

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 :

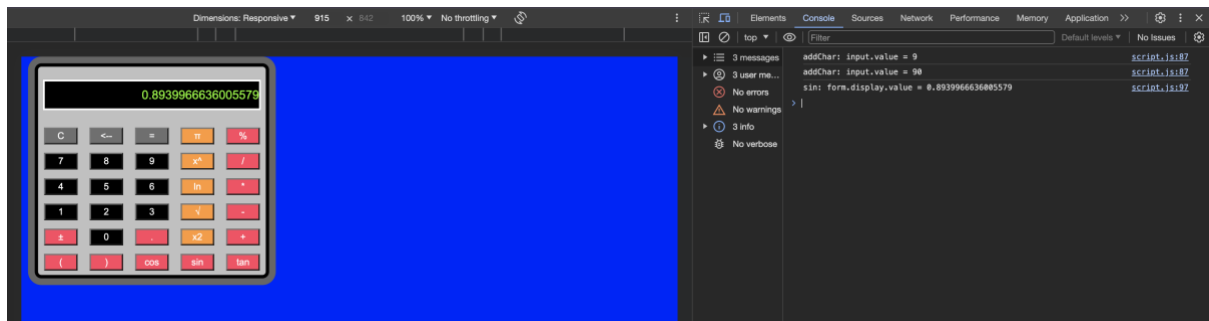
```
Click here to ask Blackbox to help you code faster
1  var chai = require("chai");
2  var chaiAsPromised = require("chai-as-promised");
3  var assert = chai.assert;
4  chai.use(chaiAsPromised);
5  // script.js
6  var calculator = {
7    sin: function (form) {
8      // Implementation of sin function
9      form.display.value = Math.sin(parseFloat(form.display.value));
10   },
11 };
12 module.exports = calculator;
13 describe("Calculator Functions", function () {
14   describe("sin", function () {
15     it("should compute sin of the input value", function () {
16       var form = {
17         display: {
18           value: "90",
19         },
20       };
21
22       calculator.sin(form);
23
24       // Use Chai's assert.approximately for the approximation
25       assert.approximately(
26         parseFloat(form.display.value),
27         Math.sin(90), 0.8775825618903728
28       );
29     });
30   });
31 });
32
```

```
15 DESCRIBE: Calculator Functions - FUNCTION 17.1
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR
● (base) nisharamprasath@Nishas-MBP dist % npx mocha assenter.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.



Test set 1 test 2 coded, explained and run :

```
1
2 var chai = require("chai");
3 var chaiAsPromised = require("chai-as-promised");
4 var assert = chai.assert;
5 chai.use(chaiAsPromised);
6 // script.js
7 var calculator = {
8   cos: function (form) {
9     // Implementation of sin function
10    form.display.value = Math.cos(parseFloat(form.display.value));
11  },
12 };
13 module.exports = calculator;
14 describe("Calculator Functions", function () {
15   describe("cos", function () {
16     it("should compute cos of the input value", function () {
17       var form = {
18         display: {
19           value: "0",
20         },
21       };
22       calculator.cos(form);
23
24       // Use Chai's assert.approximately for the approximation
25       assert.approximately(
26         parseFloat(form.display.value),
27         Math.cos(0), 1
28       );
29     });
30   });
31 });
```

PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

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

Calculator Functions

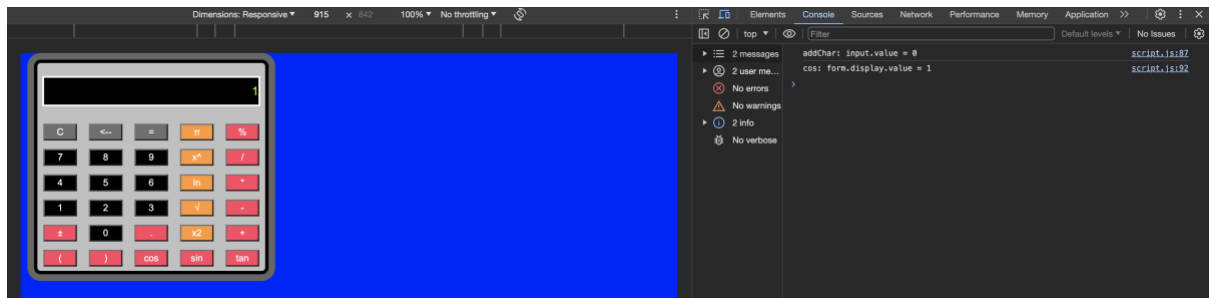
cos

✓ 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.



Test set 1 test 3 coded, explained and run :

```

1
2   var chai = require("chai");
3   var chaiAsPromised = require("chai-as-promised");
4   var assert = chai.assert;
5   chai.use(chaiAsPromised);
6   // script.js
7   var calculator = {
8     tan: function (form) {
9       // Implementation of sin function
10      form.display.value = Math.tan(parseFloat(form.display.value));
11    },
12  };
13  module.exports = calculator;
14  describe("Calculator Functions", function () {
15    describe("tan", function () {
16      it("should compute tan of the input value", function () {
17        var form = {
18          display: {
19            value: "45",
20          },
21        };
22
23        calculator.tan(form);
24
25        // Use Chai's assert.approximately for the approximation
26        assert.approximately(
27          parseFloat(form.display.value),
28          Math.tan(45), 1.6197751905438615
29        );
30      });
31    });
32  });
33  });

```

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  SEARCH ERROR

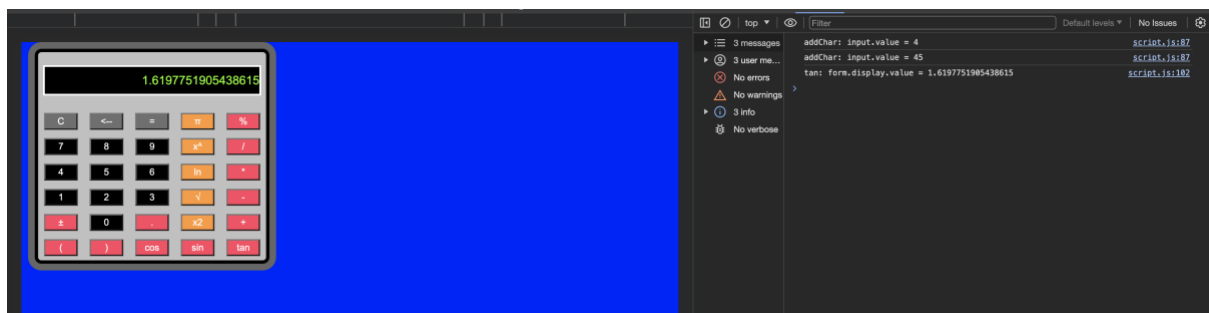
● (base) nisharamprasath@Nishas-MBP dist % npx mocha assenter.js

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.



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):

```

35
36 var chai = require("chai");
37 var chaiAsPromised = require("chai-as-promised");
38 var assert = chai.assert;
39 chai.use(chaiAsPromised);
40
41 var calculator = {
42   sqrt: function (form) {
43     form.display.value = Math.sqrt(parseFloat(form.display.value));
44   },
45 };
46
47
48 module.exports = calculator;
49
50 describe("Calculator Functions", function () {
51   describe("sqrt", function () {
52     it("should compute the square root of the input value", function () {
53       var form = {
54         display: {
55           value: "10",
56         },
57       };
58
59       calculator.sqrt(form);
60
61       assert.approximately(
62         parseFloat(form.display.value),
63         Math.sqrt(10), 3.1622776601683795
64       );
65     });
66   });
67 });

```

PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR

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

Calculator Functions

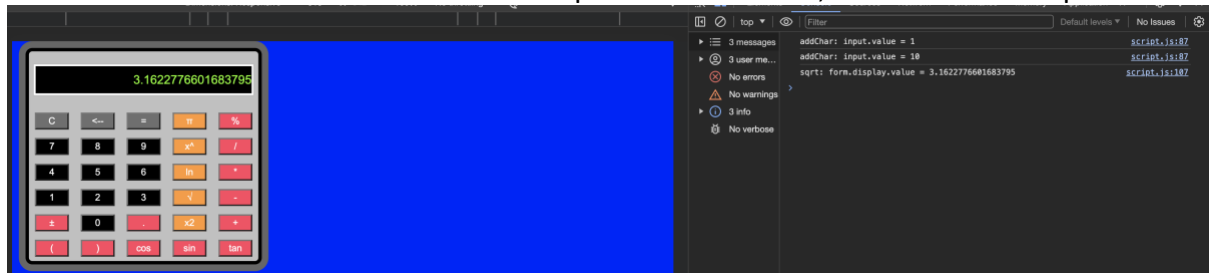
sqrt

✓ 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.



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

```
69
70 var chai = require("chai");
71 var chaiAsPromised = require("chai-as-promised");
72 var assert = chai.assert;
73 chai.use(chaiAsPromised);
74
75 var calculator = {
76   square: function (form) {
77     form.display.value = Math.pow(parseFloat(form.display.value), 2);
78   },
79 };
80
81 module.exports = calculator;
82
83 describe("Calculator Functions", function () {
84   describe("square", function () {
85     it("should compute the square of the input value", function () {
86       var form = {
87         display: {
88           value: "6",
89         },
90       };
91
92       calculator.square(form);
93
94       assert.strictEqual(
95         parseFloat(form.display.value),
96         Math.pow(6, 2)
97       );
98     });
99   });
100 });
```

✓ TERMINAL

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

Calculator Functions

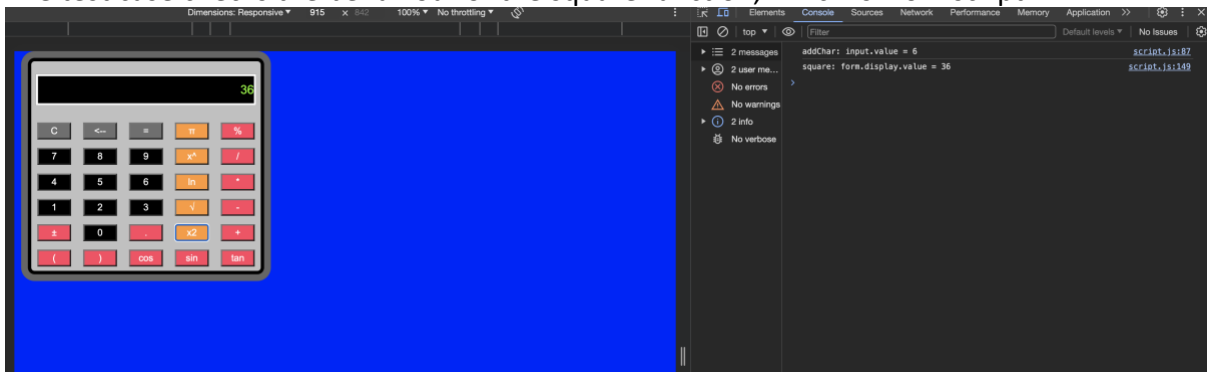
square

✓ 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.



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


```

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 assenter.js

Calculator Functions

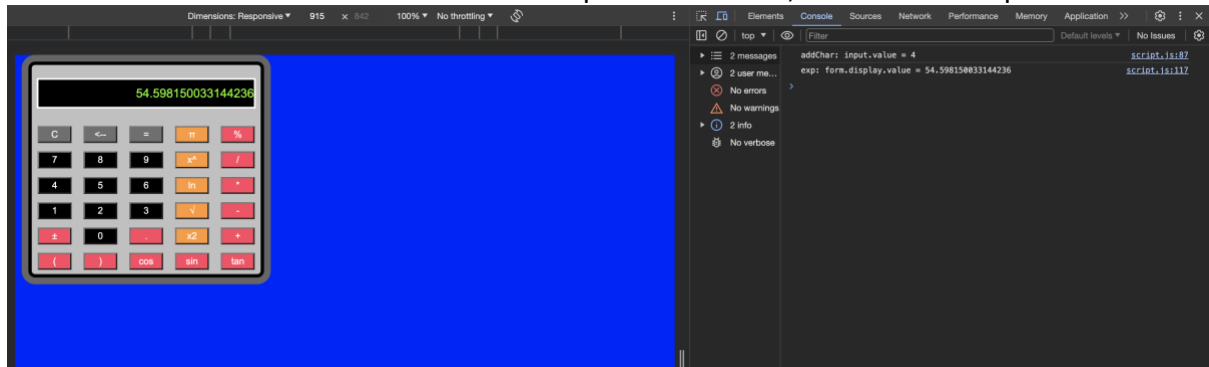
square

✓ 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.



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)

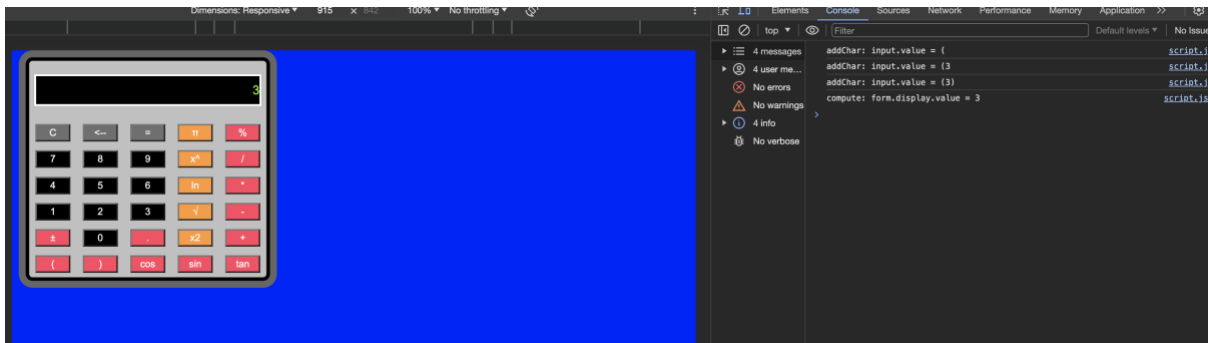
```

```

○ (base) nisharamprasath@Nishas-MBP dist % 

```

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



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

```
1
2 var chai = require("chai");
3 var chaiAsPromised = require("chai-as-promised");
4 var assert = chai.assert;
5 chai.use(chaiAsPromised);
6
7 var calculator = {
8   deleteChar: function (form) {
9     // Remove the last character from the display value
10    form.display.value = form.display.value.slice(0, -1);
11  },
12 };
13
14 module.exports = calculator;
15
16 describe("Calculator Functions", function () {
17   it("should delete the last character from the input value", function () {
18     var form = {
19       display: {
20         value: "123",
21       },
22     };
23
24     calculator.deleteChar(form);
25
26     assert.strictEqual(
27       form.display.value,
28       "12"
29     );
30   });
31 });
```

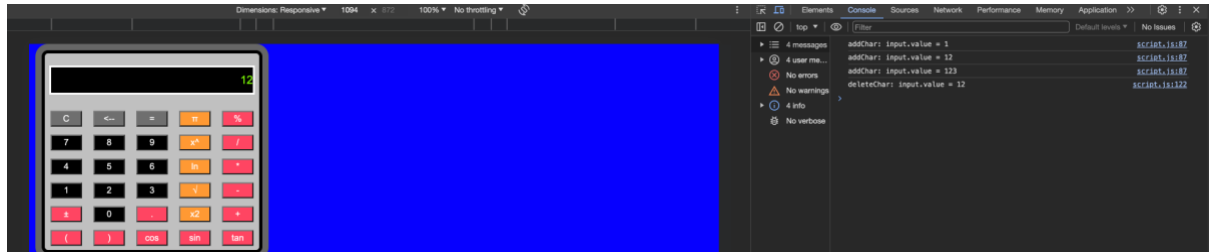
```
● (base) nisharamprasath@Nishas-MBP dist % npx mocha assenter.js

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.



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

```
232
233 var chai = require("chai");
234 var chaiAsPromised = require("chai-as-promised");
235 var assert = chai.assert;
236 chai.use(chaiAsPromised);
237
238 var calculator = {
239   compute: function (form) {
240     // Implement the compute function logic here
241     // For example:
242     form.display.value = String(eval(form.display.value));
243   },
244 };
245
246 module.exports = calculator;
247
248 describe("Calculator Functions", function () {
249   it("should compute the result of the input expression", function () {
250     var form = {
251       display: {
252         value: "2+3*4",
253       },
254     };
255
256     calculator.compute(form);
257
258     assert.strictEqual(
259       form.display.value,
260       "14"
261     );
262   });
263
264 });
265
266
```

```
• (base) nisharamprasath@Nishas-MBP dist % npx mocha assenter.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.

