**Jest:**

* In VS Code, open Demos>JavaScript>testing
* Open the terminal in VS Code and ensure you are in the testing directory
* Run npm init, leave all default settings (by pressing enter) except for “test command” set that to “jest” and continue hitting enter until it is no longer requesting input
  + Setting the test command to jest allows us to skip the step manually editing the package.json file later
* Run ‘npm install --save-dev jest’
* Walk students through the functions in the calculator.js file
* Create a new file called calculator.test.js – this file will store our jest tests
* Export all of the functions in calculator.js
  + module.exports = {add, subtract, multiply, divide};
* Import them into calculator.test.js
  + const { add, subtract, multiply, divide } = require('./calculator’);
* Write a test for the add function that takes 1 and 2 and expects 3, ex:
  + test('should return 3 given 1 and 2', () => {

expect(add(1, 2)).toBe(3);

})

* In the terminal, run ‘npm run test’ and ensure the test passes
* Set toBe() to the wrong answer and run the tests again so students know what a failing test looks like
* Write tests for the rest of the functions using:
  + toBeLessThan
  + .not
  + toEqual

**Async:**

* Open Demos>JavaScript>async>async.js in VS Code
* Walk students through the function
  + Let students know they’ll learn more about the async, await, and promise keywords soon
  + f() is an asynchronous function that promises to return the string “done!” after 3 seconds
  + Then stores this promise in the variable result and then alerts result
  + Then we call the function at the end of the file
* Open async.html in a live server and wait 3 seconds for an alert to appear that says “done”
* Remove the word await on line 7 of async.js
* Reload the web page
  + An alert should immediately appear with a promise object
  + This occurs because we are no longer awaiting the result of the promise, we are immediately returning it
* Console.log the promise object and explore it
  + It is an object with a state of pending
  + It has multiple functions that can be called on it (catch, finally, then)
* Undo the change to async.js so the await keyword is there again
* Open the terminal in VSC and ensure you’re in the async directory
* Initialize npm and install jest
* Create a file called async.test.js
* Export f() from async.js and import it into the test file
* In async.js make f() return the result so it can be tested
* In async.test.js create a test to check that f() returns “done!” ex:
  + test('the promise returns done!', () => {

return f().then(result => {

expect(result).toBe('done!');

})

})

* Once the test passes, change .toBe to .toContain and remove the exclamation mark