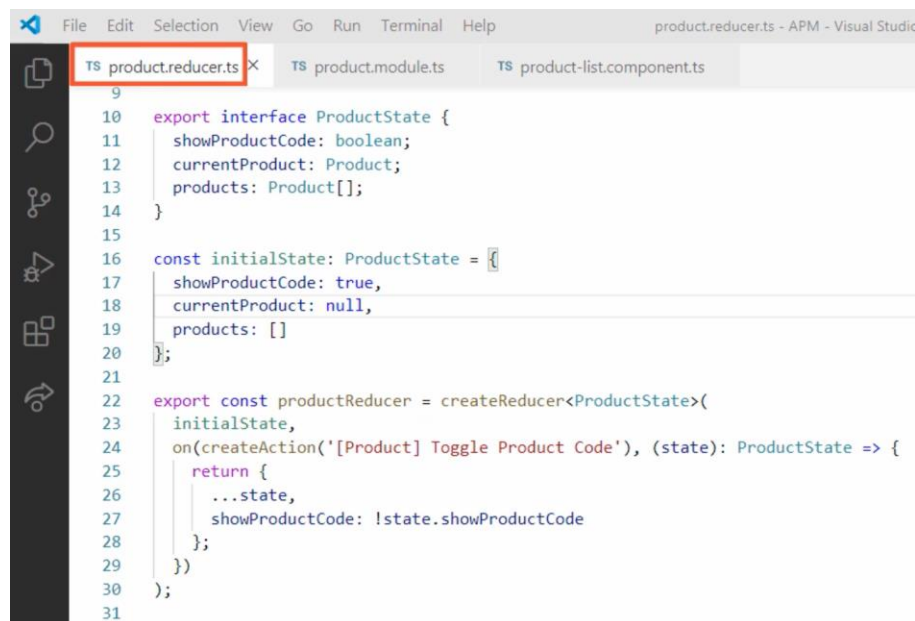


Tipado fuera del estado – Construyendo Selectores (Demo)

En esta demo, construiremos los *Selectors* para el slice de estado de nuestro producto.

Una vez más, vamos a empezar con la pregunta ¿dónde? ¿dónde definimos nuestros *Selectors*? En nuestro ejemplo, es el *Reducer* el que define las interfaces para nuestro estado, por lo que tiene sentido poner los *Selectors* en el *Reducer* también.

Así que tenemos el archivo *product.reducer.ts*:



```
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 const initialState: ProductState = {
17   showProductCode: true,
18   currentProduct: null,
19   products: []
20 };
21
22 export const productReducer = createReducer<ProductState>(
23   initialState,
24   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
25     return {
26       ...state,
27       showProductCode: !state.showProductCode
28     };
29   })
30 );
31
```

Primero creamos el *Selector* feature:



```
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 const initialState: ProductState = {
17   showProductCode: true,
18   currentProduct: null,
19   products: []
20 };
21
22 const getProductFeatureState = createFeatureSelector<ProductState>('products');
23
24 export const productReducer = createReducer<ProductState>(
25   initialState,
26   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
27     return {
28       ...state,
29       showProductCode: !state.showProductCode
30     };
31   })
32 );
```

Lo definimos con una constante llamada *getProductFeatureState*, la cual es construida usando la función *createFeatureSelector*. No exportamos esta constante, por lo que solo se podrá utilizar dentro de este archivo.

Luego añadimos un *Selector* para nuestra propiedad *showProductCode*:

```
TS product.reducer.ts X TS product.module.ts TS product-list.component.ts
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 const initialState: ProductState = {
17   showProductCode: true,
18   currentProduct: null,
19   products: []
20 };
21
22 const getProductFeatureState = createFeatureSelector<ProductState>('products');
23
24 export const getShowProductCode = createSelector(
25   getProductFeatureState,
26   state => state.showProductCode
27 );
28
29 export const productReducer = createReducer<ProductState>(
30   initialState,
31   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
```

Queremos que nuestros *Selectors* generales estén disponibles en cualquier parte de la aplicación, así que los exportamos como constantes. Llamamos a este *Selector* *getShowProductCode*, el cual construimos usando la función *createSelector*. El primer argumento que le pasamos a la función *createSelector* es el *Selector* necesario para recuperar el bit de estado deseado. En este ejemplo, se trata de la función *Selector* *getProductFeatureCode*:

```
TS product.reducer.ts X TS product.module.ts TS product-list.component.ts
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 const initialState: ProductState = {
17   showProductCode: true,
18   currentProduct: null,
19   products: []
20 };
21
22 const getProductFeatureState = createFeatureSelector<ProductState>('products');
23
24 export const getShowProductCode = createSelector(
25   getProductFeatureState,
26   state => state.showProductCode
27 );
28
29 export const productReducer = createReducer<ProductState>(
30   initialState,
```

El último argumento es la función proyectora:

```
24 export const getShowProductCode = createSelector(  
25   getProductFeatureState,  
26   state => state.showProductCode  
27 );
```

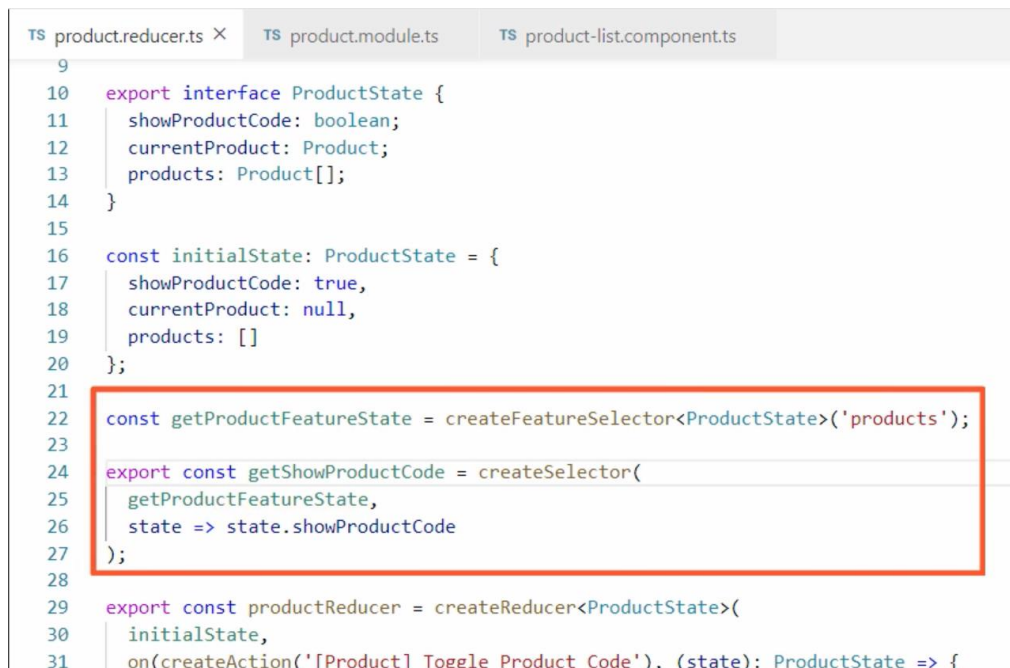
La función proyectora recibe el resultado de la función Selector (la que pasamos como primer argumento), que en este ejemplo es el slice de estado *“product”*:

```
24 export const getShowProductCode = createSelector(  
25   getProductFeatureState,  
26   state => state.showProductCode  
27 );
```

A continuación, manipulamos ese slice según sea necesario para devolver el valor de la propiedad deseada. En este caso, devolvemos el valor de la propiedad *showProductCode*:

```
24 export const getShowProductCode = createSelector(  
25   getProductFeatureState,  
26   state => state.showProductCode  
27 );
```

Note que el orden de estas constantes es importante:



```
TS product.reducer.ts X TS product.module.ts TS product-list.component.ts  
9  
10 export interface ProductState {  
11   showProductCode: boolean;  
12   currentProduct: Product;  
13   products: Product[];  
14 }  
15  
16 const initialState: ProductState = {  
17   showProductCode: true,  
18   currentProduct: null,  
19   products: []  
20 };  
21  
22 const getProductFeatureState = createFeatureSelector<ProductState>('products');  
23  
24 export const getShowProductCode = createSelector(  
25   getProductFeatureState,  
26   state => state.showProductCode  
27 );  
28  
29 export const productReducer = createReducer<ProductState>(  
30   initialState,  
31   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
```

Dado que se trata de un simple archivo de código typescript, no una clase, cada constante debe ir después de cualquier constante a la que haga referencia.

Ahora vamos a añadir *Selectors* para el *currentProduct* y el array de productos:

```
TS product.reducer.ts X TS product.module.ts TS product-list.component.ts
15
16 const initialState: ProductState = {
17   showProductCode: true,
18   currentProduct: null,
19   products: []
20 };
21
22 const getProductFeatureState = createFeatureSelector<ProductState>('products');
23
24 export const getShowProductCode = createSelector(
25   getProductFeatureState,
26   state => state.showProductCode
27 );
28
29 export const getCurrentProduct = createSelector(
30   getProductFeatureState,
31   state => state.currentProduct
32 );
33
34 export const getProducts = createSelector(
35   getProductFeatureState,
36   state => state.products
37 );
38
```

Ambos *Selectors* retornan el valor de una propiedad de nuestro Store sin ningún procesamiento adicional.

Ahora veamos cómo utilizar estos Selectors.