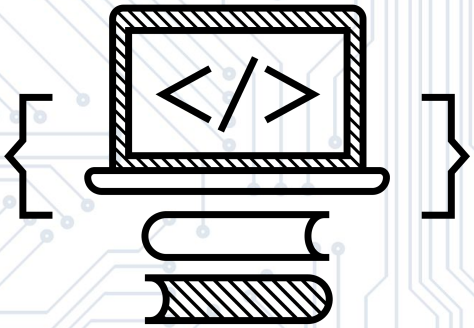
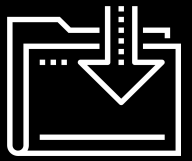


jQuery Jubilee

Web Development Boot Camp

Lesson 5.2



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.

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:

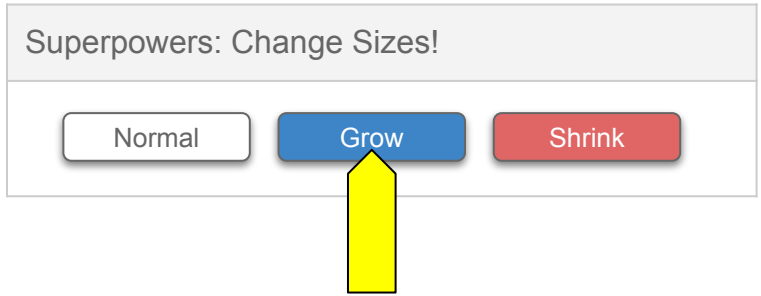
`.animate()`	`.append()`	`.remove()`

jQuery: A Common Example

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

01 Click the Grow button.

02 Make Captain Planet grow.



Use Documentation When Needed:
api.jquery.com

Captain Planet: The Game!

Captain Planet: The Game!

Superpowers: Change Sizes!

Normal | Grow | Shrink

Superpowers: Invisibility

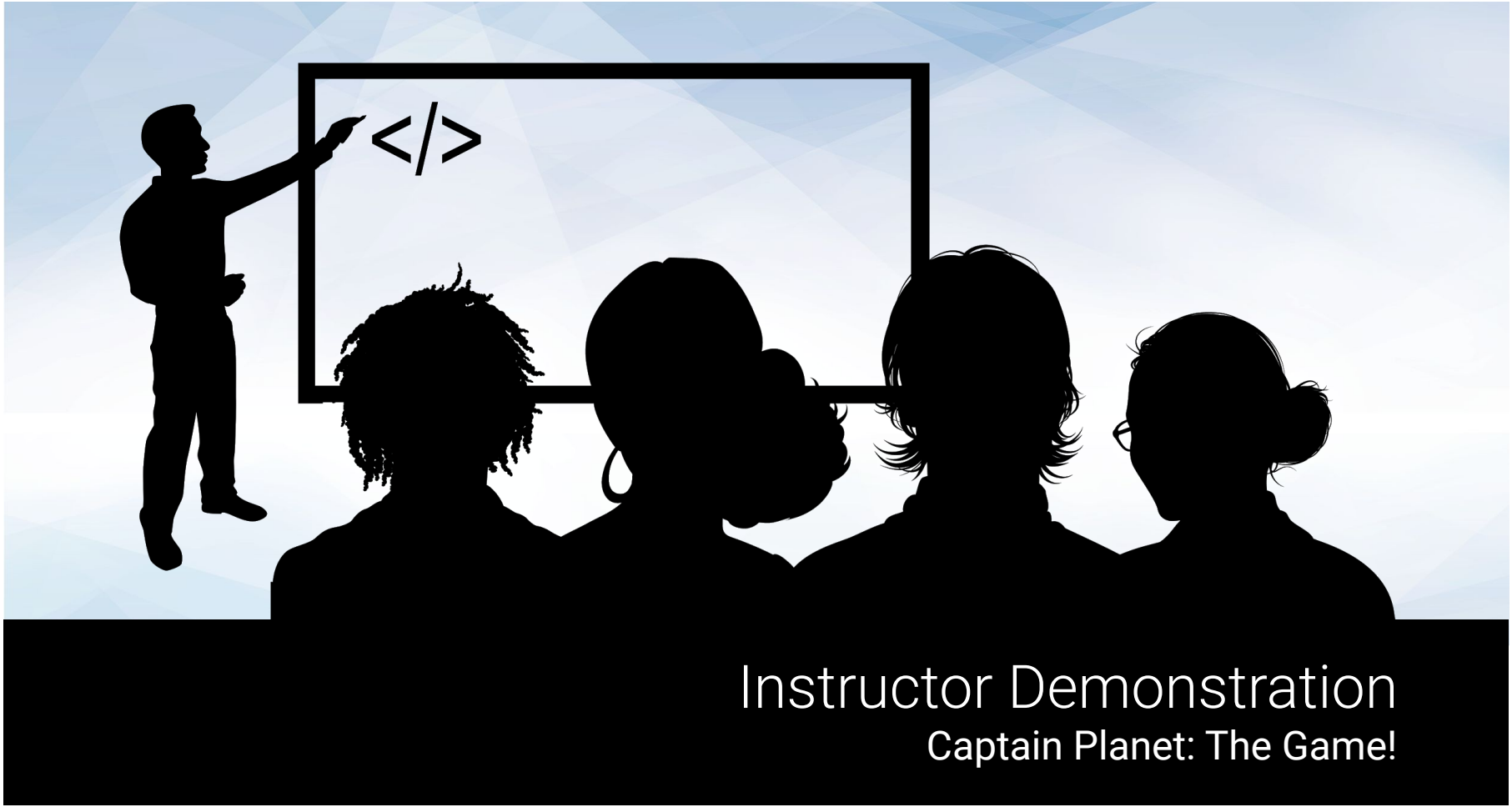
Visible | Invisible

Move Controls

⬆

⬅ ⬇ ➡

Go Planet!



Instructor Demonstration

Captain Planet: The Game!

Pseudocoding Captain Planet

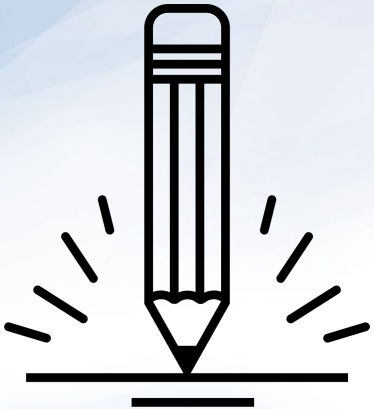
01 Create an initial HTML layout using Bootstrap.

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

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

Group Challenge:
Fridge Game

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.

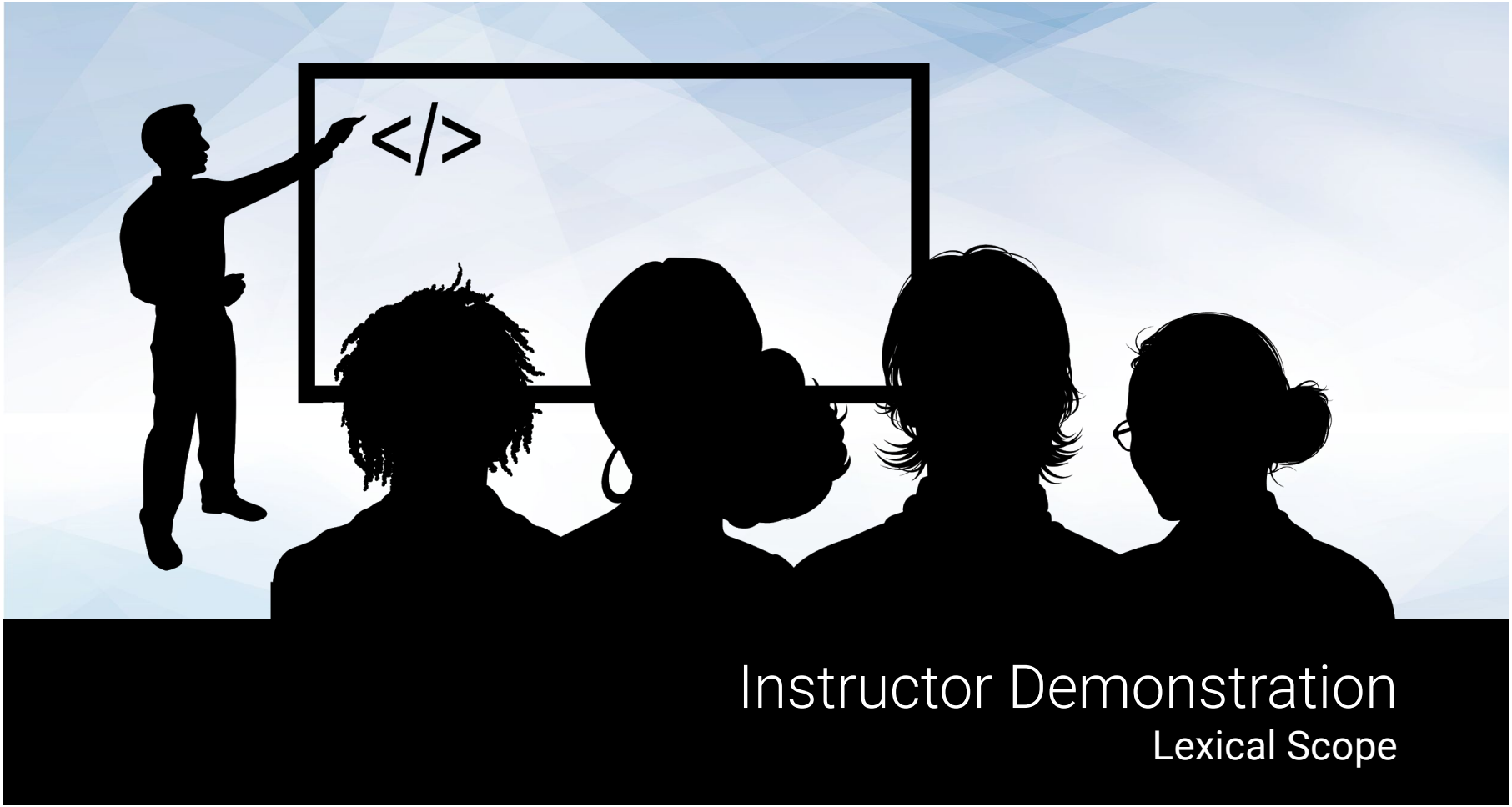
Note: This is a challenging activity. You may want one person in the group to type the code while the other two watch to catch bugs and research code snippets when necessary.

Github Pull Requests

Use Documentation When Needed:

- Git Docs: Pull Requests
- GitHub Hello World Guide to Pull Requests
- GitHub Docs: Pull Requests


Break


Instructor Demonstration
Lexical Scope

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

JavaScript Scope

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

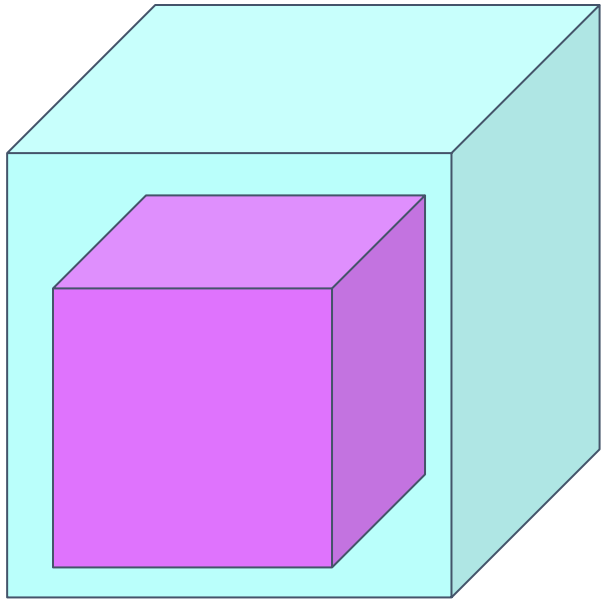
In order for the code inside the curly brackets to be executed, it must meet the condition or be called (example: functions).

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

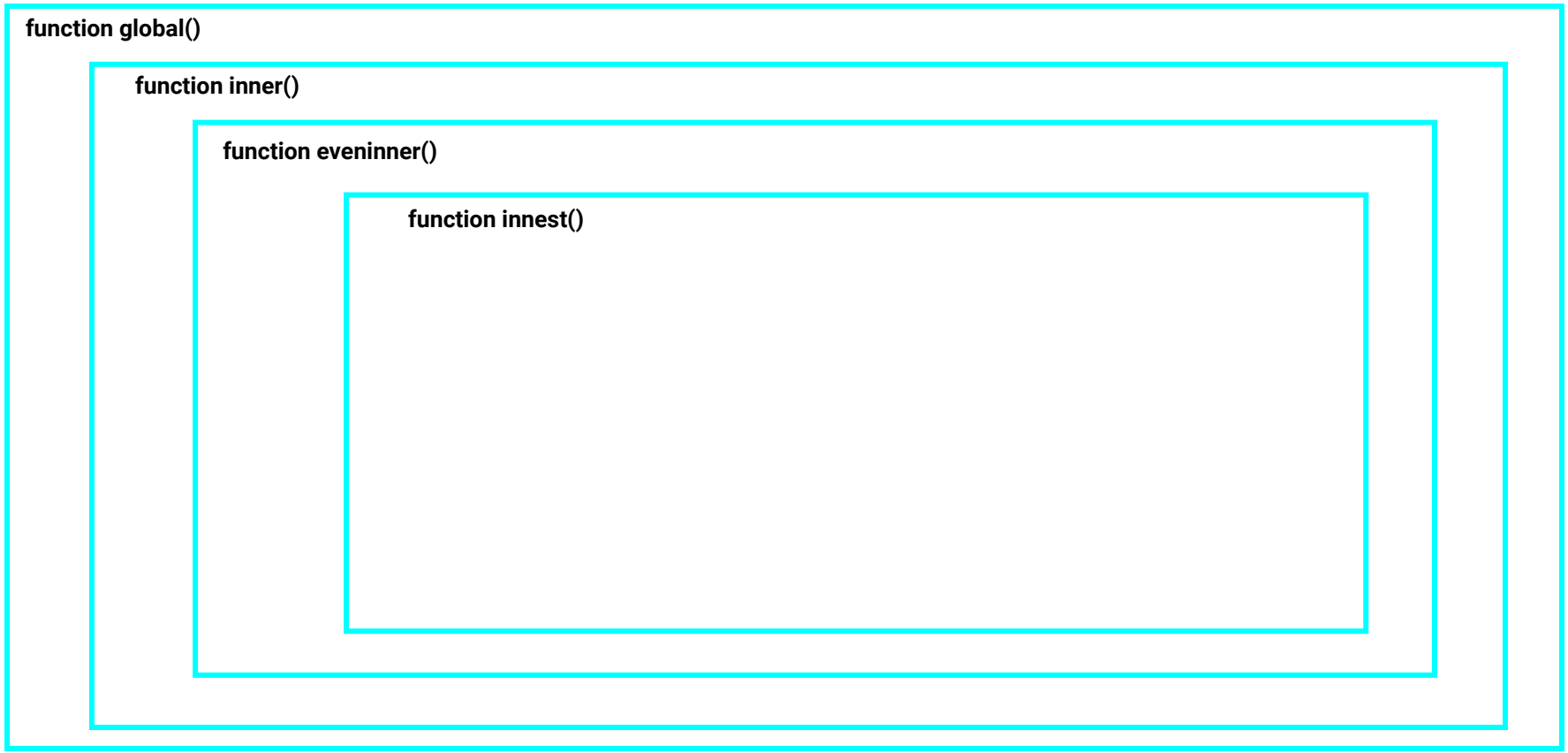
```
// Sets initial value of x
var x = 5;

// 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

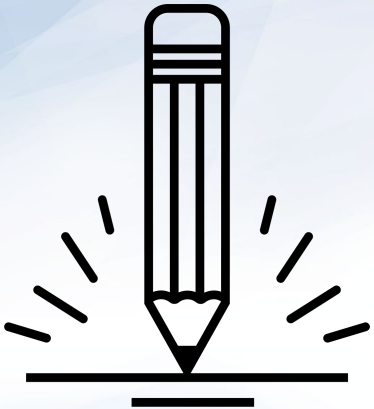


JavaScript Scope Example

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

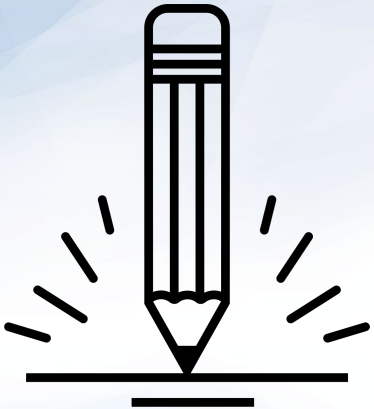
Activity: Lexical Scope 1

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



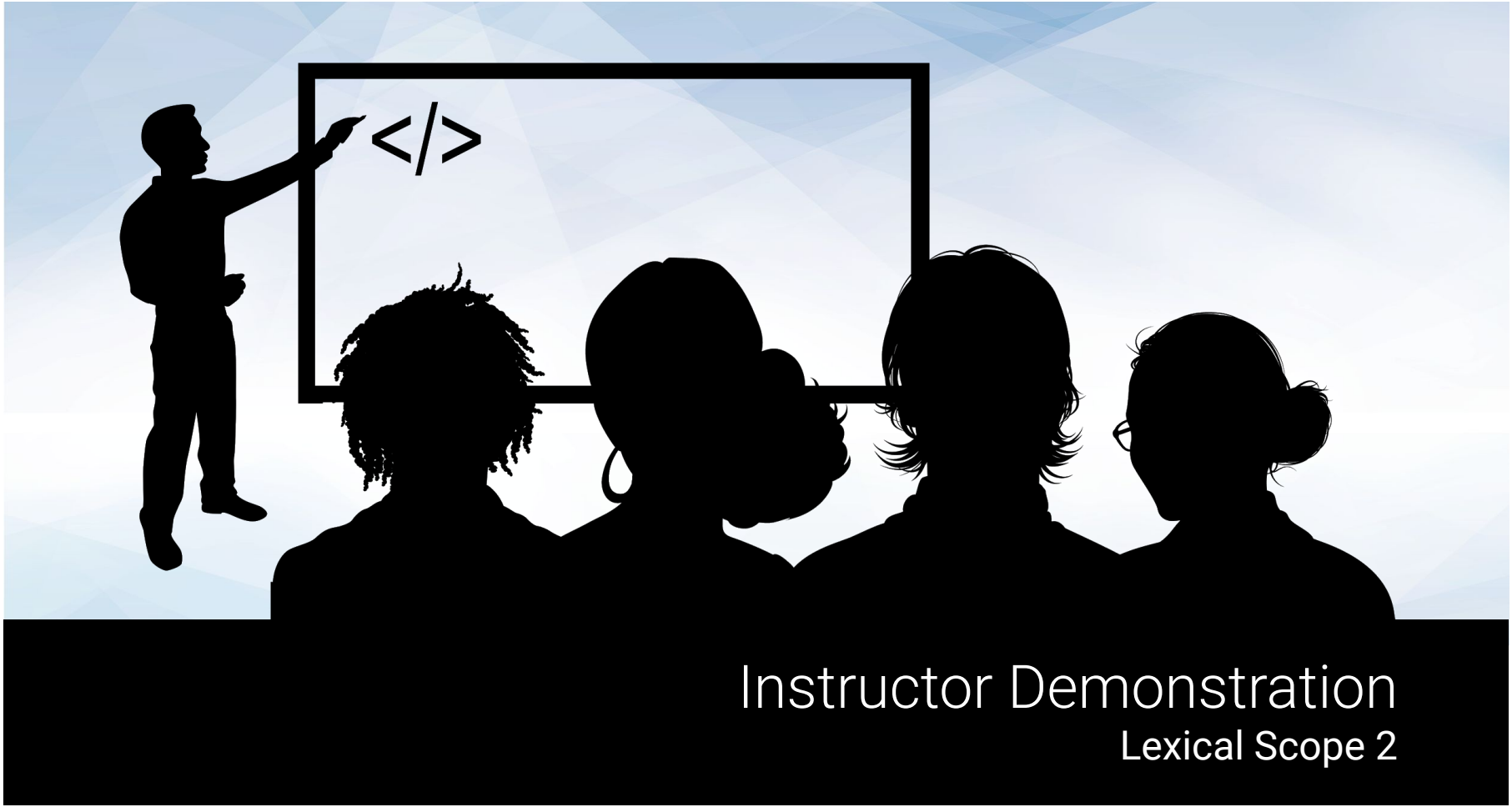
Activity:
Lexical Scope 2

Suggested Time:
7 minutes

Activity: Lexical Scope 2

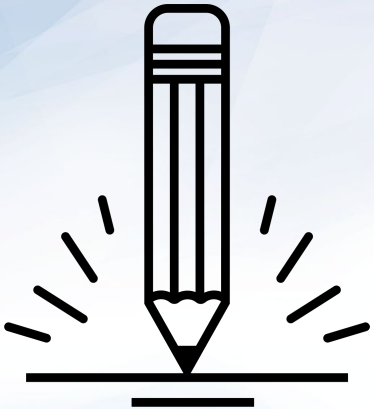
- Take a few moments to dissect the code just sent to you.
- Try to predict what will be printed in each of the examples.
- Be prepared to share!
- **Note:** Pay attention to the unusual use of the keyword *this*.

Suggested Time: 7 minutes



Instructor Demonstration

Lexical Scope 2



Activity:
Lexical Scope 3

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

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

What You Should Already Know about JavaScript Scope

spin.atomicobject.com

Questions?