


```
[ { id: 1, name: "Foo" },  
  { id: 2, name: "Bar" }  
]
```

_____ vs. _____

```
{  
  entities: {  
    1: { id: 1, name: "Foo" },  
    2: { id: 2, name: "Bar" }  
  },  
  ids: [1, 2]  
}
```

NgRx Container / Presentation

Architecture pattern to help make things reusable and testable.

Container → *Input/Output* ← *Presentation*

- Container = how things work. Loading and Dispatching.
- Presentation = presenting the data, how things work, but doesn't deal with observables.

NgRx Entity

- Provides set of the functions for CRUD operations
- @ngrx/entity lets you create entity adapters for different kinds of entities. Using an entity adapter, you can quickly write reducer operations and automatically generate selectors.

```
interface Book {
```

```
  id: string;
```

```
  title: string;
```

```
}
```

```
import { createEntityAdapter } from '@ngrx/entity';
```

```
const bookAdapter = createEntityAdapter<Book>();
```

```
export const {
```

```
  selectIds, selectEntities, selectAll, selectTotal,
```

```
} = bookAdapter.getSelectors();
```

```
[ { id: 1, name: "Foo" },  
  { id: 2, name: "Bar" }  
]
```

vs.

```
{  
  entities: {  
    1: { id: 1, name: "Foo" },  
    2: { id: 2, name: "Bar" }  
  },  
  ids: [1, 2]  
}
```