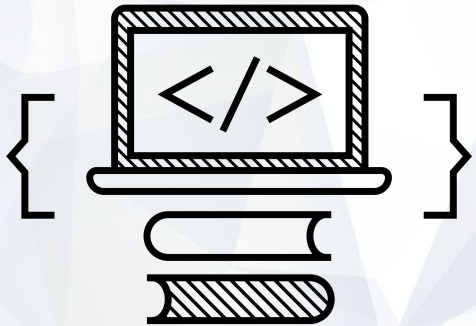
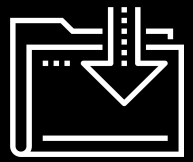


jQuery Jubilee

Skills Bootcamp in Front-End Web Development

Lesson 7.2




WELCOME

Important Reminders

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

- Feel encouraged to schedule a one-on-one during office hours.
- Instructors and TAs are here to help.
- 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.

You’ve just completed the most challenging aspect of the course!

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

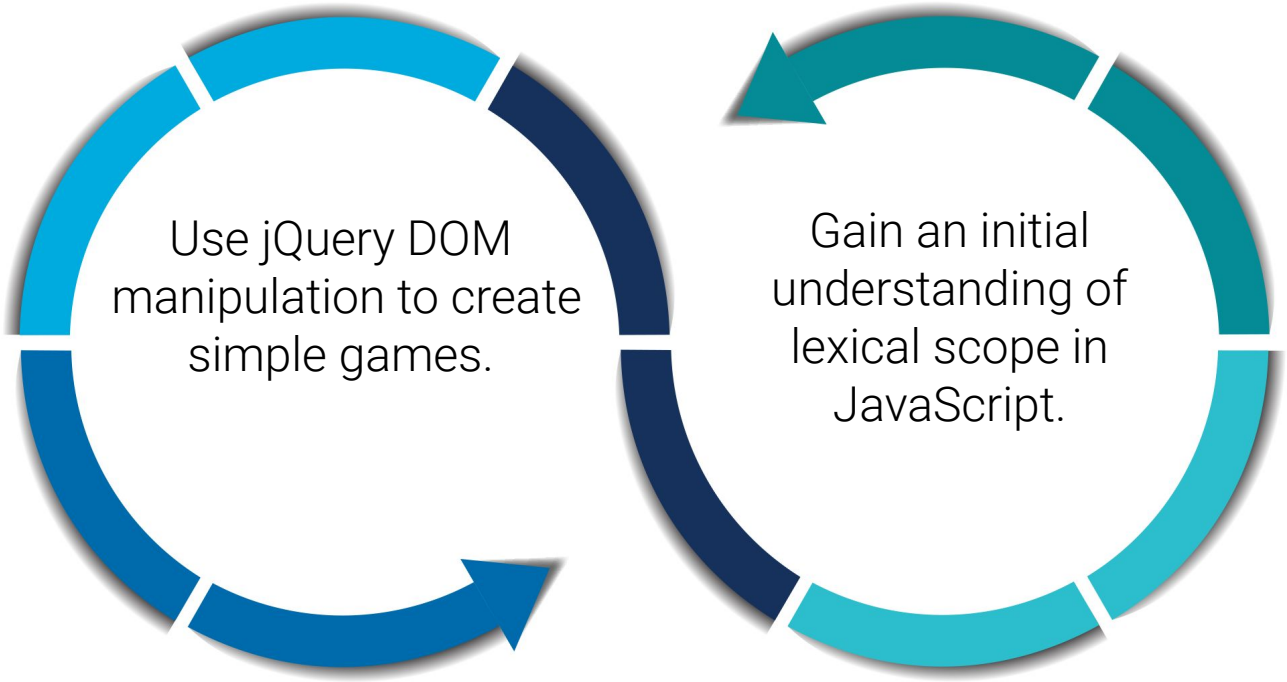
What I’ve learned is that during this period of frustration is actually when people improve the most, and their improvements are usually obvious to an outsider. If you feel frustrated while trying to understand new concepts, try to remember that it might not feel like it, but you’re probably rapidly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

—Jeff Dickey,
author of *Write Modern Web Apps with the MEAN Stack: Mongo, Express, AngularJS, and Node.JS* (Peachpit Press, 2014)

Today’s Class

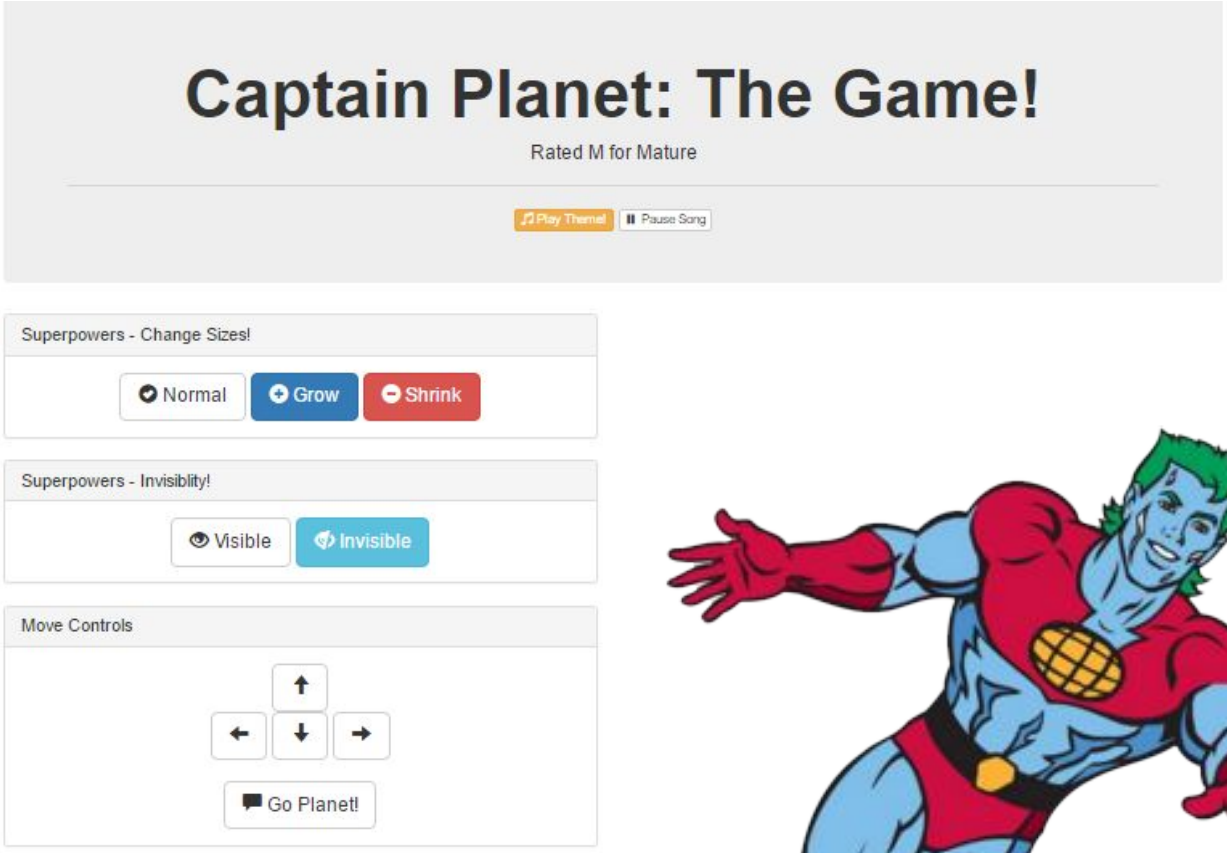
Today's Objectives

By the end of class today, you will:



Captain Planet: The Game!

Captain Planet: The Game!



Activity: Pseudocode Captain Planet

In this activity, you'll pseudocode the game's logic and structure.

Suggested Time:

Time's Up! Let's Review.

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.

Pseudocoding Captain Planet

A solution:

01 Add a reference to jQuery.

02 Create an initial HTML layout using Bootstrap.

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 Superpower for Captain Planet

In this activity, you'll dissect and add additional code to create a new superpower for Captain Planet.

Suggested Time:

Instructions: 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 superpower.

Examples:

- Click to…Stretch Captain Planet.
- Click to…Trigger a maniacal laugh.
- Click to…Create clones of Captain Planet.
- Click to…Create fire or water (**hint:** images).

Time's Up! Let's Review.

jQuery Recap

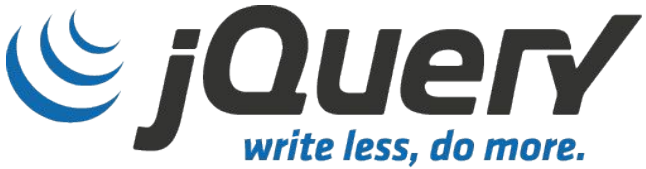
jQuery is “capturing” HTML elements using the `$()` identifier, and we are then applying various methods to that element (or a different one in response).

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" });
});
```

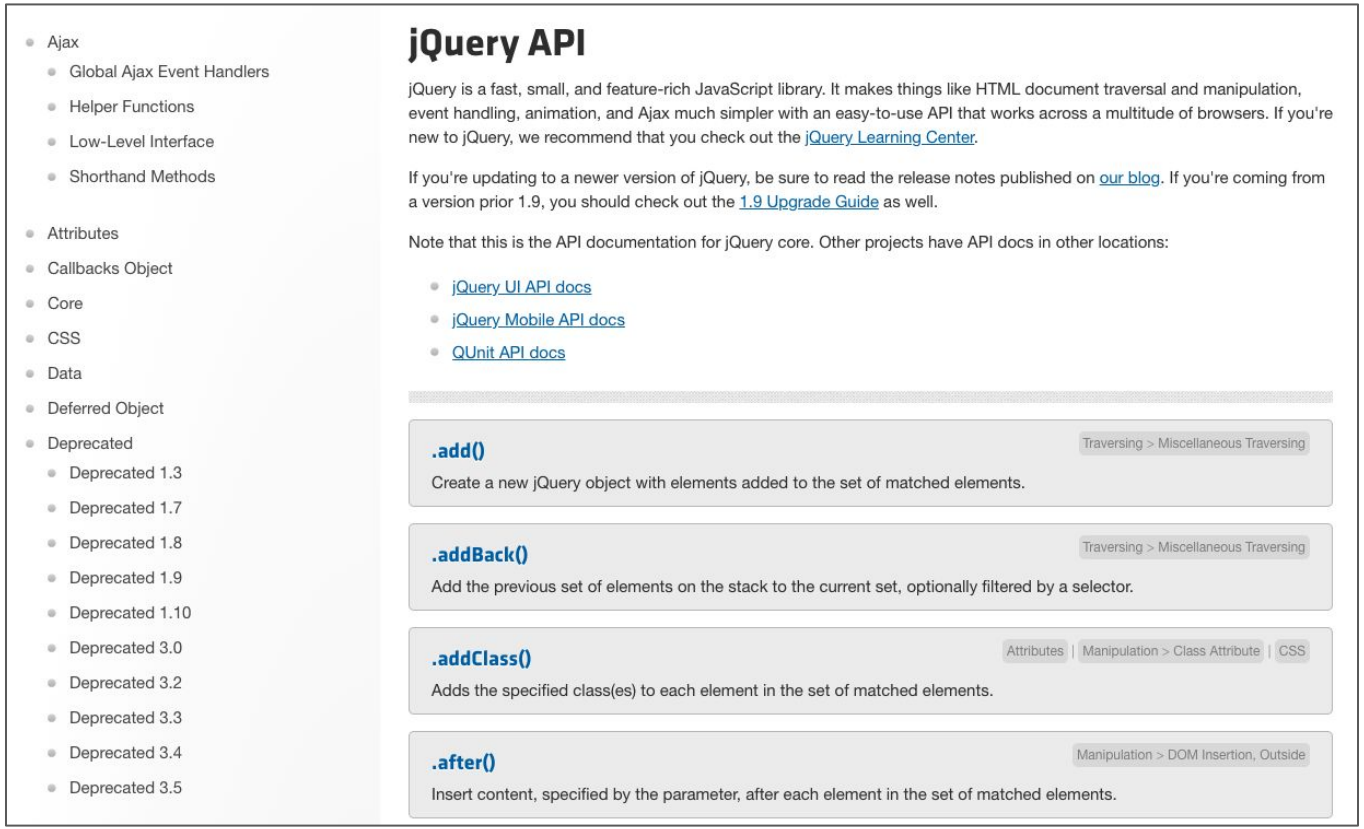
01 Click the “Grow” button.

02 Make Captain Planet grow.

Superpowers: Change Sizes!

Normal | Grow | Shrink

Use Documentation When Needed: api.jquery.com



Activity: Fridge Game

In this activity, you'll click on letters from a menu and have those letters appear on the fridge.

Suggested Time:

Instructions: 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.
- Hitting the clear button erases all of the letters from the fridge.

Note: This is a challenging exercise. Watch for bugs and/or research necessary code snippets.

Time's Up! Let's Review.

Activity Review: Fridge Game

Let's talk about:

The creation of an array that holds all of the letters.

The for loop used to take letters from the array, associate each with a data attribute and text, and then append them onto the page.

```
// 1. Create a for-Loop to iterate through the letters array.
for (var i = 0; i < letters.length; i++) {

// Inside the loop...

  // 2. Create a variable named "letterBtn" equal to $("<button>");
  var letterBtn = $("<button>");

  // 3. Then give each "letterBtn" the following classes: "letter-button" "letter" "letter-button-color".
  letterBtn.addClass("letter-button letter letter-button-color");

  // 4. Then give each "letterBtn" a data-attribute called "data-letter".
  letterBtn.attr("data-letter", letters[i]);

  // 5. Then give each "letterBtns" a text equal to "letters[i]".
  letterBtn.text(letters[i]);

  // 6. Finally, append each "letterBtn" to the "#buttons" div (provided).
  $("#buttons").append(letterBtn);

}
```

Activity Review: Fridge Game, continued

This is the **KEY POINT**.

The on-click event needed to capture button clicks. Be sure to point out how we use the data attribute (via the `.attr` method) to know which letter was clicked.

```
// 7. Create an "on-click" event (local function)(): void letter-button" class.
$(".letter-button").on("click", function() {

  // Inside the on-click event...

  // 8. Create a variable called "fridgeMagnet" and set the variable equal to a new div.
  var fridgeMagnet = $("<div>");

  // 9. Give each "fridgeMagnet" the following classes: "letter fridge-color".
  fridgeMagnet.addClass("letter fridge-color");

  // 10. Then chain the following code onto the "fridgeMagnet" variable: .text($(this).attr("data-letter"))
  fridgeMagnet.text($(this).attr("data-letter"));
```

Determines the letter via the data-letter atribute


Time For a Quick Video
Fridge Game Activity Review

Crystal Example

Introduction to Lexical Scope

This section is pretty heavy on theory.

Introduction to Lexical Scope

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

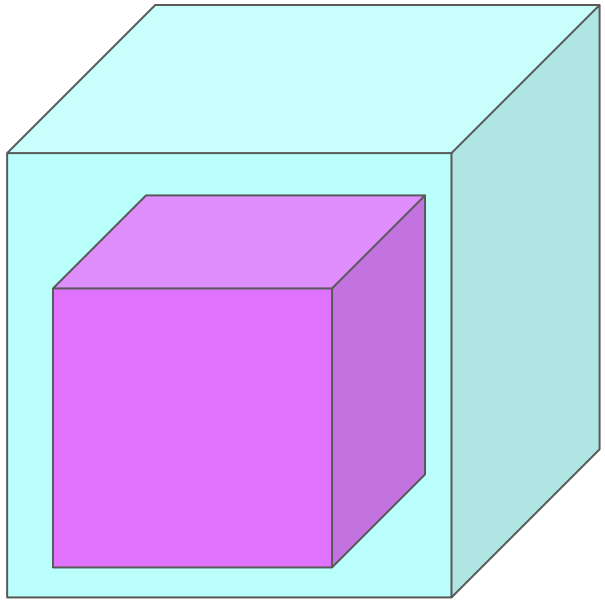
In order for the code inside the curly brackets to be executed, it must meet the condition or be called (e.g., functions).

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

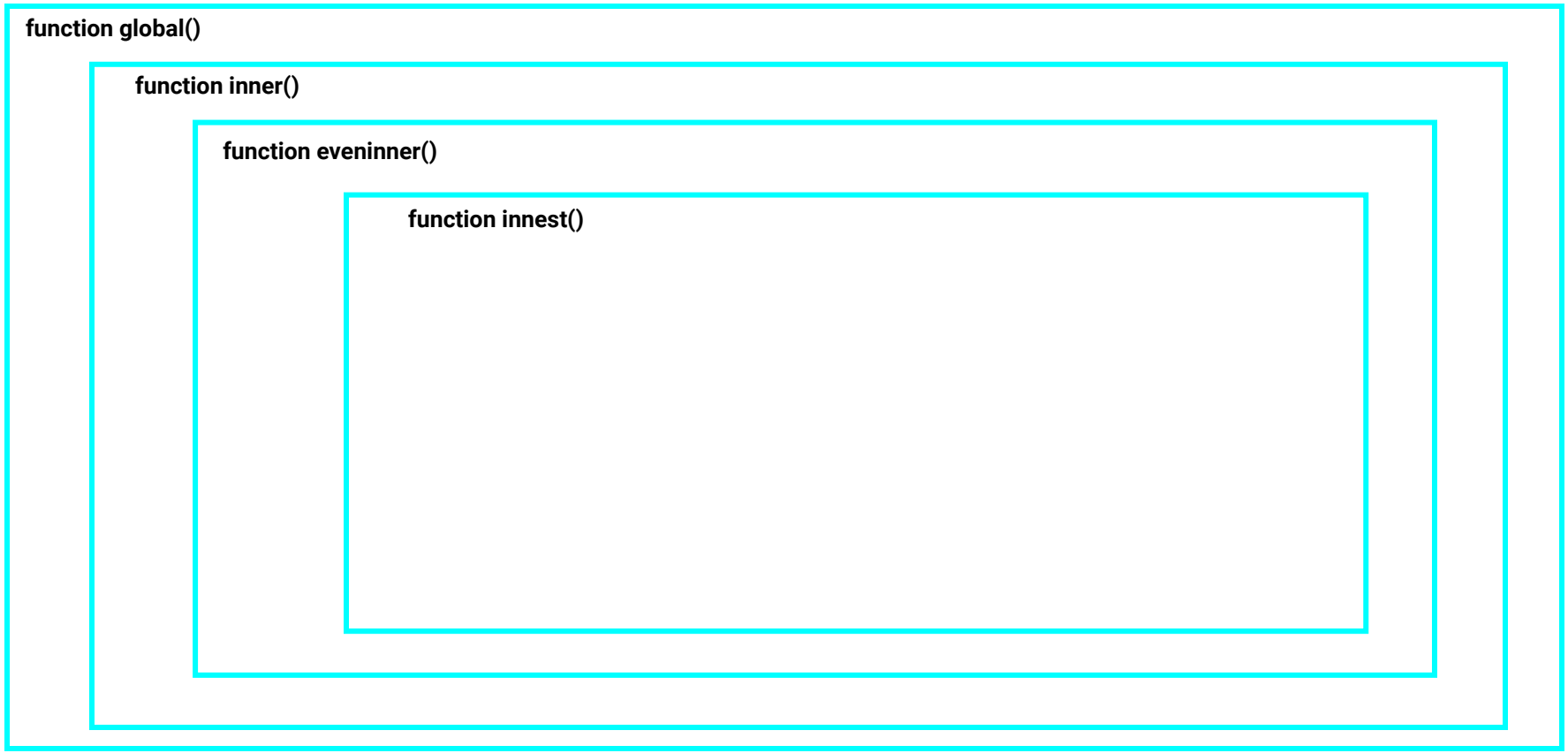
```
// Sets initial value of x
var x = 5;

// 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:

Instructions: Lexical Scope 1

Open `Unsolved/index.html` in a browser and then open the console.
With a partner, compare the results in the console with the JavaScript in `index.html` and answer the questions in the comments.

Hint: Read the MDN documentation on closures.

Time's Up! Let's Review.

Review: Lexical Scope 1

The key concept is closures:

When the return value of `outside()` is assigned to `insideOut`, a closure is created.

A closure is an object that contains both a returned function and the environment in which that function was created.

The environment consists of any local variables that were available to that function when and where it was declared.

When we call `insideOut()`, it returns the values stored in `x` and `y`, even though those were declared outside `inside()`.

Activity: Lexical Scope 2

Suggested Time:

Instructions: 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.

Time's Up! Let's Review.

Review: Lexical Scope 2

The key takeaway here is that using the keyword `this` will only print content related to the object directly above it, not from the grandparent.

```
var cat = {
  name: "Gus",
  color: "gray",
  age: 15,

  printInfo: function() {

    // What will this print? ("Object")
    console.log(this);

    // What will this print? ("Name: Gus Color: gray Age: 15")
    console.log("Name:", this.name, "Color:", this.color, "Age:", this.age);

    var nestedFunction = function() {

      // What will this print? ("Window")
      console.log(this);

      // What will this print? ("Name:  Color: undefined Age: undefined")
      console.log("Name:", this.name, "Color:", this.color, "Age:", this.age);
    };

    nestedFunction();
  }
};

// calls the printInfo function. Which subsequently calls the nestedFunction()
cat.printInfo();
```

Activity: Lexical Scope 3

Suggested Time:

Instructions: Lexical Scope 2

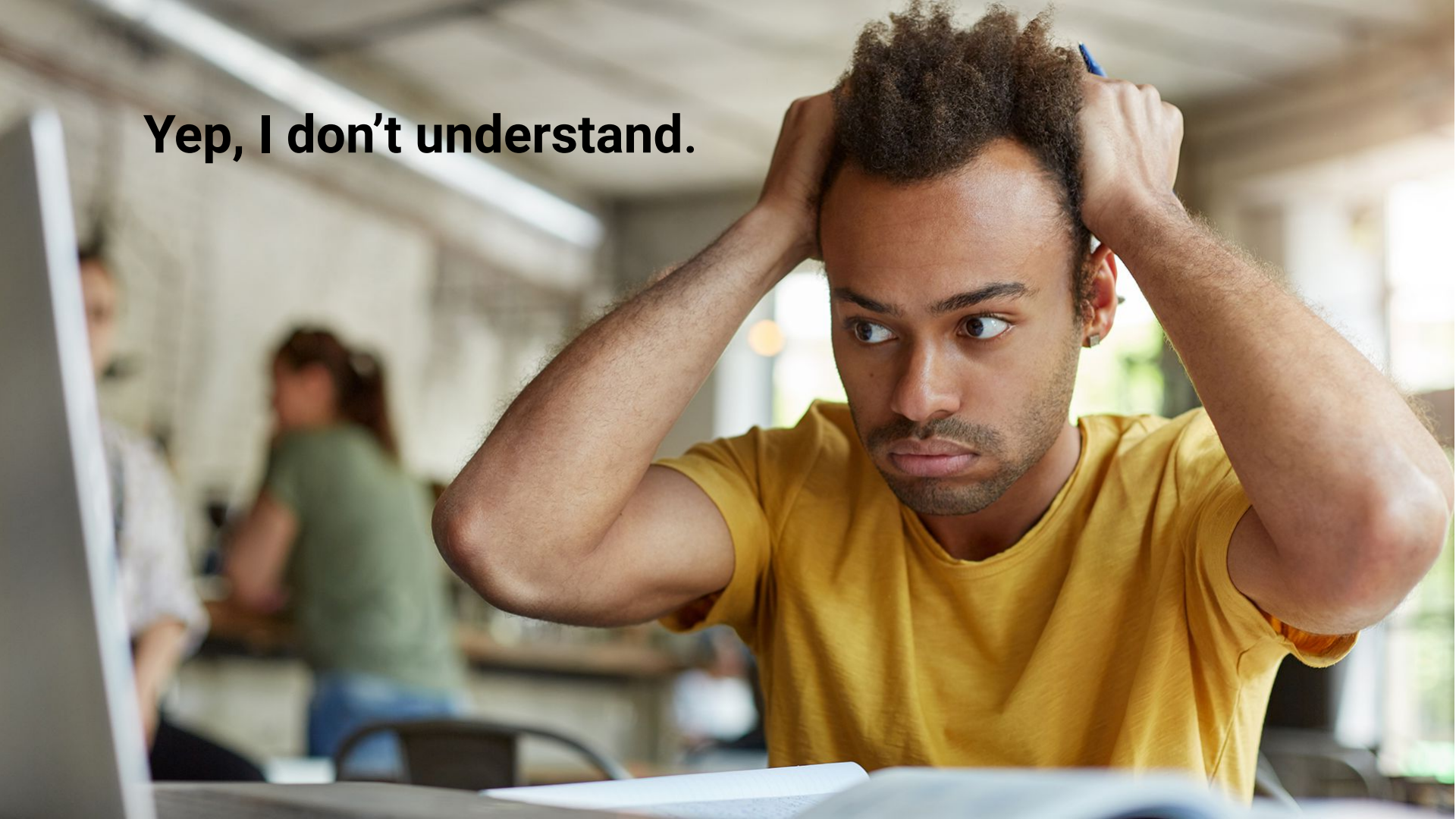
Take a few moments to dissect the code just sent to you.

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.

Time's Up! Let's Review.


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

Time to Code

Brain Teaser

Suggested Time:

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:


The End