# React初步试错

字数155 阅读351 评论0 喜欢0

1、React的return中如果使用多个标签必须放在一个div中，否则浏览器无法解析

记得加{},经常忘

2、ReactComponentWithPureRenderMixin报错，原因是antd没装好

3、JS报错 Uncaught TypeError: Cannot read property 'extend' of undefined  
原因：在business.html中引用css未生效，在main.css中引用生效，so strange

**4、"Uncaught Error: Minified exception occurred; use the non-minified dev environment for the full error message and additional helpful warnings."  
原因：render()中的注释导致，删除注释即可**

**这不是唯一的原因，但是这一条有用**

5、state树更新失败:乱用antd中的分页，导致数据显示错误

npm install classnames

var classNames **=** require('classnames');

classNames('foo', 'bar'); *// => 'foo bar'*

classNames('foo', 'bar'); *// => 'foo bar'*

classNames('foo', { bar**:** true }); *// => 'foo bar'*

classNames({ 'foo-bar'**:** true }); *// => 'foo-bar'*

classNames({ 'foo-bar'**:** false }); *// => ''*

classNames({ foo**:** true }, { bar**:** true }); *// => 'foo bar'*

classNames({ foo**:** true, bar**:** true }); *// => 'foo bar'*

*// lots of arguments of various types*

classNames('foo', { bar**:** true, duck**:** false }, 'baz', { quux**:** true }); *// => 'foo bar baz quux'*

*// other falsy values are just ignored*

classNames(null, false, 'bar', undefined, 0, 1, { baz**:** null }, ''); *// => 'bar 1'*

//////////////////////////////////////////////////////////////////////////////////////////////////////我从知乎前端界面中学到了什么

package.json

**"start"**: **"webpack-dev-server --config webpack.config.js --port 3000 --host 0.0.0.0 --progress --color --profile --history-api-fallback"**,  
**"build"**: **"webpack --config ./webpack.config.js"**

指定了

**--config webpack.config.js**

**--port 3000**

**--host 0.0.0.0**

webpack.config.js

**const** *AssetsPlugin* = *require*(**'assets-webpack-plugin'**);  
*//a generated hash in them*

在生成的文件中加入hash码

**output**: {  
 **path**: path.join(\_\_dirname, **'build'**),  
 **publicPath**: **'/static/'**,  
 **filename**: **'[name].[hash].bundle.js'**}

////////////////////////////////////////////////////

plugins: [] 部分

////////////////////////////////////////////////////

**new** *AssetsPlugin*({  
 **filename**: **'build/webpack.assets.js'**,  
 processOutput: assets => **'window.WEBPACK\_ASSETS = '** + **JSON**.stringify(assets)  
}),

#### processOutput

Optional. Defaults is JSON stringify function.

Formats the assets output.

new AssetsPlugin({

processOutput: function (assets) {

return 'window.staticMap = ' + JSON.stringify(assets)

}

})

//////////////////////////////////////////////////////////////////////////////////////////

<**script**>  
 *// webpack.assets.js 是总的入口文件* **document**.write(**'<script type="text/javascript" src="./build/webpack.assets.js?v='** + (**new** Date().getTime()) + **'"><\/script>'**);  
</**script**>  
<**script**>  
 *// 必须要等 webpack.assets.js 加载完后才会有 WEBPACK\_ASSETS 变量* **document**.write(**'<script type="text/javascript" src=".'** + **window**.**WEBPACK\_ASSETS**.**commons**.**js** + **'"><\/script>'**);  
 **document**.write(**'<script type="text/javascript" src=".'** + **window**.**WEBPACK\_ASSETS**.**index**.**js** + **'"><\/script>'**);  
</**script**>

//////////////////////////////////////////////////////////////////////////////////////////

Common Chunks 插件的作用就是提取代码中的公共模块，然后将公共模块打包到一个独立的文件中去，以便在其它的入口和模块中使用。

**new** *webpack*.**optimize**.CommonsChunkPlugin(**'commons'**, **'[name].[hash].bundle.js'**),

接下来，我们在配置项中添加CommonChunksLoader的配置，使用它来提取公共模块。

1

var webpack = require('webpack');

module.exports = {

entry:{

main1:'./main',

main2:'./main.2'

},

output:{

filename:'bundle.[name].js'

},

plugins: [

new webpack.optimize.CommonsChunkPlugin('common.js', ['main1', 'main2'])

]

};

注意第1行，添加了webpack的引用（同时也要在项目目录下安装webpack），然后添加了plugins选项，引用了webpack.optimize.CommonsChunkPlugin来提取公共模块，参数common.js表示公共模块的文件名，后面的数组元素与entry一一对应，表示要提取这些模块中的公共模块。

////////////////////////////////////////////////////////////////////////////////////////////////////

**new** *webpack*.NoErrorsPlugin(),

## NoErrorsPlugin new webpack.NoErrorsPlugin()

跳过编译时出错的代码并记录，使编译后运行时的包不会发生错误。

**//////////////////////////////////////////////////////////////**

**new** *webpack*.DefinePlugin({  
 **\_\_DEBUG\_\_**: **true**,  
 **'process.env'**: {  
 **'NODE\_ENV'**: **JSON**.stringify(**"production"**)  
 }  
})

环境设置

///////////////////////////////////////////////////////////////////////

前端部分

**import** { **createHashHistory**, **createHistory** } **from 'history'**;

**const** history = **createHistory**();  
  
**const** *Root* = **function**() {  
 **return** (  
 <**Router history=**{history}>  
 <**Route path="/" component=**{App}>  
 <**Route path="cd" getComponent=**{*getComponentB*}></**Route**>  
 </**Route**>  
 </**Router**>  
 )  
};

react-router部分对于 history的使用

///////////////////////////////////////////////////////////

如何引用style

方法1

<**div className=**{styles[**'main-wrap'**]}>

方法2

**import** styles **from './navigation.css'**;  
**import** cx **from 'classnames'**;

<**ul className=**{cx(styles[**'header-nav-profile-dropdown'**]) }>

<**i className=**{cx(styles[**'sign-icon'**],styles[**'focus-question'**])}>

//叠加两种样式

方法3

当然类名简单的情况下：

**className=**{styles.**header**) }

官方教程

# 类名操作

classSet()是一个简洁的工具，用于简单操作DOM中的class字符串。

这里是一个常见的场景，处理方式中没有使用classSet()：

// inside some `<Message />` React component

render: function() {

var classString = 'message';

if (this.props.isImportant) {

classString += ' message-important';

}

if (this.props.isRead) {

classString += ' message-read';

}

// 'message message-important message-read'

return <div className={classString}>Great, I'll be there.</div>;

}

这会很快变得单调乏味，因为指定类名的代码很难阅读，并且容易出错。classSet()解决了这个问题：

render: function() {

var cx = React.addons.classSet;

var classes = cx({

'message': true,

'message-important': this.props.isImportant,

'message-read': this.props.isRead

});

// same final string, but much cleaner

return <div className={classes}>Great, I'll be there.</div>;

}

当使用classSet()的时候，传递一个对象，对象上的键是你需要或者不需要的CSS类名。对应真值的键将会成为结果字符串的一部分。

classSet也允许传递一些类名作为参数，然后拼接这些类名：

render: function() {

var cx = React.addons.classSet;

var importantModifier = 'message-important';

var readModifier = 'message-read';

var classes = cx('message', importantModifier, readModifier);

// Final string is 'message message-important message-read'

return <div className={classes}>Great, I'll be there.</div>;

}

////////////////////////////////////////////////////////////////

简单逻辑

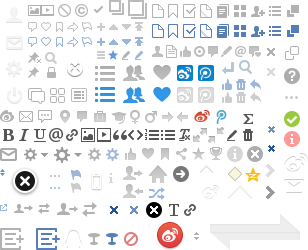
<**div className=**{cx(styles[**'header-nav-user-profile'**])} **onMouseLeave=**{**this**.handleProfileOut} **onMouseEnter=**{**this**.handleProfileHover}>

getInitialState: **function**() {  
 **return** {  
 **dropdownIsShow**:**false** }  
},  
handleProfileHover: **function**() {  
 **this**.*setState*({  
 **dropdownIsShow**: **true** })  
},  
handleProfileOut: **function**() {  
 **this**.*setState*({  
 **dropdownIsShow**: **false** })  
}

<**ProfileDropdown isShow=**{**this**.**state**.**dropdownIsShow**}/>

//////////////////////////////////////////////////////////////////////////////

如何使用icon 图片



css background-position

.**header-nav-profile-dropdown i**{  
 **display**: **inline-block**;  
 **background-image**: **url**(**../assets/images/icon.png**);

//i标签的背景图片都相同  
 **width**: 20**px**;  
 **height**: 20**px**;

这里的大小与每一个元件都相同  
 **margin**: 0 11**px** 0 15**px**;  
 **vertical-align**: -7**px**;  
}  
.**header-nav-profile-dropdown** .**icon-home**{  
 **background-position**: -6**px** -7**px**;

//通过具体的偏移选择要用的图样。  
}  
.**header-nav-profile-dropdown** .**private-message**{  
 **background-position**: -6**px** -34**px**;  
}  
.**header-nav-profile-dropdown** .**profile-setting**{  
 **background-position**: -6**px** -60**px**;  
}  
.**header-nav-profile-dropdown** .**profile-exit**{  
 **background-position**: -6**px** -87**px**;  
}

<**ul className=**{cx(styles[**'header-nav-profile-dropdown'**]) }>  
 <**li**><**a**><**i className=**{cx(styles[**'icon-home'**]) }></**i**>我的主页</**a**></**li**>  
 <**li**><**a**><**i className=**{cx(styles[**'private-message'**]) }></**i**>私信</**a**></**li**>  
 <**li**><**a**><**i className=**{cx(styles[**'profile-setting'**]) }></**i**>设置</**a**></**li**>  
 <**li**><**a**><**i className=**{cx(styles[**'profile-exit'**]) }></**i**>退出</**a**></**li**>  
</**ul**>

////////////////////////////////////////////////////////////////////////////