**Mocha:**

* [Mocha](https://mochajs.org/) is a flexible library providing developers with just the base test structure.
* Mocha includes the test structure as globals, so save time.
* Mocha is a mature and powerful testing framework for Node.js.
* To install it, simply run:

$ npm install –g [mocha@1.16.2](mailto:mocha@1.16.2)

* Mocha is a testing library for Node.js, created to be a simple, extensible, and fast testing suite.
* It's used for unit and integration testing, and it's a great candidate for BDD (Behavior Driven Development).we use Both TDD and BDD the Mocha testing framework.
* Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun.
* Mocha can be used with most JavaScript assertion libraries, including:
  + - * should.js
      * express.js
      * chai
      * better-assert.

**Chai:**

* Chai is a subset of that library(mocha library). To install command like,

$ npm install [chai@1.8.1](mailto:chai@1.8.1)

* Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.
* Chai is an assertion library, similar to Node's build in assert. It makes testing much easier by giving you lots of assertions you can run against your code.
* The assert style is very similar to node.js’ included assert module, with little changes.
* Assertion Styles are:

The [Expect / Should](http://www.chaijs.com/api/bdd/) API covers the BDD assertion styles.

The Assert API covers the TDD assertion style.

Example for testing:

file name = test.js :

const assert = require(‘chai’).assert;

const sayhello = require(‘../app’).sayhello;

describe(‘app’,function(){

it(‘app should return hello’,function(){

let result = sayhello();

assert.equal(result, ‘hello’);

});

});

file name = app.js :

module.exports = {

sayhello : function(){

return ‘hello’;

}

}

here we test code using test libraries. Those are all used to avoid bugs and improve the quality of code.

**Istanbul (code coverage):**

1 Code coverage is a term to describe which application code is exercised when the application is running. Test coverage refers to metrics in an overall test-plan.

2 It is a great tool for adding coverage reporting to your continuous-

integration flow.

3 [Istanbul](https://github.com/gotwarlost/istanbul) is a code coverage tool for JavaScript.

4 To run Istanbul with [Mocha](https://github.com/visionmedia/mocha) (a popular test framework), you can run this command: $ npm install -g istanbul.

Or

declare in json file also. That is,

**{**

**"scripts":{**

**"test": "nyc mocha"**

**}**

**}**

The Istanbul tool is used to find the code coverage. It will show the coverage through table formate.