo code), work with a neighbor to give precise, exact instructions to a dumb robot to spit out celsius, given a temperature in Fahrenheit.

- [ ] Write it down on your desk. Use regular English, but use declarations, statements, and expressions. When you finish, try to identify those in your algorithm.

- [ ] If you finish, write out an algorithm for finding out someone's age given their birth year and whether they are a dog or not.

Like the five positions in ballet, these universal concepts make up ALL programming in ALL languages.

After learning one language, it's pretty easy to pick up another, unlike human language.

A word on computer science.

“Computer Science is no more about computers than astronomy is about telescopes”

—Edsger Dijkstra

Abstraction

Some languages are ‘closer to the metal’

- ☐ All code eventually translates into electrons popping in and out of a processor’s logic gates.
- ☐ Processors get binary. Ones and Zeros, on/off.
- ☐ Nobody understands 0101010101 etc. No normal humans at least. Code is always an abstraction of that or something else which is itself abstracted.

Abstraction

- Most computers run some version of Assembly, which is an analogue to the actual machine code that a processor architecture like Intel x86, ARM, or PowerPC understands
- If you need a program to be as fast as possible, this is what you would write. Example, a friend of a friend of mine writes software for the Navy which guides air to air missiles mid flight.
- I have never written a line of any assembly in my life. It's crazy hard, so hard that a bug that nobody caught once killed people.
- Therac-25 — the ultimate bad software. Worth reading about.

HELLO WORLD IN C, ASSEMBLY, AND RUBY

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                     ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel

    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel

section     .data

msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

Assembly

```
#include<stdio.h>

main()
{
     printf("Hello World");


}
```

C

```
puts “Hello world!”
```

Ruby

What is open source?

WEB DEVELOPMENT BASICS

VCS: Version Control Systems

- allow you keep track of the version of your files
- view, retrieve, and see different between previous versions of files
- work across environments and machines
- collaborate with others and keep track of contributions
- different software: **git**, subversion, mercurial

Popular Services:

- GitHub: https://github.com/
- Bitbucket: https://bitbucket.org/

Programming languages

Programming languages

- [ ] A programming language is a series of grammar and rules that we can define towards writing source code.

- [ ] Languages are effectively different approaches towards communicating the same ideas in programming. Essentially, we can communicate in say both French and English, what mainly differs is the structure of our sentences and the actual words and sounds themselves.

- [ ] The same analogy can be made with programming languages.

Interpreted and compiled languages

- [ ] Luckily, we don't have to write arcane procedural programs anymore.

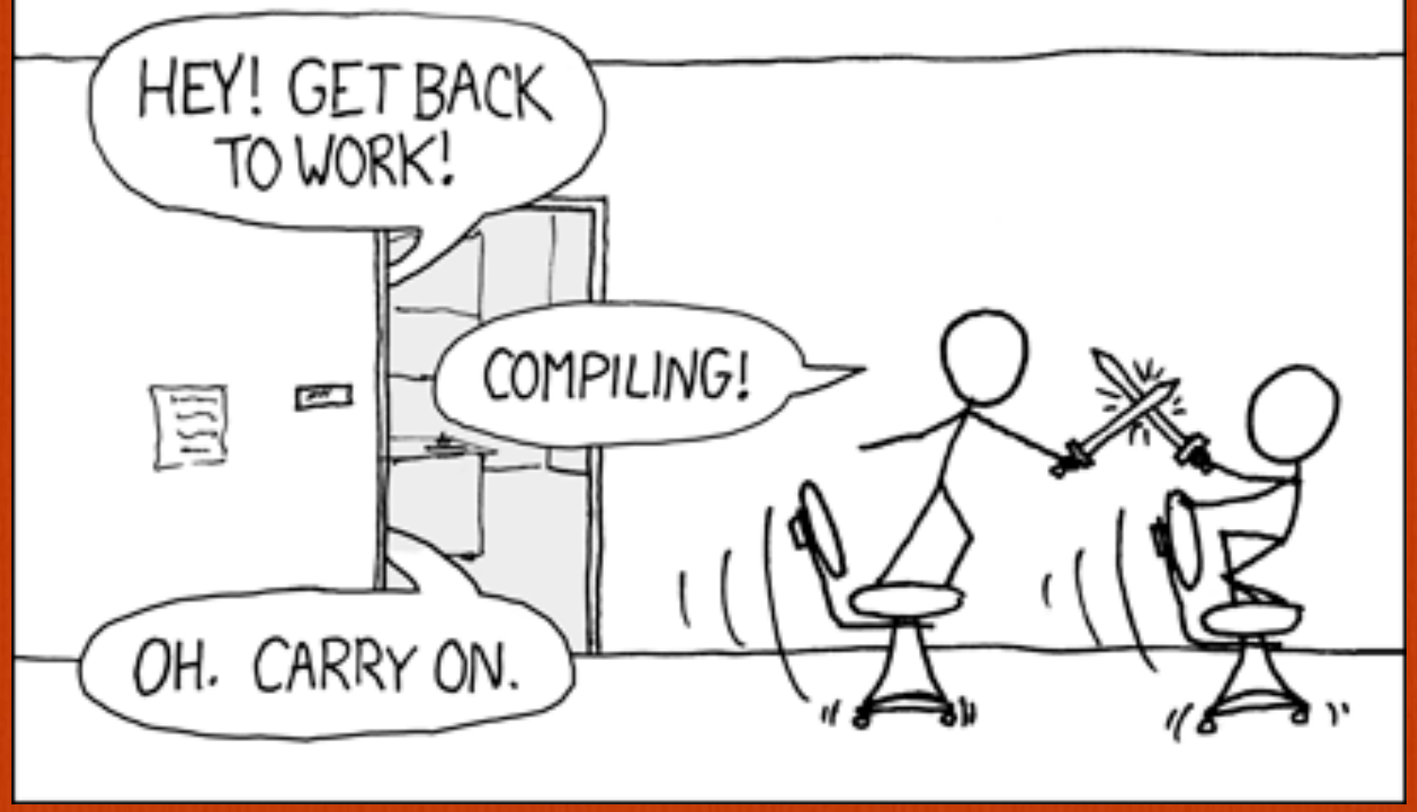
- [ ] For the most part, programming languages are either compiled or interpreted.

- [ ] C is an example of a compiled language. You write your code, something compiles it, and it either works or it doesn't.

- [ ] Languages like Ruby and Python are basically cheat codes for C. They are interpreted, meaning another program just reads your code and figures out what C code to turn it into.

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."



Interpreted languages

- ☐ Compiled languages like C are wicked fast but a pain to get your head around for most programmers.

- ☐ Ruby, Python and many others do the boring work of selecting spots in memory, clearing them out, and having deep conversations with the processor so you don't have to.

- ☐ Interpreted languages also tell you precisely where your bug is, and therefore it's so much easier to work with.

Which programming languages do what

VOCABULARY

THE WEB

WEB SITES
WEB APPS
MOBILE WEB

In a browser

- CHROME
- FIREFOX
- INTERNET EXPLORER
- SAFARI
- OPERA

NATIVE APPS

Directly on your Operating System

- IPHONE
- OSX
- ANDROID
- WINDOWS

OTHER

- DECODE THE GENOME
- OTHER HARDWARE
- XBOX
- ANYTHING ELSE I MAY HAVE MISSED
- VIDEO GAMES

Some programming languages and what they generally do

C

- ☐ Started in the 60s, still going strong.
- ☐ Used to create Unix, which lives on and forms the backbone of Linux and Mac OS X. When you count servers and phones, thats the vast majority of computers. Even Windows 10 can run a subsystem of Linux.
- ☐ Procedural — one line executes after another.
- ☐ Most people find it hard, present company included.
- ☐ Mainly used for operating systems, databases, creating other language interpreters. Great for learning how computers work, but fortunately, you don’t necessarily have to if you want to just get some stuff done.

C++

- ☐ ++ is a common idiom for iteration (this plus one)
- ☐ It's a huge improvement on C.
- ☐ Mainly used for desktop applications. Very big deal in the games industry.

Java

- ☐ Named for the amount of coffee consumed during its creation.
- ☐ Object oriented, fast, high-level, and easy to understand (compared to other languages)
- ☐ Used to be SO COOL for making interactive webpages with “applets”
- ☐ Runs in a Virtual Machine, so a Java program can theoretically run on any platform.
- ☐ Still enormous, partly because of Android, the most successful OS ever.
- ☐ Owned by a patent control freak (Oracle) unfortunately.

Python

- [ ] Made in 1995 by the gorgeously named Guido Van Rossum, a Dutch mathematician. Referred to as “benevolent dictator for life” by Pythonistas.

- [ ] All purpose, great for making web apps, desktop apps, you name it. It’s best feature is that it is very easy to learn and read.

- [ ] Truly shines in science and data analysis. Python’s library Pandas just owns other data science tools.

Ruby!

- ☐ My personal favorite.
- ☐ Made in 1995 by Matsumoto Yukihiro (Matz). He wanted to make a pleasurable language that emphasizes programmer happiness above all else.
- ☐ “Matz is nice and so we are nice”
- ☐ Amazing for making full-fledged web apps with Ruby on Rails. Also used extensively in automating tasks for devops.
- ☐ Very similar to Python but more expressive. More fun, but harder to master.

C#

- [ ] Basically Microsoft's thing. Used for Windows apps and Microsoft's .NET web framework.

Objective-C and Swift

- [ ] Apple stuff. If Mac OS X or iOS apps are your jam, you should learn one of these.

- [ ] Swift came out just over 2 years ago, and it's much easier to understand than the lower level Obj C. Swift syntax borrows heavily from JavaScript.

Which brings me to the 500 lb gorilla that is JavaScript

JavaScript

- [ ] Has the most oddball history of any programming language

JavaScript

- ☐ Created over the course of TEN FREAKING DAYS in 1995 for Netscape
- ☐ Most widely used of all, by a huge margin.
- ☐ Named JavaScript to capitalize on the contemporary popularity of Java. A crappy marketing decision that we still live with.

Make the monkey dance

- ☐ JavaScript, for years, was solely used to manipulate and animate web pages.
- ☐ It's the only real programming language that any browser you can find understands. That's just how it is.
- ☐ This makes it the sole language of front end development, but that isn't all it can do.

Why new programmers should learn JavaScript first

- ☐ Most popular, despite its massive flaws
- ☐ You can make something that does something on day one since it runs in the browser. With Python or Ruby, you have no choice but to start out with command line programs.
- ☐ The browser is the most successful runtime environment ever conceived.
- ☐ Don’t have to install a delicate development environment

Why new programmers should learn JavaScript first

- ☐ Enormous job market
- ☐ Easy to get started, no need to use the command line.
- ☐ The only language that can be your only language.
- ☐ Massive community — any problem that you come across has been made by others, and solved.
- ☐ Currently undergoing a Cambrian explosion.
- ☐ JS deserves credit for dramatically democratizing programming.

“Anything that can be written in JavaScript, will be written in JavaScript.”

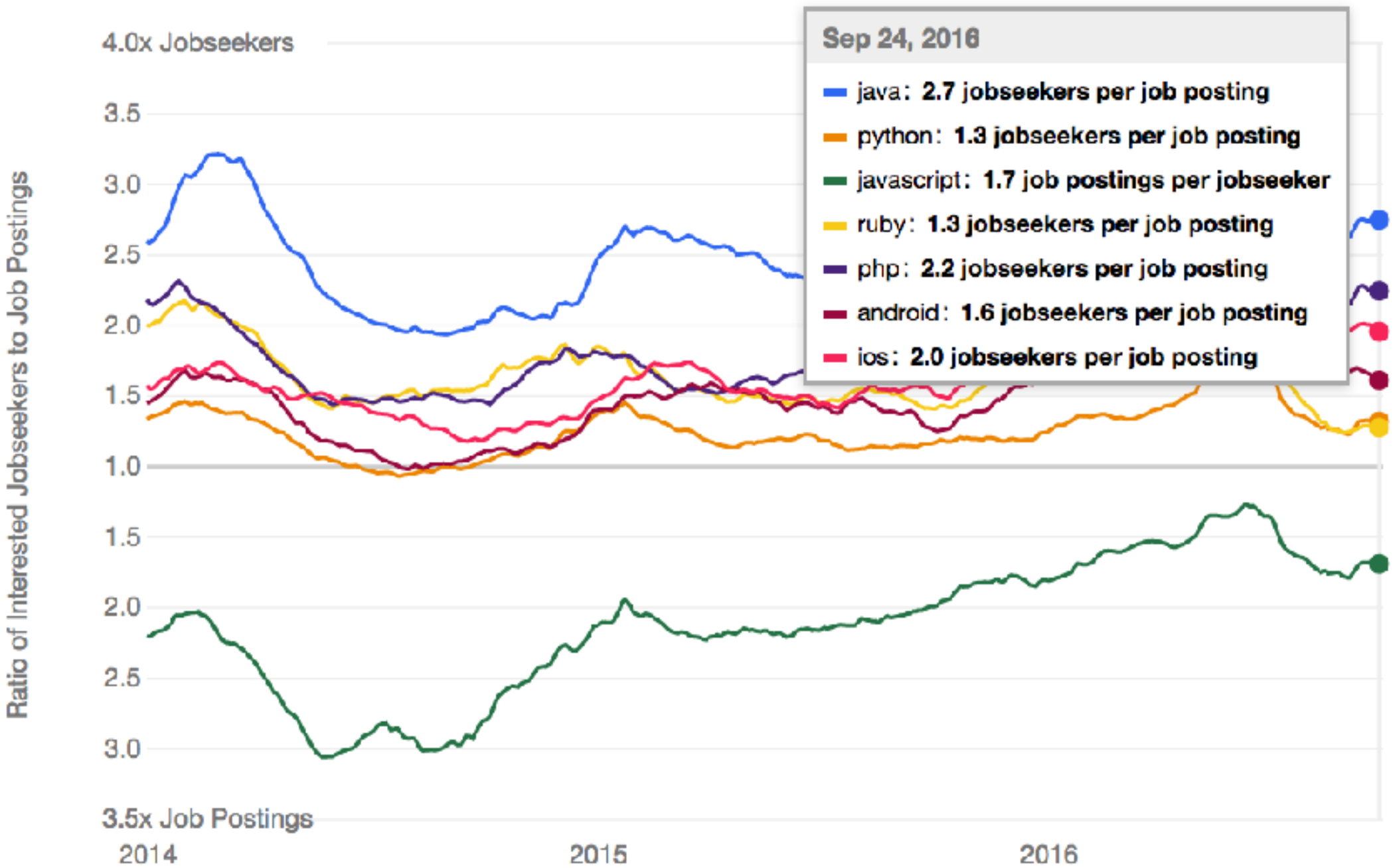
–Jeff Atwood

Node and universal JavaScript

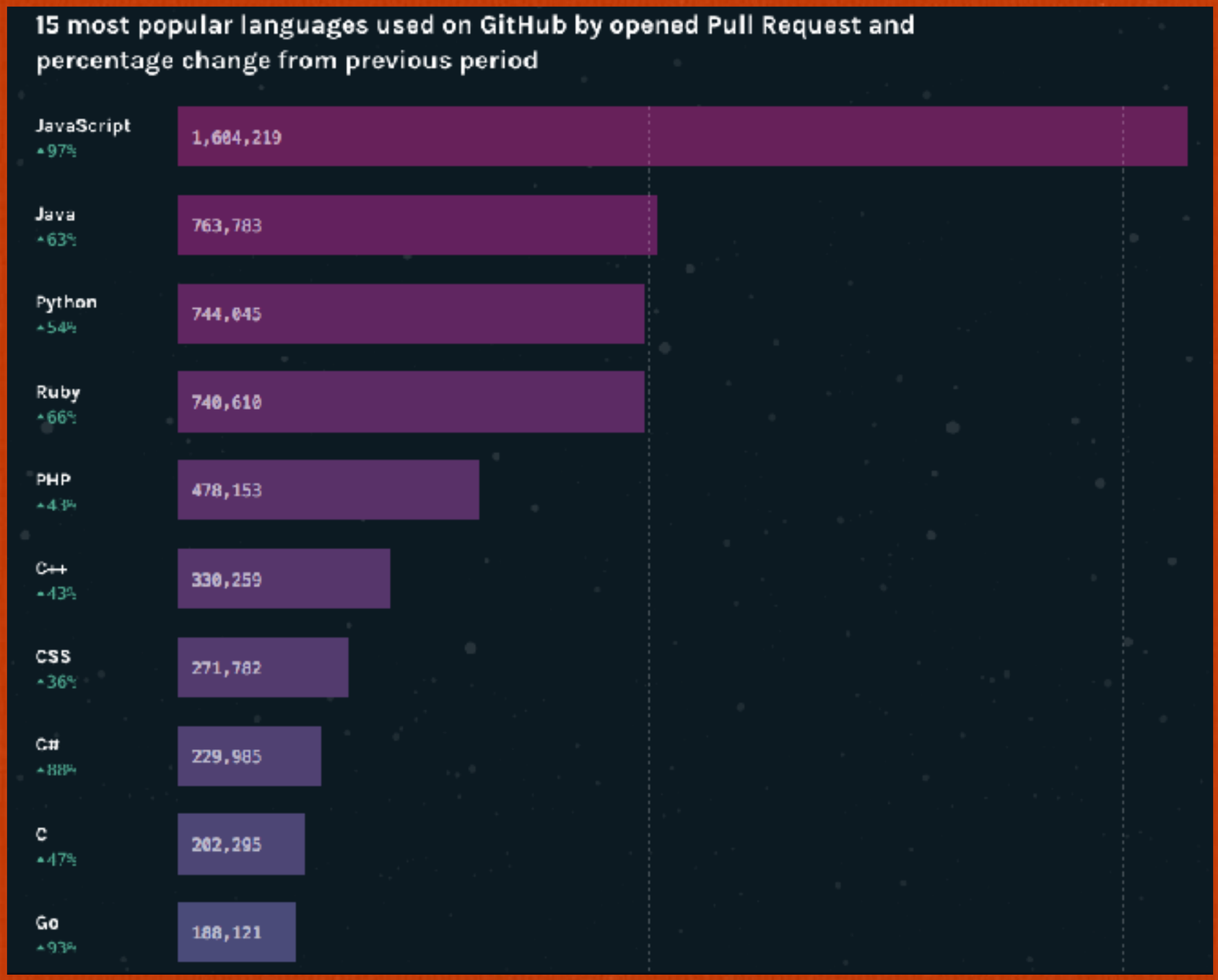
- [ ] Thanks to Node.js, you can now use JavaScript practically anywhere.

- [ ] The front end of a website is the start, but from there you can do server side code, home automation, robotics, and even fine art.

Job market


4.0x Jobseekers
3.5
3.0
2.5
2.0
1.5
1.0
1.5
2.0
2.5
3.0
3.5x Job Postings
Ratio of Interested Jobseekers to Job Postings
2014
2015
2016
Sep 24, 2016
java: 2.7 jobseekers per job posting
python: 1.3 jobseekers per job posting
javascript: 1.7 job postings per jobseeker
ruby: 1.3 jobseekers per job posting
php: 2.2 jobseekers per job posting
android: 1.6 jobseekers per job posting
ios: 2.0 jobseekers per job posting

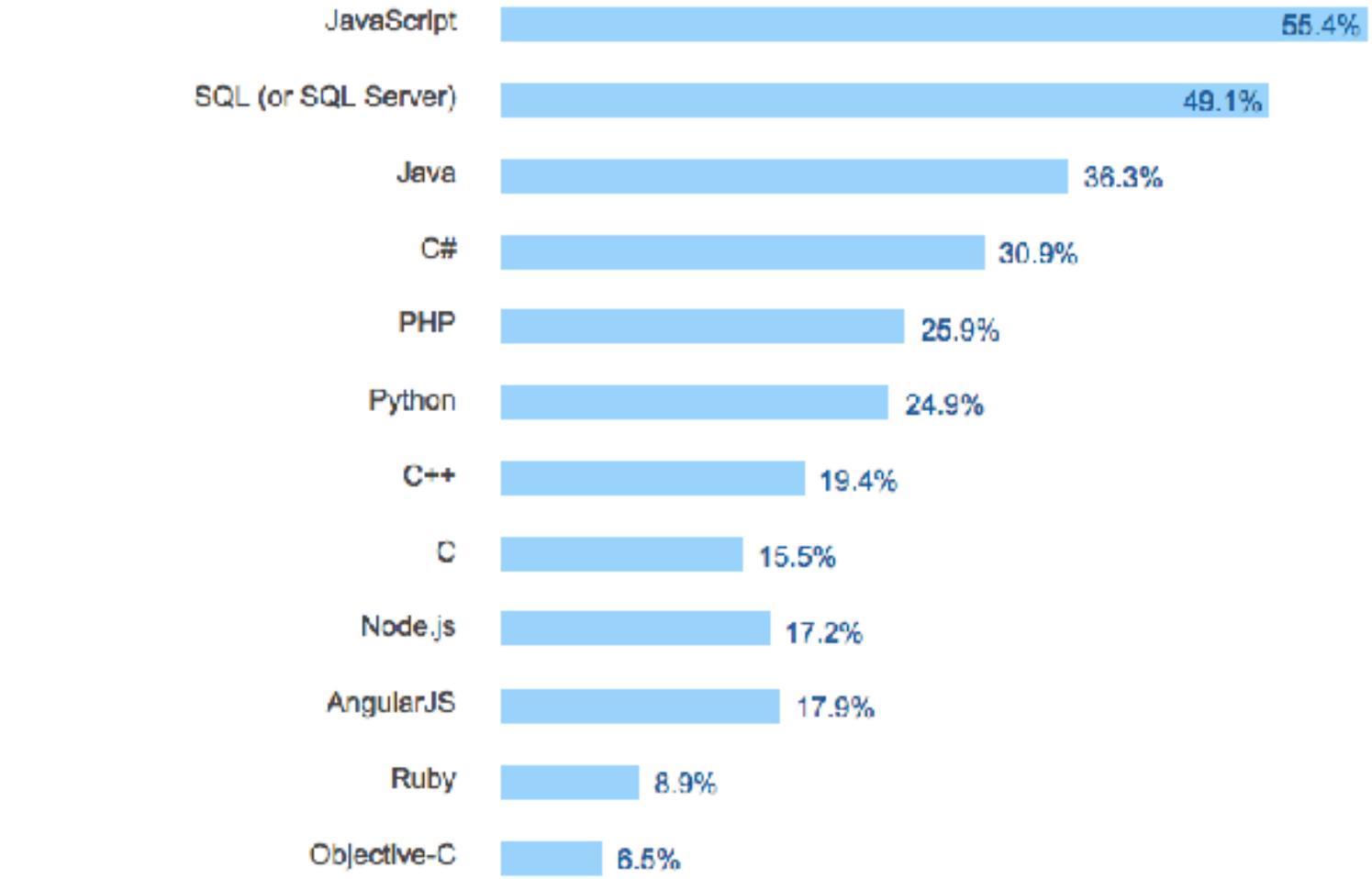
Open source action



Community

I. Most Popular Technologies

2016 | 2015 | 2014 | 2013



49,397 responses

“Never memorize something that you can look up.”

Documentation and help

- ☐ MDN
- ☐ Stack Overflow
- ☐ Google

What are some other programming languages that you would like to hear about?

Anything else that you would like to learn about that I *might* be able to teach you?

Questions?

WEB DEVELOPMENT

Programming for Non-Programmers

WHAT IS WEB DEVELOPMENT?

- The use of programming languages and concepts towards producing a system that listens to user requests over the internet and serves back HTML, CSS, and interaction code to the client.

WEB APPS VS WEB SITES

- Web sites are more static. Think of them as sort of like interactive brochures. They typically don't hold any state and are usually just information. They are amnesiac little things that could be gorgeous and interactive, but that’s it.

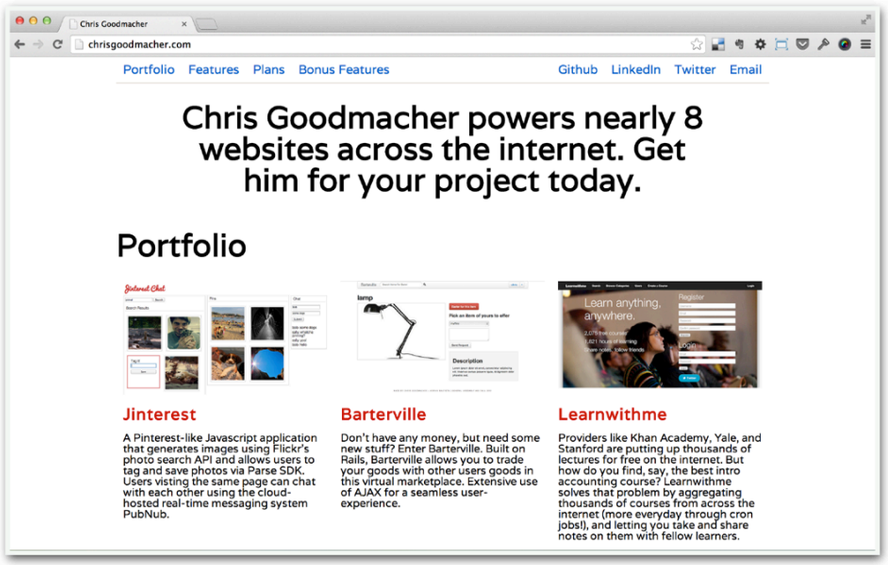
- A web app is more like an iOS or Android app. It holds state and in general does something vs showing something.

- Almost invariably, a web app is connected to a persistent data storage, a database.

- Generally, if you can log into it, its a web app.

WEB DEVELOPMENT BASICS

1. HARDCODED



ex. portfolio page

index.html

Server

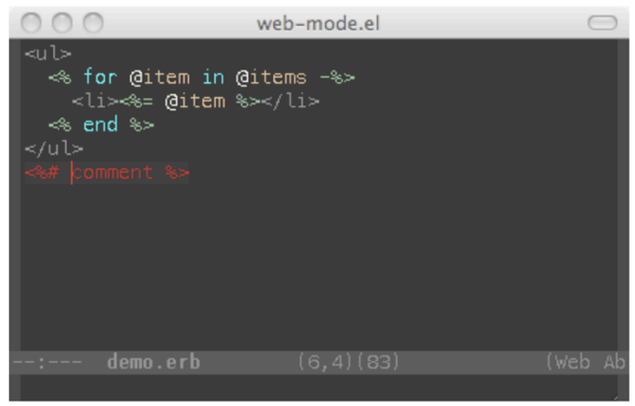
(170.149.168.130)

2. DYNAMIC

ex. Facebook

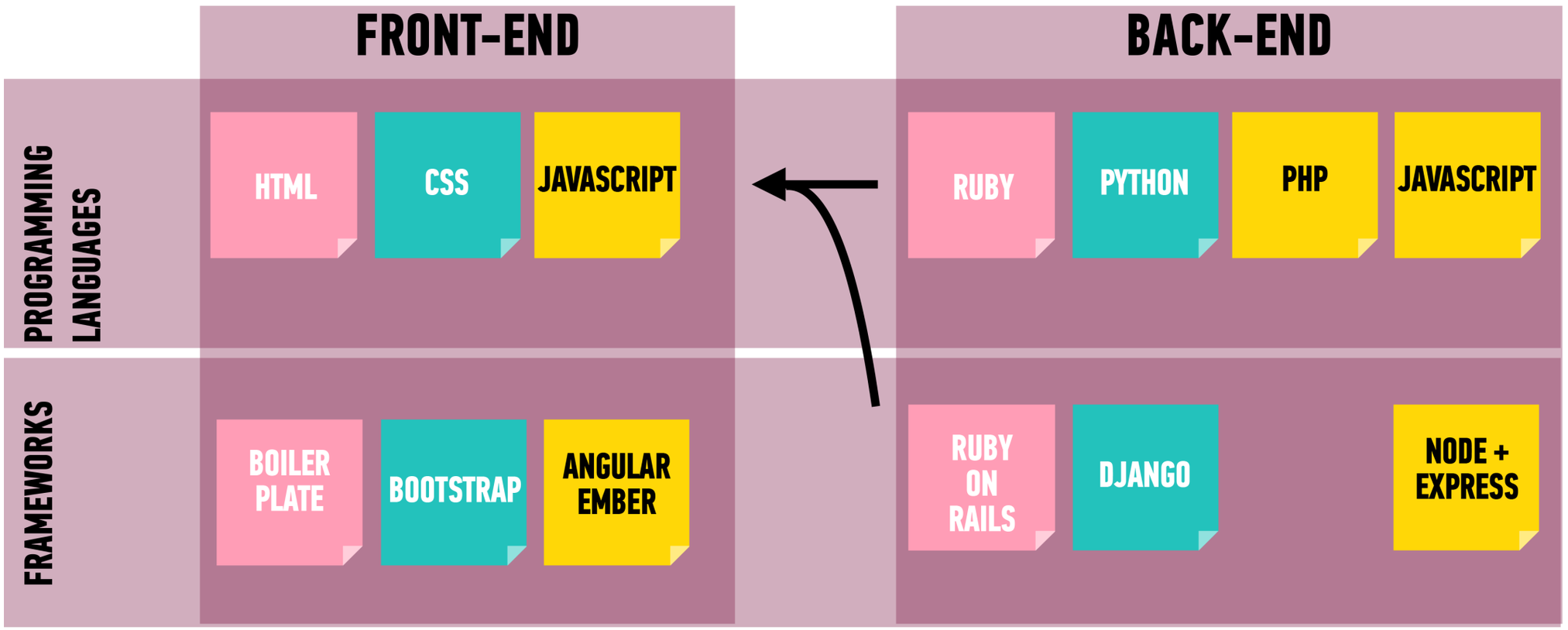
Server

(170.149.168.130)



Program that generates HTML on the server

FRONT-END VS. BACK-END



WEB DEVELOPMENT BASICS

API: Application Programming Interface

- allow you to retrieve and/or submit information from a source
- some are private, i.e. for internal use within the company
- some are public, i.e. for use by outside individuals
- public APIs may require registration, payment, security measures, and rate limits

Examples:

- NYT: http://developer.nytimes.com/docs
- Instagram: https://www.instagram.com/developer/
- Twitter: https://dev.twitter.com/overview/documentation

UX

- Big companies typically, have a UX phase, a Design phase, a Development phase bundled with extensive QA. This is how it is at Google and other large companies I have worked for.
- When, I was in a startup, design and development kind of happened simultaneously. There is a lot more room for iteration and testing out ideas by putting something up for users to use and deliver feedback.
- UX design incorporates psychology, statistics, business chops, graphic design, and an endless back and forth with developers, business stakeholders, and users themselves. A very fascinating and fast-growing discipline.

A TYPICAL FLOW

- Management has an idea, they allocate people and time to it
- Designers chuck something together that they think might work.
- Developers or technical managers weigh in to determine whether it is feasible and sometimes push back on design.
- They write some code, compare notes with design, and ship it.
- (we're not done yet!)

A TYPICAL FLOW

- Usually, the new feature would be A/B tested, meaning that only some users see the new feature, or some see one version of it, others see a different one.
- This helps inform the team on how the feature is faring with users and iterate. UX might realize a problem in analytics, talk to users directly, and perhaps find out that people hate it or just don’t care.
- Then they redesign, redevelop, ship, analyze, repeat.
- Forever.

GET READY FOR THE MOST COMMON INTERVIEW QUESTION EVER

WHAT DO YOU RECKON HAPPENS WHEN YOU VISIT A WEBSITE IN YOUR BROWSER?

TALK TO YOUR CLOSEST NEIGHBOR(S)

- Draw out what you think transpires across the internet when you do that? Go into as much detail as you can.
- Feel free to use your markers on your desk.
- Then we will talk over a few key terms and revisit this.
- Just so you know, this is the takeaway that I find most important from this class.

FIRST OF ALL

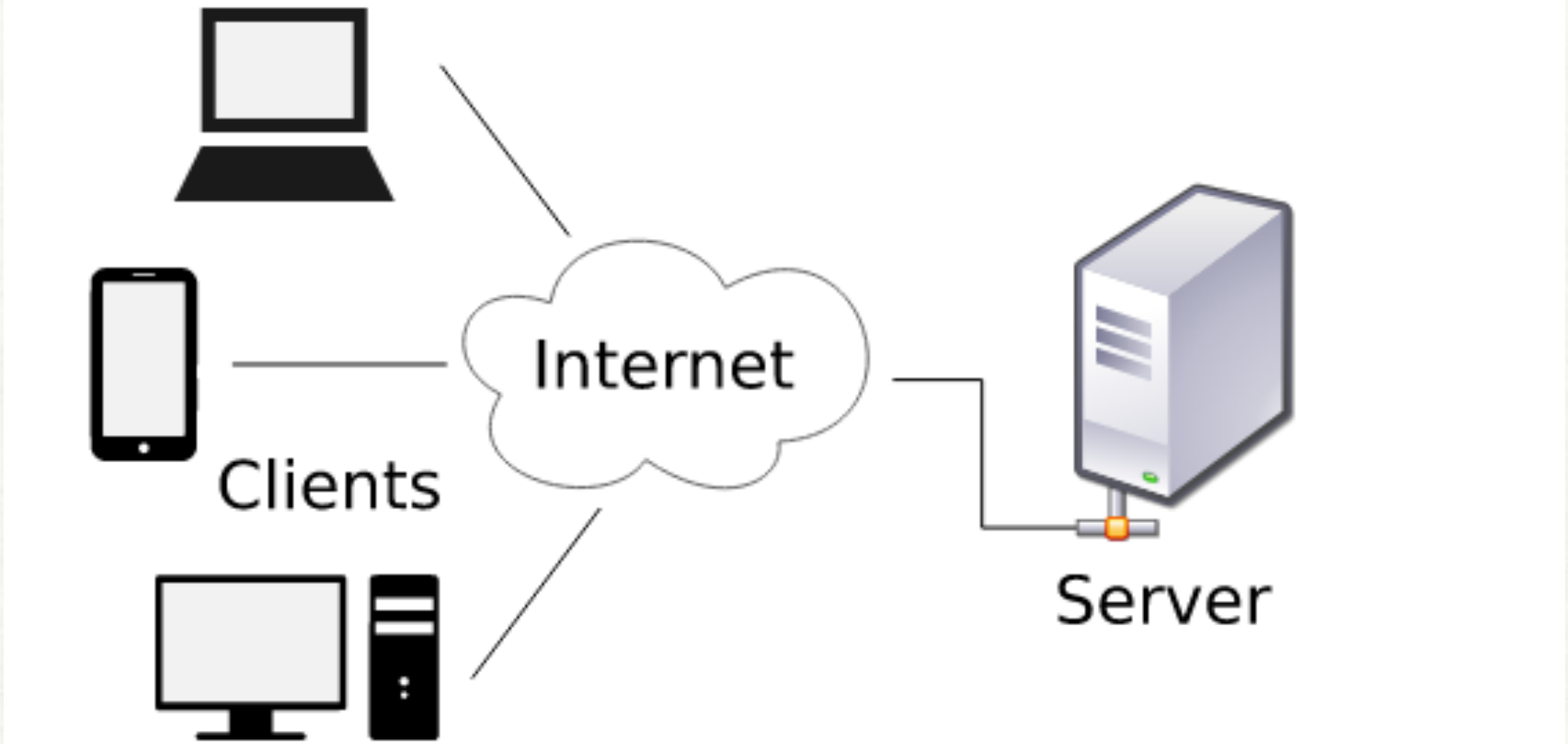
WHAT IS THE INTERNET

- First of all, it aint the web.
- The internet (lower case in AP style, finally) refers to the global telecommunications infrastructure that enables computers to send messages to one another. It’s been around since the 1960s.
- Can be partially destroyed by squirrels.

THE WEB

- The World Wide Web is a system of *servers* and *clients* that send requests through the internet and receive responses in the form of “hypertext”

- Hypertext is a type of document that can be interconnected with others, such as a link in text that opens up another document, or a photo embedded in that text.


Internet
Clients
Server

WHAT'S A SERVER?

- A server is just someone else's computer. Usually one without a screen that is just responsible for listening to HTTP requests.
- You can run a server on your own laptop too.
- You might have heard of 'THE CLOUD' which is basically another Orwellian marketing thing that really means for server infrastructure in some massive data center using ridiculous amount of electricity. It weighs a ton more than a cloud.
- Servers also have to have IP addresses that don't change periodically, as your home internet router does.

CLIENTS?

- A client is usually a browser or a mobile application.
- In modern web development, your client is often another server. For example, logging in with Facebook on a web app.

HTTP REQUESTS

What happens when you press enter in the url bar

- Chrome has no idea what http://www.nytimes.com actually means, so first, it looks up the IP Address for Google.
- DNS, or Domain Name System, is like a phone book containing IP Addresses for all the websites that exist on the internet.
- In October of 2016, assholes attacked large DNS servers, and disrupted a ton of business and my bosses were like “Vincent what did you do?” This is a fragile piece of the global internet infrastructure
- Chrome then sends an HTTP Request to the IP address associated with NY Times. An HTTP Request is essentially a text message asking the NY Times for data.

SERVER SIDE

- The NY Times server will issue a Response back to your browser. This response is parsed by Chrome. What comes back is code that we will learn today (front end code)

- The response always comes with a Status Code. Does 404 sound familiar?

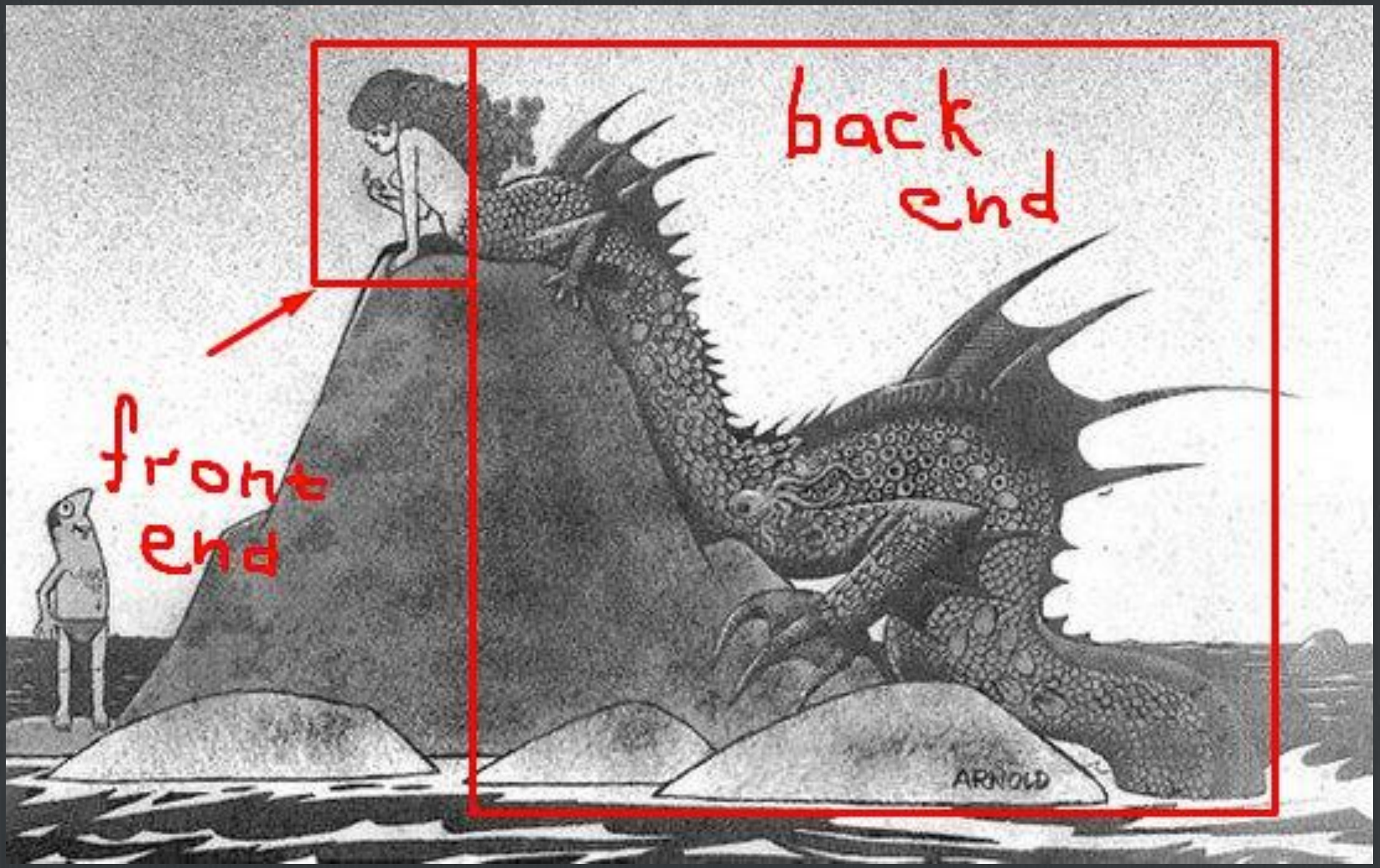
- A bunch of code gets executed on the Times server. Say you are asking for ‘section/world’ it will look in the database for recent articles in that category, assemble some code to display them, and send it back. If you are signed in, (we’ll explain what that entails soon) you will see other personalized stuff.

- Depending on what the server decides to send you, it sends code your browser understands. HTML, CSS, and JavaScript.

HTTP Methods

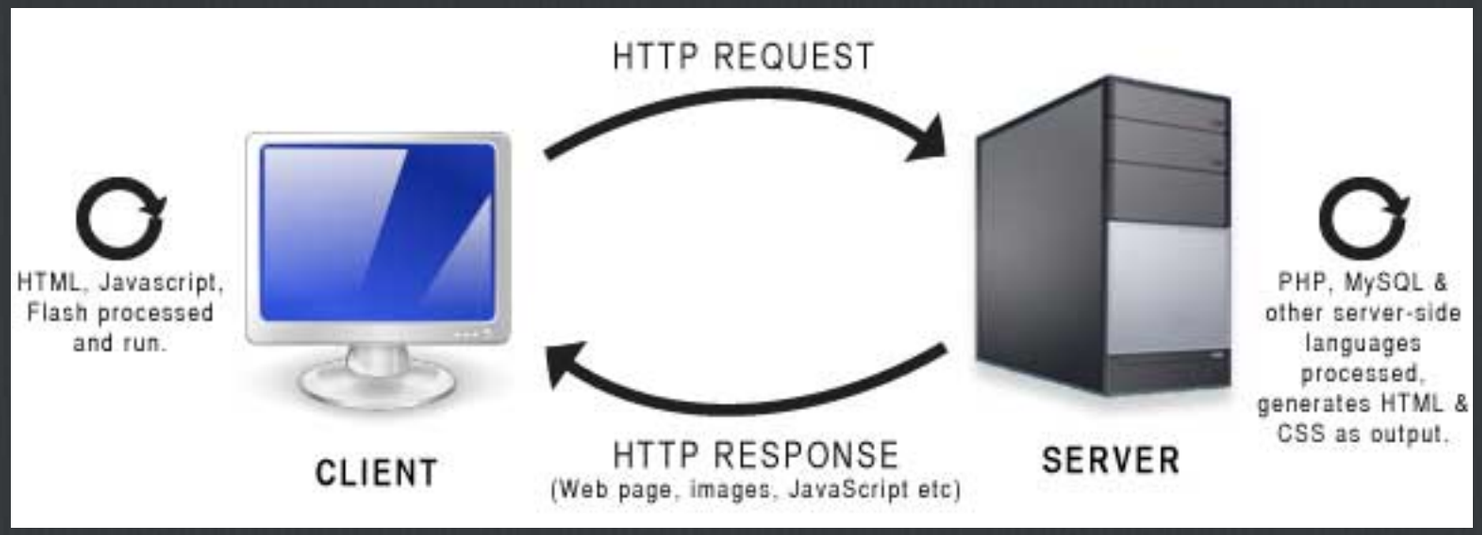
HTTP Methods and Their Meaning

Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data


back end
front end
ARNOLD

FRONT END VS BACK END

- Front end: Client side interactions and structure. This includes stuff like iOS apps and whatever you load in browser.
- Back end: Server side, data storage and retrieval. Needs a client of some sort (not necessarily a browser) to work. Deals with storing and retrieving information.
- Usually there are different programming languages used on the front and back, but nowadays, you can do it all in JavaScript



Front end

Back end

TO REITERATE

- Client sends a request.
- Server executes some server side code (maybe in Ruby, Node.js, PHP) and decides what to send back.
- The client (usually a browser) gets back a bunch of code, but this is code that the browser can understand. The browser uses this code to paint the web page for you.
- The client side (front end) code is not business critical stuff. Google isn't going to put its search algorithm into JS and send it to anyone. That happens on the black box that is the server.

- The back end of a web application is the part that happens on the server.
- Users interact with the front end, and the back end code decides what to do with it.
- Almost invariably, this involves interacting with a database for persistent storage.
- The *client* is whatever consumes the data sent from the back end. This might be a browser or a mobile app.

HTML

- HTML is the *structure* of a web page.
- Stands for Hyper Text Markup Language
- HTML 5 is the latest standard, which supports nice semantic naming of your HTML elements. 5 also supports richer media natively without the need for Flash or Java.
- It’s code, but not a programming language.

CSS

- On it’s own, HTML is quite ugly.
- Like worse than Craigslist ugly.
- To give HTML some game, we use a very simple thing called CSS — Cascading Style Sheets.
- The Cascading part just means that later declarations will take precedence over the earlier ones.
- CSS Zen Garden
- JavaScript comes tomorrow

Going further

- ☐ It's important to pace yourself and focus.
- ☐ Every programmer you meet will try and get you to follow his or her perfect holy amazing favorite thing.
- ☐ Stick to one thing, get good, then become a polyglot

Going further

- [ ] Codecademy
- [ ] Free Code Camp
- [ ] Stick with those two and you will be a maven before you know it