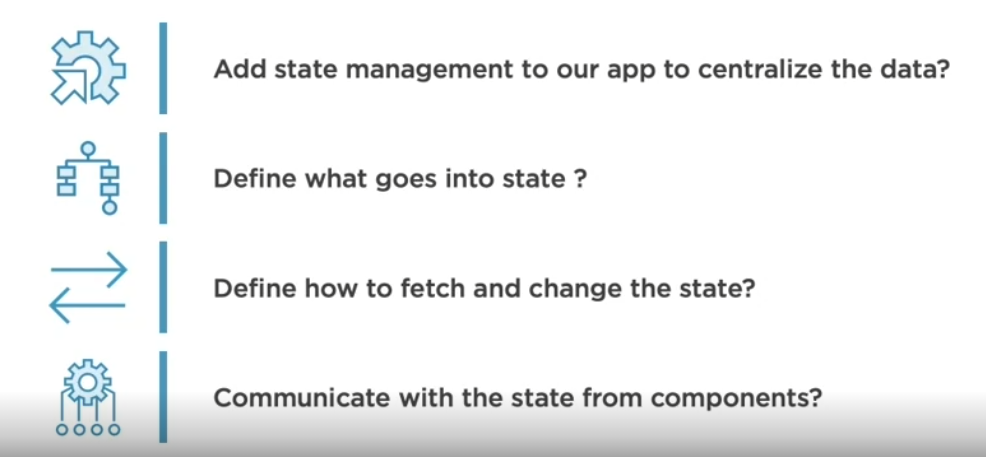
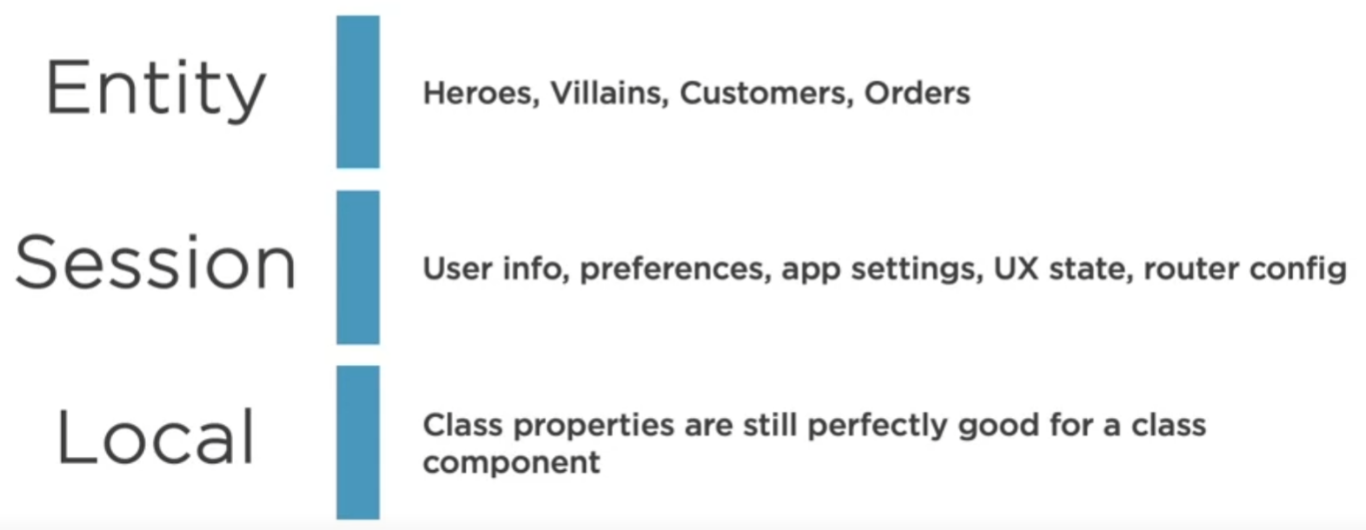
* Manage data state with vuex



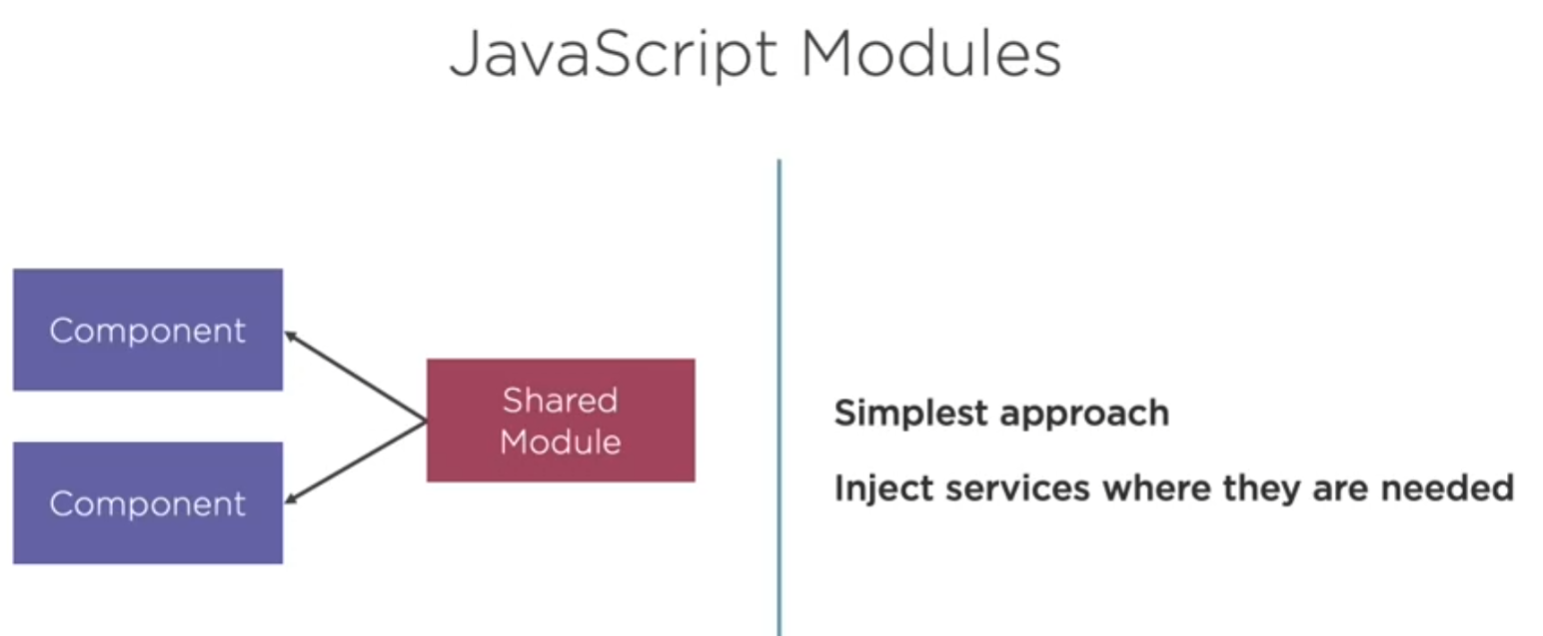
“Store” : the place we hold our data and state

* Problems that a central store can help with us:

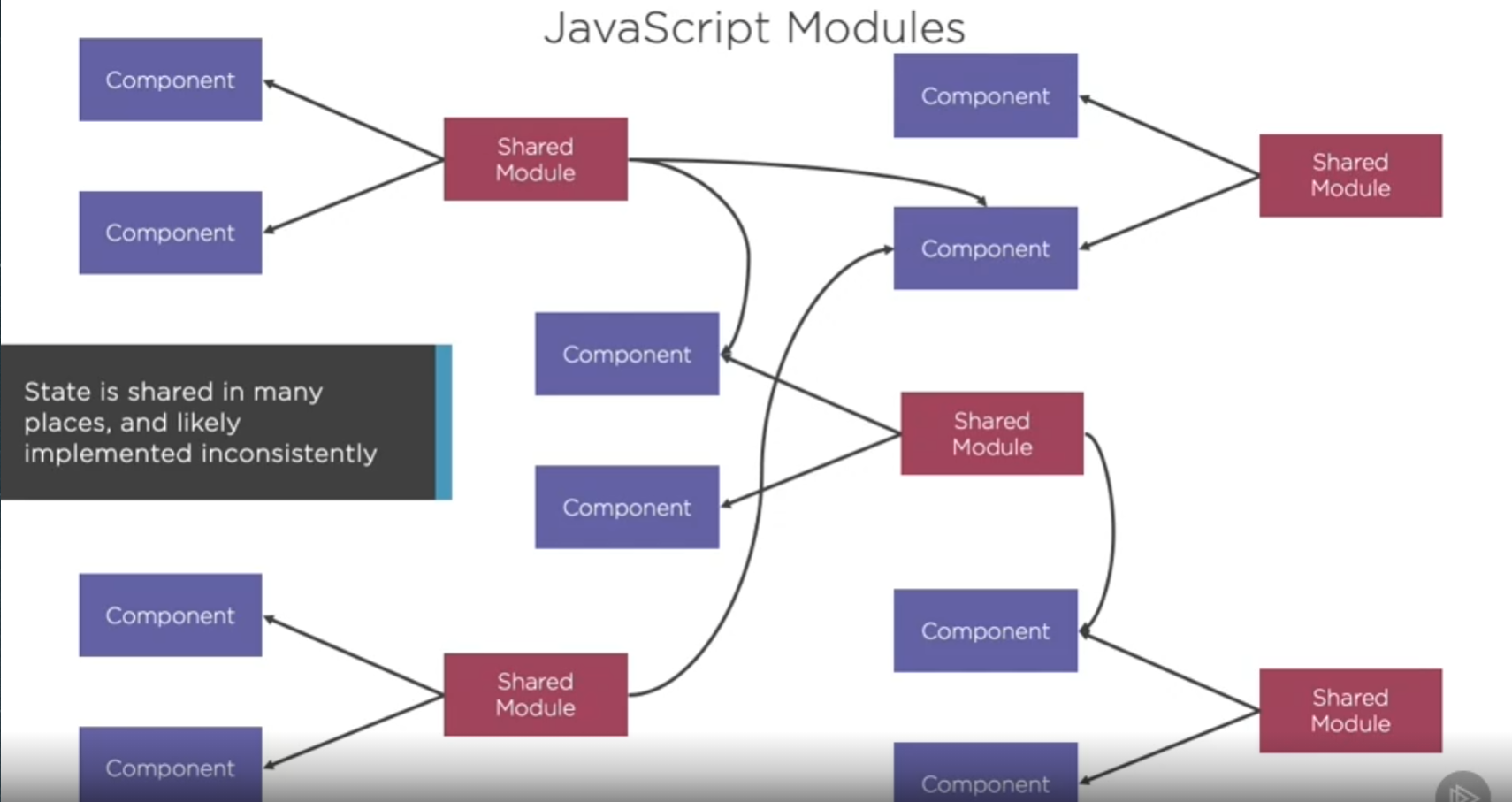
1. Types of shared state through whole app

Way of sharing data or state through out component:

1. Js module:



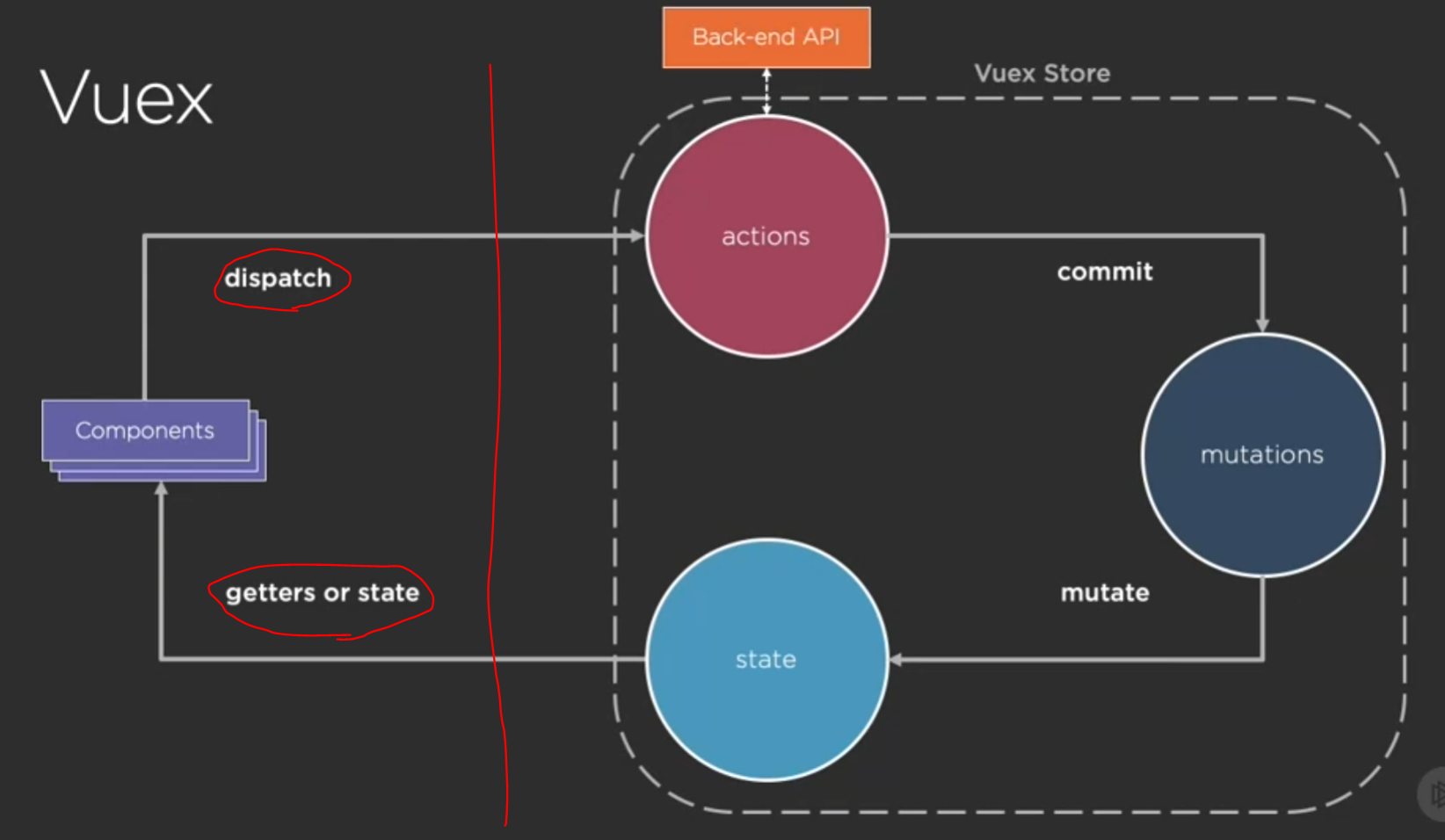
>> but if things got complicated, app will be messed up….



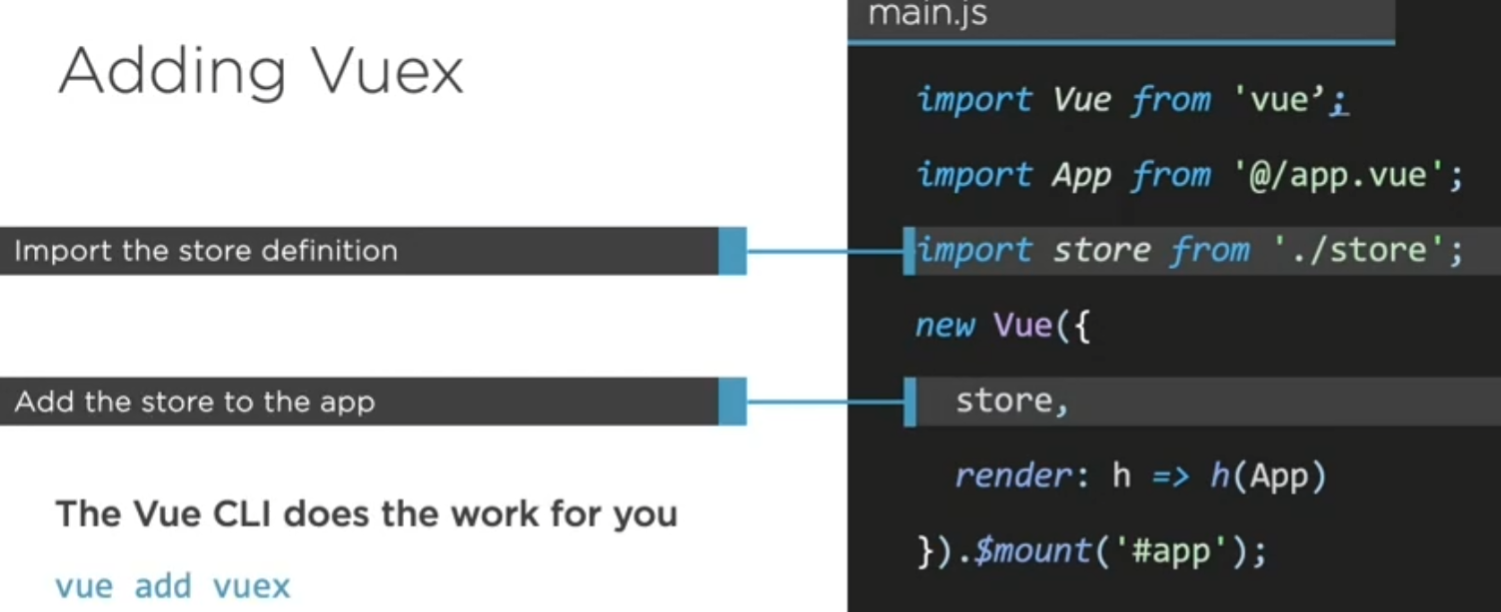
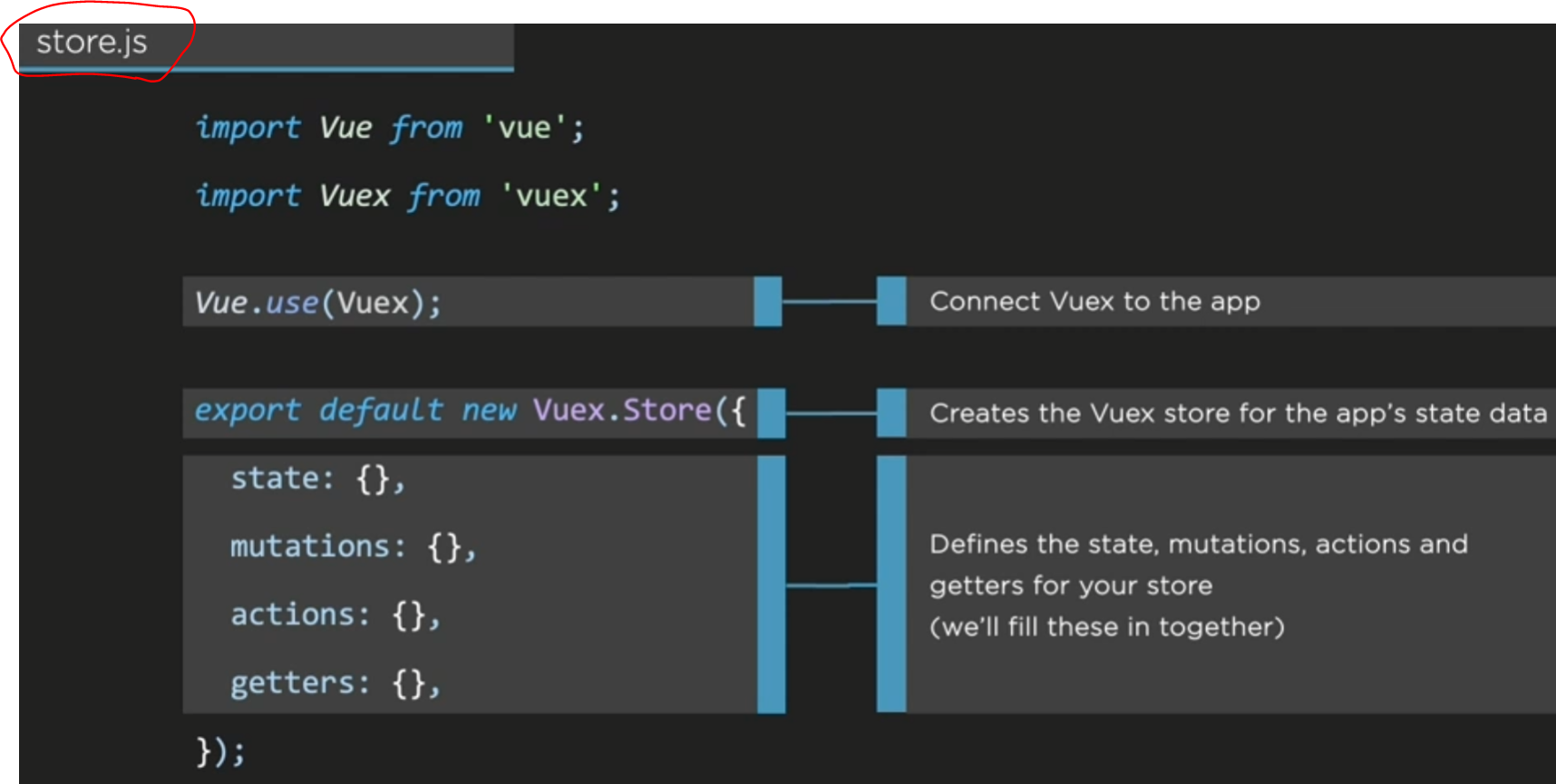
So, using vuex can create single source of truth for you

>> Data flow of vuex:

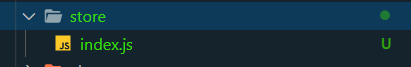
1. **Component** dispatch actions (sending message)
2. **Actions** then commit to mutations (telling to change our state)
3. **Mutation** mutate our state
4. **Finally the component get data from new state**



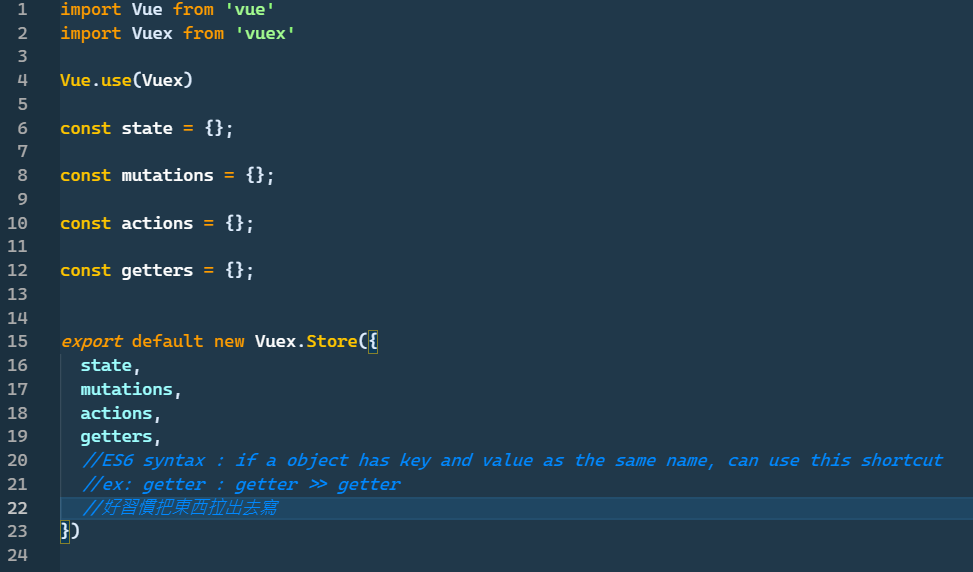
Step:

1. Add vuex >> vue add vuex (vue cli)
2. Customize your store
3. Tell your app about the store
4. Make your component communicate with your store
5. 
6. 
7. Store.js is base template provided by cli
8. Create store folder and rename store to index.js

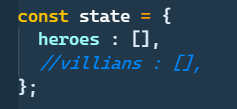
>> cause there are other files to work with store



Code Demo:



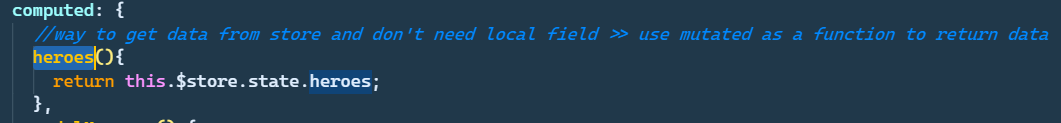
* Describe your state (data you want in your store)

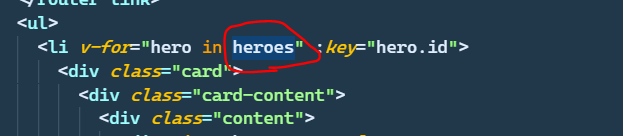


First wat to get data from state: >> this.$store.state.heroes;

>> but this need local field and have to put it in data

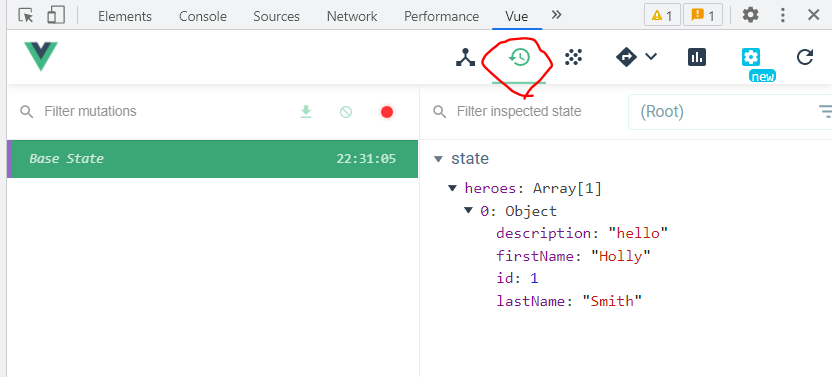
Second way >> use mutated





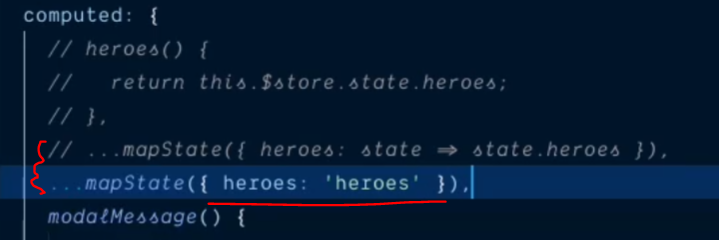
>> 這邊的heroes現在變成function，一樣可以bind到資料

* Tips in chrome extension



>> can see what the state is going on

\*\* two other way to get your state: use mapState



Line 1: go get the state object and return the state.heroes

Line 2: just directely get the heroes state from state

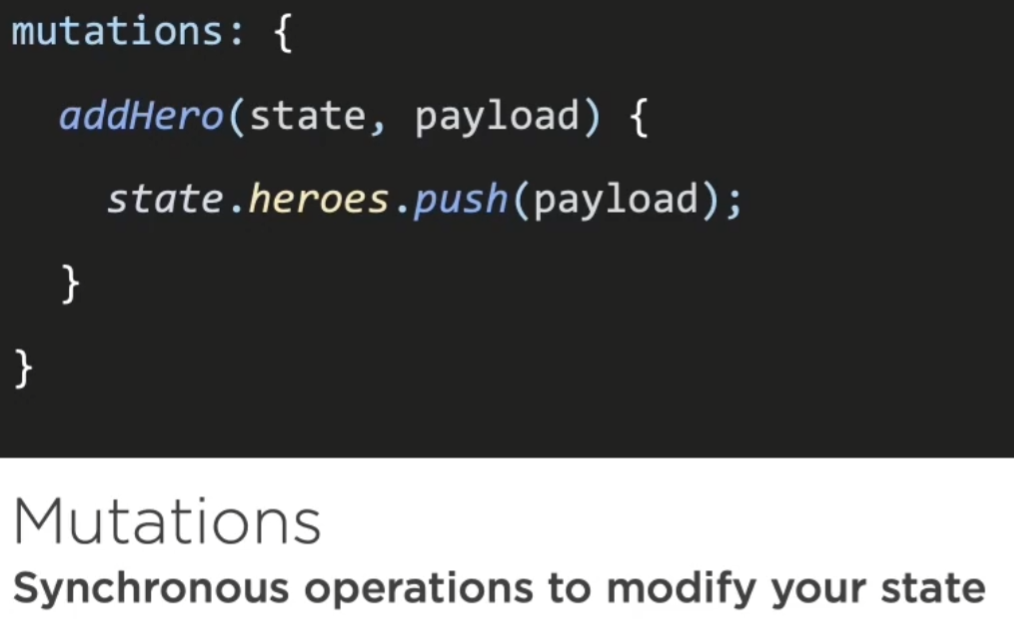
\*\* and because they share the same name (state and local assignment),it can be even simplified to:

 >> final answer!!

>> it means go get the heroes from state and back here create a local property also called heroes

**Tip: ‘… (spread)’ + ‘mapState func’ + ‘state in arr form’**

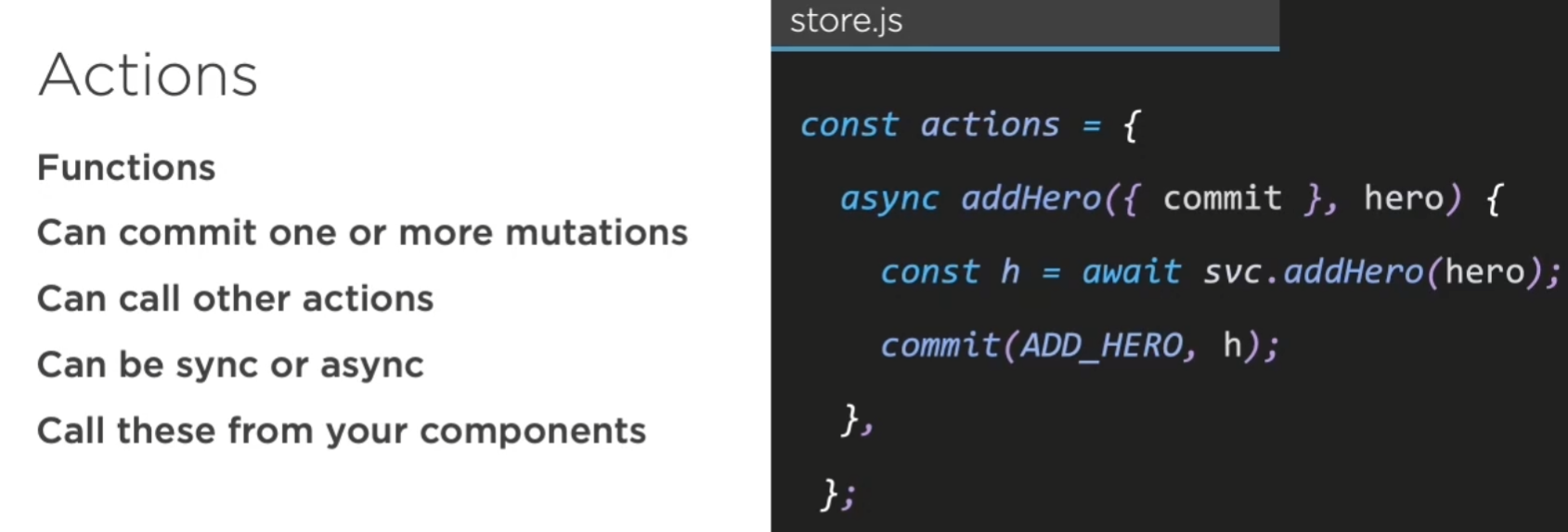
* **Once we have the state we have to mutate the state to add new things to state! The only way state can be changed is through mutation**



**Cf: es6 method naming shortcut: (上方寫法)**



**Actions**



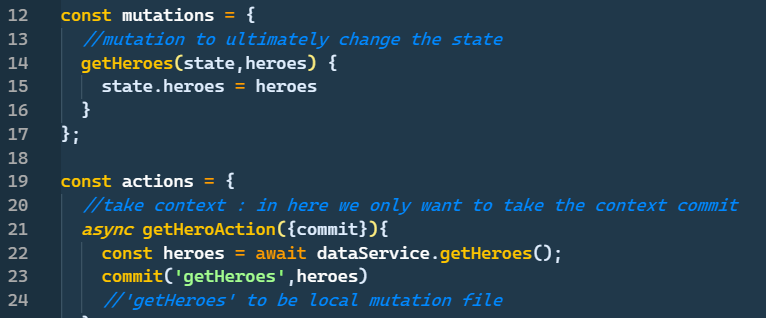


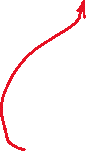
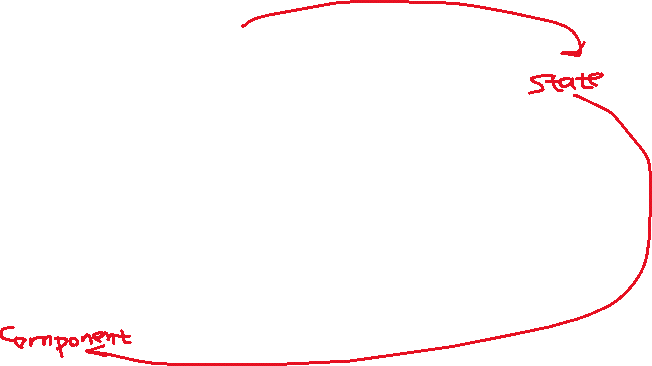
**Action and Mutation demo: (基本款)**

**(Index.js)**

**Don’t forget import your service**

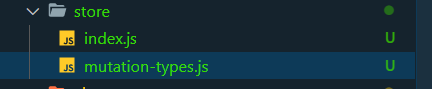




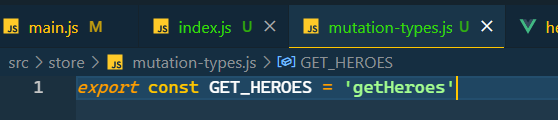


**進階整理，不希望在這邊使用string call mutations，改用 mutation-type.js**

**>> 在剛剛資料夾中新建文件**

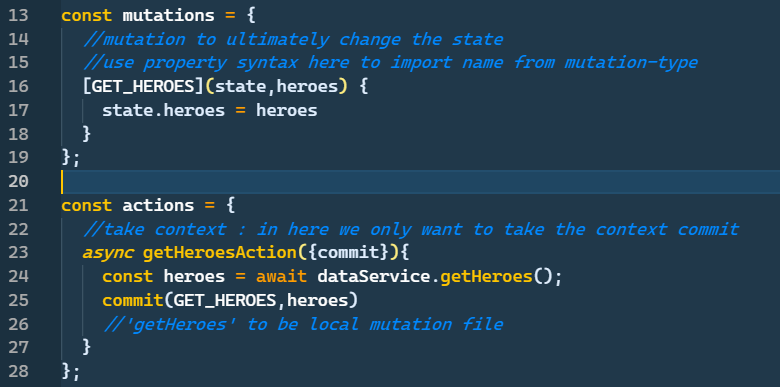


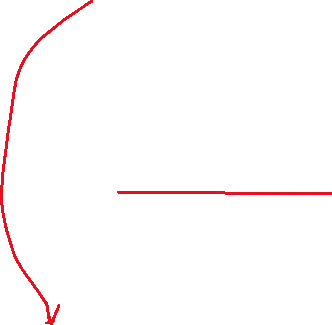
**Mutation-types.js**



**>> 避免string 可能寫錯字的問題**



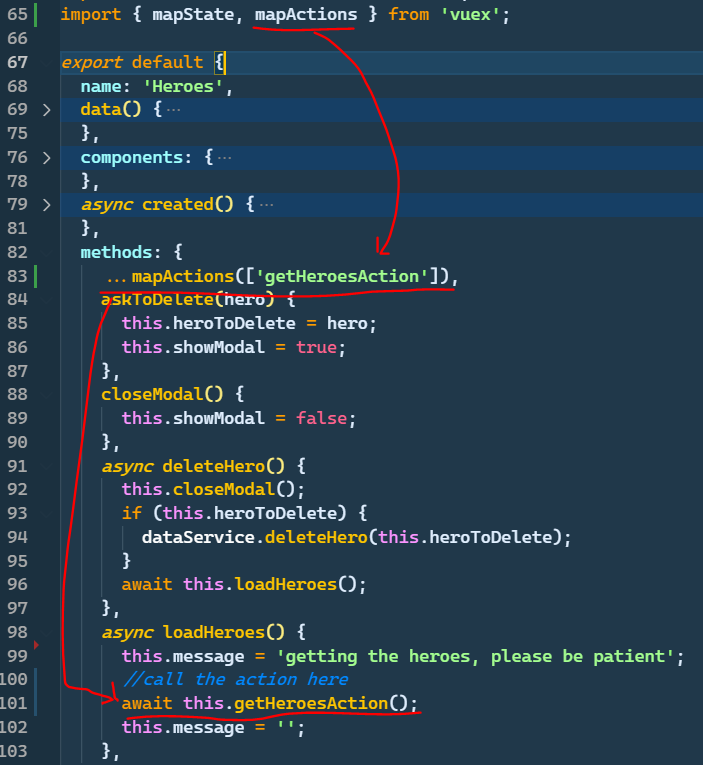




**Property 呼叫方式，從import的模組中回傳string**

**>> 這邊就會回傳’getHeroes’**

* **去到componenet建立action起點 (dispatch) >> 把action 也map 進來**

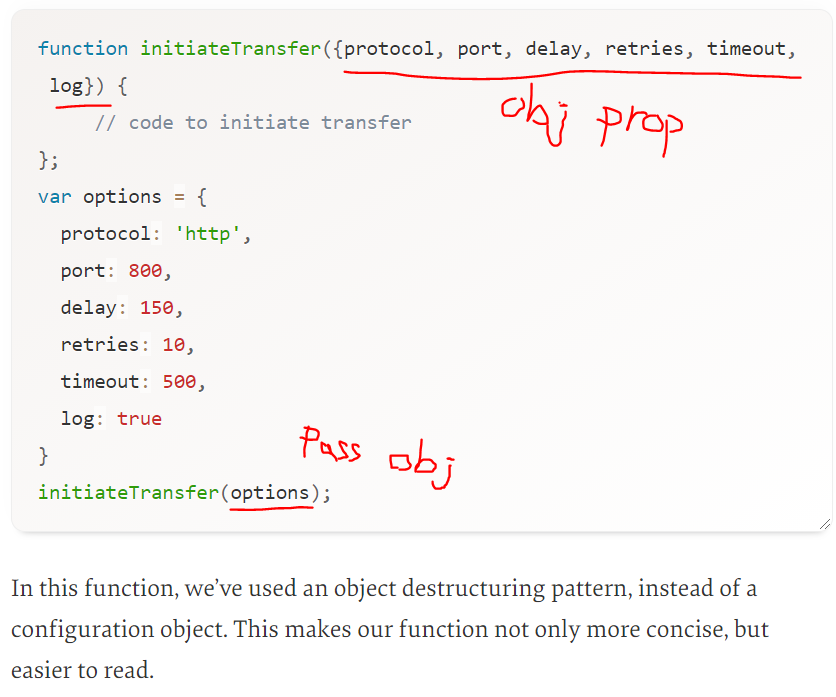


* **Get specific state using getters**

**Getters : make it easier to access parts of state**

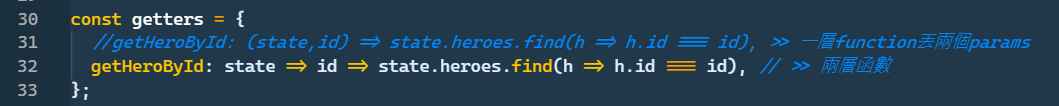


**\*\* cf: es6 用法 (obj prop傳進去作為params):**



**設置getters: (hero detail要按照id找到特定hero)**

**(index.js)**

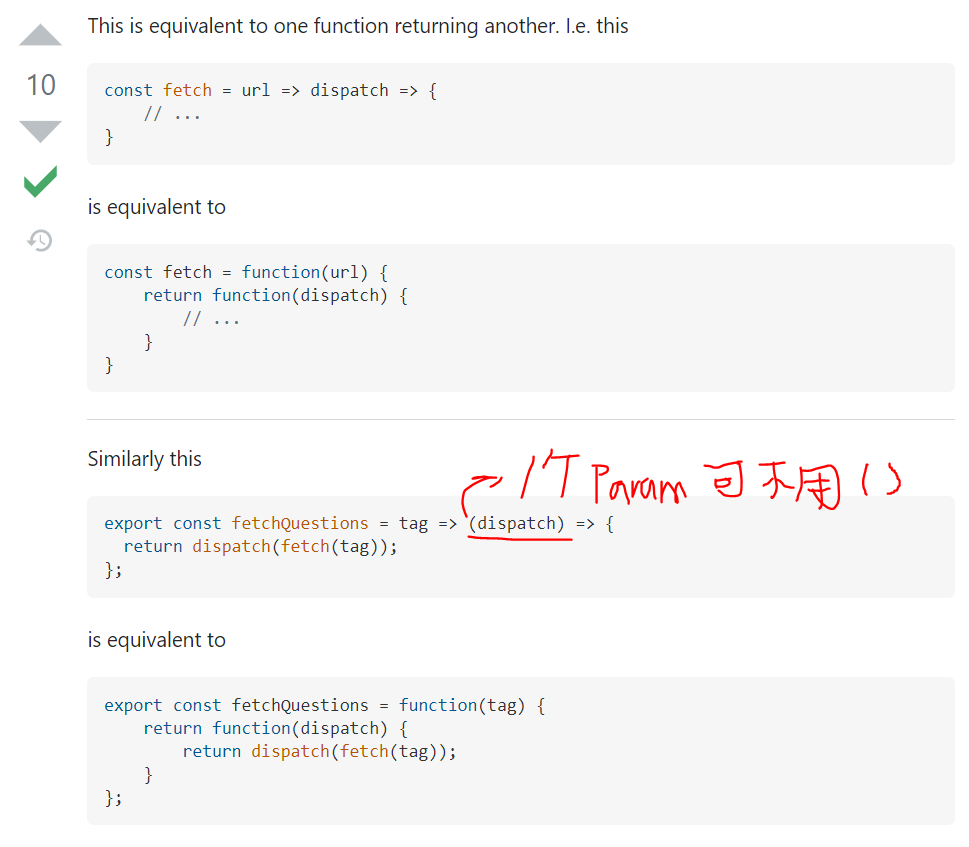




**>> 已經跟linq的where是一樣寫法了，覺得方便**



**Es6: arrow function chaining:**





**(hero-detail.js)**



* **Strict mode :**

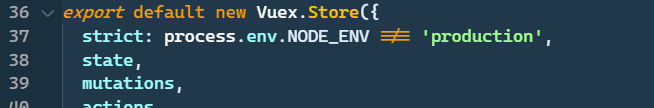
**\*\* strict should only be used in dev mode since run it is expensive!**

**\*\* strict mode will check and show error in browser,**

**tell you not to mutate the vuex store state outside the mutation handler!!**

**\*\*\*basically, we want to avoid users chage state directely in the UI, we want him to ge through the vuex circle**

**(index.js) strict mode**



**So we make a copy using lodash’s cloneDeep function**

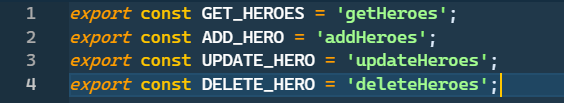
**\*\* lodash is a usful js extension library!!**

1. **Use npm I lodash**
2. **Import {cloneDeep} from ‘loadash’**

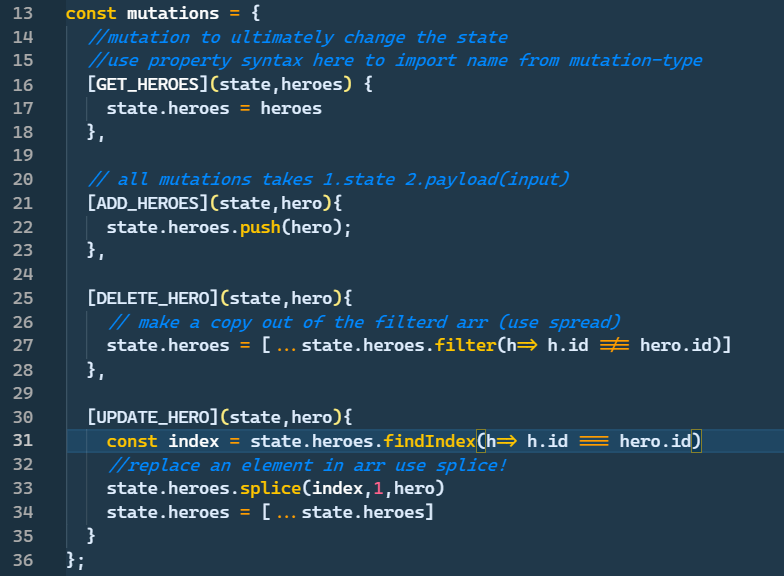


* **Mutate state through mutation circle (add update delete)**

**>> 1. Modify mutation-types.js**



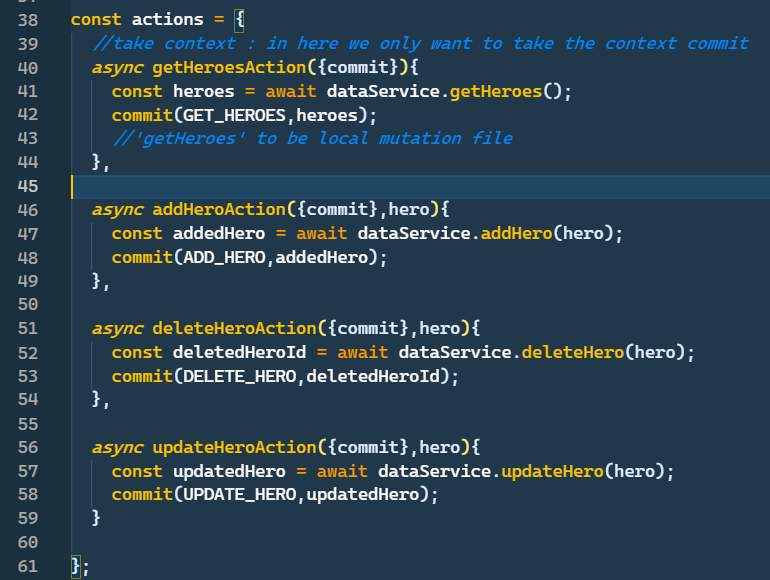
1. **Mutataion setup:**



1. **Action sertup (for component has a entry to call mutation)**

**>> Action first talk to the backend, then send request to mutation**

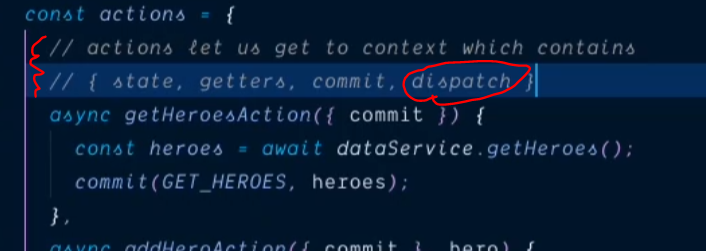
**\*\* also can use snipper ‘vmutataion’ and ‘vaction’ to get the template**



**\*補充:**

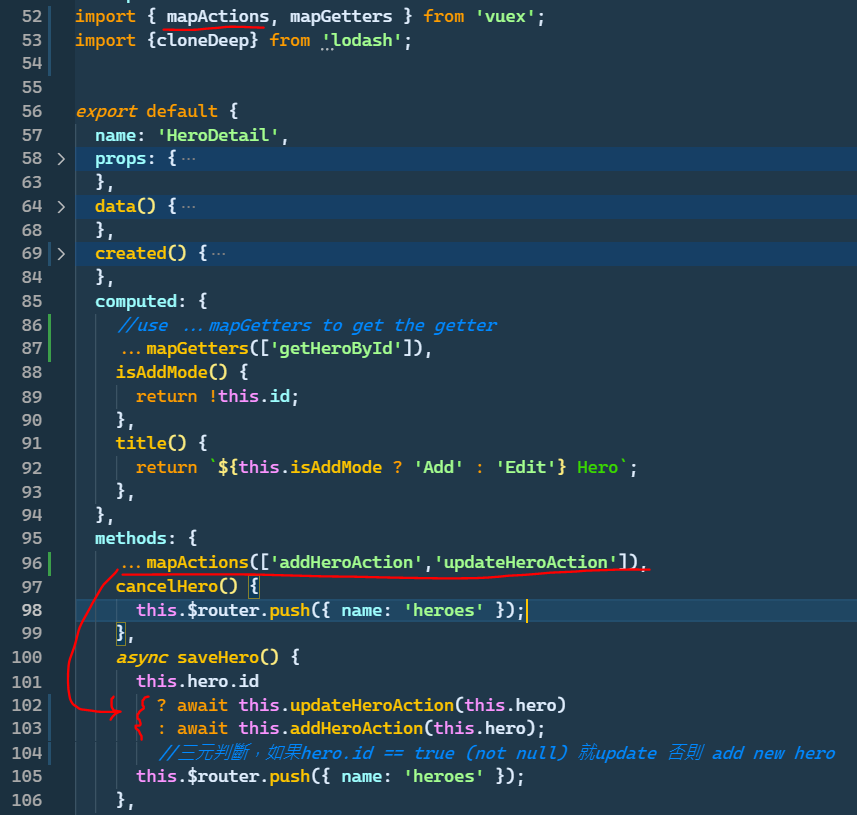
**1. 一個action裡面可以叫很多muitation，不只一個**

**2. 這邊都是叫commit出來，但其實還可以叫其他 context**

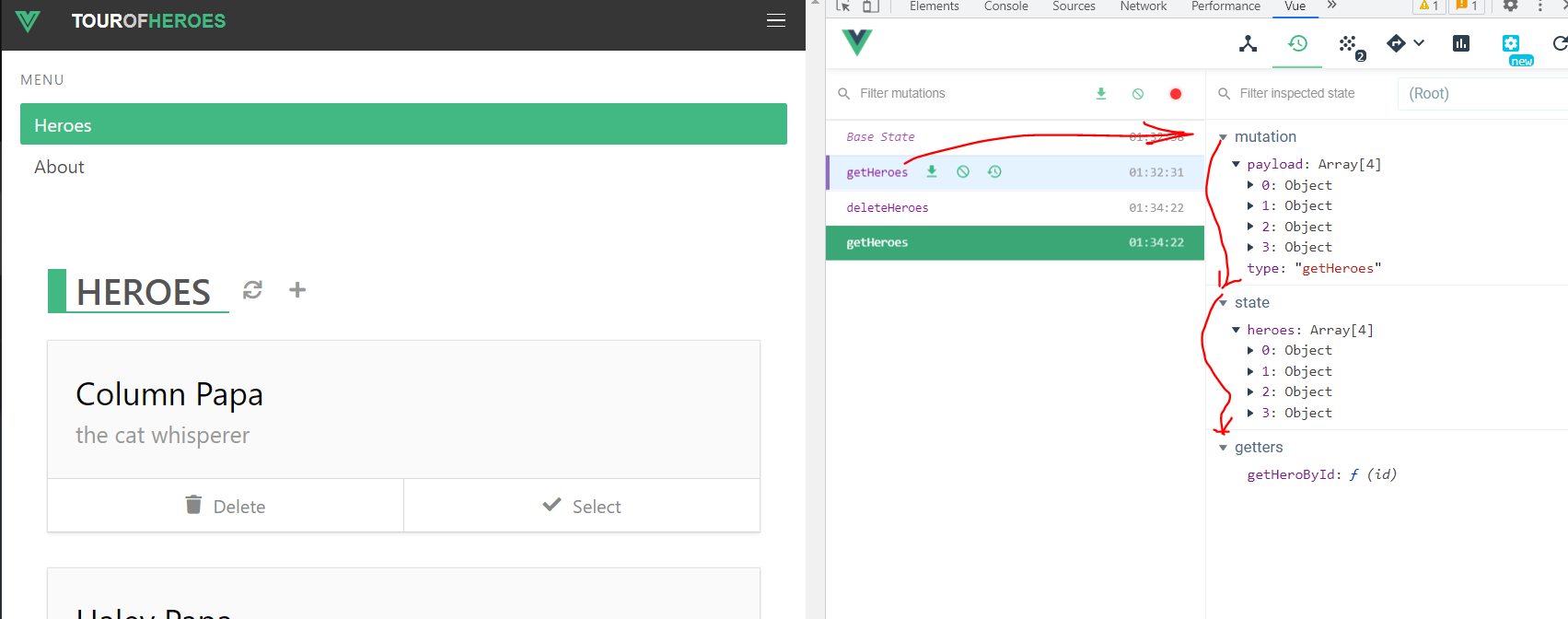


**>> dispatch 讓你可以在一個action中叫另一個action，例如在叫addHeroe 之後再叫 getHeroes**

1. **Dispatch action from component:**



**Now we can see the mutation handler in browser!**



**Recap:**

