Business web创建流程

# 1.初始化

npm init

2.babel加载器

$ npm install babel-core babel-loader --save-dev

3.配置.babelrc文件

$ npm install babel-preset-es2015 babel-preset-react --save-dev

type null>.babelrc

.babelrc文件配置

4.安装webpack

$ npm i webpack webpack-dev-server --save-dev

$ npm install webpack-cli -D

5.webpack 配置：

Entry output mode

(

1. Mode

Mode: 'development',

plugins: [

new webpack.NamedModulesPlugin(),

new webpack.DefinePlugin({ "process.env.NODE\_ENV": JSON.stringify("development") }),

],

1. 打包
2. 热加载

webpack-dev-server是一个小型服务器，在开发过程中可以实时刷新网页。

在webpack.config.js中添加：

devServer: {

historyApiFallback: true,

inlint: true,

contentBase: path.join(\_\_dirname, "dist"), //告诉服务器从哪里提供内容

port: '8080'

},

devServer.proxy？？

HMR???

React-hot-??

1. 按需加载
2. 缓存

问题：

地址间关系：

Entry

Output

devServer

html->script

6.样式loader

$ npm install --save-dev css-loader style-loader postcss-loader

$ npm install --save-dev autoprefixer

webpack.config.js文件loader添加

module: {

rules: [

{

test: /\.css$/,

exclude: /node\_modules/,

use: [

{

loader: 'style-loader',

},

{

loader: 'css-loader',

options: {

importLoaders: 1,

}

},

{

loader: 'postcss-loader'

}

]

}

]

}

postcss.config.js文件

css-loader: 在js中加载css

style-loader: 把CSS作为style加载到html中

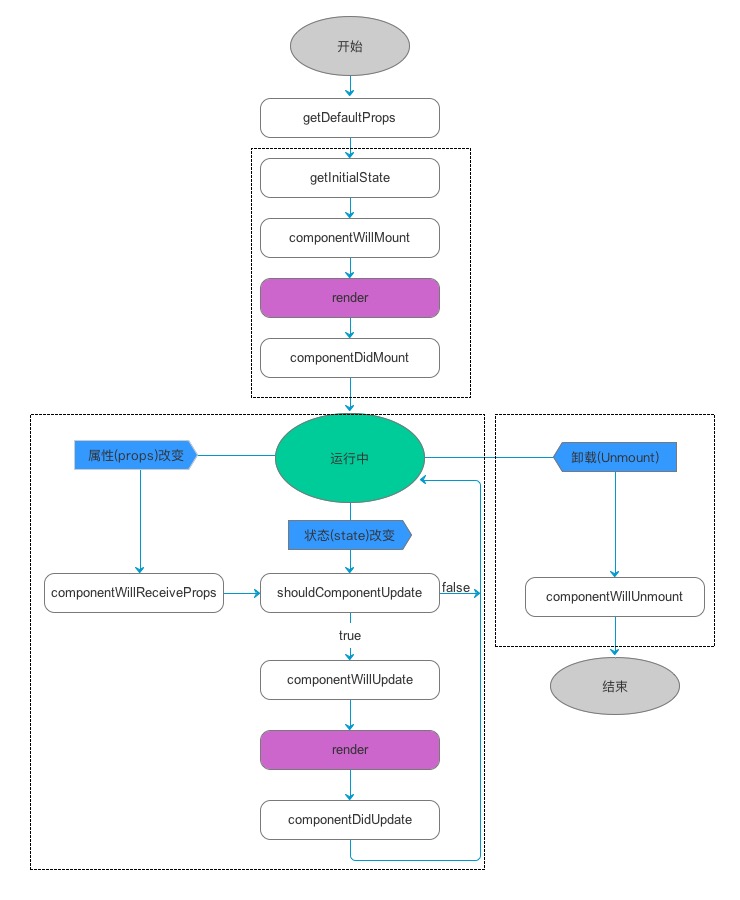
post-loader： 是一种允许使用js插件转换样式的工具，有其中Autoprefixer插件可用于兼容性转换。

# 7.编写router

# 8.CSS mix优化待解决

# 9.react

## （1）生命周期



React.Component react组件类

constructor(props)

将props挂到this上，并声明state，可以创建局部state，如： this.handleClick = this.handleClick.bind(this)。事件订阅最好在componentDidMount()上，

另外，避免在constructor(props)内部将pros赋值给state。如果想让一些state与props相关，可以使用getDerivedStateFromProps（）和 componentWillReceiveProps（）

**componentWillMount()**

组件初始化时只调用，以后组件更新不调用，整个生命周期只调用一次，此时可以修改state。

**Render()**

创建虚拟dom，进行diff算法，更新dom树都在此进行。此时就不能更改state了

**componentDidMount()**

组件渲染之后调用，只调用一次。可以在此处获取DOM节点、 load数据和订阅，对于订阅事件要在componentWillUnmount() 取消订阅。

注意：在componentDidMount()中调用this.setState()只会在浏览器更新屏幕触发render()，所以会影响性能还是什么？用回调吗？？

It will trigger an extra rendering, but it will happen before the browser updates the screen. This guarantees that even though the render() will be called twice in this case, the user won’t see the intermediate state.

**componentWillReceiveProps**

组件初始化时不调用，组件接受新的props时调用。接受newProps做参

**shouldComponentUpdate(nextProps, nextState)**

react性能优化非常重要的一环。组件接受新的state或者props时调用，我们可以设置在此对比前后两个props和state是否相同，如果相同则返回false阻止更新，因为相同的属性状态一定会生成相同的dom树，这样就不需要创造新的dom树和旧的dom树进行diff算法对比，节省大量性能，尤其是在dom结构复杂的时候

**componentWillUpdate**

组件初始化时不调用，只有在组件将要更新时才调用，此时可以修改state

**componentDidUpdate(prevProps, prevState, snapshot)**

组件更新后调用，并可以在此处操作DOM。通过比较prevProps和this.pros做网络请求。不可以直接调用**setState() 。必要时可以使用**prevProps和this.pros比较做条件，建议直接使用props。

**componentWillUnmount()**

组件将要卸载时调用，一些事件监听和定时器需要在此时清除

**static getDerivedStateFromProps()**

**getSnapshotBeforeUpdate()**

## （2）数据通信

Contex适用场景：

多余多层数据传输，或者兄弟节点共享数据时使用。通过属性将数据传递给后代

const ThemeContext = React.createContext('light');

<ThemeContext.Provider {…someProps}>

{children}

</ ThemeContext.Provider >

则<ThemeContext>内部所有子节点可以共享数据，共享方法：

Children:  
render() {

return(

<ThemeContext.Consumer>

{theme => <Button {...props} {…someProps } />}

</ThemeContext.Consumer>

)

}

Refs获取子节点数据，或者将状态更新函数作为子节点属性，通过调用函数更新状态。

# 10. Router

React中路由传参及接收参数的方式

注意： 路由表改变后要重启服务

方式 一：

通过params

1.路由表中

<Route path=' /sort/:id ' component={Sort}></Route>

2.Link处

HTML方式

<Link to={ ' /sort/ ' + ' 2 ' } activeClassName='active'>XXXX</Link>

JS方式

this.props.router.push( '/sort/'+'2' )

3.sort页面

通过

this.props.params.id 就可以接受到传递过来的参数（id）

方式 二：

通过query

前提：必须由其他页面跳过来，参数才会被传递过来

　　　 注：不需要配置路由表。路由表中的内容照常：<Route path='/sort' component={Sort}></Route>

1.Link处

HTML方式

<Link to={{ path : ' /sort ' , query : { name : 'sunny' }}}>

JS方式

this.props.router.push({ path : '/sort' ,query : { name: ' sunny'} })

2.sort页面

this.props.location.query.name

方式 三：

通过state

同query差不多，只是属性不一样，而且state传的参数是加密的，query传的参数是公开的，在地址栏

1.Link 处

HTML方式：

<Link to={{ path : ' /sort ' , state : { name : 'sunny' }}}>

JS方式：

this.props.router.push({ pathname:'/sort',state:{name : 'sunny' } })

2.sort页面

this.props.location.state.name

Router history出现问题：

问题解决：

import { Router, Route, IndexRoute, browserHistory, hashHistory } from 'react-router';

import {syncHistoryWithStore} from 'react-router-redux'

const history = syncHistoryWithStore(browserHistory, store);

出现BUG

Uncaught TypeError: Cannot read property 'listen' of undefined at syncHistoryWithStore

解决方法

import { createBrowserHistory } from 'history';

const history = syncHistoryWithStore(createBrowserHistory(), store);

## API

### <Router/>

<Router path=””/>

<Router exact/>

<Router render = {func}/>

# 11. ant design

安装：

npm install antd –save

配置：babel-plugin-import模块

babel: {

presets: ['es2015', 'stage-0', 'react'],

plugins: ['transform-runtime', ['import', {

libraryName: 'antd',

style: 'css'

}]]

}

使用less?还是postSASS

通过 npm 安装 antd 和 babel-plugin-import 。babel-plugin-import 是用来按需加载 antd 的脚本和样式的，

示例：

import { Progress, Button } from 'antd'; //引入antd

const ButtonGroup = Button.Group; //按钮组

......

<Progress percent={this.state.percent} /> //使用进度条组件，percent是组件提供的配置项

<Progress type="circle" percent={this.state.percent} />

<ButtonGroup>

<Button type="ghost" onClick={this.decline.bind(this)} icon="minus" />

<Button type="ghost" onClick={this.increase.bind(this)} icon="plus" />

</ButtonGroup>

......

表单使用：

1. 如何将父组件属性传入到表单中
2. 如何设置表单的默认选项或默认值
3. mapPropsToFields（）
4. getFieldDecorator
5. setFieldsValue（）
6. 上传文件组件
7. 父组件通过通过this.refs.WrappeForm.setFieldsValue（）设置子表单显示的

this.refs.WrappeForm.setFieldsValue({selectedChapter: ""});

(9) upLoading的fileList必须是可控组件，否则onchange只能跳入一次

（10）表单数据验证

this.props.form.validateFields

{getFieldDecorator('fileName',

rules: [ { required: true, message: 'Please select your country!' },],

valuePropName: ‘xxx',

getValueFromEvent: this.func,

})

# 11. Node

(1)node读取文件

(2)node遍历文件

（3）node读取文件夹

（4）node遍历文件夹

# 12.redux

# 注意事项

1.React-router使用 query参数造成的问题，前一页load的数据不保存

2.为什么路由变化，组件渲染不变化？

3.React的计算和渲染分离？

4.Router获取路由参数方法this.props.location 和match.pramas.id

5.规范化router

6.使用context不能取值问题?

路由组件不能取props问题传入props

<Route exact path="/business/display-chapter/:bookID" component={(props) => <ChapterDisplay {...props} updateBooks={updateBooks}/>}/>

7.如何实现兄弟组件共享数据？ 在最近节点加<context>

8.当一个兄弟节点更改了后台数据后，如何让另一个兄弟节点更新数据？在<content>添加9.一个更新函数updata()（函数内部重新load后台数据），其余兄弟节点更改数据后都调用updata()函数。

10. 抽象table组件，关于操作数据（如删除数据）的函数通过props形式传递给<Table>，在：

render() {

const { data } = this.state;

if (!data) return null

const gropsProps = {

title: 'dispalyBooks',

data: {data},

name: '书名',

itemBaseLinkPath: '/business/display-chapter',

addButtonLinkPath: '/business/add-book',

addButton: '添加书籍',

addButtonRedirect: '/business',

onItemRemoved: (book) => {

// 删除数据的操作

// 删除后更新所有数据

this.props.updateBooks();

},

}

return(

<Layout style={{ padding: '0 24px 24px' }}>

<Breadcrumb style={{ margin: '16px 0' }}>

<Breadcrumb.Item>所有书籍</Breadcrumb.Item>

</Breadcrumb>

<TabelDisplay {...gropsProps}/>

</Layout>

)

}

11.函数使用？

函数定义，什么时候fun(){}，什么时候fun =()=>{}

（1）函数在标签属性上使用场景：

<component onClick={fun}>

<component onClick={fun.bind()}>

<component onClick={() = >fun.()}>

（2）

constructor(props) {

super(props);

this.handleSubmit = this.handleSubmit.bind(this);

this.handleSubmit = this.handleSubmit;

使用场景

}

12.JSON的key问题

Key为变量：

JsonName[keyName] = nextKeys

或者

this.setState({

[keyName]: nextKeys,

});

13． Antd表单初始化默认值流程

方法一：在 getFieldDecorator 上添加initialValue:value

<FormItem>

{getFieldDecorator(`examples[${k}]`, {

rules: [{}],

initialValue:’’,

})(

<Input />

)}

</FormItem>

方法2：表单组件渲染后使用this.form.props.setFiledValue({formOD: defaultValue})设置进入页面后需要渲染的表单数据。

14. form表单提交

15. form表单提交前数据校验

16. 使用Data = Object.assign({},Data, newData)，赋值对象，防止数据丢失

# 网站

react-router <https://reacttraining.com/react-router/core/guides/testing>

react state和props赋值问题：<https://reactjs.org/blog/2018/06/07/you-probably-dont-need-derived-state.html>

react 钩子函数： <https://reactjs.org/docs/react-component.html#static-getderivedstatefromprops>

react数据通信： <https://reactjs.org/blog/2018/03/29/react-v-16-3.html>

react表单：<https://reactjs.org/docs/forms.html>

antiDesign: <https://ant.design/components/form-cn/#components-form-demo-validate-other>

crypto模块：<https://cnodejs.org/topic/504061d7fef591855112bab5>

https://nodejs.org/api/crypto.html

Koa学习网站：https://koa.bootcss.com/#context

<http://www.ruanyifeng.com/blog/2017/08/koa.html>

Koa实现登陆功能实例：https://github.com/dlyt/react-koa-login

Fetch学习网站：

https://developer.mozilla.org/zh-CN/docs/Web/API/Fetch\_API

FormData： <https://developer.mozilla.org/zh-CN/docs/Web/API/FormData>

jsvaScript: https://developer.mozilla.org/zh-CN/docs/Web/JavaScript

Http header content-type: <https://www.cnblogs.com/52fhy/p/5436673.html>

Git KOA, React, promise, fetch, JS的内置对象和String ，Type-Script, NodeJS