GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-8-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

- Functions & hoisting
- 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	Slackbot Lab / Objects & JSON
WEEK 5	Intro to the DOM / Intro to jQuery
WEEK 6	Ajax & APIs / Asynchronous JavaScript & Callbacks

EXIT TICKET QUESTIONS

1. I got the message “can’t automatically merge” on GitHub when I tried to create a pull request
2. Browser versions - when to use Babel vs Modernizr
3. What is the functional difference between the 3 function declaration styles? Why are there three?
4. Parameters and Methods within Objects
5. How to do the bonus questions with DOM.
6. How do I submit homework?

SUBMIT HOMEWORK: STEP 1

In Finder:

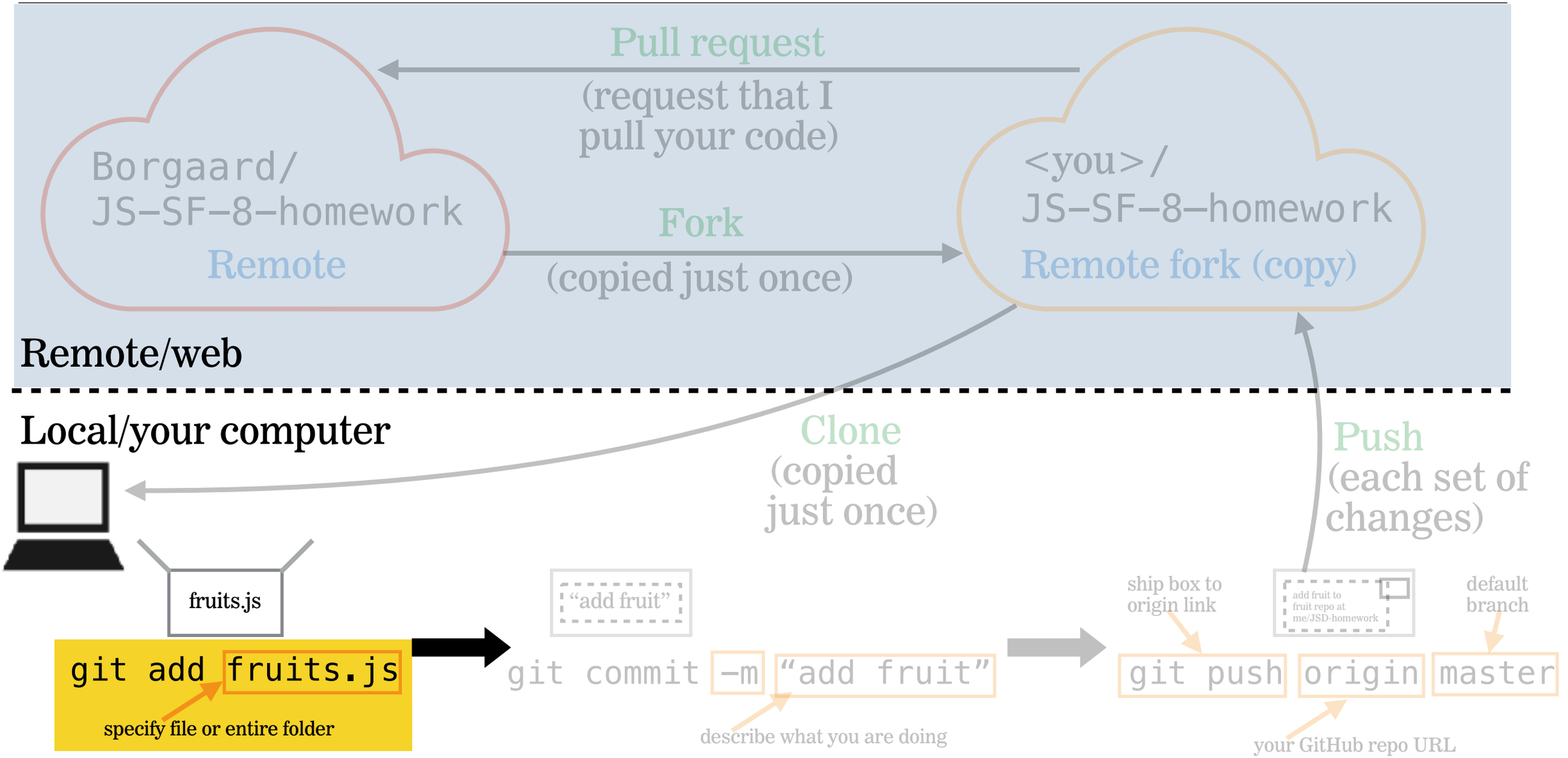
- navigate to *`firstname-username`* folder (example: `Sasha-svodnik`)
- copy your completed Homework-2 folder from last Wednesday into your *`firstname-username`* folder.

SUBMIT HOMEWORK: STEP 2

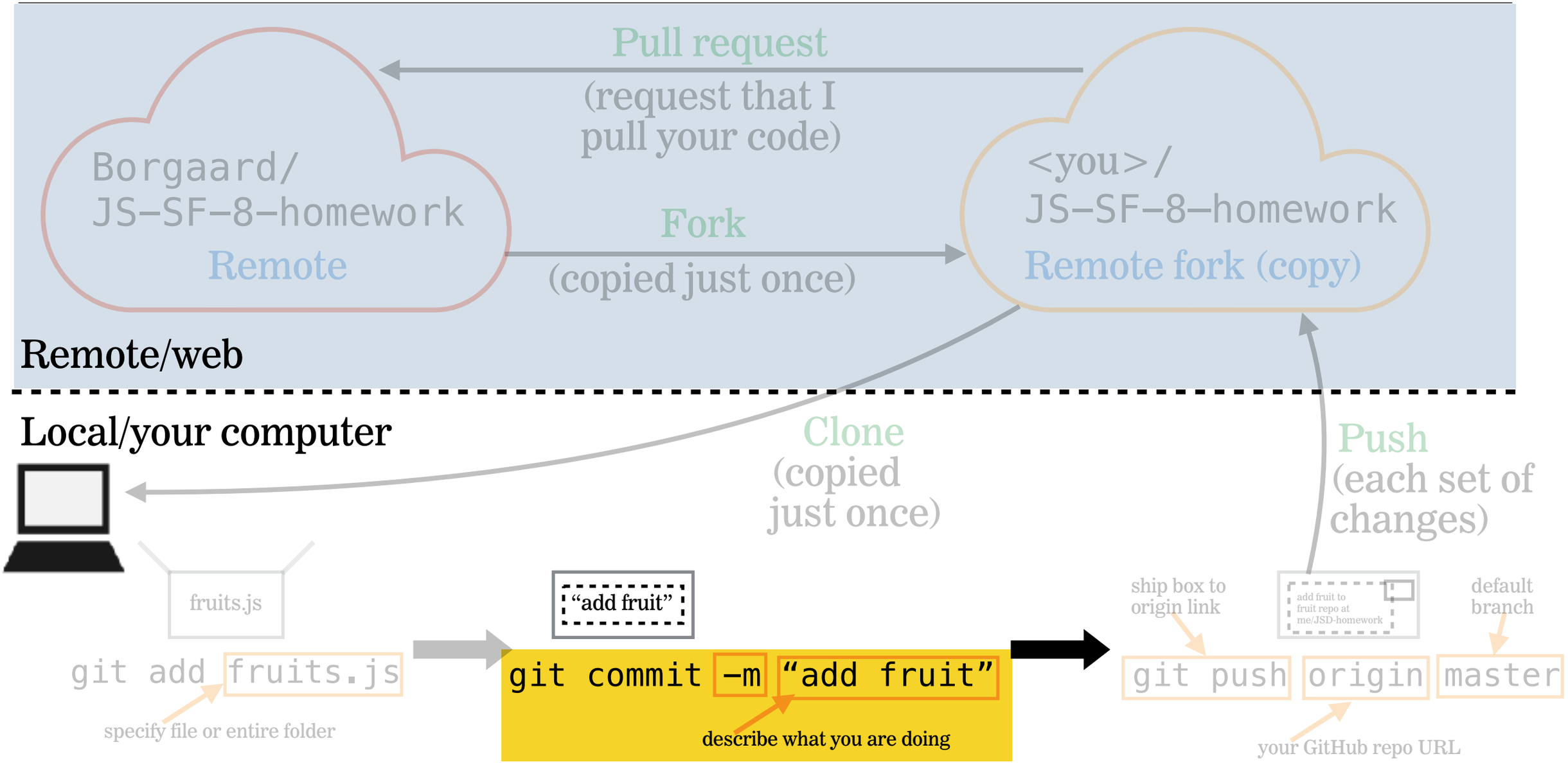
In Terminal:

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

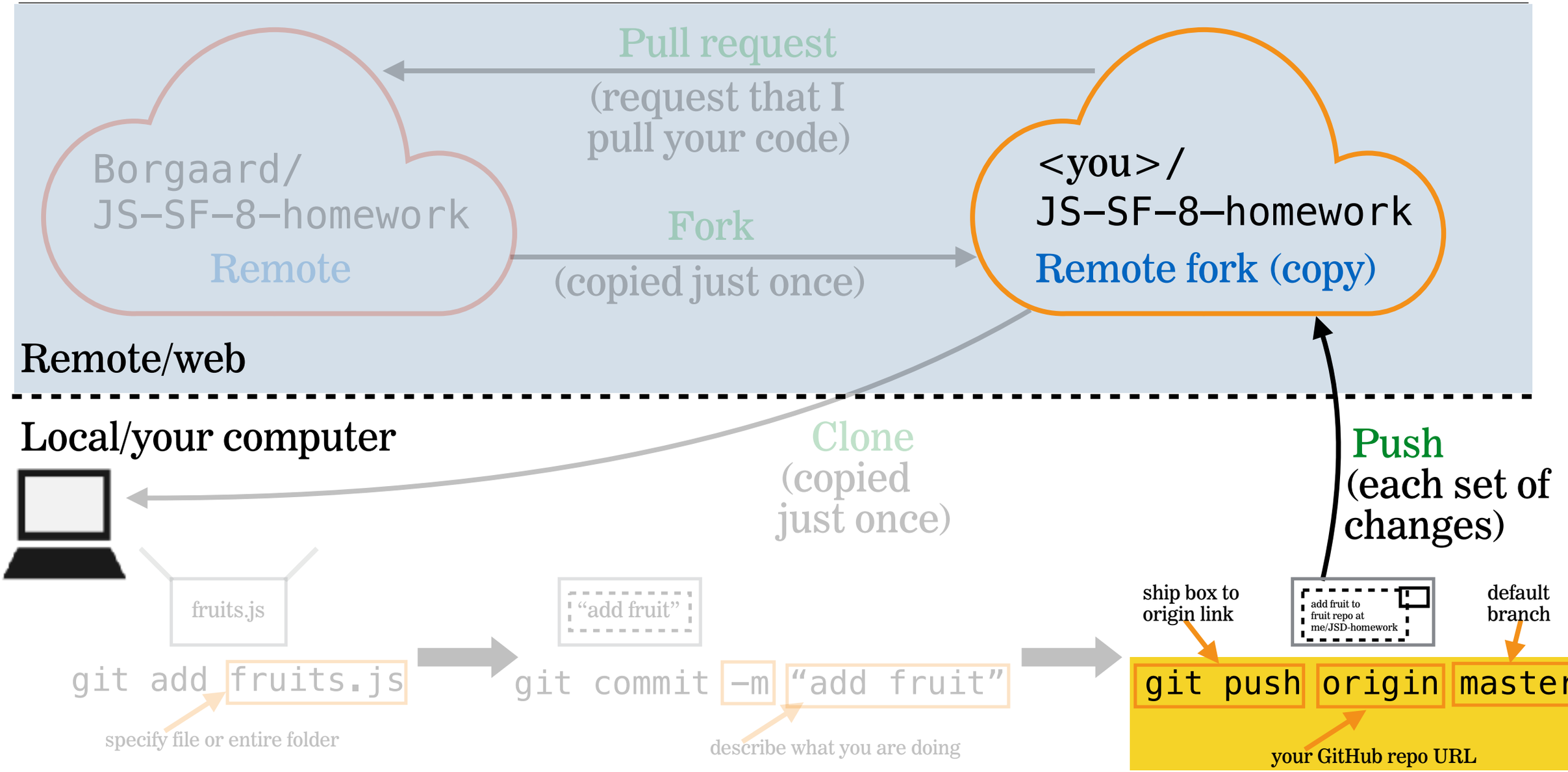
USING THE JS-SF-8-HOMEWORK REPO



USING THE JS-SF-8-HOMEWORK REPO



USING THE JS-SF-8-HOMEWORK REPO

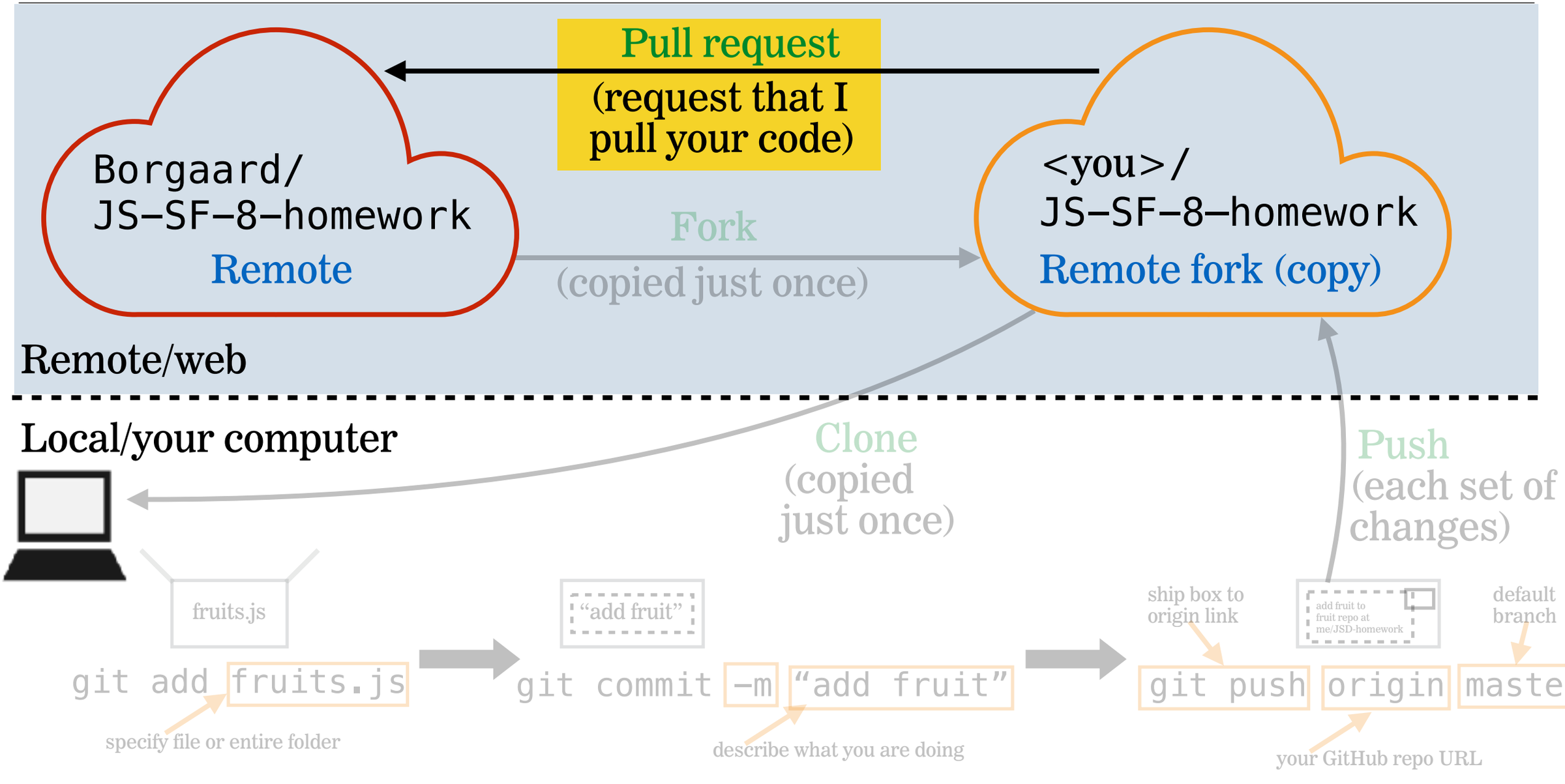


SUBMIT HOMEWORK: STEP 3

In Browser:

- Go to your fork of `JS-SF-8-homework` on github.com
- click **New pull request**
- click **Create pull request**
- click **Create pull request** (again)

USING THE JS-SF-8-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!

HOISTING

- JavaScript's behavior of moving some declarations to the top of a scope.
- This means that you are able to use some functions or variables before they have been declared.

VARIABLES DECLARED WITH `var` ARE HOISTED

CODE AS WRITTEN

```
console.log("Hello!");
var x = "What's up?";
console.log(x);
```

CODE AS INTERPRETED BY PARSER

parser hoists declaration of x to the top of the scope

```
var x;
console.log("Hello!");
x = "What's up?";
console.log(x);
```

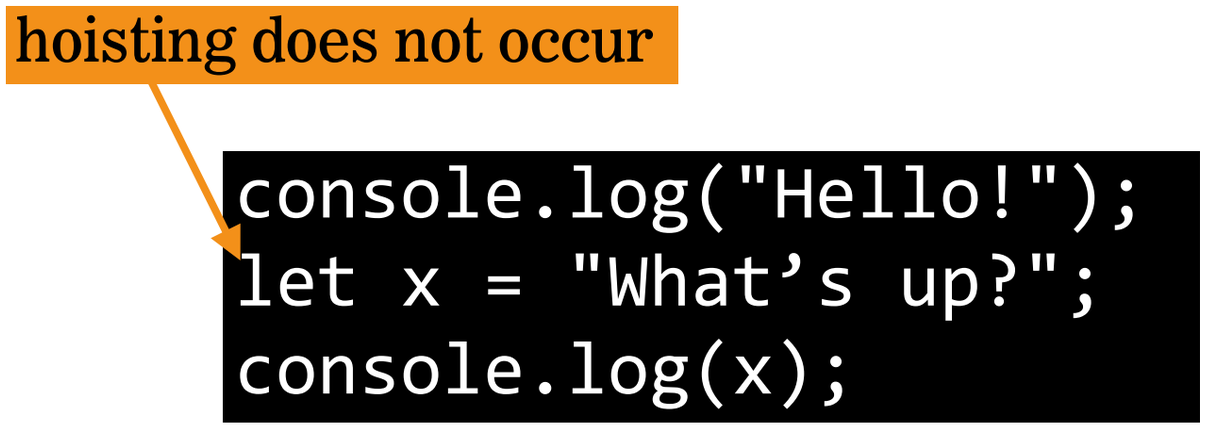
value is then assigned to existing variable

VARIABLES DECLARED WITH `let` AND `const` ARE NOT HOISTED

CODE AS WRITTEN

```
console.log("Hello!");
let x = "What's up?";
console.log(x);
```

CODE AS INTERPRETED BY PARSER



DECLARING OUT OF ORDER

`let`

parser does not know that variable exists

```
console.log(x);
> ReferenceError “x is not defined”
let x = "What’s up?";
console.log(x);
> “What’s up?”
```

`var`

parser knows that variable exists, but no value has been assigned

```
console.log(x);
> undefined
var x = "What’s up?";
console.log(x);
> “What’s up?”
```

FUNCTIONS AND HOISTING

CODE AS WRITTEN

```
foo();
bar();
baz();
var foo = function () {
  console.log("this won't run!");
}
let bar = function() {
  console.log("this won’t run!");
}
function baz() {
  console.log("this will run!");
}
```

CODE AS INTERPRETED BY PARSER

```
var foo;
function bar() {
  console.log("this will run!");
}
foo();
bar();
baz();
foo = function () {
  console.log("this won't run!");
}
let baz = function() {
  console.log("this won’t run!");
}
```

FUNCTIONS AND HOISTING – RESULTS

variable name declared with `var` is hoisted

variable name declared with `let` is **not** hoisted

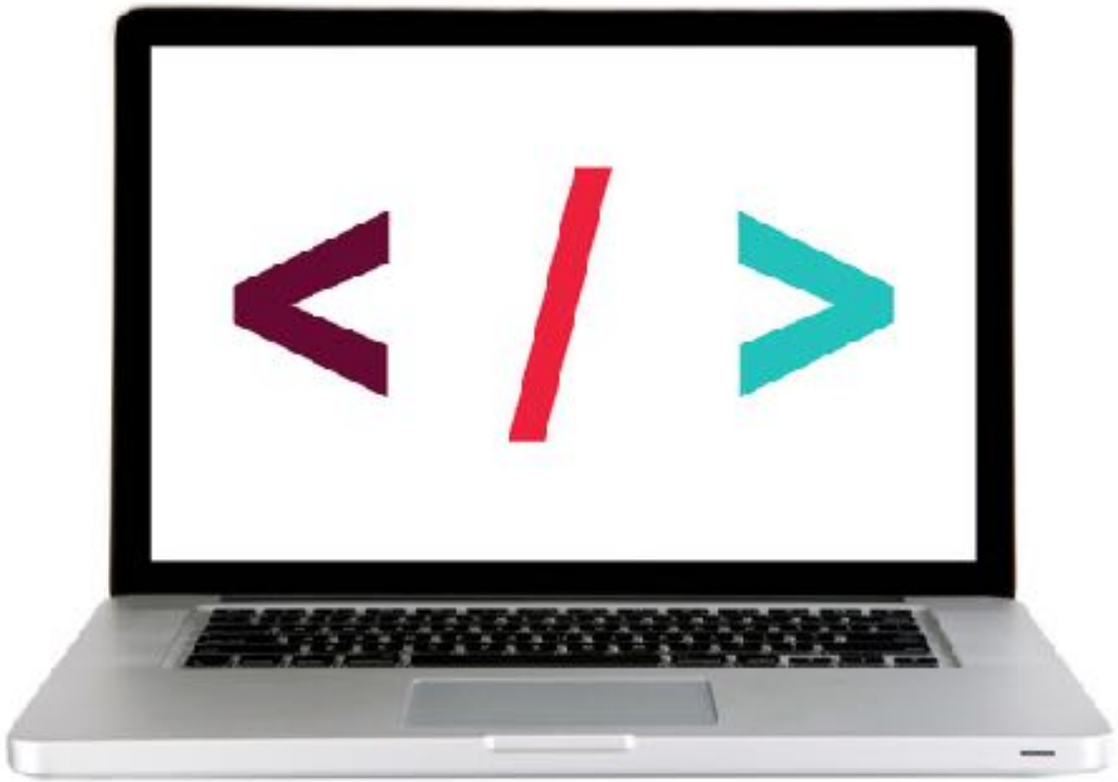
function declaration name **and value** are hoisted

```
foo();
> TypeError "foo is not a function"
bar();
> ReferenceError “bar is not defined”
baz();
> "this will run!”
var foo = function () {
  console.log("this won't run!");
}
let bar = function() {
  console.log("this won’t run!");
}
function baz() {
  console.log("this will run!");
}
```

HOISTING SUMMARY

statement type	name hoisted?	value hoisted?
function declaration	✓	✓
expression using `let`	✗	✗
expression using `var`	✓	✗

LET'S TAKE A CLOSER LOOK



HOISTING BEST PRACTICES

- Don’t rely on hoisting!
- Declare all variables at the top of the scope
- Declare all functions at the top of the scope

```
let x = "this won’t run";
let y = "this will run";
var foo = function () {
  console.log(y);
}
let bar = function() {
  console.log(y);
}
function baz() {
  console.log(y);
}
foo();
bar();
baz();
```

EXERCISE — HOISTING

KEY OBJECTIVE

- Create a program that hoists variables

TYPE OF EXERCISE

- Groups of 3

EXECUTION

2 min

1. Examine the code on the whiteboard.
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

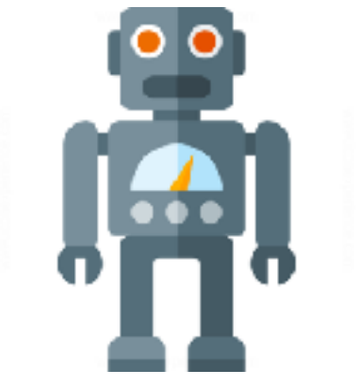
- **Bot**: A script programmed to interact with users as if it's a person
  - Slackbot
  - PlusPlus
- We will use a framework to create our own bots with interactive behaviors that we specify with our code
- These bots will be members of our class Slack organization

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

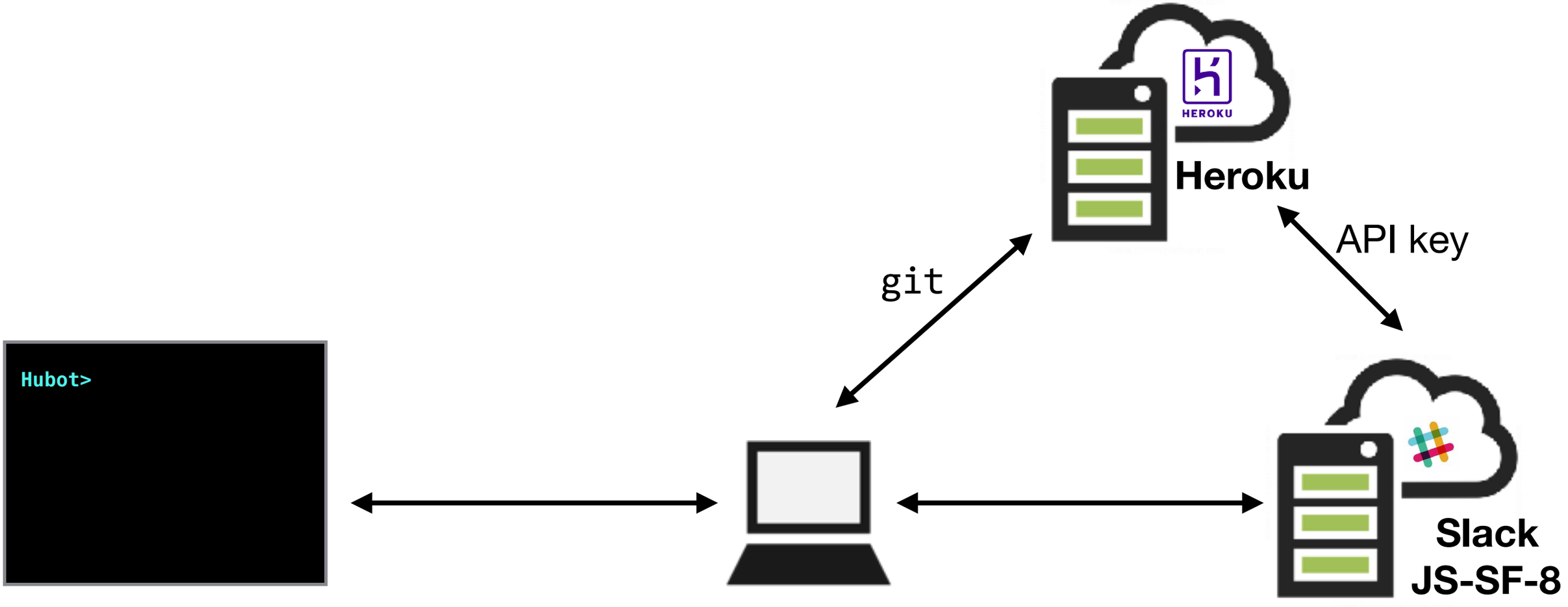
HUBOT vs SLACK BOT vs SLACKBOT

- 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

- `JS-SF-8-resources > 1-slack-bot-install-guide.md`

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 `2-slack-bot-code-samples.md`

UNDERSTANDING THE HUBOT FRAMEWORK

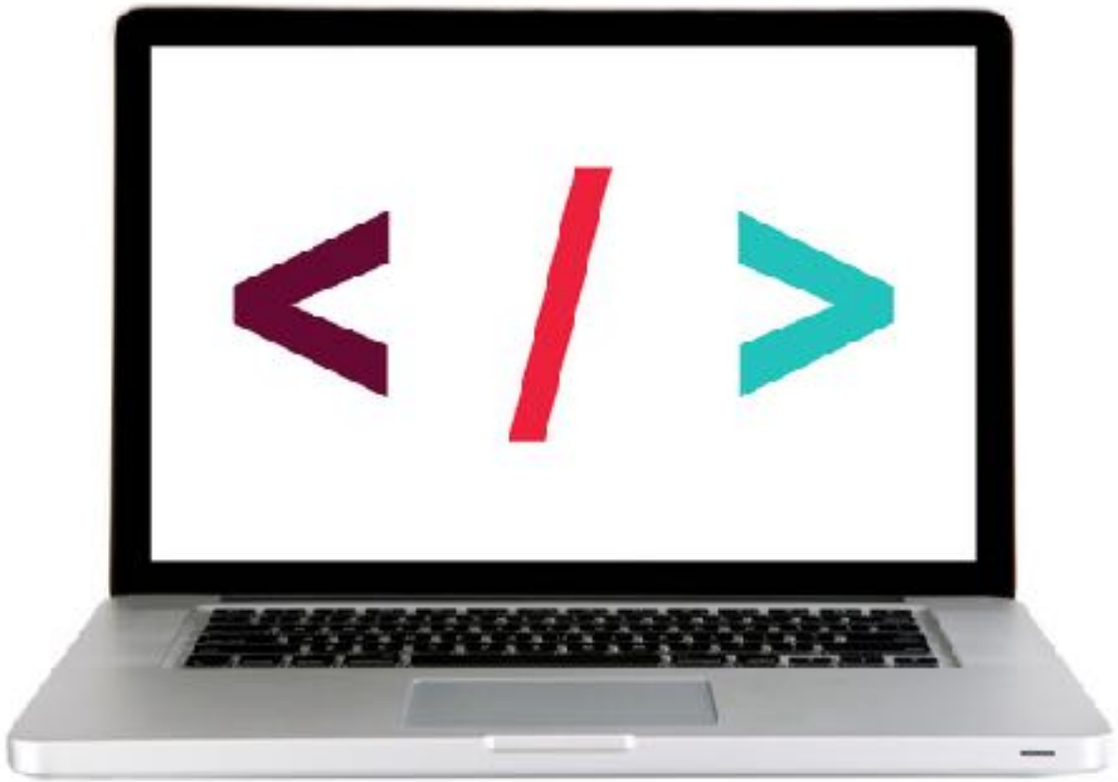
- As a framework, Hubot has its own way of doing things
- The code for your bot behaviors is structured as follows:

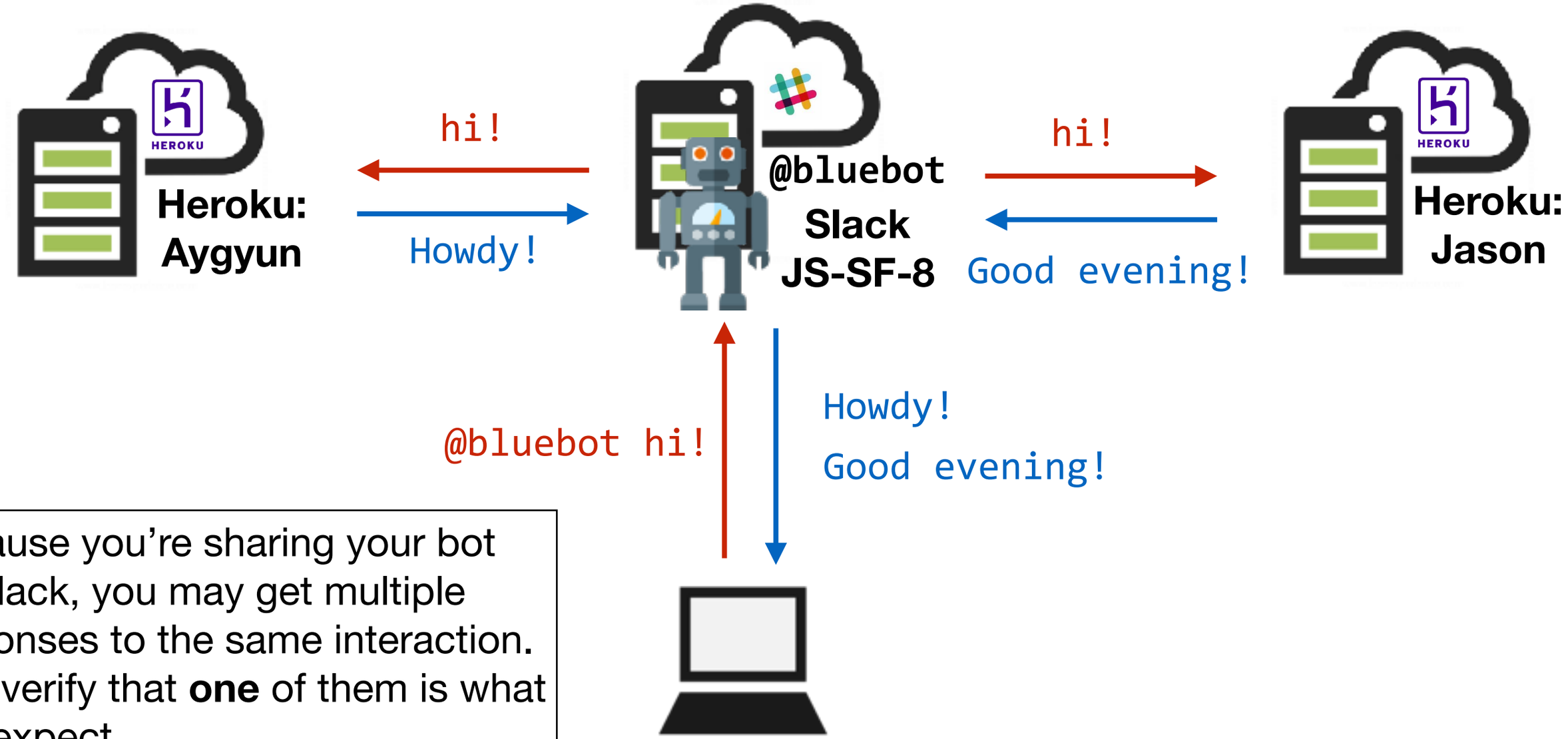
```
module.exports = function(robot) {
  robot.verb(parameter1, function(res) {
    return res.command();
  });
};
```

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 > slackbot.js`

EXECUTION

Until 9:20

1. Uncommenting portions of the sample code in slackbot.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 the `2-slack-bot-code-samples.md` file.
3. Create a plan for what you want your bot to be able to do, pseudocode it, and start building it!
4. BONUS: Experiment with advanced commands documented at https://github.com/github/hubot/blob/master/docs/scripting.md

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

Exit Tickets!

Q&A