- 
- 
- inside this file, we see 3 different functions that perform some sort of computation to a number.
- if we wanted to run these functions together we could chain them together like so.
- in this instance, we want to run them in order of their line number to compute a number from the startVal constant atop the page.
- 
- when chaining functions, the innermost function will be executed first and then continues towards the outermost function which in our case is addTen.
- Now if i execute this in my terminal, we see that the function is working correctly as it's printing out42.
- alright so let's add another function that will just divide our number in half.
- now, if I wrap these nested functions in this halfNum function, then the prompt should print 21…. and it does.
- so this is great, but there has has to be a more elegant way of performing this, right?
- Queue the compose function.
- the compose function takes in an array of fns as well as a starting point, from there, from there, it uses JavaScripts reduceRight method to map through the array of functions, and apply them to the result of the previous function being returned..
- additionally, we must pass in initial value into reduceRight to specify our start value.
- alright awesome!
- Now that compose is wired up and ready
- Since compose is a curried function, we're passing our vals in in seperate parenthesis.
- as we just learned, the left parens group is going to hold our array of functions, while the 2nd one will hold our initialValue, which in this case is the startVal constant.
- As it's name suggests, reduceRight is going to apply our functions from RTL therefore we're going to pass our functions in like so.
- first we have halfNum, then addTen, then doubles and finally squared, which is going to be the first function that will act on our startValue.