Variables Lab

In this lab we're going to work with variable scopes and hoisting. We'll continue working in your pairs with our development of the roman numeral converter.

Hoisting with var

1. Add this function to your romanNumberConverter.js source code:

```
export function showHoisting() {
  var x = 1;
  if (x == 2) {
    var y = true;
  }
  return y;
}
```

2. Edit your romanNumeralConverter.spec.js test file. Import showHoisting like so:

```
import { showHoisting } from '../src/romanNumeralConverter";
```

3. Add a unit test to your suite. Make the test pass if showHoisting() returns undefined.
4. Run your tests. Did the test pass? It should have. But wait! Wouldn't most languages have a compile time failure on the return statement? I mean, y is local to the block, right?
5. Discuss with your partner why it passed. What happed so that it became undefined?

Working with let and const

6. Now change the vars to lets.
7. Rerun your test. Did it fail again? It should have but for a different reason. Again, talk to your partner about why.
8. Go ahead and make the test pass. Any way you like is fine.
9. Lastly, change the vars/lets to consts. Depending on how you chose to fix your test in the last step, it may or may not still pass. Again talk with your partner about what might have changed.
10. Try to reassign the y const. (Should not work).
11. Get your tests passing any way you like.
12. Create a new const:

```
const z = {foo: true, bar: true, baz: true};
```

13. Your tests should still pass. Make sure.
14. Now on the next lines of code go:

```
z.foo = false;
z.qux = true;
```

15. Now z is a const, so this should fail, right? Wrong! Run your tests and they'll pass. Discuss with your teammate why a const is allowed to change.

Bonus! Go through all your code and tests and change all of your vars to consts where you can and lets everywhere else.