Manage Local State with Component Store

Duncan Hunter

Lead Frontend Developer

@dunchunter | duncanhunter.com.au

Module Introduction

Why and when use component-store?

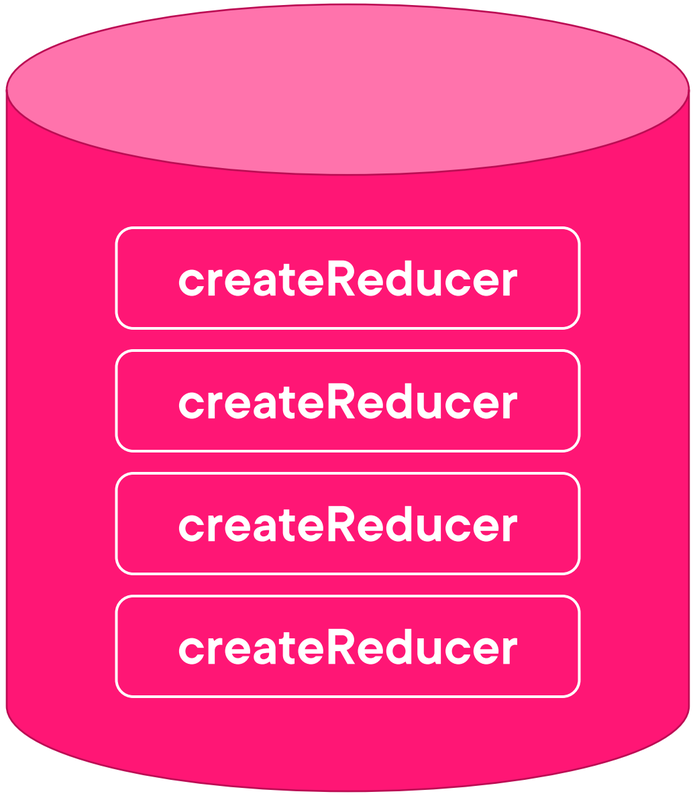
Add @ngrx/component-store

@ngrx/component-store vs @ngrx/store

ComponentStore is a stand-alone library that helps to manage local/component state.

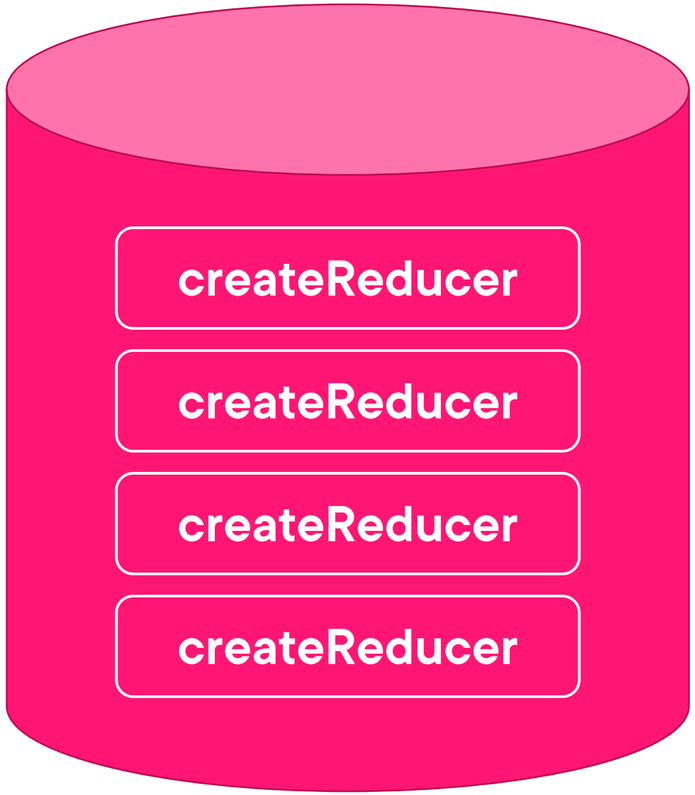
@ngrx/store

Global state



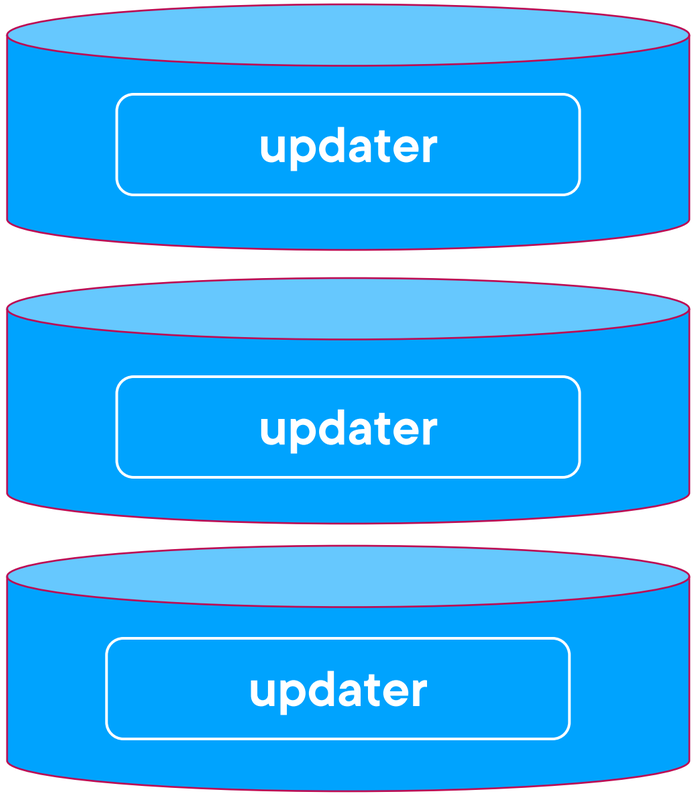
@ngrx/store

Global state



@ngrx/component-store

Independent states



Why Component Store

Simpler single file

Independent state

Similar NgRx API

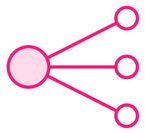
Component Store - Key Concepts

Local state must be initialized

Local state is typically tied to a single components lifecycle

Update state via setState or an updater

Read state with selectors



Manage side effects with effects

Component Store

```
@Injectable()
export class ProductsStore extends ComponentStore<ProductsState> {


 constructor(private productsService: ProductsService) {
  super({ products: [] });
 }




}
```

Component Store

```
@Injectable()
export class ProductsStore extends ComponentStore<ProductsState> {


 constructor(private productsService: ProductsService) {
  super({ products: [] });
 }

 addProducts = this.updater((state, products: Product[]) => ({
  ...state,
  products,
 }));




}
```

Component Store

```
@Injectable()
export class ProductsStore extends ComponentStore<ProductsState> {


 constructor(private productsService: ProductsService) {
  super({ products: [] });
 }

 addProducts = this.updater((state, products: Product[]) => ({
  ...state,
  products,
 }));

 getProducts = this.effect<void>((trigger$) =>
  trigger$.pipe(exhaustMap(() => this.productsService.getAll().pipe(
   tap({ next: (products) => this.addProducts(products)})
 ))));
}
```

Component Store

```
@Injectable()
export class ProductsStore extends ComponentStore<ProductsState> {
 products$ = this.select((state) => state.products);

 constructor(private productsService: ProductsService) {
  super({ products: [] });
 }

 addProducts = this.updater((state, products: Product[]) => ({
  ...state,
  products,
 }));




}
```

Component Store

```
@Injectable()
export class ProductsStore extends ComponentStore<ProductsState> {
 products$ = this.select((state) => state.products);

 constructor(private productsService: ProductsService) {
  super({ products: [] });
 }

 addProducts = this.updater((state, products: Product[]) => ({
  ...state,
  products,
 }));

 getProducts = this.effect<void>((trigger$) =>
  trigger$.pipe(exhaustMap(() => this.productsService.getAll().pipe(
   tap({ next: (products) => this.addProducts(products)})
 ))));
}
```

Component Store

```
@Component({
 selector: ‘app-products-page’,
 templateUrl: ‘./products-page.component.html’,
 styleUrls: [‘./products-page.component.css’],
 providers: [ProductsStore],
})
export class ProductsPageComponent {
 products$ = this.productsStore.products$;

 constructor(private productsStore: ProductsStore) {}

 ngOnInit() {
  this.productsStore.getProducts();
 }

}
```

Component Store

```
@Component({
 selector: ‘app-products-page’,
 templateUrl: ‘./products-page.component.html’,
 styleUrls: [‘./products-page.component.css’],
 providers: [ProductsStore],
})
export class ProductsPageComponent {
 products$ = this.productsStore.products$;

 constructor(private productsStore: ProductsStore) {}

 ngOnInit() {
  this.productsStore.getProducts();
 }

}
```

Component Store

```
@Component({
 selector: ‘app-products-page’,
 templateUrl: ‘./products-page.component.html’,
 styleUrls: [‘./products-page.component.css’],
 providers: [ProductsStore],
})
export class ProductsPageComponent {
 products$ = this.productsStore.products$;

 constructor(private productsStore: ProductsStore) {}

 ngOnInit() {
  this.productsStore.getProducts();
 }

}
```

Component Store

```
@Component({
 selector: ‘app-products-page’,
 templateUrl: ‘./products-page.component.html’,
 styleUrls: [‘./products-page.component.css’],
 providers: [ProductsStore],
})
export class ProductsPageComponent {
 products$ = this.productsStore.products$;

 constructor(private productsStore: ProductsStore) {}

 ngOnInit() {
  this.productsStore.getProducts();
 }

}
```

Demo

Add and use @ngrx/component-store