Web Development

Day 7: More JavaScript

Functions, Booleans & Conditionals

Maydm

Icebreaker!

Maydm

Today's Schedule

Morning:

- Review of JS so far
- Booleans
- Functions
- If Statements
- Pseudocode

Afternoon:

- Project: Mastermind
- Mentors visit

Maydm

JavaScript So Far

- Comments
- Variables
- Data Types
- Operating on Variables
- Template Literals

Functions

In programming, a **function** is a set of statements that performs a task or solves a problem.

```
function printName(firstName, lastName) {
    console.log(firstName + ‘ ‘ + lastName);
}
```

Maydm

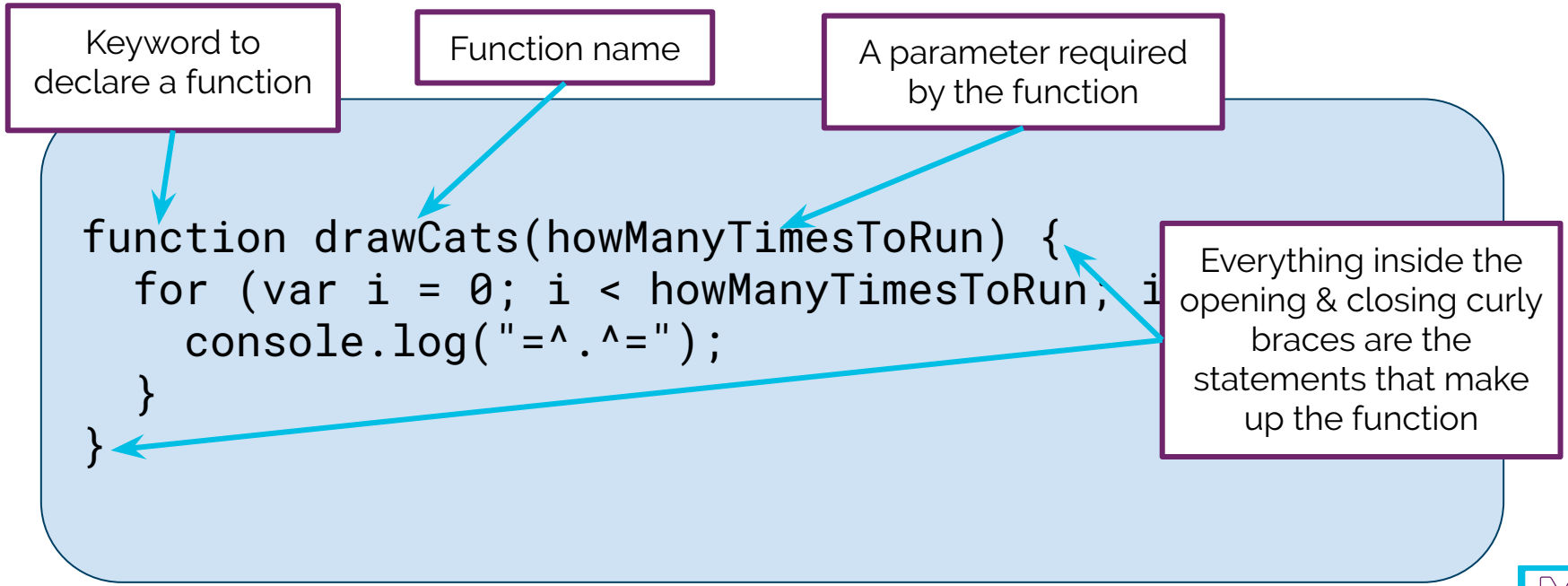
Functions

Just like **var** declares a variable, the **function keyword** declares a function. What do you think this function does?

```
function printName(firstName, lastName) {
    console.log(firstName + ‘ ‘ + lastName);
}
```

Maydm

Anatomy of a JS Function



Maydm

Naming a Function

Naming functions (and variables) is important!

Giving functions (and variables) good names helps:

- You understand what problem you're trying to solve
- Other people who might be looking at your code

camelCase

In JavaScript, variables and functions are often given long names that join words using camelCase. In camelCase, the first letter is lowercase and any other words capitalized.

```
var firstName = ‘SpongeBob’;
var lastName = ‘SquarePants’;

printName(firstName, lastName);
```

Maydm

Parameters

Sometimes a function needs some data that will be given when the function executes.

Parameters are variables that are defined as part of the function.

Parameters

In the following function, there are two parameters defined: **firstName** and **lastName**. Inside the function, the same variable names are used as placeholders for the parameters that will be given later when the function is executed.

```
function printName(firstName, lastName) {
    console.log(firstName + ‘ ‘ + lastName);
}
```

Maydm

Executing Functions

In order to execute a function, the function name is given as a statement, along with any parameters that may be required, like so:

```
function printName(firstName, lastName) {
    console.log(firstName + ‘ ‘ + lastName);
}

printName(‘Jane’, ‘Doe’);

// returns ‘Jane Doe’
```

Practice with Functions

Work through the Day 7 “Functions” exercises on JS Bin.

What can functions do?

Whatever you tell them to do!

Project: Wacky Words

Open the “Wacky Words” starter code and fork the pen.

We’ll read through the code together, starting with the HTML. Reading code someone else wrote is a regular part of being a developer.

Operators

You've been using some basic operators already in JavaScript. Now let's talk about operators that let us use logic.

Operators

- Arithmetic operators
  - +, -, *, /
  - Increment: x++ ➡️ the same as x+1
  - Decrement: x-- ➡️ the same as x-1
- Assignment operators
  - x = 0 ➡️ single equal sign is used in assigning variables
  - x += y ➡️ the same as x = x + y
  - x -= y ➡️ the same as x = x - y
  - x *= y ➡️ the same as x = x * y
  - x /= y ➡️ the same as x = x / y

Maydm

Operators

- Comparison operators
  - Equals: **==** ➡️ checks to see if values are equal

    Example:

    'hello' == 'Hello' ➡️ true

    5 == '5' ➡️ true
  - Strict equals: **===** ➡️ checks to see if values and type are equal

    Example:

    'hello' === 'Hello' ➡️ false

    5 === '5' ➡️ false

Maydm

Operators

- Comparison operators
  - Not Equal: **!=** ➡️ checks to see if values are not equal

    Example:

    'hello' != 'Hello' ➡️ false

    4 != '5' ➡️ true

Maydm

Operators

- Comparison operators
  - Greater than: **>** ➡️ is value on left greater than value on right

    Example:

    4 > 5 ➡️ false

    10 > 0 ➡️ true
  - Greater than or equal to: **>=**

    Example:

    5 >= 5 ➡️ true

    10 >= 15 ➡️ false

Maydm

Operators

- Comparison operators
  - Less than: < ➡️ is value on left less than value on right

    Example:

    4 < 5 ➡️ true

    10 < 0 ➡️ false
  - Less than or equal to: >=

    Example:

    5 <= 5 ➡️ true

    10 <= 15 ➡️ true

Maydm

Practice with Comparison Operators

Work through the Day 7 “Comparison Operators” exercises on JS Bin.

Maydm

What is logic?

Logic is a system of reasoning.

In coding, **logic** is how the program determines whether to execute a particular block of code.

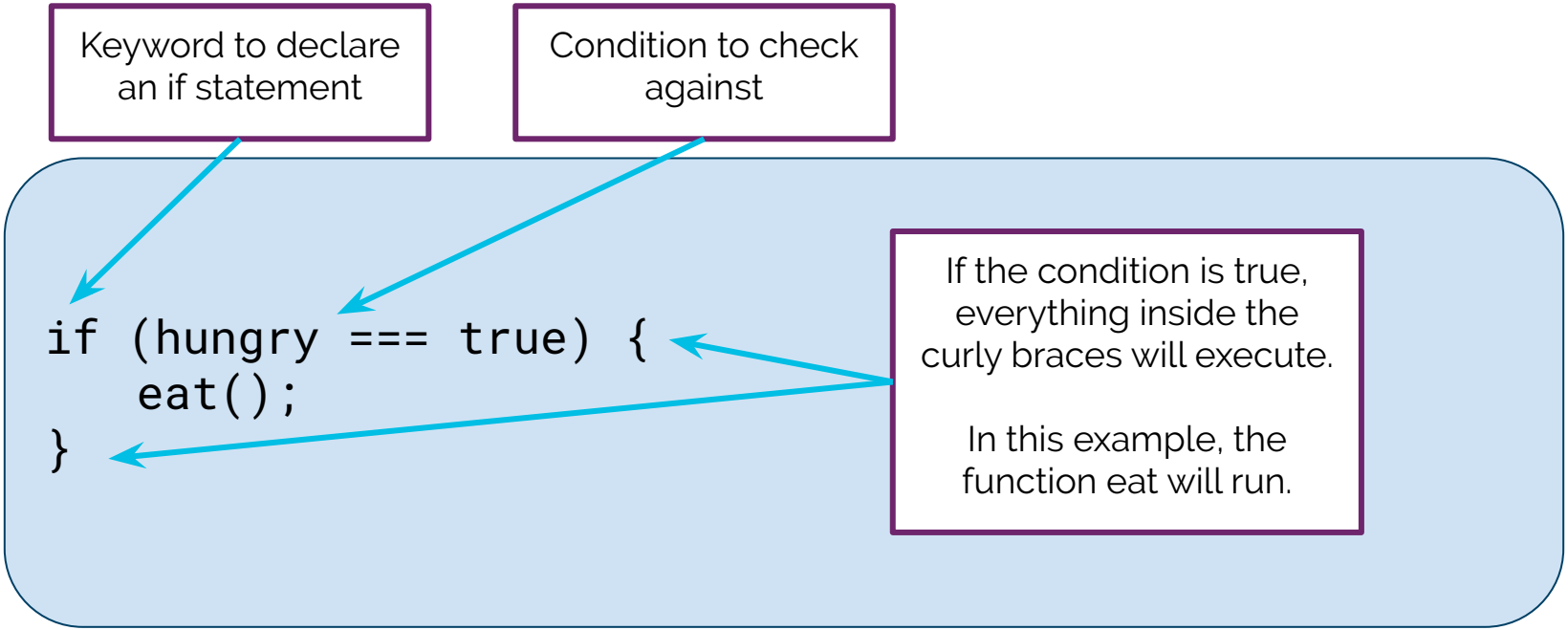
Maydm

What do we mean by logic?

An **if** block only runs if the condition is true. If that condition is **false,** the if block doesn't run at all.

A **condition** can be anything that is a Boolean, meaning it is **true or false**. Conditions involve values being equal to, greater than, or less than other values.

Anatomy of an If Statement



Maydm

Practice with If Statements

Work through the Day 7 “If Statements” exercises on JS Bin.

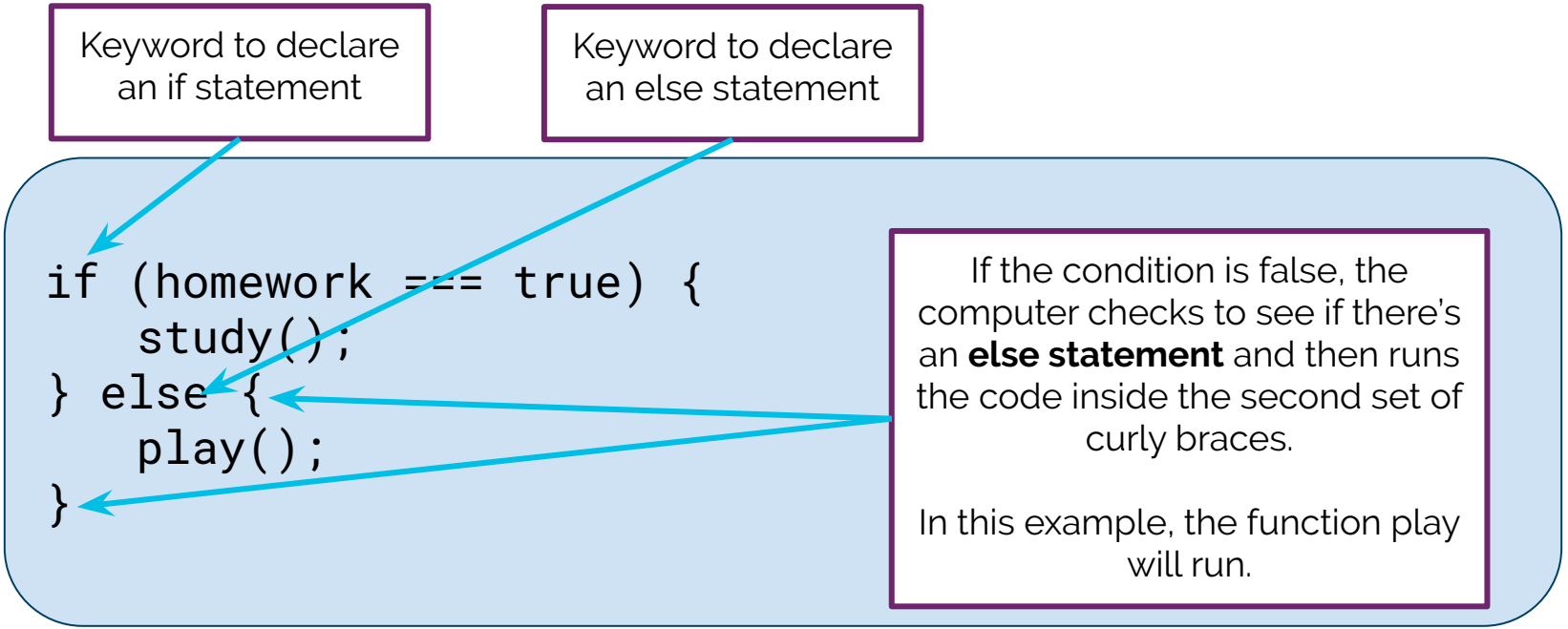
Maydm

If / Else Statements

An **if/else** block checks for the specific **condition**.

If that condition is **true**, it executes the first part of the block. If the condition not true (meaning false), it executes the second part, the **else**, of the block.

Anatomy of an If / Else Statement



Maydm

Practice with If / Else Statements

Work through the Day 7 “If / Else Statements” exercises on JS Bin.

Maydm

Pseudocode

Pseudocode is plain text that represents actual code.

It's useful for thinking about how you want to approach a problem before you start coding it.

Maydm

Project: Mastermind

Open the “Mastermind” starter code and fork the pen.

Just as before, we’ll read through the code together, starting with the HTML.

Arrays

Sometimes you need a list of related values, such as inventory for sale or grades.

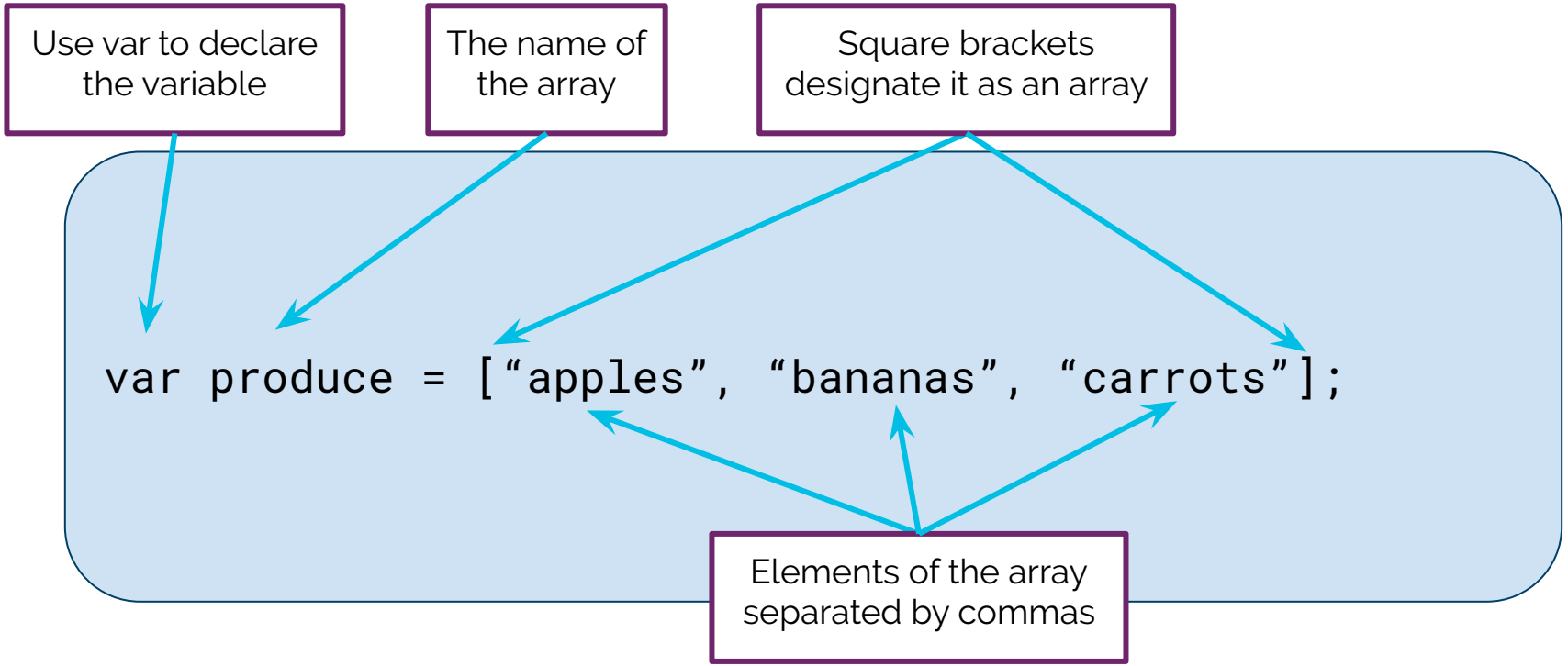
Arrays are a type of variable that are able to store multiple values. The values can be strings, numbers, Booleans, or any other data type.

var produce = [“apples”, “bananas”, “carrots”, “potatoes”];

var quizScores = [90, 88, 95];

Maydm

Anatomy of an Array



Maydm

Accessing Array Elements

Array elements can be accessed through their **index number**.

But, what is an **index**?

In programming, an **index** is a number used to indicate position in a list. Each item in an array has an index number based on its position.

Maydm

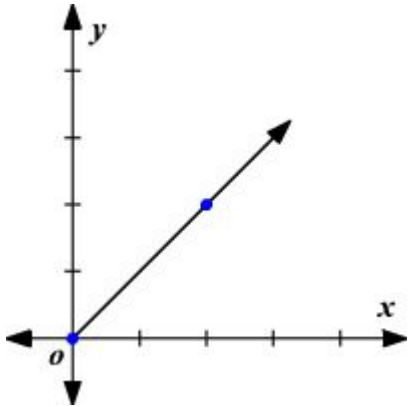
Accessing Array Elements

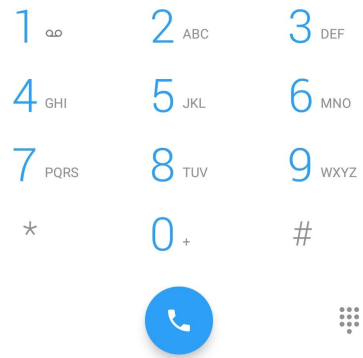
In programming, 0 is the first number.

Arrays follow the same convention, so their first element is at the zeroth (0th) index.

Where else do you see 0 as the starting number?

Zeroes





Maydm

Accessing Array Elements

Use the array name with square brackets to get an array element, like this:

```
var produce = [“apples”, “bananas”, “carrots”];

console.log(produce[0]); // returns ‘apples’
```

Maydm

Arrays

Arrays are great for holding related values, plus arrays include methods to operate on those values!

Methods are built-in functions that belong to specific objects and can only be used by those objects.

Array methods let us find elements within an array; add, remove, or replace elements; convert an array to a string; and much more.

Array Methods

We'll only focus on a few methods to start:

- Getting the number of elements in an array (**.length**)
- Remove the last element (**.pop**)
- Add a new element to the end (**.push**)
- Add or replace an element at a specific point (**.splice**)
- Merging arrays (**.concat**)

Methods

You've already been using one method already without knowing it:

```
console.log();
```

Anatomy of a Method

Arguments for the method, if required.

In this case, console.**log** will print the argument to the console.

The array that owns the method.

Notice, the

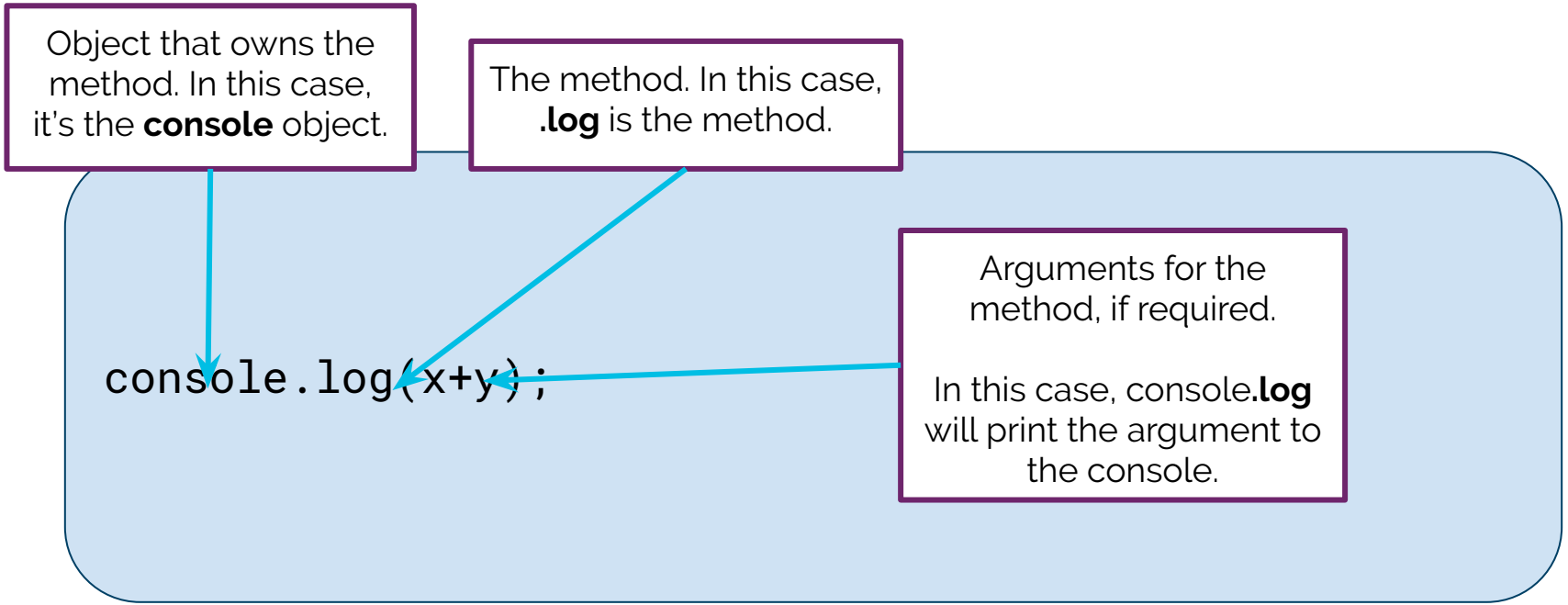
```
var produce = [“apples”, “bananas”, “carrots”];

produce.length();
```

The method. In this case, **length** is the method.

Maydm

Anatomy of an Array Method



Maydm

Array Methods: .length

Sometimes you want to know how many items are in an array, known as the **length** of the array. It takes no arguments.

```
var produce = [“apples”, “bananas”, “carrots”];

console.log(produce.length()); // returns 3
```

Maydm

Array Methods: .pop

To remove the last element in the array, use the **pop** method. It takes no arguments.

```
var produce = [“apples”, “bananas”, “carrots”];

produce.pop();

console.log(produce); // returns [“apples”, “bananas”]
```

Maydm

Array Methods: .push

To add an element to the end of the array, use the **push** method. It takes **one argument**: the value being added to the array.

```
var produce = [“apples”, “bananas”, “carrots”];

produce.push(‘grapes’);

console.log(produce);
// returns [“apples”, “bananas”, “carrots”, “grapes”]
```

Maydm

Array Methods: .splice

Splice can add an element at a specific point. It takes three arguments: the **index** to change, the **number** of elements to overwrite, and the **value** to add.

```
var produce = [“apples”, “carrots”];

produce.splice(1, 0, ‘bananas’);

console.log(produce);
// returns [“apples”, “bananas”, “carrots”]
```

Maydm

Array Methods: .splice

Splice can also replace an element. It takes the same three arguments: the **index** to change, the **number** of elements to overwrite, and the **value** to add.

```
var produce = [“apples”, “bananas”, “carrots”];

produce.splice(2, 1, ‘grapes’);

console.log(produce);
// returns [“apples”, “bananas”, “grapes”]
```

Maydm

Mozilla Docs on Splice

While learning JavaScript, reading the official docs can be really challenging but still worthwhile.

It's just like learning a new language -- exposure to the language and syntax will help you learn faster.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice#Examples

Mozilla Docs on Splice

```
array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

start

The index at which to start changing the array.

[redacted]

Maydm

Mozilla Docs on Splice

```
array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

`deleteCount` Optional

An integer indicating the number of elements in the array to remove from `start`.

If `deleteCount` is omitted, or if its value is equal to or larger than `array.length - start` (that is, if it is equal to or greater than the number of elements left in the array, starting at `start`), then all the elements from `start` to the end of the array will be deleted.

If `deleteCount` is 0 or negative, no elements are removed. In this case, you should specify at least one new element (see below).

Maydm

Mozilla Docs on Splice

```
array.splice(start[, deleteCount[, item1[, item2[, ...]]]])
```

`item1, item2, ...` Optional

The elements to add to the array, beginning from `start`. If you do not specify any elements, `splice()` will only remove elements from the array.

Maydm

Practice with Arrays

Work through the Day 7 “Arays” exercises on JS Bin.

Maydm

Reflection

Write in your journal about how you feel or what you learned today.

Prompts:

- Frustration is normal when learning new concepts. Did you struggle with any of the concepts today? How did you deal with that frustration?
- How do you use if/else statements in your daily life?
- Functions are the building block of coding. What can you do with them?

Maydm