react插件篇

1. 在react中进行数据请求：
2. fetch：
3. 检验浏览器是否支持：

if(self.fetch) {

// run my fetch request here

} else {

// do something with XMLHttpRequest?

}

1. 发起请求：

fetch('flowers.jpg')

.then(function(response) {

return response.blob(); //这里是将接受的数据转换为图片

})

.then(function(myBlob) {

var objectURL = URL.createObjectURL(myBlob);

myImage.src = objectURL;

});

1. 自定义请求的参数：

var myHeaders = new Headers();

var myInit = { method: 'GET',

headers: myHeaders,

mode: 'cors',

cache: 'default' };

fetch('flowers.jpg',myInit) //第二个参数是请求的参数

.then(function(response) {

return response.blob();

})

.then(function(myBlob) {

var objectURL = URL.createObjectURL(myBlob);

myImage.src = objectURL;

});

1. 检测请求是否成功:

由于fetch对于404等错误的返回也会当成成功，也就是说如果网不出现问题，只要有返回就会成功，基于此种情况，所以我们要对放回头重status的状态码进行判断，才能知道何时为请求数据成功；

fetch('flowers.jpg').then(function(response) {

if(response.status==”200”) { //对于返回的res中的status进行判断

response.blob().then(function(myBlob) {

var objectURL = URL.createObjectURL(myBlob);

myImage.src = objectURL;

});

} else {

console.log('Network response was not ok.');

}

})

.catch(function(error) {

console.log('There has been a problem with your fetch operation: ' + error.message);

});

1. 数据响应：

[Body.arrayBuffer()](https://developer.mozilla.org/zh-CN/docs/Web/API/Body/arrayBuffer)

使用一个buffer数组来读取 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response)流中的数据，并将bodyUsed状态改为已使用。

[Body.blob()](https://developer.mozilla.org/zh-CN/docs/Web/API/Body/blob)

使用一个[Blob](https://developer.mozilla.org/zh-CN/docs/Web/API/Blob)对象来读取 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response)流中的数据，并将bodyUsed状态改为已使用。

[Body.formData()](https://developer.mozilla.org/zh-CN/docs/Web/API/Body/formData)

使用一个 [FormData](https://developer.mozilla.org/zh-CN/docs/Web/API/FormData) 对象来读取 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response)流中的数据，并将bodyUsed状态改为已使用。

[Body.json()](https://developer.mozilla.org/zh-CN/docs/Web/API/Body/json)

使用一个 [JSON](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/JSON) 对象来读取 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response)流中的数据，并将bodyUsed状态改为已使用。

[Body.text()](https://developer.mozilla.org/zh-CN/docs/Web/API/Body/text)

使用一个[USVString](https://developer.mozilla.org/zh-CN/docs/Web/API/USVString) (文本) 对象来读取 [Response](https://developer.mozilla.org/zh-CN/docs/Web/API/Response)流中的数据，并将bodyUsed状态改为已使用。

1. 使用axios插件：（可以模拟数据）
2. 全局下载axios插件：

cnpm install –save-dev axios

1. 在每一个使用数据请求的文件中引入axios插件进行数据的请求：

axios.get('/user?ID=12345')

  .then(function (response) {

    console.log(response);

  })

  .catch(function (error) {

    console.log(error);

  });

1. 蚂蚁金服ui框架（antd）
2. 移动端使用：
3. 首先下载两个包：

$ npm install antd-mobile --save

$ npm install babel-plugin-import --save-dev

(此babel插件是用来配合es6的module语法做antd组件懒加载的)

1. 配置.babelrc文件：

注：主要配置plugin插件：

特别注意：import插件的style属性一定要设置为true，否则全局样式可能不生效

{

"presets": [

"es2015",

"stage-0",

"react"

],

plugins: [["transform-runtime",

{

"helpers": false,

"polyfill": false,

"regenerator": true,

"moduleName": "babel-runtime"

}], "syntax-dynamic-import", "transform-object-rest-spread", "transform-react-jsx", ["import", {

libraryName: "antd-mobile",

style: true,

}]]

}

1. 配置webpack.config.js:

const svgSpriteDirs = [

require.resolve('antd-mobile').replace(/warn\.js$/, ''), // antd-mobile 内置svg

//path.resolve(\_\_dirname, 'src/my-project-svg-foler'), // 业务代码本地私有 svg 存放目录

];

对于svg文件的loader加载配置：

{

test: /\.svg$/,

loader: 'svg-sprite-loader',

include: svgSpriteDirs,

}

// 使用css的module方式进行css的局部作用域划分

{

test: /\.css$/,

use: ExtractTextPlugin.extract({

fallback: "style-loader",

use: "css-loader?module"

}),

exclude: /node\_modules/

},

// 对于antd的modules中的css不可以使用css的module方式加载

{

test: /\.css$/,

use: ExtractTextPlugin.extract({

fallback: "style-loader",

use: "css-loader"

}),

include: /node\_modules/

},

// 为antd单独添加less加载器

{

test: /\.less$/,

use: [{

loader: "style-loader"

}, {

loader: "css-loader"

}, {

loader: "less-loader"

}]

},

1. 在全局中中引入公共样式:(如：main.js中)

import 'antd-mobile/dist/antd-mobile.css';

1. 在组件中使用antd的组件：

import { Button } from 'antd-mobile';

return (

<div>

<Button className="btn" type="primary">primary button</Button>

</div>

);

1. 组件的懒加载两种方式：

import { Button } from 'antd-mobile';

（此种方式必须结合babel的插件import）

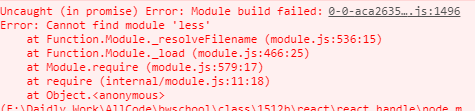
import 'antd-mobile/lib/button/style';

import Button from 'antd-mobile/lib/button'

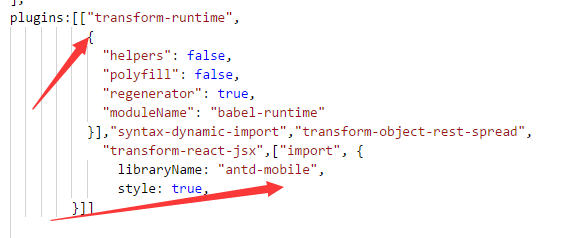
(此种方式不需要借助任何插件，但是使用此种方式，必须同时引入相应组件的样式)

注：报错处理

1、注意less加载器的添加，一定注意安装less-loader时，一同把less安装了，如下错误因为没有安装less  
cnpm install –save-dev less



2、如果报webpack等一系列错误，则需要更改如下配置



1. pc端的使用：
2. 基本配置如上一样：
3. 注意babelrc文件：

将：libraryName变成antd

{

"presets": [

"es2015",

"stage-0",

"react"

],

"plugins": [

"transform-runtime",

"transform-react-jsx",

["import", { libraryName: "antd", style: true }]]

}

1. 其他的组件懒加载等问题，如移动端使用一样；