

# Strongly Typing Actions with Action Creators

---



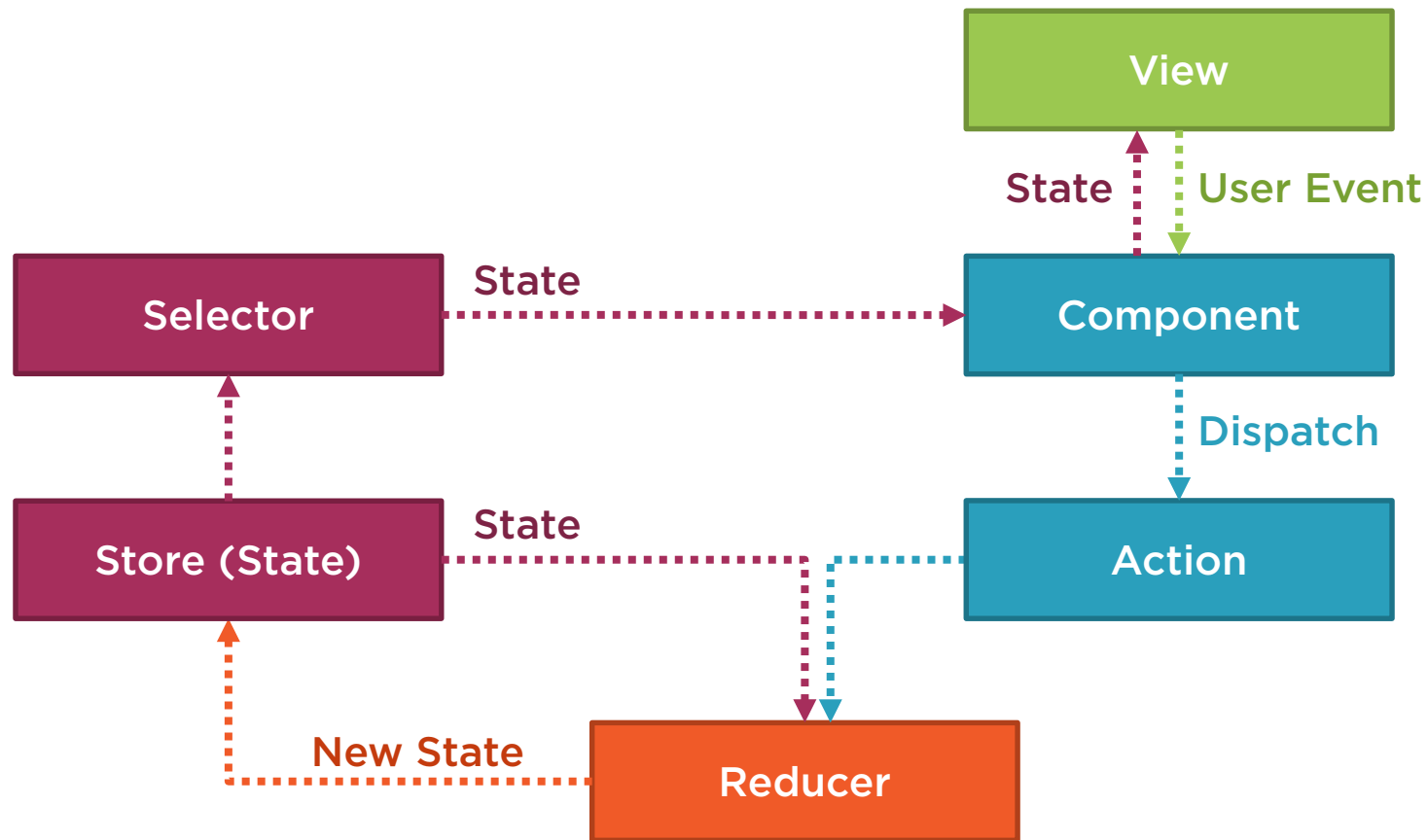
**Deborah Kurata**

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

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







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

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



# Demo



The state of our actions



# Module Overview



**Build action creators to strongly type actions**

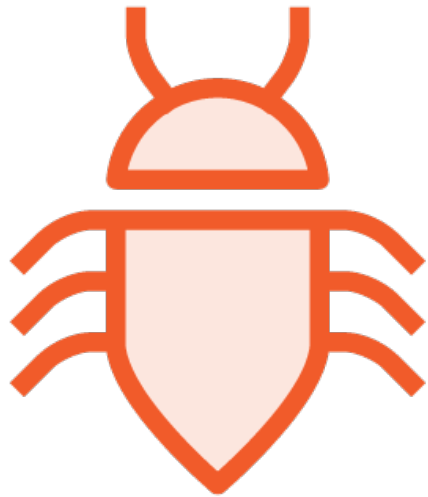
**Use action creators to create and dispatch actions**

**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

# Strongly Typing Actions with Action Creators

1. **Define the action types as named constants**
2. **Build the action creators**
3. **Define a union type for those action creators**



# Defining Action Types

Acme Product Management

HomeProduct 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 Action Types

```
export enum ProductActionTypes {  
  ToggleProductCode = '[Product List Page] Toggle Product Code',  
  SetCurrentProduct = '[Product List Page] Set Current Product',  
  ClearCurrentProduct = '[Product Edit Page] Clear Current Product',  
  InitializeCurrentProduct = '[Product List Page] Initialize Current Product',  
  Load = '[Product List Page] Load',  
  LoadSuccess = '[Product API] Load Success',  
  LoadFail = '[Product API] Load Fail'  
}
```

The screenshot displays the Redux DevTools interface. On the left, a list of actions is shown with their timestamps:

Action	Time
[User] Mask User Name	+00:23.89
[Product] Toggle Product Code	+00:13.76
[Product] Set Current Product	+00:02.84
[Product] Clear Current Product	+00:11.63
[Product] Set Current Product	+00:03.78

The right pane shows the details of the selected action, '[Product] Set Current Product'. It displays the payload and type:

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

At the bottom, there is a command input field containing 'this.' and a 'Dispatch' button. The bottom toolbar includes icons for various DevTools features: Redux DevTools logo, Redux DevTools logo, Redux DevTools logo, Pause recording, Persist, Dispatcher, Slider, Import, Export, Remote, and Settings.

# Building an Action Creator

```
export enum ProductActionTypes {  
  ToggleProductCode = '[Product] Toggle Product Code',  
  SetCurrentProduct = '[Product] Set Current Product',  
  ClearCurrentProduct = '[Product] Clear Current Product',  
  InitializeCurrentProduct = '[Product] Initialize Current Product'  
}
```

```
export class ToggleProductCode implements Action {  
  readonly type = ProductActionTypes.ToggleProductCode;  
  
  constructor(public payload: boolean) {}  
}
```

```
export class ToggleProductCode implements Action {  
  readonly type = ProductActionTypes.ToggleProductCode;  
  public payload: Boolean;  
  constructor(payload: boolean) {  
    this.payload = payload;  
  }  
}
```



# Defining a Union Type for the Action Creators

```
export type ProductActions = ToggleProductCode  
  | SetCurrentProduct  
  | ClearCurrentProduct  
  | InitializeCurrentProduct;
```

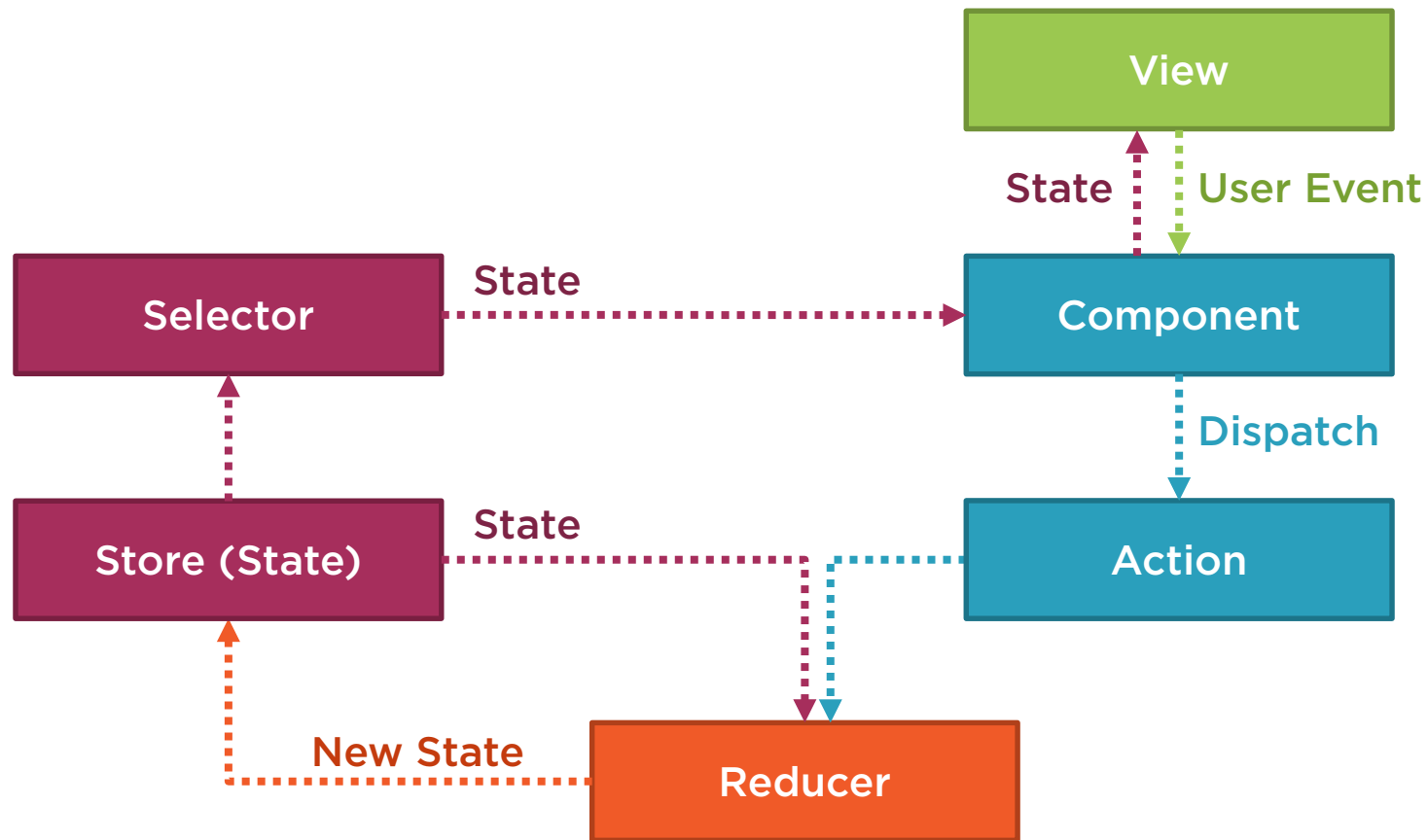


# Demo



## Building action creators





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

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



# Using Action Creators in the Reducer

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

```
export function reducer(state = initialState, action: ProductActions): ProductState {  
  switch (action.type) {  
    case ProductActionTypes.ToggleProductCode:  
      return { ...state, showProductCode: action.payload };  
    ...  
  }  
}
```



# Using Action Creators to Dispatch an Action

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

```
import * as productActions from './state/product.actions';  
...  
this.store.dispatch(new productActions.ToggleProductCode(true));
```

```
export class ToggleProductCode implements Action {  
  readonly type = ProductActionTypes.ToggleProductCode;  
  
  constructor(public payload: boolean) {}  
}
```



# 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

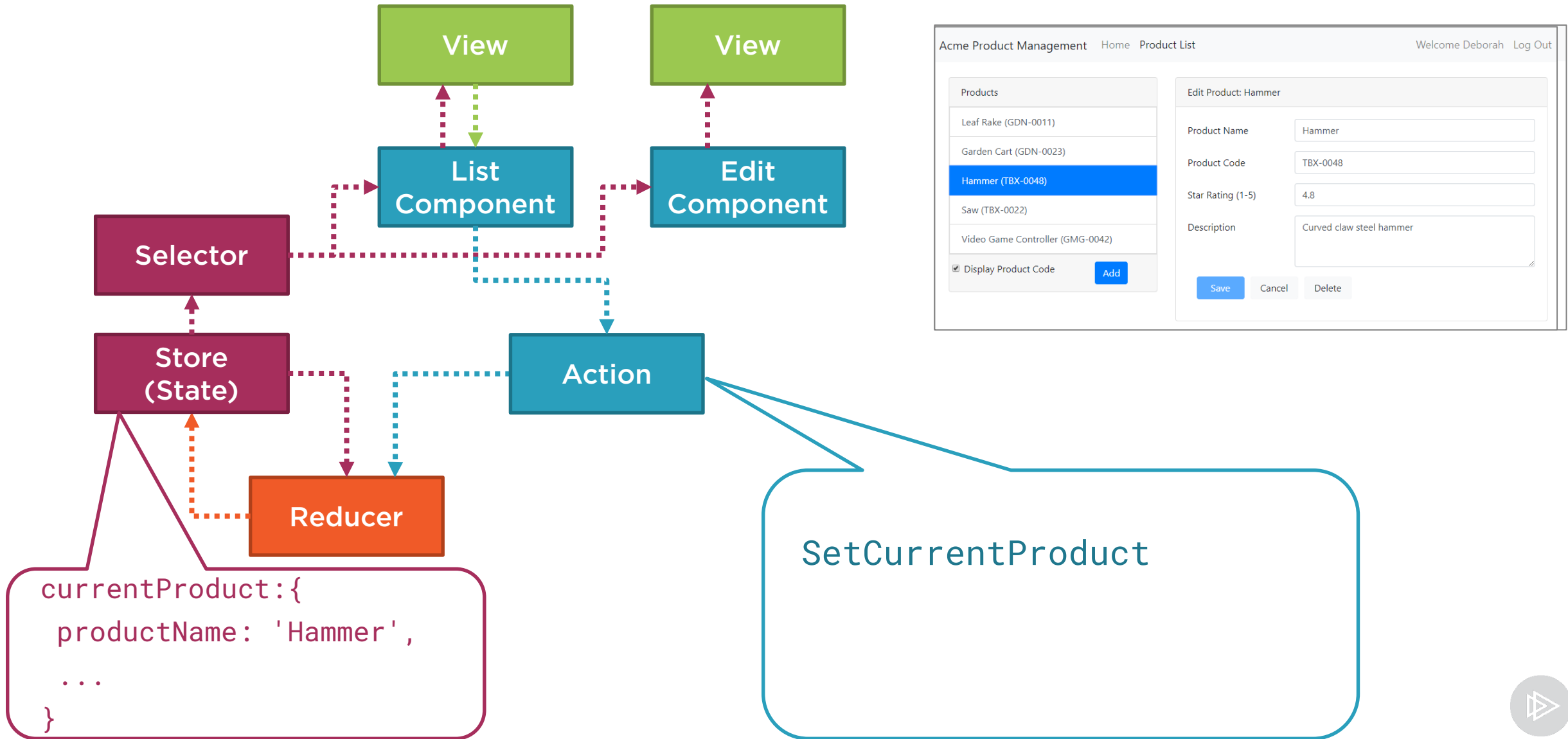
Curved claw steel hammer

Save

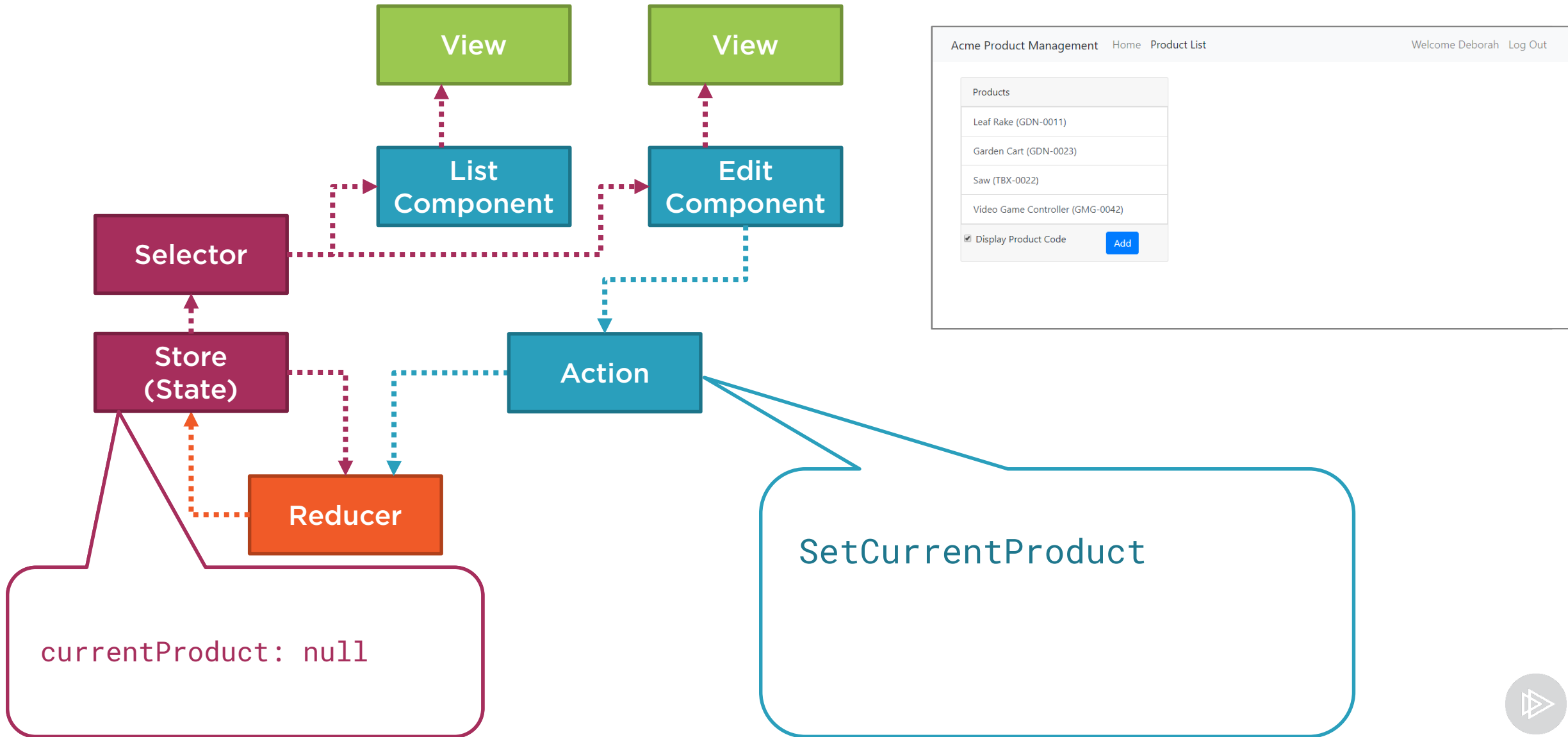
Cancel

Delete

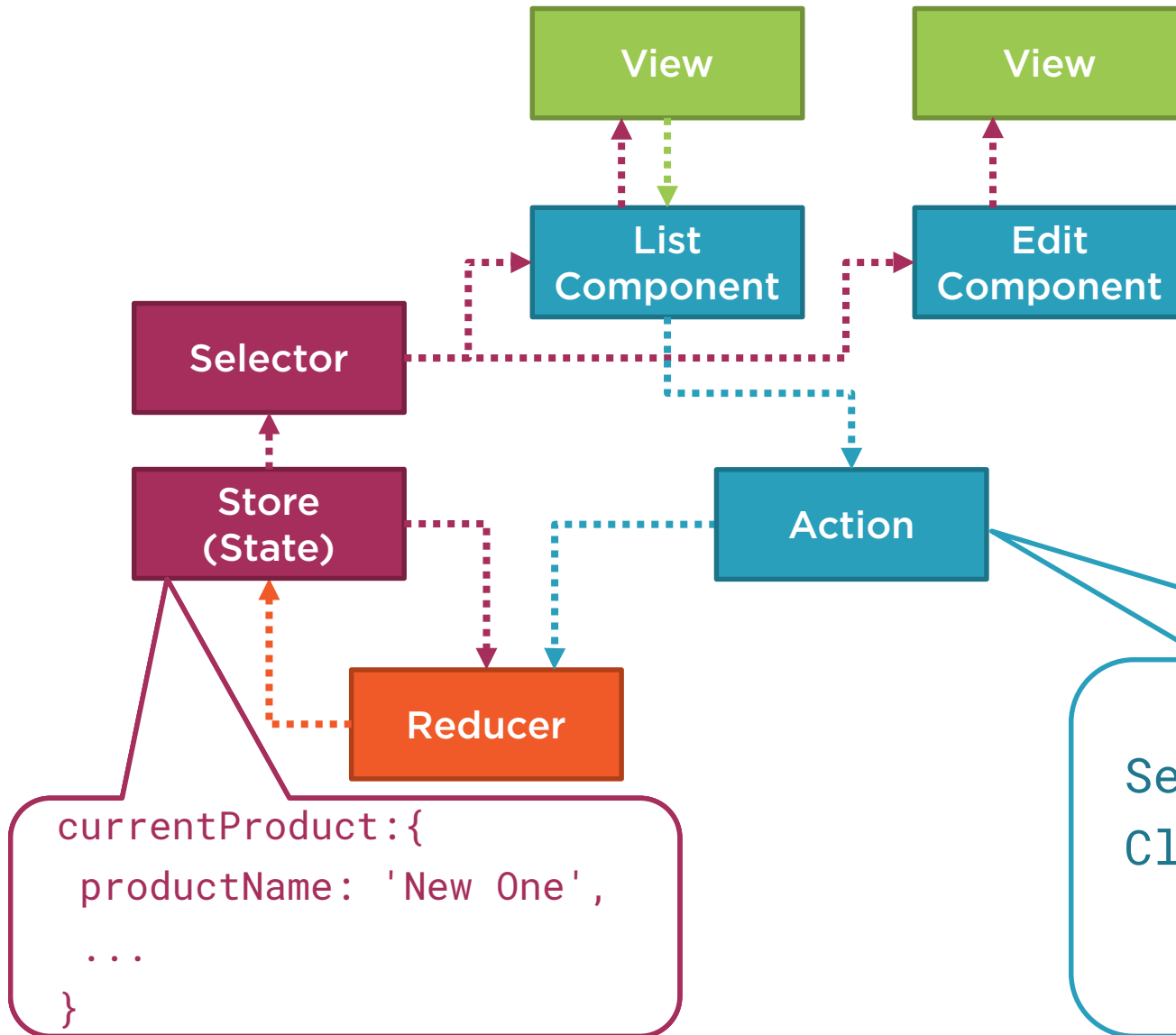
# Component Communication



# Component Communication



# Component Communication



Acme Product Management Home Product List Welcome Deborah Log Out

Products	Add Product
Leaf Rake (GDN-0011)	Product Name <input type="text" value="Name (required)"/>
Garden Cart (GDN-0023)	Product Code <input type="text" value="New"/>
Saw (TBX-0022)	Star Rating (1-5) <input type="text" value="0"/>
Video Game Controller (GMG-0042)	Description <input type="text" value="Description"/>
<input checked="" type="checkbox"/> Display Product Code <input type="button" value="Add"/>	<input type="button" value="Save"/> <input type="button" value="Cancel"/> <input type="button" value="Delete"/>

setCurrentProduct  
clearCurrentProduct



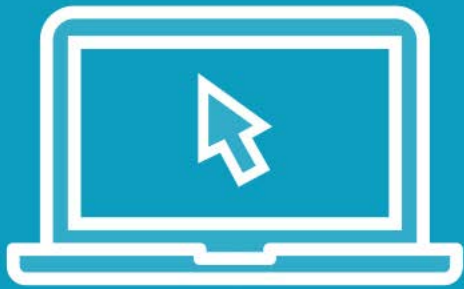
# Demo



Using actions and selectors for  
component communication

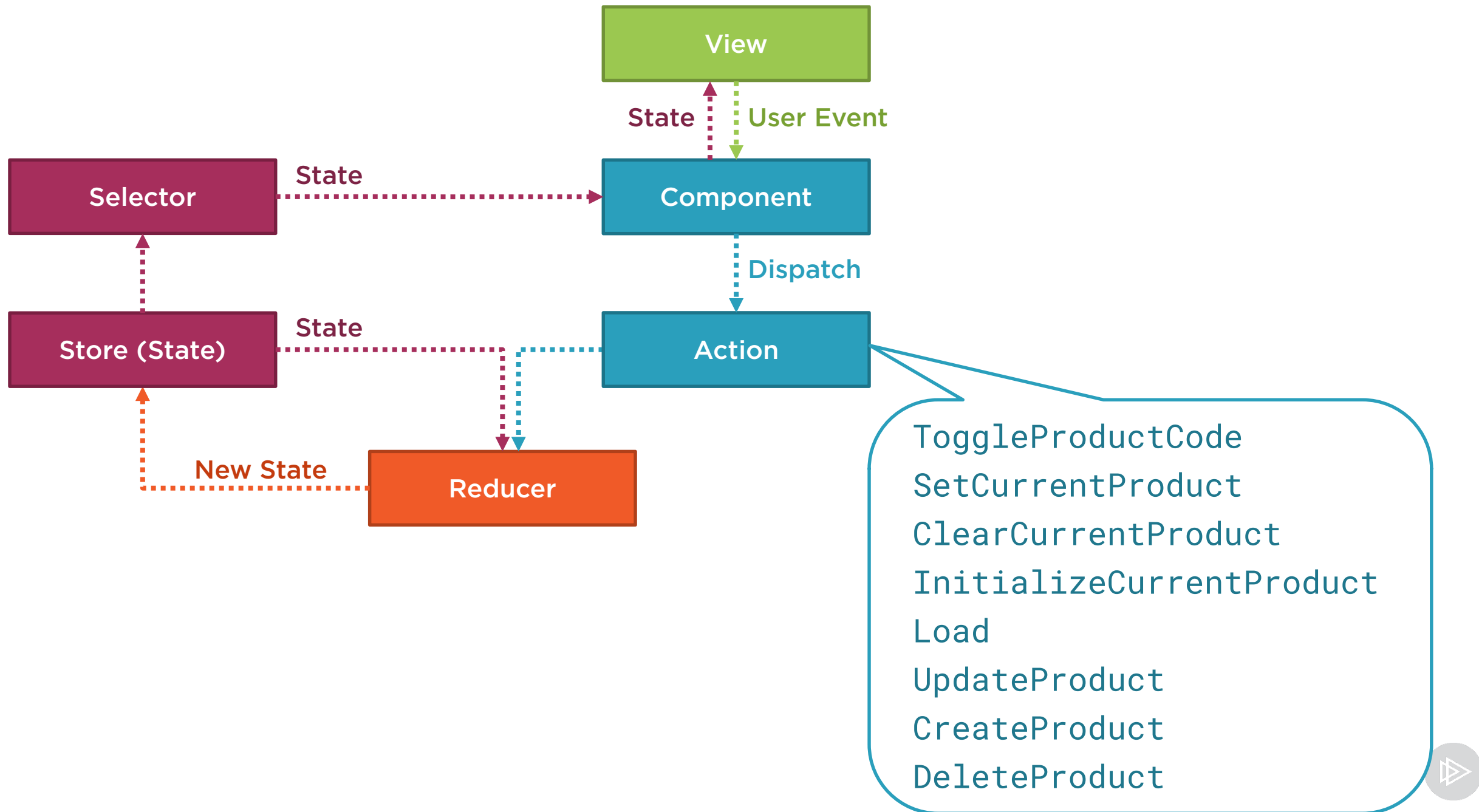


# Demo



Using actions and selectors for  
component communication:  
Edit component



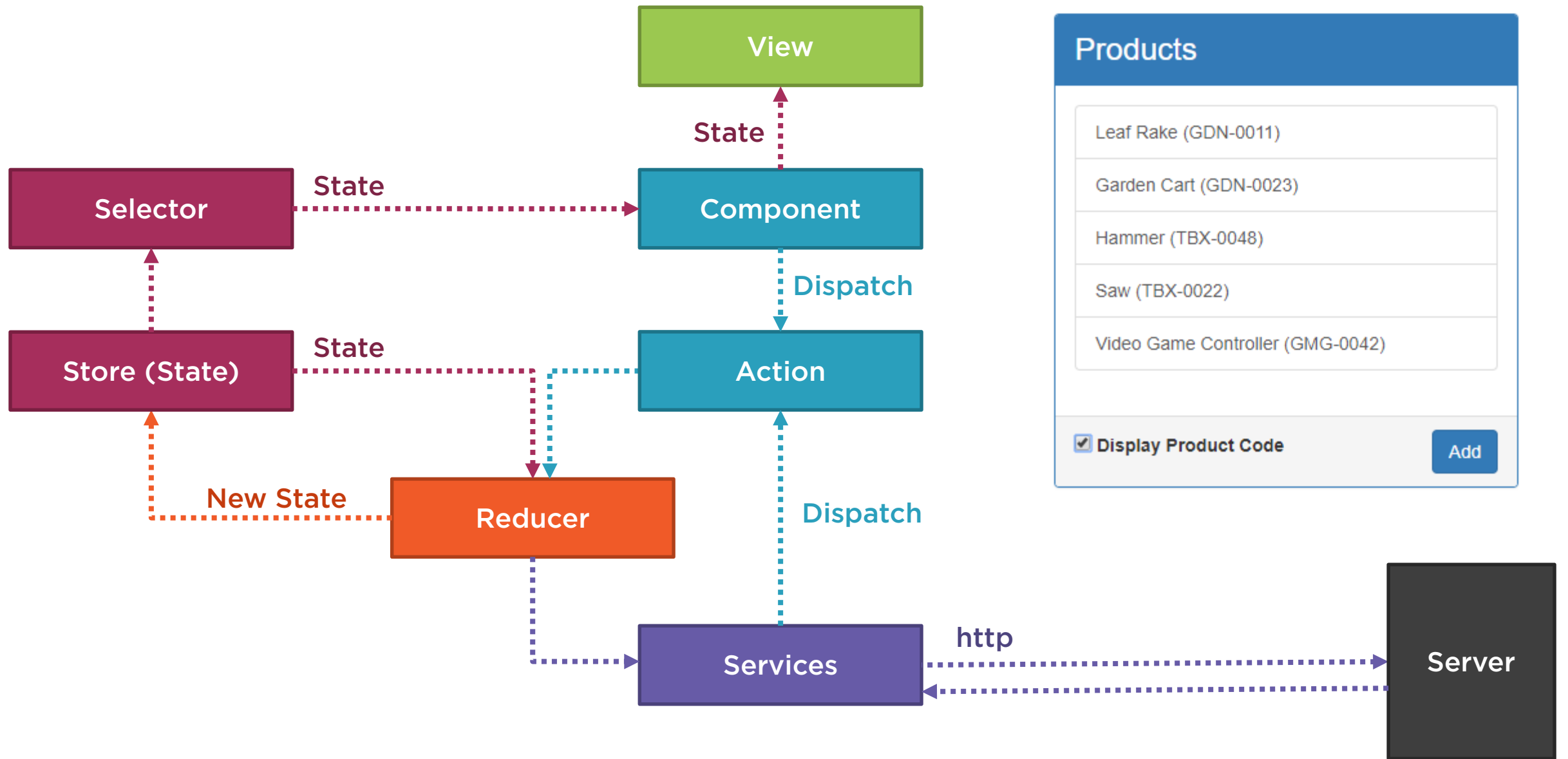


# Defining Actions for Complex Operations

```
export enum ProductActionTypes {  
  ToggleProductCode = '[Product] Toggle Product Code',  
  ClearCurrentProduct = '[Product] Clear Current Product',  
  SetCurrentProduct = '[Product] Set Current Product',  
  InitializeCurrentProduct = '[Product] Initialize Current Product',  
  Load = '[Product] Load'  
}
```







# Defining Actions for Complex Operations

```
export enum ProductActionTypes {  
  ToggleProductCode = '[Product] Toggle Product Code',  
  ClearCurrentProduct = '[Product] Clear Current Product',  
  SetCurrentProduct = '[Product] Set Current Product',  
  InitializeCurrentProduct = '[Product] Initialize Current Product',  
  Load = '[Product] Load',  
  LoadSuccess = '[Product] Load Success',  
  LoadFail = '[Product] Load Fail'  
}
```



# Demo



## Defining actions for complex operations



# Checklist: Strongly Typing Actions - 1

Action

**Define the action types**

**Use an enum to specify the set of named constants:**

```
export enum ProductActionTypes {  
  ToggleProductCode = '[Product] Toggle Product Code',  
  SetCurrentProduct = '[Product] Set Current Product',  
  ClearCurrentProduct = '[Product] Clear Current Product'  
}
```

**Specify clear action type strings**



# Checklist: Strongly Typing Actions - 2

Action

## Build an action creator

### Define a class with type and payload properties:

```
export class ToggleProductCode implements Action {  
  readonly type = ProductActionTypes.ToggleProductCode;  
  
  constructor(public payload: boolean) {}  
}
```

### Use the action creator when dispatching the action:

```
this.store.dispatch(  
  new productActions.ToggleProductCode(true)  
);
```



# Checklist: Strongly Typing Actions - 3

Action

**Union the action creators**

**Define a union type of all action creators:**

```
export type ProductActions = ToggleProductCode  
  | SetCurrentProduct  
  | ClearCurrentProduct;
```

**Use this union type in the reducer:**

```
export function reducer(state = initialState,  
  action: ProductActions)
```



# Checklist: Complex Operations

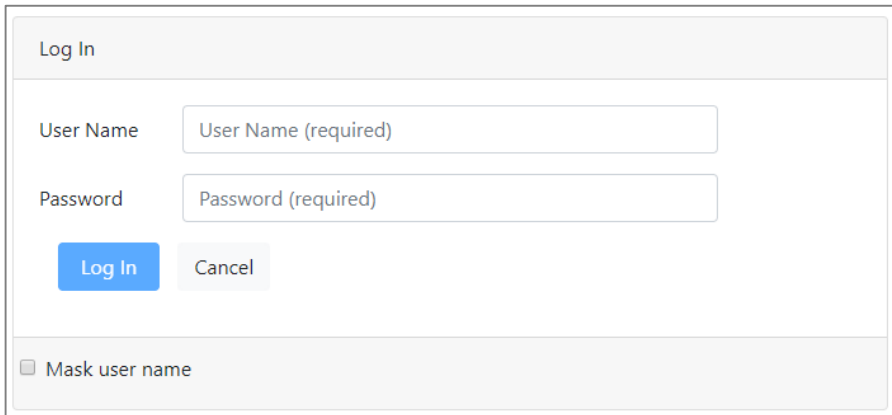
Action

**Define multiple actions:**

```
export enum ProductActionTypes {  
  ...  
  Load = '[Product] Load',  
  LoadSuccess = '[Product] Load Success',  
  LoadFail = '[Product] Load Fail'  
}
```



# Homework



Log In

User Name

Password

☐ Mask user name

Create an `user.actions.ts` file

Add an enum for the action type  
(`MaskUserName`)

Build the associated action creator

Create a union type for the action creators

Modify the reducer to use the union type

Modify the login component to use the  
action creator

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

