网页版简易计算器

-----简单的设计开发流程及问题解析

一：学习Webpack

1>>>>

基本配置及运行方式：

(1)安装Webpack

通过 npm install webpack 安装

(2)在主界面通过script标签引入打包后的bundle.js文件

(3)在入口entry.js文件中写入代码，同时还可以通过require来导入别的模块

reruire(./module.js) //module.js文件中可以写入模块的代码

(4)运行

1>, 通过命令运行

1>> 通过 webpack entry.js bundle.js 编译entry.js并打包到bundle.js中

点击运行index.html页面就行。

2>> 当导入css模块时，我们需要在入口文件中添加代码

require("!style-loader!css-loader!./style.css")其中style.css为独立的css模块

注： 再引入css模块时，必须安装css-loader和style-loader

npm install css-loader style-loader

简化导入css代码：

我们可以将entry.js中的require("!style-loader!css-loader!./style.css")代码修改为require("./style.css")

在执行 webpack entry.js bundle.js --module-bind "css=style-loader!css-loader"

点击运行index.html页面就行。

2>, 通过配置文件运行

1>>在根目录创建package.json来添加webpack需要的依赖

在package.json文件完成后，运行一次npm install

2>>然后创建配置文件webpack.config.js

3>>运行webpack命令，然后点击index.html

2>>>>

插件的使用：

在配置文件中添加plugins

例如：

plugins: [

new webpack.BannerPlugin('This file is created by 刘文亮')

]

二：根目录创建package.json来添加webpack需要的依赖

1>>>>

在配置文件webpack.config.js中添加

module.exports = {

entry: "./src/index.js", ---入口文件---

output: { ---输出文件---

path: 'dist/', ---路 径---

filename: "bundle.js" ---打包文件---

},

module: { ---解析加载---

loaders: [

{ test: /\.js$/, loader: 'babel-loader' },

{ test: /\.less$/, loader: "style-loader!css-loader!less-loader" }

]

}

};

三：需要安装的插件

正常运行该包时所需要的依赖

"dependencies": {

"babel-preset-es2015": "^6.22.0",

"babel-preset-react": "^6.23.0",

"less": "^2.7.2",

"less-loader": "^3.0.0",

"node-less": "^1.0.0",

"react": "^15.4.2",

"react-dom": "^15.4.2"

},

开发阶段完成集成测试等功能模块依赖

"devDependencies": {

"babel-core": "^6.23.1",

"babel-loader": "^6.4.0",

"css-loader": "^0.26.4",

"jasmine-core": "^2.5.2",

"less-loader": "^3.0.0",

"node-libs-browser": "^2.0.0",

"style-loader": "^0.13.2",

"webpack": "^2.2.1"

}

注：

当界面无法显示出来时,我们需要安装 react-dom;

同时必须在入口文件中添加 import ReactDOM from 'react-dom';

四：配置解析转换文件

新建.babelrc文件夹，在文件里设置参考规范，

例如：{

"presets":["es2015", "react"]

}

如果没有此文件，在终端窗口会报错；

注：我们需要安装"babel-preset-es2015","babel-preset-react"

或者在webpack.config.js配置文件中添加

query:{

'presets': ['react', 'es2015']

}

五：编写CSS样式文件，规范整体样式，设计出基础界面

1>>>>

使用Less来完成样式的设置，&指示当前选择器。

2>>>>

使用CSS3的box-shadow来对按键框添加一个或多个阴影，6个参数分别为：

h-shadow 必需。水平阴影的位置，允许负值。

v-shadow 必需。垂直阴影的位置，允许负值。

blur 可选。模糊距离。

spread 可选。阴影的尺寸。

color 可选。阴影的颜色。

inset 可选。将外部阴影 (outset) 改为内部阴影。

六：设计 Calculator.js,Button.js,Parser.js模块的代码

1>>>> 编写index.js文件，主要是将主组件Calculator插入指定的calculator-container节点。

在入口文件里利用import导入要用到的js文件和需要的库。

利用ReactDOM.render 将模板转为 HTML 语言，将主组件Calculator插入指定的calculator-container节点。

2>>>> 编写Calculator.js文件，实现主界面，即计算结果显示屏。

在Calculator.js文件中我们需要通过export来导出Calculator组件到index.js

首先必须通过 import导入需要用到的依赖的js文件，import Button from './Button'和import Parser from './Parser'

通过getInitialState()来初始化screen和keyboard的显示和按键状态值，getDefaultProps()设定默认值

updateScreen()方法更新屏幕值的变化

通过onPress()方法实现按键值到屏幕传递，同时调用updateScreen()来更新变化

在componentDidMount()方法中使用this.parser = new Parser()来实现数据的解析，同时调用updateScreen()来更新变化，使其在界面经过render渲染后执行

3>>>> 编写Button.js文件代码，实现每个按钮。

这里的按钮实现需要通过React的渲染来实现，所以之前按html的基础实现是不行的，

在这里我们需要先画好键盘布局，在通过定位布局来实现界面的初始化。再通过Less来实现布局样式设置。

我们需要给Button注册一个click监听事件，当用户点击时，会通知Calculator输入的内容是什么。

例如：

当输入的为'+', '-', '×', '/', '±'中任何一个操作符时，我们必须将其前后输入的数字分隔开，

分隔成两个运算数字，需要定一个方法来实现。

注：this.props表示那些一旦定义，就不在改变的特性；而this.state是会随着用户互动而产生变化的特性。

4>>>> 对于两次输入的数字，根据输入的字符是否为操作符来分隔

5>>>> Parser.js 用来解析用户的输入流，产生结果，本质是一个状态机

Parser.js 是计算器的核心算法，提供了两个接口

.take(letter),将用户每次点击的按钮输入到解析器中;

.getScreen(),获取屏幕上应该显示的值。

在用户点击按钮时，输入letter,然后调用 getScreen(),通过setState来更新显示

6>>>> 逻辑处理

S1：初始状态，求值操作或清空操作都会回到初始状态(this.state.RESET)

S2：左操作数(this.state.INPUT.LEFT.VALUE)

S3：右操作数(this.state.INPUT.RIGHT.VALUE)

S4：出错，除以0导致(this.state.ERROR)

根据画出的状态图来编写核心逻辑代码

我们定义一个take()和一个operate()函数来对逻辑进行编写

----take()函数用来捡测每次输入的字符，利用正则表达式来匹配，将其输入的合法字符存入数组

这里需要对输入的数字或'.'进行分析处理

"char != '.' || this.stack.indexOf(char) == -1"这个判断需要特别注意，

分析有两点原因：

a：如果当前的输入为'.',且之前没有出现过则存入数组;b：如果存入的不是'.',是数字，则直接存入数组

----operate()函数用来对操作符进行相关计算以及小数点的处理