§ NGRX Training Part 3 - Selectors

Tuesday, August 18, 2020 9:32 AM

Better Selectors

The ngrx/store selector capability is designed to support complex state definitions that include slicing and features.

Selectors are pure functions that take slices of state as arguments and return some state data that we can pass to our components.

To better understand what selectors are and what they do, it helps see NgRx state as a data structure - a tree that can be serialized to JSON. Data is added to the state tree by composing state in reducers - that's the easy part.

Now to get data out of the state tree, we have to traverse it to find our property of interest - and return it. That can become more complex, and is where selectors help us out.

You may have already seen the `store.select` method being used to get data from the store by passing it a string value:

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

The string represents the name of a slice of state in the store and we can expect this function to return data corresponding to our pizzas property - perhaps an array of pizzas. However, `store.select` can also take a function instead, which takes a slice of state and returns a property from the state (which you've likely already seen as well):

```
this.store.select(state => state.iceCream);
```

Both of these approaches represent the concept of a **selector** - we are "selecting" state.

ice-cream-selectors.ts

```
import { createFeatureSelector, createSelector } from '@ngrx/store';
import { AppState } from '../app.state';

const getIceCreamFeatureState = createFeatureSelector<AppState>('iceCream');

export const getIceCream = createSelector(
    getIceCreamFeatureState,
    state => state.iceCream
);
```

ice-cream-display.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { Store, select } from '@ngrx/store';
import { AppState } from '../app.state';
import { IceCream } from '../models/ice-cream';
import * as FlavorActions from '../actions/ice-cream-actions';
import { getIceCream } from '../selectors/ice-cream-selectors';

@Component({
  selector: 'app-ice-cream-display',
  templateUrl: './ice-cream-display.component.html',
```

```
  styleUrls: ['./ice-cream-display.component.scss']
})
export class IceCreamDisplayComponent implements OnInit {
  private iceCreams$: Observable<IceCream[]>;

  constructor(private store: Store<AppState>) {
      this.iceCreams$ = this.store.pipe(select(getIceCream));
   }

  ngOnInit() { }

  onNewFlavor(newFlavor: IceCream) {
    this.store.dispatch(FlavorActions.AddFlavor({iceCream: newFlavor}));
  }

  onDeleteFlavor(index: number) {
    this.store.dispatch(FlavorActions.RemoveFlavor({flavorId: index}));
  }
}
```