目录

[移动端 1](#_Toc7795)

[复习： 3](#_Toc11326)

[一、 移动端的事件 9](#_Toc603)

[1.1 移动端的三个事件 9](#_Toc12408)

[1.2 事件对象 11](#_Toc18423)

[下午复习： 13](#_Toc16998)

[二、 动画事件和过度事件 14](#_Toc23813)

[2.1 过度事件 14](#_Toc18768)

[2.2 动画事件 15](#_Toc553)

[三、 Zepto 17](#_Toc2012)

[3.1 模块化 17](#_Toc10684)

[3.1 touch模块 18](#_Toc8228)

[四、 百度地图 19](#_Toc2567)

## 复习：

seajs:

如果想要使用seajs必须在页面中引入该文件，引入之后会向全局暴露两个变量：seajs、define

在模块外部想要引入一个文件需要通过seajs.use方法：

该方法接收两个参数

第一个参数是一个数组，数组的每一项都是一个模块文件路径，如果引入的只有一项文件可以省略数组

第二个参数是回调函数

函数中的参数就是前面模块向外暴露的功能（一一对应的）

根目录：

以seajs所在的文件目录为根目录

定义模块：

seajs提供了define方法用于定义模块，定义模块的方式有六种：

可以传递一个参数

可以是值类型（数字、字符串、布尔值）

可以是引用类型（数组、对象）

当传递的是一个函数的时候，此时函数中有三个参数

require 用来引入其它模块的

exports 向外暴露功能的对象

module 模块信息对象

可以传递两个参数：

第一个参数是字符串表示模块的id

第一个参数可以是数组， 表示的模块的依赖集合

第二个参数是回调函数（同上面定义模块中传递函数的表现形式是一致的）

可以传递三个参数：

第一个参数是字符串，表示模块的id

第二个参数是数组，表示模块的依赖集合

第三个参数是回调函数

接口定义：

exports.接口

module.exports.接口

module.exports = 值类型

module.exports = 对象

module.exports = 函数

return 值类型

return 对象

return 函数

define(值类型)

define({})

模块信息对象：

id: 模块的id

uri: 模块的文件路径，默认情况下（没有显示的id）与模块的id是一致的

exports: 向外暴露功能的对象

require模块：

在一个模块想要引入其它模块要分为两步走

第一步是通过require方法引入模块文件

第二步通过模块文件读取模块

1 require不能被简写

2 require不能被修改

1 require不能被赋值

2 require不能赋值给其它变量

3 require不能作为子函数中的参数使用

4 在子函数中不能被修改

3 require只能是字符串不能拼接

配置：

本身这个框架具有的功能，只是隐藏了，需要开启的时候要进行配置

插件：

本身框架没有这个功能，只是别人实现了，想要使用必须引入

alias:

作用： 简化文件的引用

值是一个对象

key: 新的文件名

value：原始文件名

paths:

作用：简化文件的路径

值是一个对象

key: 新的文件路径

value： 原始路径

map:

用于批量处理文件

值是一个二维数组

数组中的每一项都是一个数组

数组中的第一项表示要匹配的文件

数组中的第二项表示要修改的文件

vars：

解决字符串不能拼接的问题

key: 模板变量名称

value: 模板变量名称对应的值

base:

改变根目录

css插件：

使用起来非常简单

第一步要在seajs文件之后引入seajs-css.js

第二步直接在模块中引入即可

需要注意的是在引入的css文件中一定要加上.css后缀

预加载文件：

第一步要在seajs之后引入seajs-preload.js

第二步要进行配置，配置preload

值是一个数组，数组中的每一项都是要预加载的文件

加载预加载文件的时间点

在seajs.use方法之后

模块加载之前

Requirejs:

想要使用requirejs与seajs一样要在页面中引入该文件

引入文件之后会向全局暴露三个变量：require 、requirejs、define

在引入requirejs文件的script标签上有一个叫做data-main属性可以引入项目的入口文件

除了以上方式可以引入入口文件，还可以通过require方法引入入口文件

require方法接收两个参数

第一个参数是数组，表示模块依赖集合

必须是数组

第二个参数是回调函数

函数中的参数就是前面模块向外暴露的功能

根目录：

如果有data-mian属性，是以data-main的属性值为根目录

如果没有data-main属性，则是以html文件所在的目录决定的

定义模块：

可以传递一个参数

可以是值类型 （不能是字符串）

可以是引用类型（不能是数组）

当传递的是一个函数的时候，此时函数遵循的是commonjs规范，requirejs提供了参数注入的技术，想要使用哪个模块就要注入哪个模块

当注入exports和module的时候

作用域是向外暴露功能的对象

如果没有注入exports和module

是全局作用域

可以传递两个参数

第一个参数是字符串 表示模块的id

第二个参数也可以是数组，表示模块的依赖集合

第三个参数是回调函数，此时函数遵循的是module transports规范，想要使用哪个模块就要在依赖集合中注入哪个模块包括内置模块(require exports module)

当注入exports和module的时候

作用域是向外暴露功能的对象

如果没有注入exports和module

是全局作用域

可以传递三个参数：

第一个参数是字符串 表示模块的id

第二个参数也可以是数组，表示模块的依赖集合

第三个参数是回调函数，此时函数遵循的是module transports规范

在module transports中定义的模块文件名称不要与内置模块是相同的

加载具有id的模块：

第一步要通过模块的依赖集合加载模块文件

第二步要通过require指定id（这里是require方法是异步的）

commonjs规范是无法加载具有id的模块的

接口定义：

exports.接口

module.exports.接口

module.exports = 值类型

module.exports = 对象

module.exports = 函数

以上方式在requirejs中同样适用

return 值类型

return 对象

return 函数

这几种方式是基于module transports规范，不依赖于require exports module, 开发起来更灵活

define(值类型)

define(对象)

模块信息对象：

id： 表示模块的id

uri: 模块的文件路径，是相对于html文件的

exports: 向外暴露功能的对象

配置：

paths:

用于简化文件的路径

值是一个对象

key: 新的文件路径

value: 原始路径

shim：

用于将文件转为模块的

key: 模块的文件路径

value: 是一个对象

exports 配置接口

deps 配置依赖集合

map:

用来配置模块文件

key： 模块的文件路径

value是一个对象

key: 引用的模块文件

value: 修改的模块文件

baseUrl:

改变根目录

baseUrl > data-main > require

加载css插件：

在requirejs中不能直接使用css文件

要作为模块来使用，所以要进行配置

map: {

“\*”： {

css: 文件目录

}

}

在引入css模块文件中要加上css!前缀

# 移动端的事件

在移动端新增了touch事件，因为手指的行为叫做“触摸”， 鼠标的行为叫做“点击”

但是它仍然支持点击事件，有300ms的延迟，检测是否双击

举例：

|  |
| --- |
| 1. // 获取元素 2. var box = document.getElementById("box"); 3. // 获取当前时间 4. var date = new Date(); 5. // 为box注册一个点击事件 6. box.onclick = function() { 7. // 改变文字的颜色 8. this.style.color = "white"; 9. console.log("onclick", new Date() - date); 10. } 11. // 移动端添加的触摸事件 12. box.addEventListener("touchstart", function() { 13. // 改变文字的颜色 14. this.style.color = "orange"; 15. console.log("touchstart", new Date() - date); 16. }) |

结果：

|  |
| --- |
|  |

这两个数值之差就是两个事件毫秒之差

## 移动端的三个事件

touchstart：触摸开始

绑定方式：

dom.addEventListener(“touchstart”, fn)

举例：

|  |
| --- |
| 1. // 获取元素 2. var box = document.getElementById("box"); 3. // 为box注册touchstart事件 4. box.addEventListener("touchstart", function() { 5. console.log("touchstart"); 6. }) |

结果：

|  |
| --- |
|  |

touchmove: 触摸移动

绑定方式：

dom.addEventListener(“touchmove”, fn)

举例：

|  |
| --- |
| 1. // 注册touchmove事件 2. box.addEventListener("touchmove", function() { 3. console.log("touchmove"); 4. }) |

当触摸移动的时候触发该事件:

|  |
| --- |
|  |

touchend: 触摸结束

绑定方式:

dom.addEventListener(“touchend”, fn)

举例：

|  |
| --- |
| 1. // 注册touchend事件 2. box.addEventListener("touchend", function() { 3. console.log("touchend"); 4. }) |

当松开鼠标的时候触发该事件：

|  |
| --- |
|  |

## 事件对象

查看：

|  |
| --- |
|  |

在touchstart和touchmove事件中获取手指相关信息的属性: e.touches

举例：

|  |
| --- |
| // 获取元素  var box = document.getElementById("box");  // 为box注册touchstart事件  box.addEventListener("touchstart", function(e) {  console.log(e);  // 在touchstart事件获取手指相关信息的属性: e.touches  console.log(e.touches[0].clientX);  console.log(e.touches[0].clientY);  })  // 注册touchmove事件  box.addEventListener("touchmove", function(e) {  // 在touchmove事件中，获取手指相关信息的属性: e.touches  console.log(e.touches[0].clientX);  console.log(e.touches[0].clientY);  }) |

在touchend事件中获取手指信息的相关属性叫做: e.changedTouches

举例：

|  |
| --- |
| // 注册touchend事件  box.addEventListener("touchend", function(e) {  // 在touchend事件中获取手指相关信息的属性: e.changedTouches  console.log(e.changedTouches[0].clientX);  console.log(e.changedTouches[0].clientY);  }) |

## 下午复习：

移动端的事件：

touchstart: 触摸开始

touchmove: 触摸移动

touchend: 触摸结束

事件对象：

在touchstart和touchmove事件中获取手指相关信息的属性：e.touches

在touchend事件中获取手指相关信息的属性：e.changedTouches

# 动画事件和过度事件

## 2.1 过度事件

当一个元素过度完成之后会触发一个事件: transitionend事件

举例：

|  |
| --- |
| <style type="text/css">  \* {  margin: 0;  padding: 0;  }  #box {  position: absolute;  width: 100px;  height: 100px;  background-color: red;  left: 0;  top: 0;  transition: all 1s;  }  #box.cur {  left: 100px;  }  </style>  </head>  <body>  <div id="box"></div>  <script type="text/javascript">  // 获取元素  var box = document.getElementById("box");  // 两秒之后添加类名  setTimeout(function() {  box.setAttribute("class", "cur");  }, 2000)  // 当一个元素过度完成之后会触发一个事件  box.addEventListener("transitionend", function() {  console.log("过度完成");  }) |

结果：当过度完成之后触发transitionend事件

|  |
| --- |
|  |

## 2.2 动画事件

当一个元素动画开始的时候会触发一个事件: animationstart

举例：

|  |
| --- |
| #box {  position: absolute;  width: 100px;  height: 100px;  background-color: red;  left: 0;  top: 0;  /\*动画的调用\*/  animation: donghua 1s ease 2s 3 alternate;  }  /\*动画的定义\*/  @keyframes donghua {  from {  left: 0;  }  to {  left: 100px;  }  } |

执行代码：

|  |
| --- |
| // 获取元素  var box = document.getElementById("box");  // 动画开始事件  box.addEventListener("animationstart", function() {  console.log("动画开始");  }) |

结果：

|  |
| --- |
|  |

当一个元素动画结束之后会触发一个事件: animationend事件

举例：

|  |
| --- |
| // 动画结束事件  box.addEventListener("animationend", function() {  console.log("动画结束");  }) |

结果：

|  |
| --- |
|  |

# Zepto

Zepto是一个轻量级的针对现代高级浏览器的JavaScript库， 它与jquery有着类似的api。 如果你会用jquery，那么你也会用zepto。

官网：<https://www.html.cn/doc/zeptojs_api/>

下载地址：<https://zeptojs.com/zepto.js>

## 3.1 模块化



zepto是分为模块化的，天生是不具备animate函数想要使用必须下载fx模块

举例：

在没有下载fx模块之前：

|  |
| --- |
| 1. <div id="box"></div> 2. <script type="text/javascript" src="js/zepto.js"></script> 3. <script type="text/javascript"> 4. // 由于引入了zepto所以可以使用$函数 5. $("#box").animate({"width": 1000}, 2000); 6. </script> |

结果：

|  |
| --- |
|  |

zepto.js:

|  |
| --- |
|  |

当下载了fx模块之后：

|  |
| --- |
|  |

zepto.js:

|  |
| --- |
|  |

## 3.1 touch模块

移动端在touch模块中自定义了一些事件：

tap: 触碰

doubleTap: 双击

singleTap: 单击

longTap: 长按 750ms之后触发

swipe: 滑动

swipeLeft: 左滑

swipeRight: 右滑

swipeUp: 上滑

swipeDown: 下滑

|  |
| --- |
| 1. // tap 2. $("#box").tap(function() { 3. console.log("tap"); 4. // console.log(new Date() - date); 5. }) 6. // doubleTap 7. $("#box").doubleTap(function() { 8. console.log("doubleTap"); 9. }) 10. // singleTap 11. $("#box").singleTap(function() { 12. console.log("singleTap"); 13. }) 14. // longTap 750ms之后触发 15. $("#box").longTap(function() { 16. console.log("longTap"); 17. }) 18. // swipe 19. $("#box").swipe(function() { 20. console.log("swipe"); 21. }) 22. // swipeLeft 23. $("#box").swipeLeft(function() { 24. console.log("swipeLeft"); 25. }) 26. // swipeRight 27. $("#box").swipeRight(function() { 28. console.log("swipeRight"); 29. }) 30. // swipeUp 31. $("#box").swipeUp(function() { 32. console.log("swipeUp"); 33. }) 34. // swipeDown 35. $("#box").swipeDown(function() { 36. console.log("swipeDown"); 37. }) |

# 百度地图

自学方式

在工作中领导交给我们一个任务，在一个页面中显示出地图并且在地图中显示自己公司的位置

既然是百度地图，所以要查找百度地图API

现成的地图有: 高德地图、百度地图、谷歌地图、腾讯地图

1：搜百度地图API



2:



3:



4:

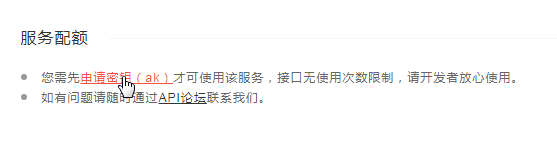


5:

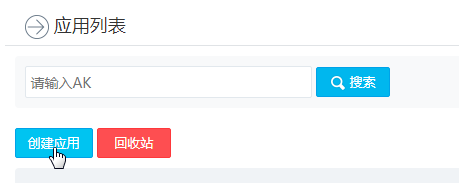


假如已经申请了百度账号并且也已经申请成为百度开发者，下一步要获取密钥

6：



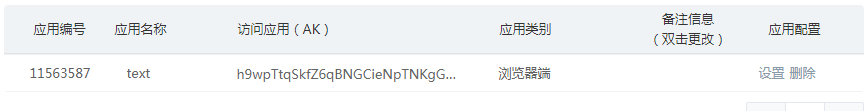
7：



8：

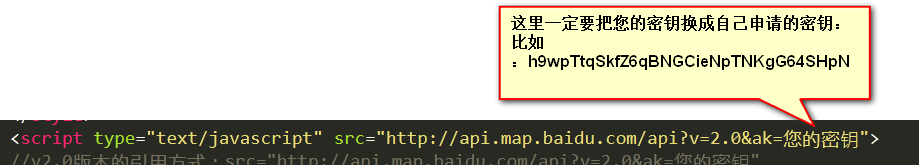


当点击提交按钮之后：



当获取密钥之后就可以去开发了

按照demo的提示一步一步走



|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |