




Using Angular Computed Signals for Cart Totals


 **Deborah Kurata**
9.39K subscribers


 Subscribed 

 163



 Share

 Download

 Thanks

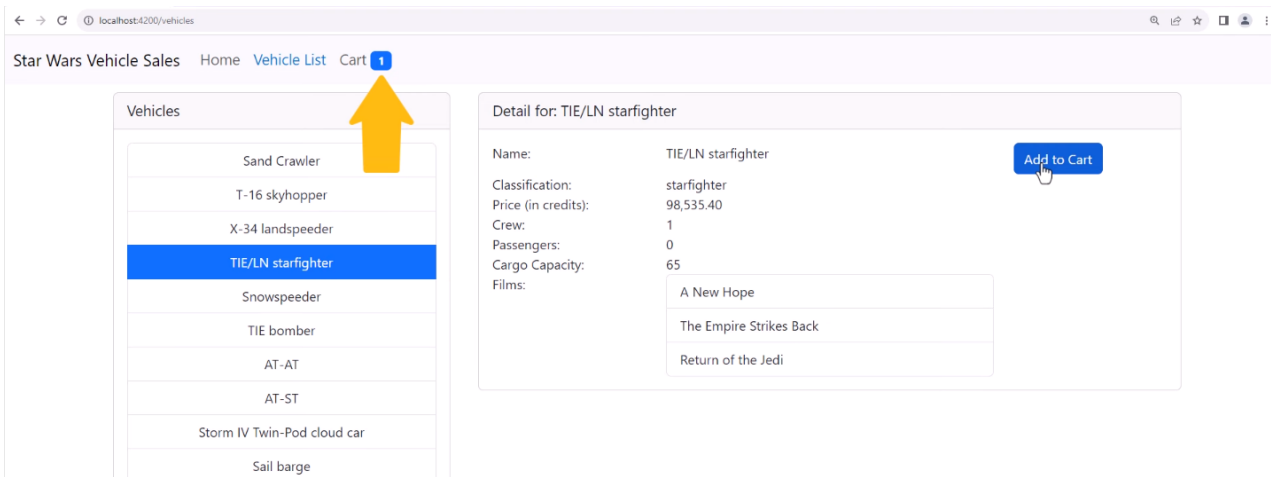
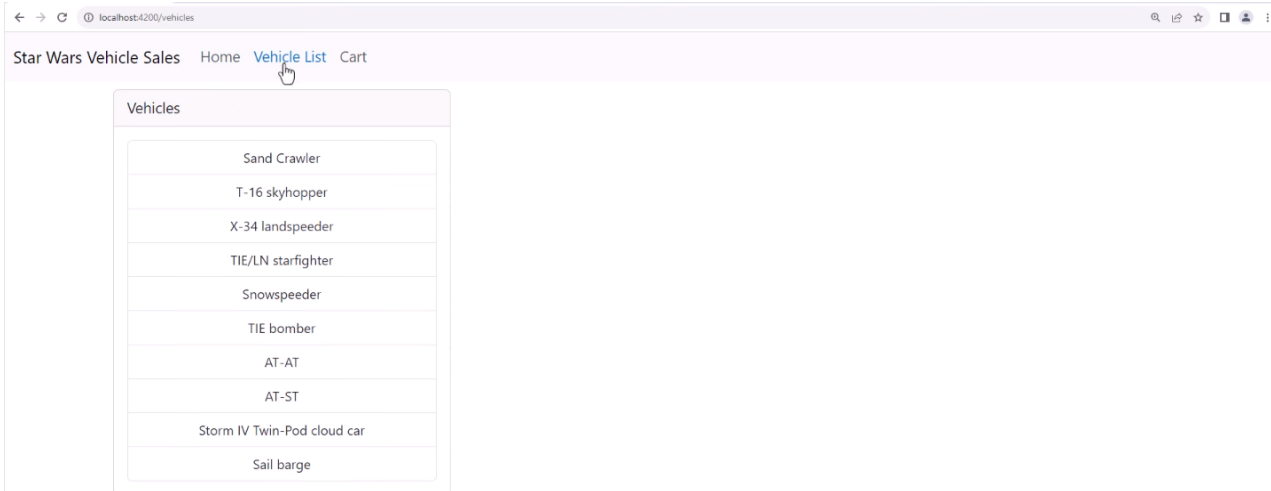
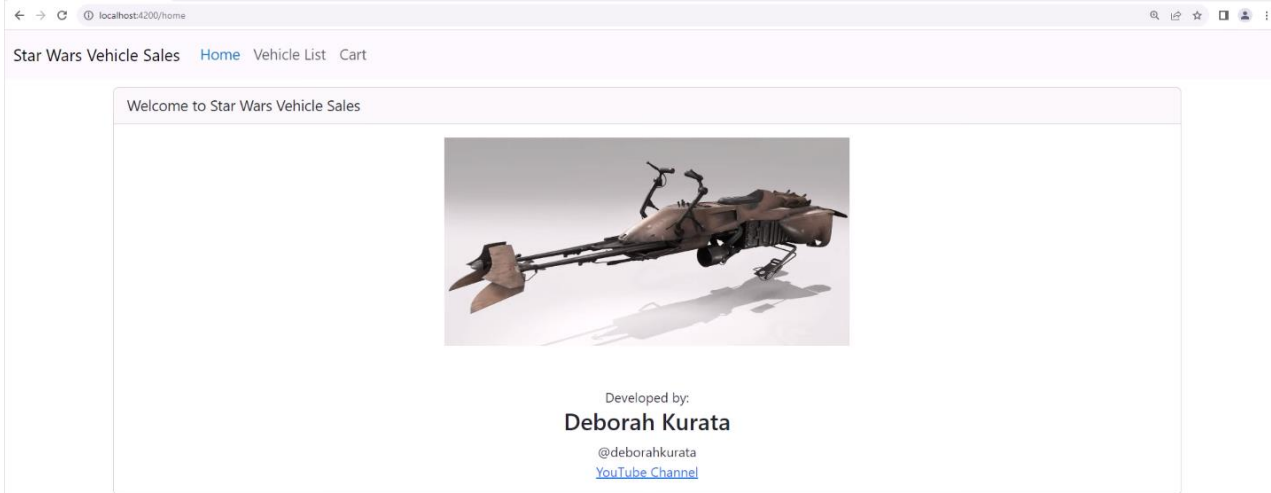


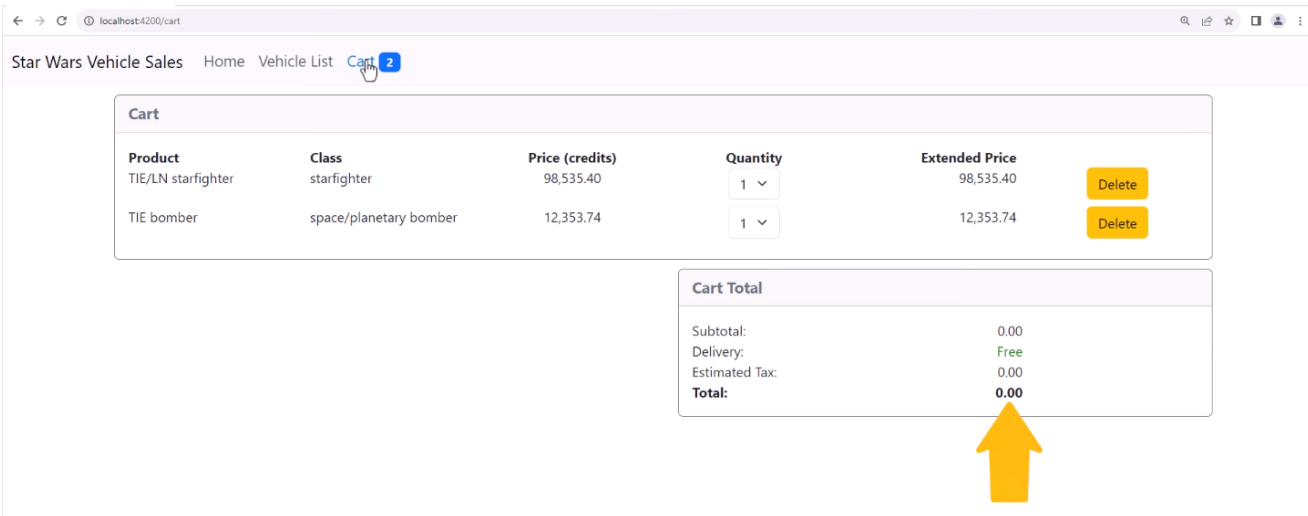
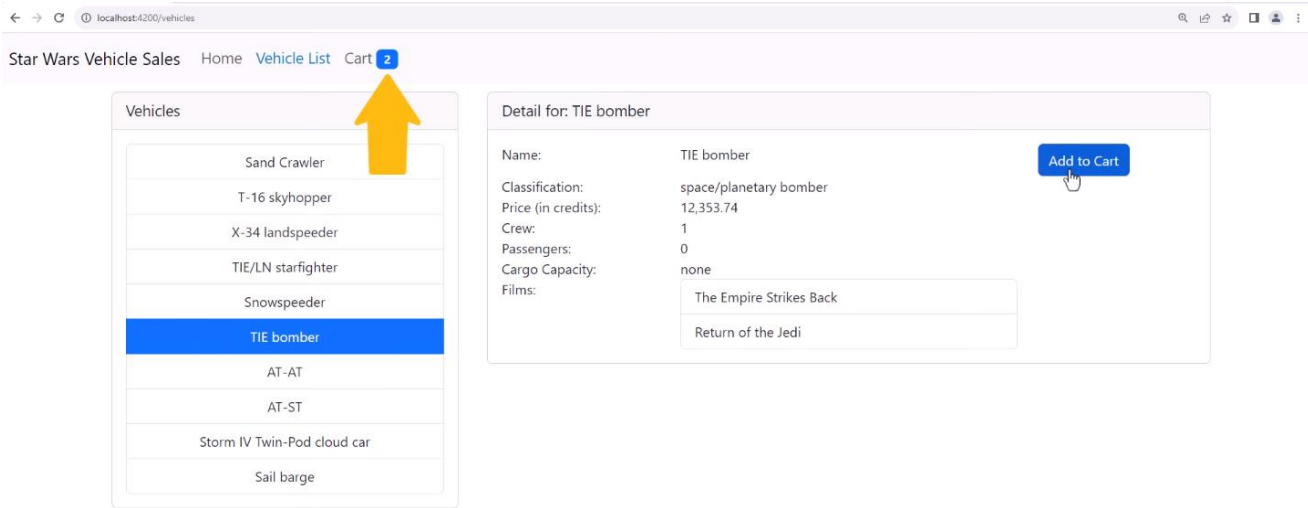
1,874 views Dec 14, 2023 [#angular](#) [#signals](#) [#angularsignals](#)
With Angular's new computed signals our data is reactive!

Computed signals derive their value from other signals. We perform a calculation or operation using one or more dependent signals. And the computed signal automatically recalculates as the dependent signals change.

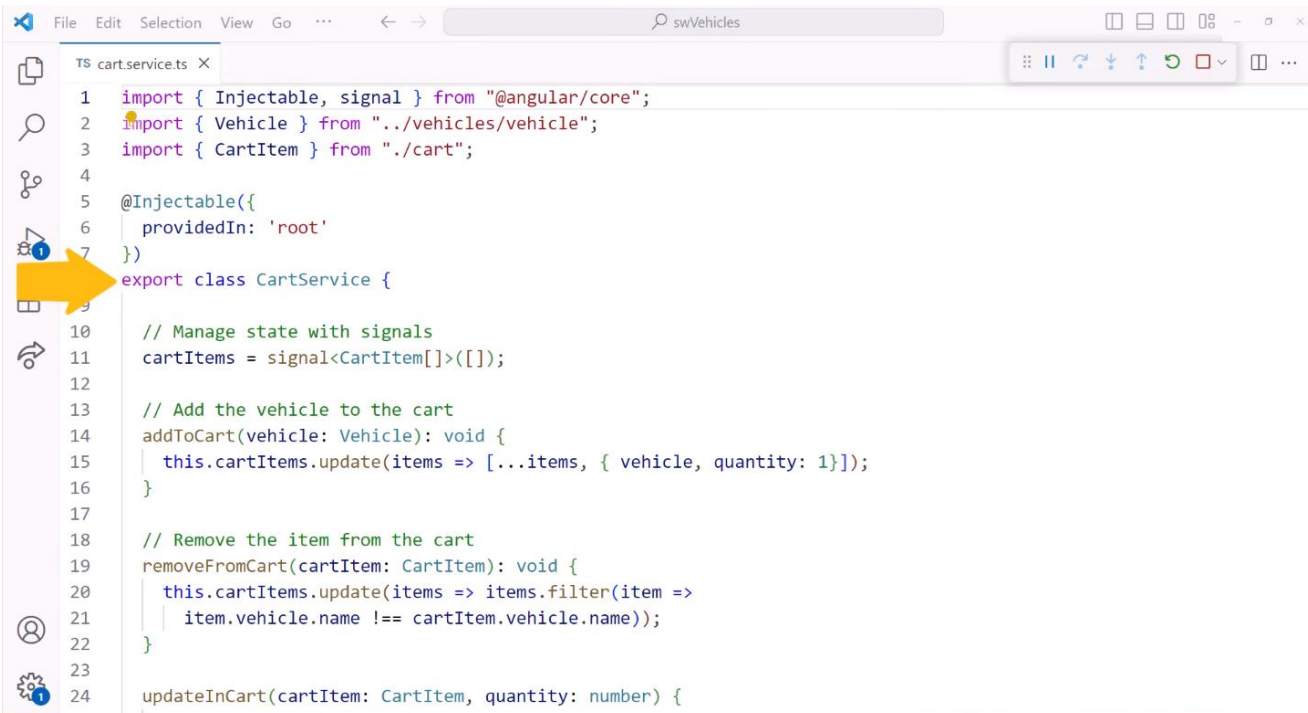
That's why computed signals are one of the most powerful reactive features of signals.

In this video, we walk through how to use computed signals to react to cart changes. We'll calculate the cart totals and those values will **automatically** react and recalculate as needed.





But the Cart totals doesn't yet react to changes in the Cart, we will use **computed()** signals to react to Cart changes

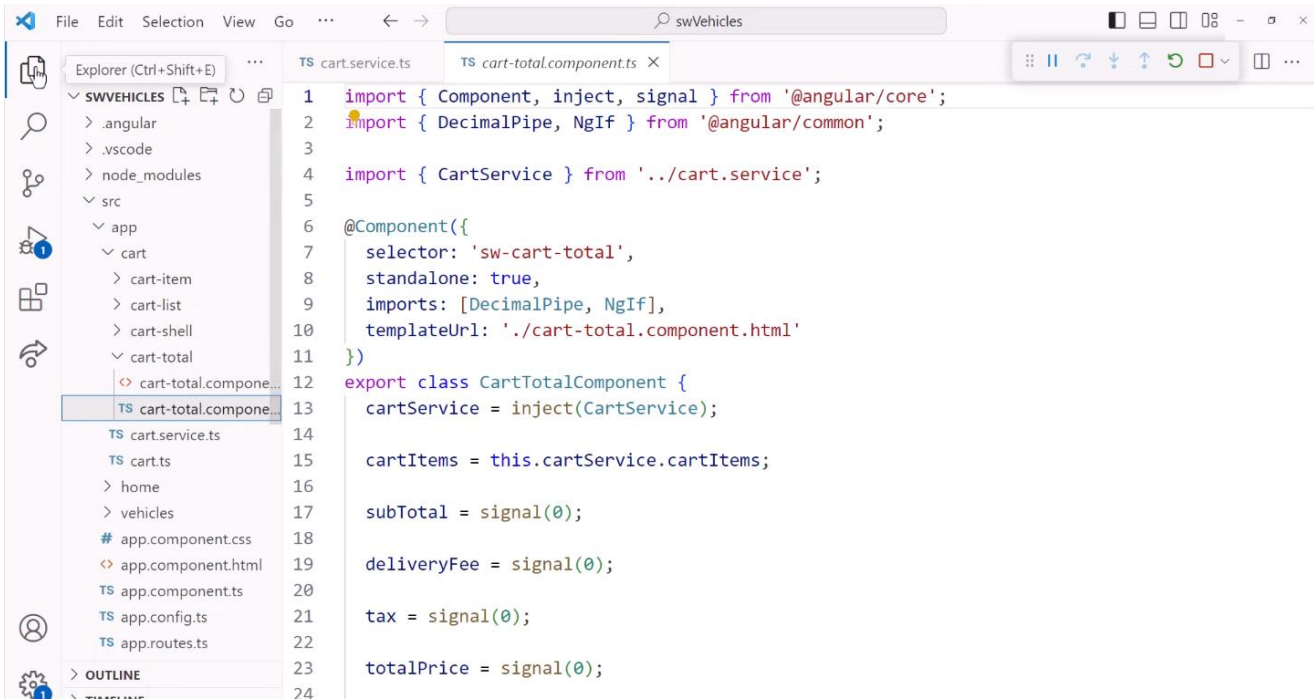


```
6   providedIn: 'root'
7   })
8   export class CartService {
9
10    // Manage state with signals
11    cartItems = signal<CartItem[]>([]);
12
13    // Add the vehicle to the cart
14    addToCart(vehicle: Vehicle): void {
15      this.cartItems.update(items => [...items, { vehicle, quantity: 1}]);
16    }
17
18    // Remove the item from the cart
19    removeFromCart(cartItem: CartItem): void {
20      this.cartItems.update(items => items.filter(item =>
21        item.vehicle.name !== cartItem.vehicle.name));
22    }
23
24    updateInCart(cartItem: CartItem, quantity: number) {
25      this.cartItems.update(items =>
26        items.map(item => item.vehicle.name === cartItem.vehicle.name ?
27          { vehicle: cartItem.vehicle, quantity } : item));
28    }
29  }
```

Next, let's add the `computed()` signals we need to calculate the cart total and make it react to changes

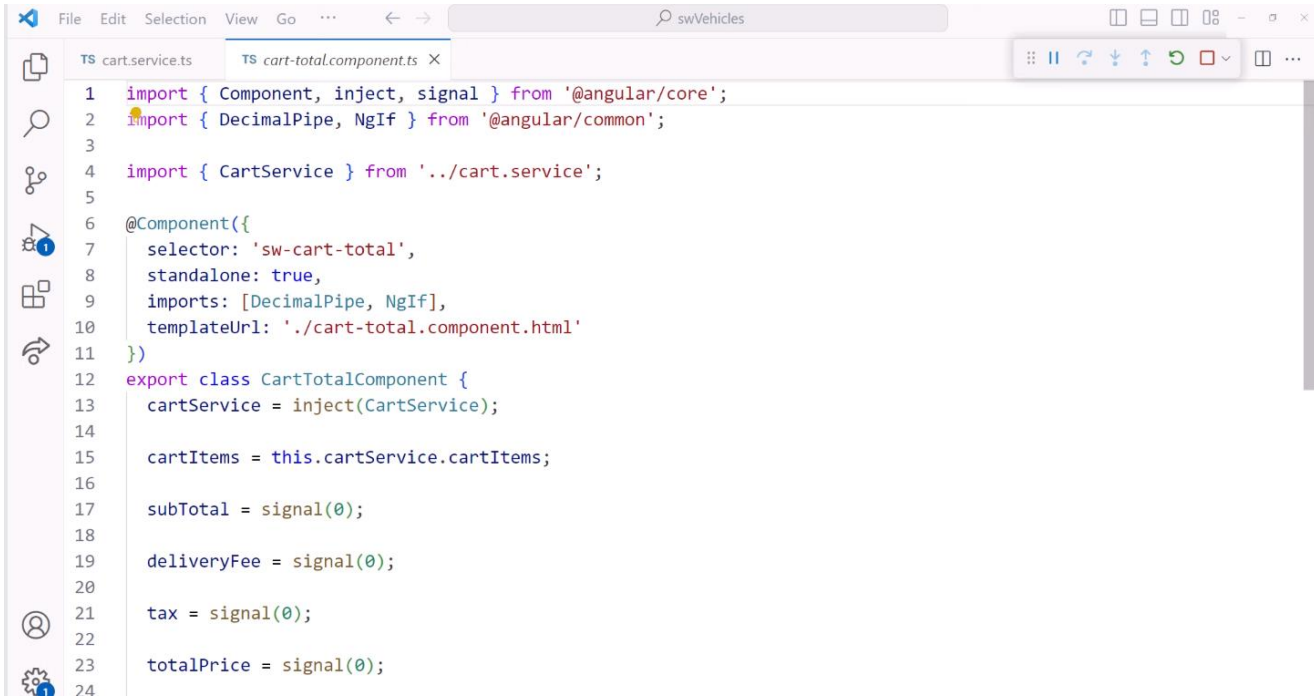
```
9
10 // Manage state with signals
11 cartItems = signal<CartItem[]>([]);
12
13 subTotal = computed(() => this.cartItems().reduce((acc, item) =>
14   acc + (item.quantity * Number(item.vehicle.cost_in_credits)), 0));
15
16 deliveryFee = computed(() => this.subTotal() < 100000 ? 999 : 0);
17
18 tax = computed(() => Math.round(this.subTotal() * 10.75) /100);
19
20 totalPrice = computed(() => this.subTotal() + this.deliveryFee() + this.tax());
21
22 // Add the vehicle to the cart
23 addToCart(vehicle: Vehicle): void {
24   this.cartItems.update(items => [...items, { vehicle, quantity: 1}]);
25 }
26
27 // Remove the item from the cart
28 removeFromCart(cartItem: CartItem): void {
29   this.cartItems.update(items => items.filter(item =>
30     item.vehicle.name !== cartItem.vehicle.name));
31 }
32
```

Next, we need to reference this `totalPrice` `computed()` signal in the Cart component as below



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure for 'SWVEHICLES'. The file 'TS cart-total.component.ts' is selected. The main editor window shows the code for this file, which includes imports from '@angular/core' and '@angular/common', an import from '../cart.service', and an @Component decorator with a selector 'sw-cart-total'. The component class 'CartTotalComponent' is defined with properties for cartService, cartItems, subTotal, deliveryFee, tax, and totalPrice, all initialized with signal(0).

```
1 import { Component, inject, signal } from '@angular/core';
2 import { DecimalPipe, NgIf } from '@angular/common';
3
4 import { CartService } from '../cart.service';
5
6 @Component({
7   selector: 'sw-cart-total',
8   standalone: true,
9   imports: [DecimalPipe, NgIf],
10  templateUrl: './cart-total.component.html'
11 })
12 export class CartTotalComponent {
13   cartService = inject(CartService);
14
15   cartItems = this.cartService.cartItems;
16
17   subTotal = signal(0);
18
19   deliveryFee = signal(0);
20
21   tax = signal(0);
22
23   totalPrice = signal(0);
24 }
```

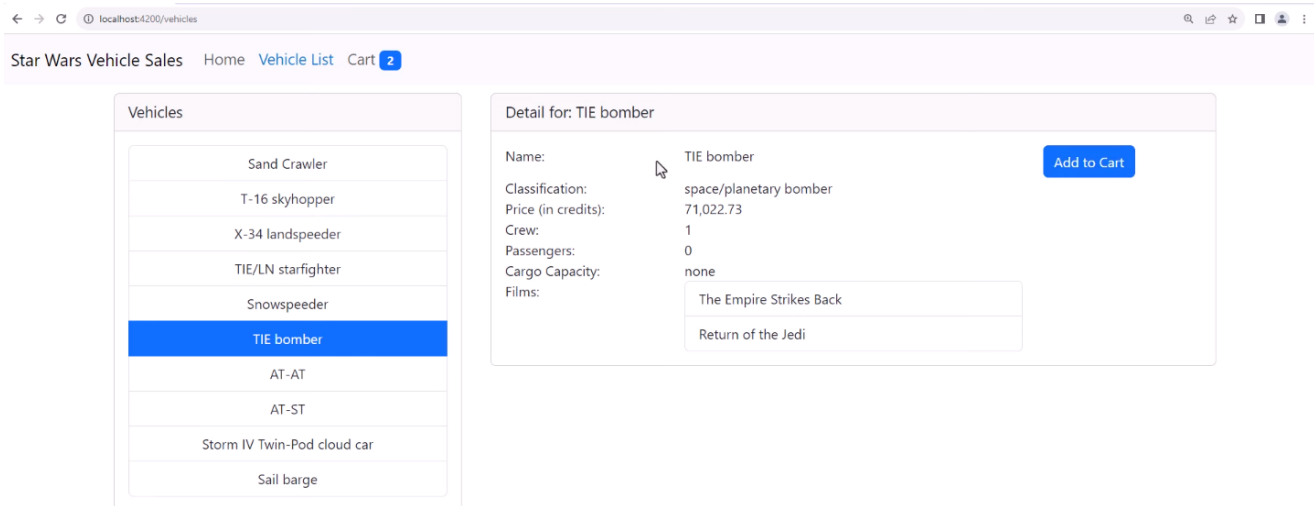
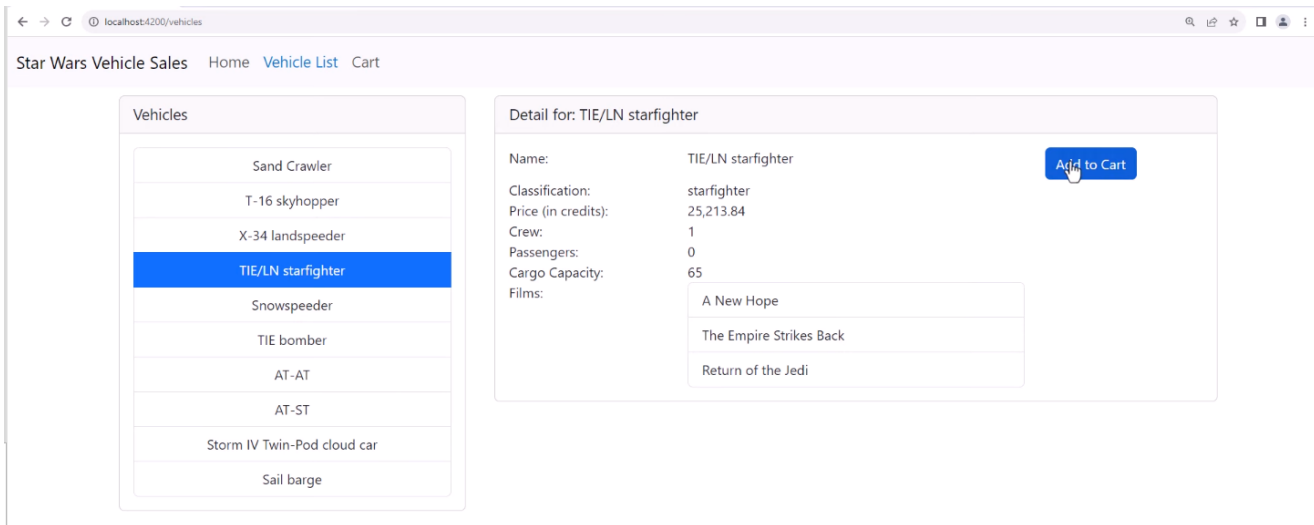


This screenshot shows the same code as the previous one, but with the Explorer sidebar collapsed. The code in the editor is identical, showing the imports, the @Component decorator, and the CartTotalComponent class with its properties and signal-based values.

```
1 import { Component, inject, signal } from '@angular/core';
2 import { DecimalPipe, NgIf } from '@angular/common';
3
4 import { CartService } from '../cart.service';
5
6 @Component({
7   selector: 'sw-cart-total',
8   standalone: true,
9   imports: [DecimalPipe, NgIf],
10  templateUrl: './cart-total.component.html'
11 })
12 export class CartTotalComponent {
13   cartService = inject(CartService);
14
15   cartItems = this.cartService.cartItems;
16
17   subTotal = signal(0);
18
19   deliveryFee = signal(0);
20
21   tax = signal(0);
22
23   totalPrice = signal(0);
24 }
```

Let' update as below to use the signals

```
File Edit Selection View Go ... swVehicles
TS cart.service.ts TS cart-total.component.ts X
1 import { Component, inject, signal } from '@angular/core';
2 import { DecimalPipe, NgIf } from '@angular/common';
3
4 import { CartService } from '../cart.service';
5
6 @Component({
7   selector: 'sw-cart-total',
8   standalone: true,
9   imports: [DecimalPipe, NgIf],
10  templateUrl: './cart-total.component.html'
11 })
12 export class CartTotalComponent {
13   cartService = inject(CartService);
14
15   cartItems = this.cartService.cartItems;
16
17   subTotal = this.cartService.subTotal;
18
19   deliveryFee = this.cartService.deliveryFee;
20
21   tax = this.cartService.tax;
22
23   totalPrice = this.cartService.totalPrice;
24 }
```



← → ↻ localhost:4200/cart


Star Wars Vehicle Sales Home Vehicle List Cart 2

Cart

Product	Class	Price (credits)	Quantity	Extended Price	
TIE/LN starfighter	starfighter	25,213.84	1 ▾	25,213.84	Delete
TIE bomber	space/planetary bomber	71,022.73	1 ▾	71,022.73	Delete

Cart Total

Subtotal:	96,236.57
Delivery:	999.00
Estimated Tax:	10,345.43
Total:	107,581.00



← → ↻ localhost:4200/cart

Star Wars Vehicle Sales Home Vehicle List Cart 2

Cart

Product	Class	Price (credits)	Quantity	Extended Price	
TIE/LN starfighter	starfighter	25,213.84	1 ▾	25,213.84	Delete
TIE bomber	space/planetary bomber	71,022.73	1 ▾	71,022.73	Delete

Cart Total

Subtotal:	96,236.57
Delivery:	999.00
Estimated Tax:	10,345.43
Total:	107,581.00

Quantity dropdown menu

--Select a quantity--
1
2
3
4
5
6
7
8

← → ↻ localhost:4200/cart


Star Wars Vehicle Sales Home Vehicle List Cart 3

Cart

Product	Class	Price (credits)	Quantity	Extended Price	
TIE/LN starfighter	starfighter	25,213.84	2 ▾	50,427.68	Delete
TIE bomber	space/planetary bomber	71,022.73	1 ▾	71,022.73	Delete

Cart Total

Subtotal:	121,450.42
Delivery:	Free
Estimated Tax:	13,055.92
Total:	134,506.34



← → ↻ localhost:4200/cart

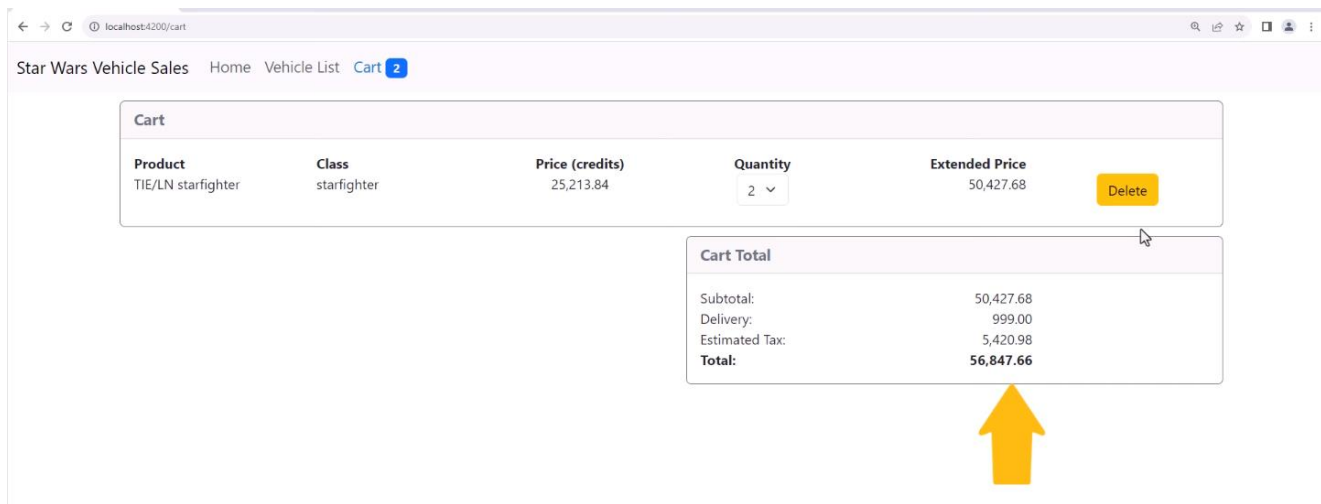
Star Wars Vehicle Sales Home Vehicle List Cart 3

Cart

Product	Class	Price (credits)	Quantity	Extended Price	
TIE/LN starfighter	starfighter	25,213.84	2 ▾	50,427.68	Delete
TIE bomber	space/planetary bomber	71,022.73	1 ▾	71,022.73	Delete

Cart Total

Subtotal:	121,450.42
Delivery:	Free
Estimated Tax:	13,055.92
Total:	134,506.34



With Angular's `computed()` signals, our data automatically reacts to changes in other signals without needing to handle events or create subjects!!!