MobX

Facilidade e Elegância Na Manipulação de Estado

Mathias Gheno Azzolini

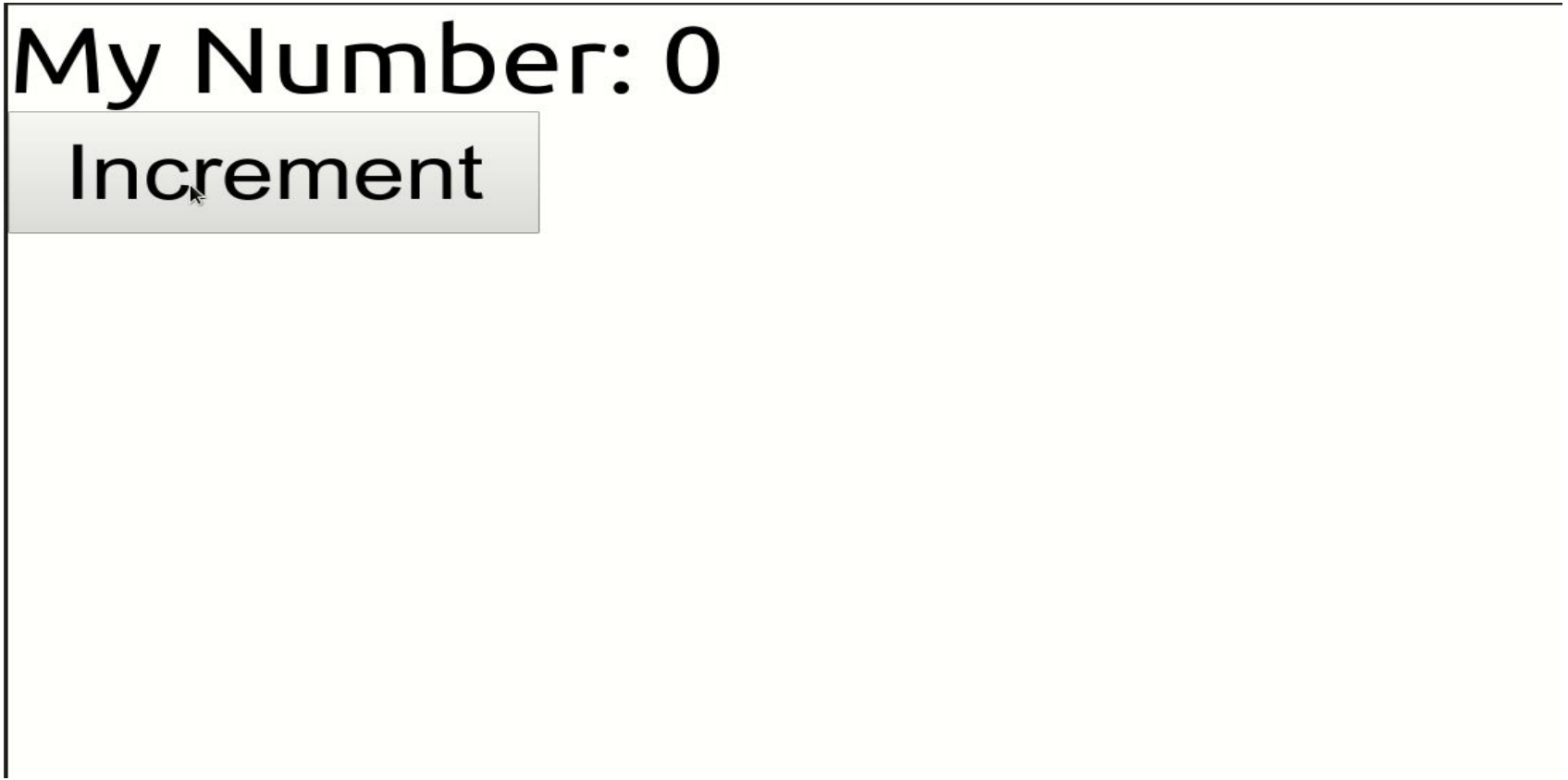
- Desenvolvedor na DBServer, por 2 Anos.
- Focado no desenvolvimento JavaScript.
- Formado em Análise e Desenvolvimento de Sistemas pelo IFRS.
- Estudante de Especialização em Engenharia de Software pela UFRGS.
- React, Angular, Typescript, Ionic, MobX, Node.js, Nest...
- Organizador novato do meetup de Node.js POA.
- Professor voluntário no Projeto Educa Transforma e Presidente do RCT São João.
- Gosto muito de música eletrônica e gatos.

Motivação

- Definição de dois novos projeto no cliente.
- Arquitetura Legado.
  - AngularJs e Knockout.js (2012)
  - Gulp / Grunt
  - ECMAscript 5
  - Sass
  - Flux
  - Node.js / Express
- Arquitetura Sugerida
  - React
  - TypeScript
  - Styled Componentes
  - Webpack
  - MobX
  - Nest.js

MobX

- Biblioteca de Manipulação de Estado
- Possui quatro principais conceitos
  - Action
  - State
  - Computed Values
  - Reactions
- React, Preact, Vue, Angular.


My Number: 0
Increment

```jsx
import React, { useState } from 'react'

const App = () => {
  const [ number, setNumber ] = useState(0)
  return (
    <div>
      My Number: {number}
      <button onClick={() => setNumber(number++)}>Incrementar</button>
    </div>
  )
}

export default App

```

```jsx
import React, { Component } from 'react';

class App extends Component {
    state = {
        number: 0,
    };

    setNumber = () => {
        const { number: value } = this.state;
        const number = value + 1;
        this.setState({ number });
    };

    render(){
        return (
            <div>
                <span>My Number: {this.state.number}</span><br/>
                <button onClick={this.setNumber}>Increment</button>
            </div>
        )
    }
}

export default App;
```

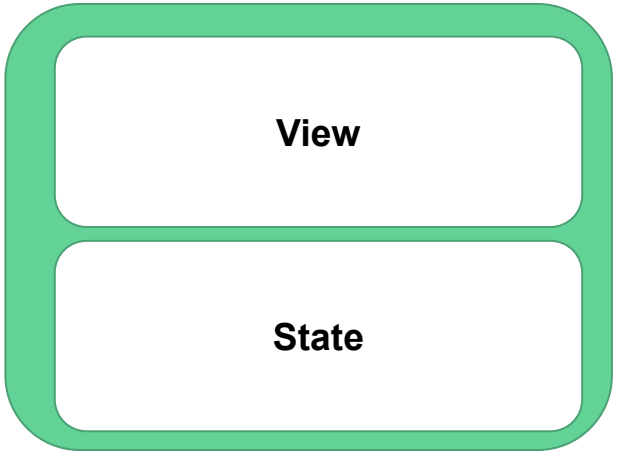
```
import React, { Component } from 'react';
import { observer } from "mobx-react";
import { observable } from "mobx";

@observer
class App extends Component {
    @observable number = 0;

    increment = () => {
      this.number ++;
    };

    render(){
        return (
            <div>
                <span>My Number: {this.number}</span><br/>
                <button onClick={this.increment}>Increment</button>
            </div>
        )
    }
}

export default App;
```

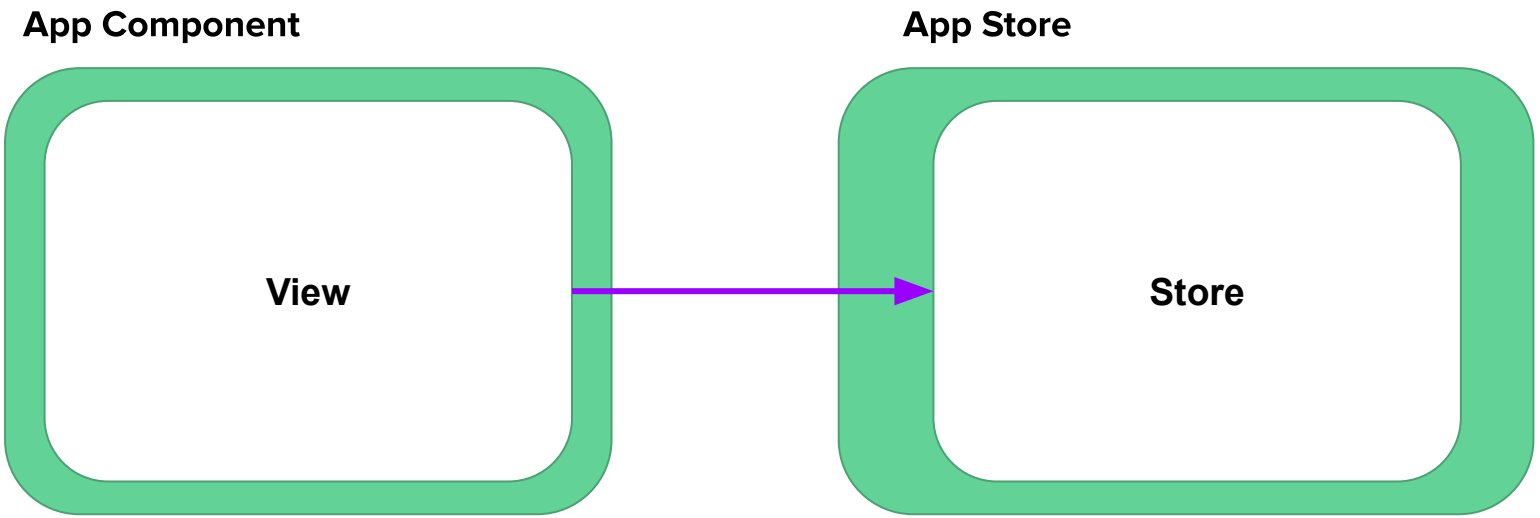
```
npx create-react-app my-app --typescript
npm install mobx --save
npm install mobx-react --save
```

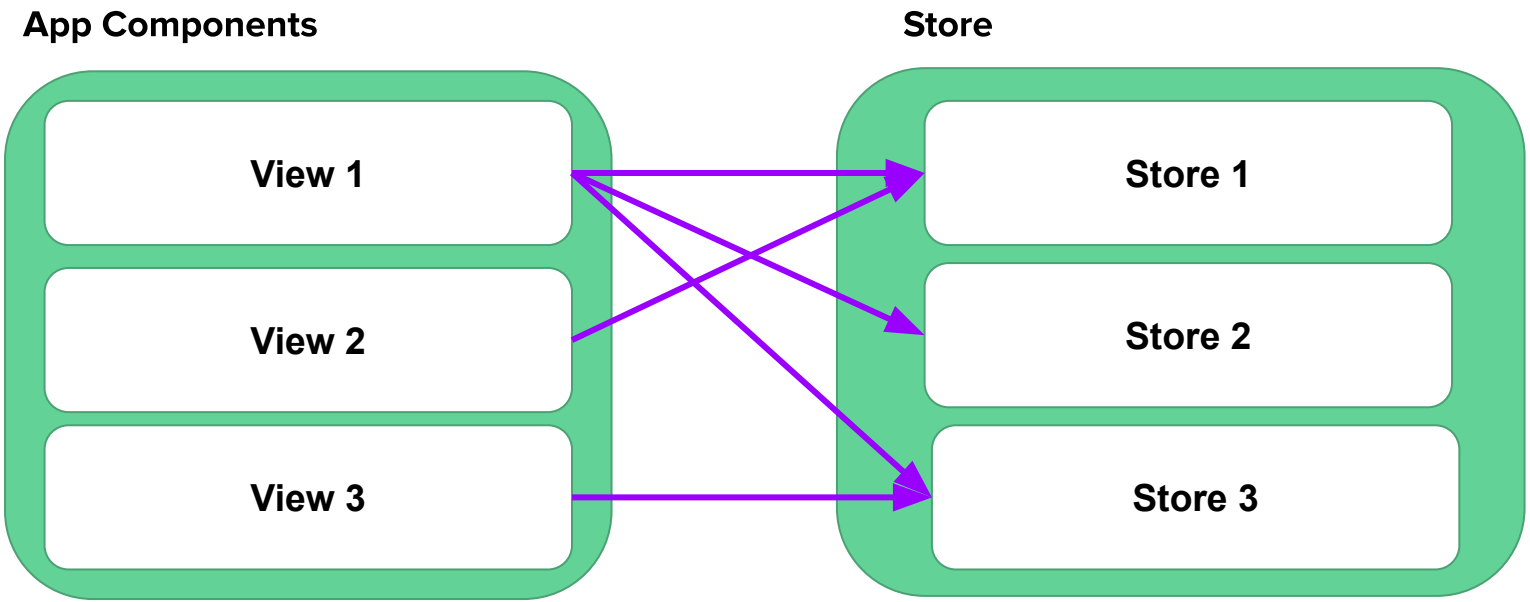
```
{
  "compilerOptions": {
    "experimentalDecorators": true,
  }
}
```

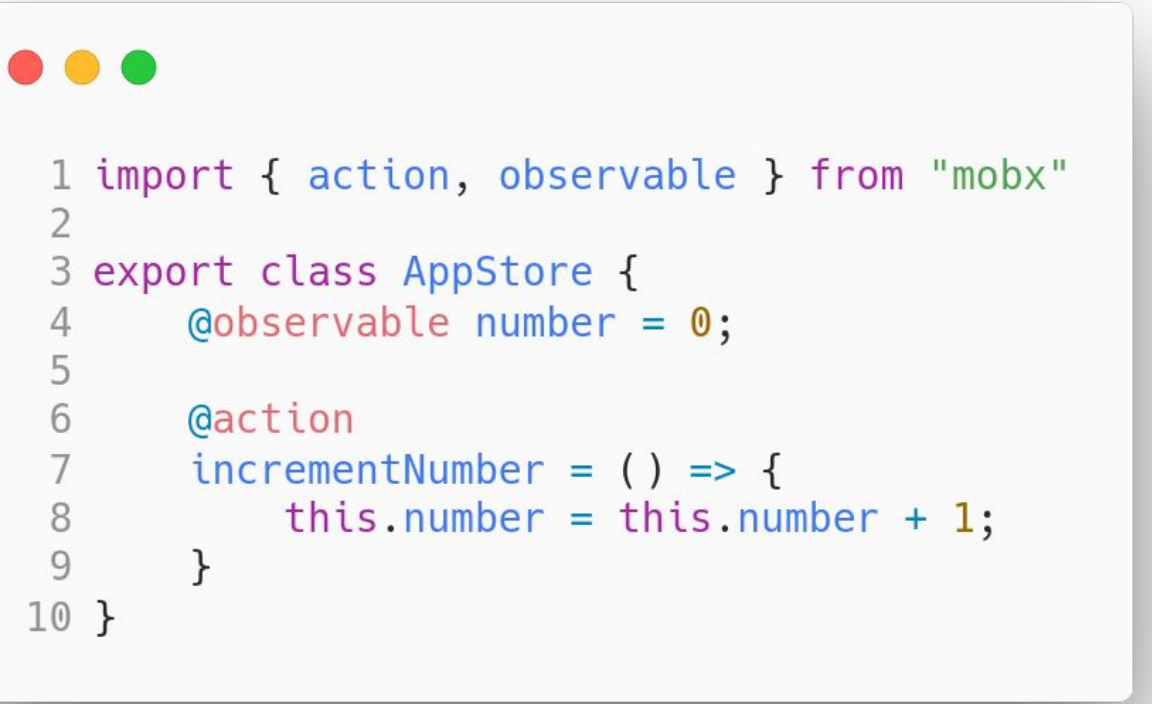
App com Hooks ou Class Component

App com MobX




App Component
App Store
View
Store


App Components
Store
View 1
View 2
View 3
Store 1
Store 2
Store 3

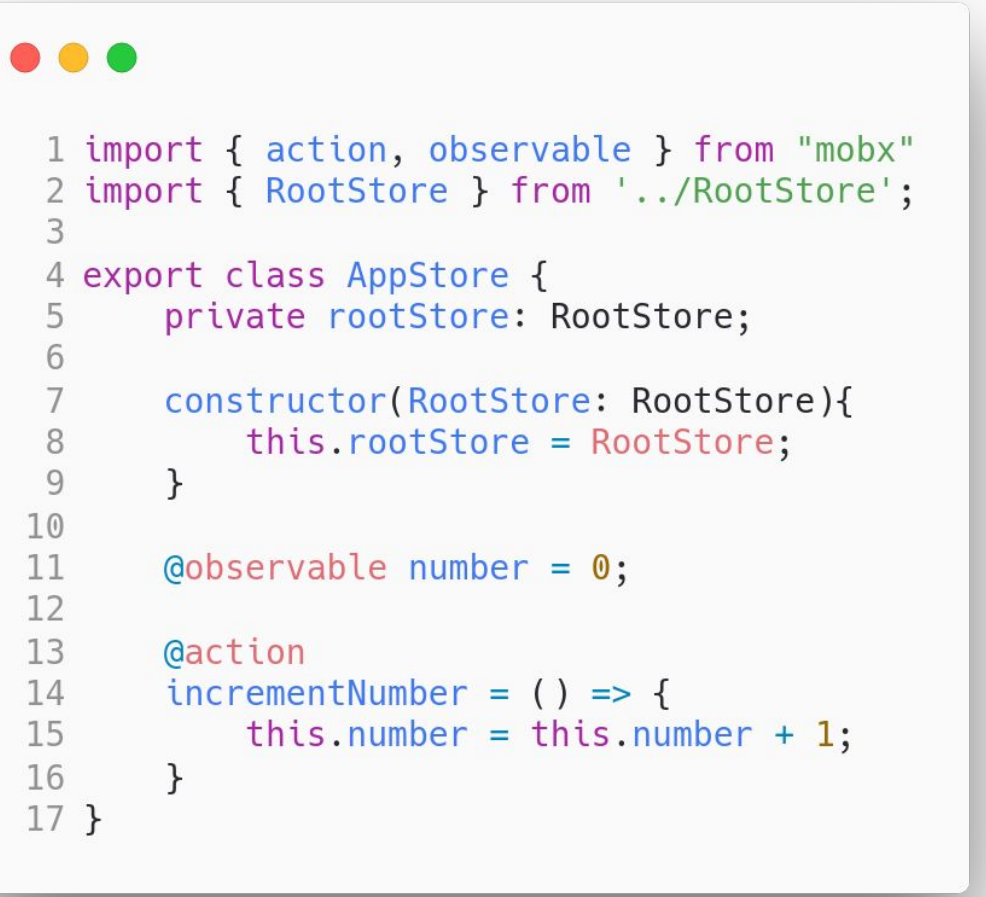


```
import { action, observable } from "mobx"

export class AppStore {
    @observable number = 0;

    @action
    incrementNumber = () => {
        this.number = this.number + 1;
    }
}
```

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4 import { AppStore } from './Store/AppStore/AppStore';
5
6 ReactDOM.render(<App AppStore={new AppStore()}/>, document.getElementById('root'));
```

```jsx
import React, { Component } from 'react';
import { observer } from "mobx-react";
import { AppStore } from "./Store/AppStore/AppStore";

@observer
class App extends Component<{AppStore: AppStore}>{
    render(){
        return (
            <div>
                <span>My Number: {this.props.AppStore.number}</span><br/>
                <button onClick={this.props.AppStore.incrementNumber}>Increment</button>
            </div>
        )
    }
}

export default App;
```

```
import { AppStore } from "./AppStore/AppStore";

export class RootStore {
    public AppStore: AppStore;

    constructor(){
        this.AppStore = new AppStore(this);
    }
}
```



```
import { action, observable } from "mobx"
import { RootStore } from '../RootStore';

export class AppStore {
    private rootStore: RootStore;

    constructor(RootStore: RootStore){
        this.rootStore = RootStore;
    }

    @observable number = 0;

    @action
    incrementNumber = () => {
        this.number = this.number + 1;
    }
}
```

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4 import { RootStore } from "./Store/RootStore";
5
6 ReactDOM.render(<App Store={new RootStore()}/>, document.getElementById('root'));
7
```

```
import React, { Component } from 'react';
import { observer } from "mobx-react";
import { RootStore } from "./Store/RootStore";

@observer
class App extends Component<{Store: RootStore}>{
    render() {
        return (
            <div>
                <span>My Number: {this.props.Store.AppStore.number}</span><br/>
                <button onClick={this.props.Store.AppStore.incrementNumber}>Increment</button>
            </div>
        )
    }
}

export default App;

```

```
1  import { AppStore } from './AppStore';
2  import { RootStore } from '../RootStore';
3
4  describe('AppStore Test', () => {
5      let appStore: AppStore;
6
7      beforeEach(() => {
8          appStore = new AppStore(new RootStore());
9      });
10
11     it('deve inicializar com number igual a zero', () => {
12        expect(appStore.number).toEqual(0);
13     });
14
15     describe('o método', () => {
16         describe('incrementNumber', () => {
17             it('deve incrementar number após executado', () => {
18                appStore.incrementNumber();
19                expect(appStore.number).toEqual(1);
20             });
21         })
22     })
23 });
```

```
import React, { ReactElement } from 'react';
import Enzyme, { shallow, ShallowWrapper } from 'enzyme';
import Adapter from 'enzyme-adapter-react-16';
import App from './App';
import { RootStore } from "./Store/RootStore";

Enzyme.configure({ adapter: new Adapter() });

describe('App Test', () => {
  let wrapper: ShallowWrapper<ReactElement<App>>;

  beforeEach(() => {
    wrapper = shallow(<App Store={new RootStore()}/>)
  });

  describe('ao renderizar', () => {
    it('deve inicializar possuir conteudo de span igual a \'My Number: 0\' ', () => {
      expect(wrapper.find('span').text()).toEqual('My Number: 0');
    });
  });

  describe('ao clicar no botão', () => {
    it('deve alterar o conteudo do span para \'My Number: 1\'', () => {
      wrapper.find('button').simulate('click');
      expect(wrapper.find('span').text()).toEqual('My Number: 1');
    });
  })
});
```

Conclusão

- Vantagens
  - Curva de aprendizado curta.
  - ESNext.
  - Pode ser utilizado por vários frameworks web.
  - Fácil de testar.
  - Pouco verboso.
- Desvantagens
  - Não sugere ou recomenda uma arquitetura de manipulação de estado.
  - Flexibiliza bastante, podendo ser ruim para times não experientes.
  - Comunidade muito direcionado ao React.

Referências

- Develop React Applications With MobX and TypeScript
- Manage Complex State in React Apps with MobX
- MobX Repository
- MobX-React Repository
- MobX Official Documentation

Obrigado!

GitHub: Mathias54
Medium: @mathiasghenoazzolini
Twitter: @mathiasgheno
Linkedin: Mathias Gheno Azzolini