

# JAVASCRIPT DEVELOPMENT

*Sasha Vodnik, Instructor*

# HELLO!

1. Pull changes from the `svodnik/JS-SF-14-resources` repo to your computer:
  - Open the terminal
  - `cd` to the `Documents/JSD/JS-SF-14-resources` directory
  - Type **`git pull`** and press **return**
2. In your code editor, open the following folder:  
`Documents/JSD/JS-SF-14-resources/02-arrays-loops`

---

**JAVASCRIPT DEVELOPMENT**

---

# **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

---

## ARRAYS & LOOPS

---

# WEEKLY OVERVIEW

**WEEK 2**

(holiday) / Arrays & Loops

**WEEK 3**

Conditionals & Functions / Scope

**WEEK 4**

Slack bot lab / Objects & JSON

## **EXIT TICKET QUESTIONS**

1. I know a lot of the stuff already taught, although I appreciate setting a solid foundation for future lessons.
2. (paraphrase) Group exercises don't always seem necessary, but expect they'll be especially useful in the later classes.
3. I'm sure this will be covered but I want to learn more about the console within the browser and how that will be useful in a practical setting.
4. quick recap on git in terminal

## **BASIC GIT WORKFLOWS**

**To add your local changes to a remote repo:**

```
$ git add .
```

```
$ git commit -m "description of changes"
```

```
$ git push origin master
```

**To download other people's changes from a remote repo:**

```
$ git pull
```



# **ARRAYS**

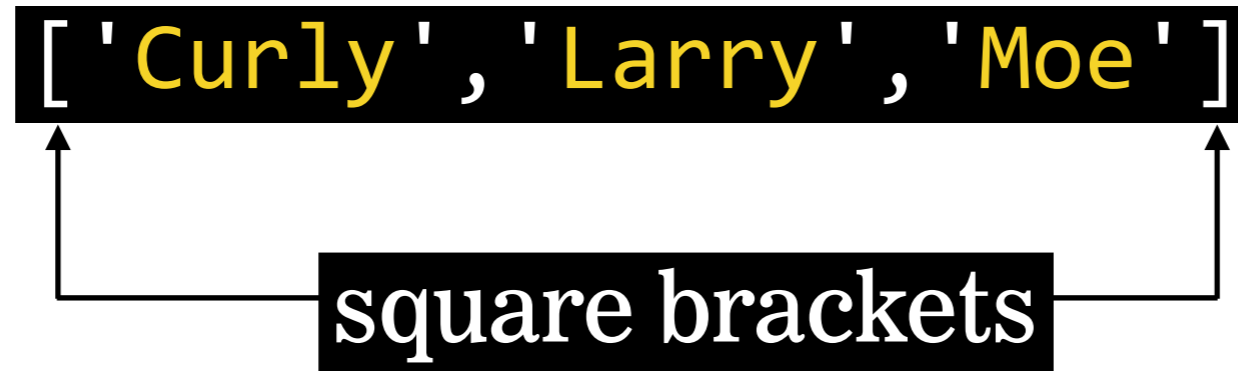
# ARRAYS

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

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

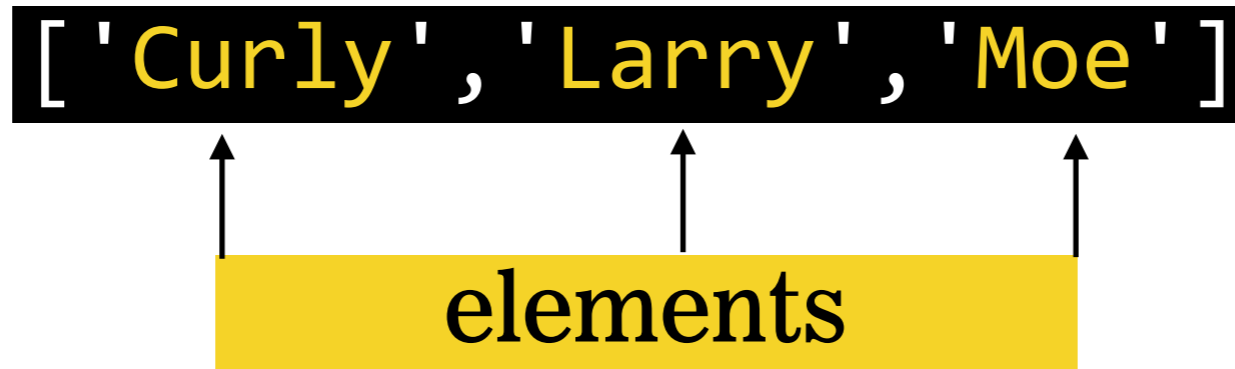
# ARRAYS

- An array is enclosed in square brackets [ ]



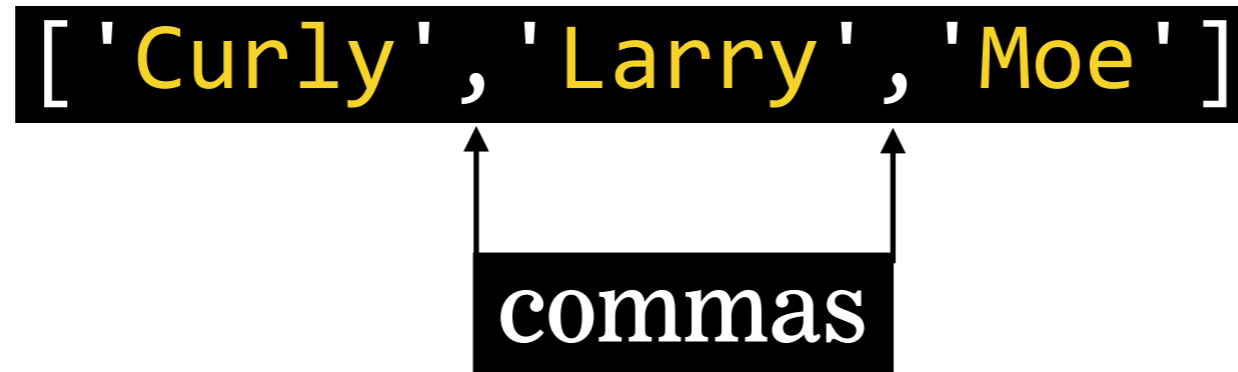
# ARRAYS

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



# ARRAYS

- Elements are separated by commas



# ARRAYS

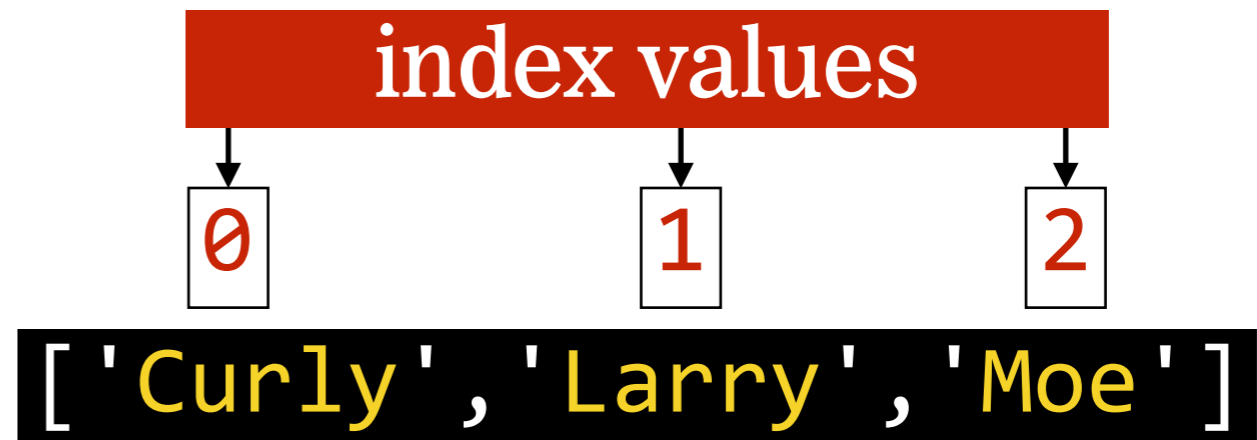
- An array is similar in concept to a list
- Good for storing, enumerating, and quickly reordering data

- 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        |
| <hr/>                    |          |
| <b>Final index value</b> | <b>2</b> |



# 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

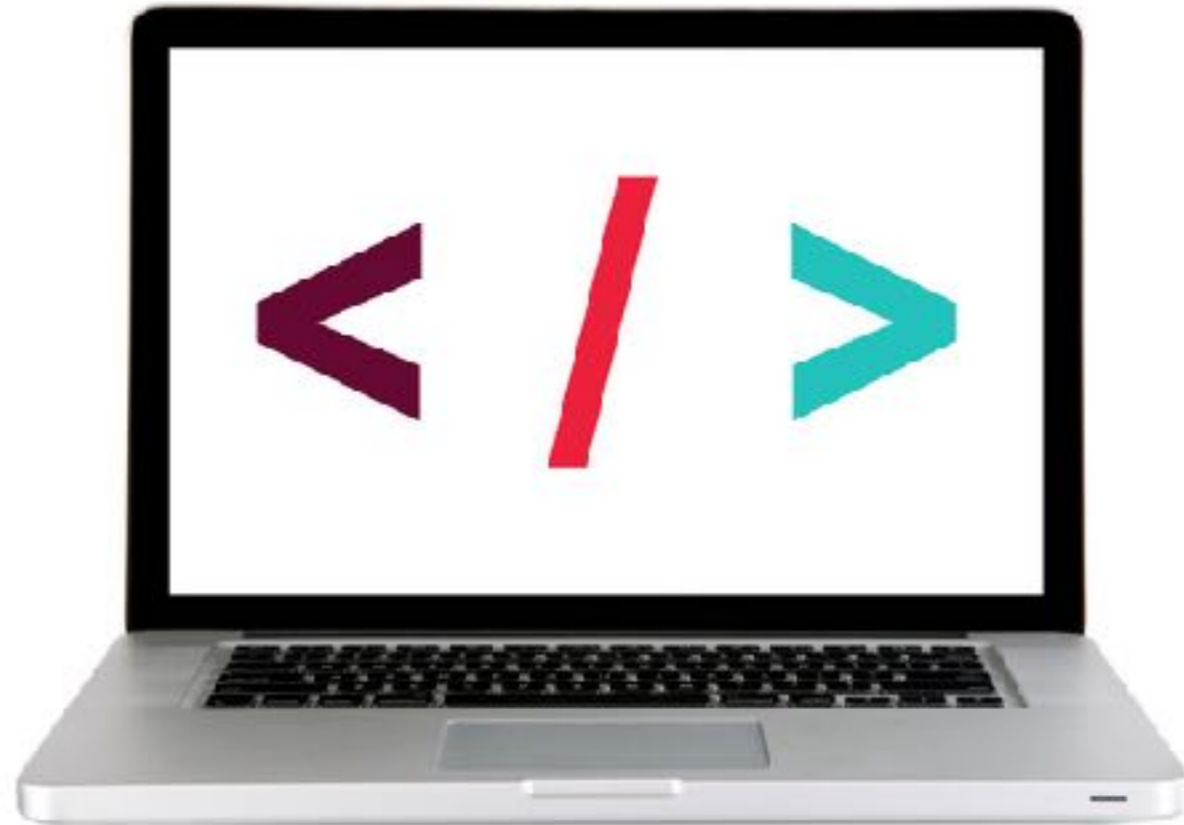
---

**Value of length property 3**

---

**LET'S TAKE A CLOSER LOOK**

---



# LAB — ARRAYS

---



## **TYPE OF EXERCISE**

---

‣ Individual / Pair

## **LOCATION**

---

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

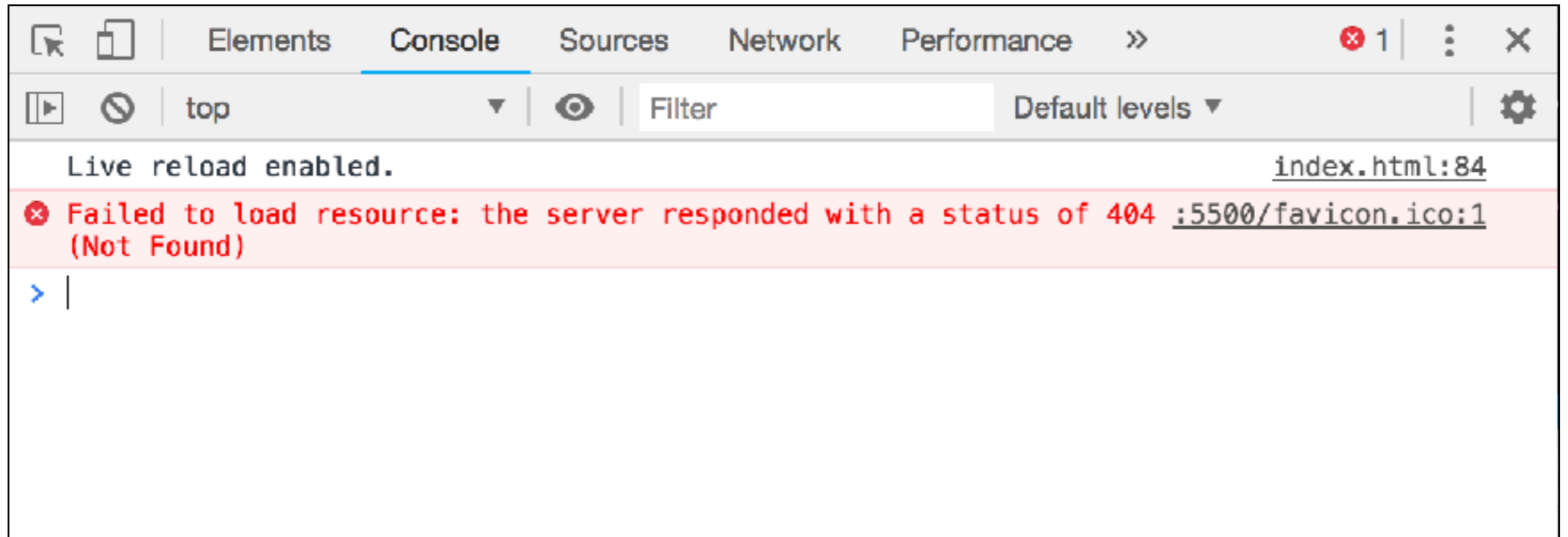
## **TIMING**

---

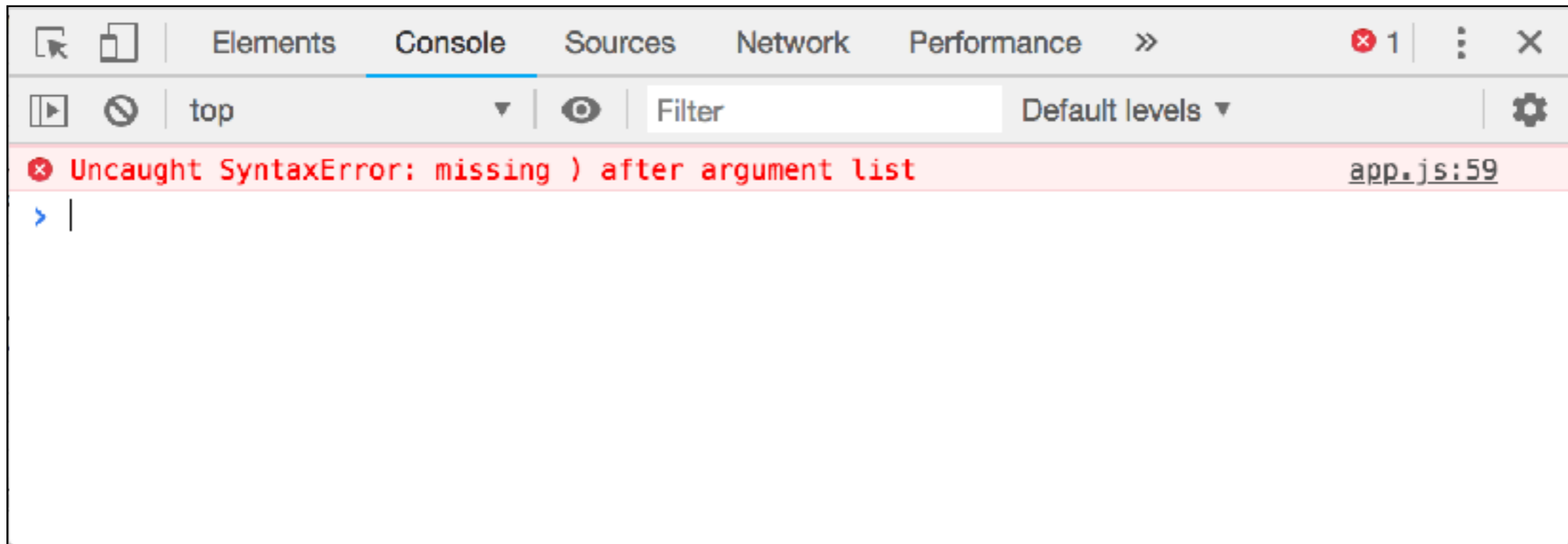
*8 min*

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.

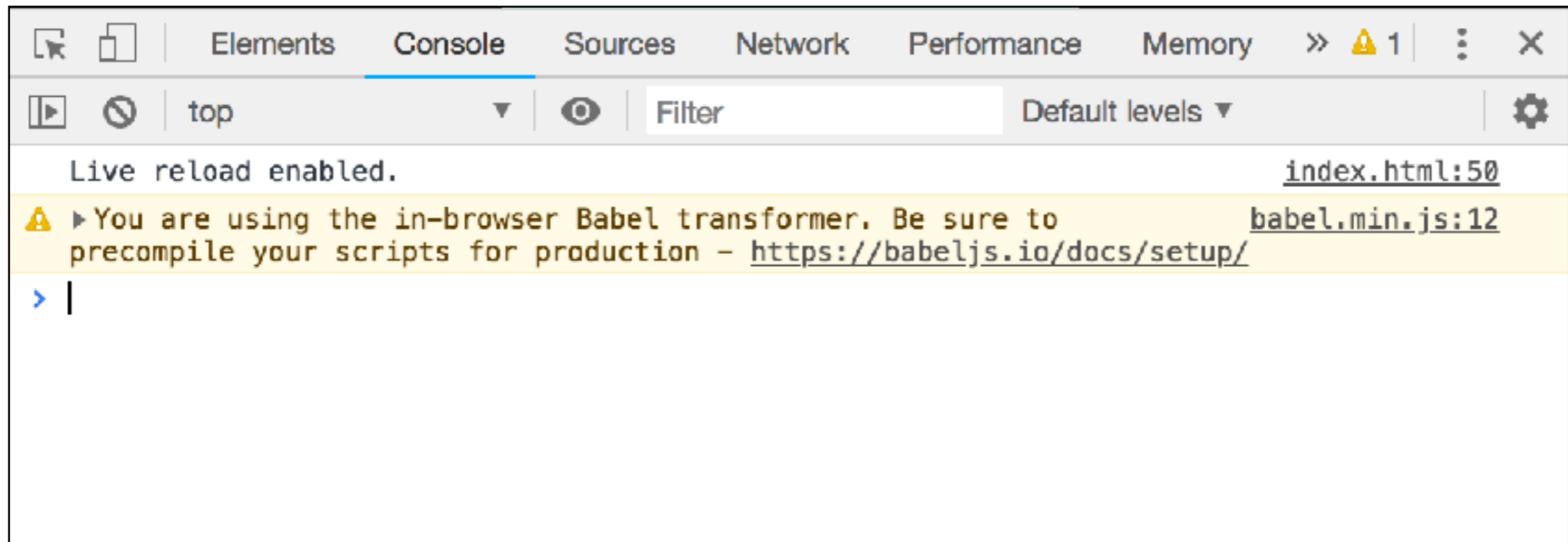
# DEBUGGING



# DEBUGGING



# DEBUGGING



# **ARRAY HELPER METHODS**

# ARRAY HELPER METHODS

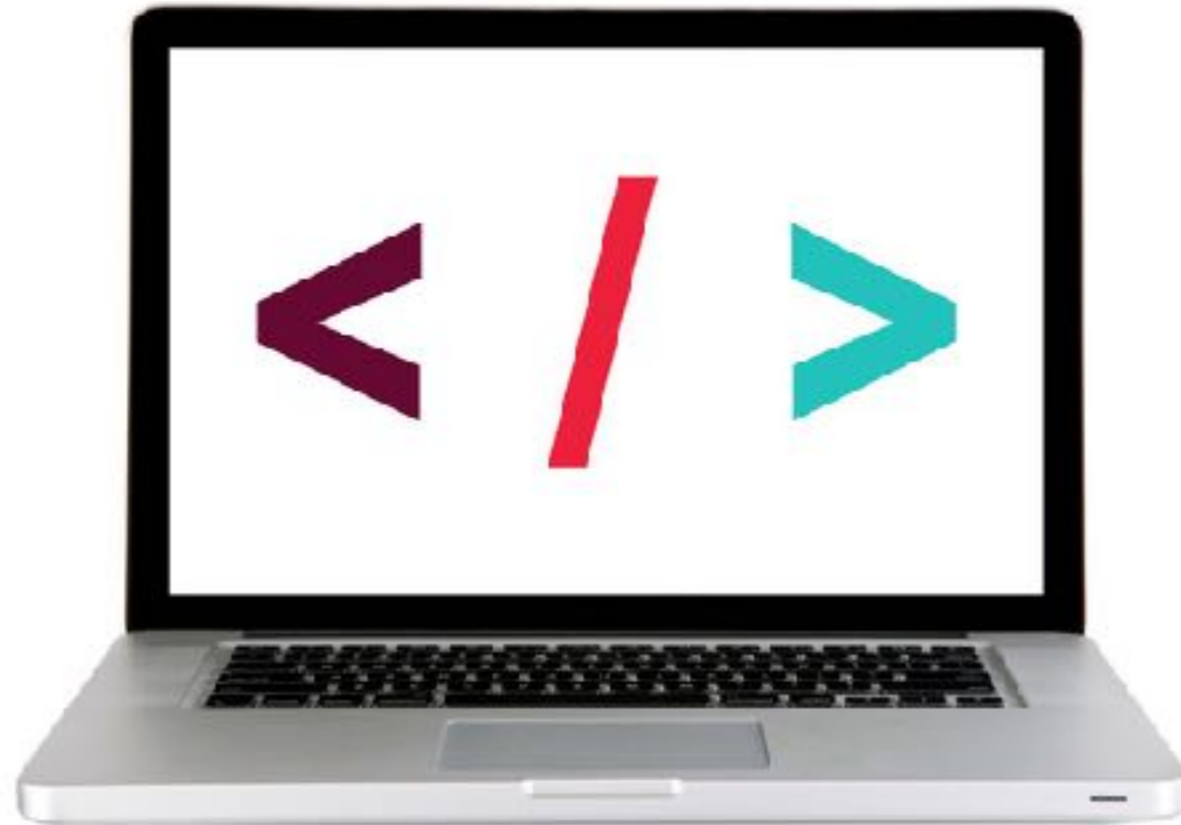
|   |   |
|---|---|
| <code>toString()</code>                 | Returns a single string consisting of the array elements converted to strings and separated by commas |
| <code>join()</code>                     | Same as <code>toString()</code> , but allows you to pass a custom separator as an argument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



---

**LET'S TAKE A CLOSER LOOK**

---



## WHY IS THIS AD FUNNY?



```
> vanilla = ["cream","milk",  
  "sugar","eggs","vanilla"];  
<- ["cream","milk","sugar",  
  "eggs","vanilla"]  
> vanilla.join();  
<- "ääh"
```

**ääh**



# **FOR LOOPS**

# **ITERATING**

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

# for STATEMENT

iterator declaration

condition (execute statements as long as this statement is true)

change to iterator at the end of each loop (increment or decrement)

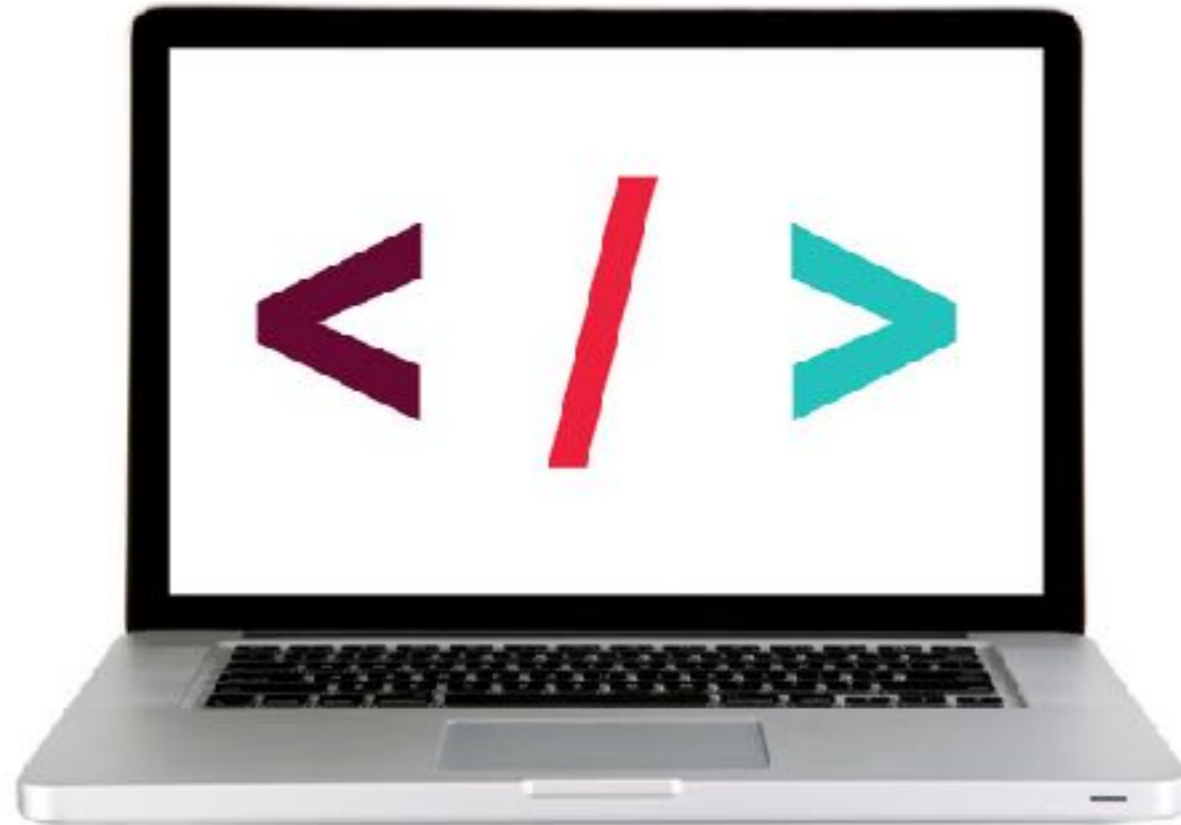
```
for (let i = 0; i < teams.length; i++) {
  console.log(teams[i]);
}
```

statement(s) to execute enclosed in braces

---

**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++) {  
  console.log(fruits[i]);  
});
```

result in console:

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



# LAB — FOR LOOPS

---



## TYPE OF EXERCISE

‣ Individual / Pair

## LOCATION

‣ starter-code > 2-loops-exercise

## TIMING

*10 min*

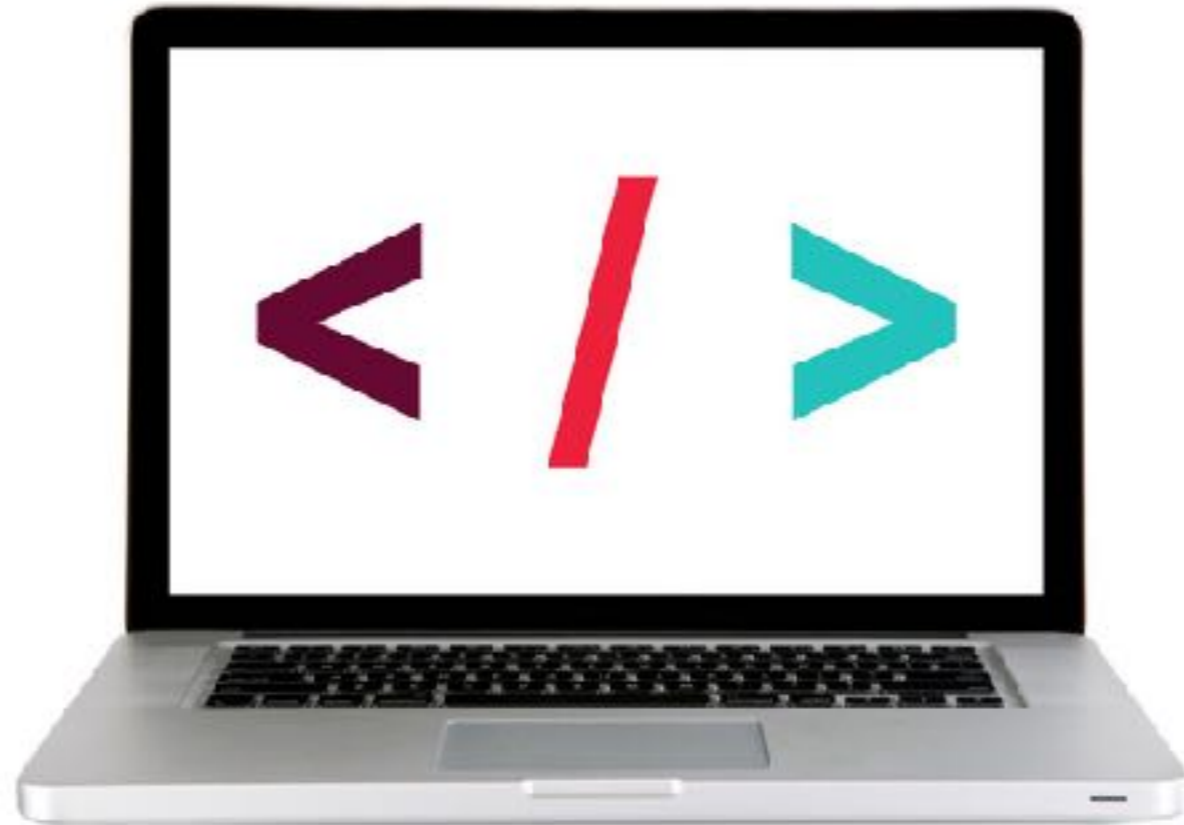
1. Write code that creates a for loop that calculates 2 to the 10th power, and console.logs each step of the calculation. (Full instructions in the `app.js` file.)
2. BONUS 1: Rewrite your code to allow a user to enter the exponent value, rather than hard-coding it into your program. (Hint: Read up on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

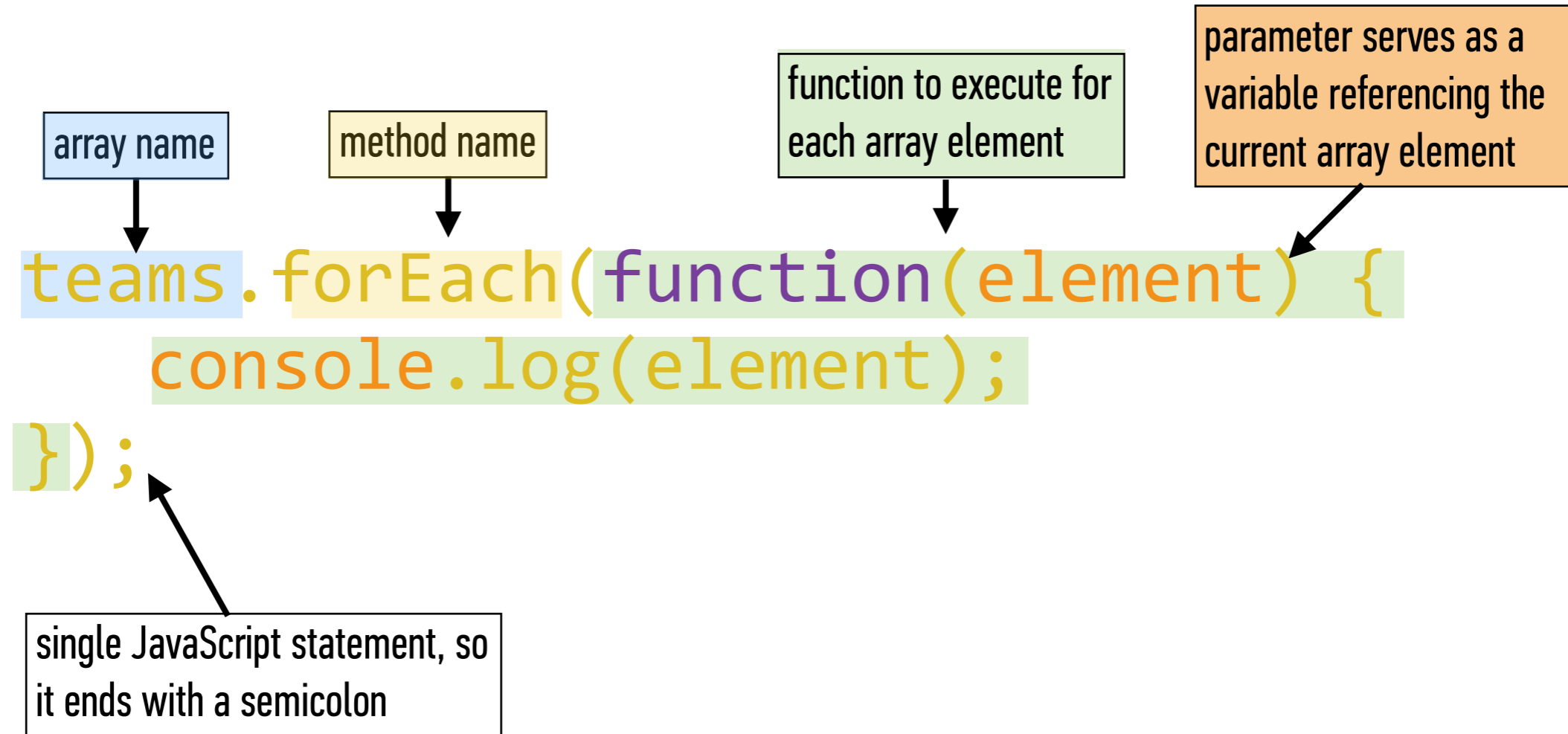
---

**LET'S TAKE A CLOSER LOOK**

---



# forEach()



## forEach() EXAMPLE

```
let teams = ['Bruins', 'Bears', 'Ravens', 'Ducks'];  
  
teams.forEach(function(element) {  
    console.log(element);  
});
```

# LAB — ARRAY LOOPS

---



## **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

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

# LAB — ARRAY LOOPS

---



## **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!

---



## TYPE OF EXERCISE

‣ Individual / Pair

## LOCATION

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

## TIMING

*until 9:20*

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!

## Subscription

Silver Extension (1yr) \$14.99

Gold Upgrade (1.41 yr) \$21.08

✓ Gold Upgrade (1.41 yr) and 1yr Extension \$51.069999999999999

Platinum Upgrade (1.41 yr) \$105.41

Platinum Upgrade (1.41 yr) and 1yr Extension \$195.399999999999998

# **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**