HELLO RAILS!

REMINDER TO VINCENT TO START THE SCREEN SHARE

How’s that midterm coming?

- This is a tough puzzle, but you can get it to work with all of the tools we learned in the last three weeks.
- You can pair and collaborate, but don't do it like painting a wall.
- This is for YOU, remember?
- You might want to try PRY

- We (for the most part) are done learning bare bones Ruby
- But that doesn’t mean we are done with Ruby. We will be taking it to the web now.

What we know so far:

- Variables
- Conditions
- Methods
- Classes
- Modules
- APIs
- Loops & iteration

You know Ruby

- Congrats! You know what you this does now:

```
puts “Hello World”
```

And this

```
name = gets.chomp
if name == “vincent”
  puts “Whats up vincent!”
end
```

And even this

```
class Person
  attr_accessor :name, :age, :birthday
end
```

```
Person.new
```

And not to mention

- Unix basics
- Git & Github workflow (not easy!)

Current Status

- We are a third of the way through
- You have 13 left after today
- Each class from here on out is 7% closer to making a full blown website

Today

- Intro To Rails
- Install Fest
- Structure Of A Rails App
- Getting Ready With Rails 4
- Make a working Application (but we will cheat)
- Lab Time

What is Ruby on Rails exactly?

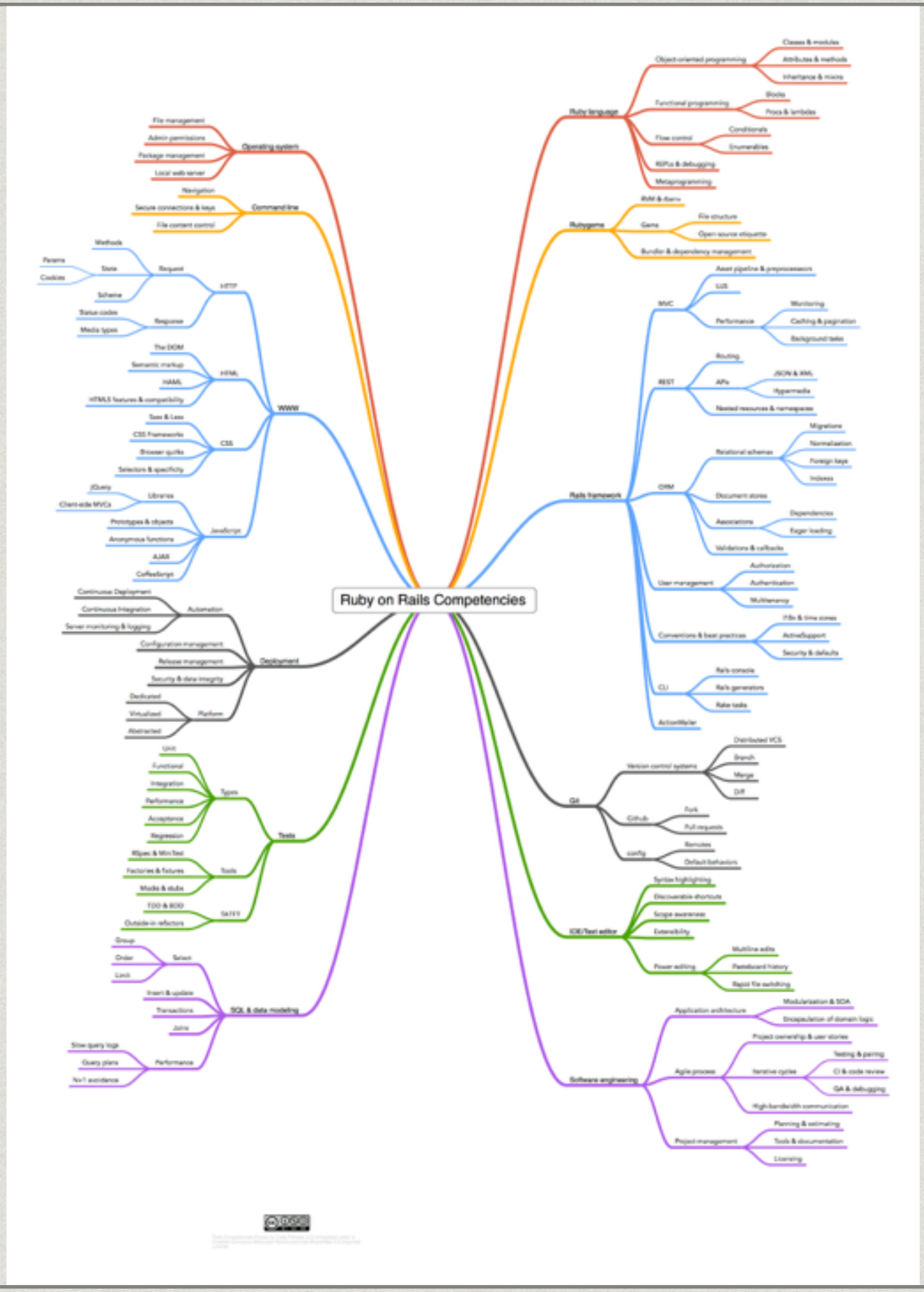
- Rails is a framework to build dynamic and robust web applications
- It is open source, which is a huge advantage.
- When we get down to it, it's mostly just a bunch of Ruby files.
- Throw in your own Ruby files and you have a working application!

Rails

- Started by David Heinemeier Hansson (dhh) at a company called 37Signals as a common-ground for their app, Basecamp.
- Over time, they slowly moved out the common components to this new framework, Rails.
- Open sourced in 2004
- Took off with “The 15 minute blog” (Search on youtube to find the video, 16:12)

- Rails conveniently abstracts SO MUCH of the process of developing a working web app. When it first appeared, people acted like DHH just gave them fire.
- It does the stuff that you would have to do anyway, letting you get to what you set out to do.
- Rails is all about convention. If you follow best practices, you can get web app or API built much quicker than you used to.

DON’T SAY IT’S EASY THOUGH


Ruby on Rails Competencies
Ruby language
Object-oriented programming
Classes & modules
Attributes & methods
Inheritance & mixins
Functional programming
Blocks
Procs & lambdas
Flow control
Conditionals
Enumerables
REPLs & debugging
Metaprogramming
Rubygems
RVM & rbenv
Gems
File structure
Open source etiquette
Bundler & dependency management
Ruby frameworks
MVC
Asset pipeline & preprocessors
I18n
Performance
Monitoring
Caching & pagination
Background tasks
REST
Routing
APIs
JSON & XML
Hypermedia
Nested resources & namespaces
ORM
Relational schemas
Migrations
Normalization
Foreign keys
Indexes
Document stores
Associations
Dependencies
Eager loading
Validations & callbacks
User management
Authorization
Authentication
Multitenancy
Conventions & best practices
I18n & time zones
ActiveSupport
Security & defaults
CLI
Rails console
Rails generators
Rake tasks
ActionMailer
Git
Version control systems
Distributed VCS
Branch
Merge
Diff
GitHub
Fork
Pull requests
config
Remotes
Default behaviors
IDE/Text editor
Syntax highlighting
Discoverable shortcuts
Scope awareness
Extensibility
Power editing
Multiline edits
Pasteboard history
Rapid file switching
Software engineering
Application architecture
Modularization & SOA
Encapsulation of domain logic
Agile process
Project ownership & user stories
Iterative cycles
Testing & pairing
CI & code review
QA & debugging
High-bandwidth communication
Project management
Planning & estimating
Tools & documentation
Licensing
Operating system
File management
Admin permissions
Package management
Local web server
Command line
Navigation
Secure connections & keys
File content control
WWW
HTTP
Request
Methods
Params
Store
Cookies
Scheme
Response
Status codes
Media types
HTML
The DOM
Semantic markup
HAML
HTML5 features & compatibility
CSS
Sass & Less
CSS Frameworks
Browser quirks
Selectors & specificity
JavaScript
jQuery
Libraries
Client-side MVCs
Prototypes & objects
Anonymous functions
AJAX
CoffeeScript
Deployment
Automation
Continuous Deployment
Continuous Integration
Server monitoring & logging
Configuration management
Release management
Security & data integrity
Platform
Dedicated
Virtualized
Abstracted
Tests
Types
Unit
Functional
Integration
Performance
Acceptance
Regression
Tools
RSpec & MiniTest
Factories & fixtures
Mocks & stubs
TATFT
TDD & BDD
Outside-in refactors
SQL & data modeling
Select
Group
Order
Limit
Insert & update
Transactions
Joins
Performance
Slow query logs
Query plans
N+1 avoidance

Some websites that use Rails

- Groupon
- Github
- Scribd
- Greenhouse
- Shopify
- General Assembly
- bloomberg.com
- Indiegogo
- SoundCloud
- Square
- Heroku
- Zendesk
- Crunch base
- How about we
- Hulu
- Funny or die
- Basecamp
- AirBnb
- DigitalOcean
- Twitter (until 2008)
- Kickstarter
- Blue Apron
- Yammer

Yeah. There are public companies that run on Rails.

Tobi Lütke
@tobi

Following

Shopify now runs on Rails 4.2. Same codebase started on Rails 0.5 in August 2004 roughly 11 years ago.

RETWEETS 187 LIKES 171

9:08 AM - 26 Aug 2015

INSTALL FEST!

Let's walk through this, one piece at a time.

- First, we need a package manager. Let's use Homebrew for Mac:
- If you have Fink or Macports installed, DO NOT DO THIS, flag down one of us instructors
- Go here: https://github.com/trivett/dev-environment-setup/
- And follow the instructions for **Homebrew**

- Installing homebrew:
- `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
- `brew doctor`
- `brew update`

- Next, we will install rbenv, which manages ruby versions so you don't need to use the system ruby.
- If you have RVM installed, DO NOT DO THIS.
- `brew install rbenv ruby-build`
- `echo 'if which rbenv > /dev/null; then eval "$(rbenv init -)"; fi' >> ~/.bash_profile`
- `source ~/.bash_profile`
- Now close your terminal, and open up another one.
- `rbenv rehash`

Now for Rails

- `rbenv install 2.2.3 #This gives us the latest`
- `rbenv global 2.2.3`
- `gem install rails --no-ri –no-rdoc`
- If the above command asks for permissions try rbenv rehash and then try again.
- `rbenv rehash`

If that worked:

- `ruby -v #=> should be 2.2.3`
- `rails –version #=> should be 4.2.4 or above`

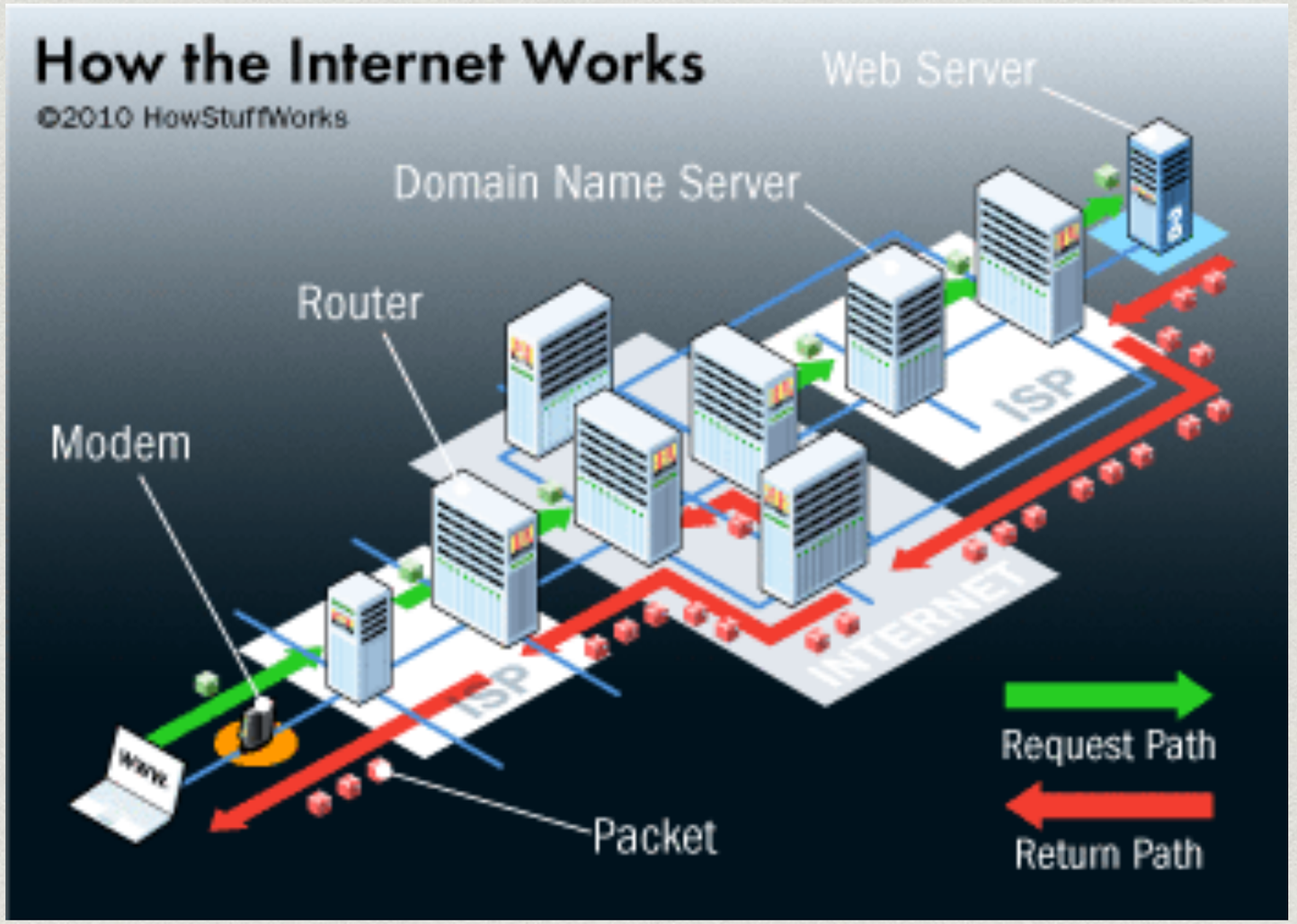
The

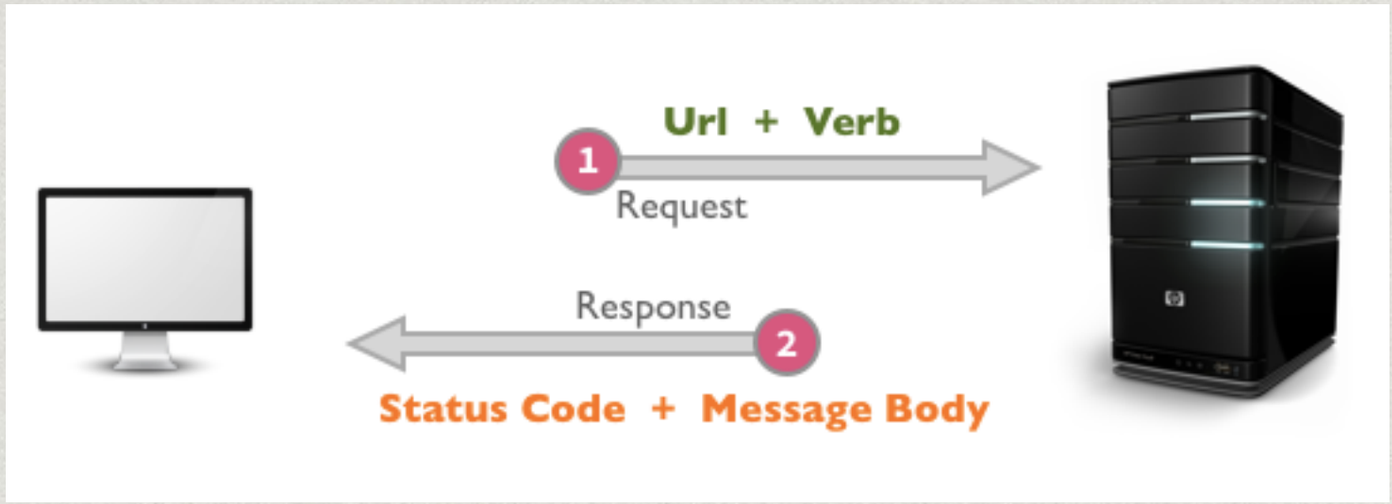
"Internet"


PROPRIETE CERN
CERN
PROPRIETE CERN
Information Management: A Proposal

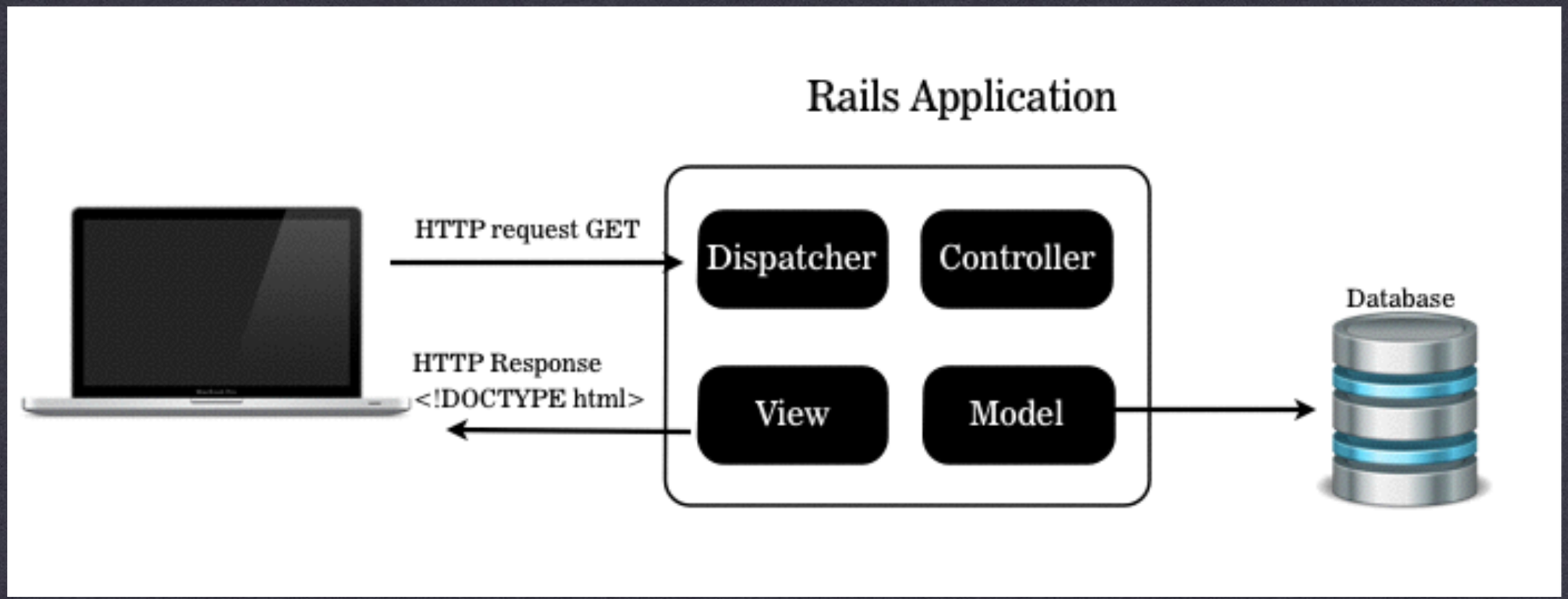
Get in your groups

- Draw a diagram that represents your understanding of how the Internet works. Here are a couple of questions to consider as you draw.
  - What happens when you hit enter on your address bar?
  - Where does a website live?


How the Internet Works
©2010 HowStuffWorks
Web Server
Domain Name Server
Router
Modem
ISP
INTERNET
ISP
Packet
Request Path
Return Path


Url + Verb
1
Request
Response
2
Status Code + Message Body

REQUEST RESPONSE CYCLE



- HTTP (the protocol that drives the Web) communicates via input/output just like the terminal.
- Stands for Hypertext Transfer Protocol
- Your Browser sends an HTTP Request to a server, and then the server then with an HTTP Response.
- Not entirely valuable for right now, but you'll hear the word "HTTP" a lot in conversation and in blog posts.
- HTTPS means "Secure"

The Dispatcher

The Dispatcher

- The Dispatcher takes the URL being requested, and figures out what to do with it.
- It bases its decision on the method of request and the path used.
  - GET /accounts
  - POST /accounts

HTTP Verbs — The main ones

- GET => Retrieve something from the server
- POST => Create something on server
- PUT => Update something on the server
- DELETE => Destroy something on the server

HTTP Status Codes

- 200 => OK, here's what you asked for
- 404 => Not found (if you haven't come across this, i can't even)
- 201 => Created successfully
- 500 => Server side error — The back end code is messed up
- And many more…

GO TIME

Your first Rails app

- Rails provides a bunch of commands to help you create and work with new applications.

- Go to your ~/Desktop directory, no need to commit this in the class repo.

- `Then rails new my_first_app`

- `cd my_first_app` `#(Easy to forget)`

What just happened

- Rails just created a bunch of files in the folder my_first_app
- That's it.
- These files make up the backbone of any Rails 4 application.

Your first Rails app

- `$ rails g scaffold Book author:string title:string abstract:text`
  - This create a new scaffold for an entity, “books”
- `$ rake db:migrate`
  - Rake is a tool for running tasks quickly
  - Creates and ensures our database is up-to-date.

You should see something like this

```
      create      app/helpers/books_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/books/index.json.jbuilder
      create      app/views/books/show.json.jbuilder
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/books.coffee
      invoke    scss
      create      app/assets/stylesheets/books.scss
      invoke  scss
      create    app/assets/stylesheets/scaffolds.scss
vincent@apple:~/my_first_app $ rake db:migrate
== 20151122211558 CreateBooks: migrating ======================================
-- create_table(:books)
   -> 0.0013s
== 20151122211558 CreateBooks: migrated (0.0014s) =============================
```

Your first Rails app

- Once we've scaffolded and ensured our database is proper, we need to run the rails server.
- The rails server responds to HTTP Request with Responses. (Your webpage, your HTML and Markup)
- To run the server
  - `$ rails server`
  - `—— OR a shortcut ——`
  - `$ rails s`

You should see something like this

```
vincent@apple:~/my_first_app $ rails s
=> Booting WEBrick
=> Rails 4.2.4 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
[2015-11-22 16:17:46] INFO  WEBrick 1.3.1
[2015-11-22 16:17:46] INFO  ruby 2.2.3 (2015-08-18) [x86_64-darwin15]
[2015-11-22 16:17:46] INFO  WEBrick::HTTPServer#start: pid=43045 port=3000
```

You are now running a server!

- Go to `localhost:3000` in your browser of choice

Now go to `localhost:3000/books`


AHHHHHHHHHHHHHHHH!
http://www.etsy.com/shop/sharpwriter
HEUSER 2012

CRUD

- Create, Read, Update, Delete
- These are the things that you can do with data.
- See how this lines up with the main HTTP Verbs?
- Mess around with the scaffolded site. Create a bunch of books in your browser. Throw in the Great Gatsby. Look back at your terminal. What's going on? Notice those HTTP status codes?

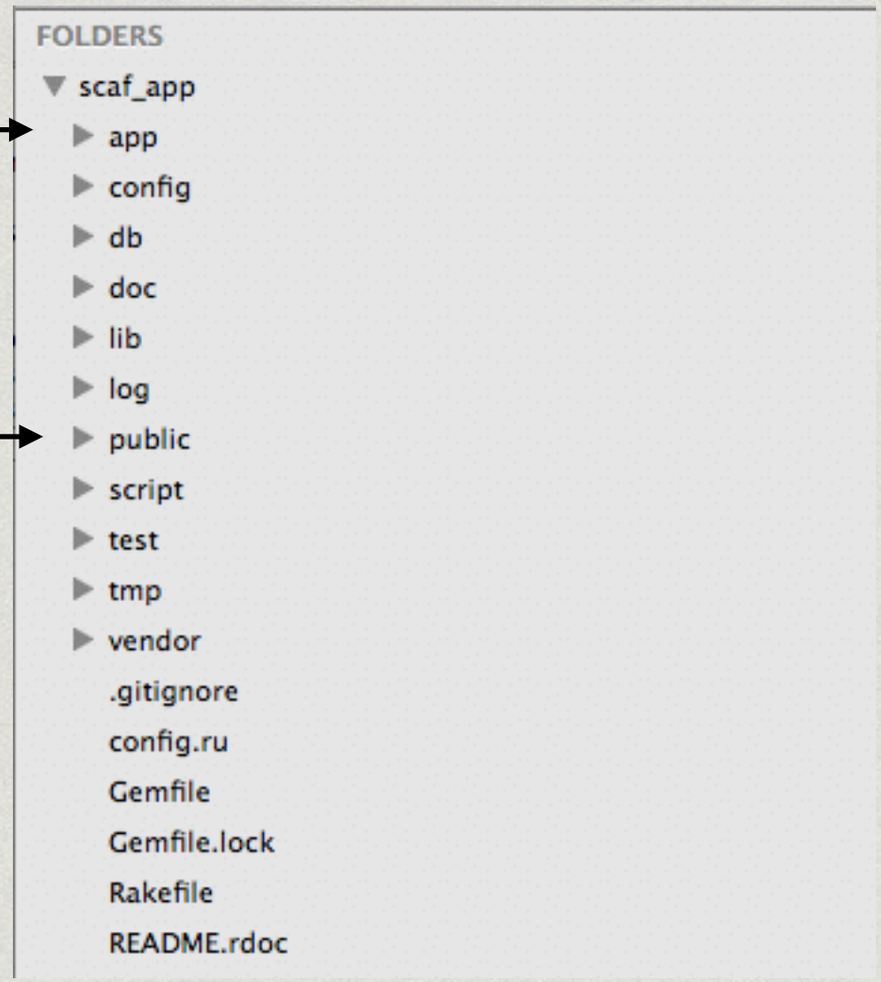
CRUD

- In your terminal, quit the server by doing `ctrl + c`
- Refresh.
- Start the server again with `rails s`
- Gatsby is still there.
- Use cmd + shift + f in sublime to search your entire project for 'gatsby'. Where is it?

App structure

Most of your app lives here

This is where directly accessible files live



Directories and their purpose

- `app/assets`: CSS, JavaScript and images used in templates.
- `app/**controllers**`: Classes which gather data and render responses for specific requests.
- `app/helpers`: Utility modules which define methods for templates to use.
- `app/mailers`: Classes that define various outbound emails in your app.
- `app/**models**`: Classes defining your data models (e.g.. User, Comment, Post, Book).
- `app/**views**`: Templates called by your controllers to render HTML.
- `app/views/layouts`: Master templates to be used throughout your website.

You try it!

- Create a new application called user_scaffold using “`rails new making_users`”
- Create a scaffold called `users` and it should have the attributes / fields:
    - First Name
    - Last Name
    - Age
    - Email
- Run your server and go to localhost:300/users and mess around
- If you finish early, see if you can find a way to make it impossible to edit or delete users.

Static files in Rails

- Shut down your server by typing “control + c”
- Run “touch public/hello.txt”
- Add some random text to the file in sublime
- Start your rails server again
- Head to “http://localhost:3000/hello.txt”

Recap

- Scaffolds provide the bare necessities for creating, editing, deleting, and viewing a resource. Otherwise known as CRUD.
- Scaffolds are great when getting started with Rails or a new app, but are generally not used in production websites.
- Scaffolding is not considered best practice, and should generally only be used initially as a means of learning (the primary drawback of using them is that you generate lots of code that you don't need, and it's difficult to undo).

HOMEWORK

Thanksgiving homework:

If you do not know basic HTML

- Sign up for Dash from GA
- https://dash.generalassemb.ly/projects
- Complete at least part 1, do part 2 if you wanna get fancy.
- HTML (and maybe a bit of CSS) knowledge will come in handy very soon.

Thanksgiving homework:

- Read about HTTP: http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177
- Just read, no need to code.
- Read a bit about HTTP Methods http://www.restapitutorial.com/lessons/httpmethods.html
- Mnemonic device https://http.cat/