# 项目总结

1. jQuery判断数据类型

$.type()

1. Ajax请求加上error状态的操作。
2. self若没有指定指向时，默认指向window。
3. React组件类中可以添加静态方法，但该静态属性可能在控制台无法通过打印this验证，但可以通过打印this.属性名验证。
4. React组件类中，函数中的this指向这个组件，静态属性值中的this指向undefined。
5. Obj1={fn1:function(event){}},则fn1是obj的方法。当不使用call、apply、bind改变this指向时，fn1函数不可能被其他对象调用，fn1函数中的this总是指向Obj1。即方法中的this总是指向该方法所属于的对象。而fn1函数的event事件对象指向触发该函数的事件本身。总之，**event.target与this无关**。
6. 使用委托的方式，父组件向子组件传递一个函数属性值，若将子组件作为该函数的参数，此时子组件调用该函数时，可将父组件自身作为一个属性值赋值给子组件(**widget.options.form = this**)。即：子组件自身作为函数的参数传递给父组件，可以与event作为函数的参数传递给父组件(React.docx p46)进行对比。
7. ...展开的使用方法

var attr={

type:"text",

disabled:true,

value:"文本框",

style:{

**opacity:"0.5",**

**height:"30px"**

}

}

ReactDOM.render(

<input {...attr}/>,

document.getElementById('container')

)

注释:①展开(...)可以展开多层。②展开可以展开style样式。③展开style样式时数值CSS属性值可加引号(原生style样式不可加引号否则不生效)。④展开style样式时标点为冒逗而不是冒分。

1. 组件的render函数中不可以修改自身状态，因为render函数中不可以执行update。

Cannot update during an existing state transition (such as within `render`). **Render methods should be a pure function of props and state.**

不可以在子组件的render函数中修改父组件的state状态，因为当子组件rende渲染时，父组件的render函数仍未完成。

1. 作用域原理:子函数有权访问父函数的参数、局部变量和其他子函数。在这句话中，函数的父子关系为**函数体**的父子关系，而不是**函数调用**的父子关系。

示例1:

a=1;

function demo1(){

a++;

alert(a);

}

demo1(); //弹出1;

示例2:

function demo1(){

a++;

alert(a);

}

function demo2(){

var a=1;

demo1();

}

demo2(); //a is not defined

总结:函数的位置是函数体的位置，而不是函数调用时的位置。

1. this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，实际上this的最终指向的是那个调用它的对象。

一个对象的方法里包裹一个函数体，此时该函数体内部的this④指向window，需要使用bind改变this的指向。一个对象的方法里包裹一个函数调用，函数调用时传的参数里的this②指向同一层作用域的this，即这个对象。一个对象的方法里包裹一个函数调用，该函数调用里包裹一个函数体参数，该函数体参数内部的this③指向window。

setValue: function(value) {

var this①.str = "";

this.loadData(this②.str, function() {

selectedIndex = this③.getIndexInData(value, false);

})

function (){this④}

}

注释:若这个对象调用setValue方法，则①②两个this指向这个对象，③④两个this指向window。

注意:函数体与函数调用的区别。

1. React组件更新的条件

①父组件状态变化引起子组件属性变化。②子组件内部状态变化。

实例:在控件是否可用以及控件是否可见的实例中，直接通过委托的方式设置父组件的状态变化，进而设置子组件和父组件是否可用以及是否可见。

1. jQuery判断对象是否是函数。

$.isFunction();

1. var a = b\*c;

当这一行代码执行完成之后，a不随b与c值的变化而变化。

1. JavaScript中的隐式转化

[1]== 1 == "1"

1. JavaScript中的容错处理

if(value==""||value==null||value==undefined){

return;

}

1. 字符串(string)方法

if(a.startsWith(b)) //判断字符串a 是不是以字符串b开头

if(a.endsWith(b)) //判断字符串a 是不是以字符串b结尾

string.substring(起始底,起始底+截个数)

1. 函数中的逻辑如果过于复杂，应该拆分逻辑为单独的函数。
2. jQuery each方法跳出循环的方法

return false;跳出所有循环；相当于 javascript 中的 break 效果。

return true;跳出当前循环，进入下一个循环；相当于 javascript 中的 continue 效果。

注释:在jQuery的map方法中return false和return true;并无此效果。

1. $.map()方法与$.each()方法的参数顺序不同

$.map(items, function(item, index){}

$.each(items, function (index, item) {}

1. 为方便组件之间通讯，一个项目应只有一个顶级组件，从该级组件中获取所有数据。如果一个项目有两个顶级组件，那么这两个顶级组件之间的数据传递将不再符合React数据流。
2. 正则表达式

amazeUIFormat = amazeUIFormat.replace(/h/g, "H");

1. 字体图标制作

<http://www.iconfont.cn/>

1. input文本框，若type等于"text"，则type可以省略。
2. 字体图标样式

div.close:after {

content: "\E617";

**display: block;**

width: 4rem;

height: 4rem;

font-size: 1.2rem;

}

1. Ajax的post请求

Chrome调试工具中

Query String Parameters和Form Date中数据区别

举例:Query String Parameters

{op:widget,widgetname:comboBox0,sessionID:3772}

Form Date

{dataFilter:[4e2d],startIndex:0,limitIndex:500}

1. Query String Parameters中的数据是放在url中的数据。

Form Date中的数据是放在data中的数据。

1. 只要数据相同，写在不同位置不影响接收到的数据。
2. 下拉复选框搜索功能

用selectedIndexes状态存储被选项的序号，selectedIndexes只有在被选项点击时才会改变。当搜索内容并无匹配项时，this.state.data为空数组([])，而此时this.selectedIndexes仍存储有匹配项顺序，故此时无法从data中获取被选项的值，故报错。

1. 数组的indexOf()方法

var selectedItem = e.target;

var items = $(selectedItem).parent().children("a");

var selectedIndex = items.indexOf(selectedItem);

1. 返回该元素在数组中的下标值。
2. 可以操作类数组。
3. Key值需要唯一，并与当前节点特征对应，否则渲染时可能出现未知问题。

在循环中生成元素，并设置index为当前节点的key。此时，若元素顺序和个数发生变化，则容易引发渲染错误。

1. For循环中变量声明

for(var i = 0, len = items.length; i < len; i ++) {循环体}

1. React中的key值在组件内无法获取，key值可以是中文，只要不重复。
2. 下拉框、下拉复选框、下拉树this.props.options.controlAttr

若不设置控件值，则{value : ""}。

若设置控件值，则对于下拉框、下拉复选框来说传递完整的值(一层)，对于下拉树来说，传递第一层的值{value : "值" , data : {Array[n]}}。

1. var 子对象 = Object.create(父对象，子对象的自有属性) 。

var treeNodeModel = Object.create(TreeNodeModel);

1. $.each(this.nodeModels,function(index,nodeModel){})

$.map(this.state.dataArray, function(data) {})

1. 对象(Object)的初始值可设置为空(null)，字符串的初始值可设置为空字符串("")。
2. 在React项目中引入jQuery，jQuery库的作用是使用其选择器和进行数据类型操作。禁止在React项目中使用jQuery操作DOM节点，例如:操作class、CSS样式、HTML属性等等。
3. 全等判断

0 == ""为trur, 0 === ""为false

"" == ""为true, "" === ""为true

若不是为了兼容false的情况，判断是否等于空字符串，应使用===。

1. touch事件相关
2. $("sel").scrollTop("数值");设置上卷高度生效的前提为$("sel")设置高度，并且设置overflow: scroll;或者overflow: auto;或者overflow: hidden;
3. 当设置overflow: hidden时，移动端默认的touch事件不生效，PC端默认的鼠标滚轮事件不生效，但此时可以通过重写touch事件，自定义移动端touch事件函数。

即:若y方向overflow: hidden，则无竖向滚动条，默认touch事件不生效，默认scroll事件不生效，但可以重写touch事件。

1. PC端文本选中与移动端长按复制

(1) PC端设置文本不可选中且移动端长按不可以复制

-moz-user-select: none;

-webkit-user-select: none;

-ms-user-select: none;

user-select: none;

1. PC端设置文本可以选中但移动端长按不可以复制

onTouchStart: function(e) {

e.preventDefault();

}

总结:①touchstart事件中使用preventDefault() => pc端可以选中，但是移动端不可以长按复制。②-webkit-user-select:none; =>pc端不可以选中，并且移动端不可以长按复制。

1. 移动端浏览器兼容
2. document的scroll事件在Safari浏览器中不会冒泡。
3. document/window的scroll事件在Android系统中只有当document出现滚动条的时候才会触发，而在苹果系统中即使没有滚动条滑动也会触发。

解决方法: 使用touchmove事件替代scroll事件。

1. 在Safari浏览器中，input文本框的获取光标时会触发两次document的scroll事件。

解决方法:使用touchmove事件替代scroll事件。

1. touchmove事件在浏览器中会冒泡。

解决方法: 阻止事件冒泡。

1. FastClick可能导致文本框的onClick事件在移动端触发两次。

//fastclick

(function () {

window.addEventListener('load', function () {

new FastClick(document.body);

}, false);

})();

1. onTouchMove事件若直接写在DOM元素上，Android浏览器不识别

解决办法: 使用JQuery绑定事件，兼容性更好

$('#div2').bind("touchmove", function(e){

if(e && e.stopPropagation){

e.stopPropagation();

}

})

1. 取消touchmove事件的事件冒泡，会导致滑动时触发点击事件。
2. 继承的方式

$.extend(true, {}, this.formOptions);

true表示深拷贝。

1. 函数包函数，子函数的返回值不等于父函数的返回值

getItemGroups: function(items) {

var self = this;

var itemGroups = [];

debugger;

$.map(items, function(item, index) {

if(item.type === "absolute" && $.isArray(item.items)) {

self.getItemGroups(item.items);

} else {

self.addItem2Groups(item, itemGroups);

}

});

return itemGroups;

}

在本示例中，如果执行getItemGroups子函数，返回值为空数组。

1. 数组添加DOM元素，其实添加的是React对象

**var** items = [];

items.push(  
 <li key={index}>  
 {FormLayout.createLayoutElement(elementOptions, index, options, widgetName, self.props.hasFormBackground)}  
 </li>  
);

即:

items.push(React.createElement(

"li",

{ key: index },

FormLayout.createLayoutElement(elementOptions, index, options, widgetName, self.props.hasFormBackground)));

正确写法:

iconItems.push(<i key={index} className={icon}></i>);

错误写法:

arrowElement = [  
 <div **class**="tab-left-arrow"></div>  
];

1. 很多递归都可以使用while循环替代，能够使用while循环就不要使用递归！
2. 文字垂直居中

父div:

display: table;

width: 300px;

height: 300px;

子div:

display: table-cell;

vertical-align: middle;

text-align: center;

注释:①此时父若不设置宽高，则收缩至子div内容。④子div高以内容为准，宽扩展到父的宽度。③子div即使设置宽高也不生效。

1. 父级元素的touchstart和touchend阻止事件默认行为之后，内层元素的onclick事件在安卓手机上不生效。

解决办法:①移动端使用tap事件替代click事件。

②父级元素的touchstart和touchend事件不阻止默认行为。

1. Object为引用数据类型，在React中对其进行修改要注意是在原值基础上进行修改，还是创建新值。
2. 在React中，不同值有以下几种存在形式:

①state状态值。 ②组件的属性值。③通过函数获取的值。

(1) 哪些值存储为state的状态值

与视图一一对应的值。

(2) 哪些值存储为组件的属性值

常量以及值不轻易改变的变量。

特点:只有手动改变的时候，值才会变化。

1. 哪些值通过函数获取

计算复杂的值。

1. 移动端浏览器后退各个浏览器表现不一

以iOS为例

safari： 后退时上一个页面不会重新请求页面本体，不会重新执行js，页面完全保持上次离开时的样子；

chrome：后退时不会重新请求页面本体，但是会重新执行js，body尝试滚动到上次的位置；

微信iOS：后退时不会重新请求页面本体，但是会重新执行js，body尝试滚动到上次的位置，同时触发window.onpopstate事件；

UIWebView：后退时不会重新请求页面本体，但是会重新执行js，body尝试滚动到上次的位置，同时触发window.onpopstate事件；

WKWebView: 后退时上一个页面不会重新请求页面本体，不会重新执行js，页面完全保持上次离开时的样子。

1. overflow:auto的生效条件

宽高都是固定值时，overflow:auto才会生效。

问题:若设置的只是maxHeight，则可能不生效。

解决方法一: maxHeight修改为height

1. Table中的几个问题

问题:设置table中单元格每一行的高度，当文字高度太大时，垂直截断。

给tr或td添加高度，不可行。td的实际高度仍然可以被td中文字撑开。

解决方法:在td中添加div元素，在div中放置内容，设置div的高度。

问题:如何让div中文字在td中垂直居中。

解决方法:不要设置div的height为td的高度，而是设置max-height。

问题:当td中文字字体太大时，是否设置td的border影响td的实际高度。

解决办法:td中div的max-height设置值要比td的高度小。

举例: var textContainerStyle = {maxHeight: options.height - 3};

注意:适合使用max-height的场景不要使用height。

1. 使用React时，如果不加上type类型(<script type="text/babel">)，则报错:Uncaught SyntaxError: Unexpected token < .
2. 使用React的错误

React.createClass = ({ })

正确写法:eact.createClass({})

1. 设置一个div的高度铺满整个屏幕的一种方法

html, body, 外层div,此div {height:100%};

从外到内的每一个盒子模型的高度都设置为100%。

1. React类中静态属性的特点

var ToolTip = React.createClass({

screenHeight : document.body.clientHeight,...

})

screenHeight只在组件类加载的时候计算一次，不再更新。

例如:执行this.setState()方法重新render，若render中使用this.screenHeight, 则该值可能不正确。

1. 使用setInterval()和setTimeout()定时器时，新添加定时器之前需要清除上一个定时器。

function() {

if(this.hideTimer) {

clearTimeout(this.hideTimer);

}

this.hideTimer = setTimeout(function() {

$("div#h5-hint")[0].style.display = "none";

}, 2000);

}

1. 利用闭包实现ajax中success函数预处理

options.success = (function(successClone) {

return function (result) {

var jsonData = BaseUtils.jsonDecode(result);

if (jsonData.exception) {

//TODO 错误信息处理

}

} else {

successClone(result);

}

}

})(options.success.clone());

1. 外层组件注册内层组件的事件

示例1:两个Outer组件，两个Inner组件，两个fireEvent函数

var Inner = React.createClass({

render: function () {

return (

<h1>Inner</h1>

)

},

fireEvent:function () {

alert(this.props.num);

},

componentDidMount: function () {

this.props.registerAPI({

fnName:"fireEvent", fn:this.fireEvent

})

}

})

var Outer = React.createClass({

render:function(){

return (

<div onClick={this.handleClick}>

<Inner registerAPI={this.registerAPI} num={this.props.num}/>

</div>

)

},

registerAPI: function (obj) {

// this.constructor.prototype[obj.fnName] = obj.fn;

//Outer组件类的原型注册Inner组件的fireEvent事件。

//Outer生成的两个Outer组件共用同一个函数。

//同一个Outer组件类注册两次不同的函数，后者覆盖前者。

//点击弹出2, 2

//Outer组件类生成的两个Outer组件注册Inner组件的fireEvent事件，两个Outer组件不共用fireEvent函数。

//两个Outer组件分别注册两个不同的函数，互不干扰。

this[obj.fnName] = obj.fn;

//点击分别弹出1, 2

},

handleClick: function () {

debugger;

this.fireEvent();

}

})

ReactDOM.render(

<div>

<Outer num="1"/>

<Outer num="2"/>

</div>,

document.body);

示例2:一个Outer组件，两个Inner组件，两个fireEvent函数

var Inner = React.createClass({

render: function () {

return (

<h1>Inner</h1>

)

},

fireEvent:function () {

alert(this.props.num);

},

componentDidMount: function () {

this.props.registerAPI({

fnName:"fireEvent",fn:this.fireEvent

})

}

})

var Outer = React.createClass({

render:function(){

return (

<div onClick={this.handleClick}>

<Inner registerAPI={this.registerAPI} num="1"/>

<Inner registerAPI={this.registerAPI} num="2"/>

</div>

)

},

registerAPI: function (obj) {

// this.constructor.prototype[obj.fnName] = obj.fn;

//点击弹出2,2

//Outer组件类的原型注册Inner组件的fireEvent事件，注册两次。

//同一个Outer组件类注册两次不同的函数，后者覆盖前者。

//点击弹出2, 2

//Outer组件类生成的一个Outer组件注册Inner组件的fireEvent事件。

//同一个Outer组件注册两次fireEvent函数，后者覆盖前者。

this[obj.fnName] = obj.fn;

//点击弹出2,2

},

handleClick: function () {

this.fireEvent();

}

})

ReactDOM.render(<Outer />,document.body);

总结: fireEvent中的this恒指向Inner组件类生成的那个组件实例，即调用fireEvent函数的组件实例。

1. APP与H5通讯

jsBridge.registerHandle(函数名, 函数体)——注册函数。

jsBridge.callHandler(函数名,参数,回调)——调用函数。

不管在h5还是app注册的函数，在h5和app都可以调用。

js调原生的方法jsBridge.callHandler(函数1，参数， 回调);

在h5中使用jsBridge.callHandler(函数1, 参数, 回调2)方法调用app中定义的函数1时，函数1在app中执行，并且数据以参数的形式从h5传递给app，回调2在h5执行。函数1执行过程与其他h5代码异步，回调2可能在许多h5代码之后执行。

注意:APP中注册的函数只能在H5中调用，H5中注册的函数只能在APP中调用。

1. 在React中，当兄弟节点的key相同时，只会渲染第一个。
2. 数组中的React组件没有key值的警告

Warning: Each child in an array or iterator should have a unique "key" prop. Check the render method of Login.

1. 当父元素的height值为默认的auto时，子元素的高度设置为100%不生效。
2. Width的默认值为auto(浏览器可计算出实际的宽度)， 而不是100%。

width属性值为auto时，横纵屏幕切换，div的宽度可能不变。

width属性值为100%时，横纵屏切换，div的宽度总是等于父级元素的宽度。

1. 当html和body的高度都为100%而不是固定像素时，可能导致某些浏览器(魅蓝自带浏览器)滚动失效。
2. react 事件和原生事件混用有的时候不能阻止冒泡。

解决方法:使用react事件替代原生事件。

1. Less中的媒体查询

@baseWidth: 375px;

@bps: 400px, 414px, 480px;

.loop(@i: 1) when (@i <= length(@bps)) {

@bp: extract(@bps, @i);

@font: @bp / @baseWidth \* @baseFont;

@media only screen and (min-width: @bp){

html {

font-size: @font;

}

}

.loop((@i + 1));

};

.loop;

1. call方法可以不传参数，含义为调用该函数。

var fn = function(){alert(1)};

fn.call();相当于调用该函数

1. 正则表达式使用变量

var rr="aa"

var reg=new RegExp(rr,"g");

var s="rr111aa1111aa1111aarr";

var x=s.replace(reg,"AA");

alert(x);

1. React与ReactNative点击事件的对比

在React中，所有div、span、input等html元素以及所有<Component />封装的组件都可以添加onClick事件。

在ReactNative中，只有TouchableOpacity，TouchableHighlight，TouchableNativeFeedback，TouchableWithoutFeedback等个别组件以及这些组件为最外层元素封装的<Component />支持onPress事件。