§ 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.

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}));
  }
}
```