Pre-class Drill

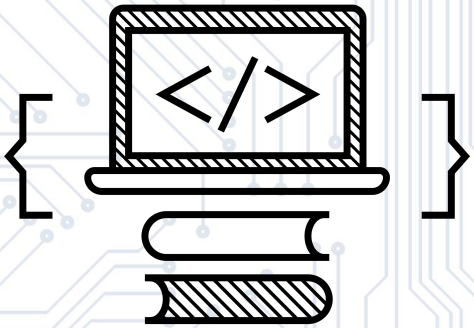
Write a function that takes in an integer and outputs the sum of all the numbers from 1 to that integer.

Ex:

Input: 2

Output: 3 (because 1 + 2 = 3)

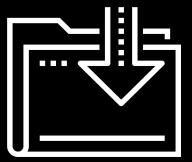
Input: 4



jQuery Jubilee

Web Development Boot Camp

Lesson 4.2



Two Corrections From Last Class

Functions: ES5 vs ES6

```
const adder = function() {

}

const adder = () => {

}
```

Functions: ES5 vs ES6

```
const adder = function() {

}
```

100 % identical

```
const adder = () => {

}
```

Functions: ES5 vs ES6

```
const adder = function() {

}
```

98.3 % identical

```
const adder = () => {

}
```

Functions: ES5 vs ES6

```
const adder = function() {

}
```

```
const adder = () => {

}
```

Differ in how they handle the `this` keyword.

Functions: ES5 vs ES6

```
const adder = function() {

}
```

`this` is bound to the function.

```
const adder = () => {

}
```

`this` is bound to it's context -- maybe an outer function or class.

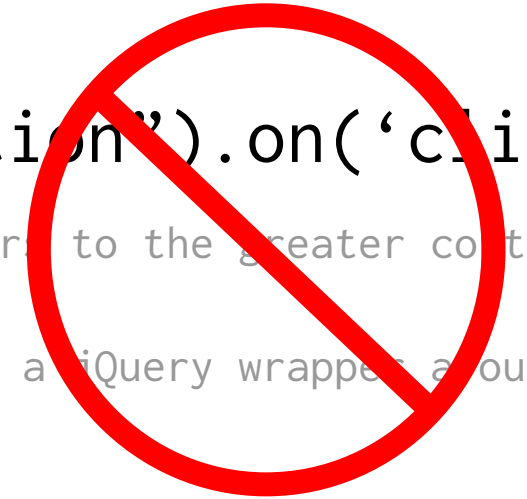
Functions: ES5 vs ES6

```
$(“.menuOption”).on(‘click’, function() {
    // `this` refers to the element that was clicked.
    // `$(this)` is a jQuery wrapper around the element.
}
```

```
$(“.menuOption”).on(‘click’,() => {
    // `this` refers to the greater context.  In this case, the document.
    // `$(this) is a jQuery wrapper around the document
}
```

Functions: ES5 vs ES6

```
$(“.menuOption”).on(‘click’, function() {
    // `this` refers to the element that was clicked.
    // `$(this)` is a jQuery wrapper around the element.
}
```



```
$(“.menuOption”).on(‘click’,() => {
    // `this` refers to the greater context.  In this case, the document.
    // `$(this) is a jQuery wrapper around the document
}
```

Correction #2:

Virgin Radio/Jakarta - Malaysia

Correction #2:

Virgin Radio/Jakarta - Malaysia

Correction #2:

Virgin Radio/Jakarta - Indonesia

Maybe You Thought Learning jQuery would be like this…

But, instead, it was more like this…

This shouldn’t be you:

Remember This:

You can't tell whether you're learning something when you're learning it—in fact, learning feels a lot more like frustration.

What I've learned is that during this period of frustration is actually when people improve the most, and their improvements are usually obvious to an outsider. If you feel frustrated while trying to understand new concepts, try to remember that it might not feel like it, but you're probably rapidly expanding your knowledge.

—Jeff Dickey, author of *Write Modern Web Apps with the MEAN Stack: Mongo, Express, AngularJS, and Node.js*

Important Reminders

This course covers a lot of material quickly, so remember:

- Instructors and TAs are here to help.
- Feel encouraged to schedule a one-on-one during office hours.
- One-on-one sessions are a great way to identify weaknesses and outline a plan to get back on track.
- Office hours are held before and after class.

Today’s Class

Objectives

01 Use jQuery DOM manipulation to create simple games.

02 Practice jQuery on Captain Planet: The Game and Fridge Game.

03 Gain an initial understanding of lexical scope in JavaScript.

04 Understand click events.

Captain Planet: The Game!

Captain Planet: The Game!

Superpowers: Change Sizes!

Normal | Grow | Shrink

Superpowers: Invisibility

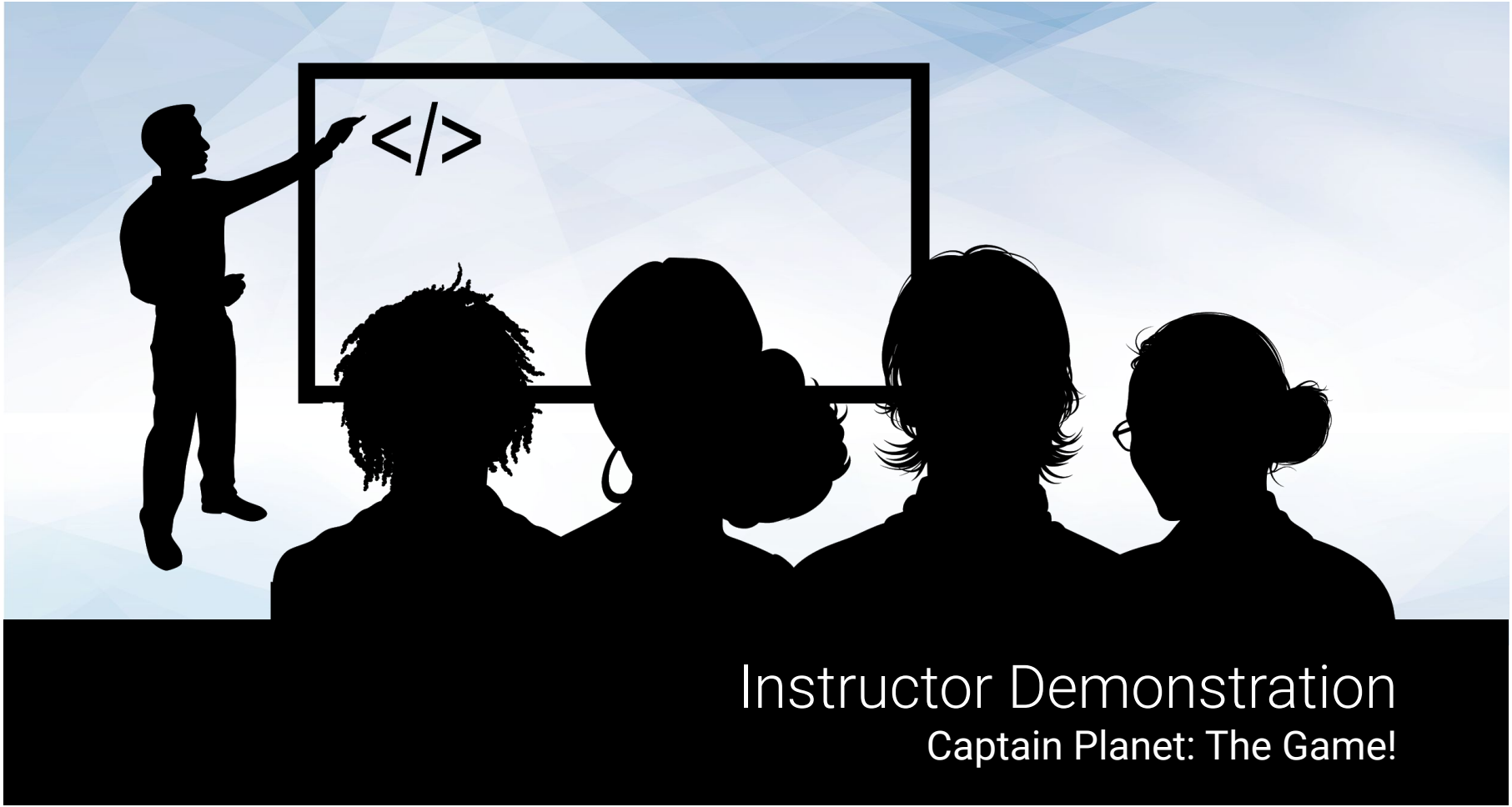
Visible | Invisible

Move Controls

⬆

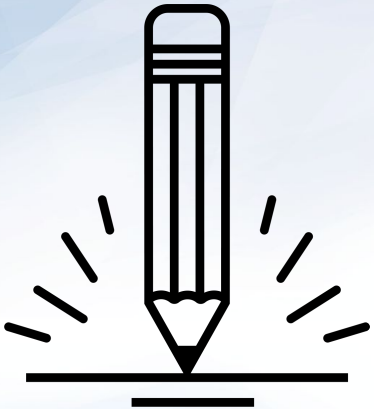
⬅ ⬇ ➡

Go Planet!



Instructor Demonstration

Captain Planet: The Game!



Group Activity:
Pseudocode Captain Planet

Suggested Time:
7 minutes

Group Activity: Pseudocode Captain Planet

Examine the code for the Captain Planet game. Then, describe how this code works in five steps.

1.
2.
3.
4.
5.

Suggested Time: 7 minutes

Pseudocoding Captain Planet

Solution:

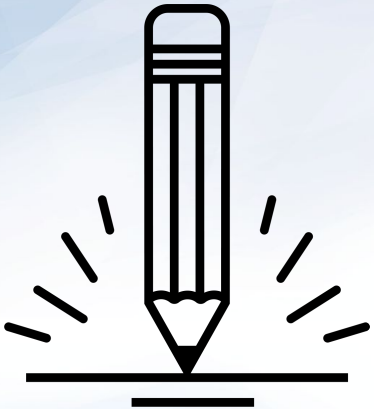
01 Create an initial HTML layout using Bootstrap.

02 Add a reference to jQuery.

03 Assign unique class names to key buttons and images.

04 Use jQuery to capture when the corresponding buttons are clicked, using the `$()` identifier with the class name inside.

05 Create code that changes the CSS of target classes in response to click events.



Activity:
Create a Captain Planet Superpower

Suggested Time:
12 minutes

Activity: Create a Captain Planet Superpower

Review the jQuery API documentation (api.jquery.com). Then, add a button of your own that gives Captain Planet a new power.

Examples:
Click to…stretch Captain Planet.
Click to…trigger a maniacal laugh.
Click to…create clones of Captain Planet.
Click to…create a shield **(hint: border)**.
Click to…create fire or water **(hint: images)**.

Suggested Time: 12 minutes

jQuery Recap

jQuery in a Nutshell

01 Find some HTML.

02 Attach to an event.

03 Do something in response.

jQuery in a Nutshell

We use the jQuery $( ) identifier to capture HTML elements:

`$(“.classname”)`	`$(“div”)`
`$(“#idname”)`	`$(“p”)`

Then, we tie the element to a jQuery method of our choice to capture events:

`.on(“click”)`	`.ready()`

Finally, we modify the selected element or add or remove elements from the DOM:

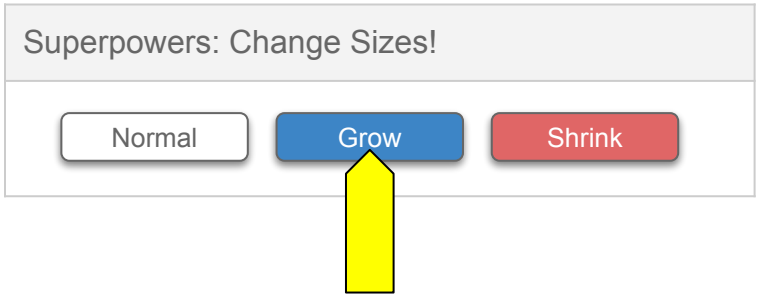
`.animate()`	`.append()`	`.remove()`

jQuery: A Common Example

```
$(".growButton").on("click", function() {
  $(".captainplanet").animate({ height: "500px" });
});
```

01 Click the Grow button.

02 Make Captain Planet grow.



jQuery – Callback Function Review

Notice the function below…

```
$(".growButton").on("click", function() {
  $(".captainplanet").animate({ height: "500px" });
});
```

jQuery – Callback Function Review

Plumber example

“Please leave your name and number at the beep.
When I’m done with my current job, I will…
***CALL YOU BACK**!”*

```
$(".growButton").on("click", function() {
  $(".captainplanet").animate({ height: "500px" });
});
```

jQuery – Callback Function Review

Plumber example

"Please leave your name and number at the beep.
When I'm done with my current job, I will…
***CALL YOU BACK**!"*

```
$(".growButton").on("click", function() {
  $(".captainplanet").animate({ height: "500px" });
});
```

Plumber

Current job

Callback

Use Documentation When Needed:
api.jquery.com

Group Challenge:
Fridge Game

Suggested Time:
35 minutes

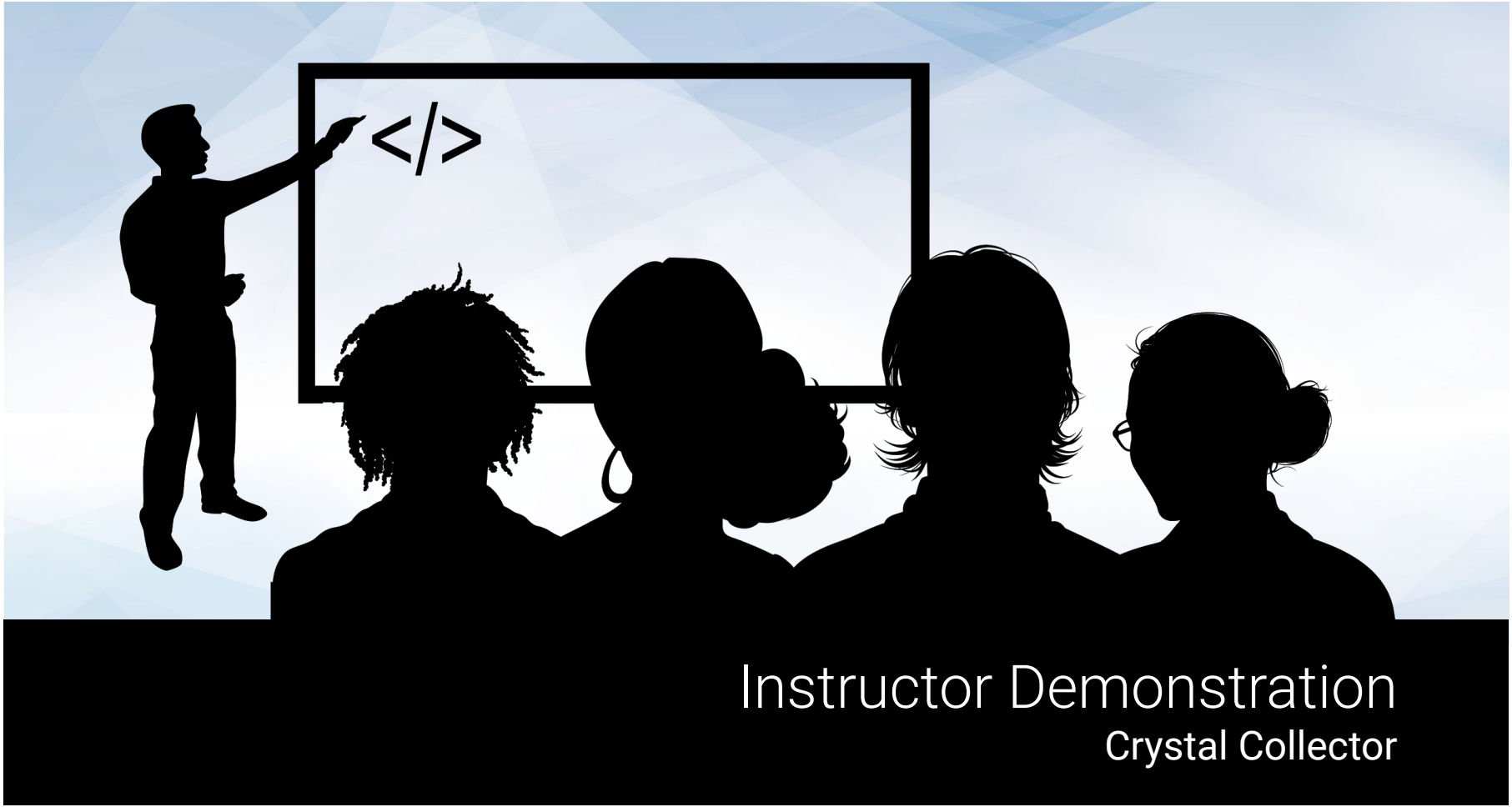
Group Challenge: Fridge Game

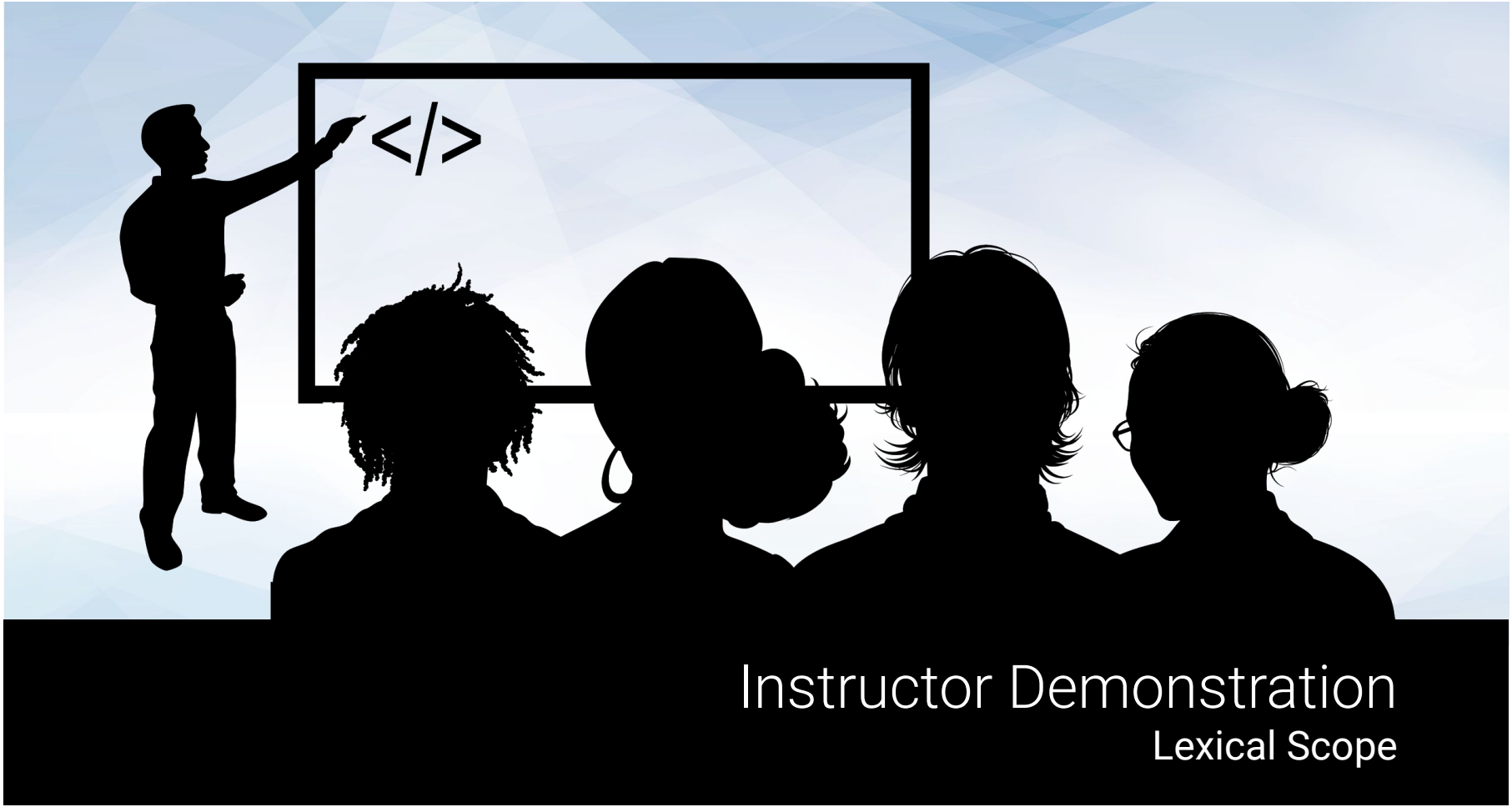
Working in groups of three, complete the code for the fridge game such that:

- JavaScript dynamically generates buttons for each of the letters on the screen.
- Clicking any of the buttons causes the same letter to be displayed on the screen.
- Clicking the Clear button erases all of the letters from the fridge.

Note: This is a challenging activity. You may want one person in the group to type the code while the other two watch to catch bugs and research code snippets when necessary.

Suggested Time: 35 minutes


Instructor Demonstration
Crystal Collector


Instructor Demonstration
Lexical Scope

This next section is **heavy** on theory.

JavaScript Scope

In Javascript, curly **brackets { }** indicate blocks of code.

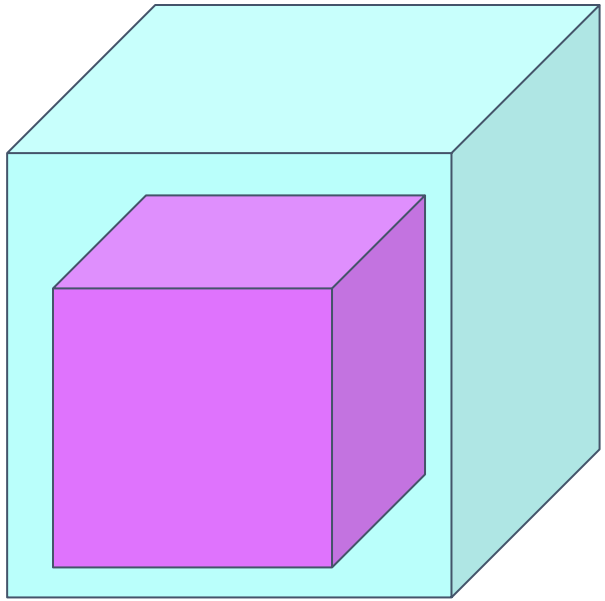
In order for the code inside the curly brackets to be executed, it must meet the condition or be called (example: functions).

These blocks of code can affect variables that were declared outside the curly brackets—so be careful!

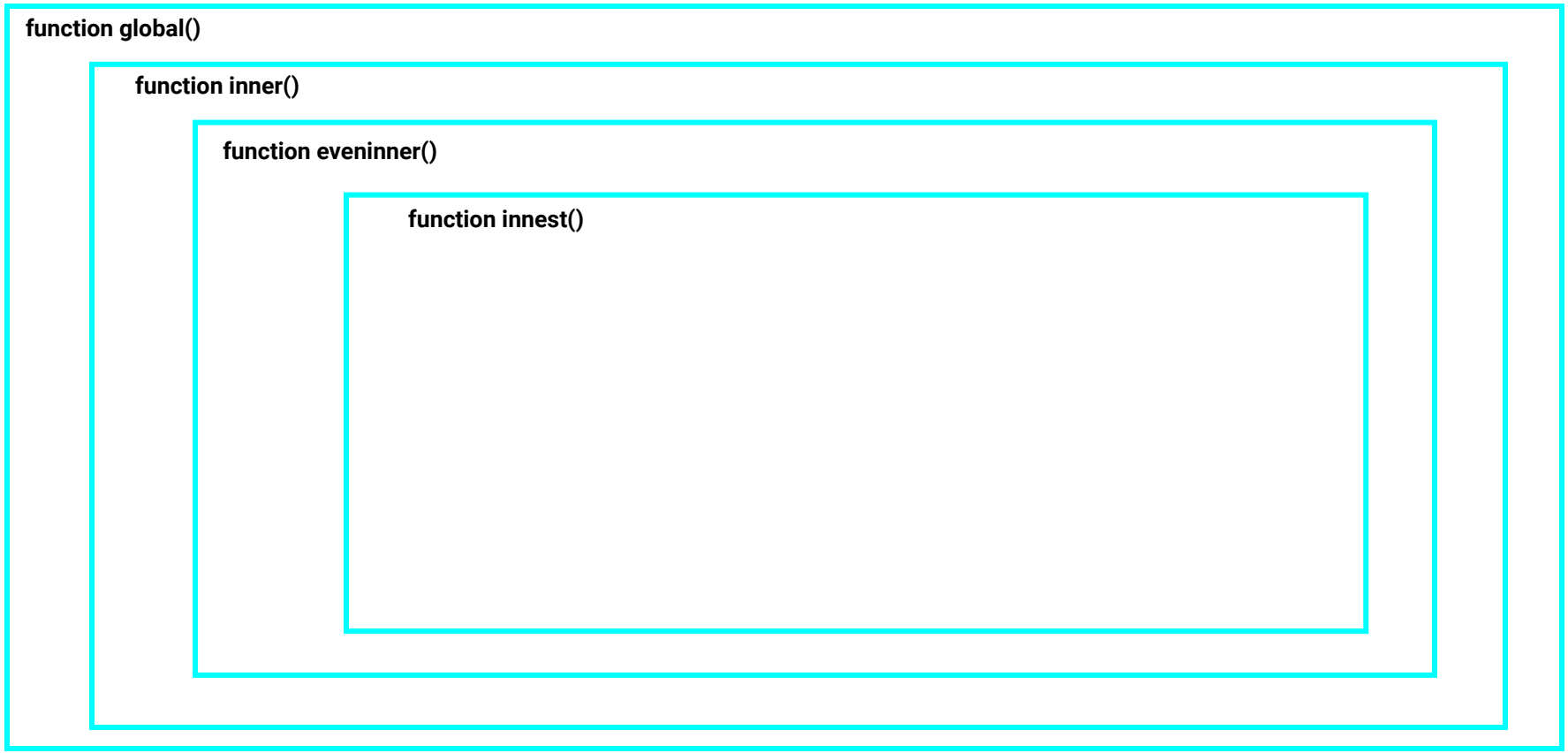
```
// Sets initial value of x
var x = 5;

// False Condition doesn't get run
if(1 > 2000) {
    x = 10
}

// Will print 5. X was unchanged.
console.log(x);
```

Scope = Boxes in Boxes

Scope impacts which variables can be accessed by which function.



Scope = Boxes in Boxes

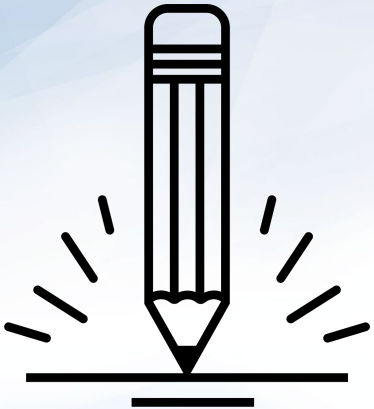


JavaScript Scope Example

Here, **inside** is clearly able to access the variables of its **parent function, outside**.

How does **insideOut** have access to **x**?

```
<script>

  function outside() {

    var x = 1;

    // what is the scope of this function and the scope of y?
    function inside(y) {

      console.log(x + y);
    }

    return inside;
  }

  var insideOut = outside();

  // What does this return?
  insideOut(2);

  // Uncaught ReferenceError: x is not defined.
  // How does insideOut have access to x?
  console.log("The value of 'x' outside 'outside()' is: " + x);

</script>
```



Activity:
Lexical Scope 1

Suggested Time:
10 minutes

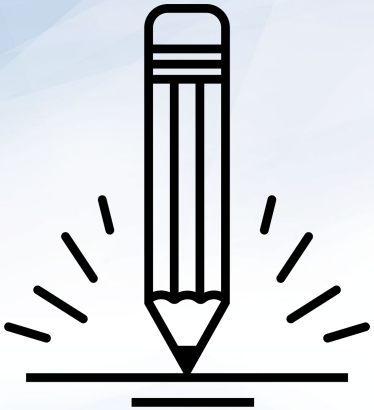
Activity: Lexical Scope 1

Review the file sent to you and explain the following to the person sitting next to you:

- What do the terms *parent function* and *child function* mean?
- Why can child functions access parent variables, but not vice versa?

Be prepared to share your answers!

Suggested Time: 10 minutes



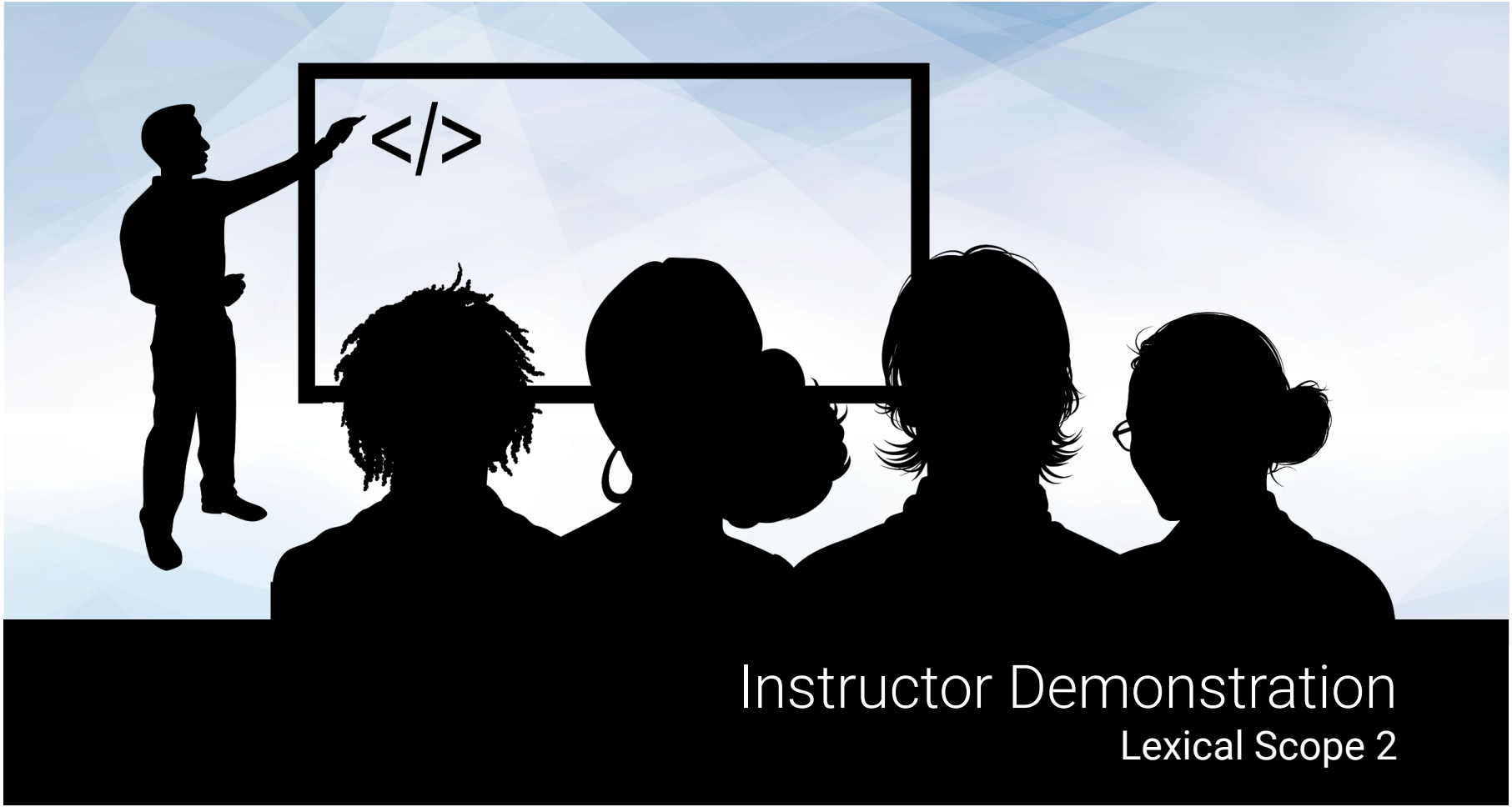
Activity:
Lexical Scope 2

Suggested Time:
7 minutes

Activity: Lexical Scope 2

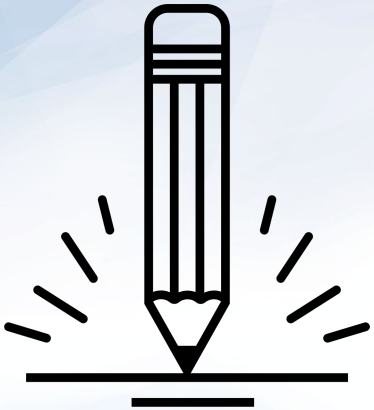
- Take a few moments to dissect the code just sent to you.
- Try to predict what will be printed in each of the examples.
- Be prepared to share!
- **Note:** Pay attention to the unusual use of the keyword *this*.

Suggested Time: 7 minutes



Instructor Demonstration

Lexical Scope 2



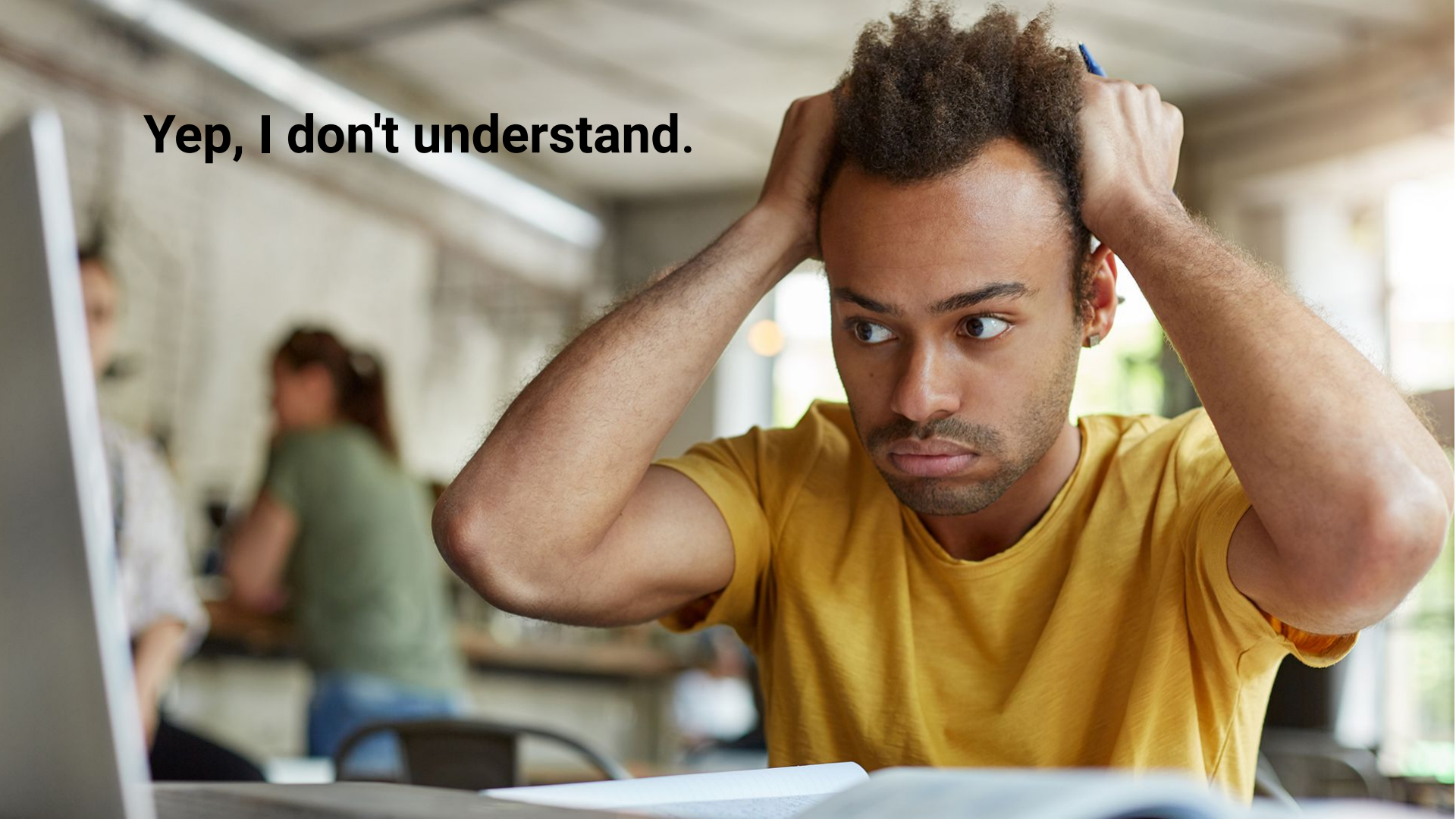
Activity:
Lexical Scope 3

Suggested Time:
7 minutes

Activity: Lexical Scope 3

- Take a few moments to dissect the code just sent to you.
- Try to predict what will be printed in each of the examples.
- Be prepared to share!
- **Note:** Pay attention to the unusual use of the keyword *this*.

Suggested Time: 7 minutes


Yep, I don't understand.

If you'd like to learn more, here's a helpful article:

What You Should Already Know about JavaScript Scope

spin.atomicobject.com

Challenge:

Color Corrector:
Build a Brain Teaser

Suggested Time:

20 minutes *plus* additional 20 minutes at home

Color Corrector: Build a Brain Teaser

Choose the color of the word shown from the list below:

teal
brown
magenta
blue
teal
coral
black

Challenge: Color Corrector: Build a Brain Teaser

Using the files sent to you as a starting point, add the missing code so that the Color Corrector game works correctly.

To win, choose the word that matches the color of the text at the top of the column.

Example:

Suggested Time: 20 minutes

Questions?