Create a Counter with Redux-ToolKit

We will be creating a simple counter with Redux-Toolkit,

Step 1

So for this we will install redux toolkit dependency in our project.

```
npm install @reduxjs/toolkit
```

Step 2

Now we will create a store for our project.

So we will create a file name : store.js

We will import configureStore from redux toolkit

```
import { configureStore } from '@reduxjs/toolkit'
```

Step 3

We will be creating an empty redux store.

```
import { configureStore } from '@reduxjs/toolkit'

export const store = configureStore({
reducer: { },
})
```

Step 4

So now once the store is created, we will make it available to our react component with index.js in our src folder by using react-redux <Provider /> around our <App /> component.

We will import redux store we just created and put <Provider > around our < App /> and pass the store as props.

(in index.js)

```
import { store } from './Store'
import { Provider } from 'react-redux'

ReactDOM.render(
<Provider store={store}>
<App />
</Provider>,
```

Step 5

Now we will be creating our Redux State Slice.

We will create new file name it as CounterSlice.js

In this file, we will import the createSlice from ReduxToolkit.

```
import { createSlice } from '@reduxjs/toolkit'
```

Creating a slice requires a string name to identify the slice, an initial state value and one or more reducer function to define how state can be updated.

```
const initialState = {
value: 0,
}

export const CounterSlice = createSlice({
name: 'counter',
initialState,
reducers: {
increment: (state) => {
state.value += 1
},
decrement: (state) => {
state.value -= 1
},
},
})
```

Once the slice is created , we can export the generated redux actions and reducer function for the whole slice.

```
export const { increment, decrement } = CounterSlice.actions

export default CounterSlice.reducer
```

It is important to note, redux requires that we write all the updates immutably by making copies of data and updating the copies. However, Redux Toolkit's createSlice allows us to write mutating update logics that becomes correct immutable updates.

Step 6

Now we will import the reducer function from the counter slice and add it to our store.
By defining field inside the reducer parameter, we are telling the store to use this slice reducer function to handle all updates to that state.

(in store.js)

```
import { configureStore } from '@reduxjs/toolkit'
import CounterSlice from './CounterSlice'

export const store = configureStore({
reducer: {
counter: CounterSlice,
},
})
```

Step 7

Now we can use react hooks to let react component interact with the redux store. We will read data from the store with useSelector() and dispatch action using useDispatch()

So we will create a file Counter.js

```
import { useSelector, useDispatch } from 'react-redux'
import { decrement, increment } from './CounterSlice'
```

Now with useSelector we will access to value of our state

```
const count = useSelector((state) => state.counter.value)
```

```
<div className="number">
<span>{count}</span>
</div>
```

Now withDispatch() we will dispatch the action.

```
const dispatch = useDispatch()

<button
class="button"
onClick={() => dispatch(increment())} >
Increment
</button>
```

we will import our Counter.js to our App.js and hence we successfully created counter app with react toolkit.

(in Counter.js)

```
import React from 'react'
import { useSelector, useDispatch } from 'react-redux'
import { decrement, increment } from './CounterSlice'

export default function Counter() {
const count = useSelector((state) => state.counter.value)
const dispatch = useDispatch()

return (
<div>
<div className="container">
<div className="number">
<span>{count}</span>
</div>
<div className="allbuttons">
<button
aria-label="Increment value" class="button"
onClick={() => dispatch(increment())}
>
Increment
</button>
<button
aria-label="Decrement value" class="button"
onClick={() => dispatch(decrement())}
>
Decrement
</button>
</div>
</div>
</div>
)
}
```