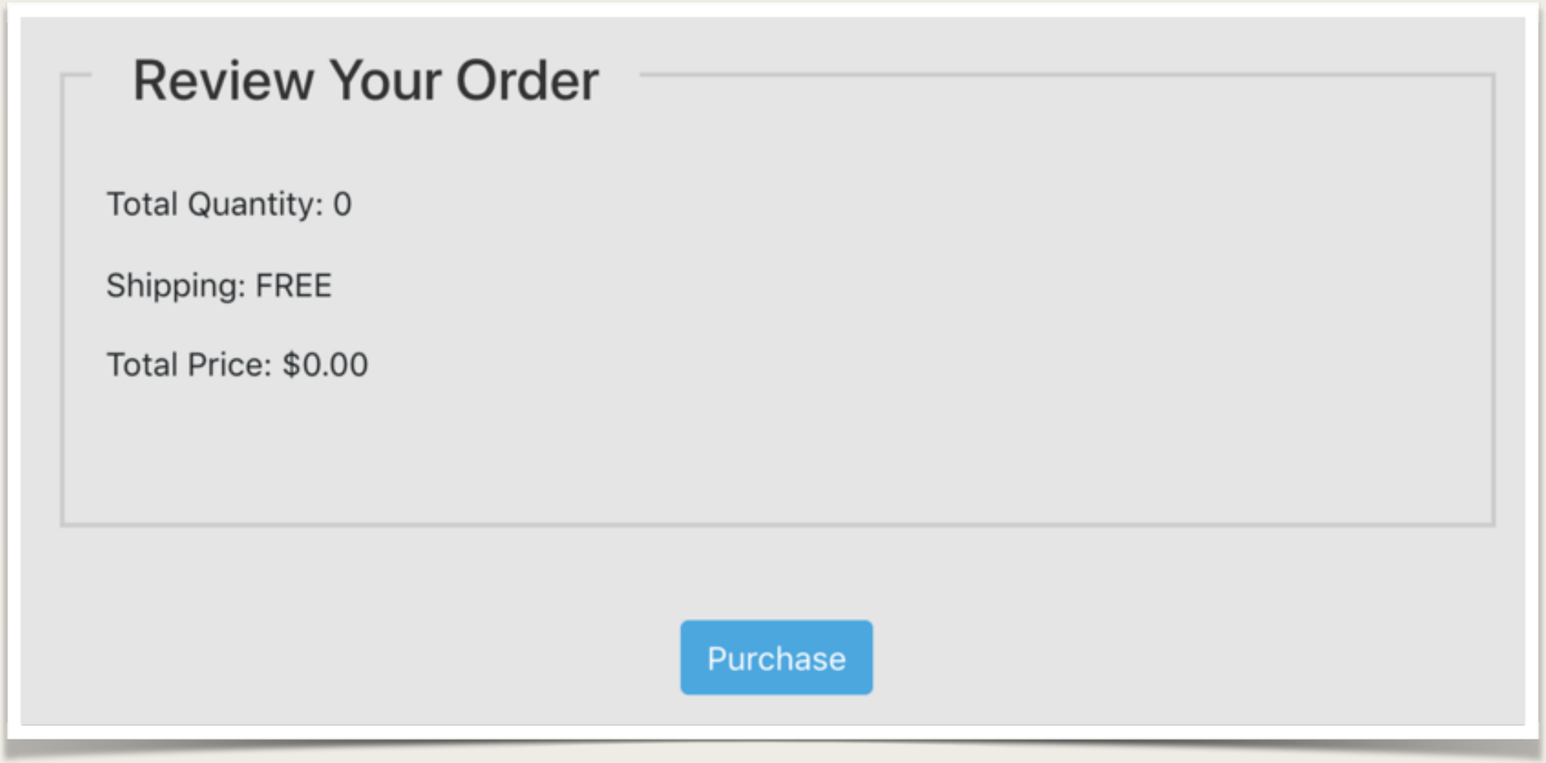
Checkout Form
Review Cart Totals

Update Checkout Form - Cart Totals

- Our form currently has a section for cart totals
- We need to add the appropriate code to support this

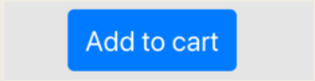


Cart Service - Publishing messages/events

- Recall that we send messages/events to other components in our application
- For example, `CartStatusComponent` will **subscribe** to the `CartService`
- `CartService` will **publish** messages for:
  - `totalPrice` and `totalQuantity`

Application Interaction

Remember



ProductListComponent

CartStatusComponent

1. subscribe for events

2. addToCart(...)

3. publish events to all subscribers

4. update UI for total price and quantity

CartService

Remember

CartService

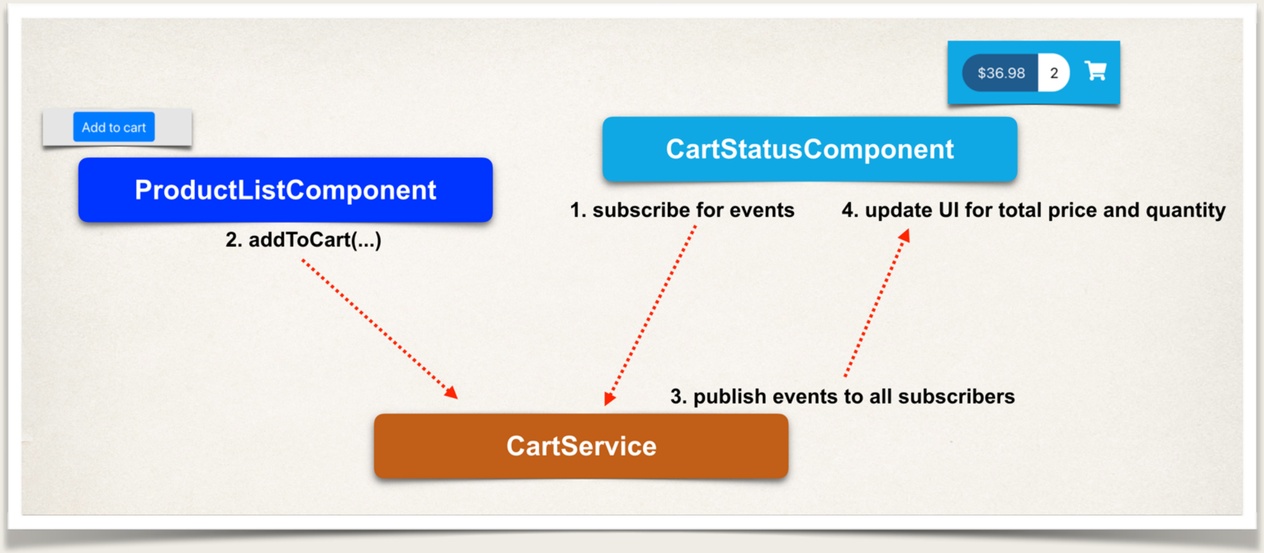
File: cart-service.ts

```
export class CartService {

  cartItems: CartItem[] = [];

  totalPrice: Subject<number> = new Subject<number>();
  totalQuantity: Subject<number> = new Subject<number>();

  ...
}
```

`Subject` is a subclass of `Observable`

We can use `Subject` to publish events in our code.

The event will be sent to all of the subscribers



Remember

CartService - Publish events

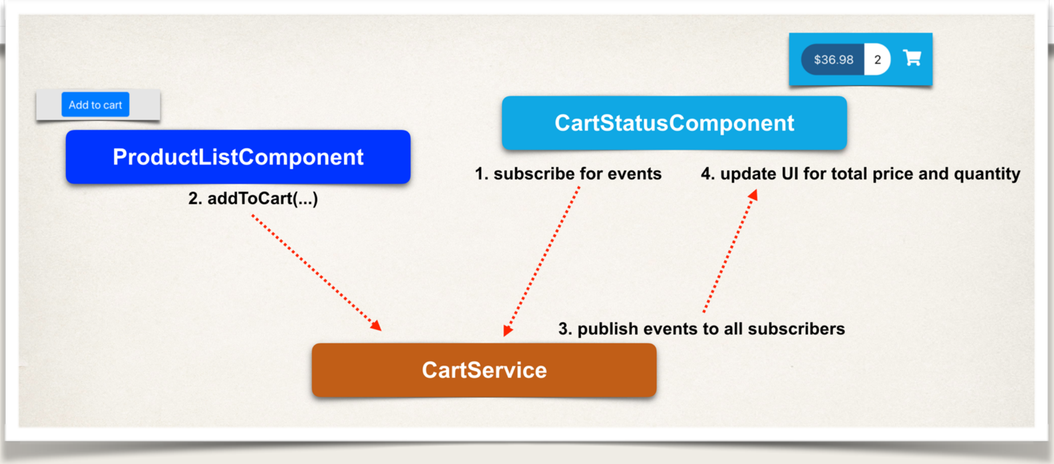
File: cart-service.ts

```
computeCartTotals() {

  let totalPriceValue: number = 0;
  let totalQuantityValue: number = 0;

  for (let currentCartItem of this.cartItems) {
    totalPriceValue += currentCartItem.quantity * currentCartItem.unitPrice;
    totalQuantityValue += currentCartItem.quantity;
  }

  // publish the new values ... all subscribers will receive the new data
  this.totalPrice.next(totalPriceValue);
  this.totalQuantity.next(totalQuantityValue);
}
```

Compute totals

This will publish events to all subscribers

one event for totalPrice
one event for totalQuantity



Remember

CartStatusComponent subscribes to CartService

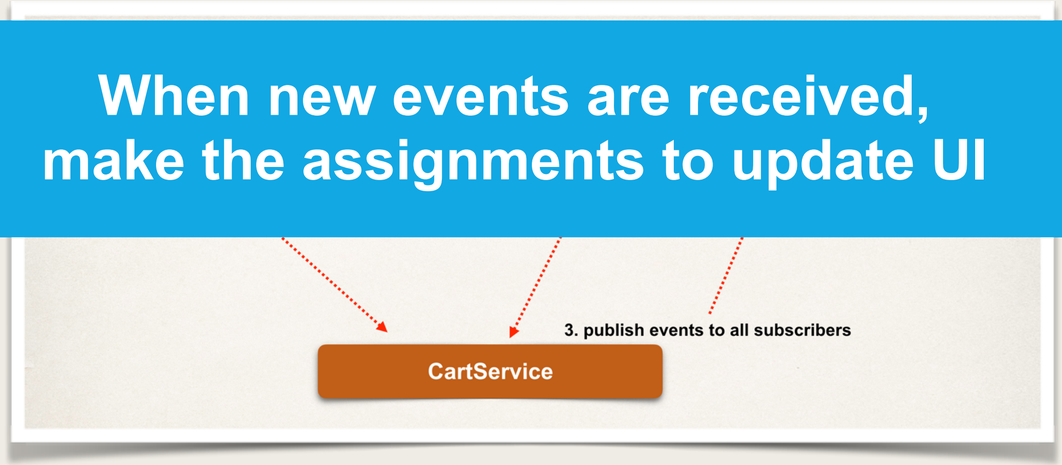
File: cart-status.component.ts

```
export class CartStatusComponent implements OnInit {

  totalPrice: number = 0.00;
  totalQuantity: number = 0;

  constructor(private cartService: CartService) { }

  ngOnInit(): void {
    this.updateCartStatus();
  }

  updateCartStatus() {

    // subscribe to the cart status totalPrice
    this.cartService.totalPrice.subscribe(
      data => this.totalPrice = data
    );

    // subscribe to the cart status totalQuantity
    this.cartService.totalQuantity.subscribe(
      data => this.totalQuantity = data
    );
  }
}
```

Inject the CartService

Subscribe for events

When new events are received, make the assignments to update UI

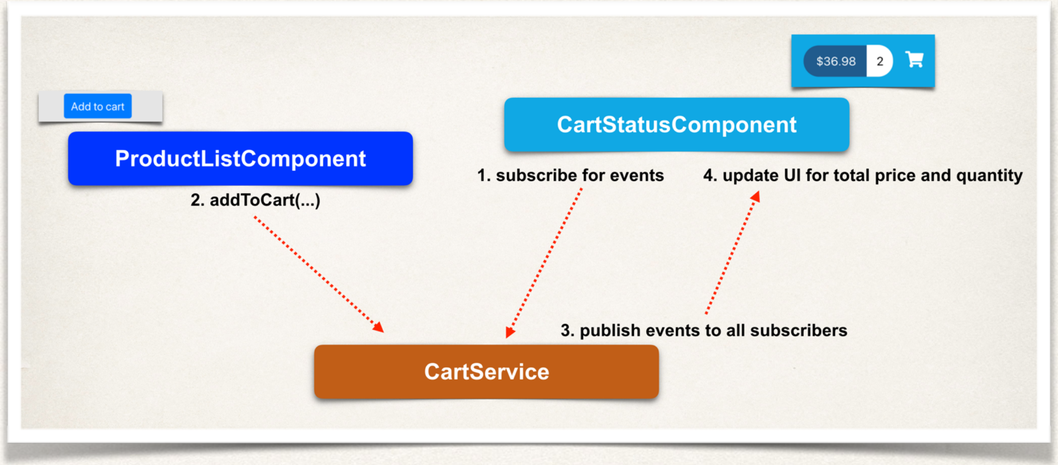


Remember

CartStatusComponent subscribes to CartService

File: cart-status.component.ts

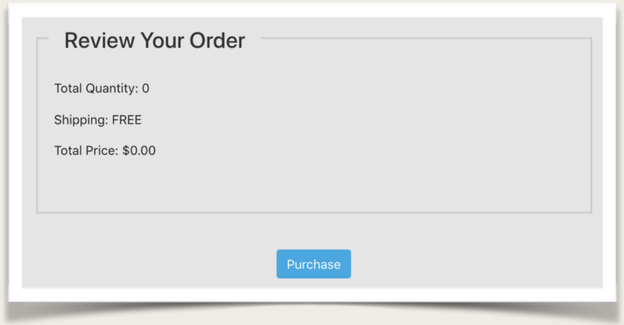
```
export class CartStatusComponent implements OnInit {

  totalPrice: number = 0.00;
  totalQuantity: number = 0;

  constructor(private cartService: CartService) { }

  ngOnInit(): void {
    this.updateCartStatus();
  }

  updateCartStatus() {

    // subscribe to the cart status totalPrice
    this.cartService.totalPrice.subscribe(
      data => this.totalPrice = data
    );

    // subscribe to the cart status totalQuantity
    this.cartService.totalQuantity.subscribe(
      data => this.totalQuantity = data
    );
  }
}
```



When new events are received, make the assignments to update UI

Publish / Subscribe

- Similar approach for `CheckoutComponent` ... almost
- `CheckoutComponent` will subscribe to events from `CartService`
- However, since `CheckoutComponent` is instantiated later in the application
  - Will miss out on previous messages
- As a result, `CheckoutComponent` cart totals will erroneously show as
  - `0` for total quantity
  - `0.00` for total price



Need to get a replay of the messages missed

Replay Messages

- Similar to the real world

Sorry, I am late to the meeting.
Can you tell me what I missed?

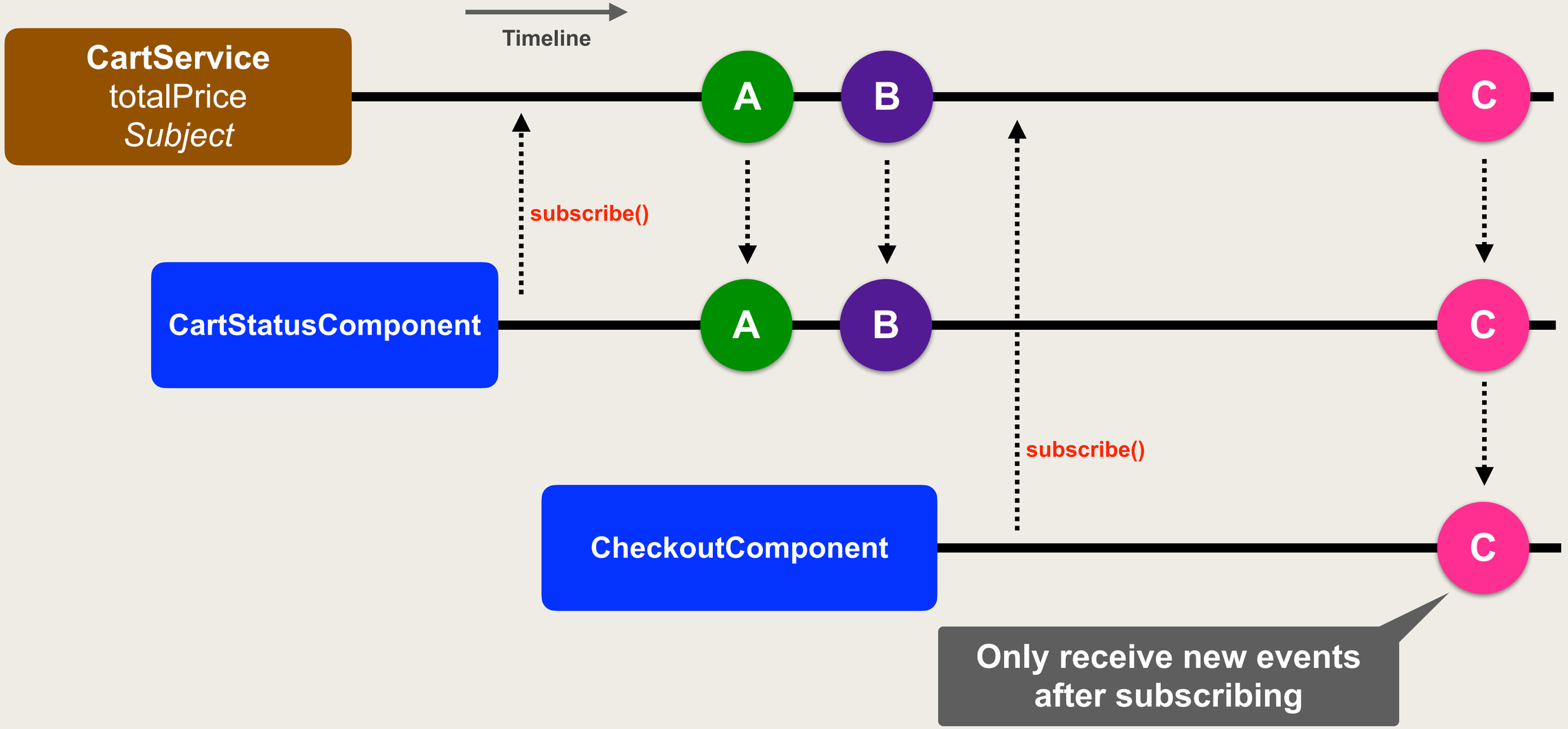
- We can get this functionality with `ReplaySubject`

ReplaySubject

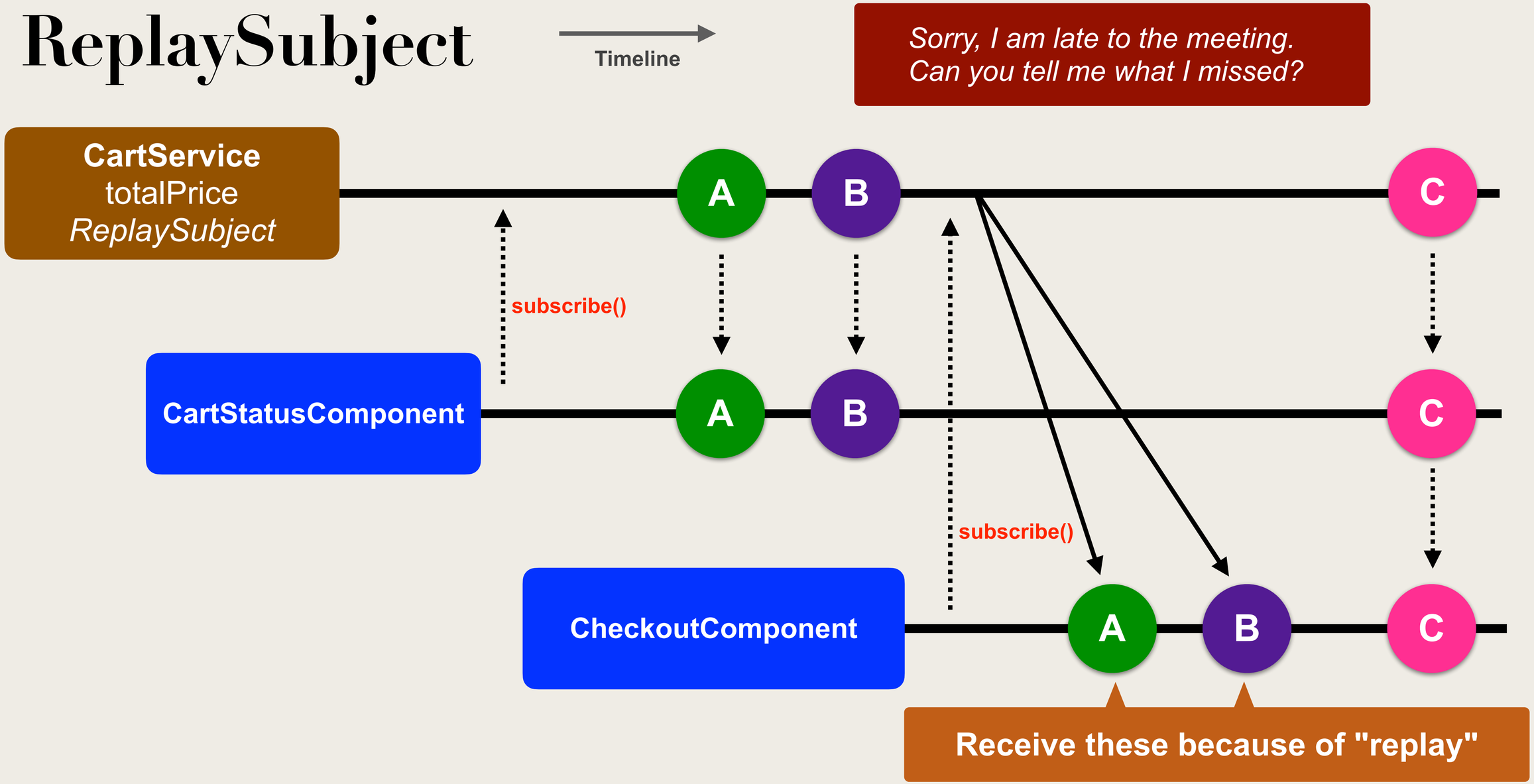
- Recall, `Subject` is used to send events to subscribers
- **`ReplaySubject`** is a subclass of `Subject`
  - Will also "replay events" for new subscribers who join later
  - Keep a buffer of previous events ... and send to **new subscribers**

Sorry, I am late to the meeting.
Can you tell me what I missed?

Subject (without replay)



ReplaySubject



ReplaySubject

```
export class CartService {

  cartItems: CartItem[] = [];

  totalPrice: Subject<number> = new ReplaySubject<number>();
  totalQuantity: Subject<number> = new ReplaySubject<number>();

  ...
}
```

Keep a buffer of previous events
Send previous events to new subscribers

ReplaySubject

- For more details on `ReplaySubject`, see online docs

http://www.luv2code.com/rxjs-replay-subject