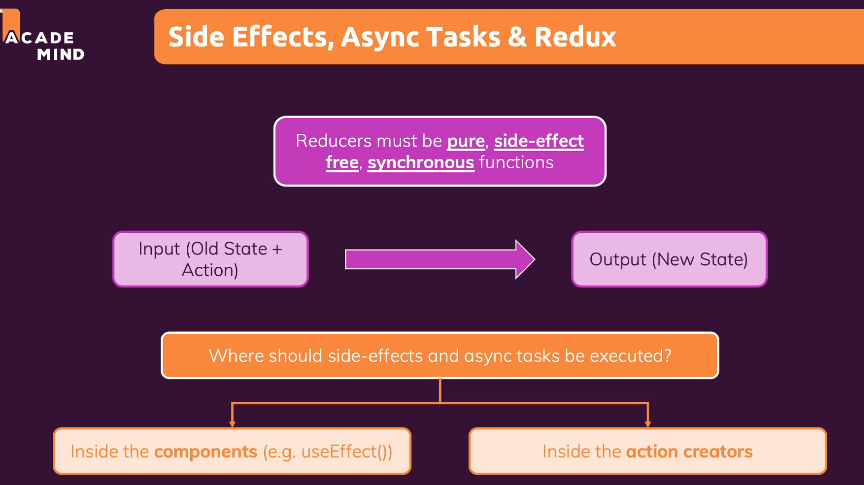
Section 19: Advanced Redux



Source to start:

251. Refresher/Practice: Part 1/2

Step 0 clone source code

<https://github.com/academind/react-complete-guide-code/tree/19-advanced-redux>

Step 1 run command npm install @reduxjs/toolkit

Step 2 run command npm install react-redux

Step 3 create new folder store in folder src

Step 4 create ui-slice.js in folder store

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

const uiSlice = createSlice({

  name: 'ui',

  initialState: { cartIsVisible: false },

  reducers: {

    toggle(state) {

      state.cartIsVisible = !state.cartIsVisible;

    }

  }

});

export const uiActions = uiSlice.actions;

export default uiSlice;

Step 5 create index.js in folder store

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

import uiSlice from './ui-slice';

import cartSlice from './cart-slice';

const store = configureStore({

  reducer: { ui: uiSlice.reducer, cart: cartSlice.reducer },

});

export default store;

Step 6 in src.index.js

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import store from './store/index';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <Provider store={store}>

    <App />

  </Provider>

);

Step 7 in CartButton.js

import { useDispatch, useSelector } from 'react-redux';

import { uiActions } from '../../store/ui-slice';

import classes from './CartButton.module.css';

const CartButton = (props) => {

  const dispatch = useDispatch();

  const cartQuantity = useSelector((state) => state.cart.totalQuantity);

  const toggleCartHandler = () => {

    dispatch(uiActions.toggle());

  };

  return (

    <button className={classes.button} onClick={toggleCartHandler}>

      <span>My Cart</span>

      <span className={classes.badge}>{cartQuantity}</span>

    </button>

  );

};

export default CartButton;

Step 8 in app.js

import { useSelector } from 'react-redux';

import Cart from './components/Cart/Cart';

import Layout from './components/Layout/Layout';

import Products from './components/Shop/Products';

function App() {

  const showCart = useSelector((state) => state.ui.cartIsVisible);

  return (

    <Layout>

      {showCart && <Cart />}

      <Products />

    </Layout>

  );

}

export default App;

Step 9 in cart-slice.js

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

const cartSlice = createSlice({

  name: 'cart',

  initialState: {

    items: [],

    totalQuantity: 0,

  },

  reducers: {

    addItemToCart(state, action) {

      const newItem = action.payload;

      const existingItem = state.items.find((item) => item.id === newItem.id);

      state.totalQuantity++;

      if (!existingItem) {

        state.items.push({

          id: newItem.id,

          price: newItem.price,

          quantity: 1,

          totalPrice: newItem.price,

          name: newItem.title

        });

      } else {

        existingItem.quantity++;

        existingItem.totalPrice = existingItem.totalPrice + newItem.price;

      }

    },

    removeItemFromCart(state, action) {

      const id = action.payload;

      const existingItem = state.items.find(item => item.id === id);

      state.totalQuantity--;

      if (existingItem.quantity === 1) {

        state.items = state.items.filter(item => item.id !== id);

      } else {

        existingItem.quantity--;

      }

    },

  },

});

export const cartActions = cartSlice.actions;

export default cartSlice;

step 10 in Shop/Products.js

import ProductItem from './ProductItem';

import classes from './Products.module.css';

const DUMMY\_PRODUCTS = [

  {

    id: 'p1',

    price: 6,

    title: 'My First Book',

    description: 'The first book I ever wrote',

  },

  {

    id: 'p2',

    price: 5,

    title: 'My Second Book',

    description: 'The second book I ever wrote',

  },

];

const Products = (props) => {

  return (

    <section className={classes.products}>

      <h2>Buy your favorite products</h2>

      <ul>

        {DUMMY\_PRODUCTS.map((product) => (

          <ProductItem

            key={product.id}

            id={product.id}

            title={product.title}

            price={product.price}

            description={product.description}

          />

        ))}

      </ul>

    </section>

  );

};

export default Products;

Step 11 in Shop/PriductItem.js

import { useDispatch } from 'react-redux';

import { cartActions } from '../../store/cart-slice';

import Card from '../UI/Card';

import classes from './ProductItem.module.css';

const ProductItem = (props) => {

  const dispatch = useDispatch();

  const { title, price, description, id } = props;

  const addToCartHandler = () => {

    dispatch(

      cartActions.addItemToCart({

        id,

        title,

        price,

      })

    );

  };

  return (

    <li className={classes.item}>

      <Card>

        <header>

          <h3>{title}</h3>

          <div className={classes.price}>${price.toFixed(2)}</div>

        </header>

        <p>{description}</p>

        <div className={classes.actions}>

          <button onClick={addToCartHandler}>Add to Cart</button>

        </div>

      </Card>

    </li>

  );

};

export default ProductItem;

Step 12 In Cart.js

import { useSelector } from 'react-redux';

import Card from '../UI/Card';

import classes from './Cart.module.css';

import CartItem from './CartItem';

const Cart = (props) => {

  const cartItems = useSelector((state) => state.cart.items);

  return (

    <Card className={classes.cart}>

      <h2>Your Shopping Cart</h2>

      <ul>

        {cartItems.map((item) => (

          <CartItem

            key={item.id}

            item={{

              id: item.id,

              title: item.name,

              quantity: item.quantity,

              total: item.totalPrice,

              price: item.price,

            }}

          />

        ))}

      </ul>

    </Card>

  );

};

export default Cart;

Step 13 in CartItem.js

import { useDispatch } from 'react-redux';

import classes from './CartItem.module.css';

import { cartActions } from '../../store/cart-slice';

const CartItem = (props) => {

  const dispatch = useDispatch();

  const { title, quantity, total, price, id } = props.item;

  const removeItemHandler = () => {

    dispatch(cartActions.removeItemFromCart(id));

  };

  const addItemHandler = () => {

    dispatch(

      cartActions.addItemToCart({

        id,

        title,

        price,

      })

    );

  };

  return (

    <li className={classes.item}>

      <header>

        <h3>{title}</h3>

        <div className={classes.price}>

          ${total.toFixed(2)}{' '}

          <span className={classes.itemprice}>(${price.toFixed(2)}/item)</span>

        </div>

      </header>

      <div className={classes.details}>

        <div className={classes.quantity}>

          x <span>{quantity}</span>

        </div>

        <div className={classes.actions}>

          <button onClick={removeItemHandler}>-</button>

          <button onClick={addItemHandler}>+</button>

        </div>

      </div>

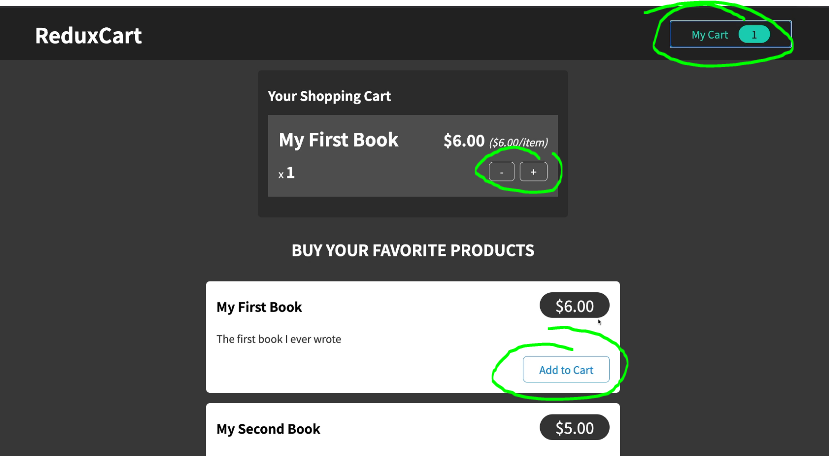
    </li>

  );

};

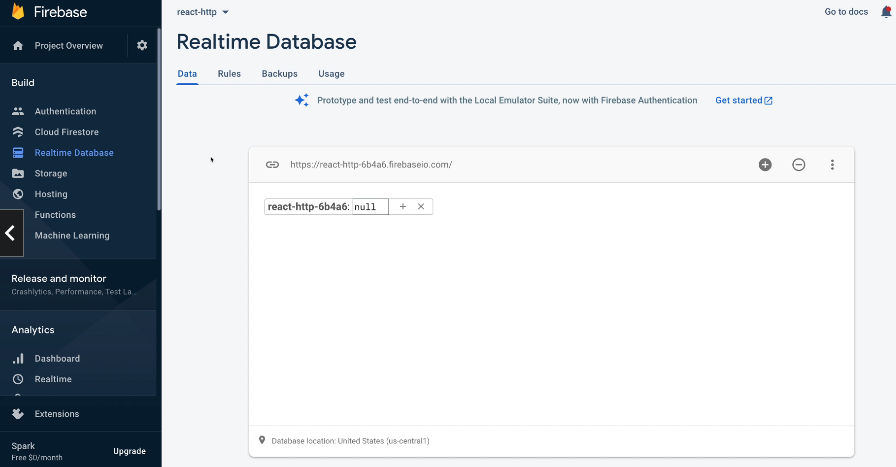
export default CartItem;

Step 14 test result all button

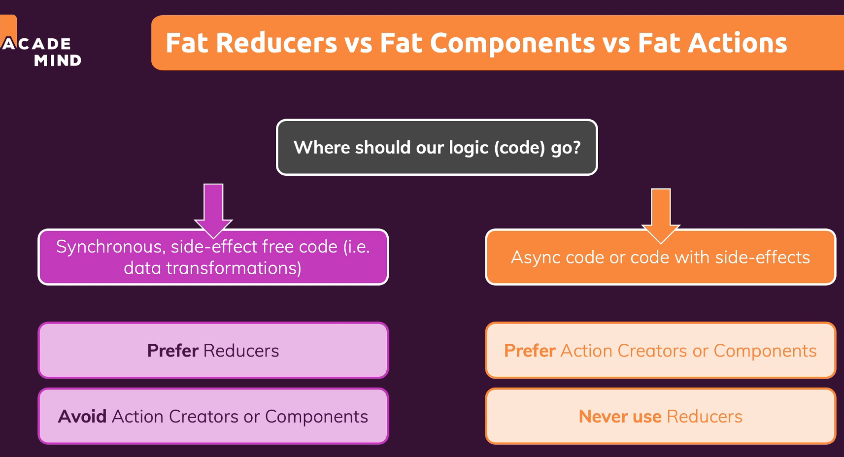


253. Redux & Sync Code

Step 1 access firebase and create new project => go to realtime database



255. Where to put our logic



256. Using useEffect with Redux

Step 1: in App.js

import { useEffect } from 'react';

import { useSelector } from 'react-redux';

import Cart from './components/Cart/Cart';

import Layout from './components/Layout/Layout';

import Products from './components/Shop/Products';

function App() {

  const showCart = useSelector((state) => state.ui.cartIsVisible);

  const cart = useSelector((state) => state.cart);

  useEffect(() => {

    fetch('https://react-http-6b4a6.firebaseio.com/cart.json', {

      method: 'PUT',

      body: JSON.stringify(cart),

    });

  }, [cart]);

  return (

    <Layout>

      {showCart && <Cart />}

      <Products />

    </Layout>

  );

}

export default App;

Step 2: Check result (now all data will send to server) (only one api in app.js)

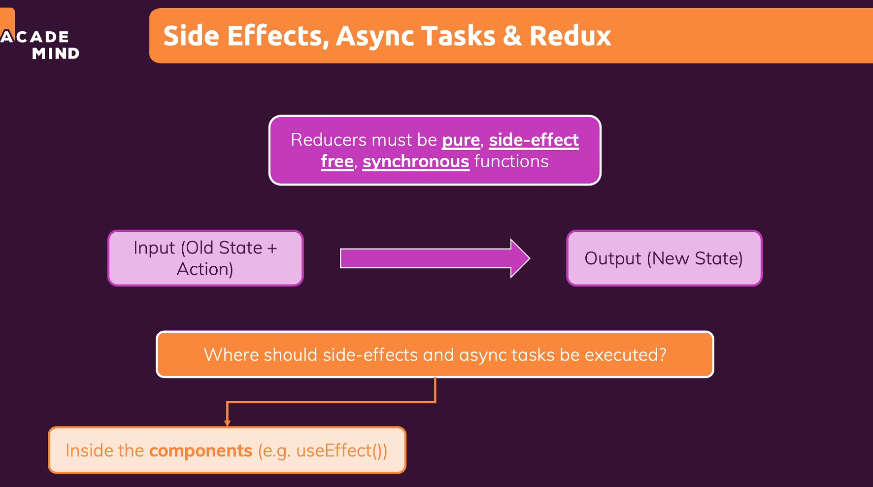
257. A Problem with useEffect

We face one problem when using useEffect the way we currently do it: It will execute when our app starts.

Why is this an issue?

It's a problem because this will send the initial (i.e. empty) cart to our backend and overwrite any data stored there.

258. Handling Http States & Feedback with Redux (Option One, write code in component parent)



1. Clone source ‘Handling Http states’ to clone file notification.js and nitification.module.js

2. in ui.slice.js

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

const uiSlice = createSlice({

  name: 'ui',

  initialState: { cartIsVisible: false, notification: null },

  reducers: {

    toggle(state) {

      state.cartIsVisible = !state.cartIsVisible;

    },

    showNotification(state, action) {

      state.notification = {

        status: action.payload.status,

        title: action.payload.title,

        message: action.payload.message,

      };

    },

  },

});

export const uiActions = uiSlice.actions;

export default uiSlice;

3. in app.js

import { Fragment, useEffect } from 'react';

import { useSelector, useDispatch } from 'react-redux';

import Cart from './components/Cart/Cart';

import Layout from './components/Layout/Layout';

import Products from './components/Shop/Products';

import { uiActions } from './store/ui-slice';

import Notification from './components/UI/Notification';

let isInitial = true;

function App() {

  const dispatch = useDispatch();

  const showCart = useSelector((state) => state.ui.cartIsVisible);

  const cart = useSelector((state) => state.cart);

  const notification = useSelector((state) => state.ui.notification);

  useEffect(() => {

    const sendCartData = async () => {

      dispatch(

        uiActions.showNotification({

          status: 'pending',

          title: 'Sending...',

          message: 'Sending cart data!',

        })

      );

      const response = await fetch(

        'https://react-http-6b4a6.firebaseio.com/cart.json',

        {

          method: 'PUT',

          body: JSON.stringify(cart),

        }

      );

      if (!response.ok) {

        throw new Error('Sending cart data failed.');

      }

      dispatch(

        uiActions.showNotification({

          status: 'success',

          title: 'Success!',

          message: 'Sent cart data successfully!',

        })

      );

    };

    if (isInitial) {

      isInitial = false;

      return;

    }

    sendCartData().catch((error) => {

      dispatch(

        uiActions.showNotification({

          status: 'error',

          title: 'Error!',

          message: 'Sending cart data failed!',

        })

      );

    });

  }, [cart, dispatch]);

  return (

    <Fragment>

      {notification && (

        <Notification

          status={notification.status}

          title={notification.title}

          message={notification.message}

        />

      )}

      <Layout>

        {showCart && <Cart />}

        <Products />

      </Layout>

    </Fragment>

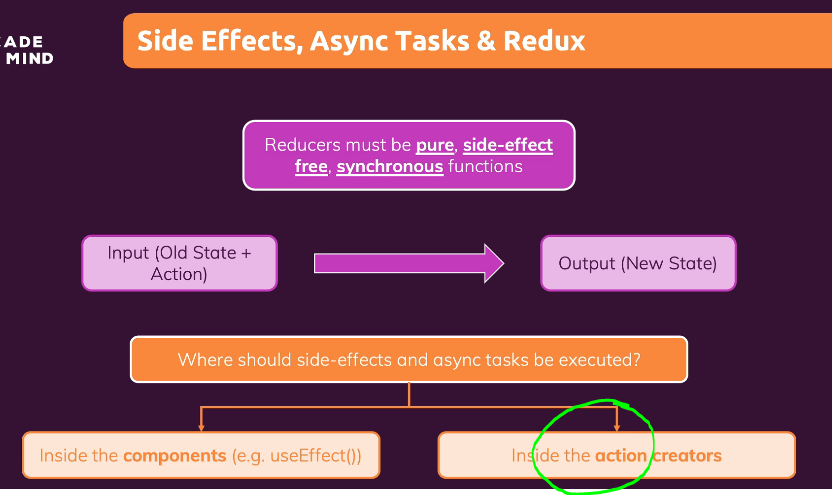
  );

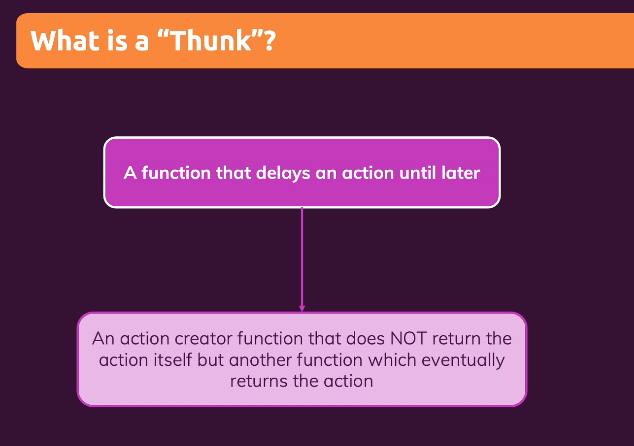
}

export default App;

4. Check result

259. Using an Action Creator Thunk (Option two, write code in action creators)





1 in cart-slice.js

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

import { uiActions } from './ui-slice';

const cartSlice = createSlice({

  name: 'cart',

  initialState: {

    items: [],

    totalQuantity: 0,

  },

  reducers: {

    replaceCart(state, action) {

      state.totalQuantity = action.payload.totalQuantity;

      state.items = action.payload.items;

    },

    addItemToCart(state, action) {

      const newItem = action.payload;

      const existingItem = state.items.find((item) => item.id === newItem.id);

      state.totalQuantity++;

      if (!existingItem) {

        state.items.push({

          id: newItem.id,

          price: newItem.price,

          quantity: 1,

          totalPrice: newItem.price,

          name: newItem.title,

        });

      } else {

        existingItem.quantity++;

        existingItem.totalPrice = existingItem.totalPrice + newItem.price;

      }

    },

    removeItemFromCart(state, action) {

      const id = action.payload;

      const existingItem = state.items.find((item) => item.id === id);

      state.totalQuantity--;

      if (existingItem.quantity === 1) {

        state.items = state.items.filter((item) => item.id !== id);

      } else {

        existingItem.quantity--;

      }

    },

  },

});

export const sendCartData = (cart) => {

  return async (dispatch) => {

    dispatch(

      uiActions.showNotification({

        status: 'pending',

        title: 'Sending...',

        message: 'Sending cart data!',

      })

    );

    const sendRequest = async () => {

      const response = await fetch(

        'https://react-http-6b4a6.firebaseio.com/cart.json',

        {

          method: 'PUT',

          body: JSON.stringify(cart),

        }

      );

      if (!response.ok) {

        throw new Error('Sending cart data failed.');

      }

    };

    try {

      await sendRequest();

      dispatch(

        uiActions.showNotification({

          status: 'success',

          title: 'Success!',

          message: 'Sent cart data successfully!',

        })

      );

    } catch (error) {

      dispatch(

        uiActions.showNotification({

          status: 'error',

          title: 'Error!',

          message: 'Sending cart data failed!',

        })

      );

    }

  };

};

export const cartActions = cartSlice.actions;

export default cartSlice;

2 in App.js

import { Fragment, useEffect } from 'react';

import { useSelector, useDispatch } from 'react-redux';

import Cart from './components/Cart/Cart';

import Layout from './components/Layout/Layout';

import Products from './components/Shop/Products';

import Notification from './components/UI/Notification';

import { sendCartData } from './store/cart-slice';

let isInitial = true;

function App() {

  const dispatch = useDispatch();

  const showCart = useSelector((state) => state.ui.cartIsVisible);

  const cart = useSelector((state) => state.cart);

  const notification = useSelector((state) => state.ui.notification);

  useEffect(() => {

    if (isInitial) {

      isInitial = false;

      return;

    }

    dispatch(sendCartData(cart));

  }, [cart, dispatch]);

  return (

    <Fragment>

      {notification && (

        <Notification

          status={notification.status}

          title={notification.title}

          message={notification.message}

        />

      )}

      <Layout>

        {showCart && <Cart />}

        <Products />

      </Layout>

    </Fragment>

  );

}

export default App;

260. Getting Started with Feching Data

1 create new cart-action.js in store

import { uiActions } from './ui-slice';

import { cartActions } from './cart-slice';

export const fetchCartData = () => {

  return async (dispatch) => {

    const fetchData = async () => {

      const response = await fetch(

        'https://react-http-6b4a6.firebaseio.com/cart.json'

      );

      if (!response.ok) {

        throw new Error('Could not fetch cart data!');

      }

      const data = await response.json();

      return data;

    };

    try {

      const cartData = await fetchData();

      dispatch(

        cartActions.replaceCart({

          items: cartData.items || [],

          totalQuantity: cartData.totalQuantity,

        })

      );

    } catch (error) {

      dispatch(

        uiActions.showNotification({

          status: 'error',

          title: 'Error!',

          message: 'Fetching cart data failed!',

        })

      );

    }

  };

};

export const sendCartData = (cart) => {

  return async (dispatch) => {

    dispatch(

      uiActions.showNotification({

        status: 'pending',

        title: 'Sending...',

        message: 'Sending cart data!',

      })

    );

    const sendRequest = async () => {

      const response = await fetch(

        'https://react-http-6b4a6.firebaseio.com/cart.json',

        {

          method: 'PUT',

          body: JSON.stringify({

            items: cart.items,

            totalQuantity: cart.totalQuantity,

          }),

        }

      );

      if (!response.ok) {

        throw new Error('Sending cart data failed.');

      }

    };

    try {

      await sendRequest();

      dispatch(

        uiActions.showNotification({

          status: 'success',

          title: 'Success!',

          message: 'Sent cart data successfully!',

        })

      );

    } catch (error) {

      dispatch(

        uiActions.showNotification({

          status: 'error',

          title: 'Error!',

          message: 'Sending cart data failed!',

        })

      );

    }

  };

};

2 remove some code in cart-slice.js

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

const cartSlice = createSlice({

  name: 'cart',

  initialState: {

    items: [],

    totalQuantity: 0,

    changed: false,

  },

  reducers: {

    replaceCart(state, action) {

      state.totalQuantity = action.payload.totalQuantity;

      state.items = action.payload.items;

    },

    addItemToCart(state, action) {

      const newItem = action.payload;

      const existingItem = state.items.find((item) => item.id === newItem.id);

      state.totalQuantity++;

      state.changed = true;

      if (!existingItem) {

        state.items.push({

          id: newItem.id,

          price: newItem.price,

          quantity: 1,

          totalPrice: newItem.price,

          name: newItem.title,

        });

      } else {

        existingItem.quantity++;

        existingItem.totalPrice = existingItem.totalPrice + newItem.price;

      }

    },

    removeItemFromCart(state, action) {

      const id = action.payload;

      const existingItem = state.items.find((item) => item.id === id);

      state.totalQuantity--;

      state.changed = true;

      if (existingItem.quantity === 1) {

        state.items = state.items.filter((item) => item.id !== id);

      } else {

        existingItem.quantity--;

        existingItem.totalPrice = existingItem.totalPrice - existingItem.price;

      }

    },

  },

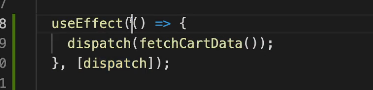
});

export const cartActions = cartSlice.actions;

export default cartSlice;

3 in App.js

Add code



import { Fragment, useEffect } from 'react';

import { useSelector, useDispatch } from 'react-redux';

import Cart from './components/Cart/Cart';

import Layout from './components/Layout/Layout';

import Products from './components/Shop/Products';

import Notification from './components/UI/Notification';

import { sendCartData, fetchCartData } from './store/cart-actions';

let isInitial = true;

function App() {

  const dispatch = useDispatch();

  const showCart = useSelector((state) => state.ui.cartIsVisible);

  const cart = useSelector((state) => state.cart);

  const notification = useSelector((state) => state.ui.notification);

  useEffect(() => {

    dispatch(fetchCartData());

  }, [dispatch]);

  useEffect(() => {

    if (isInitial) {

      isInitial = false;

      return;

    }

    if (cart.changed) {

      dispatch(sendCartData(cart));

    }

  }, [cart, dispatch]);

  return (

    <Fragment>

      {notification && (

        <Notification

          status={notification.status}

          title={notification.title}

          message={notification.message}

        />

      )}

      <Layout>

        {showCart && <Cart />}

        <Products />

      </Layout>

    </Fragment>

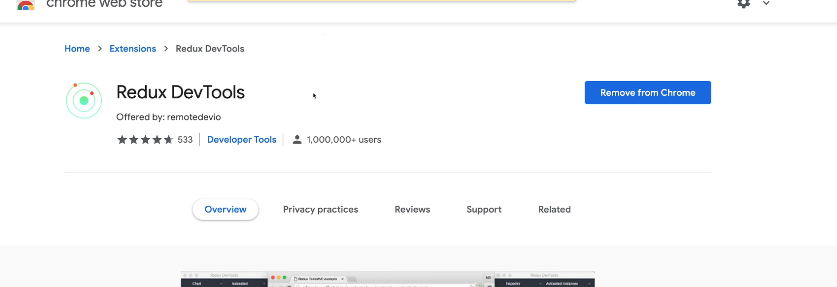
  );

}

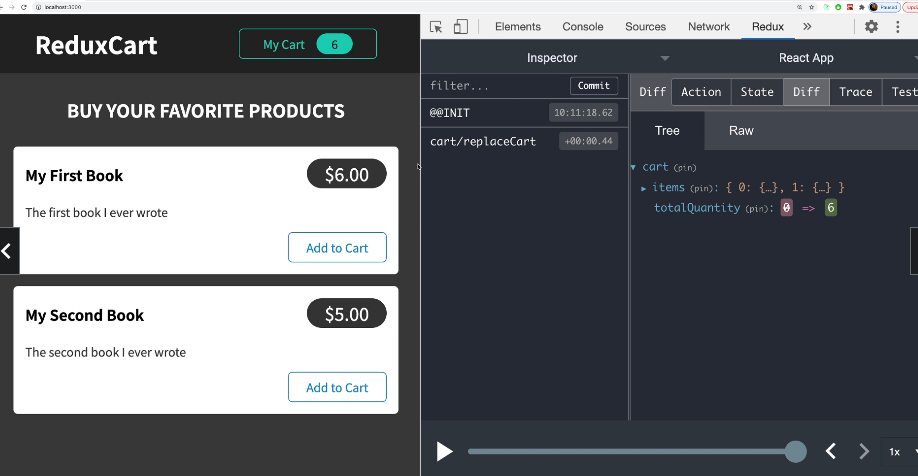
export default App;

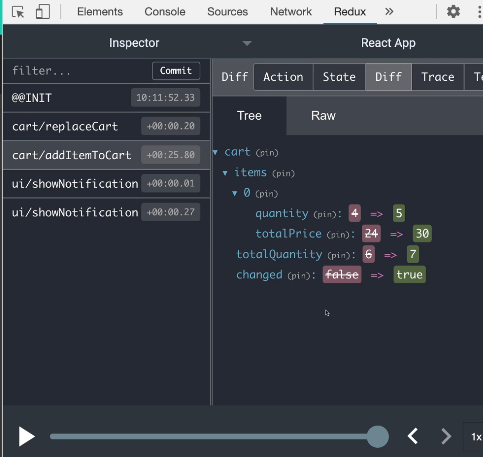
262. Exploring the redux DevTools

1 install extension

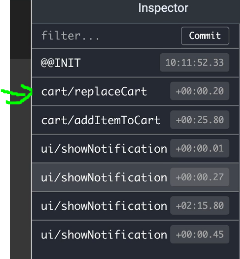


2 F12 and click Redux





+ You can jump to past, to rollback redux value



264. Module Resources

You may want to **compare your code to mine** (e.g. to find + fix errors).

For that, you find **multiple code snapshots** for this module here in this Github repository: <https://github.com/academind/react-complete-guide-code/tree/19-advanced-redux>