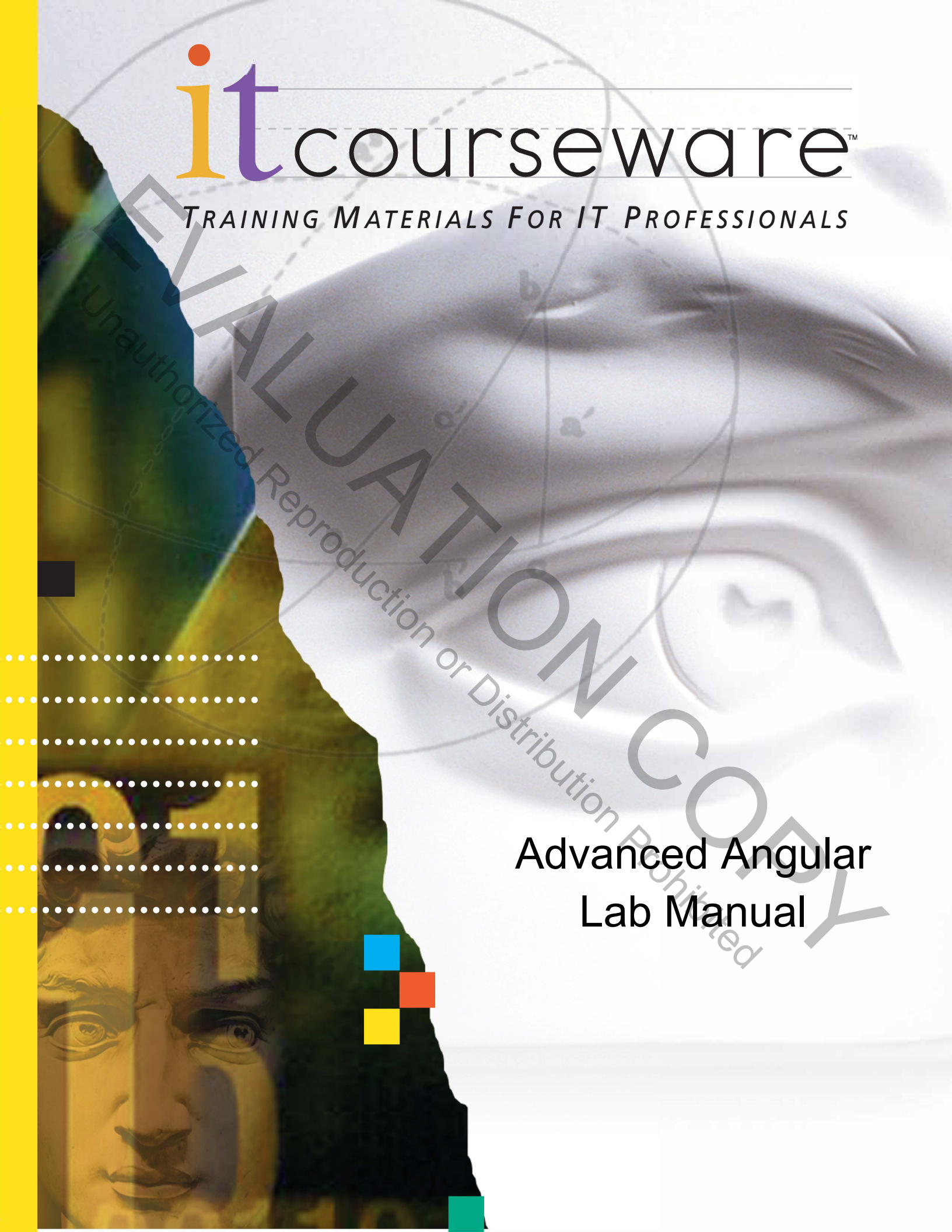

itcourseware™
TRAINING MATERIALS FOR IT PROFESSIONALS
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited
Advanced Angular
Lab Manual

Angular

Lab Manual

About this Lab Manual

This lab manual provides a series of hands-on exercises for learning how to build web applications using Angular.

Conventions

Each hands-on exercise in this manual will consist of a series of steps to accomplish a learning objective.

Code Blocks

- All paths in the are relative to the **project-manage** directory.

  So the file below will be found at:

  AngularCourse\code\labs\working\project-manage\app.module.ts

- Highlighted code indicates code that has changed. If the code is not highlighted it should already exist from a previous step.
- Code with a ~~Strikethrough~~ should be removed.
- `...` Indicates code has been omitted for formatting and clarity but you should leave these sections of code in your running application.
- Most code snippets are short and easy to type but some are longer so a file with the contents of the code to add is provided in the folder.

  AngularCourse\code\labs\snippets

  - If a code snippets is provided for a code block the file path will appear below the code block as show below.

app.module.ts

```
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { BrowserModule } from '@angular/platform-browser';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent],
})
```

snippets\lab00-step00.html

Commands

These commands should be run in a command-prompt (Windows) or terminal (Mac).

```
ng -v
```

Sidebars

The boxes are sidebars and should be read.

> The boxes with blue borders are information and tips.

> The boxes with red borders are alerts.

Completion

At the end of each lab you will see:

✓ You have completed Lab …

Lab 30: Search

Objectives

- [ ] Add the ability to search for projects

Steps

Add the ability to search for projects

1. **Add** a **listByName** method to the **ProjectService**.

src\app\projects\shared\project.service.ts

```
...
export class ProjectService {
...
  list(): Observable<Project[]> {...}

  listByName(name: string): Observable<Project[]> {
    if (!name.trim()) {
      return this.list(); // if no name was provided, list all
    }
    const url = `${this.projectsUrl}?name_like=${name}`;
    return this.http.get<Project[]>(url).pipe(
      catchError((error: HttpErrorResponse) => {
        console.error(error);
        return throwError('An error occurred searching the projects.');
      })
    );
  }
...
```

snippets\lab30-step01.txt

2. Add an **onSearch** method and a **search** method. Invoke **search** in **ngOnInit**.

src\app\projects\projects-container\projects-container.component.ts

```
export class ProjectsContainerComponent implements OnInit {
 ...
  ngOnInit() {
    this.loading = true;
    this.projectService.list().subscribe(
      ...
    );
    this.search('');
  }

  onSearch(term: string) {
    this.search(term);
  }

  search(term: string) {
    this.loading = true;
    this.projectService.listByName(term).subscribe(
      data => {
        this.projects = data;
      },
      error => {
        this.loading = false;
        this.errorMessage = error;
      },
      () => (this.loading = false)
    );
  }
 ...
}
```

snippets\lab30-step02.txt

3. **Add** a **search input** to the template and **call onSearch** on the **keyup** event.

src\app\projects\projects-container\projects-container.component.html

```
<h1>Projects</h1>
<div class="row">
  <div class="col-sm-12">
    <div class="input-group fluid">
      <input #searchBox type="text" name="searchBox"
        placeholder="Search" (keyup)="onSearch(searchBox.value)">
    </div>
  </div>
</div>
<div class="row">
```

snippets\lab30-step03.txt

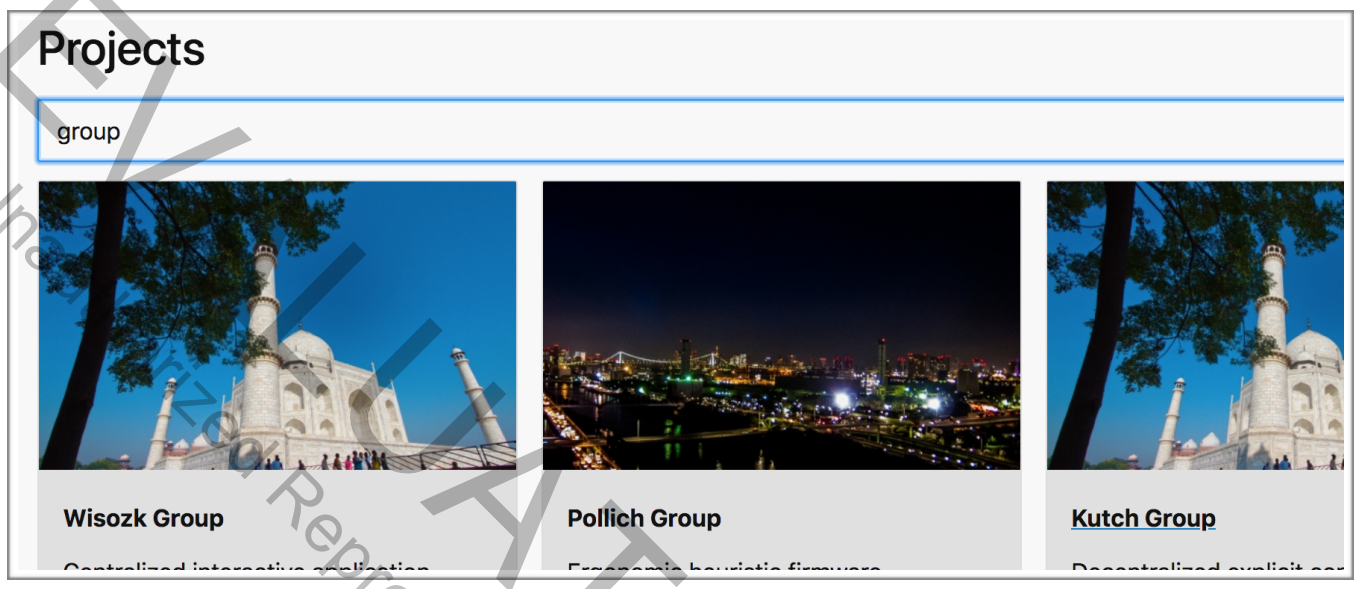
4. Verify

   a. **Save** your **code** changes.

   b. **Click** on **Projects** in the navigation if you aren't at that route already.

   c. **Type "group"** in the search input.

d. The **projects** should be **filtered** to ones with **"group"** in their name.



Notice that the screen flashes with your every keystroke resulting in a poor user experience. We will fix this in the next lab using RxJS and Observables.

✓ You have completed Lab 30

Lab 31: Search using RxJS

Objectives

- [ ] Improve the user experience when searching

Steps

Steps begin on the next page.

1. **Refactor** ProjectsContainerComponent to use an observable Subject.

src\app\projects\projects-container\projects-container.component.ts

```
...
import { Subject, Observable, Subscription } from 'rxjs';
import { debounceTime, distinctUntilChanged, switchMap } from 'rxjs/operators';

export class ProjectsContainerComponent implements OnInit, OnDestroy {
  projects: Project[];
  errorMessage: string;
  loading: boolean;
  private searchTerms = new Subject<string>();
  private subscription: Subscription;

  constructor(private projectService: ProjectService) {}

  ngOnInit() {
    this.observeSearchTerms();
    this.searchTerms.next('');
  }

  onSearch(term: string) {
    this.searchTerms.next(term);
  }

  observeSearchTerms() {
    this.subscription = this.searchTerms
      .pipe(
        // wait 300ms after each keystroke before considering the term
        debounceTime(300),

        // ignore new term if same as previous term
        distinctUntilChanged(),

        // switch to new search observable each time the term changes
        switchMap(
          (term: string): Observable<Project[]> => {
            this.loading = true;
            return this.projectService.listByName(term);
          }
        )
      )
      .subscribe(
        data => {
          this.loading = false;
          this.projects = data;
        },
        error => {
          this.loading = false;
          this.errorMessage = error;
        }
      );
  }

  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }
...
}
```

snippets\lab31-step01.txt

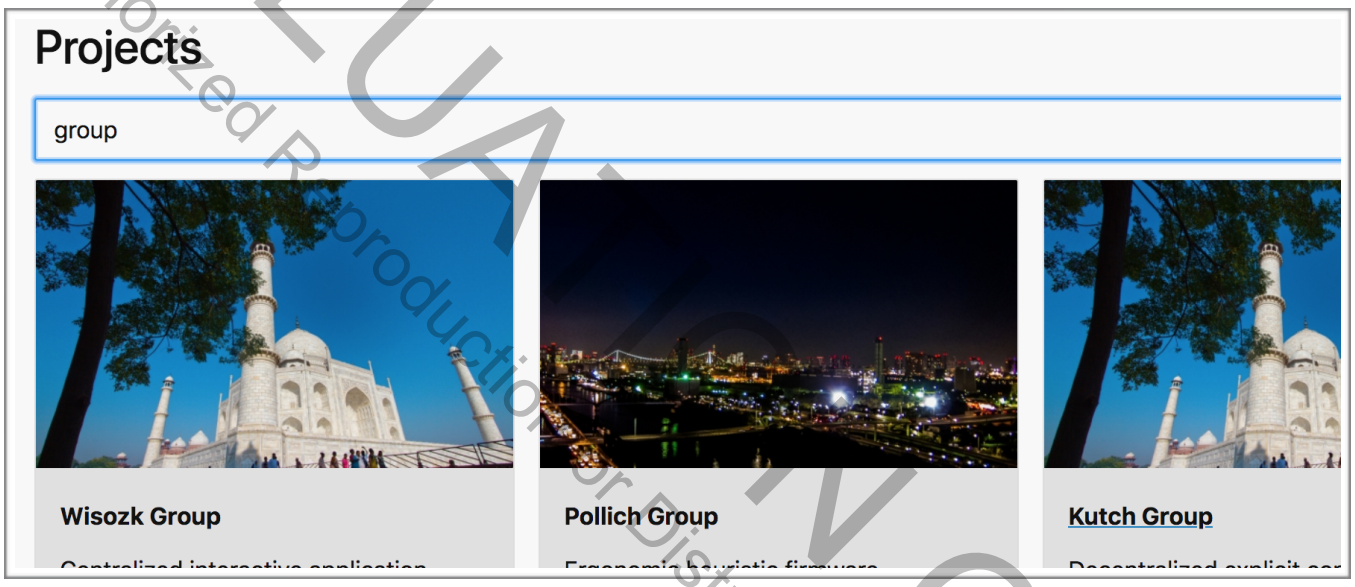
2. Verify

   a. **Save** your **code** changes.

   b. **Click** on **Projects** in the navigation if you aren't at that route already.

   c. **Type "group"** in the search input.

   d. The **projects** should be **filtered** to ones with **"group"** in their name.



Notice that the screen no longer flashes with your every keystroke resulting in a significantly improved user experience.

✓ You have completed Lab 31

Unit Testing Lab 1: First Test

Objectives

- [ ] Write your first JavaScript unit test
- [ ] Debug a unit test

Steps

Write your first JavaScript unit test

1. Close all your current editors and command prompts/terminals.

2. Open the following directory in your editor as the top level directory. This will your starting point and working directory for all the unit testing labs.

   - code\labs\unit-lab00\complete\project-manage

   > The code is the completed Angular labs up to this point. In addition, the unit test files (.spec) generated by the Angular CLI have been commented out where the tests are failing but the boiler-plate setup code for testing has been left to save us typing. We will get all the unit tests passing in the upcoming labs.

3. **Open** a **command prompt** (Windows) or **terminal** (Mac). Set the directory to **project-manage**.

4. **Run** the following **command** to install all JavaScript dependencies in this folder

```
npm install
```

5. **Run** the following **command** to build the Angular project and run the tests in both the Karma console test runner and the Jasmine HTML test runner.

```
ng test
```

6. A Chrome browser will open and run the unit tests in the Jasmine HTML test runner. Karma will run the tests and display the results at the command prompt or terminal.

```
Executed 8 of 8 SUCCESS
```

> Both processes will watch for change to files with .spec in their name and run again whenever you save a change. Note that running "npm test" runs the "ng test" command. Running either command is equivalent.

7. **Create** the following **spec file** and **add** the following **code**.

src\app\smoke-test.spec.ts

```
describe('Smoke Test', () => {
  it('should run a passing test', () => {
    expect(true).toEqual(false);
  });
});
```

8. **Save** the file and you should **see** a **failure message** similar to the one shown below.

```
Smoke Test should run a passing test FAILED
        Expected true to equal false.
...
```

9. **Change** false to true in the test.

src\app\smoke-test.spec.ts

```
describe('Smoke Test', () => {
  it('should run a passing test', () => {
    expect(true).toEqual(true);
  });
});
```

10. **Save** the file and you should **see** the following **success message**.

```
Executed 9 of 9 (SUCCESS)
```

Debug a unit test

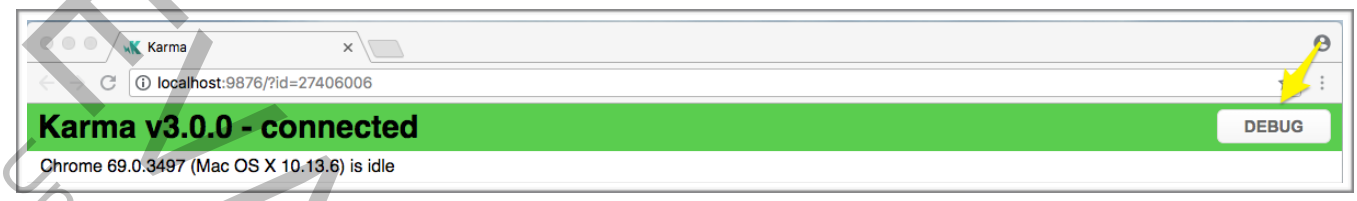
11. **Add** a **debugger statement** to the unit test as shown below.

src\app\smoke-test.spec.ts

```
describe('Smoke Test', () => {
  it('should run a passing test', () => {
    debugger;
    expect(true).toEqual(true);
  });
});
```

Your linter (tslint) will display an error that use of debugger statements is forbidden. You can safely ignore this error. It is trying to prevent you from accidentally leaving this line in production code and causing a defect. In this case, we are using it to make it easier to break into the test instead of searching for the file in the Chrome DevTools source tab.

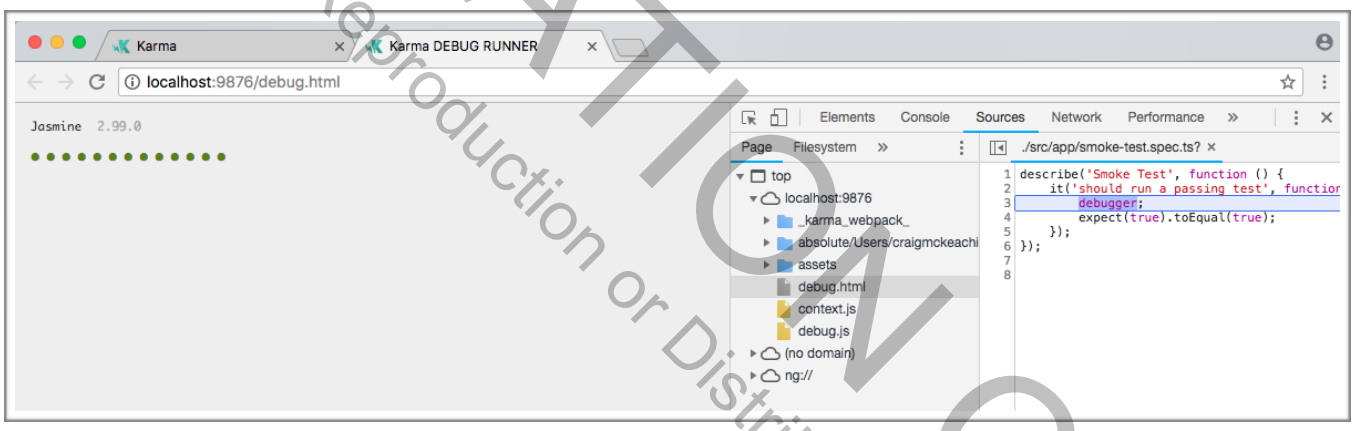
12. Find the karma browser window and click the **DEBUG** button in the upper right corner.



13. A new browser tab opens and re-runs the tests.

14. **Open** the Chrome browser's **DevTools** (**F12**).

15. **Refresh** the browser…and it **stops** at the **debugger** breakpoint.



16. **Click** the **continue** button **or F8** to let the script finish.

17. **Remove** the **debugger** statement from the test.

✓ You have completed Unit Testing: Lab 1

Unit Testing Lab 2: Component Test

Objectives

- ☐ Test a simple component
- ☐ Understand how to detect changes in a component

Steps

Test a simple component

1. As mentioned previously, your working directory for all the unit testing labs should be:
   - code\labs\unit-lab00\complete\project-manage
2. *If not already running*, **run** the command **ng test** in the working directory.

Steps continue on the next page.

3. **Add** a **variable** to hold the header element. **Query** the component for the **header element**. Write a test to **verify** the **value** of the **header element**.

src\app\home\home-container\home-container.component.spec.ts

```
...
describe('HomeContainerComponent', () => {
  let component: HomeContainerComponent;
  let fixture: ComponentFixture<HomeContainerComponent>;
  let h1: HTMLElement;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [HomeContainerComponent]
    }).compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(HomeContainerComponent);
    component = fixture.componentInstance;
    h1 = fixture.debugElement.nativeElement.querySelector('h1');
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should render title in a h1 tag', () => {
    expect(h1.textContent).toEqual('Home');
  });

});
```

4. **Save** the file and the **test** will **automatically run**.

5. **Verify** you see an additional successful test.

```
Executed 10 of 10 (SUCCESS)
```

Understand how to detect changes in a component

6. **Update** the **component** to **dynamically set** a **title property** into the header.

src\app\home\home-container\home-container.component.ts

```
...
@Component({
  selector: 'app-home-container',
  templateUrl: './home-container.component.html',
  styleUrls: ['./home-container.component.css']
})
export class HomeContainerComponent implements OnInit {
  title = '';
  constructor() {}

  ngOnInit() {}
}
```

src\app\home\home-container\home-container.component.html

```
<h1>Home</h1>
<h1>{{title}}</h1>
```

7. **Change** the "should render title…" **test** to **expect** an **empty string**.

src\app\home\home-container\home-container.component.spec.ts

```
...
describe('HomeContainerComponent', () => {
  let component: HomeContainerComponent;
  let fixture: ComponentFixture<HomeContainerComponent>;
  let h1: HTMLElement;

...

  it('should render title in a h1 tag', () => {
    expect(h1.textContent).toEqual('');
  });

});
```

8. **Save** the file to run the tests again and **verify** they all pass.

```
Executed 10 of 10 (SUCCESS)
```

9. **Add** another **test** that **sets** the **title** property on the component.

src\app\home\home-container\home-container.component.spec.ts

```
...
describe('HomeContainerComponent', () => {
  let component: HomeContainerComponent;
  let fixture: ComponentFixture<HomeContainerComponent>;
  let h1: HTMLElement;

...

  it('should render title in a h1 tag', () => {
    expect(h1.textContent).toEqual('');
  });

  it('changing title, updates h1', () => {
    const title = 'Home';
    component.title = title;
    expect(h1.textContent).not.toContain(title, 'before detectChanges');
    fixture.detectChanges();
    expect(h1.textContent).toContain(title);
  });

});
```

Notice that calling **detectChanges** on the fixture causes the component to render again and that prior to calling **detectChanges** the **h1** is not yet updated.

10. Verify the new test passes.

```
Executed 11 of 11 (SUCCESS)
```

✓ You have completed Unit Testing: Lab 2

7400 E. Orchard Road, Suite 1450 N
Greenwood Village, Colorado 80111
Ph: 303-302-5280
www.ITCourseware.com