牛客网刷题笔记

3.26：

background-attachment定义背景图片随滚动轴的移动方式

有三个值，scroll是默认值，背景图像会随着页面其余部分的滚动而移动。

fixed当页面的其余部分滚动时，背景图像不会移动。

inherit规定应该从父元素继承 background-attachment 属性的设置。

background-origin：属性规定 background-position 属性相对于什么位置来定位。

background-clip：属性规定背景的绘制区域。

常用的块状元素有：设置display:block就是将元素显示为块级元素。

、<p>、<h1>…<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>

块级元素特点：

1、每个块级元素都从新的一行开始，并且其后的元素也另起一行。

2、元素的高度、宽度、行高以及顶和底边距都可设置。

3、元素宽度在不设置的情况下，是它本身父容器的100%，除非设定一个宽度。

常用的内联元素有： 当然块状元素也可以通过代码display:inline将元素设置为内联元素

<a>、、
、<i>、、、<label>、<q>、<var>、<cite>、<code> 在html中，、<a>、<label>、 和就是典型的内联元素（行内元素）（inline）元素。

内联元素特点：

1、和其他元素都在一行上；

2、元素的高度、宽度及顶部和底部边距不可设置；

3、元素的宽度就是它包含的文字或图片的宽度，不可改变。

常用的内联块状元素有：<img>、<input>

内联块状元素（inline-block）就是同时具备内联元素、块状元素的特点，代码display:inline-block就是将元素设置为内联块状元素。

inline-block 元素特点：

1、和其他元素都在一行上；

2、元素的高度、宽度、行高以及顶和底边距都可设置。

Bootstrap 框架的网格系统工作原理如下：

1 、数据行 (.row) 必须包含在容器（ .container ）中，以便为其赋予合适的对齐方式和内距 (padding) 。

如：

2 、在行 (.row) 中可以添加列 (.column) ，但列数之和不能超过平分的总列数，比如 12 。

如：

3 、具体内容应当放置在列容器（ column ）之内，而且只有列（ column ）才可以作为行容器 (.row) 的直接子元素

4 、通过设置内距（ padding ）从而创建列与列之间的间距。然后通过为第一列和最后一列设置负值的外距（ margin ）来抵消内距 (padding) 的影响

Bootstrap 3的兼容性问题：

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	✔ 支持	✔ 支持		✖ 不支持	N/A
iOS	✔ 支持	N/A	N/A	✖ 不支持	✔ 支持
Mac OS X	✔ 支持	✔ 支持		✔ 支持	✔ 支持
Windows	✔ 支持	✔ 支持	✔ 支持	✔ 支持	✖ 不支持

页面中使用CSS的方式主要有3种：行内添加定义style属性值，页面头部内嵌调用和外面链接调用，其中外面引用有两种：link和@import。外部引用CSS两种方式link和@import的方式分别是：

XML/HTML代码

<link rel="stylesheet" rev="stylesheet" href="CSS文件" type="text/css" media="all" />

XML/HTML代码

<style type="text/css" media="screen">

@import url("CSS文件");

</style>

两者都是外部引用CSS的方式，但是存在一定的区别：

区别1：link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。

区别2：link引用CSS时，在页面载入时同时加载；@import需要页面网页完全载入以后加载。

区别3：link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。

区别4：ink支持使用Javascript控制DOM去改变样式；而@import不支持。

补充：@import最优写法

@import的写法一般有下列几种：

@import 'style.css' //Windows IE4/ NS4, Mac OS X IE5, Macintosh IE4/IE5/NS4不识别

@import "style.css" //Windows IE4/ NS4, Macintosh IE4/NS4不识别

@import url(style.css) //Windows NS4, Macintosh NS4不识别

@import url('style.css') //Windows NS4, Mac OS X IE5, Macintosh IE4/IE5/NS4不识别

@import url("style.css") //Windows NS4, Macintosh NS4不识别

由上分析知道，@import url(style.css) 和@import url("style.css")是最优的选择，兼容的浏览器最多。从字节优化的角度来看@import url(style.css)最值得推荐。

3.27：

CSS选择器如下：

1. 标签名选择器　div { color：Red;}　/即页面中的各个标签名的css样式

2.类选择器　.divClass {color:Red;}/即定义的每个标签的class 中的css样式

3.ID选择器　#myDiv {color:Red;}　/即页面中的标签的id

4.后代选择器（类选择器的后代选择器）　.divClass span { color:Red;}

5.群组选择器　div,span,img {color:Red}　/即具有相同样式的标签分组显示

选择器的优先级 1.最高优先级是 (直接在标签中的设置样式，假设级别为1000)

2.次优先级是（ID选择器 ,假设级别为100）　#myDiv{color:Red;}

3.其次优先级是（类选择器，假设级别为10）　.divClass{color:Red;}

4.最后优先级是 （标签选择器，假设级别是 1）　div{color:Red;}

5.那么后代选择器的优先级.divClass span { color:Red;}　优先级别就是：10+1=11

js延迟加载的方式：

defer 属性 defer async属性只适用于外部脚本文件 仅IE浏览器支持

async 属性 不让页面等待脚本下载和执行，从而异步加载页面其他内容。

动态创建DOM方式　使用jQuery的getScript方法

使用setTimeout延迟方法　让JS最后加载把js外部引入的文件放到页面底部

localstorage的使用

1.获取对象 [javascript] view plain copy var localstroage = window.localStorage;

2.存储值 [javascript] view plain copy localstroage.setItem('openid','ksjhfkjajkdfjkszdjfksad'); localstroage.setItem('uid','10001');

3.获取值 [javascript] view plain copy var openid = localstroage.getItem('openid');

4.删除某个值 [javascript] view plain copy localstroage.removeItem('openid');

5.删除所有值 [javascript] view plain copy localstroage.clear();

6.遍历所有键值 [javascript] view plain copy localStorage.uid = 1; localStorage.sex='nan'; localStorage.age = 23; for (var i = 0, len = localStorage.length; i < len; ++i) { console.log(localStorage.key(i) +':' + localStorage.getItem(localStorage.key(i))); }

写入字段有三种方式：localStorage["a"]=1;localStorage.b=1;localStorage.setItem("c",3);

读取字段也有三种方式：var a= localStorage.a;var b= localStorage["b"];

var c= localStorage.getItem("c");

注意：1. 如果不主动删除 localstorage 是不会自动清空的 不会像cookie一样清除 2. localstorage 几乎支持大部分浏览器 不用考虑IE8 的问题 大小为 5M

ES6 function*声明 (function关键字后跟一个星号）定义了一个 *生成器函数* (*generator function*)，它返回一个 Generator 对象。

也可以定义*生成器函数* 使用构造函数 GeneratorFunction 和一个 function* expression

JavaScript RegExp 对象有 3 个方法：test()、exec() 和 compile()。

(1) test() 方法用来检测一个字符串是否匹配某个正则表达式，如果匹配成功，返回 true ，否则返回 false；

(2) exec() 方法用来检索字符串中与正则表达式匹配的值。exec() 方法返回一个数组，其中存放匹配的结果。如果未找到匹配的值，则返回 null;

（3）compile() 方法可以在脚本执行过程中编译正则表达式，也可以改变已有表达式。

比较Ajax与Flash的优缺点

1.Ajax的优势：1.可搜索性 2.开放性 3.费用 4.易用性 5.易于开发。

2.Flash的优势：1.多媒体处理 2.兼容性 3.矢量图形 4.客户端资源调度

3.Ajax的劣势：1.它可能破坏浏览器的后退功能 2.使用动态页面更新使得用户难于将某个特定的状态保存到收藏夹中，不过这些都有相关方法解决。

4.Flash的劣势：1.二进制格式 2.格式私有 3.flash 文件经常会很大，用户第一次使用的时候需要忍耐较长的等待时间 4.性能问题

1操作符（1）typeof操作符

格式：result=typeof variable

返回值：undefined boolean string number object 、null function

（2）instanceof操作符

格式：result=variable instanceof constructor

返回值：true false

通常来说判断一个对象的类型使用typeof，但是在new String的情况下的结果会是object

数据类型	转换为true的值	转换为false的值
Boolean	true	false
String	任何非空字符串	空字符串
Number	任何非零数字值（包括无穷大）	0和Null
Object	任何对象	null
Undefined		Undefined

在JavaScript中下面的值被当作假 false. undefined. null. 空字符串 数字0 数字NAN

var name;只定义没有赋值 就是undefined

alert(undefined==null) true ECMAScript认为undefined是从null派生出来的

alert(undefined===null) false

“==”运算符（两个操作数的类型不相同时）

- 如果一个值是null，另一个值是undefined，则它们相等

- 如果一个值是数字，另一个值是字符串，先将字符串转换为数学，然后使用转换后的值进行比较。
- 如果其中一个值是true，则将其转换为1再进行比较。如果其中的一个值是false，则将其转换为0再进行比较。
- 如果一个值是对象，另一个值是数字或字符串，则将对象转换为原始值，再进行比较。

对象到数字的转换

- 如果对象具有valueOf()方法，后者返回一个原始值，则JavaScript将这个原始值转换为数字（如果需要的话）并返回一个数字。
- 否则，如果对象具有toString()方法，后者返回一个原始值，则JavaScript将其转换并返回。（对象的toString()方法返回一个字符串直接量（作者所说的原始值），JavaScript将这个字符串转换为数字类型，并返回这个数字）。
- 否则，JavaScript抛出一个类型错误异常。

空数组转换为数字0

- 数组继承了默认的valueOf()方法，这个方法返回一个对象而不是一个原始值，因此，数组到数学的转换则调用toString()方法。空数组转换为空字符串，空字符串转换为数字0.

布尔类型里只有这几参数个返回false，其它都为true
Boolean(undefined) Boolean(null) Boolean(0) Boolean(NaN) Boolean('')
布尔类型与其它任何类型进行比较，布尔类型将会转换为number类型。
Boolean([]); //true Number([]); //0
Number({}); // NaN Number(false); //0
console.log(([])?true:fasle);// => console.log((true)?true:false);
console.log([]==false?true:false); // => console.log(0==0?true:false);
console.log(({}==false)?true:false); // => console.log((NaN==0)?true:false);
常见的请求头和相应头都有什么呢
1)请求(客户端->服务端[request])
  GET(请求的方式) /newcoder/hello.html(请求的目标资源) HTTP/1.1(请求采用的协议和版本号)
  Accept: */*(客户端能接收的资源类型)
  Accept-Language: en-us(客户端接收的语言类型)
  Connection: Keep-Alive(维护客户端和服务端的连接关系)
  Host: localhost:8080(连接的目标主机和端口号)
  Referer: http://localhost/links.asp(告诉服务器我来自于哪里)
  User-Agent: Mozilla/4.0(客户端版本号的名字)
  Accept-Encoding: gzip, deflate(客户端能接收的压缩数据的类型)
  If-Modified-Since: Tue, 11 Jul 2000 18:23:51 GMT(缓存时间)
  Cookie(客户端暂存服务端的信息)
  Date: Tue, 11 Jul 2000 18:23:51 GMT(客户端请求服务端的时间)
2)响应(服务端->客户端[response])
  HTTP/1.1(响应采用的协议和版本号) 200(状态码) OK(描述信息)
  Location: http://www.baidu.com(服务端需要客户端访问的页面路径)
  Server:apache tomcat(服务端的Web服务端名)
  Content-Encoding: gzip(服务端能够发送压缩编码类型)
  Content-Length: 80(服务端发送的压缩数据的长度)
  Content-Language: zh-cn(服务端发送的语言类型)
  Content-Type: text/html; charset=GB2312(服务端发送的类型及采用的编码方式)
  Last-Modified: Tue, 11 Jul 2000 18:23:51 GMT(服务端对该资源最后修改的时间)
  Refresh: 1;url=http://www.it315.org(服务端要求客户端1秒钟后，刷新，然后访问指定的页面路径)
  Content-Disposition: attachment; filename=aaa.zip(服务端要求客户端下载文件的方式打开文件)

Transfer-Encoding: chunked(分块传递数据到客户端）

Set-Cookie:SS=Q0=5Lb_nQ; path=/search(服务端发送到客户端的暂存数据)

Expires: -1//3种(服务端禁止客户端缓存页面数据)

Cache-Control: no-cache(服务端禁止客户端缓存页面数据)

Pragma: no-cache(服务端禁止客户端缓存页面数据)

Connection: close(1.0)/(1.1)Keep-Alive(维护客户端和服务端的连接关系)

Date: Tue, 11 Jul 2000 18:23:51 GMT(服务端响应客户端的时间)

在服务器响应客户端时，带上Access-Control-Allow-Origin头信息，解决跨域的一种方法

AMD 是 RequireJS 在推广过程中对模块定义的规范化产出。

CMD 是 SeaJS 在推广过程中对模块定义的规范化产出。

commonJS:nodeJS在推广过程中对模块定义的规范化产出。

区别：

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。

2. CMD 推崇依赖就近，AMD 推崇依赖前置。

target的值：_blank 在新窗口打开链接 _self在当前框架中打开链接 _parent在父框架打开链接

_top 在当前窗口打开链接 framename在指定框架打开链接

(F.prototype)[的构造函数] ═══ Object F[的构造函数] ═══ Function

Math对象包含max()方法，用于确认一组数值中的最大值。该方法接收任意多个数值参数，不接受数组参数。

要找到数组中的最值，可以使用apply()方法，D表示将Math.max()方法的执行环境切换到null上，apply()方法接收两个参数，第二个参数是一个数组。

ul 无序列表 ol 有序列表 li 定义列表项目 dl 定义列表

call()方法,第一个参数和apply()一样,是在其中运行的作用域即this;和apply()不同的是,call()方法中的其余的参数必须直接传给函数,即在使用call()方法时参数必须逐个的列出来. this 在函数执行时，this 总是指向调用该函数的对象。要判断 this 的指向，其实就是判断 this 所在的函数属于谁。

要运用css3动画，需要运用@keyframes规则和animation属性

Array 对象方法 方法 concat() 连接两个或更多的数组，并返回结果。 join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。 pop() 删除并返回数组的最后一个元素 push() 向数组的末尾添加一个或更多元素，并返回新的长度。 reverse() 颠倒数组中元素的顺序。 shift() 删除并返回数组的第一个元素 slice() 从某个已有的数组返回选定的元素 sort() 对数组的元素进行排序 splice() 删除元素，并向数组添加新元素。 toSource() 返回该对象的源代码。 toString() 把数组转换为字符串，并返回结果。 toLocaleString() 把数组转换为本地数组，并返回结果。 unshift() 向数组的开头添加一个或更多元素，并返回新的长度。 valueOf() 返回数组对象的原始值

H5新增标签：

article: 标签定义外部的内容。

aside:标签定义 article 以外的内容。

audio:h5新增音频标签。没有高宽属性。

canvas:h5新增画布标签。

command: 定义命令按钮(未测试)

datalist：标签定义选项列表。

datalist 及其选项不会被想显示出来，它仅仅是合法的输入值列表。

details：标签用于描述文档或文档某个部分的细节。

figure：标签用于对元素进行组合。

figcaption：定义 figure 元素的标题。

footer：定义 section 或 document 的页脚。

header：定义 section 或 document 的页眉。

hgroup：用于对网页或区段（section）的标题进行组合。

keygen:标签规定用于表单的密钥对生成器字段

mark：标签定义带有记号的文本。

meter：通过min="0" max="20"的方式定义度量衡。仅用于已知最大和最小值的度量。

nav：定义document或section或article的导航。
output：定义不同的输出类型，比如脚本。
progress：定义任何类型的任务的进度。
rp:定义若浏览器不支持 ruby 元素显示的内容
rt：定义 ruby 注释的解释
ruby：定义 ruby 注释
section：标签定义文档中的节、区段。比如章节、页眉、页脚或文档中的其他部分。
source:audio和video的属性之一。为audio和video定义媒介源。
summary:为details定义标题。
time:定义日期或时间。
video：h5新增视频标签。具有高宽属性。
jQuery siblings() 方法:返回被选元素的所有同胞元素。
jQuery next() 方法:返回被选元素的下一个同胞元素。该方法只返回一个元素。
jQuery find() 方法:返回被选元素的后代元素，一路向下直到最后一个后代。
this有四种情况！
1)当在函数调用的时候指向widow
2)当方法调用的时候指向调用对象
3)当用apply和call上下文调用的时候指向传入的第一个参数
4)构造函数调用指向实例对象
包含在 ^{标签和其结束标签} 中的内容将会以当前文本流中字符高度的一半来显示，但是与当前文本流中文字的字体和字号都是一样的。
上标 下标 <mark></mark>突出显示
3.27:
with try-catch eval可以改变作用域链.while的话只是在函数局部环境全局环境运行，并不改变作用域
在Javascript定义一个函数一般有如下三种方式：
1，函数关键字(function)语句： function fnMethodName(x){alert(x);}
2，函数字面量(Function Literals)： var fnMethodName = function(x){alert(x);}
3，Function()构造函数： var fnMethodName = new Function('x','alert(x);')
ajax的事件是：
ajaxComplete(callback) ajaxError(callback) ajaxSend(callback)
ajaxStart(callback) ajaxStop(callback) ajaxSuccess(callback)
块内函数声明
不要在块内声明一个函数（严格模式会报语法错误）。如果确实需要在块中定义函数，可以使用函数表达式来声明函数。
/* Recommended if (x) { var foo = function() {}}
/*error if (x) { function foo() {}}

- **Margin(外边距)** - 清除边框外的区域，外边距是透明的。
- **Border(边框)** - 围绕在内边距和内容外的边框。
- **Padding(内边距)** - 清除内容周围的区域，内边距是透明的。
- **Content(内容)** - 盒子的内容，显示文本和图像。
- Margin

margin清除周围的元素（外边框）的区域。margin没有背景颜色，是完全透明的

- Padding（填充）

当元素的 Padding（填充）（内边距）被清除时，所"释放"的区域将会受到元素背景颜色的填充。
html语言中加粗的标签有<b>和,或者使用css设置｛font-weight:bold｝
for in遍历对象所拥有的属性(可枚举的) 由于对象和数组不同，不能用下标来访问，只能用for in遍历

CSS Sprites在国内很多人叫css精灵，是一种网页图片应用处理方式。它允许你将一个页面涉及到的所有零星图片都包含到一张大图中去，这样一来，当访问该页面时，载入的图片就不会像以前那样一幅一幅地慢慢显示出来了。
利用CSS的“background-image”，“background- repeat”，“background-position”的组合进行背景定位，background-position可以用数字精确的定位出背景图片的位置。
利用CSS Sprites能很好地减少网页的http请求，从而大大的提高页面的性能，这也是CSS Sprites最大的优点，也是其被广泛传播和应用的主要原因；
CSS Sprites能减少图片的字节，曾经比较过多次3张图片合并成1张图片的字节总是小于这3张图片的字节总和。所以C错误
解决了网页设计师在图片命名上的困扰，只需对一张集合的图片上命名就可以了，不需要对每一个小元素进行命名，从而提高了网页的制作效率。
更换风格方便，只需要在一张或少张图片上修改图片的颜色或样式，整个网页的风格就可以改变。维护起来更加方便。

```
js里面没有函数重载的概念，在其他语言中（如java）java中，可以存在同名函数，
只要传入的参数数量或者类型不同即可。在js中，定义了两个同名函数后，
后面的函数会覆盖前面定义的函数。结合这道题来说，由于函数声明提升，
所以函数声明会提前，由于存在同名函数，后面的add函数将覆盖第一个add函数，
所以两次调用add()返回的值是相同的。也就是y,z都为4.
```

enctype 属性规定在发送到服务器之前应该如何对表单数据进行编码。
属性值:application/x-www-form-urlencoded 在发送前编码所有字符（默认）
multipart/form-data 不对字符编码。 在使用包含文件上传控件的表单时，必须使用该值。
text/plain 空格转换为 "+" 加号，但不对特殊字符编码。
span标签是无法设置宽高的；.float会把浮动元素变成块级元素；.绝对定位脱离了文档流
继承就是指子节点默认使用父节点的样式属性。可以继承的属性很少，只有颜色，文字，字体间距行高对齐方式，和列表的样式可以继承
text-transform:capitalize是首字母大写 text-transfrom:lowercase是全部字母为小写
text-transfrom:uppercase是全部字母为大写 font-weight: bold;字体为粗体，
标准盒子模型 ＝ margin + border + padding + content （content = width | height）
IE盒子模型 ＝ margin + content （content = border + padding + width | height）
ES6 后新增了一类数据类型 ：Symbol，根据 JavaScript 高程, ES5 中的基本数据类型有 5 种：Undefined、Null、Boolean、Number、String.而 Object 是属于复杂数据类型，所以我认为这里说的 6 种基本数据类型是指：Undefined、Null、Boolean、Number、String 与 Symbol.
DOM中的事件对象：（符合W3C标准）
preventDefault()　　取消事件默认行为
stopImmediatePropagation() 取消事件冒泡同时阻止当前节点上的事件处理程序被调用。
stopPropagation()　　取消事件冒泡对当前节点无影响。
IE中的事件对象：
cancelBubble()　　取消事件冒泡　　returnValue()　　取消事件默认行为
Flash提供了ExternalInterface接口与JavaScript通信，ExternalInterface有两个方法，call和addCallback，call的作用是让Flash调用js里的方法，addCallback是用来注册flash函数让js调用。
label标签只有两个属性
for（ 规定 label 绑定到哪个表单元素。 ） form（规定 label 字段所属的一个或多个表单）
置换元素：浏览器根据元素的标签和属性，来决定元素的具体显示内容。
例如：浏览器会根据<img>标签的src属性的 值来读取图片信息并显示出来，而如果查看(x)html代码，则看不到图片的实际内容；<input>标签的type属性来决定是显示输入 框，单选按钮等。 (x)html中 的<img>、<input>、<textarea>、<select>、<object> 都是置换元素。这些元素往往没有实际的内容，即是一个空元素。置换元素在显示中生成了框，这也就是有的内联元素能够设置宽高的原因。
不可替换元素：(x)html 的大多数元素是不可替换元素，即其内容直接表现给用户端。
<label>label中的内容</label> 标签<label>是一个非置换元素，文字label中的内容”将全被显示。
iframe可用在以下几个场景中：
1：典型系统结构，左侧是功能树，右侧就是一些常见的table或者表单之类的。为了每一个功能，单独分离出来，采用iframe。

2：ajax上传文件。

3：加载别的网站内容，例如google广告，网站流量分析。

4： 在上传图片时，不用flash实现无刷新。

5： 跨域访问的时候可以用到iframe，使用iframe请求不同域名下的资源。

3.30：

<i>都表示斜体，如果这种斜体字对该浏览器不可用的话，可以使用高亮、反白或加下划线等样式。区别在于表示强调，<i>单纯表示斜体

特性	Cookie	localStorage	sessionStorage
数据的生命期	一般由服务器生成，可设置失效时间。如果在浏览器端生成Cookie，默认是关闭浏览器后失效	除非被清除，否则永久保存	仅在当前会话下有效，关闭页面或浏览器后被清除
存放数据大小	4K左右		一般为5MB
与服务器端通信	每次都会携带在HTTP头中，如果使用cookie保存过多数据会带来性能问题	仅在客户端（即浏览器）中保存，不参与和服务器的通信	
易用性	需要程序员自己封装，源生的Cookie接口不友好	源生接口可以接受，亦可再次封装来对Object和Array有更好的支持	

position: absolute;生成绝对定位的元素，相对于static 定位以外的第一个父元素进行定位；都绝对定位了，肯定脱离了文档流。。

position: fixed;生成绝对定位的元素，相对于浏览器窗口进行定位

position: relative;生成相对定位的元素，相对于其正常位置进行定位。生成相对定位，也就是说还在原本的上下左右之间，上下左右的元素都不变，so这个没有能脱离文档流。。就这个了

float: left;都浮动出去了，还上哪保持原位置去。

Wekbit是一个开源的Web浏览器引擎，也就是浏览器的内核。Apple的Safari, Google的Chrome, Nokia S60平台的默认浏览器，Apple手机的默认浏览器，Android手机的默认浏览器均采用的Webkit作为器浏览器内核。Webkit的采用程度由 此可见一斑，理所当然的成为了当今主流的三大浏览器内核之一。另外两个分别是Gecko和Trident，大名鼎鼎的Firefox便是使用的Gecko 内核，而微软的IE系列则使用的是Trident内核。

另外，搜狗浏览器是双核的，双核并不是指一个页面由2个内核同时处理,而是所有网页（通常是标准通用标记语言的应用超文本标记语言）由webkit内核处理,只有银行网站用IE内核

白屏时间（first Paint Time）——用户从打开页面开始到页面开始有东西呈现为止

首屏时间——用户浏览器首屏内所有内容都呈现出来所花费的时间

用户可操作时间(dom Interactive)——用户可以进行正常的点击、输入等操作，默认可以统计**domready**时间，因为通常会在这时候绑定事件操作

总下载时间——页面所有资源都加载完成并呈现出来所花的时间，即页面 onload 的时间

发送消息： $scope.$emit(name, data) 或者 $scope.$broadcast(name, data);

接收消息： $scope.on(name,function(event,data){ });

区别： $emit 广播给父controller $broadcast 广播给子controller

broadcast 是从发送者向他的子scope广播一个事件。

$emit 广播给父controller，父controller 是可以收到消息

$on 有两个参数function(event,msg) 第一个参数是事件对象，第二个参数是接收到消息信息

a^=(1<<4)-1; return a;

1<<4 左移相当于1*2^4=16 a^=16-1=15 a=a^15=10^15

^ 异或运算：10的二进制00001010 15的二进制00001111 ==========>00000101 转成十进制：5

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers. Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

js中的对象：

对象分为三类： 内置对象　　宿主对象　　自定义对象

（1）内置对象： Arguments Array Boolean Date Error Function Number String Regexp Math Object

（2）宿主对象：运行环境提供的对象。如Window和Documen，Element，form，image。

（3）自定义对象：开发人员定义的对象。

清除浮动方法

方法一：使用带clear属性的空元素

在浮动元素后使用一个空元素如 ，并在CSS中赋予.clear{clear:both;}属性即可清理浮动。亦可使用<br class="clear" />或<hr class="clear" />来进行清理。

优点：简单，代码少，浏览器兼容性好。

缺点：需要添加大量无语义的html元素，代码不够优雅，后期不容易维护。

方法二：使用CSS的overflow属性

给浮动元素的容器添加overflow:hidden;或overflow:auto;可以清除浮动，另外在 IE6 中还需要触发 hasLayout，例如为父元素设置容器宽高或设置 zoom:1。在添加overflow属性后，浮动元素又回到了容器层，把容器高度撑起，达到了清理浮动的效果。

方法三：给浮动的元素的容器添加浮动

给浮动元素的容器也添加上浮动属性即可清除内部浮动，但是这样会使其整体浮动，影响布局，不推荐使用。

方法四：使用邻接元素处理

什么都不做，给浮动元素后面的元素添加clear属性。

方法五：使用CSS的:after伪元素

结合 :after 伪元素（注意这不是伪类，而是伪元素，代表一个元素之后最近的元素）和 IEhack，可以完美兼容当前主流的各大浏览器，这里的 IEhack 指的是触发 hasLayout。

给浮动元素的容器添加一个clearfix的class，然后给这个class添加一个:after伪元素实现元素末尾添加一个看不见的块元素（Block element）清理浮动。

```
1 for(let i=0;i<12;i++){} console.log(i);
2 const a=12; a=13; console.log(a);
3 const g={b:3};console.log(g.b);g.b=12;console.log(g.b);
4 let [head,...tail]=[1,2,3,4];console.log(tail);
```

1.let 与var不同，存在块级作用域，在for循环中声明，循环之外销毁 所以 i not defined

2.const 声明一个常量无法更改，所以TypeError

3.const 声明的是一个常量所以是无法更改的

```
1 const a={x:1};
2 console.log(a.x);
3 a.x=5;
4 console.log(a);//Object {x: 5}
```

主流浏览器内核私有属性css前缀：mozilla内核 (firefox,flock等)　-moz

webkit内核(safari,chrome等)　-webkit

opera内核(opera浏览器)　　-o

trident内核(ie浏览器)　　-ms

这里主要是讲标签的嵌套。

- 其中li标签必须嵌套在ul标签或ol标签中；
- dt标签和dd标签必须嵌套在dl标签里面，并且dt标签必须位于dd标签前面；
- tr标签和td标签必须嵌套在table标签里面，其中td标签必须位于tr标签里面。

type	button checkbox file hidden image password radio reset submit text	规定 input 元素的类型。

3.31：

HTML注释 <!--内容--> CSS注释 /* 内容 */ JS注释 单行注释以 // 开头。 多行注释以 /* */

<link> 和href配合 加载css，hypertext reference超文本引用，页面加载到href时不会停下来

<script>和src配合 加载script文件，source资源，页面会停下来等待资源加载完毕js放在body最下面

<mark> 标签定义带有记号的文本。请在需要突出显示文本时使用 <m> 标签。 把文本定义为强调的内容。

 把文本定义为语气更强的强调的内容。 <dfn> 定义一个定义项目。 <code> 定义计算机代码文本。 <samp> 定义样本文本。 <kbd> 定义键盘文本。它表示文本是从键盘上键入的。它经常用在与计算机相关的文档或手册中。 <var> 定义变量。您可以将此标签与 <pre> 及 <code> 标签配合使用。 <cite> 定义引用。可使用该标签对参考文献的引用进行定义，比如书籍或杂志的标题。 CSS字体加粗的方法： font-weight 属性设置文本的粗细。使用 bold 关键字可以将文本设置为粗体 highlight 该属性用来高亮显示指定的代码行。可以通过单个数字来高亮显示单行，或者传入一个类似 [1, 2, 3] 的数组来高亮显示指定的多行。

ajax() 方法通过 HTTP 请求加载远程数据。$.ajax(opts);opts为json格式，常见参数url、type、data

.load() 方法从服务器加载数据，并把返回的数据放入被选元素$(selector).load(URL,data,callback);

必需的 URL 参数规定您希望加载的 URL。可选的 data 参数规定与请求一同发送的查询字符串键/值对集合。可选的 callback 参数是 load() 方法完成后所执行的函数名称。

$.get() 方法通过 HTTP GET 请求从服务器上请求数据。$.get(URL,callback);

必需的 URL 参数规定您希望请求的 URL。可选的 callback 参数是请求成功后所执行的函数名。

getScript() 方法通过 HTTP GET 请求载入并执行 JavaScript 文件。

jQuery.getScript(url,success(response,status));

与浏览列表有关的对象:history screen location Navigator

scope: true和transclude: true会创建新的子作用域，并且进行原型继承；

scope: {...} 会创建新的独立作用域，不会进行原型继承;

默认情况下创建directive使用了scope: false，不会创建子作用域.

javascript异步模式的编程：

回调函数，这是异步编程最基本的方法。

事件监听，另一种思路是采用事件驱动模式。任务的执行不取决于代码的顺序，而取决于某个事件是否发生。

发布/订阅，上一节的"事件"，完全可以理解成"信号"。

Promises对象，Promises 对象是CommonJS 提出的一种规范，目的为异步编程提供统一接口。

var a=b=3 时a是局部变量，而b是全局变量

var obj=[ ];声明数组对象var obj={ };声明对象var obj=/ /;声明obj为正则对象

```
var myObject = {
    foo: "bar",
    func: function() {
        var self = this;
        console.log(this.foo);
        console.log(self.foo);
        (function() {
            console.log(this.foo);
            console.log(self.foo);
        }());
    }
};
myObject.func();
```

1.第一个this.foo输出bar，因为当前this指向对象myObject。

2.第二个self.foo输出bar，因为self是this的副本，同指向myObject对象。

3.第三个this.foo输出undefined，因为这个IIFE(立即执行函数表达式)中的this指向window。

4.第四个self.foo输出bar，因为这个匿名函数所处的上下文中没有self，所以通过作用域链向上查找，从包含它的父函数中找到了指向myObject对象的self。

`console.log(1+ "2"+"2");`做加法时要注意双引号，当使用双引号时，JavaScript认为是字符串，字符串相加等于字符串合并。因此，这里相当于字符串的合并，即为122.

`console.log(1+ +"2"+"2");`第一个+"2"中的加号是一元加操作符，+"2"会变成数值2，因此1+ +"2"相当于1+2=3.然后和后面的字符串“2”相合并，变成了字符串"32".

`console.log("A"- "B"+"2");`"A"-"B"的运算中，需要先把"A"和"B"用Number函数转换为数值，其结果为NaN，在剪发操作中，如果有一个是NaN，则结果是NaN，因此"A"-"B"结果为NaN。

然后和"2"进行字符串合并，变成了NaN2.

`console.log("A"- "B"+2);`"A"-"B"结果为NaN，然后和数值2进行加法操作，在加法操作中，如果有一个操作数是NaN，则结果为NaN。

call（）方法和apply（）方法的作用相同，他们的区别在于接收参数的方式不同。对于call（），第一个参数是this值没有变化，变化的是其余参数都直接传递给函数。（在使用call（）方法时，传递给函数的参数必须逐个列举出来。使用apply（）时，传递给函数的是参数数组）如下代码做出解释：

```
function add(c, d){
return this.a + this.b + c + d;
}
var o = {a:1, b:3};
add.call(o, 5, 7); // 1 + 3 + 5 + 7 = 16
add.apply(o, [10, 20]); // 1 + 3 + 10 + 20 = 34
```

class	浏览器	打印机
.visible-print-block .visible-print-inline .visible-print-inline-block	隐藏	可见
.hidden-print	可见	隐藏

a:link; a:visited; a:hover; a:active;

a:hover必须放在a:link和a:visited之后；a:active必须放在a:hover之后。

4.2：

<HR>：表示一条横线　　
:可插入一个简单的换行符　　<TR>产生一个表行

toggleClass() 对设置或移除被选元素的一个或多个类进行切换。

该方法检查每个元素中指定的类。如果不存在则添加类，如果已设置则删除之。这就是所谓的切换效果。不过，通过使用 "switch" 参数，您能够规定只删除或只添加类。

`var x = foo();　　var foo=function foo() {...}`

语句中变量的声明会提升，但是定义不会提升。以上代码等同于：

`var foo; var x = foo(); foo = function foo() {...}`

当执行到 x = foo() 时，由于foo未被定义为函数，所以会返回 `TypeError: foo is not a function`

HTML中的标签分为闭合标签和自闭合标签。 自闭合标签有<input/><img/>
<link/><hr/>等

于History对象的属性或方法:

length　　返回浏览器历史列表中的URL数量　　back()　　加载 history 列表中的前一个URL

forward() 加载 history 列表中的下一个URL　　go()　　加载history列表中的某个具体页面。

bootstrap

.table	为任意 <table> 添加基本样式 (只有横向分隔线)	尝试一下
.table-striped	在 <tbody> 内添加斑马线形式的条纹 (IE8 不支持)	尝试一下
.table-bordered	为所有表格的单元格添加边框	尝试一下
.table-hover	在 <tbody> 内的任一行启用鼠标悬停状态	尝试一下
.table-condensed	让表格更加紧凑	

对于border-radius：Internet Explorer 9+ 支持 border-radius 和 box-shadow 属性。
Firefox 、 Chrome 以及 Safari 支持所有新的边框属性。
注释：对于 border-image ， Safari 5 以及更老的版本需要前缀 -webkit- 。
Opera 支持 border-radius 和 box-shadow 属性，但是对于 border-image 需要前缀 -o-
当在 HTML 页面中执行脚本时，页面的状态是不可响应的，直到脚本已完成。 web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行。
call()方法,第一个参数和apply()一样,是在其中运行的作用域即this;和apply()不同的是,call()方法中的其余的参数必须直接传给函数,即在使用call()方法时参数必须逐个的列出来. this 在函数执行时，this 总是指向调用该函数的对象。要判断 this 的指向，其实就是判断 this 所在的函数属于谁。把 this 出现的场景分为四类，简单的说就是： 有对象就指向调用对象 没调用对象就指向全局对象 用new构造就指向新对象 通过 apply 或 call 或 bind 来改变 this 的所指。
`(function() {var a = b = 5;})();  console.log(b);console.log(a);`
第一个考点在于var a=b=5相当于拆解成var a=b; b=5; 然后，b=5前面没有var，相当于声明为全局变量（这种方式在严格模式下会报错，此题不考虑)。所以就相当于： var b; (fun…{ var a=b; b=5; })(); console.log(b); //5 console.log(a); //报错 此处报错也就是另一个考点，a声明的是函数的局部变量，在函数结束是就销毁了，所以在全局下找不到a，于是报错。
disabled指当 input 元素加载时禁用此元素。input内容不会随着表单提交 readonly规定输入字段为只读。input内容会随着表单提交。无论设置readonly还是disabled，通过js脚本都能更改input的value
要运用css3动画，需要运用@keyframes规则和animation属性
在标准的 JavaScript 中， Ajax 异步执行调用基于Event和callback机制
Firefox 不支持 DOM 对象的 outerHTML innerText outerText 属性。
在 IE6 IE7 IE8 中，createElement 方法不仅可以通过合法的标签名创建节点对象，还可以通过传入一段合法的 HTML 代码字符串作为参数创建节点对象。
I E6 IE7 IE8(Q) 中无法通过 "Element.setAttribute("class", "AttributeValue")" 设置元素的 class 属性，而需要使用 "Element.setAttribute("className", "AttributeValue")";
IE6 IE7 IE8(Q) 中无法通过诸如 "Element.setAttribute("onclick", "alert('ok')")" 为元素绑定事件;
4.3：通过iframe设置document.domain可以实现跨域
JSONP的优点是：它不像XMLHttpRequest对象实现的Ajax请求那样受到同源策略的限制；它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest或ActiveX的支持；并且在请求完毕后可以通过调用callback的方式回传结果。JSONP的缺点则是：它只支持GET请求而不支持POST等其它类型的HTTP请求；它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题。
关于Javascript中数字的部分知识总结：
1.Javascript中，由于其变量内容不同，变量被分为基本数据类型变量和引用数据类型变量。基本类型变量用八字节内存，存储基本数据类型(数值、布尔值、null和未定义)的值，引用类型变量则只保存对对象、数组和函数等引用类型的值的引用(即内存地址)。
2. JS中的数字是不分类型的，也就是没有byte/int/float/double等的差异。

AngularJs相对于其他的框架来说，有一下的特性：angular 指令的link函数中进行dom操作和事件绑定，服务主要是封装与后台交互的数据接口提供复用性。

1 MVVM

2 模块化

3 自动化双向数据绑定

4 语义化标签

5 依赖注入

1.良好的应用程序结构，层层分离。使用 AngularJS，您可以通过MVC（模型 - 视图 - 控制器）或MVVM （模型 - 视图 - 视图模型）模式来组织源代码。HTML模板将会被浏览器解析到DOM中，DOM结构成为AngularJS编译器的输入。AngularJS将会遍历DOM模板，来生成相应的NG指令,所有的指令都负责针对view(即HTML中的ng-model)来设置数据绑定。因此，NG框架是在DOM加载完成之后，才开始起作用的。在代码处声明ng-app让浏览器知道何处开始的代码属于AngularJS的代码范畴。代码结构上面的控制层（Conttroller）和视图层（View）通过$scope来进行连接。所有从后端请求的数据，我们使用服务（Services）来获取。并通过$rootScope.$broadcast广播的方式传送出去。在控制层接受数据，通过$scope来反馈到视图层。

我们在html中先定义一个angular的app,指定一个angular的controller,则该controller会对应于一个作用域(可以用$scope前缀来指定作用域中的属性和方法等). 则在该ngCtl的作用域内的HTML标签，其值或者操作都可以通过$scope的方式跟js中的属性和方法进行绑定。所以可以说$scope的作用是连接视图层和数据的桥梁。

2.双向数据绑定

解释性语言和编译性语言的定义：

计算机不能直接理解高级语言，只能直接理解机器语言，所以必须要把高级语言翻译成机器语言，计算机才能执行高级语言编写的程序。

翻译的方式有两种，一个是编译，一个是解释。两种方式只是翻译的时间不同。

解释性语言的定义：

解释性语言的程序不需要编译，在运行程序的时候才翻译，每个语句都是执行的时候才翻译。这样解释性语言每执行一次就需要逐行翻译一次，效率比较低。

现代解释性语言通常把源程序编译成中间代码，然后用解释器把中间代码一条条翻译成目标机器代码，一条条执行。

编译性语言的定义：

编译性语言写的程序在被执行之前，需要一个专门的编译过程，把程序编译成为机器语言的文件，比如exe文件，以后要运行的话就不用重新翻译了，直接使用编译的结果就行了（exe文件），因为翻译只做了一次，运行时不需要翻译，所以编译型语言的程序执行效率高。

CSS3新增属性用法整理：

1、box-shadow（阴影效果）

2、border-color（为边框设置多种颜色）

3、border-image（图片边框）

4、text-shadow（文本阴影）

5、text-overflow（文本截断）

6、word-wrap（自动换行）

7、border-radius（圆角边框）

8、opacity（透明度）

9、box-sizing（控制盒模型的组成模式）

10、resize（元素缩放）

11、outline（外边框）

12、background-size（指定背景图片尺寸）

13、background-origin（指定背景图片从哪里开始显示）

14、background-clip（指定背景图片从什么位置开始裁剪）

15、background（为一个元素指定多个背景）

16、hsl（通过色调、饱和度、亮度来指定颜色颜色值）

17、hsla（在hsl的基础上增加透明度设置）

18、rgba（基于rgb设置颜色，a设置透明度）

要理解display:inline、block、inline-block的区别，需要先了解HTML中的块级（block）元素和行级（inline）元素的特点，行内元素也叫内联元素。 块级元素 总是另起一行开始； 高度，行高以及顶、底边距都可以控制； 宽度缺省是它所在容器的100%，除非设定一个宽度。 块级元素通常作为其他元素的容器，可以容纳内联元素和其他块元素。 block element的HTML标签如下： address-地址 blockquote-块引用 center-居中对齐块 div-常用块级元素 dl-定义列表 form-交互表单 fieldset-form控制组 hr-水平分隔线 ol-排序表单 ul-非排序列表 dir-目录列表 p-段落 pre-格式化文本 isindex-input prompt menu-菜单列表 table-表格 h1...h6-标题 noframes-frames可选内容（对于不支持frame的浏览器显示此区块内容） noscript-可选脚本内容（对于不支持script的浏览器显示此内容） 行级元素 和其他元素都在一行上； 高度，行高以及顶、底边距不可改变； 高度就是它所容纳的文字或图片的宽度，不可改变。 一般都是基于语义级（semantic）的基本元素，只能容纳文本或者其他内联元素，内联元素的HTML标签分类如下： a-锚点 abbr-缩写 acronym-首字 font-字体设定（不推荐） b-粗体（不推荐） big-bidi override em-强调 br-换行 small-小字体文本 strong-粗体强调 i-斜体 img-图片 input-输入框 label-表格标签 select-项目选择 textarea-多行文本输入框 u-下划线 var-定义变量 cite-引用 code-计算机代码（在引用源码的时候需要） dfn-定义字段 kbd-定义键盘文本 q-短引用 s-中划线（不推荐） strike-中划线 sub-下标 sup-上标 tt-电传文本 HTML中有些元素是可变元素，可根据上下文语境决定该元素为块元素或者内联元素。 applet-java applet button-按钮 del-删除文本 iframe-inline frame ins-插入的文本 map-图片区块（map） object-object对象 script-客户端脚本 display:block就是将元素显示为块级元素，display:inline则将元素显示为行内元素。 display：inline-block将元素显示为行内元素，但是元素的内容作为块元素处理。旁边的内联元素和该对象显示在同一行，并且允许空格，但是该元素具有块元素的特性，可以设置其高度，宽度等属性。在同一行内有不同高度内容的元素时，通常要设置对齐方式如vertical-align: top;来使元素顶部对齐。 兼容性 CSS中使用display:inline-block;来样式化，在Firefox, Safari, Google Chrome 和 IE 8及以上是有效的。但是在早期的IE，比如IE 7，对行内元素设置inline-block无法实现inline-block的效果。只是触发了块元素的layout，而行内元素本身就是行布局，所以触发后，依然是行布局。 对IE8以下的版本，可以采用以下两种方法来实现inline-block的效果： 先用display:inline-block属性触发块元素，然后再定义display:inline，让块元素呈递为内联对象（原理：这是IE的一个经典bug，如果先定义了display:inline-block，然后再设置display回inline或block，layout不会消失），代码如下： 1 div {display:inline-block;}2 div {display:inline;} 将块元素设置为内联对象(display:inline)，然后通过zoom:1触发块元素的layout，代码如下： div {display:inline; zoom:1;}

在js里面添加的属性名使用驼峰法，在css里面使用连接线

```
window.setTimeout(checkState(), 10000); //立即被调用 window.setTimeout(checkState, 10000); // 10s后被调用
window.setTimeout("checkState()", 10000); //10s后被调用 注意和第一个的区别 有引号
```

1.while的话只是在函数局部环境或者全局环境运行，并不会改变作用域链。

2.try catch红皮书第四章讲的清清楚楚：虽然执行环境的类型总共只有两种--全局和局部（函数），但还是有其他办法来延长作用域链。这么说是因为有些语句可以在作用域链的前端临时增加一个变量对象，该变量对象会在代码执行后被移除。在两种情况下回发生这种现象。具体来说，就是当执行流进入下列任何一个语句时，作用域链就会得到加强：

try catch语句的catch块；

with语句；

这两个语句都会在作用域链的前端添加一个变量对象。对WITH语句来说，将会指定的对象添加到作用域链中。对catch语句来说，会创建一个新的变量对象，其中包含的是被抛出的错误对象的声明。

Example：

```
function builderUrl(){
     var qs="?debug=true";
     with(location){
     var url = href + qs;
     }
    return url;
```

}在此，with语句接受的是location对象，因此其变量对象中就包含了Location对象的所有属性和方法，而这个变量对象被添加到了作用域链的前端。builderUrl()函数中定义了一个变量qs。当在with语句中引用变量href时（实际引用的是location.href），可以在当前执行环境的变量对象中找到。当引用变量qs时，引用的则是在buildUrl()中定义的那个变

量，而该变量位于函数环境的变量对象中。至于with语句内部，则定义了一个名为url的变量，因而url就成了函数执行环节的一个部分，所以可以作为函数的值被返回。

html5中不再支持的元素：<noframes>,<frameset>,<frame>,<applet>,<acronym>,<basefont>,<dir>,<tt>,<strike>,<big>,<blink>,<s>,<font>

1、能用css代替的元素 ，basefont、big、center、font、s、strike、tt、u。这些元素纯粹是为画面展示服务的，HTML5中提倡把画面展示性功能放在css中统一编辑。2、不再使用frame框架。frameset、frame、noframes。HTML5中不支持frame框架，只支持iframe框架，或者用服务器方创建的由多个页面组成的符合页面的形式，删除以上这三个标签。3、只有部分浏览器支持的元素，applet、bgsound、blink、marquee等标签。4、其他被废除的元素。废除rb，树勇ruby替代。废除acronym使用abbr替代。废除dir使用ul替代。废除isindex使用form与input相结合的方式替代。

CSS中Position属性有四个可选值，它们分别是：static、absolute、fixed、relative。

◆position:static 无定位

该属性值是所有元素定位的默认情况，在一般情况下，我们不需要特别的去声明它，但有时候遇到继承的情况，我们不愿意见到元素所继承的属性影响本身，从而可以用position:static取消继承，即还原元素定位的默认值。

◆position:absolute 绝对定位

使用position:absolute，能够很准确的将元素移动到你想要的位置，

◆position:fixed 相对于窗口的固定定位

这个定位属性值是什么意思呢？元素的定位方式同absolute类似，但它的包含块是视区本身。在屏幕媒体如WEB浏览器中，元素在文档滚动时不会在浏览器视察中移动。例如，它允许框架样式布局。在页式媒体如打印输出中，一个固定元素会出现于第一页的相同位置。这一点可用于生成流动标题或脚注。我们也见过相似的效果，但大都数效果不是通过CSS来实现了，而是应用了JS脚本。

请特别注意，IE6不支持CSS中的position:fixed属性。真的非常遗憾，要不然我们就可以试试这种酷酷的效果了。

◆position:relative 相对定位

所谓相对定位到底是什么意思呢，是基于哪里的相对呢？我们需要明确一个概念，相对定位是相对于元素默认的位置的定位。既然是相对的，我们就需要设置不同的值来声明定位在哪里，top、bottom、left、right四个数值配合，来明确元素的位置。

promise

在 JavaScript 的异步进化史中，涌现出一系列解决 callback 弊端的库，而 Promise 成为了最终的胜者，并成功地被引入了 ES6 中。它将提供了一个更好的“线性”书写方式，并解决了异步异常只能在当前回调中被捕获的问题。

Promise 就像一个中介，它承诺会将一个可信任的异步结果返回。首先 Promise 和异步接口签订一个协议，成功时，调用resolve函数通知 Promise，异常时，调用reject通知 Promise。另一方面 Promise 和 callback 也签订一个协议，由 Promise 在将来返回可信任的值给then和catch中注册的 callback。promise 有一个 then 方法，then 方法可以接受 3 个函数作为参数。前两个函数对应 promise 的两种状态 fulfilled 和 rejected 的回调函数。第三个函数用于处理进度信息

4.4：

◎Math.ceil()执行向上舍入，即它总是将数值向上舍入为最接近的整数；

◎Math.floor()执行向下舍入，即它总是将数值向下舍入为最接近的整数；

◎Math.round()执行标准舍入，即它总是将数值四舍五入为最接近的整数(这也是我们在数学课上学到的舍入规则)。

jQuery width() 和 height() 方法

width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。

innerWidth() 方法返回元素的宽度（包括内边距）。innerHeight() 方法返回元素的高度（包括内边距）。

outerWidth() 方法返回元素的宽度（包括内边距和边框）。outerHeight() 方法返回元素的高度（包括内边距和边框）。

outerWidth(true) 方法返回元素的宽度（包括内边距、边框和外边距）。outerHeight(true) 方法返回元素的高度（包括内边距、边框和外边距）。

```
typeof  1; //'number'
typeof (1);//'number'
typeof (); //SyntaxError 语法错误

void  0; //undefined
void (0);//undefined
void (); //SyntaxError 语法错误


var f = function g() {
        return 23;
    };
typeof g();
```

如果是typeof f，结果是function 如果是typeof f()，结果是number 如果是typeof g,结果是undefined. 如果是typeof g(),结果是ReferenceError，g is not defined

常用的页面的图片格式有三种，GIF、JPG、PNG。 GIF 意为Graphics Interchange format（图形交换格式）； JPEG 代表Joint Photograhic Experts Group（联合图像专家组），这种格式经常写成JPG，JPG图片的扩展名为jpg；流式网络图形格式(Portable Network Graphic Format，PNG)名称来源于非官方的“PNG's Not GIF”，是一种位图文件(bitmap file)存储格式，读成“ping”。PNG用来存储灰度图像时，灰度图像的深度可多到16位，存储彩色图像时，彩色图像的深度可多到48位，并且还可存储多到16位的α通道数据。

标签图像文件格式 （ **Tagged Image File Format** ，简写为 **TIFF** ）是一种灵活的 位图 格式，主要用来存储包括照片和艺术图在内的图像。 **TIFF文件格式适用于在应用程序之间和计算机平台之间的交换文件，它的出现使得图像数据交换变得简单。** TIFF文件以 **.tif** 为扩展名。其数据格式是一种3级体系结构，从高到低依次为：文件头、一个或多个称为IFD的包含标记指针的目录和数据。

Math.max(args...)传入参数是任意数量的值 Function.call()可以传入任意多个参数，Function.apply()第二个参数以数组形式传递

setTimeout是window的一个方法，如果把window当做全局对象来看待的话，它就是全局函数。严格来讲，它不是。全局函数与内置对象的属性或方法不是一个概念。全局函数它不属于任何一个内置对象。JavaScript 中包含以下 7 个全局函数escape( )、eval( )、isFinite( )、isNaN( )、parseFloat( )、parseInt( )、unescape( )。

JavaScript 全局函数

函数	描述
decodeURI()	解码某个编码的 URI。
decodeURIComponent()	解码一个编码的 URI 组件。
encodeURI()	把字符串编码为 URI。
encodeURIComponent()	把字符串编码为 URI 组件。
escape()	对字符串进行编码。
eval()	计算 JavaScript 字符串，并把它作为脚本代码来执行。
isFinite()	检查某个值是否为有穷大的数。
isNaN()	检查某个值是否是数字。
Number()	把对象的值转换为数字。
parseFloat()	解析一个字符串并返回一个浮点数。
parseInt()	解析一个字符串并返回一个整数。
String()	把对象的值转换为字符串。
unescape()	对由 escape() 编码的字符串进行解码。

test.innerHTML:也就是从对象的起始位置到终止位置的全部内容,包括Html标签。

test.innerText: 从起始位置到终止位置的内容, 但它去除Html标签

test.outerHTML:除了包含innerHTML的全部内容外, 还包含对象标签本身。

1、什么是HTML语义化？〈基本上都是围绕着几个主要的标签，像标题（H1~H6）、列表（li）、强调（strong em）等等〉根据内容的结构化（内容语义化），选择合适的标签（代码语义化）便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析。 2、为什么要语义化？ 为了在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构:为了裸奔时好看； 用户体验：例如title、alt用于解释名词或解释图片信息、label标签的活用； 有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息： 爬虫依赖于标签来确定上下文和各个关键字的权重；方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页； 便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。 3、写HTML代码时应

注意什么？ 尽可能少的使用无语义的标签div和span； 在语义不明显时，既可以使用div或者p时，尽量用p, 因为p在默认情况下有上下间距，对兼容特殊终端有利； 不要使用纯样式标签，如：b、font、u等，改用css设置。 需要强调的文本，可以包含在strong或者em标签中（浏览器预设样式，能用CSS指定就不用他们），strong默认样式是加粗（不要用b），em是斜体（不用i）； 使用表格时，标题要用caption，表头用thead，主体部分用tbody包围，尾部用tfoot包围。表头和一般单元格要区分开，表头用th，单元格用td； 表单域要用fieldset标签包起来，并用legend标签说明表单的用途； 每个input标签对应的说明文本都需要使用label标签，并且通过为input设置id属性，在lable标签中设置for=someld来让说明文本和相对应的input关联起来。

验证数字：^[0-9]*$

验证n位的数字：^\d{n}$

验证至少n位数字：^\d{n,}$

验证m-n位的数字：^\d{m,n}$

验证零和非零开头的数字：^(0|[1-9][0-9]*)$

验证有两位小数的正实数：^[0-9]+(.[0-9]{2})?$

验证有1-3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$

验证非零的正整数：^\+?[1-9][0-9]*$

验证非零的负整数：^\-[1-9][0-9]*$

验证非负整数（正整数 + 0） ^\d+$

验证非正整数（负整数 + 0） ^((-\d+)|(0+))$

验证长度为3的字符：^.{3}$

验证由26个英文字母组成的字符串：^[A-Za-z]+$

验证由26个大写英文字母组成的字符串：^[A-Z]+$

验证由26个小写英文字母组成的字符串：^[a-z]+$

验证由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$

验证由数字、26个英文字母或者下划线组成的字符串：^\w+$

AngularJS性能问题，编译阶段应分为两个阶段

1,compile (绑定DOM)

2,link（数据绑定）。所以绑定dom应该在dom生成以后绑定的。不过也可能是动态生成的哦。

Readonly只针对input(text/password)和textarea有效，而disabled对于所有的表单元素有效，包括select,radio,checkbox,button等。

1) <meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;" />

//强制让文档的宽度与设备的宽度保持1:1，

//文档初始化缩放比例是1:1，

//不允许用户点击屏幕放大浏览，

//允许用户缩放到的最大比例，

//尤其要注意的是content里多个属性的设置一定要用逗号+空格来隔开，如果不规范将不会起作用。其他属性有：width;height; initial-scale; minimum-scale; maximum-scale; user-scalable;

2) <meta name="apple-mobile-web-app-capable" content="yes" />

//iPhone私有标签，它表示：允许全屏模式浏览

3) <meta name="apple-mobile-web-app-status-bar-style" content="black" />

//iPhone私有标签，它指定的iPhone中safari顶端的状态条的样式

4) <meta name="format-detection" content="telephone=no; email=no" /> //不识别邮件和不把数字识别为电话号码

js可以使用jsonp进行跨域 通过修改document.domain来跨子域 使用window.name来进行跨域

JavaScript 保留关键字

Javascript 的保留关键字不可以用作变量、标签或者函数名。有些保留关键字是作为 Javascript 以后扩展使用。

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

* 标记的关键字是 ECMAScript5 中新添加的。

DHTML实现了网页从Web服务器下载后无需再经过服务的处理，而在浏览器中直接动态地更新网页的内容、排版样式和动画的功能。例如，当鼠标指针移到文章段落中时，段落能够变成蓝色，或者当鼠标指针移到一个超级链接上时，会自动生成一个下拉式子链接目录等。

包括：

①动态内容(Dynamic Content)：动态地更新网页内容，可“动态”地插入、修改或删除网页的元件，如文字、图像、标记等。

②动态排版样式(Dynamic Style Sheets)：W3C的CSS样式表提供了设定HTML标记的字体大小、字形、样式、粗细、文字颜色、行高度、加底线或加中间横线、缩排、与边缘距离、靠左右或置中、背景图片或颜色等排版功能，而“动态排版样式”即可以“动态”地改变排版样式

DNS是域名系统的缩写，他是由解析器和域名服务器组成的。域名服务器是保存有该网络中所有主机的域名和对应的ip地址。域名必对应一个ip地址，而ip地址不一定有域名。将域名映射为ip地址的过程就称为域名解析。域名虽便于人们记忆，但计算机只能互相认识ip地址。DNS就是进行域名解析的服务器。 DNS主要是UDP协议，但是当请求字节过长超过512字节时，是用TCP协议，它可以分割成多个片段。DNS协议默认端口号是53。游览器的DNS缓存：chrome对每个域名会默认缓存60s；IE将DNS缓存30min；Firefox默认缓存时间只有1分钟；Safari约为10s

:IE6.0的div的内嵌div可以把父级的高度撑大，而FireFox不可以，要自己设置高度。

: 当设置为三列布局时，IE6.0的float宽度不能达到100％，而FireFox可以。当设置为两列布局时，两种浏览器都可以。

火狐浏览器中，非float的div前面有同一父级的float的div，此div若有背景图，要使用clear：both，才能显示背景图，而IE6.0中不用使用clear：both在[text-decoration:underline]的属性下，IE6.0显示的下划线会比FireFox低一点。在FireFox中，部分笔画会在下划线的下面1个象素左右。

可能的值

值	描述
absolute	生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
fixed	生成绝对定位的元素，相对于浏览器窗口进行定位。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
relative	生成相对定位的元素，相对于其正常位置进行定位。 因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。
static	默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
inherit	规定应该从父元素继承 position 属性的值。

1.<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器的解析器，用什么文档类型 规范来解析这个文档。

2.严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。3.DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

BFC就是“块级格式化上下文”的意思，创建了 BFC的元素就是一个独立的盒子，不过只有Block-level box可以参与创建BFC， 它规定了内部的Block-level Box如何布局，并且与这个独立盒子里的布局不受外部影响，当然它也不会影响到外面的元素。 内部的Box会在垂直方向，从顶部开始一个接一个地放置。

Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生叠加

每个元素的margin box的左边， 与包含块border box的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

BFC的区域不会与float box叠加。BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之亦然。计算BFC的高度时，浮动元素也参与计算。