Part 2: Unit testing activity

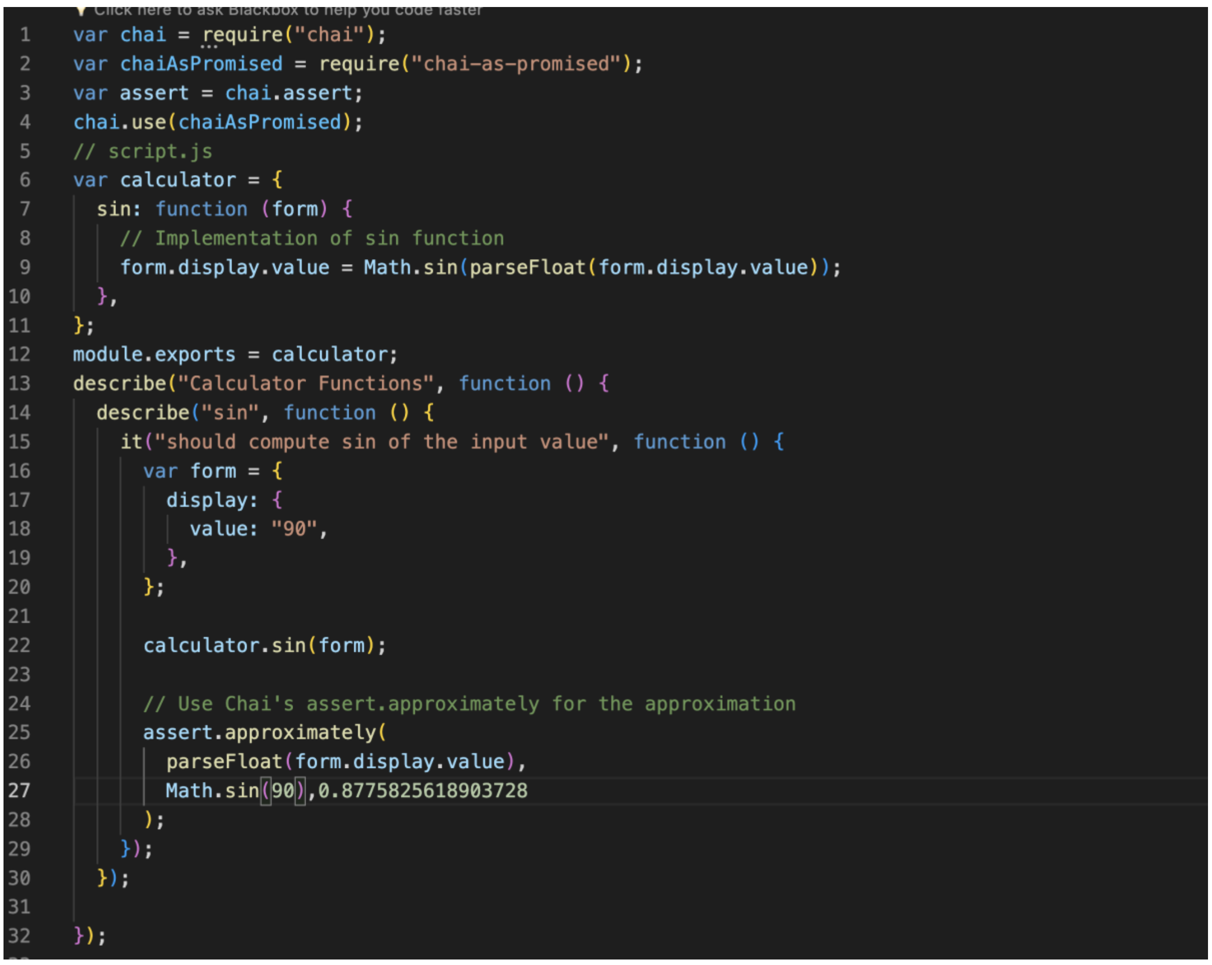
Test set 1 target function name :

Test Set 1 targets the trigonometry functions in the calculator, namely the sine (sin), cosine (cos), and tangent (tan) functions.

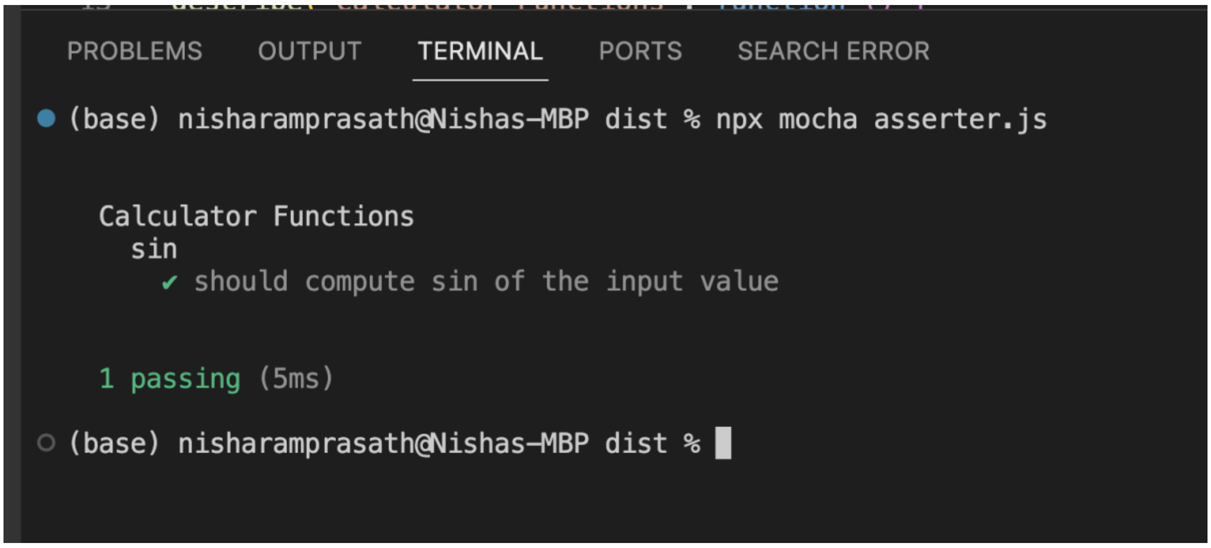
Test set 1 written explanation of strategy:

Test Set 1 targets the trigonometry functions in the calculator, namely the sine , cosine , and tangent functions. The objective is to verify the correctness and functionality of these trigonometric functions

Test set 1 test 1 coded, explained and run :

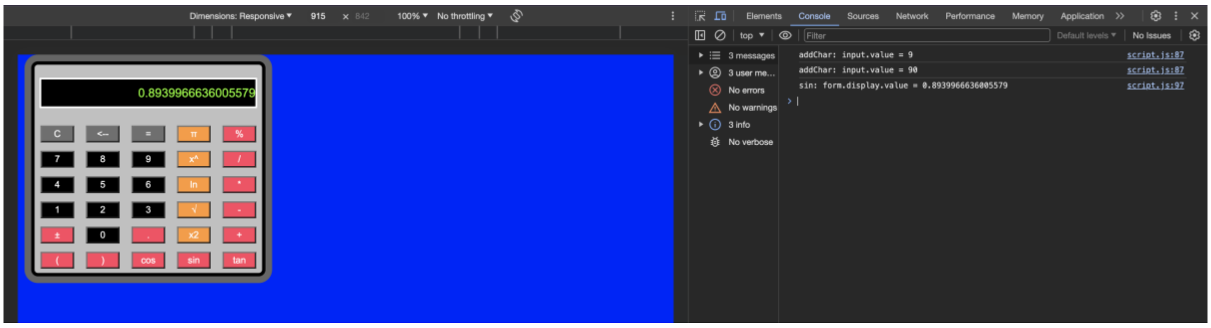


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);
// script.js
var calculator = {
  sin: function (form) {
    // Implementation of sin function
    form.display.value = Math.sin(parseFloat(form.display.value));
  },
};
module.exports = calculator;
describe("Calculator Functions", function () {
  describe("sin", function () {
    it("should compute sin of the input value", function () {
      var form = {
        display: {
          value: "90",
        },
      };

      calculator.sin(form);

      // Use Chai's assert.approximately for the approximation
      assert.approximately(
        parseFloat(form.display.value),
        Math.sin(90),0.8775825618903728
      );
    });
  });

});
```



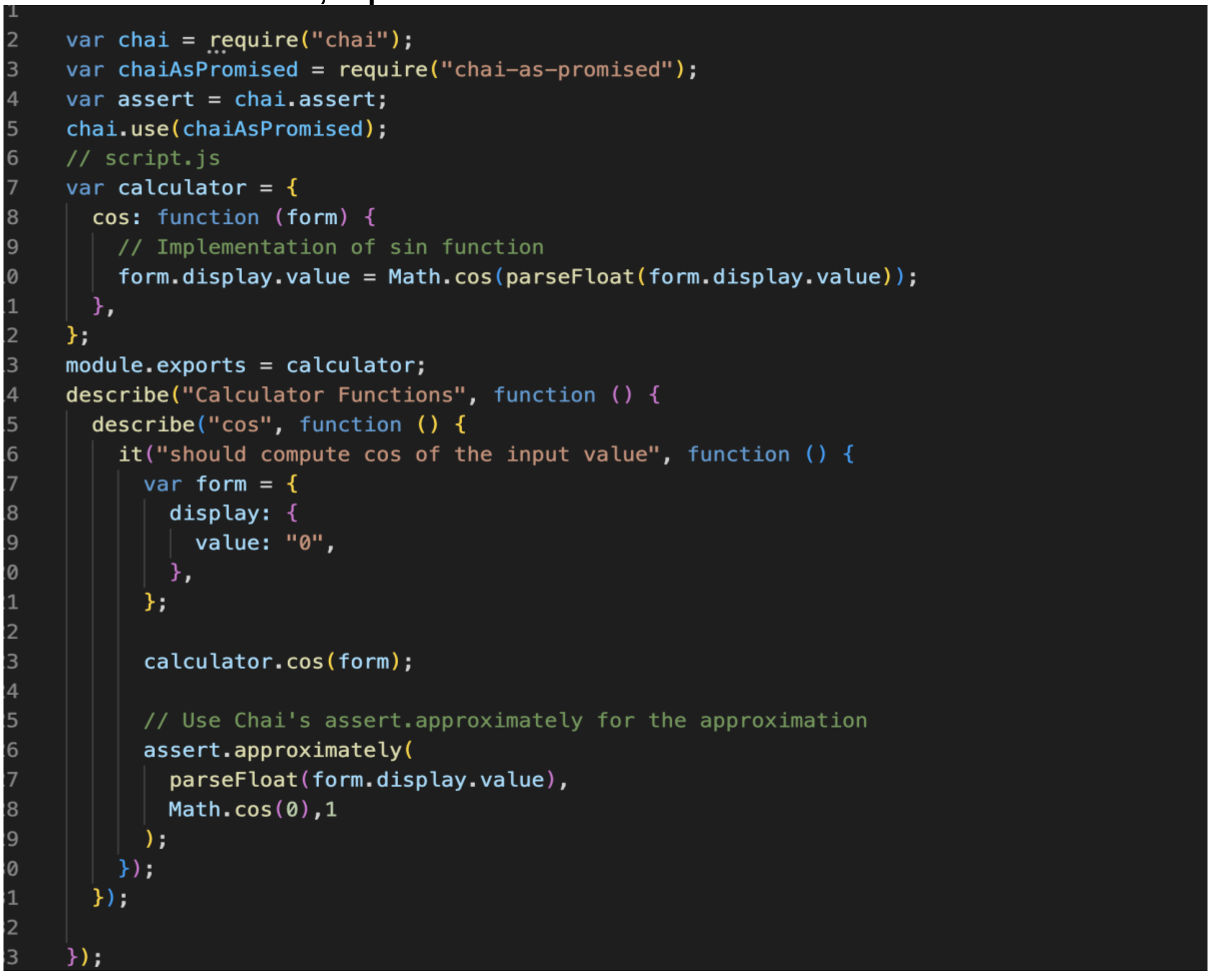
```
(base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


  Calculator Functions
    sin
      ✔ should compute sin of the input value


  1 passing (5ms)

(base) nisharamprasath@Nishas-MBP dist %
```

The test case checks the behaviour of the sin function, which is from the script.js.

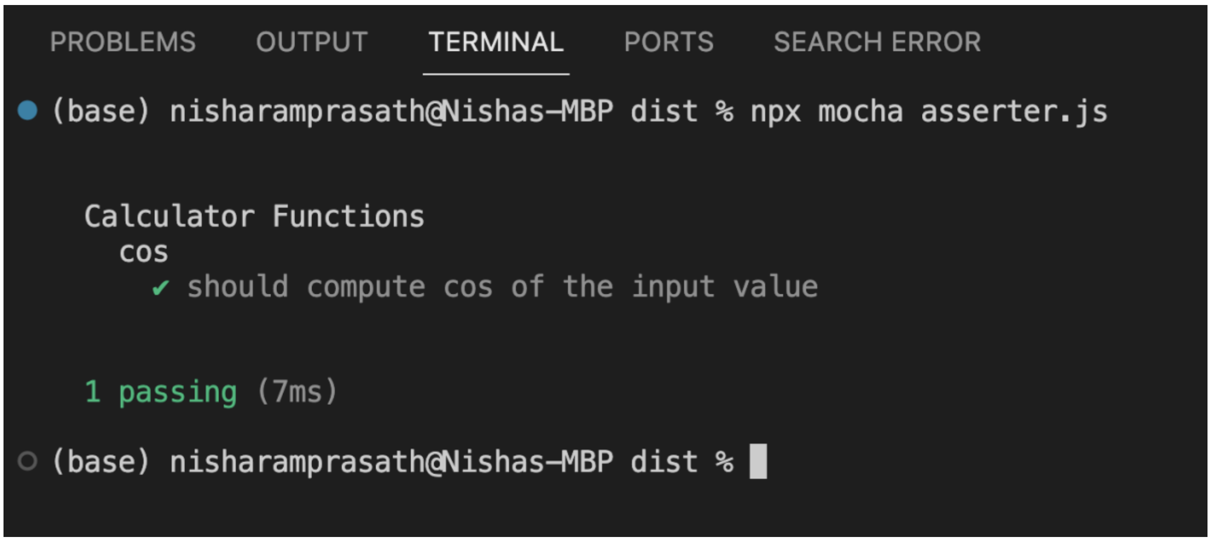


```
addChar: input.value = 9                                  script.js:87
addChar: input.value = 90                                 script.js:87
sin: form.display.value = 0.8939966636005579              script.js:97
```

Test set 1 test 2 coded, explained and run :

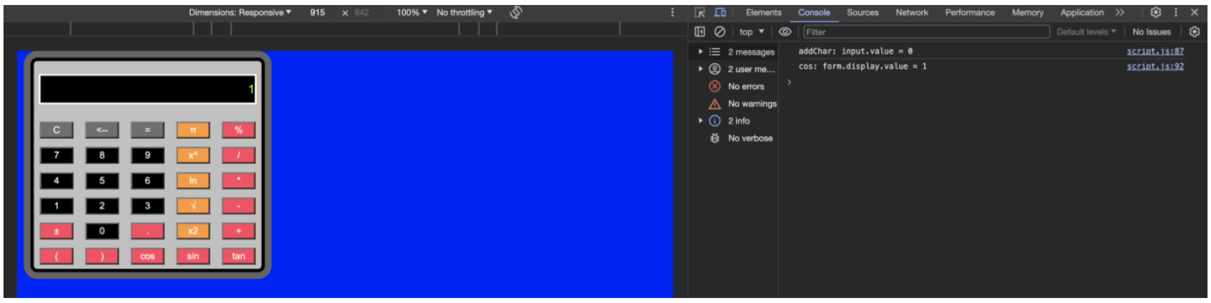


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);
// script.js
var calculator = {
  cos: function (form) {
    // Implementation of sin function
    form.display.value = Math.cos(parseFloat(form.display.value));
  },
};
module.exports = calculator;
describe("Calculator Functions", function () {
  describe("cos", function () {
    it("should compute cos of the input value", function () {
      var form = {
        display: {
          value: "0",
        },
      };

      calculator.cos(form);

      // Use Chai's assert.approximately for the approximation
      assert.approximately(
        parseFloat(form.display.value),
        Math.cos(0),1
      );
    });
  });

});
```



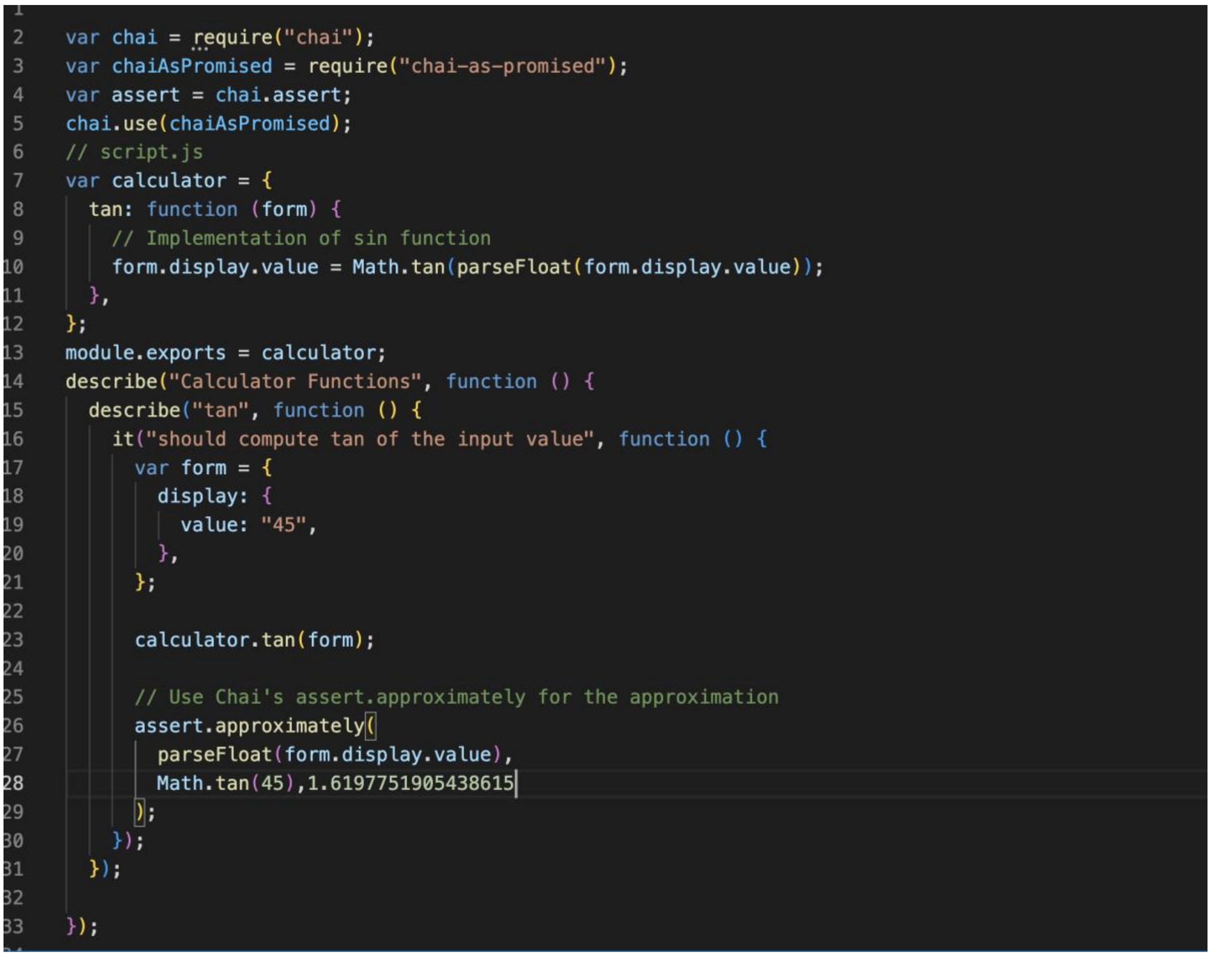
```
(base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


  Calculator Functions
    cos
      ✔ should compute cos of the input value


  1 passing (7ms)

(base) nisharamprasath@Nishas-MBP dist %
```

The test case checks the behaviour of the cos function, which is from the script.js.

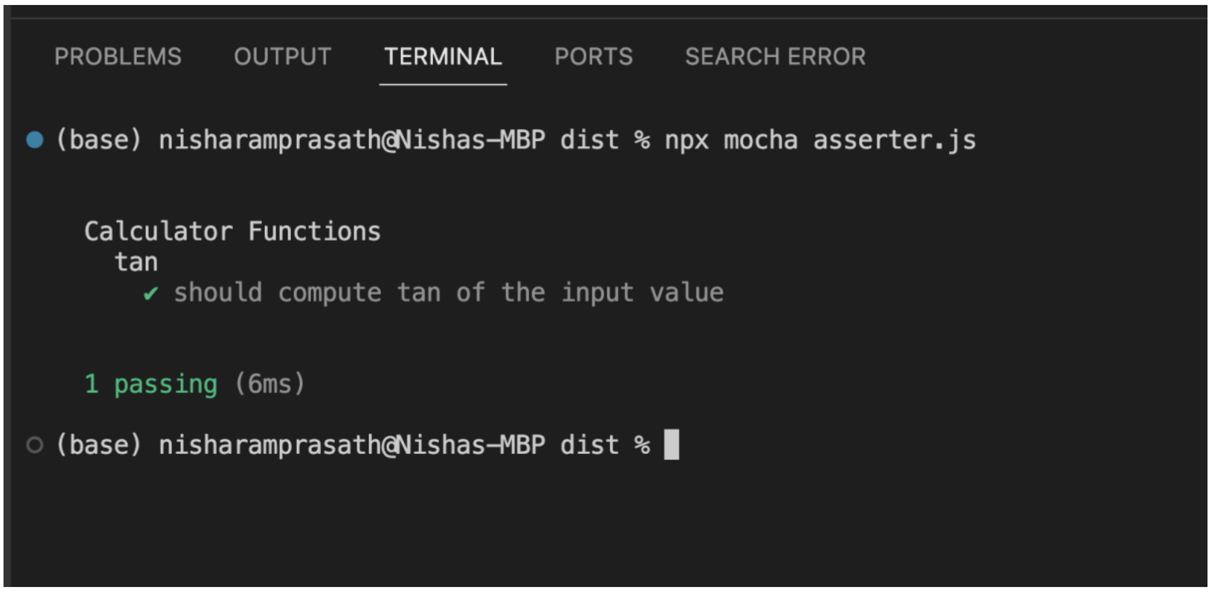


```
addChar: input.value = 0                    script.js:87
cos: form.display.value = 1                 script.js:92
```

Test set 1 test 3 coded, explained and run :

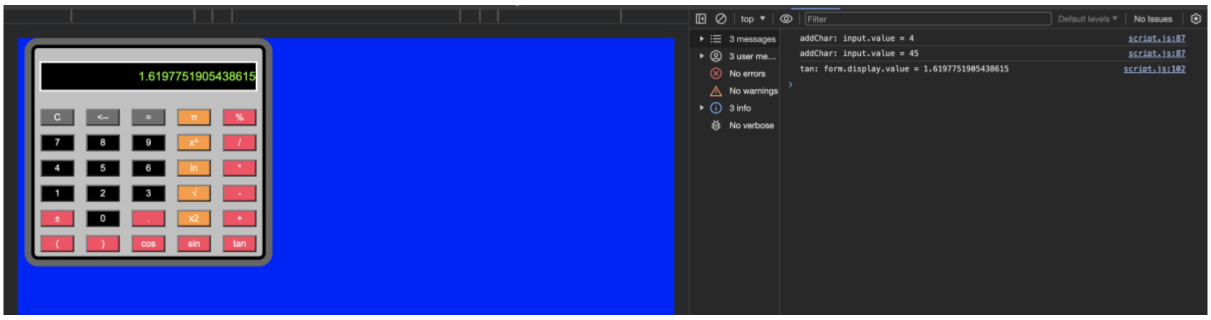


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);
// script.js
var calculator = {
  tan: function (form) {
    // Implementation of sin function
    form.display.value = Math.tan(parseFloat(form.display.value));
  },
};
module.exports = calculator;
describe("Calculator Functions", function () {
  describe("tan", function () {
    it("should compute tan of the input value", function () {
      var form = {
        display: {
          value: "45",
        },
      };

      calculator.tan(form);

      // Use Chai's assert.approximately for the approximation
      assert.approximately(
        parseFloat(form.display.value),
        Math.tan(45),1.6197751905438615
      );
    });
  });

});
```



```
● (base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


    Calculator Functions
      tan
        ✔ should compute tan of the input value


    1 passing (6ms)

○ (base) nisharamprasath@Nishas-MBP dist % █
```

The test case checks the behaviour of the tan function, which is from the script.js.



```
addChar: input.value = 4                              script.js:87
addChar: input.value = 45                             script.js:87
tan: form.display.value = 1.6197751905438615          script.js:102
```

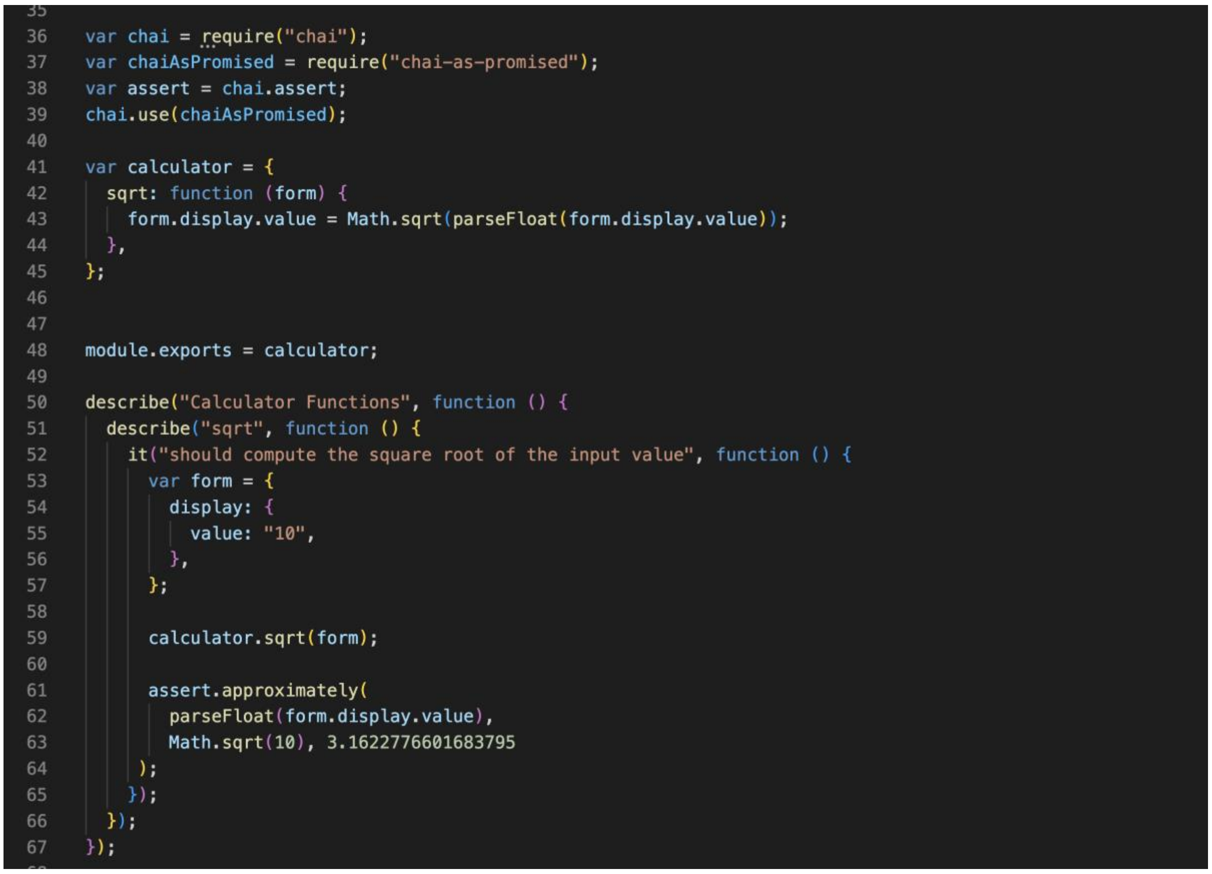
Test set 2 target function name :

Test Set 2 targets the Square Roots , Exponential Functions and Square Function. The objective is to verify the correctness and functionality of these trigonometric functions

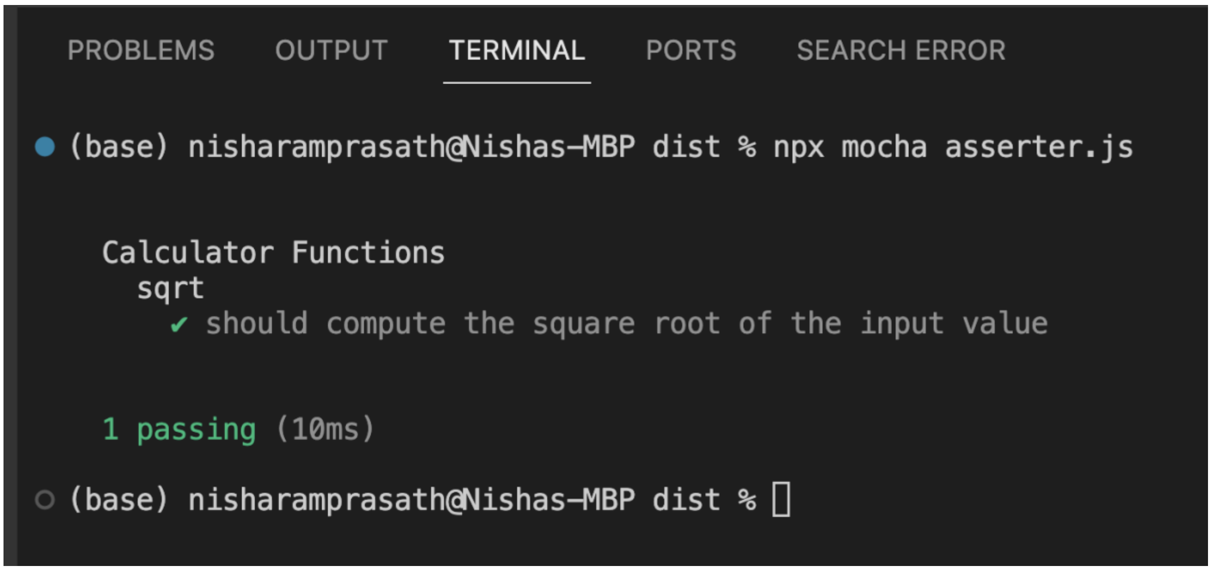
Test set 2 written explanation of strategy:

Test Set 1 targets the trigonometry functions in the calculator, namely the square , square root, and expt functions. The objective is to verify the correctness and functionality of these trigonometric functions

Test set 2 test 1 coded, explained and run (square root):

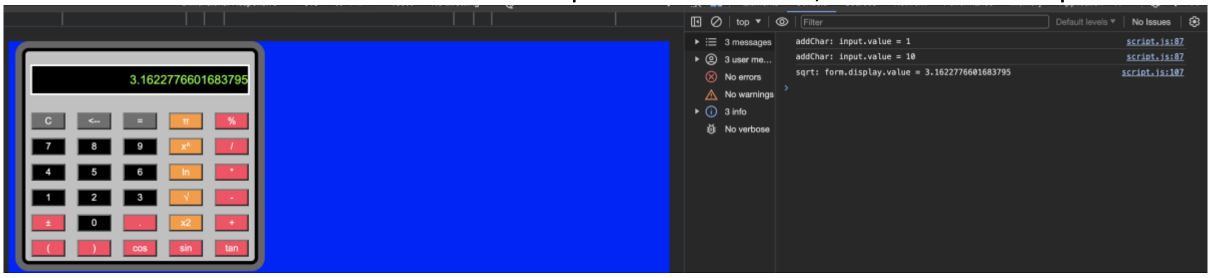


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  sqrt: function (form) {
    form.display.value = Math.sqrt(parseFloat(form.display.value));
  },
};


module.exports = calculator;

describe("Calculator Functions", function () {
  describe("sqrt", function () {
    it("should compute the square root of the input value", function () {
      var form = {
        display: {
          value: "10",
        },
      };

      calculator.sqrt(form);

      assert.approximately(
        parseFloat(form.display.value),
        Math.sqrt(10), 3.1622776601683795
      );
    });
  });
});
```



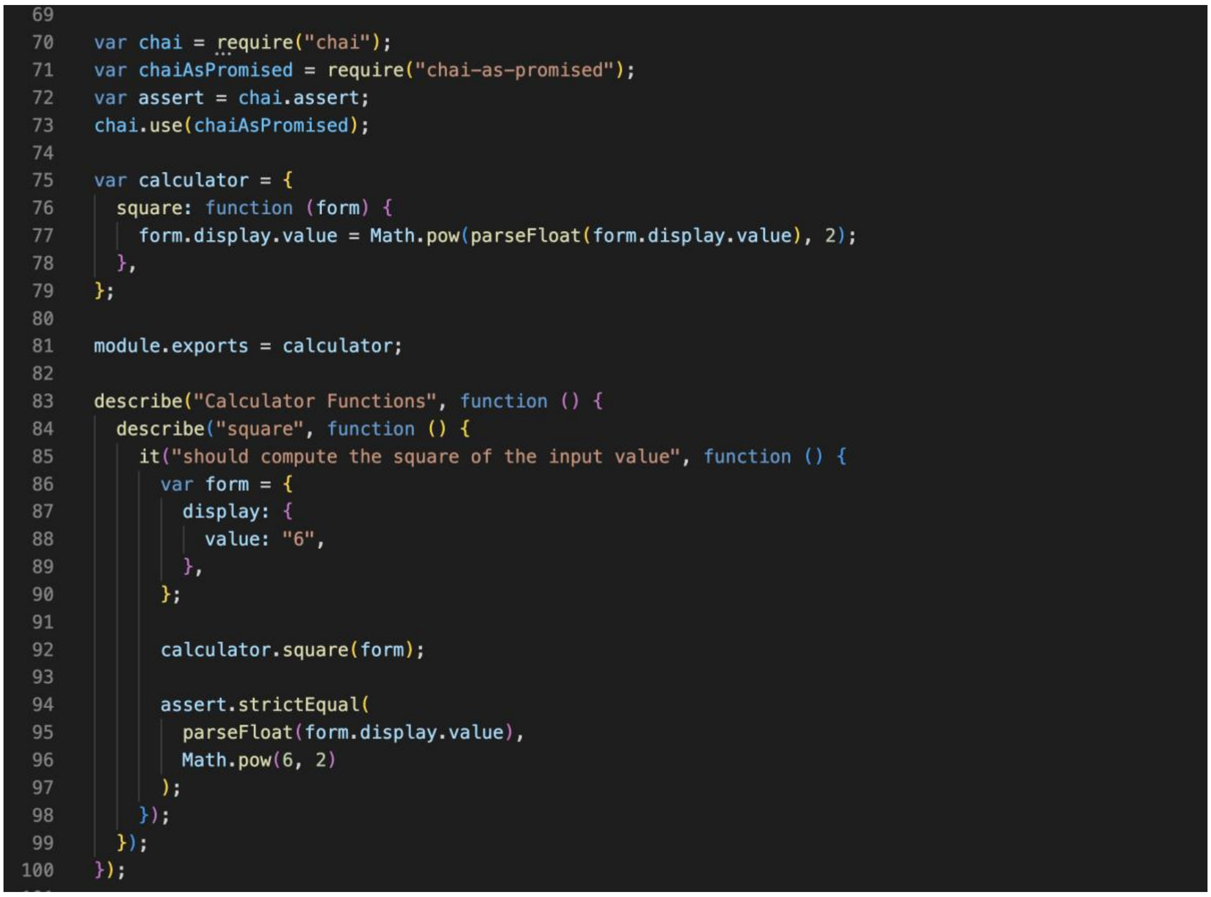
```
(base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


  Calculator Functions
    sqrt
      ✔ should compute the square root of the input value


  1 passing (10ms)

(base) nisharamprasath@Nishas-MBP dist %
```

The test case checks the behaviour of the square root function, which is from script.

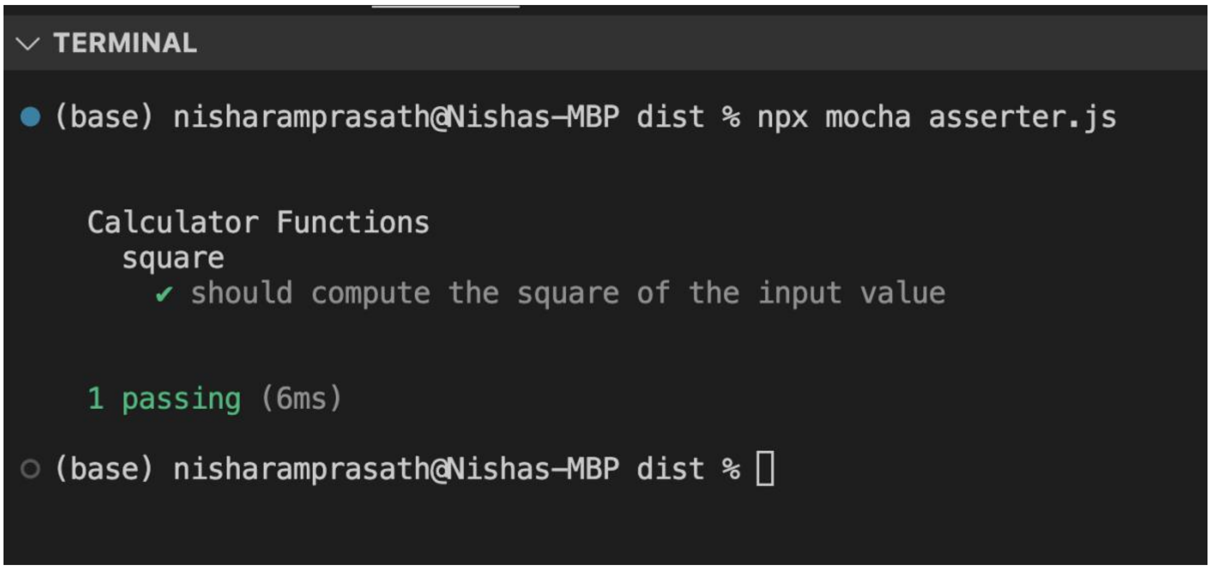


```
addChar: input.value = 1                                  script.js:87
addChar: input.value = 10                                 script.js:87
sqrt: form.display.value = 3.1622776601683795             script.js:107
```

Test set 2 test 2 coded, explained and run (sqaure function):

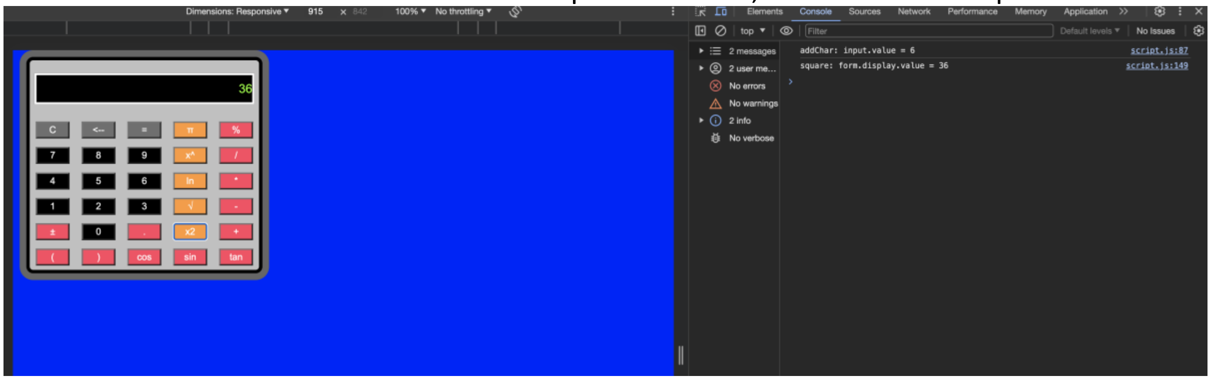


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  square: function (form) {
    form.display.value = Math.pow(parseFloat(form.display.value), 2);
  },
};

module.exports = calculator;

describe("Calculator Functions", function () {
  describe("square", function () {
    it("should compute the square of the input value", function () {
      var form = {
        display: {
          value: "6",
        },
      };

      calculator.square(form);

      assert.strictEqual(
        parseFloat(form.display.value),
        Math.pow(6, 2)
      );
    });
  });
});
```



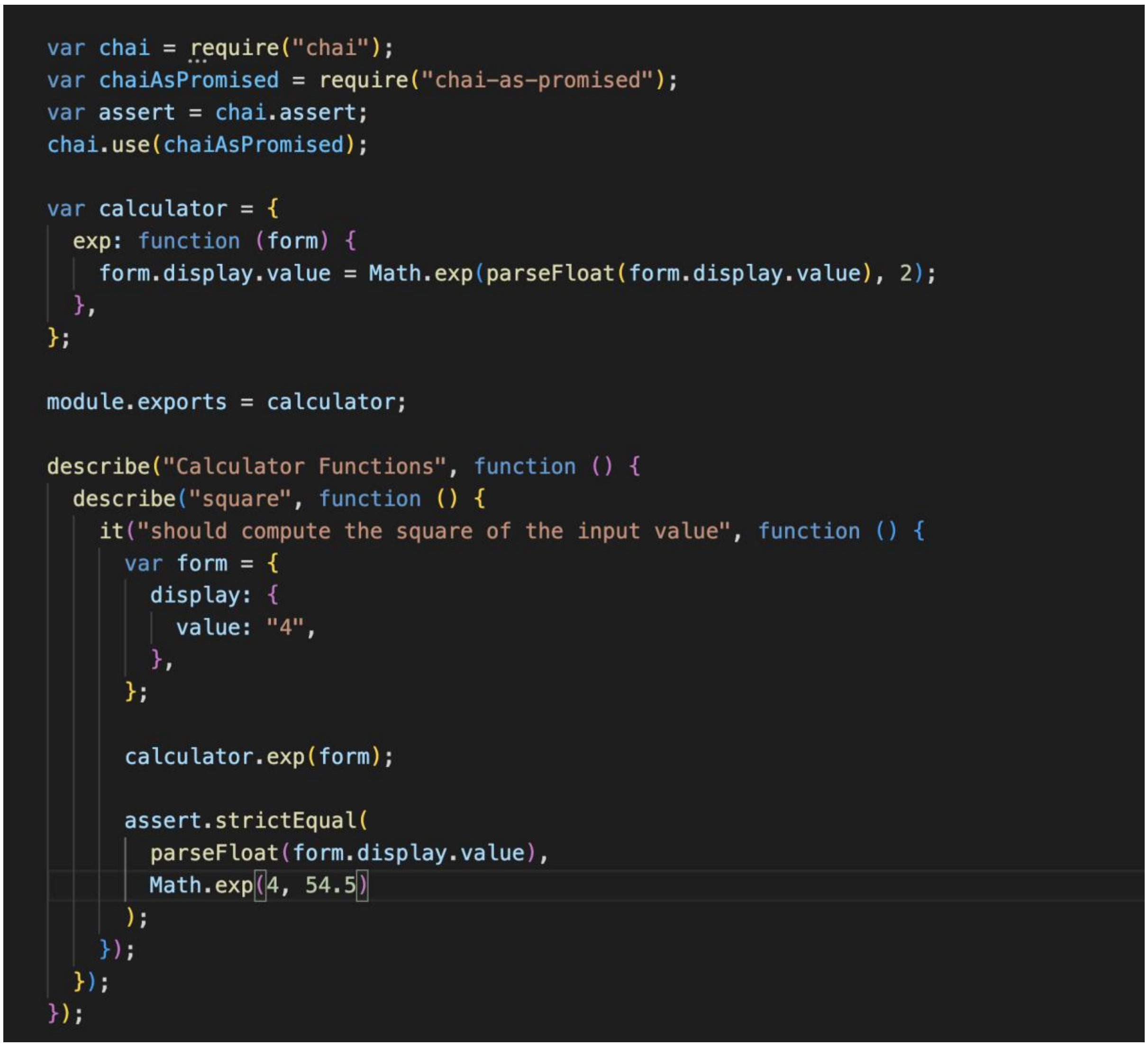
```
TERMINAL

● (base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


  Calculator Functions
    square
      ✔ should compute the square of the input value


  1 passing (6ms)

○ (base) nisharamprasath@Nishas-MBP dist % 
```

The test case checks the behaviour of the square function, which is from script.

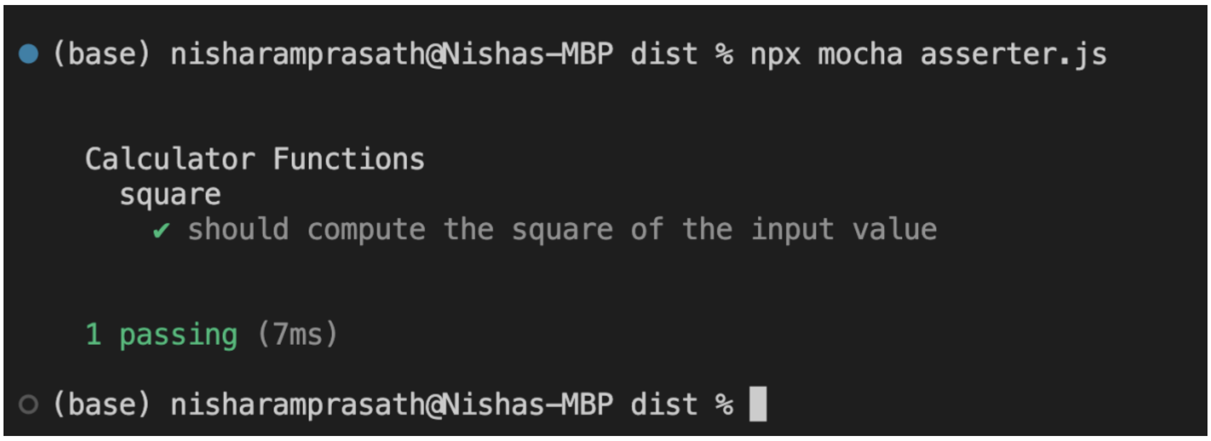


```
addChar: input.value = 6                                        script.js:87
square: form.display.value = 36                                 script.js:149
```

Test set 2 test 3 coded, explained and run (expt function):

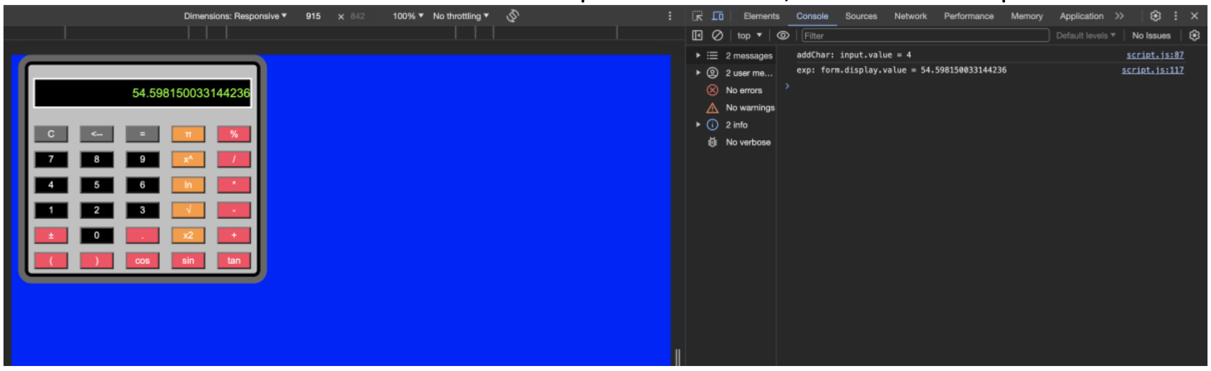


```javascript
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  exp: function (form) {
    form.display.value = Math.exp(parseFloat(form.display.value), 2);
  },
};

module.exports = calculator;

describe("Calculator Functions", function () {
  describe("square", function () {
    it("should compute the square of the input value", function () {
      var form = {
        display: {
          value: "4",
        },
      };

      calculator.exp(form);

      assert.strictEqual(
        parseFloat(form.display.value),
        Math.exp(4, 54.5)
      );
    });
  });
});
```



```
● (base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


    Calculator Functions
      square
        ✔ should compute the square of the input value


    1 passing (7ms)

○ (base) nisharamprasath@Nishas-MBP dist % 
```

The test case checks the behaviour of the expt root function, which from script.



```
addChar: input.value = 4                                          script.js:87
exp: form.display.value = 54.598150033144236                      script.js:117
```

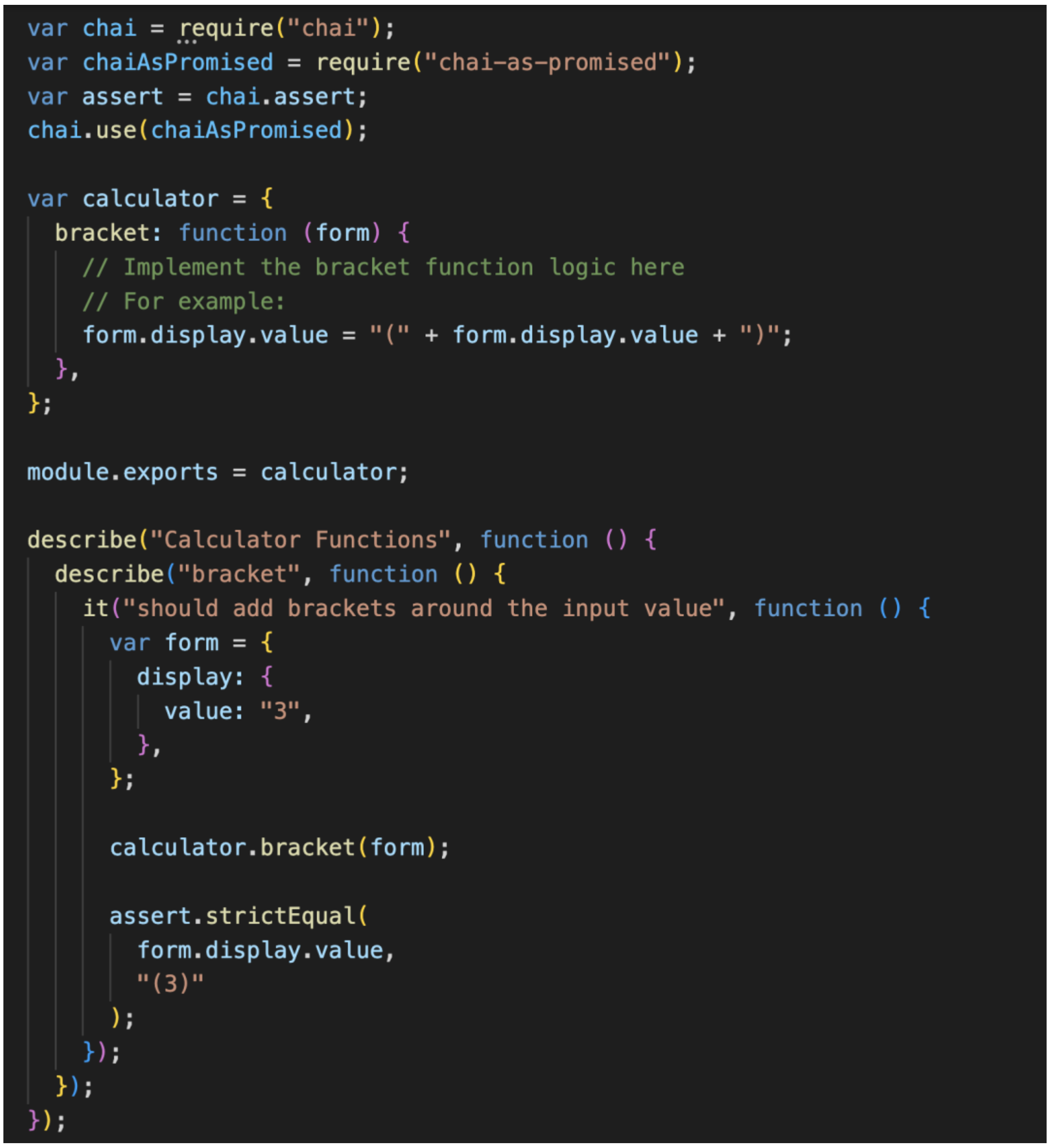
Test set 3 target function name :

Test Set 3 targets the bracket, delete char Functions and compute Function. The objective is to verify the correctness and functionality of these trigonometric functions

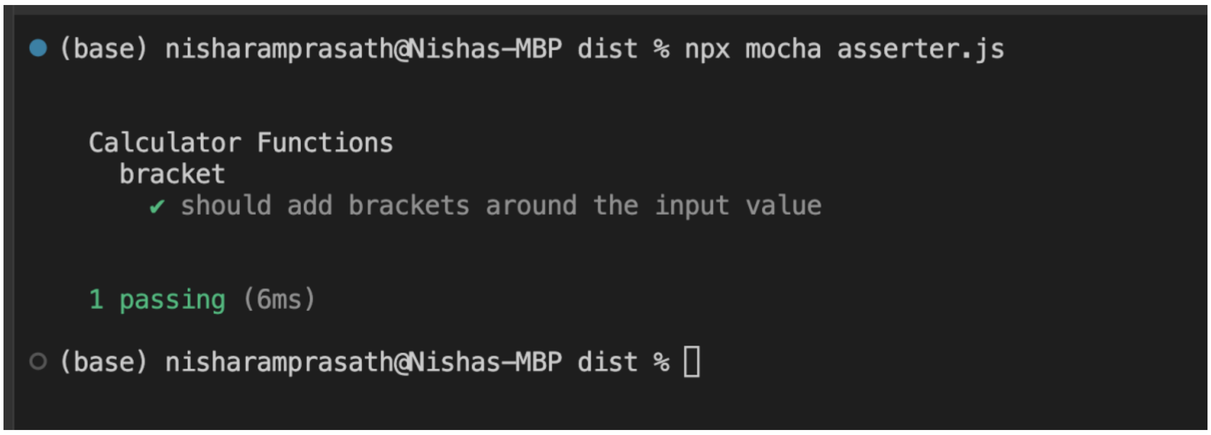
Test set 2 written explanation of strategy:

Test Set 1 targets the trigonometry functions in the calculator, the bracket, delete char Functions and compute Function. The objective is to verify the correctness and functionality of these trigonometric functions

Test set 3 test 1 coded, explained and run (bracket function):

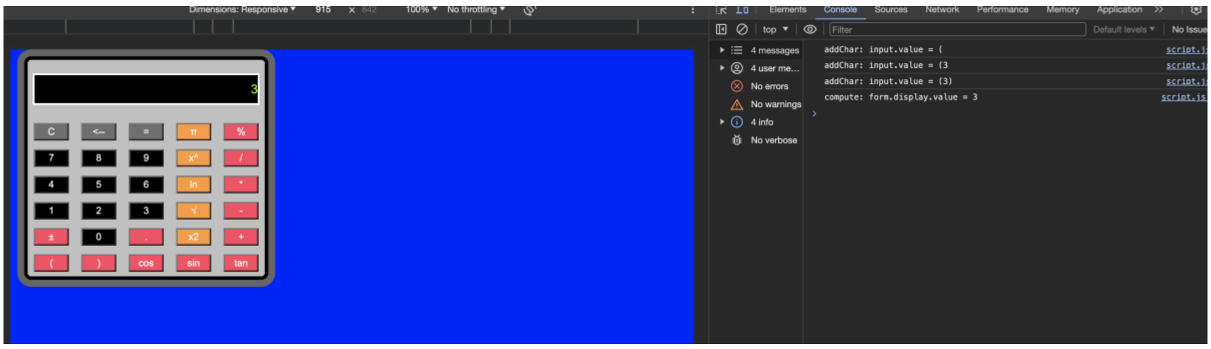


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  bracket: function (form) {
    // Implement the bracket function logic here
    // For example:
    form.display.value = "(" + form.display.value + ")";
  },
};

module.exports = calculator;

describe("Calculator Functions", function () {
  describe("bracket", function () {
    it("should add brackets around the input value", function () {
      var form = {
        display: {
          value: "3",
        },
      };

      calculator.bracket(form);

      assert.strictEqual(
        form.display.value,
        "(3)"
      );
    });
  });
});
```



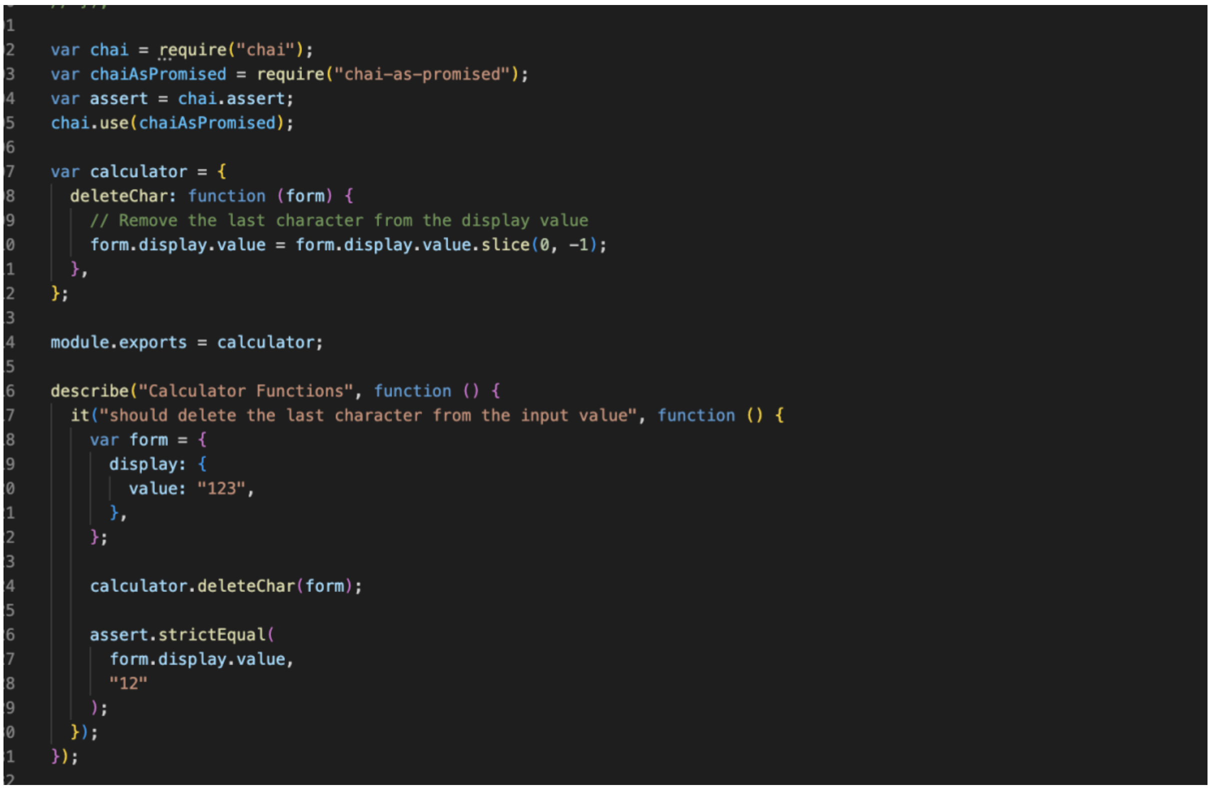
```
● (base) nisharamprasath@Nishas-MBP dist % npx mocha asserter.js


    Calculator Functions
      bracket
        ✔ should add brackets around the input value


    1 passing (6ms)

○ (base) nisharamprasath@Nishas-MBP dist % 
```

The test case checks the behaviour of the bracket function, which is from the script.js.

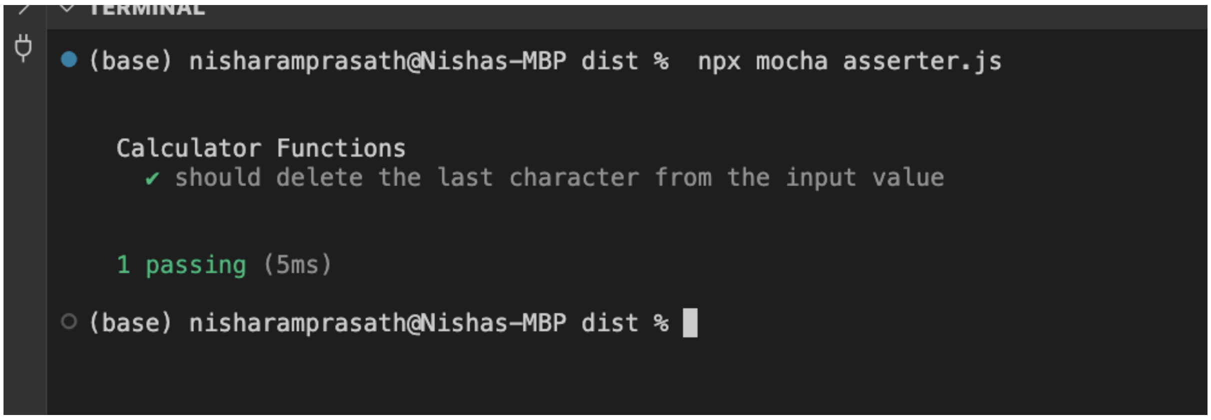


```
addChar: input.value = (                    script.js
addChar: input.value = (3                   script.js
addChar: input.value = (3)                  script.js
compute: form.display.value = 3             script.js
```

Test set 3 test 2 coded, explained and run (delete char function):

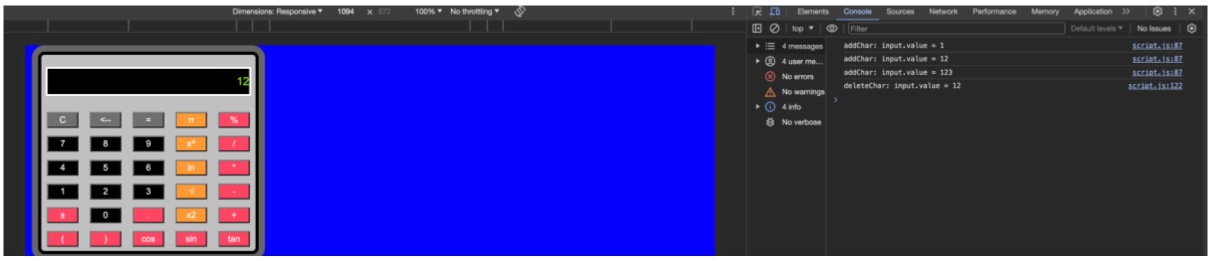


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  deleteChar: function (form) {
    // Remove the last character from the display value
    form.display.value = form.display.value.slice(0, -1);
  },
};

module.exports = calculator;

describe("Calculator Functions", function () {
  it("should delete the last character from the input value", function () {
    var form = {
      display: {
        value: "123",
      },
    };

    calculator.deleteChar(form);

    assert.strictEqual(
      form.display.value,
      "12"
    );
  });
});
```



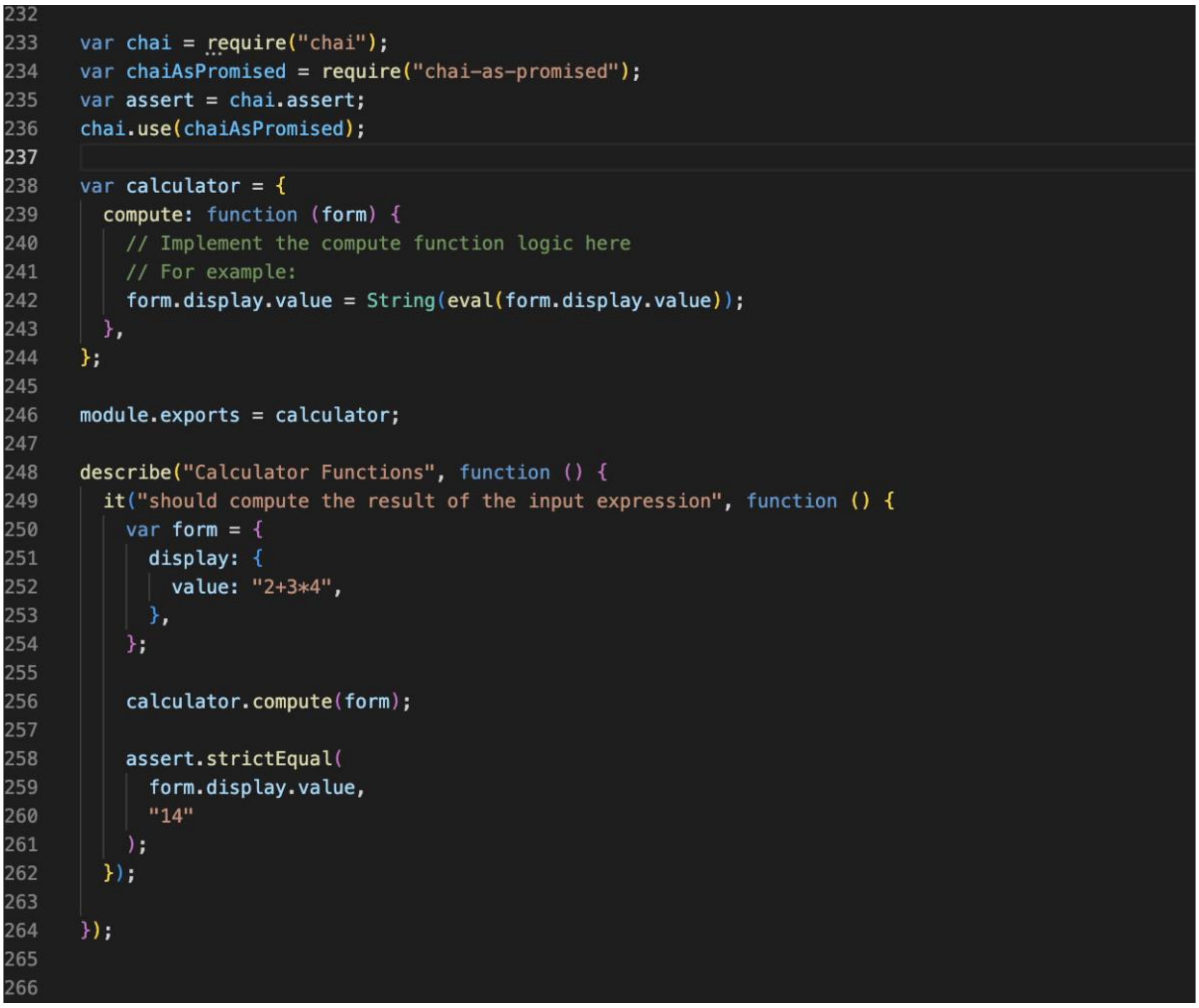
```
(base) nisharamprasath@Nishas-MBP dist %  npx mocha asserter.js


  Calculator Functions
    ✔ should delete the last character from the input value


  1 passing (5ms)

(base) nisharamprasath@Nishas-MBP dist %
```

The test case checks the behaviour of the delete function, which is from the script.js.

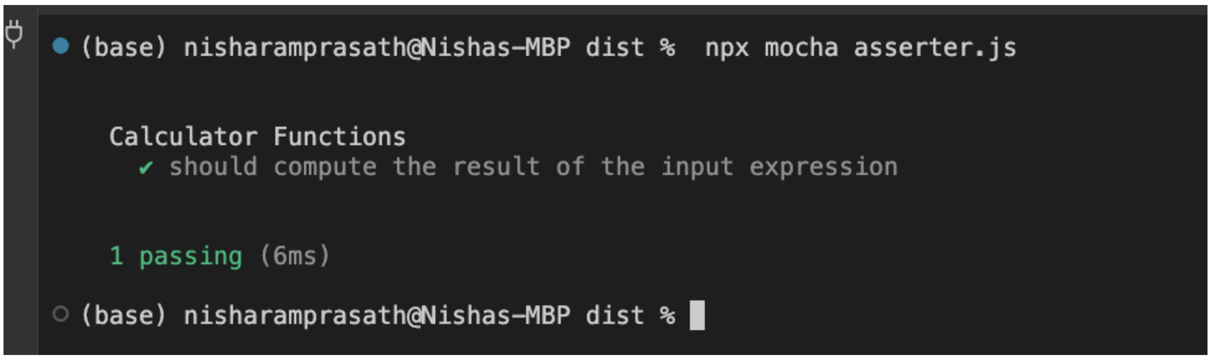


```
addChar: input.value = 1        script.js:87
addChar: input.value = 12       script.js:87
addChar: input.value = 123      script.js:87
deleteChar: input.value = 12    script.js:122
```

Test set 3 test 3 coded, explained and run (compute function):

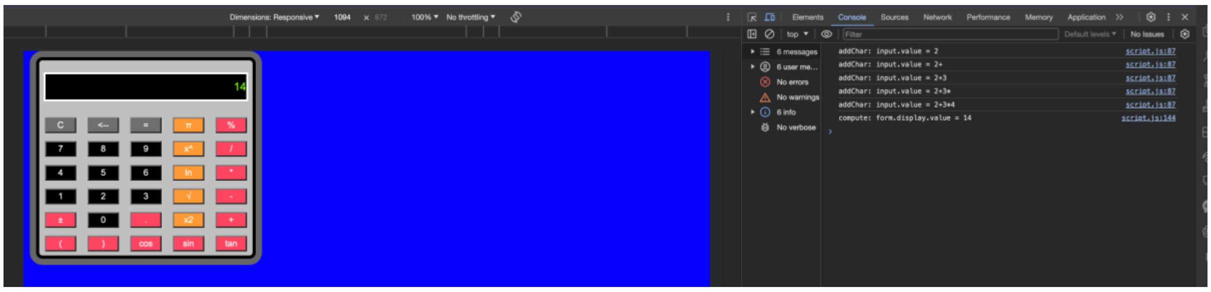


```
var chai = require("chai");
var chaiAsPromised = require("chai-as-promised");
var assert = chai.assert;
chai.use(chaiAsPromised);

var calculator = {
  compute: function (form) {
    // Implement the compute function logic here
    // For example:
    form.display.value = String(eval(form.display.value));
  },
};

module.exports = calculator;

describe("Calculator Functions", function () {
  it("should compute the result of the input expression", function () {
    var form = {
      display: {
        value: "2+3*4",
      },
    };

    calculator.compute(form);

    assert.strictEqual(
      form.display.value,
      "14"
    );
  });

});
```



```
● (base) nisharamprasath@Nishas-MBP dist %  npx mocha asserter.js


    Calculator Functions
      ✔ should compute the result of the input expression


    1 passing (6ms)

○ (base) nisharamprasath@Nishas-MBP dist % 
```

The test case checks the behaviour of the compute function, which is from the script.js.



```
addChar: input.value = 2                script.js:87
addChar: input.value = 2+               script.js:87
addChar: input.value = 2+3              script.js:87
addChar: input.value = 2+3*             script.js:87
addChar: input.value = 2+3*4            script.js:87
compute: form.display.value = 14        script.js:144
```