PSEUDO CODE

START SCEENCAST!!!!

LET'S GET EVERYTHING SET UP!

1. Navigate to the FEWD 42 Dashboard (saraheholden.com/fewd_dashboard/) and download the Lesson 7 slides. You'll want to keep the dashboard open for other links and resources we'll be referencing in class.
2. Log in to the FEWD 42 Slack (fewd42.slack.com) and join the class7 channel.

Note: There is no starter code for this lesson

WEEKLY OVERVIEW

WEEK 4	Intro to Programming / Intro to jQuery
WEEK 5	Intro to JavaScript — Variables, Conditionals, Functions
WEEK 6	JavaScript Continued — Arrays / Lab

GENERAL ASSEMBLY

PSEUDO CODE

Sarah Holden

LEARNING OBJECTIVES

- Practice programmatic thinking by writing pseudo code to solve a basic problem.
- Define web site behavior and the practical uses of JavaScript.
- Predict DOM output / changes by reading JS code.

AGENDA

- Review
- Intro to Programming
- Intro to Pseudo Code
- Intro to JS
- Reading JS
- Wireframes Lab

REVIEW

EXIT TICKET QUESTIONS

- Are there any downsides to using SASS/LESS?
- Are there any best practices for style order? text, box, display, etc?
- Is nesting and using css variables a normal standard for most developers? (SASS)
- How long is the final project supposed to take outside of class?
- Centering images and other things :)

PREPROCESSORS

CSS PRE-PROCESSORS: SASS AND LESS

- Use variables in CSS!
- Nest styles!
- Define mixins (similar to JS functions)
- Mathematical functions
- Operational functions (such as "lighten" and "darken")

$VARIABLES

```
$baseTextColor: #e51b24;

h1 {
    color: $baseTextColor;
}
```

NESTING

```
article {
    margin-bottom: 3em;
    h1 {
        font-size: 2em;
    }
}
```

REVIEW

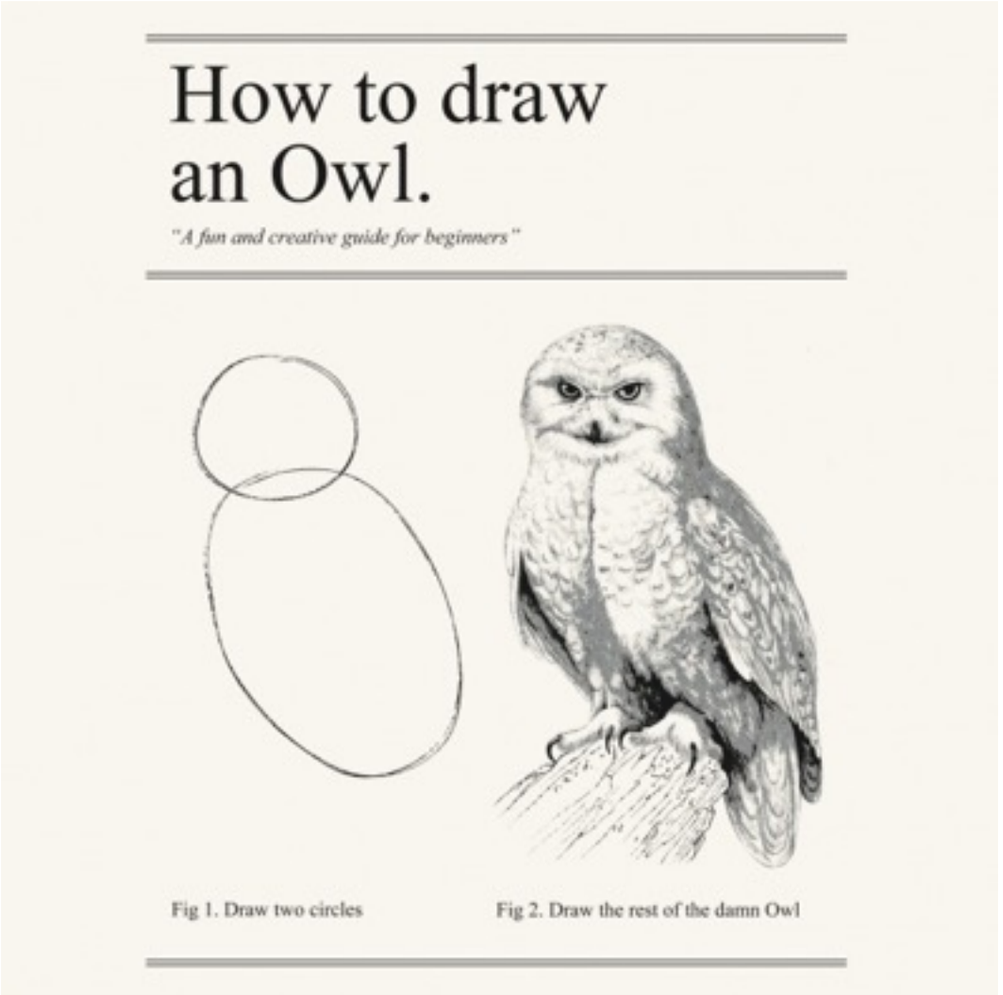
- "Lines" (borders) between sections
- Background-image
- Why use anchor instead of button elements?
- Shadow on bottom of buttons
- Text-transform
- Pixels/percentages/responsive
- Padding vs. margin for info sections
- Creating developer/designer pages

REVIEW — MULTIPLE CLASSES ON ONE ELEMENT

An element can have multiple classes. Multiple classes are separated by a space in the HTML.

```
<section class="banner clearfix"></section>
```

PROCESS


How to draw
an Owl.
"A fun and creative guide for beginners"
Fig 1. Draw two circles
Fig 2. Draw the rest of the damn Owl

PROCESS

1. Write your HTML (use HTML5 Flowchart for guidance in picking elements)
2. Get things into place. (Add floats, clear: both, get columns set up, etc.)
3. Add base styles. (Font-family/color for the body, remove text-decoration under anchors and add a base anchor color, remove bullets for list items, etc.)
4. Work through the page section by section and start "filling in the details." Resist the urge to be a perfectionist at this point.
5. Polish things up! Compare the design with your page. Are you using the right fonts? Colors? Is any of the spacing off?

PROCESS — STEP 2 (LAYOUT STYLES)

Developers | Designers | How it Works | Our Team | Blog

Startup Matchmaker

Because two heads are better than one.

Meet your Match!

Have a great idea for a product, but need help making it a reality? We're here to help. Startup Matchmaker is the best place for designers and developers to find each other.

Create a Profile

Create a Profile	Find a Developer	Find a Designer
Are you a Designer? Put yourself out there so that others can find you!	Looking for a developer to work with on the next big thing? Look no further.	Need someone who can make a product intuitive and appealing? Get ready.
Sign up Now	Start Your Search	Start Your Search

© 2013 Startup Matchmaker. Made in NY.

*Pro tip: adding a border to everything on the page can help during this process:
* {border: 1px solid black; }

PROCESS

1. Write your HTML
2. Get things into place. (Add floats, clear: both, get columns set up, etc.)
3. Add base styles. (Font-family/color for the body, remove text-decoration under anchors and add a base anchor color, remove bullets for list items, etc.)
4. Work through the page section by section and start "filling in the details." Resist the urge to be a perfectionist at this point.
5. Polish things up! Compare the design with your page. Are you using the right fonts? Colors? Is any of the spacing off?

PROCESS — STEP 3 (BASE STYLES)

PROCESS

1. Write your HTML
2. Get things into place. (Add floats, clear: both, get columns set up, etc.)
3. Add base styles. (Font-family/color for the body, remove text-decoration under anchors and add a base anchor color, remove bullets for list items, etc.)
4. Work through the page section by section and start "filling in the details." Resist the urge to be a perfectionist at this point.
5. Polish things up! Compare the design with your page. Are you using the right fonts? Colors? Is any of the spacing off?

INTRO TO PROGRAMMING

PROGRAMMING

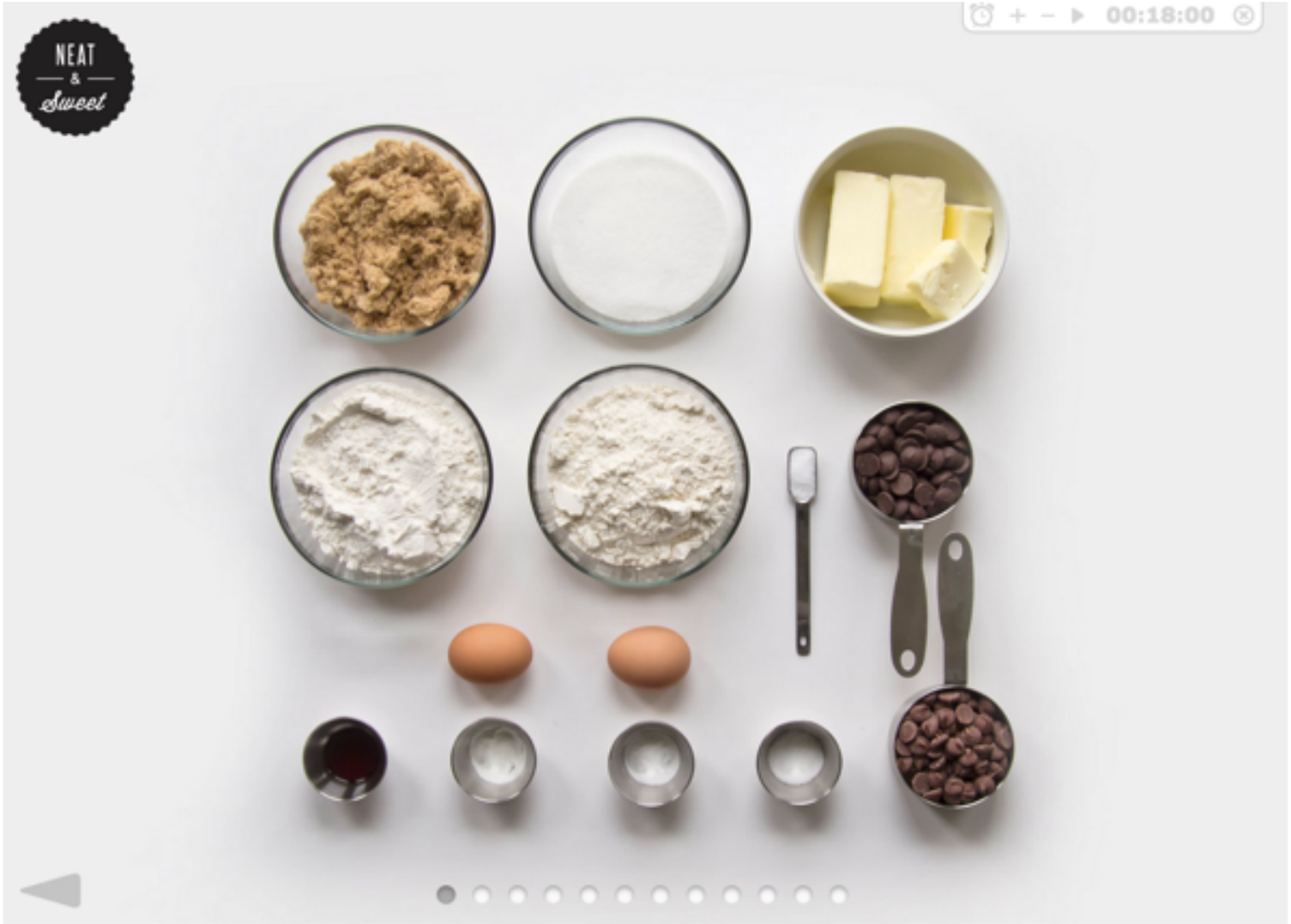
WHAT IS A PROGRAM?

- A program is a set of instructions that you write to tell a computer what to do

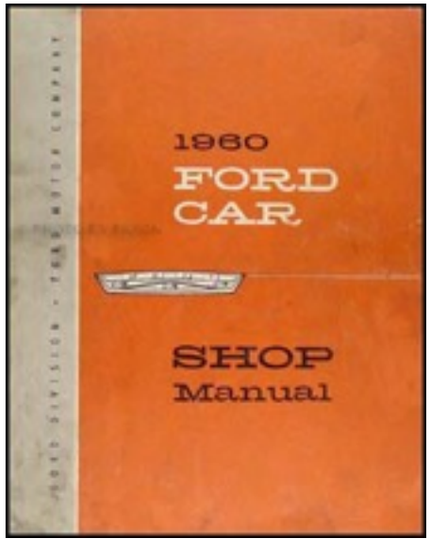
WHAT IS PROGRAMMING?

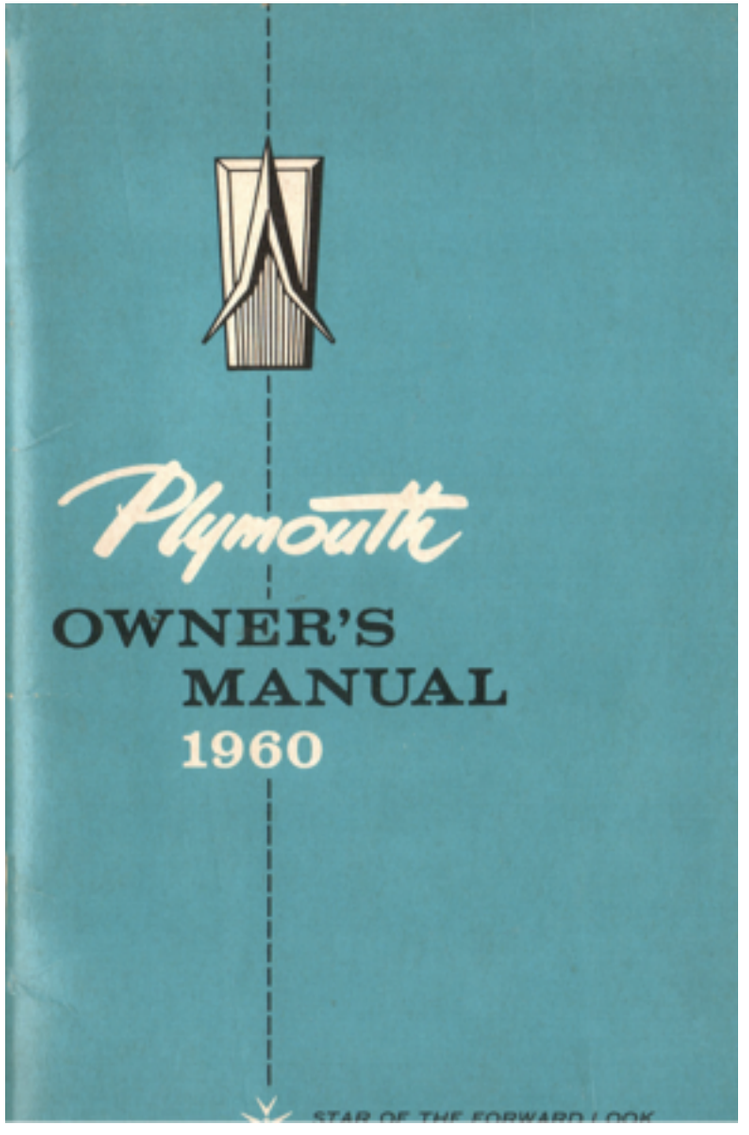
- Programming is the task of writing those instructions in a language that the computer can understand.

WHAT IS A PROGRAM?



WHAT IS A PROGRAM?





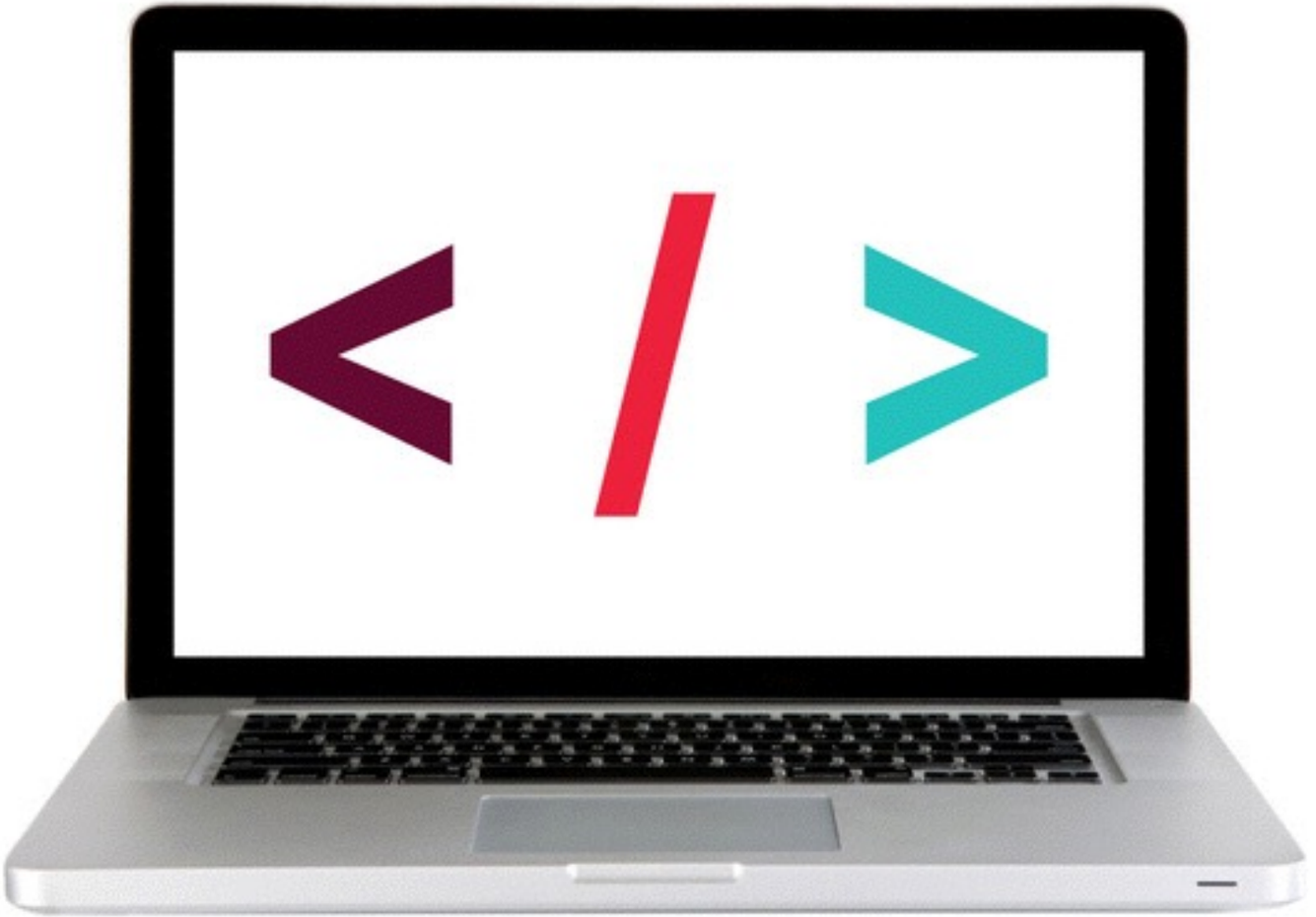
BECOMING A PROGRAMMER

It isn't about the programming language!!!
It is about changing how you think.

HOW COMPUTERS 'THINK'

- Short answer — they don’t think!
- While computers don’t think, they *act as if they do*, by sequentially executing simple instructions.
- The only things a computer knows are the things we tell it.
- A computer doesn't learn to perform tasks like you and I — it needs to follow instructions every time it performs the task.

LET'S TAKE A LOOK — SANDWICH TIME!



INTRO TO PSEUDO CODE

PSEUDO CODE

- When we write a program, we need to figure out a way to translate the ideas that are in our heads into code
- Pseudo code is a way to 'plan out' your program before coding it
- **Pseudo code** is a detailed yet readable description of what a computer program must do, expressed in plain english rather than in a programming language

THE IMPORTANCE OF PLANNING

Image credit: Minecraft HD Wallpapers

PSEUDO CODE — THERMOSTAT

Goal: *Write pseudo code for an application that would monitor the room temperature and adjust it so the room remains at a certain temperature.*

LAB — ROCK PAPER SCISSORS

LAB — ROCK PAPER SCISSORS

EXERCISE

KEY OBJECTIVE

- Practice programmatic thinking by writing pseudo code to solve a basic problem

TYPE OF EXERCISE

- Group of 3-4

TIMING

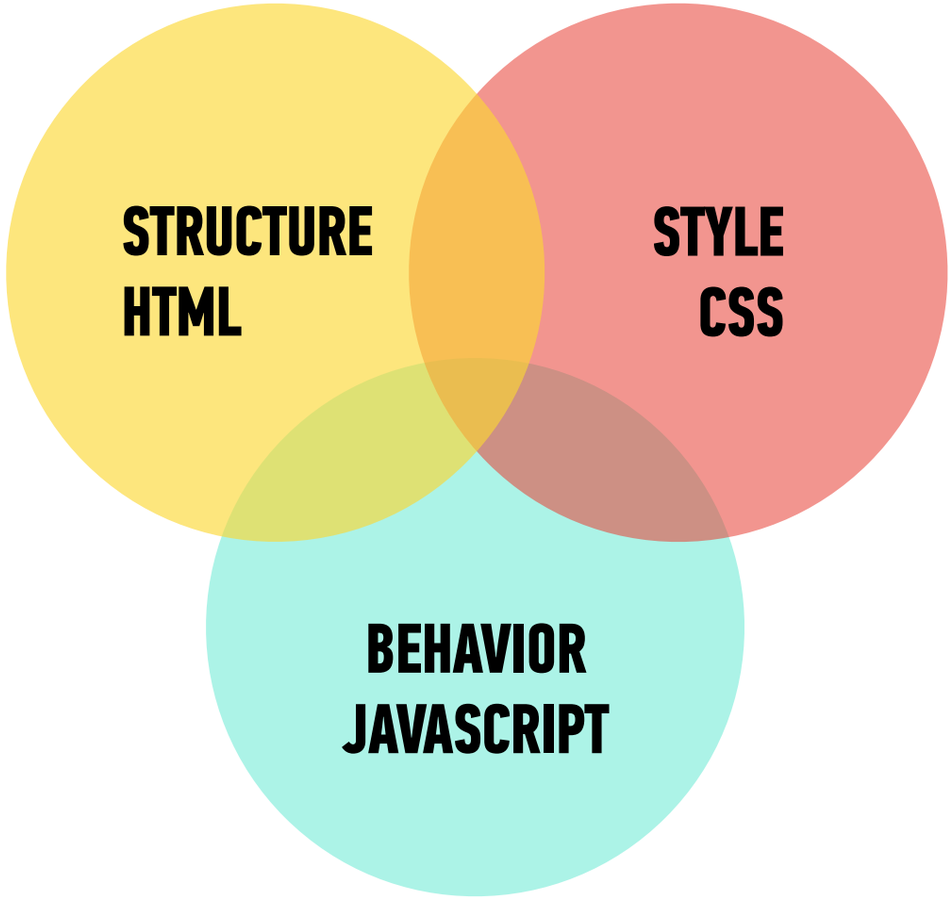
30 min

1. Write pseudo code to program a computer to play the game 'rock paper scissors'
2. Write each line of instruction onto a post it
3. Put the post its in order to form the program
4. If you finish early, walk around and view what other groups came up with

INTRO TO JS

THE THREE AMIGOS: STRUCTURE, STYLE, BEHAVIOR

- HTML = Noun
- CSS = Adjective
- Javascript = Verb



JAVA VS. JAVASCRIPT

Just a quick note! We're learning **JavaScript** *in this class, not Java. Java and JavaScript are actually two different languages.*

!==

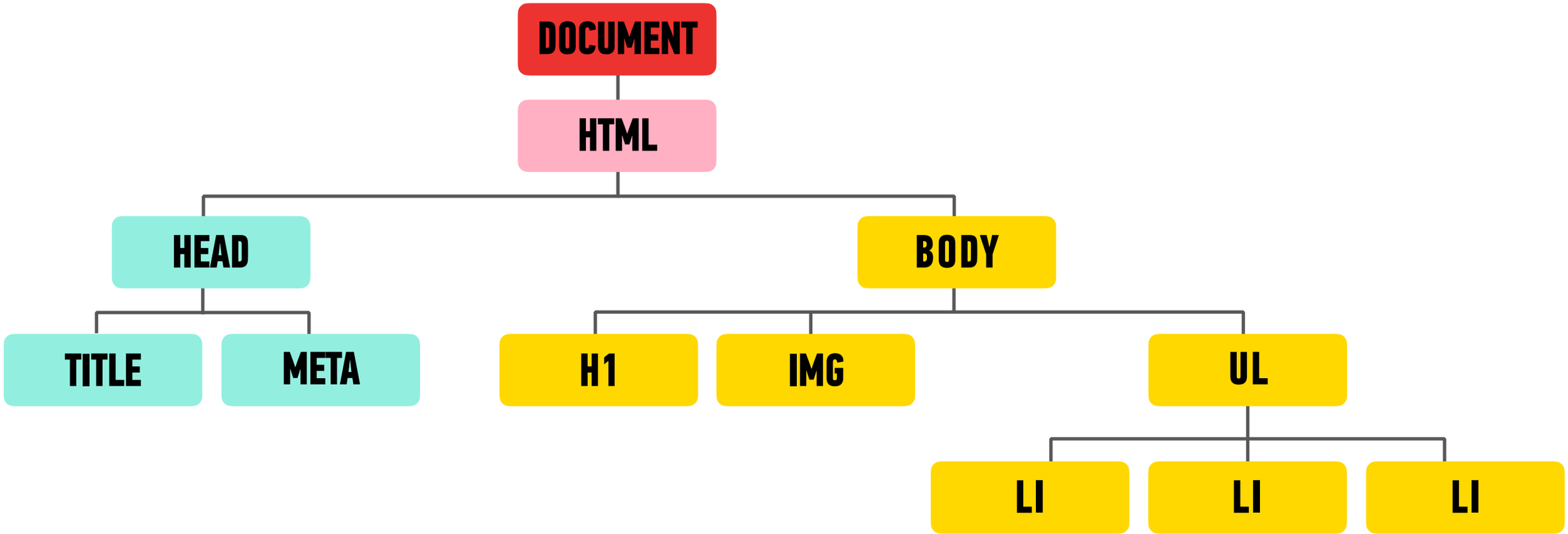
THE DOM

DOM TREE

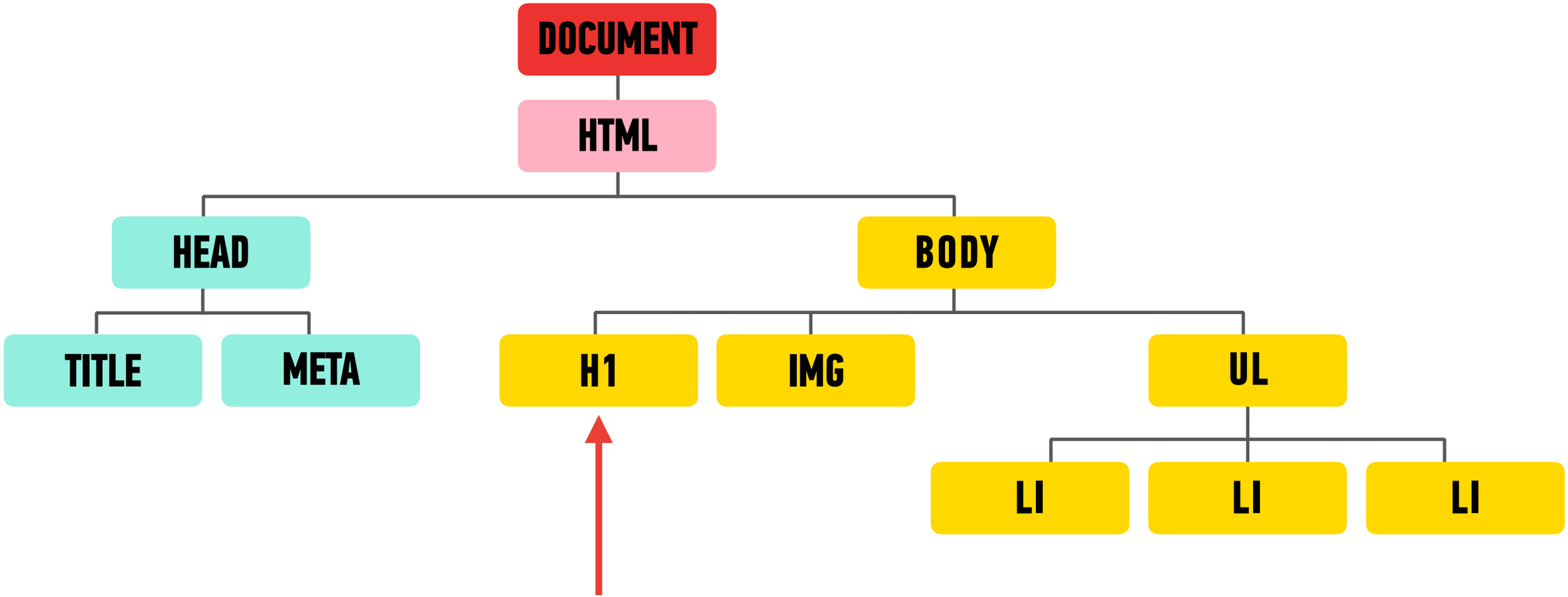
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Methods | Getting/Setting Content</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>

  <h1>Grocery List</h1>

  <ul>
    <li>Pepper Jack Cheese</li>
    <li>Hot Sauce</li>
    <li>Tortilla Chips</li>
  </ul>

  <img src="images/giphy.gif">

  <script src="js/main.js"></script>
</body>
</html>
```

DOM TREE

- The browser pulls in this HTML document, parses it, and creates an object model of the page in memory.
- This model is called the Document Object Model (DOM).
- The DOM specifies that the browser should use a DOM Tree to structure this model:



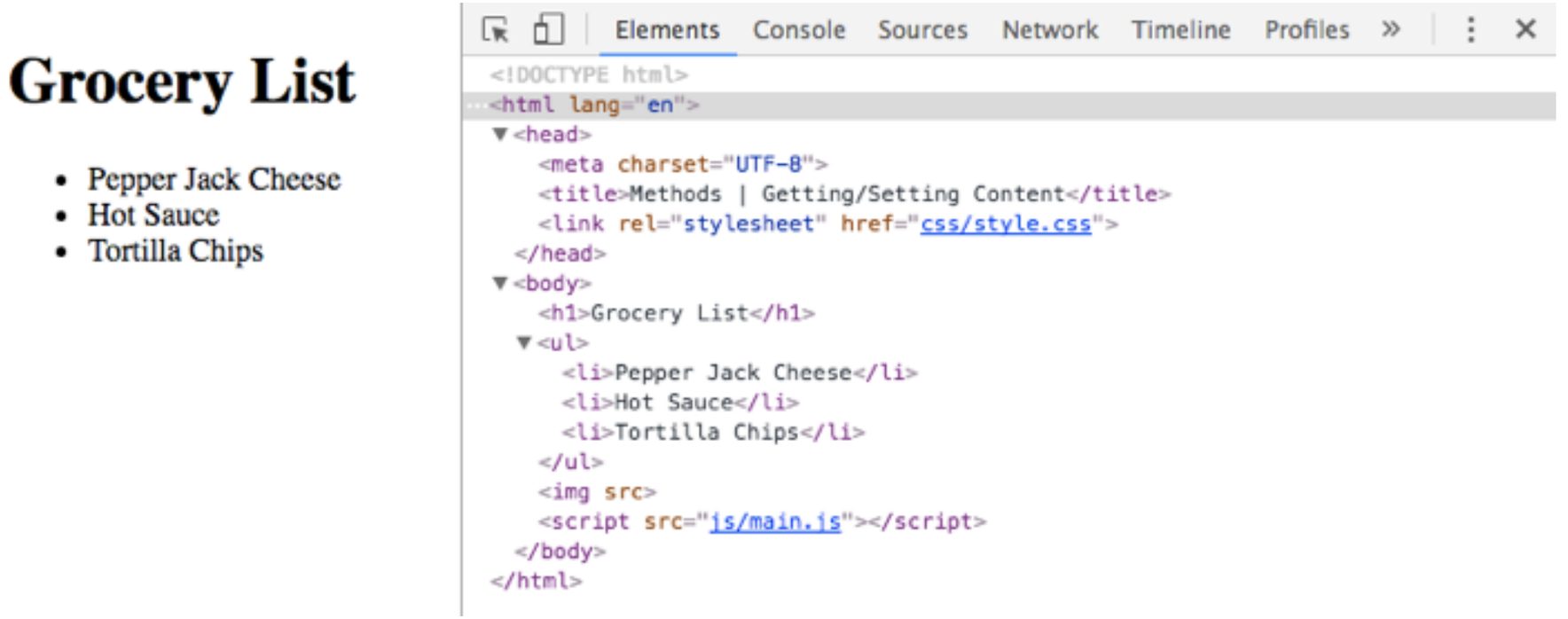
DOM TREE



- Each element in the HTML document is represented by a DOM node.
- You can think of a node as a live object that you can access and change using JavaScript.
- When the model is updated, those changes are reflected on screen.

DOM TREE

- Modern browsers come with tools developers can use to look at the DOM.
- In Chrome, you can go to View > Developer > Developer Tools and click on the Elements panel to take a look at the DOM tree.



WHAT JAVASCRIPT CAN DO!

1. Access Content
2. Modify Content
3. Program Rules
4. React to Events

WHAT JAVASCRIPT CAN DO!

1. Access Content
2. Modify Content
3. Program Rules
4. React to Events

You can use JS to select any element, attribute or text from an HTML page.

For example:

- Select the text inside all the <p> elements on a page
- Select the element that has the id attribute with a value of **email**
- Find out what the user entered into a text input when they submit a form

WHAT JAVASCRIPT CAN DO!

1 Access Content

2 Modify Content

3 Program Rules

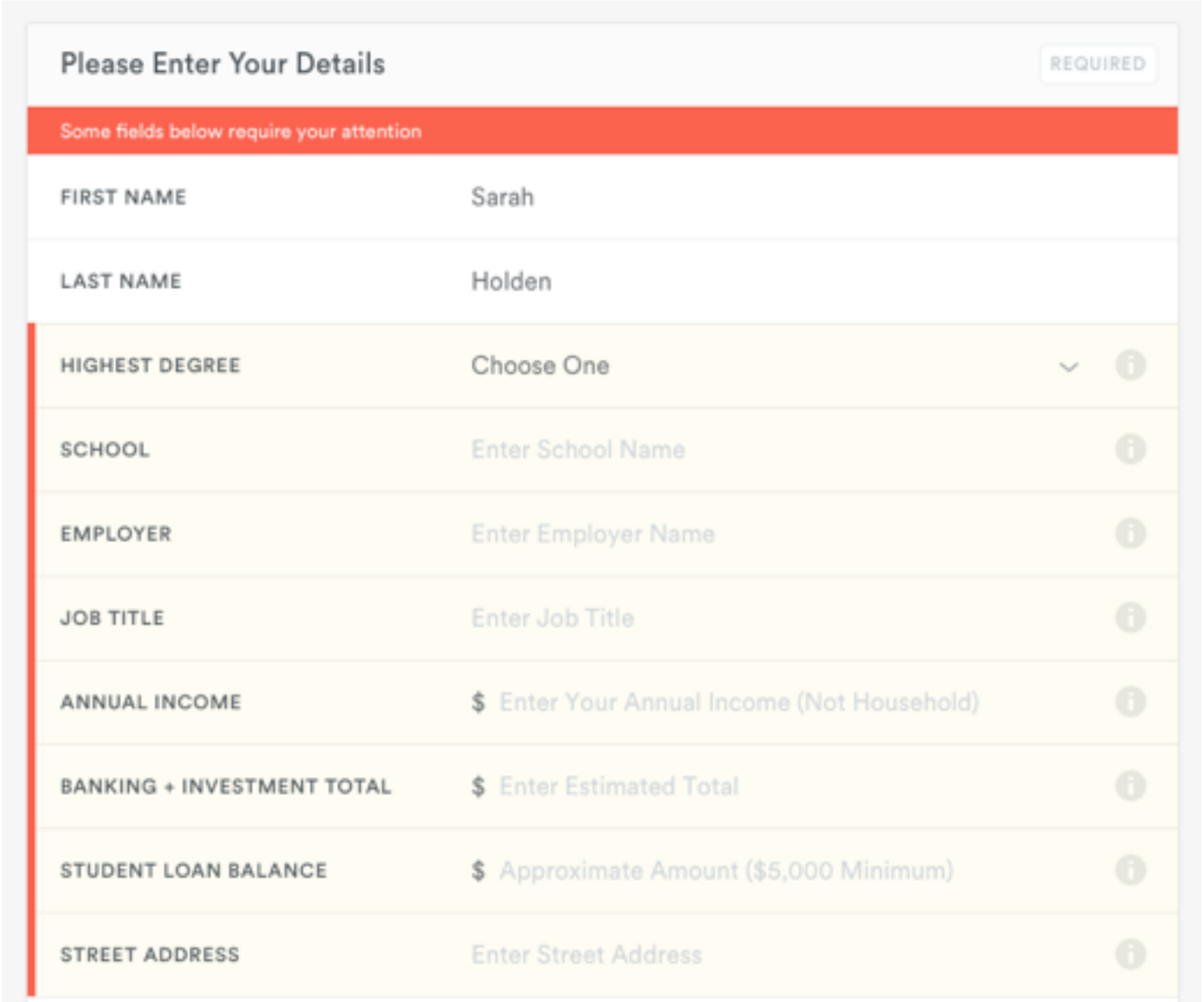
4 React to Events

You can use JS to add elements, attributes and text to the page (or remove them)

For example:

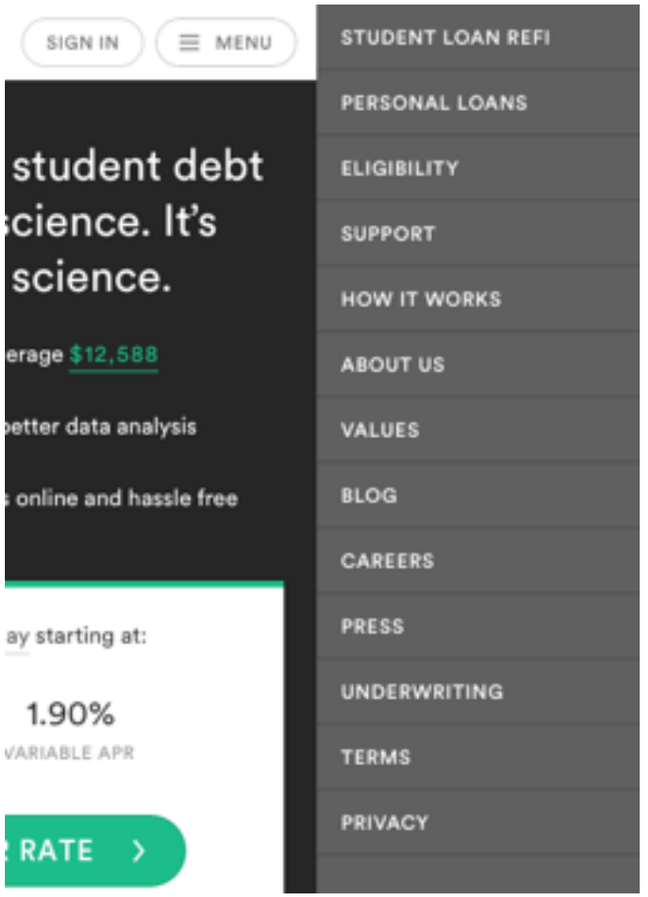
- Add an error message below a form
- Change the size, position, color, or other styles for an element
- Add or remove a class from elements to trigger new CSS rules for those elements

WHAT JAVASCRIPT CAN DO – MODIFYING CONTENT



Add an error message (and styles) to a form

WHAT JAVASCRIPT CAN DO – MODIFYING CONTENT



Change the size, position, color, or other styles for an element

WHAT JAVASCRIPT CAN DO!

1 Access Content

2 Modify Content

3 Program Rules

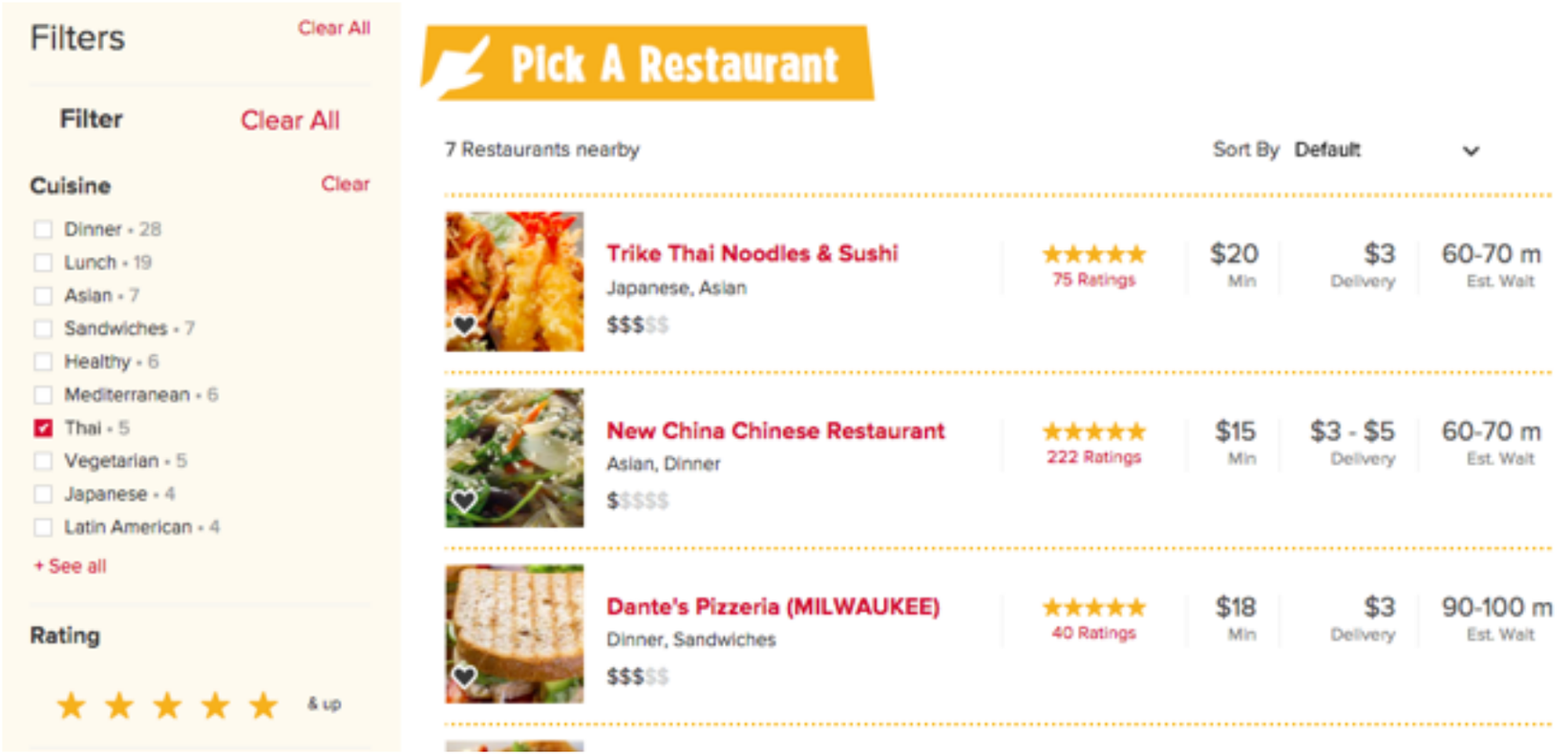
4 React to Events

You can specify a set of steps (instructions) for the browser to follow.

For example:

- Have images/text fade in as the user scrolls down the page
- Check to make sure the user has entered a valid email address into a form and display an error message if not
- Open a chat panel when the user clicks on a 'Chat with Us' button
- Filter data when the user selects a filter

WHAT JAVASCRIPT CAN DO – PROGRAM RULES



Filter data when the user selects a filter

WHAT JAVASCRIPT CAN DO!

1 Access Content

2 Modify Content

3 Program Rules

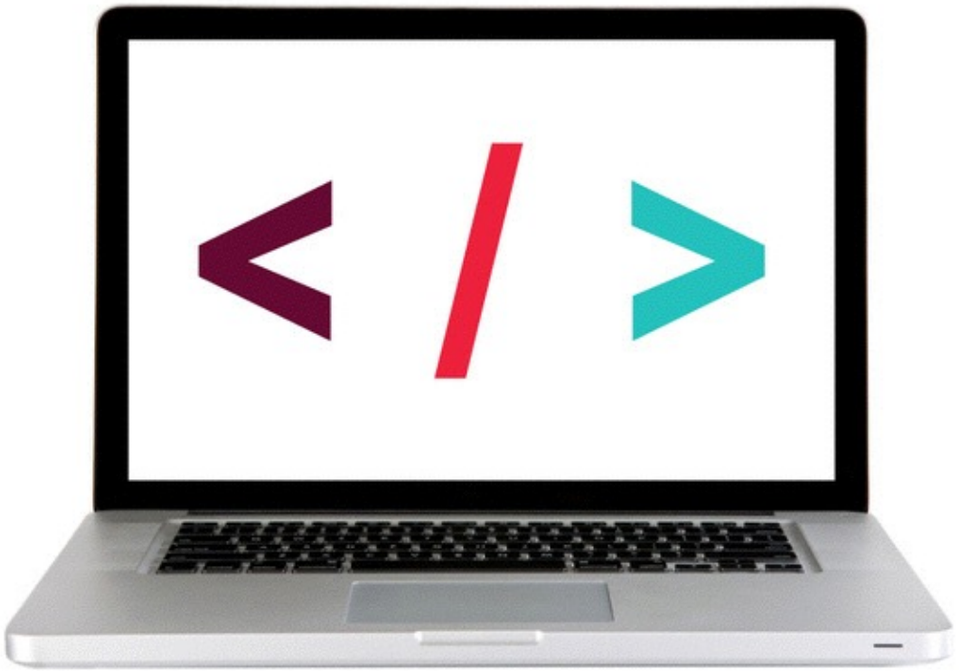
4 React to Events

You can specify that a script should run when an event occurs

For example:

- When a button is clicked
- When the cursor hovers over an element
- When the user types information into a form
- When a page has finished loading
- When the user hits enter to submit a form

LET'S TAKE A LOOK

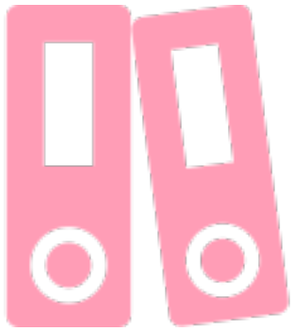


https://kinhr.com/

READING JS

READING JS

- When you are a child you learn to speak and read before you learn to write
- We learned to 'speak' JS with the discussion, video, and pseudo code



READING JS — COLOR SWITCHER WALK THROUGH

Color Switcher CodePen

LAB — TRAFFIC LIGHT

LAB — TRAFFIC LIGHT

EXERCISE

KEY OBJECTIVE

- Predict DOM output / changes by reading JS code.

TYPE OF EXERCISE

- Partner

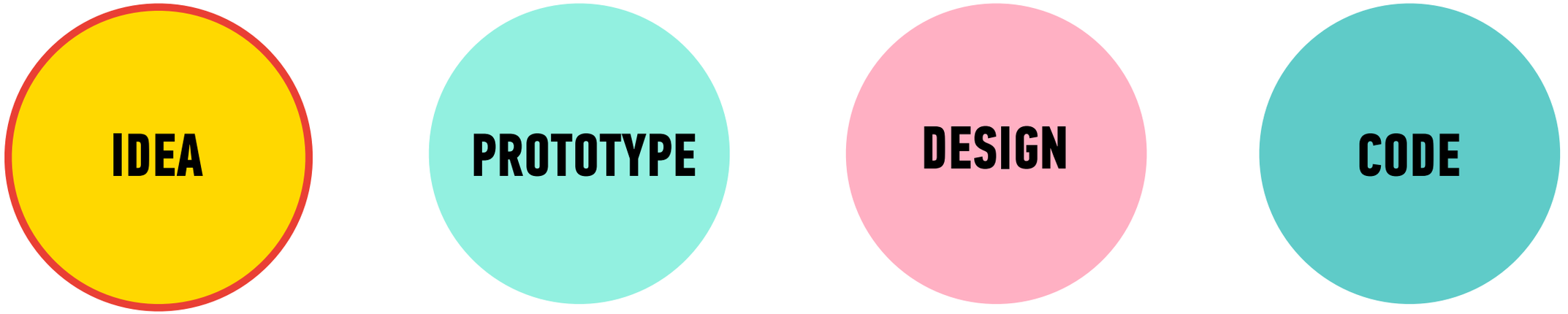
TIMING

30 min

1. Take a look at the Traffic Light code in Codepen
2. The yellow button changes the bulb to purple and the green light does not work.
3. Make some minor changes to the code so that the traffic light works correctly.

WIREFRAMES

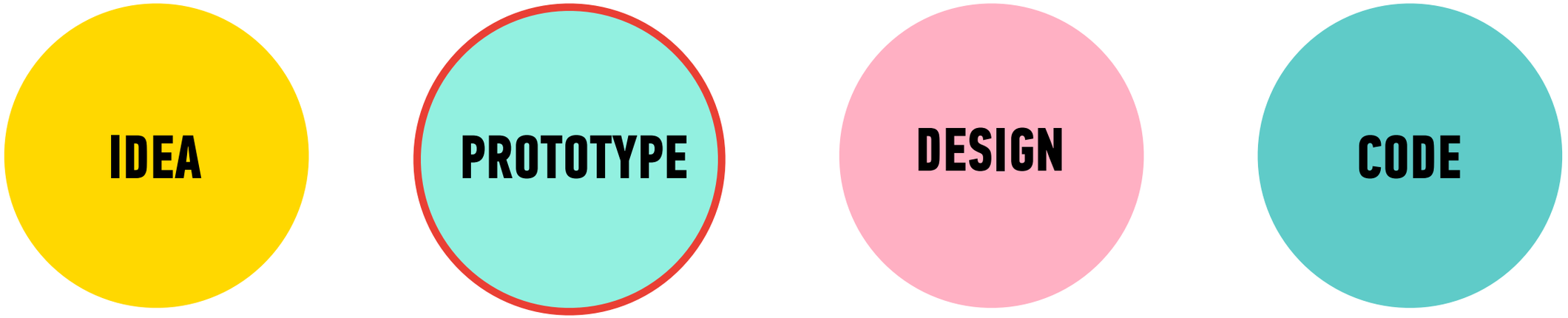
GETTING STARTED



FIND A COMPELLING PROBLEM:

- What problem are you solving? And for whom?
- How is your “customer” solving this problem now?
- Why is your proposed solution better?

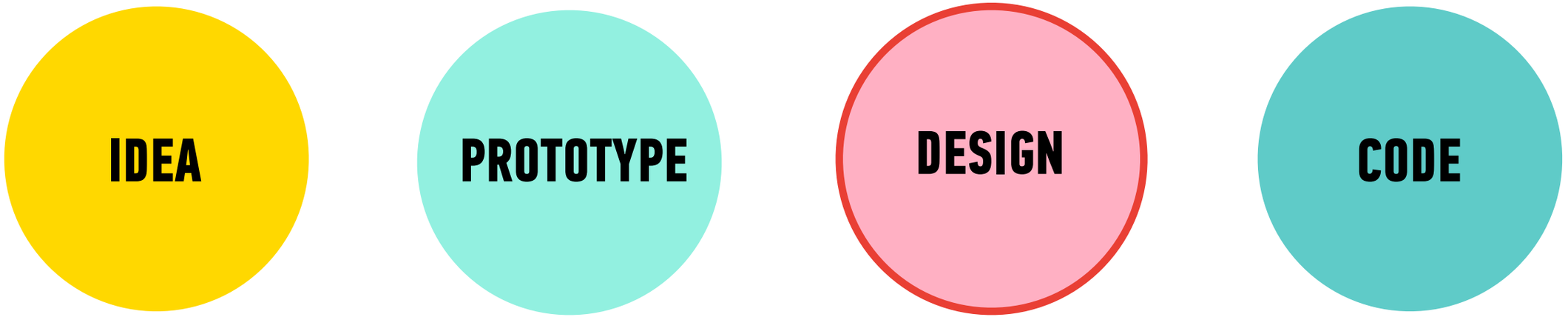
GETTING STARTED



A PICTURE IS WORTH 1000 WORDS:

- How will users most likely access my site?
- What type of content/information is most valuable? How will I structure things in a way that users will be able to easily find the most important information?
- How will users navigate the site? Will it be a single-page or multi-page site?
- What types of organizational structures are possible?

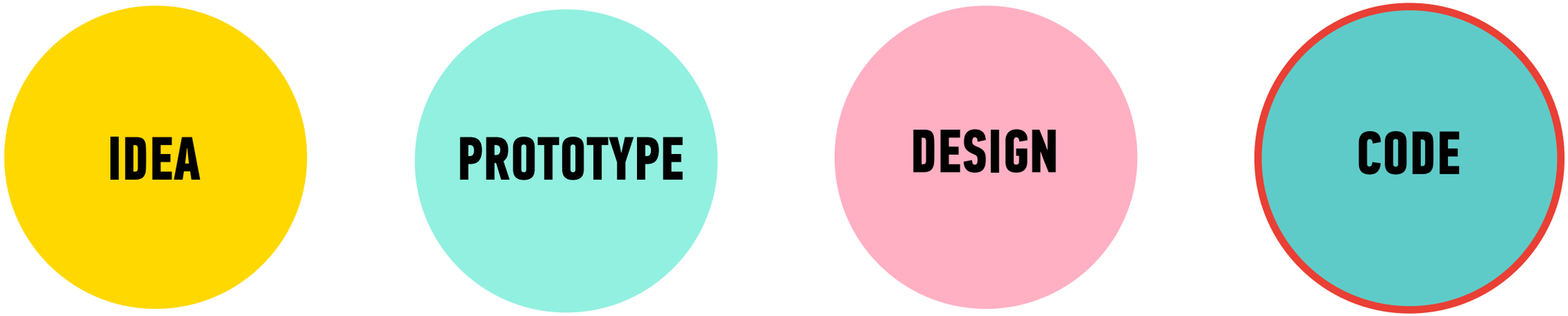
GETTING STARTED



FILL IN THE DETAILS:

- I often find it helpful to create a "styleguide" for the site
- Pick 4-5 colors for the site
- Look through Google Fonts/Font Squirrel and pick out 2-3 fonts for the site
- Use sites like https://unsplash.com/ for high-resolution stock images
- Sites like http://www.siteinspire.com/ , http://www.awwwards.com/ and http://www.webdesign-inspiration.com/ are great for inspiration!

GETTING STARTED



PLAN MORE, CODE LESS:

- Create a features/functionality list
- Start with the structure (HTML), then add styles (CSS), then work on interactions (pseudo code and JavaScript).

TIMELINE

Milestone 1	Project Proposal / Wireframes
Milestone 2	Draft of HTML / CSS (no JS)
Milestone 3	First draft of JS
Milestone 4	Final Presentation

Find more info and dates here

LAB — TRAFFIC LIGHT

EXERCISE

KEY OBJECTIVE

- Start on wireframes/project proposals

TYPE OF EXERCISE

- Individual/Partner

TIMING

Until 9:20

1. Look at http://gallery.ga.co/ for inspiration
2. Start on wireframes/project proposals

LEARNING OBJECTIVES

- Practice programmatic thinking by writing pseudo code to solve a basic problem.
- Define web site behavior and the practical uses of JavaScript.
- Predict DOM output / changes by reading JS code.

HOMEWORK

HOMEWORK

- Final project milestone #1
- Be sure to read the homework specs! There is some highly recommended reading from the textbook that will really help prepare you for next week!

EXIT TICKETS

http://goo.gl/forms/vPhCOlfESf