

---

**PFNP: THE BASICS**

---

 **GENERAL ASSEMBLY**

# **PROGRAMMING FOR NON-PROGRAMMERS**

**Joe Bliss**

**Front-End Web Developer**

# **AGENDA**

Intros / Icebreaker

What is Programming?

Thinking Programmatically

The Web Development Process

How the Web Works

HTML, CSS, JavaScript Primers

Code Examples

Personal Website



# JOE BLISS

Joe Bliss is a Freelance Front-End Web Developer who has worked with Bloomberg, American Express, Sports Illustrated. His expertise is in HTML, CSS, and Javascript, and has been building websites since Geocities weren't just ironic.

He has taught many courses at GA, where he is currently working as a Front-End Assessment Editor.

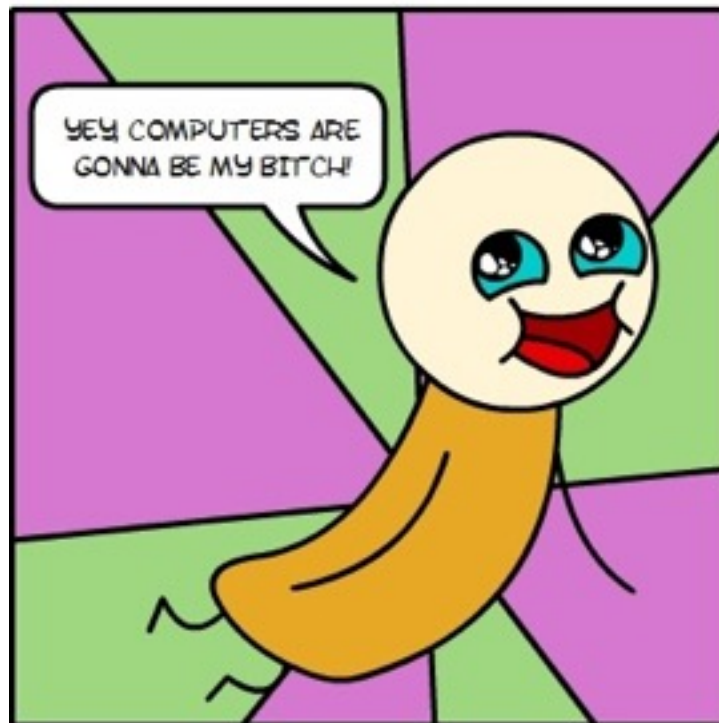
# ICEBREAKER

- Your name.
- What you do.
- Your goal in taking this class.
- Your “Jam” song.



# When I decided to take Computer Science as a second major

## PRECONCEPTIONS



## REALITY



# **THIS STUFF IS HARD**

My goal is to give you a baseline of understanding to empower you and allow you to communicate better with your team.

And maybe (just maybe) pique your interest to go on to learn more!

# **PARKING LOT**



# **WHAT IS PROGRAMMING?**



# **WHAT IS PROGRAMMING?**

In your own words.

# **WHAT IS A PROGRAM?**

You've probably already written a program before without knowing it! If you've ever written a formula or macro in Excel, you've written a program!

A program is a set of instructions that a person writes to tell a computer how to carry-out a task.

Programming is the task of writing those instructions.

# **WHAT IS A COMPUTER?**

"An electronic device for storing and processing data [...] according to instructions given to it [...]" (Wikipedia.org)

- Laptop, Desktop
- Phone, Tablet
- Watch, Fitbit, Calculator
- Thermostat, calculator, microwave, smart toaster, etc. etc. etc.



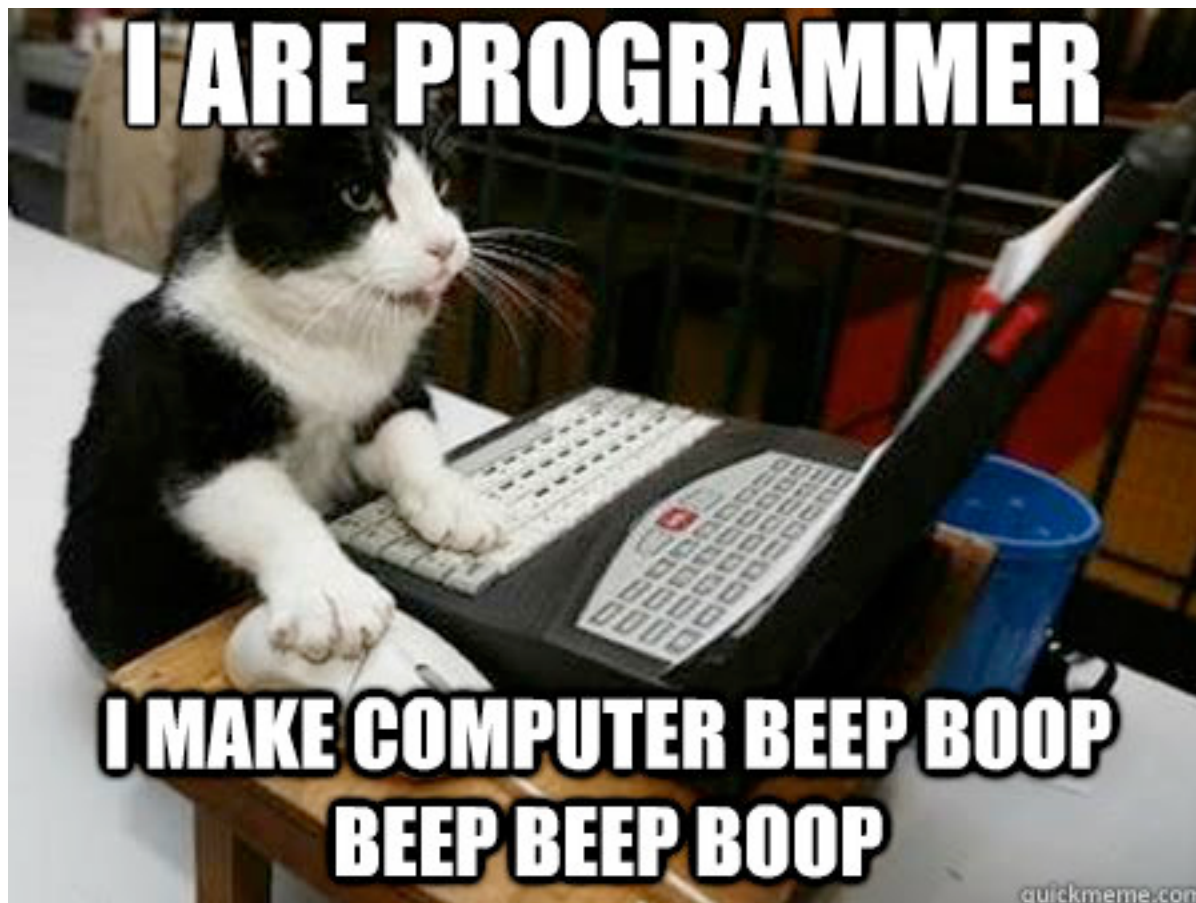
# **BECOMING A PROGRAMMER**

So, pretty much any electronic device is a computer these days. To get these computers to do what we want, we give them sets of instructions in languages the computer understands. I.e. We program them.

To program them, though, we must first learn to think like the computer thinks.

# **THINKING PROGRAMMATICALLY**

# **THINK LIKE A COMPUTER**



# **CLOSE YOUR LAPTOPS**

Seriously.



# **LAB TIME**

## **WHAT DID WE LEARN?**

You have to be speaking the same language.

You have to know what's pre-defined in the language.

Steps execute sequentially.

Steps must be small, granular.

The computer will do **ONLY** and **EXACTLY** what you tell it to do.

# **WHAT DID WE LEARN?**

When we are working on a computer, though, these commands aren't written-out in English. They are written in a Programming Language. Programming languages take many different forms.

# NAME A FEW PROGRAMMING LANGUAGES



# **TO NAME A FEW ...**

RUBY RUBY ON RAILS PHP JAVA JAVASCRIPT HTML CSS  
C++ C# OBJECTIVE C PYTHON C JQUERY NODE  
BACKBONE ANGULAR EMBER R DJANGO SINATRA  
PADRINO SCALA ERLANG HASKELL ASSEMBLY PERL SQL  
FORTRAN PASCAL PROCESSING SCRATCH HEROKU  
MONGO-DB MYSQL SMALLTALK LISP J2EE XSLT OCTAVE

# **WHY ARE THERE SO MANY?**

Because they all do different things!

Interactive, Rich Websites -> HTML, CSS, JavaScript

Enterprise applications -> Java

Mobile Apps -> Objective-C, Swift, Java

Web Apps -> PHP, Ruby (on Rails)

Data Science -> Python

# **SO HOW DO YOU CHOOSE?**

It depends on a number of factors:

- What you are trying to make.
- What languages you already know or can easily get up-to-speed on.
- The current “Technology Stack” at your company.
- Third-party integrations.
- Security concerns.
- Open-source.

# **SO HOW DO YOU CHOOSE?**

“You wouldn’t spend months brushing up on Mandarin before a trip to Germany”

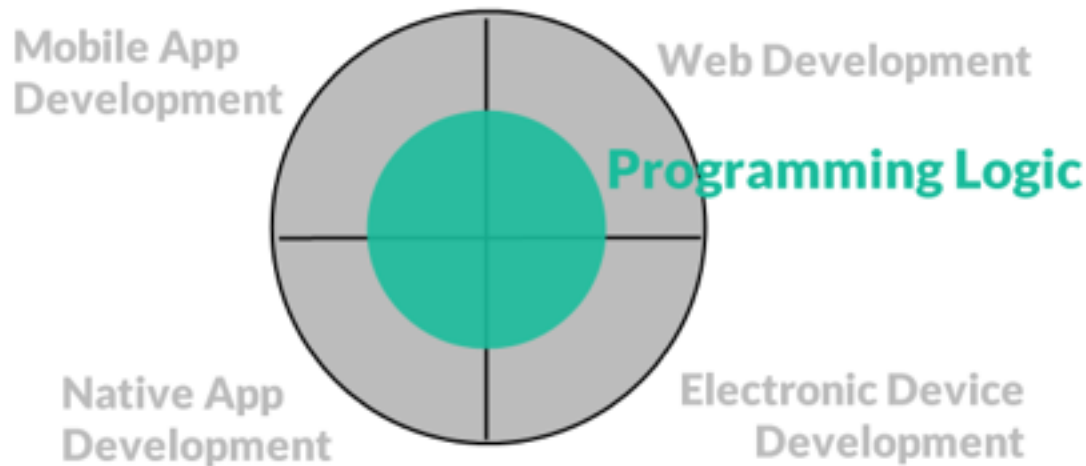
A helpful (and not totally SWF) guide:

<http://www.wfplsiu.com/>



# PROGRAMMING LOGIC

At the core of each of these, though, is the same Programming Logic - Data Types, Data Structures, Variables, Conditionals, Loops, Functions.



# **PROGRAMMING LOGIC**

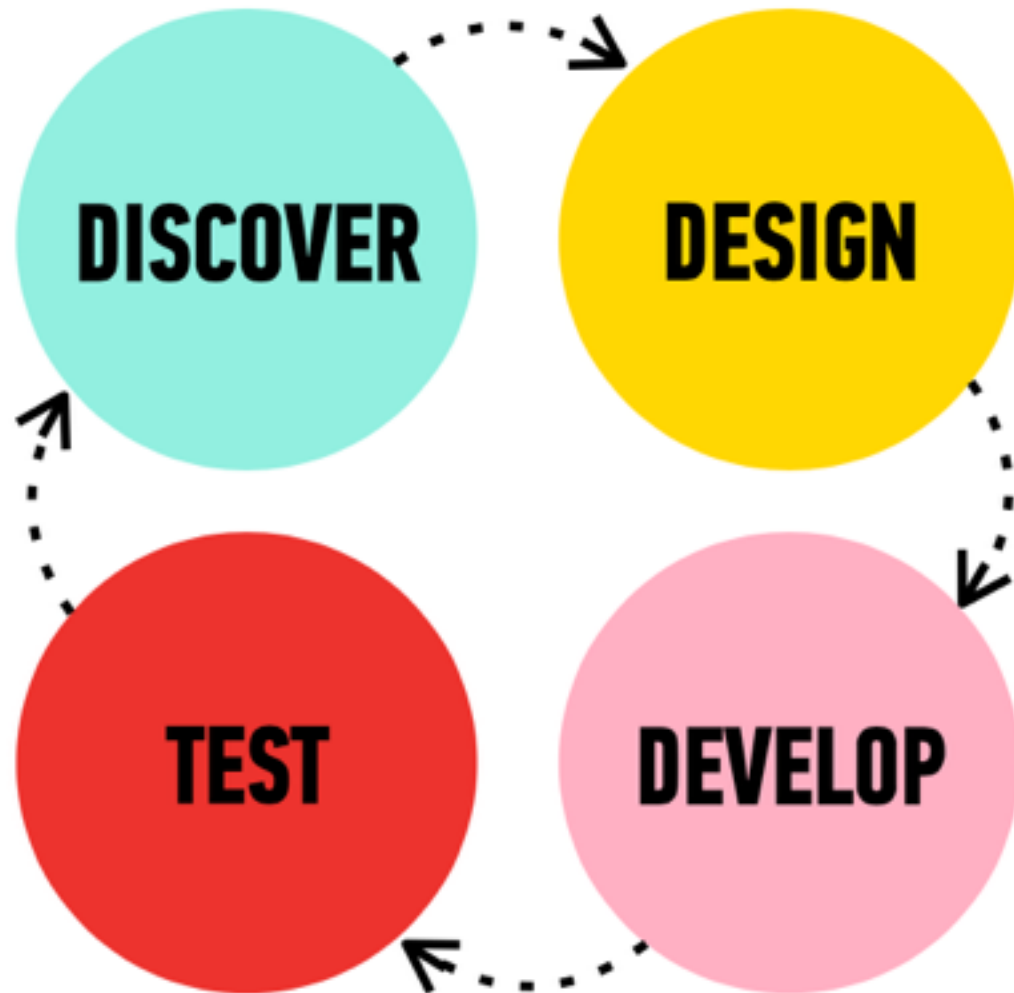
Play the following game based on FROZEN:  
<http://studio.code.org/s/frozen/stage/1/puzzle/1>



# **LAB TIME**

Code got you down? Let it go!

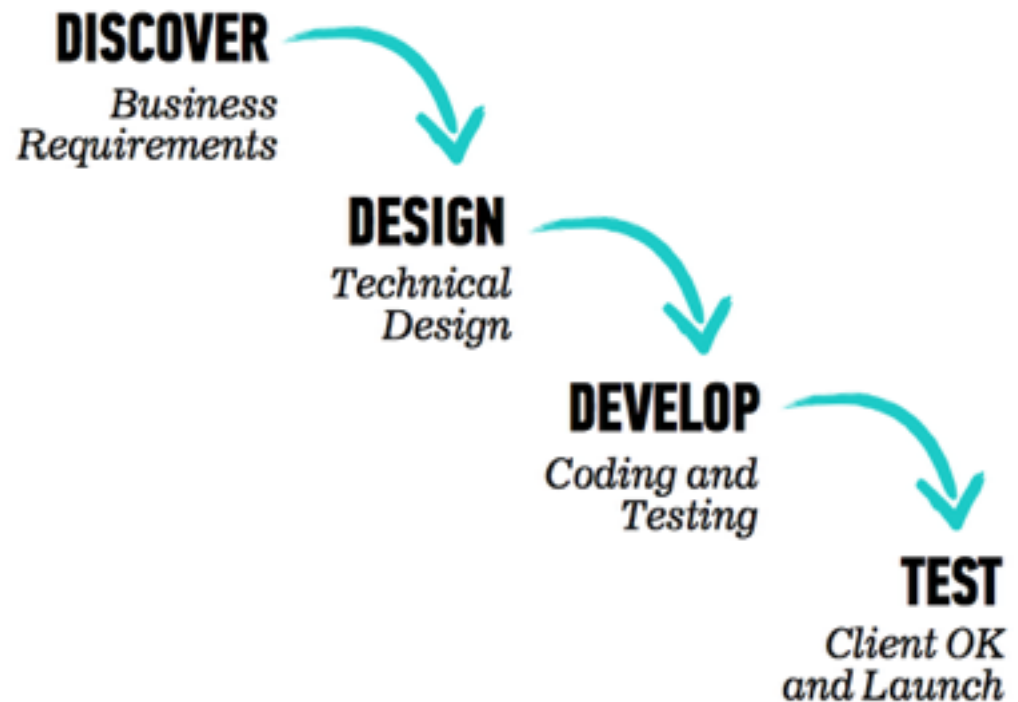
# **THE WEB DEVELOPMENT PROCESS**



# THE WATERFALL APPROACH



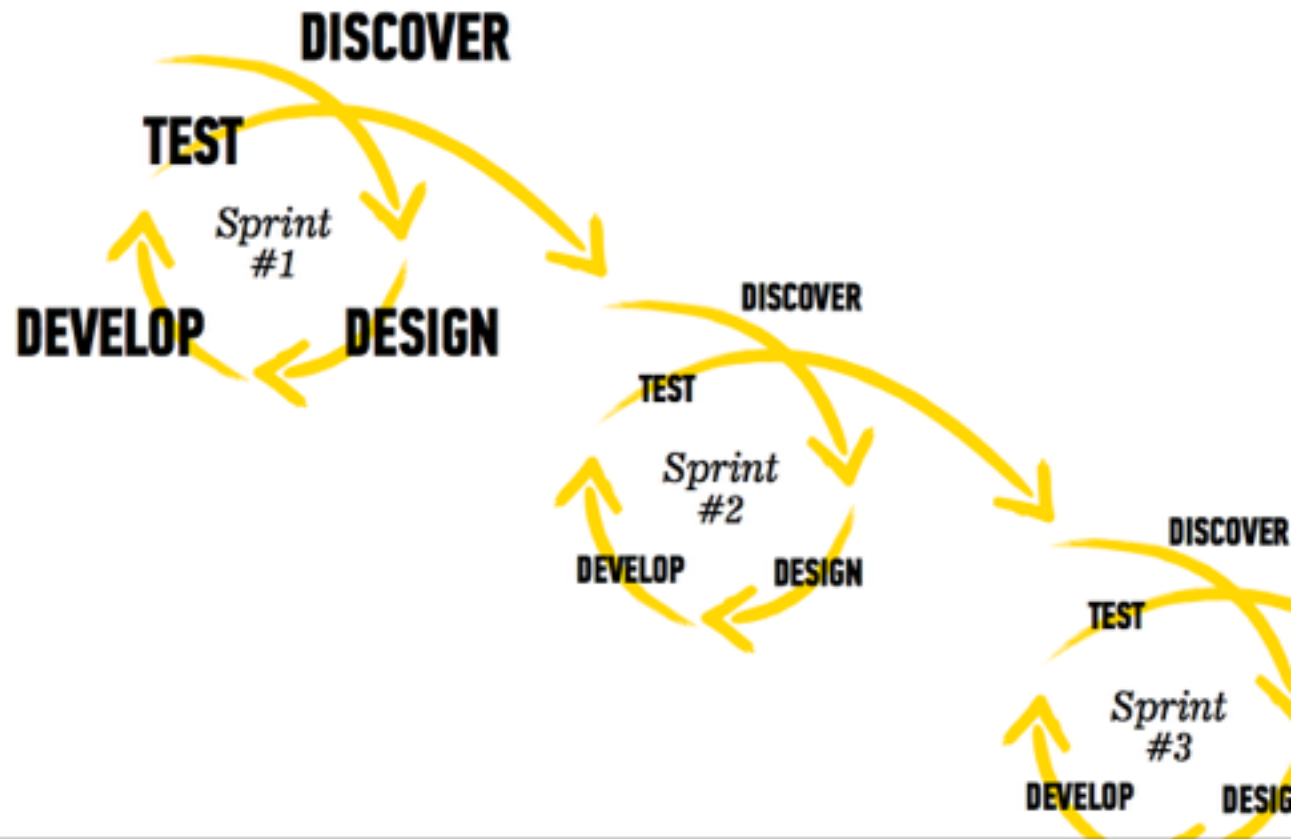
**WATERFALL**  
*Sequential*



# THE AGILE APPROACH



**AGILE**  
*Iterative*



# A COMPARISON



- › One chance to get the product right
- › Less opportunity for feedback
- › Less communication between different groups
- › Following a plan
- › Creates needless complexity



- › Many chances to get the product right
- › More opportunity for feedback
- › More communication between different groups
- › Responding to change
- › Complexity presents itself early



# **THERE ARE LOTS OF PEOPLE INVOLVED IN THE PROCESS...**



**Business  
Managers**



**Product  
Managers**



**Designers**



**Marketers**



**Programmers**



**Users**

# **COMMUNICATION EXERCISE**

# **LAB TIME**

# **TAKEAWAYS**

Requirements are extremely important.

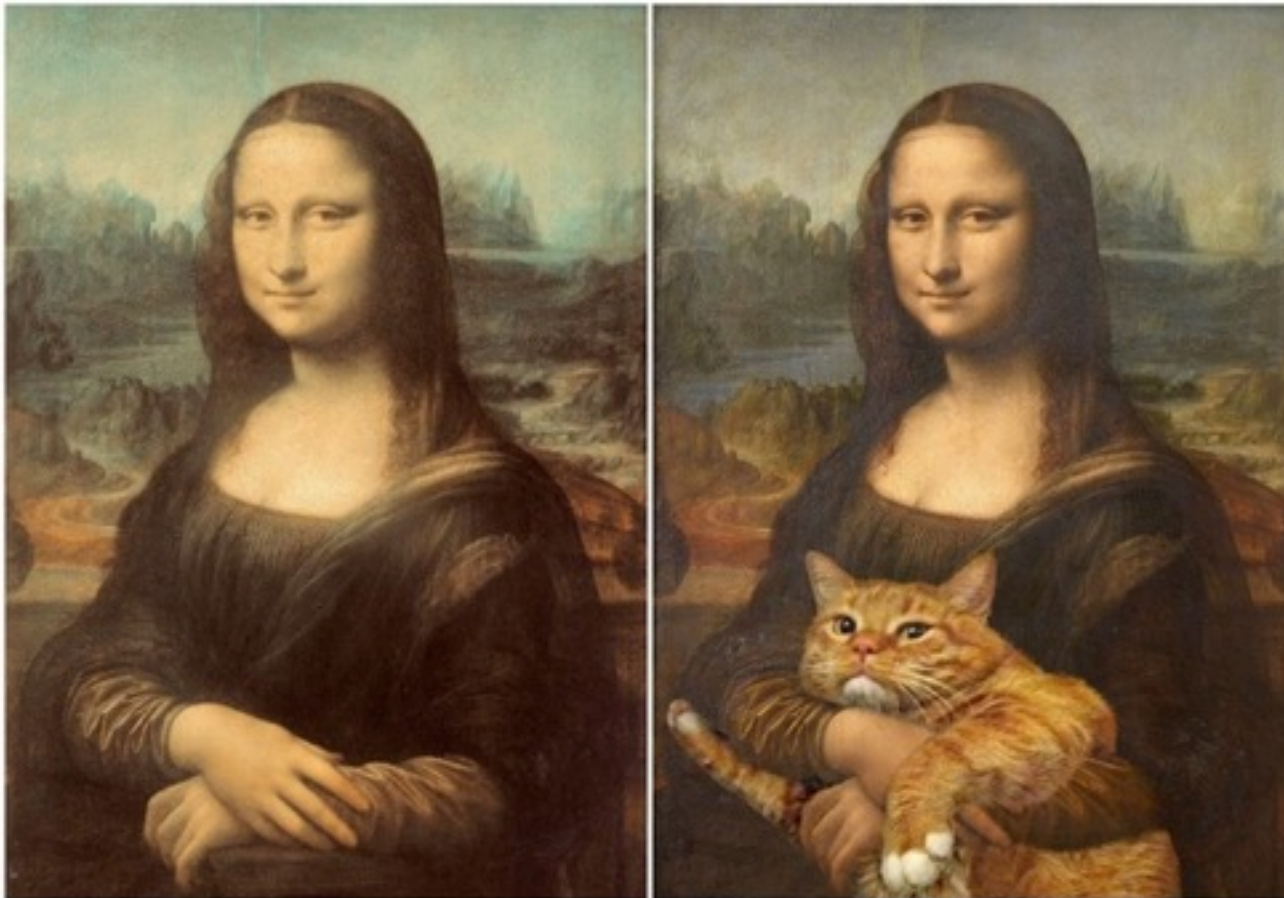
Documentation is extremely important.

Before you start Design / Development, it is important that everyone is on the same page.

# **HOW THE WEB WORKS**

## HOW THE INTERNET WORKS

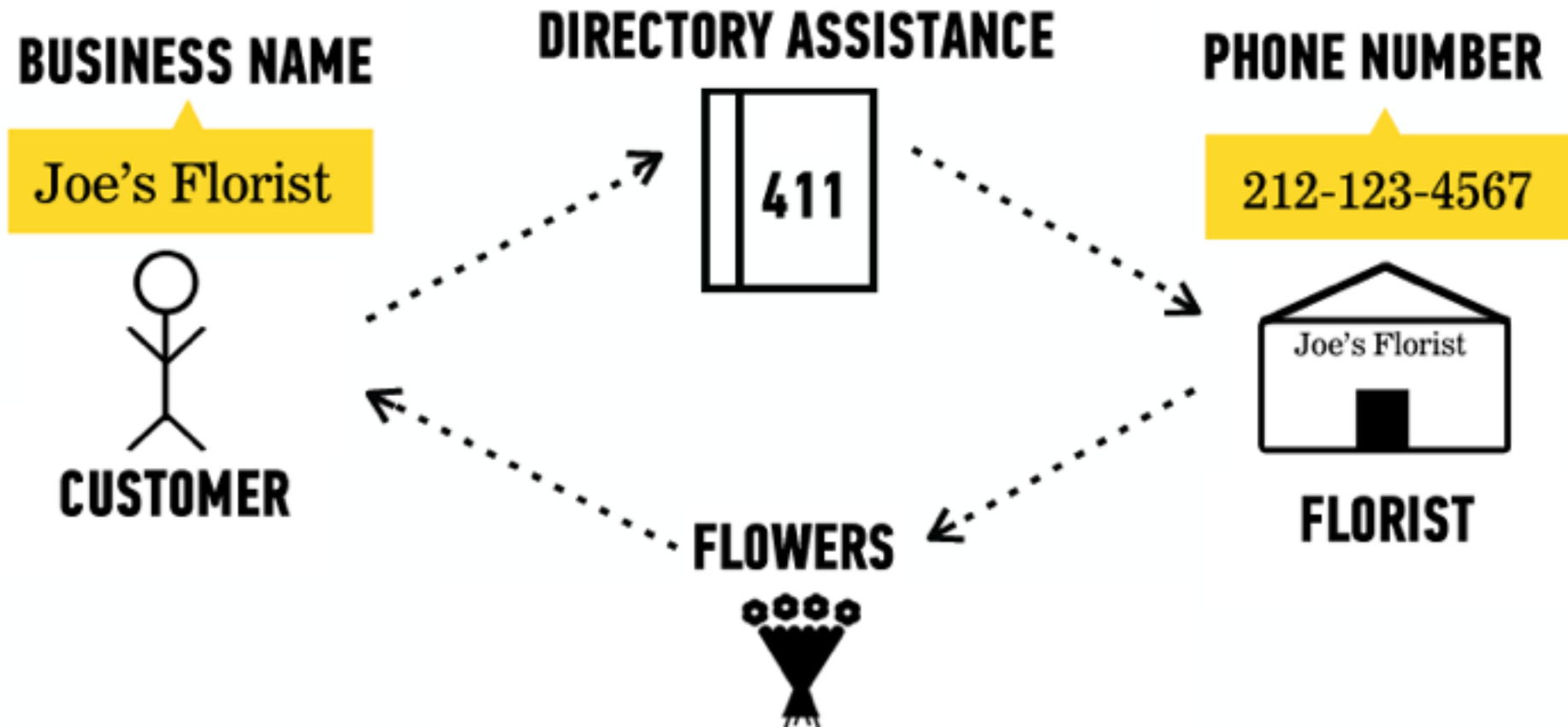
EATLIVER.COM



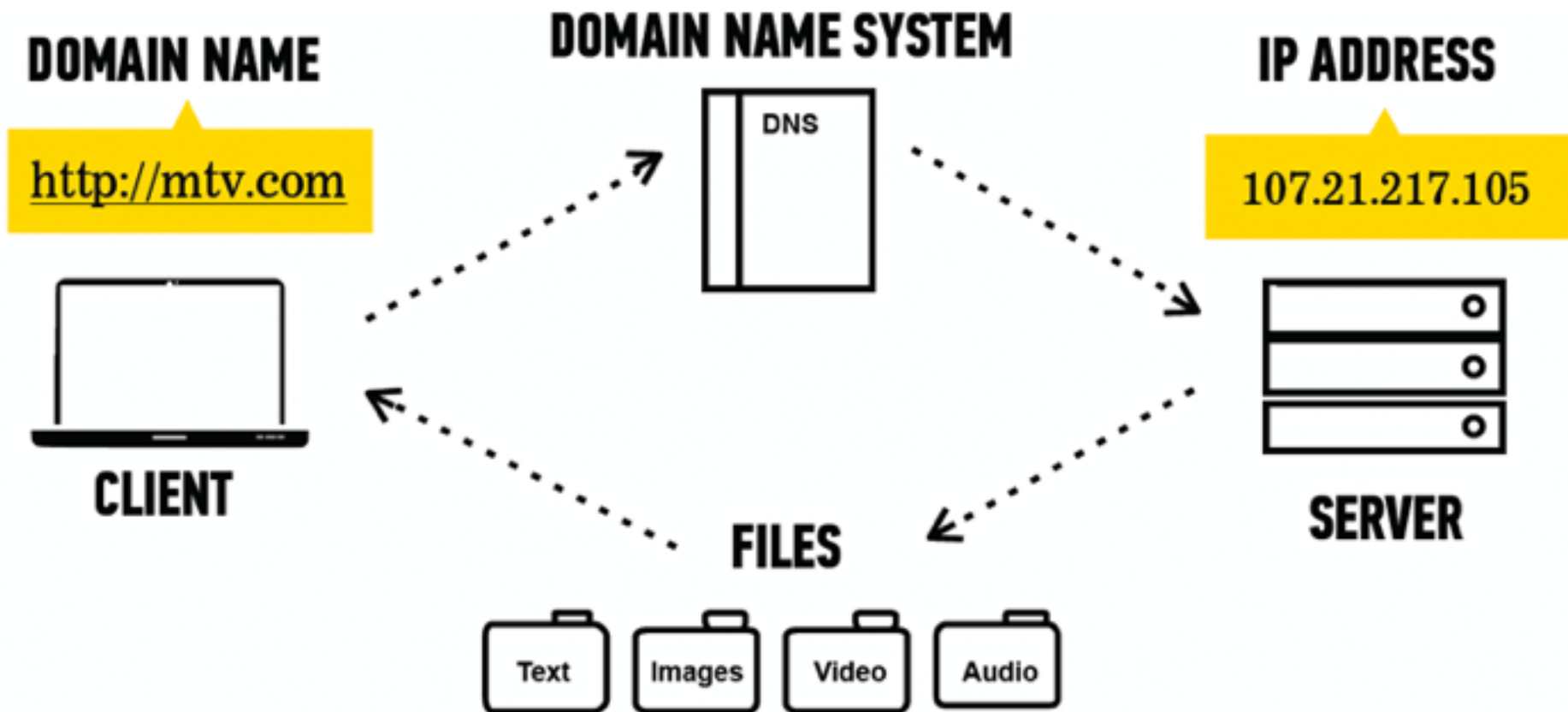
Like 0 Tweet 2 +1 0

Like 446k Tweet 5,763 +1 184

# HOW THE WEB WORKS: AN ANALOGY



# HOW THE WEB WORKS





## WHAT GETS SENT BACK?

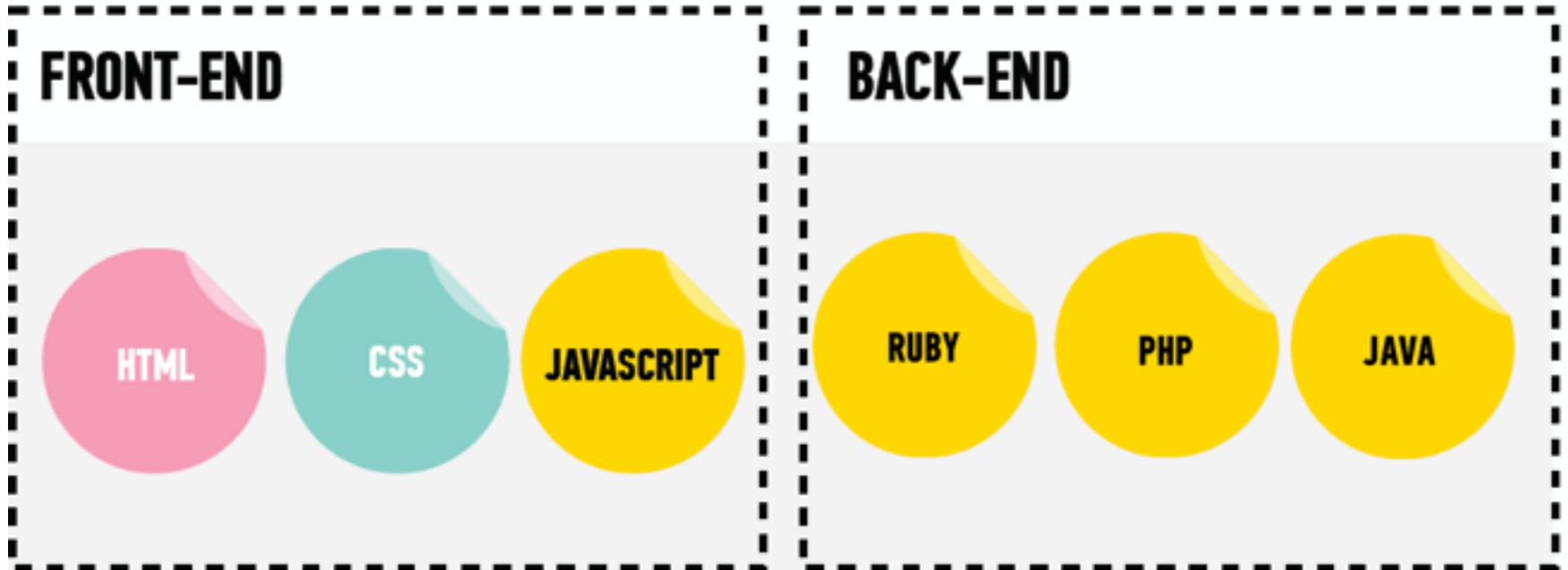


**HTML = CONTENT / STRUCTURE**

**CSS = STYLE / DESIGN**

**JAVASCRIPT = BEHAVIOR / INTERACTION**

# **FRONT-END VERSUS BACK-END**



# **HTML PRIMER**

# **I'M GONNA POP SOME TAGS ...**



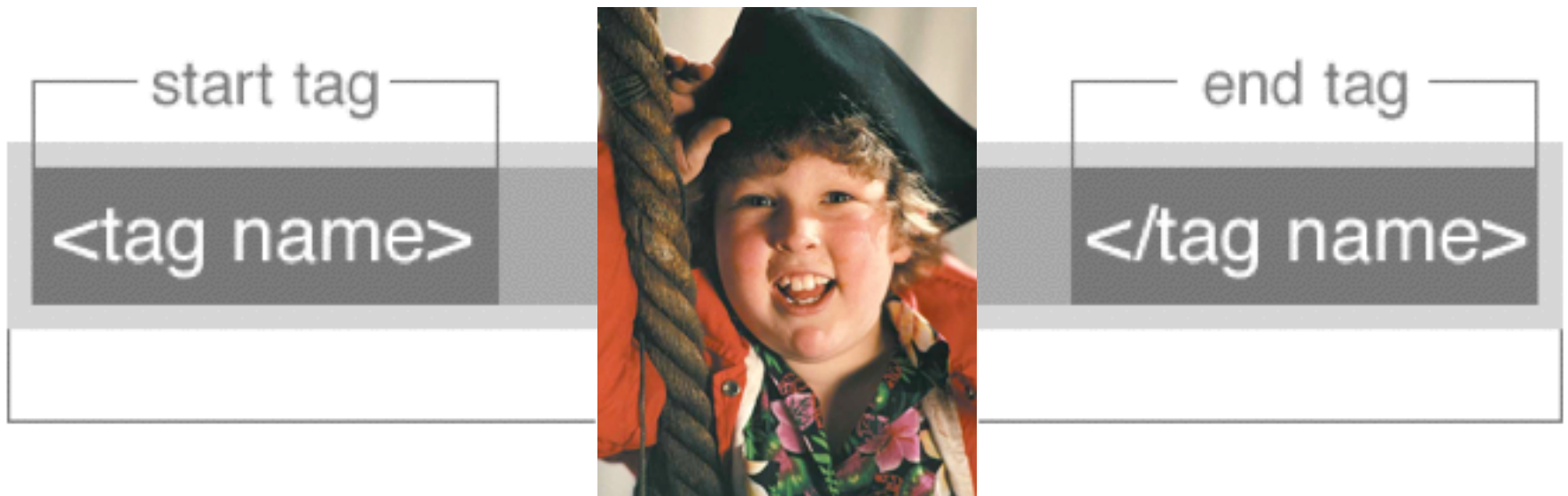
# **I'M GONNA POP SOME TAGS ...**

We use code called “tags” to group content into different chunks.

`<p>Hello!</p>`

There generally is an opening tag “`<sometag>`” and a closing tag “`</sometag>`” wrapping the chunk of content.

# GROUP “CHUNK”S OF CONTENT



# HEADING ELEMENTS

## Heading Elements

`<h1>Largest Heading</h1>`

`<h2> ... </h2>`

`<h3> ... </h3>`

`<h4> ...</h4>`

`<h5> ... </h5>`

`<h6>Smallest Heading</h6>`



# HEADING ELEMENTS

Heading tags `<h1>` through `<h6>` are meant to be used for text that you want to appear as a title or headline on your webpage. Think of the way that headlines look in a newspaper or an online news website.

# **PARA-NORMAL ACTIVITY**

`<p>`This is a paragraph.`</p>`

One of our bread-and-butter tags. The `<p>` tag gives us paragraphs of wrapping content.

Think of it as a paragraph of content in a book, article or word processing program.

# **LISTS – UNORDERED LIST**

**<ul>**

**<li>First Item</li>**

**<li>Second Item</li>**

**<li>Third Item</li>**

**</ul>**

## **OMG! <IMG>**

Images are placed using the `<img>` tag.

The `<img>` doesn't wrap content, so it doesn't close!

The `img` tag requires a `src` attribute, which tells the browser where to find the image to be placed.

```

```

## **<IMG> CONTINUED**

They can be linked relative to a local file or to an absolute address on the internet:

`` - Relative

`` - Absolute


# **CSS PRIMER**

# CSS SYNTAX

```
p { color: black; }
```

This CSS statement turns the text of every paragraph - `<p>` - on our page black.

# CSS SYNTAX



The diagram illustrates the components of a CSS rule. The text `p { color: black; }` is shown. Above the `p` is a red bracket labeled *selector*. Above the `color` is a red bracket labeled *property*. Above the `black` is a red bracket labeled *value*. A large red bracket below the `color: black;` is labeled *declaration*.

We refer to what we are changing as a “selector”, what we are changing about it as the “property”, and what we are changing that property to as the “value”.  
property: value; pairs are referred to as a “declaration”.



# CSS SYNTAX

The diagram illustrates the components of a CSS rule using the example `p { color: black; }`. Red curly braces are used to group parts of the code, with red italicized labels above or below them. A brace above `p` is labeled *selector*. A brace above `color` is labeled *property*. A brace above `black` is labeled *value*. A brace below `color: black;` is labeled *declaration*. A large brace below the entire `p { color: black; }` is labeled *rule*.

```
p { color: black; }
```

The entire block of CSS is referred-to as a “CSS rule”. Your CSS files will have many CSS rules.

# CSS SYNTAX

*rule* {  
    p {  
        color: black; } *declaration*  
        font-weight: bold; } *declaration*  
    }

One selector can have multiple declarations (that is, property: value; pairs). This is still referred-to as a rule. It's common for each declaration to be on its own line.

# CSS PROPERTIES

[http://www.w3schools.com/cssref/css\\_colors.asp](http://www.w3schools.com/cssref/css_colors.asp)

- Colors

[http://www.w3schools.com/css/css\\_font.asp](http://www.w3schools.com/css/css_font.asp)

- Fonts

Every CSS Property you (n)ever wanted to know:

<https://css-tricks.com/almanac/>

# **CODE CHALLENGE: PERSONAL WEBSITE**

# **CREATE A WEBSITE FOR YOURSELF WITH:**

Your name

Your title

A photo

A blurb about yourself

A list of your interests

# **JAVASCRIPT PRIMER**

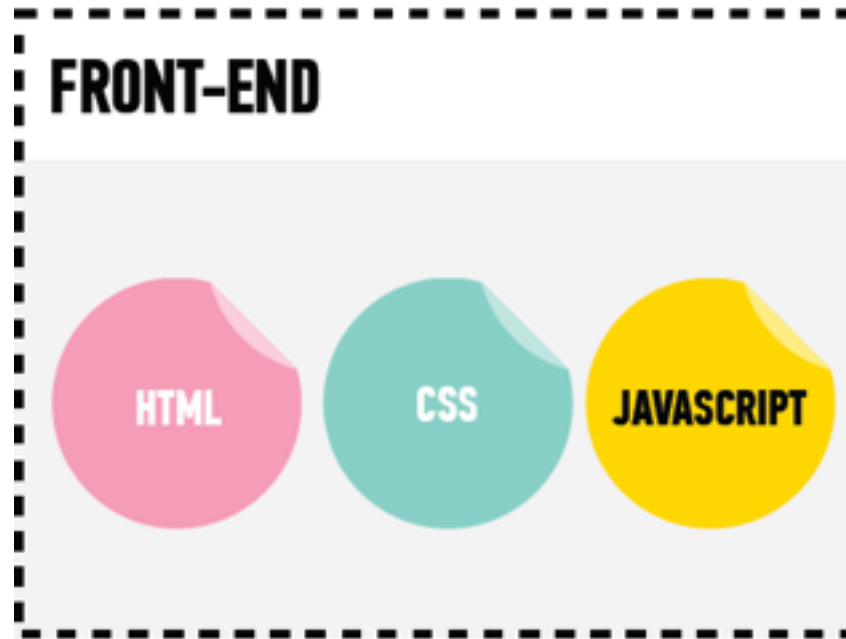
# **HTML AND CSS AREN'T PROGRAMMING LANGUAGES!**

# **SURPRISE!**





# **WHERE DOES JAVASCRIPT FIT IN?**



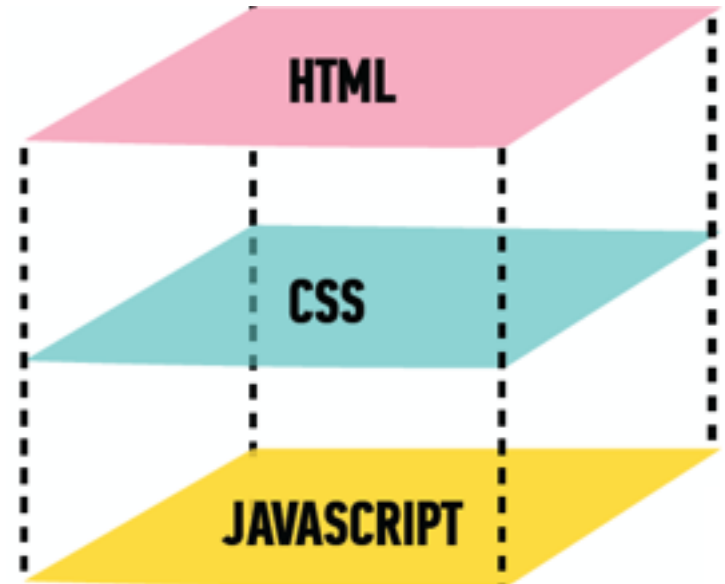
# HOW IT DIFFERS FROM HTML AND CSS

HTML and CSS are used to define the initial state of our website.

JS is used to define how this state changes.

HTML and CSS are static.

JS is dynamic.



# **WHAT CAN WE DO WITH JAVASCRIPT?**

Adding / Removing Elements

Changing CSS “on-the-fly”

Animating content

Detecting user interactions

Form validation

Loading dynamic content

Etc.

# SYNTAX

Syntax: Spelling and grammar rules of a programming language.

Like with any language, there are formal rules around how to write it. This is the syntax.



## **(SOME) JAVASCRIPT SYNTAX**

JavaScript statements end in semicolons: “;”

JavaScript is case-sensitive. Variables, function names, etc. must be consistent. `joeBliss()`; is not the same as `joebliss()`;

Javascript uses various keywords (i.e. `function`, `if`, `else`, `for`, `while`) or symbols (i.e. `()`, `{ }`, `[ ]`) to demarcate control flow.

# **CODEALONG – OUR FIRST JAVASCRIPT**

Open up <http://codepen.io/pen/>

Type into the JS panel:

```
document.write("Hello, World!");
```

# **TYPES OF DATA – NUMBERS**

Integers

1, 2, 3, 4, 5

Floats (numbers with decimal points)

3.14159, 2.718281828459045

Can be Signed or Unsigned (- or +)

6, -8.2

We can perform arithmetic on number data types

# **TYPES OF DATA – STRINGS**

## **Strings**

- A sequence of characters enclosed in quotes, i.e. “I am a String”, “Hello!”, “Joe Bliss”
- Stores textual information
- Can be “double” or ‘single’ quoted



# **SCAVENGER HUNT: TRAFFIC LIGHT**

# SEE IF YOU CAN FIGURE OUT WHAT'S WRONG WITH THIS TRAFFIC LIGHT

And fix it!

<http://codepen.io/ga-joe/pen/ONeJwL>

# **CODEALONG : COLOR SCHEME SWITCHER**

# **LET'S MAKE YOUR COLOR SCHEME DYNAMIC WITH JAVASCRIPT!**



# AMA

*Ask Me Anything*

# THANK YOU!

Joe Bliss

[joe.bliss@generalassemb.ly](mailto:joe.bliss@generalassemb.ly)  
[linkedin.com/in/joebliss](https://www.linkedin.com/in/joebliss)