爱创课堂前端培训

# 移动端

第1天课堂笔记（本课程共1天）

班级：北京前端训练营19期

讲师：彭帅伟

日期：2018年12月28日

彭帅伟老师

QQ: 284337908

爱创课堂官网 ：[www.icketang.com](http://www.icketang.com)

目录

[移动端 1](#_Toc14312)

[复习： 3](#_Toc230)

[一、 移动端的事件 9](#_Toc15357)

[1.1 移动端的三个事件 9](#_Toc19632)

[1.2 事件对象 11](#_Toc2085)

[二、 动画事件和过度事件 15](#_Toc13004)

[2.1 过度事件 15](#_Toc27313)

[2.2 动画事件 15](#_Toc28321)

[三、 zepto 17](#_Toc9831)

[3.1 模块化 18](#_Toc8458)

[3.2 touch模块 18](#_Toc17711)

[四、 百度地图 19](#_Toc28355)

# 复习：

seajs:

如果想要使用seajs必须在页面中引入文件

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另外一个模块要通过use方法

接收两个参数

第一个参数家在模块的依赖集合

如果是一个字符串，表示一个模块

如果是一个数组，表示多个模块

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接口

作用域是全局作用域

seajs的根目录是以seajs所在的文件目录为根目录

seajs对js文件敏感， 因此可以省略.js后缀

定义模块：

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（arr, obj）

当传递一个函数的时候，此时函数遵循commonjs规范，

函数中有三个参数

require 用来引入其它模块的

exports: 向外暴露接口的对象

module: 模块信息对象

传递两个参数：

第一个参数可以是字符串，表示模块的id

第一个参数可以是数组，表示模块的依赖集合

第二个参数是回调函数（与上面第三种定义模块方式中的函数表现形势一致）

传递三个参数：

第一个参数是一个字符串， 表示模块的id

第二个参数是一个数组， 表示模块的依赖集合

第三个参数是回调函数（与上面第三种定义模块方式中的函数表现形势一致）

require模块：

在一个模块中引入另一个模块

通过require引入其它模块文件

从该模块文件中读取该模块

1 require不能被简写

2 requier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不能在子函数中作为参数传递

4 require不能在子函数中被修改（以上3点）

3 require不能拼接

接口定义：

exports.接口

module.exports.接口

module.exports = 值类型

module.exports = {}

module.exports = 函数

return 值类型

return {}

return 函数

define()

模块信息：

id: 表示模块的id

uri: 表示模块的文件地址

exports: 向外暴露功能的接口对象

配置：

本身具有这些功能， 只是隐藏了， 需要开启，要进行配置

alias:

作用: 简化文件的引用

值是一个对象

key: 新的文件名

value: 原始文件路径

paths:

作用: 简化文件的路径

值是一个对象

key: 新的路径名称

value: 原始路径

map:

作用: 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

数组中的第一项是匹配到的文件

数组中的第二项是修改的文件

vars:

解决字符串不能拼接的问题

值是一个对象

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：

第一步要在seajs之后引入seajs-css.js文件

第二步直接在模块中引入css文件即可

注意：引入的css文件不能省略.css拓展名

预加载文件：

第一步在seajs文件之后引入seajs-preload.js文件

第二步要进行配置，定义preload配置

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：

seajs.use方法之后，模块执行之前

requirejs:

使用requirejs与seajs一样都是要在页面中引入该文件

引入requirejs之后会向全局暴露3个变量：require、 requirejs、 define

根目录：

如果有data-main属性，此时根目录是以data-mian的属性值为根目录

如果没有data-mian属性，此时根目录是由html文件来决定的

在引入requirejs的script标签上有一个data-main属性，它的属性值就是项目的入口文件

除了以上方式可以引入入口文件

require中有一个require方法相当于seajs中的seajs对象

因此还可以通过require来引入入口文件

接收两个参数

第一个参数是加载模块的依赖集合

必须是一个数组

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接口（一一对应的）

作用域是全局作用域

定义模块：

传递一个参数：

可以是值类型（不能是字符串）

也支持对象

当传递一个函数的时候，此时函数遵循commonjs规范，requirejs提供了参数注入的技术， 如果想要使用哪个模块就要注入哪个模块

当没有注入exports和module的时候

作用域是window

如果注入exports和module

作用域是向外暴露接口的对象

传递两个参数:

第一个参数是字符串

第一个参数是字符串

第二个参数是回调函数

如果第一个参数是字符串，表示模块的id， 此时函数遵循commonjs规范

如果第一个参数是数组， 表示模块的依赖集合，此时函数遵循module transports规范， 想要使用哪个模块，就必须在依赖集合中注入，包括内置模块（require, exports, module）

当没有注入exports和module的时候

作用域是window

如果注入exports和module

作用域是向外暴露接口的对象

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，此时函数遵循module transports规范，

在module transports规范中，定义的模块文件名称不要与内置模块的名称相同

接口定义：

exports.接口

module.exports.接口

module.exports = 值类型

module.exports = {}

module.exports = 函数

以上几种方式都是commonjs规范， require同样适用

return 值类型

return 对象

return 函数

是module transports

模块信息对象：

id: 表示模块的id， 默认是与uri不同

uri: 模块的文件地址， 是相对于html文件

exports: 向外暴露功能的接口对象

配置：

paths:

用于简化路径的

key: 新的路径名

value：原始路径

shim:

将文件转为模块的

key表示模块的文件路径

value: 是一个配置对象

exports 配置接口

deps 配置依赖集合

map:

配置模块文件的

key表示模块的文件路径

value: 对象

key: 引入的模块文件

value: 修改的模块文件

baseUrl:

改变根目录

baseUrl > data-main > require

css插件：

在requirejs中不能直接引入css文件

要作为模块来引入，要进行配置

map: {

\*: {

css: 文件目录

}

}

引入的css文件中要加上css!前缀

# 移动端的事件

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”

仍然支持点击事件，但是有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. // 按照pc端的思想， 要给box添加点击事件 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. // touchstart 12. box.addEventListener("touchstart", function() { 13. // 改变文字颜色为蓝色 14. // console.log(this); 15. this.style.color = "blue"; 16. console.log("touchstart", new Date() - date); 17. }) |

结果：

|  |
| --- |
|  |

这两个数值之差就是两个事件的毫秒差

## 移动端的三个事件

移动端添加的事件分别是：

touchstart: 触摸开始

绑定方式：

dom.addEventListener(“touchstart”, function() {})

举例：

|  |
| --- |
| 1. // touchstart 2. box.addEventListener("touchstart", function() { 3. console.log("触摸开始"); 4. }) |

结果：

|  |
| --- |
|  |

touchmove： 触摸移动

绑定方式：

dom.addEventListener(“touchmove”, function() {})

举例：

|  |
| --- |
| 1. // touchmove 2. box.addEventListener("touchmove", function() { 3. console.log("触摸移动中"); 4. }) |

结果：

当鼠标在box中移动的时候：

|  |
| --- |
|  |

touchend：触摸结束

绑定：

dom.addEventListener(“touchend”, function() {})

举例：

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

结果：  
在松开鼠标的时候：

|  |
| --- |
|  |

## 事件对象

查看事件对象：

|  |
| --- |
|  |

在touchstart和touchmove事件中， 获取手指相关的信息通过e.touches来获取

举例：

|  |
| --- |
| 1. // touchstart 2. box.addEventListener("touchstart", function(e) { 3. // 查看事件对象 4. // console.log(e); 5. // 获取手指相关信息 6. console.log(e.touches[0].clientX); 7. // console.log(e.touches[0].clientY); 8. }) |

touchstart:

|  |
| --- |
|  |

touchmove:

|  |
| --- |
| 1. // touchmove 2. box.addEventListener("touchmove", function(e) { 3. // 获取手指相关信息 4. console.log(e.touches[0].clientX); 5. }) |

结果：

|  |
| --- |
|  |

在touchend事件中，获取手指信息不是通过e.touches， 而是通过e.changedTouches来获取

举例：

|  |
| --- |
| 1. // touchend 2. box.addEventListener("touchend", function(e) { 3. // console.log(e); 4. // 获取手指相关信息 5. console.log(e.changedTouches[0].clientX); 6. }) |

结果：

|  |
| --- |
|  |

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，通过e.touches来获取手指相关信息

在touchend事件中，通过e.changedTouches来获取手指相关信息

# 动画事件和过度事件

## 2.1 过度事件

在一个元素过度完成之后会触发一个事件: transtionend事件

绑定方式：

dom.addEventListener(“transitionend”， function() {})

举例：

|  |
| --- |
| 1. // 获取元素 2. var box = document.getElementById("box"); 3. // 2s之后添加类名 4. setTimeout(function() { 5. // box.className = "cur"; 6. box.setAttribute("class", "cur"); 7. }, 2000) 8. // 过度完成事件 9. box.addEventListener("transitionend", function() { 10. console.log("过度完成"); 11. }) |

## 2.2 动画事件

在一个元素动画开始的时候会触发一个事件: animationstart事件

举例：

|  |
| --- |
| 1. <style type="text/css"> 2. \* { 3. margin: 0; 4. padding: 0; 5. } 6. #box { 7. position: absolute; 8. width: 100px; 9. height: 100px; 10. background-color: red; 11. left: 0; 12. top: 0; 13. animation: donghua 1s linear 0s 3 alternate; 14. } 15. /\*定义动画\*/ 16. @keyframes donghua { 17. from { 18. left: 0; 19. } 20. to { 21. left: 100px; 22. } 23. } 25. </style> |

执行代码：

|  |
| --- |
| 1. // 动画开始事件 2. box.addEventListener("animationstart", function() { 3. console.log("动画开始"); 4. }) |

结果：

|  |
| --- |
|  |

在一个元素动画结束的时候，会触发一个事件叫做animationend事件

举例：

|  |
| --- |
| 1. // 动画结束事件 2. box.addEventListener("animationend", function() { 3. console.log("动画结束"); 4. }) |

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.css88.com/doc/zeptojs_api/>

下载：<https://zeptojs.com/zepto.js>

zepto是模块化的，如果你想要使用animate函数， 你必须下载fx模块， 如果你想要使用移动端的事件，

必须下载touch模块

|  |
| --- |
|  |

## 3.1 模块化

zepto天生不具备animate方法， 需要使用的时候必须要下载fx模块

在没有下载之前：

|  |
| --- |
|  |

zepto.js

|  |
| --- |
|  |

在下载了fx模块之后：

|  |
| --- |
|  |

zepto.js

|  |
| --- |
|  |

## 3.2 touch模块

touch模块中自定义了一些移动端的事件：

tap: 触碰

doubleTap: 双击

singleTap: 单击

longTap: 长按事件 750ms之后执行

swipe: 滑动

swipeLeft: 左滑动

SwipeRight: 右滑动

swipeUp: 上滑动

swipeDown: 下滑动

以上事件可以使用jquery中两种方式来绑定：

$(dom).on(事件类型， fn)

$(dom).tap(fn)

# 百度地图

自学方式

在工作中，领导交给我们一个任务，在某网站上显示出地图并且要显示出自己公司的位置

既然是显示地图，要找到地图API

现在地图有: 高德地图、 百度地图、 谷歌地图

1：



2：



3：



4:



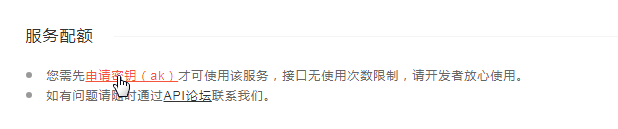
5:



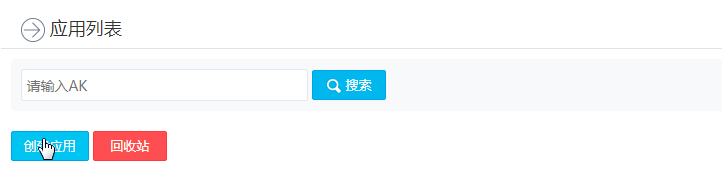
6:

假如已经注册百度账号，并且申请成为百度开发者

下一步要获取密钥



7：

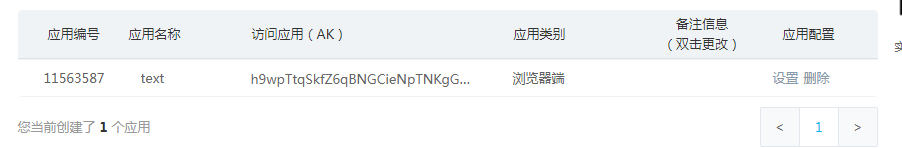


8：



9：

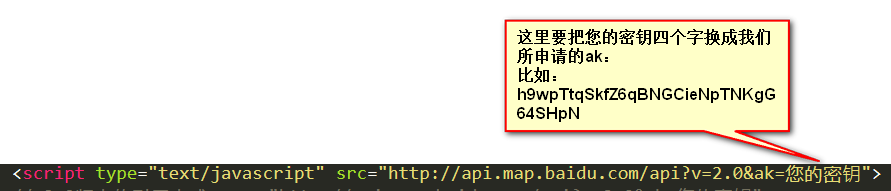
创建出来的密钥



获取密钥之后我们就可以进行开发了

按照demo提示即可

10：



|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |