Enhanced E-commerce Example with Host Elements

Introduction

This example builds on the logic found in Anatomy of a Component and demonstrates more in-depth use of Angular Component Host Elements. This code builds on the original e-commerce application logic, leveraging Angular's host element functionalities to demonstrate how to manipulate a component or directive's host element. We will: - Use `@HostBinding` to dynamically set properties or classes on the host element. - Use `@HostListener` to respond to events on the host element. - Configure the host metadata property in a component to bind classes and styles.

We will create a new directive (`HoverHighlightDirective`) to highlight product items when hovered, update the ProductItemComponent to utilize the directive and host properties, and show how everything ties together in the module.

Overview

We'll extend our existing e-commerce application by adding: 1) A directive (`HoverHighlightDirective`) that applies highlight styling to the host element when hovered (using `@HostBinding` and `@HostListener`). 2) Updates to `ProductItemComponent` to demonstrate how to: - Use the hover directive. - Bind a dynamic class to the host element via the component's host property. 3) Explanations of each file to showcase how these host element features work together.

Code

Below are the new and updated files demonstrating host element functionalities. All other files from the original example remain the same unless explicitly shown here.

hover-highlight.directive.ts

```
import { Directive, ElementRef, Renderer2, HostBinding, HostListener, Input } from
        '@angular/core';

@Directive({
  selector: '[appHoverHighlight]'
})
export class HoverHighlightDirective {
  @Input() highlightColor = 'lightyellow';

  // Dynamically set the background color of the host element
  @HostBinding('style.backgroundColor') backgroundColor: string | null = null;

  // Add or remove a class on the host element
  @HostBinding('class.highlighted') isHighlighted = false;

  constructor(private el: ElementRef, private renderer: Renderer2) {}

  // Listen to mouse enter events on the host element
  @HostListener('mouseenter')
  onMouseEnter() {
    this.backgroundColor = this.highlightColor;
    this.isHighlighted = true;
  }
```

```
  // Listen to mouse leave events on the host element
  @HostListener('mouseleave')
  onMouseLeave() {
    this.backgroundColor = null;
    this.isHighlighted = false;
  }
}
```

product-item.component.ts

Below is the updated version of `ProductItemComponent`. Changes include: - Using the host property to bind a class (`.product-item`) directly to the component's host element. - Demonstrating the newly created `HoverHighlightDirective` in the component's template.

```
import {
  Component,
  Input,
  Output,
  EventEmitter,
  OnChanges,
  SimpleChanges,
  ChangeDetectionStrategy
} from '@angular/core';

@Component({
  selector: 'app-product-item',
  template: `
    <div appHoverHighlight [highlightColor]="'lightblue'">
      <h3>{{ product.name }}</h3>
      <p>{{ product.description }}</p>
      <ng-content></ng-content>
      <button (click)="addToCart()">Add to Cart</button>
    </div>
  `,
  styleUrls: ['./product-item.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  host: {
    '[class.product-item]': 'true'
  }
})
export class ProductItemComponent implements OnChanges {
  @Input() product: any;
  @Output() add = new EventEmitter<any>();

  ngOnChanges(changes: SimpleChanges) {
    if (changes['product']) {
      console.log('Product changed:', changes['product'].currentValue);
    }
  }

  addToCart() {
    this.add.emit(this.product);
  }
}
```

app.module.ts

We add `HoverHighlightDirective` to the declarations array.

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

import { AppComponent } from './app.component';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductItemComponent } from './product-item/product-item.component';
import { CartComponent } from './cart/cart.component';
import { HoverHighlightDirective } from './hover-highlight.directive';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
    ProductItemComponent,
    CartComponent,
    HoverHighlightDirective
  ],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Example CSS for Host Element

Optionally, you can add CSS classes to style highlighted elements or the product-item host class.

```
.product-item {
  margin: 1rem 0;
  border: 1px solid #ccc;
  padding: 1rem;
  transition: background-color 0.3s ease;
}

.highlighted {
  outline: 2px solid #73cdf7;
}
```

Explanation of the Code

HoverHighlightDirective

- **Purpose**: Demonstrates Angular's host element functionalities via `@HostBinding` and `@HostListener`.
  - `@HostBinding('style.backgroundColor')`: Dynamically updates the style on the directive's host element.
  - `@HostBinding('class.highlighted')`: Adds/removes a CSS class based on the highlight state.
  - `HostListener('mouseenter') / @HostListener('mouseleave')`: Listens for mouse events on the host element to trigger highlight changes.

ProductItemComponent

- **Purpose**: Displays individual product details, extended with a `host` property.
- `host` property in `@Component`:
  - `[class.product-item]: true` ensures that the component's root element has a `product-item` class, which can be styled in the associated CSS.

- **Template**:
  - Uses `<div appHoverHighlight [highlightColor]="'lightblue'">` to apply the new directive, using a property binding to control the highlight color.

AppModule

- **Purpose**: Registers the new `HoverHighlightDirective` so that it’s recognized application-wide.

Conclusion

With these additions, you expand on the existing e-commerce Angular application to showcase host element interactions. The user can visually see components highlighted on hover, with dynamic class and style bindings on the host element, illustrating Angular’s powerful host element features.