

# Strongly Typing Actions with Action Creators

---



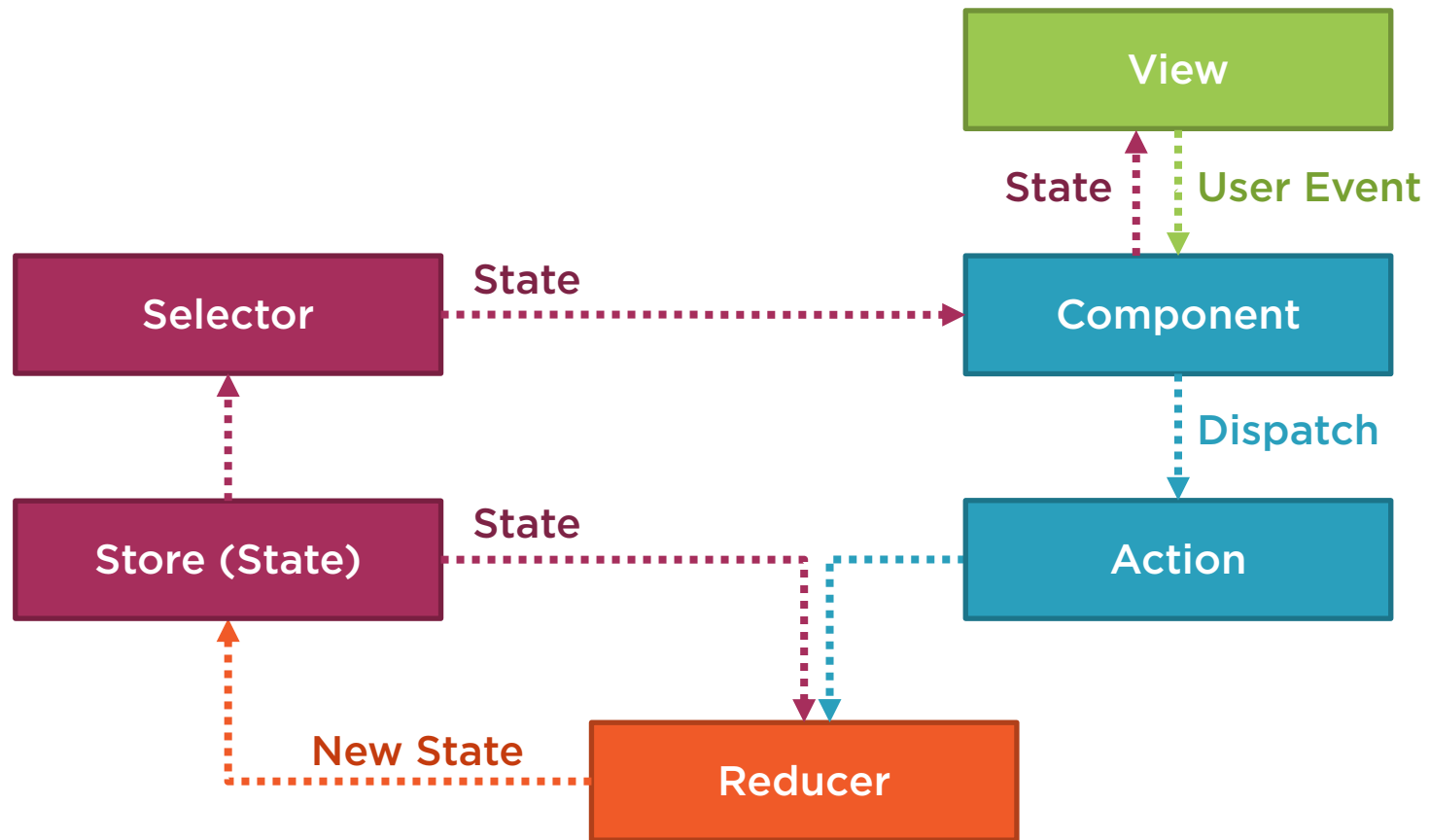
**Deborah Kurata**

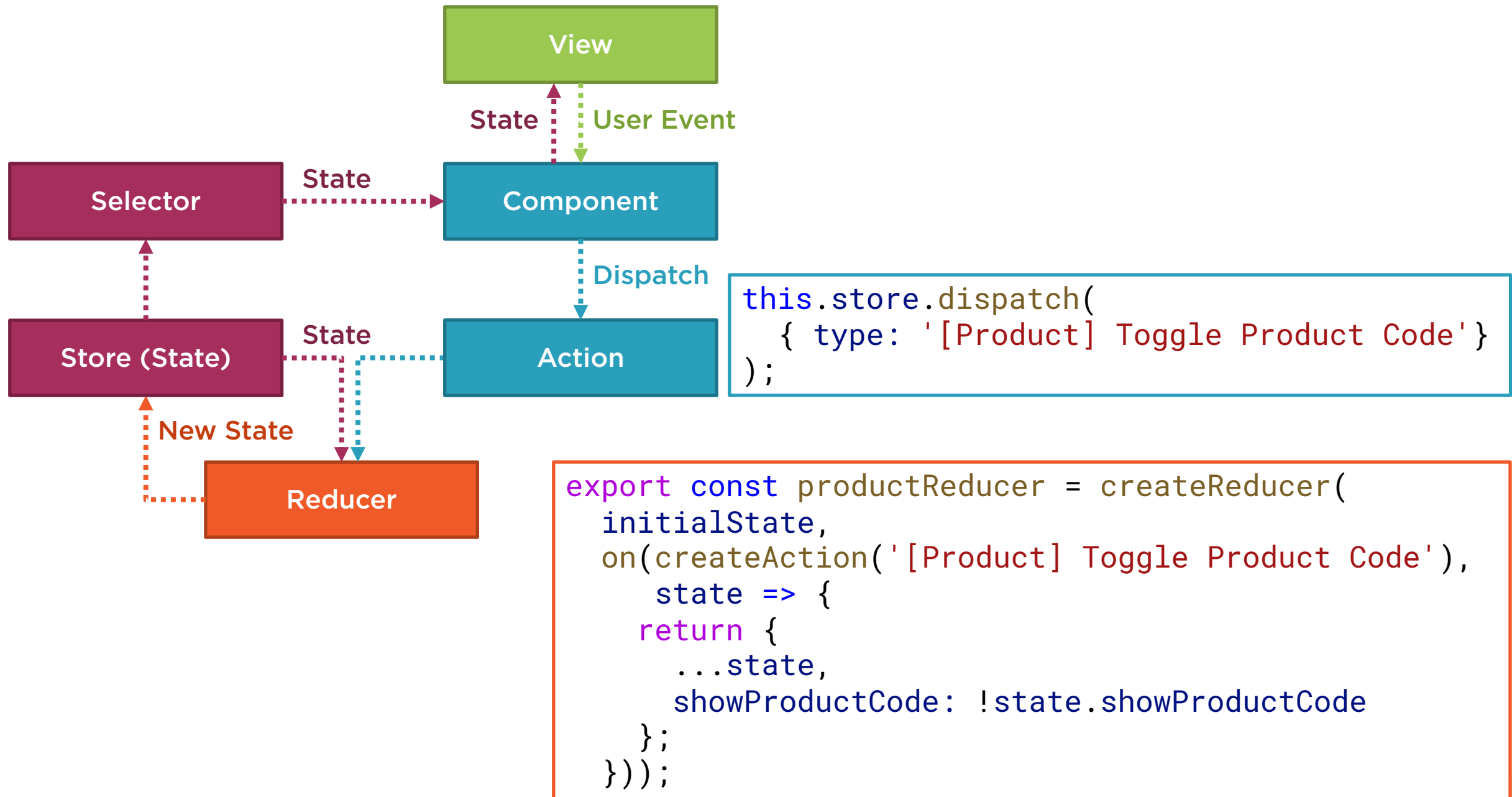
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata [blogs.msmvps.com/deborahk/](https://blogs.msmvps.com/deborahk/)









# Demo



The state of our actions



# Module Overview



**Build action creators to strongly type actions**

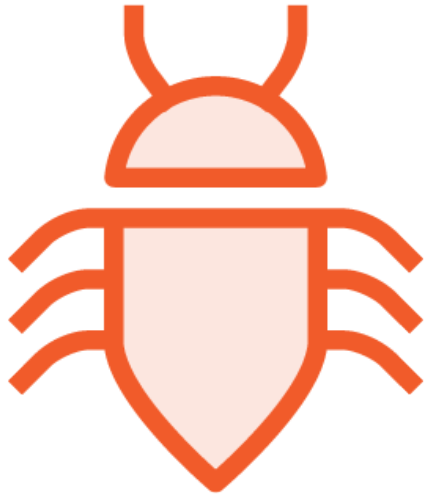
**Use action creators to create and dispatch actions**

**Use actions and selectors to communicate between our components**

**Define actions for more complex operations**



# Benefits of Strongly Typed Actions



Prevents hard to find  
errors



Improves the tooling  
experience



Documents the set of  
valid actions

# Action Creators Using createAction

```
export const productReducer = createReducer(  
  initialState,  
  on(createAction('[Product] Toggle Product Code'), state => {  
    return {...};  
  }));
```

```
export const toggleProductCode = createAction('[Product] Toggle Product Code');
```

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

```
this.store.dispatch(toggleProductCode());
```





# Defining Actions

Acme Product Management

Home

Product List

Log In

Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

☒ Display Product Code

Add

Add Product

Product Name

Name (required)

Product Code

New

Star Rating (1-5)

0

Description

Description

Save

Cancel

Delete

## 1. Toggle Product Code



# Defining Actions

```
export const toggleProductCode = createAction('[Product] Toggle Product Code');
export const setCurrentProduct = createAction('[Product] Set Current Product');
export const clearCurrentProduct = createAction('[Product] Clear Current Product');
export const initCurrentProduct = createAction('[Product] Init Current Product');
```

The screenshot displays the APM Demo App DevTools interface. On the left, the 'Inspector' panel shows a list of actions with a search filter and a 'Commit' button. The actions listed are:

Action	Time
[User] Mask User Name	+00:19.75
[Product] Toggle Product Code	+00:11.99
[Product] Set Current Product	+00:11.98
[Product] Clear Current Product	+00:07.34
[Product] Set Current Product	+00:16.30

On the right, the 'Action' panel shows a tree view of the selected action. The tree structure is:

```
product (pin): { id: 5, productName: "Hammer", productCode: "TBX-0048", ... }
type (pin): "[Product] Set Current Product"
```

At the bottom, there is a timeline with a play button, a progress bar, and a '1x' speed selector. The bottom bar contains several utility buttons: Pause recording, Lock changes, Persist, Dispatcher, Slider, Import, Export, Remote, and Settings.



# Defining Actions

```
export const toggleProductCode =
    createAction('[Product List Page] Toggle Product Code');
export const setCurrentProduct =
    createAction('[Product List Page] Set Current Product ');
export const clearCurrentProduct =
    createAction('[Product Edit Page] Clear Current Product');
export const initCurrentProduct =
    createAction('[Product List Page] Initialize Current Product');
export const loadProducts =
    createAction('[Product List Page] Load');
export const loadProductsSuccess =
    createAction('[Product API] Load Success');
export const loadProductsFailure =
    createAction('[Product API] Load Fail');
```

```
export const toggleProductCode = createAction('[Product] Toggle Product Code');
export const setCurrentProduct = createAction('[Product] Set Current Product');
export const clearCurrentProduct = createAction('[Product] Clear Current Product');
export const initCurrentProduct = createAction('[Product] Init Current Product');
```

# Defining Actions with Data

Acme Product Management

Home

Product List

Log In

Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

Hammer (TBX-0048)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

☒ Display Product Code

Add

Edit Product: Hammer

Product Name

Hammer

Product Code

TBX-0048

Star Rating (1-5)

4.8

Description

Curved claw steel hammer

Save

Cancel

Delete

Set Current Product



# Defining Actions with Data

```
export const setCurrentProduct = createAction(  
  '[Product] Set Current Product',  
  props<{ product: Product }>()  
);
```

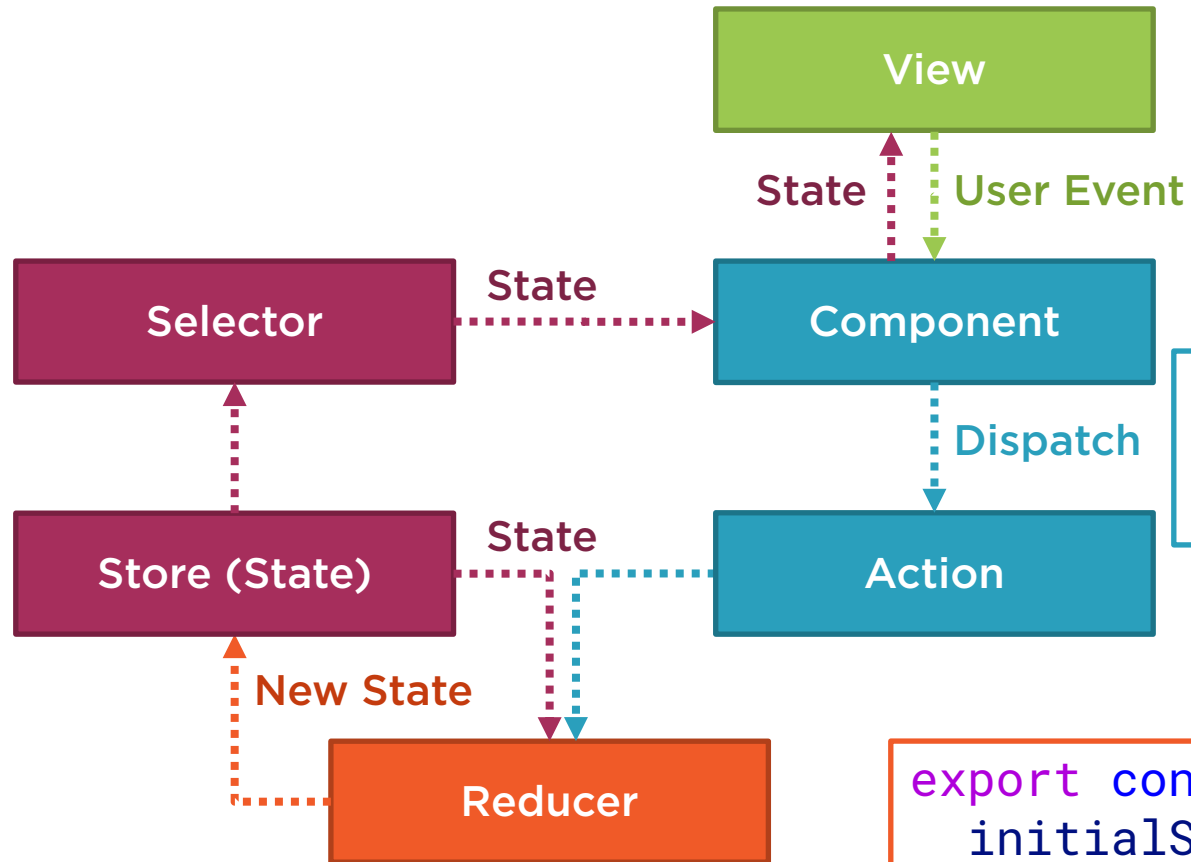


# Demo



## Building action creators





```
this.store.dispatch(  
  { type: '[Product] Toggle Product Code' }  
);
```

```
export const productReducer = createReducer(  
  initialState,  
  on(createAction('[Product] Toggle Product Code'),  
    state => {  
      return {  
        ...state,  
        showProductCode: !state.showProductCode  
      };  
    }  
  ));
```

# Using an Action in the Reducer

```
export const productReducer = createReducer<ProductState>(
  initialState,
  on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
    return {...};
  })
);
```

```
import * as ProductActions from './product.actions';

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





# Processing an Action with Data

```
import * as ProductActions from './product.actions';

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



# Dispatching an Action

```
this.store.dispatch({ type: '[Product] Toggle Product Code' });
```

```
import * as ProductActions from '../state/product.actions';  
this.store.dispatch(ProductActions.toggleProductCode());
```



# Dispatching an Action with Data

```
import * as ProductActions from '../state/product.actions';

productSelected(product: Product): void {
  this.store.dispatch(ProductActions.setCurrentProduct(
    { product: product }
  ));
}
```

```
import * as ProductActions from '../state/product.actions';

productSelected(product: Product): void {
  this.store.dispatch(ProductActions.setCurrentProduct({ product }));
}
```



# Demo



## Using action creators



# Component Communication

Acme Product Management

HomeProduct List

Welcome DeborahLog Out

Products

Leaf Rake (GDN-0011)

Garden Cart (GDN-0023)

Hammer (TBX-0048)

Saw (TBX-0022)

Video Game Controller (GMG-0042)

☒ Display Product Code

Add

Edit Product: Hammer

Product Name

Hammer

Product Code

TBX-0048

Star Rating (1-5)

4.8

Description

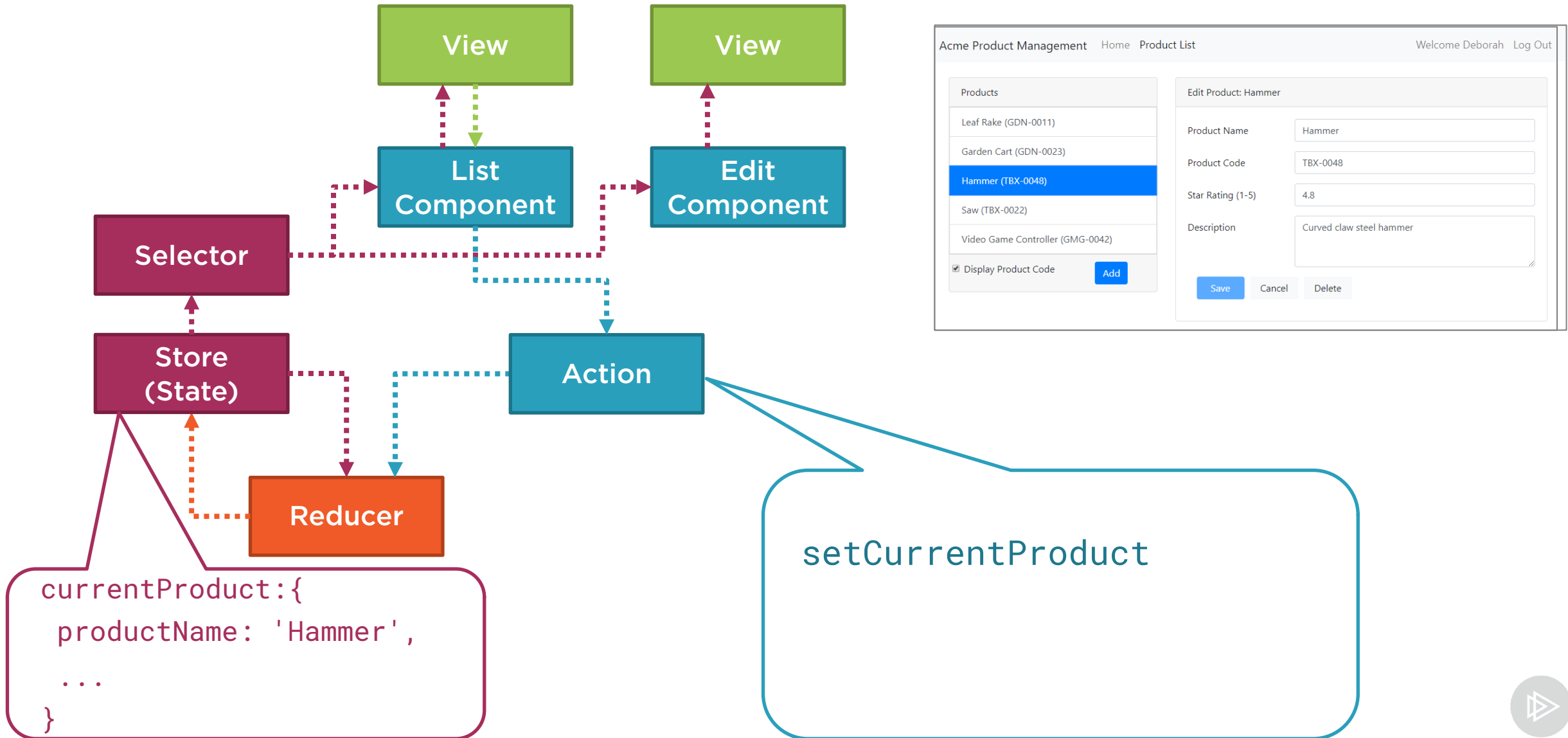
Curved claw steel hammer

Save

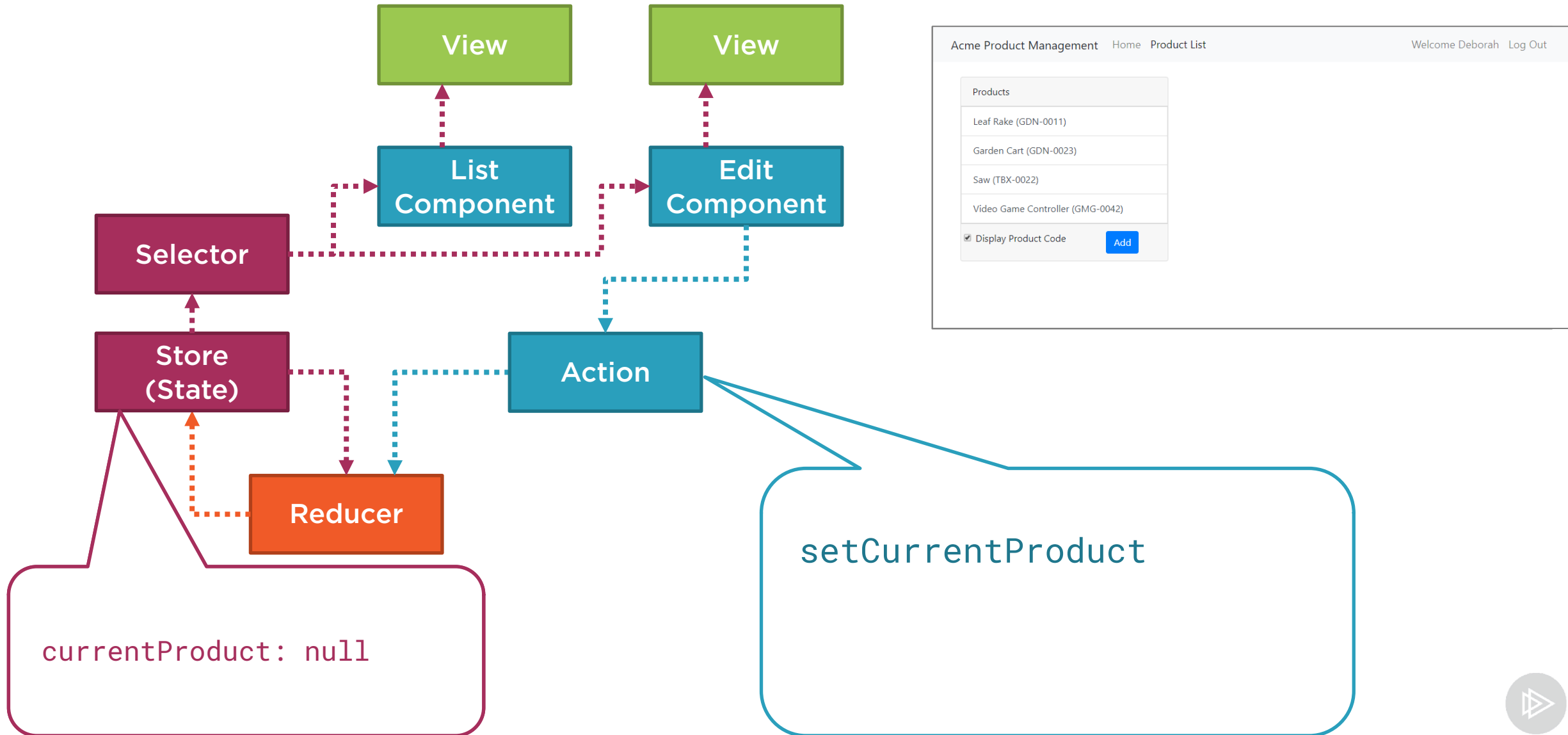
Cancel

Delete

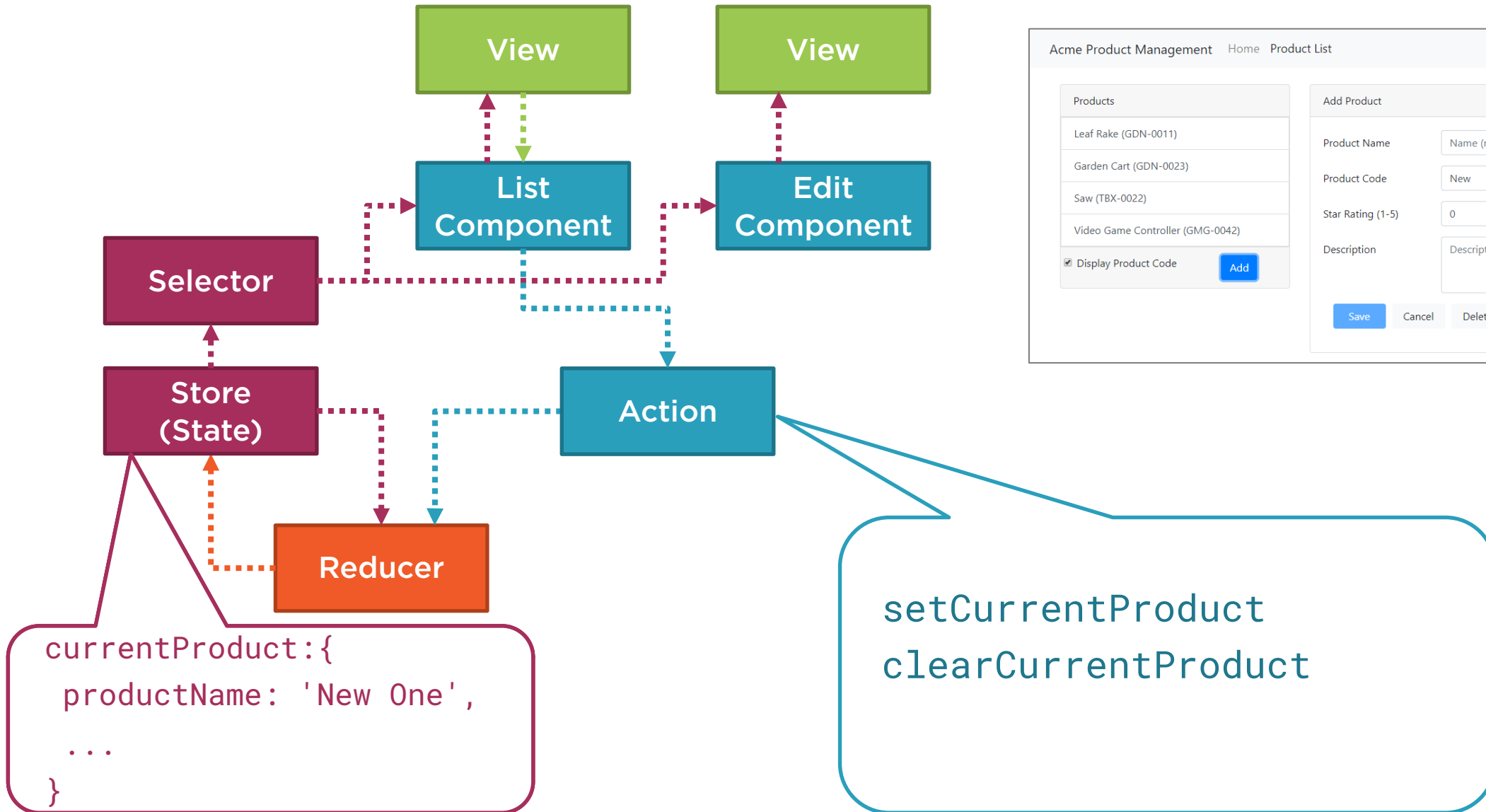
# Component Communication



# Component Communication



# Component Communication



Acme Product Management Home Product List Welcome Deborah Log Out

Products
Leaf Rake (GDN-0011)
Garden Cart (GDN-0023)
Saw (TBX-0022)
Video Game Controller (GMG-0042)
<input checked="" type="checkbox"/> Display Product Code <input type="button" value="Add"/>

Add Product	
Product Name	<input type="text" value="Name (required)"/>
Product Code	<input type="text" value="New"/>
Star Rating (1-5)	<input type="text" value="0"/>
Description	<input type="text" value="Description"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/> <input type="button" value="Delete"/>	





# Demo



Using actions and selectors for  
component communication

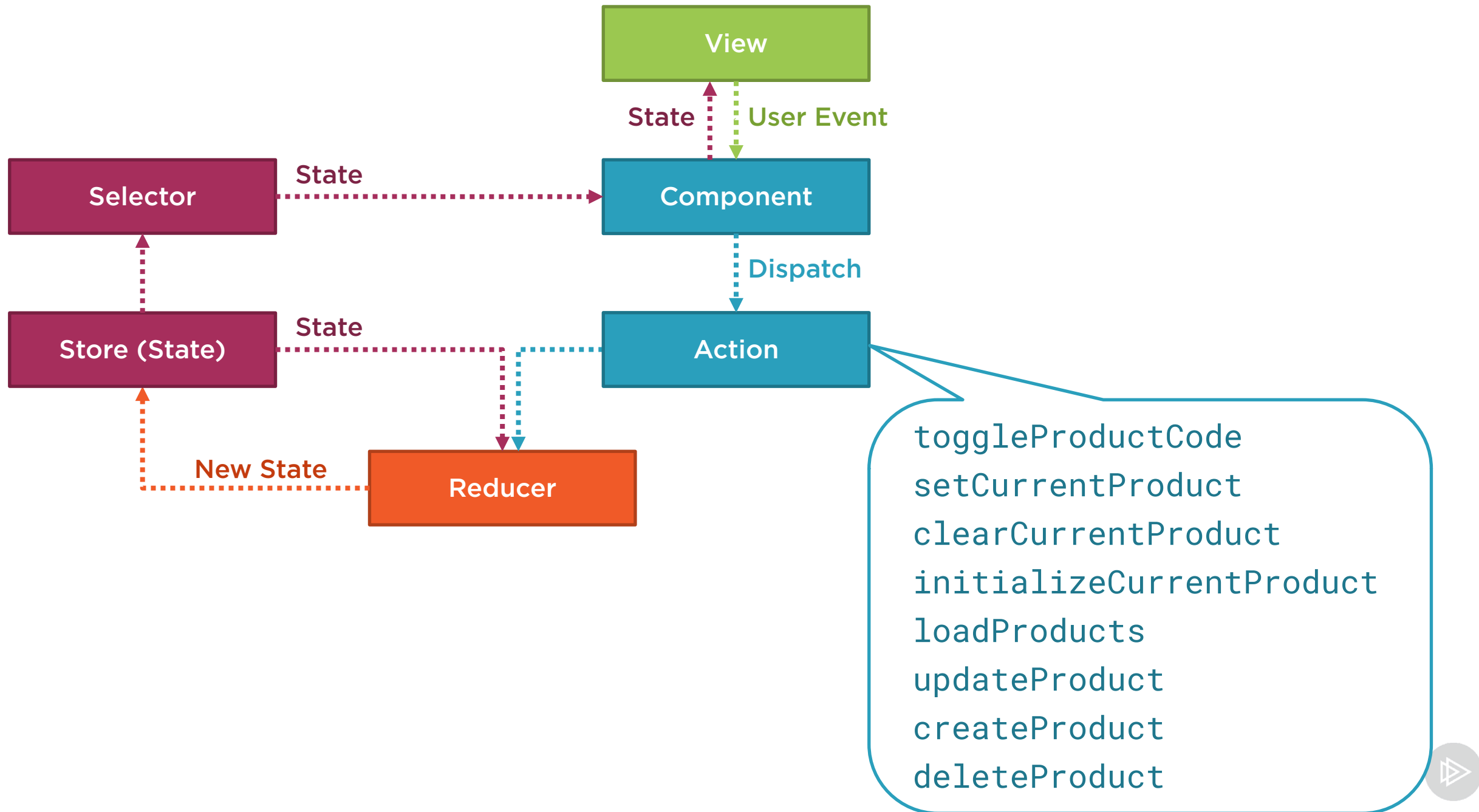


# Demo



Using actions and selectors for  
component communication:  
Edit component



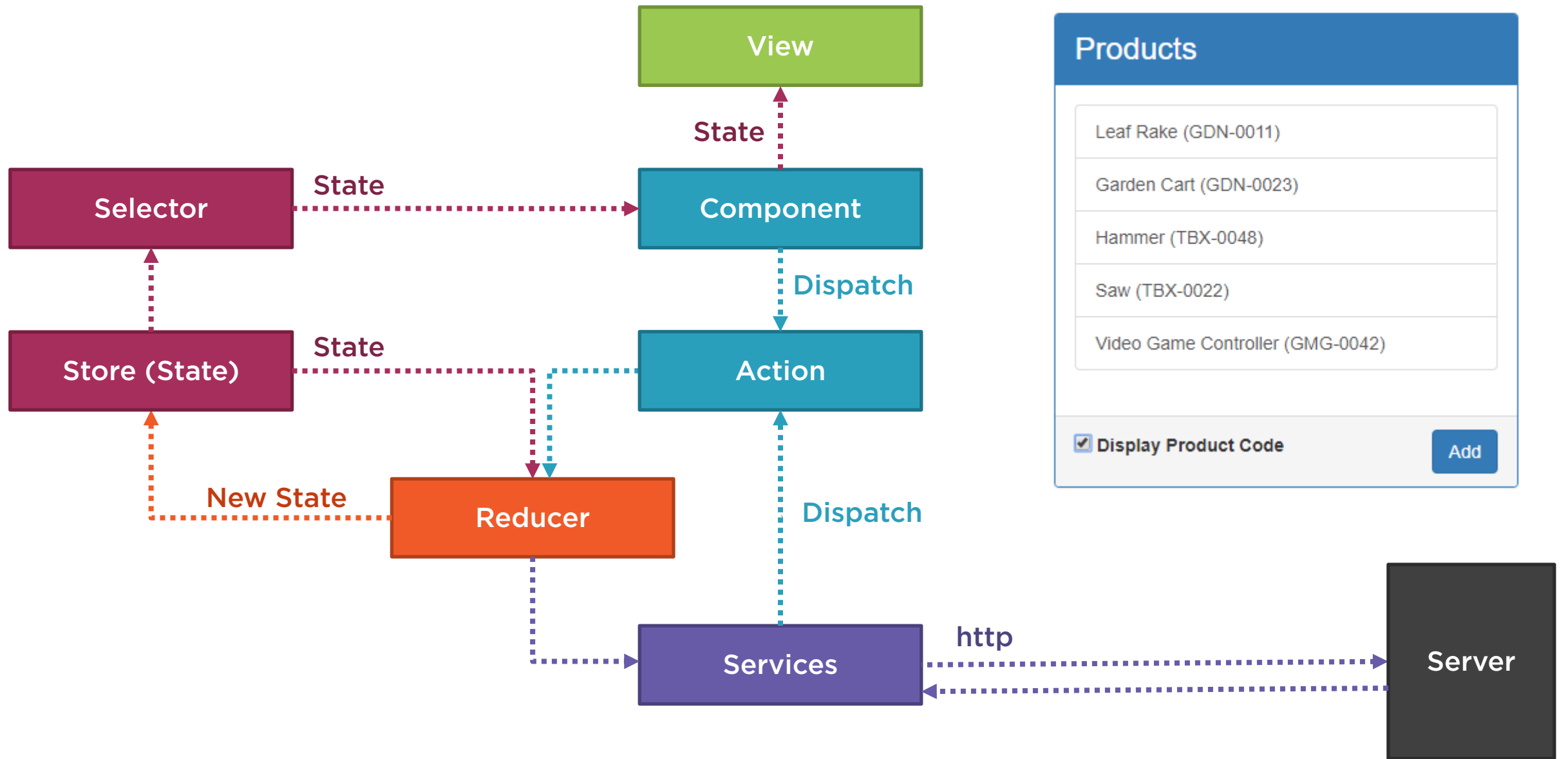


# Defining Actions for Complex Operations

```
export const toggleProductCode = createAction('[Product] Toggle Product Code');
export const setCurrentProduct = createAction(
  '[Product] Set Current Product',
  props<{ currentProductId: number }>()
);
export const clearCurrentProduct = createAction('[Product] Clear Current Product');
export const initCurrentProduct = createAction('[Product] Init Current Product');

export const loadProducts = createAction('[Product] Load');
```





# Defining Actions for Complex Operations

```
export const loadProducts = createAction(
  '[Product] Load'
);

export const loadProductsSuccess = createAction(
  '[Product] Load Success',
  props<{ products: Product[] }>()
);

export const loadProductsFailure = createAction(
  '[Product] Load Fail',
  props<{ error: string }>()
);
```



# Demo



## Defining actions for complex operations



# Checklist: Strongly Typing Actions

Action

**Define the appropriate actions**

**Export a constant that calls createAction**

**Specify a clear, unique action type string**

```
export const toggleProductCode = createAction(  
    '[Product] Toggle Product Code');
```

**Use props to define associated data as needed**

```
export const setCurrentProduct = createAction(  
    '[Product] Set Current Product',  
    props<{ product: Product }>()  
);
```





# Checklist: Complex Operations

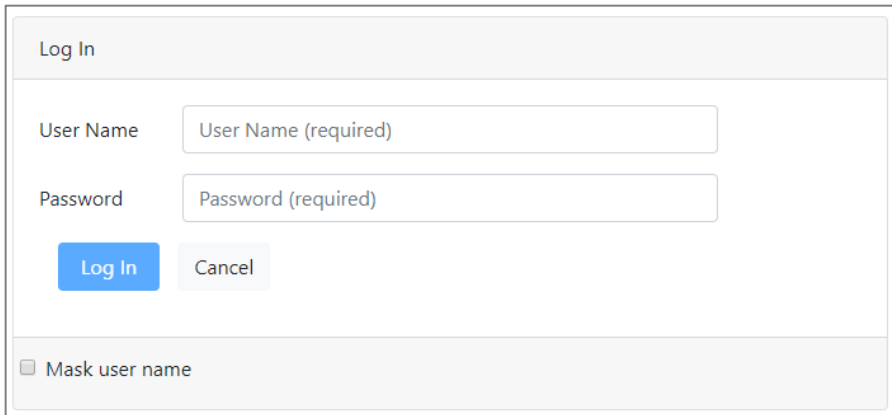
Action

## Define multiple actions:

```
export const loadProducts = createAction('[Product] Load');
export const loadProductsSuccess = createAction(
  '[Product] Load Success',
  props<{ products: Product[] }>()
);
export const loadProductsFailure = createAction(
  '[Product] Load Fail',
  props<{ error: string }>()
);
```



# Homework



The screenshot shows a 'Log In' form with a light gray header. Below the header, there are two input fields: 'User Name' with the placeholder text 'User Name (required)' and 'Password' with the placeholder text 'Password (required)'. Below these fields are two buttons: a blue 'Log In' button and a gray 'Cancel' button. At the bottom of the form, there is a checkbox labeled 'Mask user name'.

Create a `user.actions.ts` file

Build a strongly typed action  
(`maskUserName`, '[User] Mask User Name')

Modify the reducer to handle the strongly  
typed action

Modify the login component to dispatch  
the strongly typed action

<https://github.com/DeborahK/Angular-NgRx-GettingStarted/tree/master/APM-Demo2>

