Jasmine Testing Exercises

Download starter code <../jasmine-testing-exercises.zip>

Loan Calculator

You are given the HTML, CSS, and JavaScript for a loan calculator. Your goal is to fill in the JavaScript to make the loan calculator functional. The calculator takes an amount to loan (the *principle*), a term in years, and a yearly rate. The output should be the monthly payment of the loan.

To calculate the monthly payment, use the following formula:

$$\text{monthly payment} = \frac{P \times i}{1 - (1 + i)^{-n}}$$

Where:

- ***P*** = Amount of principle
- ***i*** = periodic interest rate (in our case yearly rate ÷ 12)
- ***n*** = total number of payments (years × 12)

There is also a jasmine test file (***calculator-test.js***). Your goal is to write tests for the ***calculateMontlyPayment*** function. There are a few suggested tests, but if you can think of more, feel free to add to it.

Further Study: Tip Pool

In this exercise you will take an existing code base and write tests using the Jasmine testing library.

'Tip Pool' is an application that tracks the total tips generated from a group of servers in a restaurant. It then calculates the payout for each server.

Your task is to thoroughly test each function in the app!

Step One: Complete server.test.js

Take a minute to read through the codebase, currently there is only one test in `server.test.js`

Notice that the string 'Alice' is getting left on the dom after the first test is run, this is because we didn't tear down our tests properly

To get started:

- Thoroughly read the code base
- Clean up the dom after the test is run using `afterEach`
- Write a test for each function found in `server.js`

Step Two: Test the remainder of the app

Repeat the process from step one for `payments.js` and `helpers.js`

- Create a `helpers.test.js` file and test each function in `helpers.js`
- Create a `payments.test.js` file and test each function in `payments.js`
- When finished you should have three .test.js files one for each .js file

Step Three: Test your own code

Get a feel for testing your own code

First we will build out functionality for removing a server from the server table

- Review the functionality of `appendTd(tr, value)`
- Create a `appendDeleteBtn(tr)`, it will be similar to `append(tr, value)`. This function will create a 'td' with the value 'X', when clicked it will delete the table row it belongs to
- Write the functionality for appending a 'td' to a 'tr' with the value 'X'
- Set an click event listener on the 'td' that will remove the parent 'tr' from the dom. You will have to find a way to access the parent row of the 'td' from the click event
- Write tests for `appendDeleteBtn(tr)`

  You may notice the difficulty of simulating a click with vanilla javascript so do not spend too much time on testing the html after the DOM is updated (later we will study approaches for this with other libraries).
- Repeat the process for removing a payment from the payment table

Solution

View our solutions <solution/index.html>