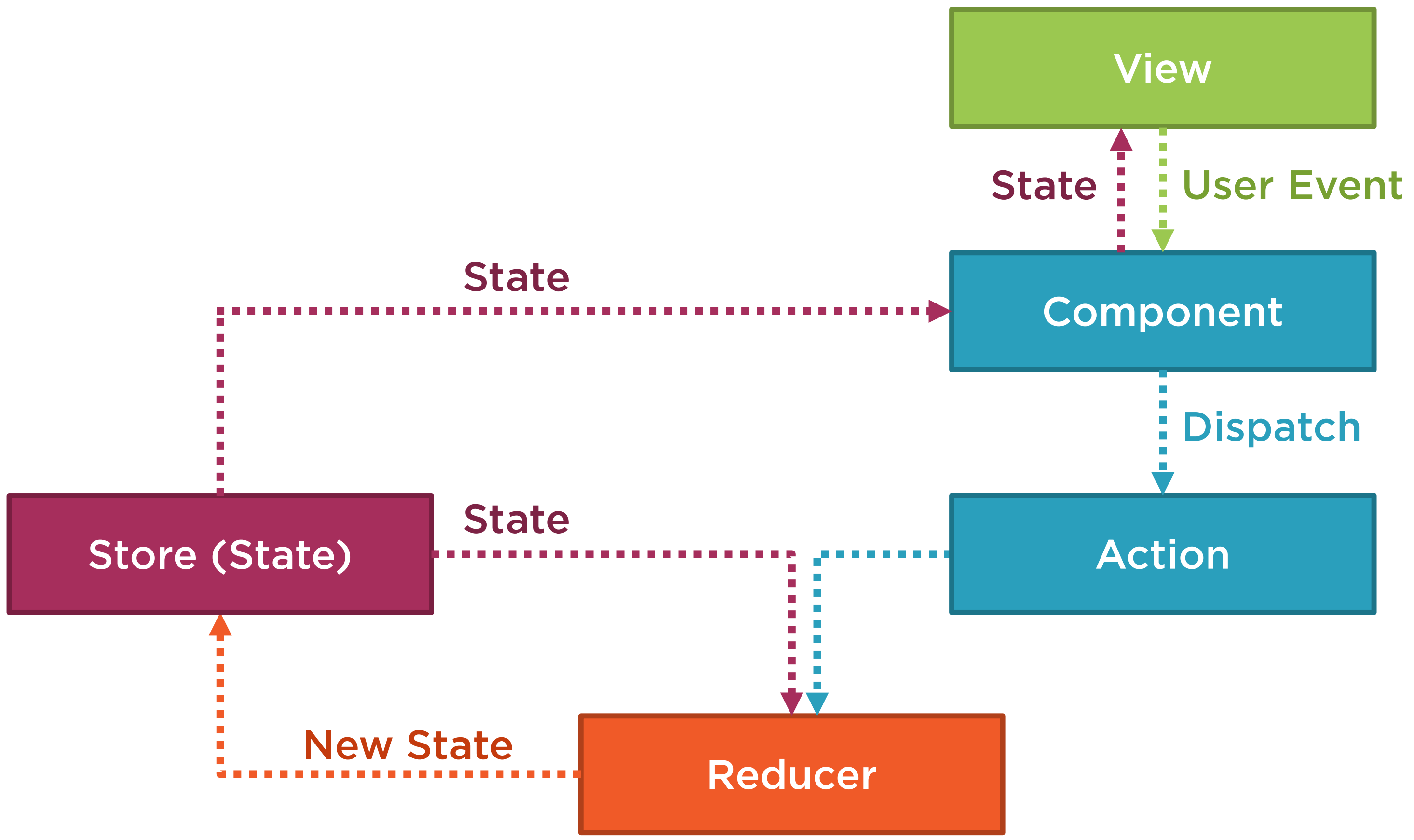
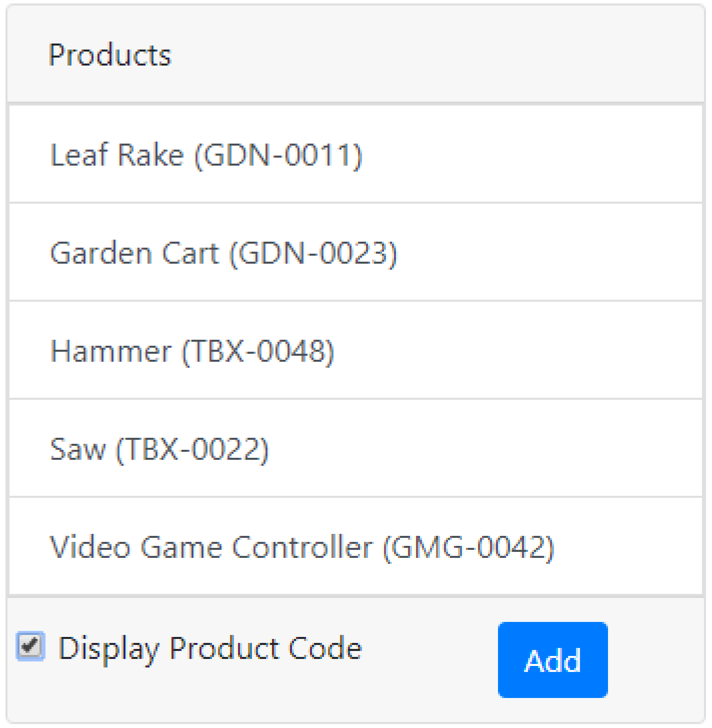
# First Look at NgRx

**Deborah Kurata**
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata blogs.msmvps.com/deborahk/




Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

Hammer (TBX-0048)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

☑ Display Product Code

## Module Overview

Set up the sample application

Install @ngrx/store

Initialize the store

Define the state and actions

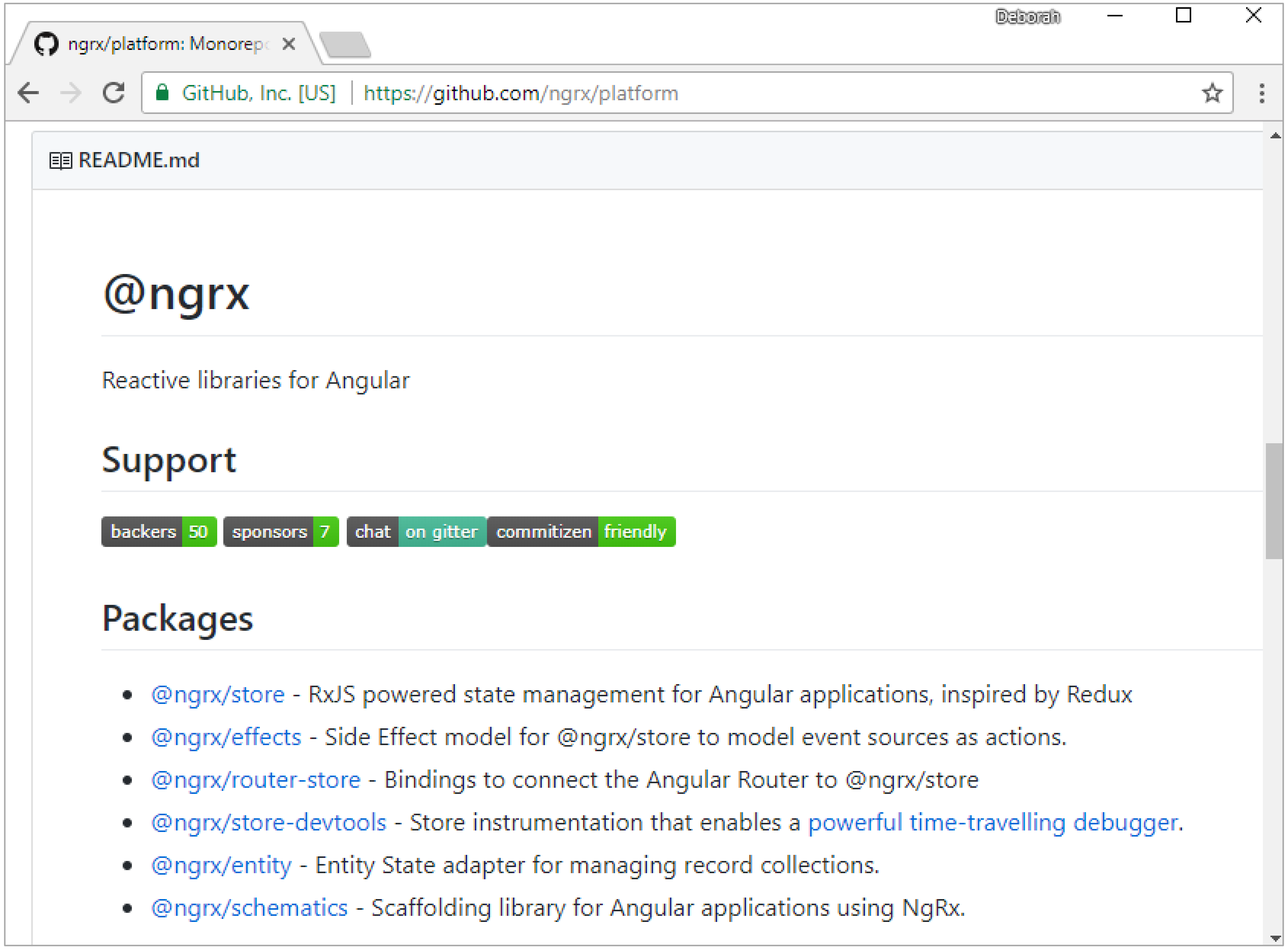
Build a reducer to process actions and set store state

Dispatch an action to change state

Subscribe to state change notifications

Demo

Setting up the sample application



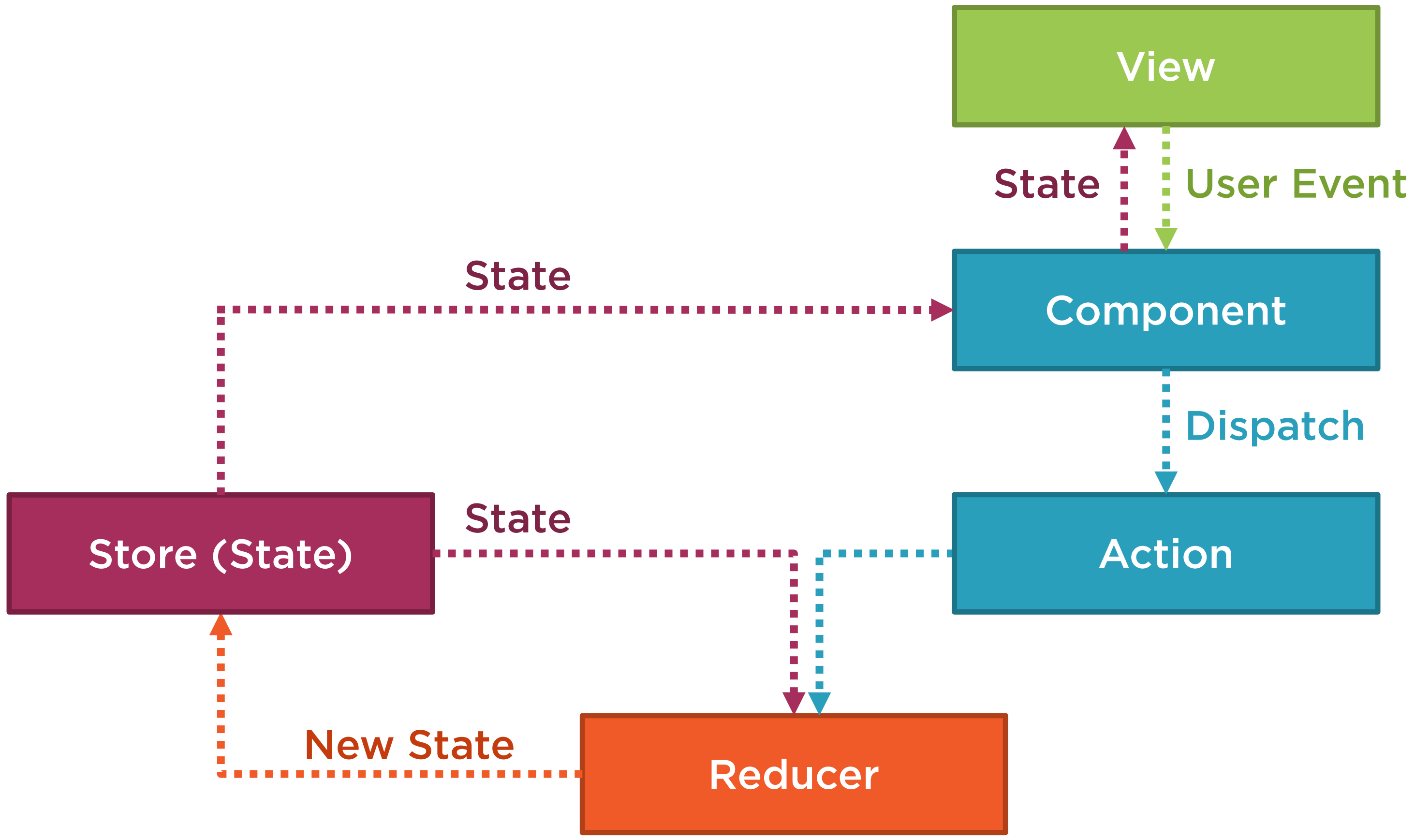
# @ngrx

Reactive libraries for Angular

## Support

## Packages

- @ngrx/store - RxJS powered state management for Angular applications, inspired by Redux
- @ngrx/effects - Side Effect model for @ngrx/store to model event sources as actions.
- @ngrx/router-store - Bindings to connect the Angular Router to @ngrx/store
- @ngrx/store-devtools - Store instrumentation that enables a powerful time-travelling debugger.
- @ngrx/entity - Entity State adapter for managing record collections.
- @ngrx/schematics - Scaffolding library for Angular applications using NgRx.

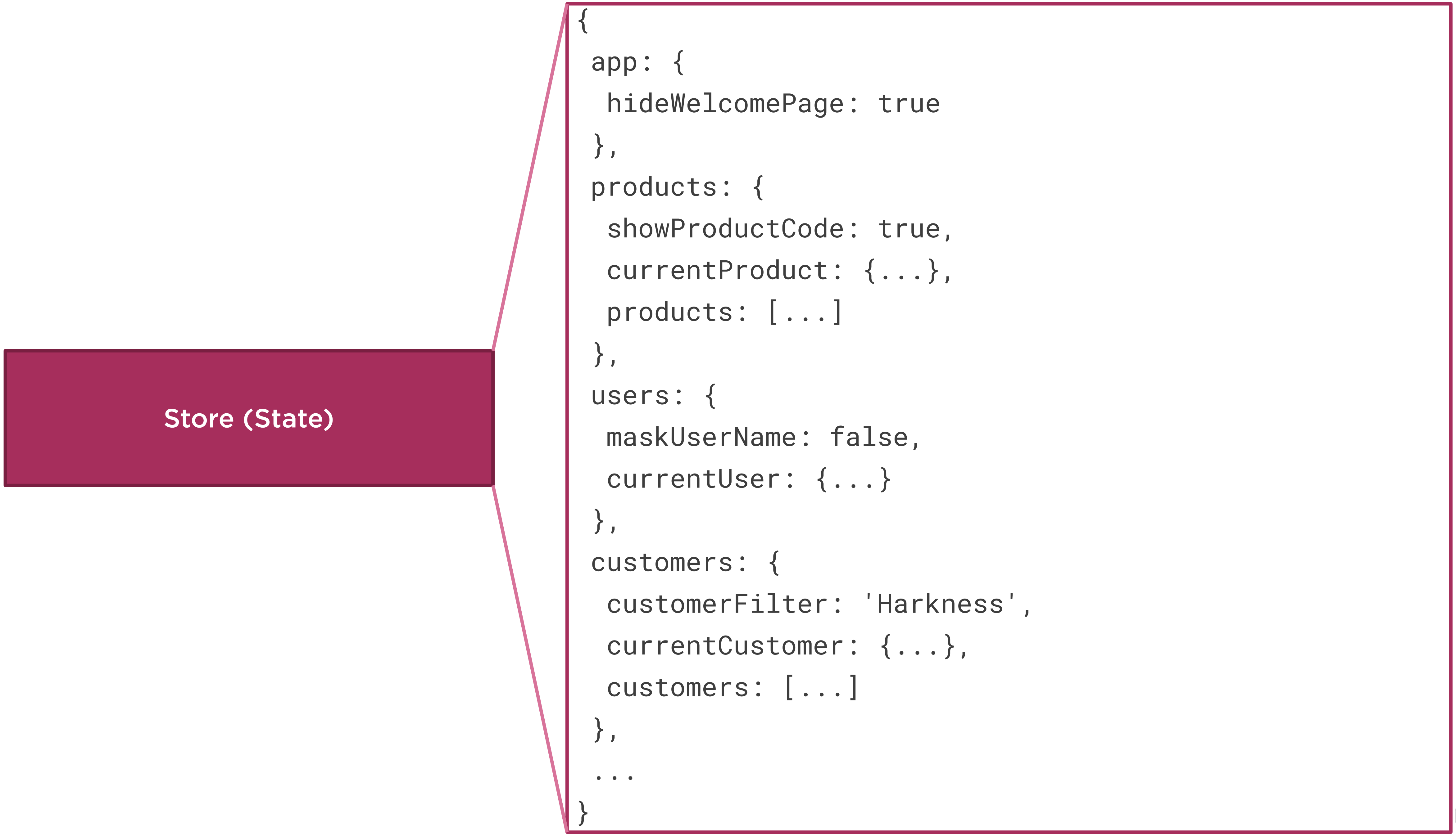

View
State
User Event
State
Component
Dispatch
Store (State)
State
Action
New State
Reducer

Store (State)

```
{
 showProductCode: true,
 currentProduct: {
    id: 5,
    productName: 'Hammer',
    productCode: 'TBX-0048',
    description: 'Curved claw steel hammer',
    starRating: 4.8
 },
 products: [
    {
      id: 1,
      productName: 'Leaf Rake',
      productCode: 'GDN-0011',
      description: 'Leaf rake with wooden handle',
      starRating: 3.2
    },
    ...
 ]
}
```

Store (State)

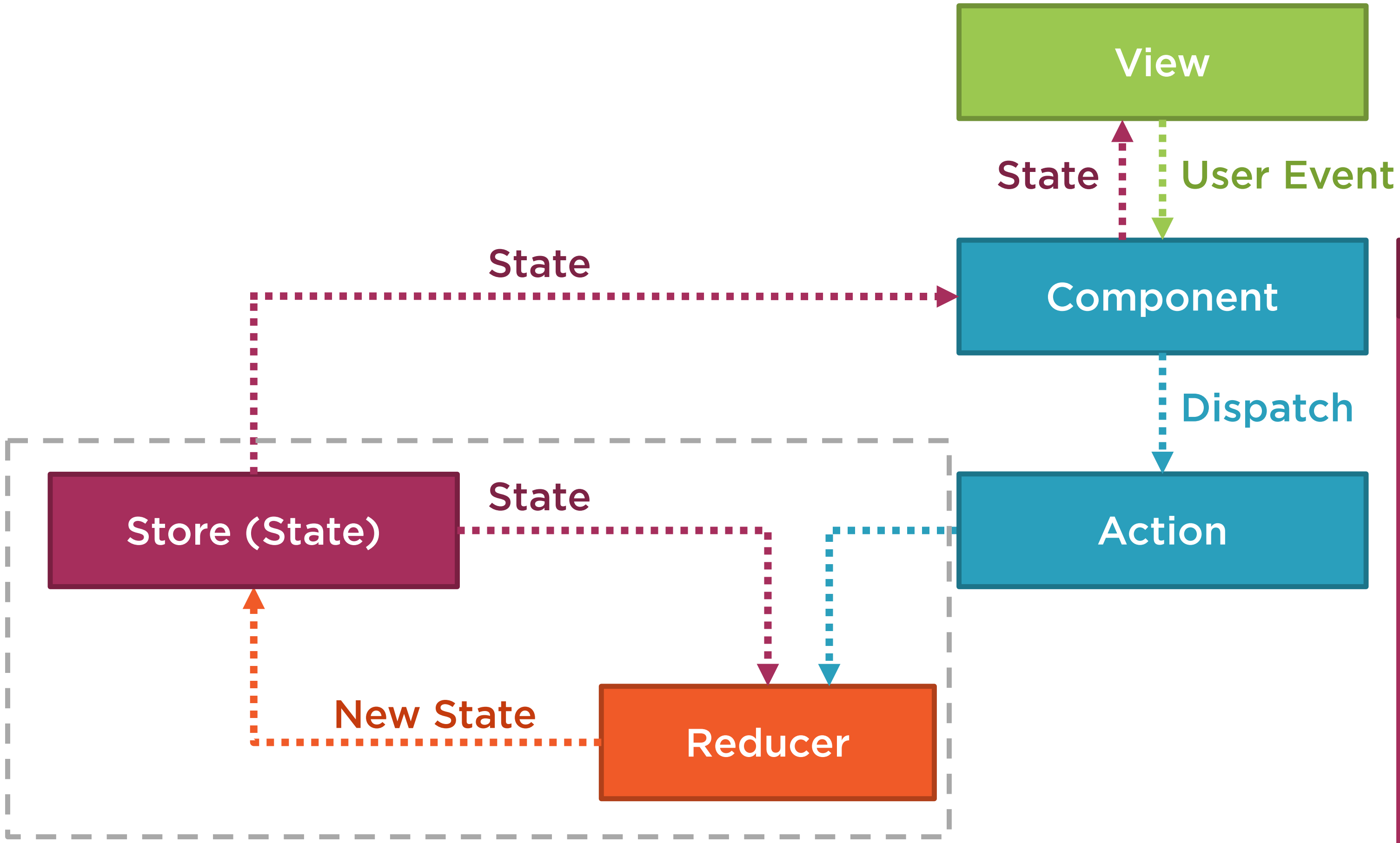
```
{
  showProductCode: true,
  currentProduct: {...},
  products: [...],
  hideWelcomePage: true,
  maskUserName: false,
  currentUser: {...},
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...],
  allowGuest: false,
  includeAds: true,
  displayCustomerDetail: false,
  allowEdit: false,
  listFilter: 'Hammer',
  displayAddress: false,
  orders: [...],
  cart: [...],
  ...
}
```



```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
 },
 users: {
  maskUserName: false,
  currentUser: {...}
 },
 customers: {
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...]
 },
 ...
}
```

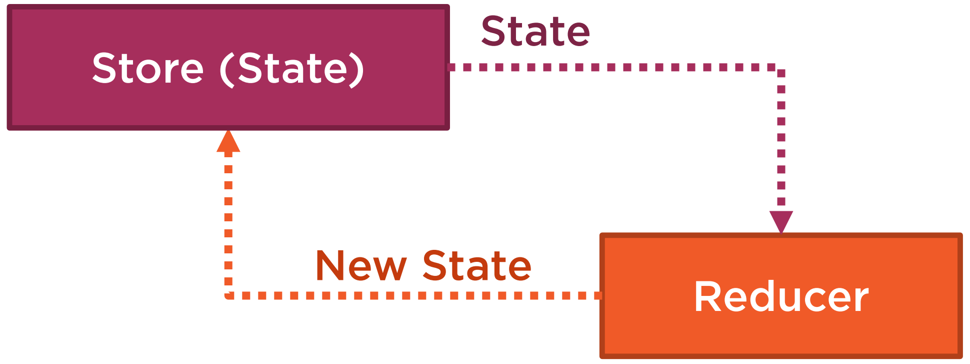
## Demo

Installing @ngrx/store

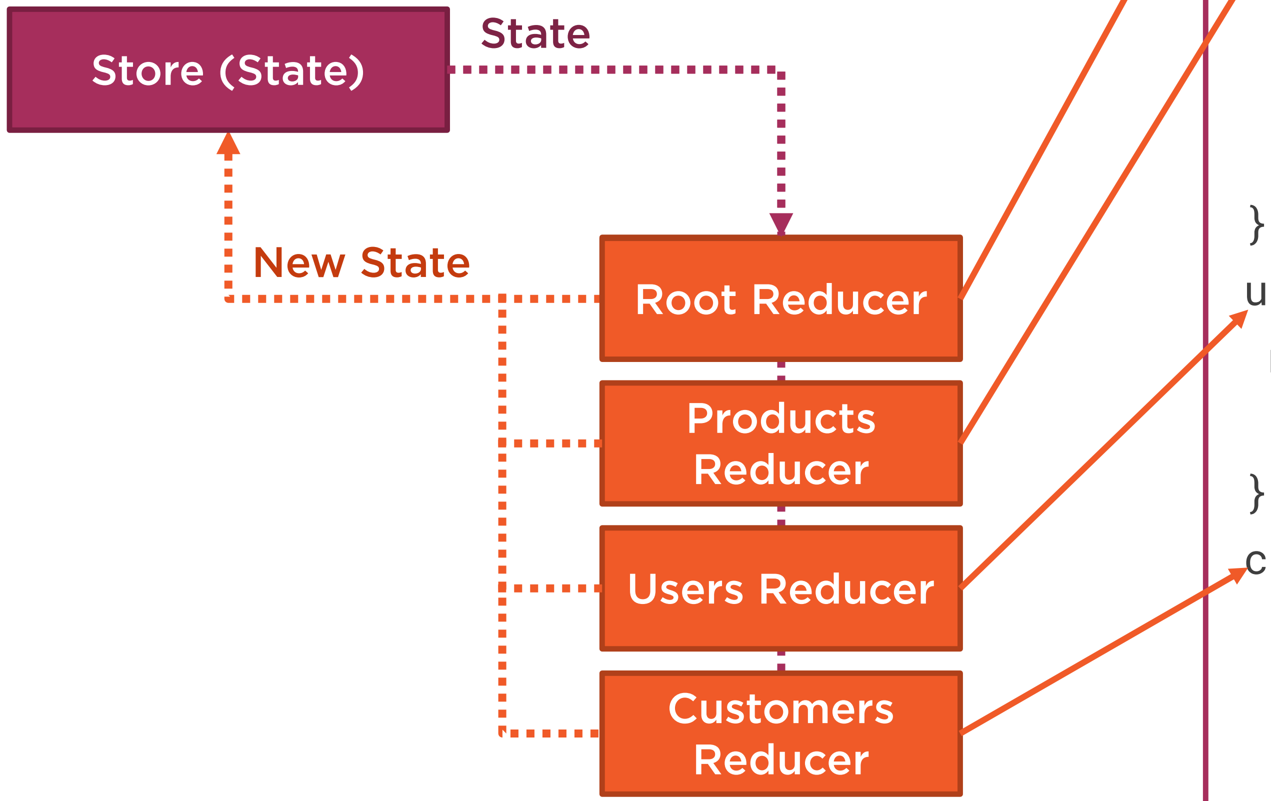


## App Module

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

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



```
{
  showProductCode: true,
  currentProduct: {...},
  products: [...],
  hideWelcomePage: true,
  maskUserName: false,
  currentUser: {...},
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...],
  allowGuest: false,
  includeAds: true,
  displayCustomerDetail: false,
  allowEdit: false,
  listFilter: 'Hammer',
  displayAddress: false,
  orders: [...],
  cart: [...],
  ...
}
```



```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
 },
 users: {
  maskUserName: false,
  currentUser: {...}
 },
 customers: {
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...]
 },
 ...
}
```

# Feature Module State Composition

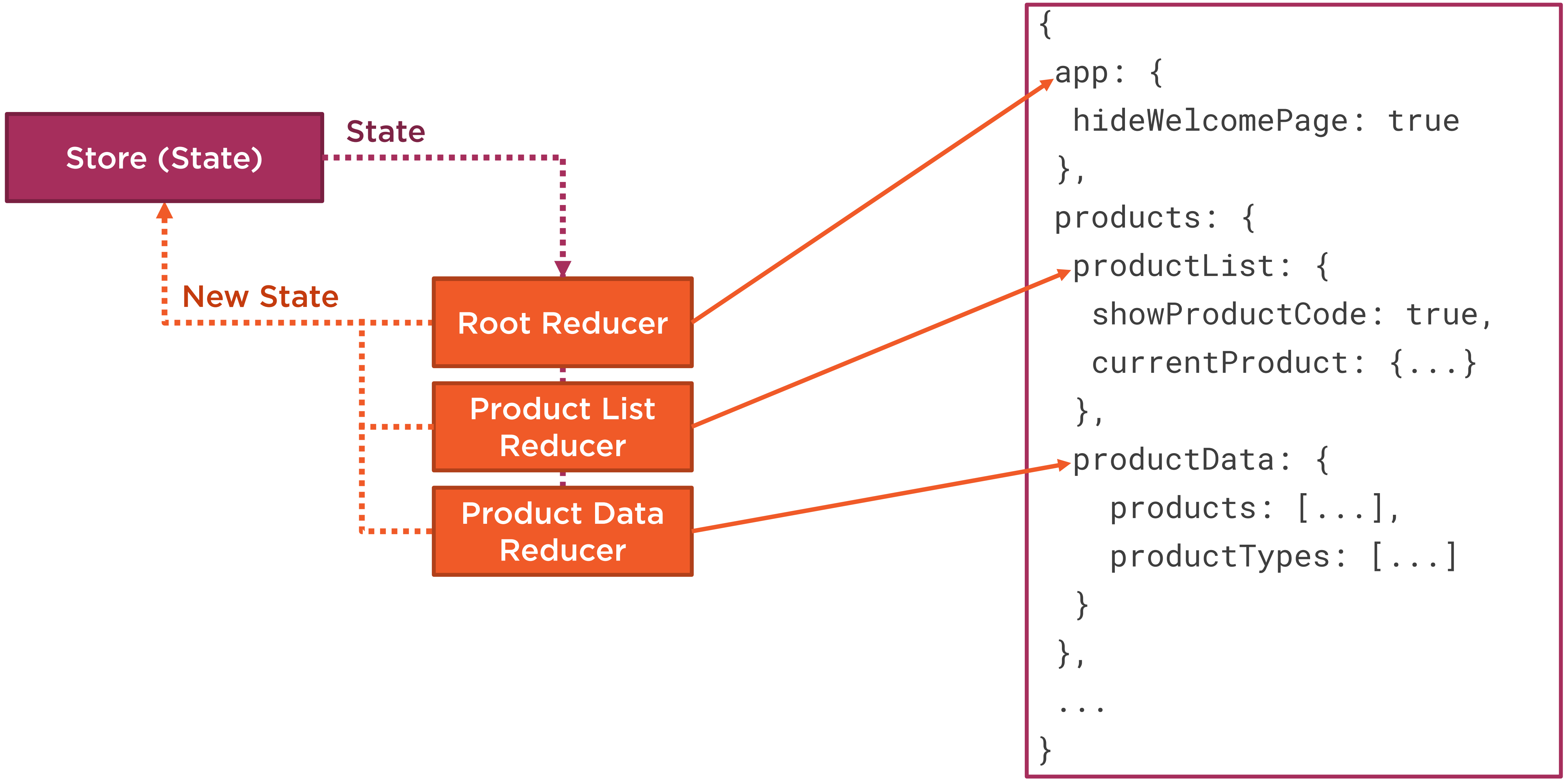
**App Module**

```
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



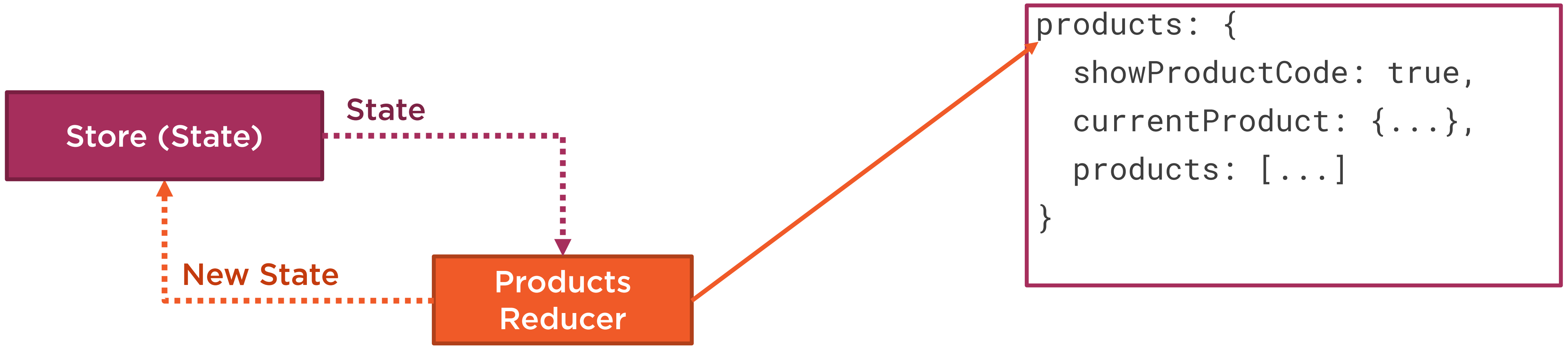
```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  productList: {
   showProductCode: true,
   currentProduct: {...}
  },
  productData: {
    products: [...],
    productTypes: [...]
  }
 },
 ...
}
```

# Sub-slice of State

**Product Module**

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

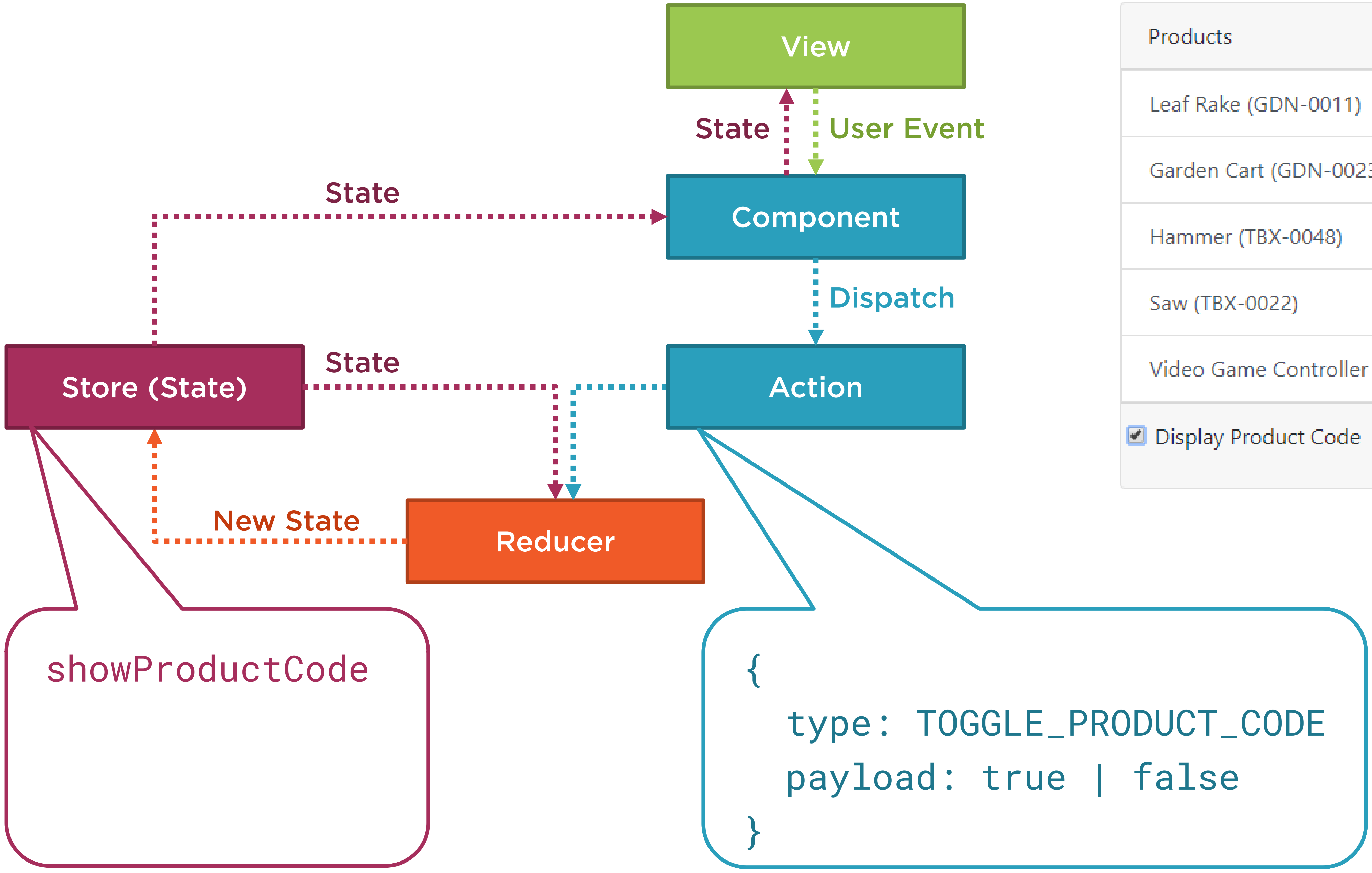
```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  productList: {
   showProductCode: true,
   currentProduct: {...},
  },
  productData: {
    products: [...],
    productTypes: [...]
  }
 },
 ...
}
```



```
products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
}
```

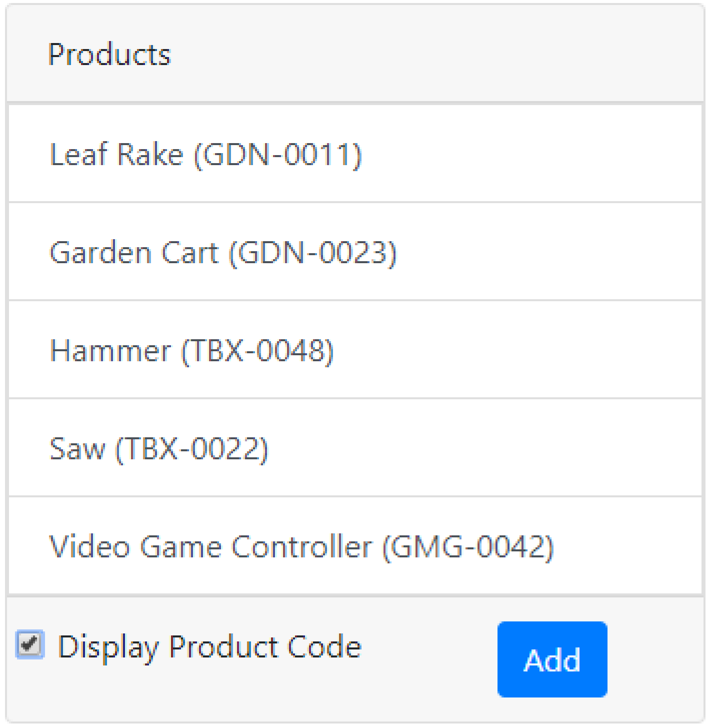
Demo

## Initializing the Store



```
showProductCode
```

```
{
  type: TOGGLE_PRODUCT_CODE
  payload: true | false
}
```


Products

Leaf Rake (GDN-0011)

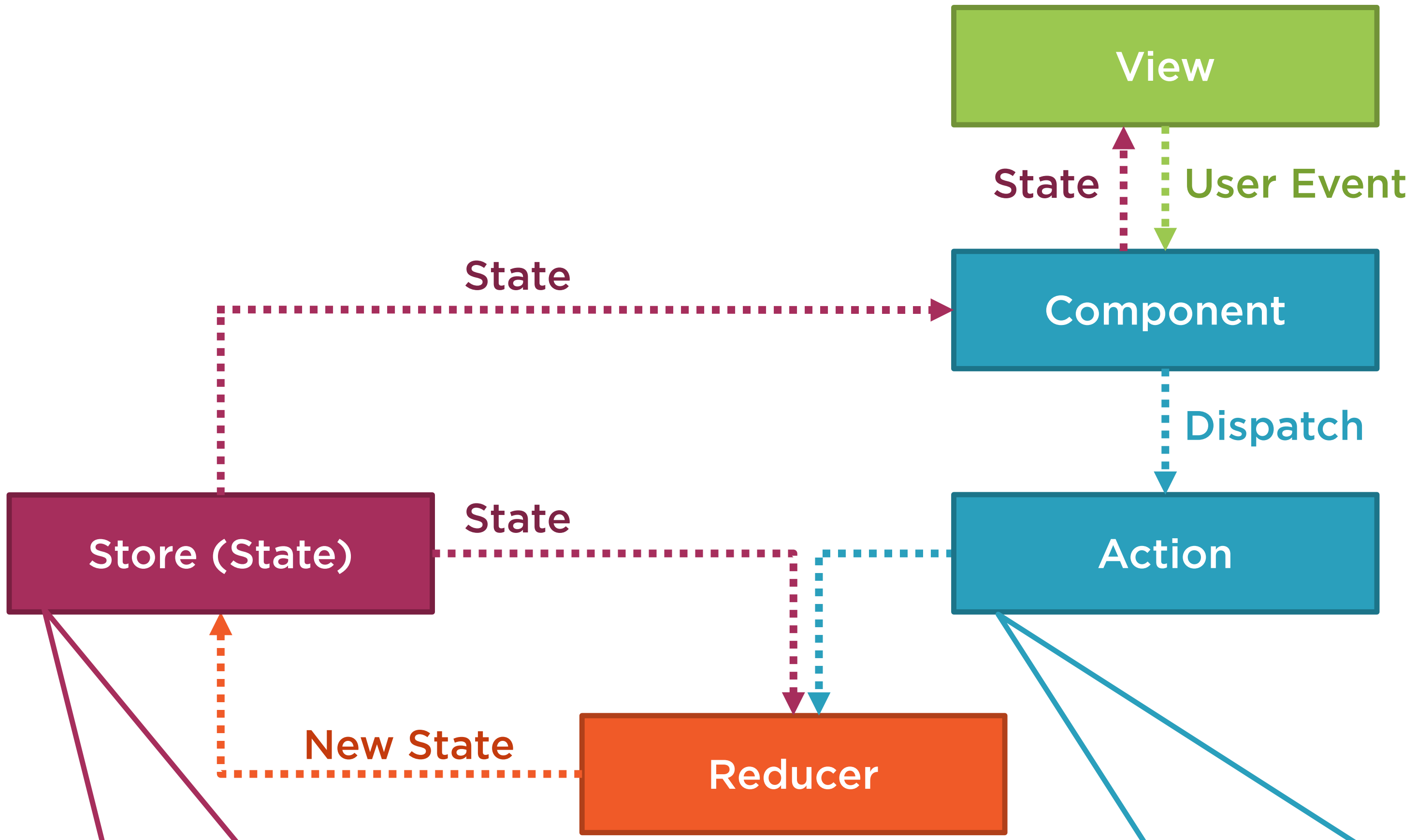
Garden Cart (GDN-0023)

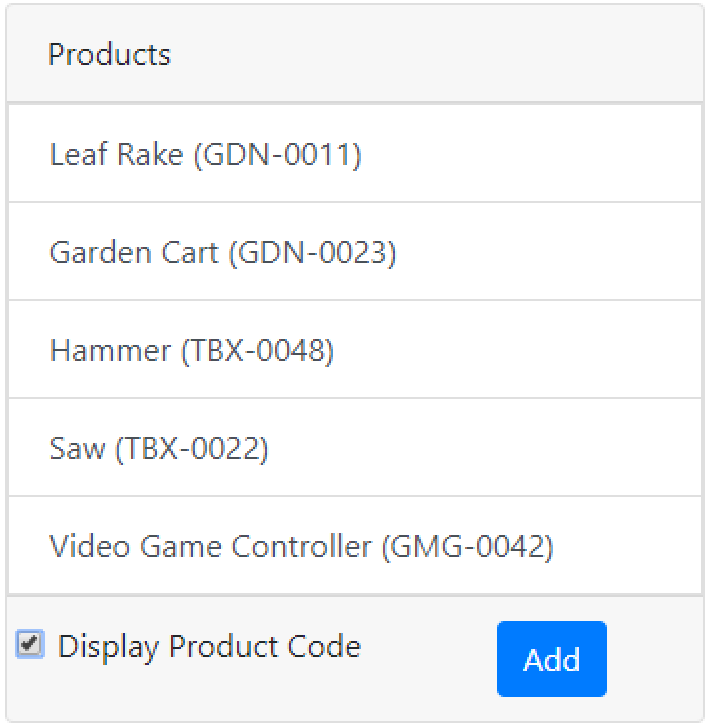
Hammer (TBX-0048)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

☑ Display Product Code




Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

Hammer (TBX-0048)

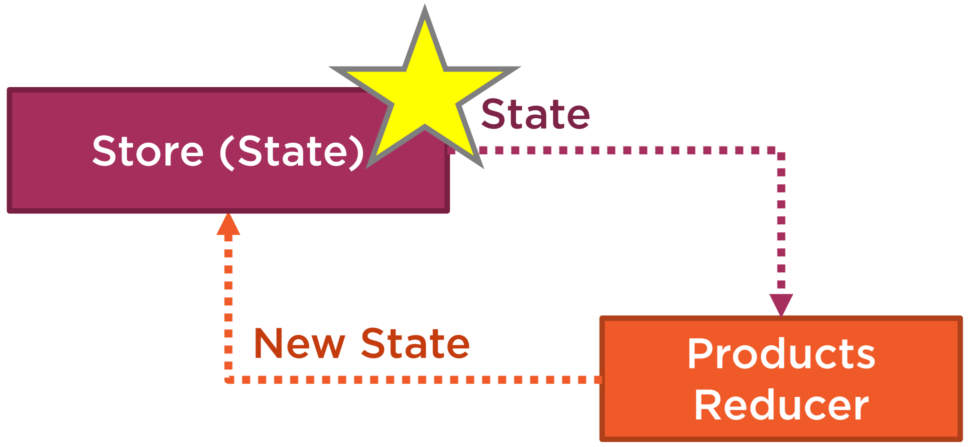
Saw (TBX-0022)

Video Game Controller (GMG-0042)

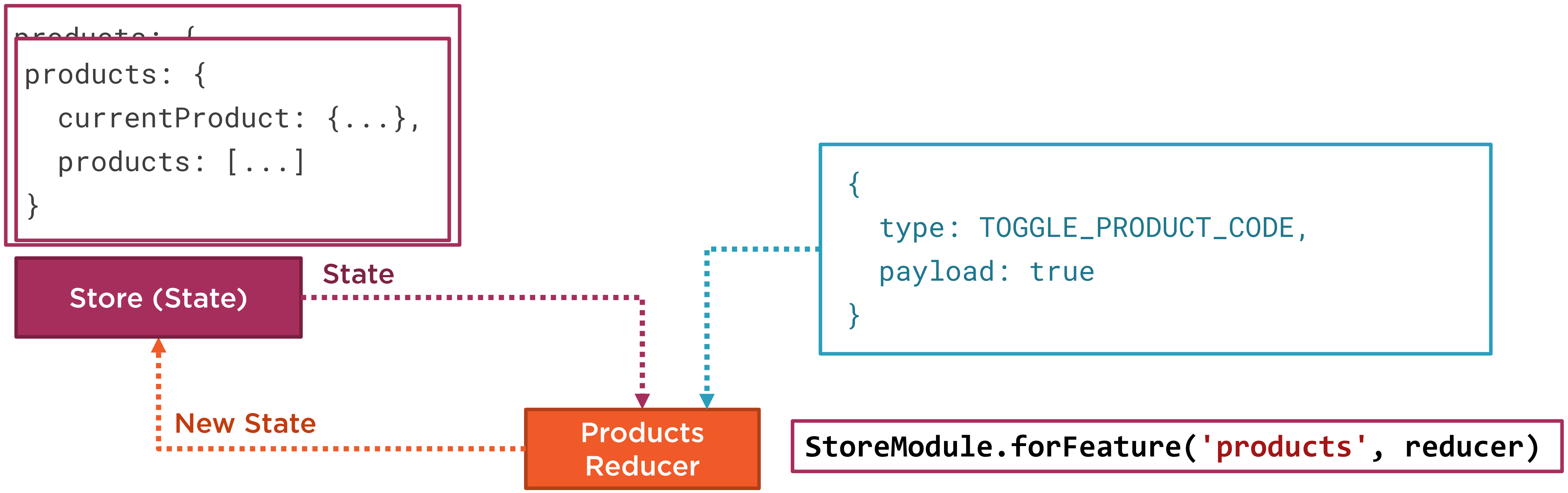
☑ Display Product Code

```
showProductCode
```

```
{
  type: TOGGLE_PRODUCT_CODE
  payload: true | false
}
```



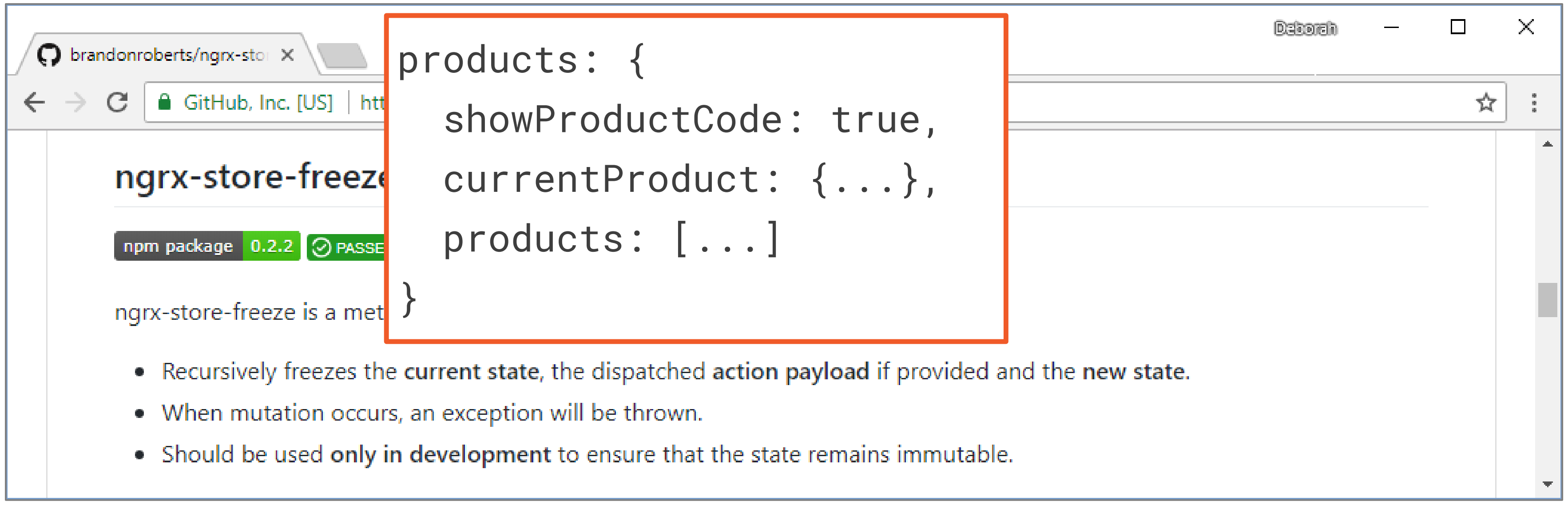
```
{
 showProductCode: true,
 currentProduct: {
    id: 5,
    productName: 'Hammer',
    productCode: 'TBX-0048',
    description: 'Curved claw steel hammer',
    starRating: 4.8
 },
 products: [
    {
      id: 1,
      productName: 'Leaf Rake',
      productCode: 'GDN-0011',
      description: 'Leaf rake with wooden handle',
      starRating: 3.2
    },
    ...
 ]
}
```



```
products: {
  currentProduct: {...},
  products: [...]
}
```

```
{
  type: TOGGLE_PRODUCT_CODE,
  payload: true
}
```

```
StoreModule.forFeature('products', reducer)
```



```
products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
}
```

ngrx-store-freeze

ngrx-store-freeze is a met

- Recursively freezes the **current state**, the dispatched **action payload** if provided and the **new state**.
- When mutation occurs, an exception will be thrown.
- Should be used **only in development** to ensure that the state remains immutable.

```
products: {
 currentProduct: {...},
 products: [...]
}
```

Store (State)

```
{
  type: TOGGLE_PRODUCT_CODE,
  payload: true
}
```

Product Reducer

```
export function reducer(state, action) {

 switch (action.type) {

  case 'TOGGLE_PRODUCT_CODE':
   return {
     ...state,
     showProductCode: action.payload
   };

  default:
   return state;
 }
}
```

```
products: {
 showProductCode: true,
 currentProduct: {...},
 products: [...]
}
```

# Anatomy of a Reducer

**Product Reducer**

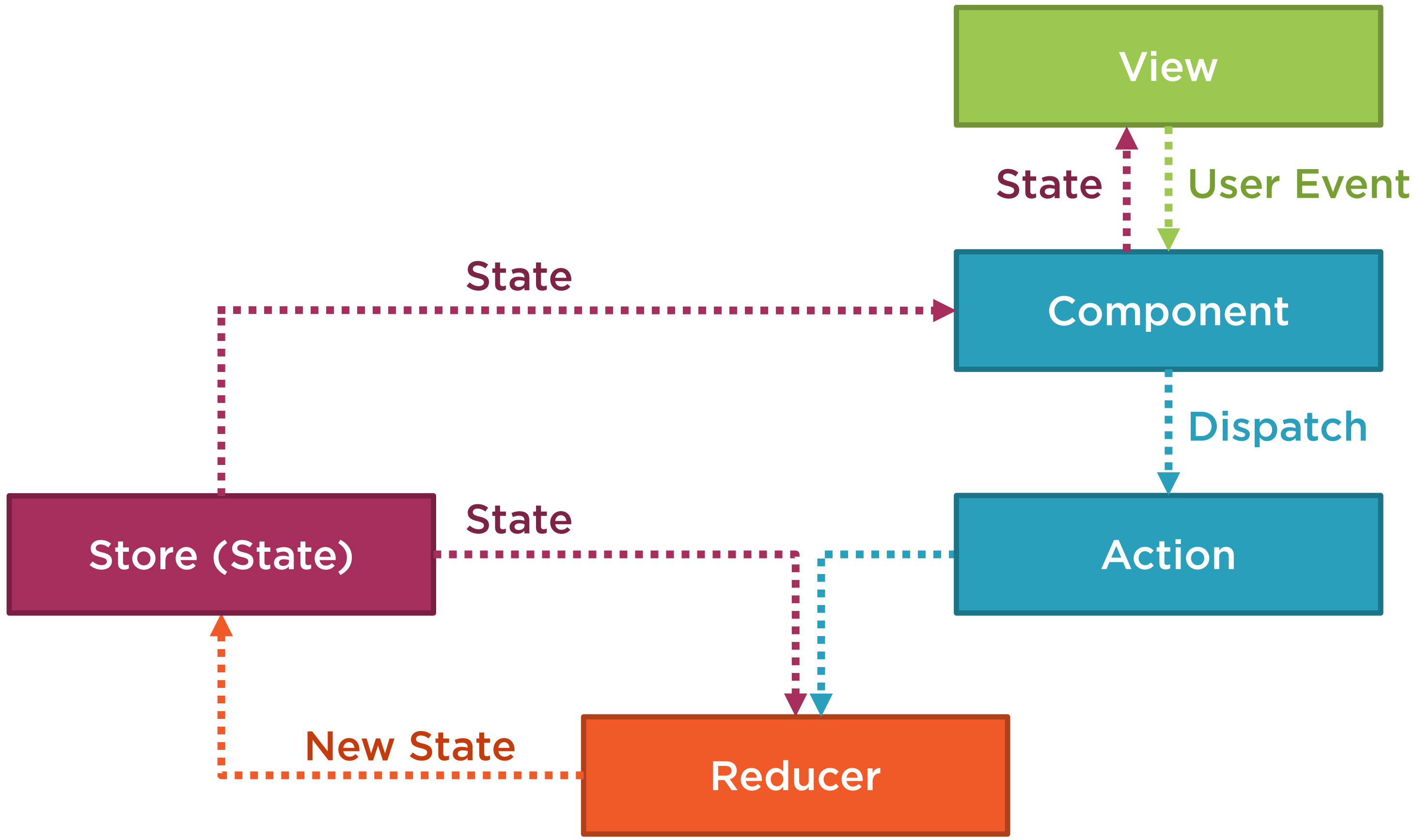
```
export function reducer(state, action) {

 switch (action.type) {

  case 'TOGGLE_PRODUCT_CODE':
   return {
     ...state,
     showProductCode: action.payload
   };

  default:
   return state;
 }
}
```

**Product Reducer**

```
export function reducer(state, action) {

 switch (action.type) {

  case 'SHOW_PRODUCT_CODE':
   return {...state, showProductCode: true};

  case 'RESET_DEFAULTS':
  case 'HIDE_PRODUCT_CODE':
   return {...state, showProductCode: false};

  default:
   return state;
  }
}
```

Demo

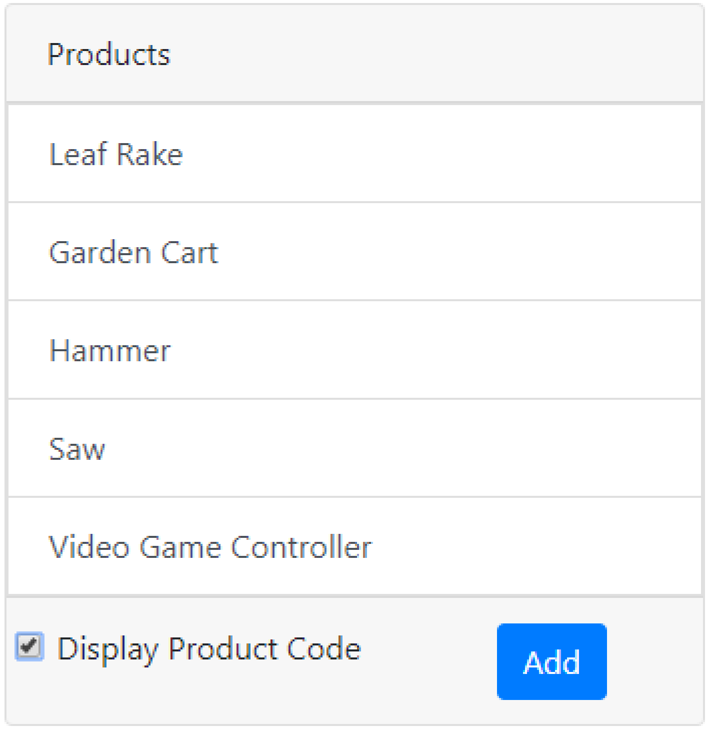
## Building a reducer to process actions


View
State
User Event
State
Component
Dispatch
State
Store (State)
Action
New State
Reducer

# Dispatching an Action

**Product List Component**

```
constructor(private store: Store<any>) {}
```


Products

Leaf Rake

Garden Cart

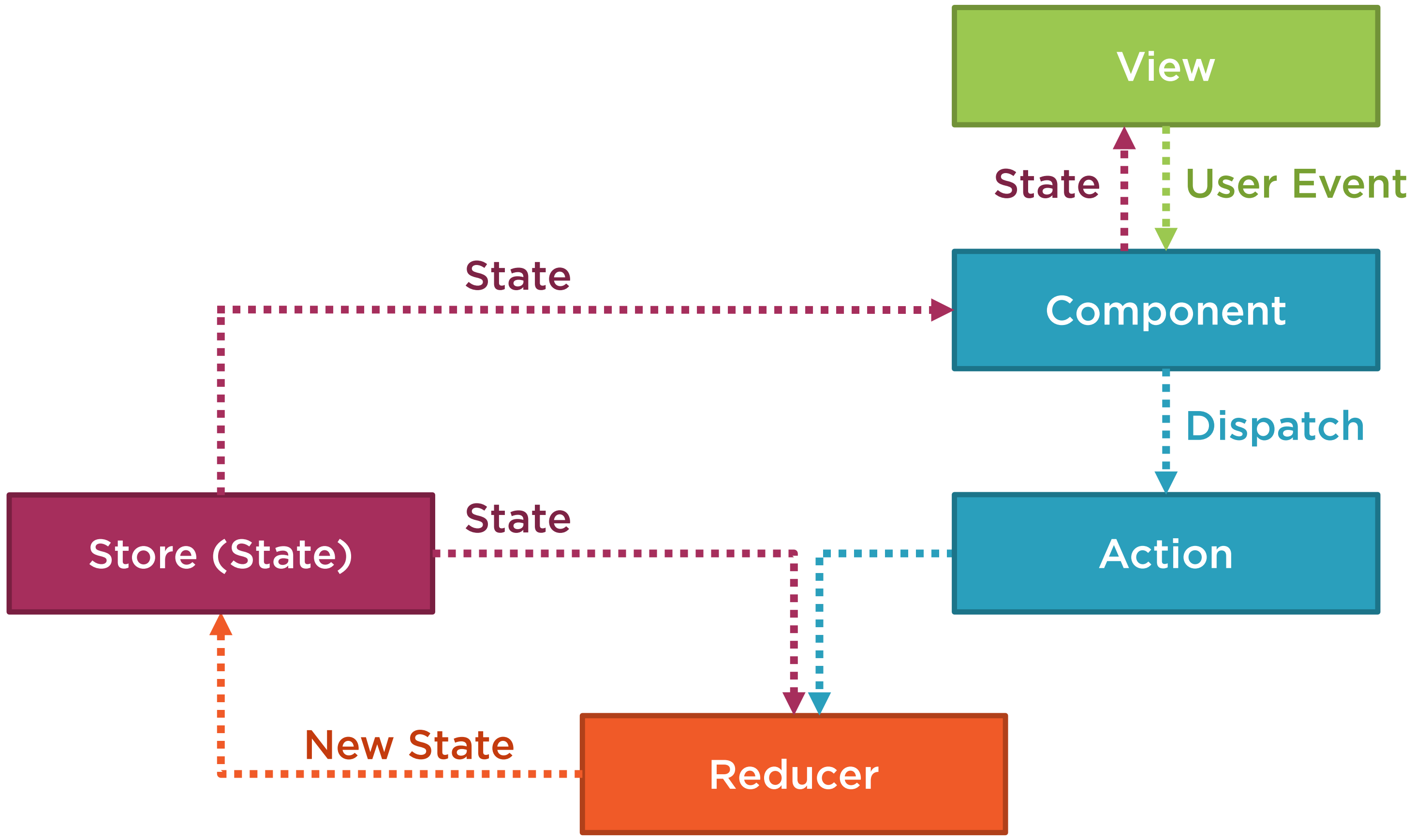
Hammer

Saw

Video Game Controller

Demo

## Dispatching an action


View
State
User Event
State
Component
Dispatch
State
Store (State)
Action
New State
Reducer

## Product List Component

```
this.store.select('products');
```

```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
 },
 users: {
  maskUserName: false,
  currentUser: {...}
 },
 customers: {
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...]
 },
 ...
}
```

## Product List Component

```
this.store.pipe(select('products'))
```

```
{
 app: {
  hideWelcomePage: true
 },
 products: {
  showProductCode: true,
  currentProduct: {...},
  products: [...]
 },
 users: {
  maskUserName: false,
  currentUser: {...}
 },
 customers: {
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...]
 },
 ...
}
```

## Demo

Subscribing to state changes

# Checklist: Store

Store (State)

**Single container for application state**

**Interact with that state in an immutable way**

**Install the @ngrx/store package**

**Organize application state by feature**

**Name the feature slice with the feature name**

**Initialize the store using:**

```
StoreModule.forRoot(reducer)
StoreModule.forFeature('feature', reducer)
```

## Checklist: Action

Action

An action represents an event

Define an action for each event worth tracking

Action is an object with a type and optional payload:

```
{
    type: 'TOGGLE_PRODUCT_CODE',
    payload: value
}
```

## Checklist: Reducer

**Responds to dispatched actions**

**Replaces the state tree with new state**

**Build a reducer function (often one or more per feature)**

**Implement as a switch with a case per action:**

```
export function reducer(state, action) {
 switch (action.type) {
  case 'TOGGLE_PRODUCT_CODE':
   return { ...state, showProductCode: action.payload };
 }
}
```

**Spread the state as needed**

## Checklist: Dispatching an Action

Action

**Often done in response to a user action or an operation**

**Inject the store in the constructor**

**Call the dispatch method of the store**

**Pass in the action to dispatch:**

```
this.store.dispatch({
   type: 'TOGGLE_PRODUCT_CODE',
   payload: value
});
```

**Passed to all reducers**

# Checklist: Subscribing to the Store

Store (State)

Often done in the ngOnInit lifecycle hook

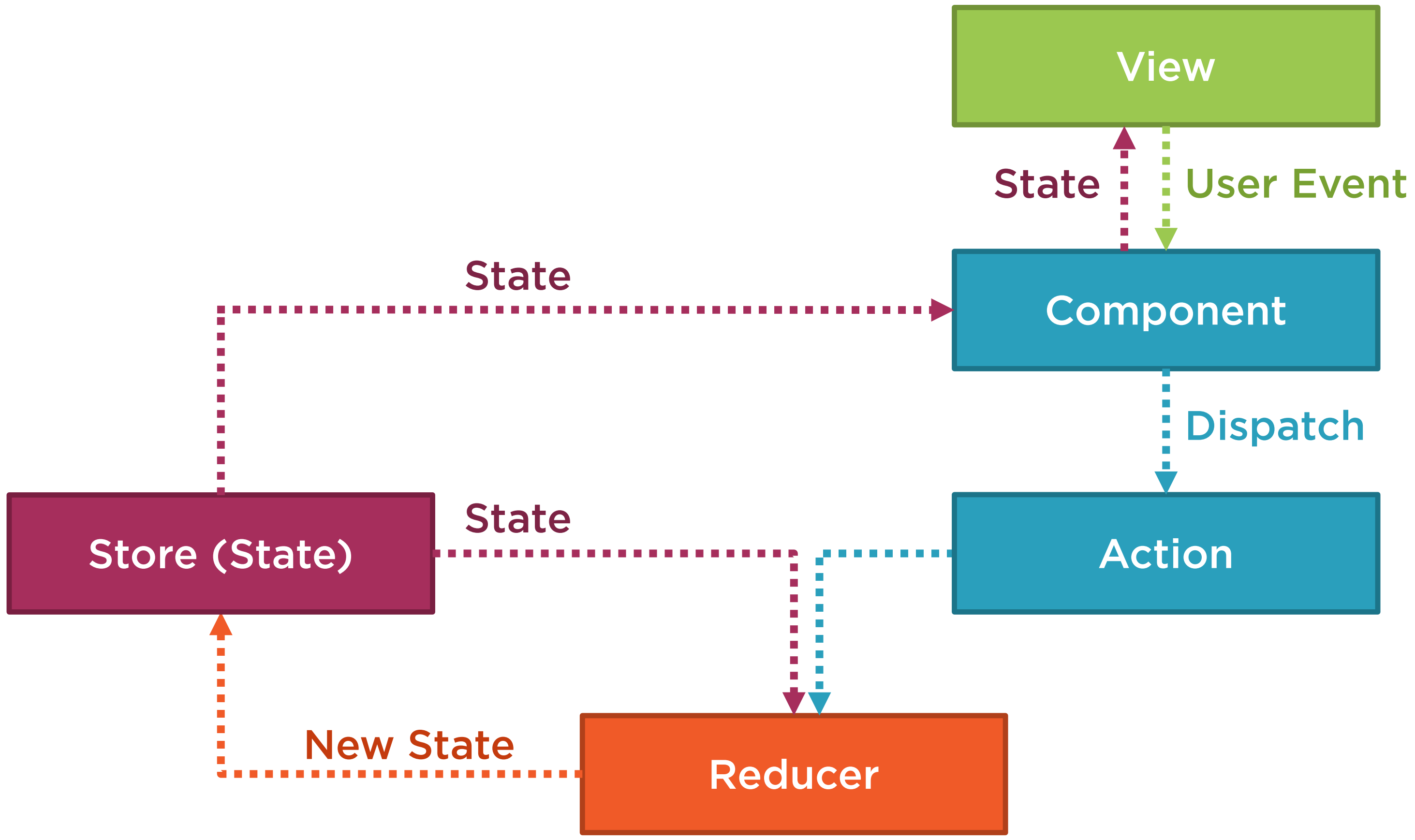
Inject the store in the constructor

Use the store's select operator, passing in the desired slice of state

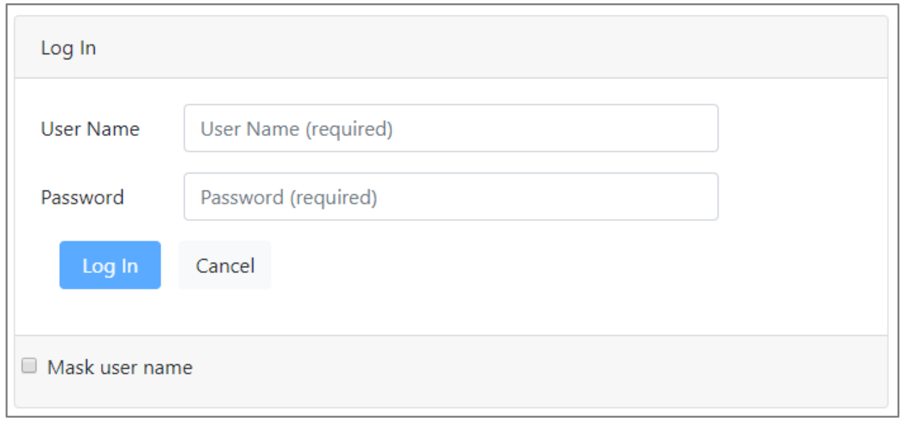
Subscribe:

```
this.store.pipe(select('products')).subscribe(
 products => this.displayCode = products.showProductCode
);
```

Receives notifications when the state changes


View
State
User Event
State
Component
Dispatch
Store (State)
State
Action
New State
Reducer

# Homework


Log In

User Name | User Name (required)

Password | Password (required)

Run the application and examine the login page

Initialize the store in the user module

Define the state (maskUserName) and action (MASK_USER_NAME)

Create the reducer function

Dispatch an action

Subscribe to the user slice of state

https://github.com/DeborahK/Angular-NgRx-GettingStarted/tree/master/APM-Demo1