目录

[2014-3-3 7](#_Toc384848473)

[Google HTML/CSS Style Guide 7](#_Toc384848474)

[General Style Rules 7](#_Toc384848475)

[General Formatting Rules 7](#_Toc384848476)

[General Meta Rules 7](#_Toc384848477)

[HTML Style Rules 8](#_Toc384848478)

[HTML Formatting Rules 8](#_Toc384848479)

[CSS Meta Rules 9](#_Toc384848480)

[2014-3-4 9](#_Toc384848481)

[CSS Style Rules 9](#_Toc384848482)

[CSS Formatting Rules 10](#_Toc384848483)

[2014-3-5 10](#_Toc384848484)

[规则一 11](#_Toc384848485)

[规则二 11](#_Toc384848486)

[规则三 11](#_Toc384848487)

[规则四 11](#_Toc384848488)

[附A：省略规则 11](#_Toc384848489)

[html 11](#_Toc384848490)

[head 11](#_Toc384848491)

[body 11](#_Toc384848492)

[li 11](#_Toc384848493)

[dt 11](#_Toc384848494)

[dd 12](#_Toc384848495)

[p 12](#_Toc384848496)

[rt 12](#_Toc384848497)

[optgroup 12](#_Toc384848498)

[option 12](#_Toc384848499)

[colgroup 12](#_Toc384848500)

[catpion 12](#_Toc384848501)

[thead 12](#_Toc384848502)

[tbody 12](#_Toc384848503)

[tfoot 12](#_Toc384848504)

[tr 12](#_Toc384848505)

[td 12](#_Toc384848506)

[th 12](#_Toc384848507)

[2014-3-6 13](#_Toc384848508)

[Introduction to HTML 13](#_Toc384848509)

[Introduction to CSS 13](#_Toc384848510)

[2014-3-7 15](#_Toc384848511)

[Css selector 15](#_Toc384848512)

[选择器 15](#_Toc384848513)

[多选择 15](#_Toc384848514)

[\*选择器 15](#_Toc384848515)

[子选择器 15](#_Toc384848516)

[兄弟选择器 15](#_Toc384848517)

[Class and id 15](#_Toc384848518)

[伪类 16](#_Toc384848519)

[CSS Element Positioning 16](#_Toc384848520)

[Javascript 18](#_Toc384848521)

[2014-3-8 18](#_Toc384848522)

[Javascript版本经历 18](#_Toc384848523)

[Javascript调试 18](#_Toc384848524)

[文档约定 18](#_Toc384848525)

[字体说明 18](#_Toc384848526)

[Javascript概述 18](#_Toc384848527)

[Values, variables, and literals 19](#_Toc384848528)

[Expressions and operators 20](#_Toc384848529)

[Operators 20](#_Toc384848530)

[Expressions 20](#_Toc384848531)

[正则表达式 21](#_Toc384848532)

[javascript Style Rules 23](#_Toc384848533)

[2014-3-9 23](#_Toc384848534)

[2014-3-10 25](#_Toc384848535)

[对象 25](#_Toc384848536)

[取值 25](#_Toc384848537)

[引用 26](#_Toc384848538)

[原型 26](#_Toc384848539)

[反射 26](#_Toc384848540)

[给类型增加方法 26](#_Toc384848541)

[回调 26](#_Toc384848542)

[闭包 26](#_Toc384848543)

[2014-3-11 27](#_Toc384848544)

[This 27](#_Toc384848545)

[document 27](#_Toc384848546)

[window 27](#_Toc384848547)

[call() 27](#_Toc384848548)

[apply() 27](#_Toc384848549)

[多态 27](#_Toc384848550)

[CSS-Getting\_started 27](#_Toc384848551)

[2014-3-12 28](#_Toc384848552)

[html5 28](#_Toc384848553)

[html5 Input form 29](#_Toc384848554)

[input list 29](#_Toc384848555)

[input step 29](#_Toc384848556)

[盒子模型 29](#_Toc384848557)

[布局 29](#_Toc384848558)

[2014-3-13 30](#_Toc384848559)

[2014-3-16 30](#_Toc384848560)

[2014-3-17 31](#_Toc384848561)

[垃圾收集 31](#_Toc384848562)

[Array 31](#_Toc384848563)

[join 31](#_Toc384848564)

[push 31](#_Toc384848565)

[pop 31](#_Toc384848566)

[shift 31](#_Toc384848567)

[reverse 31](#_Toc384848568)

[sort 31](#_Toc384848569)

[sort 32](#_Toc384848570)

[concat 32](#_Toc384848571)

[slice 32](#_Toc384848572)

[splice 32](#_Toc384848573)

[every 32](#_Toc384848574)

[some 32](#_Toc384848575)

[filter 32](#_Toc384848576)

[forEach 32](#_Toc384848577)

[map 32](#_Toc384848578)

[原型继承 32](#_Toc384848579)

[2014-3-18 33](#_Toc384848580)

[浏览器CSS定制 33](#_Toc384848581)

[DOM 33](#_Toc384848582)

[document.domain： 33](#_Toc384848583)

[特殊集合 33](#_Toc384848584)

[特性： 33](#_Toc384848585)

[创建元素 33](#_Toc384848586)

[动态添加脚本： 34](#_Toc384848587)

[动态创建link 34](#_Toc384848588)

[表格操作 34](#_Toc384848589)

[判断元素是否获取焦点（HTML5） 34](#_Toc384848590)

[当前文档是否加载完毕：readyState 34](#_Toc384848591)

[document.compatMode 34](#_Toc384848592)

[insertAdjacent： 35](#_Toc384848593)

[事件冒泡处理： 36](#_Toc384848594)

[DOM2级事件处理程序 36](#_Toc384848595)

[IE事件处理程序 36](#_Toc384848596)

[事件 37](#_Toc384848597)

[contextmenu 37](#_Toc384848598)

[beforeunload 37](#_Toc384848599)

[DOMContentLoaded 37](#_Toc384848600)

[haschange 37](#_Toc384848601)

[委托 37](#_Toc384848602)

[内存清理： 37](#_Toc384848603)

[模拟事件 37](#_Toc384848604)

[表单 38](#_Toc384848605)

[表单字段共有属性和方法 38](#_Toc384848606)

[2014-3-19 39](#_Toc384848607)

[HTML5约束验证 39](#_Toc384848608)

[required 39](#_Toc384848609)

[type 39](#_Toc384848610)

[stepUp(number),stepDown(number) 39](#_Toc384848611)

[pattern 39](#_Toc384848612)

[checkValidity 39](#_Toc384848613)

[validity 39](#_Toc384848614)

[noValidate 39](#_Toc384848615)

[选择框 39](#_Toc384848616)

[富文本编辑 40](#_Toc384848617)

[document.execCommand(p1,p2,p3) 40](#_Toc384848618)

[命令可执行检测方法： 40](#_Toc384848619)

[跨文档消息传递： 40](#_Toc384848620)

[拖放 40](#_Toc384848621)

[dataTransfer： 40](#_Toc384848622)

[媒体 41](#_Toc384848623)

[匿名闭包与自引用 41](#_Toc384848624)

[2014-3-20 41](#_Toc384848625)

[XHR 41](#_Toc384848626)

[XHR的方法 42](#_Toc384848627)

[XHR的属性 42](#_Toc384848628)

[HTTP 42](#_Toc384848629)

[GET 43](#_Toc384848630)

[POST 43](#_Toc384848631)

[进度事件 43](#_Toc384848632)

[跨域访问 43](#_Toc384848633)

[2014-3-21 43](#_Toc384848634)

[跨域访问： 43](#_Toc384848635)

[FF和其他浏览器的event： 44](#_Toc384848636)

[阻止FF的拖拽重定向脚本 44](#_Toc384848637)

[2014-3-22 44](#_Toc384848638)

[CSS鼠标样式 44](#_Toc384848639)

[2014-3-23 45](#_Toc384848640)

[常用的css命名规则： 45](#_Toc384848641)

[注释的写法： 46](#_Toc384848642)

[id的命名： 46](#_Toc384848643)

[class的命名： 47](#_Toc384848644)

[2014-3-24 48](#_Toc384848645)

[JSON-js 48](#_Toc384848646)

[stringify() 48](#_Toc384848647)

[JSON.parse 49](#_Toc384848648)

[E4X 49](#_Toc384848649)

[XML对象 49](#_Toc384848650)

[2014-3-25 50](#_Toc384848651)

[window.onerror： 50](#_Toc384848652)

[判断网页致命错误的依据 50](#_Toc384848653)

[javascript高级技巧 51](#_Toc384848654)

[防篡改对象 51](#_Toc384848655)

[constructor属性 51](#_Toc384848656)

[自定义事件（C#中的delegate） 51](#_Toc384848657)

[解耦 52](#_Toc384848658)

[性能 52](#_Toc384848659)

[部署 53](#_Toc384848660)

[2014-3-26 53](#_Toc384848661)

[javascript’s offline 53](#_Toc384848662)

[manifest文件语法 53](#_Toc384848663)

[applicationCache对象 54](#_Toc384848664)

[数据存储 55](#_Toc384848665)

[web存储机制 56](#_Toc384848666)

[IndexedDB 56](#_Toc384848667)

[2014-3-27 57](#_Toc384848668)

[IndexedDB 57](#_Toc384848669)

[IndexedDB的事件 57](#_Toc384848670)

[indexed Database对象 57](#_Toc384848671)

[IDBDatabase 58](#_Toc384848672)

[2014-3-28 58](#_Toc384848673)

[并发问题 60](#_Toc384848674)

[2014-3-31 60](#_Toc384848675)

[AngularJS 60](#_Toc384848676)

[双向绑定 61](#_Toc384848677)

[AngularJS应用的解析 61](#_Toc384848678)

[2014-4-1 68](#_Toc384848679)

[Q&A about AngularJS 68](#_Toc384848680)

[表单 69](#_Toc384848681)

[Ember 69](#_Toc384848682)

[2014-4-2 69](#_Toc384848683)

[AngularJS 69](#_Toc384848684)

[$location 69](#_Toc384848685)

[$location服务的配置 70](#_Toc384848686)

[$location的方法 70](#_Toc384848687)

[bootstrap 70](#_Toc384848688)

[Animations 71](#_Toc384848689)

[HTML Compiler 72](#_Toc384848690)

[module 72](#_Toc384848691)

[2014-4-3 72](#_Toc384848692)

[概念 72](#_Toc384848693)

[Controllers 76](#_Toc384848694)

[Working with CSS 76](#_Toc384848695)

[2014-4-4 77](#_Toc384848696)

[DI 77](#_Toc384848697)

[指令（directive） 77](#_Toc384848698)

[E2E Testing 80](#_Toc384848699)

[表达式 82](#_Toc384848700)

[Filter 82](#_Toc384848701)

[表单（Form） 83](#_Toc384848702)

[2014-4-6 83](#_Toc384848703)

[IE兼容 83](#_Toc384848704)

[Modules 84](#_Toc384848705)

[2014-4-9 85](#_Toc384848706)

[作用域Scope 85](#_Toc384848707)

[作用域层级 85](#_Toc384848708)

[作用域的声明周期 85](#_Toc384848709)

[作用域和指令 86](#_Toc384848710)

# 2014-3-3

# Google HTML/CSS Style Guide

## General Style Rules

使用省略的路径。

例如：<script src=”//localhost/js/test.js” type="text/javascript">

## General Formatting Rules

使用两个空格作为格式规范。

例如： <div>

<br/>

</div>

注：请勿使用Tab等作为通用格式规则。

使用小写拼写规则：

例如：<div style=”color:#fff”>

</div>

注：特殊字符可以使用大写 例如<![CDATA[…]]>.CDATA是针对xhtml中表示以下内容作为文本处理的一个关键字。xhtml和html的区别是xhtml是html像xml的一个过渡语言，它比html严谨性会高点，然后基本语言都还是沿用的html的标签，只不过废除了部分表现层的标签，同事在标准上要求高了点比如标签的严格嵌套，标签结束等等

清除结尾不必要的空格

例如： <p>This is test text. </p>，这样的空格是应该清除掉的。

## General Meta Rules

编码：所有的文档都要显示标注utf-8编码。没有指定特殊编码的都默认成utf-8。

注：除了要注意css和html编码之外，还要注意你的文件保存编码形式一定也要是utf-8

注释：路径、目的以及选择此解决方案的原因。

TODO：将行为标注在TODO之后，并且用“：”分割。

例如：<!-- TODO：Action -- >

## HTML Style Rules

HTML5:使用HTML5的文件头既：<!DOCTYPE html>

注：在HTML5中不需要将空白的标签关闭，例如：<br>不再需要<br/>

使用有效的HTML：尽量使用有效的标签，除非是特殊文件，可是使用<http://validator.w3.org/nu/>进行检测。

使用正确的元素：根据元素的目的选择正确的元素。

例如：一个锚点正确应该使用<a></a> 而不是 div或者其他什么元素。

多媒体的紧急处理方法：使用说明属性。

例如：<img src=”” alt=”说明属性”> 疑问：canvas和video元素使用什么作为说明？title么？

答：video标签可以使用title记录额外信息，便于爬虫识别，

分离式结构化：

保持着结构行为和说明这几个部分，尽量减少之间的影响，最好是保持在3到0个影响数。

确保“行为”在脚本遇见出现，样式在css中出现。

分离元素中的样式，尽量减少link数量。

疑问：如果将过多的样式写在一个css文件里面那样会导致第一次打开页面的时候速度降低，怎么解决这个问题？

答：可以采用优化和压缩策略。但是我还是会带着这个疑问在我的开发过程中，直到我能亲自解开这个谜底。

没必要实体引入：没必要使用&ldquo;（“）&rdquo;（”）&mdash;（-）&#x263a;（☺）这些符号，唯一例外的是<和&符号还有无间断空格。

省略规则：为了文件更小加载更快可以省略掉一些元素，省略规则：<http://www.whatwg.org/specs/web-apps/current-work/multipage/syntax.html#syntax-tag-omission>

Type属性：对于标准的css样式和javascript忽略掉type属性，除非是非标准文件。

## HTML Formatting Rules

块元素：为每一个块元素新起一行。例如table ul 等等。如果块元素对应的子元素也要

进行缩进。如果列表元素没有值，也可以放在同一行。

例如：<table>

<tr>

<td></td>

</tr>

</table>

双引号：使用双引号来包裹属性值。

## CSS Meta Rules

注释：如果可能就在每一组样式上面加上注释，格式如下：

/\*注释\*/

.teststyle{}

# 2014-3-4

## CSS Style Rules

有效性：尽可能使用有效的css，除非为了一些浏览器bug使用的Hack。

命名规则：使用具有意义的名词，最好是能够表达其目的。通用的名称只是为了和具有意义的混合使用。在可能的情况下，让名称尽量短。但是要表达清楚其目的。

选择器：在没有必要时，就不要加上父类的选择器。这样有损性能。

例如：table#login{} 这样是没有必要的，id是login的在一个页面中也就只能存在一个。

速记属性：尽量使用简短的属性表达。

例如：.test{

margin-top:10px;

margin -left:10px;

margin -bottom:10px;

margin -right:10px;

}

可以简写成：.test{

margin:10xp;

}

0和省略：当属性值为0时，不需要单位，除非特别要求。属性值为0.x的时候，只需要写.x即可。

例如：font-size: .8em;

十六进制表示：在可能的情况下，将使用三位数字表示十六进制。

例如：color:#aabbcc 可以表示成 color:#abc

注：如果coclor:#0f45df 则无法用三个数字表示。

前缀：在引入外部项目或者将被外部项目引用的css文件中可以加入项目前缀，一边是“命名空间-名称”的格式。如果内部或者是小型项目尽量不要用咯，可以减少冗余。

名词连接：在命名过程中，如果出现需要两个词语来表示，可以使用“-”连接，不要使用“\_”连接。更不要不分割。

Hack：尽量避免使用hack，先去尝试一些其他方法。

## CSS Formatting Rules

声明顺序：按照字母表排序，如果出现浏览器类型的前缀，则忽略前缀，考虑实体。

例如：#id{

color:#abc;

margin:10px;

}

块缩进：当整个代码块位于父级一下时，那么按照规则整块都要缩进。

终止：以“；”作为终止标志。

注：不要省略。虽然最后一个省略了浏览器也不一定报错。

属性与值：属性和值之间应该用“ ”割开。

例如：color: red;

大括号：大括号应该是跟在声明之后。语法类似java。

例如：#id{

margin: 0;

}

注：到此为止发现之前写的好多例子都错了。注意修改。

选择器占据新行：当多个选择器使用同一样式时，注意每个选择器都占据一行。

例如：a,

p {

color: red;

}

分离：在两个规则之间留有一行或者两行空白。

引号：在css样式中使用‘’代替“”，url是不需要‘’号的。具体规则：<http://www.w3.org/TR/CSS21/syndata.html#charset>

# 2014-3-5

主要是复习了一下前两天的google样式规范。按照规范要求去编写html和css，也通过此次编写的过程中，对html和css的学习做一个指导，通过这几次练习之后，深刻的领悟到自己在布局方面的欠缺，所以接下来的学习重点在于css的盒子模式，定位以及html的标签熟知度。

以下为前两天学习回顾：

在开发过程中存在众多规则，这些规则是为了约束，达到统一的目的。这样可以减少维护成本，也会是整个项目看上去更加美观。

规则一：引用，引用外部文件，使用省略http或者是https的路径。在引用媒体文件时应该注意要有对应的说明性属性。

规则二：格式，正确缩进方式是两个空格，小写拼写，清除尾行多余空格。

对于块结构或者是列表都要新起一行，在html中要使用双引号。Css样式声明顺序，按照字母表的顺序声明，也要注意缩进规则，以分号结束，注意在声明样式时，请将Name和{之间用空格割开，并且注意缩进方式，选择器也要新起一行，在两个规则之间使用空行割开。在css中使用单引号代替双引号。

规则三：编码注释，统一编码形式（默认utf-8），特别是文件保存的编码格式，在html中使用<!—注释--> 作为注，使用TODO标记即将要完成的事情或者是行为。在css中使用/\*\*/进行注释。

规则四：样式，html中除了要使用正确的元素之外，还要注意文件类型，要使用html5（<!DOCTYPE html>）文件类型，在适当的时候选择适当的元素，不要一味的使用div或者是table等，尽量减少文件之间的相互影响（3~0规则），去掉文件中所有可选标签，如果是标准的css或javascript则不需要type属性。 Css中主要体现在命名规则，选择有意义的名称，减少选择器的过滤条件，没有必要时，请不要添加父类选择，整体要求在于简洁，所以要多使用快速记忆标签，也要记得0省略规则（0后面的单位省略，0.x的0省略）连接符使用“-”，不要使用“\_”。在有可能的情况下不要使用hack。

# 附A：省略规则

html一个html元素的尾标签没有紧跟着一个内容，那么是可以忽略的。

head如果head的开始标签是第一个元素或者是head标签是空的，可以省略。

如果head的结束标签前面没有空格或者内容可以省略。

body如果body元素内是空白的，body的起始标签可以省略，或者是其实标签前面没有空格或者是内容。

如果body元素的结尾标签前面没有空格或者是内容可以省略。

li一个li元素的结尾标签是可以省略的，如果这个li元素存在兄弟节点或者父节点没有内容。

dt一个dt元素如果存在兄弟元素或者dd元素，可以省略结尾标签。

dd一个dd元素如果存在兄弟元素或者是dt元素或者父节点没有内容可以省略结尾标签。

p一个p元素在很多情况下都可以省略结束标签，下面我罗列一下可以省略的情况

如果在p元素后面是address,article,aside,blockquote,div,dl,fieldset,footer,form,h1~h6,header,hgroup,hr,main,menu,nav,ol,p,pre,section,table,ul.

但是如果后面是a或者父节点有内容则不能省略。

rt一个rt（rp）元素如果存在兄弟元素或者是rp元素，或者是父节点没有内容，可以省略结尾标签。

optgroup一个optgroup元素紧跟个optgroup元素或者是父元素没有其他内容，可以省略结尾标签。

option一个option元素紧跟着一个option或者是optgroup或者是父元素没有其他内容时，结尾标签可以省略。

colgroup一个colgroup元素，如果该元素下的第一个标签是col标签，则此标签的开始标签可以省略。

一个colgroup元素的结尾如果不是空格或者是注释，那么结尾元素可以省略。但是如果元素内容为空不可以省略。

catpion一个caption元素的结尾如果不是空格或者是注释，结尾百千可以省略。

thead一个thead元素的结尾如果紧跟是tbody或者是tfoot元素，那么结尾标签可以省略。

tbody一个tbody元素的内的第一个元素是tr，他的开始标签可以省略，如果不是紧跟在tbody，thead，tfoot的后面，结尾标签是被省略的。

一个tbody元素后面是tbody或者是tfoot，或者是父元素没有其他内容时，结尾标签可以省略。

tfoot一个tfoot元素后面是tbody或者是父元素没有更多内容，结尾标签可以省略。

tr一个tr后面还是tr，那么上一个tr的结尾标签可以省略。

td一个td后面如果是td或者是th或者父元素没有其他内容，结尾标签可以省略。

th一个th后面如果是td或者th或者是如元素没有其他内容，结尾标签可以省略。

# 2014-3-6

## Introduction to HTML

主要学习一些HTML的基础知识和基础的标签使用，并且在[www.codecademy.com](http://www.codecademy.com)做了练习题，主要学习的内容有

<!DOCTYPE html>,html5文件标志。

<a>,link元素，也有人喜欢叫它锚点,主要的属性是href。

<img>, 图片元素，可以显示图片，主要的属性是src，用于指向文件路径，但是根据google样式规则要求，应该要注意alt属性。

<h1>~<h6>,标题元素，一般在写文章的标题是使用到，它有自己的样式，h1最大，h6最小。

<p>, 段落元素，在写段落文字的时候使用。

<body>,<html>,<head>,这三个元素在页面中最多只能有一个，文件是这么说的，但是我不这么认为，因为在html中还可以引用其他的html，所以我认为这些元素起始是可以嵌套拥有的。

以上讲解的元素比较少可以在网站<http://www.w3school.com.cn/html5/index.asp>上学习了一些其他元素。

## Introduction to CSS

css的基础知识，主要内容有

选择器，在今天的联系中只有元素选择器，基本语法是

元素 {

/\*样式\*/

}

e.g. div {

color: #abc;

}

除此之外还要知道有 id选择器，语法如下

#id {

/\*样式\*/

}

class选择器，语法格式

.classname {

/\*样式\*/

}

基础样式表达形式： 属性名称： 属性值

主要接触到的属性有

color: #c00;

background-color: #000;

background-image: url(../1.jpg);

注：background-xxx这样的属性最好是使用快速记忆标签，这样写起来比较简单，传输成本也少很多。

font-size: 1em;

font-family: serif;

注：font-xxx的优化同background-xxx.

border-style: solid;

border-width:1px;

....

注：优化方式同boackground-xxx

text-decoration: none;

注：这个属性有5个值none，underline，overline，line-through，blink。其实text还有好多属性，在这里就不一一罗列了。

ps：在学习网站上发现有一些问题，我个人认为他的一些编码风格是不符合google样式规则要求的，比如，css属性值要按照字母顺序排序，属性值中只能存在小写，尽量使用三位数字表示6位数字的颜色等。

晚补：

List：主要有ol和ul两种。Ol是不存在序号的，ul如果没有设置样式，会根据浏览器不同显示出来的序号样式不一样，为了让样式统一我们可以使用list-style设置。

两个列表的item都是使用li实现。

嵌套样式：可以在标签中嵌套style样式。语法同css中。

table 元素存在着一些子标签，主要有

tr 行 就是一个table的行。

注：tr不能直接存储数据。

td 列 也可以说是数据元素。用于存内容 数据的。

一个table就相当于是一个html，在每一个table中都有thead元素和tbody元素，但是这两个元素不是必须的。

th表头单元格 - 包含表头信息（由 th 元素创建），元素内部的文本通常会呈现为居中的粗体文本。

colspan属性，列合并。

rowspan属性, 行合并。

em 斜体文字。Address也是斜体哦。

Div 块状霸道的元素。（独霸一行）

Span 元素，多为内含元素，为了是一段文本中的某些汉字产生特殊的样式。

# 2014-3-7

## Css selector

选择器:这里面有很多种，首先说说我学过的这几种

#Id选择器，通过元素的id指定元素样式

.class 选择器 通过元素的class指定元素的样式。

Tags 选择器 通过元素的“元素名”指定元素样式。

Id选择器只能指定固定的元素。

Class选择器可以自定所有class是这个class的元素。

Tags选择器则是指定所有tags相同的元素样式。

多选择：也叫子选择。格式 父级1 父级2 父级3 …{} 这样可以自定某一个范围内的一些样式。

\*选择器：选择全部

子选择器：“>”指的是该元素的子元素，而非孙子元素或者是更下级的元素。

兄弟选择器：“+” 指该元素的兄弟元素。是第二个元素的样式发生改变而非第一个。

## Class and id

Class样式设置主要是通过.class来完成。

例如<div class=”classname”></div>

在css中 .classname {

/\*样式\*/

}

ID样式设置主要是通过元素的ID进行样式控制

例如 <div id=”idname”></div>

Css 文件: #idname {

/\*样式\*/

}

伪类：

a:link,a:visited,a:hover

a:link {color: #000} /\* 未访问的链接 \*/

a:visited {color: #fff} /\* 已访问的链接 \*/

a:hover {color: #aaa} /\* 鼠标移动到链接上 \*/

a:active {color:#bbb} /\*选定连接时\*/

注：hover一定要在link和visited后面才会有效果，active一定要在hover后面才会有效果。

first-child : 第一个子元素。按照英文名字是这样翻译的。但是我却认为他应该是第一个元素的意思。

比如：

<p>t1

<p>t1

<p>t1

<p>t1

Css: p:first-child {

color:red;

}

这样其实是第一个p元素的文本变成了红色。

注：要注意此节点的选择，区分该伪类与“>”和“+”的区别。

nth-child（num）

注：这个伪类有一个强大的功能就是可以通过（num）连选择具体是第num个元素改变css

这个num很有意思，这个num并不是p：nth-child 中的p的个数，而是他父类元素下面的子元素的个数排序。

小结：学习了css选择器之后发现关于id class tags选择器的用法可以有多种组合使用，并且浏览器在解释的时候下面的代码样式总会覆盖之前的样式。

## CSS Element Positioning

Display：显示样式，样式有很多种主要记录几种常用的吧。

Inline 默认值。内联元素。前后没有换行符，也就是不像div那样独占一行。

Block：块状，可以使一下元素拥有类似div一样的霸道属性，可以独占一行。

None：不显示。

Inline-block：行内块元素。让元素在同一行 但是却还是块元素。

Margin：元素与父级元素的距离。

当设置margin：auto的时候，该元素会水平居中。

注：在此列出垂直居中的几种方法1：line-height方法，2 vertical-align:middle

Padding：元素与子元素的距离。

Border：边框 是元素的边界值。

疑问：当父元素足够大的时候如果我设置margin：10px 10px 5px 5px;而且该元素还是足够的小，那么该元素为什么之后left和top其左右呢，而bottom和right没有效果呢？

个人认为结果：是因为html默认的流布局是left的，如果改成right会有不同效果。

Clear：清除div的霸道特性，也就是清除block属性。主要有both left right none inherit五个值。

Position：absolute 绝对定位，相对于static以外的第一个父元素定位

Fixed 绝对定位相当浏览器。

Relative：相对定位。

Static：默认值 没有定位。正常流布局。

Inherit：从父元素集成position属性值。

z-index：是指该元素的z轴上的显示顺序，当值越大的时候就会显示在小的上面。

小结：学习了定位之后发现为了达到同样的效果起始方法有很多种，要学会变通。

总结：至此已经学过所有的css和html的基础知识了。这些东西虽然以前也用过，但是在此学习的时候才发现原来还有很多我不知道的东西。再多说都是没有意义的，动手去写练习吧，the more the better。

# Javascript

# 2014-3-8

Javascript版本经历：1.0~1.8

注：关于每个版本的更新详细和具体内容没有详细了解，在对javascript有一定认识之后再回过头来看看这些，会有一些帮助的。

Javascript调试：firebug。

文档约定：<http://server.domain/path/file.html>;

http://服务器(或者是www).网站名（domain）/服务器路径/文件

字体说明（在这个文档中，不同的字体有着自己的意义）：

1 宽体：样例代码 api 文件名 路径 以及html标签等。宽斜体：嵌入的代码。

2 斜体：书名，强调或者是变量。

3 粗体： 专用词语。

Javascript概述：

一个非独立运行的语言，可以寄宿在浏览器上运行，是一个面向对象的脚本语言。

可以响应客户端的操作，也可以和服务器进行通信，例如请求关系数据库等，它的LiveConnect功能可以实现它与java的相互操作。

LiveConnect: <https://developer.mozilla.org/zh-CN/docs/LiveConnect> 嵌套在java开发过程中。

Java和javascript的不同

Javascript没有静态的和强类型。

Javascript是基于原型模型，而不是对象模型，所以可以动态单继承。

注：在网络上查到还可以多继承，关于继承在学习过程中要注意，并且要查一下资料。

Javascript没有访问级别的限制，也就是没有public等等。

Javascript是小的并且是动态的语言，例如dbase和HyperTalk。

Javascript和ECMAscript:

ECMAscript是标准化的javascript语言。是遵从ECMA组织的。

Javascript中不存在全局对象。但是有顶级的function和属性（不知道是不是这个意思呢，在以后的学习中再领悟一下吧。）

Javascript中的构造函数都是有参数的，没有参数的数字构造函数就是0，没有参数的字符串构造函数就是””空字符串。

## Values, variables, and literals

Javascript有以下五种原始值类型：

Number 数字 1 ，0.1，…

Boolean true|false

String “this is test text”

null 这个要注意区分大小写

undefined 当一个变量声明之后为赋值也就是此类型。

变量：变量声明与赋值

var temp=1; var boolvalues=true; var stringvalues=”tt”; var nullvalues=null; var undefined;

以上的这几种赋值方式得到对应的值。

Javascript在做+ or – 运算的时候，如果存在字符串和数字，在都会首先按照字符串做计算，如果无法计算则在按照数字类型进行计算。

字符串强转 数字类型。

parseInt（“”） parseFloat（“”）

变量命名：必须是字母 \_ $ 开始，后续字符中可以有数字。

注：区分大小写，也可以使用转义字符

赋值：undefined类型的强转boolean时 对应为false；当被强转为数字时，为NaN

null类型强转boolean时，是false，强转number时 是0

javascript块作用：在javascript中可以使用，但是变量并不受块作用域影响，变量只被function影响或者是被document影响。

Javascript变量声明的作用范围和声明顺序。

全局变量：全局变量是声明在所有的function之外。也可以使用window.XXX调用。

常量: 关键字 const，常量在声明之后它的值将不能被改变。在同一个document中不能存在一个函数或者是另外一个常量的名称与其中一个常量的名称相同。

Array：数组。

Boolean：布尔。

Integers：整形

分为十六进制0x\_\_\_ 十进制0\_\_\_ 八进制 还有二进制。

Floating-point：浮点型 float

Object：对象，我们就叫他object吧。

表现一个对象使用function关键字，也可以使用类似json格式。

在取其属性值的时候要注意如果属性名称是数字和字母的区别。

String：字符串。

转义字符：

\b退格 \f翻译成“换”，我没有测试出来到底是怎么用，百度说是换页。待再测。

\n换行 \r“enter” \t“tab” \v 垂直制表符 \’ \” \\（不解释）

\XXX 三个八位数 \x XX两个十六位 \uXXXX四个十六位数字

针对编码在此不赘述。

## Expressions and operators

### Operators

算数运算符 + - \* / % 等等

逻辑运算符： == && || 等等

“>> 右移,高位补符号位” 这里右移一位表示除2  
“>>> 无符号右移,高位补0”； 与>>类似  
“<< 左移” 左移一位表示乘2，二位就表示4，就是2的n次方

不一一赘述，因为所有语言的运算符几乎都一样，一下说一下javascript中的“==”和“===”的区别吧。

==： 只进行值比较，如果值相同，那么就是true

===：除了比较值，还比较类型

e.g. if(“123”==123) //true

if(“123”===123) //false

delete运算符：与其说是运算符不如看成一个方法，可以删除对象或者对象的属性或者是array的中的值，返回true或者是false。

In 运算符：如果指定的属性是对象的属性返回true。

Instanceof 运算符：如果指定对象是此类型，就返回true。

This 当前对象。

Typeof（） 或者是typeof xxx 返回xxx的类型，并且义字符串形式呈现。

Void express 或者是void （express） 无返回值表达式。

### Expressions

表达式=变量|常量和操作符的组合。

## 正则表达式

有两种方式

var pr=/abc/;

var pr = new RegExp(“abc”);

修饰符：

i：不区分大小写匹配

g：查找所有匹配，就算是找到匹配的也要全部都查找一遍才结束。

m：执行多行匹配

**元字符**：

. ：查找单个字符，除了换行和结束符。

\w ：查找单词字符

注：当字母换成大写的时候表示查找非匹配此规则。

例如：\W 查找非单词字符

\d ：查找数字

\s ：查找空白字符

\b ：匹配单词边界

\0 ：查找null字符。

\f \n\t\v\r\xxx\xdd\uxxxx不多解释，同转义字符意义。

**量词**：

+ ： 匹配一次或者更多次

\* ： 匹配0次或者更多次

？ ：匹配0或者一次。

{x，y}：匹配x或者y个，y是可选，如果没有y就是x到更多个，如果{x}那就是只匹配x个

$ ：以xxx结尾

^ ：以xxx开始

？=xxx ：匹配任何后面紧挨着xxx的字符串。

?!xxx ：匹配没有紧挨着xxx的字符串。

**方括号**：

[] ：查找括号内的任何字符。

[^] :就是加了一个“非”。查找不在方括号之间的字符。

[0-9]: 数字

（red|blue|green）：查找任何指定的选项。

RegExp的属性

global： g标志

ignoreCase: i标志

lastIndex:一个整数，标示开始下一次匹配的位置。

multiline： m标志。

ssource： 正则原文本。

RegExp：方法

compile：编译正则表达式

exec：检索字符串中指定的值，返回找到的值，并确定位置。

test：检索字符串中指定的值，如果存在true。

表达式对象的方法：

search：查找与正则表达式匹配的值

mathc：找到一个或者是多个匹配的值。

replace：替换匹配的值

split：分割匹配。

小结：今天所学习的内容是开始接触并且 朝着javascript的大门前进，通过javascript的简介变量常量表达式等等，正式叩响了javascript的大门，不说这些没用的了，去练习吧。

晚补：

google的javascript样式要求：

var： 使用@const去声明一个常量。因为const在ie中不支持。并且命名全大写。这里也就只是做了人为区分。

注意使用；作为一块语句的结束。但是function语句块要有特殊处理。

不要再流程块中嵌套function。如果需要，可以定义一个属性=function

自定义异常处理。

标准功能总是优先于非标准功能。

标准和非标准是通过ECMA区分的么？待查。

装箱拆箱：

不做做没有必要的装箱操作。

例如 new Boolean（false）这将制造一个object出来，而不是bool。

多层次结构待查没看懂

推荐使用prototype定义属性。

建议使用JSON.parse()代替eval

不要使用with

限制this的使用，1构造函数可以使用 2 对象的方法 包括闭包。

使用正常的for循环而非for in

坚决不使用array 作为map hash和associative array（关联数组）

多行文本使用“+”拼接 而不是用 \

使用array和object符号代替关键字。

例如 [] 代替new Array {} 代替 new Object

### javascript Style Rules

命名：

变量：第一个单词小写其他单词大写。

常量：全大写。

类名：每一个单词首字母都大写。

文件名：全小写。

自定义toString（） 一定要成功并且没有边界bug。

延迟初始化是一件好事。

明确范围：不要依靠window对象。

代码格式：遵循C++规则。

在需要的时候才使用（）一元运算千万不要用。

建议使用‘’声明string在html中。

# 2014-3-9

控制流：

if else：在if中要注意一下值都是对应的false

false， NaN， undefined， 0 ， “” ，null。

除此之外，其他的值都应该为true，（注意是其他的值，不是表达式）

例如： var b = new Boolean(false);

if（b）{

//此时的b是true。因为他是一个有效的的Boolean对象。

}

switch （key）

case value;

[break;]

使用规则同其他语言。

循环：

for （[初始化];[条件];[表达式]）{

//doing

}

do {

//doing

} while（条件）

while（条件）{

//doing

}

break:

中断，一般是放在循环体中。

break lable

中断当前标签。

continue

中断当前循环，进入下一次循环。

for in 语法格式如下

for (变量 in 对象){

//doing

}

注：此种遍历方式将返回数字索引对应的名称。所以建议使用传统的for

for each in 语法如下：

for each (var 变量 in Object) {

//doing

}

注：这种直接返回对象的属性，而非属性名称

异常：

throw：抛出异常。

throw 后面可以跟着任何表达式。

try … catch … [finaly]

异常处理，使用规则和语法同C#，finaly可选。

Error对象：

可以自动以异常，这样可以伪装异常也可以起到对程序的很好保护作用。

构造一个异常对象 new Error(‘异常名称’); 这样就可以在catch模块中直接使用e.name 取到异常信息。

function：一个强大的关键字。

在javascript中，function可以定义为函数 也可以理解为对象。

首先说一下他的第一个功能：函数。

函数： 格式 function [functionname] （[arguments]）{

}

如果没有函数名就是匿名函数。

javascript中的函数有很多同java或者是C#，但是我发现一个有意思的东西，javascript不存在指针和代理，但是却可以直接将function作为参数传入，例子如下：

function a(b,c){

return b(c);

}

调用 a(function(x) {return ++x} , 10) //结果 11

注：javascript中不存在这delegate，所以定义一个var也可以指向一个function。这样就会出现一个很强大的功能，随时随地可以定义function，并且也可以把function看成是一个属性（当有 var a = function () {}）。

function 作用域：function 就像围墙，他可以取围墙外面的东西，但是外面去无法调用围墙里面的东西。

闭包：闭包和作用域有着密不可分的关系，闭包就是在function里面还有function，可是我还没有搞懂闭包，这个问题我想要留到明天有网络在去查找结果了。

总结：今天学习了流控制和function，在学习function的时候发现很多关于function的原理还没有搞懂，这些东西是我在明天需要查找的。迫切希望电信快点来给我装网络。function是个强大的东西。

# 2014-3-10

继续function。在继续function之前先来插入一个小插曲。

## **对象**

取值：取值方式可以使用[“属性名称”]或者是“.”来取值。

如果使用属性名称检索不到时，默认是返回一个undefined，但是我们是有办法改变它的。例如：var name = manInfo[“name”]||(null);

这里是好玩的，要记住此用法。

如果对一个undefined对象在取其属性，将会出现TypeError的错误。

引用：

javascript中的对象引用都是地址引用，在几个对象建立关联之后，改变其中任意一个都会导致所有的都改变。

注：a = {}， b = {}，c = {}；这样a b c 是指向不同的空对象（这里比较特殊）。

如果是 a = {}； b = a； c = b；这样就是他们几个指向同一个对象。

在javascript的引用中，除掉简单类型意外的是对象引用，简单类型的引用是值引用。

原型：prototype,这个属性（起始是对象）是每一个Object和Function 都会有的，是javascript为了实现私有变量共有化（这里的共有化就是指所有他的对象都是用同一个地址存储prototype里面的东西），这样就是先一个修改，全部对象都改变的效果了。当然他的子对象也会拥有prototype里面的东西，所以这个也是继承的基础。

反射：使用typeof 去判断一个对象是否有有该属性，如果返回undefined就是没有，但是这里要注意的是，使用typeof会反射到该对象的原型中。如果不想这样，那很好，告诉我另外一个方法吧，hasOwnProperty，用法如下，

对象.hasOwnProperty(‘属性名称’) ; 存在返回true否则返回false

插曲结束，继续function。

给类型增加方法：在做这个之前应该首先说明一下Function和function的区别，网上资料介绍说Function是对象，而function是关键字，这个我表示赞同，其他我不赞同的不在此记录。我个人理解Function 是javascript内置的，如同this ，Array一样，是内置对象（我理解为构造函数，因为是可以用new 声明的。）

好的，那我们下面说一下给一个类型加方法，

问什么Function.prototype.xxx = function () {}之后String Number 等这些就可以使用.xxx了呢？

答：通过前面学习的prototype之后，我断定Number这些都是用过Function创建出来的子对象，也就是他们都继承自Function，虽然我现在还没有找到证据呢，但是我深信这一点。

回调：回调就是我们将方法作为参数传入，在function内部处理。

闭包：闭包除了可以实现类似java中的getter和setter的功能之外，它还导致整个环境（变量参数等）都被引用到下一个对象中，所以浏览器是无法释放这些资源的。所以呢，也要小心内存溢出哦。

注：在学习闭包的时候想起来Philip今天说的this关键字，这个是我下一个要考虑的知识点。

总结：今天因为网络问题没有按照网站上的学习，但是大的方向是和教学网站一致的，和学习计划也同步，今天主要针对对象和function做了学习，发现这其中有很多奥秘，作用域在闭包的作用下发生了一些变化，而prototype也完成了javascript的继承功能。

# 2014-3-11

This：指向当前调用对象，网上的说法都是这样的，我做了一点点补充，this就是当前对象，因为对于javascript来说，每一个js文件在被browser（html）引用之后，都存在这一个window对象，如一般的调动 method() ，这样的形式起始是等同于 window.method()的，所以this.method()也就是window了，所以，method中的this也就都是window了。但是如果使用了new关键字，那么就会生成一个method对象，那么method方法体中的this就是指method对象，而外部的this却还会指向window。

document：document是顶级对象中最低级的那个，也就是window，frame 等对象都加载完成之后才会生成document，document包含了所有的html元素。

window：browser对象。

还有一些类似screen ，location等在此不作记录。

在谈过this 和 window 之后 不得不说的两个方法自然就是call()和apply()了。

call()：这个方法可以将当前方法的调用对象转变为新的对象。语法形式

object.call(newObject,var1,…varn);

apply()：功能和call是一样的，语法形式需要注意一下

object.apply(newObject,[var1,…varn]);

网站都打不开了。。。。回家在继续哦

多态：在javascript中不存在多态，如果存在两个函数名称相同的function，后加载的会覆盖掉先加载的。

## CSS-Getting\_started

继承：父元素的样式会影响到子元素。

不可继承的：display、margin、border、padding、background、height、min-height、max- height、width、min-width、max-width、overflow、position、left、right、top、 bottom、z-index、float、clear、table-layout、vertical-align、page-break-after、 page-bread-before和unicode-bidi。

所有元素可继承：visibility和cursor。

内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、 font-family、font-size、font-style、font-variant、font-weight、text- decoration、text-transform、direction。

终端块状元素可继承：text-indent和text-align。

列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。

表格元素可继承：border-collapse。【引用自网络】

看到定义列表样式的时候发现这个counter（mynum）需要注意一下，我不相信有这个属性，但是在网络上暂时还没找到。这一定是自己定义的一个css参数。

总结：这个是临时的，我会把boxes，layout，tables，media看完之后再做一个总结的。

今天学习了javascript的this关键字，还有一些常用的内置对象，起始应该算是浏览器内置对象吧，window，screen等等。把这个css入门基础看了一边，觉得和之前学的大致相同，但是温故知新，还是从中领悟了一些之前不是很清楚的东西。

# 2014-3-12

html文件头部的<DOCTYPE html> 是给浏览器说明，按着哪个版本（这个是html5）来解释当前文件。

html5为了解决html4中使用div定义文章范围不准确的问题，引进了section元素。

section

accesskey，contentediteable，

contentmenu（浏览器不支持）

在html5中添加了一些新的元素

<article><section><nav><aside>

<blockquote>常用来表示引用

<article> 定义一个独立的部分。类似div。

<aside>附近相关的内容。

< nav >导航超链接

<header> 类似div，是html5提出来的模块划分的元素，有助于seo

<footer>同样的道理

<address>通常是作者 地址 之类

<time> 便于seo 一般记录出版时间和特定日期等

注：在什么场合下使用section？

答：当文章内容可以按照单元进行划分，或者是文章被列入目录的。使用section有助于理解界面布局。

html5 Input form属性

form = “formid” 这样就使得form关闭标签</from>以外的input也关联到该from上面了。

input list 属性

list属性和datalist一起使用，

<datalist>

<option value=”” />

<datalist>

input step 属性：能够整除的基数。

<output> form中的输出标签。

总结：今天主要学了html5和html4的一些对比，大致内容就是在讲解html5中新添加的一些元素对seo和程序员带来的好处，以及这些元素的使用方法和规则。

以下是补充昨晚的遗留

盒子模型：

盒子模型主要的几点就是margin padding 和border。

外边距 内边距 和边框。在这里不得不提的就是border。

border：这个东西也许会毁掉你看似很好的布局，当布局是流模式的时候，一切都是那么工整，当再放上1px的边框，一切都彻底改变了，所以要注意border可能改变盒子的大小不再等于设置的width和height。

布局：

单位：布局中说到单位有px 和em，关于em就是一个汉字的大小，在不同的像素下大小也不是绝对的，而是一个相对值。但是要注意的是css的继承可能会导致你的文字和你想象中的不一样大，比如父级的p是0.8em，他的子节点p也是0.8em，那么实际大小是0.64em。

布局中重点就是position

Position：

1. fixed 相对浏览器的相对位置。
2. absolute 绝对位置，相对这个page的，如果设置了这个属性，那么将被排除在流布局之外，也就是不具有“占位”的能力。
3. relative：相对正常的位置做一个top left right bottom的移动。
4. static：默认值。

表格：table在此不再多说明了。去练习使用它吧。

# 2014-3-13

今天主要调整了led电子时钟页面，在调整的过程中学习到的动态创建控件的办法，使用clone方法克隆对象，使用append追加内容（append是jquery的方法）。

晚上又学习了一下闭包，原型，还有模块。关于这些东西的日记也就这么多。就是练习练习练习。

继续昨天的页面：

终于想明白为啥使用getElementById无法或得到ul元素了。是因为我把方法写在了window.onload事件上，此时的元素还没有加载。

注：虽然网上有博文说onload是在页面元素加载完成之后执行，但是通过实际测试，证明此说法不成立。

总结：对于javascript的学习中，没能熟练应用的就是this+闭包。这两个东西组合的练习做的少了点。

# 2014-3-16

工厂方法没有new 不是问题。但是在每次调用方法都new一个新的内存空间是一个问题。

如以下问题，如果没有new那就无法实现面向对象的优化工作。所以没有new还是有点问题。

问题：如果我返回一个匿名对象那么他们是指向同一个栈地址么？

答：不同的地址

问题：Function和Object是什么关系？

答：Function继承自Object

# 2014-3-17

instanceof：variable instanceof constructor； retrun true or false；

with：javascript中的with如同C#中的using（在方法中使用，非文件头）。

## 垃圾收集

标记清除：标记清除的方式和.NET的内存管理机制有几分相似，浏览器有一个定时回收机制，javascript主要负责标记失效的或者是已经无用的资源信息，等待垃圾回收期进行回收。也可以在js中强制做内存回收，IE：window.CollectGarbage()；Opera7+：window.opera.collect()；

计数引用：在IE的DOM和BOM中的引用都为计数引用，所以如果使用了DOM对象的引用在使用完成之后都要将对象赋值null（解除引用）

## Array

join：join(‘分隔符’)；

push：在Array最后追加元素，也可以使用数组方式直接赋值。

pop：取得最后一个元素，并且移除此元素。

shift：取得第一个元素 并且移除它。

reverse：降序

sort：升序

注：这两个排序是按照字符串类型排序的，而非数字类型，比如：11 会排在 2的前面。

所以，sort加强了自己的能力，它可以接受一个比较函数作为参数。

例如：function compare(v1,v2) {

if(v1 > v2) {

return 1;

} else if (v1 < v2) {

return -1;

} else {

return 0;

}

}

sort(compare);就可以对数值类型的Array进行比较了。

concat：复制当前Array并且将参数传递到副本中作为新的元素。

注：如果参数中有[],将把[]的每一个元素循环插入到副本中

slice：如同C#中substring，是一个截取功能。

注：此方法一样是制作一个副本出来，修改副本内容不影响原Array

splice：第一个参数，起始位置，第二个参数，要删除的元素个数，第三个参数到第N个参数是要从起始位置插入的元素。

注：此方法直接对原Array产生影响。

//以下方法传入参数的function都有三个参数（item，index，array）

every：参数是一个给定函数，每一项都返回true则是true

some：同上，此方法有一项是true就返回true

filter：过滤，返回满足过滤的结果组成的数组

forEach：遍历操作，无返回值。

map:遍历对元素操作，返回操作后的元素组成的数组

原型继承：原型继承就是通过把父类对象作为参数传入给子类的prototype。

代码：function Sub(o) {

function F() {};

F.prototype = o;

return new F();

}

寄生式继承：就是在原型继承外面再包一层壳。

function createSub(o) {

var sub = Sub(o);

sub.xxx = function() {

//具体实现

};

return sub;

}

# 2014-3-18

## 浏览器CSS定制

-moz- : Firefox  
-webkit- : Safari & Chrome  
-o- : Opera  
-khtml- : Konqueror  
-ms- : Internet Explorer  
-chrome- : Google Chrome专用前缀

## DOM

## document.domain：

这个值的设置必须来自同一个域名。

比如：[www.csdn.com](http://www.csdn.com) 和down.csdn.com 此时如果通过嵌套在一个页面中，是不能互相访问js文件的，可以通过设置document.domain=”csdn.com”；这样就可以相互访问了。

### 特殊集合

document.anchors:文档中所有带有name属性的 a 元素

document.applets:文档中所有applets元素。

document.forms：所有表单

document.images：所有图片

document.links：所有带有href特性的 a 元素。

### **特性**：

getAttribute(‘’): 获取特性值

setAttribute(‘’,’’):设置特性值

removeAttribute(‘’):移除特性

### 创建元素

createElement(‘’);在IE8以及更早的版本中可以使用createElement(‘<div class=”” name=””></div>’);

### 动态添加脚本：

var sc = document.createElement(‘script’);

sc.type = ‘text/javascript’;

sc.src=’’;

document.body.appentChild(sc);

如果需要在脚本里面添加脚本代码可以使用如下方法：

var code =””;

try {

sc.appendChild(document.createTextNode(code));

} catch {

sc.text = code;

}

### 动态创建link

与动态创建脚本基本一致，需要注意的是，动态创建link需要Head.appendChild(‘’);

是要动态追加到head中。

### 表格操作

caption：caption元素的指针

tBodies,tFoot,tHead,rows,cell此属性均可直接指向表格对应的元素。

createTHead(),createTFoot(),createCaptopn()等方法是创建对应元素，还有删除对应元素的方法是将create变成delete。

insertRow(index)直接创建并且插入行。

insertCell(index)直接创建并且插入单元格。

### 判断元素是否获取焦点（HTML5）

1. document.activeElement === button
2. button.hasFocus();

### 当前文档是否加载完毕：readyState

readyState：laoding和complete正在加载和加载完成，可以结合window.onload使用

### document.compatMode

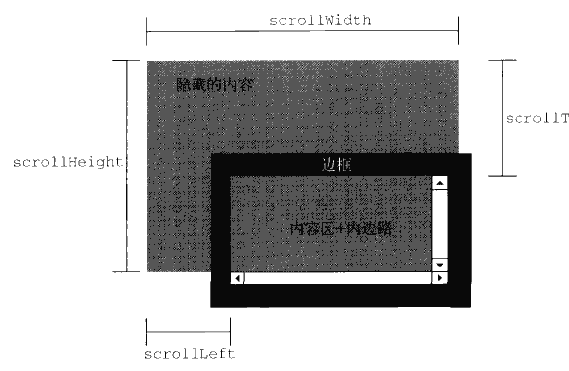
CSSlCompat 标准模式 BackCompat：混合模式

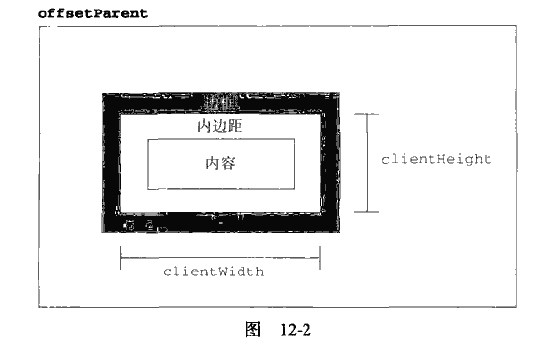
### insertAdjacent：

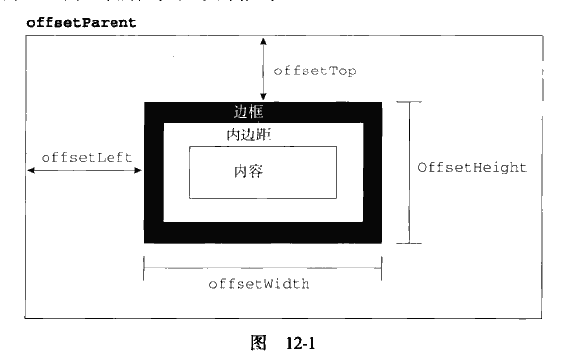
该方法需要两个参数，第一个“枚举”参数，beforebegin afterbegin beforeend afterend，第二个参数是一个string，可以包含元素，例如’<a href=””/>’

#### scrollIntoView(true or false)

关于页面和滚动条的宽高最有效说明的两张图：







### 事件冒泡处理：

在DOM中，事件的处理是采用的冒泡形式。如果想要阻止冒泡可以采用以下代码

if(e && e.stopPropagation) {

e.stopPropagation()

} else {

window.event.cancelBubble=ture;

}

### DOM2级事件处理程序

addEventListener(‘事件’,’实现的对象’,true or false);

removeEventListener(参数同上);

注：强烈建议在add的时候使用实例对象传入，而不要使用匿名方法，因为remove的参数对象必须是和add的参数是同一个对象可能remove。最后一个true or false 表明是否在冒泡阶段触发。

### IE事件处理程序

attachEvent(‘onXXX’,’实现的对象’);

detachEvent(参数同上);

注：这两个方法无论是被任何对象调用，方法内部的作用域都是window，也就是在“实现对象”的方法中如果出现this都是指向window。（嵌套this除外）

### 事件

mousewheel：鼠标滑轮滚动

event上有一个inputMethod属性对应的值和解释如下

0:不确定输入方式

1：键盘输入

2：粘贴

3：拖放

4：IME输入

5：通过在表单中选择项输入

6：通过手写输入

7：语音输入

8：混合输入

9：通过脚本输入

contextmenu：菜单事件。

beforeunload：在离开该页面之前触发

DOMContentLoaded：在形成完整的dom树之后，在load之前。

haschange：属于window，当发生此事件之后 event存在着oldURL和newURL

委托：在javascript中所谓的委托就是要学会利用冒泡原理

内存清理：在移除具有事件的元素之前，请先将他的事件链断开，这样有利于回收机制进行回收。

模拟事件：createEvent()方法。

**模拟鼠标**：createEvent(‘MouseEvents’);此方法返回initMouseEvent()方法。这个方法接收15个参数：

type：要触发的事件类型（click mousemove….）

bubbles:bool类型，是否冒泡。

cancelable：bool ，是否可以取消。

view(AbstractView):与事件关联的视图，几乎总是document.defaultView

detail：int类型，几乎总是0，

screenX：int类型，屏幕x坐标

screenY:不解释

clientX:clientY：统统不解释

ctrlKey：bool是否按下此键，默认是false。与此类似的还有shiftKey，altKey,metaKey.

button:Int类型，按下哪一个鼠标键，默认是0.

relateTarget：Object类型，表示与此事件相关的对象。这个参数是在模拟mouseover和mouseout时候使用。

模拟键盘：传入”KeyEvents”字符串。返回initKeyEvent()方法。有是个参数

type：不解释

bubbles：不解释

cancelable：不解释

view：不解释

ctrlKey，altKey，shiftKey，metaKey：不解释

keyCode：键码

charCode：默认0，按键字母对应的ASCII编码。

## 表单

防止重复提交表单的办法：

1. 按提交之后将提交按钮禁用，
2. 可以利用onsubmit事件中的方法取消后续提交的表单。

表单字段共有属性和方法：

disabled：是否禁用。

form：指向当前字段所属表单指针：只读

name, readOnly, tabIndex, type value.

#### **选择文本**：

selectionStart和selectionEnd属性：这两个属性保存的初始值是0

setSelectionRang（index1，index2）：选择开始和结束的索引值。

#### 剪切板事件

beforecopy：copy之前

copy：copy时发生

beforecut：剪切前发生

cut：剪切发生

beforepaste：粘贴钱发生

paste：粘贴发生

clipboardData：剪切板对象，在IE中是window下面的一个属性，在其他浏览器中是event下面的一个属性。但是在chrome中之后再授权之后才可以使用此对象。

此对象有三个方法：

1. getData（）
2. setData（）
3. clearData（）

# 2014-3-19

HTML5约束验证：（这些验证只有在最新的浏览器上才有可能有效果，使用时要注意）

required：必填字段，

可以使用 var boolResult = “required” in document.createElement(‘input’);

type:在html5中新添加了一些type值，在使用的过程中，如果有特殊需要可以查看api。

stepUp(number),stepDown(number)：给对象的值增加或者减少一定数值。

pattern：正则表达式验证

注意：此正则表达的内容不需要使用^和&作为开始和结束标志，默认内容认为是存在这两个规则的，如果写了这两个符号就表示必须全部匹配。

checkValidity:表单拥有的方法，会验证表单中的每一个元素的required和pattern属性，如果都通过就返回true，否则返回false。

validity：属性对应的值和意义

customError：如果设置了setCustomValidity返回true，否则false

patternMismatch：如果与制定的pattern不匹配返回true

rangeOverflow：如果值比max的值大，返回true。

rangeUnderflow：如果值比min值小，返回true。

stepMisMachch：如果步长不合理返回true。

tooLong：如果长度超出了maxLength返回true。

tyoeMismatch：如果值不是要求格式，返回true。

valid：其他属性都是false返回true。

valueMissing：如果required标记的元素没有值，返回true。

noValidate：表单禁用验证，设置为表单的属性为noValidate即可。

### 选择框

提供以下属性和方法，

add(newOption,relOption):向空间插入新的option元素。relOption表示与位置相关项

multiple：bool类型，是否可以多选。

options：所有节点

remove（index）：移除指定的option

selectedIndex：如果没有选择就是-1.

size：选择框一次可见行数。

### 富文本编辑

使用iframe 作为富文本编辑器，一定要在document全部加在完成之后设置它的designMode为on才可以编辑，默认是false

**更好的办法**

可以对元素使用contenteditable属性，该属性有三个可选值 true false 和inherit。

针对富文本也有对应的命令

### document.execCommand(p1,p2,p3)

p1：命令

p2：始终false，为了兼容火狐。

p3：可以是null或者对应的值。

### **命令可执行检测方法**：

queryCommandEnabled（‘命令’）如果命令可执行返回true，否则返回false。

### **跨文档消息传递**：

XDM的核心就是postMessage（）方法，此方法有两个参数，第一个参数是消息内容，第二个参数表示iframe的rul来源，如果匹配就发送，如果不匹配就不发送。

注：第二个参数 “\*”，表示无规则发送给所有框架。

使用processMessage（event.data）接收数据

### 拖放

dragstart：开始拖动

drag：拖动中

draggend：已放开

被动控件事件：

dragenter：拖进来

dragover：在范围内移动

dragleave|drop：离开

**拖放数据重点**

dataTransfer：这个对象有两个常用的主要方法，getData(‘key’) 和setData(‘key’,’text’);

key:html支持mime类型，例如text/plain等 还有url

**两个重要的属性**

dropEffect：这个属性有四个可能值：none：不能把拖动的元素放在这里。

move：把拖动的元素放置在移动的目标。

copy：把拖动元素复制放置在目标。

link：放置目标会打开拖动元素。

### 媒体

检测浏览器支持媒体类型：canPlayType（audio|video/编码样式），此方法返回probable，maybe和“”。

### 匿名闭包与自引用

javascript中存在这样一种写法

匿名闭包

(function(a) {

//方法执行

}());

这种就是匿名闭包+自执行。

var t = (function(a) {

//方法执行

}(t));

匿名闭包+自执行+自引用。

这样做在function声明是就在内存中产生一个副本。

# 2014-3-20

## XHR

以下这段代码就是创建一个XHR对象，兼容了IE7以前的版本。

Ajax.prototype.createXHR = function() {

if(typeof XMLHttpRequest !== 'undefined') {

return new XMLHttpRequest();

} else if(typeof ActiveXObject !== 'undefined') {

var versions = ['MSXML2.XMLHttp.6.0','MSXML2.XMLHttp.3.0','MSXML2.XMLHttp'];

for(i = 0; i < versions.length; i++) {

try{

return new ActiveXObject(versions[i]);

break;

} catch(ex) {

continue;

}

}

} else {

throw new Error('No XHR Object probable！');

}

};

### XHR的方法

open：主要有三个参数：（post or get），url，bool

前两个参数一目了然，第三个参数的意思为是否异步请求。

注：open方法并不是真正的请求，只是做好一个发送的备选。

send：有一个参数：作为请求主体要发送的数据。如果不需要发送数据就使用null。

例如：send（null）；

### XHR的属性

status：在http下面，status的值为200表示成功（一般都是在200到300之间，不包括300），304表示请求数据未发生改变。此时可以使用**responseText**属性获取到文本，如果是xml类型的可以使用**responseXML**获取。

readyState：0：为构建open，1：表示未send，2：已经send但是未收到数据，3：收到部分数据，4：已经接收完毕，可以使用。

onreadystatechange：在readyState每次改变的时候都会触发onreadystatechange事件。但是这个事件的使用一定要保证在open之前。

### HTTP

setRequestHeader（key，value）

key值列表：Accept浏览器能够处理的内容

Accept-Charset浏览器能够显示的字符集

Accept-Encoding浏览器能够处理的压缩编码

Accept-Language浏览器当前设置的语言。

Connection浏览器与服务器之间的连接类型

Cookie当前页面设置的任何Cookie

Host发送请求页面的所在域

Referer发送请求页面的URI

user-Agent浏览器用户代理字符串。

注：此方法的调用一定是在open和send之间

### GET

在数据请求时，如果有参数，那么要注意编码类型一定要使用encodeURIComponent()进行编码。

### POST

这里需要注意的Content-Type：application/x-www-form-urlencode 当设置RequeHeader为此类型时，就是表单提交。

### 进度事件

loadstart:在接收到第一个字节时候就发生

progress：在接收期间持续发生。

error：请求发生错误的时候发生。

abort:在因为调用abort（）方法终止发生

load：在接收到完整的相应数据时发生

loadend：在通信完成或者是error abort 和load事件完成后发生。

onprogress：在事件触发时候会接收到一个event对象，其target实行是XHR对象。但是这个event对象还包含三个属性，lengthComputable，position和totalSize。

lengthCompuable:bool类型，表示进度信息是否可以用bool值。

position：表示已接收的字节。

totalSize：需要接收的总字节。

### 跨域访问

XDomainRequest:在IE下使用此对象进行跨域访问，他的属性和方法与XHR类似。

在火狐浏览器上跨域是不需要做任何改变的，但是在chrome上，虽然理论上也支持，但是在实际使用过程中却不能跨域访问，需要对服务器的Request做配置。

# 2014-3-21

## **跨域访问**：

跨域访问有很多种，关于xhr的跨域访问在html5中提供了跨域资源共享的功能，只要在服务端的程序配置RequestAddHeader(“Access-Control-Allow-Origin”,”授权域名+端口”);

注：这个是一定要在服务端配置的，在客户端添加此访问头信息无效。

## **FF和其他浏览器的event**：

在FF浏览器中，如果需要使用event，一定要在事件调用的时候把event作为参数传入，否则无法使用。

## 阻止FF的拖拽重定向脚本

//阻止FF拖拽之后重定向新的url（e：event）

if(e.stopPropagation) {

e.stopPropagation();

}

# 2014-3-22

由于默认的字体大和小不一样，所以在有文字的地方要注意使用overflow： hidden

这样可以防止自动换行的问题。

### CSS鼠标样式

|  |  |
| --- | --- |
| 默认 | default |
| 文字/编辑 | text |
| 自动 | auto |
| 手形 | pointer, hand(hand是IE专有) |
| 可移动对象 | move |
| 不允许 | not-allowed |
| 无法释放 | no-drop |
| 等待/沙漏 | wait |
| 帮助 | help |
| 十字准星 | crosshair |
| 向上改变大小(North) | n-resize |
| 向下改变大小(South) | s\_resize 与n-resize效果相同 |
| 向左改变大小(West) | w-resize |
| 向右改变大小(East) | e-resize 与w-resize效果相同 |
| 向左上改变大小（NorthWest） | nw-resize |
| 向左下改变大小（SouthWest） | sw-resize |
| 向右上改变大小（NorthEast） | ne-resize 与sw-resize效果相同 |
| 向右下改变大小（SouthEast） | se-resize 与nw-resize效果相同 |

# 2014-3-23

## 常用的css命名规则：

头：header

内容：content/container

尾：footer

导航：nav

侧栏：sidebar

栏目：column

页面外围控制整体布局宽度：wrapper

左右中：left right center

登录条：loginbar

标志：logo

广告：banner

页面主体：main

热点：hot

新闻：news

下载：download

子导航：subnav

菜单：menu

子菜单：submenu

搜索：search

友情链接：friendlink

页脚：footer

版权：copyright

滚动：scroll

内容：content

标签页：tab

文章列表：list

提示信息：msg

小技巧：tips

栏目标题：title

加入：joinus

指南：guild

服务：service

注册：regsiter

状态：status

投票：vote

合作伙伴：partner

## 注释的写法：

/\* Footer \*/

内容区

/\* End Footer \*/

## id的命名：

(1)页面结构

容器: container

页头：header

内容：content/container

页面主体：main

页尾：footer

导航：nav

侧栏：sidebar

栏目：column

页面外围控制整体布局宽度：wrapper

左右中：left right center

(2)导航

导航：nav

主导航：mainbav

子导航：subnav

顶导航：topnav

边导航：sidebar

左导航：leftsidebar

右导航：rightsidebar

菜单：menu

子菜单：submenu

标题: title

摘要: summary

(3)功能

标志：logo

广告：banner

登陆：login

登录条：loginbar

注册：regsiter

搜索：search

功能区：shop

标题：title

加入：joinus

状态：status

按钮：btn

滚动：scroll

标签页：tab

文章列表：list

提示信息：msg

当前的: current

小技巧：tips

图标: icon

注释：note

指南：guild

服务：service

热点：hot

新闻：news

下载：download

投票：vote

合作伙伴：partner

友情链接：link

版权：copyright

## class的命名：

(1)颜色:使用颜色的名称或者16进制代码,如

.red { color: red; }

.f60 { color: #f60; }

.ff8600 { color: #ff8600; }

（2）字体大小,直接使用”font+字体大小”作为名称,如

.font12px { font-size: 12px; }

.font9pt {font-size: 9pt; }

(3)对齐样式,使用对齐目标的英文名称,如

.left { float:left; }

.bottom { float:bottom; }

(4)标题栏样式,使用”类别+功能”的方式命名,如

.barnews { }

.barproduct { }

注意事项::

一律小写;

尽量用英文;

不加中杠和下划线;

尽量不缩写，除非一看就明白的单词.

主要的 master.css

模块 module.css

基本共用 base.css

布局，版面 layout.css

主题 themes.css

专栏 columns.css

文字 font.css

表单 forms.css

补丁 mend.css

打印 print.css

# 2014-3-24

## JSON-js

### stringify()

此方法有三个参数：1、即将转化的对象

2、过滤器

3、缩进字符或者是缩进长度。

#### 过滤器:

样例一，‘--’缩进：

console.info(JSON.stringify(jsonObj,null,'--'));

样例二，[] 过滤：

console.info(JSON.stringify(jsonObj,['title','year'],2));

样例三，function 过滤：

alert(JSON.stringify(jsonObj,function(key,value) {

switch(key) {

case 'Author':

return value.join('#');

break;

case 'year':

var date = new Date(value);

return date.toLocaleDateString();

default:

return value;

}

},2));

#### 缩进字符：

可以使用数字表示缩进长度，也可以使用‘--’、‘@@’等等作为缩进符号

#### toJSON：

在javascript对象字面量中加入toJSON属性，并且采用闭包方式返回一个规则，这个规则将作为序列化字面量的第一规则。

序列化字面的顺序：

1. 如果存在toJSON，则首先按照toJSON规则执行，返回整体字面量对象。
2. 如果stringify传入了第二个参数，在进行第二遍过滤。
3. 序列化
4. 进行格式化（是否缩进等）

### JSON.parse

prase接收两个参数，第一个是字符串，第二个是解析规则（这个是可选，功能与过滤器相同）

console.info(JSON.parse(jsonText,function(key,value) {

if(key === 'year') {

var date = new Date(value);

return date.getFullYear();

} else {

return value;

}

}));

## E4X

Tips：FF不支持在定义XML对象时传入标准的xml文件头信息，<?xml version = xxx?>.

<script type="text/javascript;**e4x=1**"> </script>要注意e4x=1

虽然有这么多声明，但是e4x在FF18已经被移除了。所以可以使用DOMParser/DOMSerializer

### XML对象

//以下是旧版本，也是书上讲解版本，在新版的浏览器中都放弃了e4x.使用DOMParser代替

Object.prototype.createXMLDocument = function() {

if(typeof ActiveXObject !== 'undefined') {

return new ActiveXObject("Microsoft.XMLDOM");

} else {

return document.implementation.createDocument("","",null);

}

};

#### DOMParser

parseFromString(‘目标文本’,’mime类型’);此方法返回一个类型的对象。

例如

var x = new DOMParser();

var xmldoc = x.parseFromString('<aaa>Lampard</aaa>','text/xml');

alert(xmldoc.childNodes[0].childNodes[0].nodeName);

以下是针对XML操作的API

documentElement 属性：XML 文档的根节点。  
nodeName 属性：节点的名称(只读)。  
nodeValue 属性：节点的值。  
nodeType 属性：节点的类型。  
childNodes 属性：返回子节点集合。  
parentNode 属性：返回父节点。  
firstChild 属性：返回第一个子节点。  
lastChild 属性：返回最后一个子节点。  
nextSibling 属性：返回下一个兄弟（同级）节点。  
previousSibling 属性：返回前一个兄弟（同级）节点。

注：文本节点的 nodeName 永远是 #text

#### 获取节点的方法

getElementById和getElementsByTagName和for遍历

# 2014-3-25

## window.onerror：

三个参数。message、url、linenum。这三个参数中有用的是第一个，第二个和第三个是鸡肋。

## 判断网页致命错误的依据

1. 应用程序无法进行。
2. 影响了用户的主要使用功能。
3. 导致连带错误。

## javascript高级技巧

Object.prototype.toString.call(XXX) 得到 [Object XXX类型]；

bind()：原生javascript方法，接收方法和调用对象的绑定

函数柯里化：

curry：就是利用arguments属性和bind方法组合。

### 防篡改对象

Object.preventExtensions(XXX):XXX代表具体对象，调用这个方法之后无论给xxx对象在加任何属性都是无效的。（检测方法 Object.isExtensible()）

Object.seal(XXX):xxx代表具体对象。调用这个方法之后的对象被密封，不可添加属性也不可删除已有属性。（检测方法 Object.isSeal()）

Object.frozen(XXX):既是seal也是preventExtensions

### constructor属性

constructor 属性返回对创建此对象的数组函数的引用

样例

var test = new Boolean();

alert(test.constructor);//这个结果就是Boolean

### 自定义事件（C#中的delegate）

function EventTarget() {

this.handlers = {};

}

EventTarget.prototype.addEventHandler = function(type,handler) {

if(typeof this.handlers[type] === 'undefined') {

this.handlers[type] = new Array();

}

this.handlers[type].push(handler);

};

EventTarget.prototype.removeEventHandler = function(type,handler) {

if(this.handlers[type] instanceof Array) {

var handlerArray = this.handlers[type];

for(i = 0; i < handlerArray.length; i++) {

if(handlerArray[i] == handler) {

handlerArray.splice(i,1);

break;

}

}

}

};

EventTarget.prototype.trigger = function(event) {

if(!event.target) {

event.target = this;

}

if(this.handlers[event.type] instanceof Array) {

var handlerArray = this.handlers[event.type];

for(i = 0; i < handlerArray.length; i++) {

handlerArray[i](event);

}

}

};

### 解耦

1. 解耦HTML和javascript。HTML是用来显示数据，javascript是用来定义行为，如果在页面中标签中存在过多的onclick=“”等等事件指向的时候，应该考虑把这些剥离出来，使用DOM进行动态绑定。
2. 解耦javascript和HTML，如果在js中存在着大量的HTML输出，那么就应该考虑解耦，可以采用先display：none的方式加载，在需要的时候在去渲染他们。也可以使用ajax进行一步请求php ruby等得到输出html代码进行输出。
3. 解耦javascript和css，尽量不要再js中去动态添加css样式（除非很少的属性），可以先定义好一个class，在使用js动态给element.className=“xxx”；

### 性能

避免在一个function或者是一个js文件中出现大量的document。

解决办法：var doc = document；将document进行copy，放在内存中循环使用。

尽量避免使用with，因为with相当于（C#中的using——在method中，而非文件头的引用），会创建自己的作用域。但是不同的而是在js中没有自动释放的能力。

**最小化语句数**

当声明多个变量是可以这样 var a, b, d =0;或者 var a=0,b=’’,c=true;

## 部署

Ant构建工具

<http://ant.apache.org>

检测工具

<http://www.jslint.com>

压缩工具：

JSMin

<http://www.crockford.com/javascript/jsmin.html>

Dojo

<http://shrinksafe.dojotoolkit.org>

YUI Compressor

<http://yuilibrary.com/projects/yuicompressor>

单元测试

JsUnit

<http://www.jsunit.net>

# 2014-3-26

## javascript’s offline

### manifest文件语法

1. 每一个指令都另起一行。
2. 在文件头用“#”进行注释。
3. CACHE MANIFEST 在词标题下面列出需要缓存的文件
4. NETWORK 在此标题下面列出需要网络连接的文件
5. FALLBACK 在此标题下面列出当离线访问出错时替换文件。

注：1、“#”的注释可以作为文件更新的重要操作，当需要更新缓存文件时，可以更改“#”注释内容（日期版本号）。

2、manifest文件的后缀名推荐使用“.appcache”, manifest 文件需要配置**正确的 MIME-type**，即 "text/cache-manifest"。必须在 web 服务器上进行配置.

3、浏览器对缓存文件的限制大小不一，一般不建议超过5M（因为有些浏览器控制了最大不能超过5M）

#### 样例：

# manifest 2014-3-26 v=0.1

CACHE:

/main.css

/main.js

/logo.jpg

NETWORK:

/index.aspx

FALLBACK:

/image/ /404.html

至此整个文件都写完了，以下是关于这个文件的一点说明。

1. 如果需要使用“所有文件”，那么请用“\*”。
2. FALLBACK下面的语法 第一个是“请求资源路径”，第二个是“替补路径”。

**注：一直都在说manifest文件的配置，却没有说怎么使用这个文件。使用这个文件的办法就是要在html中指定。例如：<html manifest = “xxx.appcache”>就是这样指定了这个页面将按照xxx.appcache清单缓存离线文件。**

### applicationCache对象

#### status属性

这个属性的值是从0~5.

0：无缓存，没有与页面相关的缓存。

1:闲置，应用缓存没有得到更新。

2:检查中，正在下载清单文件，并检查更新

3:下载中，正在下载清单中所列资源

4:更新完成，所有资源已经下载完成，可以通过使用swapCache()来使用

5:废弃，缓存清单已经无效或者不存在了，无法访问应用缓存。

#### 缓存状态改变时对应的事件

checking：在为应用缓存查找更新时触发

error：在检查更新或者下载更新出错时触发

noupdate：检测过清单之后，发现没有需要更新的内容时触发。

downloading：在下载缓存文件时触发。

progress：在下载应用缓存的时候持续触发。

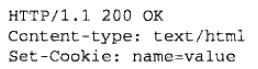
updateready：下载完毕，且可以通过swapCache()使用时触发。

cached：在应用缓存完整可用时触发。

### 数据存储

#### cookie

在访问的时候可以设置requestHeader控制set-Cookie信息，



Set-Cookie的值类型是name value的样式。

注：不同的浏览器对同一个域名下的cookie有限制，30~50不等。并且每一个cookie的大小是有限制的4096B（+-1）长度，因为浏览器不尽相同，保守起见，设置4095B是最合理的，如果cookie的长度超过来指定长度，浏览器将丢掉该cookie。

##### cookie的组成：

名称：不区分大小写

值：cookie中的字符串 经过url编码。

域：cookie对那个域是有效的。

路径：对于指定域中的那个路径应该向服务器发送cookie。

失效时间：到指定时间cookie就删除

安全标志（secure）：制定后，只有在ssl连接时才发送cookie

cookie的创建方法见练习代码。

注：一般为了能够让网站下所有页面都能访问cookie会将path设置成 “/”，也就是根目录。

也可以通过设置domain=xxx.xxx设置在同一个域名之下的所有网站都能访问，非同域名仍然不能访问。

#### IE数据存储

behavior：css属性。

behavior：url（url|#userID|#default#behaviorName）

url(url):使用绝对或相对url地址指定DHTML行为组件（.htc）

url(#objID):使用二进制实现(作为ActiveX控件)的行为。此处的#objID为object对象的id属性值

url(#default#behaviorName):IE的默认行为。由行为的名称标识(#behaviorName)指定

**可以使用url(#default#userData)方式进行数据存储，但是要注意调用save()方法。**

### web存储机制

#### Storage对象

clear（）：清空数据 FF为实现

getItem(name)通过名字获取Item

key（index） 通过索引获取key值

removeItem（name） 通过名字移除值

setItem（name，value）设置Item的name和value

#### sessionStorage

随着回话的建立存储数据，当回话结束之后所有数据即将删除。

如果有大量的数据存储的时候，可以使用begin（）开始和commit（）结束。

#### globalStorage

对于globalStorage这个对象则不能像其他对象那样简单的使用sessionStorage.xxx，

globalStorage[域].xxx=xxxx;

例如

globalStorage[baidu.com].name = ‘XXXX’;只有baidu.com这个域下的地址可以访问此数据。

#### localStroage

localStroage是在HTML5中推出替代globalStorage的，在localStroage只能在同域名同端口访问。使用方法和sessionStorage一样

### IndexedDB

这个对象在不同的浏览器下面有不同的名字，所以采取一种兼顾的办法就是使用“||”

例如：

var IndexedDB = window.IndexedDB||window.msIndexedDB||window.mozIndexedDB||window.webkitIndexedDB;

# 2014-3-27

## IndexedDB

var IndexedDB = window.IndexedDB||window.msIndexedDB||window.mozIndexedDB||window.webkitIndexedDB;这是一个DB对象

### IndexedDB的事件

onabort：当一个事务（transcaction）被取消的时候触发

onblocked：由于其他活动中的事务导致改变版本信息事务失败时触发

oncomplete：当事务提交（commite）所有未完成事务成功之后触发

onerror：无论做任何数据操作，只要发生错误都会触发

onsuccess：当一个请求成功之后触发

onupgrandeneeded：当打开的数据库版本号发生改变时触发。在创建数据库的时候也会触发。

### indexed Database对象

以下摘自网络

[*IDBFactory*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBFactory) *提供了对数据库的访问。这是由全局对象 indexedDB 实现的接口，因而也是该 API 的入口。*

[*IDBCursor*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBCursor) *遍历对象存储空间和索引。*

[*IDBCursorWithValue*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBCursorWithValue) *遍历对象存储空间和索引并返回游标的当前值。*

[*IDBDatabase*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBDatabase) *表示到数据库的连接。只能通过这个连接来拿到一个数据库事务。*

[*IDBEnvironment*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBEnvironment) *提供了到客户端数据库的访问。它由* [*window*](https://developer.mozilla.org/en-US/docs/DOM/window) *对象实现。*

[*IDBIndex*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBIndex) *提供了到索引元数据的访问。*

[*IDBKeyRange*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBKeyRange) *定义键的范围。*

[*IDBObjectStore*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBObjectStore) *表示一个对象存储空间。*

[*IDBOpenDBRequest*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBOpenDBRequest) *表示一个打开数据库的请求。*

[*IDBRequest*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBRequest) *提供了到数据库异步请求结果和数据库的访问。这也是在你调用一个异步方法时所得到的。*

[*IDBTransaction*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBTransaction) *表示一个事务。你在数据库上创建一个事务，指定它的范围（例如你希望访问哪一个对象存储空间），并确定你希望的访问类型（只读或写入）。*

[*IDBVersionChangeEvent*](https://developer.mozilla.org/en-US/docs/IndexedDB/IDBVersionChangeEvent) *表明数据库的版本号已经改变。*

### IDBDatabase

coreateIndex(‘indexname’,’keyname’[,{unique:true|false}];

参数：索引名称，索引对应的key，是否可重复（可选）；返回一个

createObjectStore（tablename，option）

此方法就是创建一张表，option中指定{keypath：xxx[，autoIncrement：true|false]}

keypath：就是主键，可以指定是否为自动增长类型。

注：此方法返回一个objectStore对象，与此同时也会创建一个事务，但是没有搞清楚这个事务是什么时候提交和结束的。

transaction（[‘tb1’,’tb2’],’Enum’）

可以同时创建多个objectStore的事务。

Enum包括：

readonly：只读

readwrite：读写

versionchange：提供读取和写入访问权来修改对象存储定义，或者创建一个新的对象存储。

objectStore：

get（）：此方法只能通过keypath进行查询

delete（）：此方法只能通过keypath进行删除

clear（）：清楚所有record

put（）：插入或者是更新

add（option）：插入，如果之前存在keypath相同的则发生error

# 2014-3-28

继续昨天的知识点。

键范围：虽然这个东西不能单独使用，但是再学游标之前有必要学习一下，这个东西就是一个查询条件——keypath范围条件

var IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange;

因为chrome的实现和其他浏览器不太一样，so，这里做了一个兼容。

键范围有四中形式：

1. only

var onlyKeyRange = IDBKeyRange.only(‘keyValue’);唯一查询。

1. lowerBound：定义一个最小值，然后就向后遍历做最后。

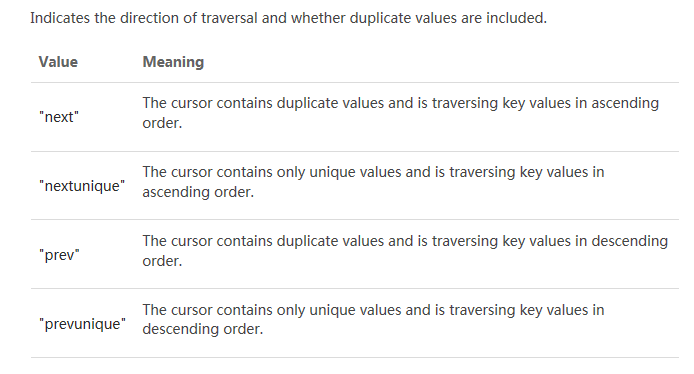
var lowerBound = IDBKeyRange.lowerBound(‘keyMinValue’[,true|false]);

第二个参数为可选，如果是true则包含keyMinValue。否则不包含。

1. upperBound：与lowerBound用法一样，但是这个是指定最大值。也就是从头开始遍历到指定的值结束。
2. bound：这个方法是需要同时指定上下边界和是否包含。

var bound = IDBKeyRange.bound(‘startValue’,’endValue’[,true|false,true|false]);

游标方向：



注：可以直接使用textValue，这样就可以避免声明IDBCursor变量了。

游标：openCursor（[IDBKeyRange,orientation]）

指定范围和方向是可选的，默认就是全部范围和next。

返回一个游标对象，

游标对象有属性：

key：就是keypath的值。

value：就是object啦。也就是存储进去的那个。

primaryKey：游标使用键。

方法：

continue([key])：移动到下一个或者是指定的key，如果不调用这个方法一个游标就只会执行一次，不会遍历所有结果出来。

advance（count）：向前移动count行。

注意这里是count 不是key

### 并发问题

注：网上和书上有很多文章说使用setVersion（）方法测试，但是请注意，此方法已经删除，并且在所有浏览器上都没有得到支持。在chrome中要是用-webkit注明（但是实际测试最新版本的也不支持）<https://developer.mozilla.org/en-US/docs/Web/API/IDBVersionChangeRequest>

# 2014-3-31

## AngularJS

<!DOCTYPE html>

<html ng-app>

<head>

<meta charset="utf-8" />

<script type="text/javascript" src="angular.js"></script>

<title>angular</title>

</head>

<body>

Hello {{'world!'}}

</body>

</html>

对应的解释如下图



### 双向绑定

<p>双向绑定</p>

Your name<input ng-model="yourname" placeholder="GGG"/><br>

Hello {{yourname||'world!'}}

placeholder：占位符的效果就是如下图



### AngularJS应用的解析

本节描述AngularJS应用程序的三个组成部分，并解释它们如何映射到模型-视图-控制器设计模式：

#### **模板**（Templates）

模板是您用HTML和CSS编写的文件，展现应用的视图。󰀃您可给HTML添加新的元素、属性标记，作为AngularJS编译器的指令。󰀃AngularJS编译器是完全可扩展的，这意味着通过AngularJS您可以在HTML中构建您自己的HTML标记！

#### 应用程序逻辑（Logic）和行为（Behavior）

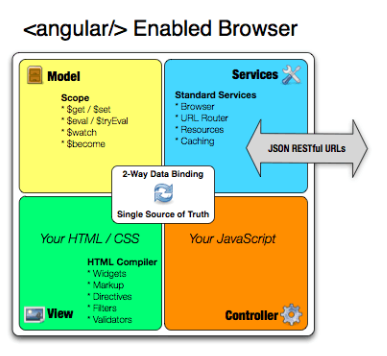
应用程序逻辑和行为是您用JavaScript定义的控制器。AngularJS与标准AJAX应用程序不同，您不需要另外编写侦听器或DOM控制器，因为它们已经内置到AngularJS中了。这些功能使您的应用程序逻辑很容易编写、测试、维护和理解。

#### 模型数据（Data）

模型是从AngularJS作用域对象的属性引申的。模型中的数据可能是Javascript对象、数组或基本类型，这都不重要，重要的是，他们都属于AngularJS作用域对象

AngularJS通过作用域来保持数据模型与视图界面UI的双向同步。一旦模型状态发生改变，

AngularJS会立即刷新反映在视图界面中，反之亦然。

下面的插图显示了AngularJS应用的各部分，以及它们协同工作原理：

#### AngularJS应用引导过程有3个重要点：

**1、注入器(injector)将用于创建此应用程序的依赖注入(dependency󰀃injection)；**

**2、注入器将会创建根作用域作为我们应用模型的范围；**

**3、AngularJS将会链接根作用域中的DOM，从用ngApp标记的HTML标签开始，逐步处理DOM中指令和绑定。**

#### 模板|控制器

**view代码（html文件）**

<p>模板</p>

<div **ng-controller="PhoneListCtrl"**>

<ul>

<li **ng-repeat="phone in phones"**>

{{phone.name}}

<p>{{phone.snippet}}</p>

</li>

</ul>

</div>

**controller代码（js文件）**

function PhoneListCtrl($scope) {

$scope.phones = [

{"name": "Nokia",

"snippet": "This is Nokia"},

{"name": "Moto",

"snippet": "This is Moto"},

{"name": "iphone",

"snippet": "This is iphone"},

];

}

#### 模板+过滤器

<p>模板+过滤器</p>

<div ng-controller="PhoneListCtrl">

Search:<input ng-model="query"/>

<ul>

<li ng-repeat="phone in phones | filter: query">

{{phone.name}}

<p>{{phone.snippet}}</p>

</li>

</ul>

</div>

其他内容都一样，需要注意上面红色字体

#### 模板+OrderBy

view代码（html）

<p>模板+过滤器+排序</p>

<div ng-controller="PhoneListCtrl">

Search:<input ng-model="query"/>

<select ng-model="orderProp">

<option value="name">Alphabetical</option>

<option value="snippet">Newest</option>

</select>

<ul>

<li ng-repeat="phone in phones | filter: query | orderBy:orderProp">

{{phone.name}}

<p>{{phone.snippet}}</p>

</li>

</ul>

</div>

controller代码（js）

function PhoneListCtrl($scope) {

$scope.phones = [

{"name": "Nokia",

"snippet": "a、 This is Nokia"},

{"name": "Moto",

"snippet": "b、 This is Moto"},

{"name": "iphone",

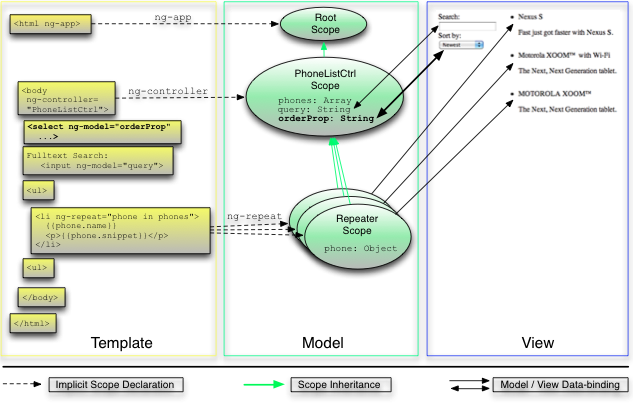
"snippet": "c、 This is iphone"},

];

$scope.orderProp = 'name';

}

注意：controller红色字体表示默认排序方式，如果去掉则没有默认方式。图片仅供参考



#### XHR注入

view代码（html）

<div ng-controller="PhoneListCtrlByjson">

<ul>

<li ng-repeat="phone in phones">

{{phone.name}} <br> age: {{phone.age}}

<p>{{phone.snippet}}</p>

</li>

</ul>

</div>

重点在controller代码（js）

function PhoneListCtrlByjson($scope,$http) {

$http.get('json.json').success(function(data){

$scope.phones = data;

});

$scope.orderProp = 'name';

}

这里需要请求的路径为相对路径

#### 文件压缩

如上所述，在编写controller的时候总要传入一些固定的注入参数，这些参数名字是不能改变的（如果改变了则无效），那么怎么样避免因为js文件压缩导致的参数名称改变呢，可以使用如下方法

function PhoneListCtrlByjson($scope,$http) {

$http.get('json.json').success(function(data){

$scope.phones = data;

});

$scope.orderProp = 'name';

}

PhoneListCtrlByjson.$inject = ['$scope','$http'];

也可以

var PhoneListCtrl = ['$scope', '$http', function($scope, $http) { /\* constructor body \*/ }];

#### 路由与多视图

angular.module('phonecat', ['ngRoute']).

config(['$routeProvider', function($routeProvider) {

$routeProvider.

when('/phones', {templateUrl: 'phoneDetial.html', controller: PhoneListCtrl}).

when('/phones/:phoneId', {templateUrl: 'phoneDetial.html', controller: PhoneDetailCtrl}).

otherwise({redirectTo: 'phoneDetial.html'});

}]);

需要注意的红色字体。最下面的一堆红色字体要注意的是语法结构

#### Module

angular.module(‘moduleName’,[‘arg’]).XXXX

XXXXX可是是Filter，config，controller等。

关于AngularJS的介绍学习就这么多，如果工作中需要使用可以参考<http://docs.angularjs.org/api>进行查学。

补：

#### $watch

从这个名字我们也能理解的差不多，他到底做了些什么，没错，他就是一个监视器（个人理解）那么他是一个监视器？NO，NO，NO！每一个绑定到UI上的属性都会创建一个$watch，也就是“一对一”。

#### 浏览器事件循环和Angular.js扩展

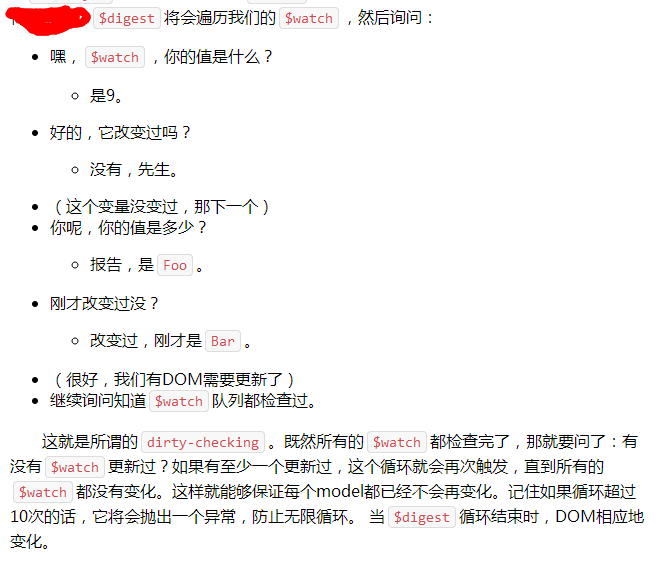
*我们的浏览器一直在等待事件，比如用户交互。假如你点击一个按钮或者在输入框里输入东西，事件的回调函数就会在javascript解释器里执行，然后你就可以做任何DOM操作，等回调函数执行完毕时，浏览器就会相应地对DOM做出变化。 Angular拓展了这个事件循环，生成一个有时成为angular context的执行环境（记住，这是个重要的概念），为了解释什么是context以及它如何工作，我们还需要解释更多的概念。（摘自网络*[*http://www.angularjs.cn/A0a6*](http://www.angularjs.cn/A0a6)*）*

#### $**digest**

*还记得我前面提到的扩展的事件循环吗？当浏览器接收到可以被angular context处理的事件时，$digest循环就会触发。这个循环是由两个更小的循环组合起来的。一个处理evalAsync队列，另一个处理$watch队列，这个也是本篇博文的主题（摘自网络）*

它就是一个收集器，每一个$watch都要向他汇报，每一监视的改变都会触发$digest(实际上是$apply触发的,下面在详细说明)，它会与每一$watch对话，来确保每一个属性发生过改变没有，直到这一次循环中每一个$watch都告诉他自己没有改变过。，下面是摘自网友的一段模拟对话，很有趣，也很能反应这种情况

注：在下图中提到询问循环次数不会超过10次，否则会出现异常。



#### $apply

他来决定哪些东西会进入angular context，也就是决定了哪些会进行dirty-checking。也就是决定了哪些会触发$digest。

*谁决定什么事件进入angular context，而哪些又不进入呢？$apply！如果当事件触发时，你调用$apply，它会进入angular context，如果没有调用就不会进入。现在你可能会问：刚才的例子里我也没有调用$apply啊，为什么？Angular为了做了！因此你点击带有ng-click的元素时，时间就会被封装到一个$apply调用。如果你有一个ng-model="foo"的输入框，然后你敲一个f，事件就会这样调用$apply("foo = 'f';")。（摘自网络）*

如果想要自定义事件也能触发监视那么可以是使用这样的代码

element.bind('click', function() {

scope.$apply(function() {

scope.foo++;

scope.bar++;

});

})

在这里针对$watch $digest $apply 三个方法的协作形式也就说的比较清楚了。

注：在整个模板结构中要注意 ng-app 和 ng-contraller等等，最主要的ag-app，这个要指定一个module模块，否则在模块内定义的任何controller都是无效的。

# 2014-4-1

## Q&A about AngularJS

1. **使用Route功能路由其它html是以什么形式加载的**？

**答**：在传统界面中如果需要引入外部的html可以使用object元素或者是iframe等，但是在AngularJS中是使用div,将属性ng-view。将另外一个页面的内容（扣除html、body、head标签）加载到新页面的DOM中。

注意：如果在head或者body中存在script元素，虽然会加载，但是是无效的，避免此问题的办法就是将js文件独立完成作为父页面的直接引用。

1. **上面提到过module，What is module**？

**答：**module就是一个容器，可以封装services，fiter，directives，config，controller等。

1. What is “services”？

答：services就是一组方法，一种能执行一个常见操作的单例。

1. What is “directives”？

答：*使我们用来扩展浏览器能力的技术之一。在DOM编译期间，和HTML关联着的指令会被检测到，并且被执行。这使得指令可以为DOM指定行为，或者改变它。AngularJS有一套完整的、可扩展的、用来帮助web应用开发的指令集，它使得HTML可以转变成“****特定领域语言(DSL)****”（摘自网络*<http://www.angularjs.cn/A00r>）

1. 如何手动触发$apply()方法？

答：网站给出的答案是使用$scope.$apply(function(){}); 但是在测试时提示$scope没有apply方法。待解

有如下几个方法需要知道怎么使用，在什么时候使用，factory，directive[[1]](#footnote-1)

#### factory

工厂函数是用来创建指令的。它只会被调用一次：就是当编译器第一次匹配到相应指令的时候。你可以在其中进行任何初始化的工作。调用它时使用的是 $injector.invoke， 所以它遵循所有注入器的规则。

### 表单

<form name="myForm"></form>

<ng-form name="myForm"></ng-form>

这两种写法都能定义一个angularJS的表单

**表单验证预提交**

<form name="myForm" ng-submit="save()">

<input name="personEmail" required type="email" ng-model="person.email"/>

<span ng-show="!myForm.personEmail.$valid">有错</span>

<span ng-show="myForm.personEmail.$error.required">必填</span>

<span ng-show="myform.personEmail.$error.email">email 地址不对</span>

</form>

注：在AngularJS 1.0.7 input form的名字要用驼峰， 否者有问题。

## Ember

环境搭建：

页面需要引用如下几个文件

<script src="js/libs/jquery-1.10.2.js"></script>

<script src="js/libs/handlebars-1.1.2.js"></script>

<script src="js/libs/ember-1.5.0.js"></script>

# 2014-4-2

## AngularJS

### $location

$loaction就是对当前URL操作的一个对象。

### $location服务的配置

要配置$location服务，检索$locationProvider并把参数设置成以下这样：

**html5Mode**(模式): {boolean}

**true** - HTML5模式

**false** - Hashbang模式

**default**: false

**hashPrefix**(前缀): {string}

**Hashbang** URLs的前缀 (在Hashbang模式中或者低级浏览器中使用)

**default**: '!'

**样例**

var app = angular.module('app',[function($locationProvider) {

$locationProvider.html5Mode(true).hashPrefix('!');

}]);

定义一个控制器将$location赋值给全局$scope。

function XXXX($scope, $location) {

$scope.$location = $location;

}

注：红色字体可以是任意名称，如果为$aaa则在绑定时就只能用$aaa，符合命名意义，所以使用$location

$location只能修改url，但是不能进行页面跳转或者重新加载。

疑问：想不出来$location的用途在哪里，等待后面学习能够找到答案。

问什么我只设置了path，而url却自动添加了“#”。

### $location的方法

[只读方法] absUrl() 绝对url

[只读方法] protocol() http

[只读方法] host() localhost

[只读方法] port() 端口

[读写方法] url()

[读写方法] path()

[读写方法] search()

[读写方法] hash() #后面的东西

### bootstrap

动态指定ng-app和初始化工作，

angular.bootstrap(element,[function(){//初始化工作}])；

### Animations

要是用Animations必须要引用angular-animate.js文件，并且指定ng-app=“ngAnimate”

能够支持动画的directives有：ngRepeat，ngInclude，ngIf，ngSwitch，ngShow，ngHide，ngView和ngClass。

| ***Directive*** | ***Supported Animations*** |
| --- | --- |
| [*ngRepeat*](http://docs.angularjs.org/api/ng/directive/ngRepeat#usage_animations) | *enter, leave and move* |
| [*ngView*](http://docs.angularjs.org/api/ngRoute/directive/ngView#usage_animations) | *enter and leave* |
| [*ngInclude*](http://docs.angularjs.org/api/ng/directive/ngInclude#usage_animations) | *enter and leave* |
| [*ngSwitch*](http://docs.angularjs.org/api/ng/directive/ngSwitch#usage_animations) | *enter and leave* |
| [*ngIf*](http://docs.angularjs.org/api/ng/directive/ngIf#usage_animations) | *enter and leave* |
| [*ngClass*](http://docs.angularjs.org/api/ng/directive/ngClass#usage_animations) | *add and remove (the CSS class(es) present)* |
| [*ngShow*](http://docs.angularjs.org/api/ng/directive/ngShow#usage_animations)*&*[*ngHide*](http://docs.angularjs.org/api/ng/directive/ngHide#usage_animations) | *add and remove (the ng-hide class value)* |
| [*form*](http://docs.angularjs.org/api/ng/directive/form#usage_animations)*&*[*ngModel*](http://docs.angularjs.org/api/ng/directive/ngModel#usage_animations) | *add and remove (dirty, pristine, valid, invalid & all other validations)* |

**样例**

.module {

margin: 0;

border: solid 1px #888;

background: #efefee;

width: 90%;

line-height: 20px;

}

.module.ng-hide-add,

.module.ng-hide-remove {

display: block !important;

}

.module.ng-hide-add.ng-hide-add-active,

.module.ng-hide-remove.ng-hide-remove-active {

transition: all linear 0.5s;

-webkit-transition:all linear 0.5s;

}

.module.ng-hide {

line-height: 0.1;

opacity: 0;

}

备注：ng-hide-add-active执行动作。动画在这里。当然也可以改变组合方式，而达到相同的效果。所有的动画效果都是使用css3实现的，所以要注意浏览器版本。IE6,7就放弃吧~！

### HTML Compiler

这里的使用会牵涉到Directives的内容。所以在此就简单做一个定义描述：

HTML Compiler就是完成自定义元素或者元素行为，对基本HTML的一个扩展。

### module

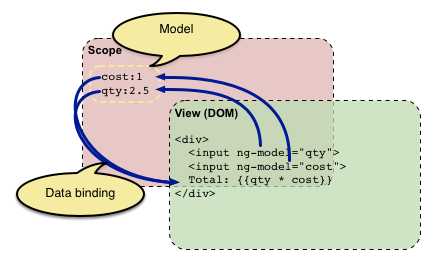
module就是Angular的程序入口，也就相当于其他语言的main函数。那么所有初始化工作都可以在这里执行。

# 2014-4-3

### 概念

| **Concept** | **Description** |
| --- | --- |
| [Template](http://docs.angularjs.org/guide/concepts#template) | HTML with additional markup |
| [Directives](http://docs.angularjs.org/guide/concepts#directive) | extend HTML with custom attributes and elements  扩展HTML的属性或者是元素 |
| [Model](http://docs.angularjs.org/guide/concepts#model) | the data that is shown to the user and with which the user interacts  显示并且与用户交互的数据 |
| [Scope](http://docs.angularjs.org/guide/concepts#scope) | context where the model is stored so that controllers, directives and expressions can access it  能够访问的一个上下文，全局变量 |
| [Expressions](http://docs.angularjs.org/guide/concepts#expression) | access variables and functions from the scope  从Scope访问的一个变量或者是函数。 |
| [Compiler](http://docs.angularjs.org/guide/concepts#compiler) | parses the template and instantiates directives and expressions  解析模板，实例化指令和表达式 |
| [Filter](http://docs.angularjs.org/guide/concepts#filter) | formats the value of an expression for display to the user  格式化表达式或者是值以便呈献给用户。（过滤） |
| [View](http://docs.angularjs.org/guide/concepts#view) | what the user sees (the DOM) |
| [Data Binding](http://docs.angularjs.org/guide/concepts#databinding) | sync data between the model and the view  同步model和view之间的数据 |
| [Controller](http://docs.angularjs.org/guide/concepts#controller) | the business logic behind views |
| [Dependency Injection](http://docs.angularjs.org/guide/concepts#di) | Creates and wires objects / functions  创建和激活objects和functions |
| [Injector](http://docs.angularjs.org/guide/concepts#injector) | dependency injection container |
| [Module](http://docs.angularjs.org/guide/concepts#module) | configures the Injector  配置注入 |
| [Service](http://docs.angularjs.org/guide/concepts#service) | reusable business logic independent of views  抽象业务逻辑功能成为可重用的。 |

图解各个部门的定义



#### 业务独立：Services

**代码如下**

**功能抽象**

**angular.module('finance2', [])**

**.factory('currencyConverter', function() {**

**var currencies = ['USD', 'EUR', 'CNY'];**

**var usdToForeignRates = {**

**USD: 1,**

**EUR: 0.74,**

**CNY: 6.09**

**};**

**var convert = function (amount, inCurr, outCurr) {**

**return amount \* usdToForeignRates[outCurr] / usdToForeignRates[inCurr];**

**};**

**return {**

**currencies: currencies,**

**convert: convert**

**};**

**});**

**Controller引用Factory**

angular.module('invoice2', ['finance2'])

.controller('InvoiceController', ['currencyConverter', function(currencyConverter) {

this.qty = 1;

this.cost = 2;

this.inCurr = 'EUR';

this.currencies = currencyConverter.currencies;

this.total = function total(outCurr) {

return currencyConverter.convert(this.qty \* this.cost, this.inCurr, outCurr);

};

this.pay = function pay() {

window.alert("Thanks!");

};

}]);

HTML中重点代码

<div>

<b>Total:</b>

<span ng-repeat="c in invoice.currencies">

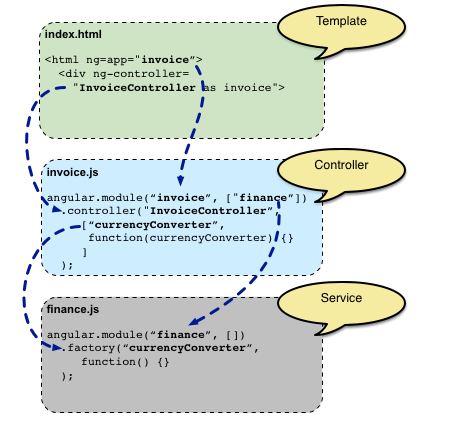
{{invoice.total(c) | currency:c}}

</span>

<button class="btn" ng-click="invoice.pay()">Pay</button>

</div>

#### 整个逻辑的图例说明



### Controllers

使用Controllers完成以下事情

**1、设置好作用域对象的初始状态。**

**2、给作用域对象增加行为。**

不要使用Controllers完成的事情

**1、控制器应该只关心业务逻辑。DOM操作（表现层逻辑）通常会把测试弄得很难。将任何表现层逻辑放到控制器中都会显著地增加对业务逻辑的测试难度。AngularJS提供**

**directives和databinding用来自动进行DOM操作。如果你需要手动操作DOM，将表现层的逻辑抽离到指令中。**

**2、对输入格式化 — 你应该用AngularJS的表单控制来实现格式化。.**

**3、对输出格式化 — 该用AngularJS的过滤器实现。.**

**4、在控制器中运行无状态或者有状态但在控制器中共享的代码 — 该用服务来实现.**

**实例化组件或者控制其它组件的生命周期（比如创建一个服务的实例）.**

### Working with CSS

ng-binding：所有绑定数据的元素都会有这个class，也就是元素中存在{{}}的。

ng-scope：绑定了ng-app的元素

# 2014-4-4

## DI

标准的ID设计模式，通过字符串进行对象传递，降低耦合。

样例代码

controller('say',['$scope','sayHello',function($scope,sayhello) {

$scope.say = function(text) {

sayhello.say(text);

}

}])

factory('sayHello',function() {

return {

say: function(text) {

alert('Hello,' + text);

}

};

});

注：只有factory才可以注入，directive是不可以注入的。

## 指令（directive）

指令是angular渲染的核心技术，针对html的渲染首先是按照标准api生成DOM，angular调用$compile形成一个队列添加进来，然后进行compile（整个过程是有序的），为DOM生成对应的link，link就是监听了每一个DOM节点。

命令中标准的格式（有一些是可以省略）

var myModule = angular.module(...);

myModule.directive('directiveName', function factory(injectables) {

var directiveDefinitionObject = {

priority: 0,

template: '<div></div>',

templateUrl: 'directive.html',

replace: false,

transclude: false,

restrict: 'A',

scope: false,

compile: function compile(tElement, tAttrs, transclude) {

return {

pre: function preLink(scope, iElement, iAttrs, controller) { ... },

post: function postLink(scope, iElement, iAttrs, controller) { ... }

}

},

link: function postLink(scope, iElement, iAttrs) { ... }

};

return directiveDefinitionObject;

});

针对每一个命令的解释

*名称name - 当前作用域的名称，在注册是可选的。*

*优先级priority - 当一个DOM上有多个指令时，有会需要指定指令执行的顺序。 这个优先级就是用来在执行指令的compile函数前先排序的。高优先级的先执行。 相同优先级的指令顺序没有被指定谁先执行。*

*终端terminal - 如果被设置为true，那么该指令就会在同一个DOM的指令集和中最后被执行。任何其他“terminal”的指令也仍然会执行，因为同级的指令顺序是没有被定义的。*

*作用域scope- 如果被定义成：*

*那么就会为当前指令创建一个新的作用域。如果有多个在同一个DOM上的指令要求创建新作用域，那么只有一个新的会被创建。 这一创建新作用域的规则不适用于模板的根节点，因为模板的根节点总是会得到一个新的作用域。*

*{} 对象哈希 - 那么一个新的“孤立的”作用域就会被创建。这个“孤立的”作用域区别于一般作用域的地方在于，它不会以原型继承的方式直接继承自父作用域。这对于创建可重用的组件是非常有用的，因为可重用的组件一般不应该读或写父作用域的数据。 这个“孤立的”作用域使用一个对象哈希来表示，这个哈希定义了一系列本地作用域属性， 这些本地作用域属性是从父作用域中衍生出来的。这些属性主要用来分析模板的值。这个哈希的键值对是本地属性为键，它的来源为值。*

*@ 或 @attr - 将本地作用域成员成员和DOM属性绑定。绑定结果总是一个字符串，因为DOM的属性就是字符串。如果DOM属性的名字没有被指定，那么就和本地属性名一样。比如说<widget my-attr="hello {{name}}"> 和作用域对象: { localName:'@myAttr' }。当name值改变的时候， 作用域中的LocalName也会改变。这个name是从父作用域中读来的（而不是组件作用域）。*

*= 或 =expression(表达式) - 在本地作用域属性和父作用域属性间建立一个双向的绑定。如果没有指定父作用域属性名称，那就和本地名称一样。 比如 <widget my-attr="parentModel"> 和作用域对象: { localModel:'=myAttr' }, 本地属性localModel会反映父作用域中parentModel的值。localModel和parentModel的任一方改变都会影响对方。*

*& 或 &attr - 提供了一种能在父作用域下执行表达式的方法。如果没有指定父作用域属性名称，那就和本地名称一样。 比如 <widget my-attr="count = count + value">和作用域对象：{ localFn:'increment()' }。本地作用域成员localFn会指向一个increment表达式的函数包装。通常你可以通过这个表达式从本地作用域给父作用域传值， 操作方法是将本地变量名和值得对应关系传给这个表达式的包装函数。比如说，这个表达式是increment(amount)，那么你就可以用调用localFn({amount:22})的方式指定amount的值。*

*控制器controller - 控制器的构造对象。这个控制器函数是在预编译阶段被执行的，并且它是共享的，其他指令可以通过它的名字得到（参考依赖属性[require attribute]）。这就使得指令间可以互相交流来扩大自己的能力。会传递给这个函数的参数有：*

*$scope - 当前元素关联的作用域。*

*$element - 当前元素*

*$attrs - 当前元素的属性对象。*

*$transclude - 模板链接功能前绑定到正确的模板作用域：function(cloneLinkingFn)。*

*请求require - 请求将另一个控制器作为参数传入到当前链接函数。 这个请求需要传递被请求指令的控制器的名字。如果没有找到，就会触发一个错误。请求的名字可以加上下面两个前缀：*

*?- 不要触发错误，这只是一个可选的请求。*

*^ - 没找到的话，在父元素的controller里面也查找有没有。*

*限制restrict - EACM中的任意一个之母。它是用来限制指令的声明格式的。如果没有这一项。那就只允许使用属性形式的指令。*

*E - 元素名称：<my-directive></my-directive>*

*A - 属性： <div my-directive="exp"> </div>*

*C - 类名：<div class="my-directive: exp;"></div>*

*M - 注释： <!-- directive: my-directive exp -->*

*模板template - 将当前的元素替换掉。 这个替换过程会自动将元素的属性和css类名添加到新元素上。更多细节请查考章节“创建widgets”。*

*模板templateUrl - 和template属性一样，只不过这里指示的是一个模板的URL。因为模板加载是异步的，所有编译和链接都会等到加载完成后再执行。*

*替换replace - 如果被设置成true那么现在的元素会被模板替换，而不是被插入到元素中。*

*编译模板transclude - 将元素编译好，使得指令可以开始使用它。一般情况下需要和ngTransclude指令一起使用。 使用嵌入的好处在于链接好书可以获取到预绑定在作用域上的函数。在一个典型的初始化过程中，widget会创建一个孤立的作用域，但是嵌入并不是其中一个子成员，而是这孤立作用域的兄弟成员。这使得widget可以有一个私有的状态，并且嵌入被绑定在父作用于上。*

*true - 嵌入指令的内容。*

*'element' - 嵌入整个元素，包括优先级较低的指令。*

*编译compile - 这就是后面将要讲到的编译函数。*

*链接link - 这就是后面将要讲到的链接函数。只有没有提供编译函数时才会用到这个值。*

注：这段完全copy自网络，记不全，只能记住常用的几个，其他的记录下来方便以后查阅

### E2E Testing

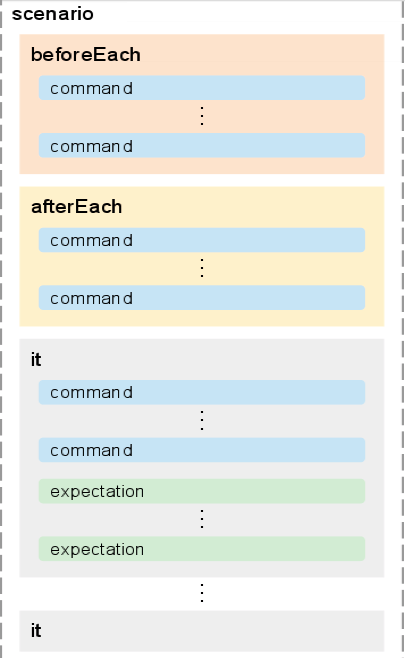
组成：

一个discribe包含一个或者多个 it 模块。

一个it 模块是有表达式和command组成。

beforEach和afterEach：在遍历每一个it模块之前或者之后， 不管it模块是否通过。

组成结构图：



API介绍：

pause（）：暂停当前的测试单元，一直等到在控制台调用resume（）或者在测试UI上点击resume（）的连接。

sleep（ns）：暂停执行测试单元n秒。

browser（）.navugateTo（url）：将url加载到测试框架内。

browser（）.navugateTo（url，fn）：Loads the URL returned by fn into the testing frame. The given url is only used for the test output. Use this when the destination URL is dynamic (that is, the destination is unknown when you write the test).（这个是原文，不明白是怎么使用）

答案：在test frame中加载fn返回的url地址。这里的url参数只是用作测试输出。当目的url是动态的时候可以使用这个API（写测试的时候，地址还是未知的）。

browser（）.reload（）：重新加载

browser（）.window（）.href（）：返回window.href， 与此类似的用法还有，browser（）.window（）.path（）、search（），hash（）等。

与window对应的还有location的使用，browser（）.location（）.path（）、search，hash，url（）等。

expect（future）.{matcher}：断言 future和matcher匹配。futher：表达式等，matcher：指定结果（期望值）。

与之相反的还有expect（futher）.not（）.{matcher}.

using(selector,label):选择范围内的下一个DSL元素。

binding（name）：返回匹配绑定的值（只找到第一个）

input（name）.enter(value)：将指定的value键入指定的input。

注：是匹配ng-model=name，而不是name=name的。

与此类似的就是input（name）.check（），check一次checkbox，等同于用鼠标点击一下checkbox。input（name）select（value），使radio button 选择指定的value。

input（name）.val（）：返回值。

repeater（selector，label）.count（）：lable用于测试输出。返回行数。selector是jquery的selector方式。与此类似的还有repeater（selector，lable）.row(index)返回指定行的数据，还有colume（binding）：指定列的值。binding格式是怎么样的？

答案：repeater('#products table', 'Product List').column('product.name')

select（name）.option(value[,value2,value3…])：选择被指定值的对象。

element（selector，label）.count()：返回满足selector的元素数量。

element（selector，label）click（）：触发满足selector的click事件。

element（selector，label）.query(fn)：fn就是function（selecteElements，done），也就是找到匹配的元素执行done回调函数。

element（selector，label）.{mehod}（）：method：可能是jQuery中的val，

text，height，innerHeigth，outerHeight，width，innerWidth，outerWidth，pasition，scrollLeft，scrollTop，offset等方法。

element（selector，label）.{method}(value):类似set方法，设置value。

element（selector，label）.{method}(key，[value]):返回结果。可以是jQuery中的attr，prop，css等方法，如果有value参数，则是要设置对应的值。

注：下面咱就说一下matcher都可以是什么形式

1. expect(value).toEqual(value)
2. expect(value).toBe(value)
3. expect(value).toBeDefind()
4. expect(value).toBeTruethy()
5. expect(value).toBeFalsy()
6. expect(value).toMatch(expectedRegExp)
7. expect(value).toBeNull()
8. expect(value).toContain(expected)
9. expect(value).toBeLessThan(expected)
10. expect(value).toBeGreaterThan(expected)

### 表达式

注：表达式中不存在流控制语句。

强大的容错能力。不会出现undefined.a的Type Error。

在事件处理时会有$event参数，但是此参数是绑定在作用域外面的，可是使用它，但是{{$event}}则不会绑定数据。

### Filter

内置Filter(来自<http://docs.angularjs.org/api/ng#filter>，我加上翻译)

|  |  |
| --- | --- |
| **Name** | **Description** |
| [filter](http://docs.angularjs.org/api/ng/filter/filter) | Selects a subset of items from array and returns it as a new array.  对array进行过滤得到新的array返回。 |
| [currency](http://docs.angularjs.org/api/ng/filter/currency) | Formats a number as a currency (ie $1,234.56). When no currency symbol is provided, default symbol for current locale is used.  格式化为货币样式，会提供一个本地默认符号。 |
| [number](http://docs.angularjs.org/api/ng/filter/number) | Formats a number as text.  格式化为数字 |
| [date](http://docs.angularjs.org/api/ng/filter/date) | Formats date to a string based on the requested format.  字符串格式化成日期 |
| [json](http://docs.angularjs.org/api/ng/filter/json) | Allows you to convert a JavaScript object into JSON string.  将js对象格式化为json字符串 |
| [lowercase](http://docs.angularjs.org/api/ng/filter/lowercase) | Converts string to lowercase.  小写 |
| [uppercase](http://docs.angularjs.org/api/ng/filter/uppercase) | Converts string to uppercase.  大写 |
| [limitTo](http://docs.angularjs.org/api/ng/filter/limitTo) | Creates a new array or string containing only a specified number of elements. The elements are taken from either the beginning or the end of the source array or string, as specified by the value and sign (positive or negative) of limit.  边界值，创建一个新的array或者是string，这个指定的值是原数据源的上下边界值或者是符号。 |
| [orderBy](http://docs.angularjs.org/api/ng/filter/orderBy) | Orders a specified array by the expression predicate.  通过制定的表达式进行排序 |

### 表单（Form）

表单样式：

ng-valid：有效的

ng-invalid：无效的

ng-pristine：从未输入

ng-dirty：已经输入的

ng-model-option：是无效的，官网给的意思是可以通过制定这个属性来改变默认的行为，不如失去焦点更新数据等等。

非立即更新模式（建立在ng-model-option之上）：可以设置延迟更新，例如

ng-model-options="{ updateOn: 'default blur', debounce: { default: 500, blur: 0 } }"

疑问：编写的测试文件怎么用，引用之后没有错，那么在哪里能看到测试结果呢？明天找答案

# 2014-4-6

i18n和l10n：

i18n：国际化。

l10n：本地化。

可以通过配置不同的localID来控制不同的js文件。

1、在\*nix的操作系统里，可以使用命令

cat angular.js i18n/angular-locale\_de-de.js > angular\_de-de.js

2、可以在html文件中进行引用js文件。

<script src="i18n/angular-locale\_de-de.js"></script>

注：当进行本地化之后需要注意的是就是金额的filter，一定要制定金钱的filter:USD$,否则$100进行本地化之后就变成了￥100，少了很多。

### IE兼容

1. 首先要保证浏览器可以使用JSON对象，如果在&lt; IE8可以引用json2.js
2. 不要使用自定义标签，例如<myElement /> 可以使用<div myElement />
3. 如果想在IE8一下使用自定义标签可以做如下设置

<html xmlns:ng="http://angularjs.org">

<head>

<!--[if lte IE 8]>

<script>

document.createElement('ng-include');

document.createElement('ng-pluralize');

document.createElement('ng-view');

// Optionally these for CSS

document.createElement('ng:include');

document.createElement('ng:pluralize');

document.createElement('ng:view');

</script>

<![endif]-->

</head>

<body>

</body>

</html>

注：xmls:ng - 命名空间 - 对于每一个我们计划使用的自定义标签，都需要有一个命名空间。

document.createElement(“自定义标签名称”) - 自定义标签名称的创建 - 因为这是旧版IE一个问题，我们需要通过IE判断注释（<!--[if lte IE 8]>…<![endif]-->）来特殊处理。对于每一个没有命名空间或者非HTML默认标签，都需要进行这种预定义，以让IE不会没样式等。

如果想让CSS选择器对自定义元素有效，那么自定义元素必须通过document.createElement(“元素名称”)进行预定义

### Modules

使用modules的好处在于

1、这过程是声明描述的，更加容易读懂。

2、在单元测试中，不需要加载所有module，这对写单元测试很有帮助。

3、额外的module可以被加载到情景测试中，可以覆盖一些设置，帮助进行应用的端对端测试（end-to-end test）。

4、第三方代码可以作为可复用的module打包到angular中。

5、module可以通过任意顺序或并行加载（取决于模块执行的延迟性，due to delayed nature of module execution）。

#### config模块

angular.module('myModule', []).

value('a', 123).

factory('a', function() { return 123; }).

directive('directiveName', ...).

filter('filterName', ...);

// 等同于

angular.module('myModule', []).

config(function($provide, $compileProvider, $filterProvider) {

$provide.value('a', 123)

$provide.factory('a', function() { return 123; })

$compileProvider.directive('directiveName', ...).

$filterProvider.register('filterName', ...);

});

通过上面的代码对比就能清楚的看到config的显示和隐式使用

#### Run Blocks（应用块）

run block是在angular中最接近main方法的东东。run block是必须执行，用于启动应用的代码。它将会在所有service配置、injector创建完毕后执行。run block通常包含那些比较难以进行单元测试的代码，就是因为这个原因，这些代码应该定义在一个独立的module中，让这些代码可以在单元测试中被忽略

# 2014-4-9

## 作用域Scope

### 作用域层级

每一个AngularJS应用都有一个绝对的根作用域。但是可能有多个子作用域。

一个应用可以有多个作用域，因为有一些指令会生成新的子作用域（在指令中使用scope会创建新的作用域）。当新作用域被创建的时候，他们会被当成子作用域添加到父作用域下，这使得作用域会变成一个和相应DOM结构一个的树状结构。

当AngularJS执行表达式{{username}}，它会首先查找和当前节点相关的作用域中叫做username的属性。如果没找到，那就会继续向上层作用域搜索，直到根作用域。在Javascript中，这被称为原型类型的继承，子作用域以原型的形式继承自父作用域。

### 作用域的声明周期

浏览器接收到事件后的一般工作流程是执行一个相应的Javascript回调。回调一执行完，浏览器就会重新渲染DOM并且重新回到等待事件的状态。

当浏览器调用AngularJS上下文之外的Javascript代码时，AngularJS是不知道模型的更改的。要正确处理模型的更改，就要使用$apply方法进入AngularJS的执行上下文。只有在$apply方法内执行的模型修改才会正确地被AngularJS处理。比如，一个指令监听DOM事件，比如ng-click，它必须在$apply方法中来执行表达式。

执行完表达式之后，$apply会进入$digest阶段。在$digest阶段作用域会检查所有的$watch表达式，并将它们和之前的值比较。这个检查工作是异步执行的。这意味着赋值语句，如$scope.username="angular"不会马上导致$watch被通知，取而代之的是它会等到$digest阶段才被通知。这种方式是合理的，因为它将多个模型的更新整合到一个$watch通知里，并且保证了一个$watch通知期间不会有其他$watch执行。如果一个$watch改变了模型的值，那么它会产生一个额外的$digest阶段。

### 作用域和指令

在编译阶段，编译器在DOM中匹配指令。指令通常分为两种：

1、观察型的指令，例如双花括号表达式{{expression}}，会用$watch来注册一个监听者。无论表达式什么时候改变，这类型的指令都会被通知，并且能更新视图。

2、监听者型的指令，比如ng-click，会向DOM注册一个监听者。当DOM监听者触发，指令会执行相关的表达式并且使用$apply方法更新视图。

注：会创建作用域的指令有比如ng-controller和ng-repeat会创建新的作用域，并关联到相应的DOM元素上，你该可以使用angular.element(aDomElement).scope()方法来获得某一个DOM元素相关的作用域。

# 2014-4-10

## 服务（services）

AngularJS服务是一种能执行一个常见操作的单例。个人理解，AngularJS的服务就是通过单例模式生成一个（或者多个不同的操作对象）单例对象，提供一系列的功能（function）

注：AngularJS通过“构造器注入”来注入依赖（通过工场函数来传递依赖）。因为Javascript是动态类型语言，AngularJS无法通过静态类型来识别服务，所以你必须使用$inject属性来显式地指定依赖，如下：myController.$inject = ['$location'];

疑问：如何理解$inject

### 服务依赖

服务可以被controller依赖，也可以依赖服务。比如：在服务中可以依赖$document，$window等服务。

注：所有的服务只有被依赖的时候才会实例化。

总结：关于service也没有太多需要记录的，很多东西在前面已经涉及到，更多的就是练习吧。gogogo~！

## github

今天学习了github网页版的使用，包括上传 修改 下载等功能。

上传只能粘贴代码，不能选择文件呢，这一点有点郁闷，明天去问问胡傲果。

学习了github for window的使用，这个简单明了，符合window用户的“鼠标法”。

总结：学会这这两种方式，可以正式步入github的使用，github的命令在使用中学习和充实。

# 2014-4-14

## AngularJS开发：

### 模块依赖

angular.model(‘modelName’,[‘relyName1’,’relyName2’…..];

通过这样的方式可以实现model之间的依赖于继承。

### 开发之前知晓

使用karma编写并且运行单元测试和E2E测试（注：工具有很多，选择一个适合自己的最重要）。

扩展Chrome’s Batarang。angularJS测试工具。

可以使用RequireJS继承angularJS应用。

建议使用Yeoman。

注：如果想要使用Yeoman，最好先熟练一下\*nux平台，在windows下使用它绝对是很大的挑战。当然最好还要先装一个Node库，这样就更方便AngularJS的测试了。

1. 这个地方没有看懂怎么使用，特此记录，以便日后学习 [↑](#footnote-ref-1)