GENERAL ASSEMBLY

PRODUCT MANAGEMENT

TECHNOLOGY FOR PRODUCT MANAGERS

AGENDA

- How the Internet Works
- Programming Languages & Terminology
- Constraints & Solutions
- Bringing It All Home

LEARNING GOALS

- Define key terminology regarding creating digital products
- Recognize common constraints and solutions that product managers face with digital products.

YOU DON'T NEED TO BE AN ENGINEER, BUT YOU NEED TO SPEAK IT.

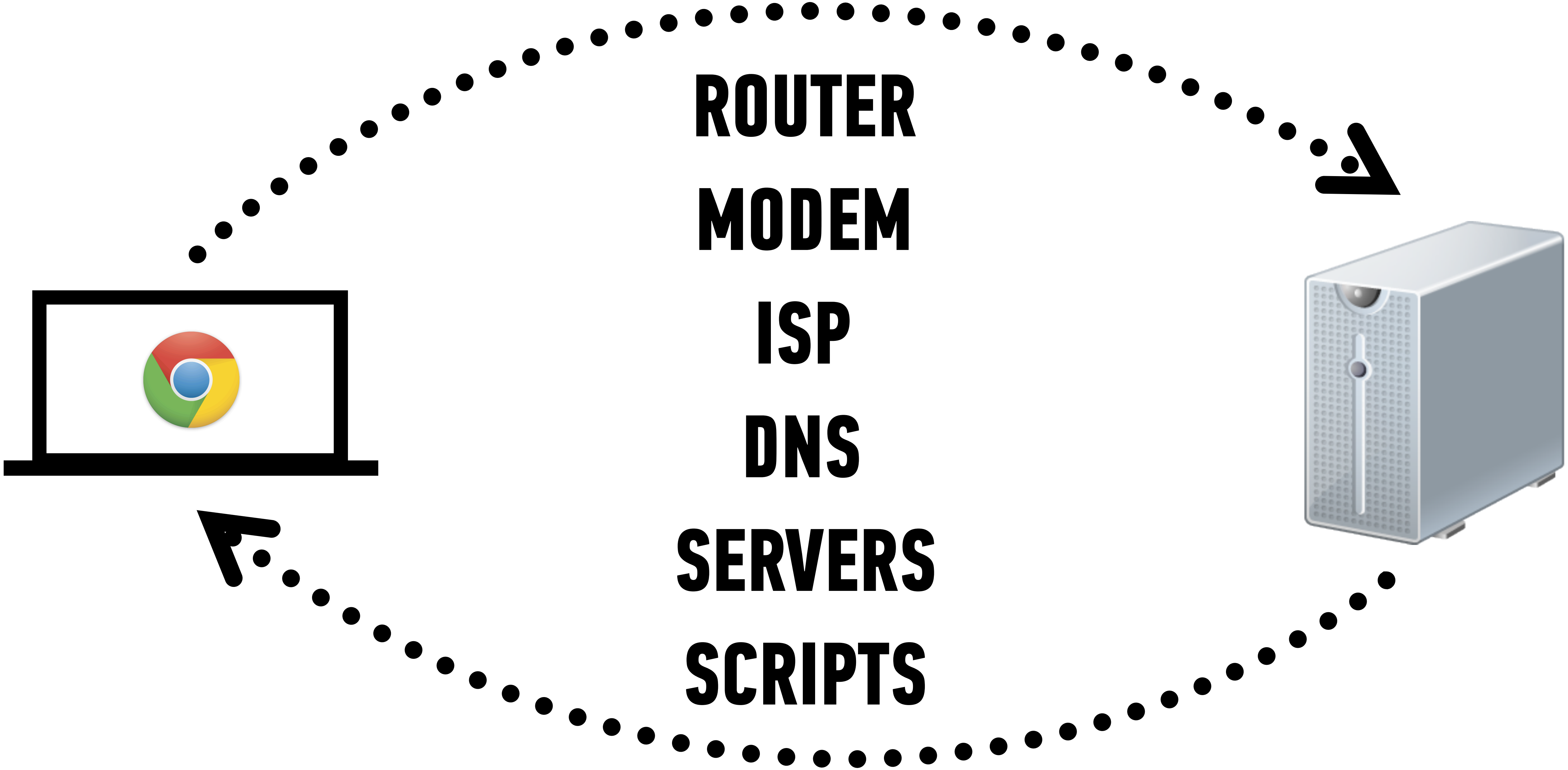
- Having a foundational understanding of how products are built is key to:
  - Creating something that can realistically be built
  - Building a relationship with engineers
  - Understanding the implications of the decisions you make

YOU DON’ T NEED TO BE AN ENGINEER, BUT YOU NEED TO SPEAK IT.

THIS LESSON IS JUST THE BEGINNING. YOU WILL NOT LEARN EVERYTHING YOU NEED TO KNOW BY THE END OF THIS LESSON.

HOW DOES THE INTERNET WORK?

HOW THE INTERNET WORKS



ROUTER

MODEM

ISP

DNS

SERVERS

SCRIPTS

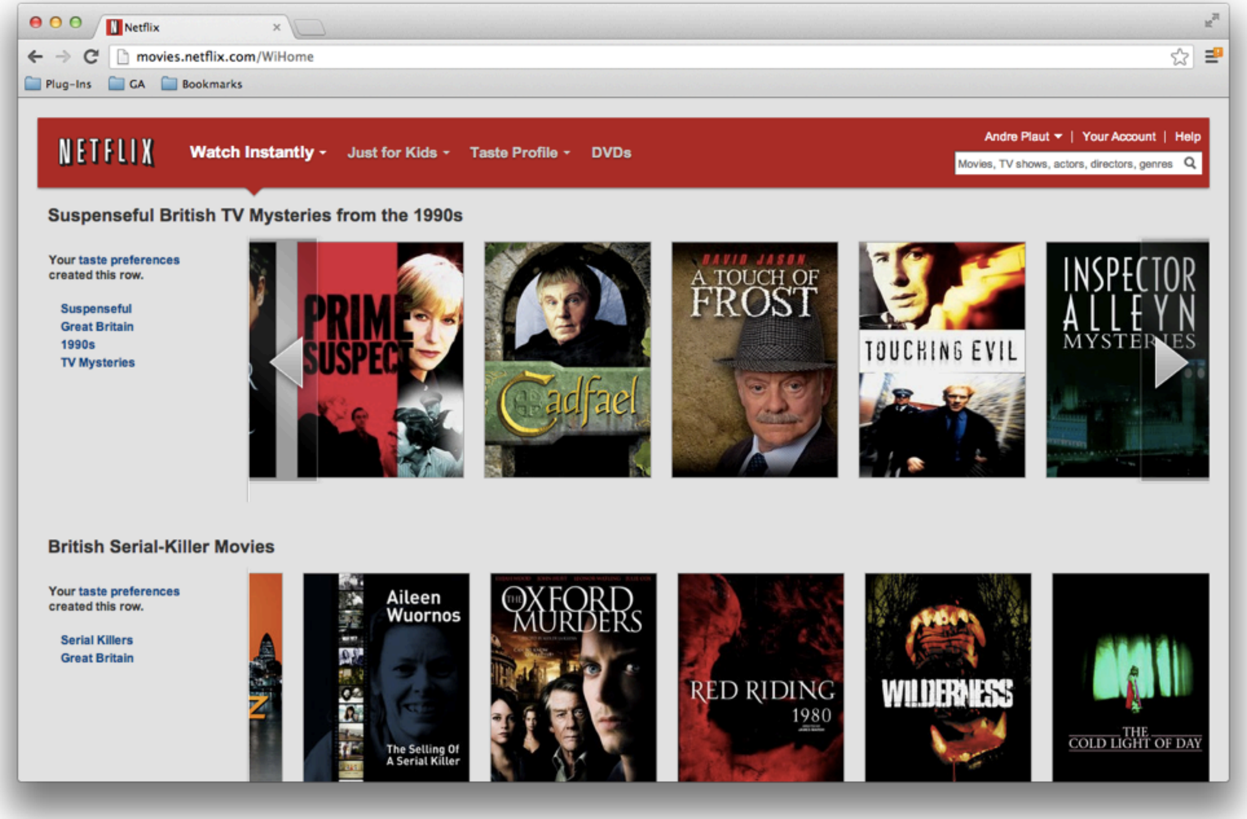
WHO HERE HAS SOME PROGRAMMING EXPERIENCE?

FRONT-END

WHAT USERS SEE AND INTERACT WITH

BACK-END

THE PROCESSES THAT ALLOW THE FRONT-END TO REACT, ADAPT, AND FUNCTION



Suspenseful British TV Mysteries from the 1990s

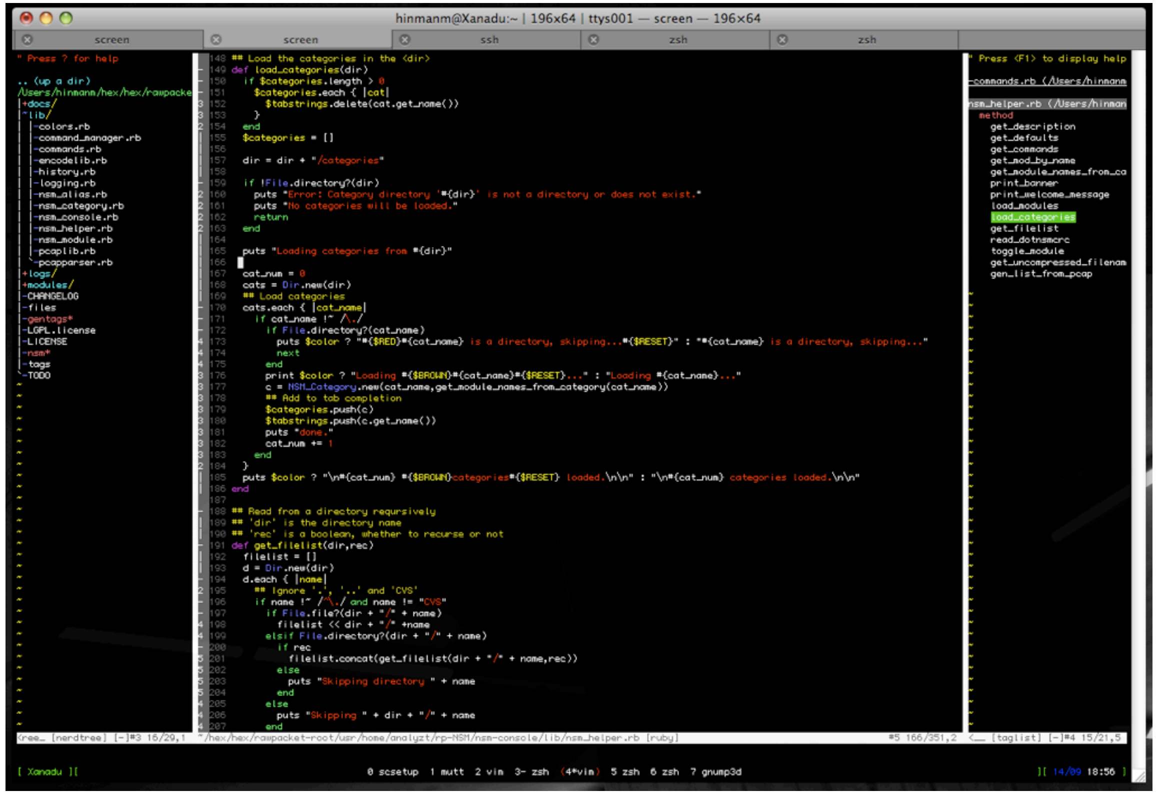
Your taste preferences created this row.

Suspenseful
Great Britain
1990s
TV Mysteries

British Serial-Killer Movies

Your taste preferences created this row.

Serial Killers
Great Britain



```
" Press ? for help

.. (up a dir)
/Users/hinmann/hex/hex/rawpacke
|+docs/
|~lib/
| |-colors.rb
| |-command_manager.rb
| |-commands.rb
| |-encodelib.rb
| |-history.rb
| |-logging.rb
| |-nsm_alias.rb
| |-nsm_category.rb
| |-nsm_console.rb
| |-nsm_helper.rb
| |-nsm_module.rb
| |-pcaplib.rb
| `-pcapparser.rb
|+logs/
|+modules/
|-CHANGELOG
|-files
|-gentags*
|-LGPL.license
|-LICENSE
|-nsm*
|-tags
`-TODO
```

```ruby
## Load the categories in the <dir>
def load_categories(dir)
  if $categories.length > 0
    $categories.each { |cat|
      $tabstrings.delete(cat.get_name())
    }
  end
  $categories = []

  dir = dir + "/categories"

  if !File.directory?(dir)
    puts "Error: Category directory '#{dir}' is not a directory or does not exist."
    puts "No categories will be loaded."
    return
  end

  puts "Loading categories from #{dir}"

  cat_num = 0
  cats = Dir.new(dir)
  ## Load categories
  cats.each { |cat_name|
    if cat_name !~ /\./
      if File.directory?(cat_name)
        puts $color ? "#{$RED}#{cat_name} is a directory, skipping...#{$RESET}" : "#{cat_name} is a directory, skipping..."
        next
      end
      print $color ? "Loading #{$BROWN}#{cat_name}#{$RESET}..." : "Loading #{cat_name}..."
      c = NSM_Category.new(cat_name,get_module_names_from_category(cat_name))
      ## Add to tab completion
      $categories.push(c)
      $tabstrings.push(c.get_name())
      puts "done."
      cat_num += 1
    end
  }
  puts $color ? "\n#{cat_num} #{$BROWN}categories#{$RESET} loaded.\n\n" : "\n#{cat_num} categories loaded.\n\n"
end

## Read from a directory reqursively
## 'dir' is the directory name
## 'rec' is a boolean, whether to recurse or not
def get_filelist(dir,rec)
  filelist = []
  d = Dir.new(dir)
  d.each { |name|
    ## Ignore '.', '..' and 'CVS'
    if name !~ /^\./ and name != "CVS"
      if File.file?(dir + "/" + name)
        filelist << dir + "/" +name
      elsif File.directory?(dir + "/" + name)
        if rec
          filelist.concat(get_filelist(dir + "/" + name,rec))
        else
          puts "Skipping directory " + name
        end
      else
        puts "Skipping " + dir + "/" + name
      end
```

```
" Press <F1> to display help

-commands.rb (/Users/hinmanm

nsm_helper.rb (/Users/hinman
  method
    get_description
    get_defaults
    get_commands
    get_mod_by_name
    get_module_names_from_ca
    print_banner
    print_welcome_message
    load_modules
    load_categories
    get_filelist
    read_dotnsmcrc
    toggle_module
    get_uncompressed_filenam
    gen_list_from_pcap
```

```
<ree_ [nerdtree] [-]#3 16/29,1  ~/hex/hex/rawpacket-root/usr/home/analyzt/rp-NSM/nsm-console/lib/nsm_helper.rb [ruby]   #5 166/351,2  <__ [taglist] [-]#4 15/21,5

[ Xanadu ][          0 scsetup  1 mutt  2 vim  3- zsh  (4*vim)  5 zsh  6 zsh  7 gnump3d          ][ 14/09 18:56 ]
```

<h1>PROGRAMMING LANGUAGES</h1>

<h2>FRONT-END WEB DEVELOPMENT</h2>

HTML: STRUCTURE & CONTENT (WHAT'S ON THE PAGE?);

CSS: STYLING (HOW DOES IT LOOK?);

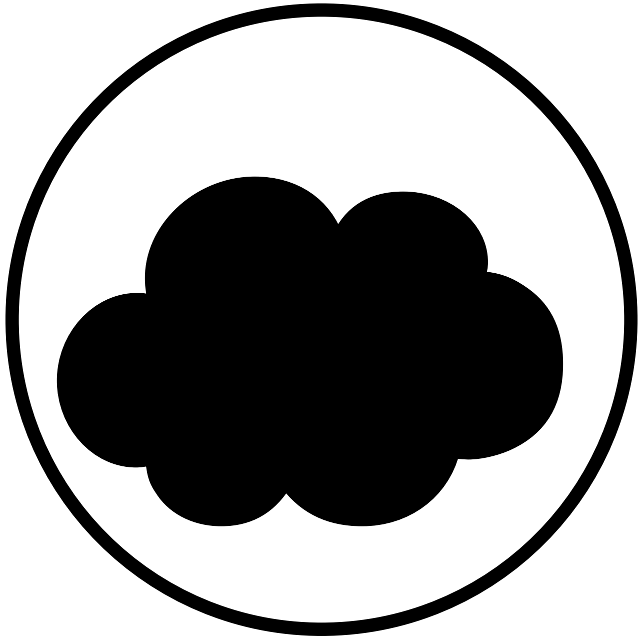
JAVASCRIPT: RESPONSE & USER ACTIVITY (HOW DO YOU INTERACT?);

<h2>BACK-END WEB DEVELOPMENT</h2>

RUBY ON RAILS

PYTHON

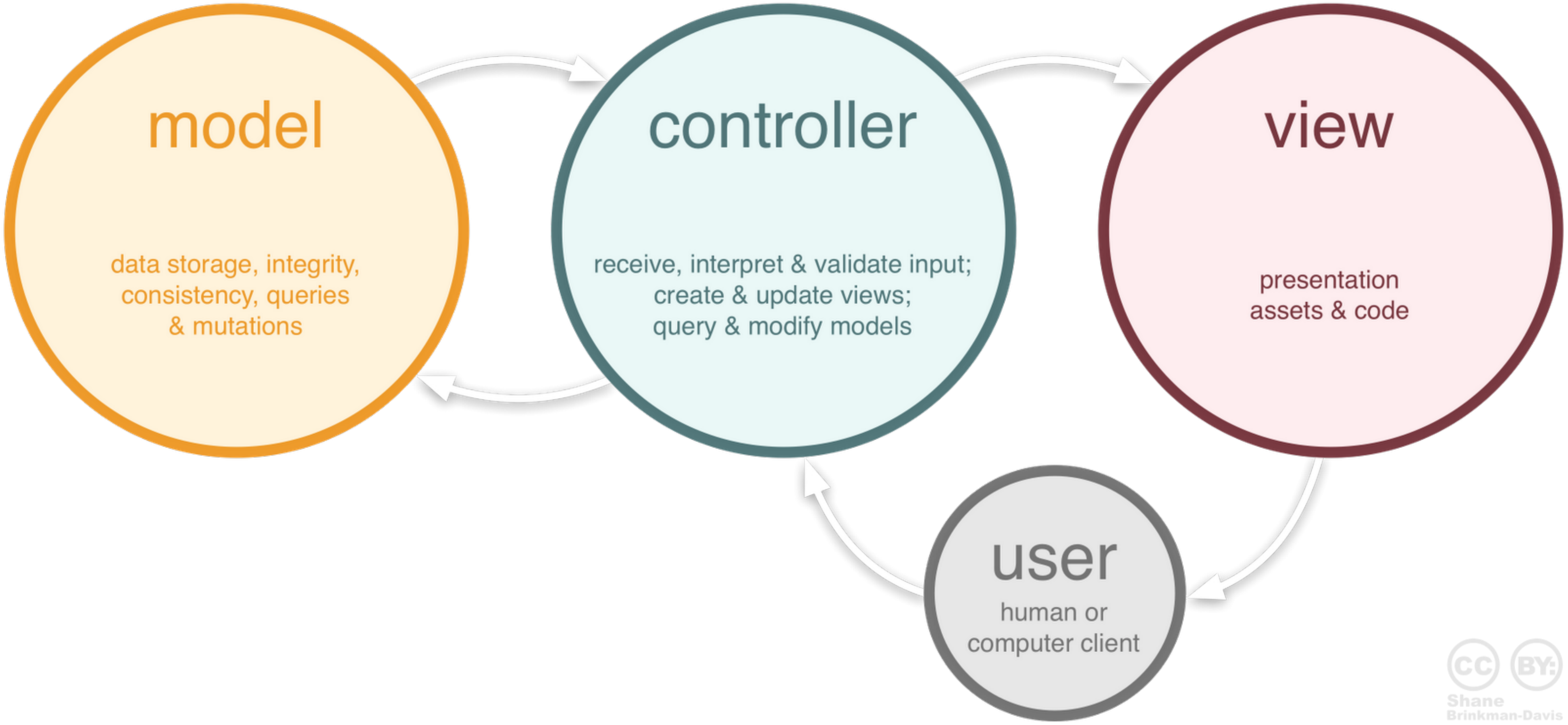
PHP



LET’S TAKE A BREAK.

5 MINUTES

MODEL-VIEW-CONTROLLER


model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client

MVC

MODEL

- CORE BUSINESS LOGIC
- WRITTEN IN RUBY, PYTHON, JAVA, C#, C++, ETC.

VIEW

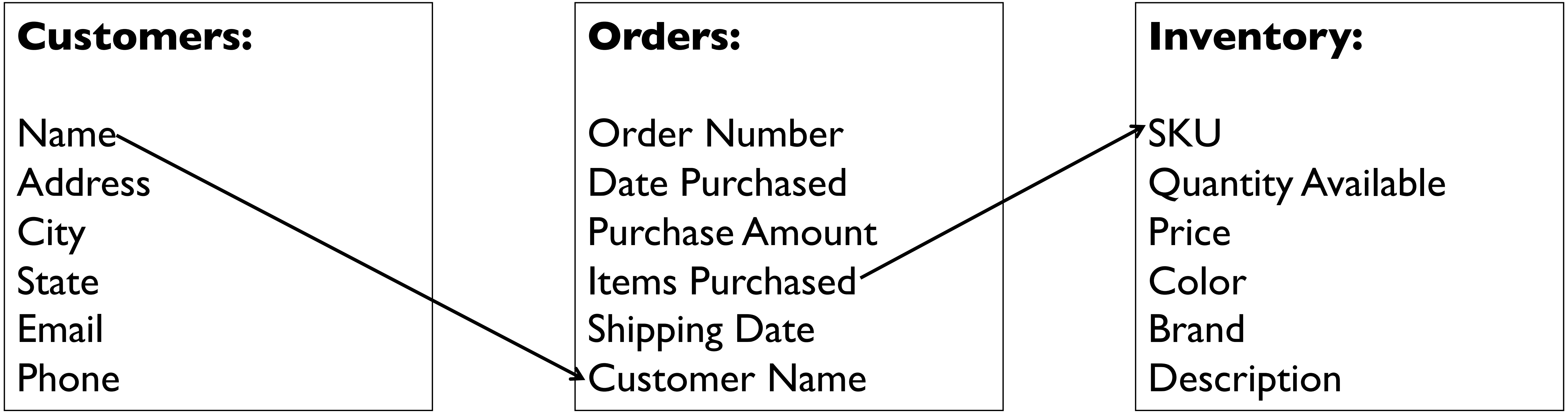
- USER INTERFACE- ALL OF YOUR ASSETS AND STYLE
- HTML, CSS, JAVASCRIPT, ETC.

CONTROLLER

- CONNECTS THE “FRONT END” AND “BACKEND”
- SAME LANGUAGE AS MODEL

HOW DOES A DATABASE ACTUALLY WORK?

What data does an e-commerce site need?



Customers:

Name
Address
City
State
Email
Phone

Orders:

Order Number
Date Purchased
Purchase Amount
Items Purchased
Shipping Date
Customer Name

Inventory:

SKU
Quantity Available
Price
Color
Brand
Description

CRUD

CREATE

ADDS A NEW RECORD IN THE DATABASE

READ

RETRIEVES A RECORD FROM THE DATABASE

UPDATE

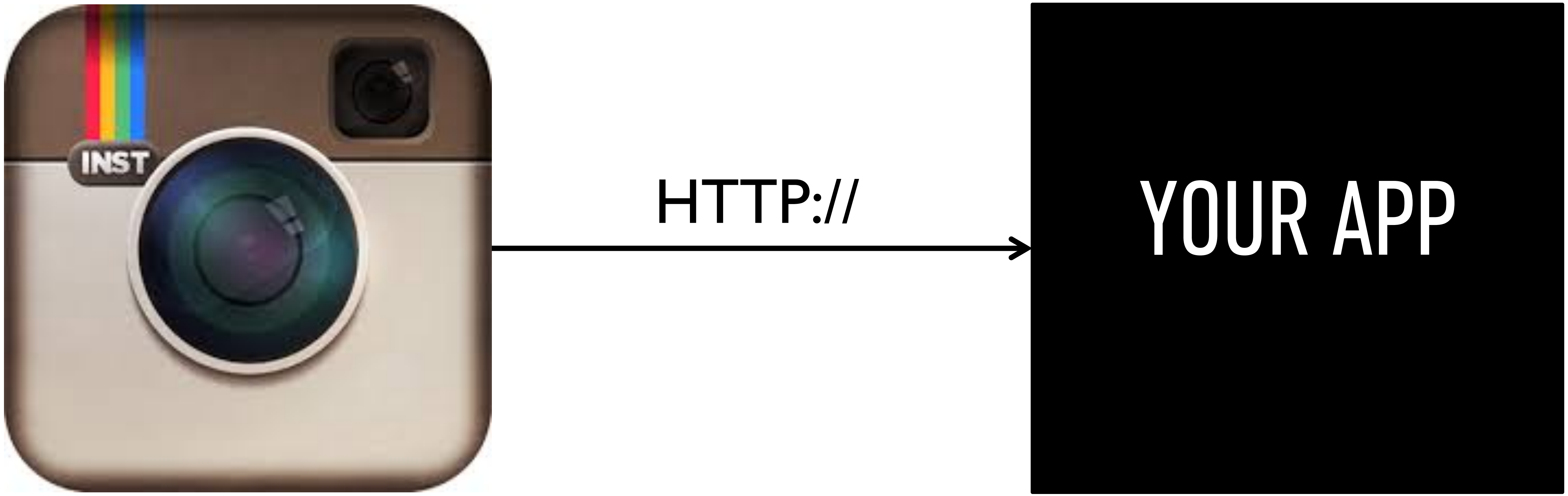
CHANGES A RECORD IN THE DATABASE

DELETE

DELETES A RECORD IN THE DATABASE

WHAT IS AN API ANYWAY?

APPLICATION PROGRAMMING INTERFACE



HTTP://

YOUR APP

HTTP Verbs – CRUD EQUIVALENT

POST

ADDS THE ENCLOSED DATA TO AN EXISTING RESOURCE

PUBLISH THIS PHOTO TO THE USERS ACCOUNT

GET

RETRIEVES A RESOURCE

GET ALL OF THE PHOTOS FOR THE SPECIFIED USER

PUT

UPDATES THE SPECIFIED RESOURCE

CHANGE THE SPECIFIED PHOTO FOR THIS USER

DELETE

DELETES THE SPECIFIED RESOURCE

DELETES THE SPECIFIED PHOTO

Development Stacks

LAMP

- LINUX (OPERATING SYSTEM)
- APACHE (WEB SERVER)
- MYSQL (DATABASE)
- PHP/ PYTHON (SCRIPTING LANGUAGE)

MEAN

- MONGODB
- EXPRESS.JS
- ANGULAR.JS
- NODE.JS

SO WHAT DO YOU DO WITH ALL OF THIS INFORMATION?

WORK WITH YOUR DEVELOPERS!

WORKING WITH DEVELOPERS

- Should be brought in as early in the project as possible & have a sense of ownership.
- Present the problem you' re trying to solve & solicit possible solutions.
- As project progresses, develop requirements that the solution needs to address, but avoid prescribing how to do it.
- Understand & remember that the developer is the expert in this area.

BUILD

Do we have the in-house capability and expertise?

Is this part of our distinct competitive advantage?

Is the solution proprietary?

Can we maintain and improve this long term?

Is there no existing solution that satisfies our needs?

Can we build this cheaper or faster than an alternative?

BUY

Is there a known and trusted solution in the market?

Does this NOT play to our strengths our use our core capabilities?

Will building this distract from our core focus or prevent us from delivering more meaningful features?

Will we get to market significantly faster or less expensive?

Is it too risky to build?

PARTNER

Is there a trustworthy partner in the market?

Does their technology integrate with ours?

Does the partnership constitute a proprietary advantage?

Are we at high risk is the agreement is dissolved or the partner goes out of the business?

Are we good at technical partnerships?

BUILD or BUY/PARTNER

BUILD	BUY/PARTNER
Payments	Square
Maps/Location	Google Maps
CRM	Salesforce
Business Data	Foursquare
A/B Testing	Optimzely

TECH AMA TIME