Selecting State with Selectors

Duncan Hunter
Lead Frontend Developer

@dunchunter | duncanhunter.com.au

Module Introduction

Why use selectors

Creating selectors

Updating the demo to use selectors

products-page.component.ts

src > app > products > products-page > products-page.component.ts > ProductsPageComponent

```
@Component({
  selector: 'app-products-page',
  templateUrl: './products-page.component.html',
  styleUrls: ['./products-page.component.css'],
})
export class ProductsPageComponent {
  products$ = this.store.select((state: any) => state.products.products);
  total = 0;
  loading$ = this.store.select((state: any) => state.products.loading);
  showProductCode$ = this.store.select(
    (state: any) => state.products.showProductCode
  );
  errorMessage = '';

  constructor(private productsService: ProductsService, private store: Store) {
    this.store.subscribe((store) => console.log({ store }));
  }

  ngOnInit() {
    this.getProducts();
  }

  getProducts() {
    this.store.dispatch(ProductsPageActions.loadProducts());
    this.productsService.getAll().subscribe({
      next: (products) => {
        this.store.dispatch(
          ProductsAPIActions.productsLoadedSuccess({ products })
        );
      },
```

master

products-page.component.ts

src > app > products > products-page > products-page.component.ts > ProductsPageComponent

```
@Component({
  selector: 'app-products-page',
  templateUrl: './products-page.component.html',
  styleUrls: ['./products-page.component.css'],
})
export class ProductsPageComponent {
  products$ = this.store.select((state: any) => state.products.products);
  total = 0;
  loading$ = this.store.select((state: any) => state.products.loading);
  showProductCode$ = this.store.select(
    (state: any) => state.products.showProductCode
  );
  errorMessage = '';

  constructor(private productsService: ProductsService, private store: Store) {
    this.store.subscribe((store) => console.log({ store }));
  }

  ngOnInit() {
    this.getProducts();
  }

  getProducts() {
    this.store.dispatch(ProductsPageActions.loadProducts());
    this.productsService.getAll().subscribe({
      next: (products) => {
        this.store.dispatch(
          ProductsAPIActions.productsLoadedSuccess({ products })
        );
      },
```

master

products-page.component.ts

src > app > products > products-page > products-page.component.ts > ProductsPageComponent

```
@Component({
  selector: 'app-products-page',
  templateUrl: './products-page.component.html',
  styleUrls: ['./products-page.component.css'],
})
export class ProductsPageComponent {
  products$ = this.store.select((state: any) => state.products.products);
  total = 0;
  loading$ = this.store.select((state: any) => state.products.loading);
  showProductCode$ = this.store.select(
    (state: any) => state.products.showProductCode
  );
  errorMessage = '';

  constructor(private productsService: ProductsService, private store: Store) {
    this.store.subscribe((store) => console.log({ store }));
  }

  ngOnInit() {
    this.getProducts();
  }

  getProducts() {
    this.store.dispatch(ProductsPageActions.loadProducts());
    this.productsService.getAll().subscribe({
      next: (products) => {
        this.store.dispatch(
          ProductsAPIActions.productsLoadedSuccess({ products })
        );
      },
```

master

products-page.component.ts

src > app > products > products-page > products-page.component.ts > ProductsPageComponent

```
@Component({
  selector: 'app-products-page',
  templateUrl: './products-page.component.html',
  styleUrls: ['./products-page.component.css'],
})
export class ProductsPageComponent {
  products$ = this.store.select((state: any) => state.products.products);
  total = 0;
  loading$ = this.store.select((state: any) => state.products.loading);
  showProductCode$ = this.store.select(
    (state: any) => state.products.showProductCode
  );
  errorMessage = '';

  constructor(private productsService: ProductsService, private store: Store) {
    this.store.subscribe((store) => console.log({ store }));
  }

  ngOnInit() {
    this.getProducts();
  }

  getProducts() {
    this.store.dispatch(ProductsPageActions.loadProducts());
    this.productsService.getAll().subscribe({
      next: (products) => {
        this.store.dispatch(
          ProductsAPIActions.productsLoadedSuccess({ products })
        );
      },
```

master

Selectors are pure functions used for obtaining slices of store state

Benefits of Selectors

- Portability
- Memoization
- Composition
- Testability
- Type Safety

Two Create Selector Helper Functions

```
import { createSelector} from ‘@ngrx/store’;

createSelector();
```

Two Create Selector Helper Functions

```
import { createSelector, createFeatureSelector } from '@ngrx/store';

createSelector();

createFeatureSelector();
```

Store State

```
{
    products: {
        showProductCode: boolean;
        loading: boolean;
        products: Product[];
    }
}
```

Store State

```
{
    products: {
        showProductCode: boolean;
        loading: boolean;
        products: Product[];
    },
    auth: {
        userId: number;
        favouriteProducts: number[];
    }
}
```

Selecting Typed State

```
export interface ProductsState {
    showProductCode: boolean;
    loading: boolean;
    products: Product[];
}

export interface AuthState {
    userId: number;
    favouriteProducts: number;
}
```

Selecting Typed State

```
export interface ProductsState {
    showProductCode: boolean;
    loading: boolean;
    products: Product[];
}

export interface AuthState {
    userId: number;
    favouriteProducts: number;
}

export interface AppState {
    products: ProductState;
    auth: AuthState;
}
```

Selecting Typed State

```
export interface ProductsState {
    showProductCode: boolean;
    loading: boolean;
    products: Product[];
}

export interface AuthState {
    userId: number;
    favouriteProducts: number;
}

export interface AppState {
    products: ProductState;
    auth: AuthState;
}

export const selectProductsState = (state: AppState) => state.products;
```

Selecting Feature State

```
import { createFeatureSelector} from ‘@ngrx/store’;

export const selectProductsState = createFeatureSelector<ProductsState>(‘products’);
```

Composing Selectors

```
import { createFeatureSelector} from ‘@ngrx/store’;

export const selectProductsState = createFeatureSelector<ProductsState>(‘products’);

export const selectProducts = createSelector(
    selectProductsState,
    (productsState) => productsState.products;
);
```

Aggregating Data

```
import { createFeatureSelector, createSelector } from '@ngrx/store';
import { sumProducts } from ./utils';

export const selectProductsState = createFeatureSelector<ProductsState>('products');

export const selectProducts = createSelector(
    selectProductsState,
    (state) => state.product;
);

export const selectProductsTotal = createSelector(
    selectProducts,
    (products) => sumProducts(products);
);
```

Aggregating Data

```
import { createFeatureSelector, createSelector } from '@ngrx/store';
import { sumProducts } from ./utils';

export const selectProductsState = createFeatureSelector<ProductsState>('products');

export const selectProducts = createSelector(
    selectProductsState,
    (state) => state.product;
);

export const selectProductsTotal = createSelector(
    selectProducts,
    sumProducts
);
```

Composing 2 Feature Slices of State

```
export const selectProducts = createSelector(
     selectProductsState,
     (state) => state.products;
);

export const selectFavouriteProducts = createSelector(
      selectAuthState,
     (state) => state.favouriteProducts;
);
```

Composing 2 Feature Slices of State

```
export const selectProducts = createSelector(
    selectProductsState,
    (state) => state.products;
);

export const selectFavouriteProducts = createSelector(
     selectAuthState,
    (state) => state.favouriteProducts;
);

export const selectFavouriteProducts = createSelector(
    selectProducts,
    selectFavouriteProducts,
);
```

Composing 2 Feature Slices of State

```
export const selectProducts = createSelector(
    selectProductsState,
    (state) => state.products;
);

export const selectFavouriteProducts = createSelector(
     selectAuthState,
    (state) => state.favouriteProducts;
);

export const selectFavouriteProducts = createSelector(
    selectProducts,
    selectFavouriteProducts,
    (products, favouriteProducts) => products.filter(
        (product) => favouriteProducts.includes(product.id)
    );
);
```

Using Selectors in Components

```
import { selectProducts } from '../state/products.actions';

products$ = this.store.select(selectProducts);
```

Demo

Adding selectors

- Create products.selectors.ts file
- Add selectors
- Swap out inline selectors