## JS文件结构是两对括号

示例：（function(){}）()

解释：先定义一个匿名函数，然后直接调用。

## use strict

使用严格模式。

## prefix

URL前缀

大量URL有一部分相同的前缀，把它提取出来，作为URL的公共成分。

## index.config.js中config参数，有的有‘$’，有的没有？

设置的服务名字不同，没有什么特别之处

## Restangular服务

访问服务端，同$http。

在配置阶段配置基础路由路径：.setBaseUrl

## $httpProvider

在请求发送之前或数据接收到之前进行预处理。

在$httpProvider.interceptors中定义了一系列特定方法的factory。push方法用于将定义的函数加入数组，这些特定函数又被称为拦截器。实例中authInterceptorService就是定义的函数。

在index.factory.js文件中的authInterceptor函数中又有两个公共方法：

request：拦截成功的请求并处理。在$http向服务器发送请求之前被调用，在此函数中可以对成功的http请求进行处理。包含一个http:config对象作为参数，然后直接返回config对象。

request实例功能：在请求头中加入token以便验证身份。

requestError；response

responseError:拦截失败的响应并处理（从服务器接收到响应之前）此方法会在前一个拦截器抛出异常或进行了reject操作时被调用，在这里可以进行恢复响应的操作，进行了一些针对错误的处理。

## index.provider.js中唯一的服务：pageTitle

注：暂时不重要

pageTitle翻译为页面标题，defaultTitle翻译为默认标题

statechange翻译为状态变更

arguments是函数内置方法，是个数组对象，用于存放函数的参数。

1. 只能通过length和索引访问
2. 不需要声明即可直接使用
3. 只在非箭头函数中存在
4. 如果有名称为arguments的参数时，内置方法失效。

由于arguments不是一个真正的数组，很多数组方法都用不了，所以要把arguments转化为真正的数组：Array.prototype.slice.call(arguments)。

## 事件传递，监听

在angularJS中提供了emit,broadcast和on服务用于控制器之间基础事件的传递交流。  
 emit一直向根作用域发送事件，并通知在传递路径上的监听器，如果一个监听器接收到了这个事件，会注销这个事件，且事件停止传播。

broadcast一直向子作用域发送事件，并通知在传递路径上的监听器，如果一个监听器接收到了这个事件，事件也会继续往下传播。

on是各个作用域设置的监听器，获取从emit或者broadcast发送的事件

on也可以监听自定义的状态改变（路由变化）：

AngularJS路由状态变化时可以通过stateChangeStart、stateChangeSuccess,startChangeError监听。多用于随着改变导航标题时路由产生变化然后改变页面标题和相关样式。

参数：

event:改事件的基本信息

toState:当前路由信息

toParams:当前路由的参数

fromState:上一个路由的信息

fromParams:上一个路由的参数

## Provider

provider有三个部分

1. 普通的私有变量和私有函数，这些变量和函数在以后会被修改。没有什么特别。
2. 第二部分是在config函数里可以访问的变量和函数，所以，他们可以在其他地方使用之前被修改，这些变量和函数一定要添加到this上面
3. 在控制器里可以访问的变量和函数，通过this.$get函数返回；同时在this.$get注入需要用到的外部服务（而非function）。即依赖注入。

provider是创建服务的最底层的方式,任何提供一个带有返回值的$get方法即为provider（说法有待商榷）。

唯一可以使用config的创建服务方法

## factory

工厂模式

返回一个包含各种方法和数据的对象

当仅仅需要一个方法和数据的存放的地方，不需要处理复杂的逻辑。

config配置服务时不能用factory。

## service

构造函数方式创建，创建方法或变量时加this

factory定义方法，service调用factory方法并传递参数,controller使用service（依赖注入）

config配置服务时不能使用service

## x=x||{}

如果x已经被初始化后，则没有变化。

如果x没有被初始化，则赋一个对象给它，即初始化。

多用于为变量指定默认值

## dialogs

dialogs用于对话框

backdrop 对话框背景元素。true时显示背景且点击背景可关闭对话框；false时无背景；static时显示背景但点击背景不会关闭对话框。

keyboard true时可以按ESC退出；false时不行。

equals（a,b）:===或者相同的正则表达式

## translate

translate: 国际化，方便系统在不同语言之间切换

.translations: 注册指定的语言

preferredLanguage: 设置默认语言

## run

初始化全局数据，仅对$rootScope有效

## toaster

炫酷的提示信息显示插件

## injector

注入器。

$provide和$module注册的（定义，没有创建，即定义需要创建某个服务的任务）服务都储存在injector中。

injector中实现服务的创建。

$provide和$module提供provider,constant,value,factory,service来创建服务。

angular.injector([模块名])：创建

若加载了多个模块，则injector可以获取到多个模块下的服务

若angular.injector对同一个模块调用多次，则每次调用的都是新的对象，即injector[“myApp”]!=injector[“myApp”]

1. $injector.get(name)：根据名字获得服务的实例；当对象不存在会尝试实例化对象。
2. $injector.has(name): 判断一个对象是否存在
3. $injector.annotate(name): 获得name的所有依赖项
4. invoke(fn,self,locals):执行自定义函数。fn为需要执行函数名称;self表示函数中this变量；locals为提供变量给执行函数。（未理解）
5. instantiate: 类似invoke，自定义函数是构造函数

任何依赖注入都要用到injector,有些情况是简写。

例如：

angular.module(“myApp”,[]).controller(“myCtrl”,function(aC,bC{}));

//aC,bC涉及依赖注入，都会用到injector，此处是简写。

## angular.extend

把一个或多个对象中的方法和属性扩展到一个对象中，使得这个对象有其他对象相同的方法和属性。

angular.extend(a,b,c,...): 第一个参数是要被扩展的对象，后面可以传入一个或多个对象。注：若后续引用了对象，a只会引用后续引用的同一个对象。

## angular.module

模板：angular.module(name,[requires],[configFn])

name:字符串，模块名字

requires:字符串数组，该模块依赖的其他模块；不依赖则为空

congfigFn:对该模块进行一些配置,一般用module.config代替。

angular.module(“myApp”,[]): 若不止一个参数，代表新建模块，空数组表示该模块不依赖其他模块。

angular.module(“myApp”): 若只有一个参数，代表获取模块，模块不存在则会报错。

## constant

无法修改，是常量（value可以修改）。多用于配置一些常用的数据

多用于依赖注入

配置阶段的依赖注入（value不能往配置里传值）。

angular.model().constant(name,value): 创建全局变量。多用于某个服务只要返回数据即可。

## 懒加载

资源不会在第一时刻加载，而是在页面需要使用的时候加载。提高页面加载效率。

例如页面使用到的js文件。正常是HTML文件使用script载入js文件；懒加载是当路由跳转到当前页面时在路由即使在当前页面需要的js文件。

## $(“.classname”)

所有CSS class=”classname”的元素

## 表达式a && 表达式b

若a为true，则执行b

## $watch

$watch(watchFnName,watchAction,deepWatch)

watchFnName: angularjs表达式或函数的字符串

watchAction: watchFnName被监控的对象发生变化会被调用

deepWatch: 可选。布尔值，检查被监控的对象的某些个属性是否发生变化

返回一个函数，用于注销watch。

## directive

用于创建自定义指令。

在创建时元素名应该用驼峰命名法：例如abCd。使用时<ab-cd></ab-cd>

restrict: 表示会限制使用

priority: 指明单个DOM上的指令的优先级

terminal： true/false,true时启用。优先级低于此指令的其他指令不会被调用（调用无效；同级会执行）

template: 字符串（一段HTML文本）；匿名函数function（tElement,tAttrs）{},tElement指使用此指令的元素，tAttrs指实例的属性（该指令中所有属性组成的对象）

templateUrl: 字符串（HTML文件路径）；函数（同template）

replace: true/false,true时启用。启用时，指令失效。

scope： false|true|{}。默认为false,和父作用域使用同一个作用域；true时继承父作用域且创建自己的作用域；{}表示创建一个全新的隔离作用域。

当隔离作用域时候有三种方法可以使隔离作用域间交互：

1. @（或@attr）绑定一个scope属性到当前DOM节点的属性值。父scope影响子scope，子scope不影响父scope。

用法：<div my-directive my-name=”{{name}}”></div>

1. &(或&attr)提供方法，执行一个表达式在父scope的上下文中。directive中提供一个方法，该方法能够在父scope中执行父scope的函数。

用法：<div my-directive change-my-name=”changName()”></div>

1. =(或=attr)通过directive的attr属性的值在局部scope的属性和父scope属性之间建立双向绑定。类似scope默认（false）。

用法：<div my-directive name=“name”></div>

scope绑定方法示例：

scope:{

name:’@myName’,

age:’=’,

changeName:’&changeMyName’

}

注：”=?”后面有个问号代表该属性是可选的，如果不加”?”如果父作用域不存在该属性则会抛出NON\_ASSIGNABLE\_MODEL\_EXPRESSION异常。用于非必填属性。（没有实现异常抛出，有待研究。）

transclude: false/true,true时启用。禁止指令内部的内容被template替换,与ng-transclude一起使用。

controller: 字符串（控制器名字，查找注册在应用中的控制器的构造函数）；匿名函数（原本控制器内容移到这儿）

controller的可注入参数（一般不用）：

1. $scope: 指令元素关联的作用域
2. $element: 当前指令对应的元素
3. $attrs： 当前元素的属性组成的对象
4. $transclude: 嵌入链接函数时用于执行克隆元素和操作DOM的函数。

controllerAs: 设置控制器的别名

require： 字符串（另一个指令的名字，作为link函数的第四个参数），两个指令有些重合的方法可以放在第三个指令的controller中，原指令用require获取第三个指令，通过link链接函数的第四个参数就可以引用重合的方法。提供前缀可以改变查找控制器的行为：1.没有前缀。指令会在自己的控制器里查找，找不到则报错。2.？前缀。如果在当前指令没有找到所需的控制器，将null传给link链接函数的第四个参数。3.^前缀。如果在当前指令没有找到所需的控制器，查找父元素的控制器。4.？^组合前缀。

link(scope,iElement,iAttrs,controller): 链接函数，用于注册DOM事件和更新DOM。即操作DOM,在directive被angular编译后即执行。

scope:指令所在的作用域。

iElement: 指令所在的元素。可以调用angular封装的JQ方法和属性。如$(‘my-directive’)

iAttrs: 指令元素的属性列表。属性在link函数之间共享。

示例：<my-directive 属性1=“” 属性2=“”></my-directive>

controller：控制器实例。当前指令通过require请求的指令内部的controller。需要配合require。

## $window

浏览器窗口对象的引用，全局对象。、

$window对象可以获得浏览器窗口的的各项属性。

## $.extend()

jQuery.extend() 函数用于将一个或多个对象的内容合并到目标对象。

注： 如果只为$.extend()指定了一个参数，则意味着参数target被省略。此时，target就是jQuery对象本身。通过这种方式，我们可以为全局对象添加新的函数。

如果多个对象具有相同的属性，则后者会覆盖前者的属性值。

## 路由

路由能让用户从一个视图跳转到另一个视图。

源库的ngroute不好用，使用第三方插件ui-router。

优点：1.可嵌套（父子）2.可同一页面用不同视图

ui-view替换之前的ng-view;$stateProvider替换$routeProvider

$stateProvider.state(‘规则名’，{属性}): 创建state对象，包含路由规则的配置信息。并对state用规则名进行唯一标识。

不建议用href进行跳转

父子路由定义方法：

1. 规则名：parentName.childName 一般方法
2. 属性：parent:’parentName’

成功跳转state会触发$stateChangeSuccess事件（监听），然后模板重新渲染。

属性：

parent: 定义当前路由的父路由（不常用）

templateUrl: 模板路径

views: 多视图情况下的多模板路径对象。viewName@stateName。viewName一般是ui-view=”viewName”,可为空；stateName默认情况是父路由的规则名，可为空（表示根路由），也可以为任意的祖先路由名。注：重写冲突时子路由配置启用。

controller：控制器。向作用域写数据（可依赖注入）。

reslove: 解决项。“键值”对象，键是依赖注入时候的函数名；值是可以依赖注入的服务和其函数，返回promise或值。通常用于数据获取操作。注：子路由resolve和controller可以被依赖注入父路由resolve提供的服务；自身的controller可以被依赖注入自身的resolve。

abstract: true/false，默认为false。值为true时，如果未找到页面，则导航至默认路由。

params: 定义给当前页面传递的参数。传参可通过ui-sref和$state.go或者toStatus，toStatus保存路由参数。

## flex

flex属性用于设置或检索弹性盒子模型对象的子元素是如何分配空间的。

flex(flex-grow,flex-shrink,flex-basis)。默认0,1,auto。可修改默认值；可从父元素继承。

单值：auto对应1,1,auto；none对应0,0,auto。

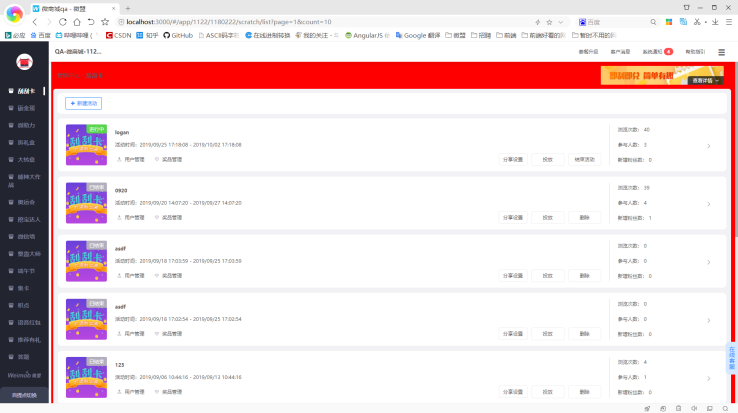
flex-grow:设置或检索弹性盒子的扩展比率（相对于其他灵活的项目进行扩展），默认0。可从父元素继承，可修改默认值。

flex-shrink:指定了flex元素的收缩规则（指定宽度大于默认宽度，多余部分按规则收缩），默认值为1。可从父元素继承，可修改默认值。

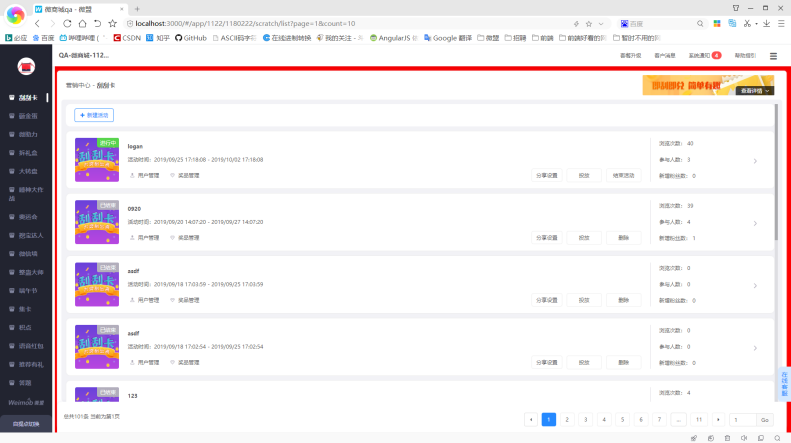
flex-basis:设置或检索弹性盒子伸缩基准值（非弹性盒对象的元素不起作用）。默认auto。可指定number;可修改默认值；可从父元素继承。

## 嵌套div实现CSS样式

删除第二个div:



删除前：



## loading-container

loading-container用于实现加载动画

## 浅拷贝和深拷贝

浅拷贝指向的是同一块内存。只是用”=”进行复制。

深拷贝是不同的两块内存

## 复制

对象的深拷贝分为合并和克隆

克隆是特殊的合并：以空对象作为目标对象，非空对象作为源对象进行合并。要求两者构造（constructor）相同.

克隆的源对象只有一个，合并的源对象可以是多个

合并：

1. Object.assign()

obj3=Object.assign(obj1,obj2)

浅合并

obj2中重复属性会覆盖obj1

obj1也会被改变，且obj===obj3

1. $.extend
   1. obj3=$.extend(obj1,obj2)

浅合并

obj2中重复属性会覆盖obj1（obj2中值为Null或undefined则忽略）

obj1也会被改变，且obj===obj3

obj1===$.extend(obj1)

$.extend(基本数据类型（Null,undefined除外），obj1)，则调用对应的基本包装类型

* 1. obj3=$.extend({},obj1,obj2)

浅合并

和（1）基本类似，只是赋给一个新对象，obj1不变

* 1. obj3=$.extend(true,obj1,obj2)

深合并

属性为对象则会把对象属性合并而非覆盖

$.extend(true,{},obj1)经常用到，将obj1赋给一个新对象。多用于给ajax请求传参数时给参数使用，创建一个新的对象传递参数，防止参数被修改。

## JS require import

require配 exports（对象） CommonJs规范（运行时加载）

import 配 export （关键词） ES6模块化（编译时加载）

## href="javascript:;"

预防页面跳转，相当于href=”#”

# 10.16.2

## export

一个模块就是一个独立的文件，文件内部的所有变量，外部无法获取。

ES6使用export导出模块。

export规定的是对外的接口，且需要与模块内部的变量建立一一对应关系。（允许外部读取内部的某个变量）

用法:

1. export var a=1;export var b=2;
2. var a=1,b=2; export{a,b}; 优先使用此方法
3. export function getX(){};
4. function getX(){};function getY(){};

export{

getX,

getX as x1,

getX as x2,

getY as y1

} //使用as重命名函数对外接口，同一个函数可以用不同的名字输出（普通变量也可以用as；函数也可以不用as）

1. export class{} //可以输出类(es6)

export的输出接口和对应内部变量的值是动态绑定，可以获取实时的值。

export需要处于模块的最外层，但可在最外层的任何位置。

## import

ES6使用import来导入export导出的模块。

import命令用于输入其他模块提供的功能。

用法：

import {x1,x2,y as yY} from ‘./test.js’

所有import的变量都是只读的，不能改写。对象可以改写其内部属性（不建议）。

from后面指定模块文件的位置可以是相对路径，也可以是绝对路径。

import命令会提升至模块首位运行，因为import是在编译阶段执行的。

整体加载：import \* as something from ‘./test’

## export default

当模块import时候可以使用任意名称（而非指定名称，也无需知道模块输出的函数名）指向export的输出的方法（无需大括号）。

用法：

export default function (){} //匿名函数或非匿名函数都可以，都视为匿名函数加载。

\*\*\*\*\*\*\*\*\*\*\*

import x from ‘./geta.js’

export default只适用于一个模块里只有一个输出为默认。

“ export default 1 “也是正确的，变量为default。

同时输入默认接口和其他方法：

import x,{a,b} from ‘test’

## export import 复合写法

一个模块中输入输出同一个模块：

export {a,b} from ‘./test’; //作用只是转发，当前模块无法使用

## 模块的继承

export \* from ‘test’ //export \* 会忽略default

export var x=1;

上面的代码转发了test，并附带了自身的x，也可重命名。

若要加载默认方法：

export \* as tt from ‘test’

export x from ‘test’

## 跨模块常量

export const A=1;

export const B=2;

import \* as constants from ‘test’

import {A,B} from ‘test’

若常量较多，可专门放置在一个文件中。

## 箭头函数

代替匿名函数，且不会创建自己的this,只会从作用域的上一层继承。

function(){} 变为 ()=>{}

语法1：

(参数1，参数2...)=>{}

语法2：

单个参数 =>{}

语法3：

没有参数：()=>{}

语法4：

.....=>{return x} 只有return可省略为 .....=>x

语法4：

.....=>({对象}) 加括号的函数体返回对象字面量

语法5：

支持剩余参数，默认参数和参数列表解构

## require

require是Nodejs的语法，用于引入模块。

‘./test’: ./表示当前目录；/表示绝对路径；无后缀表示为默认后缀.js

文件夹可以作为模块。把程序和库放在独立的文件夹，提供单一的入口指向它们。

## 检查HTML文件的变量值

任何HTML文件写的变量值都可以在同一范围内使用”{{变量}}”来查看当前变量值。

## angularjs的Promise：$q

对执行结果不确定的一种预先定义，如果成功、如果失败等。

伪代码用法：do().then(success(),error())

$q是angularJS封装的Promise

有三种状态：等待状态，完成状态，拒绝状态。

状态变更不可逆。

等待状态可变成完成或拒绝状态。

常用的三种方法

1. defer()

defer()创建一个deferred对象，这个对象可以执行几个常用的方法。

resolve: 变成完成状态

reject: 变成拒绝状态

defer.promise用于创建promise对象，promise对象用于定义then方法，then有三个方法：成功回调，失败回调，状态变更回调。

resolve(reject)中传入变量或函数返回的结果作为第一个(第二个)then方法的参数。

then又会返回一个promise对象。

then..catch..finally类似try..catch..finally

1. all()

可以把多个promise的数组合并成一个。当所有promise执行成功后，会执行后面的回调。回调中的参数是每个promise执行的结果

1. when()

当传入的参数不确定时使用该方法。可以传入一个值或者一个符合promise标准的外部变量。

## setInterval

setInterval(function(){},time): 会不停地调用函数，直到clearInterval()被调用或窗口被关闭。setInterval返回的ID值可用作clearInterval的参数。

## 组件和模块

组件: 代码复用，强调复用

模块: 一个功能应该封装成一个模块，强调封装

## tableParams-列表数据

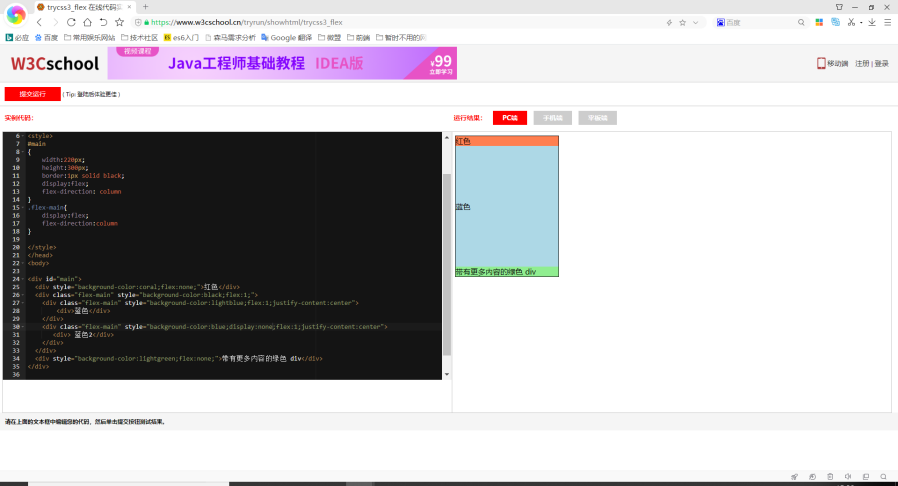
HTML文件一般用 vm in $data.list获取列表数据

可以用{{}}获取值或者借用click之类的函数从HTML文件向JS文件传递列表数据。

## 弹性盒子的实战1

上下固定，中间弹性伸展：第一个和末尾设置flex:none,中间设置flex:1。

当有一个时候居中，当有两个时候在纵轴上平均分布（同理水平垂直居中）：大框架小框架都设置为display:flex，字体div外包的弹性盒子设置子元素垂直居中。



## 脏检查（双向绑定）

angular的双向绑定机制被称为脏检查。

数据的双向绑定可以分为两个阶段，从用户到内存；从内存到用户。

### 从用户到内存

从用户到内存主要是一些触发事件而引起的数据传递。

### 从内存到用户

浏览器没有提供数据监测的API，所以任何内存数据的变动都无法被Listen，也就无法处理。

于是我们对大部分会产生数据变动的事件进行封装，即触发事件。每次事件发生并执行完成后，检查一遍数据的变化，若有变化，则执行对应的回调函数（view的数据展现、函数处理等）。

检查会将所有“注册”过的值检查一遍。一次检查为一个周期，每次至少两个周期。因为要第二遍确认第一遍时是否又有数据变动，直到最后两次数据一致，则停止检查（递归）。考虑到死循环，上限设置为10遍。

$scope.apply() & $scope.$digest() 即遍历检查的function。

脏检查是$digest执行的，$apply是对$digest的简单封装。一般无需调用，angularjs基本已做好封装。如果要调用，优先调用$apply，这是常用公开接口。

内部所有绑定表达式都会转换为$scope.$watch(),每个$watch记录了上一次表达式的值。

示例：ng-bind=”a” =>自动转换 $scope.$watch(‘a’,callback)

联系到$timeout,$timeout里有封装$digest。每次定义涉及数据的函数时，可以最外面包裹一层$timeout，保证函数使用到的是最新的数据。

## 实际长度

实际长度和指定长度区别在于padding，若padding+内容长度大于指定长度，则实际长度也会大于指定长度。

注意padding

## 层级选择器

CSS有优先级 层级选择器有高优先级

首先在浏览器里显示越上面代表优先级越高

有层级选择器的比单纯的类优先级高

## ！important

提高指定样式规则的应用优先权，使该条样式声明具有最高优先级。

## 阻塞和非阻塞

阻塞是指纯粹按顺序执行

非阻塞是指请求资源时程序依旧执行，但挂个回调函数等，等资源到位则调用回调函数。

## nodejs事件

nodejs所有的异步I/O操作在完成时都会发送一个事件到事件队列。events.EventEmitter用于事件分发和事件监听

nodejs所有事件机制都是用设计模式中观察者模式实现。  
 事件驱动：当web服务模块收到请求，就把请求处理并关闭，然后去服务下一个请求。模块会一直接受请求而不等待任何读写操作。整个驱动模型会有一个主循环来监听事件，检测到事件时触发回调函数。

## CSS3 appearance

使div元素看上去像其余模型

实例：

-moz-appearance:button; //firefox

-webkit-appearance:button; //safari,chrome

其余主流浏览器都不支持

属性值：icon(将元素呈现为小图标)、window(将元素呈现为视口)、button(将元素呈现为按钮)、menu(将元素呈现为一套供用户选择的选项)、field(将元素呈现为输入字段)

## 字体抗锯齿渲染

使字体看起来更清晰舒服

用法：

-webkit-font-smoothing: antialiased; /\*chrome、safari\*/

属性值：none(低像素文本友好)、antialiased(抗锯齿很好)、subpixel-antialiased(默认)

-moz-osx-font-smoothing: grayscale; /\*firefox\*/

## Font Awesome

Font Awesome是一款基于css框架的网页字体图标库

提供矢量图标，可定制大小，颜色，阴影

可以用自定义图标设置成图标库

## 函数补充

函数的名字仅仅是一个包含指针的变量。即使在不同的环境下，全局的test函数和a.test()指向的仍然是同一个函数

## 在一个js文件调用另一个js文件内容（例如jq）

js文件的加载实际上是把js文件一个一个的加载到同一个页面。 如果一个js文件要调用另一个js文件，则把被调用js文件的调用语句写在前面即可。

## vscode实现ES6

vscode不能完全实现ES6的功能，如import等

所以需要在vscode借用babel将es6代码文件转换为es5代码文件，再进行编译

## ES6引入less文件（待测试）

import @url(地址)

## 如何循环

forEach一般用于遍历对象

for用来遍历数

for in遍历对象的所有可枚举属性（包括原型链的属性，但顺序不确定），多用于遍历不知属性数量的对象

for of能响应break,continue,return,支持Map,Set对象，支持字符串，但不支持普通对象。

for循环时，如果数组长度在循环时不发生变化，则应该提前存储起来：for(let i = 0; len = arr.length,i < len;i++)

## 访问对象属性

let a = {x:1,y:2}

a[‘x’]

## 屏幕自适应

方法一：calc() 对应元素用屏幕总高度减去固定元素高度

方法二：flex布局 有flex：属性的子元素会填满父元素的剩余空间