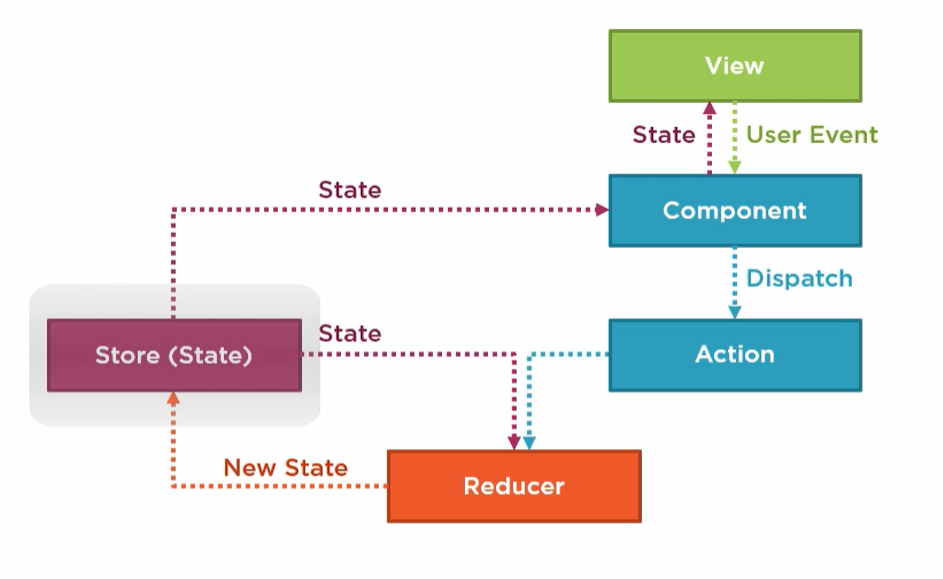
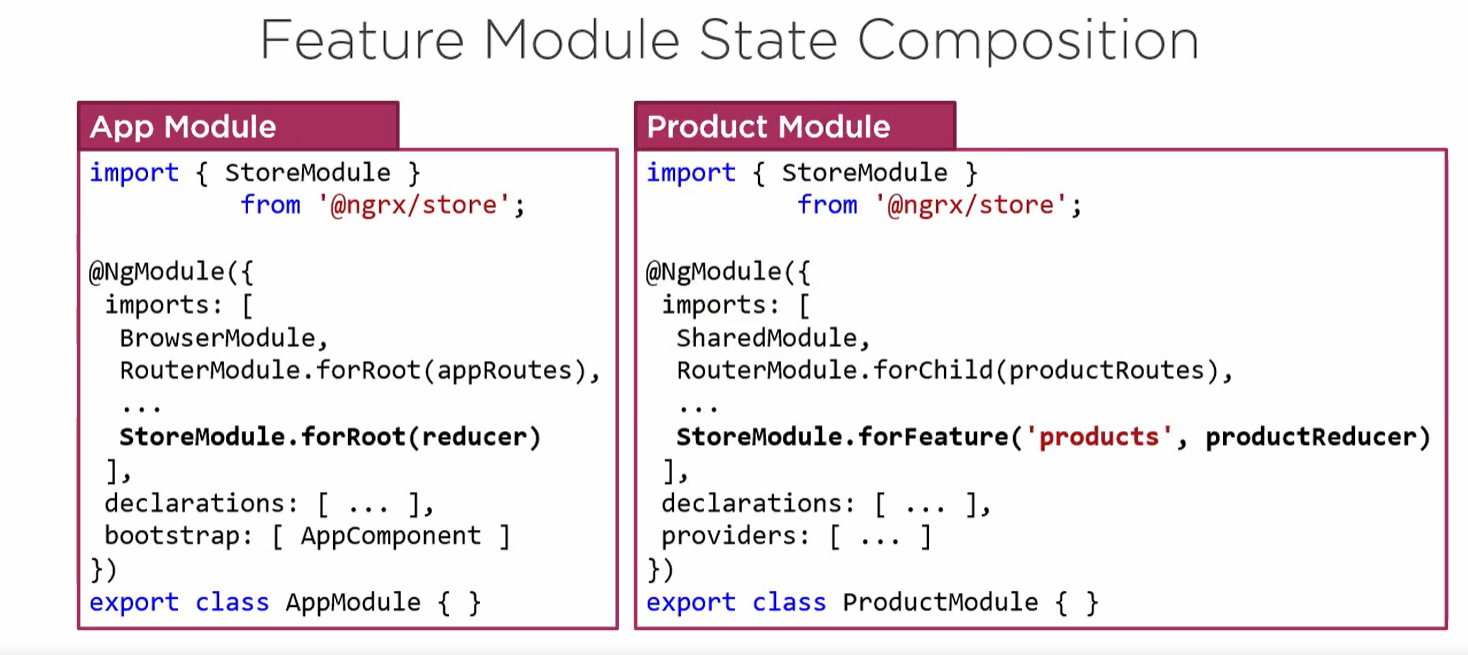
**First Look at NgRx**

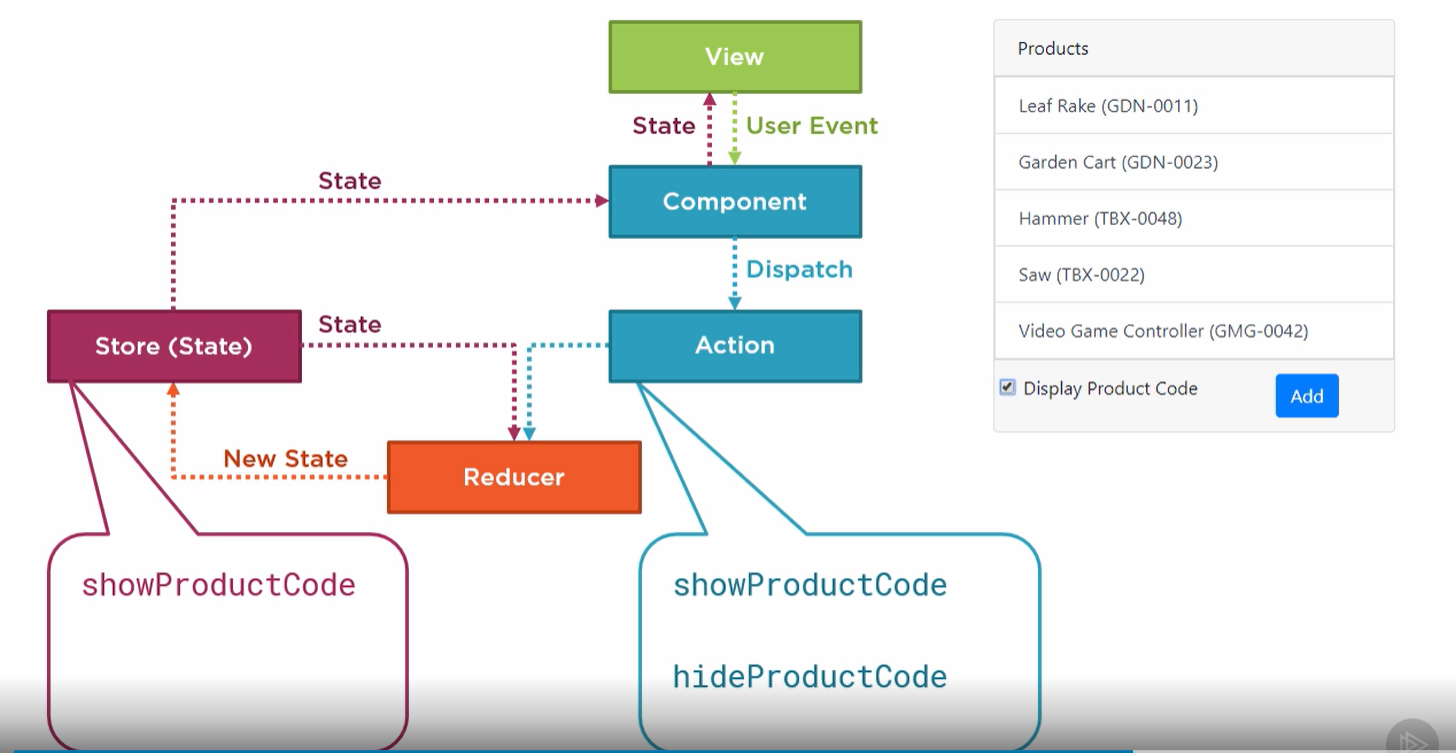


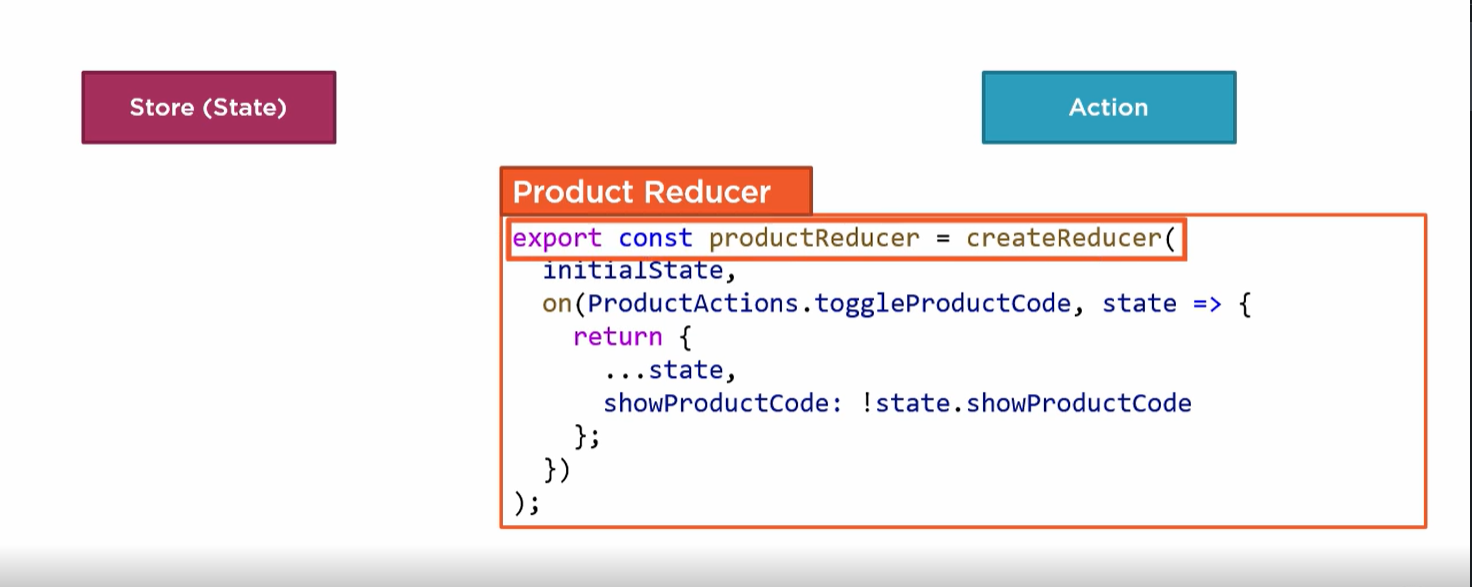
1. არის NgRx - ის დაინსტალირების რამდენიმე გზა: 1. Npm install @ngrx/store. 2. Yarn add @ngrx/store. 3. Ng add @ngrx/store. მოცემულ შემთხვევაში, გამოვიყენებთ მესამე გზას, რადგან ამ გზის ბენეფიტს წარმოადგენს ის, რომ ანგულარი თავად დააიმპორტებს NgRx - ს App Module - ში.
2. @ngrx/store - ის დამატების შემდეგ, App Module - ში დაემატება შემდეგი იმპორტი: StoreModule.forRoot({}, {});



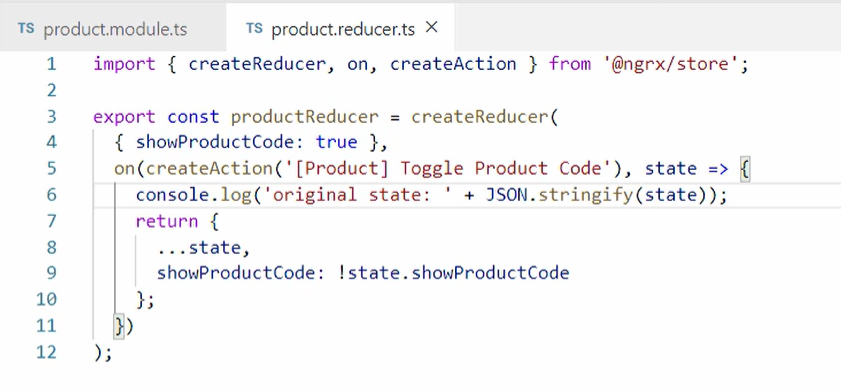


1. StoreModule.forRoot({}, {}) - ღებულობს 2 პარამეტრს, პირველი არის Store - ის Reducer, ხოლო მეორე პარამეტრი კი არის ნებაყოფლობითი, კონფიგურაციისთვის. იმ შემთხვევაში, თუ არ გვაქვს Reducer, App Module - სთვის, პარამეტრებად შეგვიძლია გადავცეთ ორი ცარიელი ობიექტი.
2. იმისათვის, რომ სხვა მოდულში შევძლოთ Store - ის გამოყენება, აუცილებელია თავდაპირველად დავაიმპორტოთ StoreModule, შესაბამის მოდულში. Import { StoreModule } from “@ngrx/store”; ხოლო თავად მოდულის იმპორტების სიაში კი შემოვიტანოთ, StoreModule.forFeature(“products”, {});





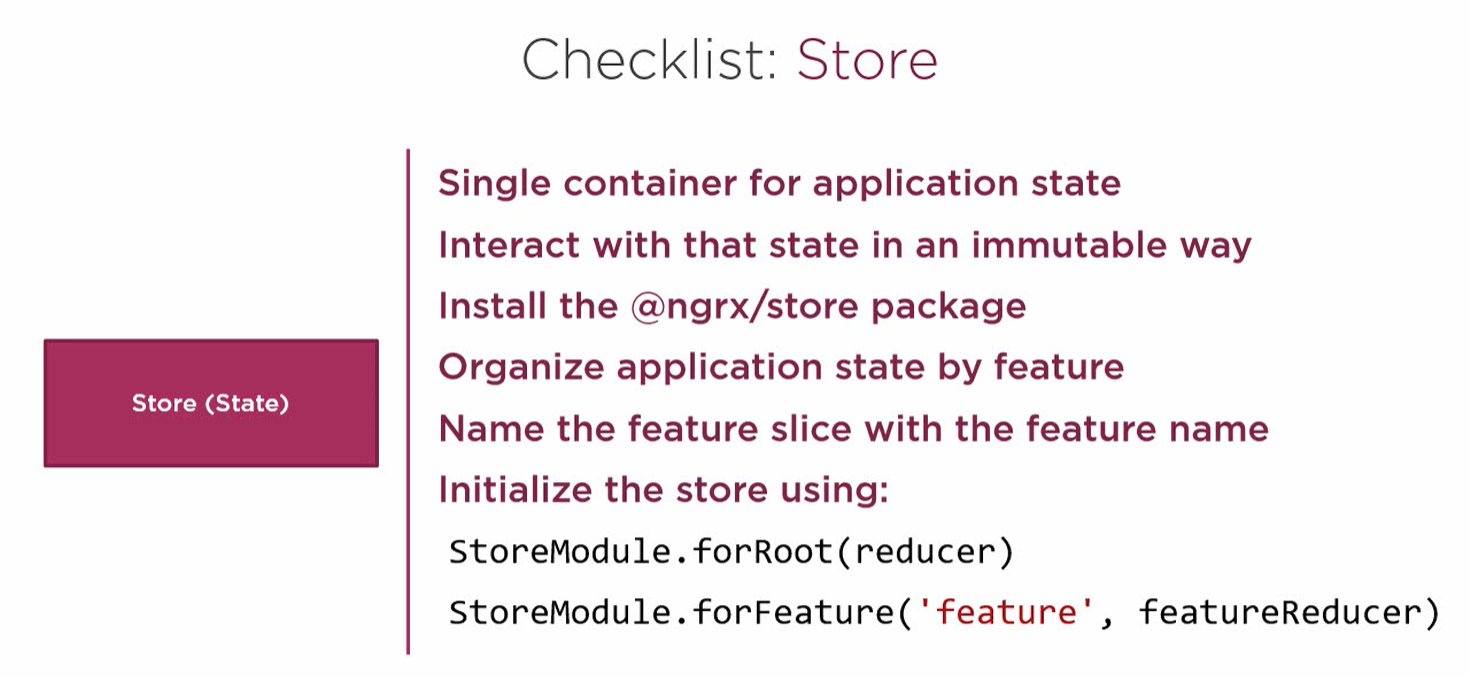
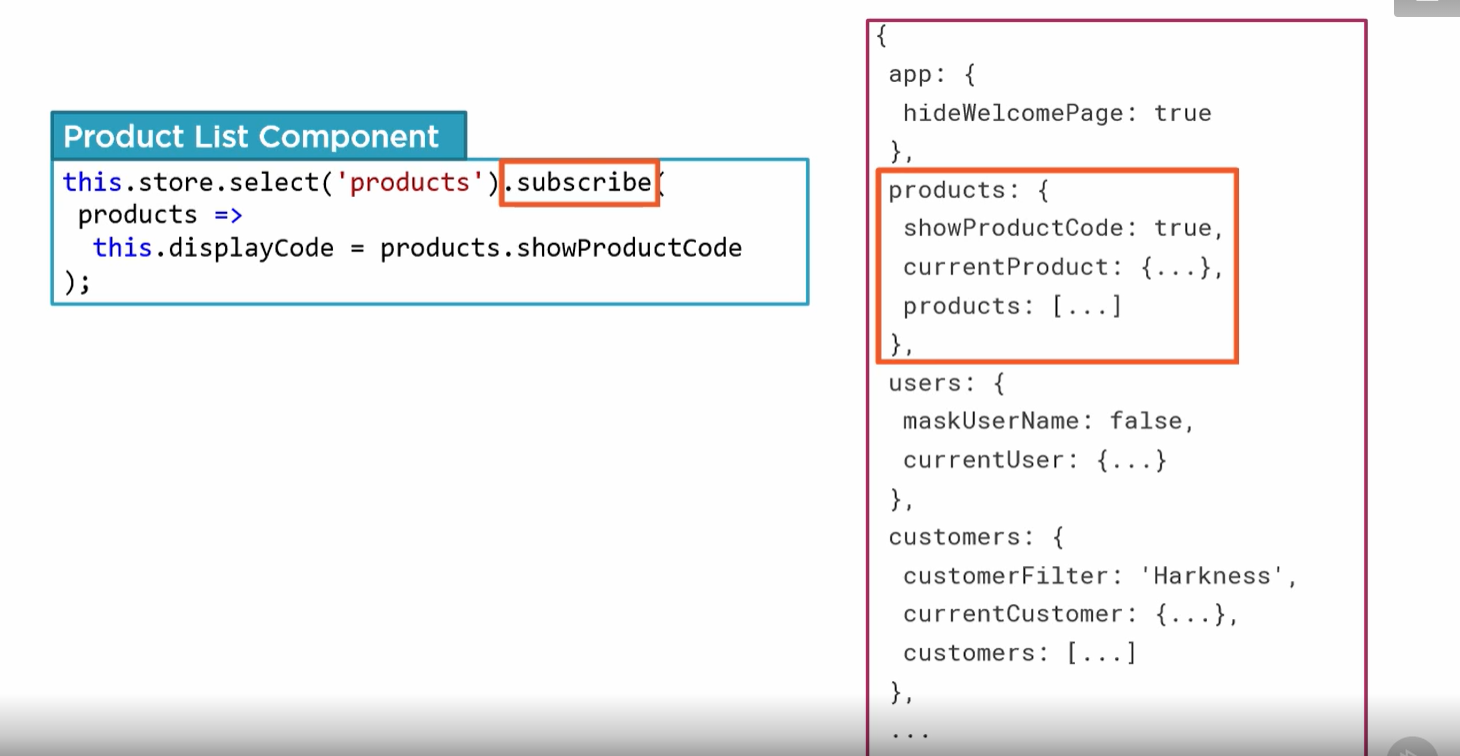
1. ზემოთ განთავსებულ სურათზე, გამოსახულია Reducer. Reducer არის ფუნქცია, რომელიც იღებს 2 არგუმენტს, პირველი არგუმენტი არის თავდაპირველი State, ხოლო მეორე არგუმენტი კი არის Action. ჩვენ განვსაზღვრავთ Action - ებს და მათ Handler - ებს, ერთი ან მეტი on ფუნქციის გამოყენებით. on ფუნქცია, უნდა დავაიმპორტოთ @ngrx/store - დან. on ფუნქცია, იღებს თავად Actions და Handler ფუნქციას. როდესაც მოხდება Action, გამოიძახება Handler ფუნქცია, რომელიც დააბრუნებს ახალ State - ს. Reducer მთლიანად არასდროს არ შეცვლის უკვე არსებულ State - ს. როგორც სურათშია აღნუშნული, ის სპრედ ოპერატრის საშუალებით, დააკოპირებს ძველ, State - ს და მასთან გააერთიანებს ახალ Property - ს, იმ შემთხვევაში, თუ ეს Property უკვე არსებობს, მაშინ ფუნქცია, უბრალოდ შეცვლის მის მნიშვნელობას.

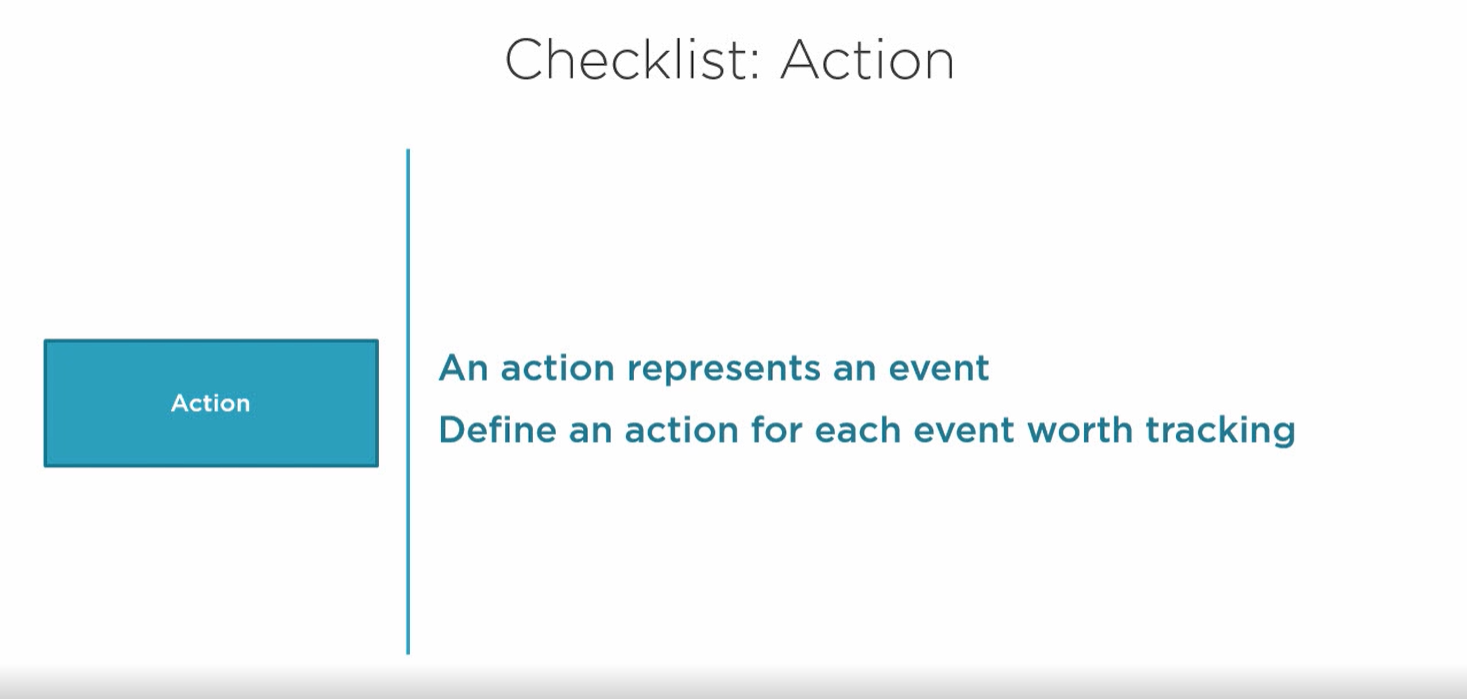


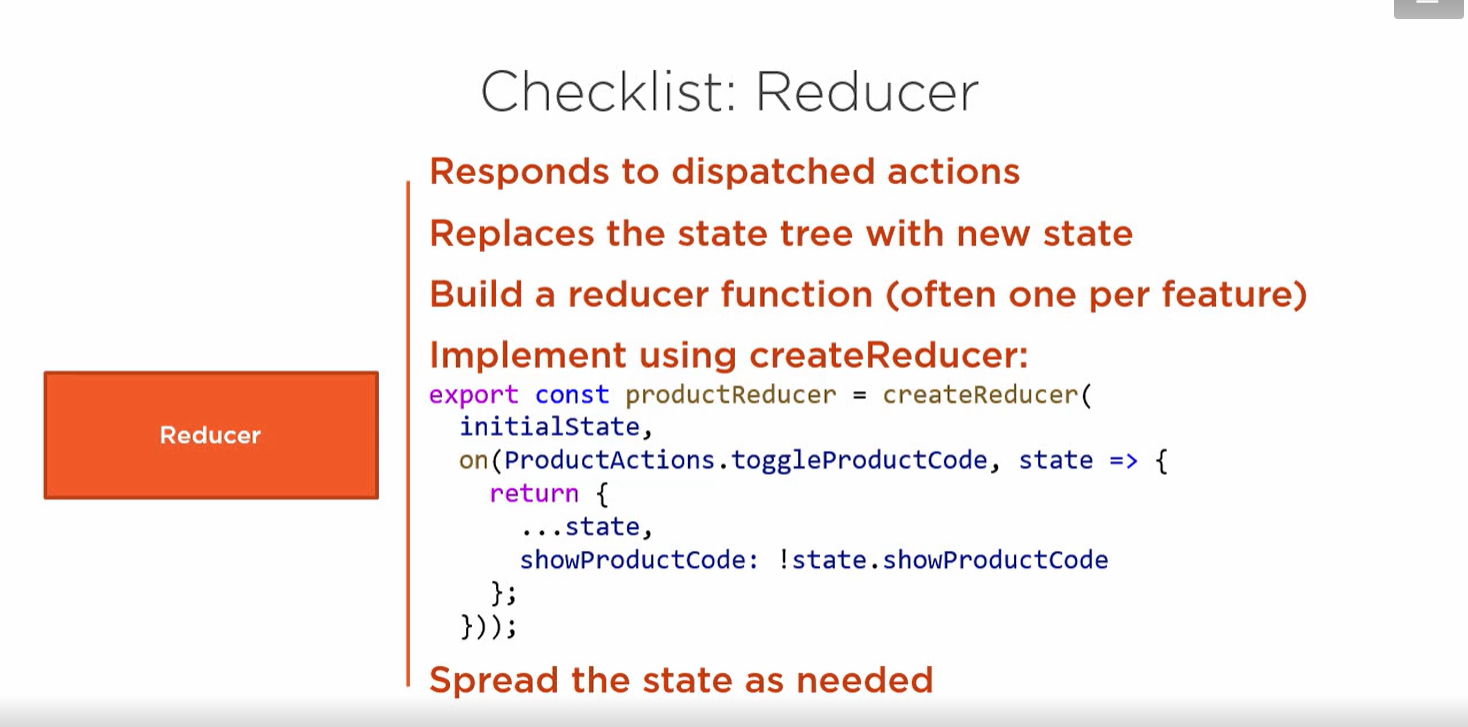
1. მოცემულ სურათზე, Action არ არის ცალკეულად განსაზღვრული და განსაზღვრულია on ფუნქციაში, createAction - ით, რომელმაც მიიღო არგუმენტი, რომელიც წარმოადგენს Action - ის აღწერას.
2. როდესაც შევქმნით Reducer - ის ფუნქციას, ამის შემდეგ აუცილებელია, რომ შესაბამის მოდულში დავამატოთ აღნიშნული რიდიუსერი, StoreModule.forFeature(“products”, . productReducer);

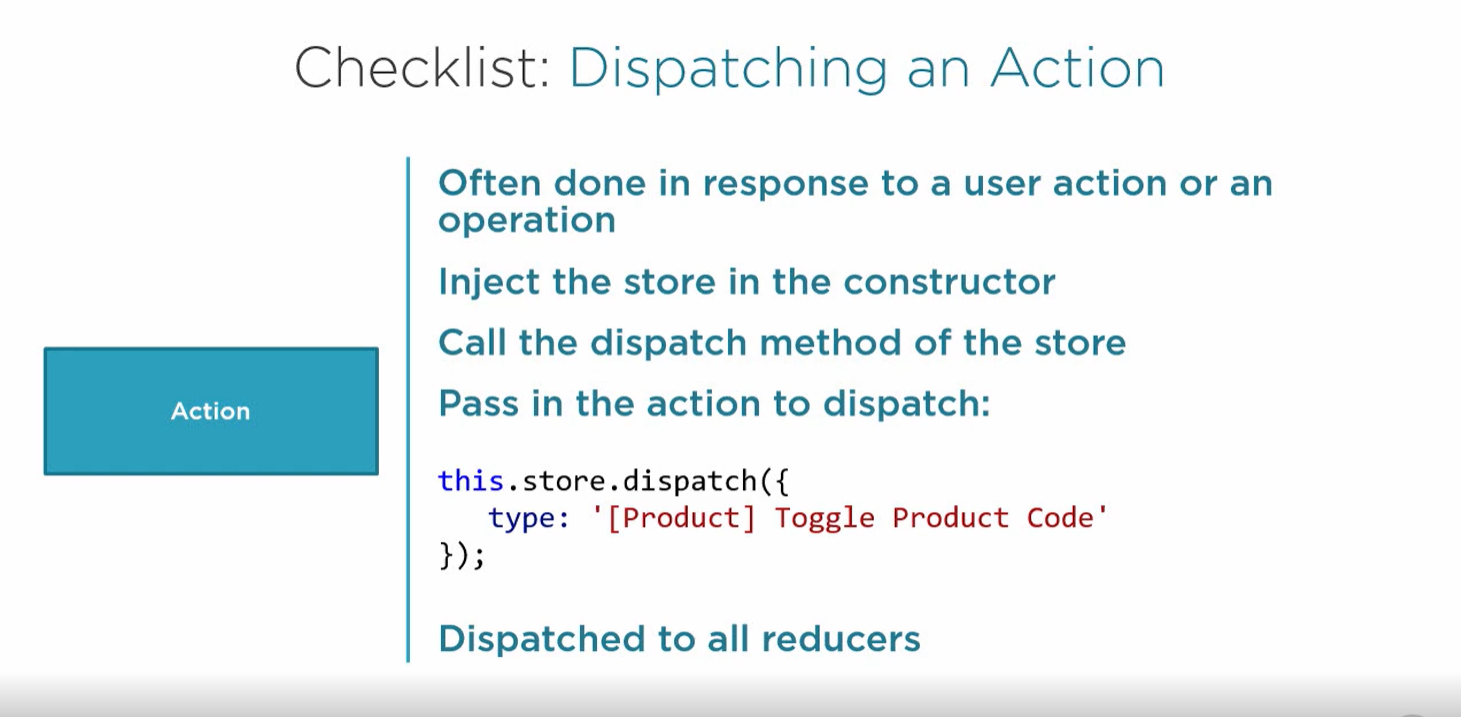


1. იმისათვის, რომ შევძლოთ Action - ის გამოძახება (Dispatch) - ამისათვის, აუცილებელია, ჩვენთვის სასურველ კომპონენტში, დავაინჯექთოთ Store. Store - მა უნდა მიიღოს Generic type ამიტომ იგი უნდა დავაინჯექთოთ შემდეგნაირად: private store: Store<any>.
2. იმისათვის, რომ შევძლოთ Action - ის გამოძახება, უნდა მოვიქცეთ შემდეგნაირად: this.store.dispatch({ type: “[Product] Toggle Product Code” }) . მოცემულ შემთხვევაში, ვინაიდან Action ცალკეულად არ გვაქვს გაწერილი, მას შეგვიძლია მივწვდეთ თავისი აღწერილობით.
3. იმისათვის, რომ Store - დან შევძლოთ ჩვენთვის სასურველი Slice - ის წამოღება, და შემდეგ ამ სლაისში არსებულ ფროფერთიზე წვდომა ამისათვის შეგვიძლია გამოვიყენოთ ორი მეთოდი. პირველი მეთოდი არის: this.store.select(“products”), ხოლო რადგან Store წარმოადგენს Observables - მეორე მეთოდად შეგვიძლია გამოვიყენოთ: this.store.pipe(select(“products”))

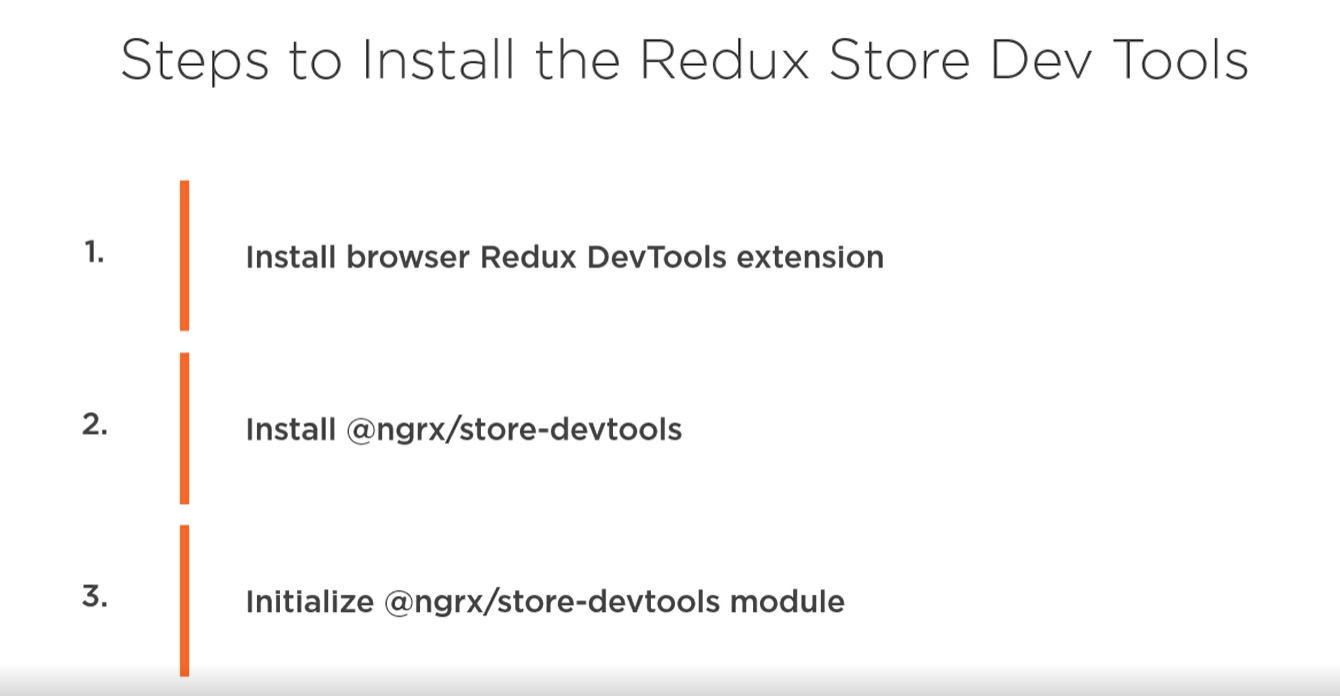












**Strongly Typing the State**

