

[< What About Coding />](#)[Home](#) > [React Js](#) > **Build an application using createAsyncThunk and Redux Toolkit**

React Js

# Build an application using createAsyncThunk and Redux Toolkit

By Himanshu Shekhar    March 24, 2023

[Write a Comment](#)

Calling an API is not so straightforward while using the redux toolkit, and that's why redux has a middleware name "createAsyncThunk()" which provided us with all the superpowers needed for handling API and response.

In this tutorial, we are going to build a complete CRUD functionality, but using [mockAPI](#), so that you get a complete end-to-end knowledge of how to deal with API while performing Create, Read, Update, and Delete Operations.

## Table of Contents

[Pre-requisites](#)[Diagram](#)[Adding redux to the project](#)[Let's create Slice now](#)[Adding Frontend Now](#)[Conclusion](#)

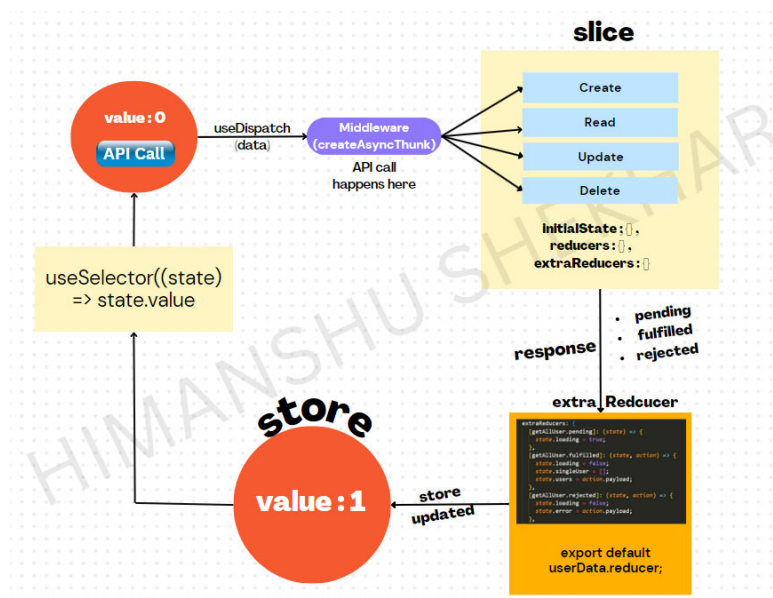
## Pre-requisites

You must have an idea about redux beginner level, if not, learn from [here](#)

Basic knowlddge of HTML, CSS , Java Script ,  
React and Redux

## Diagram

Have a closer look on the diagram to understand the redux and asyncThunk flow



**Lets understand the flow -**

**Step 1** – An action is performed on the front-end (let's say a button click)

**Step 2** – That action is dispatched (by using `useDispatch` hook) to the middleware `"createAsyncThunk()"` written inside slice file

**Step 3** – Inside `createAsyncThunk()` an API is made, using `fetch` or `Axios`, depending upon the method ie. GET, POST, DELETE, OR PUT

**Step 4** – Now the response from the above is handled by the **extraReducer**, written inside **createSlice** method

**Step 5** – And finally the state (or the global **store**) is updated

**Step 6** – The store data is displayed back to frontend using **useSelector** hook

## Adding redux to the project

Well, I have explained this in-depth, in my preview article, do check it out. [Click here](#)

### Install Package

```
npm install --save react-redux @reduxjs/toolkit
```

### src/app/store.js

```
1. import { configureStore } from "@reduxjs/to
2. import gitUser from "../features/gitUserSlice"
3.
4. export const store = configureStore({
5.   reducer: {
6.     app: gitUser,
7.   },
8. });
```

Don't forget to provide the store globally.

### src/index.js

```
1. import React from "react";
2. import ReactDOM from "react-dom/client";
3. import "./index.css";
4. import App from "./App";
5. import { store } from "./app/store";
6. import { Provider } from "react-redux";
```

```

7.
8. const root = ReactDOM.createRoot(document)
9. root.render(
10.   <React.StrictMode>
11.     <Provider store={store}>
12.       <App />
13.     </Provider>
14.   </React.StrictMode>
15. );

```

## Let's create Slice now

Slice is the only file that will contain all the things needed to perform our operation

- initialState
- reducers
- extraReducers

If you want a quick overview of how API call work using createAsyncThunk and extraReducers, do watch the below video



src/features/getUserSlice.js

```

1. import { createAsyncThunk, createSlice } from
2.
3. //Get all user action
4. export const getAllUser = createAsyncThunk(
5.   "getUsers",
6.   async (args, { rejectWithValue }) => {
7.     try {

```

```
8.     const response = await fetch(
9.       "https://629f5d82461f8173e4e7db69.mo
10.    );
11.    const result = await response.json();
12.    return result;
13.  } catch (err) {
14.    return rejectWithValue("Opps found an e
15.  }
16. }
17. );
18.
19. //get single user
20. export const getSingleUser = createAsyncTh
21.   "getSingleUser",
22.   async (id, { rejectWithValue }) => {
23.     const response = await fetch(
24.       `https://629f5d82461f8173e4e7db69.moc
25.     );
26.
27.     try {
28.       const result = await response.json();
29.       return result;
30.     } catch (err) {
31.       return rejectWithValue(err.message);
32.     }
33.   }
34. );
35. //create action
36. export const createUser = createAsyncThur
37.   "createUser",
38.   async (data, { rejectWithValue }) => {
39.     const response = await fetch(
40.       "https://629f5d82461f8173e4e7db69.moc
41.     {
42.       method: "POST",
43.       headers: {
44.         "Content-Type": "application/json",
45.       },
46.       body: JSON.stringify(data),
47.     }
48.   );
49.   const result = await response.json();
50.   return result;
51. }
52. );
53.
54. //delete single user
55. export const deleteUser = createAsyncThun
```

```

56.     "deleteUser",
57.     async (id, { rejectWithValue }) => {
58.         try {
59.             const response = await fetch(
60.                 `https://629f5d82461f8173e4e7db69.mo
61.                 {
62.                     method: "DELETE",
63.                 }
64.             );
65.             const result = await response.json();
66.             return result;
67.         } catch (err) {
68.             console.log(err);
69.             return rejectWithValue(err.response.data);
70.         }
71.     }
72. );
73.
74. //update user
75. export const updateUser = createAsyncThunk(
76.     "updateUser",
77.     async ({ id, name, email, age, gender }, { rejectWithValue }) => {
78.
79.         try {
80.             const response = await fetch(
81.                 `https://629f5d82461f8173e4e7db69.mo
82.                 {
83.                     method: "PUT",
84.                     headers: {
85.                         "Content-Type": "application/json",
86.                     },
87.                     body: JSON.stringify({ name, email, age, gender }),
88.                 }
89.             );
90.             const result = await response.json();
91.             return result;
92.         } catch (err) {
93.             return rejectWithValue(err);
94.         }
95.     }
96. );
97.
98. export const gitUser = createSlice({
99.     name: "gitUser",
100.    initialState: {
101.        users: [],
102.        loading: false,
103.        error: null,

```

```
104.   searchData: [],
105. },
106. reducers: {
107.   searchUser: (state, action) => {
108.     state.searchData = action.payload;
109.   },
110. },
111. extraReducers: {
112.   [getAllUser.pending]: (state) => {
113.     state.loading = true;
114.   },
115.   [getAllUser.fulfilled]: (state, action) => {
116.     state.loading = false;
117.     state.singleUser = [];
118.     state.users = action.payload;
119.   },
120.   [getAllUser.rejected]: (state, action) => {
121.     state.loading = false;
122.     state.error = action.payload;
123.   },
124.   [createUser.fulfilled]: (state, action) => {
125.     state.loading = false;
126.     state.users.push(action.payload);
127.   },
128.   [deleteUser.pending]: (state) => {
129.     state.loading = true;
130.   },
131.   [deleteUser.fulfilled]: (state, action) => {
132.     state.loading = false;
133.     const { id } = action.payload;
134.     if (id) {
135.       state.users = state.users.filter((post) =>
136.         post.id !== id
137.       );
138.     }
139.   },
140.   [deleteUser.rejected]: (state, action) => {
141.     state.loading = false;
142.     state.error = action.payload.message;
143.   },
144.   [getSingleUser.pending]: (state) => {
145.     state.loading = true;
146.   },
147.   [getSingleUser.fulfilled]: (state, action) => {
148.     state.loading = false;
149.     state.singleUser = [action.payload];
150.   },
151.   [getSingleUser.rejected]: (state, action) => {
152.     state.loading = false;
153.     state.error = action.payload.message;
```

```

152.     },
153.     [updateUser.pending]: (state) => {
154.       state.loading = true;
155.     },
156.     [updateUser.fulfilled]: (state, action) => {
157.       console.log("updated user fulfilled", action);
158.       state.loading = false;
159.       state.users = state.users.map((ele) =>
160.         ele.id === action.payload.id ? action.payload : ele
161.       );
162.     },
163.     [updateUser.rejected]: (state, action) => {
164.       state.loading = false;
165.       state.error = action.payload.message;
166.     },
167.   },
168. });
169.
170. export const { searchUser } = gitUser.action
171. export default gitUser.reducer;

```

returning an response, error handling, updating the store based on the response, Delete)

- returning an response
- error handling
- updating the store based on the response

exporting the reducer to the store

## Adding Frontend Now

### App.js

```

1. import { BrowserRouter, Route, Routes } from 'react-router-dom';
2. import './App.css';
3. import Create from './components/Create';
4. import Read from './components/Read';
5. import Edit from './components/Edit';
6. import Navbar from './components/Navbar';
7.
8. function App() {

```



```

9.   return (
10.     <div className="App">
11.       <BrowserRouter>
12.         <Navbar />
13.         <Routes>
14.           <Route exact path="/" element={<Rea
15.           <Route exact path="/create" element=
16.           <Route exact path="/edit/:id" element=
17.         </Routes>
18.       </BrowserRouter>
19.     </div>
20.   );
21. }
22.
23. export default App;

```



### Navbar.js

```

1.  import React, { useState } from "react";
2.  import { useDispatch, useSelector } from "re
3.  import { Link } from "react-router-dom";
4.  import { searchUser } from "../features/gitUs
5.
6.  const Navbar = () => {
7.    const [searchData, setSearchData] = useS
8.    const totalCount = useSelector((state) =>
9.
10.   const dispatch = useDispatch();
11.
12.   dispatch(searchUser(searchData));
13.
14.   return (
15.     <>
16.       <nav class="navbar navbar-expand-lg r
17.         <div className="container-fluid">
18.           <div class="collapse navbar-collapse"
19.             <ul class="navbar-nav">
20.               <li class="nav-item">
21.                 <Link to="/create" class="nav-link">
22.                   Create Post
23.                 </Link>
24.               </li>
25.               <li class="nav-item">
26.                 <Link to="/" class="nav-link">
27.                   All Post ({totalCount.length})
28.                 </Link>
29.               </li>

```

```

30.         </ul>
31.     </div>
32.
33.     <input
34.         class="form-control "
35.         type="search"
36.         placeholder="Search"
37.         value={searchData}
38.         onChange={(e) =>
39.             dispatch(searchUser(setSearchData
40.         )
41.     ></input>
42. </div>
43. </nav>
44. </>
45. );
46. };
47.
48. export default Navbar;

```



### Create.js

```

1.  import React, { useState } from "react";
2.  import { useDispatch } from "react-redux";
3.  import { useNavigate } from "react-router-c
4.  import { createUser } from "../features/gitUs
5.
6.  const Create = () => {
7.      const [data, setData] = useState({});
8.      const dispatch = useDispatch();
9.      const navigate = useNavigate();
10.
11.     const updateData = (e) => {
12.         setData({
13.             ...data,
14.             [e.target.name]: e.target.value,
15.         });
16.     };
17.
18.     const handleSubmit = (e) => {
19.         e.preventDefault();
20.         console.log("user data...", data);
21.         dispatch(createUser(data));
22.         navigate("/");
23.     };
24.
25.     return (

```

```
26.   <div>
27.     <h2>Enter the data</h2>
28.
29.     <form onSubmit={handleSubmit}>
30.       <div>
31.         <input
32.           type="text"
33.           name="name"
34.           placeholder="enter name"
35.           onChange={updateData}
36.         />
37.       </div>
38.       <div>
39.         <input
40.           type="email"
41.           name="email"
42.           placeholder="enter email"
43.           onChange={updateData}
44.         />
45.       </div>
46.       <div>
47.         <input
48.           type="number"
49.           name="age"
50.           placeholder="enter age"
51.           onChange={updateData}
52.         />
53.       </div>
54.       <div>
55.         <input
56.           type="radio"
57.           name="gender"
58.           // checked={updateData.gender ==:
59.           value="Male"
60.           onChange={updateData}
61.         />
62.         <label>Male</label>
63.         <input
64.           type="radio"
65.           name="gender"
66.           // checked={this.state.selectedOptio
67.           value="Female"
68.           onChange={updateData}
69.         />
70.         <label>Female</label>
71.       </div>
72.       <div>
73.         <button type="submit">Submit</butto
```

```

74.     </div>
75.   </form>
76. </div>
77. );
78. };
79.
80. export default Create;

```



```

1. import React, { useEffect, useState } from "react";
2. import { useSelector, useDispatch } from "redux";
3. import { getAllUser, deleteUser } from "../features/user";
4. import { Link } from "react-router-dom";
5. import UserModal from "../UserModal";
6.
7. const Read = () => {
8.   const dispatch = useDispatch();
9.   const [show, setShow] = useState(false);
10.  const handleClose = () => setShow(false);
11.  const handleShow = () => setShow(true);
12.  const [radioCheck, setRadioCheck] = useState(false);
13.
14.  const [id, setId] = useState();
15.
16.  const data = useSelector((state) => {
17.    return state.app;
18.  });
19.
20.  console.log("radio...", radioCheck);
21.
22.  useEffect(() => {
23.    dispatch(getAllUser());
24.    // eslint-disable-next-line react-hooks/exhaustive-deps
25.  }, []);
26.
27.  if (data.loading) {
28.    return <h2>Loading...</h2>;
29.  }
30.
31.  if (data.error !== null) {
32.    return <h3>{data.error}</h3>;
33.  }
34.
35.  console.log("final data to loop", data);
36.
37.  return (

```

```

38.   <div>
39.     <UserModal
40.       handleShow={handleShow}
41.       handleClose={handleClose}
42.       show={show}
43.       setShow={setShow}
44.       id={id}
45.     />
46.   <div className="d-flex justify-content-
47.     <h1>All Users</h1>
48.     <div className="d-flex gap-2">
49.       <div>
50.         <input
51.           class="form-check-input"
52.           type="radio"
53.           name="gender"
54.           checked={radioCheck === ""}
55.           onChange={(e) => setRadioCheck('
56.         />
57.         <label class="form-check-label">All<
58.       </div>
59.       <div class="form-check">
60.         <input
61.           class="form-check-input"
62.           type="radio"
63.           name="gender"
64.           value="Male"
65.           checked={radioCheck === "Male"}
66.           onChange={(e) => setRadioCheck('
67.         />
68.         <label class="form-check-label">Mal
69.       </div>
70.       <div>
71.         <input
72.           class="form-check-input"
73.           type="radio"
74.           name="gender"
75.           value="Female"
76.           checked={radioCheck === "Female"
77.           onChange={(e) => setRadioCheck('
78.         />
79.         <label class="form-check-label">Fem
80.       </div>
81.     </div>
82.   </div>
83. </div>
84.
85.   {data?.users

```

```

86.     .filter((item) => {
87.       if (data.searchData.length === 0) {
88.         return item;
89.       } else {
90.         return item.name
91.           .toLowerCase()
92.           .includes(data.searchData.toLowerCase())
93.       }
94.     })
95.     .filter((item) => {
96.       if (radioCheck === "") {
97.         return item;
98.       } else if (radioCheck === "Male") {
99.         return item.gender === radioCheck;
100.      } else if (radioCheck === "Female") {
101.        return item.gender === radioCheck;
102.      }
103.    })
104.    .map((ele) => (
105.      <div key={ele.id} className="card w-100">
106.        <div className="card-body">
107.          <h5 className="card-title">{ele.name}</h5>
108.          <h6 className="card-subtitle mb-2 text-muted">{ele.email}</h6>
109.          <h6 className="card-subtitle mb-2 text-muted">{ele.phone}</h6>
110.
111.          <button
112.            type="button"
113.            class="btn btn-primary"
114.            //onClick={() => setId(ele.id) && handleDelete(ele.id)}
115.            onClick={() => [setId(ele.id), handleDelete(ele.id)]}
116.          >
117.            View
118.          </button>
119.
120.          <Link
121.            onClick={() => dispatch(deleteUser(ele.id))}
122.            className="card-link mx-2"
123.          >
124.            Delete
125.          </Link>
126.          <Link to={`/edit/${ele.id}`}>
127.            <span className="card-link mx-2">Edit</span>
128.          </Link>
129.        </div>
130.      </div>
131.    ))}
132.  </div>
133. );

```

```

134.   };
135.
136.   export default Read;

```

```

1.   import React, { useEffect, useState } from "react";
2.   import { useDispatch, useSelector } from "redux";
3.   import { useParams, useNavigate } from "react-router-dom";
4.   import { updateUser } from "../features/gitUser";
5.
6.   const Edit = () => {
7.     const dispatch = useDispatch();
8.     const { id } = useParams();
9.     const navigate = useNavigate();
10.    const initialState = {
11.      name: "",
12.      email: "",
13.      age: "",
14.      gender: "",
15.    };
16.    const [updatedData, setUpdatedData] = useState(initialState);
17.
18.    //get all data
19.    const { users, loading } = useSelector((state) => state.users);
20.
21.    useEffect(() => {
22.      //retrieving single data from user list
23.      if (id) {
24.        const singleData = users.find((user) => user.id === id);
25.        console.log("singledata preload on edit", singleData);
26.        setUpdatedData({ ...singleData });
27.      }
28.    }, [id]);
29.
30.    //updating state as use changes input field
31.    const newData = (e) => {
32.      setUpdatedData({ ...updatedData, [e.target.name]: e.target.value });
33.    };
34.
35.    const handleSubmit = (e) => {
36.      e.preventDefault();
37.      console.log("update data..", updatedData);
38.      dispatch(updateUser(updatedData));
39.      setUpdatedData(initialState);
40.      navigate("/");
41.    };

```

```
42.
43.   if (loading) {
44.     return <h2>Loading.</h2>;
45.   }
46.
47.   return (
48.     <div>
49.       <h2>Update the data</h2>
50.
51.       {updatedData && (
52.         <form onSubmit={handleSubmit}>
53.           <div>
54.             <input
55.               type="text"
56.               name="name"
57.               placeholder="enter name"
58.               value={updatedData.name}
59.               onChange={newData}
60.             />
61.           </div>
62.           <div>
63.             <input
64.               type="email"
65.               name="email"
66.               placeholder="enter email"
67.               value={updatedData.email}
68.               onChange={newData}
69.             />
70.           </div>
71.           <div>
72.             <input
73.               type="number"
74.               name="age"
75.               placeholder="enter age"
76.               value={updatedData.age}
77.               onChange={newData}
78.             />
79.           </div>
80.           <div>
81.             <input
82.               type="radio"
83.               name="gender"
84.               checked={updatedData.gender ===
85.               value="Male"
86.               onChange={newData}
87.             />
88.             <label>Male</label>
89.             <input
```



```
90.         type="radio"
91.         name="gender"
92.         checked={updatedData.gender === "Male"}
93.         value="Female"
94.         onChange={newData}
95.       />
96.       <label>Female</label>
97.     </div>
98.   <div>
99.     <button type="submit">Submit</button>
100.   </div>
101. </form>
102.   )}
103. </div>
104. );
105. };
106.
107. export default Edit;
```



While working on any real-life project, you have to deal with API and if you are working on redux, the middleware will definitely come into the picture.

The combination of createAsyncThunk and extraReducers makes API handling easy and simple.

Frankly, it's more of a process, than the logic, so once you are comfortable with the flow, it will be a piece of cake for you.

I have tried to break it down and explain you as simply as I can. Don't forget to watch my tutorial on the same.



**Help Others**

## DOM and Window Object in JavaScript | JavaScript Interview series

[Previous](#)



Himanshu Shekhar

Hey. I am a software engineer with 3+ years.

Along with my job, I create content regarding coding and software engineering in general. I would request you to follow my blog and my video, which will help you in your software engineering journey. Follow

me on [Youtube](#) →

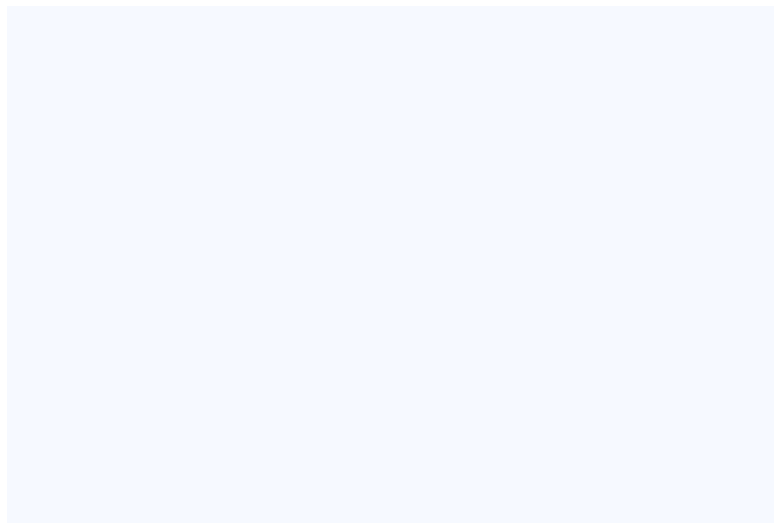
[View All Articles](#)

## Related Posts



## useRef In React Js | React Js for Beginners

November 23, 2022   React Hooks   React Js



## Revising React Basics (Fast)

June 1, 2022   React Js

## Redux Toolkit Tutorial

December 9, 2022   React Js

---

### Leave a Reply

Your email address will not be published. Required fields are marked \*

**Comment \***

**Name \***

**Email \*****Website**

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

---

Search this blog...

### Recent Posts

---

Build an application using createAsyncThunk and Redux Toolkit

DOM and Window Object in JavaScript | JavaScript Interview series

Build a simple MERN Stack Application | CRUD using "this" keyword | JavaScript Interview Series

Scope , Scope chain and Lexical Scope | JavaScript Interview Series

## Category

---

Frontend Interview Question	(5)
Java Script	(10)
JavaScript Interview Series	(5)
Jquery	(4)
Node JS	(3)
React Basics	(2)
React Hooks	(2)
React Js	(11)
Sales and Offers	(2)

## Help others

---

[Privacy Poilicy](#)[Sitemap](#)[Terms and Conditions](#)

Copyright © 2023 What About Coding