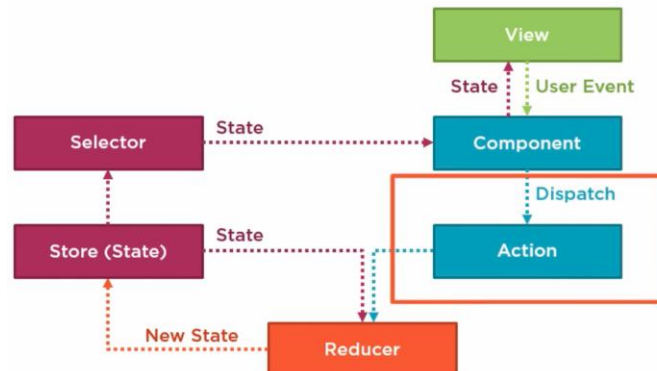


## Tipado fuerte de Actions mediante Action Creators - Introducción

Hemos tipado fuertemente nuestro estado, pero ¿qué pasa con nuestros *Actions*? Ahora, usaremos *Action Creators* para tipar fuertemente nuestros *Actions*.

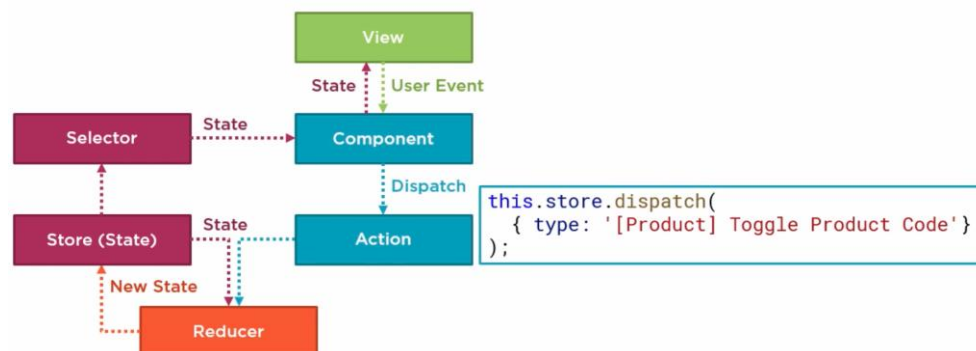
Con NgRx, es el *Action* el que hace funcionar la aplicación:



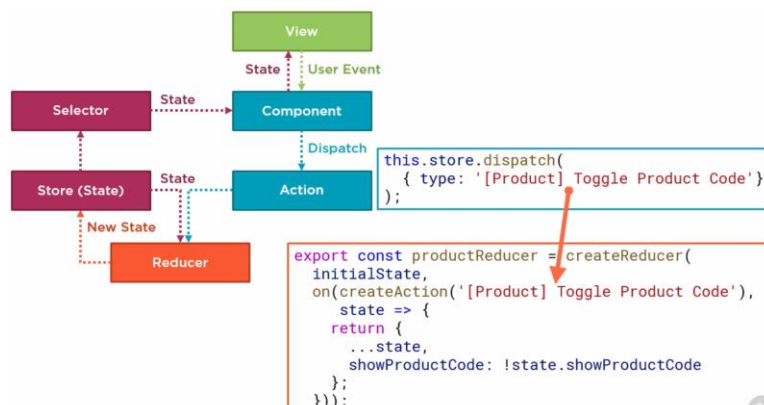
Si el usuario hace clic en algo, despachamos un *Action*; si necesitamos cargar datos, despachamos un *Action*; Si el usuario actualizar un valor y hace clic en guardar, despachamos otro *Action*.

El resultado del *Action* es a menudo un cambio de estado.

Actualmente, despachamos nuestros *Actions* usando un string:



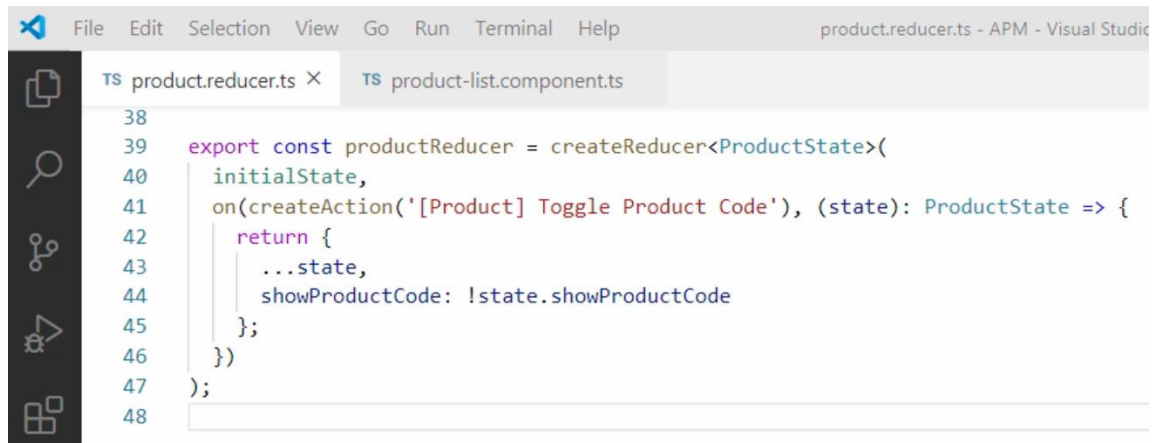
El *Reducer* luego coincide con ese string para procesar el *Action* y devolver el nuevo estado al *Store*:



Si tipado fuerte, es fácil cometer un error. Además, no podemos obtener una lista del conjunto disponible de strings de tipo *Action*.

Echemos un vistazo más de cerca al estado de nuestros Actions y los posibles problemas con nuestro enfoque actual sin tipado fuerte.

Volvamos a nuestro proyecto de ejemplo:



```
38
39 export const productReducer = createReducer<ProductState>(
40   initialState,
41   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
42     return {
43       ...state,
44       showProductCode: !state.showProductCode
45     };
46   })
47 );
48
```

Comencemos en el *product.reducer.ts*. Aquí llamamos a *createAction*, pasando un string en *hard-coded*, por lo que es fácil cometer un error tipográfico:



```
38
39 export const productReducer = createReducer<ProductState>(
40   initialState,
41   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
42     return {
43       ...state,
44       showProductCode: !state.showProductCode
45     };
46   })
47 );
48
```



```
38
39 export const productReducer = createReducer<ProductState>(
40   initialState,
41   on(createAction('[Product] Toggle Product Code'), (state): ProductState => {
42     return {
43       ...state,
44       showProductCode: !state.showProductCode
45     };
46   })
47 );
48
```

Sin tipado fuerte significa que no hay ayuda con nuestros errores tipográficos.

Mirando el *product-list.component.ts*, aquí despachamos el Action, de nuevo con un string en hard-coded:

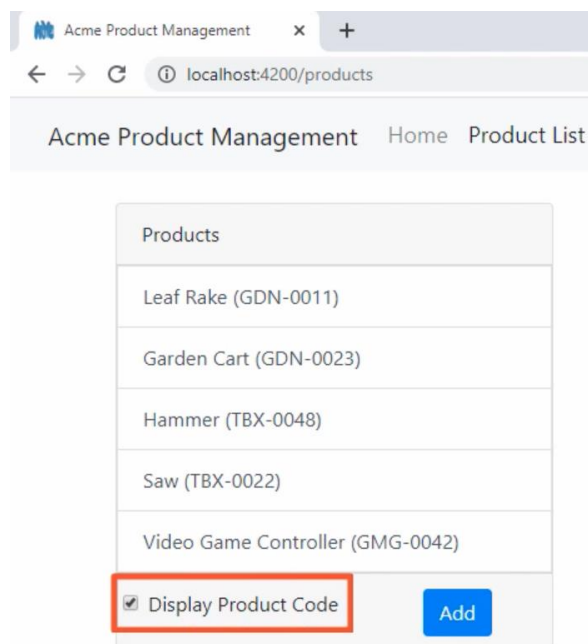
```
TS product.reducer.ts TS product-list.component.ts X
48
49 checkChanged(): void {
50   this.store.dispatch(
51     { type: '[Product] Toggle Product Code' }
52   );
53 }
54
55 newProduct(): void {
56   this.productService.changeSelectedProduct(this.productService.newProduct());
57 }
58
59 productSelected(product: Product): void {
60   this.productService.changeSelectedProduct(product);
61 }
62
63
64 }
```

En esta ocasión hemos escrito correctamente el nombre del Action. Veamos si corre:

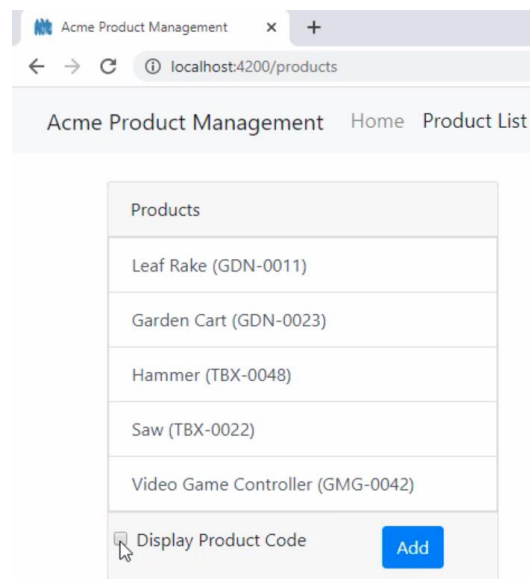
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: node
5 unchanged chunks
chunk {products-product-module} products-product-module.js, products-product-module.js.map (products-product-module) 53 kB [rendered]
Time: 191ms
: Compiled successfully.
```

No hay errores de sintaxis, por lo que nada nos dice sobre el error que acabamos de crear.

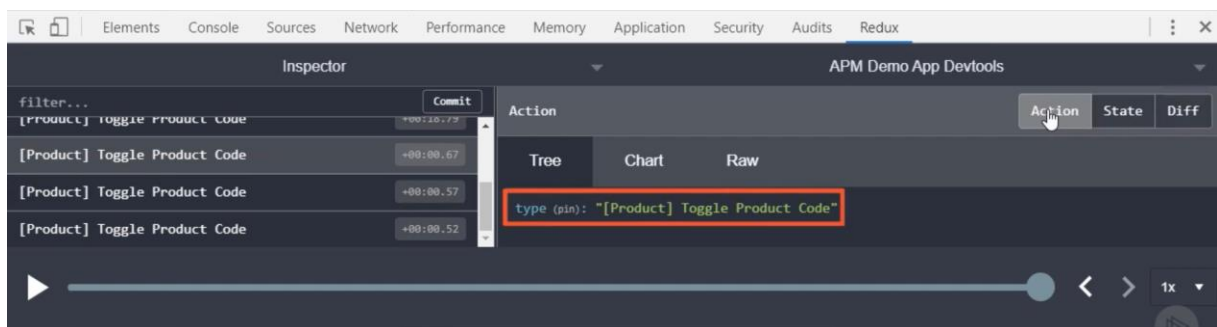
Al verlo en el navegador, nuestro checkbox *Display Product Code* se establece a true de forma predeterminada, según el código de estado inicial que agregamos en el módulo:



Pero el checkbox ya no alterna los códigos de producto:



Abramos las herramientas de desarrollo de NgRx para ver si pueden ayudarnos a resolver este problema. Seleccionamos uno de los *Actions* de *Toggle Product Code*:



Se ve bien.

No hay nada que nos ayude con nuestro error tipográfico de nuestro Reducer.