



# Angular Unit Testing

▼ Status	writing
🔗 Link	<a href="https://www.youtube.com/watch?v=HBaid2cPT98&amp;list=PL1ano0qwNuBxyiYXCmO_OjaPwc-GV-L9O&amp;index=1">https://www.youtube.com/watch?v=HBaid2cPT98&amp;list=PL1ano0qwNuBxyiYXCmO_OjaPwc-GV-L9O&amp;index=1</a>
☰ Summary	This is a tutorial documentation for angular unit testing
👤 Author	Ⓐ Ahmed Hrabı
▼ Type	Tutorial documentation

Unit testing is a software testing technique that allows individual units of code to be tested in isolation from the rest of the application. It is a code that test a code.

## Why unit testing:

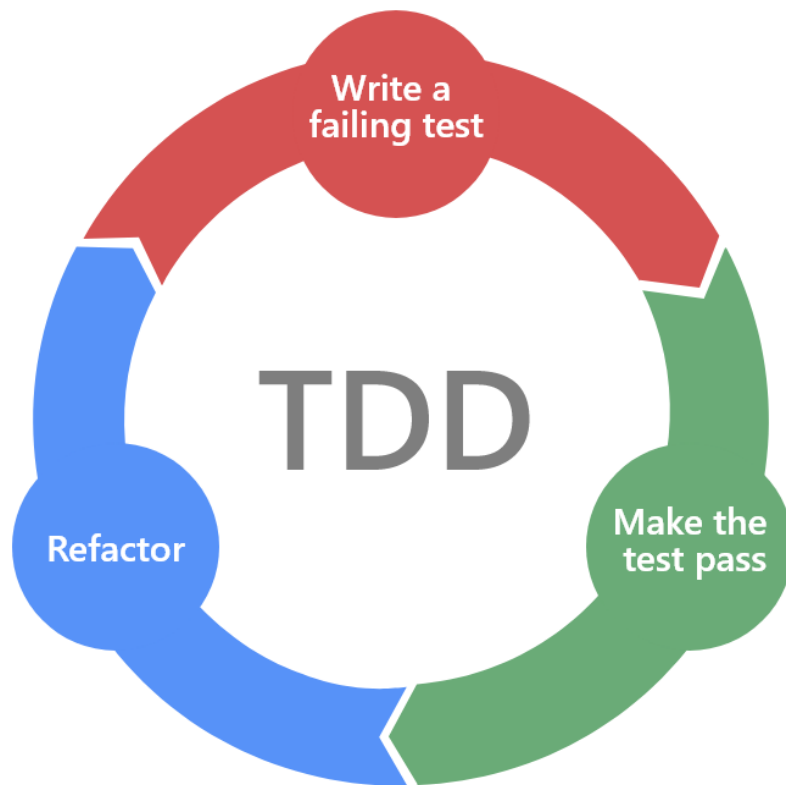
- To deploy the application with confidence, ensure that every part of the application is working as intended.
- Enable developers to confidently apply refactoring while avoiding undesired code behavior.
- Unit tests can detect early flaws in cod

## Test Driven Development (TDD):

Is a development process that emphasizes writing automated tests before writing the code. This ensures that the code is testable and that all requirements are met. The process involves three main steps:

1. **Red:** Write a test that fails. This test should capture the intended behavior of the code.
2. **Green:** Write the simplest code that passes the test.
3. **Refactor:** Refactor the code to improve its design while ensuring that all tests pass.

Here is a flowchart that represents the TDD process:



## Behavior Driven Development (BDD):

Is a software development methodology that emphasizes collaboration among developers, testers, and business stakeholders to define, implement, and verify system

behavior based on real-world **scenarios** expressed as understandable specifications and tests.

! BDD is a part of TDD.

! The way writing test cases does not require a technical knowledge in order to understand it.

## ▼ Simple examples:

### Hello world example

In AppComponent.ts :

```
hello(){  
  return 'hello World!'  
}
```

In AppComponent.spec.ts

```
describe('clicking in hellow funtion',()=>{ // (1) describe of the scenario  
  it('must return Hello World!',()=>{ // (2) describe the behavior  
    const com = new AppComponent();  
    expect(com.hello()) // (3) the methode to be tested  
      .toBe('Hello World!'); // (3) expect the return of the method  
  })  
})
```

Run the command :

```
ng test  
// or  
ng test --code-coverage
```

### Testing a service example:

[auth.service:](#)

```
isAuth():boolean {  
  return !!localStorage.getItem('token');  
}
```

auth.service.spec.ts:

```
describe('check returning value for isAuthenticated', () => {
  let auth : AuthService;
  beforeEach(() => {
    auth = new AuthService(); //a new instance before each test
  });
  afterEach(() => {
    localStorage.removeItem('token'); // remove token after each test
  })
  it('must return true if there is a token in local storage ', () => {
    localStorage.setItem('token', 'mytoken');
    expect(auth.isAuthenticated()).toBeTruthy(); //expect to return true
  });

  it('must return false if there is no token in local storage ', () => {
    expect(auth.isAuthenticated()).toBeFalsy(); //expect to return false
  })
})
```

## Angular Test Bed (ATB):

Angular component is a combination of HTML template and typescript class. An adequate component test should test both template and typescript work together as intended.

The class-only tests can tell you about only : the class behavior.

they cannot

- Tell you If the component is going to render properly
- Respond to user input and gesture
- integrate with parent or child component.

ATB is a higher level *Angular Only* testing framework that allows us to easily test behaviours that depend on the Angular Framework.

When to use ATB :

We use ATB because:

- It allows us to test the interaction of a directive or component with its template.
- It allows us to easily test change detection.

- It allows us to test and use Angular's DI framework.
- It allows us to test using the `NgModule` configuration we use in our application.
- It allows us to test user interaction via clicks and input fields

## ▼ Simple example:

`AppComponent.ts:`

```
export class AppComponent {
  title = 'ngUnitTest';
  constructor(private authService:AuthService){
  }
  canLogin(username:string,password:number):boolean{
    return this.authService.isAuthenticated(username,password);
  }
}
```

`AuthService.service.ts:`

```
isAuthenticated(username:string,password:number):boolean{
  if(username&&password){
    return true;
  }else{
    return false;
  }
}
```

`App.component.spec.ts:`

```
//import TestBed
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';
import { AuthService } from './services/auth.service';

// TestBed
describe('App component',()=>{
  let component: AppComponent;
  //a fixture is a wrapper (emballage) for a component and its template
  let fixture:ComponentFixture<AppComponent>;
  let authService: AuthService;
  beforeEach(function(){
    TestBed.configureTestingModule({

    })
    // create component and test fixture
  })
}
```

```

    fixture = TestBed.createComponent(AppComponent);
    // get test component from the fixture
    component = fixture.componentInstance;
    // the service provided to the TestBed
    authService = TestBed.inject(AuthService);
  });
  it('should should create app component',()=>{
    expect(component).toBeTruthy();
  });
  it('should have title ng unit testing',()=>{
    expect(component.title).toEqual('ngUnitTest')
  })
  it('can login',()=>{
    expect(component.canLogin('my uuser',123)).toBeTruthy();
    expect(component.canLogin('',123)).toBeFalsy();
  })
})

```