GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-15-resources` repo to your computer:
   - Open the terminal
   - `cd` to the `Documents/JSD/JS-SF-15-resources` directory
   - Type **`git pull`** and press **return**
2. In your code editor, open the following folder: `Documents/JSD/JS-SF-15-resources/02-arrays-loops`

ARRAYS & LOOPS

LEARNING OBJECTIVES

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

- Create arrays and access values in them.
- Build iterative loops using `for` statements.
- Iterate over and manipulate values in an array.

AGENDA

- Arrays
- Loops
- Array iterators

WEEKLY OVERVIEW

WEEK 2	Arrays & Loops / Conditionals & Functions
WEEK 3	Scope / Slack bot lab
WEEK 4	Objects & JSON / DOM & jQuery

EXIT TICKET QUESTIONS

1. Are there any tutorials that you recommend regarding the material that was taught today or in future classes?
2. does my git repository need to have the same name everytime? (username.git.) Can I have multiple repositories on my Github?
3. What are the primary syntactical differences between Javascript and other coding languages?
4. Still somewhat confused; what is Node as a program?
5. Am I able to hold Javascript files locally for a web based program I use.

ARRAYS

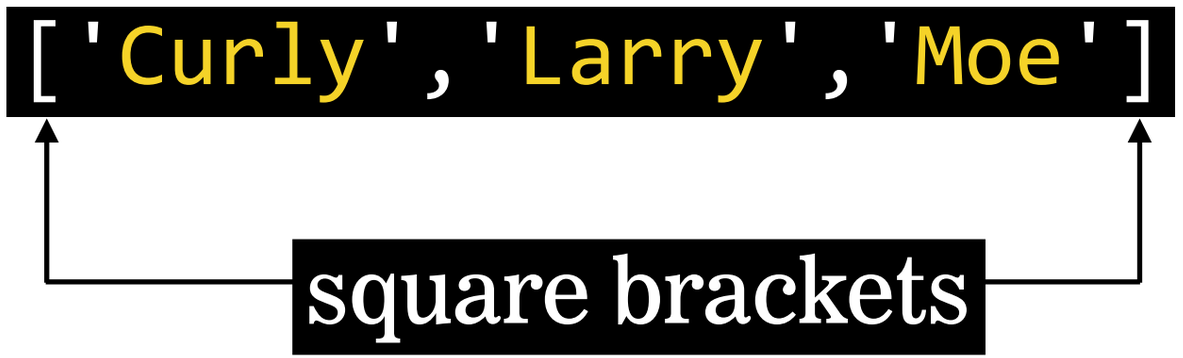
ARRAYS

- An **array** is a collection of data that you can use efficiently

```
['Curly','Larry','Moe']
```

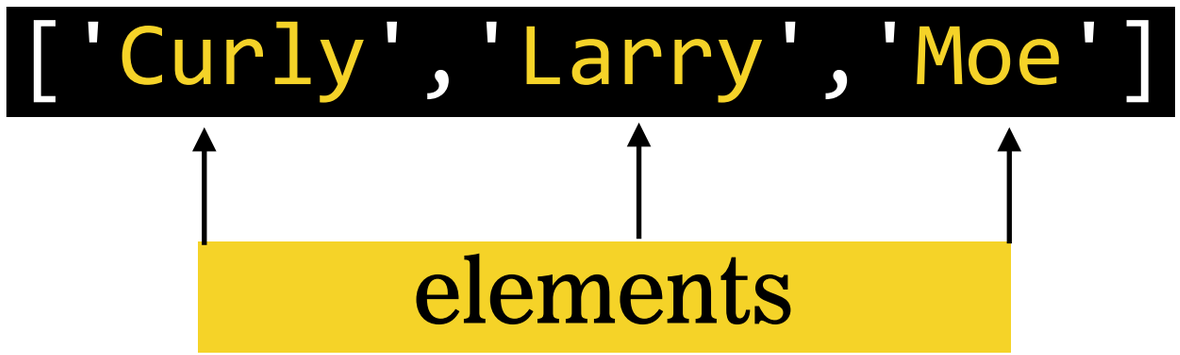
ARRAYS

- An array is enclosed in square brackets [ ]



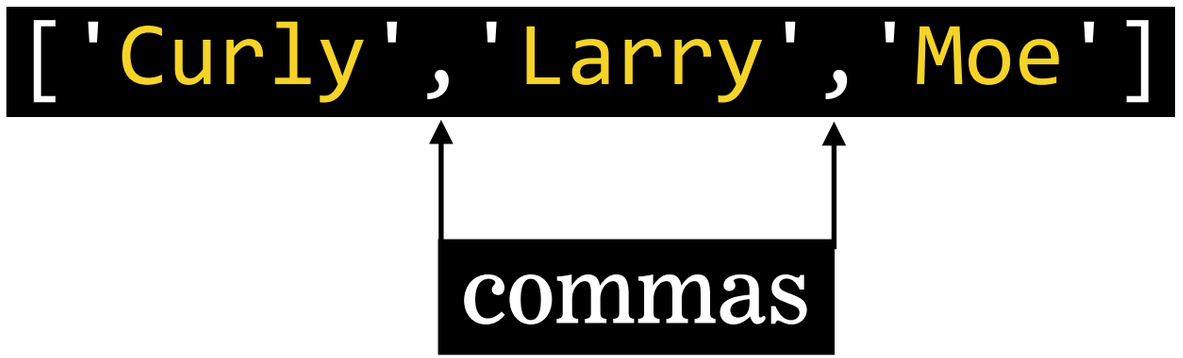
ARRAYS

- Each item in an array is called an **element**
- An element can be any data type



ARRAYS

- Elements are separated by commas



ARRAYS

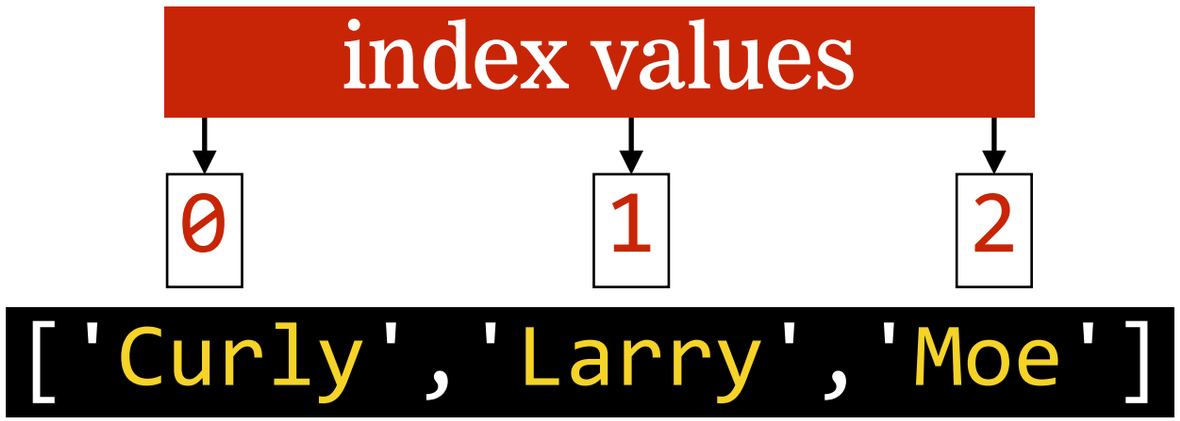
- An array is similar in concept to a list
- Good for storing, enumerating, and quickly reordering data

- Curly
- Larry
- Moe

```
['Curly','Larry','Moe']
```

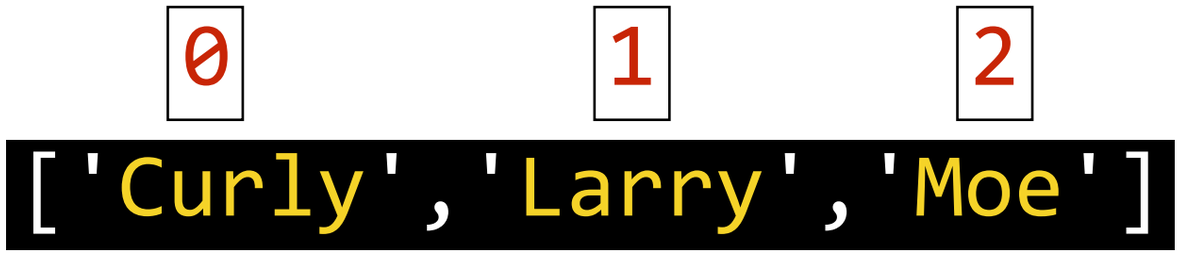
ARRAY INDEX

- Each array element is assigned an **index**, which is a number used to reference that element
- Index starts at 0



ARRAY INDEX

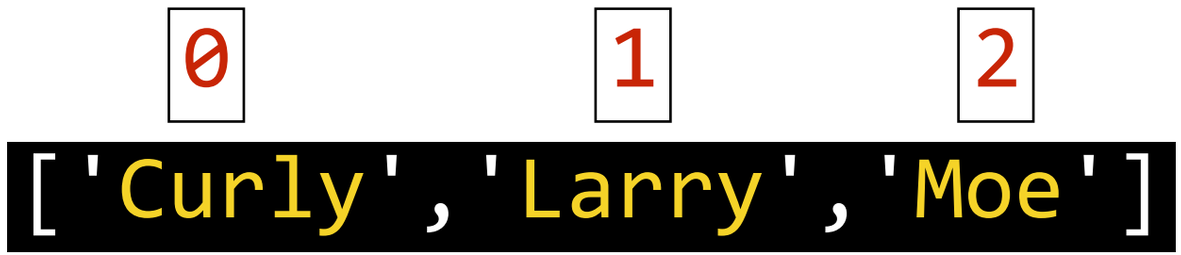
- The final index value is always the length of the array minus 1



Array length	3
-	1
Final index value	**2**

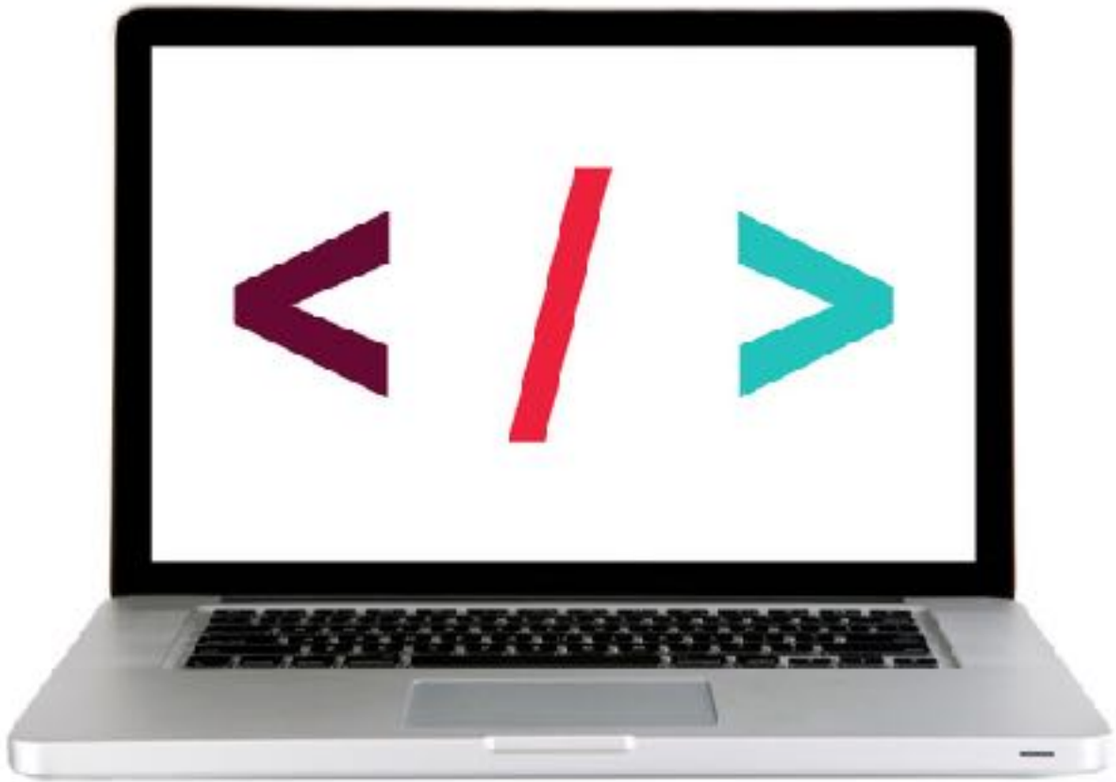
LENGTH PROPERTY

- `length` property is a number 1 greater than the final index number
- `length !==` number of elements in the array



Final index	2
+	1
Value of `length` property	**3**

LET'S TAKE A CLOSER LOOK



LAB — ARRAYS

EXERCISE

TYPE OF EXERCISE

- Individual / Pair

LOCATION

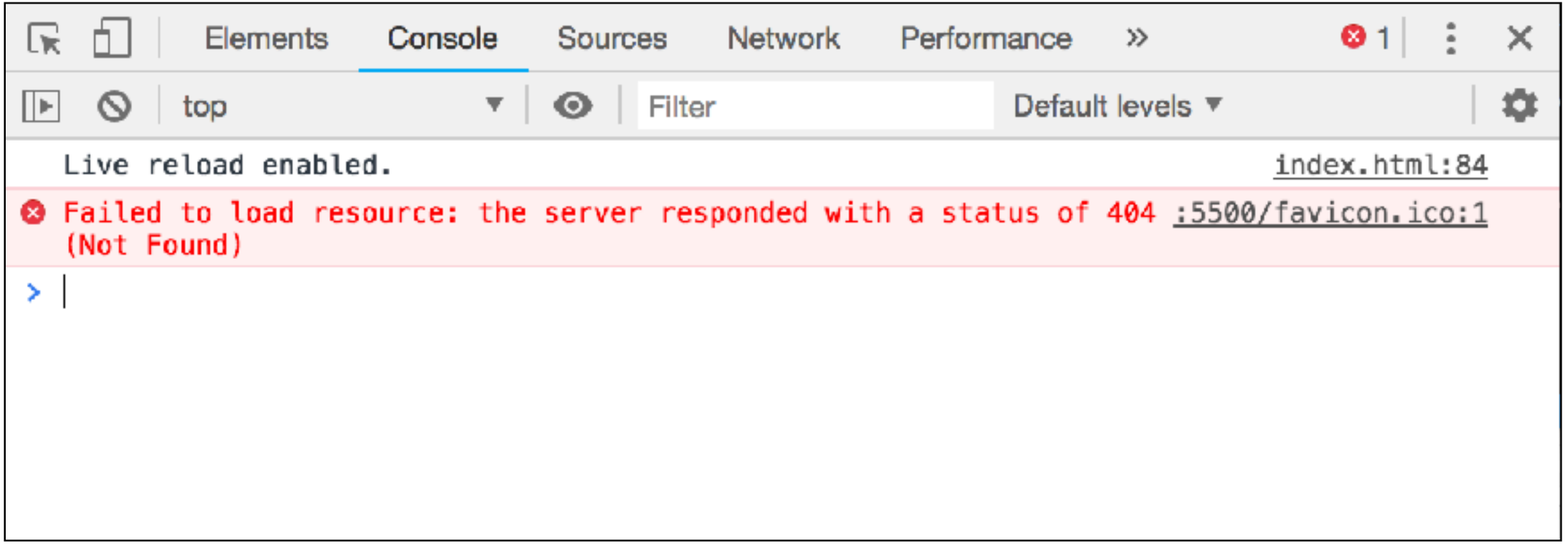
- `starter-code > 0-arrays-loops-exercise`

TIMING

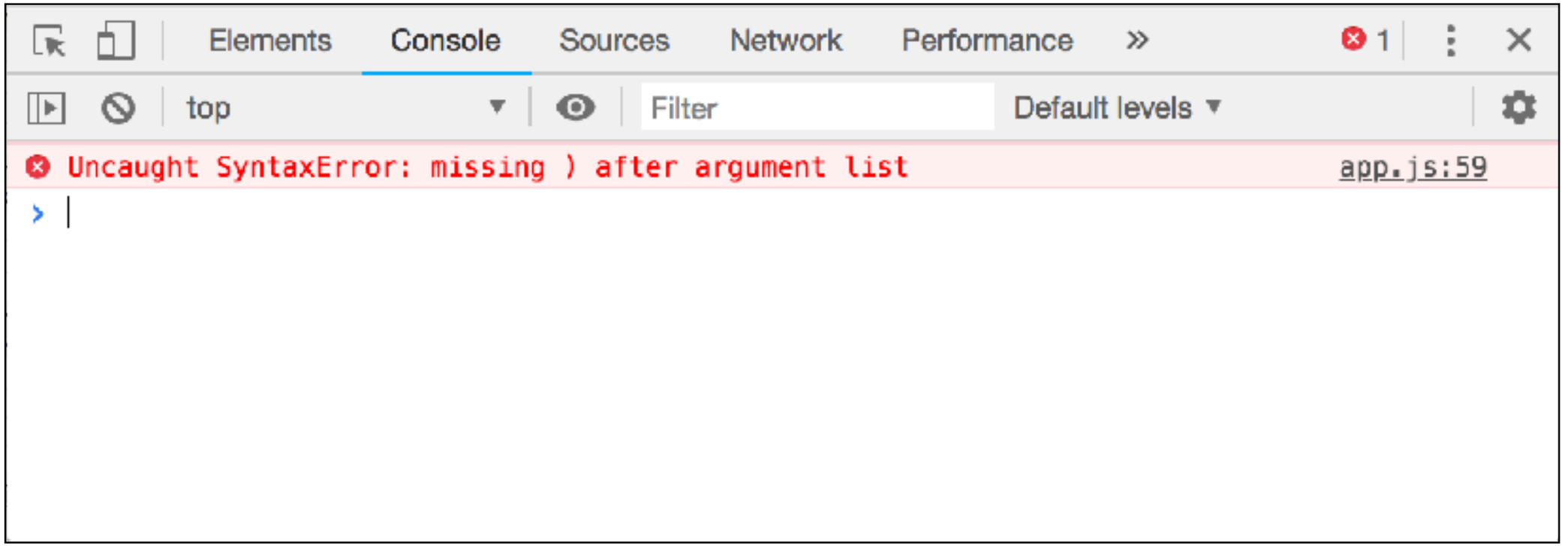
8 min

1. In the `app.js` file, complete questions 1-4.
2. Note that most of your answers should be stored in variables called q1, q2 etc., and the variables printed to the console. See Question 0, which is already completed, for an example.
3. You will work on the remaining questions later in class today.

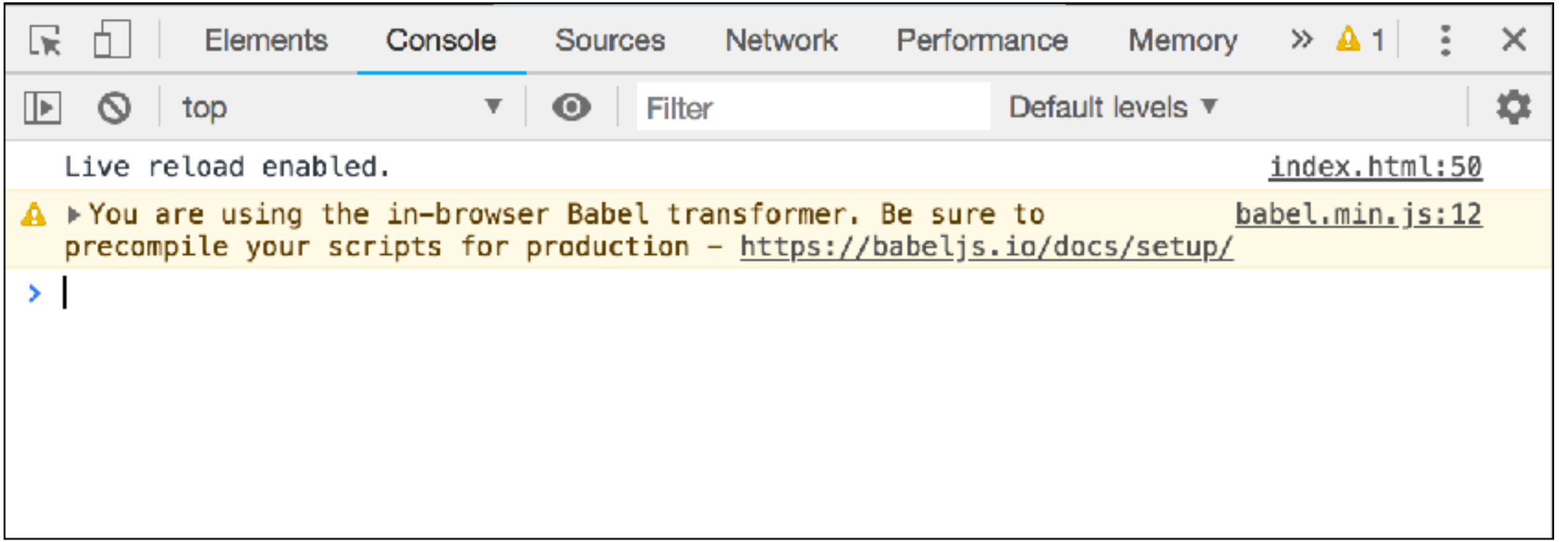
DEBUGGING


Elements
Console
Sources
Network
Performance
1
top
Filter
Default levels
Live reload enabled.
index.html:84
Failed to load resource: the server responded with a status of 404 :5500/favicon.ico:1
(Not Found)
>

DEBUGGING


Elements
Console
Sources
Network
Performance
1
top
Filter
Default levels
Uncaught SyntaxError: missing) after argument list
app.js:59

DEBUGGING

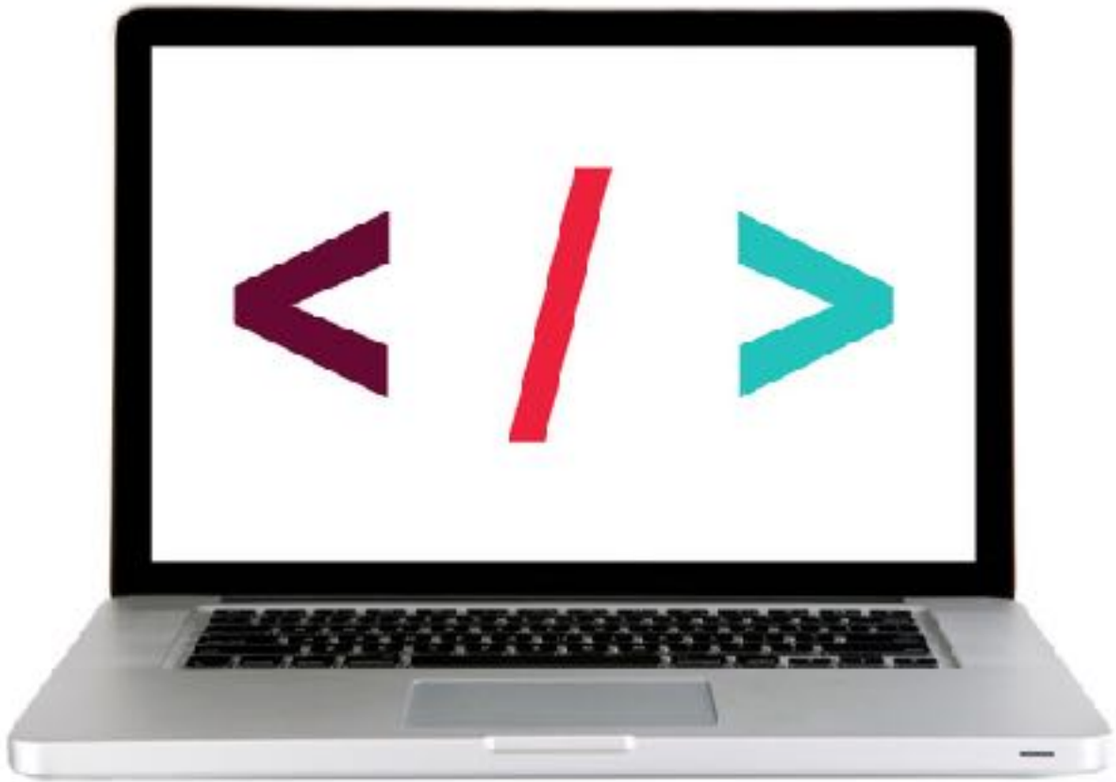

Elements
Console
Sources
Network
Performance
Memory
1
top
Filter
Default levels
Live reload enabled.
index.html:50
You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - https://babeljs.io/docs/setup/
babel.min.js:12

ARRAY HELPER METHODS

ARRAY HELPER METHODS

`toString()`	Returns a single string consisting of the array elements converted to strings and separated by commas
`join()`	Same as `toString()`, but allows you to pass a custom separator as an argument
`pop()`	Removes and returns the item at the end of the array
`push(item1, …, itemN)`	Adds one or more items to the end of the array
`reverse()`	Reverses the array
`shift()`	Removes and returns the item at the start of the array
`unshift(item1, …, itemN)`	Adds one or more items to the start of the array

LET'S TAKE A CLOSER LOOK



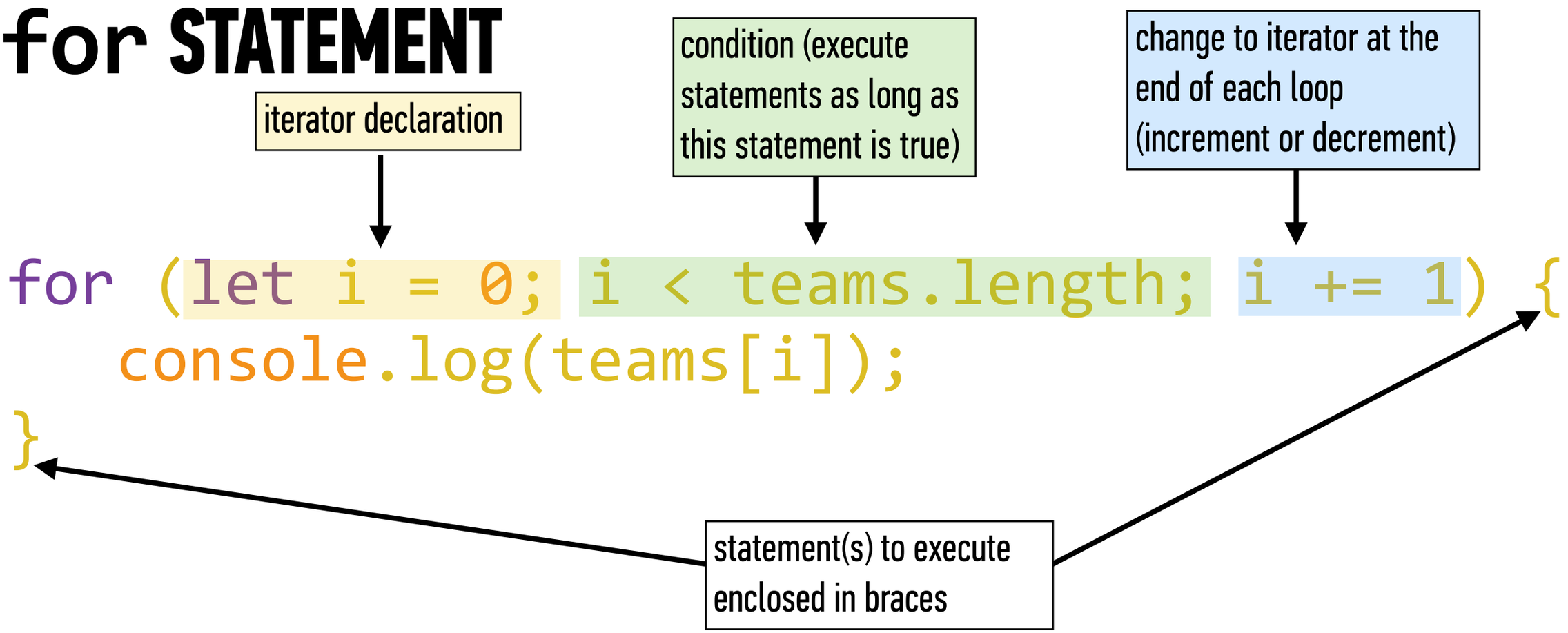
WHY IS THIS AD FUNNY?

FOR LOOPS

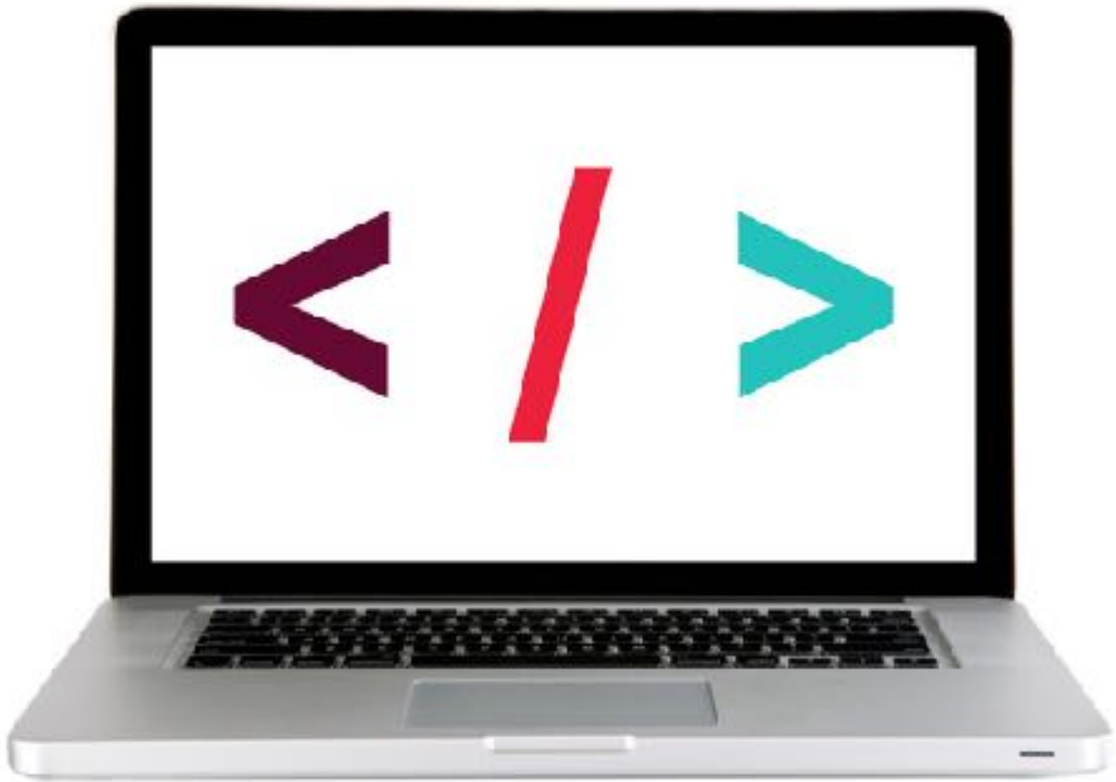
ITERATING

Going through the same process with a bunch of items, one at a time

for STATEMENT



LET'S TAKE A CLOSER LOOK



STRICT MODE

```
"use strict";
```

- Goes at the top of the file
- Tells browsers to be unforgiving in interpreting our code
- Helps us write good code by ensuring that even little mistakes trigger errors

for STATEMENT

```
let fruits = ['apples', 'oranges', 'bananas'];

for (let i = 0; i < fruits.length; i += 1) {
   console.log(fruits[i]);
});
```

result in console:

```
< "apples"
< "oranges"
< "bananas"
```

LAB — FOR LOOPS

EXERCISE

TYPE OF EXERCISE

- Individual / Pair

LOCATION

- `starter-code > 2-loops-exercise`

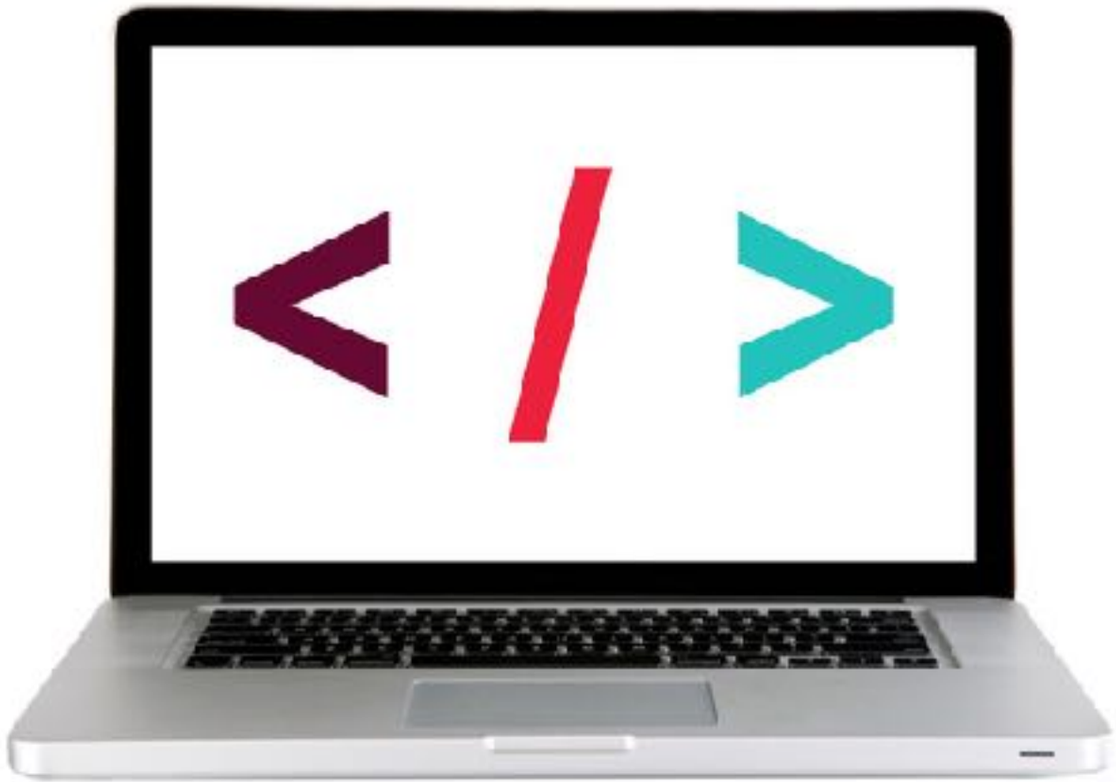
TIMING

10 min

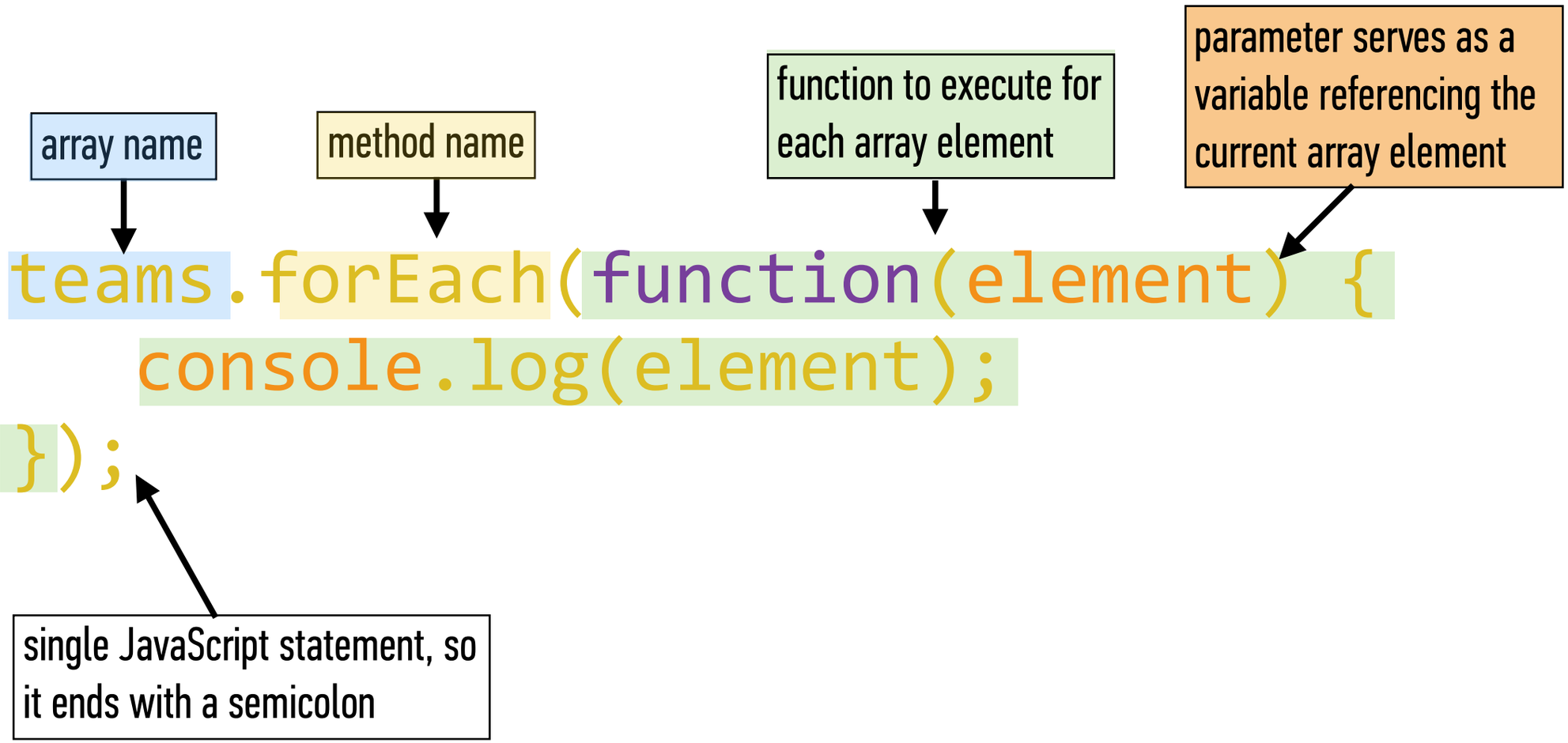
1. Write code that creates a `for` loop that calculates 2 to the 10th power, and console.logs each step of the calculation. (Full instructions in the `app.js` file.)
2. BONUS 1: Rewrite your code to allow a user to enter the exponent value, rather than hard-coding it into your program. (Hint: Read up on the window.prompt method.)
3. BONUS 2: Rewrite your code to use a while loop rather than a `for` loop.
4. BONUS 3: Rewrite your code to use a do/while loop rather than a `for` loop or `while` loop.

ARRAY ITERATOR METHODS

LET'S TAKE A CLOSER LOOK



forEach()


array name
method name
function to execute for each array element
parameter serves as a variable referencing the current array element
teams.forEach(function(element) {
console.log(element);
});
single JavaScript statement, so it ends with a semicolon

forEach() EXAMPLE

```
let teams = ['Bruins', 'Bears', 'Ravens', 'Ducks'];

teams.forEach(function(element) {
    console.log(element);
});
```

LAB — ARRAY LOOPS

EXERCISE

TYPE OF EXERCISE

- Individual / Pair

LOCATION

- `starter-code > 0-arrays-loops-exercise`

TIMING

10 min

1. In the `app.js` file, complete questions 5-6.
2. As in the section you did earlier, your answers should be stored in variables called q1, q2 etc., and the variables logged to the console.
3. Answer these questions using `forEach()` loops, not `for` loops.

ARRAY ITERATOR METHODS

`forEach()`	Executes a provided function once per array element
`every()`	Tests whether all elements in the array pass the test implemented by the provided function
`some()`	Tests whether some element in the array passes the text implemented by the provided function
`filter()`	Creates a new array with all elements that pass the test implemented by the provided function
`map()`	Creates a new array with the results of calling a provided function on every element in this array

LAB — ARRAY LOOPS

EXERCISE

TYPE OF EXERCISE

- Individual / Pair

LOCATION

- `starter-code > 0-arrays-loops-exercise`

TIMING

5 min

1. In the `app.js` file, complete question 7.
2. As in the section you did earlier, your answer should be stored in a variable called q7 and the variable logged to the console.

LAB — PUTTING IT ALL TOGETHER!

EXERCISE

TYPE OF EXERCISE

- Individual / Pair

LOCATION

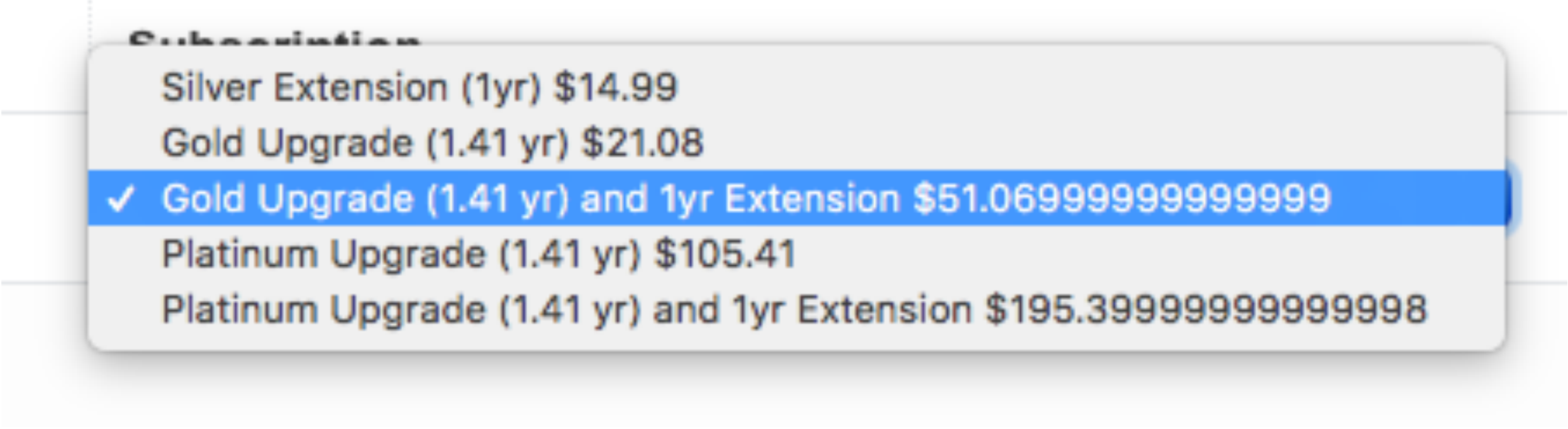
- `starter-code > 4-arrays-loops-exercise-2`

TIMING

until 9:25

1. Write code for a website shopping cart that calculates the sales tax for each item in a cart array and stores the result in a 2nd array. (Full instructions in the `app.js` file.)
2. Calculate the total with tax of all cart items and store the result in a new variable.
3. BONUS: Update your code to round each item to the nearest cent. (Hint: Read up on `Math.round`)
4. BONUS: Rewrite your code to use the `array.map` method.

ROUNDING IS IMPORTANT FOR UX!



Exit Tickets!

(Class #2)

LEARNING OBJECTIVES: REVIEW

- Create arrays and access values in them.
- Build iterative loops using `for` statements.
- Iterate over and manipulate values in an array.

Next class preview: Conditionals & Functions

- Use Boolean logic to combine and manipulate conditional tests.
- Use `if/else` conditionals to control program flow based on Boolean tests.
- Differentiate among true, false, truthy, and falsy.
- Describe how parameters and arguments relate to functions
- Create and call a function that accepts parameters to solve a problem
- Define and call functions defined in terms of other functions
- Return a value from a function using the `return` keyword
- Define and call functions with argument-dependent return values

Q&A