GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-10-resources` repo to your computer:
   - Open the terminal
   - `cd` to the `~/Documents/JSD/JS-SF-10-resources` directory
   - Type **`git pull`** and press **return**
2. In your code editor, open the following folder: `Documents/JSD/JS-SF-10-resources/04-scope-objects`

JAVASCRIPT DEVELOPMENT

SCOPE & OBJECTS

LEARNING OBJECTIVES

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

- Determine the scope of local and global variables
- Create a program that hoists variables
- Identify likely objects, attributes, and methods in real-world scenarios
- Create JavaScript objects using object literal notation

AGENDA

- Variable scope
- The `var`, `let`, and `const` keywords
- Hoisting
- Objects

WEEKLY OVERVIEW

WEEK 3	(holiday) / Scope & objects
WEEK 4	Slackbot Lab / JSON & DOM
WEEK 5	The DOM & jQuery / Advanced jQuery

EXIT TICKET QUESTIONS

1. What happens if I just write "return", will that just break out of the function? Does any value get assigned? If so, to what?
2. A little confused about default parameters
3. Is it safe to say that function declaration is the least problematic approach when defining a function due to the fact that it is accessible anywhere throughout the program?
4. I feel like I am understanding the materials but I get stuck trying to solve the exercises. I think it's got to do with thinking like a computer when writing the code to solve the problem. Can you give an example of a question or exercises and demonstrate how you would solve it and write the program?

HOMEWORK REVIEW

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

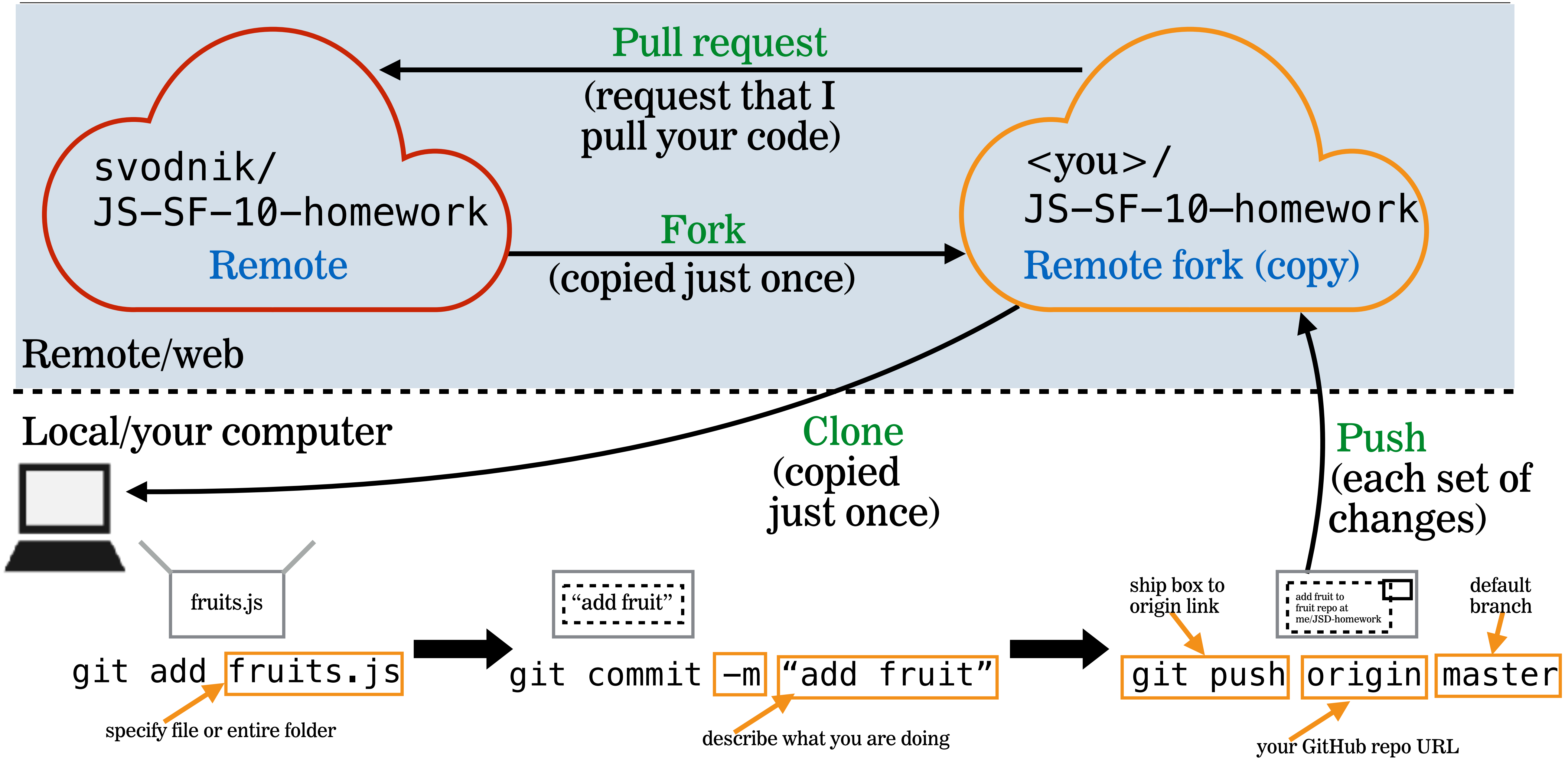
- Groups of 3

TIMING

5 min

1. Take turns showing and explaining your code.
2. Share 1 thing you're excited about being able to accomplish.
3. Have each person in the group note 1 thing they found challenging for the homework. Discuss as a group how you think you could solve each problem.
4. Did you work on the Random Address Generator bonus exercise? Show your group what you did!

USING THE JS-SF-10-HOMEWORK REPO



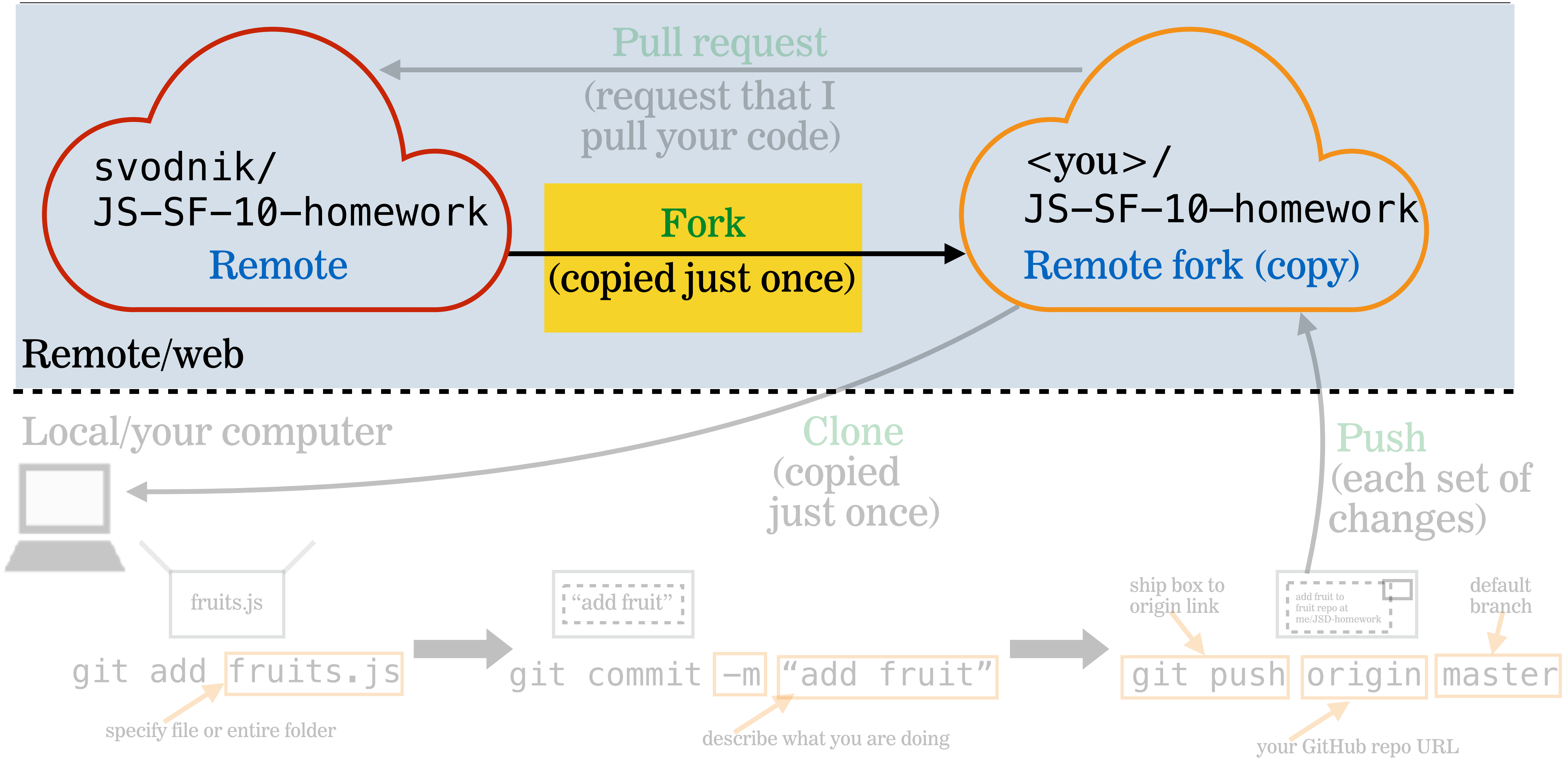
```
git add fruits.js
git commit -m “add fruit”
git push origin master
```

SUBMIT HOMEWORK: SETUP (ONE TIME ONLY)

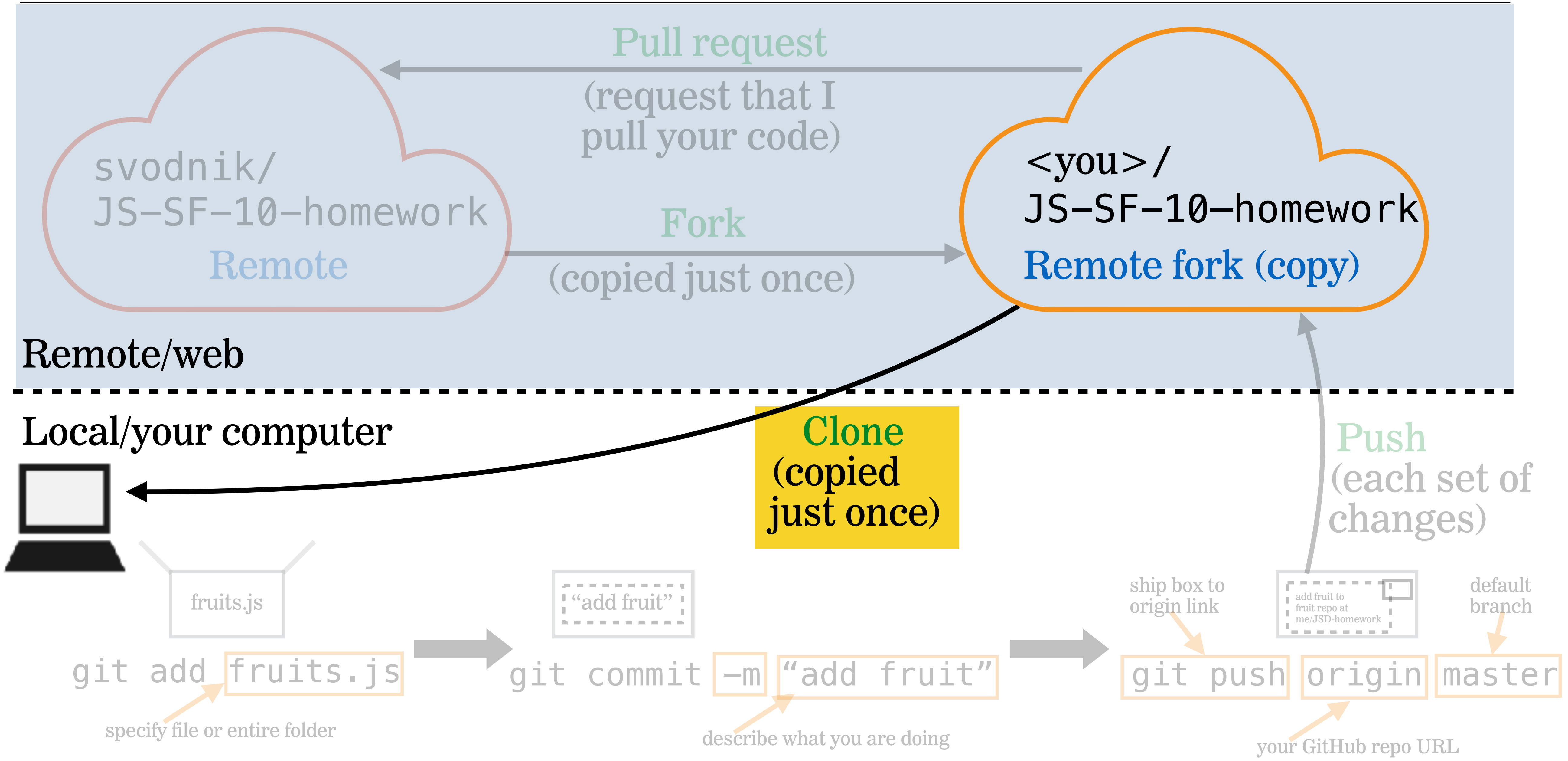
On github.com:

- Open https://github.com/svodnik/JS-SF-10-homework
- Fork this repo to your GitHub account
- Clone your fork to your computer, within your JSD folder

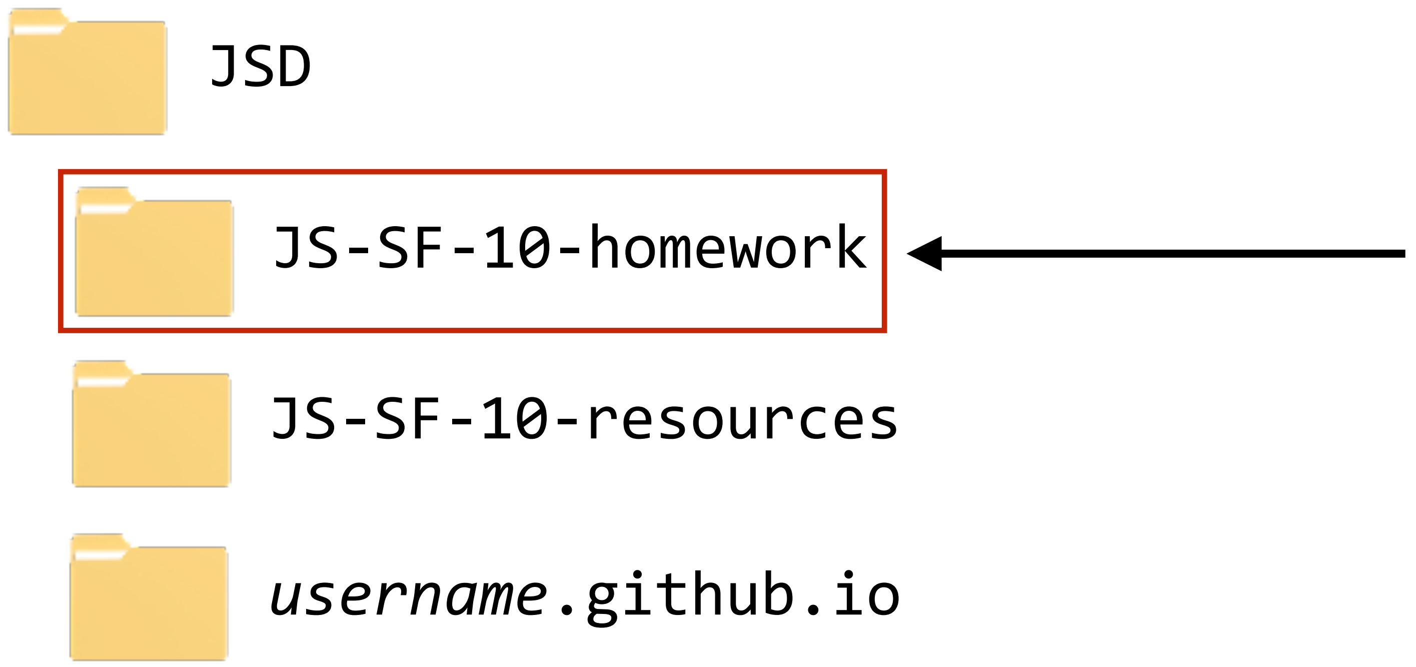
USING THE JS-SF-10-HOMEWORK REPO



USING THE JS-SF-10-HOMEWORK REPO



HOMEWORK FOLDER LOCATION

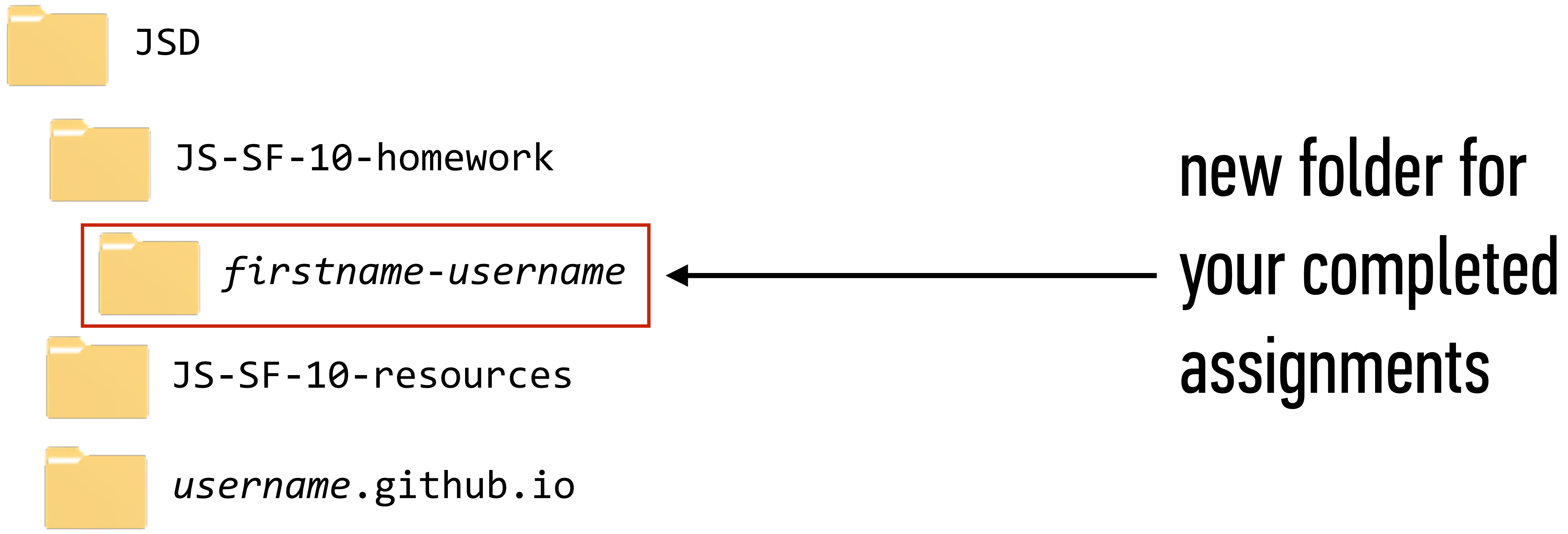


new folder for your clone of the homework repo

SUBMIT HOMEWORK: SETUP (CONTINUED)

- Within your new `JS-SF-10-homework` folder, create a new subfolder and name it your first name, a hyphen, and your github name. For instance, Sasha's folder would be `Sasha-svodnik`.

PERSONAL ASSIGNMENTS FOLDER LOCATION



SETUP DONE!

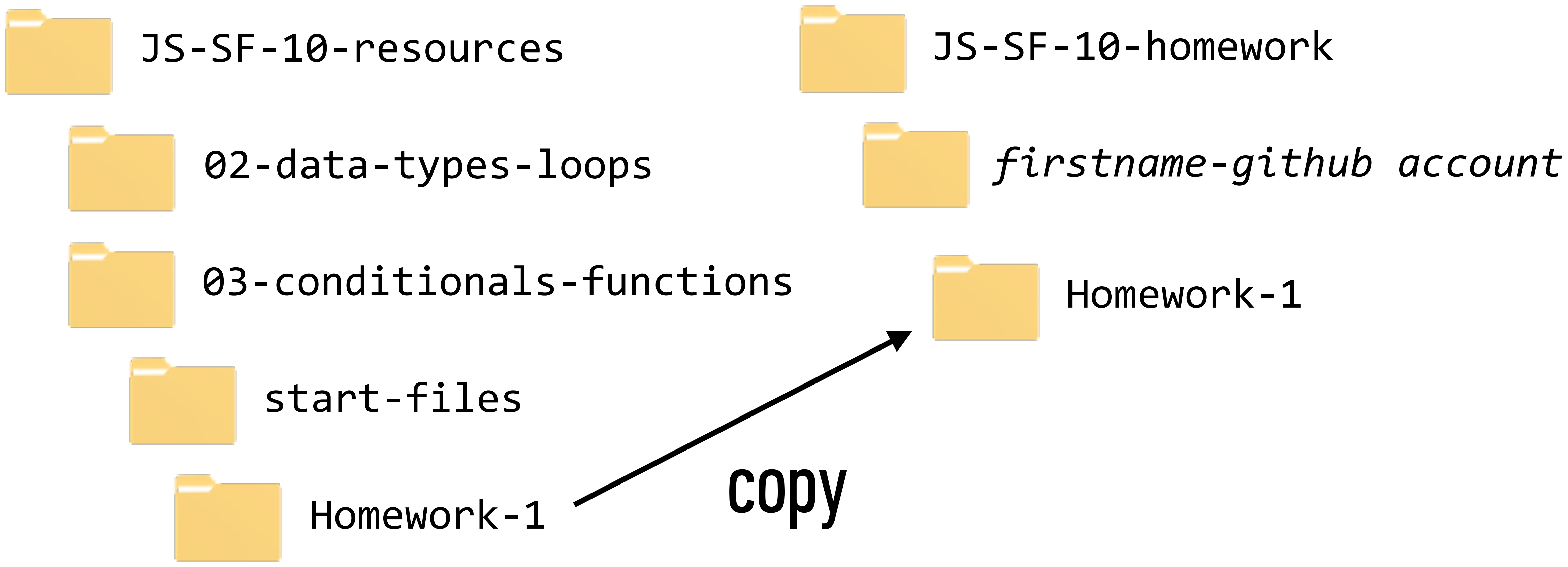
- Reminder: Now that you’ve completed the preceding setup, you never have to do it again!
- Each time you submit homework for the rest of this course, you’ll repeat only the steps that follow.

SUBMIT HOMEWORK: STEP 1

In Finder:

- navigate to `*firstname-username*` folder (example: `Sasha-svodnik`)
- copy your completed `Homework-1` folder from last Wednesday into your *`firstname-username`* folder.

SUBMIT HOMEWORK: STEP 1 ILLUSTRATION

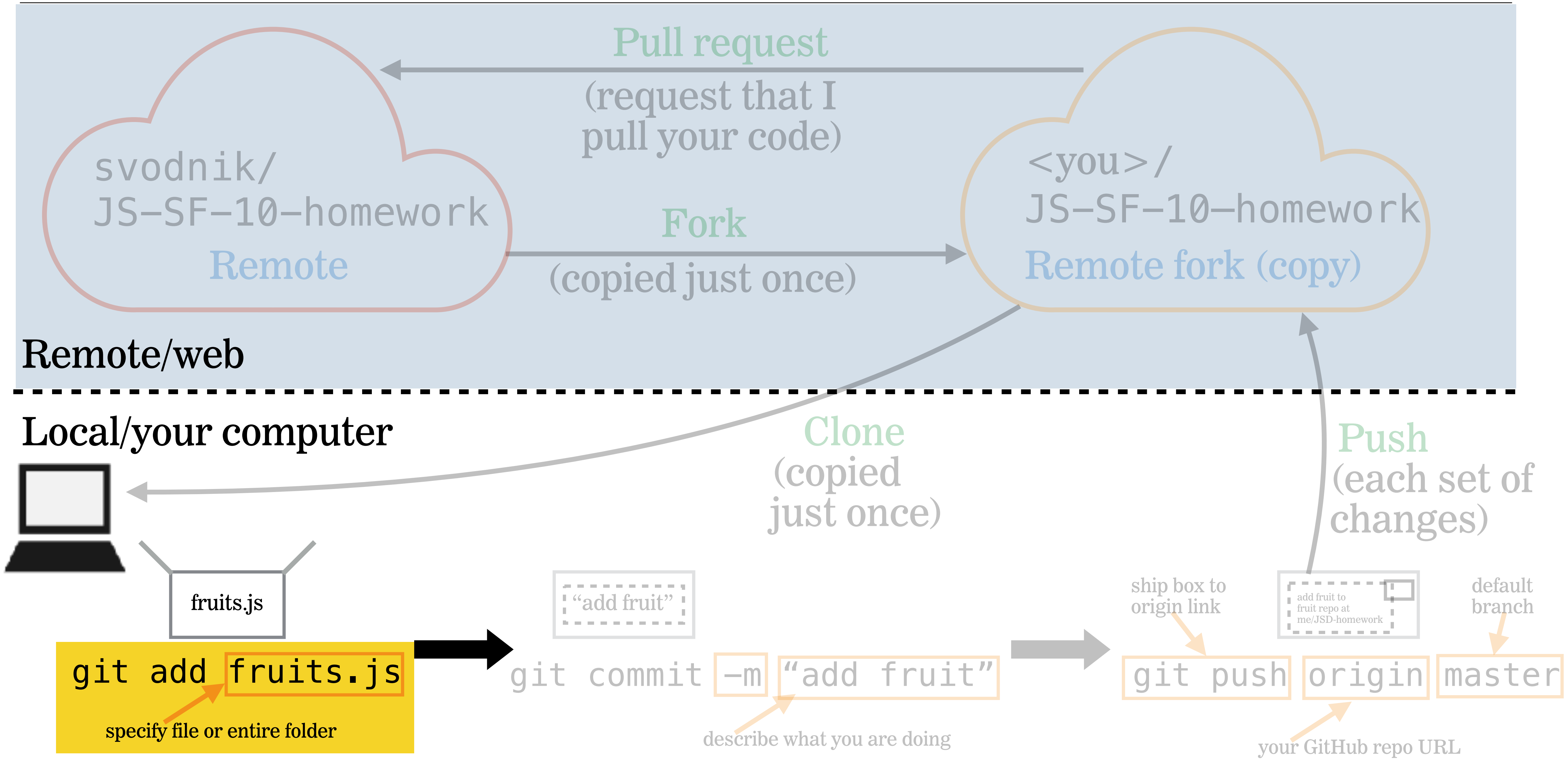


SUBMIT HOMEWORK: STEP 2

In Terminal:

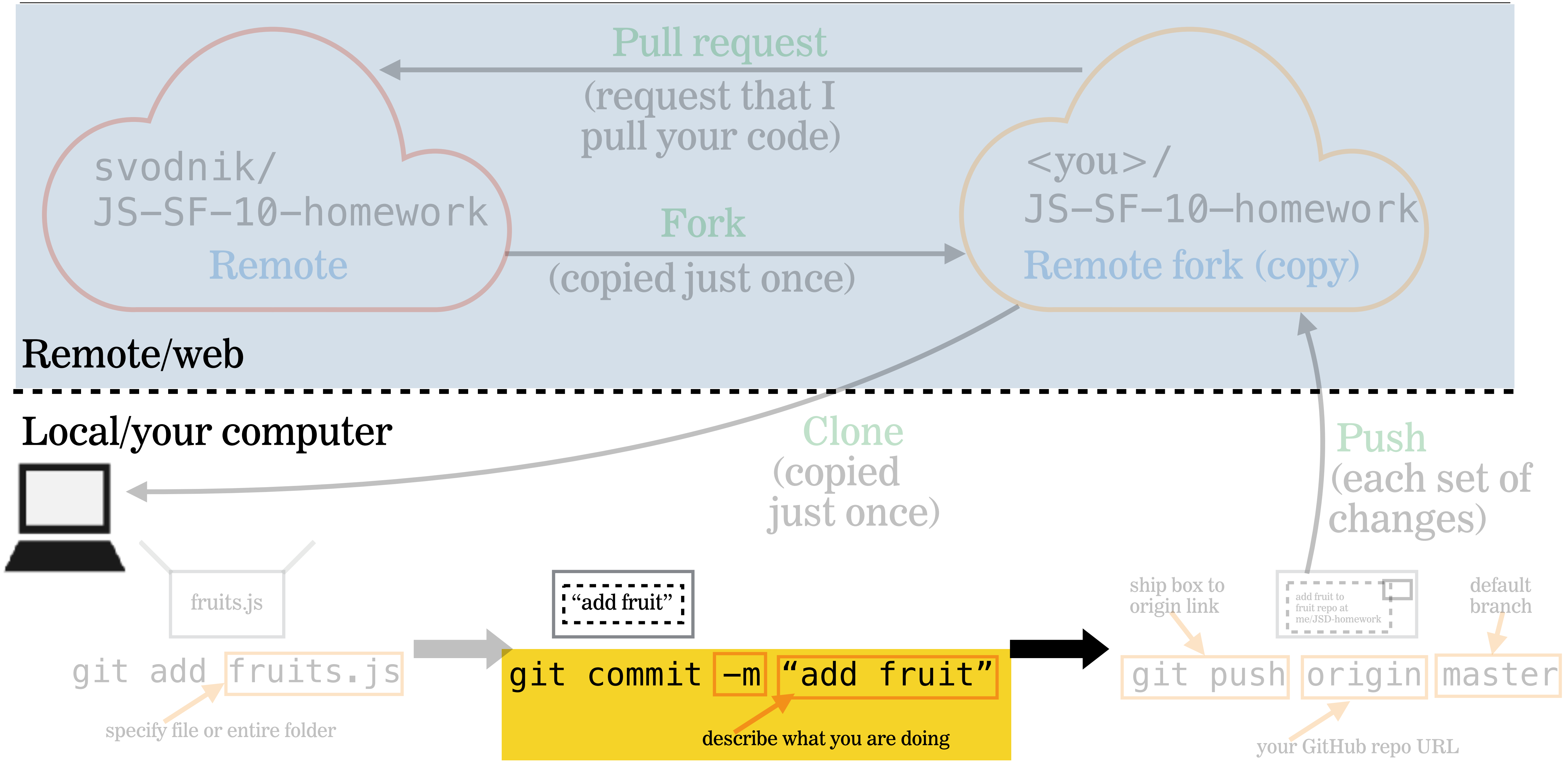
- navigate to *`firstname-username`* folder
- `git add .`
- `git commit -m “submitting Homework 1”`
- `git push origin master`

USING THE JS-SF-10-HOMEWORK REPO



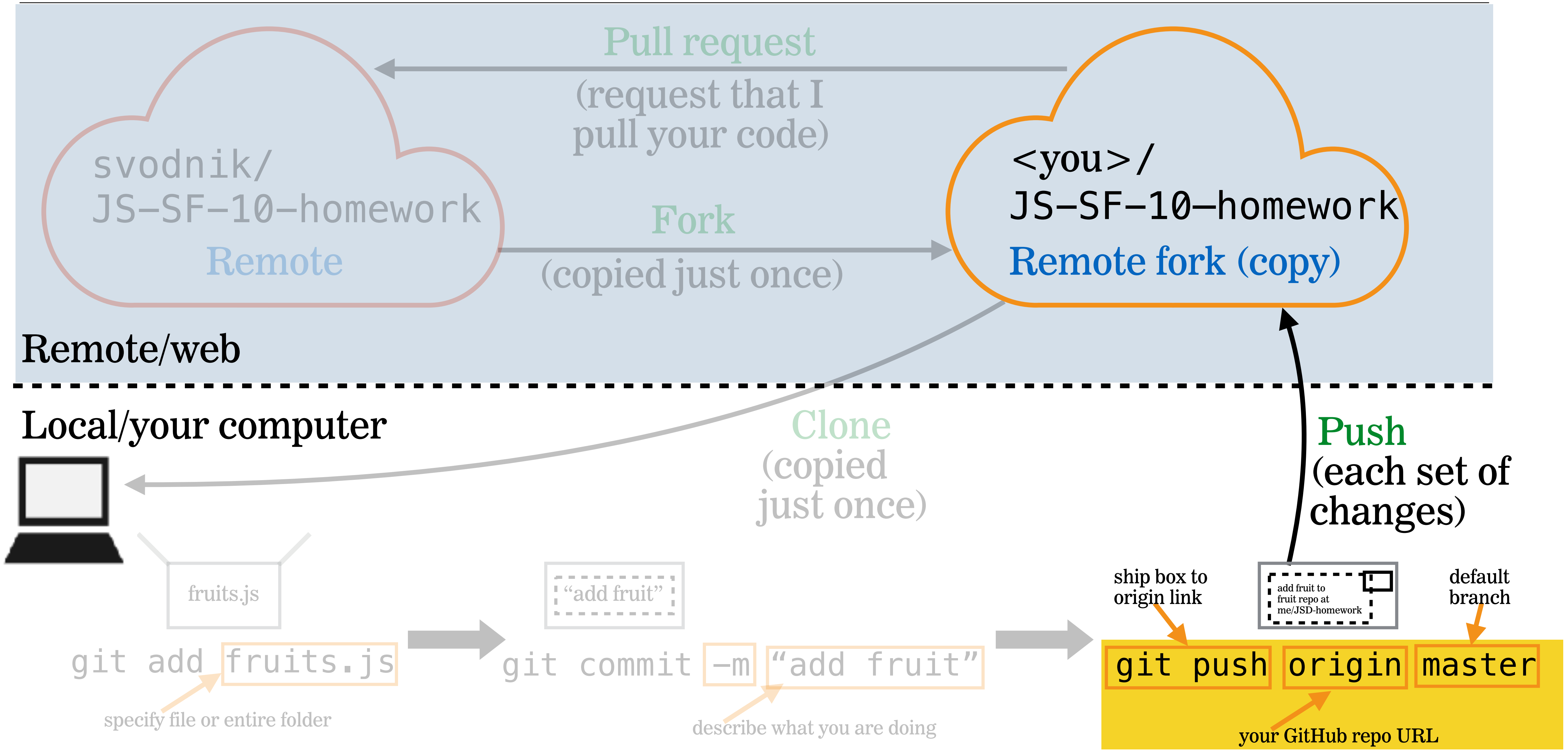
```
git add fruits.js
git commit -m "add fruit"
git push origin master
```

USING THE JS-SF-10-HOMEWORK REPO



```
git add fruits.js
git commit -m “add fruit”
git push origin master
```

USING THE JS-SF-10-HOMEWORK REPO

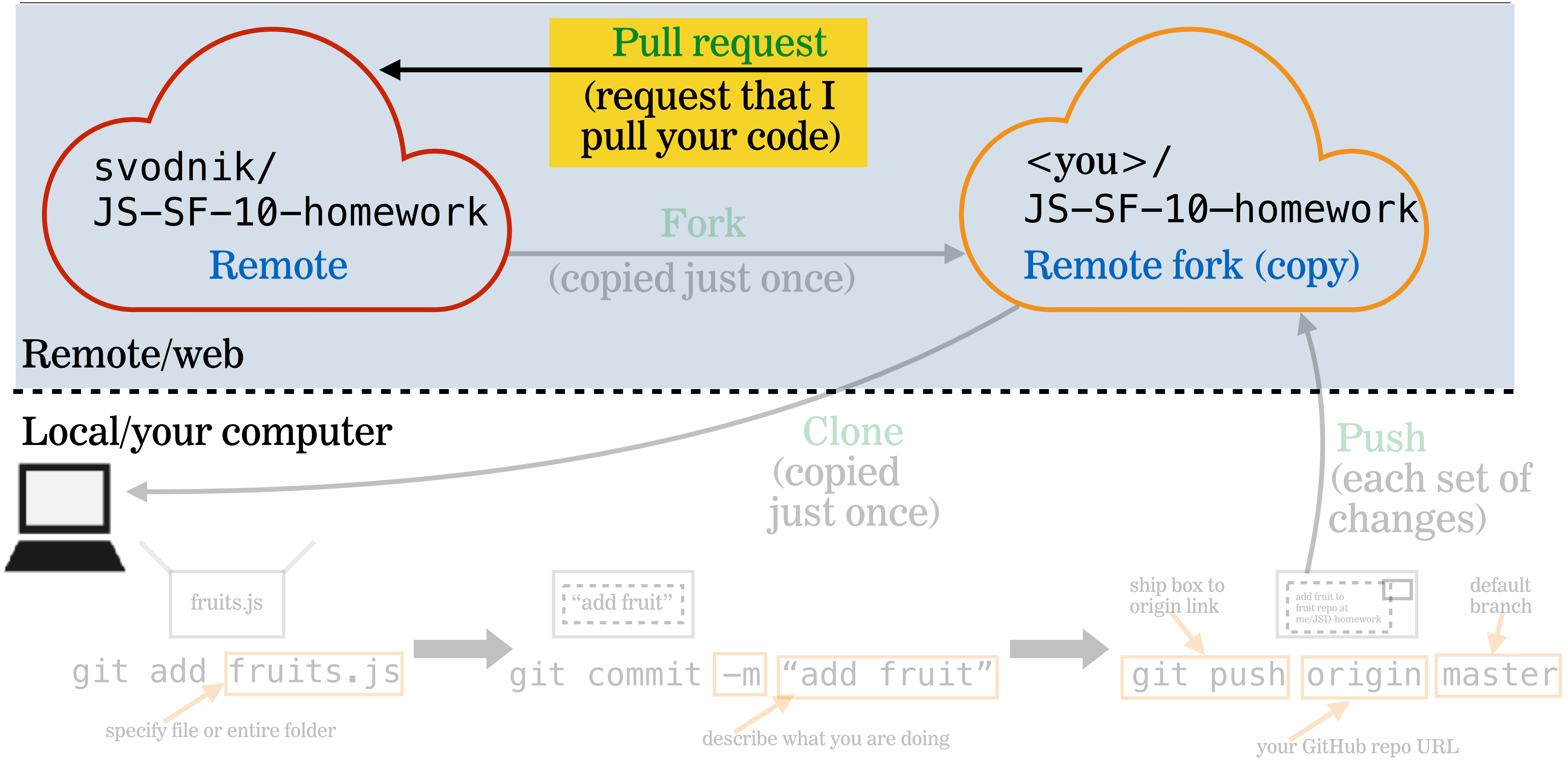


SUBMIT HOMEWORK: STEP 3

In Browser:

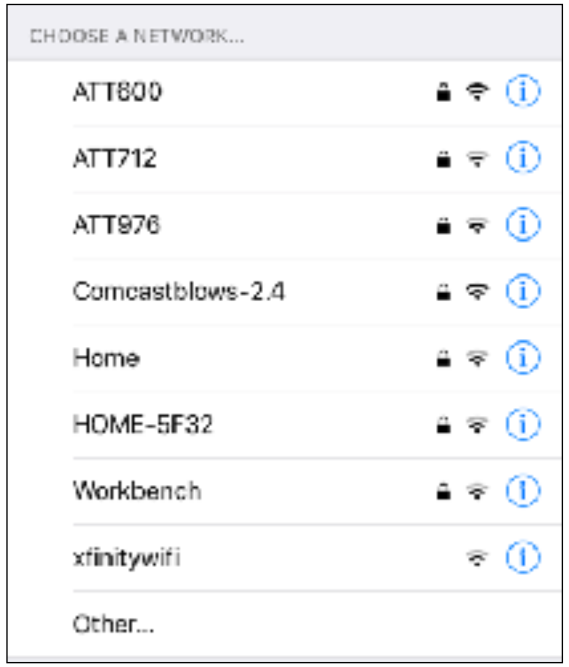
- Go to your fork of `JS-SF-10-homework` on github.com
- click **New pull request**
- click **Create pull request**
- click **Create pull request** (again)

USING THE JS-SF-10-HOMEWORK REPO

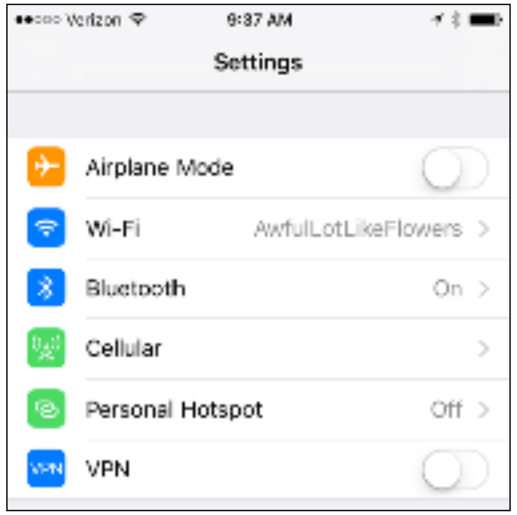


Why do we use different networks to connect to the Internet when we're in different places?

- home
- GA
- in a car
- on BART/MUNI


CHOOSE A NETWORK...

ATT800
ATT712
ATT976
Comcastblows-2.4
Home
HOME-5F32
Workbench
xfinitywifi
Other...


●●○○○ Verizon 9:37 AM

Settings

Airplane Mode	
Wi-Fi	AwfulLotLikeFlowers
Bluetooth	On
Cellular	
Personal Hotspot	Off
VPN	

SCOPE

SCOPE

- Describes the set of variables you have access to

GLOBAL SCOPE

- A variable declared outside of a function is accessible everywhere, even within functions. Such a variable is said to have **global scope**.

a variable declared outside of the function is in the global scope

```
let temp = 75;
function predict() {
  console.log(temp); // 75
}
console.log(temp); // 75
```

LOCAL SCOPE

- A variable declared within a function is not accessible outside of that function. Such a variable is said to have **local scope**.

```
let temp = 75;
function predict() {
  let forecast = ‘Sun’;
  console.log(temp + " and " + forecast); // 75 and Sun
}
console.log(temp + " and " + forecast);
// ‘forecast’ is undefined
```

a variable declared within a function is in the local scope of that function

a local variable is not accessible outside of its local scope

BLOCK SCOPE

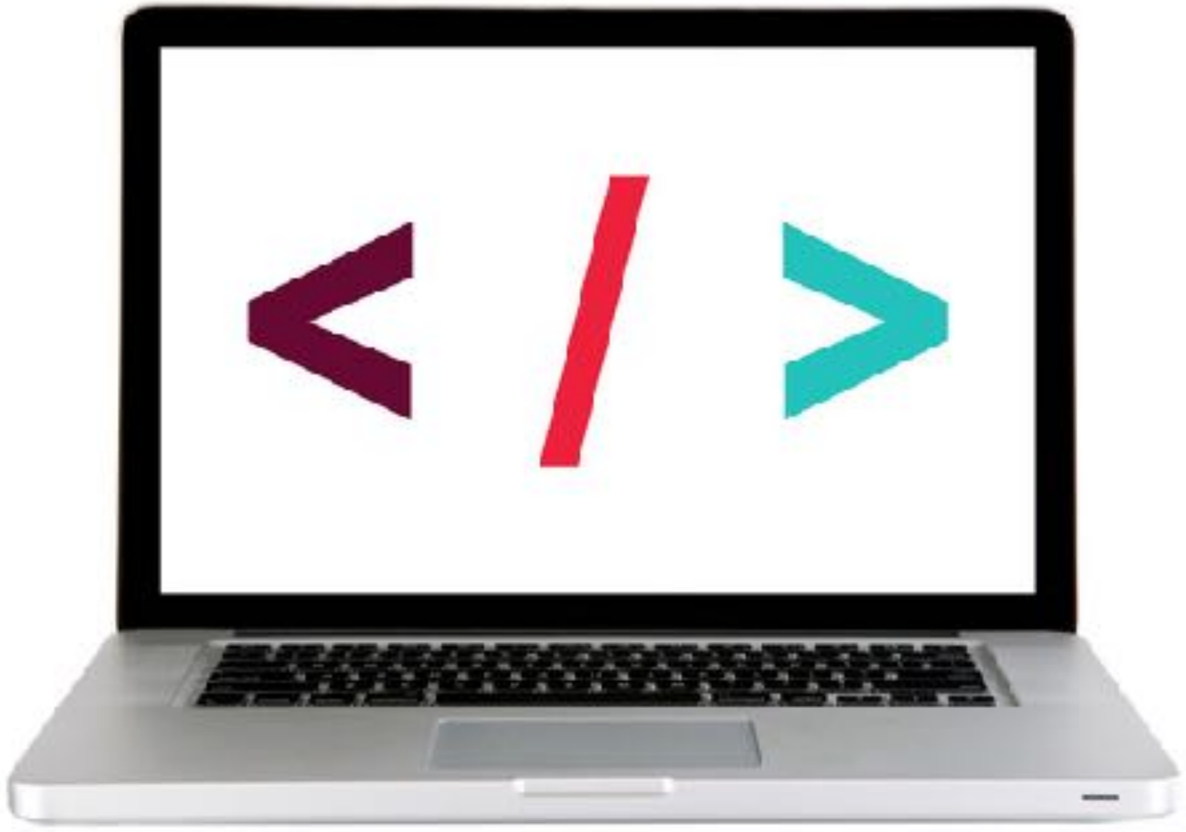
- A variable created with `let` or `const` creates local scope within any block, including blocks that are part of loops and conditionals.
- This is known as **block scope**.

```
let temp = 75;
if (temp > 70) {
  let forecast = ‘It’s gonna be warm!’;
  console.log(temp + “! ” + forecast); // 75! It’s gonna be warm!
}
console.log(temp + “! " + forecast); // ‘forecast’ is undefined
```

`let` creates a local variable within any block, such as an `if` statement

a variable with block scope is not accessible outside of its block

LET'S TAKE A CLOSER LOOK



EXERCISE — SCOPE

EXERCISE

KEY OBJECTIVE

- Determine the scope of local and global variables

TYPE OF EXERCISE

- Turn and Talk

EXECUTION

3 min

1. Describe the difference between global and local scope
2. Collaborate to write code that includes at least one variable with local scope and one variable with global scope

LAB — SCOPE

KEY OBJECTIVE

- Determine the scope of local and global variables

TYPE OF EXERCISE

- Pairs

LOCATION

- `starter code > 3-scope-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 > main.js` to complete parts A and B.

`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

```
let 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];
```

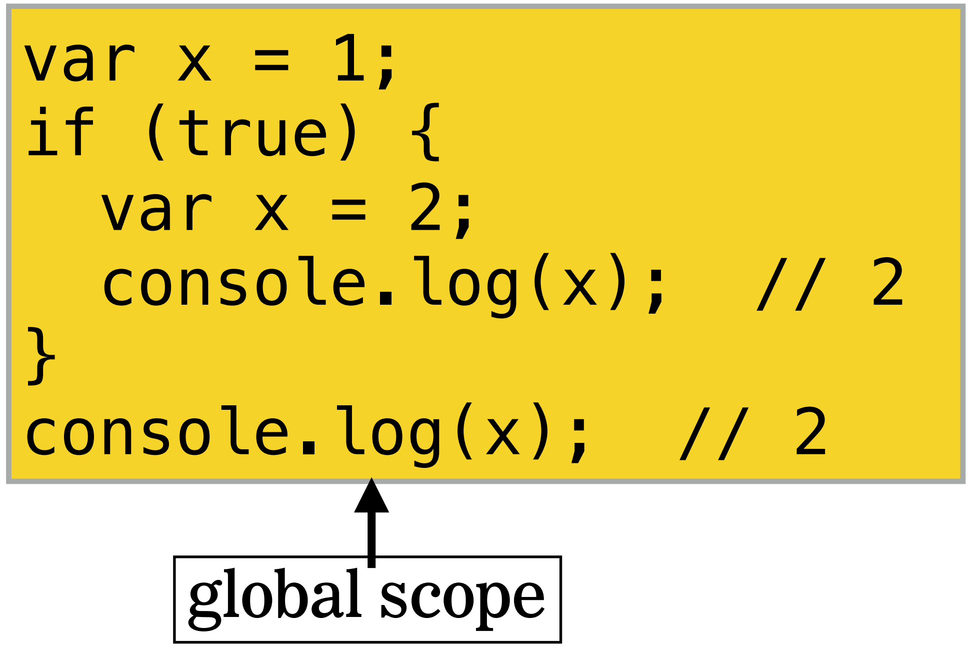
`const`

- 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

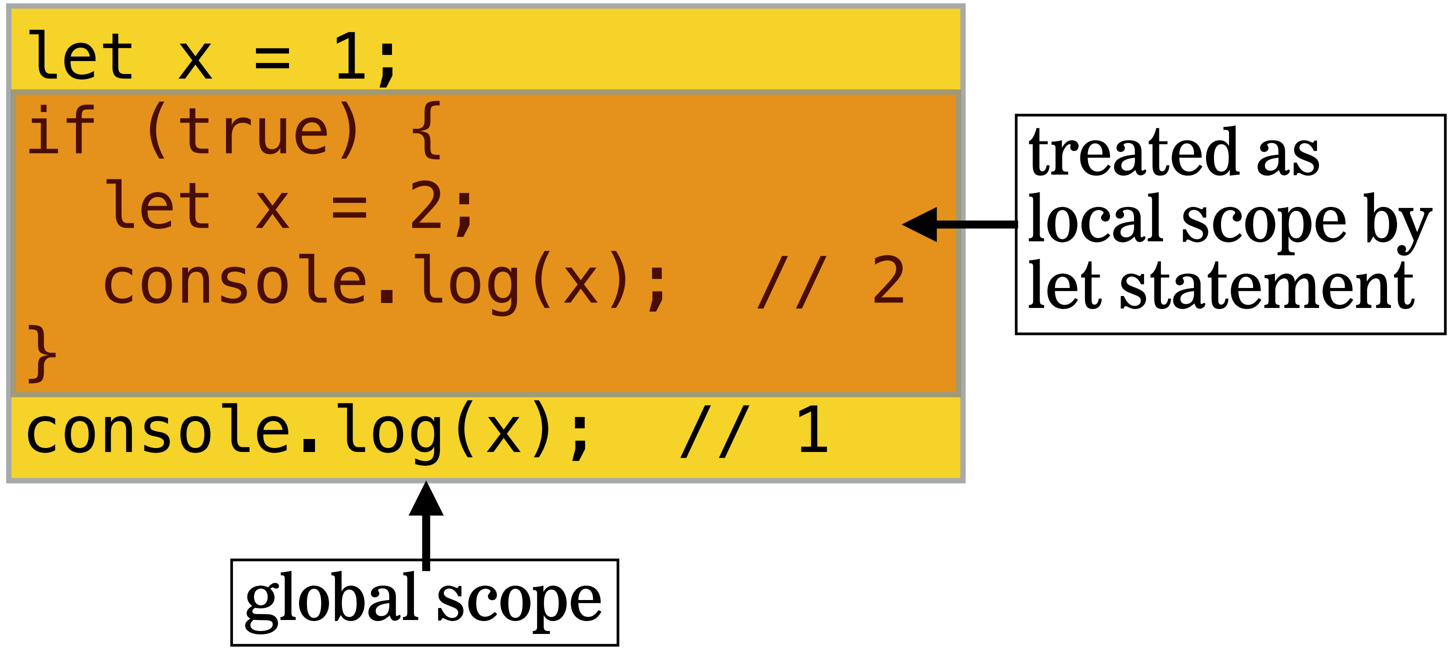
```
const salesTax = 0.0875;
```

`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
```

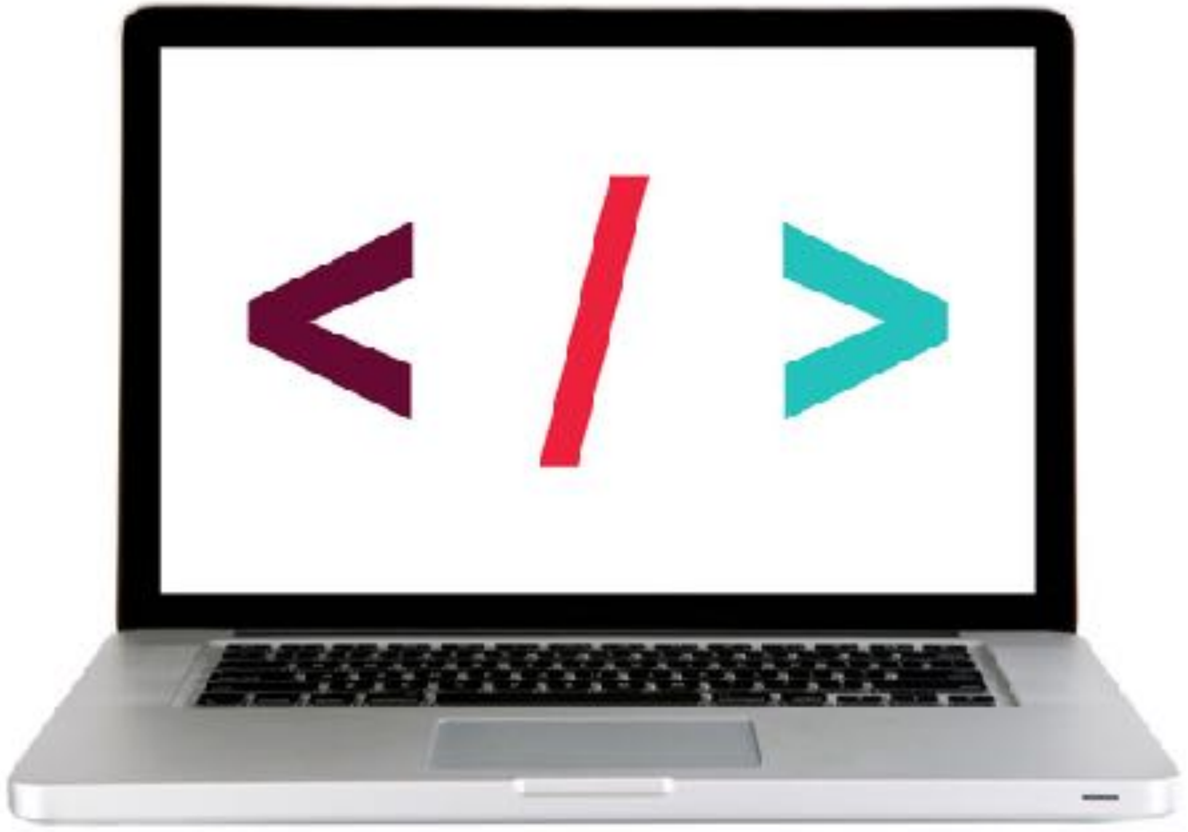


```
let x = 1;
if (true) {
  let x = 2;
  console.log(x);  // 2
}
console.log(x);  // 1
```

`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 `let` and `const` in class

LET'S TAKE A CLOSER LOOK



EXERCISE — VAR, LET, AND CONST

EXERCISE

KEY OBJECTIVE

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

TYPE OF EXERCISE

- Individual or pairs

EXECUTION

2 min

1. Draw the table shown on the next slide, which compares a few aspects of `var`, `let`, and `const` usage.
2. Complete the table without referring to your cheat sheet.

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

keyword	where does it create local scope?	can you change the value in the current scope?	which browsers support it? (modern or all)
`var`			
`let`			
`const`			

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

keyword	where does it create local scope?	can you change the value in the current scope?	which browsers support it? (modern or all)
`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

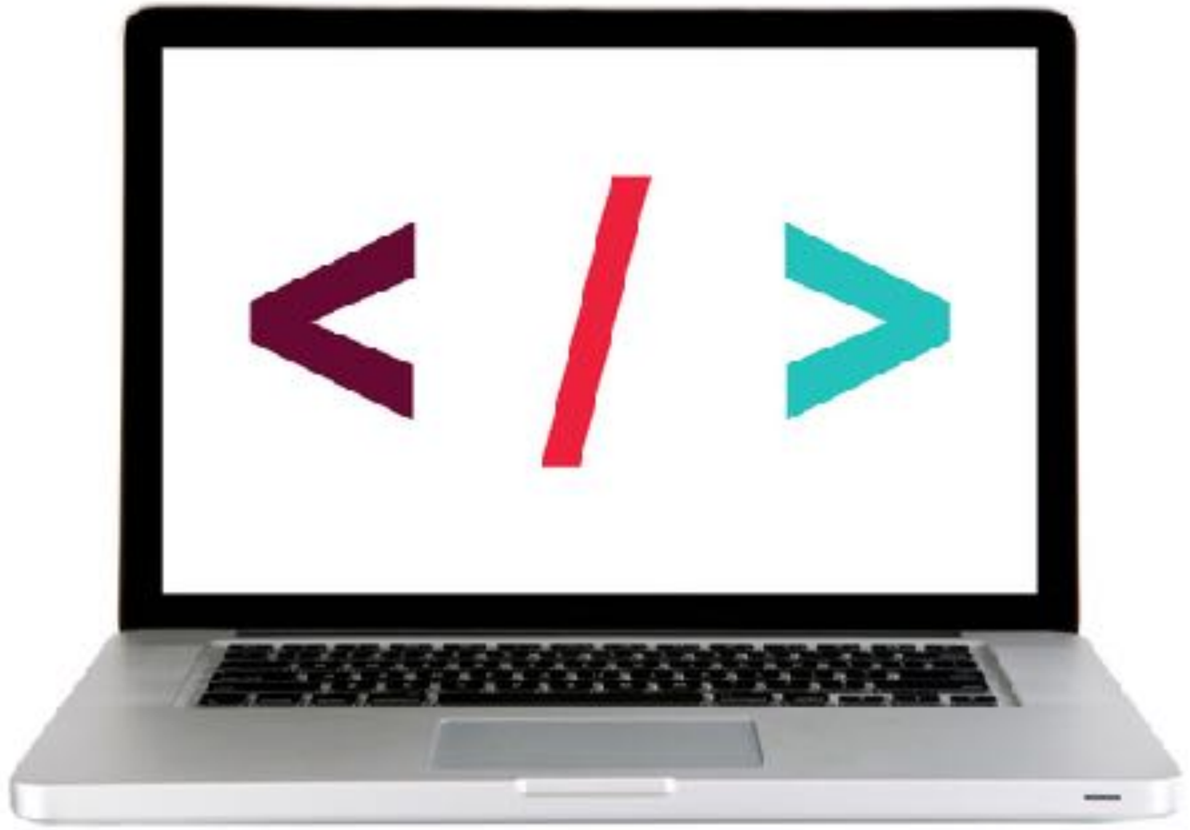
FUNCTIONS AND HOISTING

- 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

EXERCISE

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.

OBJECTS

OBJECTS ARE A SEPARATE DATA TYPE

STRING

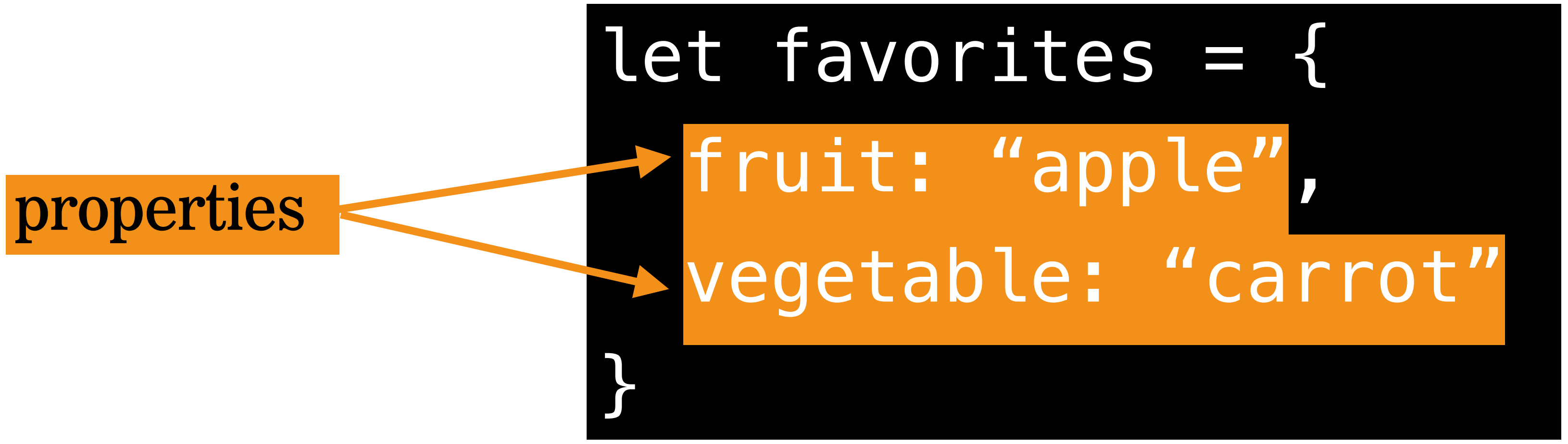
NUMBER

ARRAY

BOOLEAN

OBJECT

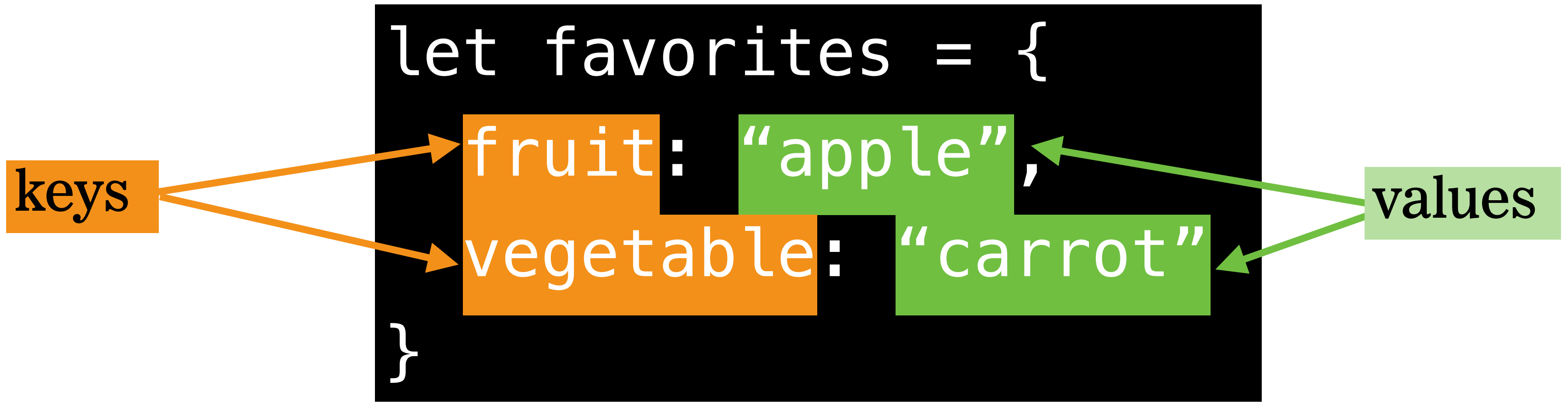
AN OBJECT IS A COLLECTION OF PROPERTIES



```
let favorites = {
  fruit: “apple”,
  vegetable: “carrot”
}
```

PROPERTY = KEY & VALUE

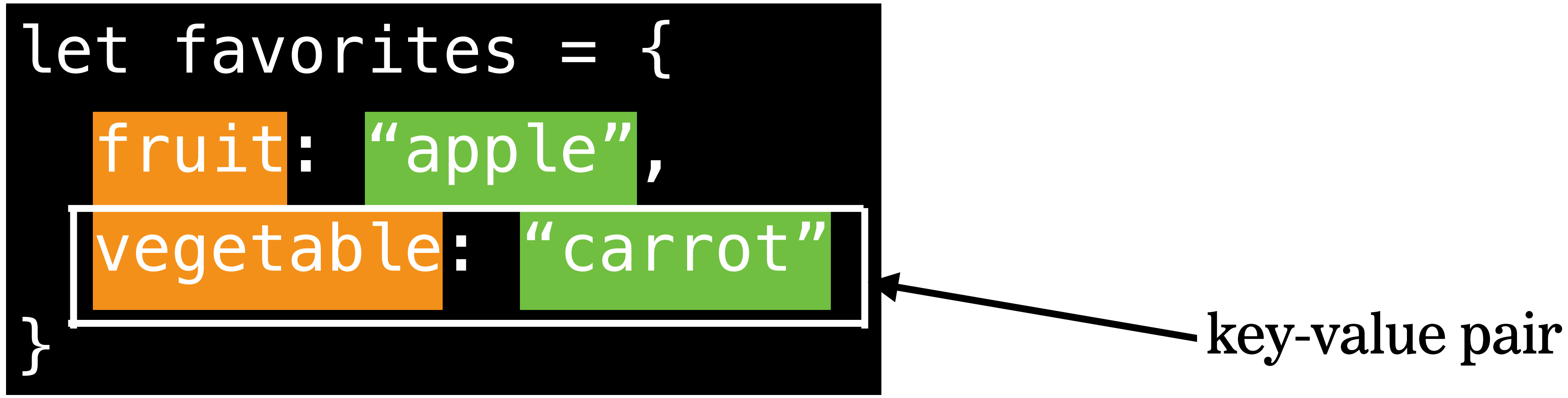
- A **property** is an association between a key and a value
  - **key**: name (often descriptive) used to reference the data
  - **value**: the data stored in that property
- A property is sometimes referred to as a **key-value pair**



```
let favorites = {
  fruit: "apple",
  vegetable: "carrot"
}
```

KEY-VALUE PAIR

- A property is sometimes referred to as a **key-value pair**



```
let favorites = {
  fruit: “apple”,
  vegetable: “carrot”
}
```

AN OBJECT IS NOT ORDERED

```
  [
0   “apple”,
1   “pear”,
2   “banana”
  ]
```

ARRAY
ordered

```
{
  fruit: “apple”,
  vegetable: “carrot”,
  fungus: “trumpet mushroom”
}
```

OBJECT
not ordered

A METHOD IS A PROPERTY WHOSE VALUE IS A FUNCTION

```
let favorites = {
  fruit: “apple”,
  vegetable: “carrot”,
  declare: function() {
    console.log(“I like fruits and vegetables!”);
  }
}
```

method

TWO WAYS TO GET/SET PROPERTIES

dot notation

square bracket notation

GETTING A PROPERTY VALUE WITH DOT NOTATION

object

```
let favorites = {
  fruit: “apple”,
  veg: “carrot”,
  declare: function() {
    console.log(“I like fruit and veg”);
  }
}
```

object name

getting properties

property name

```
favorites.fruit
> “apple”
favorites.veg
> “carrot”
```

object name

calling a method

method name

```
favorites.declare()
> “I like fruit and veg”
```

SETTING A PROPERTY VALUE WITH DOT NOTATION

object

```
let favorites = {
  fruit: “apple”,
  veg: “carrot”,
  declare: function() {
    console.log(“I like fruit and veg”);
  }
}
```

setting properties

```
favorites.fungus = ‘shiitake’;
favorites.pet = ‘hamster’;
```

setting a method

```
favorites.beAmbivalent = function() {
  console.log(“I like other things”);
};
```

GETTING A PROPERTY VALUE WITH SQUARE BRACKET NOTATION

object

```
let favorites = {
  fruit: “apple”,
  veg: “carrot”,
  declare: function() {
    console.log(“I like fruit and veg”);
  }
}
```

object name

getting properties

```
favorites[fruit]
> “apple”
favorites[veg]
> “carrot”
```

property name

SETTING A PROPERTY VALUE WITH SQUARE BRACKET NOTATION

object

```
let favorites = {
  fruit: “apple”,
  veg: “carrot”,
  declare: function() {
    console.log(“I like fruit and veg”);
  }
}
```

setting properties

```
favorites[fungus] = ‘shiitake’;
favorites[pet] = ‘hamster’;
```

setting a method

```
favorites[beAmbivalent] = function() {
  console.log(“I like other things”);
};
```

EXERCISE — OBJECTS

EXERCISE

KEY OBJECTIVE

- Create JavaScript objects using object literal notation

TYPE OF EXERCISE

- Pairs (same pair as for previous exercise)

TIMING

3 min

1. On your desk or on the wall, write code to create a variable whose name corresponds to the thing you were assigned in the previous exercise (cloud, houseplant, nation, office chair, or airplane).
2. Write code to add a property to the object and specify a value for the property.
3. Write code to add a method to the object, and specify a value for the method (use a comment or console.log() statement for the function body).
4. BONUS: Rewrite your answers for 1-3 as a single JavaScript statement.

REAL WORLD SCENARIO

A user, browsing on a shopping website, searches for size 12 running shoes, and examines several pairs before purchasing one.

OBJECTS = NOUNS

A user, browsing on a shopping website, searches for size 12 running shoes, and examines several pairs before purchasing one.

implicit object:

shopping cart

PROPERTIES = ADJECTIVES

A user, browsing on a shopping website, searches for size 12 running shoes, and examines several pairs before purchasing one.

implicit properties:

for each pair of shoes:

price
color

for the shopping cart:

contents
total
shipping
tax

METHODS = VERBS

A user, browsing on a shopping website, searches for size 12 running shoes, and examines several pairs before purchasing one.

implicit methods:

for each pair of shoes:

add to cart

for the shopping cart:

calculate shipping
calculate tax
complete purchase
remove item

EXERCISE — REAL WORLD SCENARIOS & OBJECTS

EXERCISE

KEY OBJECTIVE

- Identify likely objects, properties, and methods in real-world scenarios

TYPE OF EXERCISE

- Groups of 3-4

TIMING

10 min

1. Read through your scenario together.
2. Identify and write down likely objects, properties, and methods in your scenario. (Remember to consider implicit objects as well as explicit ones.)
3. Choose someone to report you results to the class.

LAB — OBJECTS

LAB

KEY OBJECTIVE

- Create JavaScript objects using object literal notation

TYPE OF EXERCISE

- Individual or pair

TIMING

20 min

1. Open `starter-code > 1-object-exercise > monkey.js` in your editor.
2. Create objects for 3 different monkeys each with the properties `name`, `species`, and `foodsEaten`, and the methods `eatSomething(thingAsString)` and `introduce`.
3. Practice retrieving properties and using methods with both dot notation and bracket syntax.

Exit Tickets!

(Class #4)

LEARNING OBJECTIVES - REVIEW

- Determine the scope of local and global variables
- Create a program that hoists variables
- Identify likely objects, attributes, and methods in real-world scenarios
- Create JavaScript objects using object literal notation

NEXT CLASS PREVIEW

Hubot Lab

- 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

Q&A