# NgRx

Ramesh Maharaddi

## What Is NgRx?

NgRx = redux pattern + Angular

## What Is Redux?

- Redux is a JavaScript library, which helps you to manage the state of the application.

- This state can include server responses and cached data, as well as locally created data that has not yet been persisted to the server.

- Useful for medium to large single page application(SPA)

## Three Principles

- Single source of truth – The Store
  - The state of your whole application is stored in an object tree within a single store.
  - A single state tree also makes it easier to debug or inspect an application

- State is read-only - Dispatching actions
  - The only way to change the state is to emit an action, an object describing what happened.
  - This ensures that neither the views nor the network callbacks will ever write directly to the state. Instead, they express an intent to transform the state

- Changes are made with pure functions - Reducers
  - To specify how the state tree is transformed by actions, you write pure reducers.
  - Reducers are just pure functions that take the previous state and an action, and return the next state.
  - Remember to return new state objects, instead of mutating the previous state

## Building blocks of Redux

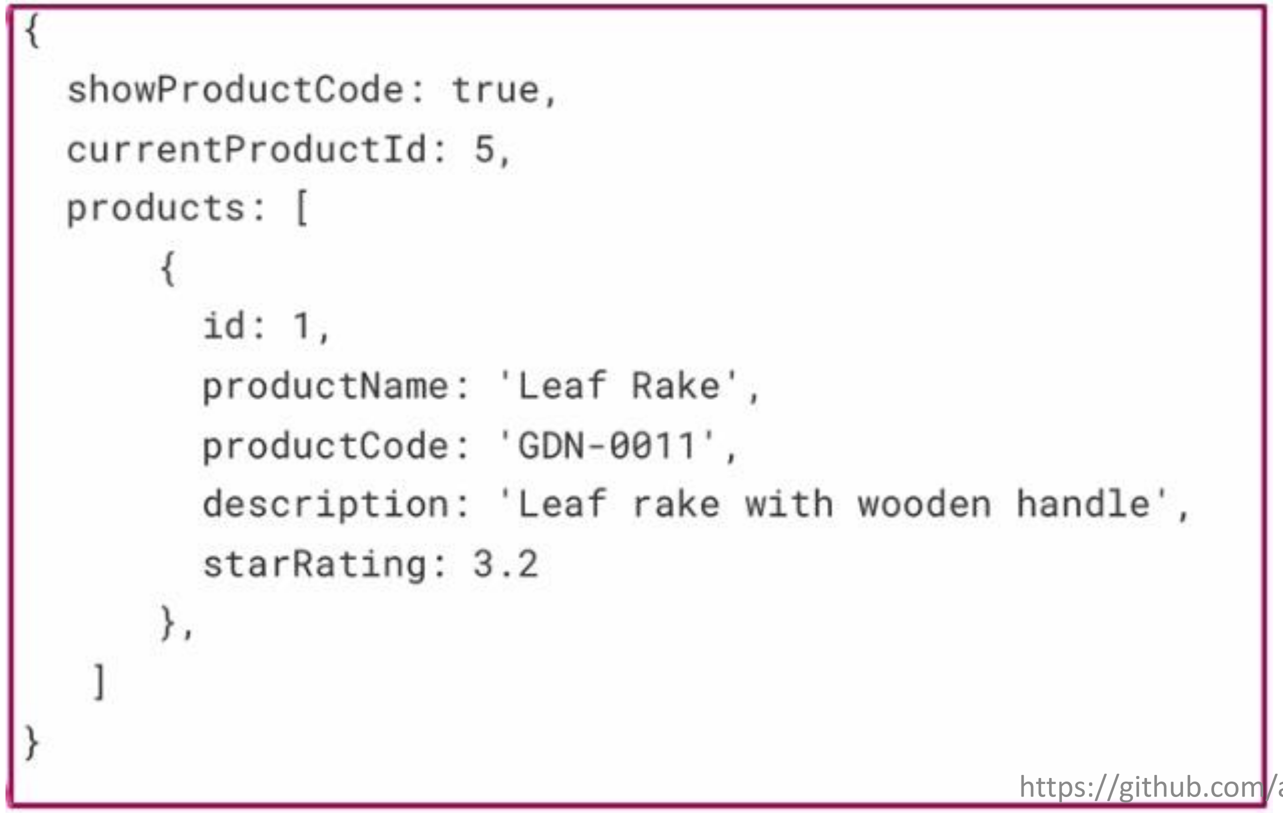
Store

Actions

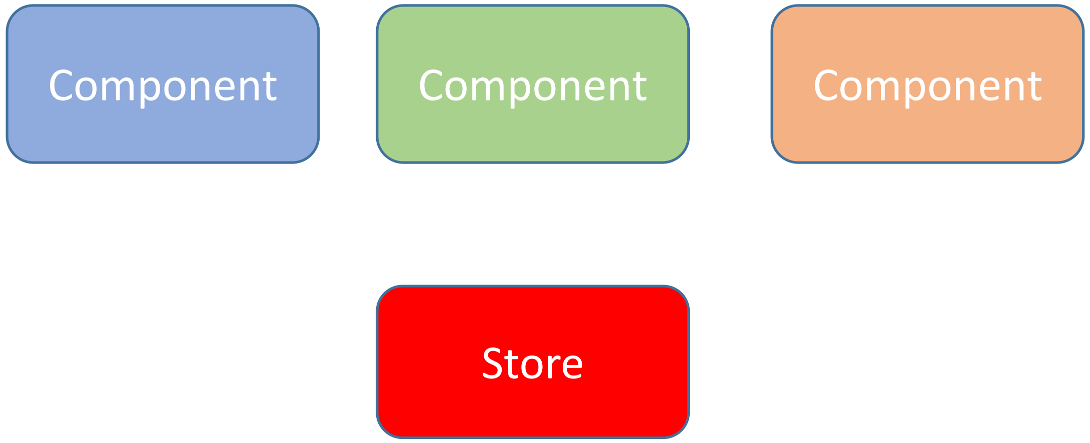
Reducers

## The Store

- ➢ A Single JS object that holds the state of the application
- ➢ You'll only have a single store in a Redux application.
- ➢ It's a good idea to think of its shape before writing any code
- ➢ Npm install @ngrx/store --save



```
{
  showProductCode: true,
  currentProductId: 5,
  products: [
      {
        id: 1,
        productName: 'Leaf Rake',
        productCode: 'GDN-0011',
        description: 'Leaf rake with wooden handle',
        starRating: 3.2
      },
   ]
}
```



## Actions

- Plain JS object that represents that something has happened in the application
- Actions are payloads of information that send data from your application to your store.
- Actions must have a type property that indicates the type of action being performed

Examples

```
{
   type: MARK_AS_READ
 }
{
   type: LOGIN,
   payload: {username: ‘scott’, password:’tiger’}
}
```

## Reducer

- A function that specify how the application's state changes in response to actions
- Reducers specify how the application's state changes in response to actions
- It does not modify the state, it returns the new state
- Not allowed to modify the state, So Reducer should be **pure function**

## Pure Functions

- Given the same input, will always return the same output.
- Produces no side effects.
- In pure function we should not mutate or modify any of the arguments

## Examples of Pure and Impure functions

| **Impure Functions** | **Why impure** |
|---|---|
| function increment(input){<br>    input.count++;<br>} | Mutating Argument<br>We should not modify any argument |
| function increment(input){<br>  input.count+=math.random();<br>} | For the same input we get different output each time<br>Or Date.now() produces different output |

### <u>Pure Function</u>

```
function increment(input){
    return { count: input.count+1};
}
```

```
function sum(a, b) {
    const result = a + b;
    return result;
}
```

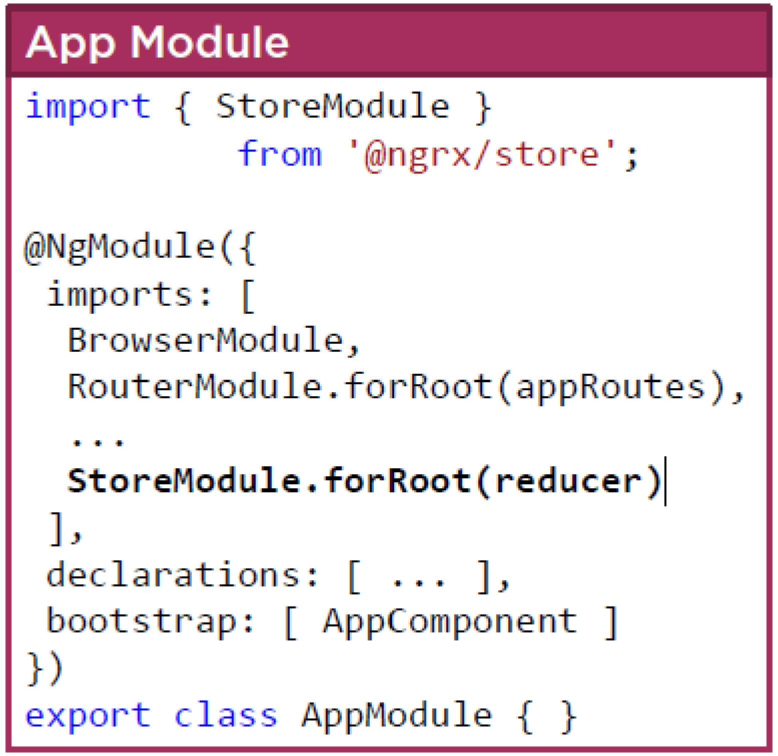
## Reducer Function

```
function reducer (state, action){
        switch(action.type){
                case ‘INCREMENT’:
                        return { count: input.count+1};
        }
}
```

- Reducer function takes two argument
  - current state and action
  - Based on the action type, return new state

# Setting up NgRx-Angular application

- ng new ngrx-demo --skip-install
- Npm install @ngrx/store --save
- ng serve –o
- import { StoreModule } from '@ngrx/store';
- imports StoreModule.forRoot(reducer)



App Module

```
import { StoreModule }
          from '@ngrx/store';

@NgModule({
 imports: [
  BrowserModule,
  RouterModule.forRoot(appRoutes),
  ...
  StoreModule.forRoot(reducer)
 ],
 declarations: [ ... ],
 bootstrap: [ AppComponent ]
})
export class AppModule { }
```

# Feature Module State Composition

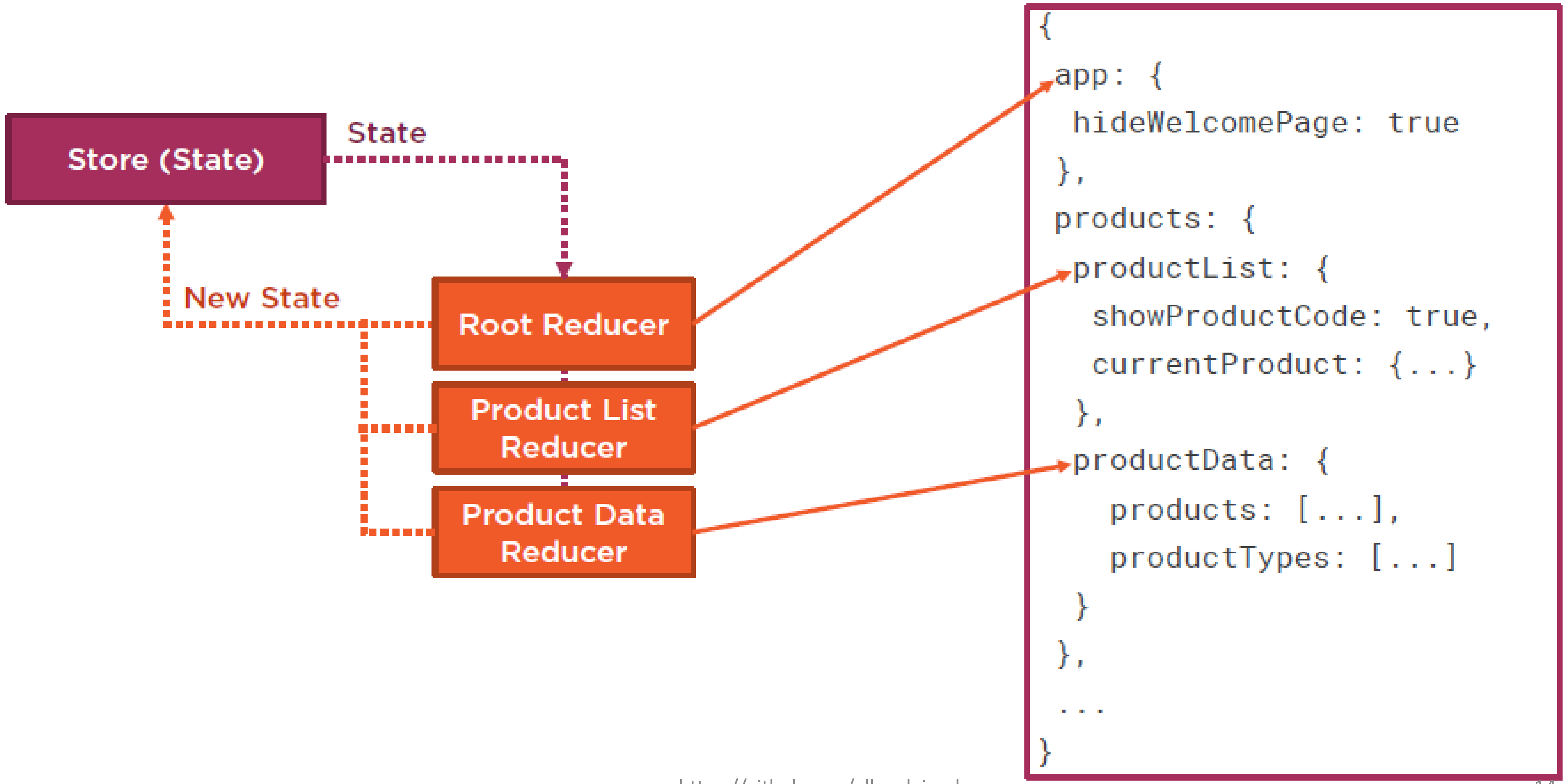
**App Module**

```
import { StoreModule }
         from '@ngrx/store';

@NgModule({
 imports: [
  BrowserModule,
  RouterModule.forRoot(appRoutes),
  ...
  StoreModule.forRoot(reducer)
 ],
 declarations: [ ... ],
 bootstrap: [ AppComponent ]
})
export class AppModule { }
```

**Product Module**

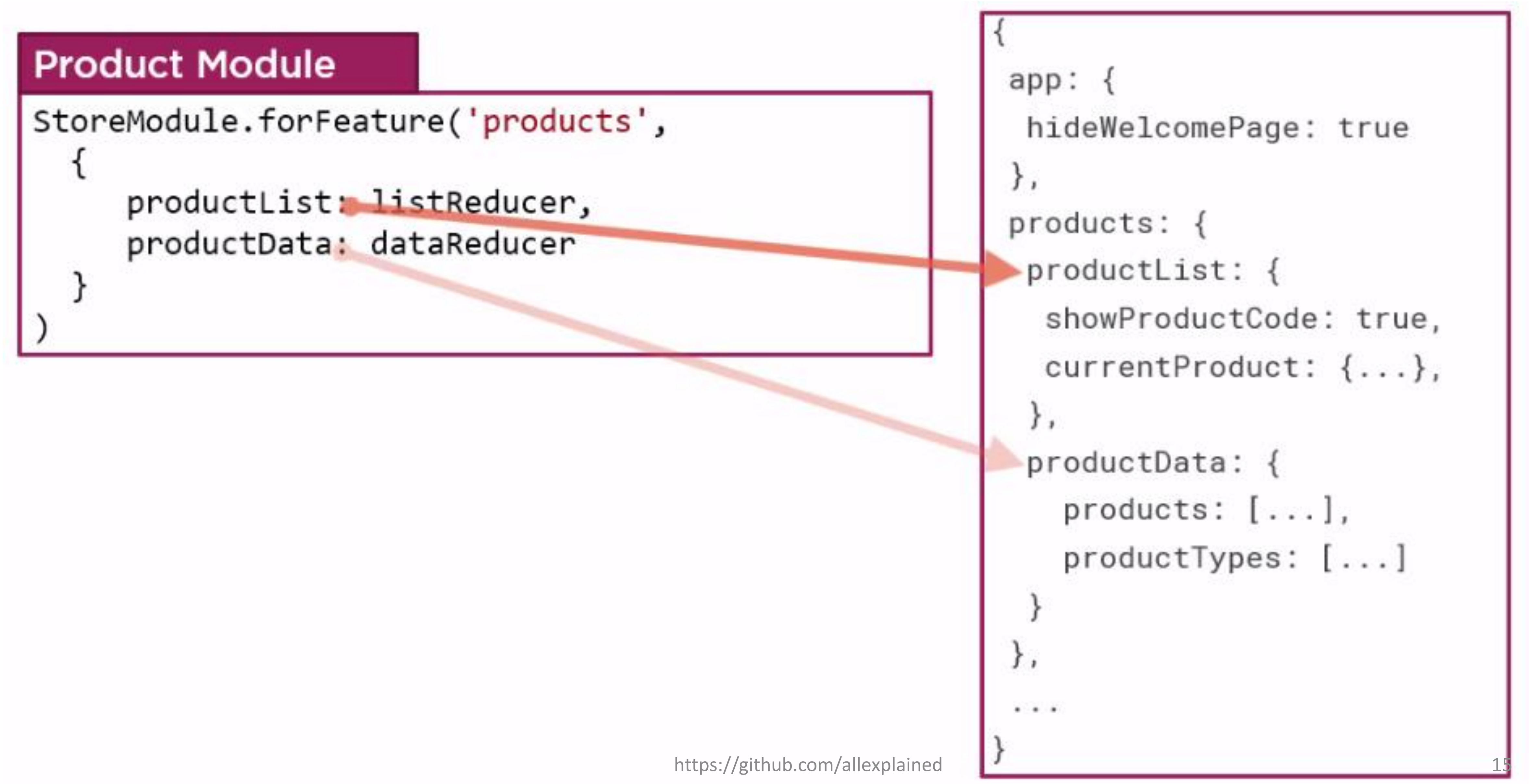
```
import { StoreModule }
         from '@ngrx/store';

@NgModule({
 imports: [
  SharedModule,
  RouterModule.forChild(productRoutes),
  ...
  StoreModule.forFeature('products', reducer)
 ],
 declarations: [ ... ],
 providers: [ ... ]
})
export class ProductModule { }
```

# Sub-slice of State



```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  productList: {
   showProductCode: true,
   currentProduct: {...}
  },
  productData: {
    products: [...],
    productTypes: [...]
  }
 },
 ...
}
```

# Sub-slice of State



**Product Module**

```
StoreModule.forFeature('products',
  {
     productList: listReducer,
     productData: dataReducer
  }
)
```

```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  productList: {
   showProductCode: true,
   currentProduct: {...},
  },
  productData: {
    products: [...],
    productTypes: [...]
  }
 },
 ...
}
```