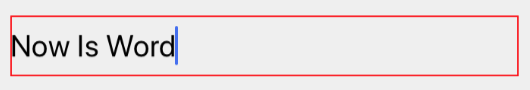
TextInput组件：

1. autoCapitalize(capitalize: 用大写字母写的意思)

是一个枚举类型。可选值：characters，words， sentences，none

autoCapitalize="characters" 

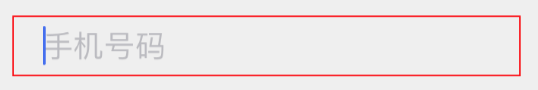
autoCapitalize="words"



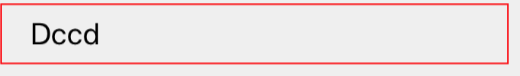
autoCapitalize="sentences"



1. **autoFocus：true时设置组件挂载后获得焦点，默认false**



1. caretHidden: true隐藏光标（caret：补字号，插入符号）



1. defaultValue和 placeholder 的表现上的区别就是颜色不一样，然后placeholder的文本在获得光标有输入后就会消失。defaultValue则不会。
2. placeholderTextColor='#0000ff' 设置占位符字体的颜色

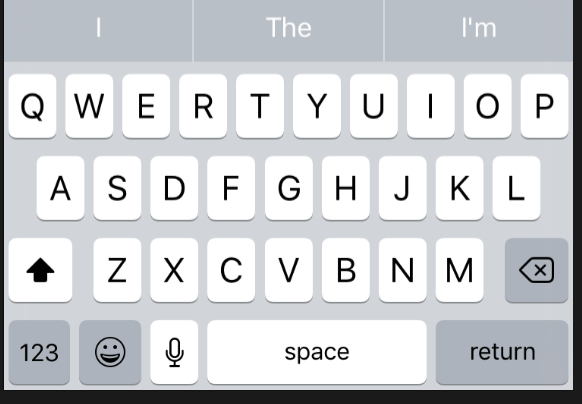


1. editable: true 为可编辑，false不可编辑
2. keyboardType: 枚举类型，设置弹出键盘的类型

numeric： 纯数字键盘



default：英文键盘



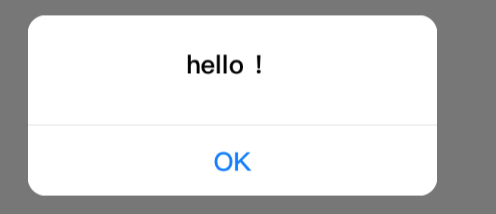
phone-pad:



1. maxLength: 限制输入字符的长度

maxLength={10} 只能输入10个字符

1. multiline： true时可以输入多行文本
2. onBlur={() => {Alert.alert('hello !')}} 文本框失去焦点的时候执行的函数



1. onChange={() => {Alert.alert("content changed")}}

当文本框中的内容发生改变(不用失去焦点，有输入就会执行)的时候执行的函数，

1. onChangeText={(s) => {Alert.alert(s)}}

和onChange函数基本相同，只是还可以实时获取到当前文本框中的内容。

1. onEndEditing={() => {Alert.alert("njdsknv")}}

是当所有的文本输入完成之后按下回车键之类的操作时执行的函数。

1. onLayout={() => {Alert.alert("组件挂载了")}}

在组件挂载或者布局发生变化的时候偶调用。



onLayout={() => {Alert.alert("组件高度变了")}}

state设置：

this.state = {

length: 90,

};

高度样式：

style={{ height: this.state.length }

点击改变state：

<TouchableOpacity onPress={() => (this.setState({length: 40}))}>

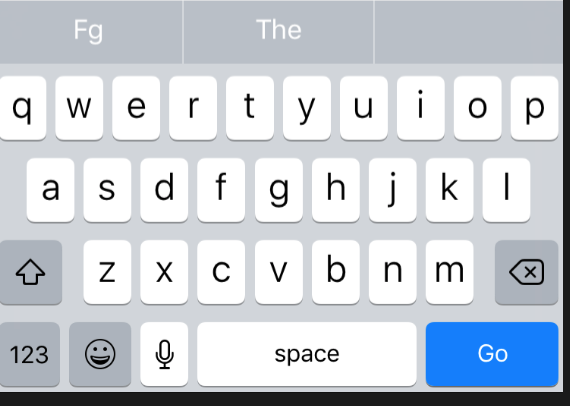
<Text>点击</Text>

</TouchableOpacity>



1. onSubmitEditing：只有在软键盘的 确定/提交 按钮按下之后才会触发这个函数。和onEndEdiiting不同，如果是焦点离开当前输入框也会触发onEndEditing这个函数。
2. returnKeyType 设置确定按钮显示的内容

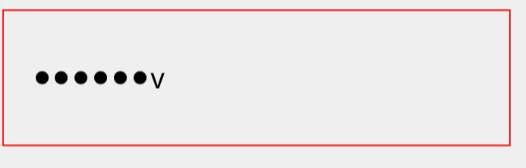
returnKeyType ="go"



returnKeyType ="send"



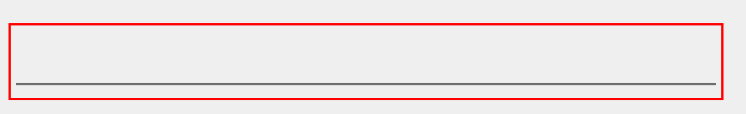
1. secureTextEntry：true时会遮住之前输入的文本，可以用于密码框



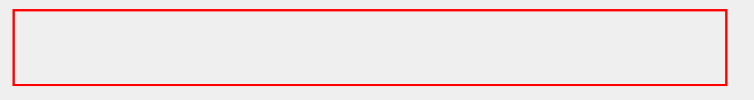
1. style： 继承了所有Text 组件的样式。
2. value：控制文本框中显示的内容，文本框中的值会和value属性的值保持一致。所以会导致文本框不可编辑，但是这种情况下使用editable 更好，不会导致有光标闪烁的情况。
3. android专用： returnKeyLabel 可以自定义return 键的显示，用来代替returnKeyType。
4. android专用 ：underlineColorAndroid 设置文本框下划线的颜色 ，当想取消下划线的时候设置为：

underlineColorAndroid = ‘transparent’

未设置：

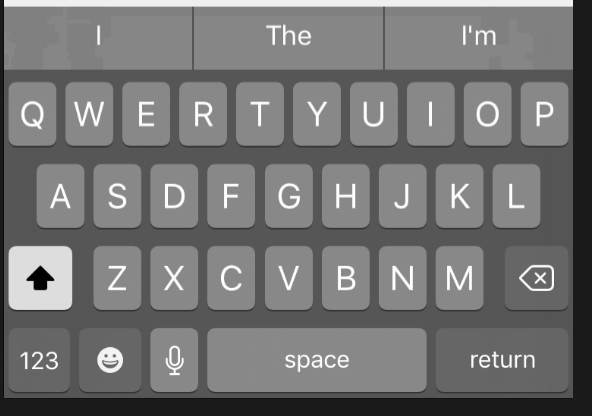


设置后：

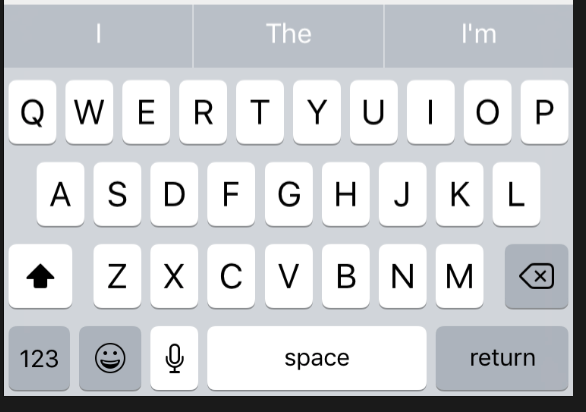


1. keyboardAppearance : 枚举类型，用于ios，设置键盘的颜色

keyboardAppearance="dark"

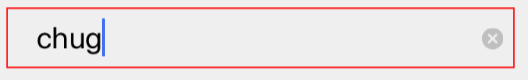


keyboardAppearance="light"



1. clearButtonMode: 用于ios，枚举类型，设置是否显示文本框右边的清除按钮。

clearButtonMode="while-editing"

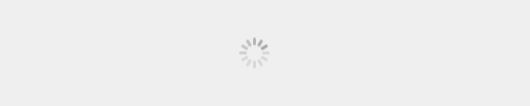


clearButtonMode="always"



ActivityIndicator组件: 加载符号

animating={true}： 是否显示



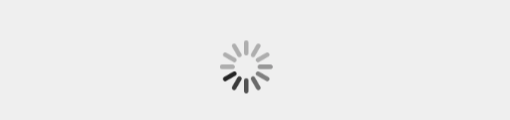
color="#ff0000"： 设置颜色



size: 枚举类型

small/large

size =“large”

(ios)

(android)

Button组件：一般用TouchableOpacity来自定义按钮组件

<Button title="按钮" />



<Button

title="按钮" //设置按钮的显示文本

color=”red”

/>

(ios)

(android)

onPress={() => Alert.alert('我被点击了')}



注意：

1. onEndEditing在失去焦点和点击软键盘的确定按钮的时候都会触发。

但是onSubmitEditing只会在点击软键盘的确定按钮的时候才会触发。

1. Value属性最好不用，可用其他属性替代。（defaultValue和editable，maxlength等。）
2. onChange和onChangeText的区别：onChangeText可以获得当前的文本框中的内容。
3. 受约束的(Controlled)组件：TextInput，ListView，DatePickerIOS等

约束性组件（值是由状态state来进行管理的，然后有了输入后触发事件来改变state中属性的值，然后在渲染到输入框中，虽然在输入框中看起来是用户输入的，实际上是组件重新渲染的。）和非约束性组件：

<http://blog.csdn.net/lihongxun945/article/details/46730835>

1. TODO: ListView FlatView ListView.DataSource WenView

进阶指南：

1. 颜色取值
2. “#fff” / “#ffffff” / “red” / “transparent” / ‘rgb(255,255,255)’ / ‘rgba(255,255,255,0.5)’
3. 对于有名字的颜色，RN遵循CSS3的规范，凡在CSS3里面可以使用的名字，在RN里面也都可以使用。
4. 图片：

行内引用：<Image source={require('./my-icon.png')} />

@2x/@3x后缀：<Image source={require('./img/check.png')} />

Packager打包所有的图片并且根据不同的设备像素比提供对应的资源。

require里面的东西一定是一个字符串，不能含有变量，因为require是在编译的时候执行而不是在运行的时候执行。

另一个值得一提的变动是我们把src属性改为了source属性，而且并不接受字符串，正确的值是一个带有uri属性的对象。

网络图片：

<Image

source={{uri: 'https://facebook.github.io/react/img/logo\_og.png'} style={{width: 400, height: 400}}

/>

背景图片ImageBackground

return (

<ImageBackground source={...}>

<Text>Inside</Text>

</ImageBackground>

);

TODO: PC端图片

1. ios/android后缀：

不同平台使用不同文件：比如引用图片a.png,如果在ios和android平台下面使用不同的图片，那么只需两个文件命名a.ios.png和a.android.png。不同平台会使用不同文件。

1. 处理触摸事件：

移动设备上的操作基本就是：点击 | 滑动 | 缩放

捕捉点击操作：

使用Touchable开头的一系列组件：

TouchableOpacity：点下去后透明度会变高

TouchableHighLight: 点下后会有一个背景色

onLongPress：该事件在按下按钮一段时间后会触发

处理上下滑动和左右滑动:

ScrollView

ListView是一个特殊的ScrollView

双指缩放：

使用ScrollView，并且里面只放置一个组件。设置maximumZoomScale和minimumZoomScale属性来使用户进行缩放。

1. 定时器

setTimeout, clearTimeout

setInterval, clearInterval

InteractionManager：让交互在动画之后执行

InteractionManager.runAfterInteractions(() => {

// ...需要长时间同步执行的任务...

});

//TODO: 什么时候要把操作放到交互执行完成之后执行，现有代码中很多操作都在交互完成后执行？？？？？

//TODO：setTimeout:为什么要使用setTimeout定时器？？使用定时器那么什么时候清除定时器？？

触摸处理系统会把一个或多个进行中的触摸操作认定为'交互'，并且会将runAfterInteractions()的回调函数延迟执行，直到所有的触摸操作都结束或取消了。

1. 特定平台代码

特定平台扩展名：

BigButton.ios.js

BigButton.android.js

使用：import BigButton from './components/BigButton';

<Text style={

Platform.select({

ios: {

backgroundColor: 'red',

},

android: {

backgroundColor: 'blue',

},

})}

; >按钮</Text>

（ios）

（android）

使用api： Platform.OS=’ios’

1. 网络

React Native提供了和web标准一致的[Fetch API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) 来进行访问网络。网络请求都是一个异步请求。React Native中已经内置了[XMLHttpRequest API](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest), 有第三方库封装了XMLHttpRequest ,比如axios ，可以直接使用。但是不能使用JQuery，JQuery中有很多浏览器才有的东西，RN中没有。

1. 浏览器(BOM)、RN、JS、 DOM：

RN中有和Web中一样的定时器，Fetch API，XMLHttpRequest 对象，

浏览器(BOM)：(Web API: https://developer.mozilla.org/zh-CN/docs/Web/API/Window)

window 对象表示一个包含DOM文档的窗口, 其 document 属性指向窗口中载入的 [DOM文档](https://developer.mozilla.org/zh-CN/docs/Web/API/cn/DOM/document)

window(属性：Location, Navigator, History；方法：setTimeout，setInterval，cleatTimeout, cleatInterval, alert)

TODO：BOM和DOM http://blog.csdn.net/xiao\_\_gui/article/details/8315148

TODO: HTML DOM 的对象（Style）

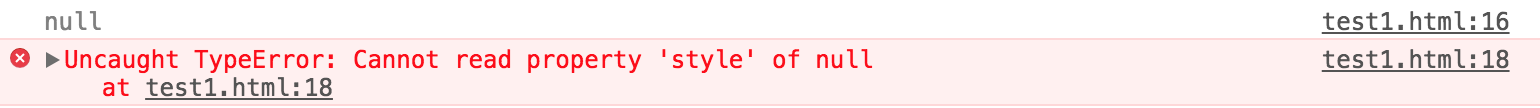
Style对象通过元素element.style.prop访问，拥有几乎所有的css属性，不过都是采用驼峰命名的方式。

1. 在head里面引入js脚本文件

代码：

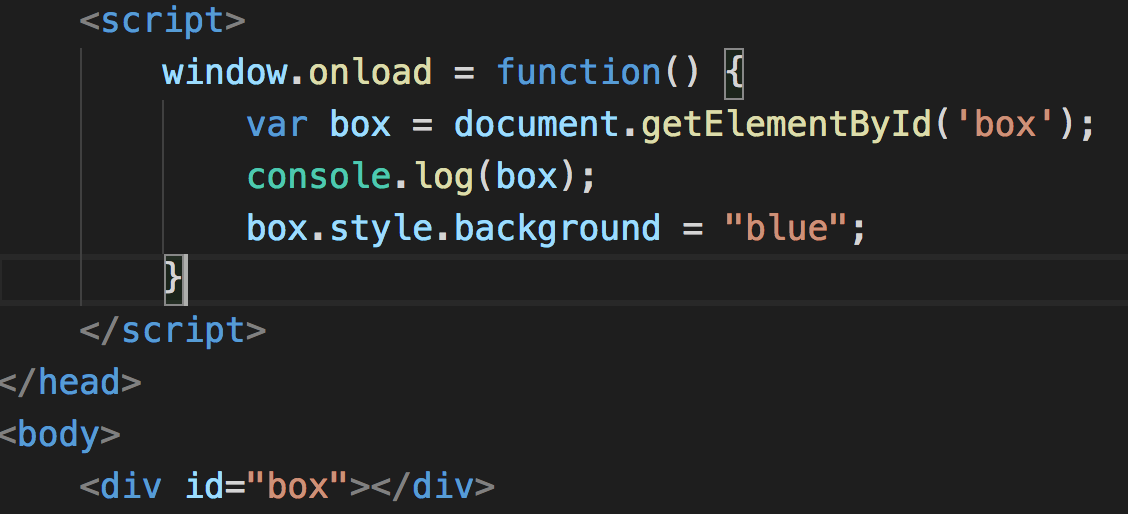


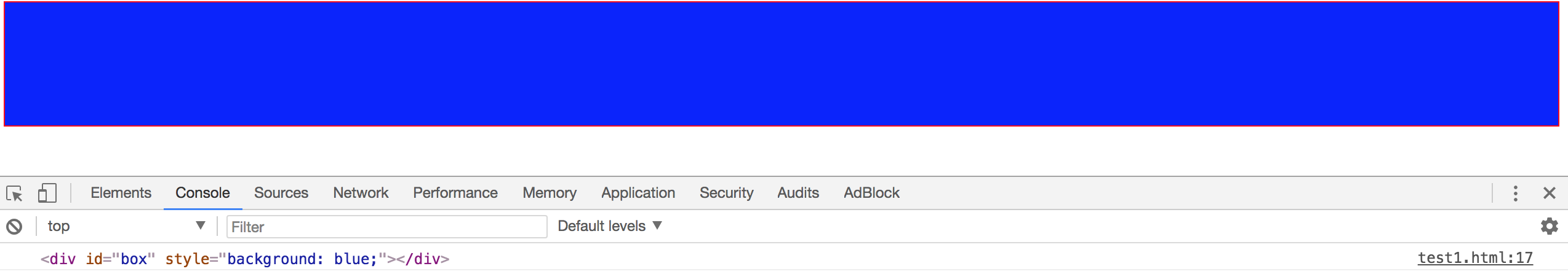
代码运行结果：



如果是放在body的最后面代码运行正常。

或者将代码放到window.onload触发后执行的function中：





1. CSS代码的优先级

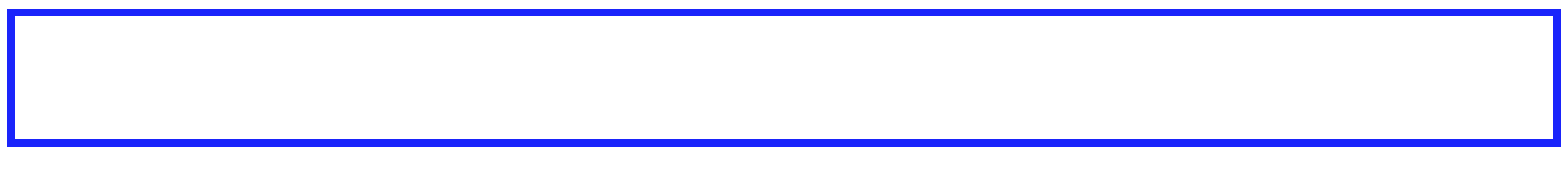
代码：test.css



html代码：



效果：



事实证明：CSS样式优先级的顺序：行内样式 > 内联样式 > 外部样式。

是建立在外部样式的引入 <link> 标签必须放在<style> 标签的前面。否则外部样式会覆盖内联样式。

所以实际上样式的应用遵循的是：就近原则。 优先级顺序的本质是就近原则。

1. HTML DOM对象
2. 标签对象： 一个标签就会是一个对象，通过document的方法获取。每一个对象都有一个style属性，类似img 元素拥有 src 属性。
3. Style对象：每一个标签上都挂载了一个style 对象。
4. DOM 事件和 JS 函数

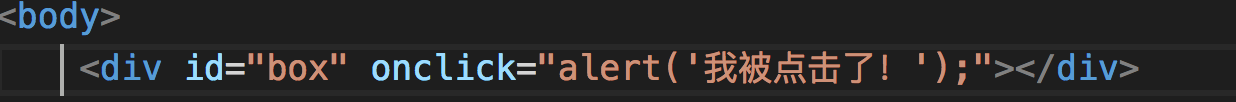
事件和函数配合使用，通过事件来驱动 JS 函数的执行。

通过事件属性可以为 DOM 事件指定相应的JS 执行函数。

代码：



或者：



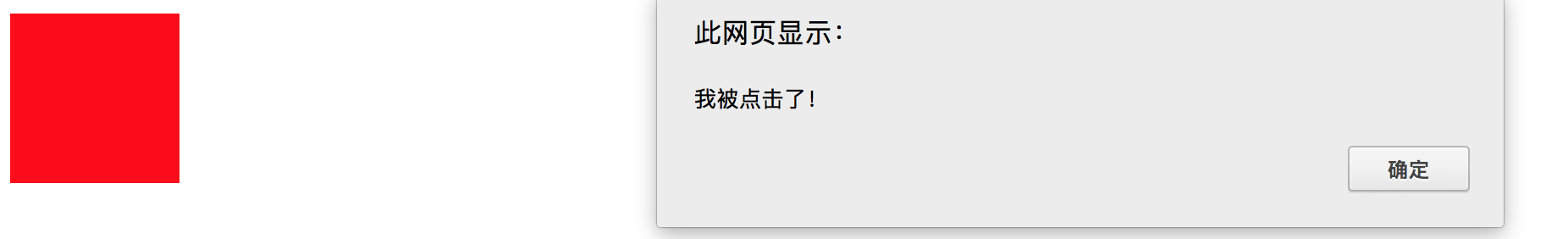
（说明：js代码存在html文档里面哪些地方：事件属性里面写代码字符串，script标签，外部引用）

或者：



（注意：为onclick事件属性指定的函数命名不可以是click，否则无效，但是在JS 中是可以命名为click的。）

效果：



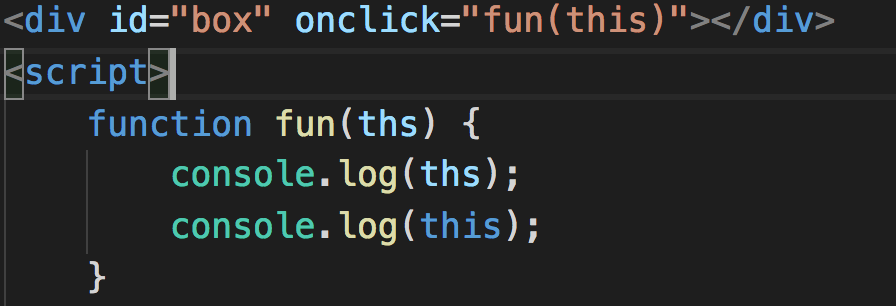
关于this：

代码：

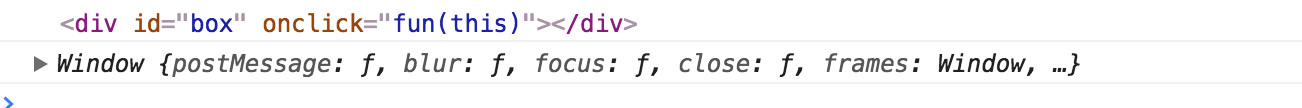
<div id="box" onclick="console.log(this)"></div>



代码：



输出：



TODO：阻止冒泡/浏览器的默认行为， addEventListener DOM事件流

1. 阻止事件冒泡

IE8 及以前 window. event. cancelBubble = true;

现代浏览器：event. stopPropagation();

阻止事件冒泡：

function myfn (e) {

window. event ? window. event. cancelBubble = true : e.stopPropagation();

}

阻止浏览器的默认行为：

function myfn(e) {

window. event ? window. event. returnValue = false : e.preventDefault();

}

阻止浏览器默认行为：TODO： http://blog.csdn.net/judas\_jia/article/details/51166406