Avoiding Common Mistakes

Promise composition allows for much more readable code than the nested callback syntax that preceded it. However, it can still be easy to make mistakes. In this exercise, we’ll go over two common mistakes with promise composition.

**Mistake 1:** Nesting promises instead of chaining them.

returnsFirstPromise()  
.then((firstResolveVal) => {  
  return returnsSecondValue(firstResolveVal)  
    .then((secondResolveVal) => {  
      console.log(secondResolveVal);  
    })  
})

Let’s break down what’s happening in the above code:

* We invoke returnsFirstPromise() which returns a promise.
* We invoke .then() with a success handler.
* Inside the success handler, we invoke returnsSecondValue() with firstResolveVal which will return a new promise.
* We invoke a second .then() to handle the logic for the second promise settling all **inside** the first then()!
* Inside that second .then(), we have a success handler which will log the second promise’s resolved value to the console.

Instead of having a clean chain of promises, we’ve nested the logic for one inside the logic of the other. Imagine if we were handling five or ten promises!

**Mistake 2:** Forgetting to return a promise.

returnsFirstPromise()  
.then((firstResolveVal) => {  
  returnsSecondValue(firstResolveVal)  
})  
.then((someVal) => {  
  console.log(someVal);  
})

Let’s break down what’s happening in the example:

* We invoke returnsFirstPromise() which returns a promise.
* We invoke .then() with a success handler.
* Inside the success handler, we create our second promise, but we forget to return it!
* We invoke a second .then(). It’s supposed to handle the logic for the second promise, but since we didn’t return, this .then() is invoked on a promise with the same settled value as the original promise!

Since forgetting to return our promise won’t throw an error, this can be a really tricky thing to debug!