Array and Object Destructuring

Goals

- Understand what destructuring is
- Use object destructuring to write less code
- Use array destructuring to swap values and extract nested values

Object Destructuring

JavaScript programmers take things out of objects all the time.

Here’s how you used to have to extract values into variables.

```
let userData = {
  username: 'smith',
  id: 12345,
  password: 'fiddlesticks',
  firstName: 'Angela',
  lastName: 'Smith',
  age: 'guess',
  isLegit: undefined
};

let username = userData.username;
let firstName = userData.firstName;
let lastName = userData.lastName;
let id = userData.id;
```

That’s A Lot of Typing

So they came up with some syntactic sugar.

```
 let userData = {
   username: 'smith',
   id: 12345,
   password: 'fiddlesticks',
   firstName: 'Angela',
   lastName: 'Smith',
   age: 'guess',
   isLegit: undefined
 };

/*
  declare variables: username, firstName, lastName, id
   values taken from the keys of the same name in userData
 */
 let { username, firstName, lastName, id } = userData;

 console.log(username);  // smith
 console.log(id);        // 12345
```

Destructuring + Spread

```
const userData = {
  username: 'smith',
  id: 12345,
  password: 'fiddlesticks',
  firstName: 'Angela',
  lastName: 'Smith',
  age: 'guess',
  isLegit: undefined
};

// extract the password key; collect the rest in 'user'
const { password, ...user } = userData;

console.log(user);
/*
{
  username: 'smith',
  id: 12345,
  firstName: 'Angela',
  lastName: 'Smith',
  age: 'guess',
  isLegit: undefined
}
*/
```

Renaming with destructuring

```
const instructorData = {
  name: "Colt",
  job: "Instructor"
}

const { name: instructorName, job: occupation } = instructorData;

instructorName // "Colt"
occupation // "Instructor"
```

Defaults with destructuring

```
const options = {
  refreshTime: 200
}
const { refreshTime = 750, waitTime = 1000 } = options;
```

```
console.log(refreshTime); // 200 - initialized in options
console.log(waitTime); // 1000 - fallback to default
```

Destructuring nested objects

```
const instructor = {
  id: 44,
  name: 'Colt',
  isHilarious: true,
  funFacts: {
    favoriteFood: 'Burrito',
    favoriteDrink: 'Old Fashioned',
  }
};
const {funFacts: {favoriteFood, favoriteDrink}} = instructor;
console.log(favoriteFood); // 'Burrito'
```

Destructuring functions

We can use destructuring to extract key/value pairs from an object into variables.

```
function makeInstructor(settings) {
  let name = settings.name;
  let age = settings.age;
}
```

We're going to assume the function is passed an object with a key of name and age

```
function myFunc({name, age}) {
  let name = name;
  let age = age;
}
```

But what happens if the object does not contain a key of name or age?

We can use default parameters!

```
function myFunc({name = "Xie", age=38}) {
  let name = name;
  let age = age;
}
```

You Can Apply The Same Concept To Arrays!

```
const myFavoriteThings = ['teaching', 'music',
                          'hiking', 'dank memes'];

const [first, second, ...others] = myFavoriteThings;
```

```
console.log(first);   // 'teaching'
console.log(second);  // 'music'
console.log(others);  // ['hiking', 'dank memes']
```

Fancy 1-Line Array Value Swap

```
let a = 1;
let b = 3;

[a, b] = [b, a];

console.log(a); // 3
console.log(b); // 1
```