Web Components binding for Redux

Sheeshpaul Kamboj Follow
Mar 21 · 4 min read

Quick Summary

All the major browsers support Web Components, and being natively supported mean the Web Components can be used in any environment, such as, vanilla JS pages, React pages, Angular pages, or any other framework powered pages, making Web Components write once share anywhere components. In my exploration of Web Components, I found lack of Redux binding for Web Components, like react-redux, making Redux state management difficult for Web Components. To mitigate this lack, I have created webcomponents-redux binding package.

In this article, I will create a Counter Web Component which uses webcomponents-redux package to connect with Redux and show the webcomponent-redux API in this process. The code example is available at codesandbox. Let's get started.

Counter Component

Below is the simple Counter component that I will be using to connect with Redux. The component takes value attribute, who's content is displayed as Counter value.

```
class CounterElement extends HTMLElement {
    constructor() {
        super();
        this.count = null;
        this.attachShadow({ mode: 'open' });
    }

    static get observedAttributes() {
        return ['value'];
    }

    connectedCallback() {
        this.shadowRoot.innerHTML = `
            <div>Counter value is
                <span>${this.getAttribute('value')}</span>
            </div>`;
        this.count = this.shadowRoot.querySelector('span');
    }

    attributeChangedCallback(name, oldValue, newValue) {
        if (!this.count) {
            return;
        }

        if (name === 'value' && newValue !== oldValue) {
            this.count.innerText = newValue;
        }
    }
}
```

Counter component implementation

```
<counter-element value="1"></counter-element>
```

Counter component HTML markup

Counter value is 1

Counter component UI

Connecting with Redux

In this section, I will go through the steps to connect Counter component with Redux.

Step 1: Install webcomponents-redux Package

Use npm to install redux and webcomponents-redux packages. See command below.

```
npm install --save redux webcomponents-redux
```

Step 2: Setup Redux Store

For Counter component, I have setup one reducer, which takes increment and decrement actions to update the count and created the store object.

```
import { createStore, combineReducers } from 'redux';

function counterReducer(state = { count: 0 }, action) {
    switch (action.type) {
        case 'INCREMENT':
            return { count: state.count + 1 };
        case 'DECREMENT':
            return { count: state.count - 1 };
        default:
            return state;
    }
}

const reducers = combineReducers({
    counter: counterReducer
});

export const store = createStore(reducers);
```

Redux store setup

Step 3: Connect Component to Redux

Import connect function from webcomponents-redux. Call the connect function, passing Web Component class, and the Redux store object. The Web Component class in this example is CounterElement.

```
import { connect } from 'webcomponents-redux';
connect(CounterElement, store);
```

Connecting component to Redux

Now the Counter component is connected to Redux store and I can implement functions to receive state change and to dispatch actions.

Step 4: Implement mapStateToProps

In Counter component class, implement mapStateToProps function. This function is called by the Redux binding logic on state change. The first time mapStateToProps function is called, is during connectedCallback lifecycle and thereafter on any state change.

For Counter component, on state change, I'm comparing new count value with the old count, and when different, call attributeChangeCallback to update the Counter component UI.

```
mapStateToProps(oldState, newState) {
    if (oldState === undefined) {
        this.attributeChangedCallback(
            'value', null, newState.counter.count);
        return;
    }

    if (newState.counter.count !== oldState.counter.count) {
        this.attributeChangedCallback(
            'value', oldState.counter.count, newState.counter.count);
    }
}
```

mapStateToProps implementation for Counter component

Step 5: Implement mapDispatchToProps

In Counter component class, implement mapDispatchToProps function. The mapDispatchToProps function is only called one time during connectedCallback lifecycle, and should return an object, where each field of the object is a function, which is expected to dispatch an action to the store.

For Counter component, I'm returning two functions to send count increment and decrement actions.

```
mapDispatchToProps(dispatch) {
    return {
        increment: () => dispatch({ type: 'INCREMENT' }),
        decrement: () => dispatch({ type: 'DECREMENT' })
    };
}
```

mapDispatchToProps implementation for Counter component

I have also updated Counter component UI to include increment and decrement buttons, which when clicked call the action methods.

```
connectedCallback() {
    this.shadowRoot.innerHTML = `
        <div>
            <div>Counter value is <span>${this.getAttribute('value')}</span></div>
            <button>Increment</button>
            <button>Decrement</button>
        </div>`;

    this.shadowRoot.querySelectorAll('button')[0].addEventListener('click', () => {
        this.increment();
    });
    this.shadowRoot.querySelectorAll('button')[1].addEventListener('click', () => {
        this.decrement();
    });
    this.count = this.shadowRoot.querySelector('span');
}
```

Counter component UI with increment and decrement buttons

Counter component is now fully connected to Redux, receives state change and is able to dispatch actions.

Summary

In this article I have used webcomponents-redux package to connect Web Components to Redux for state management. The binding package uses well established practices of popular react-redux package, to lessen the learning curve for devs trying to use Redux with Web Components.

Related Resources

The following resources will give you more detail about webcomponents-redux.

- "webcomponents-redux" Github repo
- "webcomponents-redux-sample" Module based sample
- "webcomponents-redux-script-sample" Script based sample
- "webcomponents-redux demo page" Sample demo page

JavaScript · Web Development · Web Components · Redux

WRITTEN BY
Sheeshpaul Kamboj Follow
Software developer. Specialize in web development and web performance.

Write the first response