

# Setup Jasmine + Unit Test Example

#example: **calculator**

Site:

<https://sixsentix.udemy.com/course/unit-testing-your-javascript-with-jasmine/learn/lecture/6574984#overview>

GitHub

<https://github.com/juanlizarazo/jasmine-casts>

<https://github.com/reypena/jasmine-casts>

*\*Navigate to Releases Tab and download latest version*

Unit Tests:

Testing individual units of code

Units are the smallest testable part (function)

Why Unit Tests?

Trust changes you make on existing (already tested) code.

Code quality

Changes occur quickly

Documents your own code

Automation

Why Jasmine?

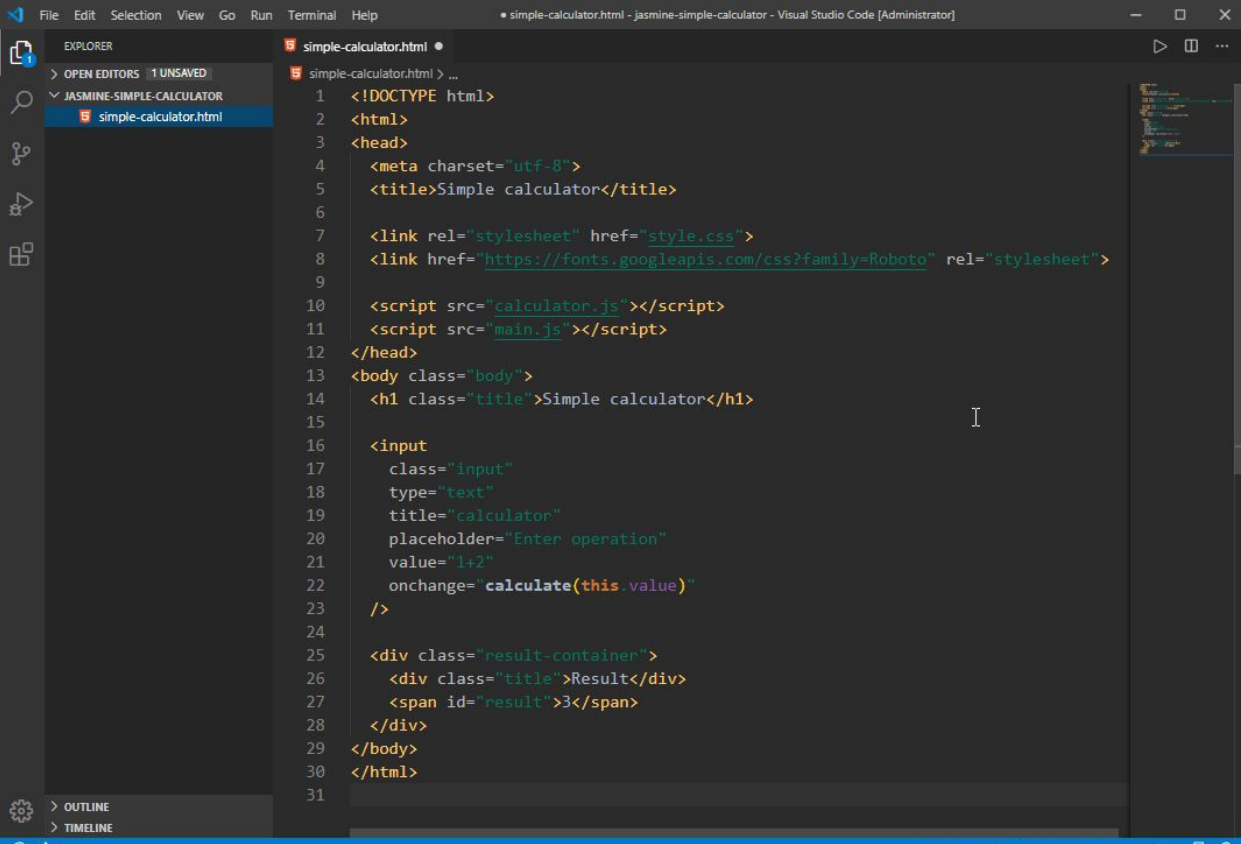
Comes out of the box with everything you need to test your code.

Course Overview

1. Course intro
2. Jasmine functions
3. Matchers
4. Organizing your specs
5. Spying on your code
6. Testing asynchronous code
7. Test reports and continuous integration
8. What next!

# Jasmine Setup with Example

- Create New File "simple-calculator"  
(C:\Users\username\Documents\TestingProjects\my-projects\jasmine-simple-calculator)
- Open VSCode
- Open Folder  
(C:\Users\Furman\Documents\TestingProjects\my-projects\jasmine-simple-calculator)
- Create New File simple-calculator.html

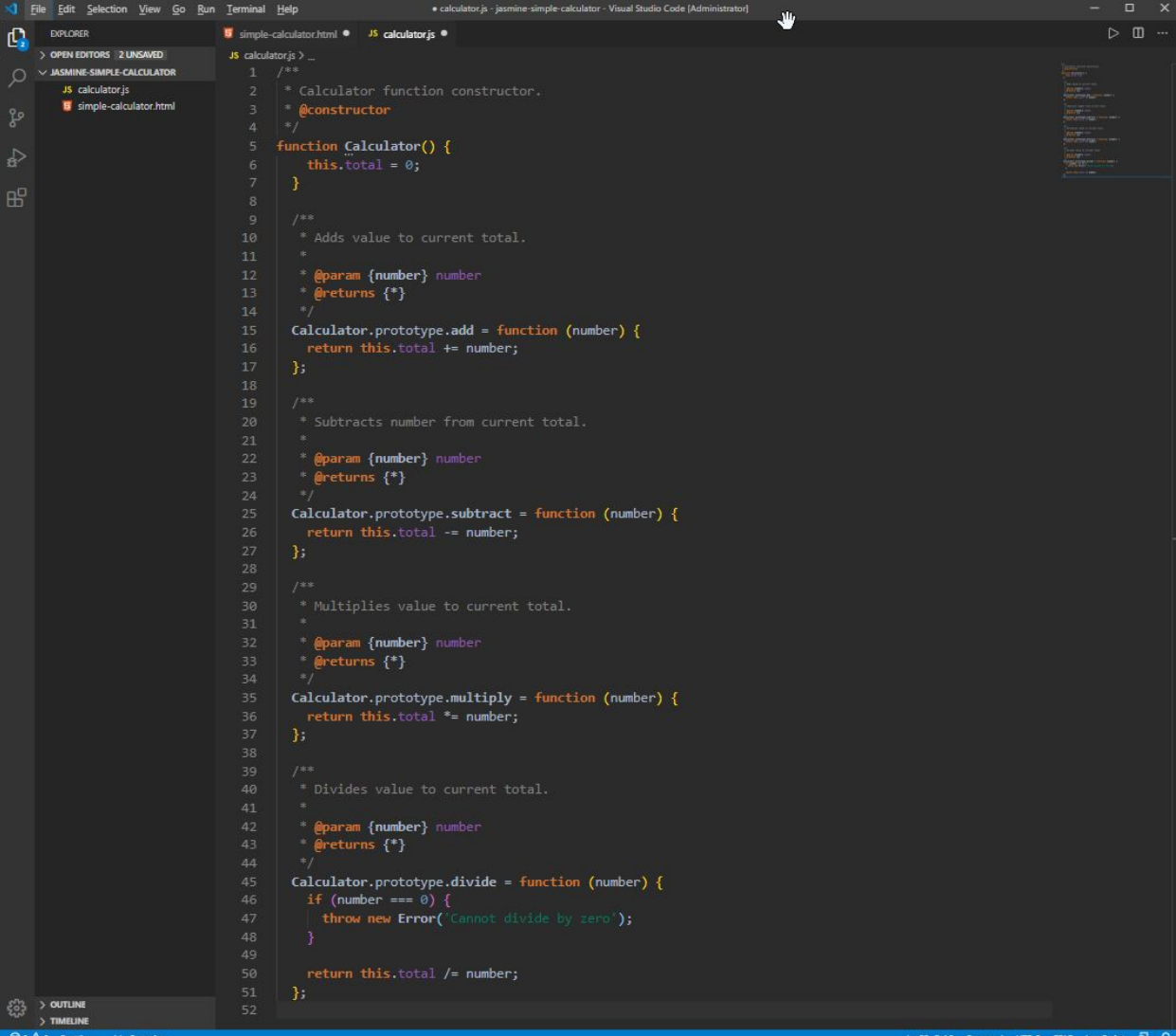


The screenshot shows the Visual Studio Code editor with the file 'simple-calculator.html' open. The Explorer sidebar on the left shows the project structure with 'JASMINE-SIMPLE-CALCULATOR' and 'simple-calculator.html'. The main editor area displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Simple calculator</title>
6
7   <link rel="stylesheet" href="style.css">
8   <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
9
10  <script src="calculator.js"></script>
11  <script src="main.js"></script>
12 </head>
13 <body class="body">
14   <h1 class="title">Simple calculator</h1>
15
16   <input
17     class="input"
18     type="text"
19     title="calculator"
20     placeholder="Enter operation"
21     value="1+2"
22     onchange="calculate(this.value)"
23   />
24
25   <div class="result-container">
26     <div class="title">Result</div>
27     <span id="result">3</span>
28   </div>
29 </body>
30 </html>
31
```

The status bar at the bottom indicates 'Ln 31, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'HTML', and a search icon.

- Create a New File: calculator.js



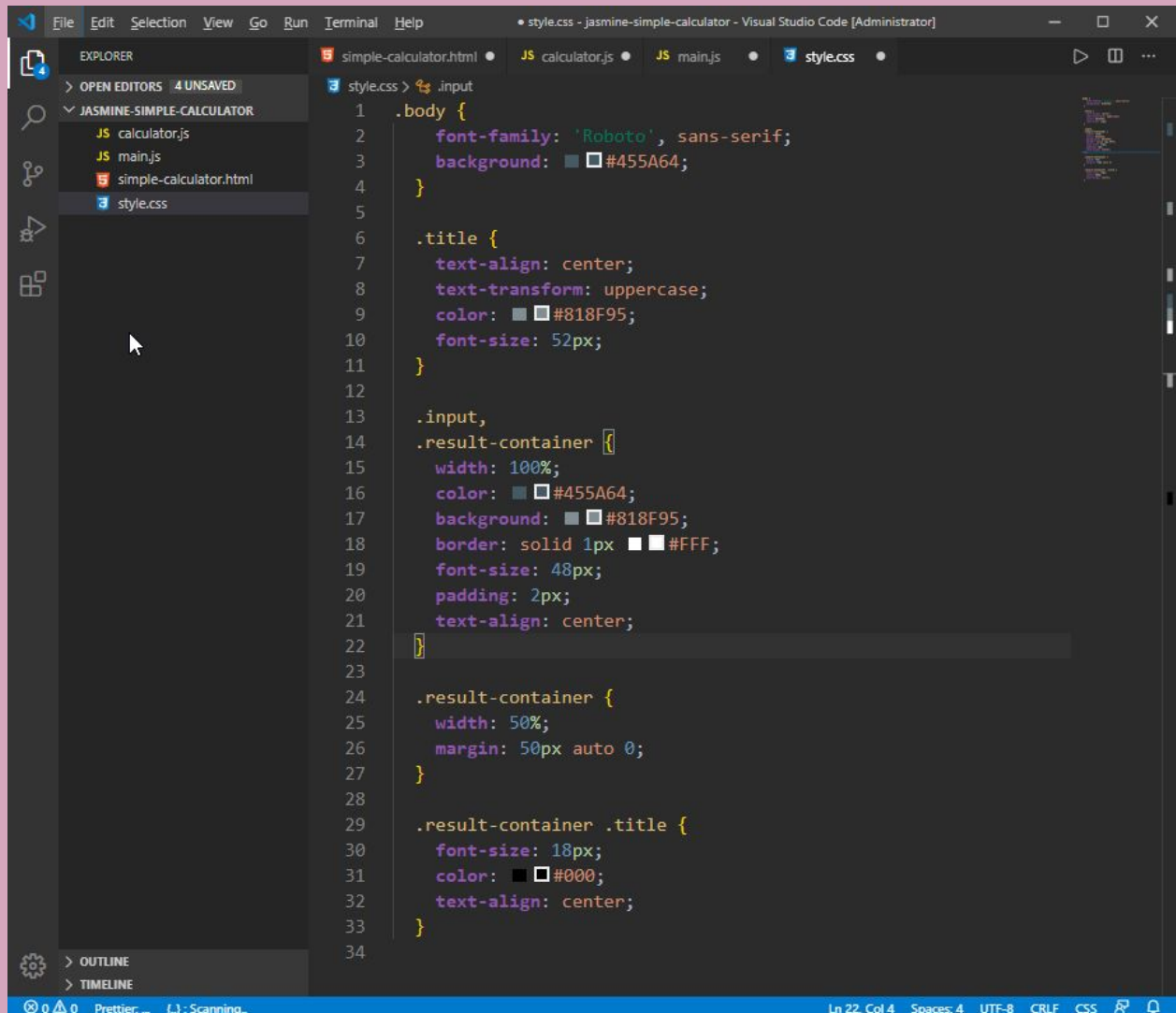
The screenshot shows the Visual Studio Code editor with the file explorer on the left, the editor window in the center, and the terminal at the bottom. The file explorer shows a project named 'JASMINE-SIMPLE-CALCULATOR' with two files: 'calculator.js' and 'simple-calculator.html'. The editor window displays the content of 'calculator.js', which is a JavaScript file defining a 'Calculator' constructor function. The code includes comments for each method and uses JSDoc-style annotations for parameters and returns. The methods implemented are 'add', 'subtract', 'multiply', and 'divide'. The 'divide' method includes a check for division by zero, throwing an error if the divisor is zero. The status bar at the bottom indicates the current line is 52, column 3, with 4 spaces, using UTF-8 encoding, CRLF line endings, and the JavaScript language.

```
1  /**
2   * Calculator function constructor.
3   * @constructor
4   */
5  function Calculator() {
6      this.total = 0;
7  }
8
9  /**
10   * Adds value to current total.
11   *
12   * @param {number} number
13   * @returns {*}
14   */
15  Calculator.prototype.add = function (number) {
16      return this.total += number;
17  };
18
19  /**
20   * Subtracts number from current total.
21   *
22   * @param {number} number
23   * @returns {*}
24   */
25  Calculator.prototype.subtract = function (number) {
26      return this.total -= number;
27  };
28
29  /**
30   * Multiplies value to current total.
31   *
32   * @param {number} number
33   * @returns {*}
34   */
35  Calculator.prototype.multiply = function (number) {
36      return this.total *= number;
37  };
38
39  /**
40   * Divides value to current total.
41   *
42   * @param {number} number
43   * @returns {*}
44   */
45  Calculator.prototype.divide = function (number) {
46      if (number === 0) {
47          throw new Error('Cannot divide by zero');
48      }
49
50      return this.total /= number;
51  };
52
```

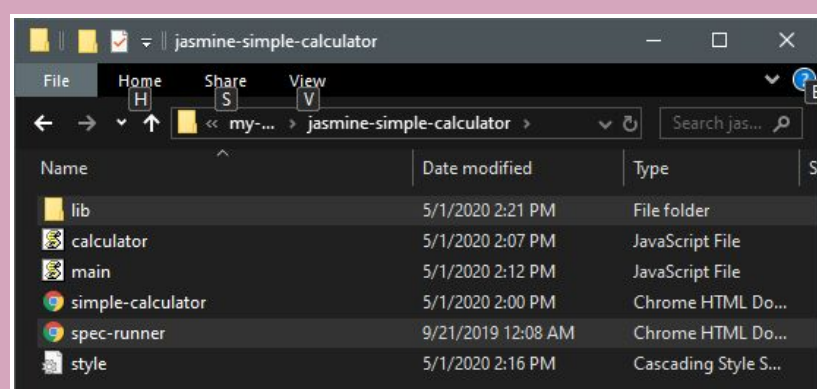
- Create New File main.js

```
1 //Calculates result for a simple mathematical expression.
2
3 function calculate(inputValue) {
4   const expression = /\+|\-|\*|\/|/;
5   const numbers = inputValue.split(expression);
6
7   const numberA = parseInt(numbers[0]);
8   const numberB = parseInt(numbers[1]);
9
10  const operation = inputValue.match(expression);
11
12   if (Number.isNaN(numberA) || Number.isNaN(numberB) || operation === null) {
13     updateResult('Expression not recognized');
14     return;
15   }
16
17   const calculator = new Calculator();
18   calculator.add(numberA);
19
20   let result;
21   switch(operation[0]) {
22     case '+':
23     result = calculator.add(numberB);
24     break;
25     case '-':
26     result = calculator.subtract(numberB);
27     break;
28     case '*':
29     result = calculator.multiply(numberB);
30     break;
31     case '/':
32     result = calculator.divide(numberB);
33     break;
34     default:
35     result = 'Operation not recognized';
36   }
37
38   updateResult(result);
39 }
40
41 /**
42  * Updates result in DOM element.
43  * @param {string} result
44  */
45 function updateResult(result) {
46   const element = document.getElementById('result');
47
48   if (element) {
49     element.innerText = result;
50   }
51 }
52
```

- Create New File style.css



- Download Jasmine Framework
- Go to: <https://github.com/jasmine/jasmine/releases> and download: jasmine-standalone-3.5.0.zip
- Open jasmine-standalone-3.5.0 file and copy SpecRunner.html, and lib folder



- Open lib folder and delete jasmine\_favicon.png and console.js files
- Change spec.runner.html

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Simple calculator</title>
6
7   <link rel="stylesheet" href="lib/jasmine.css">
8
9   <script src="lib/jasmine.js"></script>
10  <script src="lib/jasmine-html.js"></script>
11  <script src="lib/boot.js"></script>
12
13 </head>
14 </html>
15

```

- Create new file calculator.spec.js

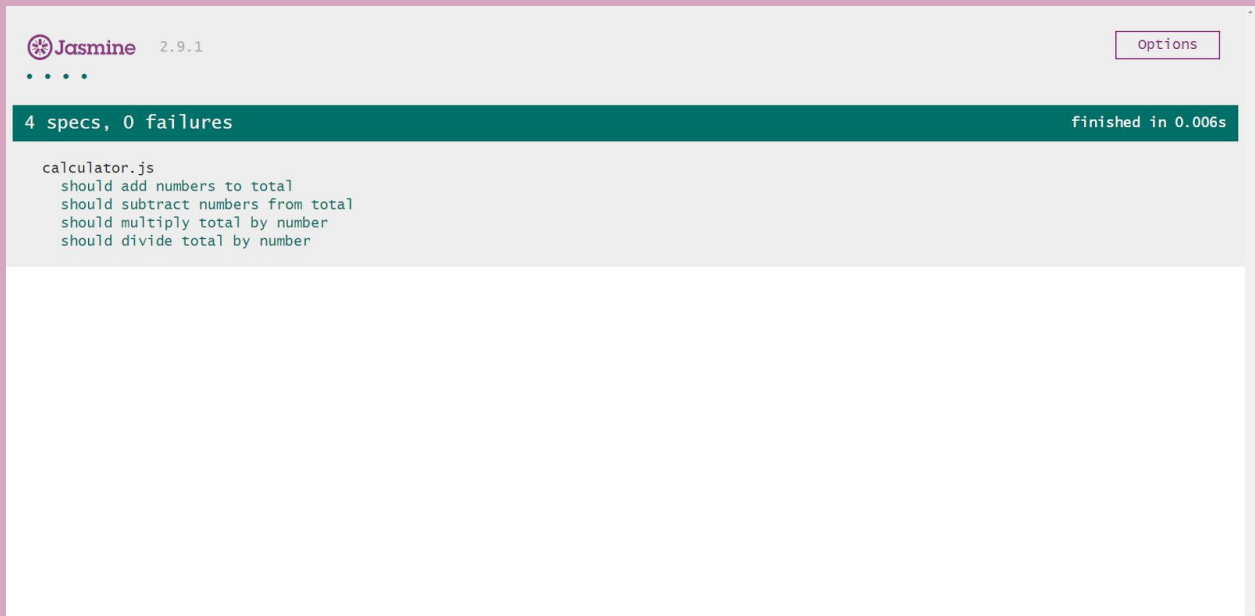
```

1 describe('calculator.js', function() {
2   it('should add numbers to total', function() {
3     const calculator = new Calculator();
4     calculator.add(5);
5
6     expect(calculator.total).toBe(5);
7   });
8
9   it('should subtract numbers from total', function() {
10    const calculator = new Calculator();
11    calculator.total = 30;
12    calculator.subtract(5);
13
14    expect(calculator.total).toBe(25);
15  });
16
17  it('should multiply total by number', function() {
18    const calculator = new Calculator();
19    calculator.total = 100;
20    calculator.multiply(2);
21
22    expect(calculator.total).toBe(200);
23  });
24
25  it('should divide total by number', function() {
26    const calculator = new Calculator();
27    calculator.total = 200;
28    calculator.divide(2);
29
30    expect(calculator.total).toBe(100);
31  });
32 });
33

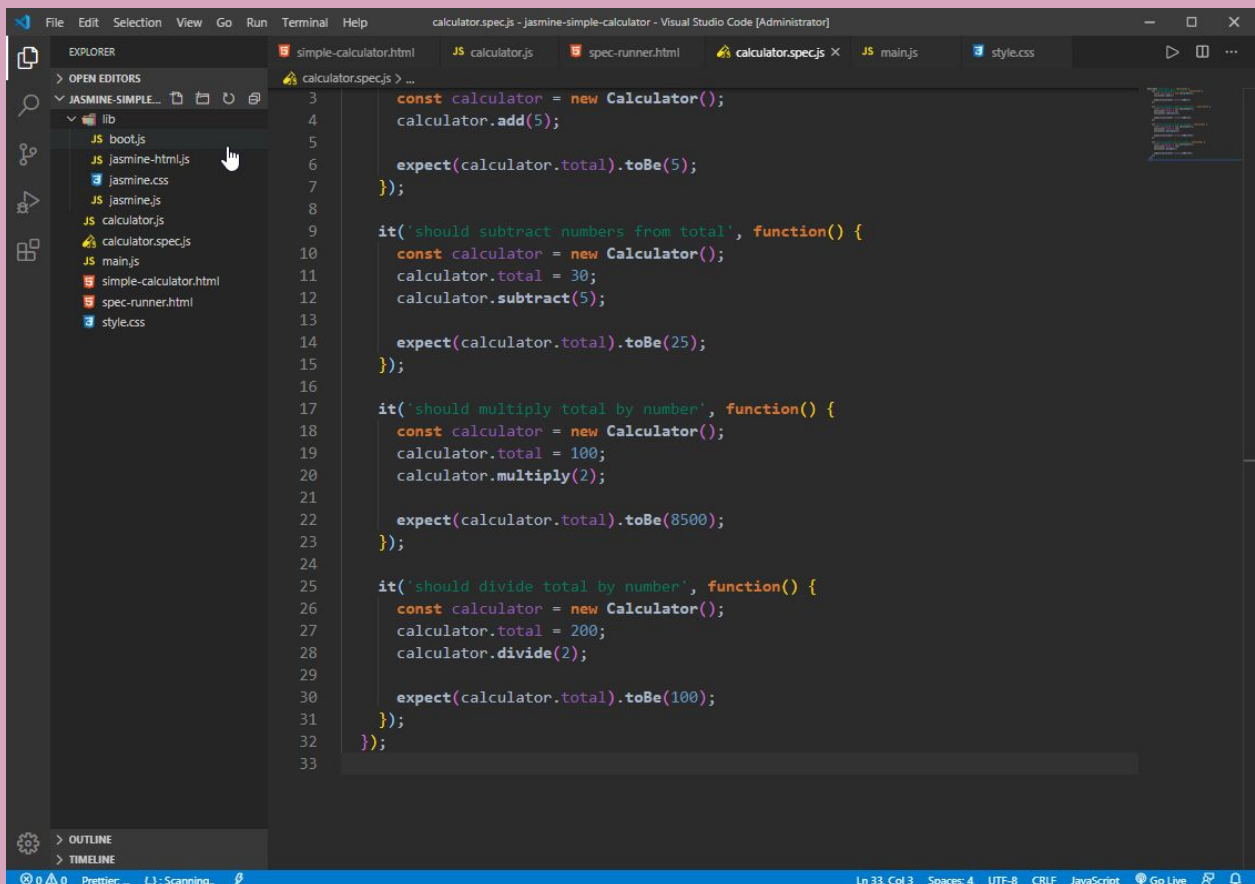
```



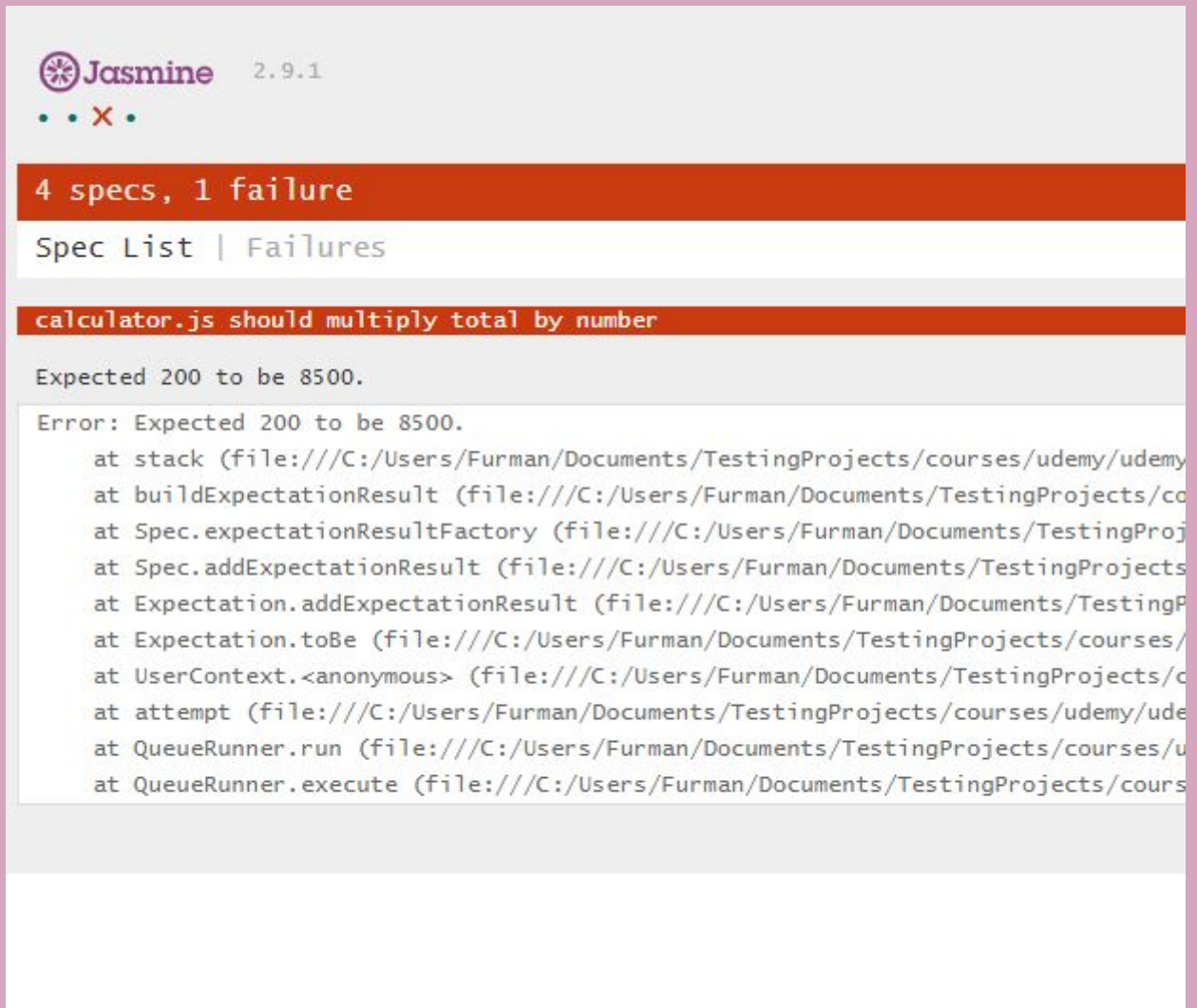
- Run spec-runner.html in your browser



- Change one test expectations toBe(8500)



- Run spec-runner.html in your browser
- 



Later:

- USING JASMINE MATCHERS  
<https://github.com/JamieMason/Jasmine-Matchers>
- ORGANIZING YOUR SPEC