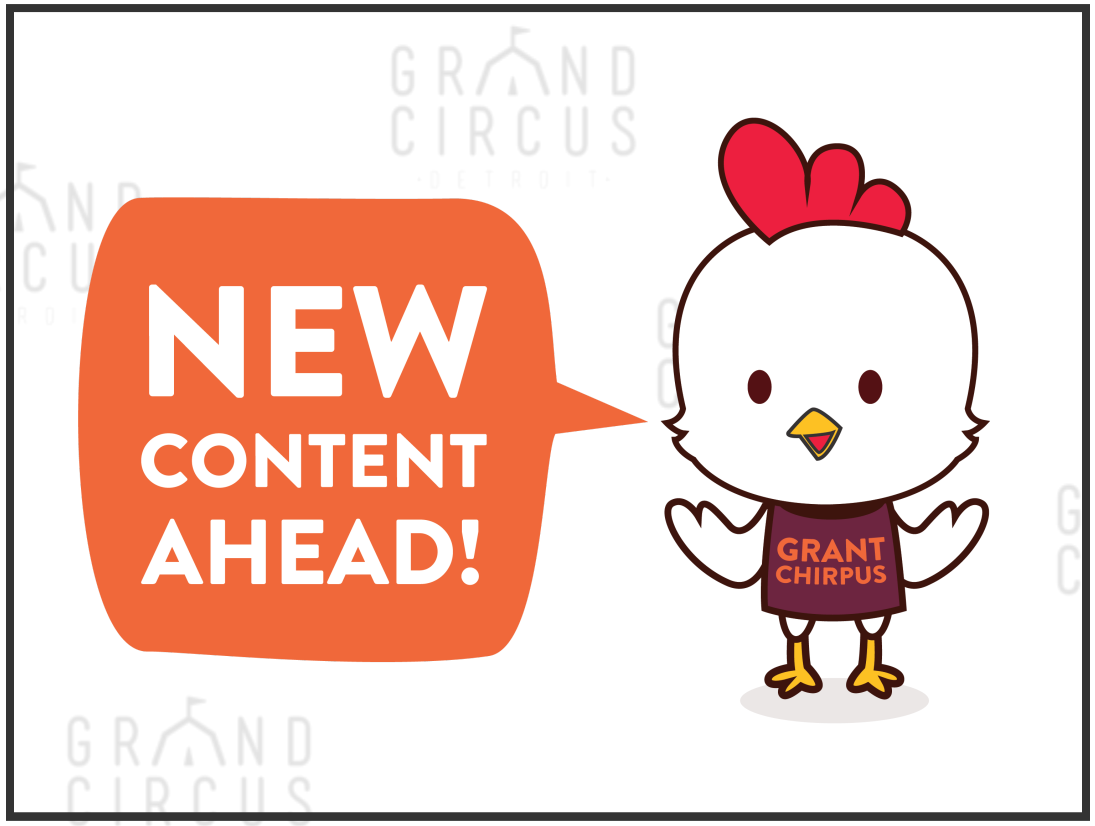

NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

ARRAYS, METHODS, & OBJECTS

GOALS FOR THIS UNIT

1. Review
2. Arrays
3. Built-in methods
4. Objects in JavaScript

REVIEW

DEMOS

DISCLAIMER

I realize that this is a lot of information. It's okay if you feel like this:

ARRAYS

ARRAYS

In JavaScript, an array is a data structure consisting of a collection of elements (values or variables), each identified by a numerical index or key.

ARRAYS

Arrays in JavaScript are special because the individual elements can be anything that can be stored in a variable.

ARRAY DEMO

```
// A perfectly legal array in JavaScript
var myArray = [
  'thing',
  12,
  ['another', 'array'],
  { my: 'object' },
  function() {}
];
```

ARRAYS

Granted, *most* of the time arrays are collections of like elements, but this is a feature that is used all over JS i.e. function argument lists.

ARRAYS

Fortunately, other array functionality works as expected (if you've worked in other languages before).

```
var coolColors = [
  'green',
  'blue',
  'purple'
];

coolColors[0];  // > 'green'

coolColors.length // 3 (number of items in array)
```

ARRAYS

Bracket notation can be used to access, reassign elements in an array, or add new elements.

```
var coolColors = [
  'green',
  'blue',
  'purple'
];

coolColors[1] = 'lavender';
coolColors[3] = 'indigo';
coolColors.length; // > 4
```

ZERO-BASED INDEXING

Notice anything unexpected in that last example?

Assigning a new color to index 3 increased the length of our 3-item `coolColors` array. That's because indexing begins with 0.

```
var coolColors = [
  'green',
  'blue',
  'purple'
];

coolColors[0]; // > 'green'
```

The `length` property returns the length of an object. E.g., if the object is an array, the number of elements is returned; if a string, the number of characters.

```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
var name = "Aisha";
rainbow.length // > 5
name.length // > 5
```

METHODS

WHAT'S A METHOD?

Methods are very similar to functions, but they have a more specific purpose. Methods allow objects (which we'll talk about in more depth very soon) do things or have things done to them.

USES

Often, methods will be used to do the following:

- Return some information about the object
- Change something about the object

BUILT-IN METHODS

There are a number of methods given to us for free by the JavaScript language. You don't need to understand exactly *how* they work for right now, but you should know how to make effective use of them.

We'll go over several examples:

The `toString()` method is used to convert many other types of data into a string that represents that data.

```
var ageAtHeart = 5;
var ageAsString = ageAtHeart.toString(); // > "5"
```

The `charAt()` method returns the character at a specified index of a string.

```
var faveThings = "raindrops on roses";
var zeroIn = faveThings.charAt(0); // > "r"
```

The `concat()` method is used to combine multiple strings into one. It can also be used to join arrays together.

```
var faveThings = "raindrops on roses";
var badThings = " when the dog bites";
var newString = faveThings.concat(badThings);
// > "raindrops on roses when the dog bites"
```

The `indexOf()` method returns the first index at which the specified value occurs.

```
var faveThings = "raindrops on roses";
var rainbow = ["red", "orange", "yellow", "green", "blue"];
faveThings.indexOf("r"); // > 0
rainbow.indexOf("orange"); // > 1
```

The `pop()` method removes the last element in an array and returns it.

```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
rainbow.pop(); // > "blue"
rainbow; // > "red", "orange", "yellow", "green"
```

The `push()` method adds one or more elements to the end of an array and returns the updated length of the array.

```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
rainbow.push("indigo", "violet"); // > 7
rainbow; // > "red", "orange", "yellow", "green", "blue", "indigo", "violet"
```

The `shift()` method removes the first element in an array and returns it.

```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
rainbow.shift(); // > "red"
rainbow; // > "orange", "yellow", "green", "blue"
```

The `unshift()` method adds one or more elements to the beginning of an array and returns the updated length of the array.

```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
rainbow.unshift("pink"); // > 6
rainbow; // > "pink", "red", "orange", "yellow", "green", "blue"
```

The `forEach()` method accepts a *function* as an argument. That functions accepts an element from the array as an argument. Then the body of that function is executed *for each* (see what they did there?) element in the array. `forEach()` is an even safer option for iterating over an array than a regular `for` loop.

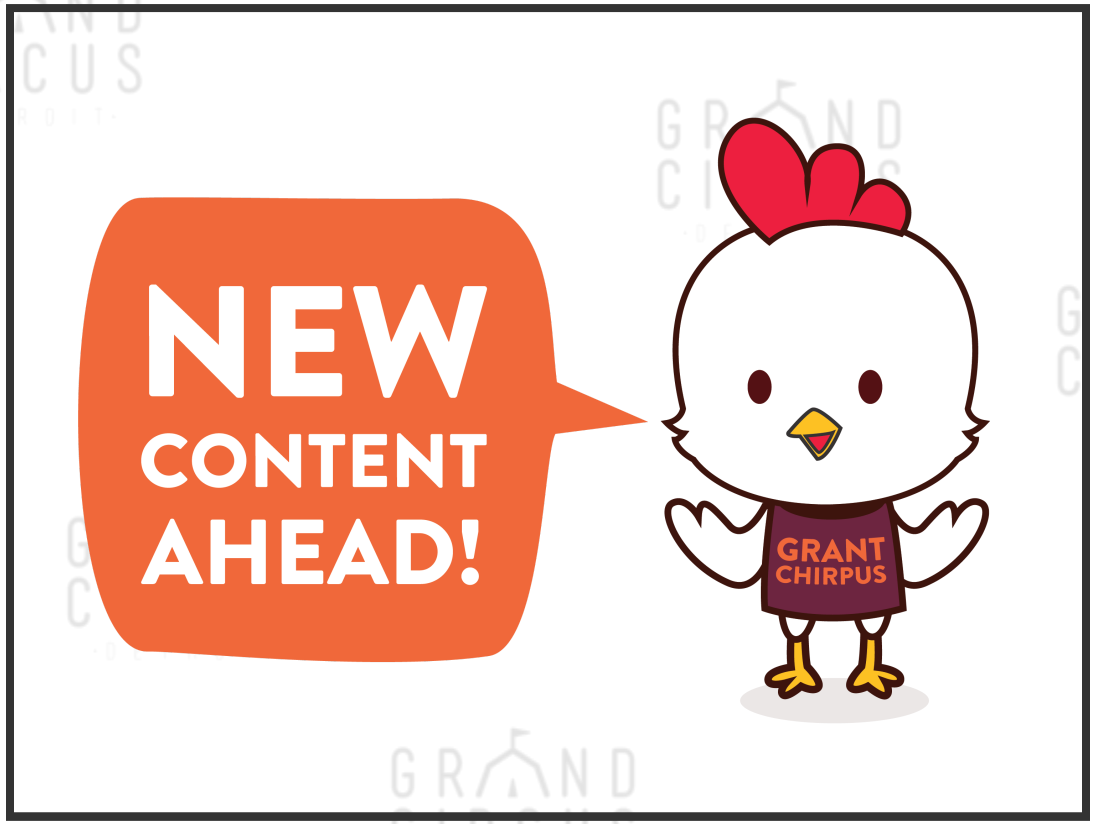
```
var rainbow = ["red", "orange", "yellow", "green", "blue"];
rainbow.forEach(function(element){
  console.log(element)
});

"red", "orange", "yellow", "green", "blue"
```

Function as an argument say whaaat?!

STOP! White board time!

FINDING METHODS

There is no point in trying to memorize all of this nonsense. You MUST learn to FIND the answers you seek! See this giant list of methods and try some of them out.


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

OBJECTS

Objects are a data structures that allow us to store collections of data including properties (variables) which can include arrays, other objects, or functions.

OBJECTS

```
var myInfo = {
  name: 'James',
  age: 36,
  married: true,
  likes: ['Mythbusters', 'Jim Butcher', 'JavaScript'],
  family: [
    {
      name: "Rebecca",
      relation: "spouse"
    },
    {
      name: "Evangeline",
      relation: "daughter"
    },
    {
      name: "Josephine",
      relation: "daughter"
    }
  ],
  listFamilyMembers: function() { }
};
```

OBJECTS

Object properties can be accessed using dot notation.

```
var name = myInfo.name;
var age = myInfo.age;
var maritalStatus = myInfo.married;
var familyMembers = myInfo.listFamilyMembers();
```

OBJECTS

You can also use dot notation to add new properties to objects.

```
myInfo.hobbies = ['video games', 'quadcopters', 'volunteering'];
myInfo.pets = 4;
```

THIS

"This" is a keyword that is helpful for use within objects and functions.

"This" always refers to a single object, but when and where it is used determines which object.

THIS

In the global context, "this" refers to the global object.

Inside a function, "this" varies depending on how it is called.

THIS

When using "this" within an object method (a function defined within an object), it refers to that particular object.

```
var javascriptStudent = {
  name: "Curly Sue",
  age: 27,
  sayName: function (){
    console.log("My name is " + this.name);
  },
};
javascriptStudent.sayName();
// > "My name is Curly Sue"
```

RECAP

You should understand and be able to use:

- Zero-based indexing
- Arrays
- Array Methods
- For Each loop
- Objects
- Functions as arguments
- "this"

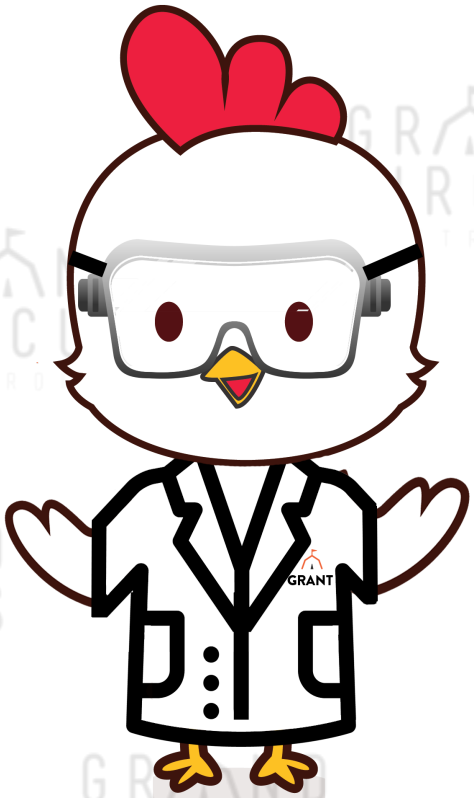
HOMEWORK

From JavaScript & jQuery:

- Chapter 3: 98-99
- Chapter 5: 183-242
- Chapter 6: 243-292

LAB 7

CONSOLE SHOPPING LIST



INSTRUCTIONS

Write a simple shopping list program. We'll build on this in the next lab.

1. Create several grocery item objects with properties of name and price.
2. Store the grocery item objects in an array.
3. Loop through the array and print out the name and price on a new line.
4. Total up the amount with a label ‘total’.

Be prepared to demo your work.

PRE-GAME

Solve the problem, then write the code

FIGURE IT OUT

Write a JavaScript function to generate an array. The elements in the array should be integers in a range between two integers given as arguments.

```
makeArray(-4, 2);
> [-4, -3, -2, -1, 0, 1, 2]
```