FEWD

Week 5 · Class 12

Arrays

Quick Review

- What does the `+` do when one or more of the values that are being operated on is not a number?
- What is one way to run some code conditionally?
- How do I tell Javascript that something is a string?
- Do variable names get quotes?

What We’ll Cover

- What are Arrays
- How to create arrays
- How to get the stuff that's inside of them
- How to add stuff to them
- How to loop through them

Working with Arrays

Arrays are like Lists

In programming, we use arrays to hold multiple pieces of related data. They are kind of like lists that we can assign to a variable.

What are they good for?

Arrays are great for storing related information in our programs.

```
var officeLocations = ["New York", "Boston", "San Fransisco"];
```

What data can Arrays hold?

Arrays can hold any combination of the *data types* we've been using so far, like strings, booleans and numbers, but they can also hold other arrays and objects.

```
var scores = [5502, 10300, 6578, 4329, 12023];
```

Creating an Array

```
var shoppingList = []; /* creates an empty array */

var fruits = ["🍏", "🍊", "🍋", "🍇", "🍓", "🍑"];

var years = [1980, 1969, 2000, 2001, 2011, 2018];
```

Getting at Stuff in an Array

Arrays are *indexed*. The stuff inside of an array is assigned to a specific position in the array. We can access the thing stored in an array with the number that represents its position starting with `0`!

```
var fruits = ["🍏", "🍊", "🍋", "🍇", "🍓", "🍑"];

console.log(fruits[0]); /* outputs: 🍏 */
console.log(fruits[4]); /* outputs: 🍓 */
```

Try Some Others...

```
var years = [1980, 1969, 2000, 2001, 2011, 2018];

var cities = ["Boston", "Paris", "London", "Frankfurt"];

var months = ["jan", "feb", "mar", "apr", "may", "jun"];
```

- `2000 == years[2]`
- `"Boston" == cities[0]`
- `months[1] == "feb"`

Setting Values in Arrays

We can set values in an array the same way as we access them to retrieve values.

```
var fruits = ["🍏", "🍊", "🍋"];

fruits[2] = "banana";

console.log(fruits); /* outputs:  ["🍏", "🍊", "banana"] */
```

Be Careful

Arrays can have "empty" indices.

```
var fruits = ["🍏", "🍊", "🍋"];

fruits[4] = "🍌";

console.log(fruits[3]); /* outputs:  undefined */
```

Array Length

If you want to know how many elements are in your array, you use the `length` property.

```
var fruits = ["🍏", "🍊", "🍋"];

console.log(fruits.length)  /* outputs:  3 */

fruits[4] = "🍌"; /* add banana in the 4th index */

console.log(fruits.length); /* outputs:  5 */
```

"Ummmm, that's nice, but what can you do with them?"

Looping over Arrays

Iterating over Arrays

A **for** loop allows us to loop over the array and do something with every value in it.

```
var shoppingList = ["Coffee", "Wine", "Chocolate", "Emergency Wine"];

for(var i = 0; i < shoppingList.length; i++) {

  console.log((i + 1) + ". " + shoppingList[i]);

}
  /* Results */
  /* 1. Coffee */
  /* 2. Wine */
  /* 3. Chocolate */
  /* 4. Emergency Wine */
```

for...of Loop

```
var shoppingList = ["Coffee", "Wine", "Chocolate", "Emergency Wine"];

for(var item of shoppingList) {

  $("ol").append("<li>"+item+"</li>");

}
```

HEADS UP: Way easier, but not supported in <= Internet Explorer 11 (only IE Edge). Also, you don't get the index by default.

CODE ALONG

Looping Over Arrays

Image URL Array

Array Methods

Working with Arrays

Arrays have some special built in methods that makes it easy*(ier)* to work with them.

Push and Unshift

Sometimes we just want to **add** to the end or beginning of an array...

```
var months = ["feb", "mar"];

months.unshift("jan"); /* unshift adds to the beginning */
months.push("apr", "may"); /* shift adds to the end */

console.log(months); /* ["jan", "feb", "mar", "apr", "may"] */
```

Pop

Other times we need to **remove** elements from the end of an array...

```
var months = ["jan", "feb", "mar", "apr", "may"];

months.pop(); /* removes the element from the end */
              /* it also returns that element */

console.log(months); /* ["jan", "feb", "mar", "apr"] */
console.log(months.pop()); /* outputs: apr */
console.log(months); /* ["jan", "feb", "mar"] */
```

Shift

We can also **remove** elements from the beginning of an array...

```
var months = ["jan", "feb", "mar", "apr", "may"];

months.shift(); /* removes the element from the start */
                /* it also returns that element */

console.log(months); /* ["feb", "mar", "apr", "may"] */
console.log(months.shift()); /* outputs: feb */
console.log(months); /* ["mar", "apr", "may"] */
```

Reversing the Order

The `reverse` method lets us swap the order of the array.

```
var months = ["jan", "feb", "mar"];

months.reverse();

console.log(months); /* ["mar", "feb", "jan"] */
```

Sorting the Elements

The `sort` method lets us sort the array elements in an ascending direction alphabetically.

```
var names = ["Jerry", "Jackie", "John", "Cathy"];

console.log(names.sort()); /* ["Cathy", "Jackie", "Jerry", "John"] */

var scores = [5502, 10300, 6578, 4329, 12023];

console.log(scores.sort()); /* [10300, 12023, 4329, 5502, 6578] */
```

Methods Worth Knowing

- `.includes()`
- `.join()`
- `.splice()`
- `.slice()`
- `.indexOf()`

- `.filter()`
- `.find()`
- `.every()`
- `.forEach()`
- `.map()`

Go Build Awesome Things!