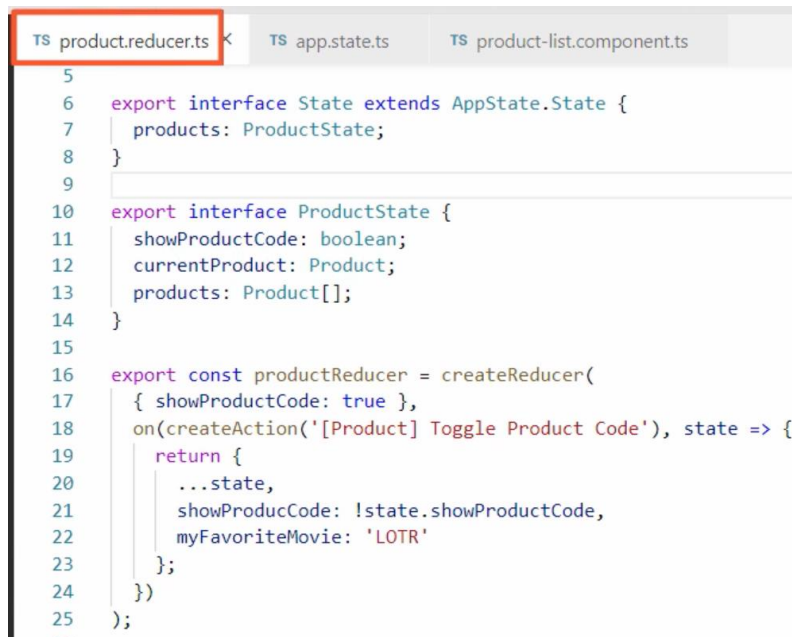


Tipado fuerte del estado – Agregando tipado fuerte al estado

Una vez definidas las interfaces, las utilizamos para tipear fuertemente el estado. Empecemos con nuestro *productReducer*:



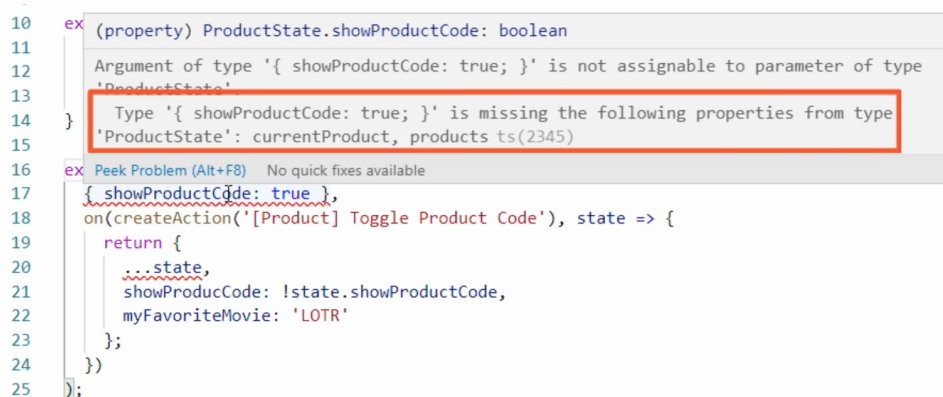
```
5
6 export interface State extends AppState.State {
7   products: ProductState;
8 }
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 export const productReducer = createReducer(
17   { showProductCode: true },
18   on(createAction('[Product] Toggle Product Code'), state => {
19     return {
20       ...state,
21       showProductCode: !state.showProductCode,
22       myFavoriteMovie: 'LOTR'
23     };
24   })
25 );
```

Añadimos el tipo del *createReducer* como su argumento genérico:



```
16 export const productReducer = createReducer<ProductState>({
17   { showProductCode: true },
18   on(createAction('[Product] Toggle Product Code'), state => {
19     return {
20       ...state,
21       showProductCode: !state.showProductCode,
22       myFavoriteMovie: 'LOTR'
23     };
24   })
25 });
```

Vemos un error en nuestro objeto de estado inicial que dice “missing the following properties”:



```
10 ex (property) ProductState.showProductCode: boolean
11
12 Argument of type '{ showProductCode: true; }' is not assignable to parameter of type
13 'ProductState'
14 }
15 Type '{ showProductCode: true; }' is missing the following properties from type
16 'ProductState': currentProduct, products ts(2345)
17
18 ex Peek Problem (Alt+F8) No quick fixes available
19
20 { showProductCode: true },
21 on(createAction('[Product] Toggle Product Code'), state => {
22   return {
23     ...state,
24     showProductCode: !state.showProductCode,
25     myFavoriteMovie: 'LOTR'
26   };
27 })
28 });
```

Recordemos que el primer argumento del `createReducer` es el valor inicial de nuestro *slice* de estado “*product*”. Vamos a agregar un tipado fuerte añadiendo “*as ProductState*”:

```
16 export const productReducer = createReducer<ProductState>(  
17   { showProductCode: true } as ProductState,  
18   on(createAction('[Product] Toggle Product Code'), state => {  
19     return {  
20       ...state,  
21       showProductCode: !state.showProductCode,  
22       myFavoriteMovie: 'LOTR'  
23     };  
24   })  
25 );  
26
```

A continuación, añadimos tipado fuerte para el tipo de retorno de la función *callback* que pasamos a la función “*on*”, agregando “*as ProductState*”:

```
16 export const productReducer = createReducer<ProductState>(  
17   { showProductCode: true } as ProductState,  
18   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {  
19     return {  
20       ...state,  
21       showProductCode: !state.showProductCode,  
22       myFavoriteMovie: 'LOTR'  
23     };  
24   })  
25 );
```

Este tipo *ProductStore* es el tipo de estado que nuestro *Reducer* devolverá al *Store*:

```
16 export const productReducer = createReducer<ProductState>(  
17   { showProductCode: true } as ProductState,  
18   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {  
19     return {  
20       ...state,  
21       showProductCode: !state.showProductCode,  
22       myFavoriteMovie: 'LOTR'  
23     };  
24   })  
25 );
```

Observe que ahora nuestro código tiene un error de sintaxis. El tipado fuerte encontró inmediatamente un error en el nombre del flag, así que lo corregimos:

```
16 export const productReducer = createReducer<ProductState>(  
17   { showProductCode: true } as ProductState,  
18   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {  
19     return {  
20       ...state,  
21       showProductCode: !state.showProductCode,  
22       myFavoriteMovie: 'LOTR'  
23     };  
24   })  
25 );  
26
```

Con eso corregido, ahora nos dice que no podemos simplemente añadir un bit de estado al Store:

```
15 (property) myFavoriteMovie: string
16 export { showProductCode: boolean; myFavoriteMovie: string; currentProduct: Product;
17   on(c products: Product[]: }' is not assignable to type 'ProductState'.
18   re Object literal may only specify known properties, and 'myFavoriteMovie' does not
19   exist in type 'ProductState'. ts(2322)
20   Peek Problem (Alt+F8) No quick fixes available
21   myFavoriteMovie: 'LOTR'
22   };
23   });
24   );
25   );
```

Debemos considerar si queremos esa propiedad en el Store o no. En caso afirmativo, lo añadimos a la interfaz. En caso contrario, lo eliminamos:

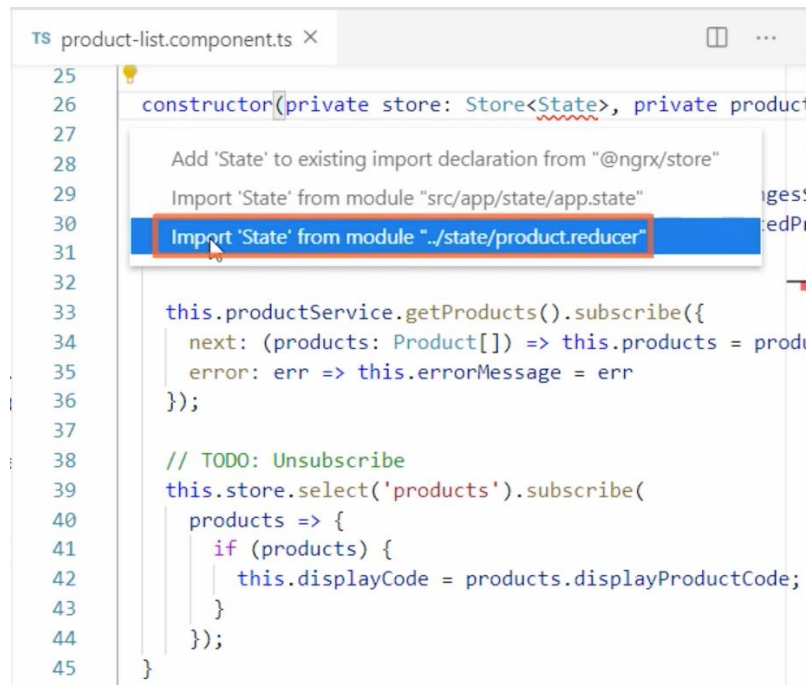
```
16 export const productReducer = createReducer<ProductState>(
17   { showProductCode: true } as ProductState,
18   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
19     return {
20       ...state,
21       showProductCode: !state.showProductCode
22     };
23   })
24 );
```

Contar con tipado fuerte agiliza la identificación de errores tipográfico u ortográficos y ayuda a garantizar que no se añada información no deseada al Store.

Miremos ahora el componente *product.list.component.ts*, podemos ahora tipear fuertemente el Store cuando lo inyectamos:

```
TS product-list.component.ts X
25
26 constructor(private store: Store<any> private productService
27
28 ngOnInit(): void {
29   this.sub = this.productService.selectedProductChanges!
30   selectedProduct => this.selectedProduct = selectedPi
31   );
32
33   this.productService.getProducts().subscribe({
34     next: (products: Product[]) => this.products = produ
35     error: err => this.errorMessage = err
36   });
37
38   // TODO: Unsubscribe
39   this.store.select('products').subscribe(
40     products => {
41       if (products) {
42         this.displayCode = products.displayProductCode;
43       }
44     });
45 }
```

Al inyectar el *Store* en nuestro componente, queremos todo el estado global de la aplicación. Eso nos permitirá acceder a cualquier estado de la aplicación desde nuestro componente. Por ejemplo, las listas de productos pueden necesitar acceso a la información del usuario, como sus preferencias. Escribimos el tipado fuerte en la inyección del *Store* usando su argumento genérico y lo establecemos a *State*, importando su referencia correspondiente:



```
TS product-list.component.ts
25
26 constructor(private store: Store<State>, private productService: ProductService) {}
27
28 // Add 'State' to existing import declaration from "@ngrx/store"
29 // Import 'State' from module "src/app/state/app.state"
30 // Import 'State' from module "../state/product.reducer"
31
32
33 this.productService.getProducts().subscribe({
34   next: (products: Product[]) => this.products = products,
35   error: err => this.errorMessage = err
36 });
37
38 // TODO: Unsubscribe
39 this.store.select('products').subscribe(
40   products => {
41     if (products) {
42       this.displayCode = products.displayProductCode;
43     }
44   }
45 );
46
47 }
```

Recordemos que nuestro módulo principal no incluye el *slice* de estado “*product*”, por lo que debemos importar el estado extendido, como lo definimos en nuestro archivo *product.reducer.ts*.

Ahora tenemos un nuevo error, ya que la propiedad *displayProductCode* a la que queremos acceder en realidad se llama *showProductCode*:

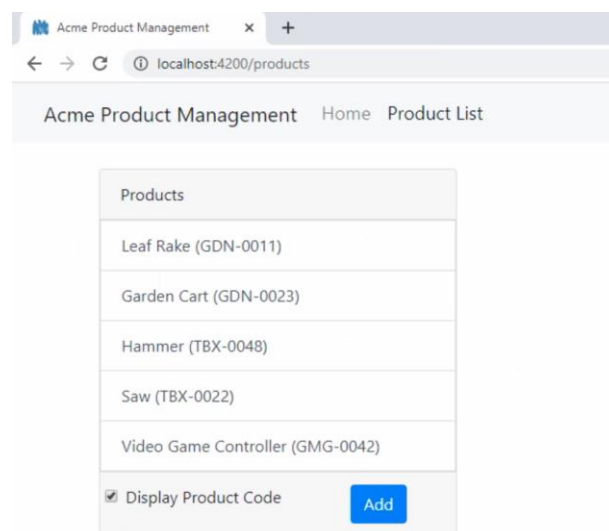


```
TS product-list.component.ts
25 sub: Subscription;
26
27 constructor(private store: Store<State>, private productService: ProductService) {}
28
29 ngOnInit(): void {
30   this.sub = this.productService.selectedProductChanges$.subscribe(
31     selectedProduct => this.selectedProduct = selectedProduct
32   );
33
34   this.productService.getProducts().subscribe({
35     next: (products: Product[]) => this.products = products,
36     error: err => this.errorMessage = err
37   });
38
39   // TODO: Unsubscribe
40   this.store.select('products').subscribe(
41     products => {
42       if (products) {
43         this.displayCode = products.displayProductCode;
44       }
45     }
46   );
47 }
```

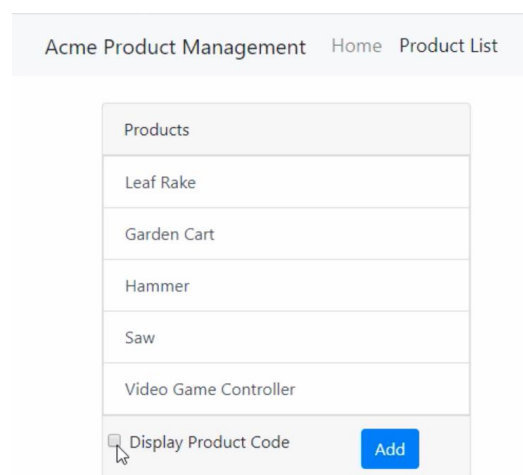
Ahora, si eliminamos la propiedad incorrecta, la función de autocompletar de VS Code proporciona una lista de las propiedades válidas entre las que elegir:

```
39 // TODO: Unsubscribe
40 this.store.select('products').subscribe(
41   products => {
42     if (products) {
43       this.displayCode = products.;
44       currentProduct
45       products
46     } showProductCode (property) ProductSta...
47   }
48 }
```

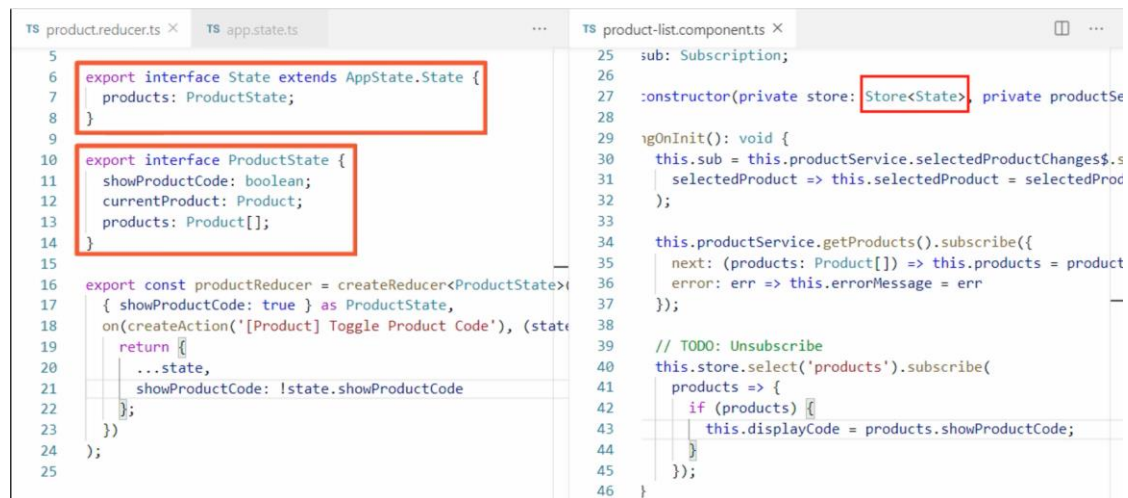
Veamos si funciona. Volvemos al navegador, y nuestro estado inicial marca el checkbox mostrando los códigos de los productos:



Desmarcando el checkbox, los códigos de los productos se ocultan:



Mirando el código, todo lo que se necesitaba para tipear fuertemente nuestro estado era definir un conjunto de interfaces que establecieran la estructura de ese estado. Luego usamos esas interfaces como tipo cada vez que hacemos referencia al estado global de la aplicación o a cualquier slice de estado:



```
TS product.reducer.ts X TS app.state.ts ... TS product-list.component.ts X ...
5
6 export interface State extends AppState.State {
7   products: ProductState;
8 }
9
10 export interface ProductState {
11   showProductCode: boolean;
12   currentProduct: Product;
13   products: Product[];
14 }
15
16 export const productReducer = createReducer<ProductState>(
17   { showProductCode: true } as ProductState,
18   on(createAction('[Product] Toggle Product Code'), (state) => {
19     return {
20       ...state,
21       showProductCode: !state.showProductCode
22     };
23   });
24 );
25
25 sub: Subscription;
26
27 constructor(private store: Store<State>, private productService: ProductService) {}
28
29 ngOnInit(): void {
30   this.sub = this.productService.selectedProductChanges$.subscribe(
31     selectedProduct => this.selectedProduct = selectedProduct;
32   );
33
34   this.productService.getProducts().subscribe({
35     next: (products: Product[]) => this.products = products,
36     error: err => this.errorMessage = err
37   });
38
39   // TODO: Unsubscribe
40   this.store.select('products').subscribe(
41     products => {
42       if (products) {
43         this.displayCode = products.showProductCode;
44       }
45     }
46   );
47 }
```

Sin embargo, sería un poco más limpio si definiéramos claramente los valores iniciales de nuestro estado.

Hagámoslo a continuación.