Enhanced E-commerce Example with Custom Events with Outputs

Introduction

This example builds on the logic found in Anatomy of a Component and demonstrates more in-depth use of Angular Component Outputs with Custom Events, including multiple custom event outputs, child-to-parent and parent-to-child interactions, and some more advanced patterns described in the Angular docs about outputs. This expanded example introduces a “Wishlist” feature, illustrating how components can emit events to parents and siblings, and how those parents can respond accordingly.

Code

app.component.ts

```
import { Component } from '@angular/core';
import { Product } from './models/product.model';

@Component({
  selector: 'app-root',
  template: `
    <h1>E-commerce Store</h1>
    <!-- Display product list and handle adding to cart/wishlist -->
    <app-product-list
      (addToCart)="handleAddToCart($event)"
      (addToWishlist)="handleAddToWishlist($event)"
    ></app-product-list>

    <!-- Display the shopping cart -->
    <app-cart
      [cartItems]="cartItems"
      (removeFromCart)="handleRemoveFromCart($event)"
    ></app-cart>

    <!-- Display the wishlist -->
    <app-wishlist
      [wishlistItems]="wishlistItems"
      (moveToCart)="handleMoveToCart($event)"
      (removeFromWishlist)="handleRemoveFromWishlist($event)"
    ></app-wishlist>
  `,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  cartItems: Product[] = [];
  wishlistItems: Product[] = [];

  /**
   * Handles adding item to the cart.
   * @param product The product being added.
   */
  handleAddToCart(product: Product): void {
    this.cartItems.push(product);
  }

  /**
```

```
    * Handles removing item from the cart.
    * @param product The product being removed.
    */
  handleRemoveFromCart(product: Product): void {
    this.cartItems = this.cartItems.filter(item => item.id !== product.id);
  }

  /**
    * Handles adding item to the wishlist.
    * @param product The product being added.
    */
  handleAddToWishlist(product: Product): void {
    // Prevent duplicates for demonstration
    const isInWishlist = this.wishlistItems.some(item => item.id === product.id);
    if (!isInWishlist) {
      this.wishlistItems.push(product);
    }
  }

  /**
    * Handles moving an item from the wishlist to the cart.
    * @param product The product to move.
    */
  handleMoveToCart(product: Product): void {
    // Remove from wishlist
    this.handleRemoveFromWishlist(product);
    // Add to cart
    this.handleAddToCart(product);
  }

  /**
    * Handles removing item from the wishlist.
    * @param product The product being removed.
    */
  handleRemoveFromWishlist(product: Product): void {
    this.wishlistItems = this.wishlistItems.filter(item => item.id !== product.id);
  }
}
```

product-list.component.ts

```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
import { ProductService } from '../product.service';
import { Product } from '../models/product.model';

@Component({
  selector: 'app-product-list',
  template: `
    <h2>Products</h2>
    <app-product-item
      *ngFor="let product of products"
      [product]="product"
      (add)="onAddToCart($event)"
      (wishlist)="onAddToWishlist($event)"
    ></app-product-item>
  `,
  styleUrls: ['./product-list.component.css']
})
export class ProductListComponent implements OnInit {
  products: Product[] = [];
  @Output() addToCart = new EventEmitter<Product>();
  @Output() addToWishlist = new EventEmitter<Product>();
```

```
  constructor(private productService: ProductService) {}

  ngOnInit(): void {
    this.products = this.productService.getProducts();
  }

  /**
   * Emits addToCart event to the parent when product is added to the cart.
   * @param product Product to be added to cart
   */
  onAddToCart(product: Product): void {
    this.addToCart.emit(product);
  }

  /**
   * Emits addToWishlist event to the parent when product is added to the wishlist.
   * @param product Product to be added to wishlist
   */
  onAddToWishlist(product: Product): void {
    this.addToWishlist.emit(product);
  }
}
```

product-item.component.ts

```
import {
  Component,
  Input,
  Output,
  EventEmitter,
  OnChanges,
  SimpleChanges,
  ChangeDetectionStrategy
} from '@angular/core';
import { Product } from '../models/product.model';

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

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

```
  /**
   * Emits 'add' event with the product when the "Add to Cart" button is clicked.
   */
  addToCart(): void {
    this.add.emit(this.product);
  }

  /**
   * Emits 'wishlist' event with the product when the "Add to Wishlist" button is
        clicked.
   */
  addToWishlist(): void {
    this.wishlist.emit(this.product);
  }
}
```

cart.component.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Product } from '../models/product.model';

@Component({
  selector: 'app-cart',
  template: `
    <h2>Shopping Cart</h2>
    <div *ngFor="let item of cartItems">
      {{ item.name }} - {{ item.price | currency }}
      <button (click)="removeFromCart(item)">Remove</button>
    </div>
  `,
  styleUrls: ['./cart.component.css']
})
export class CartComponent {
  @Input() cartItems: Product[] = [];
  @Output() removeFromCart = new EventEmitter<Product>();

  /**
   * Emits an event to remove an item from the cart.
   * @param item The product item to remove.
   */
  removeFromCart(item: Product): void {
    this.removeFromCart.emit(item);
  }
}
```

wishlist.component.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Product } from '../models/product.model';

@Component({
  selector: 'app-wishlist',
  template: `
    <h2>My Wishlist</h2>
    <div *ngIf="wishlistItems.length; else emptyWishlist">
      <app-wishlist-item
        *ngFor="let item of wishlistItems"
        [item]="item"
        (moveToCart)="onMoveToCart($event)"
        (remove)="onRemoveWishlistItem($event)"
```

```
      ></app-wishlist-item>
    </div>
    <ng-template #emptyWishlist>
      <p>Your wishlist is empty</p>
    </ng-template>
  `,
  styleUrls: ['./wishlist.component.css']
})
export class WishlistComponent {
  @Input() wishlistItems: Product[] = [];
  @Output() moveToCart = new EventEmitter<Product>();
  @Output() removeFromWishlist = new EventEmitter<Product>();

  /**
   * Emits moveToCart event to parent to move item from wishlist to cart.
   * @param item The product to move to cart.
   */
  onMoveToCart(item: Product): void {
    this.moveToCart.emit(item);
  }

  /**
   * Emits removeFromWishlist event to parent to remove an item from wishlist.
   * @param item The product to remove.
   */
  onRemoveWishlistItem(item: Product): void {
    this.removeFromWishlist.emit(item);
  }
}
```

wishlist-item.component.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
import { Product } from '../models/product.model';

@Component({
  selector: 'app-wishlist-item',
  template: `
    <div class="wishlist-item">
      <h4>{{ item.name }}</h4>
      <button (click)="moveToCartItem()">Move to Cart</button>
      <button (click)="removeItem()">Remove</button>
    </div>
  `,
  styleUrls: ['./wishlist-item.component.css']
})
export class WishlistItemComponent {
  @Input() item!: Product;
  @Output() moveToCart = new EventEmitter<Product>();
  @Output() remove = new EventEmitter<Product>();

  /**
   * Emits "moveToCart" event when user clicks "Move to Cart".
   */
  moveToCartItem(): void {
    this.moveToCart.emit(this.item);
  }

  /**
   * Emits "remove" event to remove this item from the wishlist.
   */
  removeItem(): void {
```

```
    this.remove.emit(this.item);
  }
}
```

product.service.ts

```
import { Injectable } from '@angular/core';
import { Product } from './models/product.model';

@Injectable({
  providedIn: 'root'
})
export class ProductService {
  private products: Product[] = [
    {
      id: 1,
      name: 'Laptop',
      description: 'A high-performance laptop',
      price: 1299.99
    },
    {
      id: 2,
      name: 'Smartphone',
      description: 'A powerful smartphone',
      price: 799.99
    },
    {
      id: 3,
      name: 'Headphones',
      description: 'Noise-cancelling headphones',
      price: 199.99
    },
    {
      id: 4,
      name: 'Smart Watch',
      description: 'Track your fitness and notifications',
      price: 299.99
    }
  ];

  /**
   * Returns an array of products to be displayed in the ProductListComponent.
   */
  getProducts(): Product[] {
    return this.products;
  }
}
```

product.model.ts

```
export interface Product {
  id: number;
  name: string;
  description: string;
  price: number;
}
```

app.module.ts

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

import { AppComponent } from './app.component';
import { ProductListComponent } from './product-list.component';
import { ProductItemComponent } from './product-item.component';
import { CartComponent } from './cart.component';
import { WishlistComponent } from './wishlist.component';
import { WishlistItemComponent } from './wishlist-item.component';
import { ProductService } from './product.service';

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

Explanation of the Code

Overview

Below is a high-level overview, following the same style and structure as the Anatomy of a Component reference, and focusing on the new features and interactions introduced:

AppComponent

- **Purpose**:
  - Acts as the root component of the application, orchestrating interactions between the product list, cart, and wishlist components.
- **Template**:
  - `<app-product-list>`: Displays the list of products. It emits two events: `addToCart` and `addToWishlist`, which are handled by the `AppComponent`.
  - `<app-cart>`: Displays the shopping cart items. It receives `cartItems` from the `AppComponent` and emits `removeFromCart` events.
  - `<app-wishlist>`: Displays the wishlist items. It receives `wishlistItems` and emits `moveToCart` and `removeFromWishlist` events.
- **Logic**:
  - **State Management**: Maintains two arrays, `cartItems` and `wishlistItems`, to track products in the cart and wishlist, respectively.
  - **Event Handlers**:
    - `handleAddToCart(product)`: Adds a product to the `cartItems` array.
    - `handleRemoveFromCart(product)`: Removes a product from the `cartItems` array.
    - `handleAddToWishlist(product)`: Adds a product to the `wishlistItems` array if it’s not already present.
    - `handleMoveToCart(product)`: Moves a product from the wishlist to the cart by removing it from `wishlistItems` and adding it to `cartItems`.
    - `handleRemoveFromWishlist(product)`: Removes a product from the `wishlistItems` array.

ProductListComponent

- **Purpose**:
  - Displays a list of products and allows users to add products to the cart or wishlist.
- **Template**:
  - Iterates over the products array using `*ngFor` and renders a `<app-product-item>` for each product.
  - Binds each product to the `[product]` input property of `ProductItemComponent`.
  - Captures the `add` and `wishlist` events from `ProductItemComponent` and calls `onAddToCart()` and `onAddToWishlist()` respectively.
- **Logic**:
  - Fetches the list of products from `ProductService` in `ngOnInit()`.
  - Emits `addToCart` and `addToWishlist` events to notify the parent component (`AppComponent`) when a product is added to the cart or wishlist.

ProductItemComponent

- **Purpose**:
  - Displays individual product details and provides buttons to add the product to the cart or wishlist.
- **Template**:
  - Displays the product name and description.
  - Includes “Add to Cart” and “Add to Wishlist” buttons that trigger the respective methods.
- **Logic**:
  - Implements the `ngOnChanges()` lifecycle hook to detect changes to the `product` input.
  - Emits `add` and `wishlist` events when the respective buttons are clicked.

CartComponent

- **Purpose**:
  - Displays the items added to the shopping cart and allows users to remove items.
- **Template**:
  - Iterates over the `cartItems` array using `*ngFor` and displays each item’s name and price.
  - Includes a “Remove” button for each item that triggers the `removeFromCart()` method.
- **Logic**:
  - Emits a `removeFromCart` event to notify the parent component (`AppComponent`) when an item is removed from the cart.

WishlistComponent

- **Purpose**:
  - Displays the items added to the wishlist and allows users to move items to the cart or remove them.
- **Template**:
  - Iterates over the `wishlistItems` array using `*ngFor` and renders a `<app-wishlist-item>` for each item.
  - Uses an `ng-template` to display a message when the wishlist is empty.
- **Logic**:
  - Emits `moveToCart` and `removeFromWishlist` events to notify the parent component (`AppComponent`) when an item is moved to the cart or removed from the wishlist.

WishlistItemComponent

- **Purpose**:
  - Displays individual wishlist item details and provides buttons to move the item to the cart or remove it from the wishlist.
- **Template**:
  - Displays the item name.
  - Includes “Move to Cart” and “Remove” buttons that trigger the respective methods.
- **Logic**:
  - Emits `moveToCart` and `remove` events when the respective buttons are clicked.

ProductService

- **Purpose**:
  - Provides product data to components.
- **Logic**:
  - Defines a private `products` array containing product objects.
  - Implements `getProducts()` method to return the list of `products`.

ProductModel

- **Purpose**:
  - Defines the structure of a product object with properties such as `id`, `name`, `description`, and `price`.

AppModule

- **Purpose**:
  - The root module that bootstraps the application.
- **Declarations**:
  - Lists all components used in the application.
- **Imports**:
  - `BrowserModule` is imported to run the app in a browser.
- **Providers**:
  - Registers `ProductService` as a provider.
- **Bootstrap**:
  - Bootstraps the `AppComponent` to launch the application.

Conclusion

This enhanced example demonstrates a more complex interaction between components using custom events and showcases how Angular’s event system can be used to manage state and interactions in a modular way.