WELCOME TO

ARRAYS DEEP DIVE

WHAT IS THIS?

- An experimental meetup to bring together people interested in learning to code or enhancing their coding abilities
- If anyone knows of a space that will host us for free/cheap, please let me know: markgromano31@gmail.com

ICE BREAKER 😸

- Who are ya and what brings ya here?

WHAT IS JAVASCRIPT?

- JavaScript - or JS - is a core language of the web
- It’s what allows for web pages to be interactive. Without it, our web would be very static

RECAP OF FUNDAMENTALS

- Variables
- Data types
- Conditionals
- Functions

VARIABLES

- Hold references to values
- Declare using the **var**, **let**, and **const** keywords
    - Let’s just use **let** for now

```
let foo = 1
let bar = 2
let fooBar = foo + bar
```

FUNCTIONS

- A chunk of code that can be executed”on demand
- Often **returns** a value
- Can “take in” values as **arguments**

```
function addTwo (number) {
  return number + 2
}

let value = addTwo(4)// 6

function addTwoAndWhatever(baseNumber, additionalNumber) {
  return baseNumber + 2 + additionalNumber
}

let anotherValue = addTwoAndWhatever(4, 3)// 9

let yetANOTHERvalue = addTwo(1) + addTwoAndWhatever(2, 3) //10
```

LETS DIVE INTO ARRAYS 🐳

- A container for multiple values
  - You reference values by their position, or index, within the container
  - Position numbers, or indices, start at 0

```
let arr = ['a', 'b', 1, 'foo']
arr[0] //'a'
arr[2] //1
arr[3] //'foo'

arr.length //4

arr[arr.length - 1] //foo
```

ARRAY METHODS

- Array **methods** are functions available to the array that can manipulate the values inside of it or give us information about the values inside

ARRAY.PUSH

- Adds an additional value(s) to the array

```
let fruits = [ 'apple', 'orange', 'banana' ]
fruits.push('pear')
fruits.length // 4
fruits// [ 'apple', 'orange', 'banana', 'pear' ]

fruits.push('tomato', 'cherries')
fruits// [ 'apple', 'orange', 'banana', 'pear', 'tomato', 'cherries' ]
```

ARRAY.POP

- The opposite of **push**
- Removes the last item in the array

```
let fruits = [ 'apple', 'orange', 'banana' ]
fruits.pop()
fruits.length // 2
fruits// [ 'apple', 'orange' ]
```

ARRAY.INCLUDES

- Returns a **boolean** (a **true** or **false** value) indicating if the specified value is contained within the array

```
let fruits = [ 'apple', 'orange', 'banana' ]
let hasApple = fruits.includes('apple') //true
let hasStrawberry = fruits.includes('strawberry') //false
```

ARRAY.INDEXOF

- Returns the index of the specified value
- If the value exists at multiple indices, it returns the index of the first
- If that value is not contained in the array, it returns -1

```
let fruits = [ 'apple', 'orange', 'banana' ]
let indexOfOrange = fruits.indexOf('orange') //1

fruits = [ 'apple', 'orange', 'banana', 'banana' ]
let indexOfBanana = fruits.indexOf('banana') //2
let indexOfStrawberry = fruits.indexOf('strawberry') //-1
```

ARRAY.LASTINDEXOF

- Like indexOf, but returns the *last* index of a specified value
- If that value is not contained in the array, it returns -1

```
let fruits = [ 'apple', 'orange', 'banana', ‘banana' ]
let indexOfBanana = fruits.indexOf('banana') //3
```

ARRAY.SLICE

- “Slices” a specified part of the given array into a new array
- The first argument is the index where you’d like to start slicing values
- The second argument is the index where you’d like to stop slicing values. The value at that index is not included
- If you’d like to copy the entire array, omit all arguments

```
                    0         1          2         3        4
let fruits = [ 'apple', 'orange', 'banana', 'pear', 'peach' ]
```

Start at 0, stop before 2

```
let theseFruits = fruits.slice(0, 2) //[ 'apple', ‘orange’]
```

Start at 2, stop before 5

```
let thoseFruits = fruits.slice(2, 5) //[ 'banana', 'pear', 'peach' ]
```

alternatively, omit 2nd arg to take rest of items

```
thoseFruits = fruits.slice(2) //[ 'banana', 'pear', 'peach' ]
```

Omit all args to copy entire array

```
let sameFruits = fruits.slice()
//[ 'apple', 'orange', 'banana', 'pear', 'peach' ]
```

CHALLENGES TIME!

- We are going to use the technique of **pair programming**
  - A **driver** role and a **navigator** role
  - The **driver** writes the code
  - The **navigator** guides the driver (*doesn’t mean dictating code*)
- Why?
  - *Two minds are better than one when facing an unknown*

GUIDELINES FOR CHALLENGES

- Google smartly!
- Keep your code tidy
- Don’t get super stuck on a problem; move on to another
- Seek other groups for help! It’s not a contest

Challenges

https://repl.it/@jsct/arrays

Slideshow

https://jsctarrays.herokuapp.com/

Email

markgromano31@gmail.com