



[Course](#) > [Redux](#) > [Store](#) > Store

Audit Access Expires Jun 20, 2020

You lose all access to this course, including your progress, on Jun 20, 2020.

Store

The screenshot shows a video player interface. The main content is a CodeSandbox editor window. The editor has a dark theme and shows a file named `index.js`. The code is as follows:

```
39 var store = createStore(reducer);
40
41
42
43 const unsub = store.subscribe(()=>{
44   console.log(store.getState())
45 })
46
47 store.dispatch(addItem('apple'))
48 store.dispatch(addItem('banana'))
49 store.dispatch(addItem('carrot'))
50
```

Below the code editor is a console panel with three tabs: Console, Problems, and Tests. The Console tab is active and shows three log messages:

- `["apple"]`
- `["apple", "banana"]`
- `["apple", "banana", "carrot"]`

On the right side of the video player, there is a white modal overlay with the text "Hello Coc" and "Start editing". The video player controls at the bottom show a progress bar at 2:15 / 2:39, a play button, a 1.50x speed setting, and icons for volume, full screen, and subtitles.

Video

[Download video file](#)

Transcripts

[Download SubRip \(.srt\) file](#)

[Download Text \(.txt\) file](#)

Store

Store

The Store is the object that holds all of your application state. It is important to only have a single store.

Creating a Store

You can create a store using the `createStore(reducer, [preloadedState], [enhancer])` method from the Redux library. The `createStore` takes the following arguments:

1. `reducer` - reducer that describes how dispatched actions should update the state
2. `preloadedState` - optional initial state
3. `enhancer` - optional middleware

You can import `createStore` by doing the following:

```
import {createStore} from 'redux'
```

You can create a store by calling the `createStore` method with a reducer:

```
const reducer = (state, action) => {  
  //return updated state  
}  
  
const store = createStore(reducer)
```

Getting state from a store

You can access the state from the store using the `store.getState()` method:

```
var state = store.getState()  
console.log(state)  
// {...}
```

Dispatching actions to update the store

You can dispatch actions using the `store.dispatch(action)` method:

```
store.dispatch(addItem('apple')) //'apple' is now added to the iter
```

try running this code to add a few items and delete a few items and log the store!

Subscribing to store updates

The `store.subscribe()` method is used to trigger a function whenever the store updates.

Here is an example to log the store everytime it updates:

```
//trigger a console log every time the store updates
const unsubscribe = store.subscribe(() => console.log(store.getState))

//stop subscribing to store updates
unsubscribe()
```

To unsubscribe, call the method that is returned by `store.subscribe()`:

Try out this live example on CodeSandbox: <https://codesandbox.io/s/m4j9656xwx>