Cloud Computing & Web Request Frameworks

Hello!

I am xinYu

NUS BiZiT
(Chairman)

Year 3 Information Systems

DBS Risk Compliance Intern

MEET ME!:
x.x.x.x@u.nus.edu / xinyu@nusbizit.org
H/P: 96538159

NUS BIZIT

Nurturing BIZIT Talents

- Side-projects
- Networking
- Competition
- Internships

FB: Facebook.com/groups/nusbizit

Fun Facts

Focus Areas

- Cloud Computing
- Social Media Analysis
- Enterprise Systems Development
- Risk Compliance (Controls Testing)

Fun Facts

About me

NO PERSONAL facebook account (FAN PAGE)

I hate “programming” | LOVE Code generation

I hate start ups (LOVE the culture not the HYPE)

I flunk CS courses
CS1010/1020/1231/2100/2102
Left to flunk: CS2105

Instructions Before “Workshop”

Subscribe to Github Student Developers Pack

https://education.github.com/pack

Microsoft DreamSparks (NUS)

Sign in to DL Windows Products Server / Windows Products etc.

https://e5.onthehub.com/WebStore/Security/Signin.aspx?ws=b6ca5e81-649b-e011-969d-0030487d8897&vsro=8&rurl=%2fWebStore%2fProductsByMajorVersionList.aspx%3fcmi_cs%3d1%26cmi_mnuMain%3dbdba23cf-e05e-e011-971f-0030487d8897%26ws%3db6ca5e81-649b-e011-969d-0030487d8897%26vsro%3d8

AWS Educate

Sign up as AWS Educate for Student

https://aws.amazon.com/education/awseducate/

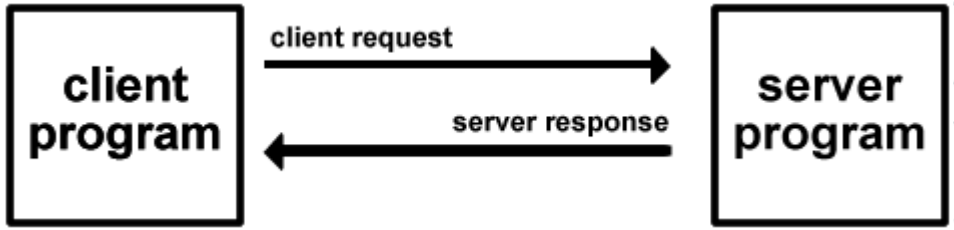
1.

Cloud computing

Building Castles in the SKY

The practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.

Client-Server Computing



Client	**Server**
Front-end	**Back-end**
Peer	**Host**
Leecher	**“Seeder”**
Download bias	**Upload bias**
Browser	**Web Server**
Native App	**App Server**
	DB Server

2.

Cloud Collaboration

Code Focus

Collaboration Tools

- C9.io
- https://github.com/
- https://about.gitlab.com
- https://www.sourcetreeapp.com/

Cloud Learning Tools

- https://www.codecademy.com/
- https://www.hackerrank.com/

3.

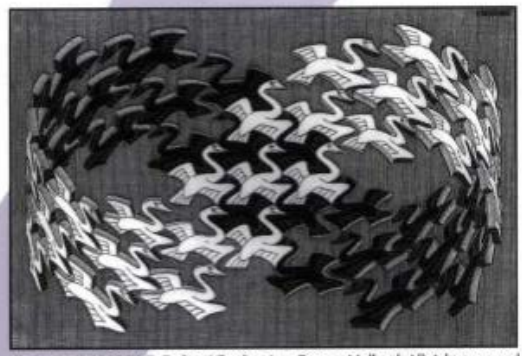
Web Methodologies

MVC Focus

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Acronyms

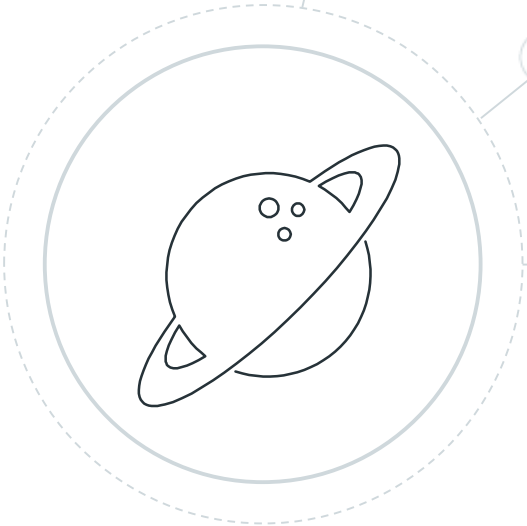
- CRUD (Create Read Update Delete)
- DRY vs WET (Dnt Repeat Yourself)
- SSOT (Single Source of Truth)
- SOC (Separation of Concerns)
- ROT (Rule of Three)

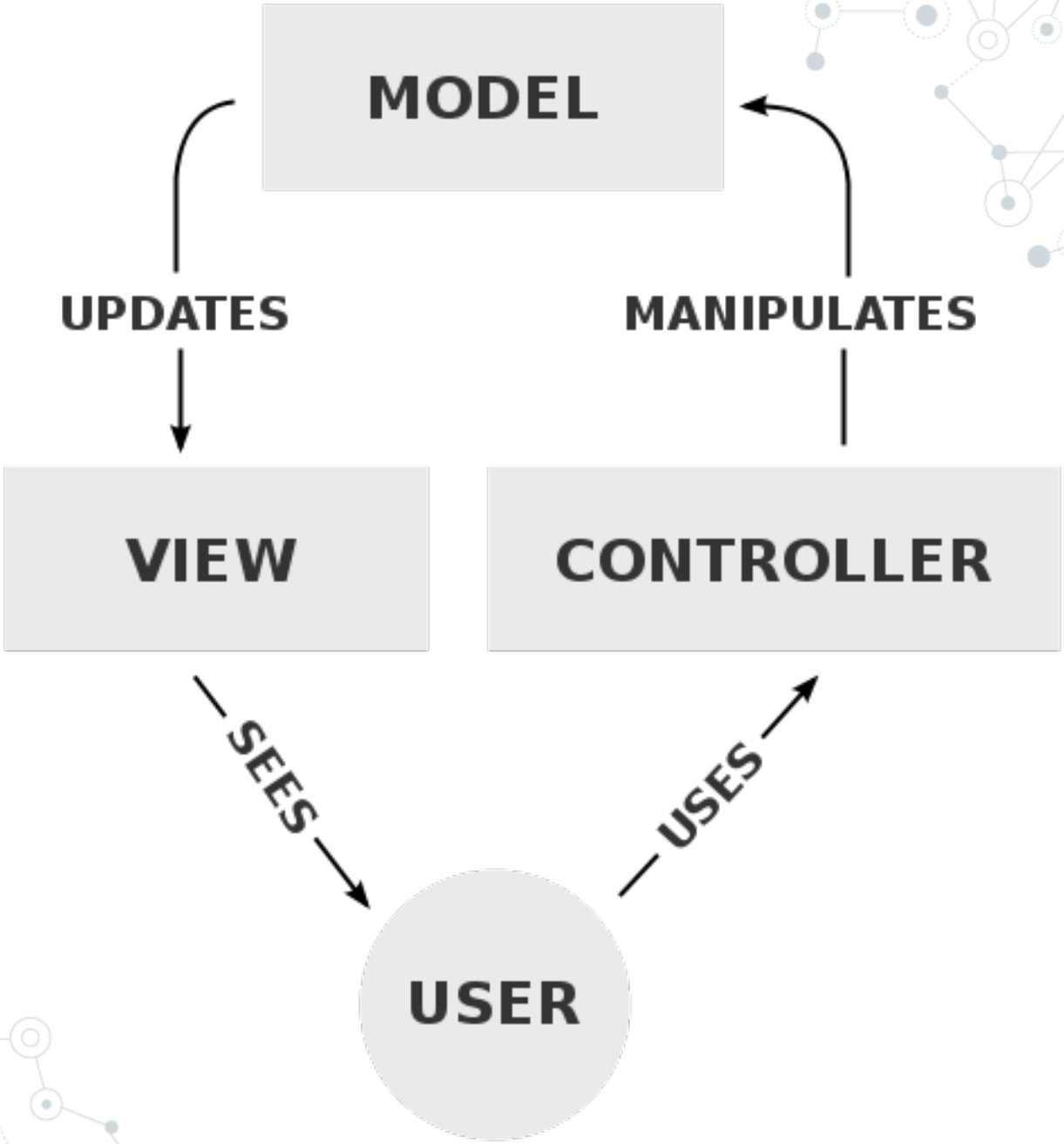
SOLID!

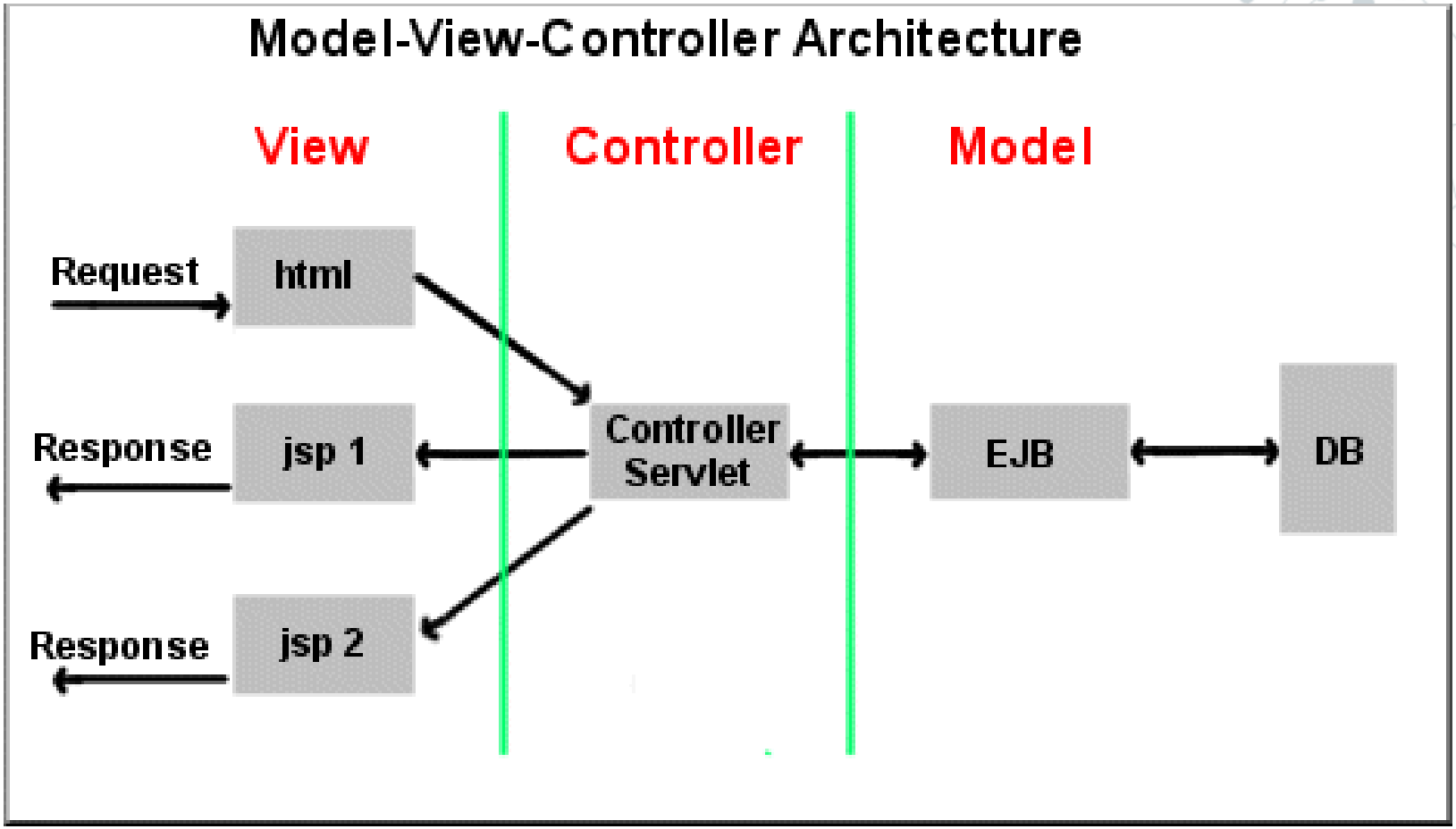
Single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion

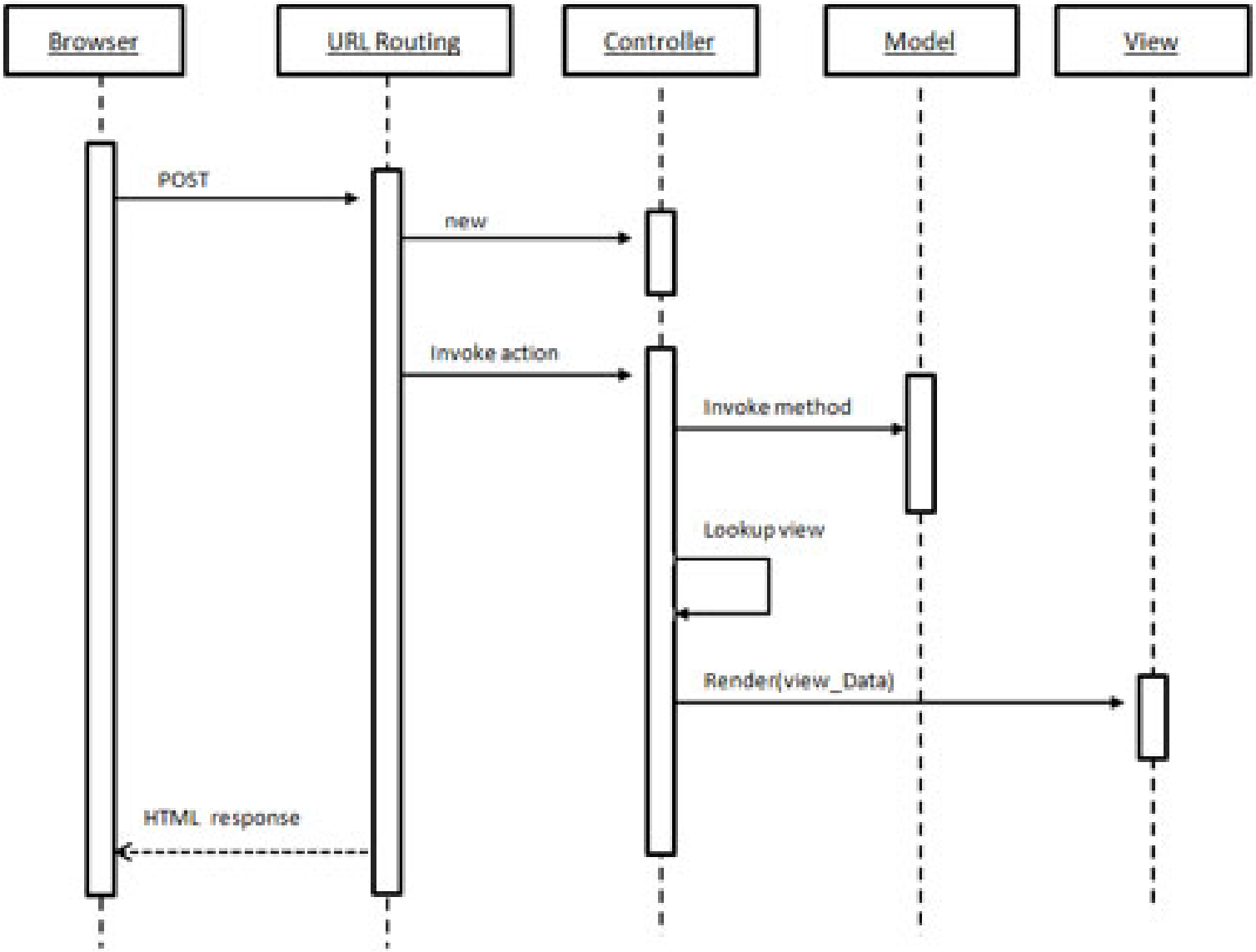
MVC

Model
View
Controller




MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

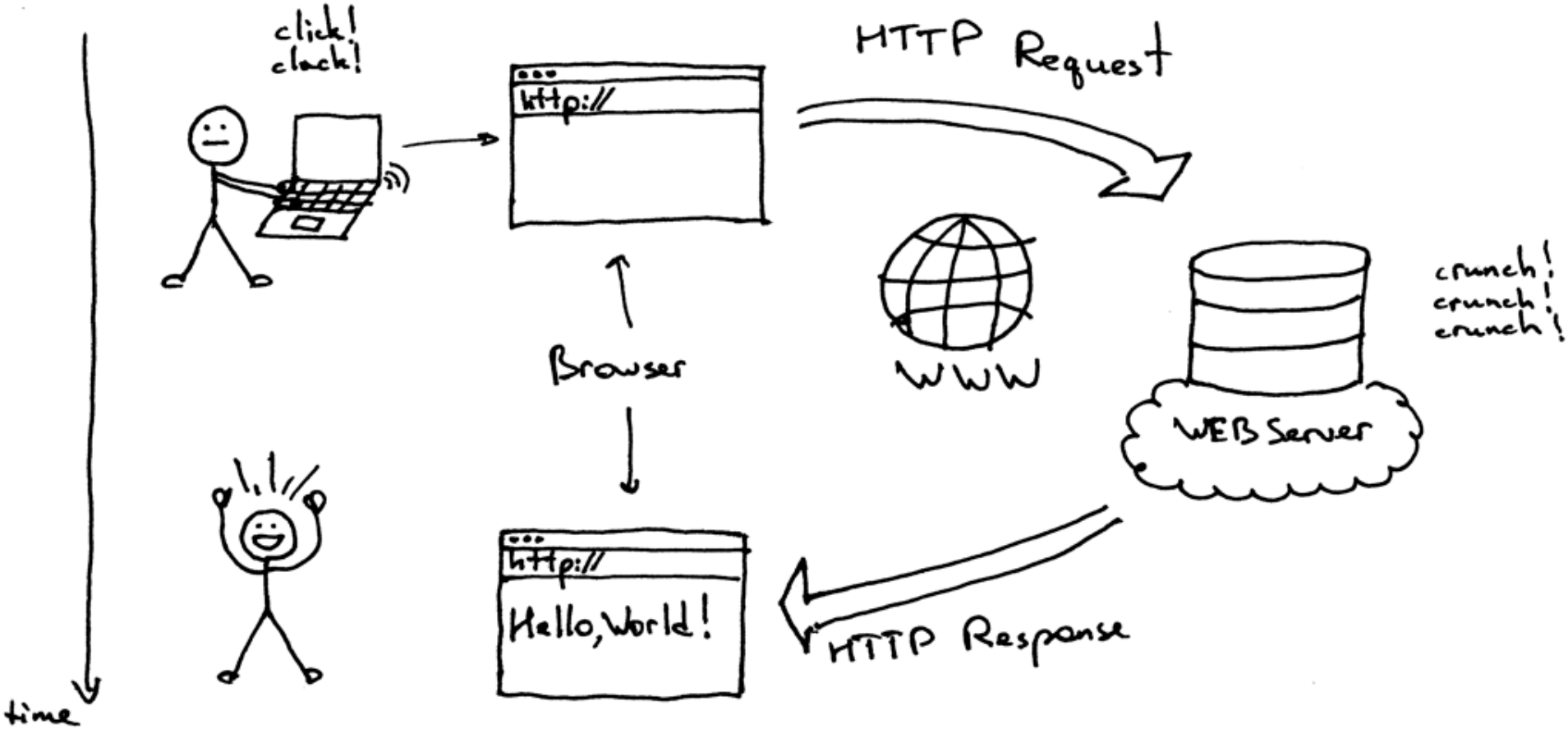

Model-View-Controller Architecture
View
Controller
Model
Request
html
Response
jsp 1
Response
jsp 2
Controller Servlet
EJB
DB


Browser
URL Routing
Controller
Model
View
POST
new
Invoke action
Invoke method
Lookup view
Render(view_Data)
HTML response

4.

Web Requests

Just basic HTTP Requests


click!
clack!
http://
HTTP Request
Browser
WWW
WEB Server
crunch!
crunch!
crunch!
http://
Hello, World!
HTTP Response
time

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

method path protocol

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: net.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```

HTTP headers as Name: Value

Status Codes

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error

Status Codes

- 200 OK
- 403 Forbidden
- 404 Not Found
- 408 Request Timeout
- 500 Internal Server Error

OOPS!

500
(Server)
Internal Server Error

Webapp Crashed!
WebServer Crashed!
Path wrong…
NullpointerExceptions…
I HATE YOU!

Basically your SERVER screwed up

403/408>404
(Client)
Unauthorised?

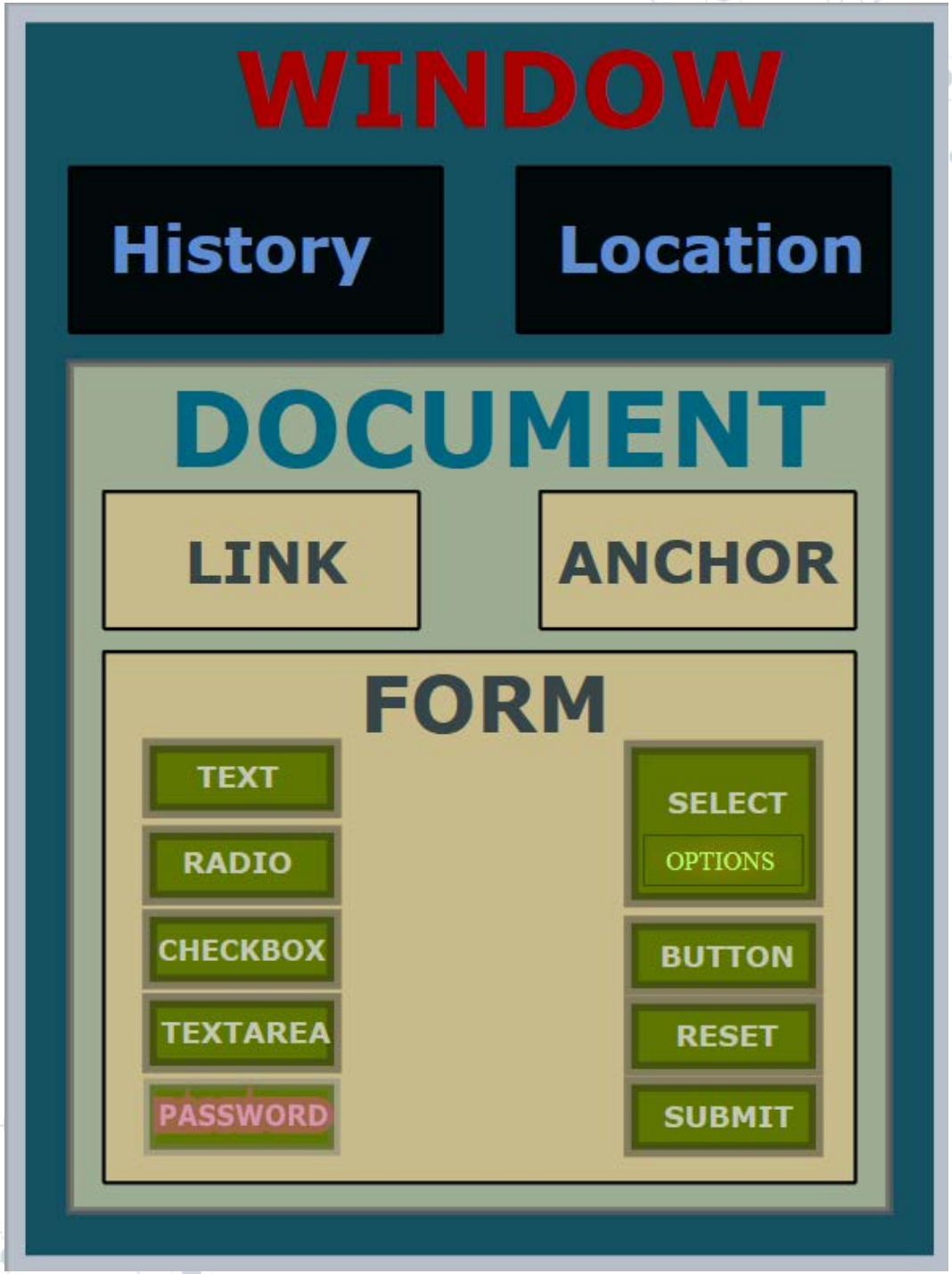
Time-out…

Client side is unable to access interface on the server side…

Resource is not found as there is no such URI

200

- GET an entity corresponding to the requested resource is sent in the response;
- HEAD the entity-header fields corresponding to the requested resource are sent in the response without any message-body;
- POST an entity describing or containing the result of the action;
- TRACE an entity containing the request message as received by the end server.


WINDOW
History
Location
DOCUMENT
LINK
ANCHOR
FORM
TEXT
RADIO
CHECKBOX
TEXTAREA
PASSWORD
SELECT
OPTIONS
BUTTON
RESET
SUBMIT

5.

Frameworks

Common Web Frameworks

Why use Frameworks

- Re-invent the wheel
- Easier Maintenance*
- Good Coding Standards (PSR2)
- Pretty URLs (mydomain.com/nice)
- Helpers Function (Import includes)
- Low Level Errors

(SQL Injection/Routing errors)


METEOR


RAILS


ANGULARJS
by Google


Sinatra


django


spring
by Pivotal


Flask


Symfony


laravel

What Framework to USE?

How to choose Frameworks?

AJAX (Asynchronous Javascript and XML)
DB Migration (DBMS)
Testing (Inbuilt)
Security (Sanitisation of inputs, web app firewall)
Templating (View)
Caching (Cluster / Static cache)
Form Validation (For user input dat)

Specific
User account controls / Mail server interface etc.
E-commerce | Games | Payment |

Bootstrap

Foundation

Start here, build everywhere.

jQuery

write less, do more.

AJAX

Asynchronous Javascript And XML

IDE LEARNING CURVE

6.

Stacks

Overflow…

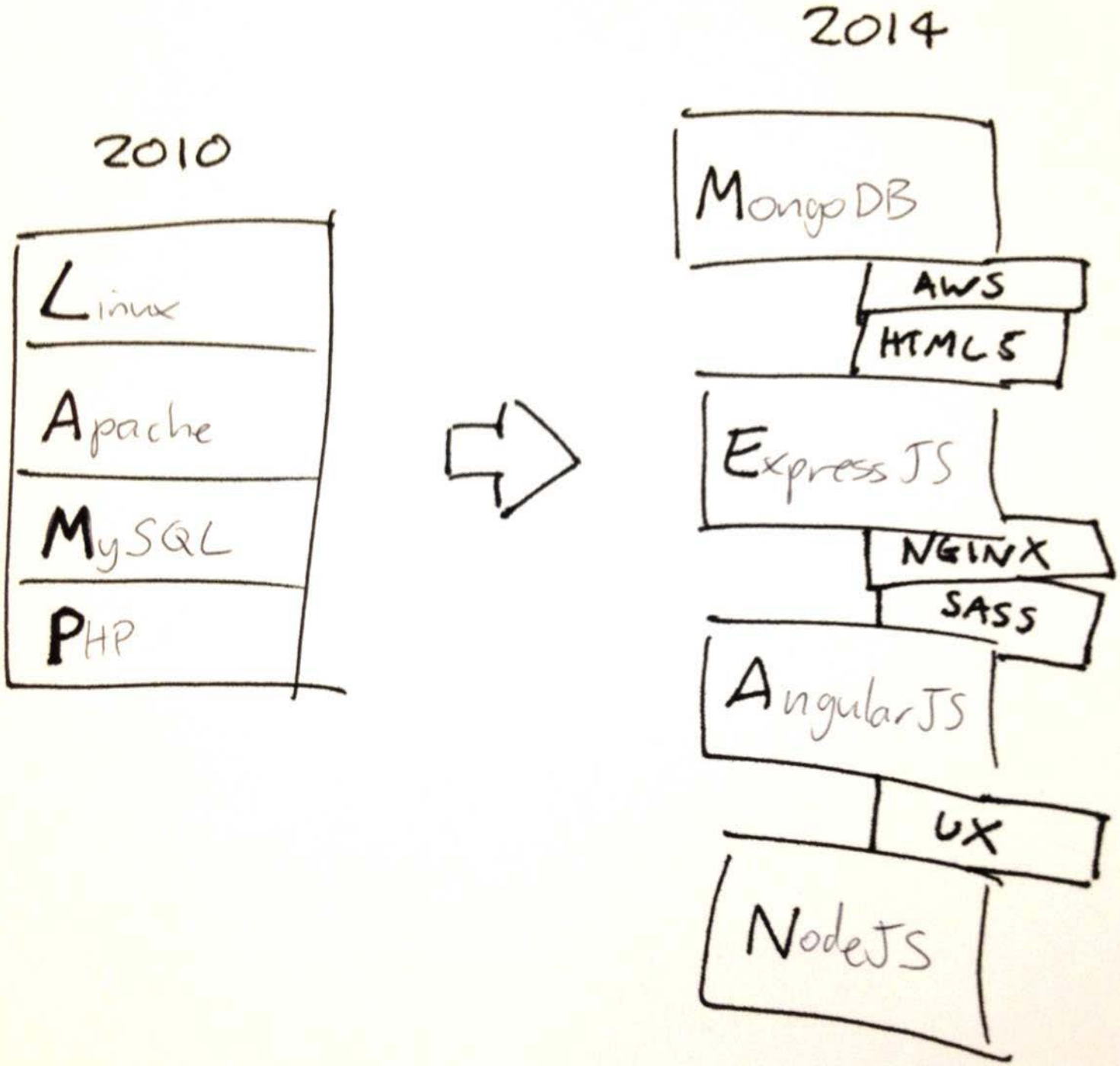
FRONTEND DEVELOPER VS. BACKEND DEVELOPER

The frontend of an application is distinctly human. It's what the user sees, touches and experiences. Whereas, the backend of a web application works behind the scenes enabling the frontend experience. Let's take a look at what the frontend and backend developers bring to the table.

	FRONT END	BACK END
Philosophy	Insatiable passion for building amazing applications that customers love to use	Hacking away to create a sterling architecture and working infrastructure without reinventing the wheel
Quoted Saying	"Dude, it's all about the user experience and has to look good (form over function)"	"If this approach doesn't work, let's try another one."
Favorite Tools	Angular, CSS, HTML5, JavaScript, jQuery, Bootstrap, Ionic	PHP, C#, Ruby, Python, MySQL, Node.js, MongoDB
Strengths	Creative, Imaginative, Innovative, Driven, Empathy, Curiosity, Personability	Flexible, Organized, Detail-Oriented, Handle Complexity, Analytical, Logical
Lifestyle	Blazin' monitor and MacBook Pro on desk, iPhone and Red Bull in hand, wearing a snarky t-shirt and skinny jeans	Empty coffee cups littered on desk next to an old school but powerful laptop with a game console nearby, sitting in a freebie t-shirt and cargo pants

"THE FULL STACK DEVELOPER"

(no other developers required)


2014
2010
Linux
Apache
MySQL
PHP
MongoDB
AWS
HTML5
ExpressJS
NGINX
SASS
AngularJS
UX
NodeJS

7.

VM Hosts

Demo

VM Hosts

https://www.digitalocean.com/
Own stack! Cheap!

https://www.heroku.com/
Deployment Integration

https://aws.amazon.com/
I am too RICH

https://azure.microsoft.com/en-us/
Microsoft nuff said

https://cloud.google.com/
Sponsored…

7.

Deployment Checklist

What to consider minimally

Mini Checklist on deployment

-Server Platform
-CLOUD/Self-hosted
-Setup (Running costs)
-Security* (Sanitisation esp)
-Mail servers (Self-hosted, Mail Service)
-Payment Function (APIs)
-Routing (mod_rewrite)
-Encryption (Hash Suite)
-Domain (DNS, CA)

8.

Reality Check

Can I do this?
Managing Expectations

Some Jokes about Learning Web

They tell you you don’t have to learn anything else except **Rails/Django**, that this is one of the greatest advantages it has.

> Oh, except you **need** to learn **Ruby/Python**

> Oh, but you **do** know about **MVC** right? The whole Model, View Controller philosophy? Right?

> No, no not that hard, a couple of months and you'll start to begin to understand it.

Ok, so I'm learning **Ruby/Python**, loving it as a language, will never go back to **PHP** again

> Well, you can't do shit without **Gems/Apps**

What?
Gems, apps, that’s where the magic of Ruby and Rails/Django lies, well the added magic that is, Rails/Apps has some magic all its own.

> For example there’s a gem/app for writing HTML forms, instead of writing all the <input> and html crap, you install the gem/app, write a few lines of code and it creates a whole form for you.

> Sure, it’s awesome, once you get the hang of how to install and maintain your gems.

> **RVM**, which is actually better than the built-in gem management system, which sorta works, but then gets confused and the gems and Ruby versions that ship with OS X are crap and broken, so use RVM, it’s better.

> Hold it! RVM has its issues as well, it can be temperamental and refuse to listen, plus it relies on **Xcode** to compile Ruby and some gems and if you have the latest version of Xcode……

> Easy, just use **rbenv**, it’s better than RVM anyway because it’s more unix-like

Yeah, good luck, first you have to install it with Homebrew

What?
and then you have to learn its syntax and then maybe you can begin using it.

Well shit, how many things do I have to learn to be able to make a Rails app?

- **Ruby**
- Rails **philosophy**
- Rails **commands**
- **ActiveRecord** (the bit that talks to your database)
- A **database** language, such asMysql though Rails promises you won't need to learn any of that
- **HTML** (you're making web pages right?)
- **CSS** (those pages are gonna look like shit if you don't know CSS)
- **Gem management**
- **Bundler** (Gem to manage other gems, one gem to rule them all)
- **Rails asset pipeline,** meaning how it serves images, CSS and javascript
(not straightforward, no)
- **Rake**, it's yet another command line tool to do things like manage your asset pipeline by compiling images, CSS and javascript files
- **Routing in Rails**. How to get Rails to show you the page you want. Out of all of Rails, routing should be the easiest and most straightforward, yet it's probably the most obtuse bit of all
- **Javascript** or at least **jQuery** since javascript is awful
- **ERB syntax** (Erbs are the equivalent of HTML to Rails)

But after you learn ERB syntax, you realize it sucks and so you have to learn to use alternatives like HAML which is much nicer, but the syntax is quite tricky for such a simple language

So you learned CSS, good for you. But, again, in the process of learning CSS you saw that it has a lot of shortcomings

>So time to learn another alternative called **SASS**

you're kidding right?

Well, you gotta test before you do anything.

What do you mean, like on a browser?

Please. You have to learn to write tests in Rails.

How do you do that?

Easy, just learn the built-in test syntax and structure and methodology and you’re all good.

Ok, let me be honest, the built-in tests suck, you have to learn them, but they suck and you won’t even end up using them cause they suck.

>You want to use **Cucumber**.

You’re sure? Nothing else, you won’t tell me to switch to something else after this will you?

Oh, all right, **Cucumber** is better than the built-in shit, but it’s lengthy and kinda retarded because your tests are actually stories that you write, and end up being probably longer than the actual code it takes to make things happen so you can then test them, plus we’re programmers right, not failed copy writers so forget Cucumber. What you want is **Rspec**.

After six months you'll be lucky if you have this

But that's the default "**Welcome to Rails**" page you get when you create a new app, you can get that by simply typing

rails new app_name

Yeah, that's right.

Ok fine how now let me **ftp** into **webserver**

Nah you call it **deploying**?

What do you mean deploying? It’s a web app, you just ftp your folder to a server and you’re off.

This is Rails, you have to deploy your application.
What the hell does that mean?
Easy…

What you want is **Capistrano**.

But you know what, I’ll save you some heartache right now and tell you to go **with Phusion Passenger** instead, it’s easier and more modern than Capistrano.

It needs to compile against the **Mysql** source so you need to get those, make sure you've installed Mysql the proper way (as opposed to just using a double-click installer,

Make sure your Ruby installation is kosher (good luck if you had to fight with **RVM** and **rbenv** and the local installations),

Then you need to edit your **apache httpd.conf** files to make sure your Rails app will work

Oh and you need to use **Virtual Hosts** in apache, which, you know, means you have to learn how to edit the apache httpd.conf files properly, and then you…

OK GO **** YOURSELF…

*Do not **Framework Hop** during learning phrase*

Choose one STICK to it

“

*Please Still use **Frameworks***

ROR/DJANGO/LAVAREL

“

Be humble

(Do not think like ninja coder)

“

Google your problems/errors!

Read APIs Documentations

Thanks!

Any questions?

You can find me at:
@xyme_xinyu (Slack)
Me.contact.xy@gmail.com