Data and State Management

- Angular is flexible and doesn't prescribe a recipe for managing data in your apps
- Since observables are integrated into Angular, you can take advantage of observables to manage data and state
- You ca use services to manage streams that emit models
- Components can subscribe to the streams maintained by services and render accordingly.
    - For example, you can have a service for a Todo app that contains a stream of todos and a `ListComponent` can listen for todos and render when a new task is added.
    - You may have another component that listens for the user that has been assigned to a task provided by a service.
- The steps for creating different parts of an app can be summarized in three steps:
    - Defining a Model using a class
    - Defining the service
    - Defining the component

Observables

- Observables can help manage async data
- Observables are similar to Promises but with a lot of differences
- Observables emit multiple values over time as opposed to one
- Angular embraces observables using the RxJS library.
- Observables emit events and observers observe observables.
- An observer *subscribes* to events emitted from an observable.
- RxJS has an object called *subject* that can be used both as an observer or an observable. *Subject* can be imported from `RxJS` very easily:

```
1  import {Subject} from 'rxjs/Subject';
```

- A subscription can be canceled by calling the `unsubscribe` method.

TODO

State Management with Observables

- There are several ways to manage state, one of them is using observables
- Observables can be used to represent the state of the app
- Changes in the state are represented as an observable

TODO