Destructuring Assignment

- Due No Due Date
- Points 1
- Submitting a website url

[(https://github.com/learn-co-curriculum/phase-1-destructuring-assignment)](https://github.com/learn-co-curriculum/phase-1-destructuring-assignment) [(https://github.com/learn-co-curriculum/phase-1-destructuring-assignment/issues/new)](https://github.com/learn-co-curriculum/phase-1-destructuring-assignment/issues/new)

Learning Goals

- Use *destructuring assignment* to assign data to variables

Introduction

As developers, sometimes we receive information in a collection (e.g., an `Object`) and we want to "pick and choose" elements out of the collection. It's a major pain to individually extract each property / value pair out of an `Object` and then assign it to a variable.

Destructuring lets us type less and be more clear about what we want to pull out of an `Object` . Not only does destructuring help when working with data in your application, it's essential for understanding how to get JavaScript to include third-party code (like you find on **npm** **(https://www.npmjs.com/)**).

Use Destructuring Assignment to Assign Data to Variables

In JavaScript, when we want to assign data from an object to single variables, we know how do it individually like so:

```
const doggie = {
  name: 'Buzz',
  breed: 'Great Pyrenees',
  furColor: 'black and white',
  activityLevel: 'sloth-like',
  favoriteFood: 'hot dogs'
};
const name = doggie.name;
const breed = doggie.breed;
name; // => "Buzz"
breed; // => "Great Pyrenees"
```

This is repetitive code. The process is:

1. Declare a variable with a name (e.g. `name` or `breed`)
2. Use that variable's name to point to an attribute in the `Object` whose name matches the name of the variable (e.g. `doggie.breed` or `doggie.name`)
3. Assign the attribute's value to the created variable

JavaScript gives us the ability to perform this task with *one* simple line of code.

```
const doggie = {
  name: 'Buzz',
  breed: 'Great Pyrenees',
  furColor: 'black and white',
  activityLevel: 'sloth-like',
  favoriteFood: 'hot dogs'
};

const { name, breed } = doggie;
name; // => "Buzz"
breed; // => "Great Pyrenees"
```

The `{}` around the variable names tells the JavaScript engine that it's going to be pulling values from an `Object` . The engine looks inside the `doggie` object for the attributes `name` and `breed` and assigns the values associated with those keys to the corresponding variable names. This is known as *destructuring assignment*.

Note that because the engine is looking for the attributes by their keys, the order inside the `{}` doesn't matter — this works as well:

```
const { breed, name } = doggie;
name; // => "Buzz"
breed; // => "Great Pyrenees"
```

We can also use destructuring assignment with a nested data structure:

```
const doggie = {
  name: 'Buzz',
  breed: 'Great Pyrenees',
  furColor: 'black and white',
  activityLevel: 'sloth-like',
  favoriteFoods: {
    meats:{
      ham: 'smoked',
      hotDog: 'Oscar Meyer',
    },
    cheeses:{
      american: 'kraft'
    }
  }
};

const { ham, hotDog } = doggie.favoriteFoods.meats;
ham; // => "smoked"
hotDog; // => "Oscar Meyer"
```

We've simply "drilled down" to the object we want to access by chaining the keys: `doggie.favoriteFoods.meats` .

Using Destructuring Assignment with Arrays

Destructuring does not just work on objects — we can use the same syntax with `Array` s.

```
const dogs = ['Great Pyrenees', 'Pug', 'Bull Mastiff'];
const [medium, small, giant] = dogs;
console.log(medium, small, giant); // LOG: Great Pyrenees Pug Bull Mastiff
```

Note that, this time, we've wrapped the variables we're declaring in `[]` instead, so the engine knows we're destructuring an `Array` . In this case, the order *does* matter: the engine assigns the first element to `medium` , the second to `small` and the third to `giant` .

The cool part is we can pick the parts of the `Array` that we want to assign!

```
const dogs = ['Great Pyrenees', 'Pug', 'Bull Mastiff'];
const [, small, giant] = dogs;
console.log(small, giant); // LOG: Pug Bull Mastiff
```

The initial comma tells the engine to skip the first element and start the assignments with the second element.

Using Destructuring Assignment with Strings

We can also destructure with strings by using the `String.prototype.split()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split) method to turn the string into an array:

```
const dogsName = 'Sir Woody BarksALot';
const [title, firstName, lastName] = dogsName.split(' ');
console.log(title, firstName, lastName); // LOG: Sir Woody BarksALot
```

Because the `split()` method returns an array, we can pick and choose just as we did before:

```
const dogsName = 'Sir Woody BarksALot';
const [title, ,lastName] = dogsName.split(' ');
```

```
console.log(title, lastName); // LOG: Sir BarksALot
```

Instructions

Take a look in `index.js` . You'll see that we've given you several variables containing `String` s, `Array` s, and `Object` s. For this lab, you need to write several destructuring assignments for each. Specific instructions are provided at the bottom of the `index.js` file. Let the test output guide you through the process.

Conclusion

Destructuring assignment is a fast, and efficient way to assign data to variables from objects, arrays, and strings. It allows us to easily pick and choose the pieces of data that we want to assign. With practice, you'll be proficient at it in no time.

Resources

- **Destructuring assignment (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment)**