# 复习指南

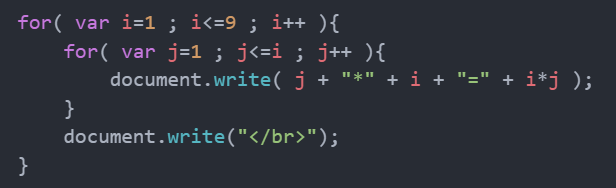
1 HTML

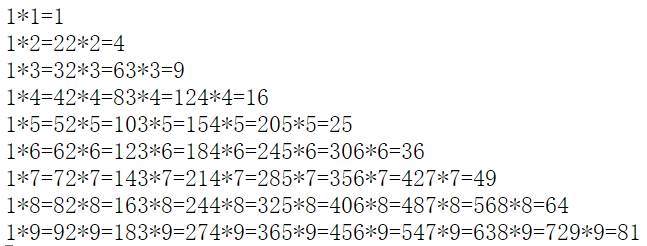
## 2 css

## 3 js基础

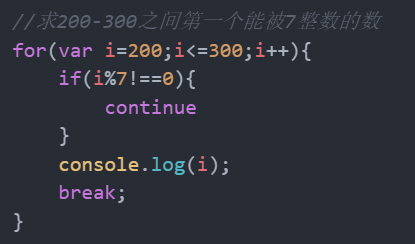
### 3.1 js代码题

#### 3.1.1 九九乘法表

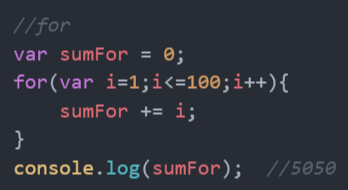
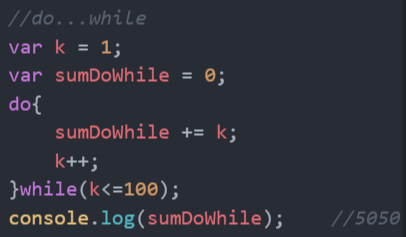




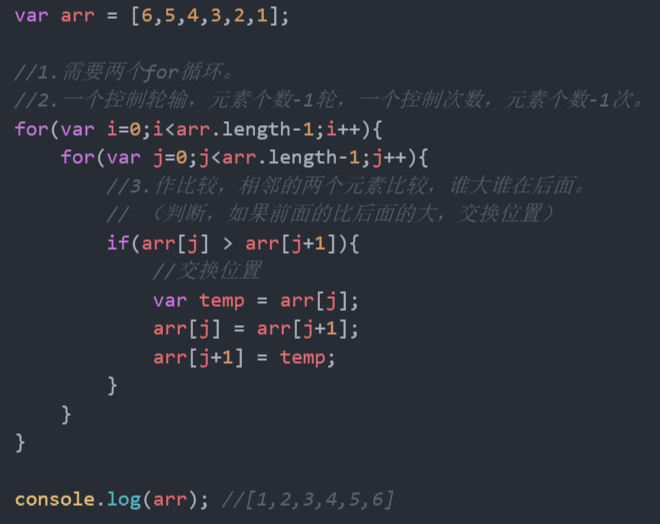
#### 3.1.2 求第一个能被7整除的数



#### 3.1.3 求1加到100

#### 3.1.4 冒泡排序



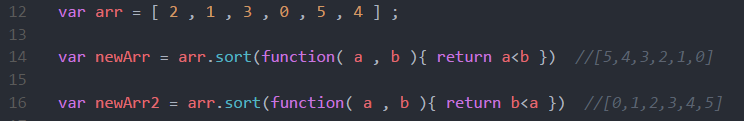
#### 3.1.5 斐波那契数列



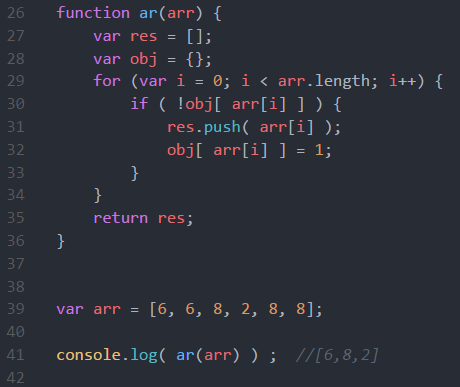
3.1.6 数组反转



3.1.7 数组排序

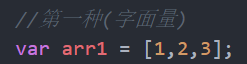


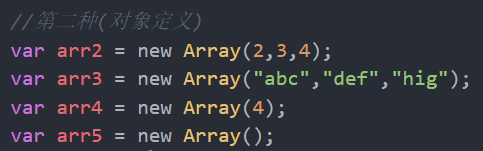
3.1.8 数组去重



### 3.2 数组

#### 3.2.1 数组的定义





#### 3.2.2 数组的操作

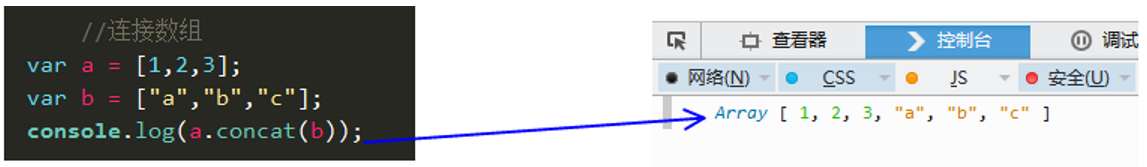
##### shift 删除数组里的第一个，并且返回该值

##### unshift 往数组的第一个位置插入，并且返回数组的长度

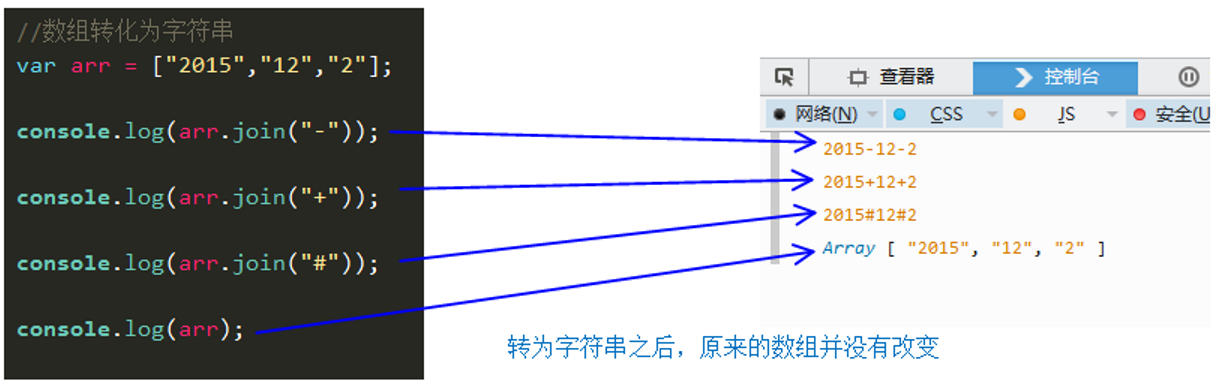
##### pop 删除数组的最后一个，并且返回该值

##### push往数组的后面添加，并且返回数组的长度

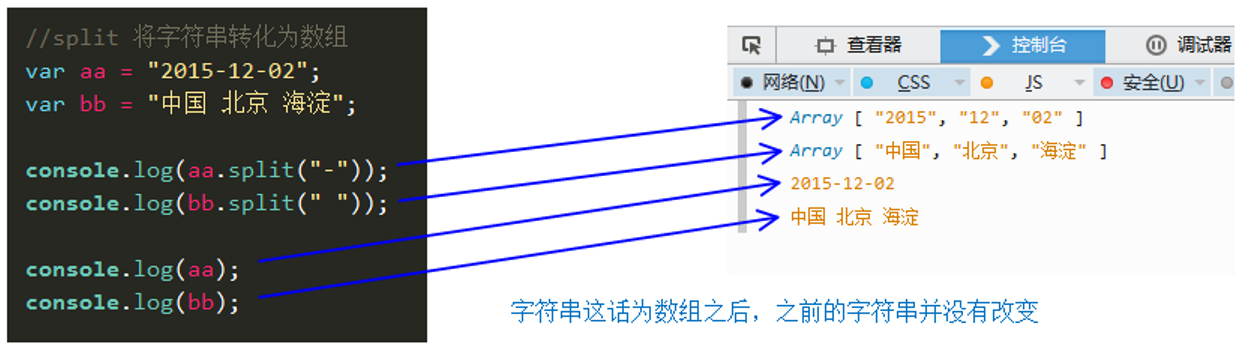
##### concat 连接数组



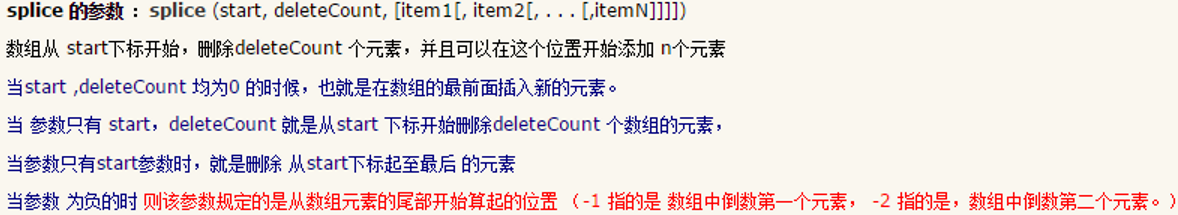
##### join 将数组转换为字符串

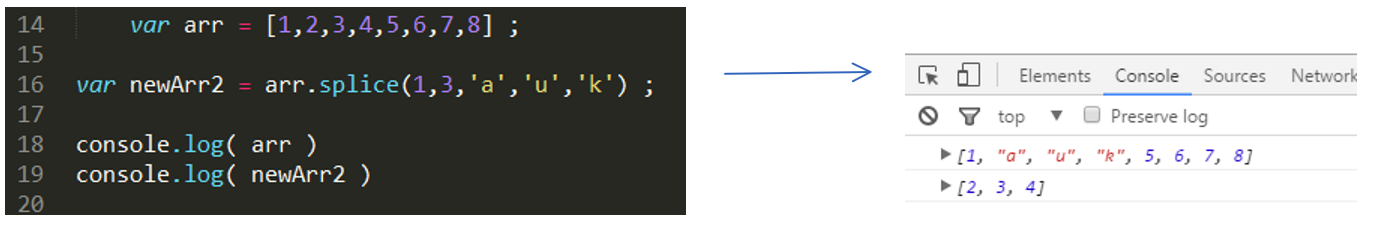


##### split 将字符串分割成字符串数组



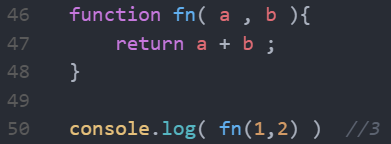
##### splice 用与插入，删除，替换数组的元素

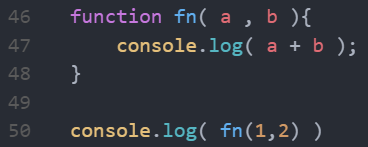




### 3.3 函数

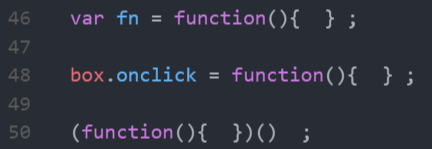
#### 3.3.1 函数的返回值





#### 3.3.2 匿名函数



#### 3.3.3 闭包

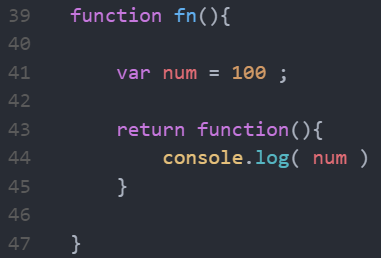
##### 闭包的定义

闭包就是一个函数引用另外一个函数的变量，因为变量被引用着所以不会被回收，因此可以用来封装一个私有变量。这是优点也是缺点，**不必要的闭包只会徒增内存消耗！**

由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成"定义在一个函数内部的函数"。

##### 闭包的应用

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。



### 3.4 对象

创建对象





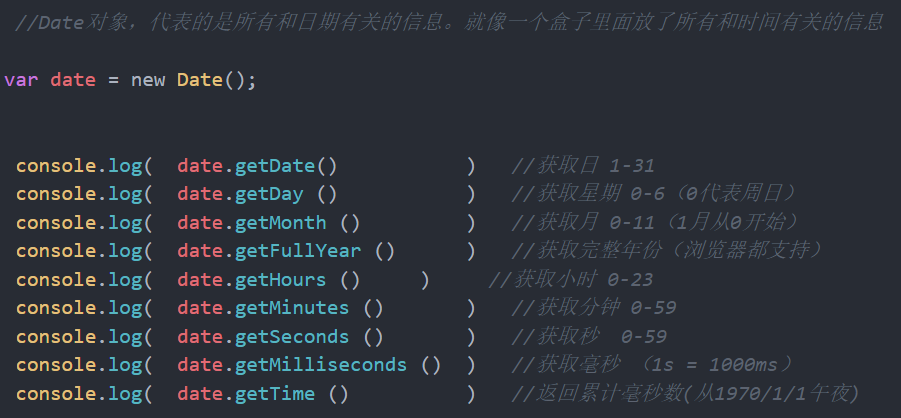
对象的动态特性

var obj = { }

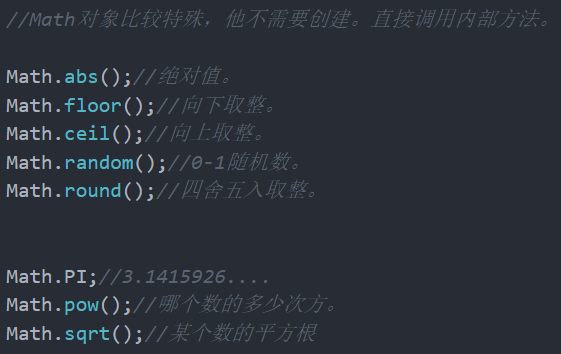
obj.name = 'tom'

obj有name则更改，没有则增加

3.5 date



3.6 Math



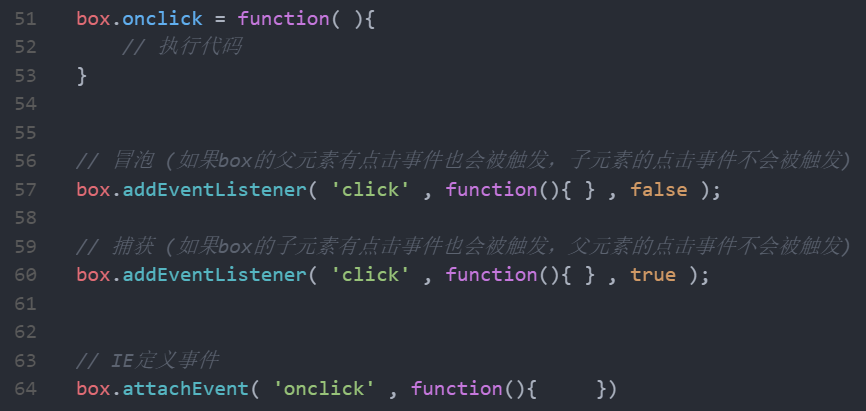
3.7 事件

定义事件

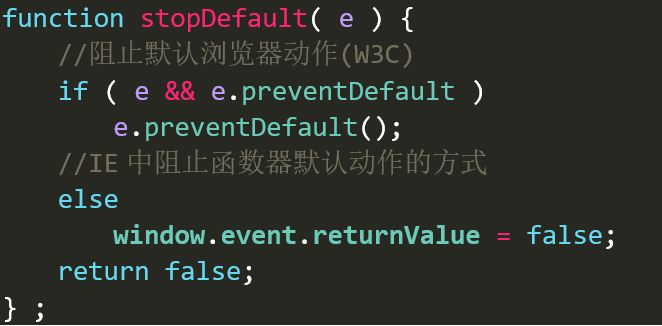
<div father>

<div son></div>

</div>



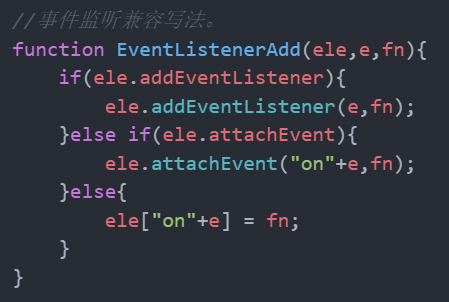
阻止默认事件



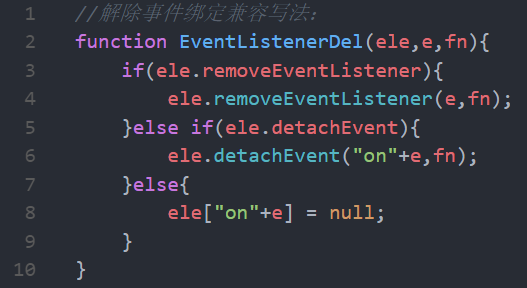
阻止冒泡



事件的兼容写法



解除事件绑定兼容写法



事件委托



3.8 dom

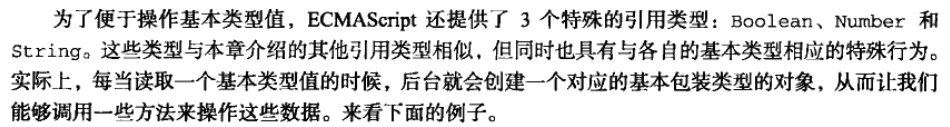
dom树结构

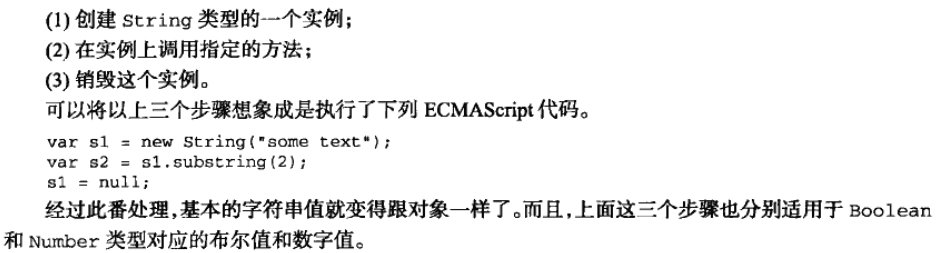


dom操作方法



3.9 三大基本包装类型





3.10 offset系列

Offset系列简介

offset这个单词本身是--偏移，补偿，位移的意思。

js中有一套方便的获取元素尺寸的办法就是offset系列；

offsetWidth和offsetHight 以及offsetLeft和offsetTop以及offsetParent

共同组成了offset系列。

offsetWidth和offsetHeight （检测盒子自身宽高+padding+border）

这两个属性，他们绑定在了所有的节点元素上。获取之后，只要调用这两个属性，我们就能够获取元素节点的宽和高。

offset宽/高 = 盒子自身的宽/高 + padding +border；

offsetWidth = width+padding+border；

offsetHeight = Height+padding+border；

offsetLeft和offsetTop （检测距离父盒子有定位的左/上面的距离）

返回距离上级盒子（带有定位）左边s的位置

如果父级都没有定位则以body为准

offsetLeft 从父亲的padding 开始算,父亲的border 不算。

在父盒子有定位的情况下，offsetLeft == style.left(去掉px)

offsetParent （检测父系盒子中带有定位的父盒子节点）

1、返回改对象的父级 （带有定位）

如果当前元素的父级元素没有进行CSS定位 （position为absolute或 relative，fixed）， offsetParent为body。

2、如果当前元素的父级元素中有CSS定位 （position为absolute或 relative，fixed）， offsetParent取最近的那个父级元素。

offsetLeft和style.left区别

1. 最大区别在于offsetLeft可以返回没有定位盒子的距离左侧的位置。

而 style.left不可以

二、offsetTop 返回的是数字，而 style.top 返回的是字符串，除了数字外还带有单位：px。

三、offsetTop 只读，而 style.top 可读写。（只读是获取值，可写是赋值）

四、如果没有给 HTML 元素指定过 top 样式，则style.top 返回的是空字符串。

3.11 scroll系列

Scroll家族组成

ScrollWidth和scrollHeight（不包括border）

检测盒子的宽高。（调用者：节点元素。属性。）

盒子内容的宽高。（如果有内容超出了，显示内容的高度）

IE567可以比盒子小。 IE8+火狐谷歌不能比盒子小

scrollTop和scrollLeft

网页，被浏览器遮挡的头部和左边部分。

被卷去的头部和左边部分。

兼容性问题

1. 未声明 DTD（谷歌只认识他）

document.body.scrollTop

1. 已经声明DTD（IE678只认识他）

document.documentElement.scrollTop

1. 火狐/谷歌/ie9+以上支持的

window.pageYOffset

兼容写法

var aaa = **window**.**pageYOffset** || **document**.**documentElement**.**scrollTop** || **document**.**body**.**scrollTop**;

var aaa = **document**.**documentElement**.**scrollTop** + **document**.**body**.**scrollTop**;

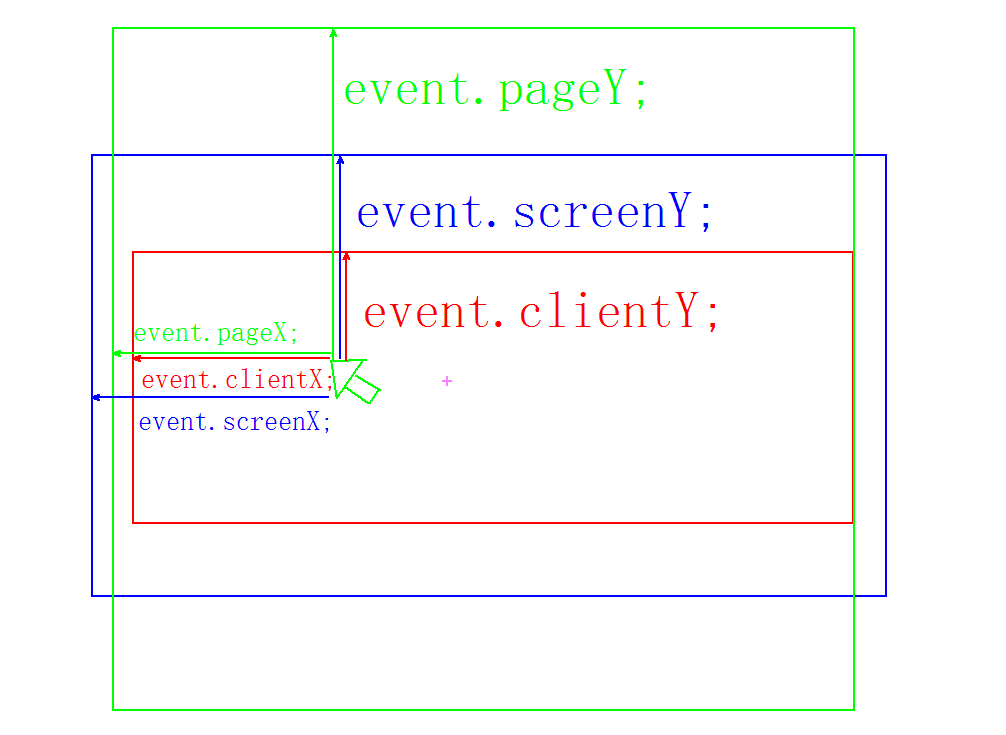
3.12 三大坐标系统

PageY/pageX: 鼠标位于整个网页页面的顶部和左侧部分的距离。（页面）

ScreenY/screenX: 鼠标位于屏幕的上方和左侧的距离。（屏幕）

ClientX/clientY: 鼠标位于浏览器的左侧和顶部的距离。（浏览器大小和位置）

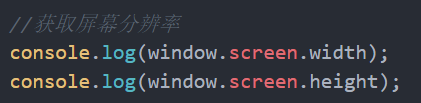
screenX、pageX和clientX的区别



3.13 获取浏览器可视区域宽高



3.14 获取屏幕分辨率



3.15 正则知识点

正则表达式的特点是：

1. 灵活性、逻辑性和功能性非常的强；

2. 可以迅速地用极简单的方式达到字符串的复杂控制。

3. 对于刚接触的人来说，比较晦涩难懂。

比如：

匹配国内电话号码：\d{3}-\d{8}|\d{4}-\d{7}

验证手机号：

/^((13[0-9])|(15[^4,\D])|(18[0,5-9]))\d{8}$/

很难记住。

正则表达式声明

1.通过构造函数定义

var 变量名= new RegExp(/表达式/);

2.通过直接量定义 （较为常用）

var 变量名= /表达式/;

console.log(regexp1.test(345));

console.log(/\d/.test(567));（直接使用）

正则表达式的组成是：(总结)

有一些普通字符和元字符组成，普通字符就是字母和数字，元字符具有特殊意义的字符

匹配腾讯QQ号：[1-9][0-9]{4,}

评注：腾讯QQ号从10000开始

比如 \d

预定义类: 表示数字 [0-9]

test() 方法

正则对象方法，检测测试字符串是否符合该规则，返回true和false，参数（测试字符串）

使用语法：

Boolean = 表达式.test("要验证的内容");

**console**.log(/\d/.test(567));

验证 567 符不符合 \d 的规范

正则内部类（帮我们写好的工具直接使用）

预定义类

. [^\n\r] 除了换行和回车之外的任意字符（“”不行）

\d [0-9] 数字字符

\D [^0-9] 非数字字符

\s [ \t\n\x0B\f\r] 空白字符

\S [^ \t\n\x0B\f\r] 非空白字符

\w [a-zA-Z\_0-9] 单词字符

\W [^a-zA-Z\_0-9] 非单词字符

简单类（正则：//中什么特殊符号都不写，和[]的加入）

1、/string/.test(“string”); 必须是完整的，只多不能少

/andy/.test(“andy”) // true

/andy/.test(“andylv”) // true

/andy/.test(“an”) // false

一句话，只要完整包含了andy 就可以了（有他就行）

2、/[string]/.test(“string”); 只要包含里面的任何一个就可以

/[andy]/.test("andy"); // true

/[andy]/.test("an"); // true

/[andy]/.test("ady"); // true

/[andy]/.test("anll"); // true

/[andy]/.test("assd"); // true

/[andy]/.test("ss"); // false

/[3aH8]/.test("ss"); // false

负向类（不能是其中的整体或者一部分）

中括号内，前面加个元字符^进行取反，不是括号里面的字符（一部分也不行）。

(可以不够，但是不能多)（不够和正好，返回false；多了或者没有返回true）

console.log(/[^abc]/.test('a'));

console.log(/[^abc]/.test('gg'));

注意: 这个符号 ^ 一定是写到方括号里面

范围类

有时匹配的东西过多，而且类型又相同，全部输入太麻烦，我们可以在中间加了个横线

console.log(/[a-z]/.test('1111'));

console.log(/[A-Z]/.test('aa'));

组合类

用中括号匹配不同类型的单个字符。

console.log(/[a-m1-5]/.test("b"))//true

正则边界（重点）

^ 会匹配行或者字符串的起始位置

注：^在[]中才表示非！这里表示开始

$ 会匹配行或字符串的结尾位置

^$在一起 表示必须是这个（精确匹配）

// 边界可以精确说明要什么  
**console**.log(/lily/.test("lilyname")); // true  
**console**.log(/^lily$/.test("lily")); // true  
**console**.log(/^lily$/.test("ly")); // false

console.log(/^andy$/.test("andy")); // true

这个的最终意思就是 说， 必须是 andy 这四个字母

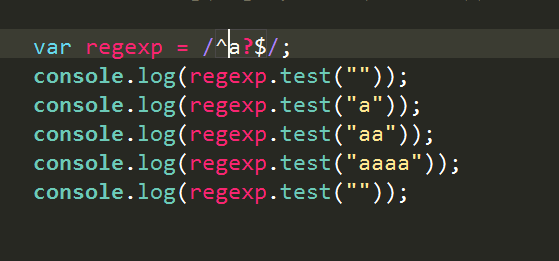
量词（重点）

（多个字母，重复最后一个）

\* 重复零次或更多 (>=0)

+ 重复一次或更多次 (>=1)

? 重复零次或一次 （0||1） 要么有 要么没有

{} 重复多少次的意思 可以有多少个

您的银行卡密码只能是 6位 {6}

{n} n次 （x=n）

{n,} 重复n次或更多 (x>=n)

{n,m} 重复出现的次数比n多但比m少 (n<=x<=m)

\* {0,}

+ {1,}

? {0,1}

x|y 一个 | x 或者 y（没有&，用的是，代替的）

（）提高权限，有限计算

正则优先级

//1.（） 2.^$ 3.| 4.量词

案例：

1. 匹配座机号

var regexp = /^(0\d{2}-\d{8})|(0\d{3}-\d{7})$/;  
var demo = /^0\d{2}-\d{8}$|^0\d{3}-\d{7}$/;

1. 匹配中文

（ /^[\u4e00-\u9fa5]{2,4}$/ ）

3.匹配验证表单（注册QQ）

replace 函数（replace、trim、search）

replace() 方法用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

语法格式：(返回值是新字符串)

需要匹配的对象.replace(正则式/字符串，替换的目标字符)

正则表达式的匹配模式支持的2个标志

g：表示全局模式（global），即模式将被应用于所有字符串而非发现一个而停止

i：表示不区分大小写（ease-insensitive）模式，在确定匹配想时忽略模式与字符串的大小写



封装自己的trim 函数

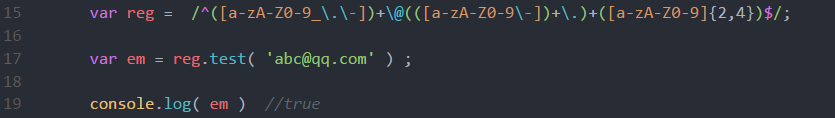
function trim(str) {

return str.replace(/（^\s+）|（\s+$）/g,""); // 去掉前面和后面的空格

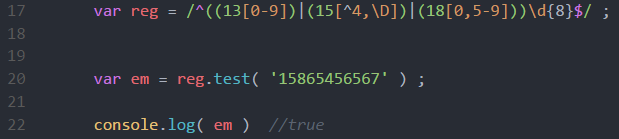
}

3.16 正则匹配

33.1 匹配邮箱



33.1 匹配手机

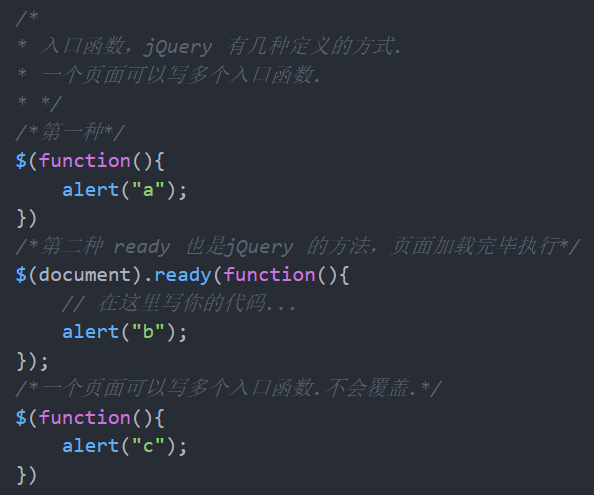


4 jQuery

兼容

jQuery1.12 版本兼容ie6,7,8

入口函数



选择器

id选择器



类选择器



层级选择器

直接子代选择器（儿子）



后代选择器



过滤选择器

获取ul下的第一个元素，并且该元素必须是li



获取奇数li元素



获取偶数li元素



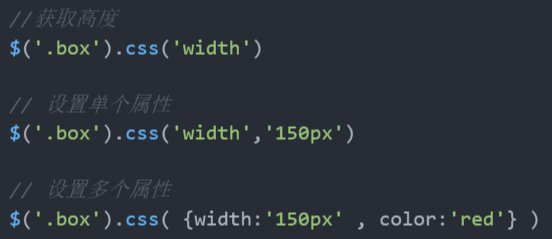
获取所有li元素中的第一个li



属性选择器

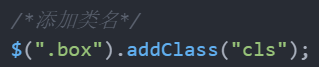


css方法

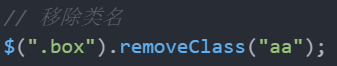


类名操作

添加类名



移除类名



判断是否有该类



有则删除，无则增加

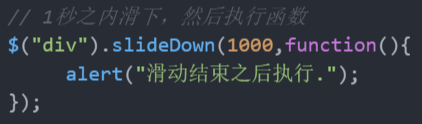


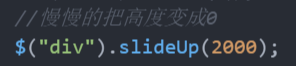
动画方法



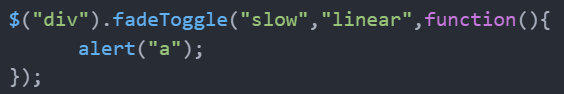


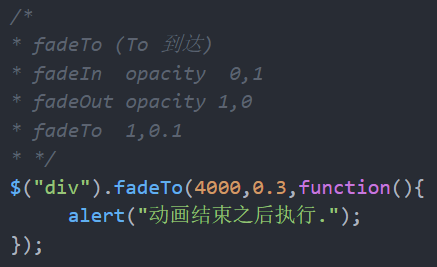














属性设置

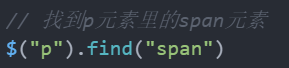






筛选

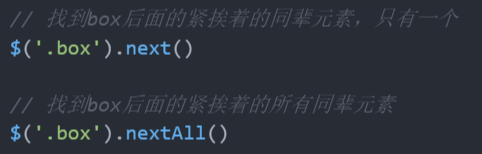
find()



children()



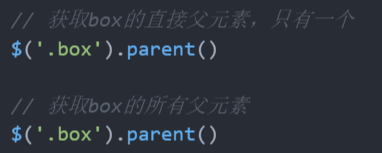
next()/nextAll()



prev()/prevAll()

和next()/nextAll()相同

parent()/parents()



siblings()

获取所有类名为.selected的同辈元素

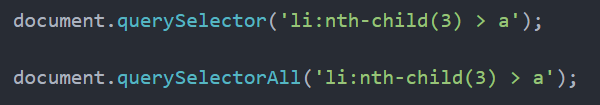


5 HTML5

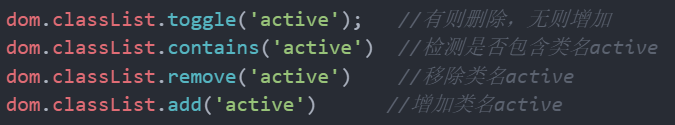
新标签和特性

**1. 新的Doctype**尽管使用<!DOCTYPE html>，即使浏览器不懂这句话也会按照标准模式去渲染   
**2. Figure元素**用<figure>和<figcaption>来语义化地表示带标题的图片   
<figure>   
<img src=”path/to/image” alt=”About image” />   
<figcaption>   
<p>This is an image of something interesting. </p>   
</figcaption>   
</figure>   
  
**4. 去掉link和script标签里面的type属性   
  
7. Email Inputs**如果我们给Input的type设置为email，浏览器就会验证这个输入是否是email类型，当然不能只依赖前端的校验，后端也得有相应的校验   
**8. Placeholders**这个input属性的意义就是不必通过javascript来做placeholder的效果了   
**9. Local Storage**   
使用Local Storage可以永久存储大的数据片段在客户端（除非主动删除），目前大部分浏览器已经支持，在使用之前可以检测一下window.localStorage是否存在   
**10. 语义化的header和footer   
11. 更多的HTML5表单特性**  
**14. Required属性**required属性定义了一个input是否是必须的，你可以像下面这样声明   
<input type=”text” name=”someInput” required>   
或者   
<input type=”text” name=”someInput” required=”required”>   
**15. Autofocus属性**   
正如它的词义，就是聚焦到输入框里面   
<input type=”text” name=”someInput” placeholder=”Douglas Quaid” required autofocus>   
**16. Audio支持**HTML5提供了<audio>标签，你不需要再按照第三方插件来渲染音频，大多数现代浏览器提供了对于HTML5 Audio的支持，不过目前仍旧需要提供一些兼容处理，如   
<audio autoplay=”autoplay” controls=”controls”>   
<source src=”file.ogg” /><!–FF–>   
<source src=”file.mp3″ /><!–Webkit–>   
<a href=”file.mp3″>Download this file.</a>   
</audio>   
**17. Video支持**   
和Audio很像，<video>标签提供了对于video的支持，由于HTML5文档并没有给video指定一个特定的编码，所以浏 览器去决定要支持哪些编码，导致了很多不一致。Safari和IE支持H.264编码的格式，Firefox和Opera支持Theora和Vorbis 编码的格式，当使用HTML5 video的时候，你必须都提供：   
<video controls preload>   
<source src=”cohagenPhoneCall.ogv” type=”video/ogg; codecs=’vorbis, theora’” />   
<source src=”cohagenPhoneCall.mp4″ type=”video/mp4; ’codecs=’avc1.42E01E, mp4a.40.2′” />   
<p> Your browser is old. <a href=”cohagenPhoneCall.mp4″>Download this video instead.</a> </p>   
</video>   
**18. 预加载视频**preload属性就像它的字面意思那么简单，你需要决定是否需要在页面加载的时候去预加载视频   
<video preload>   
**19. 显示视频控制**<video preload controls>   
**20. 正则表达式**由于pattern属性，我们可以在你的markup里面直接使用正则表达式了   
<form action=”" method=”post”>   
<label for=”username”>Create a Username: </label>   
<input type=”text” name=”username” id=”username” placeholder=”4 <> 10″ pattern=”[A-Za-z]{4,10}” autofocus required>   
<button type=”submit”>Go </button>   
</form>   
**21. 检测属性支持**   
除了Modernizr之外我们还可以通过javascript简单地检测一些属性是否支持，如：   
<script>   
if (!’pattern’ in document.createElement(’input’) ) {   
// do client/server side validation   
}   
</script>   
**22. Mark元素**把<mark>元素看做是高亮的作用，当我选择一段文字的时候，javascript对于HTML的markup效果应该是这样的：   
<h3> Search Results </h3>   
<p> They were interrupted, just after Quato said, <mark>”Open your Mind”</mark>. </p>   
**23. 什么时候用<div>**HTML5已经引入了这么多元素，那么div我们还要用吗？div你可以在没有更好的元素的时候去用。   
**24. 想立即使用HTML5?**不要等2022了，现在就可以使用了，just do it.   
**25. 哪些不是HTML5**1)SVG   
2)CSS3   
3)Geolocation   
4)Client Storage   
5)Web Sockets   
**26. Data属性**<div id=”myDiv” data-custom-attr=”My Value”> Bla Bla </div>   
CSS中使用：   
<style>   
h1:hover:after {   
content: attr(data-hover-response);   
color: black;   
position: absolute;   
left: 0;   
}   
</style>   
<h1 data-hover-response=”I Said Don’t Touch Me!”> Don’t Touch Me </h1>   
**27. Output元素**<output>元素用来显示计算结果，也有一个和label一样的for属性   
**28. 用Range Input来创建滑块**HTML5引用的range类型可以创建滑块，它接受min, max, step和value属性   
可以使用css的:before和:after来显示min和max的值   
<input type=”range” name=”range” min=”0″ max=”10″ step=”1″ value=”">   
input[type=range]:before { content: attr(min); padding-right: 5px;   
}   
input[type=range]:after { content: attr(max); padding-left: 5px;}

获取元素的方法



类名操作



自定义属性

在HTML5中我们可以自定义属性，其格式如下data-\*=""，例如

data-info="我是自定义属性"，通过Node.dataset['info'] 我们便可以获取到自定义的属性值。

Node.dataset是以类数组形式存在的

当我们如下格式设置时，则需要以驼峰格式才能正确获取

data-my-name="itcast"，获取Node.dataset['myName']

6 css3

兼容写法



选择器

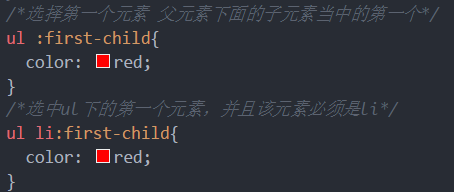
属性选择器



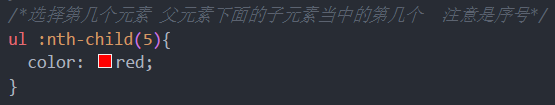


伪类选择器

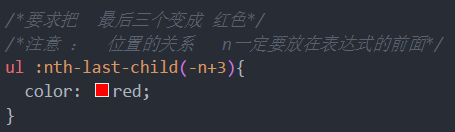
:first-child



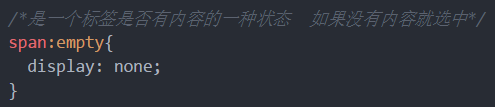
:nth-child()





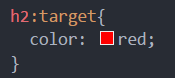


:empty



:target

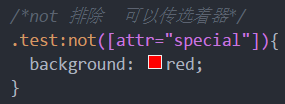
h2被选中之后，字体变为红色







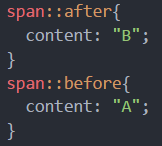
:not



::before/::after

伪元素选择器

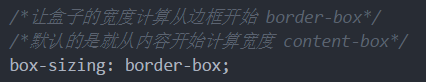
在span的前面和后面各添加一个元素，内容分别为A和B



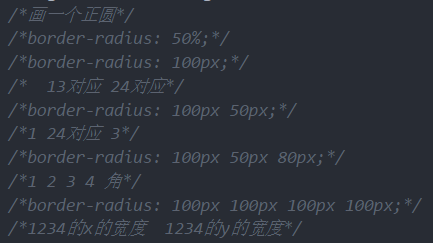
:selection



box-sizing



border-radius

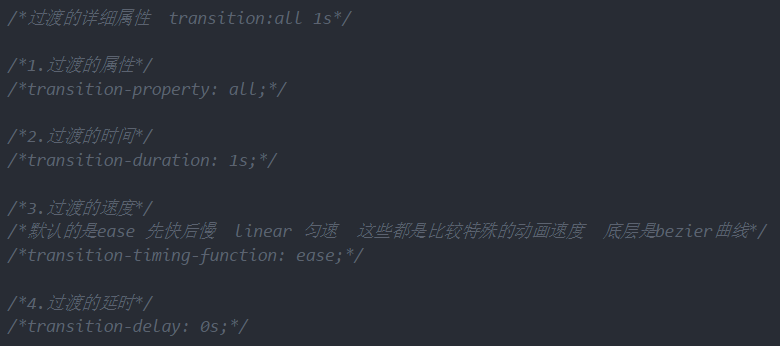


box-shadow



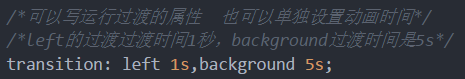


过渡





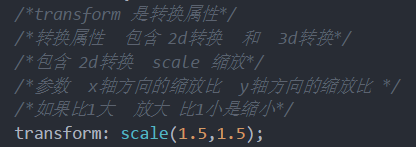




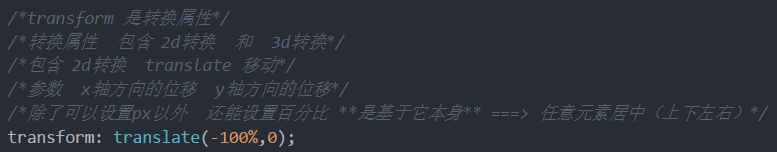
transform转换



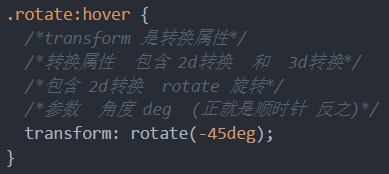
缩放scale



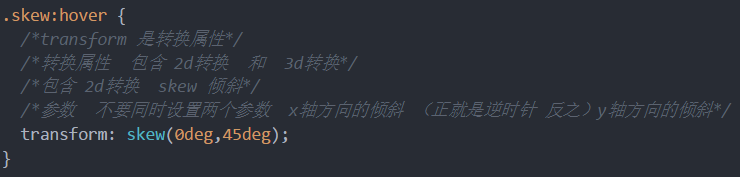
移动translate



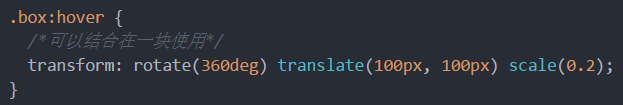
旋转rotate



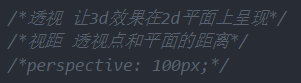
倾斜skew

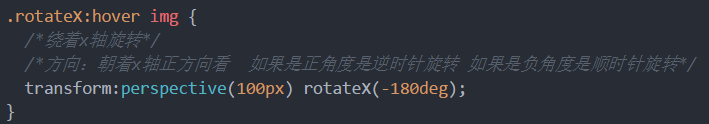


综合使用



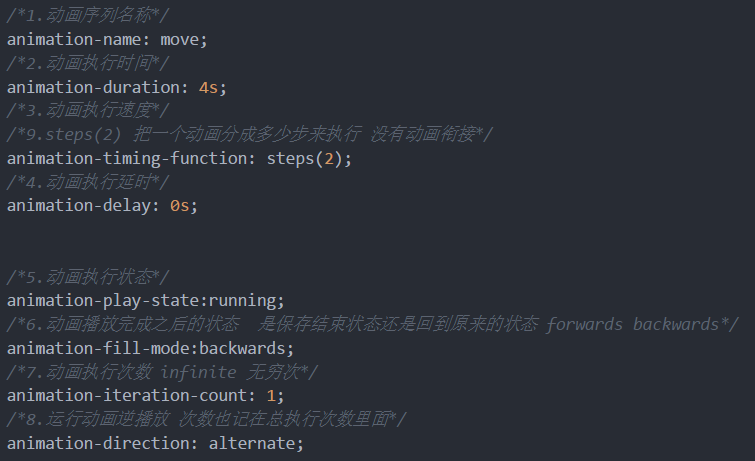
3D转换





@keyframes动画

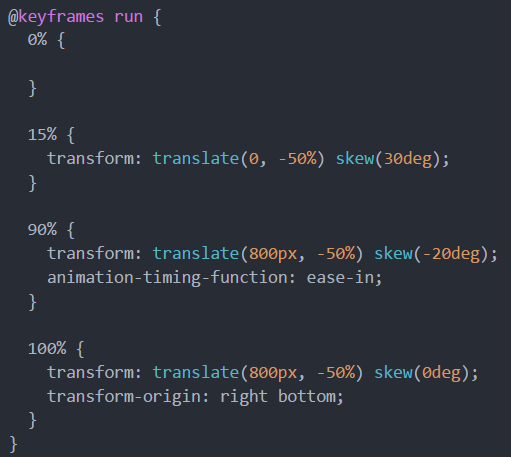
参数详解



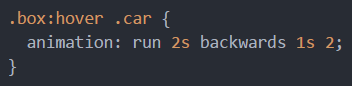
停止动画



定义一个run动画

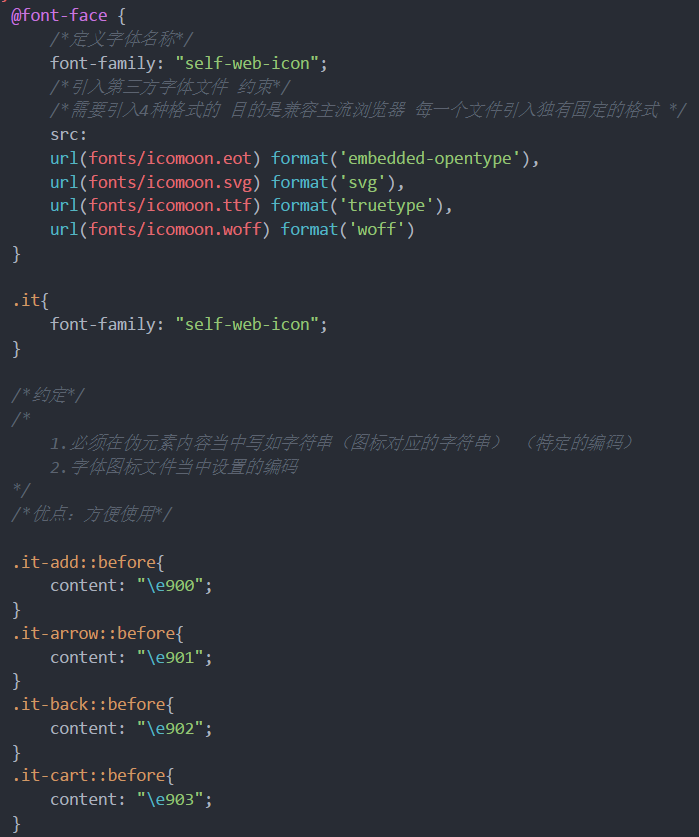


使用动画



字体图标

定义字体图标



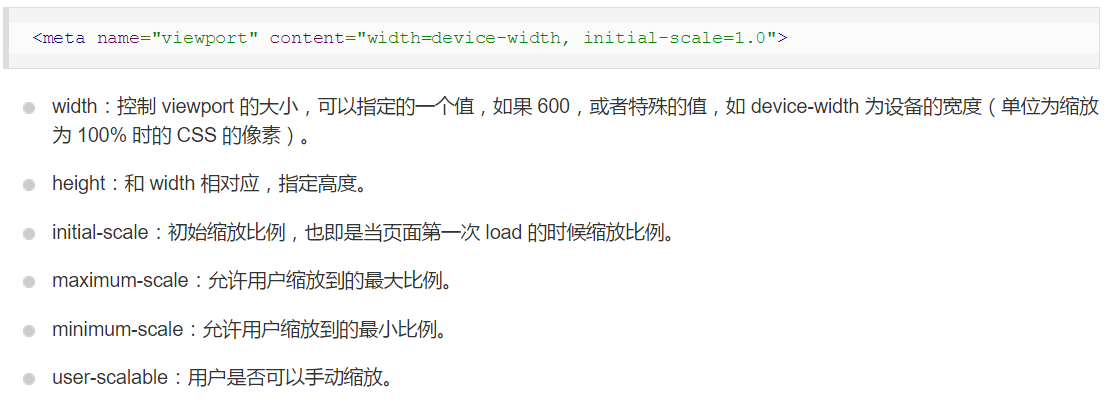
使用字体图标



7 移动web

7.1 视口viewport

手机浏览器是把页面放在一个虚拟的"窗口"（viewport）中，通常这个虚拟的"窗口"（viewport）比屏幕宽，这样就不用把每个网页挤到很小的窗口中（这样会破坏没有针对手机浏览器优化的网页的布局），用户可以通过平移和缩放来看网页的不同部分。



### 7.2 ios系统中元素被触摸时产生的半透明灰色遮罩怎么去掉

ios用户点击一个链接，会出现一个半透明灰色遮罩, 如果想要禁用，可设置-webkit-tap-highlight-color的alpha值为0，也就是属性值的最后一位设置为0就可以去除半透明灰色遮罩。

a,button,input,textarea{-webkit-tap-highlight-color: rgba(0,0,0,0;)}

### 7.3 webkit表单元素的默认外观怎么重置

.css{-webkit-appearance:none;}

### 7.4 webkit表单输入框placeholder的颜色值能改变么

input::-webkit-input-placeholder{ color:#AAAAAA; }

input:focus::-webkit-input-placeholder{ color:#E }

### 7.5 禁止文本缩放

当移动设备横竖屏切换时，文本的大小会重新计算，进行相应的缩放，当我们不需要这种情况时，可以选择禁止：

html {

　　 -webkit-text-size-adjust: 100%;

}

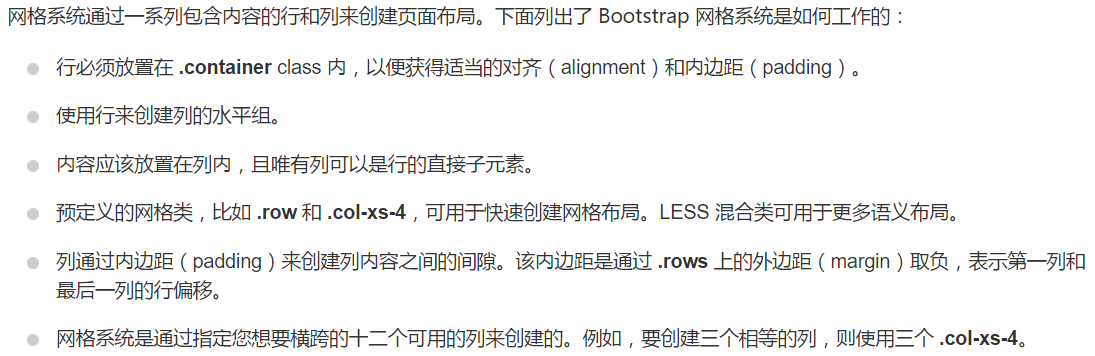
需要注意的是，PC端的该属性已经被移除，该属性在移动端要生效，必须设置 meta viewport。

8 bootstrap

网格系统是什么

Bootstrap 包含了一个响应式的、移动设备优先的、不固定的网格系统，可以随着设备或视口大小的增加而适当地扩展到 12 列。它包含了用于简单的布局选项的预定义类，也包含了用于生成更多语义布局的功能强大的混合类

网格系统怎么使用







响应式工具类



静态/自适应/流式/响应式布局

静态布局（Static Layout）

即传统Web设计，对于PC设计一个Layout，在屏幕宽高有调整时，使用横向和竖向的滚动条来查阅被遮掩部分；

对于移动设备，单独设计一个布局，使用不同的域名如wap.或m.。

自适应布局（Adaptive Layout）

自适应布局（Adaptive）的特点是分别为不同的屏幕分辨率定义布局。布局切换时页面元素发生改变，但在每个布局中，页面元素不随窗口大小的调整发生变化。  
你可以把自适应布局看作是静态布局的一个系列。

流式布局（Liquid Layout）

流式布局（Liquid）的特点（也叫"Fluid") 是页面元素的宽度按照屏幕进行适配调整，主要的问题是如果屏幕尺度跨度太大，

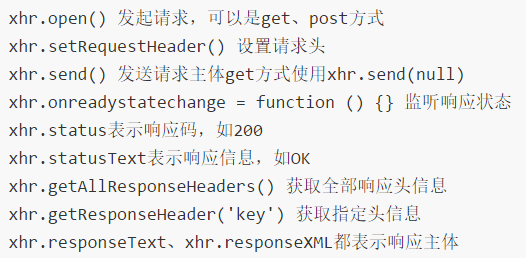
那么在相对其原始设计而言过小或过大的屏幕上不能正常显示。

响应式布局（Responsive Layout）

分别为不同的屏幕分辨率定义布局，同时，在每个布局中，应用流式布局的理念，即页面元素宽度随着窗口调整而自动适配。  
可以把响应式布局看作是流式布局和自适应布局设计理念的融合。

9 ajax

原生ajax







xhr.readyState状态码

\*\*readyState\*\*

0：请求未初始化（还没有调用 open()）。

1：请求已经建立，但是还没有发送（还没有调用 send()）。

2：请求已发送，正在处理中（通常现在可以从响应中获取内容头）。

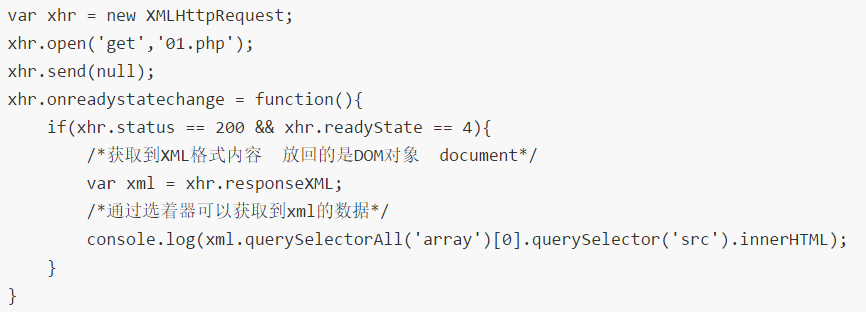
3：请求在处理中；通常响应中已有部分数据可用了，但是服务器还没有完成响应的生成。

4：响应已完成；您可以获取并使用服务器的响应了。

xhr.status

常见的有200代表成功、304文档未修改、403没有权限、404未找到、500服务器错误

返回的数据是xml



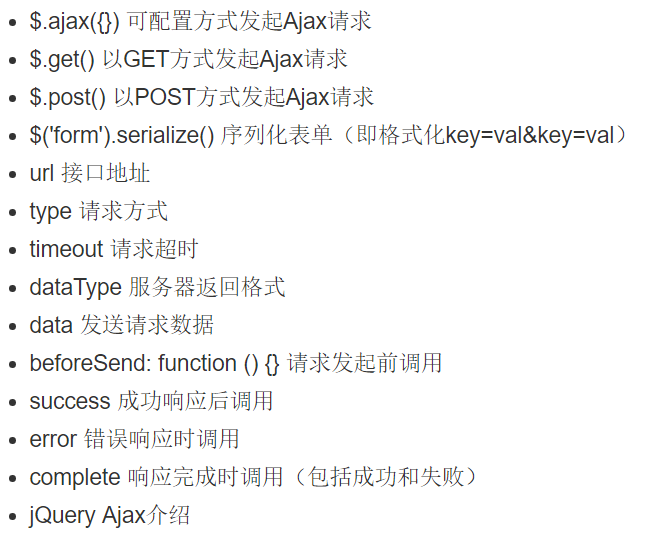
返回的数据是json



兼容IE



jQuery中的$.ajax()

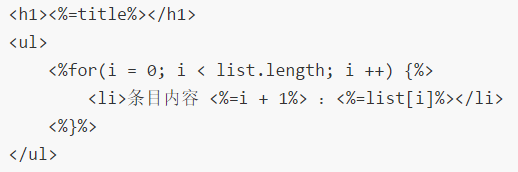


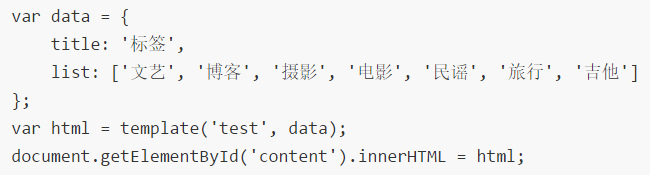
script src = 'www.baidu.com/index.data.php?callback=fn'

var xhr = new XMLhTTP

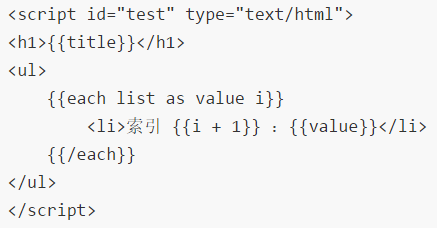
artTemplate模板

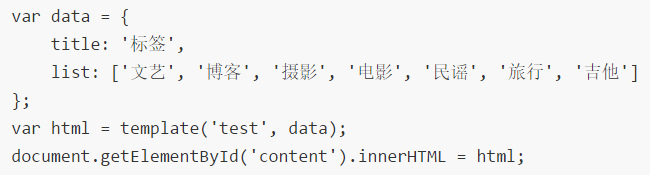
原生语法





简洁语法





同源策略

同源策略是浏览器的一种安全策略，所谓同源是指，域名，协议，端口完全相同。

三个任意一个不同都是跨域

解决跨域JSONP

jsonp跨域原理

其本质是利用了标签具有可跨域的特性， 由服务端返回一个预先定义好的Javascript函数的调用，并且将服务器数据以该函数参数的形式传递过来， 此方法需要前后端配合完成。

jsonp跨域实现



10 canvas

获取绘图上下文



绘制直线



beginPath开启新状态



填充一个矩形



11 面向对象

什么是面向对象

**面向对象是一种编程思想**，又叫**OOP**(Object Oriented Programming)

举个生活中的例子：苹果公司的iphone，并不是苹果公司生产每一个零件，cpu由三星生产，富士康公司来组装，其中三星和富士康可以视为一个对象

面向对象的特点

封装，继承，多态

封装

用函数将一个功能进行封装，封装的目的在于把对象的设计者和对象的使用者分开，使用者不必知晓行为实现的细节，只须用设计者提供的消息来访问该对象

继承

继承性是子类自动共享父类之间数据和方法的机制。

JavaScript中是通过原型链实现继承的

多态

JavaScript本身就是动态语言，所以天然支持多态

原型

|  |
| --- |
| \_\_prtoto\_\_ 谁将我new出来了我就指向谁的原型  function fn(){ }  系统会自动创建一个fn.prototype = { }  prototype是fn的一个属性，所以prototype是fn的原型属性  又由于prototype是一个对象，所以该对象叫做原型对象  原型属性和原型对象是同一个东西，只是相对角度不同所以叫法不同  fn.\_\_proto\_\_ 指向创建出fn的函数  var fn = new Function() ; |

构造函数

|  |
| --- |
| 构造函数和普通函数没有区别，只是使用不同所以叫法不同  function fn(){}  fn() //函数  var f = new fn() //构造函数  构造函数继承  function fn( nme , age ){  this.name = nme  this.age = age  this.run = function(){ console.log('百米冲刺！') }  }  var f = new fn( 'tom' , 18 ) ; //f是什么 fn又是什么  f.name = ?  f.age = ?  f.run = ?  系统开辟了三个空间用来存储name , age , run  为什么f有name属性和age属性，因为继承了fn的name和age,这叫做构造函数继承  var f2 = new fn( 'jack' , 19 ) ;  同理系统开辟了三个空间用来存储name , age , run  构造函数的弊端：浪费内存，相同方法随着new多次，也会创建多次 |

经典继承

|  |
| --- |
| function create( baseObj ) {  function F() {}  F.prototype = baseObj;  return new F();  }  var fn = new fnc()  var f = create(fn)  f.name = 'gom' |

原型继承

|  |
| --- |
| function fn (){ }  fn.prototype.run = function(){ '百米冲刺！' }  fn.prototype.name = 'gom'  fn.prototype.age = 100  var f = new fn()  var f2 = new fn()  整个过程中fun 方法指在内存中开辟一次存储空间 |

混合继承 构造函数继承+原型继承

|  |
| --- |
| function fn( nme ){ this.name = nme }  fn.prototype.run = function(){ console.log( this.name + '快跑') }  var f = fn( 'tom' ) ;  var f2 = fn( 'jack' ) ;  f的name继承自构造函数fn  f的run方法继承自fn的原型属性prototype  所以f的属性继承自两个地方 ， 一个是构造函数，一个是原型，这就叫混合继承， |



Function = new Function()

var obj = {}

var obj = new Object() ;

Object = new Function()

计算机生成了可选文字:
t 的 实 例 
"ZnewFooo 
FOO 的 实 例 

深浅拷贝

|  |
| --- |
| 深拷贝  var num = 100 ;  var num2 = num ;  浅拷贝  var obj = { country:'usa' , city:{ home:'uk' } }  var obj2 = obj  深拷贝  obj2.country = obj.country  obj2.city.home = obj.city.home |

12 angular

angular是什么

AngularJS 是一个 JavaScript 框架。它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 [指令](http://www.runoob.com/angularjs/angularjs-directives.html) 扩展了 HTML，且通过 [表达式](http://www.runoob.com/angularjs/angularjs-expressions.html) 绑定数据到 HTML。

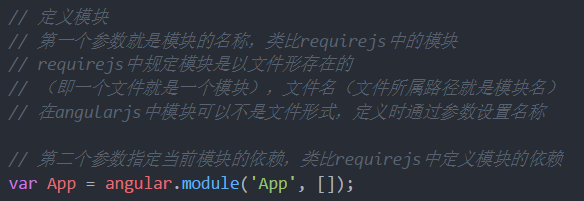
一种构建动态Web应用的结构化框架

AngularJS是为了克服HTML在构建应用上的不足而设计的

angularjs是一个MVC框架

MVC：（Model-模型 View-视图 Controller-控制器）一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码

定义模块



双向数据绑定

双向数据绑定是 AngularJS 的核心机制之一。当 view 中有任何数据变化时，会更新到 model ，当 model 中数据有变化时，view 也会同步更新，显然，这需要一个监控。

原理就是，Angular 在 scope 模型上设置了一个 监听队列，用来监听数据变化并更新 view 。每次绑定一个东西到 view 上时 AngularJS 就会往 $watch 队列里插入一条 $watch ，用来检测它监视的 model 里是否有变化的东西。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，遍历所有的 $watch ，最后更新 dom。

脏检查机制

ng只有在指定事件触发后，才进入$digest cycle：

* DOM事件，譬如用户输入文本，点击按钮等。(ng-click)
* XHR响应事件 ($http)
* 浏览器Location变更事件 ($location)
* Timer事件($timeout, $interval)
* 执行$digest()或$apply()

$watch和$digest是相辅相成的。两者一起，构成了Angular作用域的核心：数据变化的响应。

使用$watch，可以在Scope上添加一个监听器。当Scope上发生变更时，监听器会收到提示。给$watch指定如下两个函数，就可以创建一个监听器：

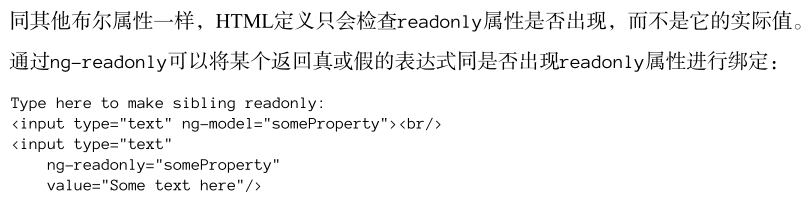
* 一个监控函数，用于指定所关注的那部分数据。
* 一个监听函数，用于在数据变更的时候接受提示。

属性指令

ng-disabled



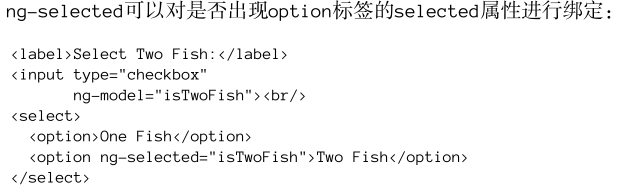
ng-readonly



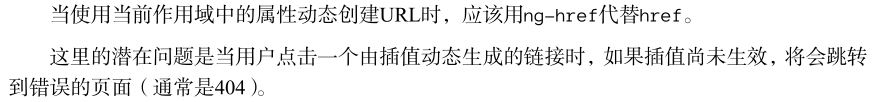
ng-checked



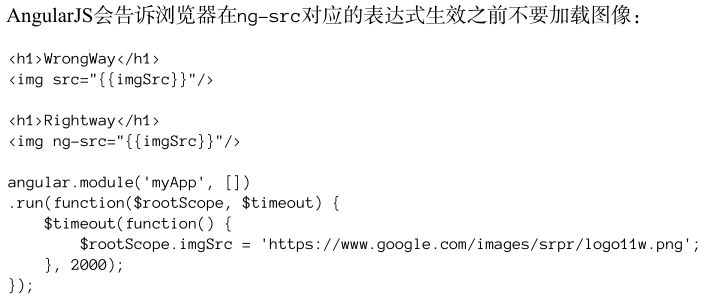
ng-selected



ng-href



ng-src



自定义指令

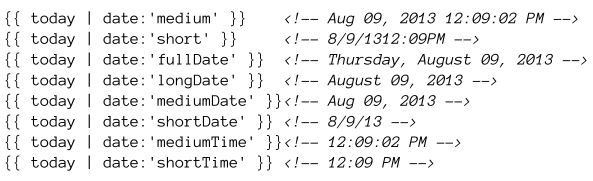


内置过滤器

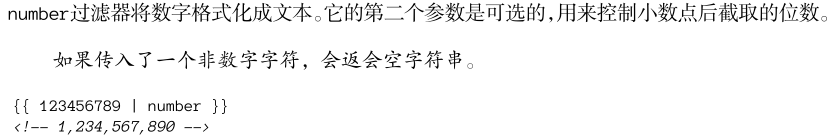
currency



date

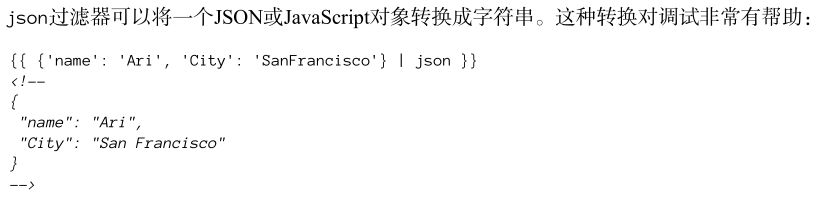


number



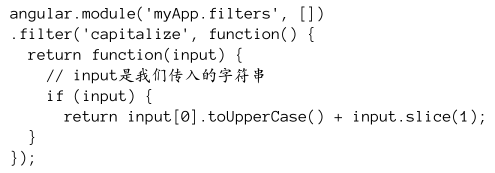


json



自定义过滤器





$location服务



面试点

面试之前：

1-简历上的知识点都能说出来；

2-学过的知识点要大概知道有哪些内容能描述出来；

3-项目能说出你找的项目用到那些知识点你怎么做的就可以不要写代码把学到的知识点都强行加到项目中；

自我介绍：用写在纸上，记住

不要写代码，多记知识点，面试气场，

闭包，

(function(){ })()

原型链，

angular，

this ,

call/apply ,

上家公司的人认不认识，炸你而已，

css脱离标准流的属性，

未知宽高如何让盒子居中，

1. 定位，left:50%, top:50% , marign-left:负一般，margin-top:负一半

2. 对于未知宽高，translateX(-50%,-50%)

VUE一个星期能不能上手，

人事试探不要上当态度要坚定

jQuery和angular的区别

jQuery注重dom操作，angular注重数据展示，两者没有可比性，设计目的不一样

jQuery是库，angular是框架（框架更重量级提供整套解决方案，库只是封装了一些方法）

解释一下原型链

继承的几种方法

创建对象的方法

http协议

http概念

http（超文本传输协议）是一个基于请求与响应模式的、无状态的、应用层的协议，常基于TCP的连接方式，HTTP1.1版本中给出一种持续连接的机制，绝大多数的Web开发，都是构建在HTTP协议之上的Web应用。

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。HTTP协议的主要特点可概括如下：  
1.支持客户/服务器模式。  
2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。  
3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。  
4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。  
5.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

URL

http://host[":"port][abs\_path]  
http表示要通过HTTP协议来定位网络资源；host表示合法的Internet主机域名或者IP地址；port指定一个端口号，为空则使用缺省端口80；abs\_path指定请求资源的URI；如果URL中没有给出abs\_path，那么当它作为请求URI时，必须以“/”的形式给出，通常这个工作浏览器自动帮我们完成。

请求

http请求由三部分组成，分别是：请求行、消息报头、请求正文

1、请求行以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如下：Method Request-URI HTTP-Version CRLF    
其中 Method表示请求方法；Request-URI是一个统一资源标识符；HTTP-Version表示请求的HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。

请求方法（所有方法全为大写）有多种，各个方法的解释如下：  
GET     请求获取Request-URI所标识的资源  
POST    在Request-URI所标识的资源后附加新的数据  
HEAD    请求获取由Request-URI所标识的资源的响应消息报头  
PUT     请求服务器存储一个资源，并用Request-URI作为其标识  
DELETE  请求服务器删除Request-URI所标识的资源  
TRACE   请求服务器回送收到的请求信息，主要用于测试或诊断  
CONNECT 保留将来使用  
OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求  
应用举例：  
GET方法：在浏览器的地址栏中输入网址的方式访问网页时，浏览器采用GET方法向服务器获取资源，eg:GET /form.html HTTP/1.1 (CRLF)

POST方法要求被请求服务器接受附在请求后面的数据，常用于提交表单。

响应

    在接收和解释请求消息后，服务器返回一个HTTP响应消息。

HTTP响应也是由三个部分组成，分别是：状态行、消息报头、响应正文  
1、状态行格式如下：  
HTTP-Version Status-Code Reason-Phrase CRLF  
其中，HTTP-Version表示服务器HTTP协议的版本；Status-Code表示服务器发回的响应状态代码；Reason-Phrase表示状态代码的文本描述。  
状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：  
1xx：指示信息--表示请求已接收，继续处理  
2xx：成功--表示请求已被成功接收、理解、接受  
3xx：重定向--要完成请求必须进行更进一步的操作  
4xx：客户端错误--请求有语法错误或请求无法实现  
5xx：服务器端错误--服务器未能实现合法的请求  
常见状态代码、状态描述、说明：  
200 OK      //客户端请求成功  
400 Bad Request  //客户端请求有语法错误，不能被服务器所理解  
401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用   
403 Forbidden  //服务器收到请求，但是拒绝提供服务  
404 Not Found  //请求资源不存在，eg：输入了错误的URL  
500 Internal Server Error //服务器发生不可预期的错误  
503 Server Unavailable  //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

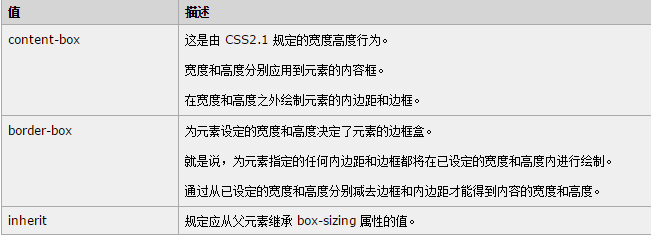
# 面试试卷

试卷一

1 隐藏元素的几种方法

|  |
| --- |
| { display: none; /\* 不占据空间，无法点击 \*/ }  { visibility: hidden; /\* 占据空间，无法点击 \*/ }  { position: absolute; top: -999em; /\* 不占据空间，无法点击 \*/ }  { position: relative; top: -999em; /\* 占据空间，无法点击 \*/ }  { position: absolute; visibility: hidden; /\* 不占据空间，无法点击 \*/ }  { height: 0; overflow: hidden; /\* 不占据空间，无法点击 \*/ }  { opacity: 0; filter:Alpha(opacity=0); /\* 占据空间，可以点击 \*/ } |

2 box-sizing常用属性



3 让一个元素垂直居中的方法

|  |
| --- |
| 第一种方法:  div.box{  weight:200px;  height:400px;  <!--把元素变成定位元素-->  position:absolute;  <!--设置元素的定位位置，距离上、左都为50%-->  left:50%;  top:50%;  <!--设置元素的左外边距、上外边距为宽高的负1/2-->  margin-left:-100px;  margin-top:-200px;  }  \*兼容性好;缺点:必须知道元素的宽高  -------------  第二种方法：  div.box{  weight:200px;  height:400px;  <!--把元素变成定位元素-->  position:absolute;  <!--设置元素的定位位置，距离上、左都为50%-->  left:50%;  top:50%;  <!--设置元素的相对于自身的偏移度为负50%(也就是元素自身尺寸的一半)-->  transform:translate(-50%,-50%);  }    \*这是css3里的样式;缺点:兼容性不好，只支持IE9+的浏览器    ---------------  第三种方法  div.box{  weight:200px;  height:400px;  <!--把元素变成定位元素-->  position:absolute;  <!--设置元素的定位位置，距离上、下、左、右都为0-->  left:0;  right:0;  top:0;  bottom:0;  <!--设置元素的margin样式值为 auto-->  margin:auto;  }  weq |

4 Doctype的作用，html5为什么要写<!DOCTYPE HTML>

<!DOCTYPE> 声明不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。

在 HTML 4.01 中，<!DOCTYPE> 声明引用 DTD，因为 HTML 4.01 基于 SGML。DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。

HTML5 不基于 SGML，所以不需要引用 DTD。

