我从这个项目中学到的：

package.json

最朴素的 webpack-dev-server 用法默认端口 8080

**"start"**: **"webpack"**,  
**"dev"**: **"webpack-dev-server"**

webpack.config.js

**devServer**: {  
 **contentBase**: **'./public'**,  
 **color**: **true**,  
 **historyApiFallback**: **true**,  
 **inline**: **true**}

这里public下面放着 index.html 是这个应用的html模版

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

{  
 **test**: /\.css$/,  
 **loader**: **'style!css?modules!postcss'**},

**postcss**: [  
 *require*(**'autoprefixer'**)  
]

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

.babelrc

{  
 **"presets"**: [**"react"**,**"es2015"**]  
}

没有这个文件， babel无法正常工作， es6 语法会被报错。

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

前端部分

这个项目使用了bootstrap

<**link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"**>

多个类直接放在一个双引号下：

<**span className="glyphicon glyphicon-chevron-up"**> </**span**>

前端逻辑

利用 media 组建构建一个问题条目 QuestionListItem， 利用一个 表单构建 QuestionForm ， 最后汇合在一个 QuestionApp 当中。

数据直接通过json 文件引入。

{  
 **"id"**: 1,  
 **"title"**:**"产品经理与程序员矛盾的本质是什么？"**,  
 **"description"**:**"理性探讨，请勿。。。。。。。。**

**"voteCount"**: 30  
}

在具体的每一条问题当中进行展示

<**span className="vote-count"**>{ **this**.**props**.**value**.**voteCount** }</**span**>

<**h4 className="media-heading"**>{ **this**.**props**.**value**.**title** }</**h4**>  
<**p**>{ **this**.**props**.**value**.**description** }</**p**>

由于涉及到父子元素之间的，所以开始必须要有

constructor(props){  
 **super**(props);  
}

继承父类的 props state

子类必须在constructor方法中调用super方法，否则新建实例时会报错。这是因为子类没有自己的this对象，而是继承父类的this对象，然后对其进行加工。如果不调用super方法，子类就得不到this对象。

**onClick=**{**this**.addHandleClick.bind(**this**)}

<**button className="btn btn-default" onClick= this**.reduceHandleClick.bind(**this**) }>

addHandleClick(e) {  
 **this**.**props**.vote(1);  
 e.preventDefault();

//不要执行与事件关联的默认动作（如果存在这样的动作）  
 e.stopPropagation();

//不要冒泡  
}  
reduceHandleClick(e) {  
 **this**.**props**.vote(-1);  
 e.preventDefault();  
 e.stopPropagation();  
}

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

form表单一级

重点的是采集文本框中的信息，生成新的问题对象。

handleSubmit(e) {  
 **let** newQuestion = {  
 **title**: **this**.**refs**.qtitle.**value**,  
 **description**: **this**.**refs**.qdesc.**value**,  
 **voteCount**: 0  
 };  
 **this**.**refs**.qtitle.**value** = **''**;  
 **this**.**refs**.qdesc.**value** = **''**;  
 **if**(!newQuestion.**title**) **return**;  
 **this**.**props**.saveQuestion(newQuestion);  
 e.preventDefault();  
 e.stopPropagation();  
}

<**button className="btn btn-success pull-right" onClick=**{ **this**.handleSubmit.bind(**this**)}>提交</**button**>

此外独特的一点是该表单的隐藏与现实。

首先有一个toggle按键能够在任何时刻显示和隐藏该表单，此外就是表单内的取消键，取消新建问题，隐藏表单；另一个是提交，不仅新建一个问题，同时也能隐藏表单。

<**div …… style=**{ styleObj }>

最外层div

**var** styleObj = {  
 **"display"**: **this**.**props**.shown ? **'block'** : **'none'**};

<**button …… onClick=**{ **this**.handleSubmit.bind(**this**)}>提交</**button**>  
<**button …… onClick=**{ **this**.**props**.hideQuestionForm }>取消</**button**>

注意在handleSubmit 函数的最后

**this**.**props**.saveQuestion(newQuestion);

**this**.**props**.hideQuestionForm

saveQuestion hideQuestionForm 都是父类中定义的两个函数，这样就实现了通过props利用父类函数来对子类进行操作。

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

最高层 App

<**QuestionForm shown=**{**this**.**state**.**showQuestionForm**} **saveQuestion=**{**this**.saveQuestion.bind(**this**)} **hideQuestionForm=**{ **this**.hideQuestionForm.bind(**this**)}/>  
<**div id="questions"**>  
 {QuestionLists}  
</**div**>

从中可以看出 属性就是 函数

**shown=**{**this**.**state**.**showQuestionForm**}

**"display"**: **this**.**props**.shown ? **'block'** : **'none'**

通过改变 showQuestionForm 这个state就能实现对form显示的控制

能够改变 该state 值的还有最外层的toggle按键。

<**button 。。。 onClick=**{**this**.handleClick.bind(**this**)}>添加问题</**button**>

hideQuestionForm() {  
 **this**.setState({  
 **showQuestionForm**: **false** })  
}

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

handleClick() {  
 **console**.log(**'click'**);  
 **this**.setState({  
 **showQuestionForm**: !**this**.**state**.**showQuestionForm** });  
}

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

数据读入

**import** QuestionsData **from '../data/QuestionsData.json'**;

constructor(props) {  
 **super**(props);  
 **this**.**state** = {  
 **QuestionsData**: QuestionsData,  
 **showQuestionForm**: **false** }  
}

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

saveQuestion(newQuestion) {  
 newQuestion.**id** = **this**.**state**.**QuestionsData**.**length**+1;  
 **let** QuestionsData = **this**.**state**.**QuestionsData**.concat(newQuestion);

//concat 连接两个数组  
 **this**.setState({  
 **QuestionsData**: QuestionsData,  
 **showQuestionForm**:**false** })  
}

/\* **this**.**props**.saveQuestion(newQuestion); \*/

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

**let** QuestionLists = [],  
 QuestionsData = **this**.**state**.**QuestionsData**;  
QuestionsData.forEach((value,key) => {  
 QuestionLists.push(<**QuestionListItem key=**{key} **value=**{value} **vote=**{**this**.vote(key)} />)  
});

/\* **this**.**props**.vote(1); Item 孙子类\*/

**vote=**{**this**.vote(key)} 爷爷类 这里的key 也就是index 与该问题一一对应，

孙子类里面的1 是指operation。

vote(index) {  
 **return** (operation) => {  
 **let** QuestionsData = **this**.**state**.**QuestionsData**;  
 QuestionsData[index].**voteCount** += operation;  
 QuestionsData = **this**.sortVote(QuestionsData);  
 **this**.setState({  
 **QuestionsData**: QuestionsData  
 })  
 }  
}

//这里直接返回一个函数

// vote(1)

// function(operation) {

**let** QuestionsData = **this**.**state**.**QuestionsData**;  
 QuestionsData[index].**voteCount** += operation;  
 QuestionsData = **this**.sortVote(QuestionsData);  
 **this**.setState({  
 **QuestionsData**: QuestionsData  
 })

}

sortVote(QuestionsData) {  
 QuestionsData.sort((a,b)=>{  
 **return** b.**voteCount** - a.**voteCount**;  
 })  
 **return** QuestionsData;  
}

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

套路

**父类state 在父级逻辑中改变后赋值给子类props**

**子类props 直接指向父类函数**

<**QuestionForm shown=**{**this**.**state**.**showQuestionForm**} **saveQuestion=**{**this**.saveQuestion.bind(**this**)} **hideQuestionForm=**{ **this**.hideQuestionForm.bind(**this**)}/>