trunk 19 Branches 14 Tags Go to file Go to file Code

benjamind chore: update to v2.2.2 release 270e2ee · last year 90 Commits

.circleci	fix(ci): reverts to earlier playwright vers…	last year
.github	chore: switches github action	3 years ago
demo	feat: updates to lit v2, fixes #85 (#93)	3 years ago
src	fix: mobx6 removed IDerivation type (#…	last year
test	feat: updates to lit v2, fixes #85 (#93)	3 years ago
.editorconfig	chore: adds linting, commit-lint and ch…	4 years ago
.eslintignore	chore: adds linting, commit-lint and ch…	4 years ago
.eslintrc.json	chore: adds linting, commit-lint and ch…	4 years ago
.gitignore	feat: adds option to specify custom Re…	last year
.npmrc	chore: adds linting, commit-lint and ch…	4 years ago
.prettierrc.yaml	Rework after initial review (#5)	5 years ago
CHANGELOG.md	chore: update to v2.2.2 release	last year
CODE_OF_CONDUCT.md	Adds license and other boilerplate	5 years ago
LICENSE	Adds license and other boilerplate	5 years ago
README.md	feat: adds option to specify custom Re…	last year
commitlint.config.cjs	fix: fixes commitlint config under esm (…	last year
license.js	chore: adds linting, commit-lint and ch…	4 years ago
package-lock.json	chore: update to v2.2.1 release	last year
package.json	chore: update to v2.2.2 release	last year
tsconfig.json	Adds unit tests, fixing #7	5 years ago
web-dev-server.config.js	fix: correctly sets package as esm type…	last year
web-test-runner.config.js	fix: correctly sets package as esm type…	last year

About

Mixin and base class for using mobx with lit-element

#mobx #web-components #mixin #lit-element

Readme
Apache-2.0, Unknown licenses found
Code of conduct
Security policy
Activity
Custom properties
270 stars
35 watching
16 forks
Report repository

Releases

14 tags

Packages

No packages published

Contributors 7

Languages

TypeScript 95.1% **JavaScript** 4.9%

README Code of conduct Apache-2.0 license License Security

Snyk security monitored

lit-mobx

Mixin and base class that allow easy usage of mobx observables with `lit`.

The mixin implementation is based heavily on the work of Alexander Weiss in his `mobx-lit-element` implementation. This has been rewritten to support multiple forms of usage (mixin, or base class) as well as to be based on typescript to get type definitions.

Installation

As a dependency:

```
npm install --save @adobe/lit-mobx lit mobx
```

LitElement 2.0 Usage

This library has been updated to the latest version of Lit which we highly recommend you update to. However if you wish to continue using LitMobx with LitElement 2.0 then install the 1.0 major version:

```
npm install --save @adobe/lit-mobx@^1.0.0
```

We will backport any security patches (though don't expect there to be any) in this major version as necessary.

Demo

```
npm install
npm run demo
```

Usage

See the JavaScript and TypeScript example projects on StackBlitz. See this example for a demonstration of usage with Mobx v6 in Typescript without the use of decorators.

```
import { html, TemplateResult } from 'lit';
import { customElement, LitElement } from 'lit/decorators.js';
import { observable, action } from 'mobx';
import { MobxLitElement } from '@adobe/lit-mobx';

// create a mobx observable
class Counter {
    @observable
    public count = 0;

    @action
    public increment() {
        this.count++;
    }
}

// create instance that can be shared across components
const counter = new Counter();

// create a new custom element, and use the base MobxLitElement class
// alternatively you can use the MobxReactionUpdate mixin, e.g. `class MyElement extends MobxReactionUpdate(LitElemen
@customElement('my-element')
export class MyElement extends MobxLitElement {
    private counter = counter;

    // any observables accessed in the render method will now trigger an update
    public render(): TemplateResult {
        return html`
            Count is ${this.counter.count}
            <br />
            <button @click=${this.incrementCount}>Add</button>
        `;
    }

    private incrementCount() {
        this.counter.increment();
    }
}
```

For further examples see the demo folder.

NOTE: observables are only hooked for updating the render function if those observables are directly used within the render function. The implementation of this library is such that we essential wrap the render function in an `autorun` block to hook these observables for re-running render. If you wish to response to an observable property to calculate some other result that then itself is used in the render function, then you need to use the regular MobX methods for observability in a property changed callback or some other lifecycle function to setup that observation and the write to a property in your component that is appropriately decorated to drive Lit's regular update cycle.

Custom Reaction

In some rare cases applications may need to have multiple different major versions of Mobx within a single app. This can lead to issues where the version of mobx used in various places needs to be specifically imported from aliased versions of mobx, e.g. aliasing `mobx 6.x` to `mobx6`.

To help support these cases lit-mobx provides the custom implementation of the MobxReactionUpdate mixin, `MobxReactionUpdateCustom`, that supports taking the `Reaction` constructor as a second argument. This allows the user to define which version of mobx to pass, and ensures that no side effects are caused by lit-mobx importing `mobx` directly.

Example usage:

```
import { MobxReactionUpdateCustom } from '@adobe/lit-mobx/lib/mixin-custom.js';

// import your specific mobx version
import { Reaction } from 'mobx6';

// and pass it to the mixin to create your own mobx lit element base class
class Mobx6LitElement extends MobxReactionUpdateCustom(LitElement, Reaction) {}
```

Contributing

Contributions are welcomed! Read the Contributing Guide for more information.

Licensing

This project is licensed under the Apache V2 License. See LICENSE for more information.