GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-9-resources` repo to your computer
2. Open the `starter-code` folder in your code editor

SLACK BOT LAB

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Create a program that hoists variables
- Install and configure all utilities needed to run a Hubot
- Write scripts that allow your Hubot to interact with users of the class Slack organization

AGENDA

- Install and configure Slack bot utilities and accounts
- Explore sample code for bots
- Plan what you'd like your bot to do
- Create a basic bot to verify that your setup works
- Expand on your basic code to add your planned functionality

WEEKLY OVERVIEW

WEEK 4	Slack bot lab / Objects & JSON
WEEK 5	Intro to the DOM / Intro to jQuery
WEEK 6	Advanced jQuery / Ajax & APIs

EXIT TICKET QUESTIONS

SUBMIT HOMEWORK: STEP 1

In Finder:

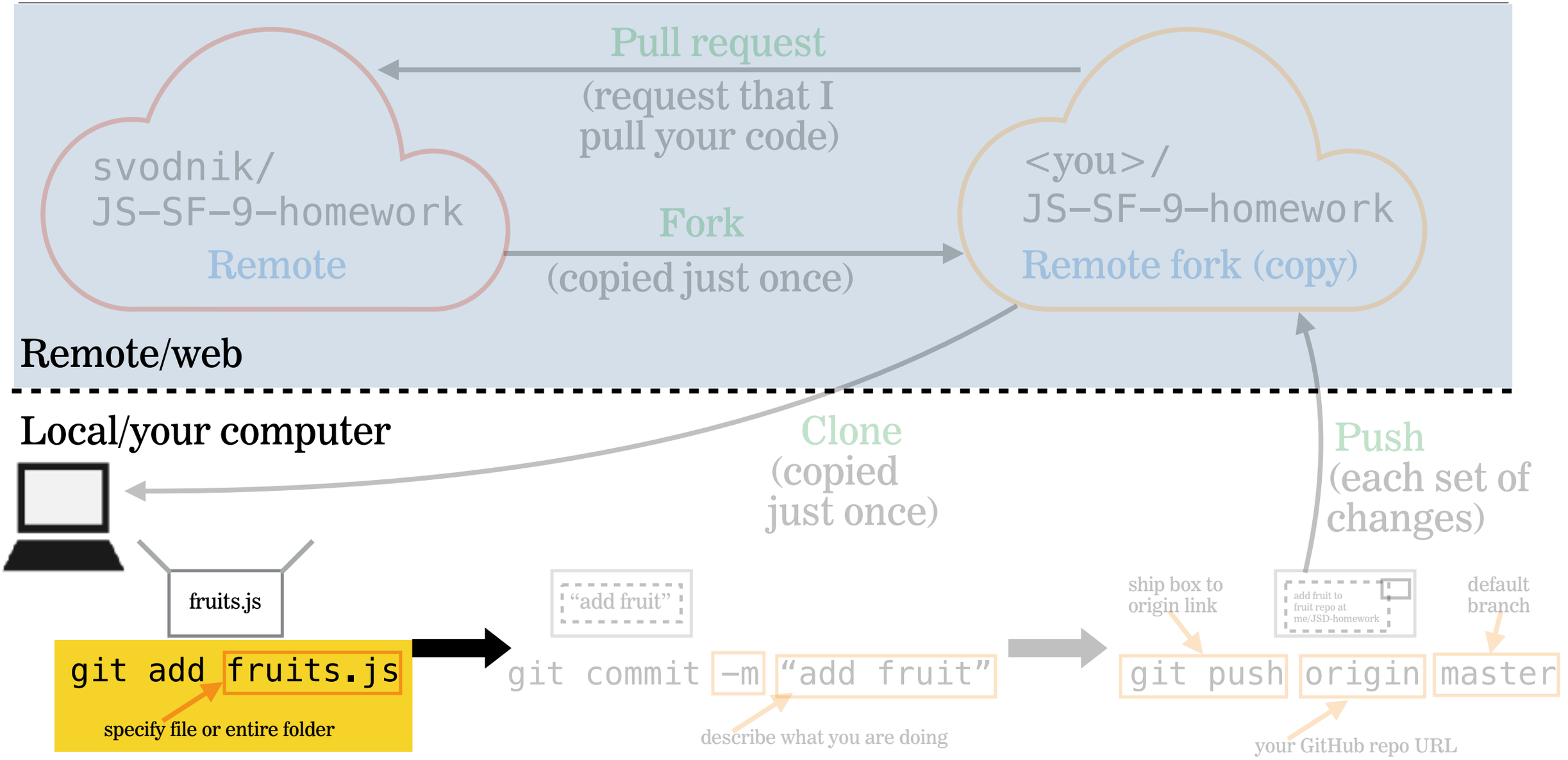
- navigate to *`firstname-username`* folder (example: `Sasha-svodnik`)
- copy your completed Homework-2 folder from last Thursday into your *`firstname-username`* folder.

SUBMIT HOMEWORK: STEP 2

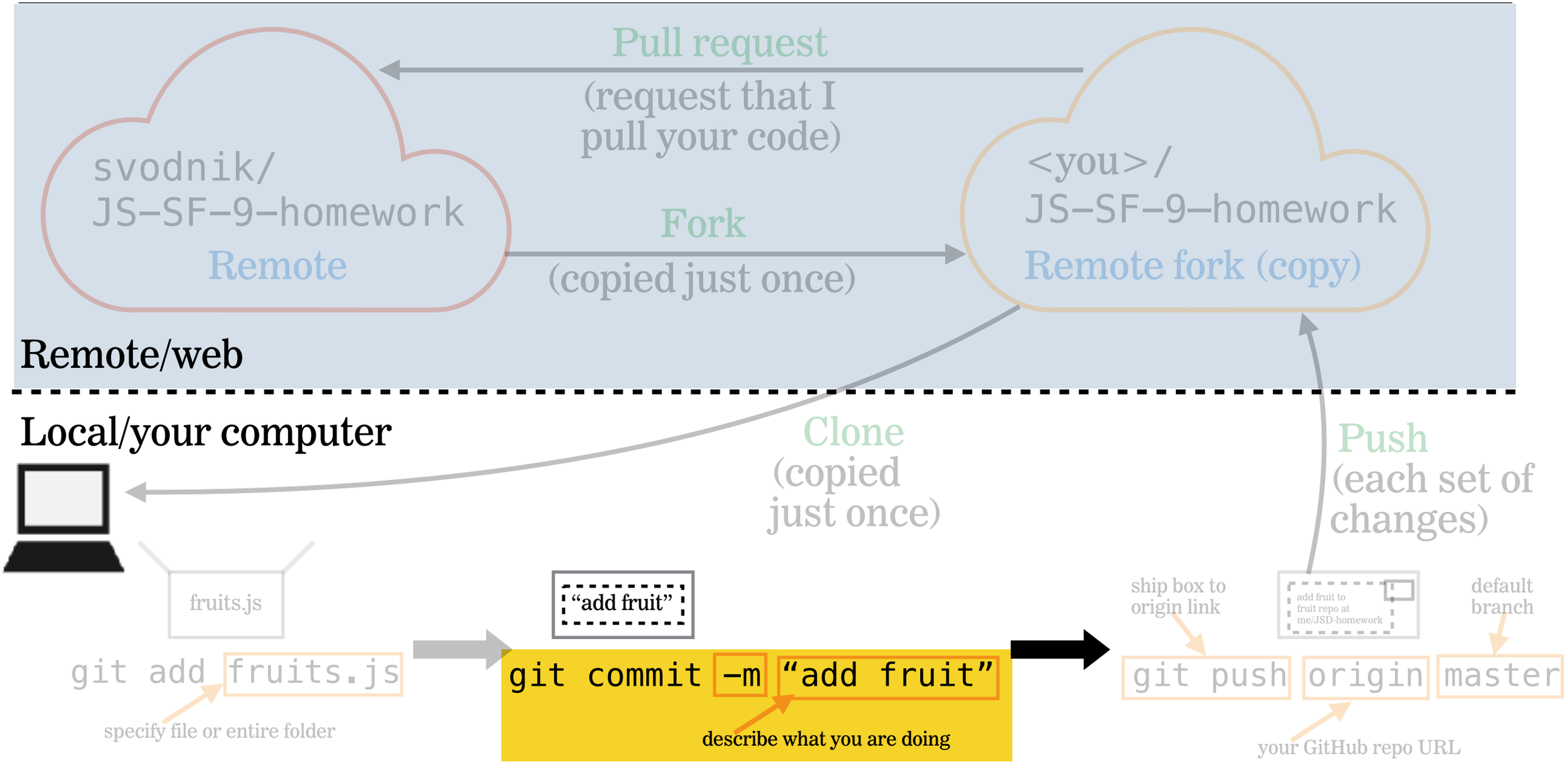
In Terminal:

- navigate to *`firstname-username`* folder
- `git add .`
- `git commit -m “submitting homework 2”`
- `git push origin master`

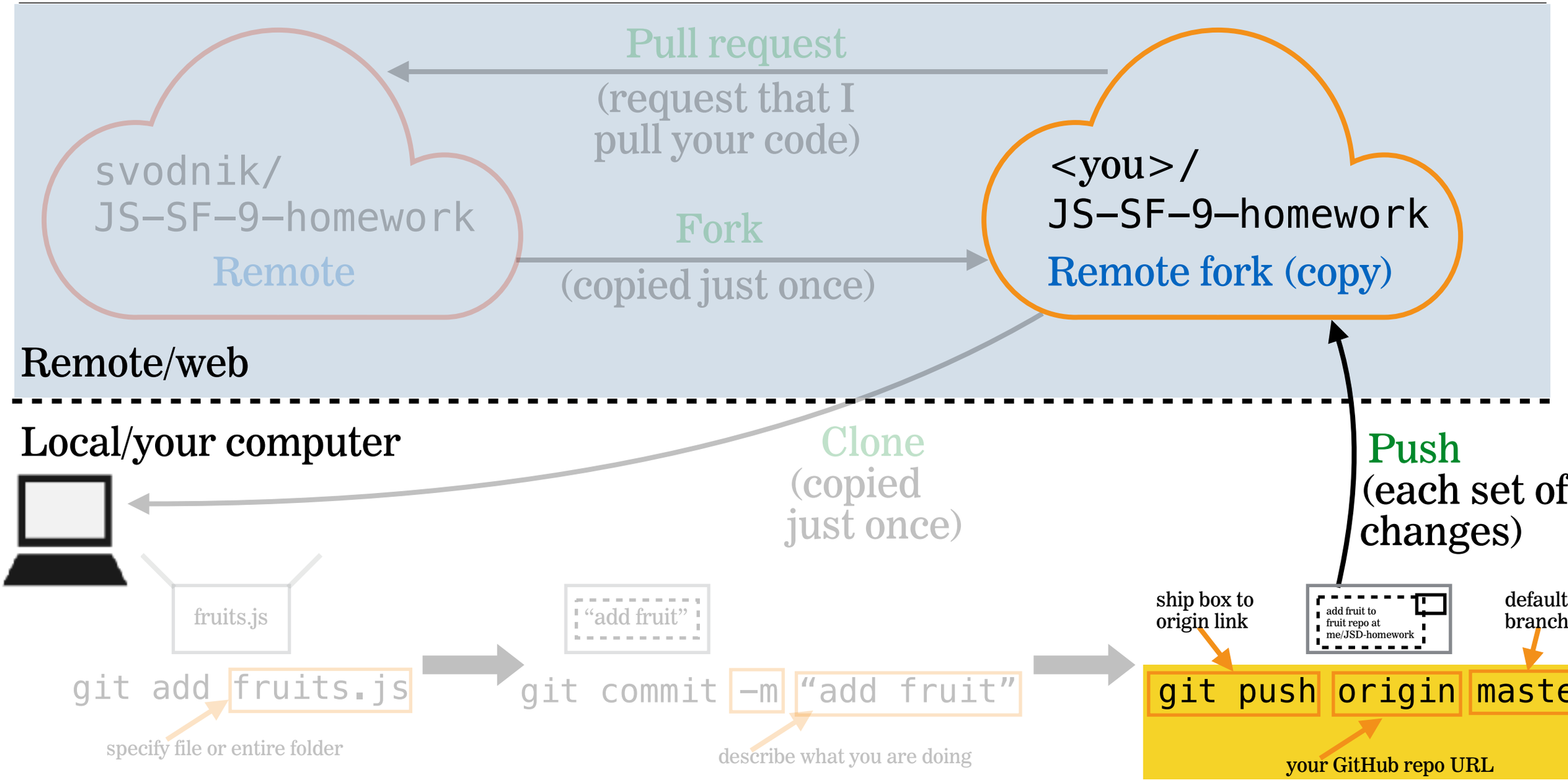
USING THE JS-SF-9-HOMEWORK REPO



USING THE JS-SF-9-HOMEWORK REPO



USING THE JS-SF-9-HOMEWORK REPO

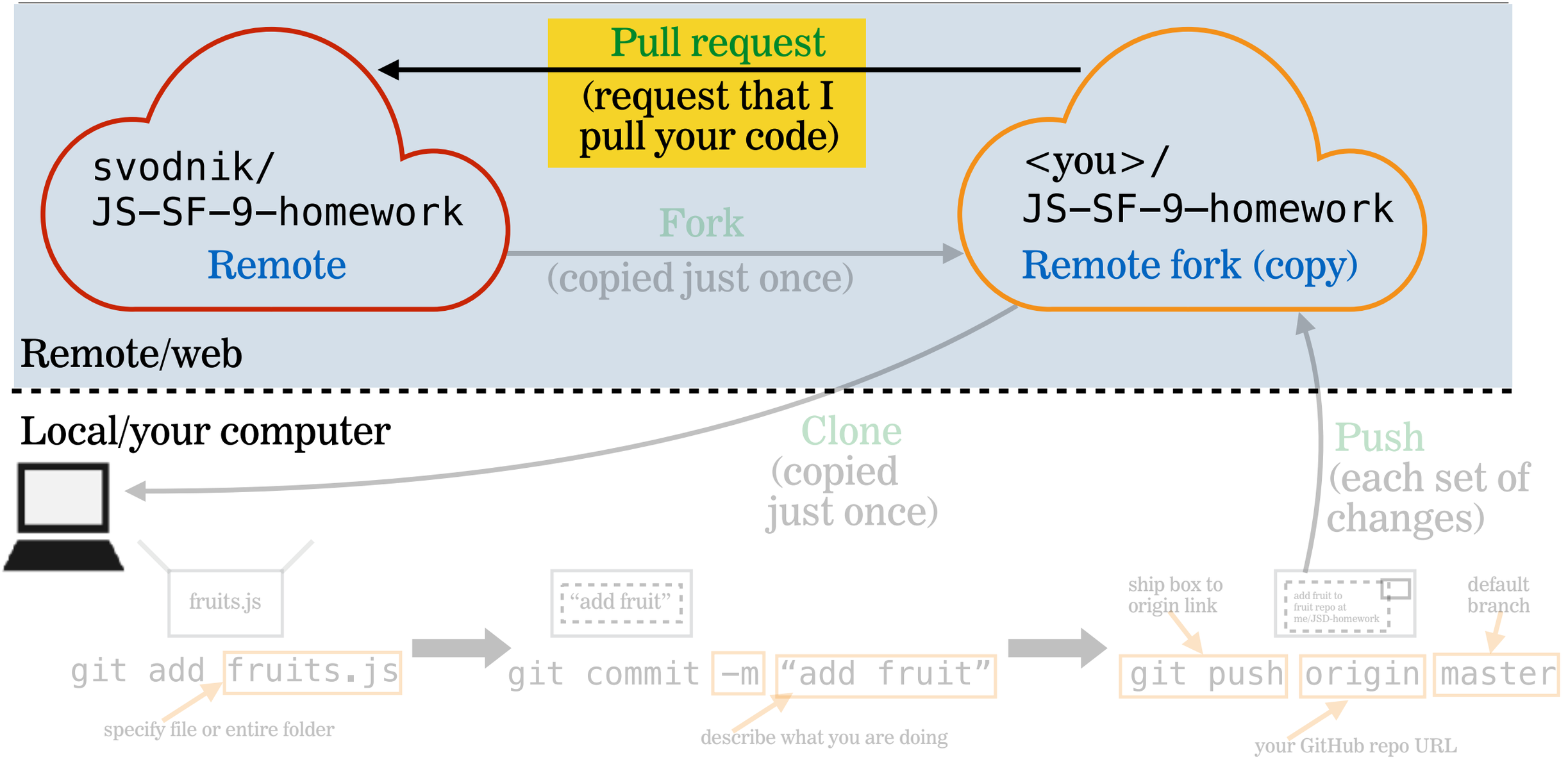


SUBMIT HOMEWORK: STEP 3

In Browser:

- Go to your fork of `JS-SF-9-homework` on github.com
- click **New pull request**
- click **Create pull request**
- click **Create pull request** (again)

USING THE JS-SF-9-HOMEWORK REPO



HOMEWORK REVIEW

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

6 min

1. Share 1 thing you're excited about being able to accomplish.
2. Have each person in the group note 1 thing they found challenging for the exercises and make note. Discuss as a group how you think you could solve that problem.
3. Did you complete either of the bonus exercises? Demonstrate it and show your group how you did it!

REVIEW – CATCH PHRASE!

EXERCISE

TYPE OF EXERCISE

- Groups of 2-3

TIMING

3 min

1. Get your partner to guess the word on each piece of paper by giving clues describing it.
2. Take turns giving clues and guessing words.

SCOPE & HOISTING

`var`, `let`, `const`, AND SCOPE

- `var` obeys the scoping rules we've just seen
  - » "generic" way to create variables
- `let` and `const` are newer keywords with different scoping rules
  - » local scope within functions **and** within any block (including loops and conditionals)

var

- creates local scope only within functions

```
var results = [0,5,2];
```

let

- used in the same situations as `var`, but with different scoping rules for code blocks

```
let results = [0,5,2];
```

const

- used to declare constants
  - **immutable**: once you've declared a value using `const`, you can't change the value in that scope
  - by contrast, variables declared with `var` or `let` are **mutable**, meaning their values can be changed
- some developers use all capital letters for constant names, but this is not required

```
const SALESTAX = 0.0875;
```

let/const vs var

- `let` & `const` create local scope within any block (including loops and conditionals) but `var` does not

```
var x = 1;
if (true) {
  var x = 2;
  console.log(x);  // 2
}
console.log(x);  // 2
```

global scope

```
let x = 1;
if (true) {
  let x = 2;
  console.log(x);  // 2
}
console.log(x);  // 1
```

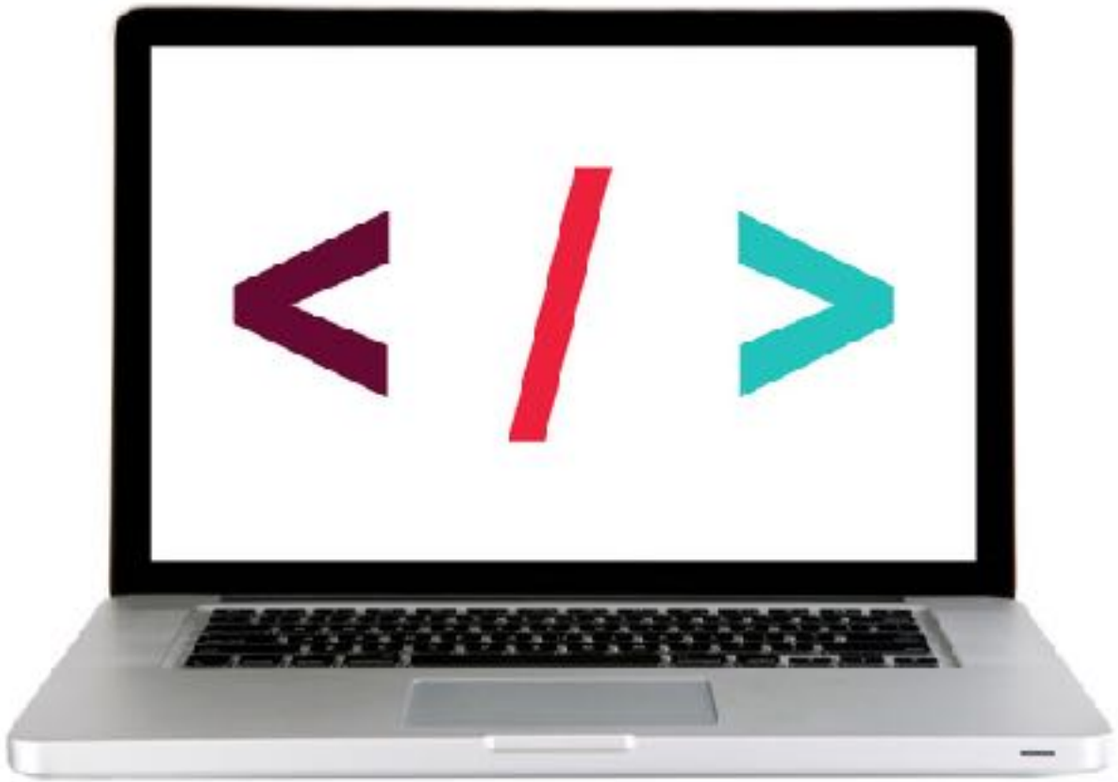
treated as local scope by let statement

global scope

`var`, `let`, `const`, AND BROWSER SUPPORT

- `let` and `const` are not supported by older browsers
  - » see caniuse.com, search on `let`
- babel.js (babeljs.io) allows you to transpile newer code into code that works with older browsers as well
- we will rely on `const` and `let` in class

LET'S TAKE A CLOSER LOOK



EXERCISE — VAR, LET, AND CONST

KEY OBJECTIVE

- Distinguish between `var`, `let`, and `const`

TYPE OF EXERCISE

- Individual or pairs

EXECUTION

2 min

1. Draw the table shown on the whiteboard, which compares a few aspects of `var`, `let`, and `const` usage.
2. Complete the table.

`var`, `let`, AND `const`

keyword	local scope	can you change the value in the current scope?	browser support
`var`	within the code block of a **function** only	yes	all browsers
`let`	within any code block	yes	**only modern browsers**
`const`	within any code block	**no**	**only modern browsers**

LAB — LET, VAR, AND CONST

KEY OBJECTIVE

- Determine the scope of local and global variables

TYPE OF EXERCISE

- Pairs

LOCATION

- `starter code > 4-let-var-const-lab`

EXECUTION

5 min

1. Open the index.html file in your browser, view the console, and examine the error.
2. Follow the instructions in `js > app.js` to complete parts A and B.

HOISTING

- JavaScript's behavior of moving declarations to the top of a scope.
- This means that you are able to use a function or a variable before it has been declared.
- Variables declared with `var` are hoisted
- Variables declared with `let` and `const` are not hoisted

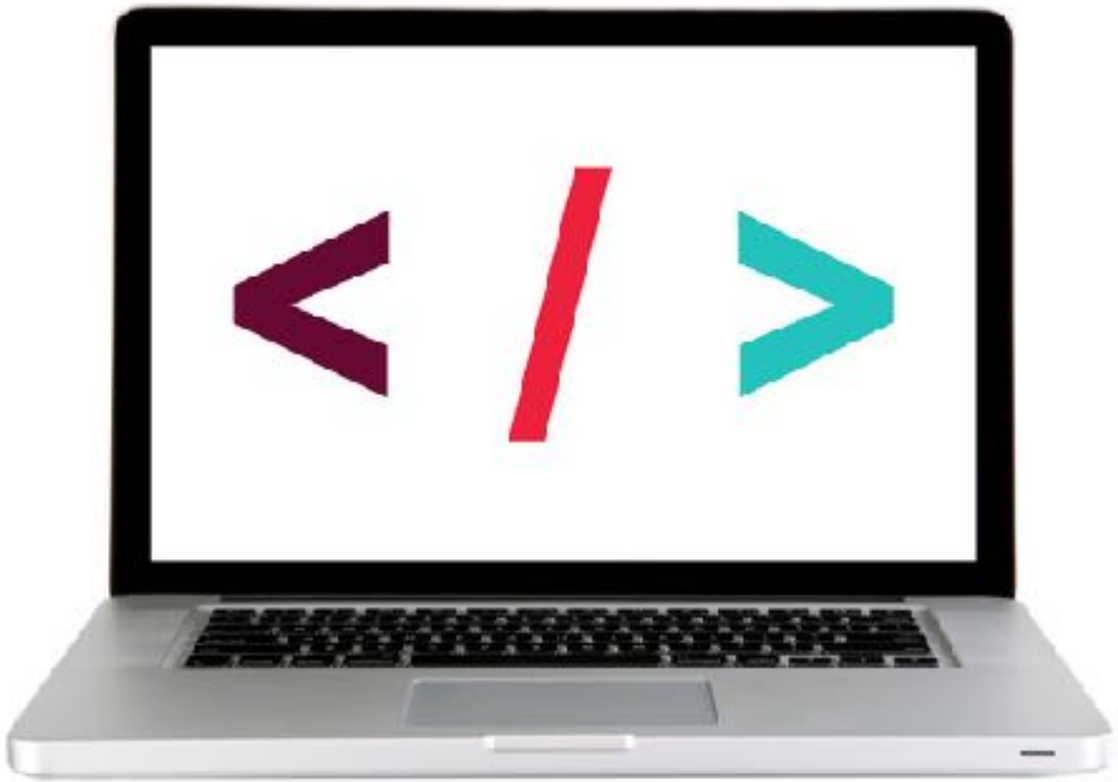
FUNCTIONS AND HOISTING

- Function expressions are treated like other variables
  - when declared with `var`, only the name is hoisted, not the value
  - when declared with `let`, they are not hoisted
- Function declarations are treated differently
  - the code for the entire function is hoisted along with a function declaration

FUNCTIONS AND HOISTING

function type	function name hoisted?	function content hoisted?
function declaration	yes	yes
function expression using `let`	no	no
function expression using `var`	yes	no

LET'S TAKE A CLOSER LOOK



EXERCISE — HOISTING

KEY OBJECTIVE

- Create a program that hoists variables

TYPE OF EXERCISE

- Groups of 3

EXECUTION

2 min

1. Examine the code in the Slack channel.
2. Discuss with your group which parts of the code are hoisted.
3. Predict the result of each of the first four statements.

SLACK BOTS

SLACK AND BOTS

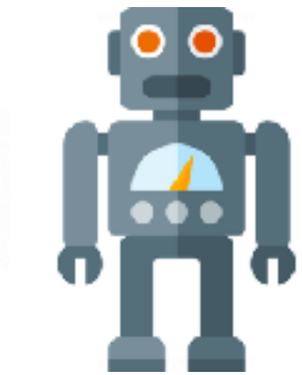
- **Bot**: A script programmed to interact with users as if it's a person
  - Slackbot
  - PlusPlus
- We will use a framework to create our own bots with interactive behaviors that we specify with our code
- These bots will be members of our class Slack organization

HUBOT

- **Hubot**: A framework meant to speed the process of developing bots for a variety of platforms, including Slack
- Includes built-in functionality for performing common bot tasks, such as posting images.
- We will use the Hubot framework to create our bots

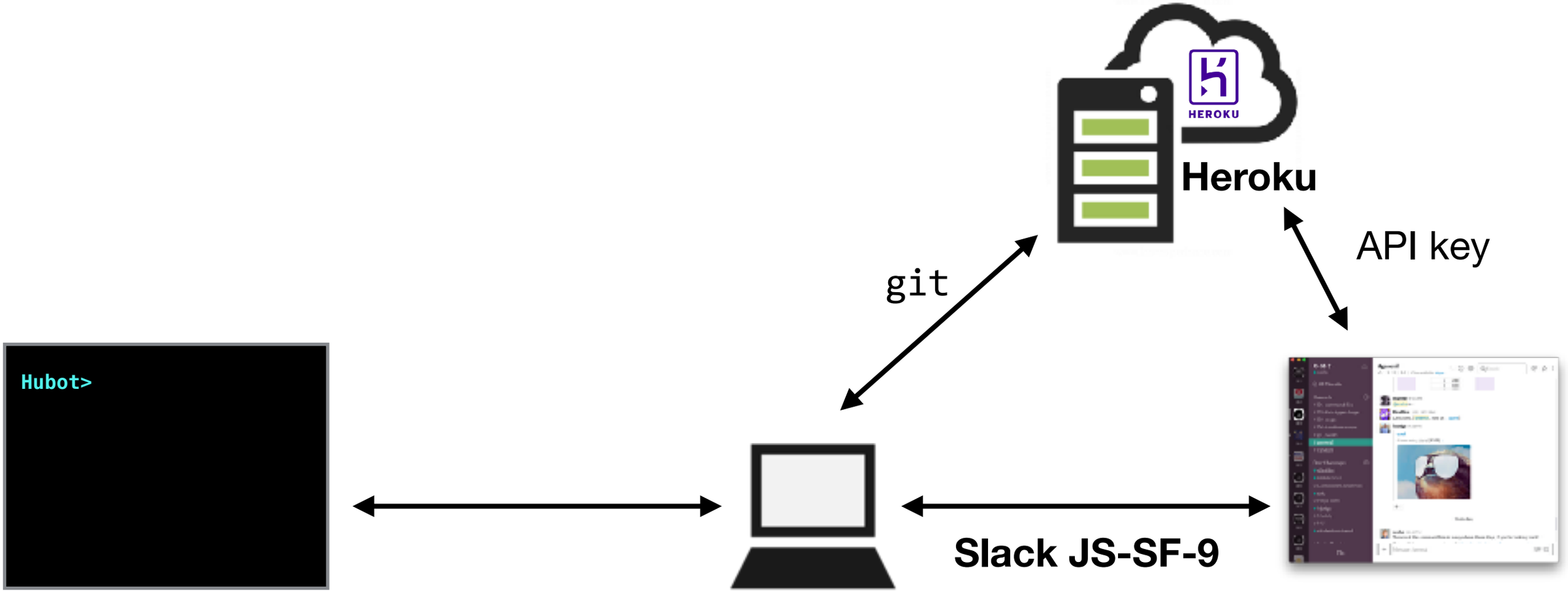
HUBOT vs SLACK BOT vs SLACKBOT

- Hubot is the framework we're using
- Each of us will be building a bot for Slack === a Slack bot
- Slackbot is the name of a specific bot already installed in our Slack organization; it answers questions about how to use Slack



HEROKU

- **Heroku**: A platform for hosting and running apps in the cloud.
- We will create our code on our computers, then push it to Heroku so it can run even when our computers are sleeping or shut down



Interacting with your bot at the command line involves local files on your computer only.

Interacting with your bot on the class Slack organization involves the files you published to your Heroku instance.

YEOMAN

- **Yeoman**: A set of tools that provides a scaffolding (basic structure) for getting web apps up and running quickly
- We'll use a Yeoman tool called yo, which automates a lot of behind-the-scenes work

YEOMAN

COFFEESCRIPT

- **CoffeeScript**: A variant of JavaScript, intended to be more readable and faster to type.
- Only JavaScript can run in browsers
  - Before being used, CoffeeScript code must be compiled, which is a process that translates it into JavaScript
- Many Hubot examples are written in CoffeeScript, but you can write Hubot code in vanilla JavaScript without any problem

MARKDOWN

- **Markdown**: A markup language used for creating formatted text documents.
- Easier to use than HTML for basic tasks
- Comes in different flavors; GitHub has its own
- Used to create README files that document projects in GitHub repos
- You will use Markdown to create a README file explaining what your bot does and how to use it

ACTIVITY — HUBOT CONFIGURATION

KEY OBJECTIVE

- Install and configure all utilities to run a Hubot

LOCATION

- Slack Bot Lab - Install Guide
  (first link in Resources on website for today's class)

EXECUTION

20 min

1. Follow the instructions to install command line utilities for building Hubots.
2. When you finish, start reading and exploring the sample code in Slack Bot Lab - Sample Code (second link in Resources on website for today's class)

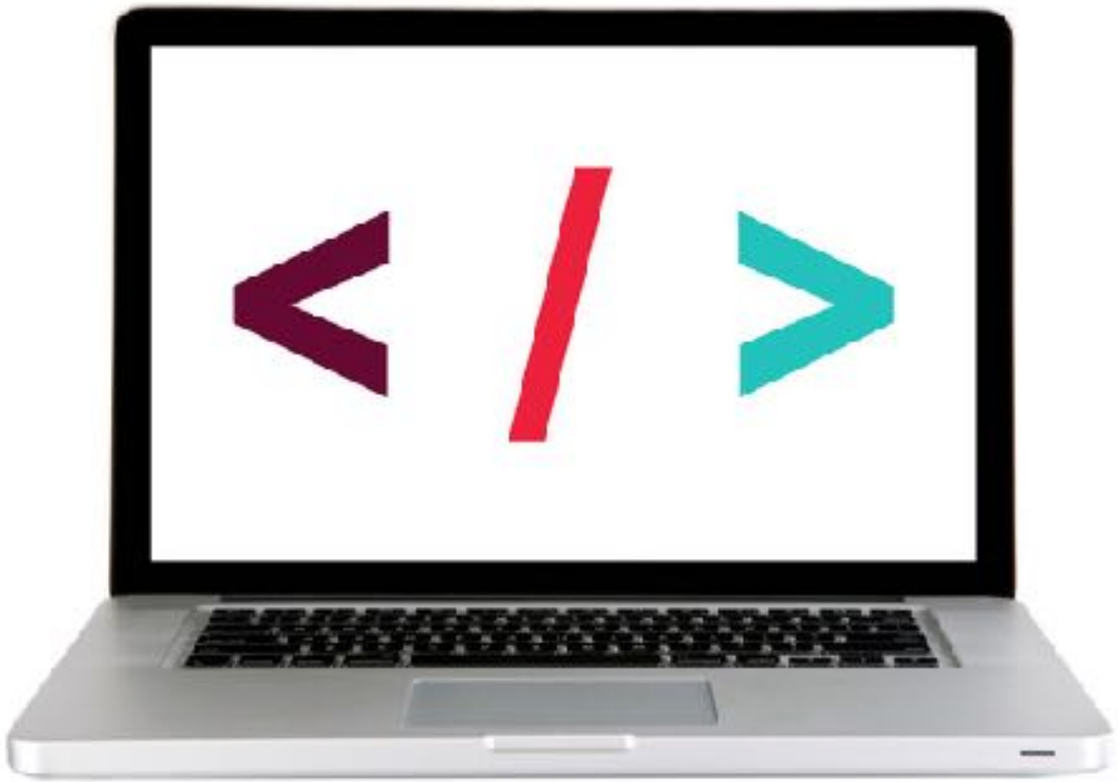
UNDERSTANDING THE HUBOT FRAMEWORK

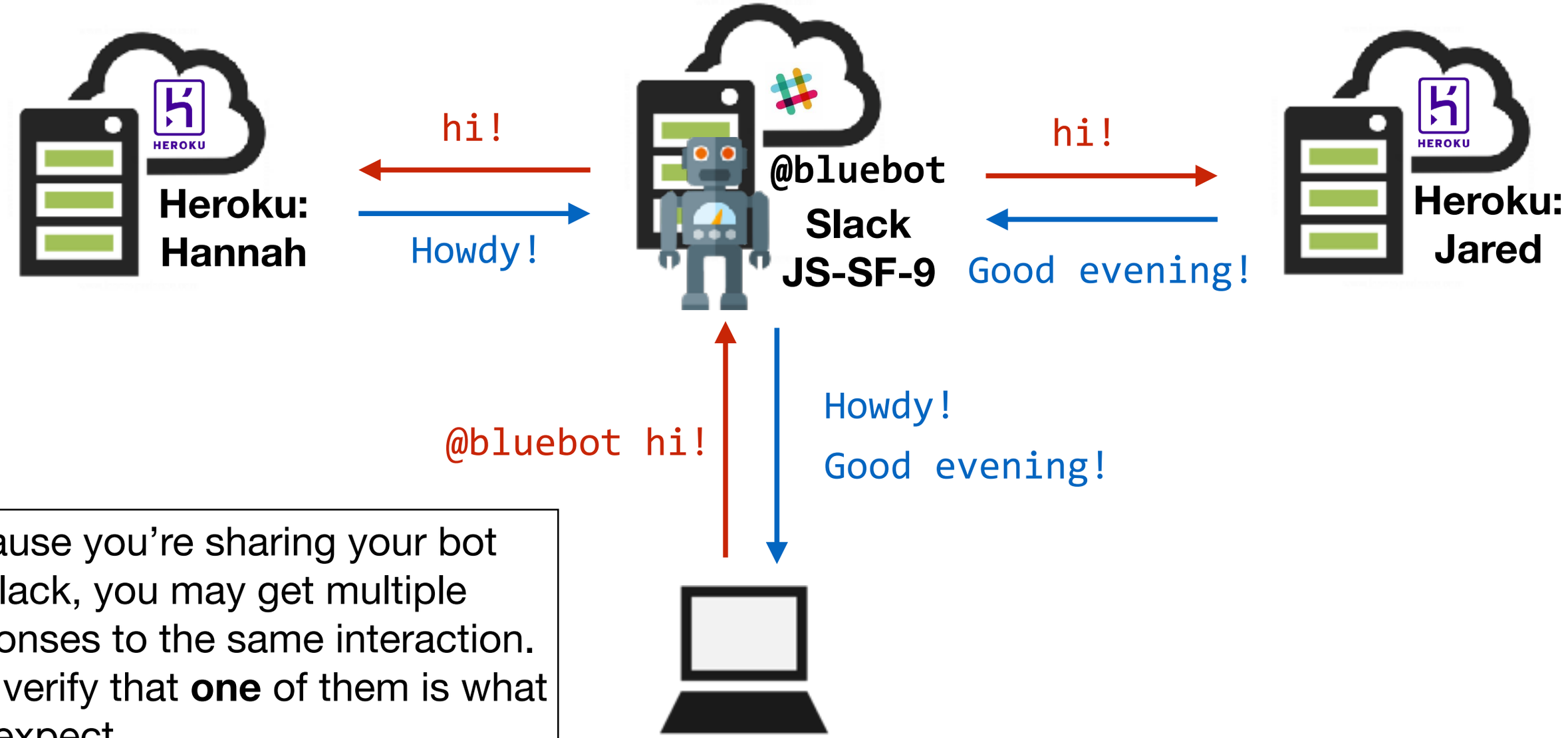
```
module.exports = function(robot) {
  robot.verb(parameter1, function(res) {
    return res.command();
  });
  robot.verb(parameter1, function(res) {
    return res.command();
  });
  ...
};
```

BASIC HUBOT VERBS

- `hear`: called anytime a message's text matches
- `respond`: called for messages immediately preceded by the robot's name or alias

LET'S TAKE A CLOSER LOOK





Because you're sharing your bot on Slack, you may get multiple responses to the same interaction. Just verify that **one** of them is what you expect.

LAB — BUILD A SLACK BOT

KEY OBJECTIVE

- Write scripts that allow your Hubot to interact with users of the class Slack organization

LOCATION

- `[hubot folder] > scripts > script.js`

EXECUTION

Until 9:20

1. Uncommenting portions of the sample code in `script.js` to explore how to code in the Hubot framework, and what a bot can do in Slack.
2. Try out some of the code samples in today's start code files.
3. Create a plan for what you want your bot to be able to do, pseudocode it, and start building it!
4. Test using the steps in Slack bot lab - Testing & Troubleshooting (third link on class resources on website)
5. BONUS: Experiment with advanced commands documented at https://github.com/github/hubot/blob/master/docs/scripting.md

Exit Tickets!

(Class #5)

LEARNING OBJECTIVES - REVIEW

- Create a program that hoists variables
- Install and configure all utilities needed to run a Hubot
- Write scripts that allow your Hubot to interact with users of the class Slack organization

NEXT CLASS PREVIEW

Objects and JSON

- Identify likely objects, attributes, and methods in real-world scenarios
- Create JavaScript objects using object literal notation
- Implement and interface with JSON data

Q&A