

Object-oriented JavaScript Lab

At this point, you probably have a file called `romanNumeral.js` with an exported function called 'convert'. Let's convert that to a class called `RomanNumeralConvertor` and see if you like it better.

1. First, change your tests. Wherever you're calling `convert`, change the "convert" to "convertor.convert". Then, instantiate what will become your class like so:

```
const convertor = new RomanNumeralConvertor();
```
2. Find all the places that you're importing `convert`. You'll want to change those to import `RomanNumeralConvertor` instead.

Your tests will fail because `RomanNumeralConvertor` isn't defined. Let's fix that.

3. Open `romanNumeral.js` in your IDE. Wrap your function in a class. You'll have to change the syntax to make it work. You may have to remove the function keyword among other things.
4. Don't forget to change your export. You're no longer exporting the function. You now want to export the entire class.
5. At this point, your tests should pass. Go ahead and run them. If they don't pass, work with them until they do.

Using class accessors

Let's say that we were told our `convert` function is difficult to use and that it would be easier if someone could do it like this:

```
convertor.romanNumeral = "xiv";  
const decimalNumber = convertor.decimalNumber;  
console.log(decimalNumber); // Should say "14"
```

We can make that happen!

6. Write a new test. Something like this will do:

```
it("should convert roman to decimal using a setter and getter", () => {  
  const convertor = new RomanNumeralConvertor();  
  convertor.romanNumeral = "iii";  
  expect(convertor.decimalNumber).toEqual(3);  
});
```
7. That test will fail if you try it.
8. Bonus! Write 3-4 tests right now for roman numerals that you think will work in your convertor.

Writing the accessors

9. First create two properties called `_decimalNumber` and `_romanNumeral`.
10. Next create a getter for `decimalNumber` kind of like this:

```
get decimalNumber() {  
  return this._decimalNumber;  
}
```
11. Then do the same for a getter for `romanNumeral`.
12. Create a setter for `romanNumeral` something like this:

```
set romanNumeral(value) {  
  this._romanNumeral = value;  
  this._decimalNumber = this.convert(value);  
}
```
13. At this point, you've created enough to get your tests to pass. Work with them until they do.