BUILT-IN DATA STRUCTURES

Through the lens of Big O

Now that we've covered BIG O...

Let's spend a couple minutes analyzing the things we do all the time in JS: working with Arrays, Objects, and built-in methods.

We spend a lot of this course talking about data structures.

Let's start by discussing the ones we get for free

OBJECTIVES

- Understand how objects and arrays work, through the lens of Big O
- Explain why adding elements to the beginning of an array is costly
- Compare and contrast the runtime for arrays and objects, as well as built-in methods

OBJECTS

Unordered, key value pairs!

```
let instructor = {
    firstName: "Kelly",
    isInstructor: true,
    favoriteNumbers: [1,2,3,4]
}
```

When to use objects

- When you don't need order
- When you need fast access / insertion and removal

Big O of Objects

Insertion - **O(1)**

Removal - **O(1)**

Searching - **O(N)**

Access - **O(1)**

When you don't need any ordering, objects are an excellent choice!

Big O of Object Methods

Object.keys - **O(N)**

Object.values - **O(N)**

Object.entries - **O(N)**

hasOwnProperty - **O(1)**

ARRAYS

Ordered lists!

```
let names = ["Michael", "Melissa", "Andrea"];

let values = [true, {}, [], 2, "awesome"];
```

WHEN TO USE ARRAYS

- When you need order
- When you need fast access / insertion and removal (sort of....)

Big O of Arrays

Insertion - **It depends....**

Removal - **It depends....**

Searching - **O(N)**

Access - **O(1)**

Let's see what we mean by that!

Big O of Array Operations

- push - **O(1)**
- pop - **O(1)**
- shift - **O(N)**
- unshift - **O(N)**
- concat - **O(N)**
- slice - **O(N)**
- splice - **O(N)**
- sort - **O(N * log N)**
- forEach/map/filter/reduce/etc. - **O(N)**

You don't need to know all this...

Limitations of Arrays

Inserting at the beginning is
not as easy as we might think!
There are **more efficient** data
structures for that!