REACTJS*

Минимизируем код React Redux приложения

ИЗ ПЕСОЧНИЦЫ

rovkin1 27 февраля в 17:17

Мне хотелось поделиться собственными размышлениями и способом построения структуры приложения с использованием react-redux. Так как я относительно недавно стал писать код на JS и поэтому не претендую на истину и на действительно верный способ разработки. Надеюсь эта статья поможет начинающим разработчикам писать меньше костылей которые я усердно плодил в своем проекте. Продвинутым гуру в JS я думаю не скажу ничего нового, но пишите в комментариях все что думаете по этому поводу.

Основные принципы и советы по JS

- Не считаю что чем больше кода, тем лучше. Более короткий код, на мой взгляд, легче читается и выглядит более изящным. Если что то можно написать короче, при условии что оно работает таким же образом, то я делаю это.
- Ничего плохого в том чтобы писать кучу условий на проверку тех или параметров, но если условия никогда не будут работать это плохо и это лишний код, который загрязняет проект.

Пример:

```
checkImg(user) {
 if (!user) return;
        ...
 var src = user && user.profileImageURL;
        ...
 return(
        ...)
}
```

Взглянув на этот код можно заметить, что условие user && — никогда не сработает правильно, потому что условие выше выбросит из функции раньше. Обычно такие костыли возникают, когда человек делает вставки или торопиться (как это делал я).

- Оператор && очень удобен для проверки переменной user на наличие объекта, а затем обращении к свойству объекта. Нельзя обращаться к свойствам объекта, если он равен undefined.
- Избегать чрезмерной вложенности условий. Для этого можно использовать такое правило в коде. Я бы назвал это правило метод выброса.

Пример:

```
mainCalc(obj){
if(!obj) return; // условие выхода из функции
...
... какой-то код
...
if(...) return; // след. условие выхода
...
...
if (...) return; // каждый след. часть кода зависит от предыдущих
частей
...
...
```

```
return; // таким образом производится допуск к коду без вложенности
}
```

— Простой пример уменьшения кода, где я написал функцию проверки и вывода картинки. В случае если значение свойства равнялось null или пустой строке, то вывести картинку по умолчанию.

BAD

```
checkImg(user) {
  if (user){
    if (user.profileImageURL !== undefined){
      if (user.profileImageURL === null) {
        return (<DefultImage />)
      }
      else{
        if (user.profileImageURL.length>0){
          return (<img src={user.profileImageURL} className="avatar
photo" alt="" />)
        }else{
          return (<DefultImage />);
        }
      }
    }
    else {
      return;
    }
  }
  return;
}
```

GOOD

```
checkImg(user) {
  var u = user&&user.profileImageURL;
  if (u===undefined) return;
  if (!u) return (<DefultImage />);
```

```
  return (<img src={u} className="avatar photo" alt=""/>);
},
```

Важно отметить, что необходимость проверки зависит от разных условий. Если из примера выше мы знаем, что в user у нас всегда попадает значение типа объект, то проверять его нет смысла, так как в JS даже пустой объект всегда true. Поэтому будет правильней обратиться сразу к свойству объекта, которое там должно быть.

```
if (user.profileImageURL !== undefined){...
```

— Условия проверки можно писать более коротким способом:

```
if (user.profileImageURL){...
```

Это работает одинаково, а наша задача уменьшить код как это возможно.

— Но часто бывают ситуации, когда у нас вложенность выше, чем просто объект –> свойство, допустим, есть вложенный объект и обращаться к его свойствам напрямую уже нельзя, поэтому нужна проверка вложенного объекта

— Установка библиотек это отдельная тема потому что одни библиотеки экономят кучу времени другие могут дать вам обратный эффект прежде чем вы найдете подходящую и полностью настроите ее под ваши нужды, но если можно обойтись тремя строчками кода, то я думаю это плохая затея ставить

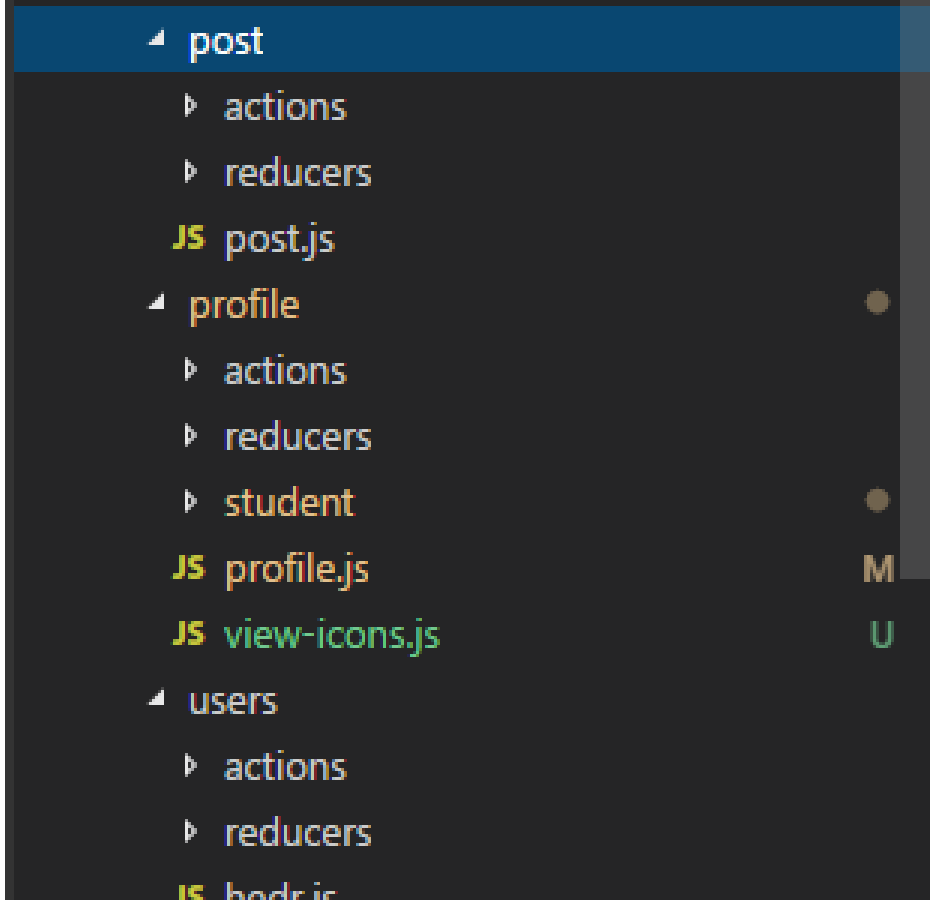
отдельную либу для этого.

Переходим к react-redux

1. Для начинающих делать проект на JS с использованием react-redux я советую не начинать проект с настройки WebPack, Babel, linter. Ответ на вопрос почему, находится в этой статье.
Ден Абрамов – создатель redux — также советует использовать Create React App.

2. Структура построения архитектуры приложения это тоже отдельная тема и поэтому я решил изучить этот вопрос подробнее после исследования десятков статей мне попалась статья разработчика. Который советует строить структуру не так как это сделано в большинстве примеров. Для больших проектов это огромный плюс, если следовать этим правилам.

Вот структура компонентов моего проекта и в каждом разделе свои actions и reducers:



И поэтому при высокой вложенности папок не приходится писать вот такие вещи:

```
import React from 'react';
import {connect } from "react-redux";
import * as Pnotify from "../../../../helper-modules/pnotify/pnotify";
import * as userActions from "../../../../actions/users";
import Profile from '../../../profile'
```

Также не приходится видеть в папке actions 100500 файлов. Для небольшого проекта это конечно может быть не очень важно, но для большого мне кажется очень удобно. Вы конечно все actions можете хранить в одном файле, но это на мой взгляд тоже не очень правильно, так как в каждый компонент импортируются функции actions всех компонентов.

3. Вот ссылка на мой написанный проект — там не все идеально,

но в нем можно посмотреть структуру приложения. Но так уж сложилось, что в этом проекте я больше не участвую. Пусть этот код не лежит без дела.

4. При помощи PropTypes мы можем определить, какие параметры должен принимать компонент и в случае передачи в компонент неверных параметров будет выведена ошибка или предупреждение, а также можем установить значения по умолчанию. Но без этой штуки можно вполне обойтись и не добавлять их в проект по одной простой причине вы можете проверить данные до передачи их в компонент:

```
{this.state.visibleID&&(<ViewIcons location={this.state.location}
visibleID={this.state.visibleID} />)}
```

Важное дополнение, компонент ViewIcons будет рендерится только после того как в переменную visibleID попадет какое либо значение, плюс если в доп. компоненте идет обращение к API то это поможет нам получить эти параметры в процедуре componentDidMount() в которой и следует делать все запросы. В свою очередь значения по умолчанию для компонента можно сделать через state. Ну если вам кажется что стоит использовать PropTypes, то используйте, потому что цель этой статьи уменьшать код а не увеличивать.

5. Так как в redux store у меня получает данные с сервера, то до момента пока данные не придут он равен пустому объекту, но допустим с сервера придет не объект, а массив объектов и обратится к какому либо свойству уже нельзя, для этого можно

проверить первый элемент массива что он не равен undefined.

```
{this.props.Menu[0]&&this.createMenu()}
```

6. Локальный state компонента можно использовать для того чтобы сначала получить значения из store, но это мне кажется не всегда оправданно, тут как грань двух концов:

- в первом случае придется делать переменные локального state и присваивать им значения затем проверять их в функции render;
- во втором случае если не использовать локальный state, мы будем обращаться к свойствам объектов или элементу массива которые находятся в store;
- и третий, лучший на мой взгляд, это присваивать по умолчанию для элементов redux store пустую строку или ноль и тогда не будет необходимости обращаться к свойствам объектов или первым элементам массива (undefined назначить элементу store нельзя).

```
const menu = (state='', action) => {
switch(action.type) {
case "MENU_TREE":
```

Это просто моменты, с которыми я столкнулся на практике. Если кто-то знает как сделать изящней, то непременно пишите, буду рад.

+9

Поделиться:

Сохранить:

Комментарии (22)

Похожие публикации

Нянчим проект на React-redux с пелёнок

32

xnim • 21 сентября 2016 в 10:28

Честный MVC на React + Redux

44

MrCheater • 18 июля 2016 в 04:59

Redux-Redents — (еще) один модуль для работы с серверными данными из React-Redux приложений.

11

qualife • 7 июля 2016 в 13:38

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на русском языке

14

alexkuku • вчера в 16:12

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада

4

lahmatiy • вчера в 13:05

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе

25

ПЕРЕВОД

Smileek • вчера в 10:32

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации

2

ПЕРЕВОД

ru_vds • вчера в 12:04

Как адаптировать игру на Unity под iPhone X к апрелю

0

P1CACHU • вчера в 16:13

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

33

HostingManager • вчера в 13:49

Обзор рынка моноколес 2018

70

lozga • вчера в 06:58

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

40

alizar • вчера в 15:14

Стивен Хокинг и его работа — что дал ученый человечеству?

8

marks • вчера в 14:46

Sunlike — светодиодный свет нового поколения

17

AlexeyNadezhin • вчера в 20:32