academeez

@ngrx/effects

Dispatch actions from other actions

@ngrx/effects

academeez

- In the previous lesson, we had a component send message to another component using the store
- Which means the component that sent a message triggered an action on the store
- Not all actions originate from components, for example:
  - We might want to perform some action when the app starts
  - We might want to trigger some actions as a result of another action
  - Might be useful to separate server logic from component, especially if the data from the server needs to be consumed from multiple components
  - We might want to trigger action based on router events

What I can do with @ngrx/effects

academeez

- With @ngrx/effects you can:
  - Dispatch actions based on other actions
  - Deal with server grabbing logic outside the components
  - Makes the components leaner and more testable
  - Connect to any stream and issue an action based on that stream

Todo list example

academeez

- Let’s try to understand the purpose of @ngrx/effects with the following example
- In this example we want to add @ngrx/store and have it manage the following state
- The todo array will be loaded from a REST server

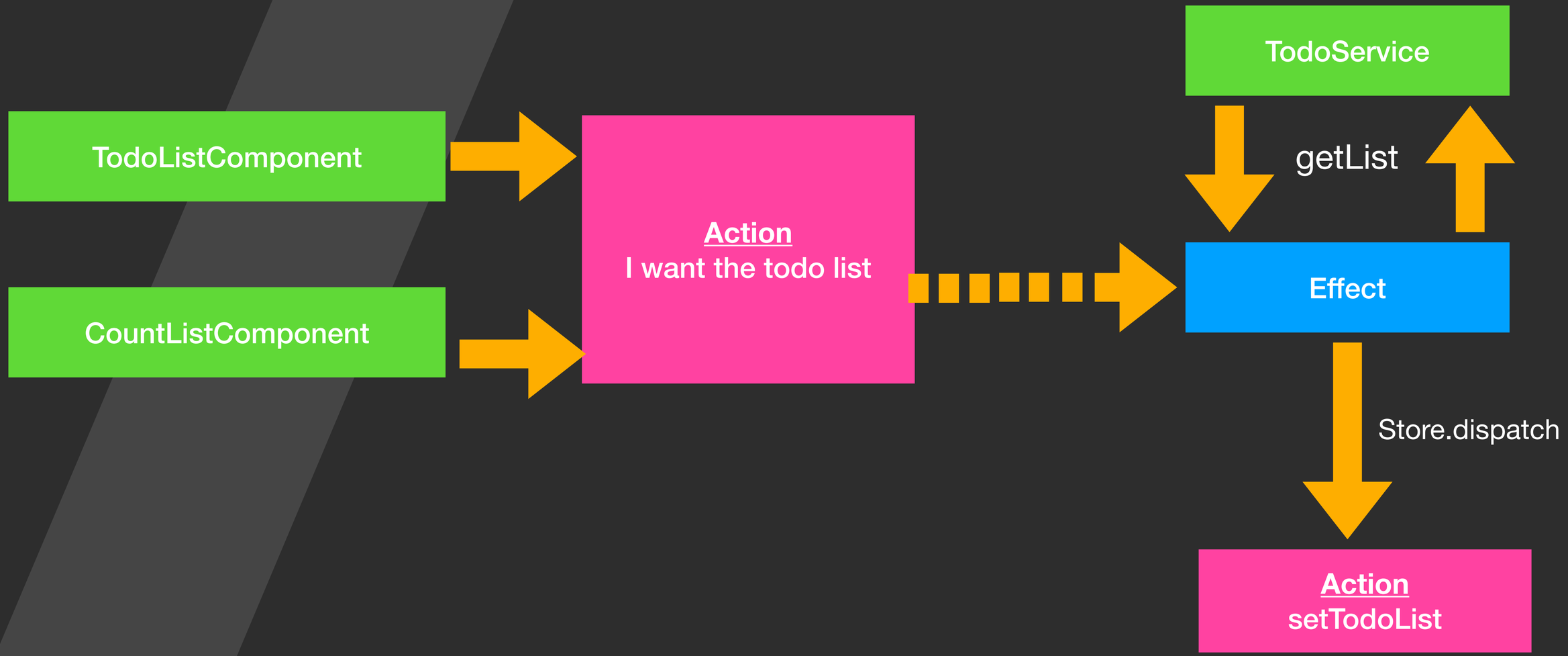
```
const state = {
  todo: [
    {id: 1, title: 'Buy soya milk', description: 'shopping'},
    {id: 2, title: 'Play with dogs', description: 'play with Piglet and Sweetness'},
    ...
  ]
}
```

Todo list components

- Our app will contain 2 components:
  - **TodoListComponent** - will display the list of todo tasks
  - **CountListComponent** - will display the number of list items
- Both components will need the list from the server
- Which one will be in charge of sending the request to the server?
- None of them, they will just send an action representing their intent to consume the list
- An effect will listen for an action of someone that wants to consume the list and send the request himself

Components Effects

academeez



Actions

academeez

- We will create 2 actions:
  - **getTodo** - this will be called from the components expressing interest in reading the list of todo
  - **setTodo -** when the server returns the list this will be called

TodoEffect

academeez

- Our Effect will listen for the action **getTodo** and will send a request to grab the list when the action arrives
- An Effect is an Injectable service.
- Usually an Effect will request in the constructor the Observable of the actions to listen to a specific action
- We can use the operator **ofType** to listen to a specific action

todoReducer

- Our reducer will update the list of tasks in the state when it gets the **setTodo** action

Components

- The components will dispatch the action **getTodo** to express their interest in the list
- The components will register for the array of todo from the state

Summary

academeez

- @ngrx/effects allows us to distinguish between actions that should originate from components, and action the should originate from other action or some non related component event that happens in our app
- With @ngrx/effects we create Effects which can listen to other observables and emit actions
- They usually also listen to the stream of action and emit action based on other action

academeez

Thank You

Next Lesson: @ngrx/entity