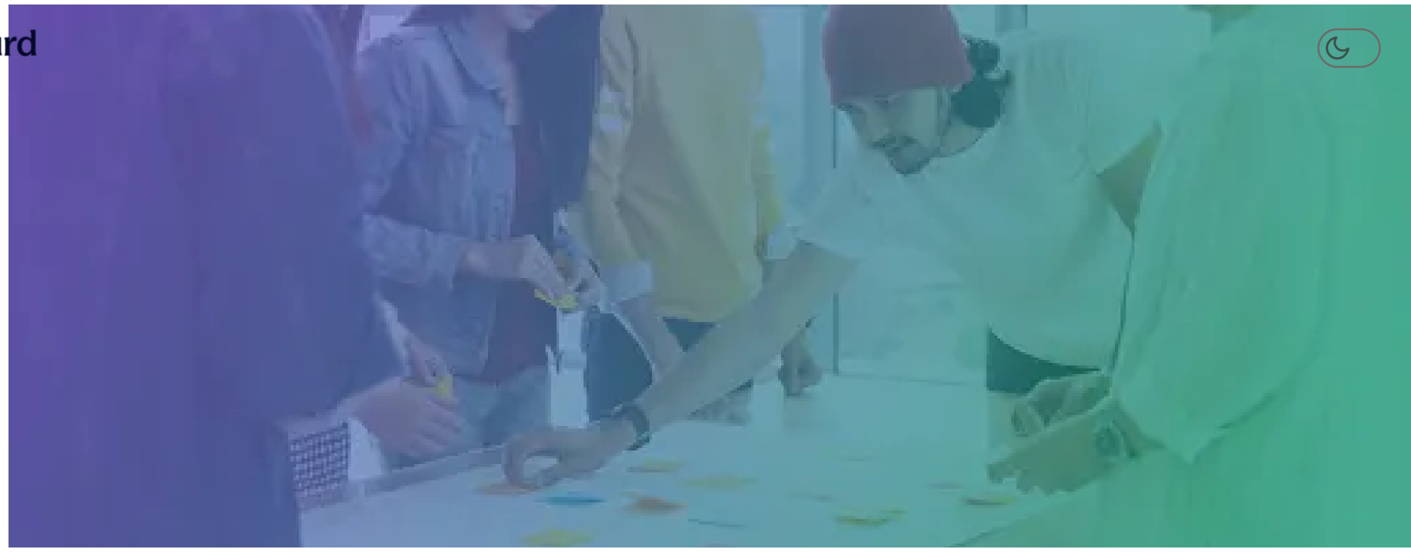
Springboard



Objects

Goals

- Describe what objects are
- Compare objects and arrays
- Access and update key values pairs in an object

Objects

What Is An Object?

An object is an **unordered** collection of key-value pairs.

If you know the key, you can get the value!

Arrays Vs. Objects

- Arrays are ordered by index
- You access array values by their index
- Arrays are helpful when you have data that is ordered
- Objects are unordered
- You access object values by their key
- Objects are helpful when you have unordered data

Why Should I Care?

Objects are everywhere in JavaScript

Most languages have analogues of what an object is in JavaScript

A useful alternative to arrays - let's see why!

Limitations Of Arrays

```
const movie = [ "Titanic", 1997, "PG-13", 659363944, "James
Cameron" ];
```

- Rating? ***movie[2]***
- Director? ***movie[4]***
- Year? ***movie[1]***

It would be much easier to do

```
movie.rating movie.director movie.year
```

Creating Objects

- Objects use curly braces (object literal syntax)
- Keys and values are separated by a colon
- Key-value pairs are separated by a comma

```
const movie = { title: "Titanic", year: 1997, rating: "PG-13",
revenue: 659363944, director: "James Cameron" };
```

Keys In Objects

These are equivalent:

```
const movie = { title: "Titanic", year: 1997, rating: "PG-13",
revenue: 659363944, director: "James Cameron" }; const movie =
{ "title": "Titanic", "year": 1997, "rating": "PG-13",
"revenue": 659363944, "director": "James Cameron" };
```

Object keys in JavaScript are (almost) strings, symbols can also be keys … but we won't talk about Symbols yet

Values In Objects

Values can be variables!

```
const movieTitle = "Titanic"; const releaseYear = 1997; const
dudeWhoDirectedIt = "James Cameron"; const movie = { title:
movieTitle, year: releaseYear, director: dudeWhoDirectedIt };
```

Accessing Values In An Object

To access a value in an object, you need to know the value's key

Given the key, you can obtain the value either with dot notation or bracket notation

```
const language = { name: "JavaScript", hasObjects: true,
yearReleased: 1995, isSuperFun: true }; language.name; //
"JavaScript" language.hasObjects; // true language["name"]; //
"JavaScript"
```

Dot Vs. Bracket - What???

When using bracket notation, the key is evaluated as an expression

When using dot notation, the key is NOT evaluated as an expression

If you don't know with 100% certainty what the name of the key is that you are looking for, use bracket notation.

Otherwise always use dot notation.

Updating Values In An Object

Like with arrays, values in an object can be updated with a simple assignment.

```
const obj = { key: "old value" }; obj.key = "new value"
obj["key"] = "newer value"
```

Removing Keys From An Object

To remove a key-value pair from an object, you can use the delete keyword.

```
const obj = { key: "old value" }; delete obj.key; // true
```

You can use this to remove values from an array as well, though this is less common.

Storing Values From An Object

Storing values by accessing them

```
const language = { name: "JavaScript", hasObjects: true,
yearReleased: 1995, isSuperFun: true }; const name =
language.name; // dot notation works const hasObjects =
language["hasObjects"]; // brackets work too! const
yearReleased = language.yearReleased;
```

hasOwnProperty

- Called on an object
- Accepts the name of a key
- Returns true if the key exists in the object, otherwise false

```
const cat = { name: "Blue", hairColor: "gray", eyeColor:
"orange" } cat.hasOwnProperty("name"); // true
cat.hasOwnProperty("favoriteFood"); // false
```