

Tutoring for Angular

Stefano Secondo

03 Giugno 2025

Contents

Angular Mobile Responsiveness Exercise	1
Learning Objectives	1
Prerequisites	1
Exercise Overview	2
Step 1: Install Angular CDK Layout (5 minutes)	2
Step 2: Create a Simple Breakpoint Service (10 minutes)	2
☒ Quick Observable Primer for Students	2
Step 3: Add Mobile Navigation Imports (5 minutes)	3
Step 4: Create Responsive Template (15 minutes)	3
☒ Understanding the Async Pipe (Important for Students!)	5
Step 5: Update App Component for Responsive Layout (5 minutes)	5
Step 6: Add Responsive Styles (15 minutes)	6
Step 7: Add Professional Header Styles (10 minutes)	6
Step 8: Test Responsive Breakpoints (5 minutes)	9
Step 9: Bonus - Add Transition Effects (5 minutes)	9
Final Testing Checklist	10
Desktop (≥1024px)	10
Tablet (768px-1023px)	10
Mobile (<768px)	10
Very Small Mobile (<480px)	10
What You Learned	10
☒ Key Angular Concepts Mastered:	11
Next Steps	11
Common Issues & Solutions	11
Resources	11

Angular Mobile Responsiveness Exercise

Learning Objectives

By the end of this exercise, you will: - Understand CSS media queries and breakpoints - Use Angular CDK Layout for responsive design - Implement responsive navigation patterns - Properly integrate router-outlet with responsive navigation - See real-time responsive changes in the browser

Prerequisites

- Basic Angular knowledge
- Understanding of CSS
- Familiarity with Angular Material

Exercise Overview

We'll transform the current desktop-only navigation into a responsive system that:

- Shows tabs on desktop/tablet
- Shows a hamburger menu with drawer on mobile
- Properly displays routed content in both layouts
- Adapts content layout for different screen sizes

Step 1: Install Angular CDK Layout (5 minutes)

First, ensure Angular CDK is installed with the latest version:

```
npm install @angular/cdk@^19.2.18
```

Test Point: Open your browser dev tools and resize the window. Notice that the current navigation doesn't adapt to smaller screens.

Step 2: Create a Simple Breakpoint Service (10 minutes)

Create a service to detect screen size changes:

File: src/app/services/breakpoint.service.ts

```
import { Injectable } from '@angular/core';
import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';
import { map } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class BreakpointService {
  // Observable that emits true when screen is mobile size
  isMobile$;

  constructor(private breakpointObserver: BreakpointObserver) {
    this.isMobile$ = this.breakpointObserver
      .observe([Breakpoints.Handset])
      .pipe(map(result => result.matches));
  }
}
```

What's happening: This service uses Angular CDK to detect when the screen is "mobile size" (typically phones). We initialize the observable in the constructor to avoid property initialization errors.

☒ Quick Observable Primer for Students

What is isMobile\$? - The \$ suffix indicates this is an Observable - a special type of data stream in Angular - Think of it as a "live news feed" about screen size that continuously updates - Unlike regular variables that hold a single value, observables can emit multiple values over time - When you resize your browser window, this observable will emit true (mobile) or false (desktop)

Why use observables for responsive design? - Reactive: Automatically responds to screen size changes without manual checking - Real-time: Updates instantly when the user resizes the browser - Efficient: Only updates when something actually changes - Clean: No need to add event listeners or cleanup code

Coming up: In the next step, you'll see how we use the async pipe to "read" the current value from this observable stream in our template.

Test Point: No visible changes yet, but we now have a reactive way to detect screen size.

Step 3: Add Mobile Navigation Imports (5 minutes)

Update the header component to include necessary Material imports:

File: src/app/components/header/header.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatTabsModule } from '@angular/material/tabs';
import { MatSidenavModule } from '@angular/material/sidenav'; // ADD THIS
import { MatListModule } from '@angular/material/list'; // ADD THIS
import { MatIconModule } from '@angular/material/icon'; // ADD THIS
import { MatButtonModule } from '@angular/material/button'; // ADD THIS
import { RouterModule } from '@angular/router';
import { LogoComponent } from '../shared/logo/logo.component';
import { BreakpointService } from '../../services/breakpoint.service'; // ADD THIS

@Component({
  selector: 'app-header',
  standalone: true,
  imports: [
    CommonModule,
    MatToolbarModule,
    MatSidenavModule, // ADD THIS
    MatListModule, // ADD THIS
    MatIconModule, // ADD THIS
    MatButtonModule, // ADD THIS
    MatTabsModule,
    RouterModule,
    LogoComponent
  ],
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.scss']
})
export class HeaderComponent {
  navLinks = [
    { path: '/dashboard', label: 'Dashboard', icon: 'dashboard' }, // ADD ICON
    { path: '/employees', label: 'Employees', icon: 'people' }, // ADD ICON
    { path: '/offices', label: 'Office assignments', icon: 'business' }, // ADD ICON
    { path: '/floor-plan', label: 'Floor Plan', icon: 'map' } // ADD ICON
  ];

  constructor(public breakpointService: BreakpointService) {} // ADD THIS
}
```

What's happening: We're importing the components we'll need for mobile navigation and injecting our breakpoint service. Make sure to add all the import statements at the top to avoid "Cannot find name" errors.

Test Point: Still no visual changes, but we're ready for the next step.

Step 4: Create Responsive Template (15 minutes)

Replace the header template with a responsive version:

File: src/app/components/header/header.component.html

```

<!-- Desktop Navigation (shows on large screens) -->
@if (!(breakpointService.isMobile$ | async)) {
  <div class="desktop-nav">
    <mat-toolbar color="primary">
      <div class="brand">
        <app-logo [size]="180"></app-logo>
        <span class="brand-text">Office Management</span>
      </div>
    </mat-toolbar>
    <nav mat-tab-nav-bar [tabPanel]="tabPanel" backgroundColor="primary">
      @for (link of navLinks; track link.path) {
        <a mat-tab-link
          [routerLink]="link.path"
          routerLinkActive #rla="routerLinkActive"
          [active]="rla.isActive"
          [routerLinkActiveOptions]="{exact: false}">
          <mat-icon>{{link.icon}}</mat-icon>
          <span class="nav-label">{{link.label}}</span>
        </a>
      }
    </nav>
    <mat-tab-nav-panel #tabPanel></mat-tab-nav-panel>
  </div>
}

<!-- Mobile Navigation (shows on small screens) -->
@if (breakpointService.isMobile$ | async) {
  <div class="mobile-nav">
    <mat-toolbar color="primary">
      <button mat-icon-button (click)="drawer.toggle()">
        <mat-icon>menu</mat-icon>
      </button>
      <app-logo [size]="120"></app-logo>
      <span class="brand-text-mobile">Office Management</span>
    </mat-toolbar>

    <mat-sidenav-container class="mobile-container">
      <mat-sidenav #drawer mode="over" position="start" class="mobile-drawer">
        <div class="drawer-header">
          <h3>Navigation</h3>
        </div>
        <mat-nav-list>
          @for (link of navLinks; track link.path) {
            <a mat-list-item
              [routerLink]="link.path"
              (click)="drawer.close()">
              <mat-icon matListItemIcon>{{link.icon}}</mat-icon>
              <span matListItemTitle>{{link.label}}</span>
            </a>
          }
        </mat-nav-list>
      </mat-sidenav>
      <mat-sidenav-content>

```

```

        <router-outlet></router-outlet>
      </mat-sidenav-content>
    </mat-sidenav-container>
  </div>
}

```

What's happening: - We use @if with async pipe to show different navigation based on screen size - Desktop shows tabs, mobile shows hamburger menu with drawer - The #drawer template reference allows us to control the drawer - We use @for with track for optimal list rendering - IMPORTANT: The router-outlet is duplicated - it appears both in the main app component AND inside mat-sidenav-content for mobile layout

☒ Understanding the Async Pipe (Important for Students!)

In the template above, you'll notice we use breakpointService.isMobile\$ | async. Let's break this down:

What is an Observable? - isMobile\$ is an Observable (the \$ suffix is a convention indicating observables) - Observables are streams of data that can change over time - Think of it like a "live TV channel" that continuously broadcasts screen size changes - When you resize your browser, the observable emits true (mobile) or false (desktop)

What is the Async Pipe? - The | async is Angular's async pipe - It subscribes to the observable and automatically unsubscribes when the component is destroyed - It extracts the current value from the observable stream - Without async, you'd have a reference to the observable itself, not its current value

Why do we need it?

```

// This WON'T work - you're checking the observable object, not its value
@if (breakpointService.isMobile$) { ... }

```

```

// This WORKS - async pipe extracts the actual boolean value
@if (breakpointService.isMobile$ | async) { ... }

```

What happens behind the scenes: 1. Without async pipe: Angular sees an Observable object (always truthy) → condition always true
2. With async pipe: Angular subscribes to the observable, gets the actual boolean value (true/false), and updates the DOM when the value changes

Real-world analogy: - Observable = A thermometer that continuously measures temperature - Async pipe = Reading the current temperature display - Without async pipe = Looking at the thermometer device itself (not helpful!)

Memory Management: - The async pipe automatically handles subscription cleanup - This prevents memory leaks when the component is destroyed - You don't need to manually unsubscribe in ngOnDestroy

This is why the navigation magically switches when you resize the browser - the observable detects screen size changes and the async pipe updates the template in real-time!

Test Point: Open dev tools, resize the window. You should now see the navigation switch between desktop tabs and mobile hamburger menu! Click the hamburger menu to see the drawer slide out with navigation options. Click any navigation item to see it close the drawer and navigate to that page.

Step 5: Update App Component for Responsive Layout (5 minutes)

Update the main app component to support responsive content:

File: src/app/app.component.html

Update to:

```

<div class="app-container">
  <app-header></app-header>
  <main class="main-content">
    <router-outlet></router-outlet>
  </main>
</div>

```

```
</main>
</div>
```

What's happening: We keep the router-outlet in the main app component for desktop layout, but we also have it in the mobile sidenav for mobile layout. The CSS will handle showing/hiding the appropriate content area.

Test Point: Navigate to different pages in both desktop and mobile views. Content should display properly in both layouts.

Step 6: Add Responsive Styles (15 minutes)

Update the app component styles to handle responsive layout:

File: src/app/app.component.scss

```
.app-container {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.main-content {
  flex: 1;
  padding: 2rem;
  background-color: #f5f5f5;

  // Mobile layout adjustments
  @media (max-width: 767px) {
    position: absolute;
    top: 64px; // Height of mobile toolbar
    left: 0;
    right: 0;
    bottom: 0;
    padding: 1rem;
    overflow: auto;
  }

  // Tablet layout
  @media (min-width: 768px) and (max-width: 1023px) {
    padding: 1.5rem;
  }
}
```

What's happening: - Desktop uses normal flex layout - Mobile uses absolute positioning to avoid conflicts with the sidenav layout - The mobile content starts at top: 64px to account for the toolbar height - Responsive padding: 2rem desktop, 1.5rem tablet, 1rem mobile

Test Point: Resize the browser and notice how the content area adapts to different screen sizes.

Step 7: Add Professional Header Styles (10 minutes)

Update the header styles to make it look polished:

File: src/app/components/header/header.component.scss

```
:host {
  display: block;
}
```

```

// Desktop Navigation Styles
.desktop-nav {
  .brand {
    display: flex;
    align-items: center;
    gap: 1rem;
    margin-right: 3rem;

    .brand-text {
      font-size: 1.2rem;
      font-weight: 500;
      color: white;
    }
  }
}

mat-toolbar {
  padding: 0 1rem;
}

nav {
  background-color: white;
  padding: 0 1rem;

  a[mat-tab-link] {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 4px;
    min-width: 120px;
    padding: 12px 16px;

    mat-icon {
      font-size: 20px;
    }

    .nav-label {
      font-size: 0.875rem;
    }
  }

  ::ng-deep {
    .mdc-tab-indicator__content--underline {
      border-color: #1976d2;
    }

    .mat-mdc-tab-link {
      &.mdc-tab--active {
        color: #1976d2;
      }
    }
  }
}
}

```

```

// Mobile Navigation Styles
.mobile-nav {
  height: 100vh;
  display: flex;
  flex-direction: column;

  mat-toolbar {
    flex-shrink: 0;
    z-index: 1100;
  }

  .brand-text-mobile {
    font-size: 1.1rem;
    font-weight: 500;
    color: white;
    margin-left: 1rem;
  }

  .mobile-container {
    flex: 1;
    position: relative;
    overflow: hidden;

    mat-sidenav-content {
      height: 100%;
      overflow: auto;
    }
  }

  .mobile-drawer {
    width: 250px;
    background: white;
    box-shadow: 0 2px 10px rgba(0,0,0,0.2);

    .drawer-header {
      padding: 20px;
      background-color: #1976d2;
      color: white;

      h3 {
        margin: 0;
        font-size: 1.2rem;
        font-weight: 500;
      }
    }
  }

  mat-nav-list {
    padding-top: 0;

    a[mat-list-item] {
      color: #333;
      padding: 16px 24px;
    }
  }

```



```

    &:hover {
      background-color: rgba(0, 0, 0, 0.04);
    }

    mat-icon {
      color: #1976d2;
      margin-right: 16px;
    }
  }
}
}
}

```

What's happening: - Desktop styles make the tabs look professional with icons and labels - Mobile styles create a full-height layout with a slide-out drawer - The drawer has a blue header matching the primary color and clean white content area - Proper z-index management prevents overlay issues

Test Point: Resize your browser. The interface should now look polished on both desktop and mobile! Click the hamburger menu to see the navigation drawer slide out beautifully.

Step 8: Test Responsive Breakpoints (5 minutes)

Open your browser developer tools:

1. Open DevTools: Press F12 or right-click → Inspect
2. Toggle Device Toolbar: Click the device icon (📱) or press Ctrl+Shift+M
3. Test Different Sizes:
 - Desktop ($\geq 1024\text{px}$): Should show tab navigation
 - Tablet ($768\text{px} - 1023\text{px}$): Should show tab navigation
 - Mobile ($< 768\text{px}$): Should show hamburger menu

Breakpoint Testing Checklist: - [] At 1200px width: Tabs visible, content displays properly - [] At 800px width: Tabs visible, content displays properly - [] At 600px width: Hamburger menu visible, content displays properly - [] At 400px width: Hamburger menu visible, content displays properly - [] Click hamburger menu: Drawer slides out - [] Click nav item in drawer: Drawer closes and navigates, content shows

Step 9: Bonus - Add Transition Effects (5 minutes)

Make the responsive changes smoother by adding these transitions to the header styles:

File: `src/app/components/header/header.component.scss` (add to existing styles)

```

// Smooth transitions (add to the end of the file)
.desktop-nav,
.mobile-nav {
  transition: all 0.3s ease-in-out;
}

mat-sidenav {
  transition: transform 0.3s ease-in-out;
}

// Responsive logo sizing
app-logo {

```

```
    transition: all 0.3s ease-in-out;
}
```

Test Point: Resize the browser slowly and notice smooth transitions between layouts.

Final Testing Checklist

Test your responsive implementation:

Desktop ($\geq 1024\text{px}$)

- ☐ Tab navigation visible
- ☐ Logo size 180px
- ☐ Tabs show icons and labels
- ☐ Content displays below tabs
- ☐ Navigation works properly

Tablet (768px-1023px)

- ☐ Tab navigation still visible
- ☐ Tabs adapt if needed
- ☐ Content has proper spacing
- ☐ Navigation works properly

Mobile ($< 768\text{px}$)

- ☐ Hamburger menu visible
- ☐ Logo size 120px
- ☐ Drawer opens/closes smoothly
- ☐ Navigation items have icons
- ☐ Content displays in absolute positioned area
- ☐ Cards/buttons are touch-friendly
- ☐ No content overlap issues

Very Small Mobile ($< 480\text{px}$)

- ☐ Content has reduced padding
 - ☐ All elements still accessible
 - ☐ No horizontal scrolling
 - ☐ Drawer navigation works
-

What You Learned

1. CSS Media Queries: How to use `@media` rules for responsive design
2. Angular CDK Layout: Reactive breakpoint detection with `BreakpointObserver`
3. Observables & Reactive Programming: Understanding data streams that emit values over time
4. Async Pipe: How to subscribe to observables in templates and extract their current values
5. Conditional Templates: Using `@if` with observables for responsive UI
6. Material Design Patterns: Tabs for desktop, drawer for mobile
7. Dual Router Outlets: Managing router-outlet in both desktop and mobile layouts
8. Absolute Positioning: Using CSS positioning for mobile-specific layouts
9. Memory Management: How async pipe prevents memory leaks through automatic subscription cleanup

☒ Key Angular Concepts Mastered:

- Observable streams vs regular variables
- Async pipe subscription and unsubscription
- Reactive UI patterns that respond to data changes
- Template conditional rendering with modern Angular syntax (@if, @for)
- Service injection and dependency management

Next Steps

To further enhance your responsive skills:

1. Add more breakpoints (large desktop, small mobile)
2. Implement responsive tables that stack on mobile
3. Add touch gestures for mobile drawer
4. Optimize images for different screen densities
5. Test on real devices not just browser simulation

Common Issues & Solutions

Issue: Navigation doesn't switch at expected breakpoint Solution: Check that Angular CDK is properly imported and Breakpoints.Handset matches your target devices

Issue: Drawer doesn't open Solution: Ensure mat-sidenav-container has proper height and #drawer template reference is correct

Issue: Content doesn't display properly on mobile Solution: ☒ CRITICAL - Make sure the mobile content uses absolute positioning with top: 64px and the sidenav content area is properly configured

Issue: Content scrolling issues on mobile Solution: Check height and overflow properties on containers, ensure mobile-container has proper flex settings and the absolute positioned content has overflow: auto

Issue: Duplicate content or layout conflicts Solution: The dual router-outlet approach means content appears in both desktop and mobile areas - the CSS media queries handle showing/hiding the appropriate layout

Resources

- Angular CDK Layout Documentation
- Material Design Responsive UI
- CSS Media Queries MDN
- Angular Router Integration

Happy coding! ☒