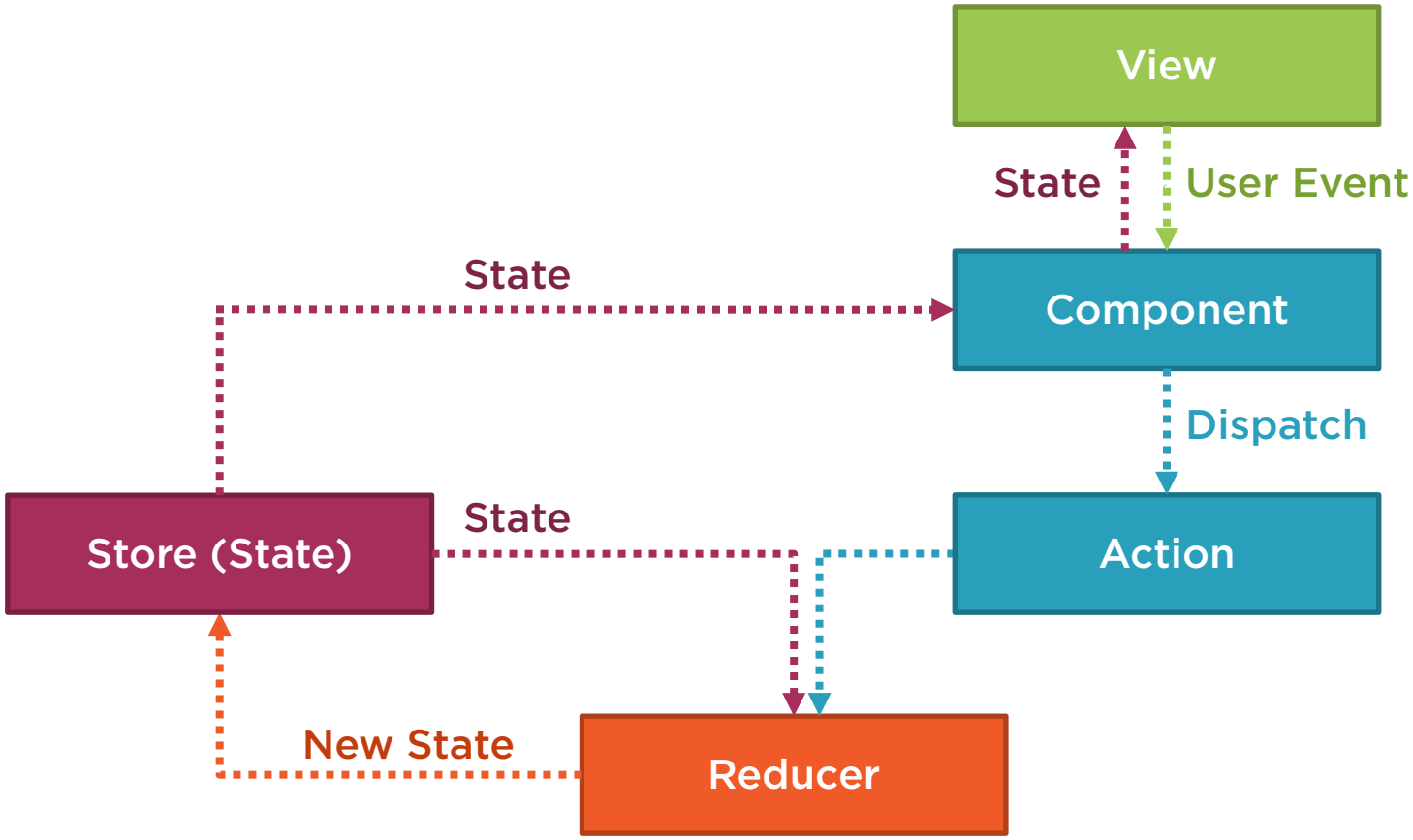
# First Look at NgRx


**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

Add

# Module Overview

Set up the sample application

Install @ngrx/store

Initialize the store

Define the state and actions

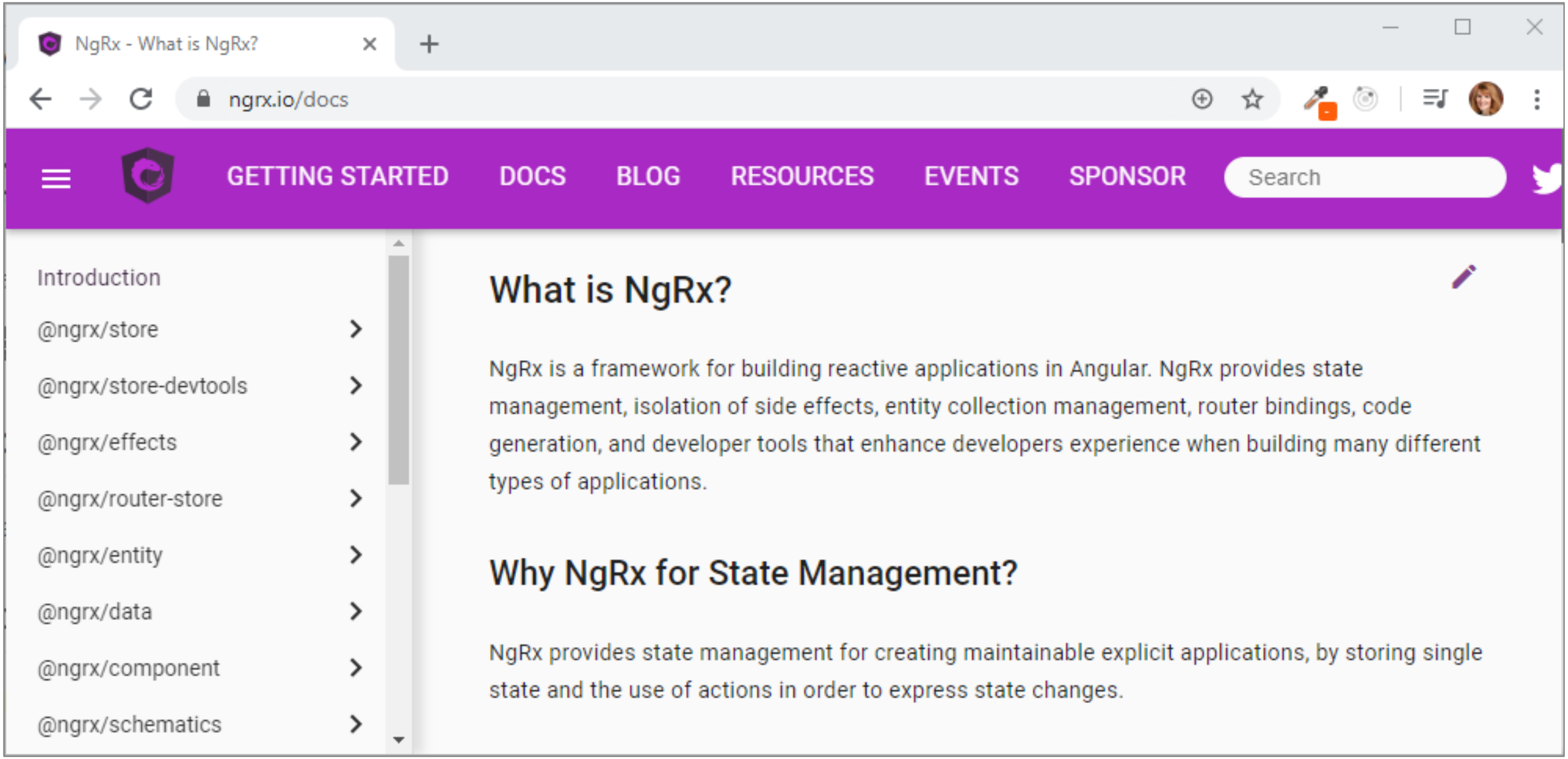
Build a reducer to process actions and set store state

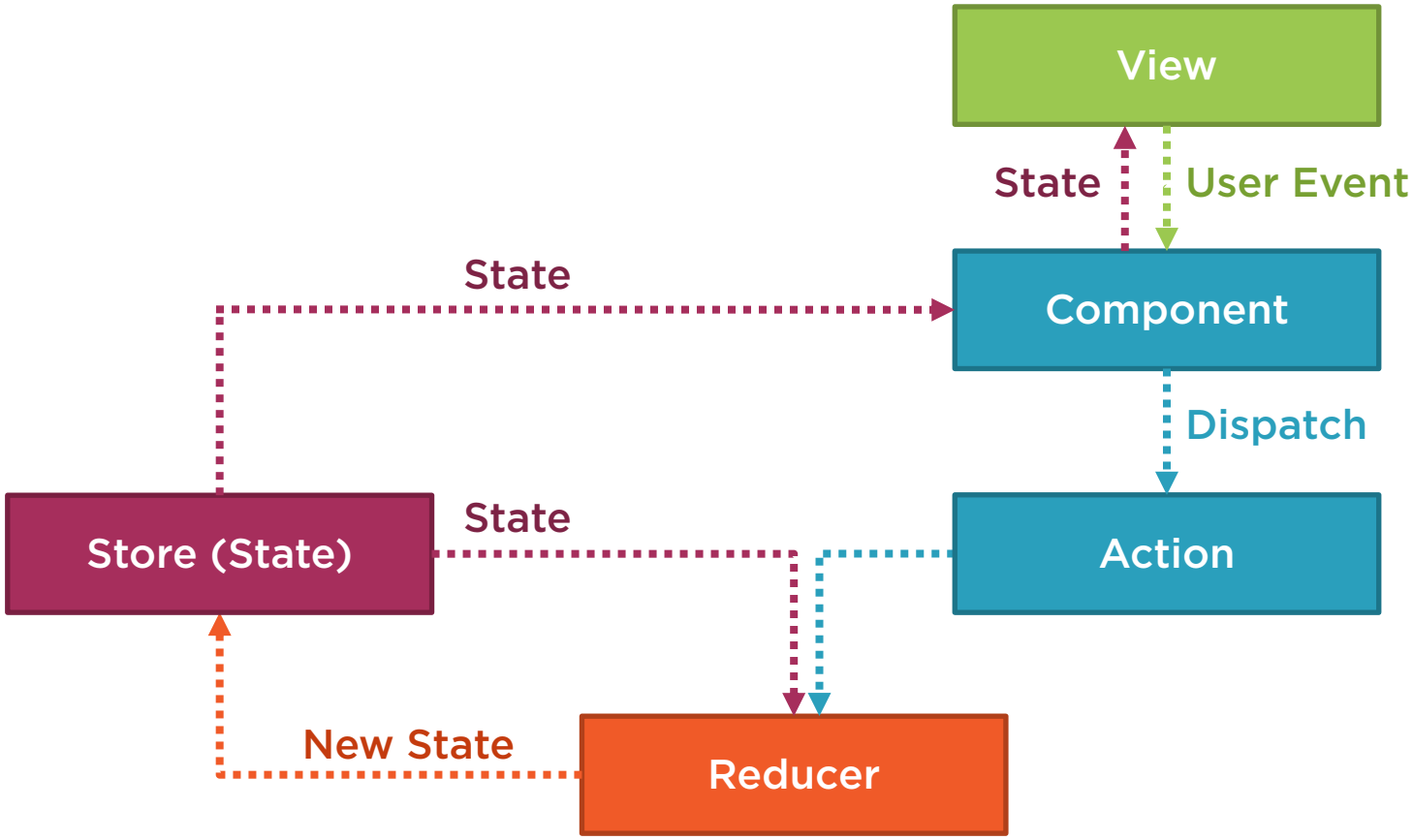
Dispatch an action to change state

Subscribe to state change notifications

# Demo

## Setting up the sample application


NgRx - What is NgRx?
ngrx.io/docs
GETTING STARTED
DOCS
BLOG
RESOURCES
EVENTS
SPONSOR
Search
Introduction
@ngrx/store
@ngrx/store-devtools
@ngrx/effects
@ngrx/router-store
@ngrx/entity
@ngrx/data
@ngrx/component
@ngrx/schematics
What is NgRx?
NgRx is a framework for building reactive applications in Angular. NgRx provides state management, isolation of side effects, entity collection management, router bindings, code generation, and developer tools that enhance developers experience when building many different types of applications.
Why NgRx for State Management?
NgRx provides state management for creating maintainable explicit applications, by storing single state and the use of actions in order to express state changes.


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)

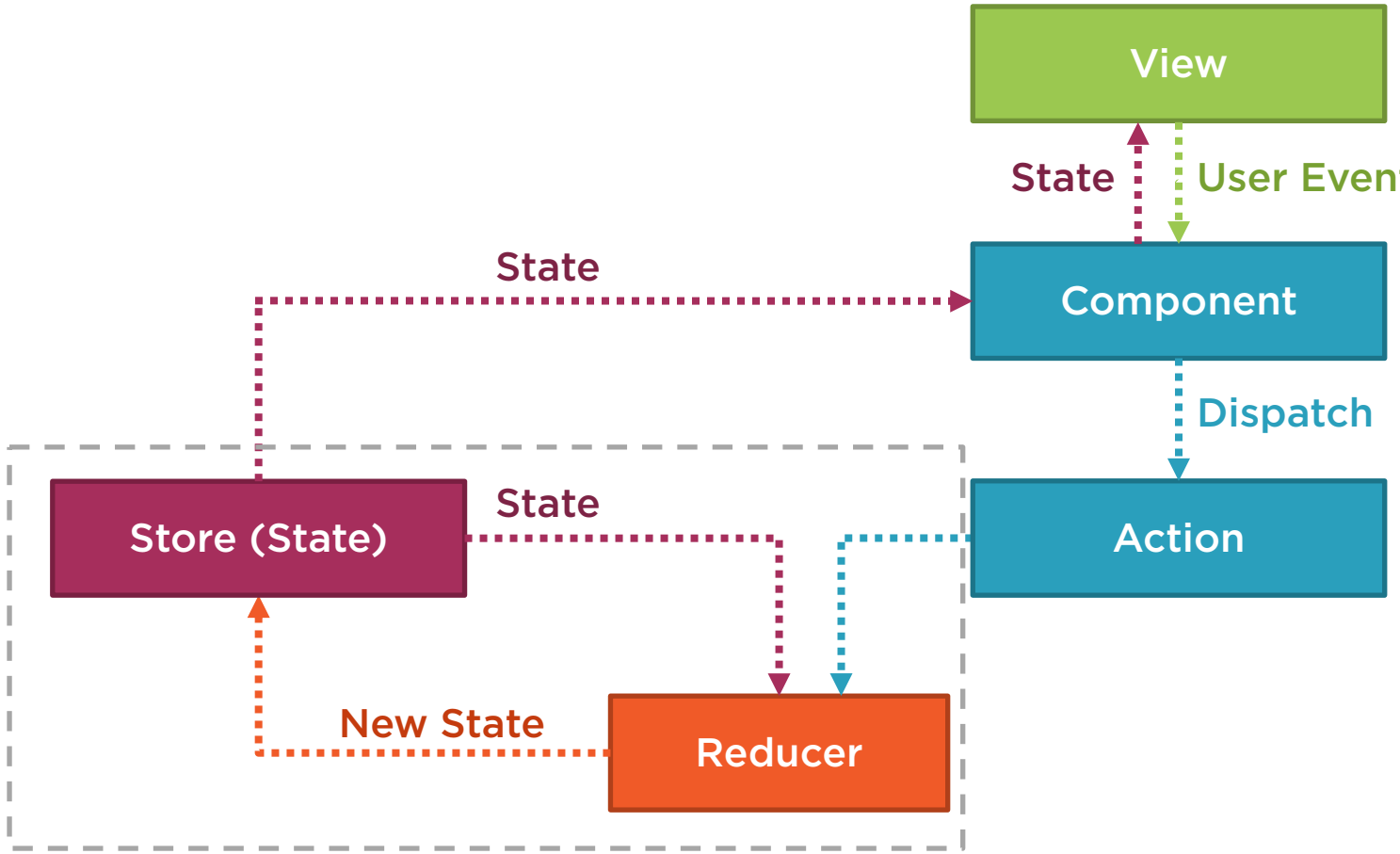
```
{
  showProductCode: true,
  currentProduct: {...},
  products: [...],
  hideWelcomePage: true,
  maskUserName: false,
  currentUser: {...},
  customerFilter: 'Harkness',
  currentCustomer: {...},
  customers: [...],
  allowGuest: false,
  includeAds: true,
  displayCustomerDetail: false,
  allowEdit: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

Store (State)

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

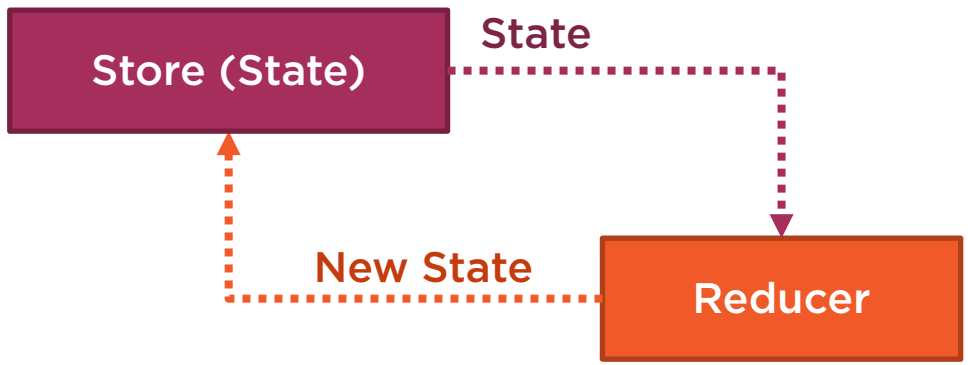
# Demo

## Installing @ngrx/store

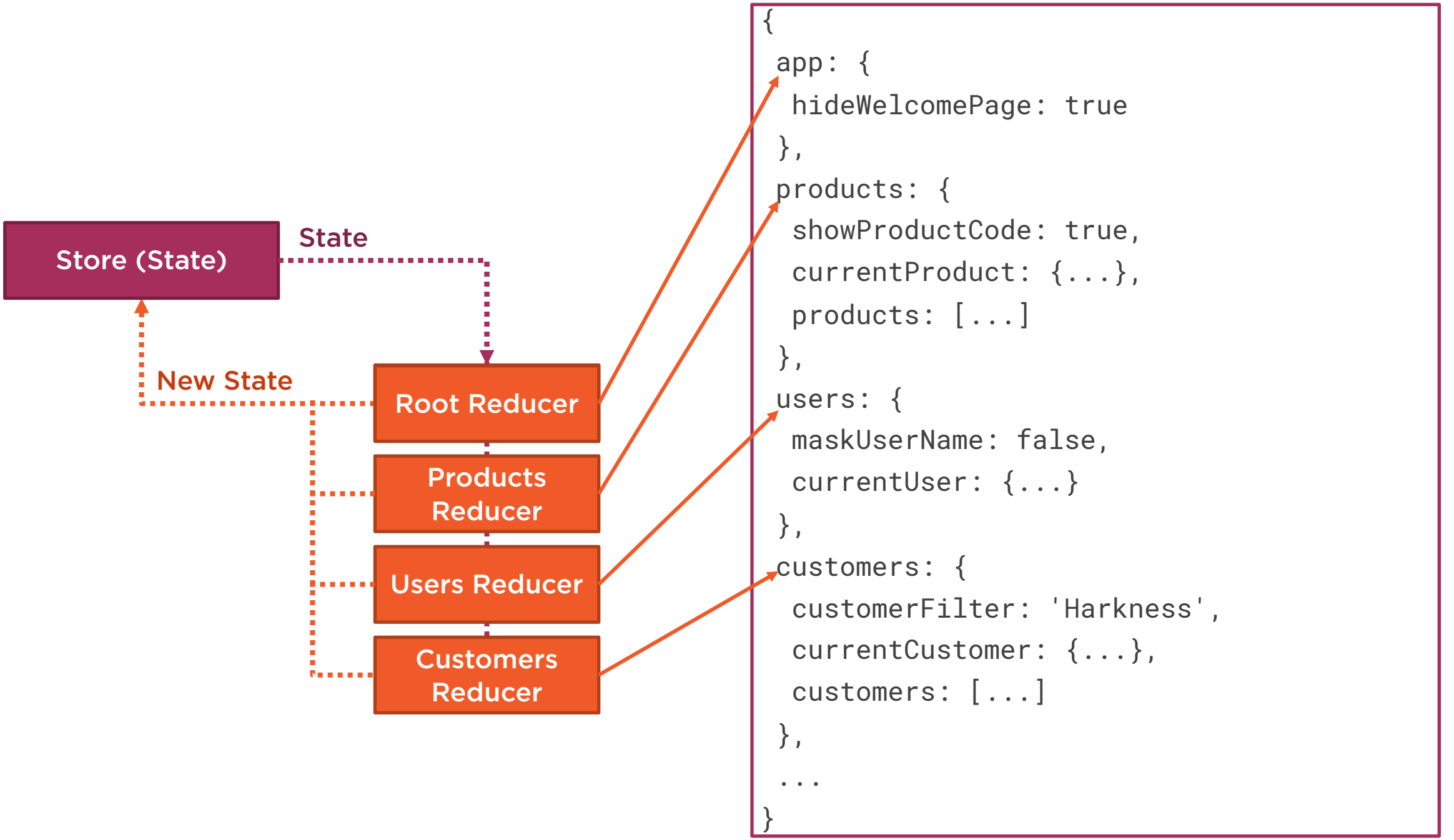



## App Module

```
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: [...],
  ...
}
```


Store (State)
State
New State
Root Reducer
Products Reducer
Users Reducer
Customers Reducer
{
app: {
hideWelcomePage: true
},
products: {
showProductCode: true,
currentProduct: {...},
products: [...]
},
users: {
maskUserName: false,
currentUser: {...}
},
customers: {
customerFilter: 'Harkness',
currentCustomer: {...},
customers: [...]
},
...
}

# Feature Module State Composition

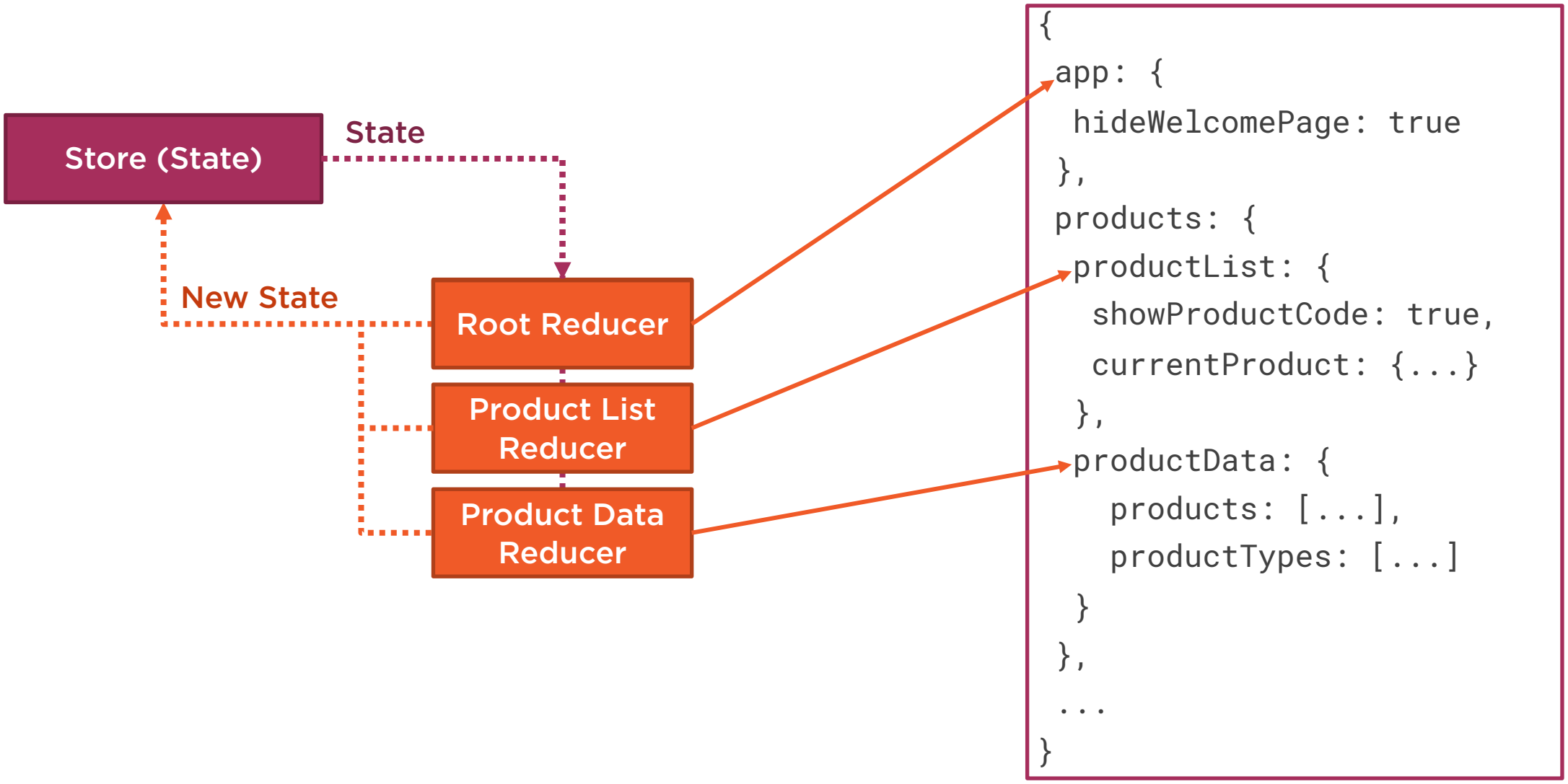
**App Module**

```
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', productReducer)
 ],
 declarations: [ ... ],
 providers: [ ... ]
})
export class ProductModule { }
```

# Sub-slice of State

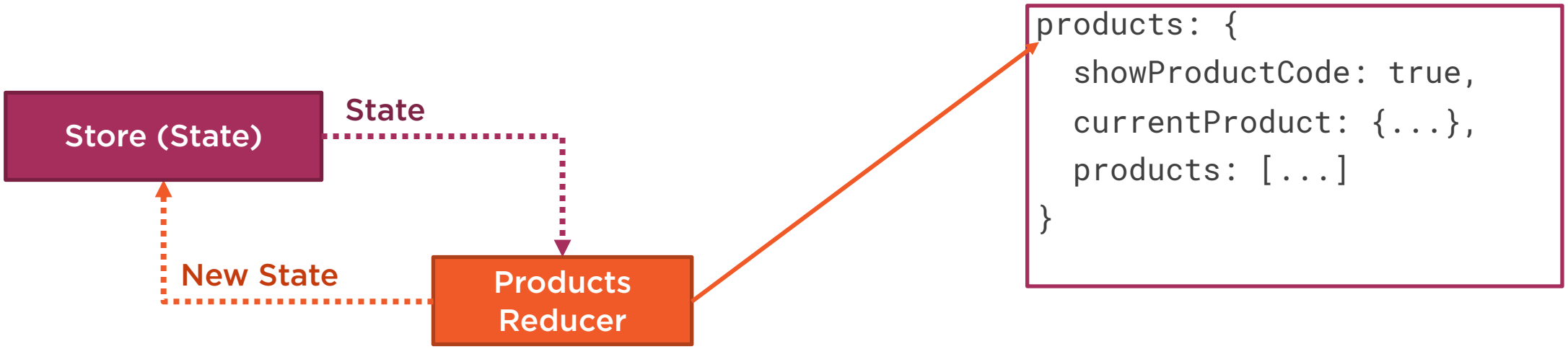


# Sub-slice of State

**Product Module**

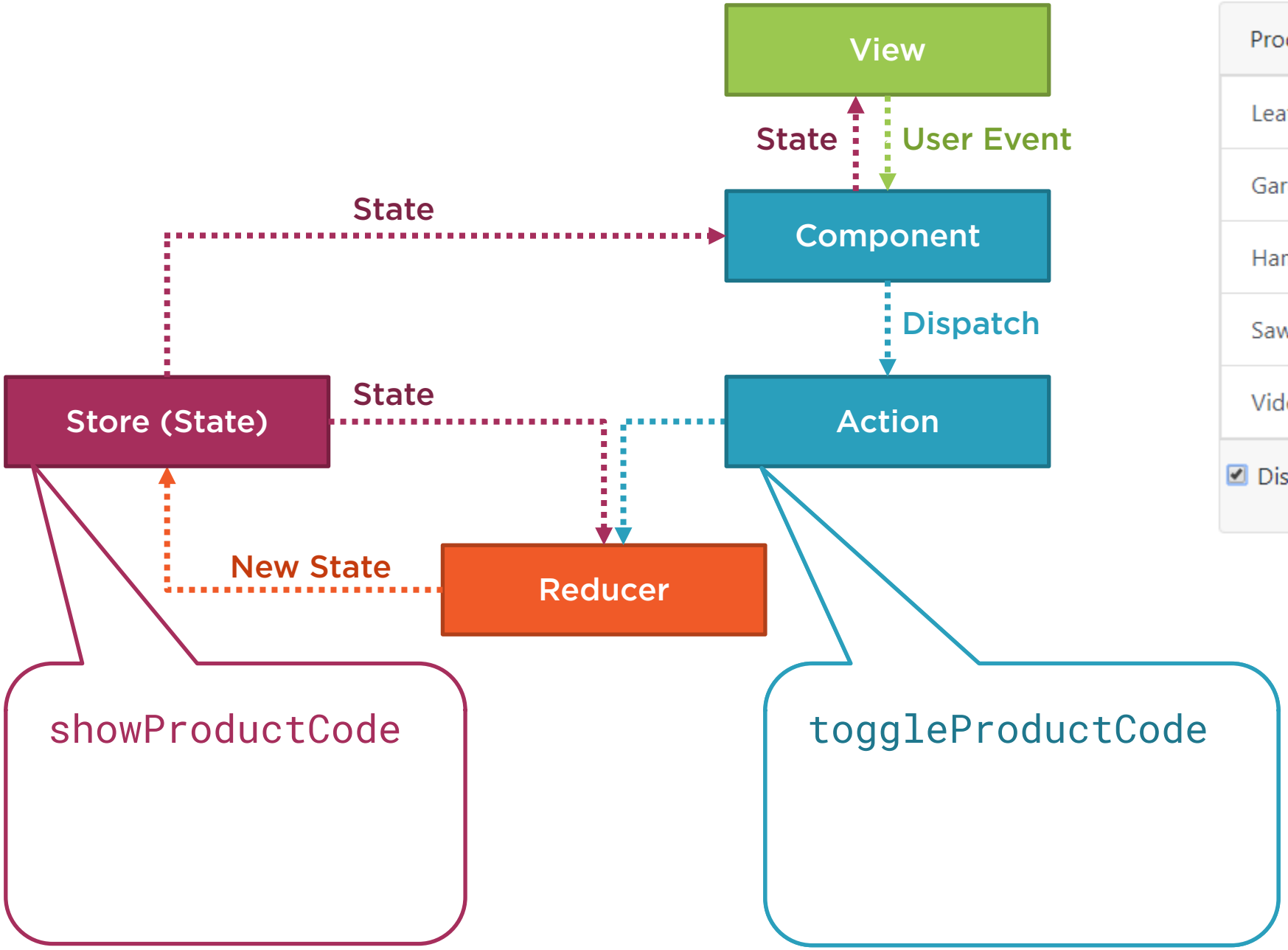
```
StoreModule.forFeature('products',
  {
     productList: listReducer,
     productData: dataReducer
  }
)
```

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


Store (State)
State
New State
Products
Reducer
products: {
showProductCode: true,
currentProduct: {...},
products: [...]
}

# Demo

## Initializing the Store




Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

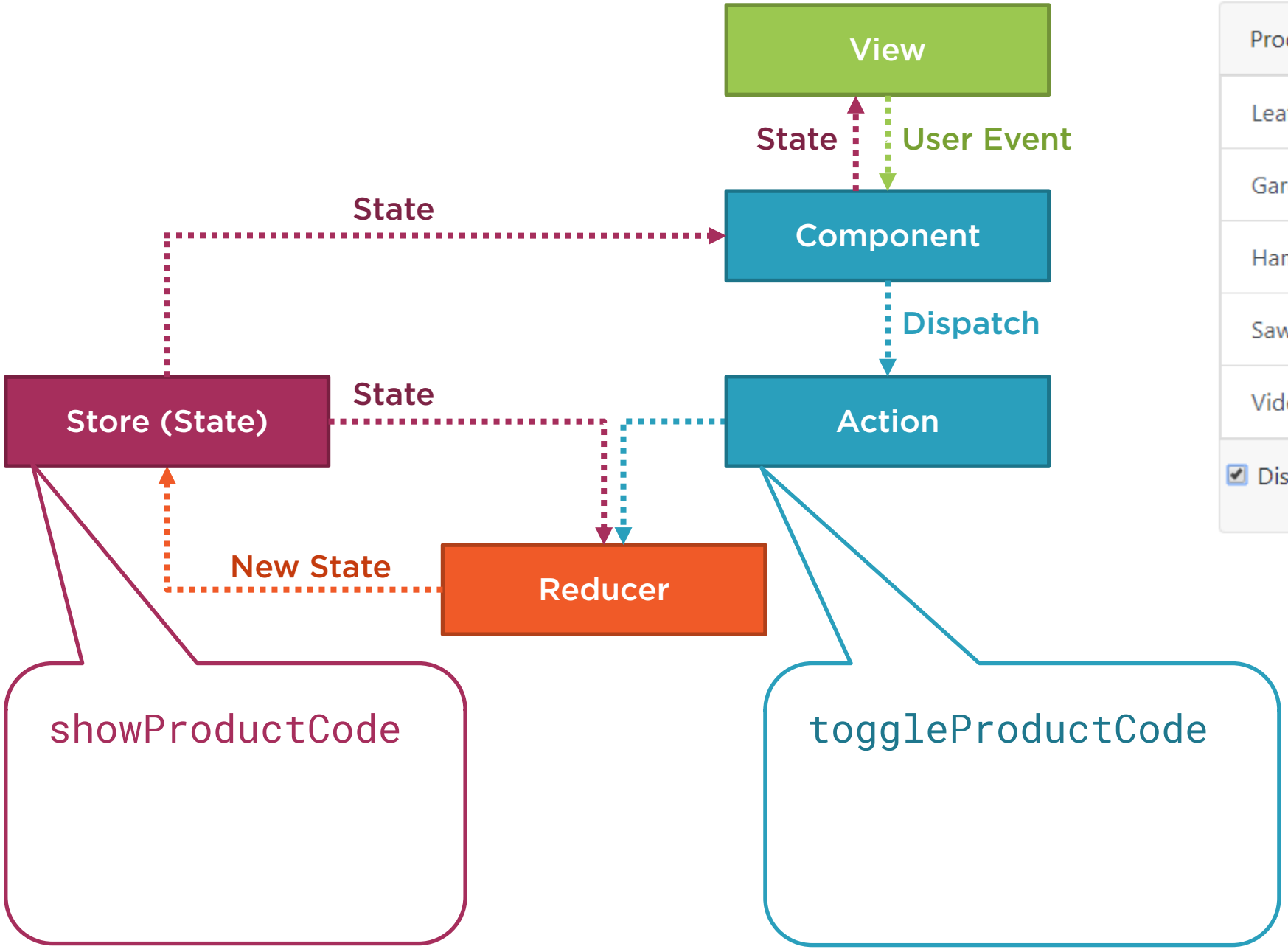
Hammer (TBX-0048)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

Display Product Code

Add




Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

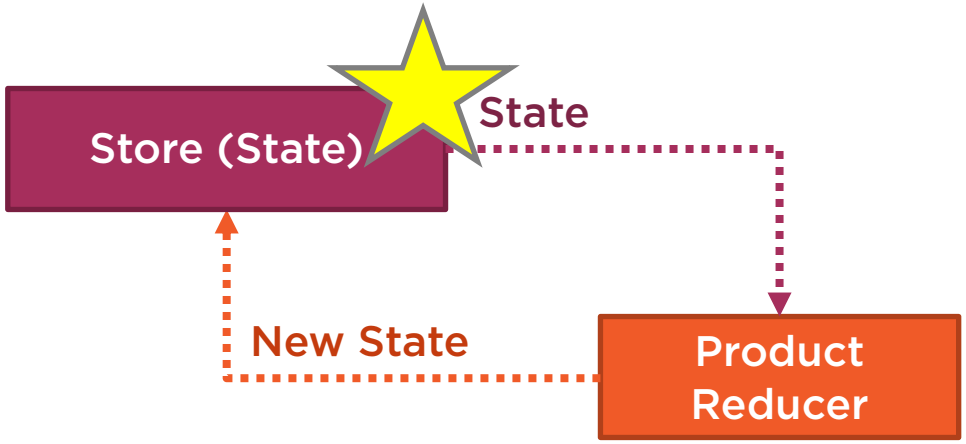
Hammer (TBX-0048)

Saw (TBX-0022)

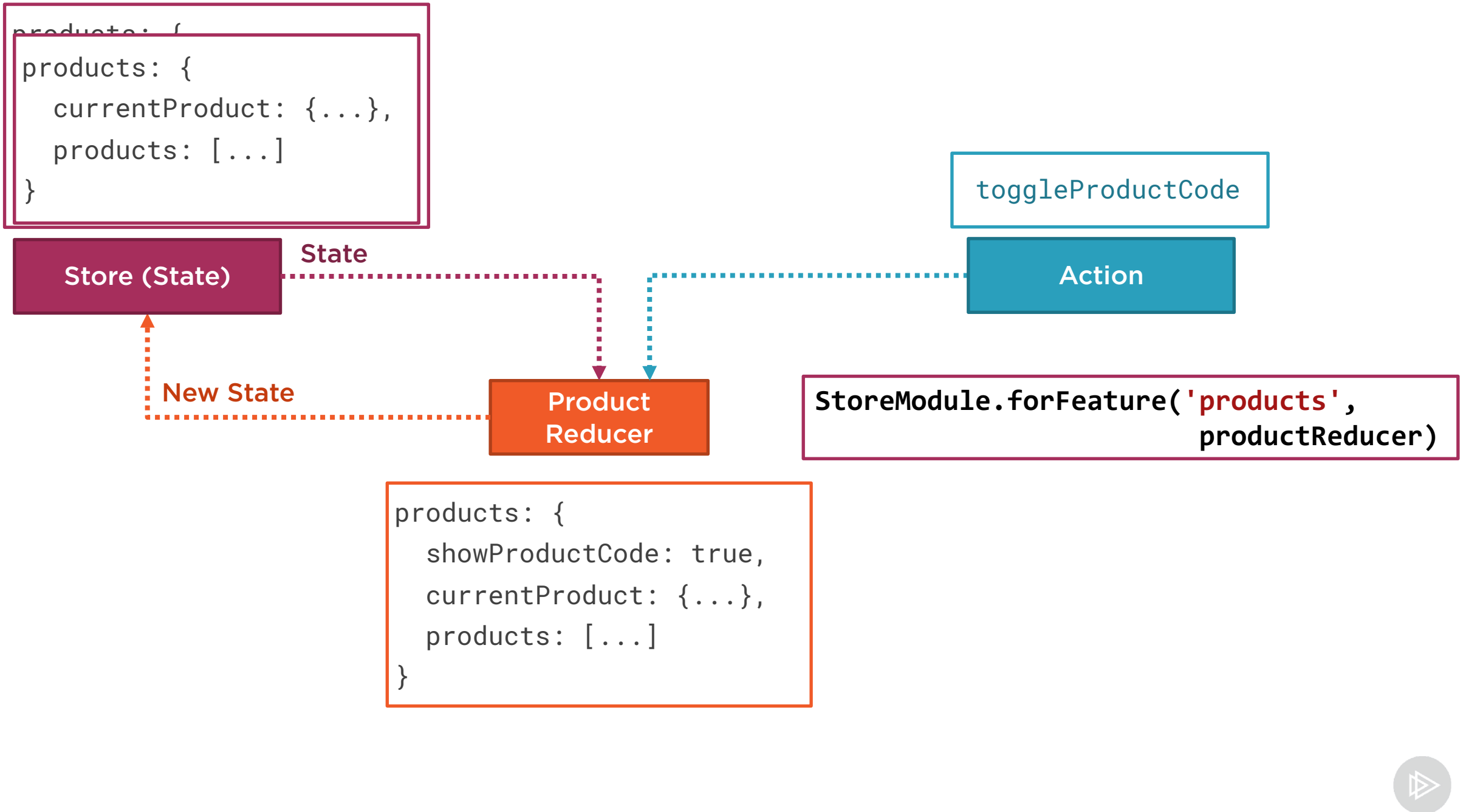
Video Game Controller (GMG-0042)

Display Product Code

Add




```
{
 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: [...]
}
Store (State)
State
toggleProductCode
Action
New State
Product Reducer
StoreModule.forFeature('products', productReducer)
products: {
showProductCode: true,
currentProduct: {...},
products: [...]
}

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

Store (State)

toggleProductCode

Action

Product Reducer

```
export const productReducer = createReducer(
  initialState,
  on(ProductActions.toggleProductCode, state => {
    return {
      ...state,
      showProductCode: !state.showProductCode
    };
  })
);
```

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

# Anatomy of a Reducer

**Product Reducer**

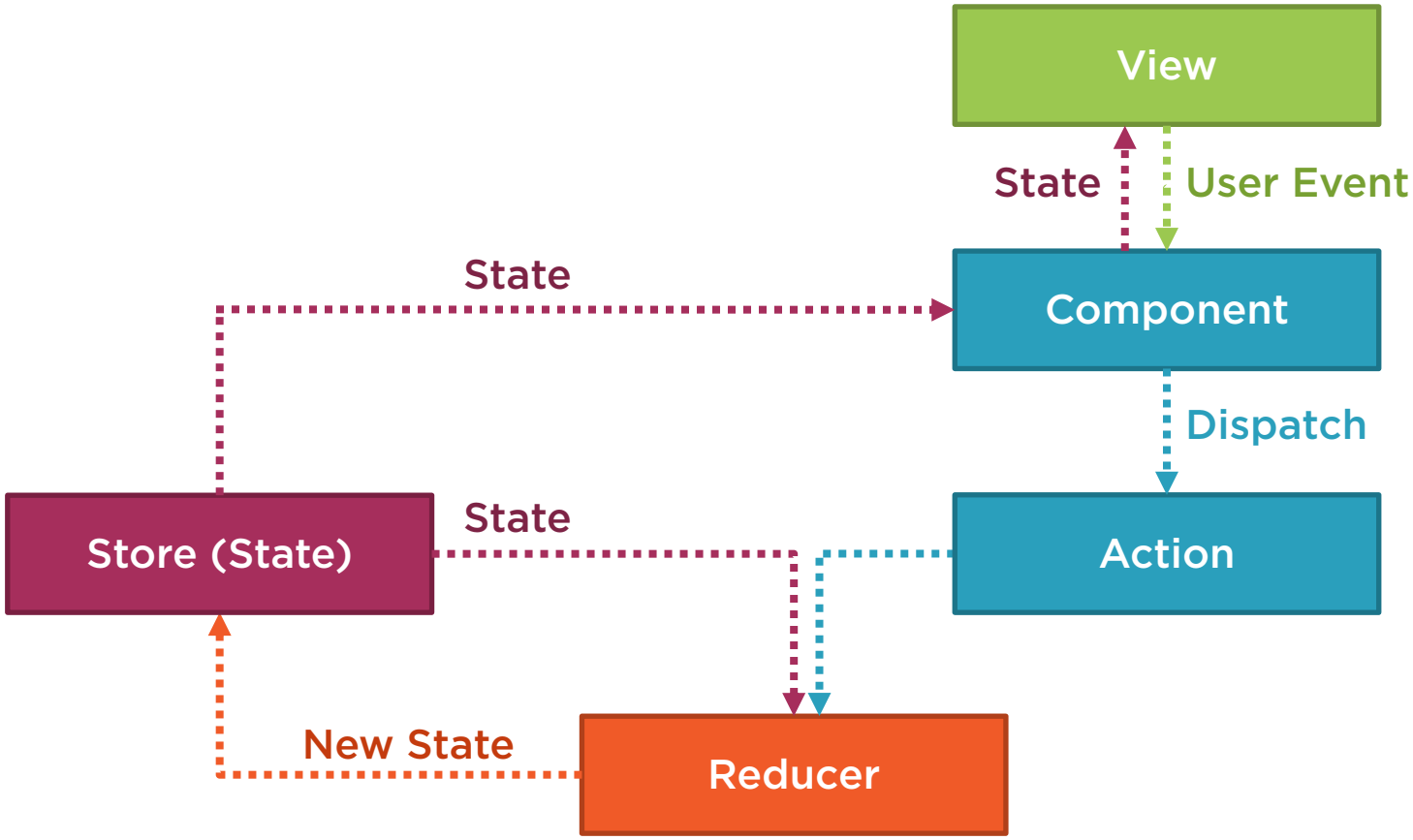
```
export const productReducer = createReducer (
  initialState,
  on(ProductActions.toggleProductCode,
    return {
      ...state,
      showProductCode: !state.showProdu
    };
  })
);
```

**Product Reducer**

```
export const productReducer = createReducer (
  initialState,
  on(ProductActions.showProductCode, state => {
    return {
      ...state,
      showProductCode: true
    };
  }),
  on(ProductActions.hideProductCode,
     ProductActions.resetDefaults, state => {
    return {
      ...state,
      showProductCode: false
    };
  })
);
```

# Demo

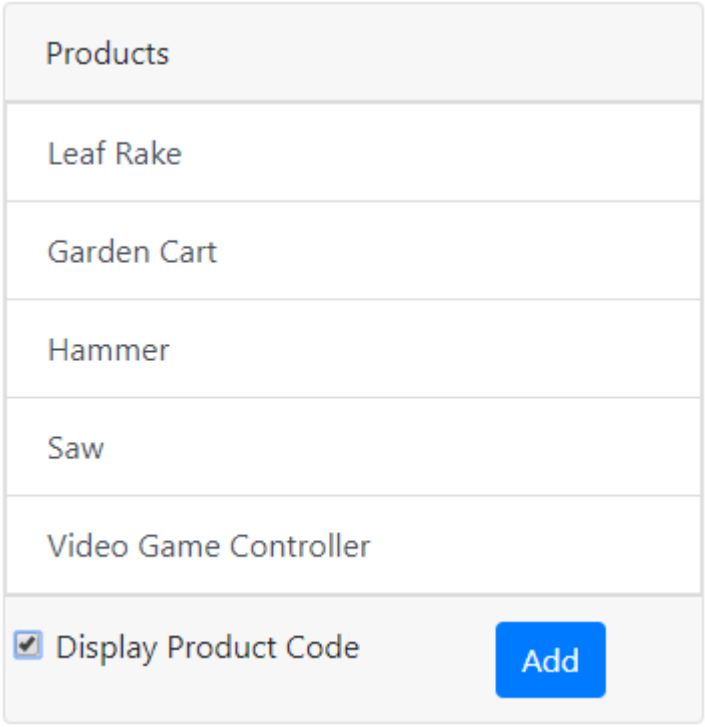
## Building a reducer to process actions


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

# Dispatching an Action

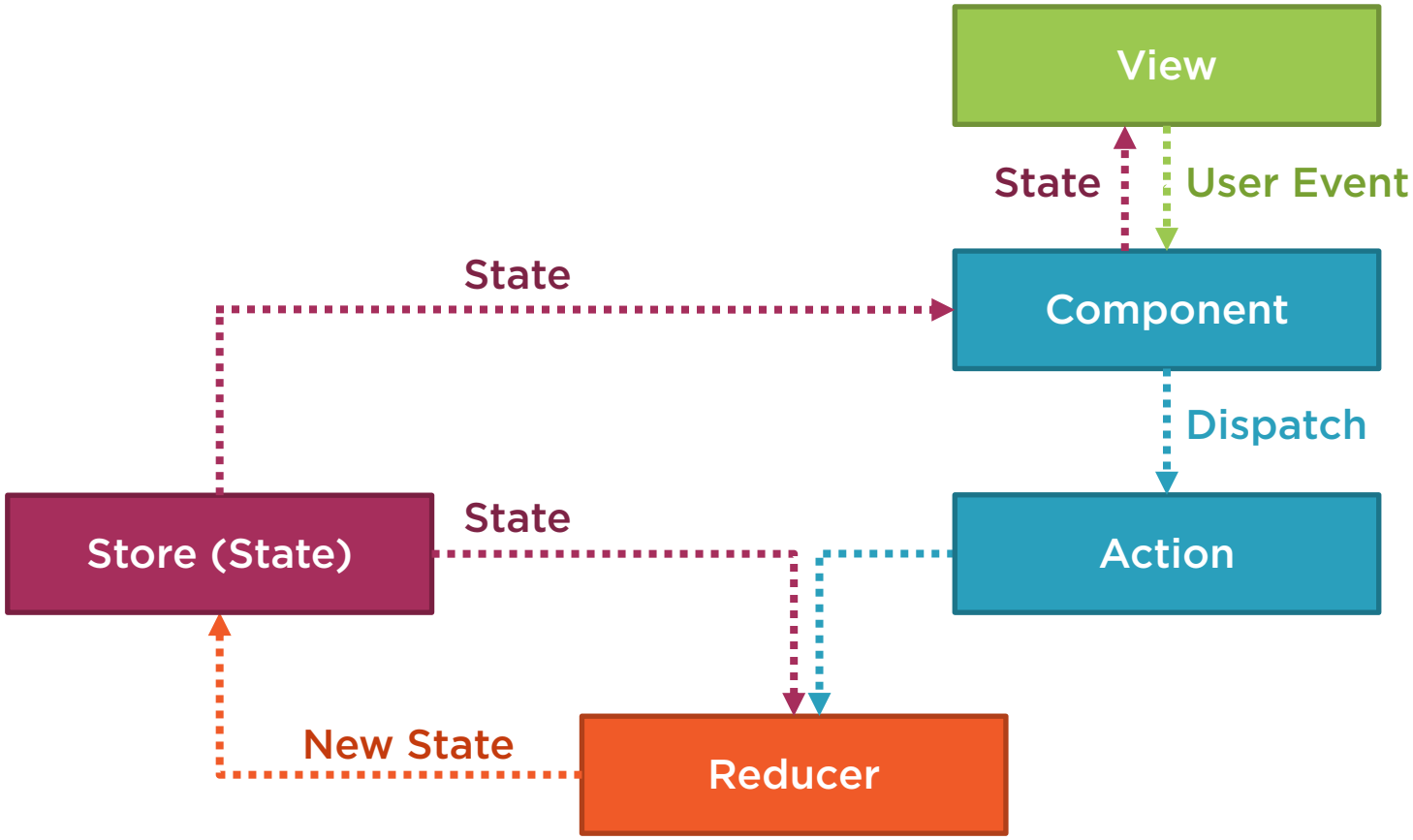
**Product List Component**

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



# Demo

## Dispatching an action


View
State
User Event
State
Component
Dispatch
Store (State)
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.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', featureReducer)
```

# Checklist: Action

Action

An action represents an event

Define an action for each event worth tracking

# Checklist: Reducer

Reducer

Responds to dispatched actions

Replaces the state tree with new state

Build a reducer function (often one per feature)

Implement using createReducer:

```
export const productReducer = createReducer(
  initialState,
  on(ProductActions.toggleProductCode, state => {
    return {
      ...state,
      showProductCode: !state.showProductCode
    };
  }));
```

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: '[Product] Toggle Product Code'
});
```

Dispatched to all reducers

# Checklist: Subscribing to the Store

Store (State)

**Often done in the ngOnInit lifecycle hook**

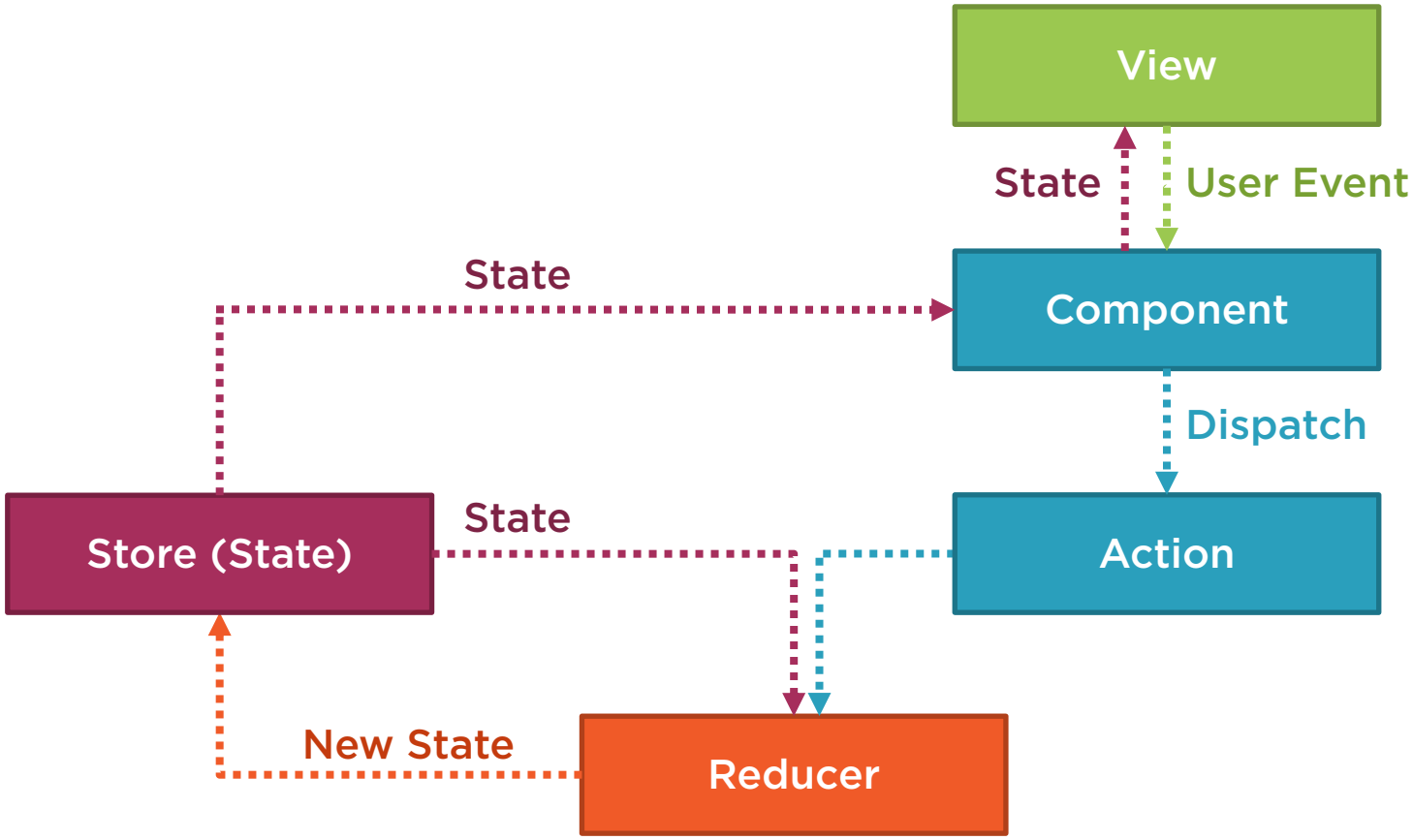
**Inject the store in the constructor**

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

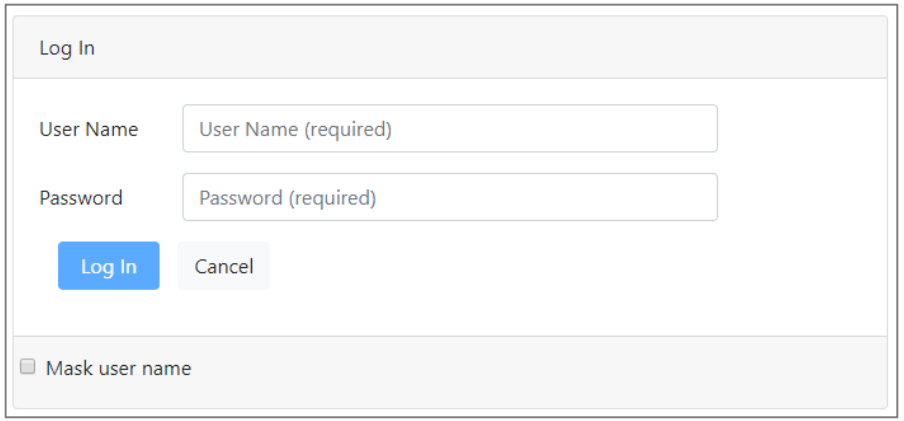
**Subscribe:**

```
this.store.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




Run the application and examine the login page

Initialize the store in the user module

Define the state (maskUserName) and action ('[User] 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