



Maydm

Web Development

# Day 8: Even More JavaScript

Arrays & Loops

# Warm up!

---

# Today's Schedule

---

## Morning:

- JavaScript So Far...
- Arrays
- Loops
- Project: Todo List

## Afternoon:

- Field Trip!

# JavaScript So Far

---

- Comments
- Variables
- Data Types
- Operating on Variables
- Template Literals
- Booleans
- Functions
- If Statements

# Arrays

---

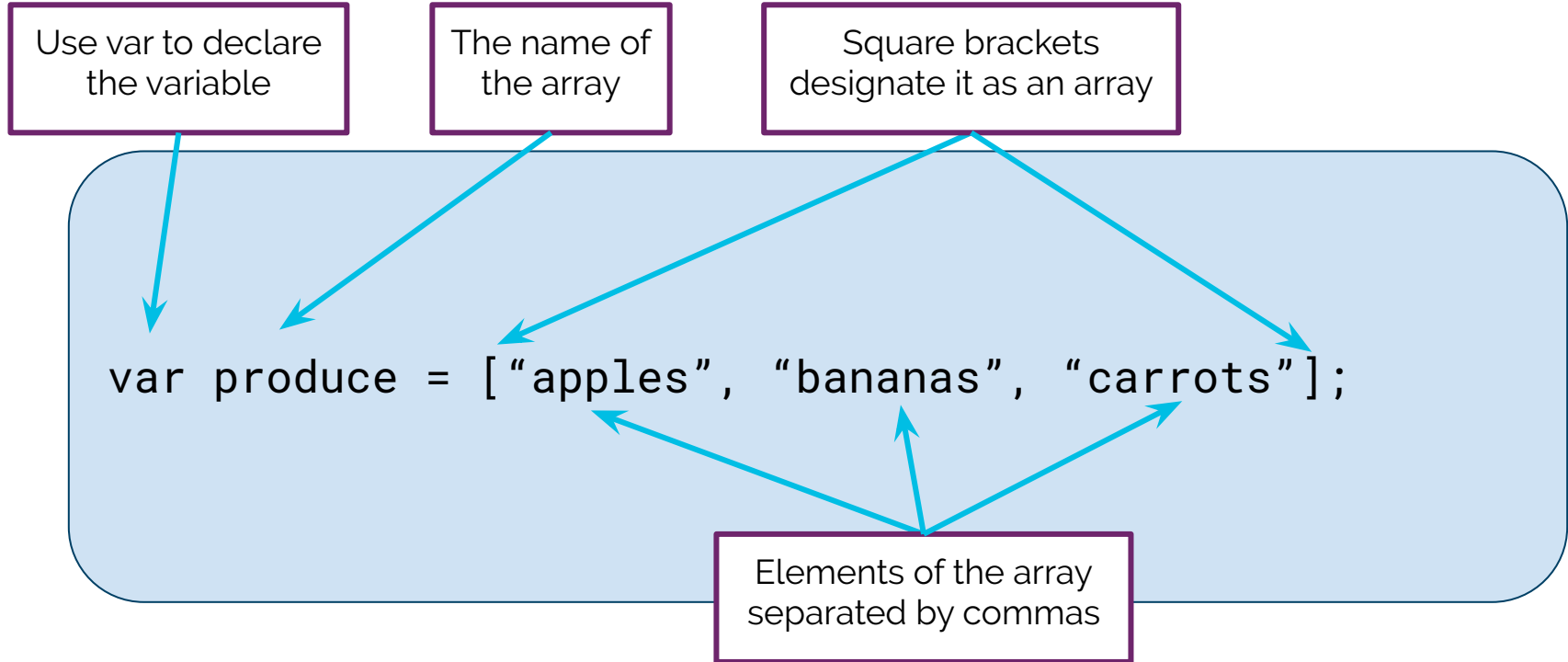
Sometimes you need a list of related values, such as inventory for sale or grades.

**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];
```

# Anatomy of an Array



# 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.



# Accessing Array Elements

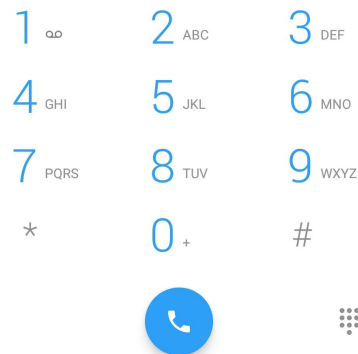
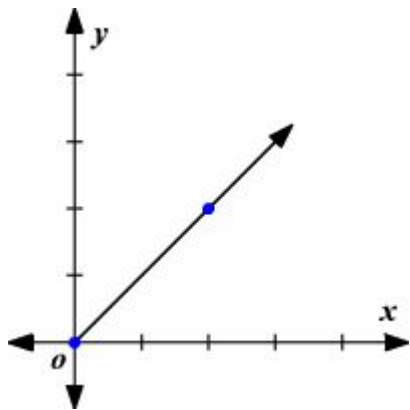
---

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



# 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'
```

# 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

The array that owns the method.

Notice, the

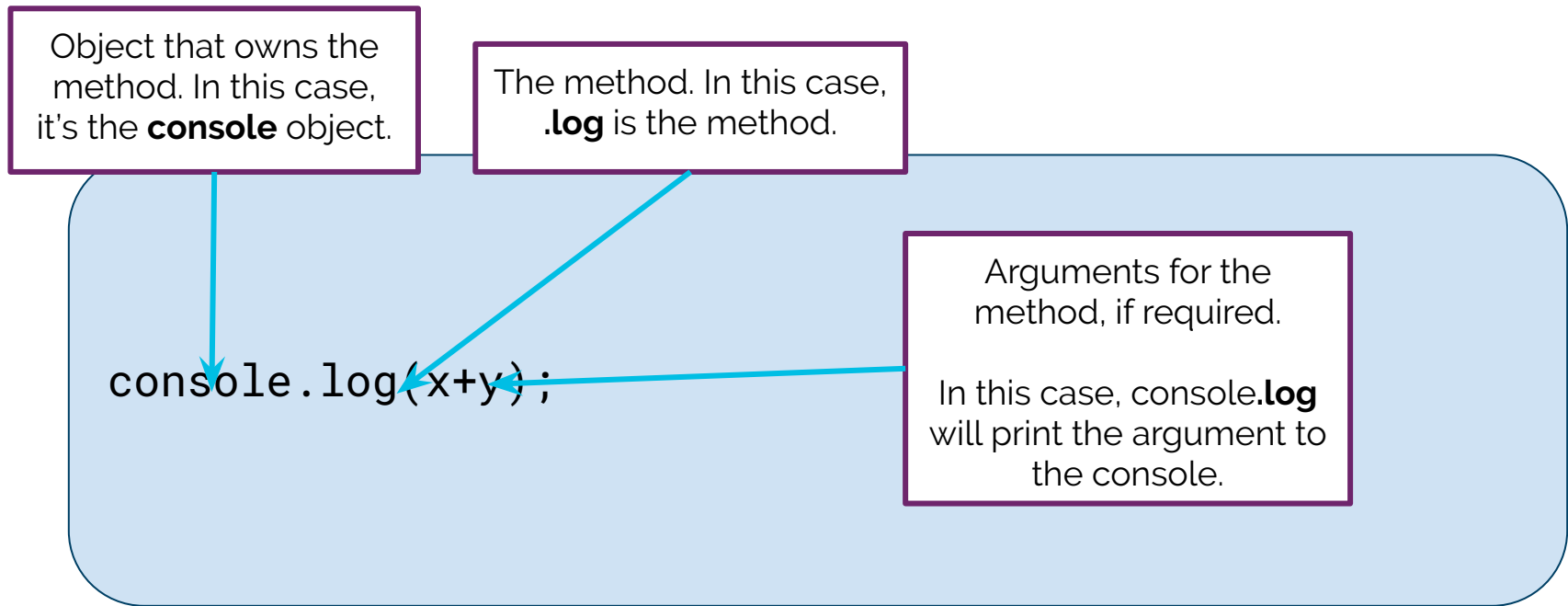
Arguments for the method, if required.

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

```
var produce = ["apples", "bananas", "carrots"];  
produce.length();
```

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

# Anatomy of an Array Method





# 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
```

# 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"]
```

# 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"]
```

# 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"]
```

# 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"]
```

# 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](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.

# 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).



# 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.

# Practice with Arrays

---

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

# Loops

---

<https://www.flocabulary.com/unit/coding-for-loops/>

# For Loops

---

When you want to run something for a specific number of times, use a **for** loop.

```
for (var i = 0; i < 10; i++) {  
    console.log("=^.^=");  
}
```

# Anatomy of a For Loop

Use for keyword to declare a for loop

The terms for the loop to run. In this case:

- variable **i** starts at 0
- loop runs while as **i is less than 10**
- **i increases** after each loop

```
for (var i = 0; i < 10; i++) {  
    console.log("=^.^=");  
}
```

Everything inside the opening & closing curly braces runs each time the loop runs

# Practice with For Loops

---

Work through the Day 8 “For Loops” exercises on JS Bin.

# For Loops

---

When might a for loop be useful?

# While Loops

---

Sometimes you don't know how many times you need a statement to execute. You need it to run while a certain condition is true. Use a while loop:

```
while (hungry === true) {  
    eatBurrito();  
    hungry = false;  
}
```



# Anatomy of a While Loop

Use while keyword to declare loop

The terms for the loop to run. In this case while the hungry variable is true.

```
while (hungry === true) {  
  eatBurrito();  
  hungry = false;  
}
```

Everything inside the curly braces runs each time the loop runs:

- eatBurrito() function
- hungry variable switches to false

# While Loops

---

Sometimes the condition for a while loop always uses a comparison operator, like greater than, less than, or equal to.

```
while (hungryPeople > 0) {  
    makeBurrito();  
    hungryPeople--;  
}
```

# Anatomy of a While Loop

Use while keyword  
to declare loop

The terms for the loop to run. In this case while  
the hungryPeople variable is greater than 0.

```
while (hungryPeople > 0) {  
    makeBurrito();  
    hungryPeople--;  
}
```

Everything inside the curly braces  
runs each time the loop runs:

- makeBurrito() function
- hungryPeople decreases by 1

# Practice with While Loops

---

Work through the Day 8 “While Loops” exercises on JS Bin.

# While Loops

---

When might a while loop be useful?

# Project: Todo List

---

Open the “Todo List” starter code and fork the pen.

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

# All Girls Gaming Hackathon

---

What would you like to create now that you know how to program?



# Reflection

---

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

Prompts:

- What can you store in arrays? Why would you use an array?
- How are loops useful? Can you think of a way you use loops in everyday life?
- What was the most exciting thing you learned on the field trip? Was there a STEM job that interested you?