Jasmine introduction



Unit testing involves the execution of a JavaScript function to evaluate if the output returned is the expected one. It helps us test if one or more features of a JavaScript function are working as per expectation. In general, these features indicate the extent to which the JavaScript application being tested:

* meets the requirements that guided its design and development,
* responds correctly to all kinds of inputs,
* achieves the general result its stakeholders desire
* JavaScript is a powerful programming language that has no compilerand no static type-checking in the code. As JavaScript developers we continuously write new code or extend the existing code. In either of the scenarios, if code fails, as developers we can see it only on the browser during execution and not before that.
* To debug or write endless console statements in our code are manual ways of testing the JavaScript code. This process is too slow.
* We should have automated ways of finding the bug in the code and avoid the chaotic and time consuming debugging. Automated testing will enable us to programmatically check or test the functionality of our code before we execute it and watch it fail.
* In this course we will use **Jasmine** unit testing framework.

Steps for installing Jasmine and karma

<https://jasmine.github.io/pages/getting_started.html>

## JASMINE FOR NODE.JS

Add Jasmine to your package.json

npm install --save-dev jasmine

Initialize Jasmine in your project

npx jasmine init

Set jasmine as your test script in your package.json

"scripts": { "test": "jasmine" }

Run your tests

npm test

## JASMINE FOR BROWSERS

Add Jasmine to your package.json

npm install --save-dev jasmine-browser-runner jasmine-core

Initialize Jasmine in your project

npx jasmine-browser-runner init

Set jasmine as your test script in your package.json

"scripts": { "test": "jasmine-browser-runner runSpecs" }

Run your tests

npm test

open command Prompt (administrator)

npm install -g karma jasmine-core

npm install jasmine-node

npm view jasmin

npm view karma

karma init

Which testing framework do you want to use ?

Press tab to list possible options. Enter to move to the next question.

> jasmine

Do you want to use Require.js ?

This will add Require.js plugin.

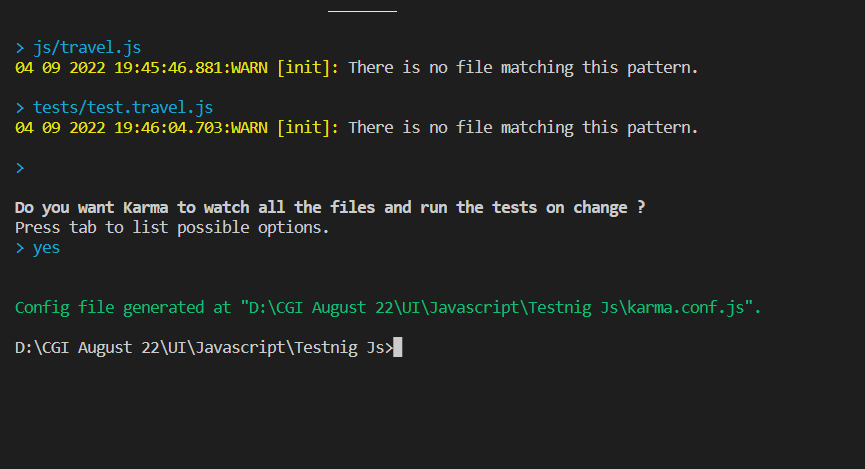
Press tab to list possible options. Enter to move to the next question.

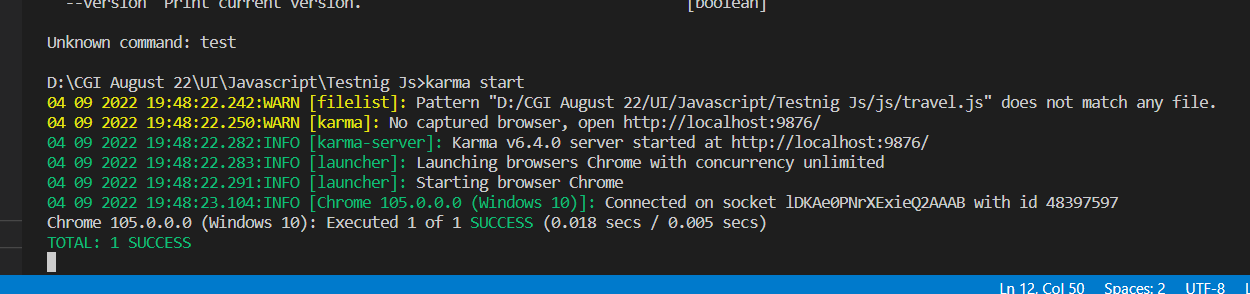
> no

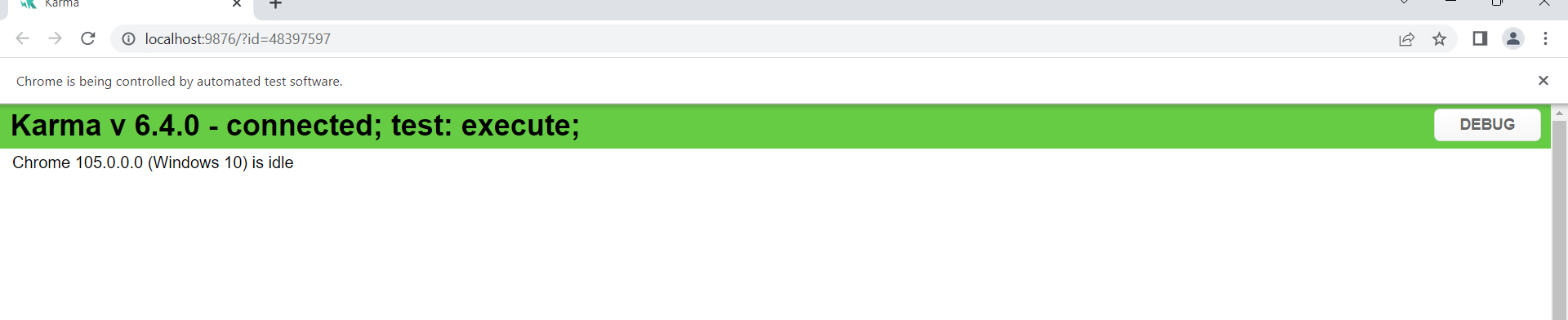
Do you want to capture any browsers automatically ?

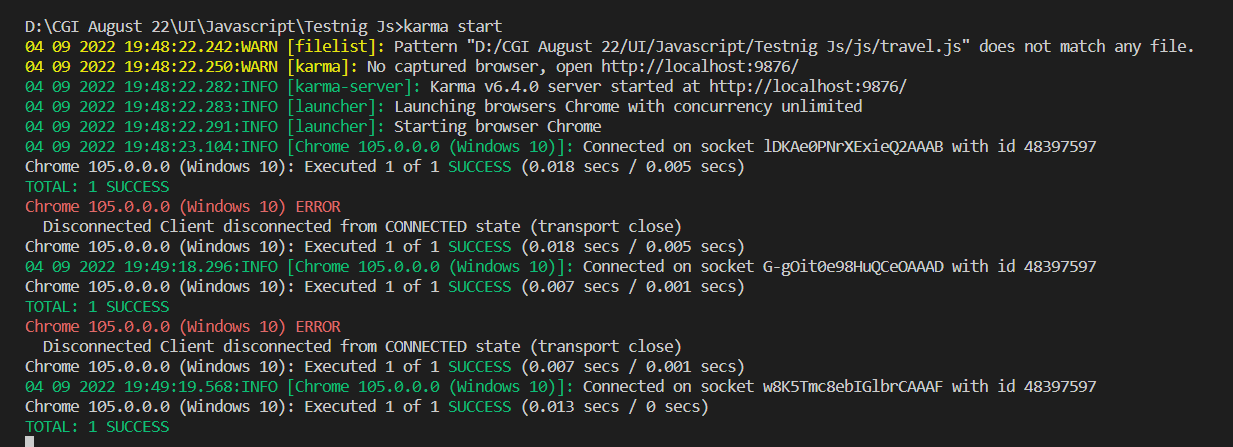
Press tab to list possible options. Enter empty string to move to the next question.

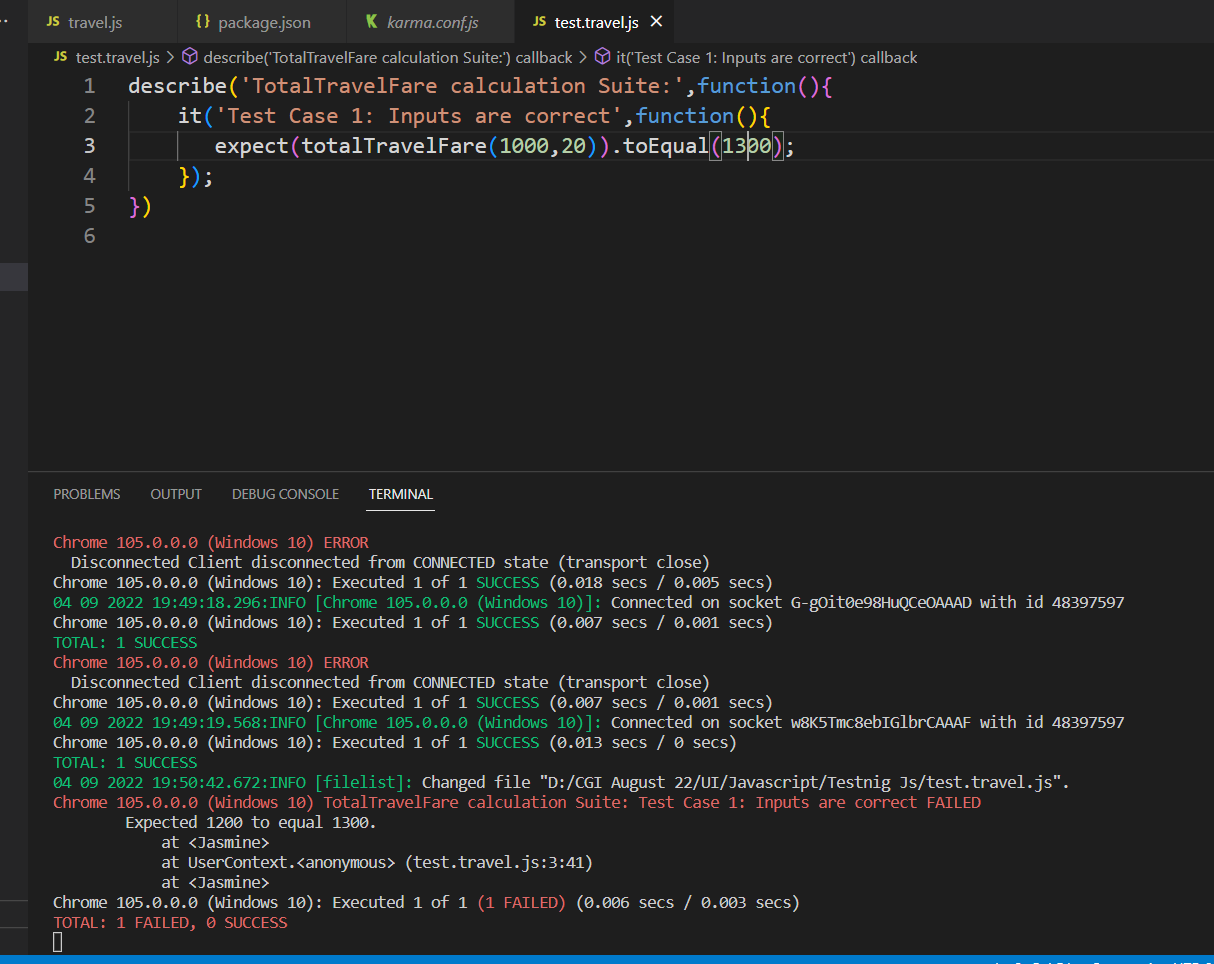
> Chrome











Jasmine components

To understand the features of **Jasmine**framework and to use it to solve the actual requirement, let us consider a simple function **totalTravelFare**(baseFare, taxPercentage) as shown below:

1. function totalTravelFare(baseFare, taxPercentage){
2. var finalFare;
3. finalFare = baseFare\*(1+taxPercentage/100);
4. return finalFare;
5. }

To test this code, we have to follow the below steps:

**Step 1**: A test suite should be created

**Step 2**: Test specs should be created for all the possible scenarios

**Step 3**: Test specs should be created in order to check whether the result variable is defined or not

**Step 4**: Test suite should be executed with the help of Karma

A test suite is a grouping of relevant test cases which are executed together. We can create a test suite with the help of built-in function describe().

**describe(title, function(){})**

* It is a global Jasmine function
* It helps in defining the test context by creating a new test suite
* It accepts two parameters:
  + A title (of string type) or name of the test suite
  + A function containing specs which belong to this suite

For example, we can create a test suite for **totalTravelFare** as shown below.

1. describe('TotalTravelFare calculation Suite:',function(){
2. });

A test spec is the actual test case. Now we will write spec inside the suite, with the help of Jasmine function "it".

**it()**

* It is also a global Jasmine function
* It helps translate the acceptance criteria into Jasmine spec
* It accepts two parameters:
  + Title or name of the test suite
  + Function containing spec code

For example, we can create a test spec for **totalTravelFare** as shown below.

1. describe('TotalTravelFare calculation Suite:',function(){
2. it('Test Case 1: Inputs are correct',function(){
3. expect(totalTravelFare(1000,20)).toEqual(1200);
4. });
5. })

# 

**expect()**

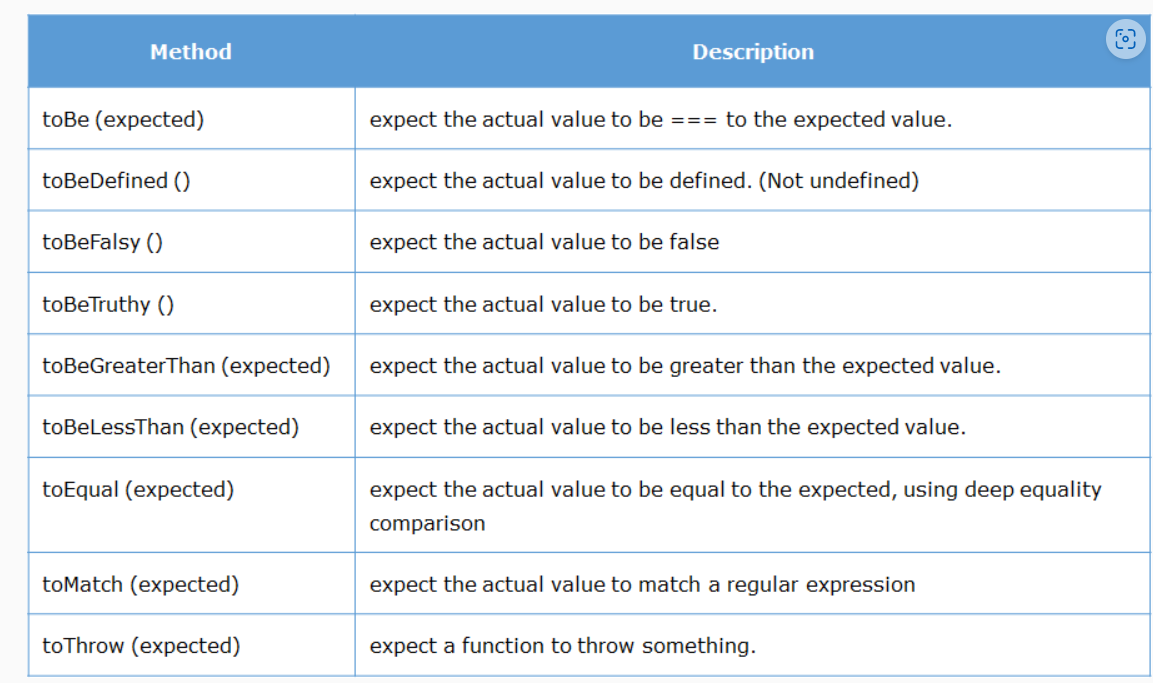
* It is a global Jasmine function that helps in writing the assertions.
* It takes only one parameter: Actual value to be tested

# 

**toEqual()**

* Matcher is used to compare the actual and expected output

Now we will see a simple demo, where we can create a simple test case and run it using Karma test runner.



So far we have just written some test cases with .js extension. To run these test cases we need to install jasmine and karma. Karma is a test runner tool. These tools need to be downloaded and installed. Node has a something called npm ( Node Package Manager ) which helps us install such tools.

Type the below commands in the vscode terminal:

1. npm install -g karma jasmine-core

Karma is:

* A tool that spawns a web server which executes the source code against the test code for each browsers connected.
* When executed, it automatically captures the browser specified by the developer during Karma configuration.
* It then displays the results on the command line.
* It watches all the files specified within configuration file and if there are any changes, it will trigger the corresponding spec again on the browser.

In the last section we learnt the following about unit testing in JavaScript.

* Create a test suite
* Create a test spec
* Create positive and negative test cases
* Run test suites with the help of Karma
* Use different matchers in Jasmine