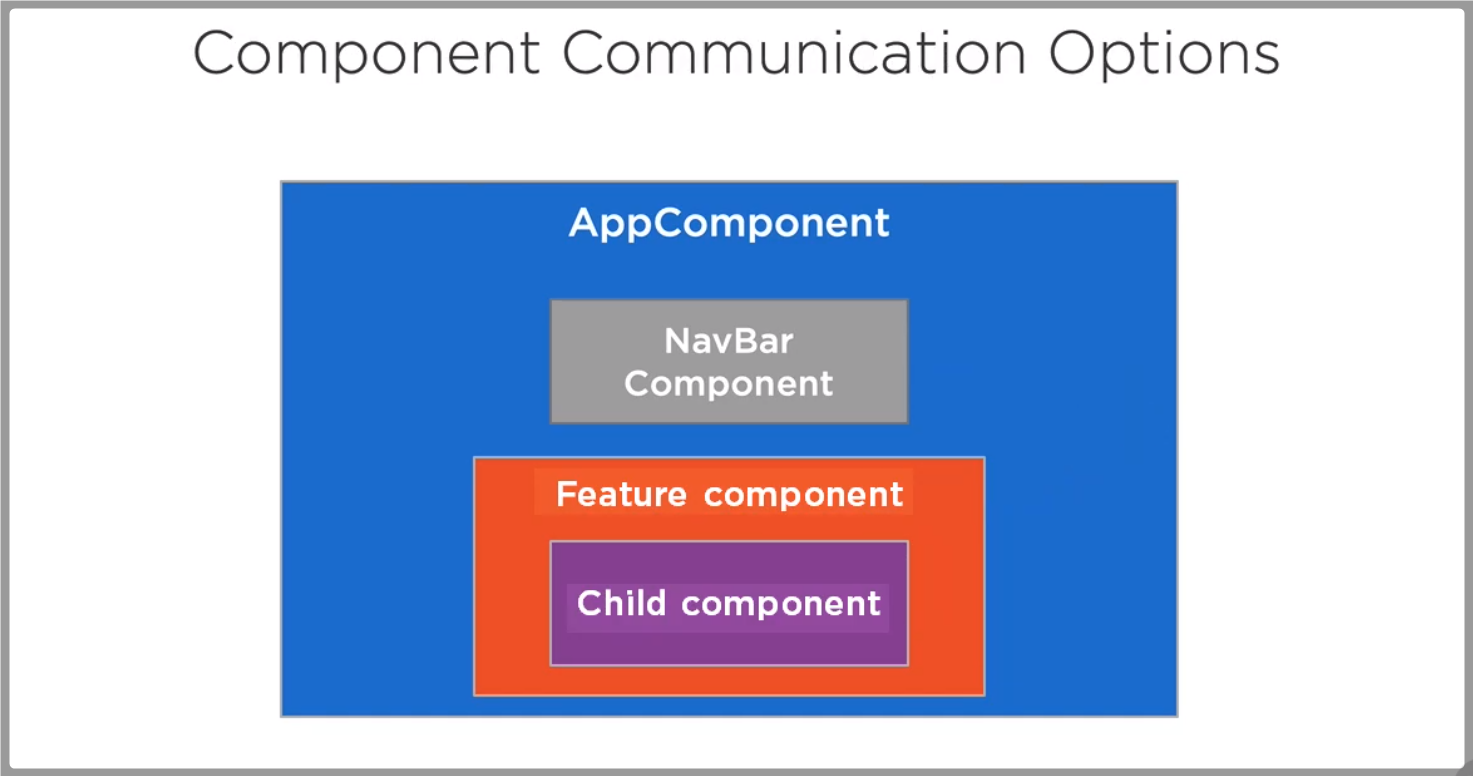
**COMPONENT COMMUNICATION**

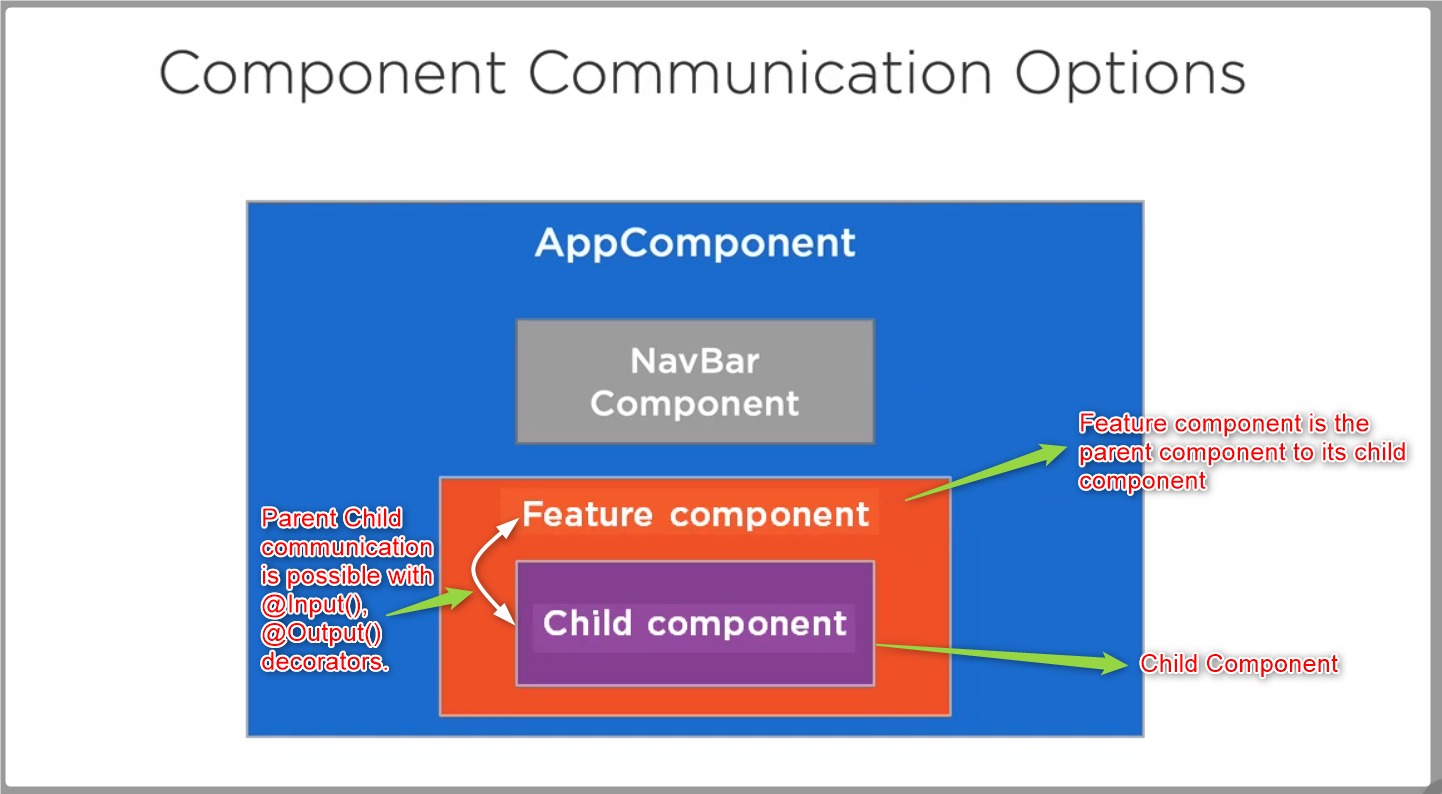
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Communication between the Components in Angular will help you to pass data from child components to parent components and vice-versa and pass data from one component to some other components as well which are not related to each other.**

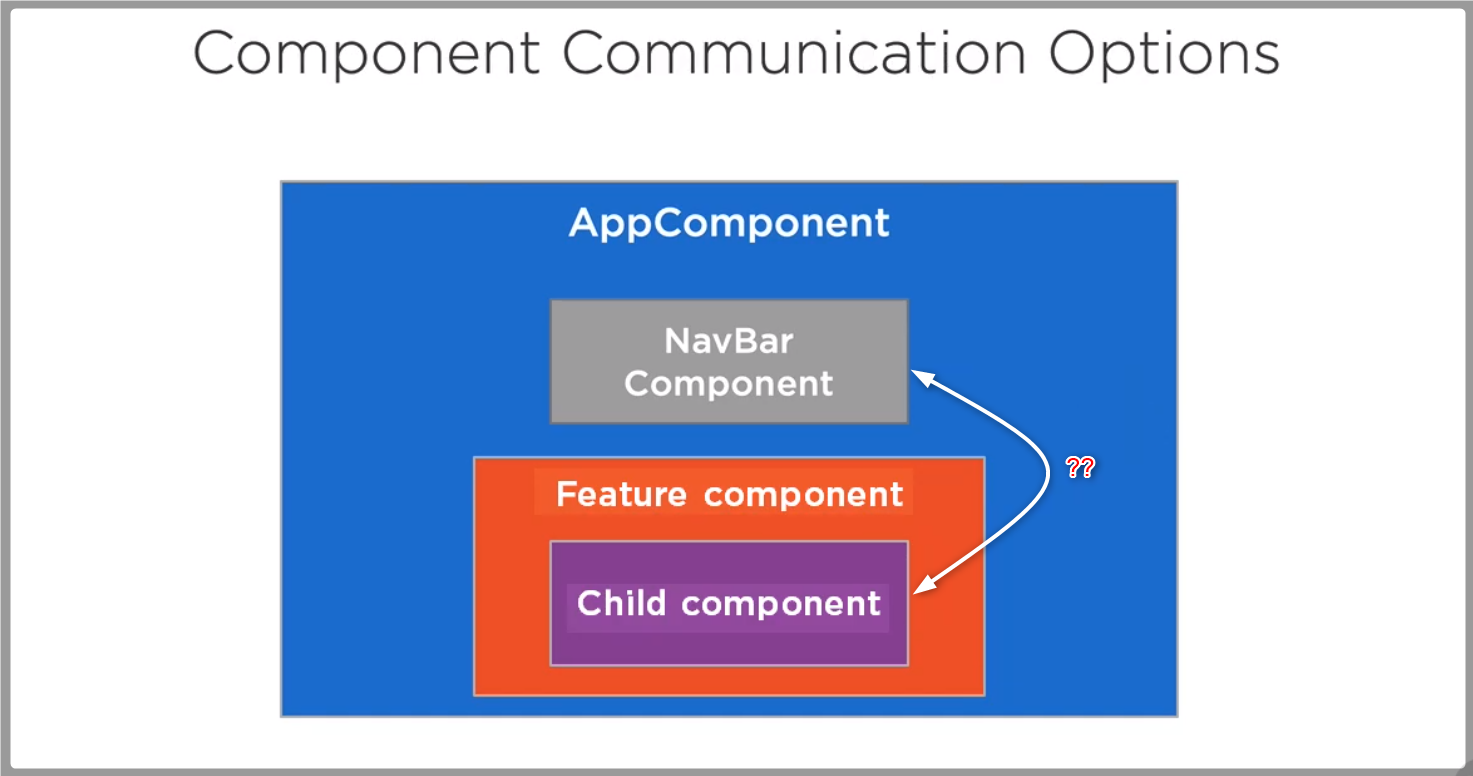
**Ex.**

****

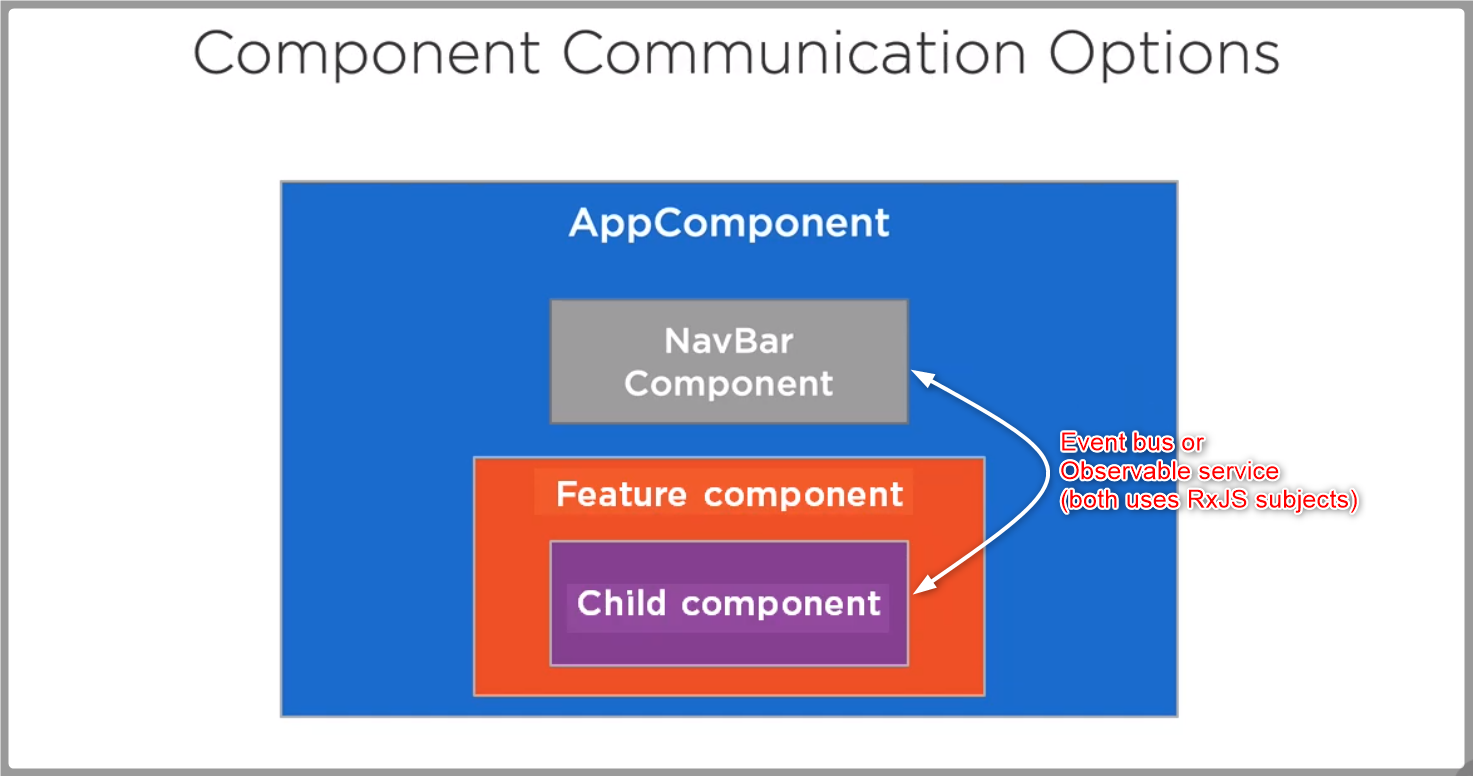
**See above Parent Child Components are communicated using @Input(), @OutPut() decarators.**

****

**This article completely focused on communication between non related components.**

****

**As per the above screen Child Component & NavBar Component are nowhere related, how they can communicate with each other ?**

****

**Non related components are communicated by using two mechanisms.**

1. **Event bus**
2. **Observable Service**

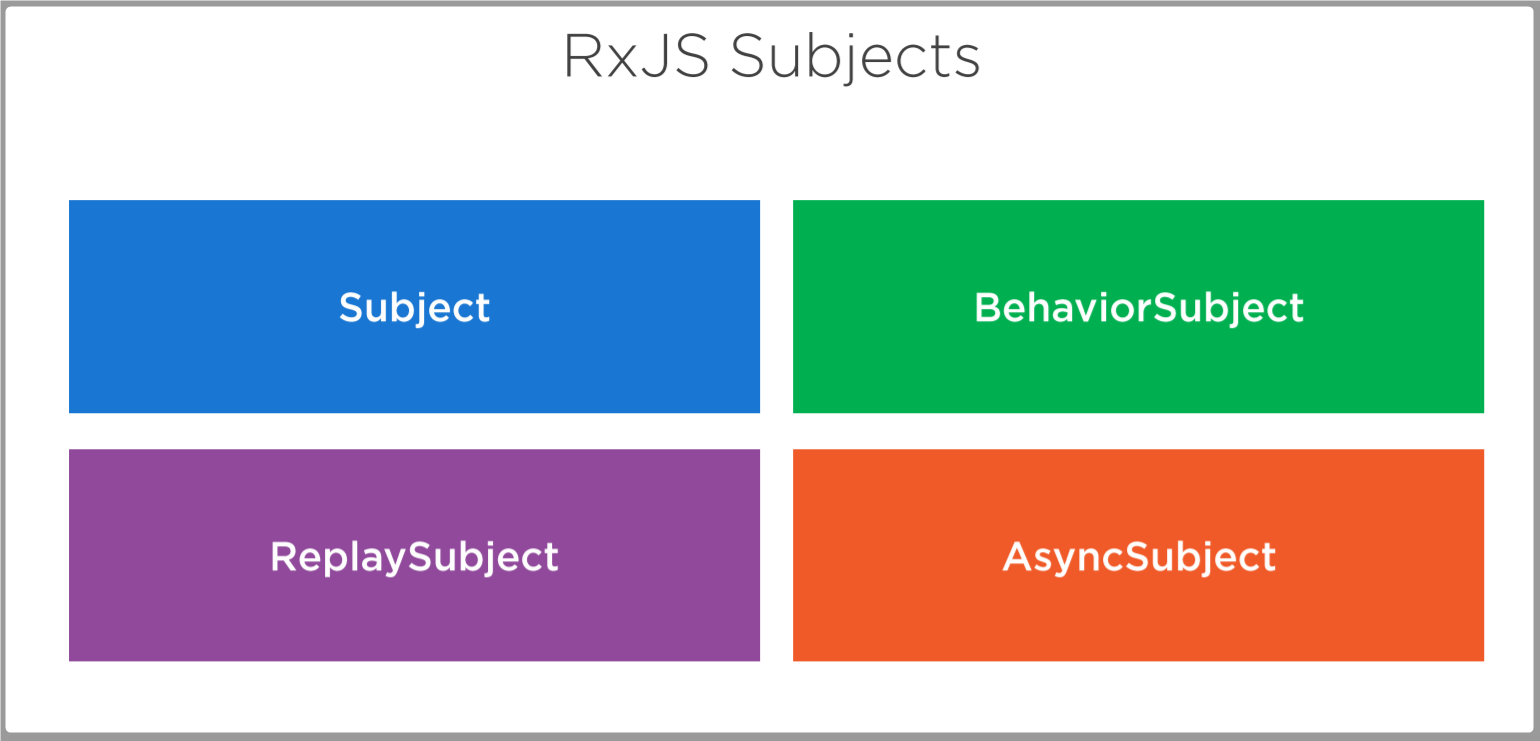
**RxJS Subjects**

RxJS is great. It helps you with composing and subscribing to data streams. RxJS offers multiple classes to use with data streams, and one of them is Subject.

A Subject is like an Observable. It can be subscribed to, just like you normally would with Observables. It also has methods like next(), error() and complete() just like the observer you normally pass to your Observable creation function.

The main reason to use Subjects is to multicast. An Observable by default is unicast. Unicasting means that each subscribed observer owns an independent execution of the Observable.

**There are 4 main types of subjects available in RxJS Subjects.**

****

**Subject: Newly subscribed observers can’t receive the old emitted values.**

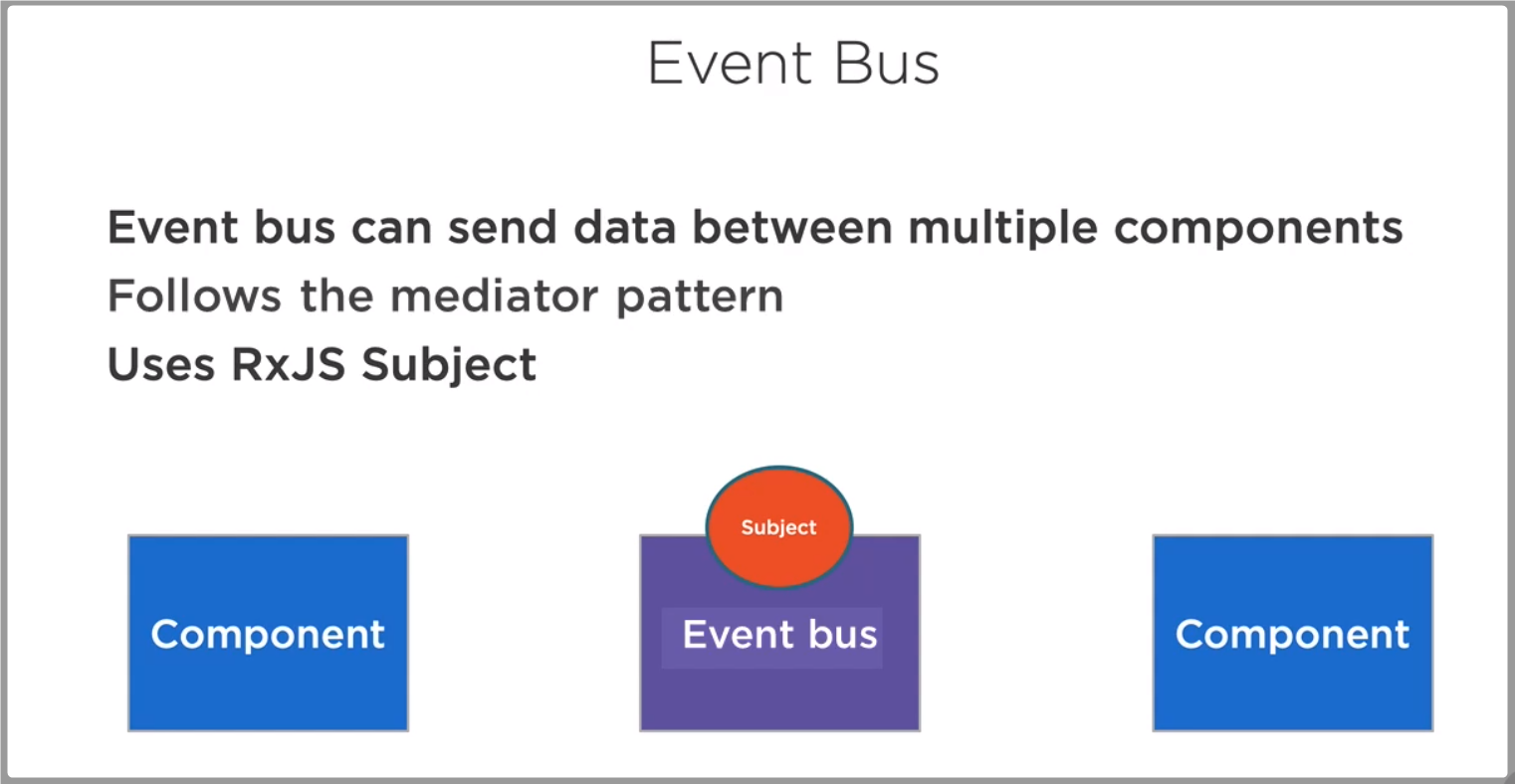
**Behavior Subject: Newly subscribed observers can receive the last emitted value.**

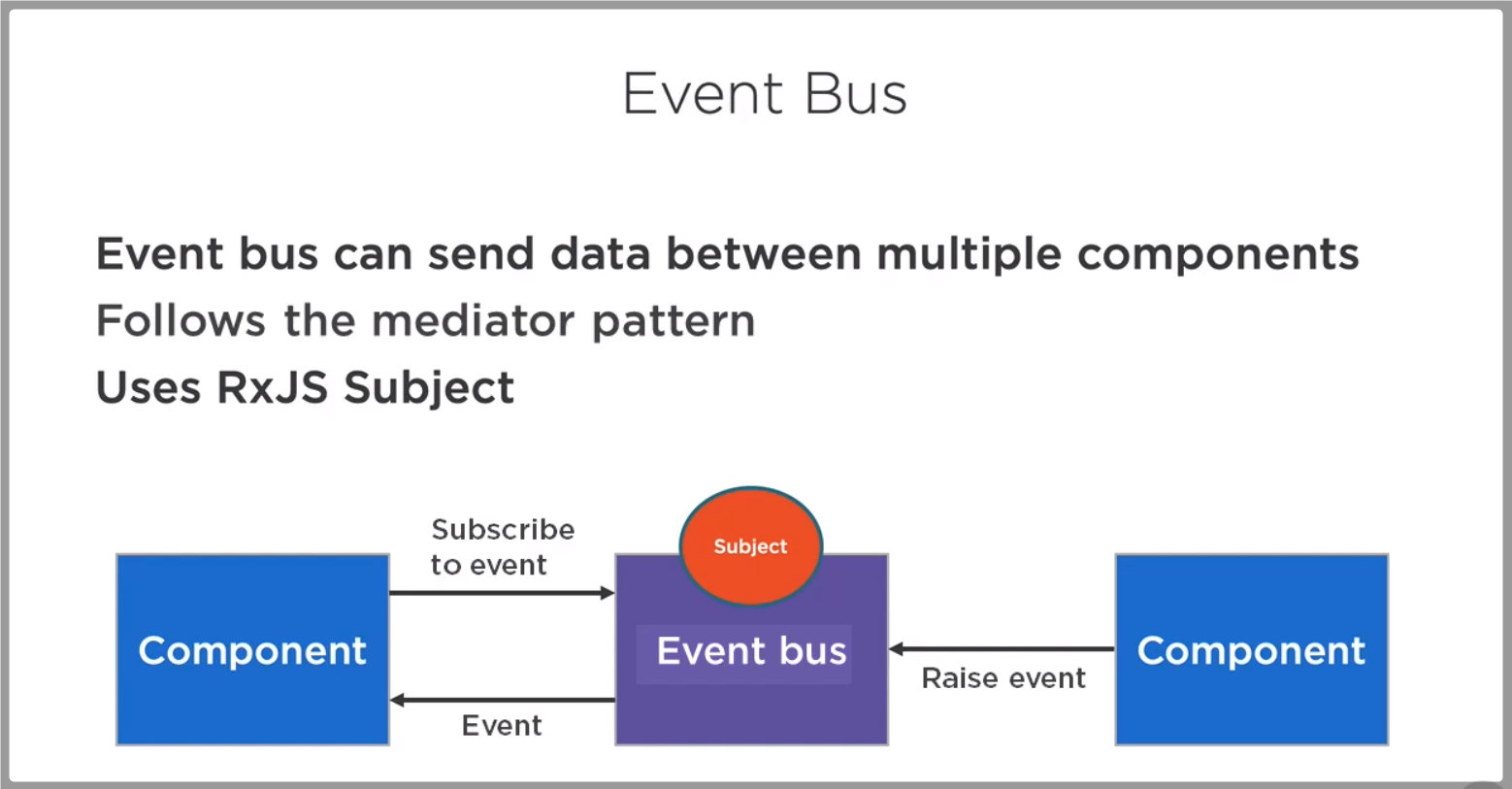
**Replay Subject: Newly subscribed observers can receive all previously emitted values.**

**Async Subject: Any subscribed observer can receive only the last value when the sequence is completed.**

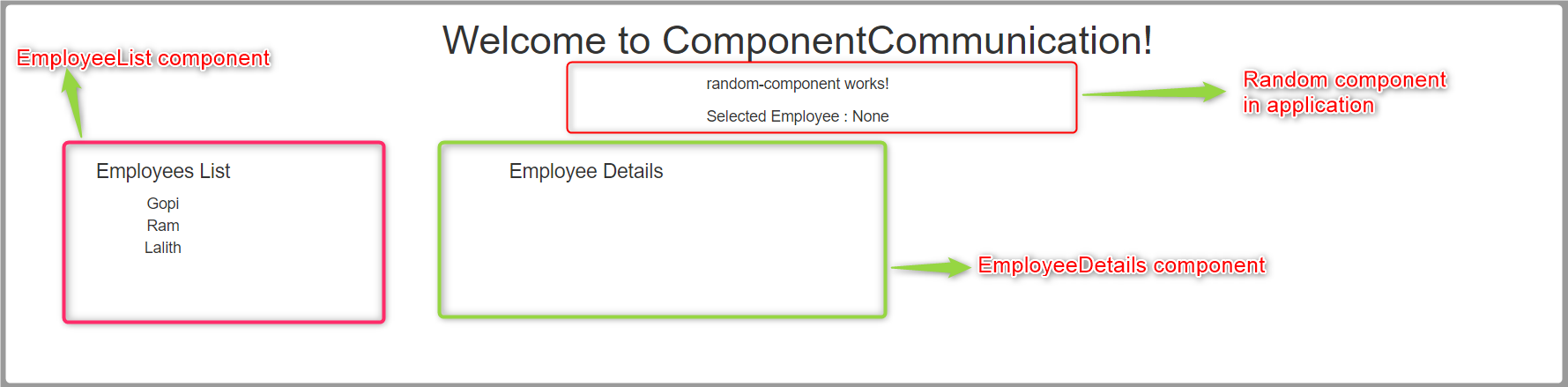
**Event Bus:**

Event Bus Service is useful to send data between multiple components by using RxJS subjects following mediator pattern.

****

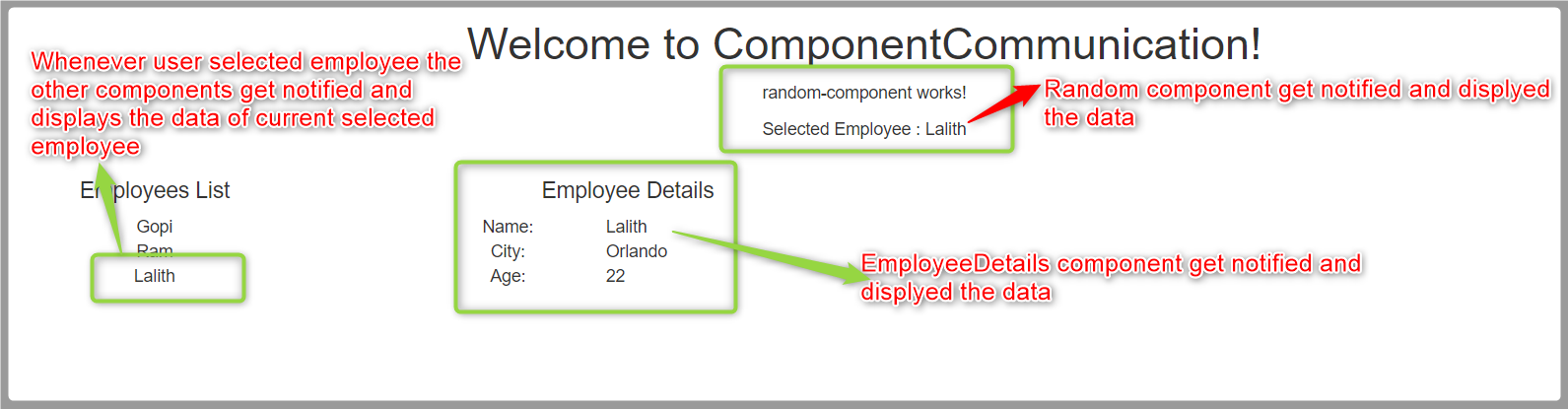
****

**Ex.**

****

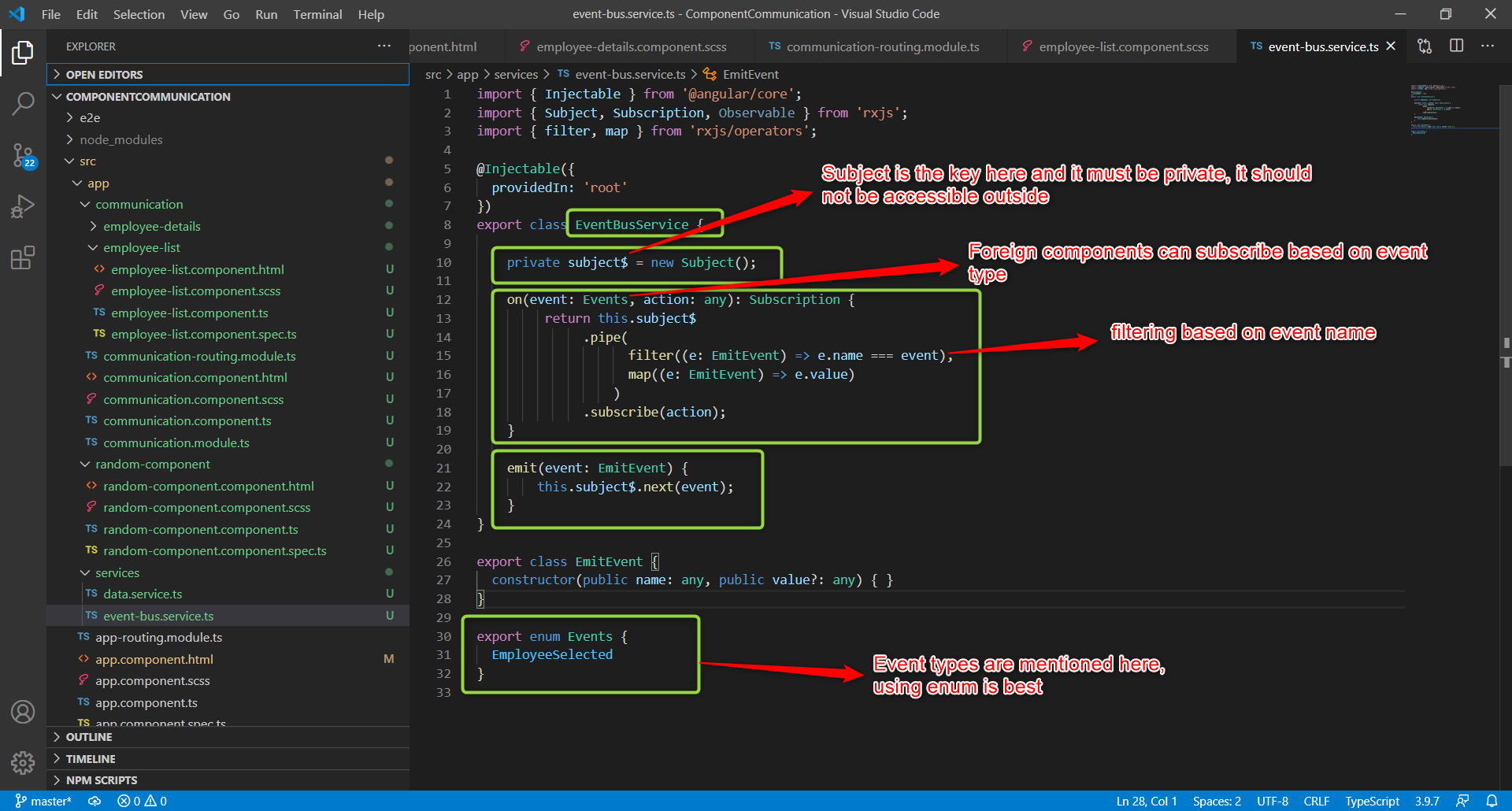
**When user selects an employee from EmployeeList component, the selected employee data needs to be communicated to other components in application by using EventBus mechanism.**

**Output screen:**

****

**Code:**

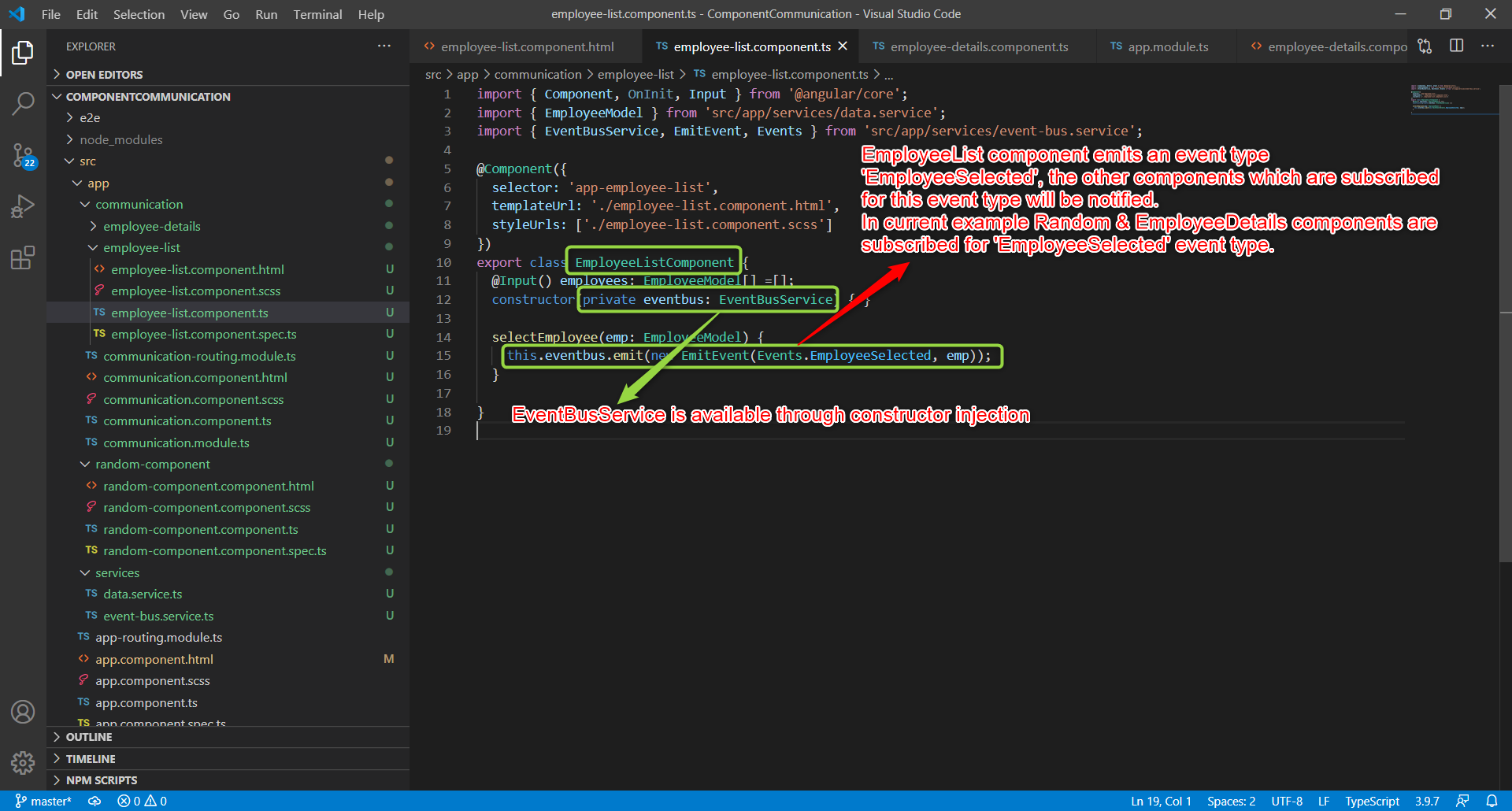
**Event Bus Service code**

****

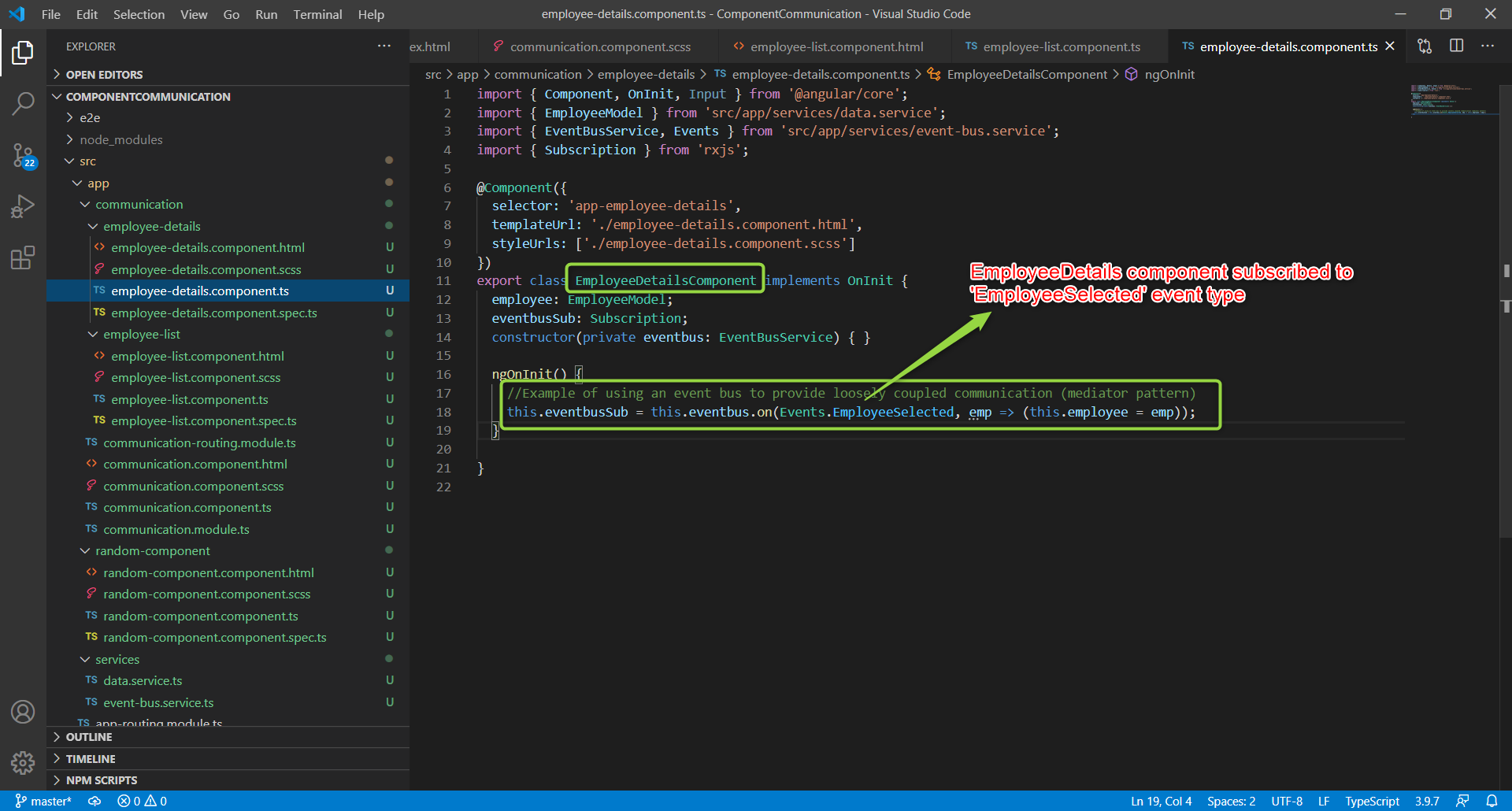
**EmployeeList Component**

EmployeeList component emits an event type 'EmployeeSelected', the other components which are subscribed for this event type will be notified.

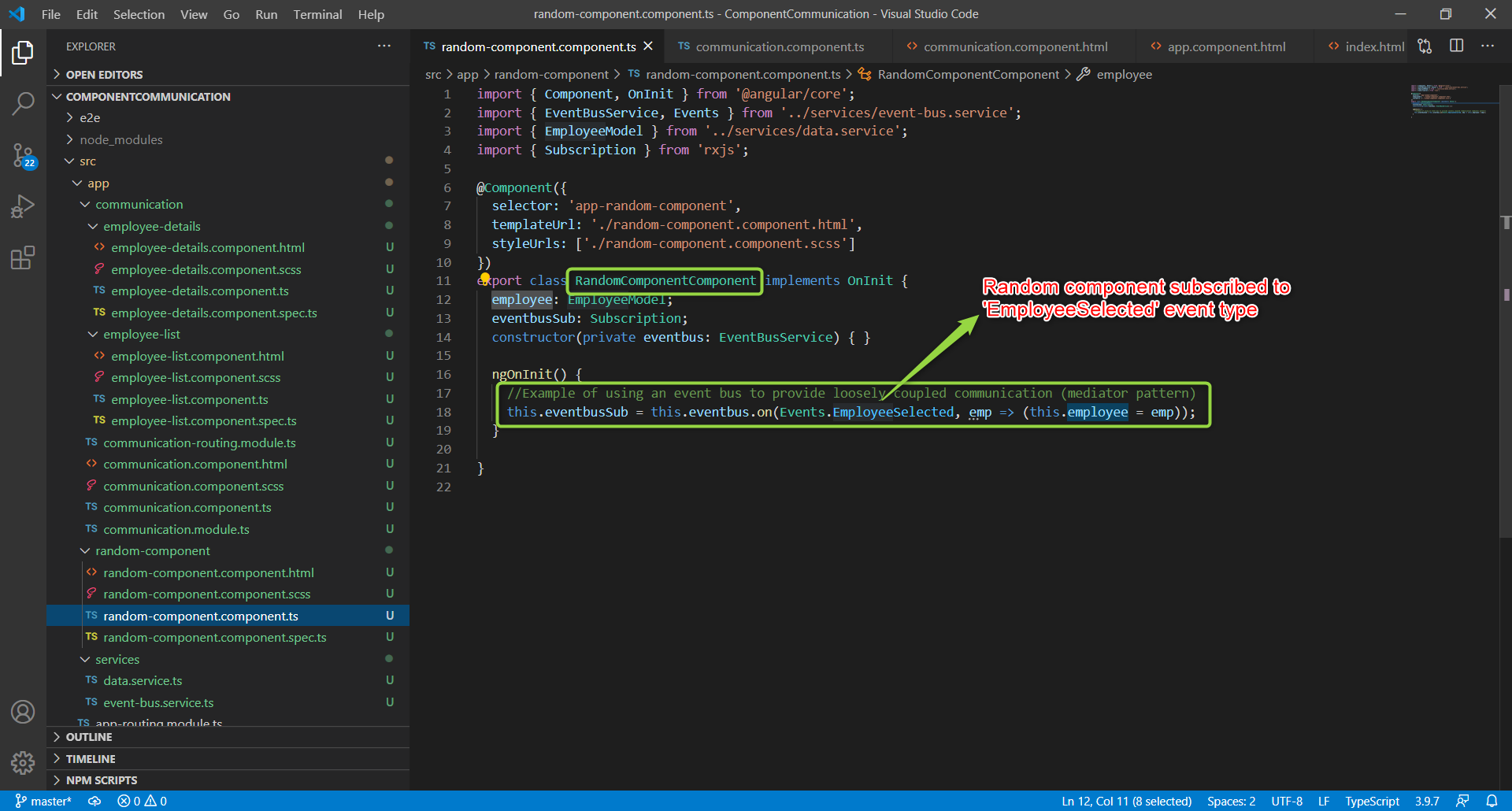
In current example Random & Employee Details components are subscribed for 'EmployeeSelected' event type.

****

**EmployeeDetails component** subscribed to EmployeeSelected event type as follows.

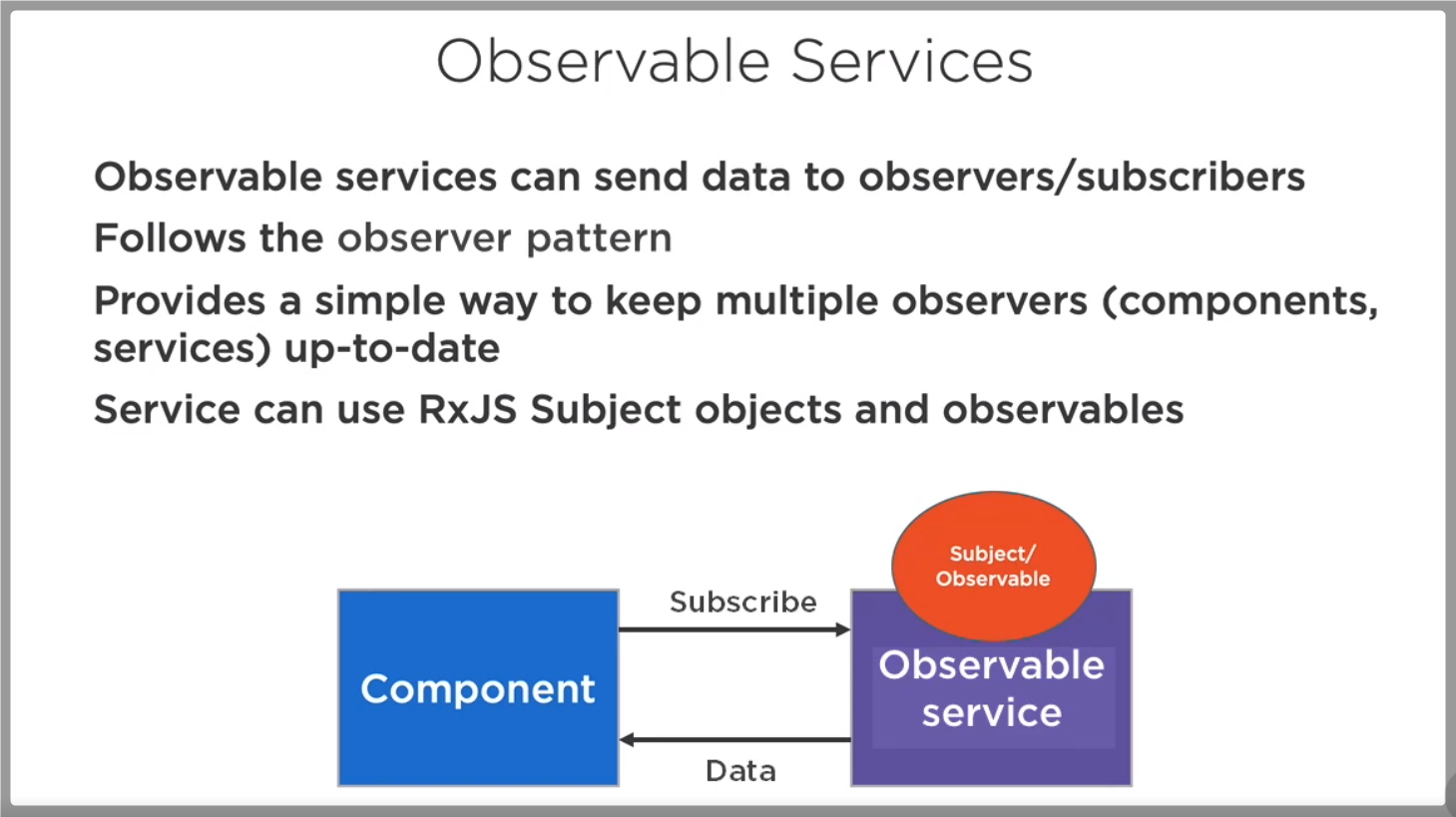
****

**Random Component**

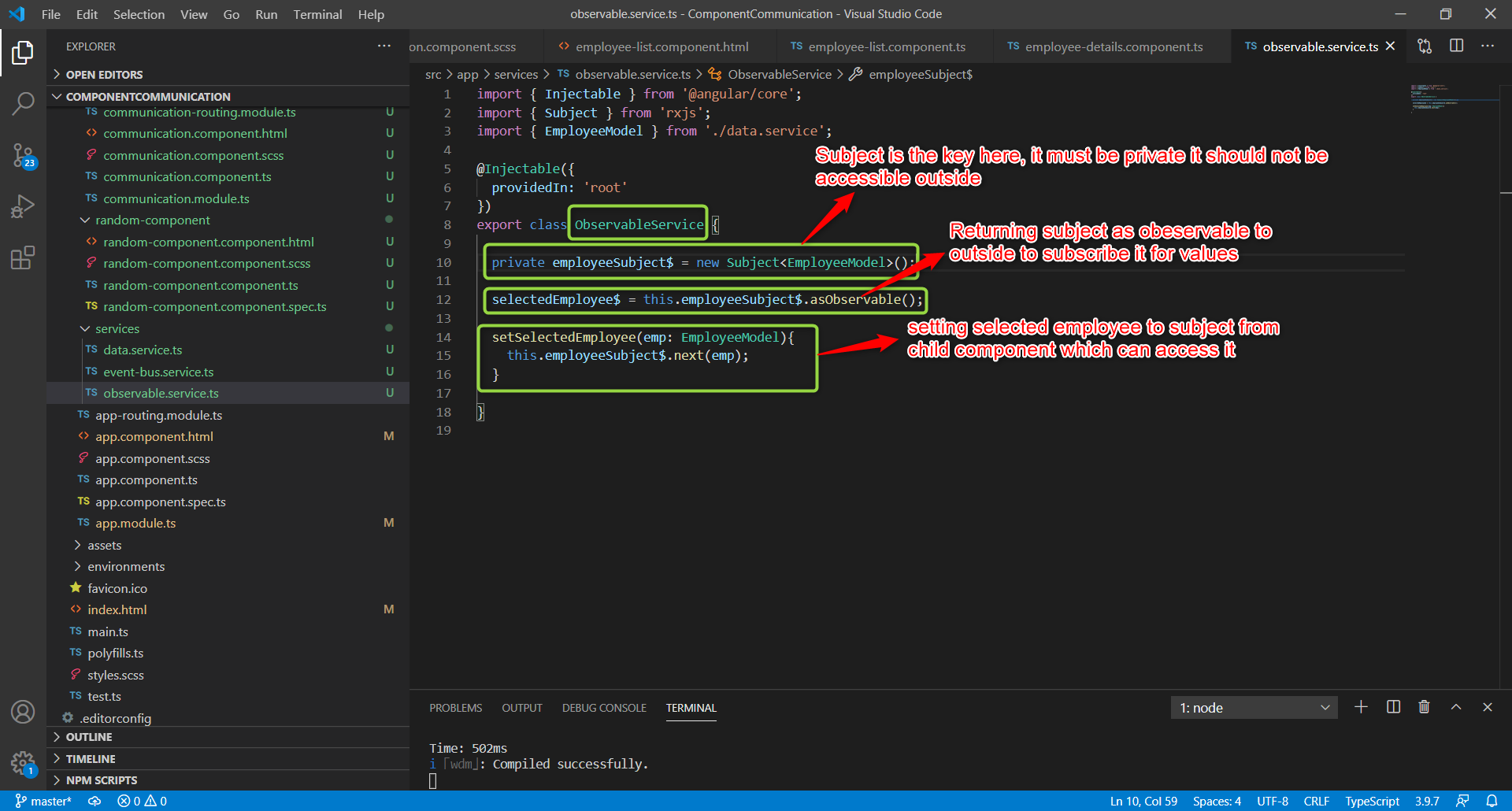
****

**Observable Service**

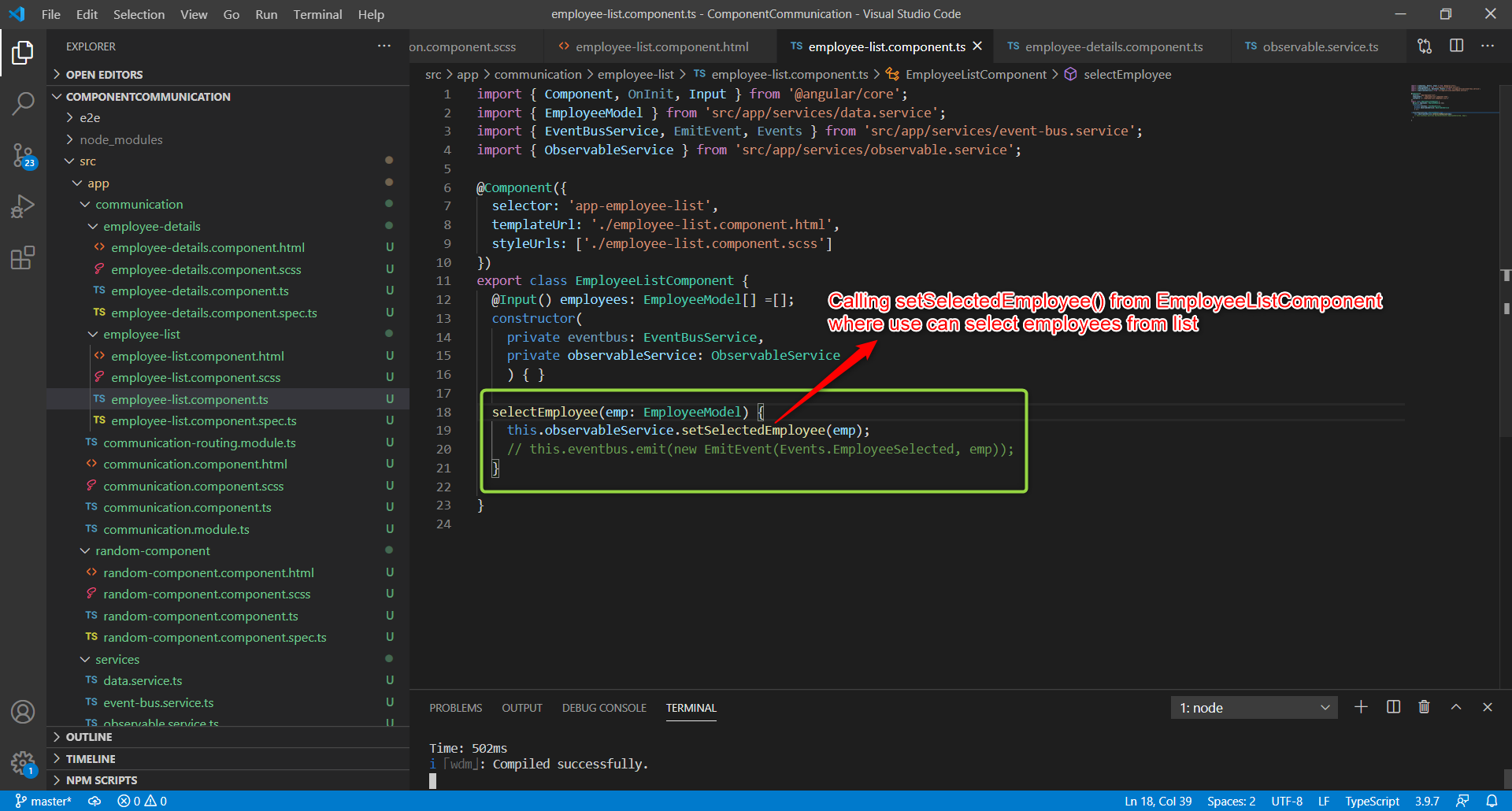
Observable Service is useful to send data between multiple components by using RxJS subjects following observer pattern.



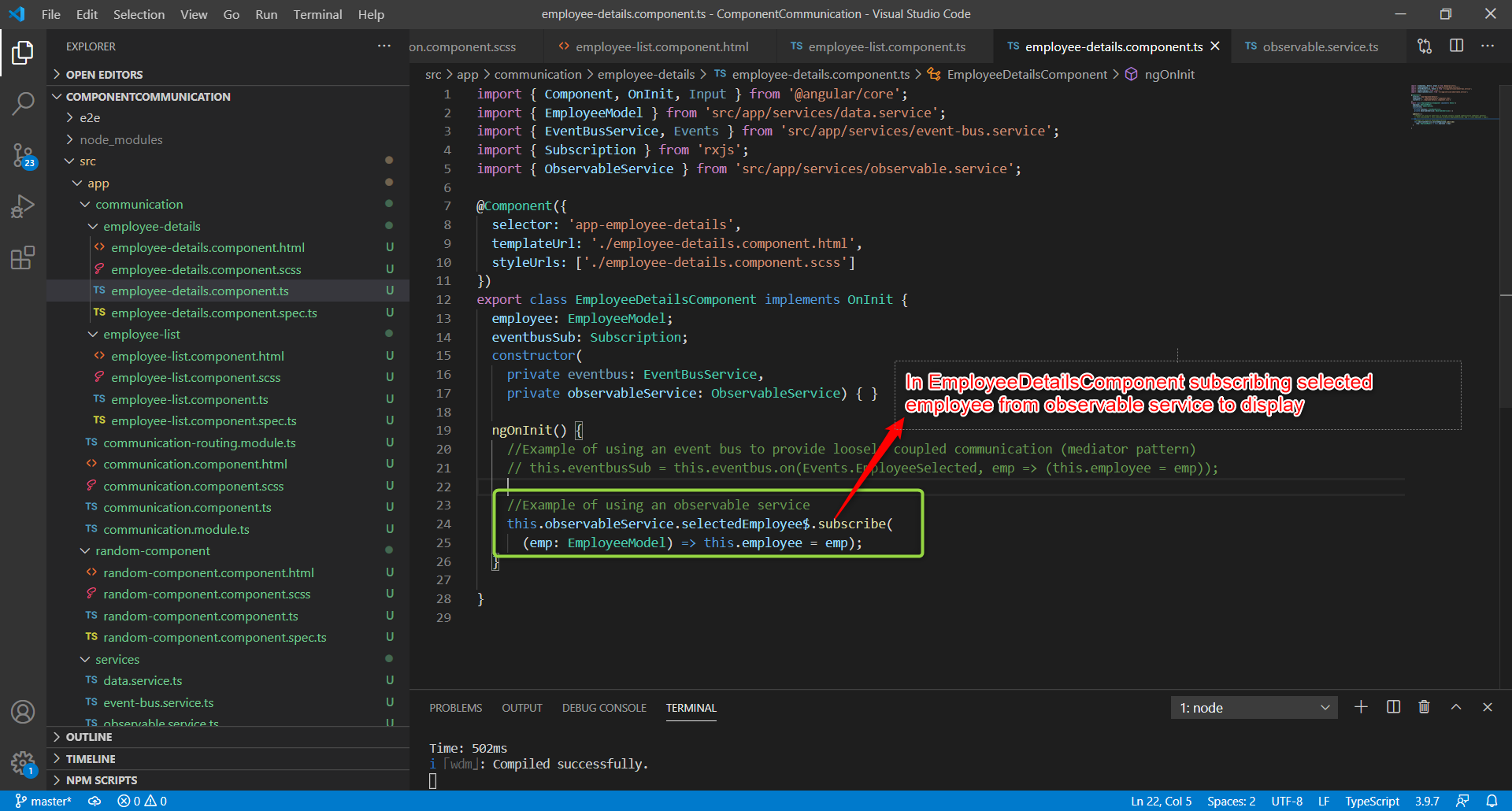
**Observable Service code**



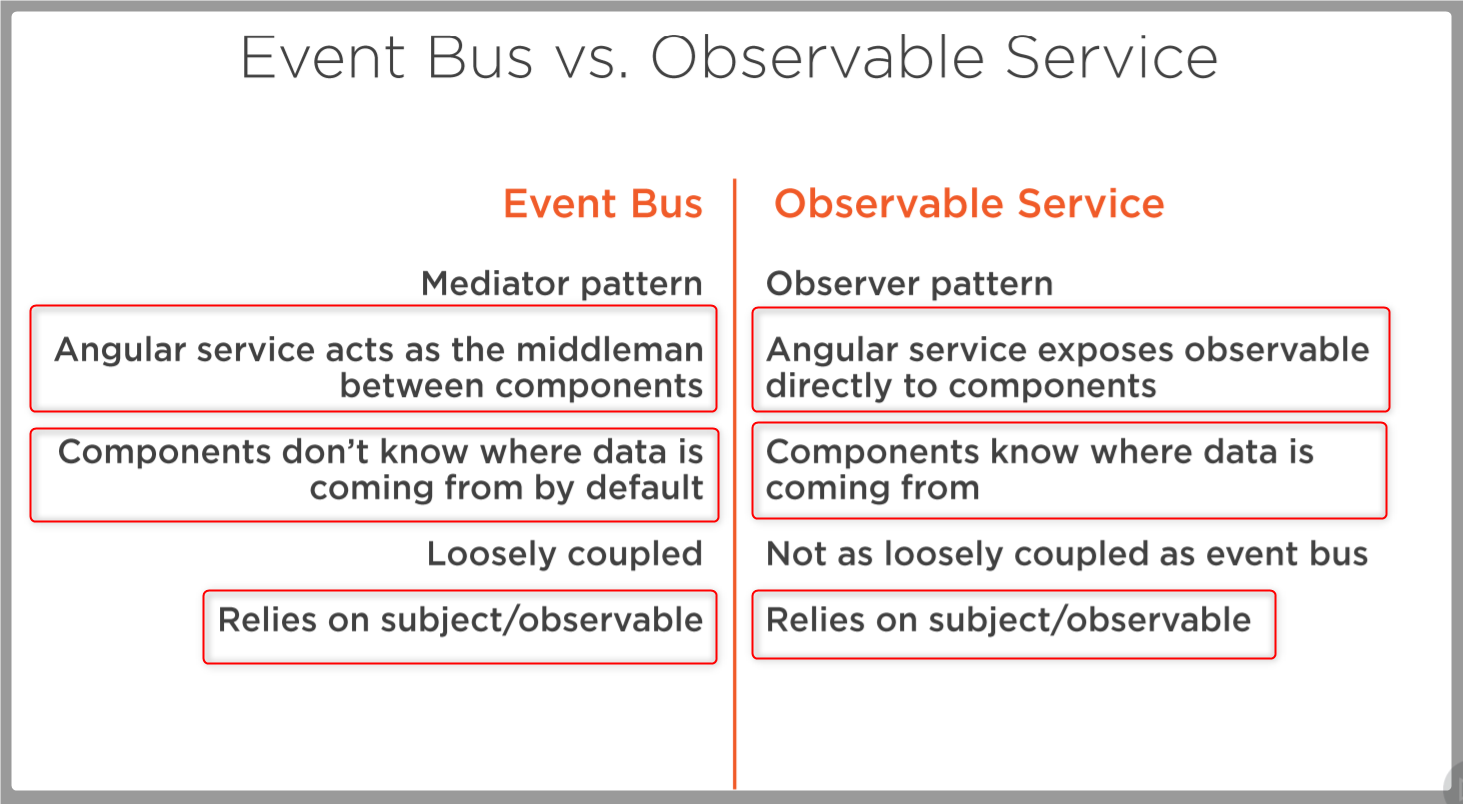
**EmployeeList Component code**

****

**EmployeeDetails Component code**

****

**Event bus vs Observable service**

****