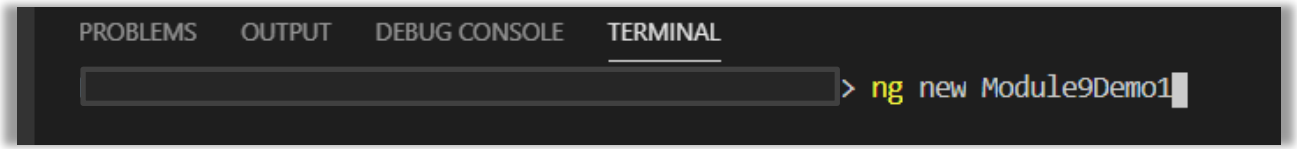
Module 11: Testing and Application Deployment in Angular

Demo Document 1: Unit testing using Jasmin and Karma

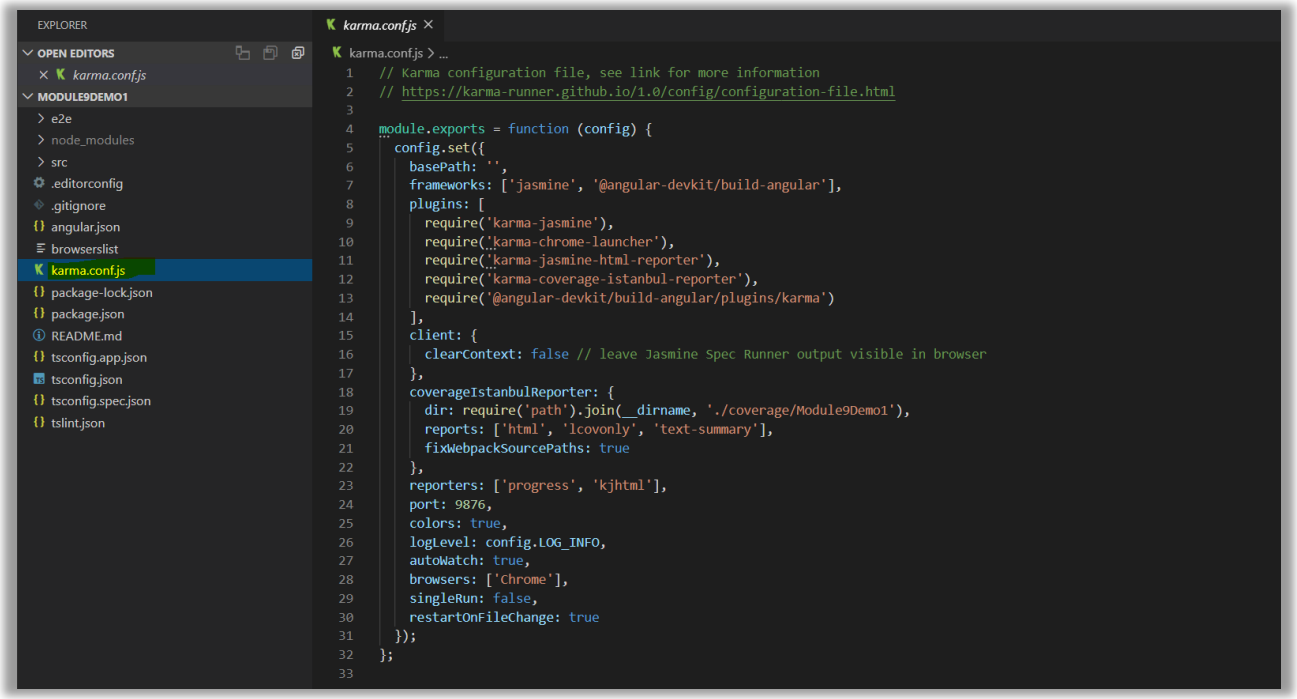
Creating unit testing steps for angular application using karma & jasmine

In this demo, we will see how to create and execute Unit testing of angular app with the help of karma & jasmine.

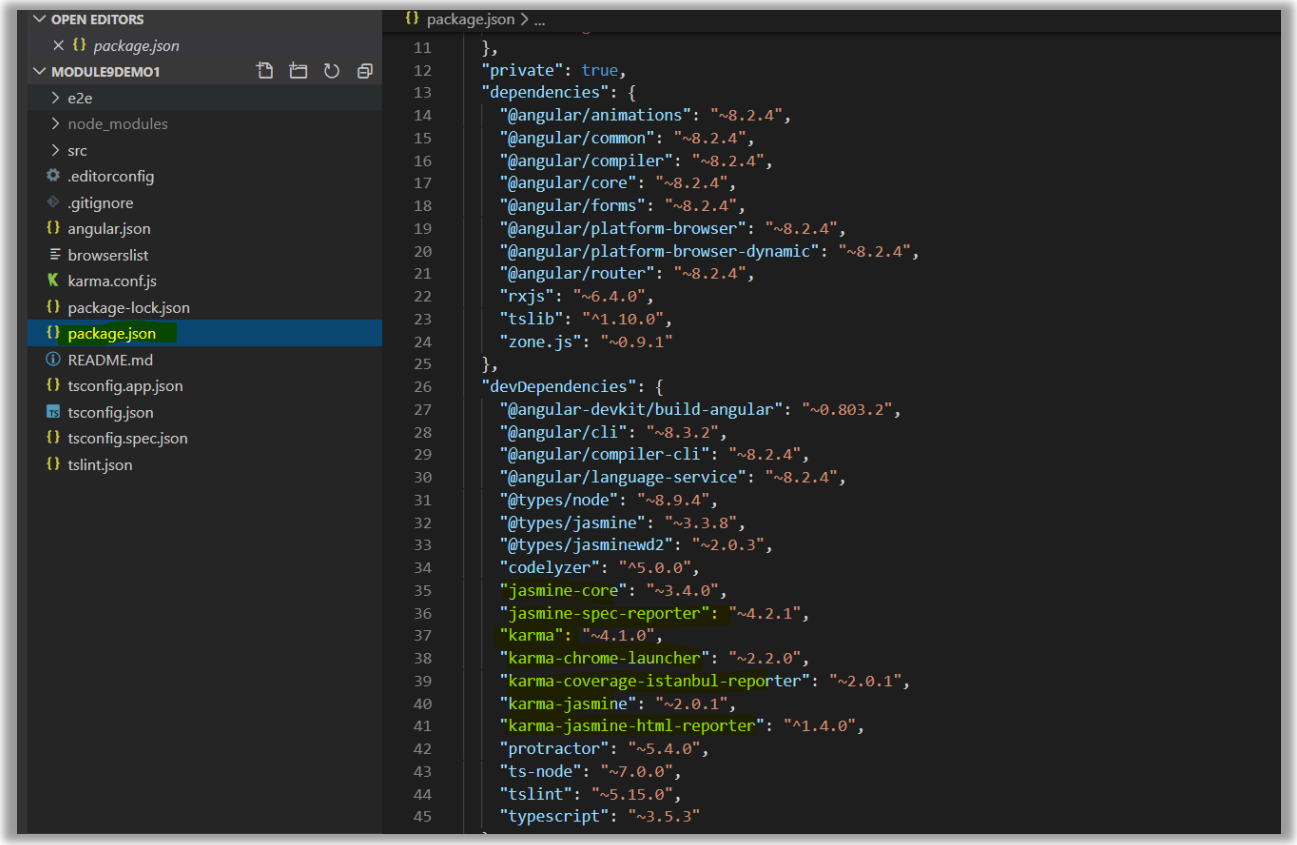
Step 1: Create new project using command 'ng new Module11Demo1' .



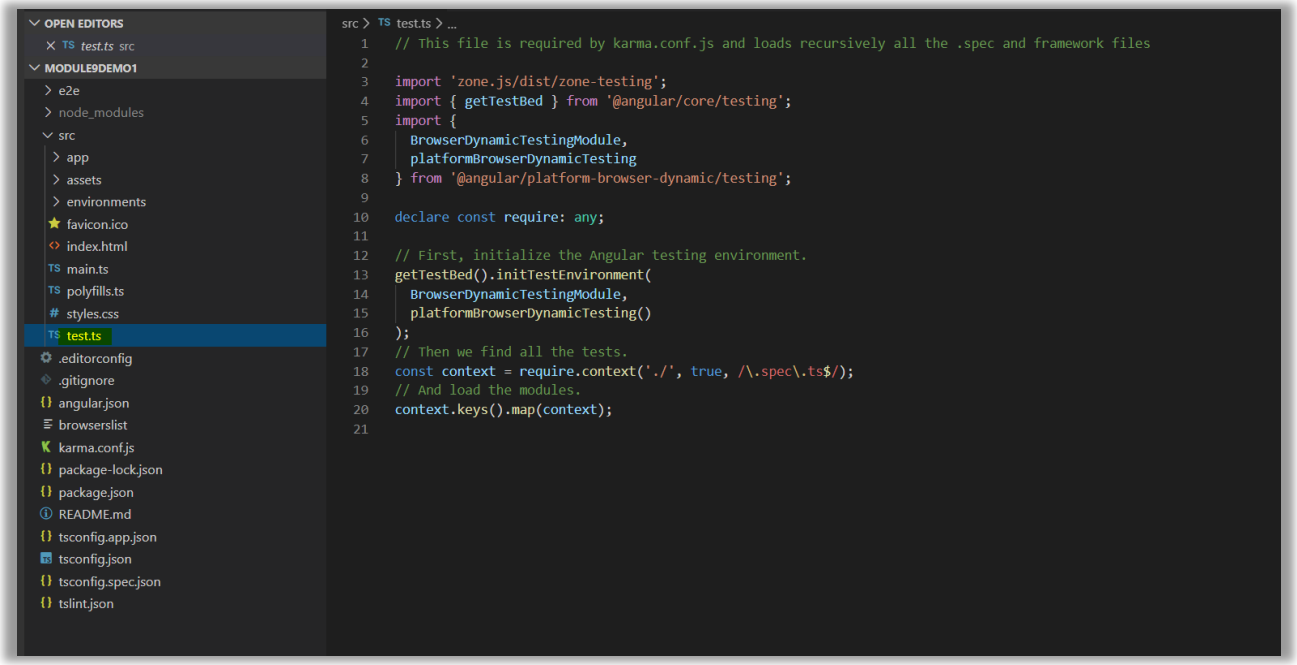
Step 2: When we create new app in angular, angular cli will automatically create karma.config.js file at root location



Step 3- When you create the project all the dependencies get installed. Open package.json file



Step 4 – The angular-cli configuration of karma uses the file "test.ts" as the entry point to test the application. Open test.ts file



Step 5- Create first test, open app.component.html and remove all html code except router-outlet as below

app.component.html

```
<router-outlet></router-outlet>
```

Step 6- Open inde.html and add bootstrap reference

app.component.html | index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Module9Demo1</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Step 7- Ppen app.component.html and add tile and styles

app.component.html | app.component.css | index.html

```
<style>
  .content {
    display: flex;
    margin: 32px auto;
    padding: 0 16px;
    max-width: 960px;
    flex-direction: column;
    align-items: center;
  }
</style>

<div class="content" role="main">

  <div class="card highlight-card card-small">
    <span>{{ title }} app is running!</span>
  </div>



<router-outlet></router-outlet>
```

Step 8- Open app.component.spec.ts (where we will write test cases). It has already written some test cases as below.

app.component.html | app.component.spec.ts | index.html

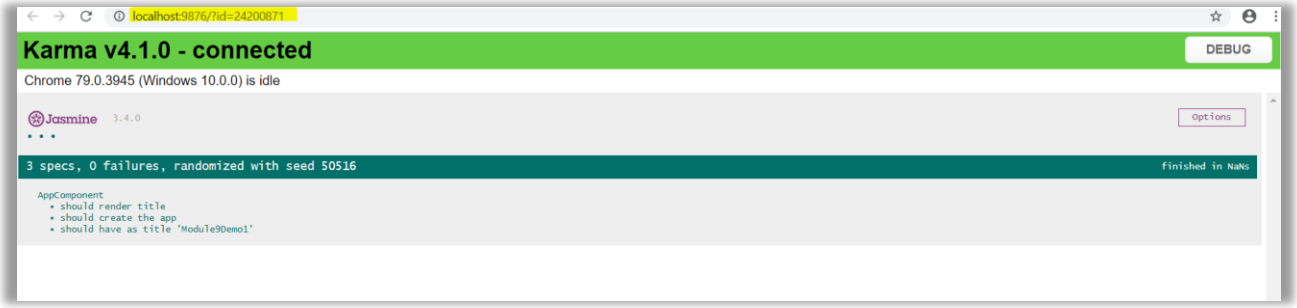
src > app > app.component.spec.ts > describe('AppComponent') callback > it(`should have as title 'Module9Demo1'`) callback

```
import { TestBed, async } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'Module9Demo1'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('Module9Demo1');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('.content span').textContent).toContain('Module9Demo1 app is running!');
  });
});
```

Step 9- As we have already written test case for title, run test using command ‘ng test’ as below.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Work\TapChief\Edureka\Angular8Demo\Module9\Module9Demo1> ng test
```

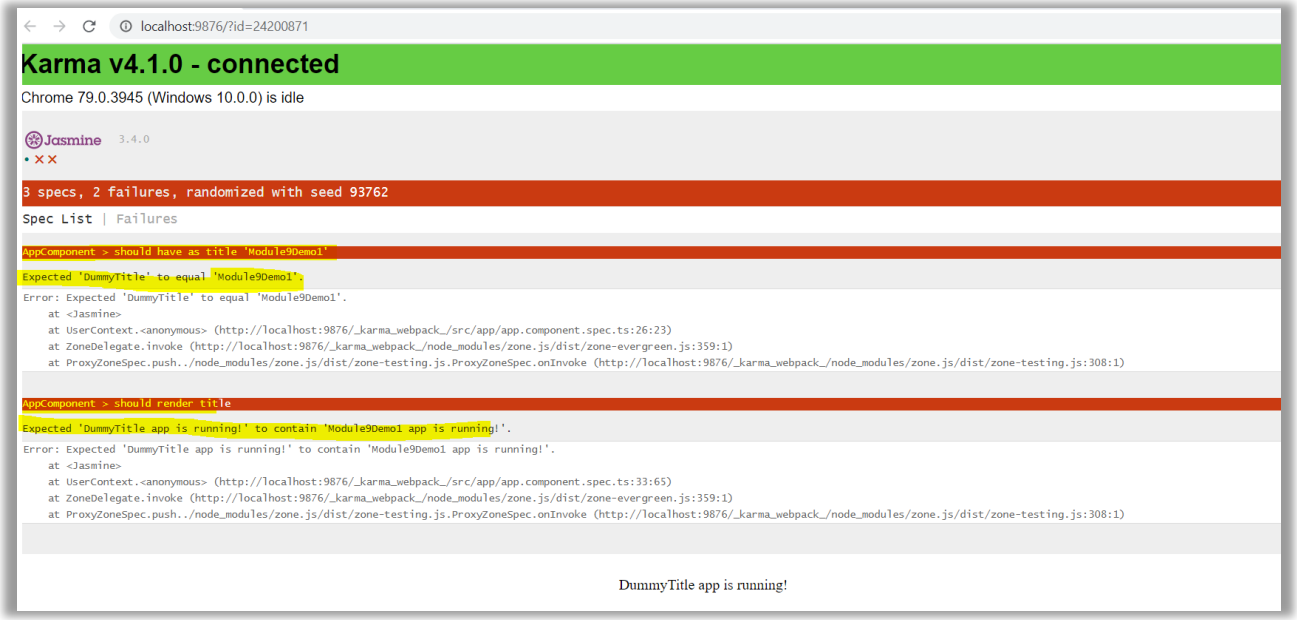
Chrome will open below window and show how many test cases passed.



Step 10 – Open app.component.ts file and change title of app from ‘Module9Demo1’ to something else as below.

Step 11 – Save file and run command ng test again.

2 specs will fail as we have modified title.



Similarly, we can write specs (test cases) for other components as well