第1单元 html5概述及语义化标签

1.1 什么是HTML5

首先了解HTML，HTML即超文本标记语言（HyperText Markup language）,这是一种语法简单、结构清晰的语解释型文档,他不同于其他的编程语言。HTML5就是HTML网页标记语言的第五次重大更新产品，在这个版本中，新功能不断推出，以帮助Web应用程序的作者，努力提高新元素互操作性。

1.2 HTML5的目标

HTML5他的产生，最简单的目标就是，用更简洁的HTML代码创建更多功能的网页程序！并且HTML5让网页结构变得更加的清楚明了，增加了更加语义化的结构标签，这样<br>一个网页的结构就非常清晰，那个部位显示的什么内容，让人一目了然！

1.3 HTML5的好处

为开发人员节约时间，这句话怎么理解，因为之前我们所使用的HTML/XHTML在各大浏览器之间存在的不兼容问题显示的非常严重的问题，然而HTML5的出现就是为了解决这一个问题，HTML5的目标就是将HTML5网页上的音视频、图像、动画等等都带入一个国际标准化！

1.4 结构清晰的HTML5

结构清晰的HTML5，在HTML5中增加了主体元素，比如新增的NAV标签，表示的就是导航的意思，而之前呢，就是用DIV并没有实际的意义

1.5新增的语义化标签

1.5.1新增的结构性元素

<header>元素

header元素表示页面中的一个内容区块或者整个页面的标题/头部区域！

<nav>元素

nav元素表示页面中的导航链接部分。

<article>元素

<article> 标签定义外部的内容。外部内容可以是来自一个外部的新闻提供者的一篇新的文章，或者来自 blog 的文本，或者是来自论坛的文本。亦或是来自其他外部源内容。

article元素表示页面中的一块上下文不相关的的独立内容。

<section>元素

section表示页面中的一块内容区块，比如章节、页眉、页脚等等。也可以和Hn（h1、h2..)等一起使用，标示出文档的结构！

<footer>元素

footer表示页面或者是页面中的一块区域的页脚，比如存放文件的创建时间、作者、联系方式等等。

1.5.2新增的行内语义性元素

主要完成web页面具体内容的引用和描述，是丰富内容展示的基础.

meter：表示特定范围内的数值，可用于工资、数量、百分比等。

time：表示时间值。

progress：用来表示进度条，可通过对其max、min、step等属性进行控制，完成对进度的表示和监事。

video：视频元素，用于支持和实现视频文件的直接播放，支持缓冲预载和多种视频媒体格式。

audio：音频元素，用于支持和实现音频文件的直接播放，支持缓冲预载和多种音频媒体格式。

1.5.3新增的块级性元素

级块性元素主要完成web页面区域的划分，确保内容的有效分割

aside：用于表达注记、贴士、侧栏、摘要、插入的引用等作为补充主体的内容。

figure：用于图像，通常与figurecaption联合使用。

code：表示一段代码块。

1.5.4新增的交互性元素

交互性元素主要用于功能性的内容表达，会有一定的内容和数据的关联，是各种事件的基础。

details：用来表示一段具体的内容，但是内容默认可能不显示，通过某种手段（如单击）与legend交互才会显示出来。

menu：主要用于交互菜单（曾被废弃又被重新启用的元素）。

command：用来处理命令按钮。

课后练习：

1、什么是html5?

2、html5新增了哪些标签？

3、nav标签的作用？

4、header标签的作用？

5、footer标签的作用？

6、section标签的作用？

7、progress标签的作用？

8、article标签的作用？

9、语义化标签的优点，好处？

10、html5的更新了哪些东西？

技能练习：练习布局，使用html5新标签完成

第2单元：HTML5智能表单

参考http://www.w3school.com.cn/html5/att\_input\_type.asp

Type类型

button：定义可点击的按钮（通常与 JavaScript 一起使用来启动脚本）。<input type="button" name="" id="" value="按钮"><br><br>

图片 15图片 14  
checkbox：定义复选框。<br><input type="checkbox" name="" id="">男<br><input type="checkbox" name="" id="" checked>女<br><br>

color：定义拾色器。<br><input type="color" name="" id=""><br><br>图片 17

Date：定义 date 控件（包括年、月、日，不包括时间）。<br><input type=“date" name="" id=""><br><br>图片 16

datetime：定义 date 和 time 控件（包括年、月、日、时、分、秒、几分之一秒，基于 UTC 时区）。<br><form action="">生日 (日期和时间):<input type="datetime" name="" id=""> <input type="submit" name="" id="" value=“提交"></form><br><br>

**注释：**Internet Explorer、Firefox 或者 Chrome 不支持 <input type="datetime"> 元素，Safari 中部分支持。Opera 12 以及更早的版本中完全支持。图片 18

图片 19datetime-local：定义 date 和 time 控件（包括年、月 、日、时、分、秒、几分之一秒，不带时区）。<br><input type="datetime-local" name="" id=""><br><br>

图片 20month：定义 month 和 year 控件（不带时区）。<br><input type="month" name="" id=""><br><br>

图片 21week：定义 week 和 year 控件（不带时区）。<br><input type="week" name="" id=""><br><br>

图片 22time：定义用于输入时间的控件（不带时区）。<br><input type="time" name="" id=""><br><br>

email：定义用于 e-mail 地址的字段。<br><form action=""></form>E-mail:<input type="email" name="" id=""></form><br><br>图片 23

图片 25file：定义文件选择字段和 "浏览..." 按钮，供文件上传。<br><input type="file" name="" id=""><br><br>

图片 24hidden：定义隐藏输入字段。<br><input type="hidden" name="" id=""><br><br>

图片 26image：定义图像作为提交按钮。<br><input type="image" name="" id=""><br><br>

图片 27number：定义用于输入数字的字段。<br><input type="number" name="" id=""><br><br>

图片 28password：定义密码字段（字段中的字符会被遮蔽）。<br><input type="password" name="" id=""><br><br>

radio：定义单选按钮。<br><input type="radio" name="" id="">男<br>

<input type="radio" name="" id="" checked>女<br><br>

range：定义用于精确值不重要的输入数字的控件（比如 slider 控件）。<br>

<input type="range" name="points" min="1" max="10" />  
  


图片 30reset：定义重置按钮（重置所有的表单值为默认值）。<br>

<input type="reset" name="" id=""><br><br>

search：定义用于输入搜索字符串的文本字段。搜索字段的外观与常规的文本字段无异。

图片 31<input type="search" name="" id=""><br><br>

submit：定义提交按钮。<br>

图片 32<input type="submit" name="" id="" value="提交"><br><br>

tel：定义用于输入电话号码的字段。<br>

图片 33输入电话：<input type="tel" name="" id=""><br><br>

图片 34text：默认。定义一个单行的文本字段（默认宽度为 20 个字符）。<br>

<input type="text" name="" id=""><br><br>

url：定义用于输入 URL 的字段。<br>

添加你的主页网址：<input type="url" name="" id="">

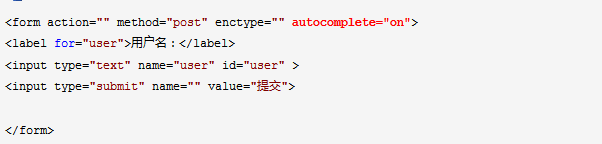
图片 35

属性

参考 http://www.w3school.com.cn/html5/html\_5\_form\_attributes.asp

autocomplete 表单自动完成： 书写格式autocomplete=“on/off”  不加此属性则默认是on,出于某些方面考虑需要关闭的话就要加上，然后值设置为off;

autocomplete控件是指用户在文本框输入前几个字母或是汉字的时候， autocomplete控件就能从存放数据的文本或是数据库里将所有以这些字母开头的数据提示给用户，供用户选择，提供方便。

autocomplete 适用于 <form> 标签，以及以下类型的 <input> 标签：text, search, url, telephone, email, password, datepickers, range 以及 color。

效果：

2. autofocus 自动聚焦  当页面加载完成后，光标自动聚焦到设置有此属性的栏目中，书写格式autofocus =“autofocus ”；

3. form 规定输入域所属的表单  书写格式: form="formID1  formID2"(需要为指定form指定ID);

4. formaction  重新指定表单的action属性；书写格式：formaction="demo,jsp"(可以让一个表单的内容通过不同的提交按钮提交到不同 地方)；

5. height width 指定input type="image"时，input标签的宽高（对其他input项无效 ）书写格式input type="image" width="100" height="100px"；

6. list 规定输入域的datalist 书写格式：list="namelist"(其中namelist为对应的datalist列表的ID名)；

7. min max 规定输入域中允许输入的最小和最大值，书写格式：min="10" max="100"

8. step 规定输入域中输入的内容的步进数，书写格式：step="3", 如果step="3"则能够输入的合法字符为：0,3,6....;

9. multiple 规定输入域中是否可选多个值，书写格式：multiple="multiple";  如：加此属性后file可选择多个文件；

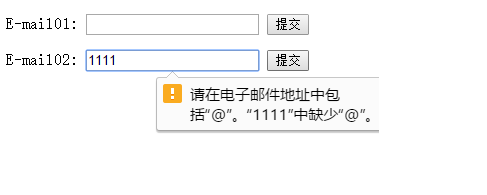
10. novalidate  规定在表单提交时不验证form或input项；书写格式：novalidate="true"/novalidate="novalidate"/novalidate;

novalidate 属性的一个boolean 属性.

novalidate 属性规定在提交表单时不应该验证 form 或 input 域。

表单01 type类型email用了novalidate属性，提交表单时不需要验证则提交成功

表单02 type类型email没用了novalidate属性，提交表单时需要验证。

效果

11. pattern 规定用于验证input输入的内容是否为正则表达式的匹配，书写格式：pattern =“正则表达式”；

12. placeholder 规定input输入域在空白时给出的提示内容，获得焦点时其内容消失，书写格式： placeholder="想要显示的内容"；

placeholder属性提供一种提示（hint），描述输入域所期待的值。

简短的提示在用户输入值前会显示在输入域上。如图，灰色字体的请输入用户名...

placeholder 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email 以及 password。



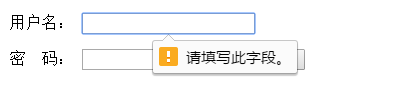
效果

13、required  规定form中的必填项，书写格式：required="required"/required;

required 属性是一个 boolean 属性.

required 属性规定必须在提交之前填写输入域（不能为空）。

required 属性适用于以下类型的 <input> 标签：text, search, url, telephone, email, password, date pickers, number, checkbox, radio 以及 file。

效果

课后练习：

1. required属性的作用是什么？
2. placeholder 属性的作用是什么？
3. autofocus 属性的作用是什么？
4. pattern属性的作用是什么？
5. autocomplete属性的作用是什么？
6. type=range 生成什么效果
7. 怎么使form的表单标签脱离form的嵌套

第3单元：HTML5 canvas画布

什么是 Canvas？

HTML5 的 canvas 元素使用 JavaScript 在网页上绘制图像。

画布是一个矩形区域，您可以控制其每一像素。

canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法

创建一个画布（Canvas）

一个画布在网页中是一个矩形框，通过 <canvas> 元素来绘制.

注意: 默认情况下 <canvas> 元素没有边框和内容。

<canvas>简单实例如下:

<canvas id="myCanvas" width="200" height="100"></canvas>

注意: 标签通常需要指定一个id属性 (脚本中经常引用), width 和 height 属性定义的画布的大小.

提示:你可以在HTML页面中使用多个 <canvas> 元素.

使用 style 属性来添加边框:

通过 JavaScript 来绘制

canvas 元素本身是没有绘图能力的。所有的绘制工作必须在 JavaScript 内部完成：

<script type="text/javascript">

var c=document.getElementById("myCanvas");

var cxt=c.getContext("2d");

cxt.fillStyle="#FF0000";

cxt.fillRect(0,0,150,75);

</script>

JavaScript 使用 id 来寻找 canvas 元素：

var c=document.getElementById("myCanvas");

然后，创建 context 对象：

var cxt=c.getContext("2d");

getContext("2d") 对象是内建的 HTML5 对象，拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

Canvas 坐标

canvas 是一个二维网格。

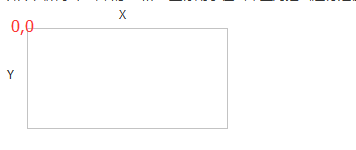
canvas 的左上角坐标为 (0,0)

上面的 fillRect 方法拥有参数 (0,0,150,75)。strokeRect()

意思是：在画布上绘制 150x75 的矩形，从左上角开始 (0,0)。

坐标实例

如下图所示，画布的 X 和 Y 坐标用于在画布上对绘画进行定位。鼠标移动的矩形框上，显示定位坐标。



Canvas - 路径

在Canvas上画线，我们将使用以下两种方法：

moveTo(x,y) 定义线条开始坐标

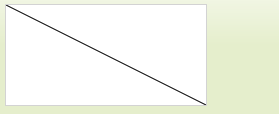
lineTo(x,y) 定义线条结束坐标

绘制线条我们必须使用stroke().

定义开始坐标(0,0), 和结束坐标 (200,100)。然后使用 stroke() 方法来绘制线条:

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); ctx.moveTo(0,0); ctx.lineTo(200,100); ctx.stroke();

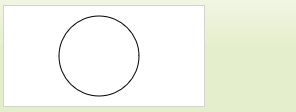
效果

在canvas中绘制圆形, 我们将使用arc(x,y,r,start,stop)

使用 arc() 方法 绘制一个圆

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); ctx.beginPath(); ctx.arc(95,50,40,0,2\*Math.PI); ctx.stroke();



Canvas - 文本

使用 canvas 绘制文本

font - 定义字体

fillText(text,x,y) - 在 canvas 上绘制实心的文本

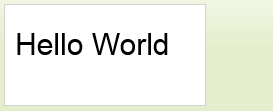
strokeText(text,x,y) - 在 canvas 上绘制空心的文本

使用 fillText():

使用 "Arial" 字体在画布上绘制一个高 30px 的文字（实心）

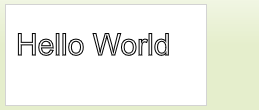
JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); ctx.font="30px Arial"; ctx.fillText("Hello World",10,50);

效果

使用 strokeText()

使用 "Arial" 字体在画布上绘制一个高 30px 的文字（空心）

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); ctx.font="30px Arial"; ctx.strokeText("Hello World",10,50);

Canvas - 渐变

渐变可以填充在矩形, 圆形, 线条, 文本等等, 各种形状可以自己定义不同的颜色。

以下有两种不同的方式来设置Canvas渐变：

createLinearGradient(x,y,x1,y1) - 创建线条渐变

createRadialGradient(x,y,r,x1,y1,r1) - 创建一个径向/圆渐变

当我们使用渐变对象，必须使用两种或两种以上的停止颜色。

addColorStop()方法指定颜色停止，参数使用坐标来描述，可以是0至1.

使用渐变，设置fillStyle或strokeStyle的值为 渐变，然后绘制形状，如矩形，文本，或一条线。

使用 createLinearGradient()创建一个线性渐变。使用渐变填充矩形:

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); // 创建渐变 var grd=ctx.createLinearGradient(0,0,200,0); grd.addColorStop(0,"red"); grd.addColorStop(1,"white"); // 填充渐变 ctx.fillStyle=grd; ctx.fillRect(10,10,150,80);

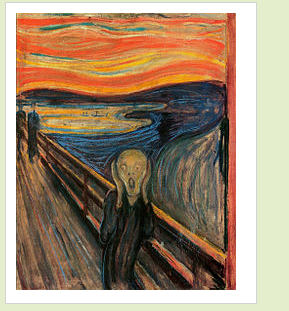
使用 createRadialGradient()创建一个径向/圆渐变。使用渐变填充矩形:

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); // 创建渐变 var grd=ctx.createRadialGradient(75,50,5,90,60,100); grd.addColorStop(0,"red"); grd.addColorStop(1,"white"); // 填充渐变 ctx.fillStyle=grd; ctx.fillRect(10,10,150,80);

Canvas - 图像

把一幅图像放置到画布上, 使用drawIma ge(image,x,y)

JavaScript:

var c=document.getElementById("myCanvas"); var ctx=c.getContext("2d"); var img=document.getElementById("scream"); ctx.drawImage(img,10,10);

课后练习：

1. 怎么绘制直线
2. 怎么绘制矩形
3. 怎么绘制圆形
4. 怎么绘制三角形
5. 怎么清空画布
6. 添加closePath 有什么效果
7. beginPath的作用是什么
8. MoveTo 和 lineTo 的区别是什么
9. 二次贝塞尔和三次贝塞尔曲线的关键字是什么
10. Stroke（）的作用是什么
11. Fill()的作用是什么
12. 怎么更改线的颜色以及粗细

第4单元HTML5音频与视频

音视频的发展史

早期：<embed>+<object>+文件

问题：不是所有浏览器都支持，而且embed不是标准。

现状：Realplay、window media、Quick Time 、Flash

问题：每个厂商每个标准，网站编码和格式也都不相同，flash的出现解决了面的问题，但是apple在07年决定任何设备将不再支持flash。

HTML5认为浏览器应该原生支持音视频，因为他们现在也是web中的一等公民了！

视频格式的简单介绍

1、常见的视频格式

视频的组成部分：画面、音频、编码格式

视频编码：H.264、Theora、VP8(google开源)

常见的音频格式

音频编码：ACC、MP3、Vorbis

HTML5支持的格式

HTML5能在完全脱离插件的情况下播放音视频

但是不是所有格式都支持。

HTML5支持的视频格式：

Ogg =带有Theora视频编码+Vorbis音频编码的Ogg文件

支持的浏览器:F、C、O

MEPG4=带有H.264视频编码+AAC音频编码的MPEG4文件

支持的浏览器: S、C

WebM=带有VP8视频编码+Vorbis音频编码的WebM格式 支持的浏览器: I、F、C、O

<Video>的使用

<video src="文件地址" controls="controls"></video>

< video src="文件地址" controls="controls">

您的浏览器暂不支持video标签。播放视频

</ video >

< video controls="controls" width="300">

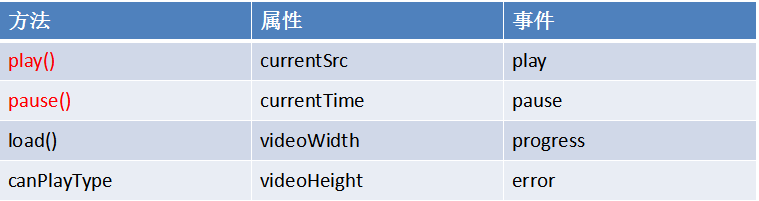
<source src="move.ogg" type="video/ogg" >

<source src="move.mp4" type="video/mp4" >

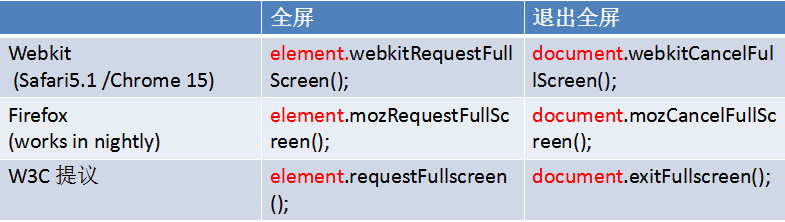
您的浏览器暂不支持video标签。播放视频

</ video >

Video的常见属性



video的API方法



Video的API属性



Video的常用事件



HTML5支持的音频格式

HTML5在不使用插件的情况下也可以原生的支持音频格式文件的播放，当然支持格式是有限的

HTML5支持的音频格式：

Ogg 免费 支持的浏览器:C、F、O

MP3 收费 支持的浏览器: I、C、S

Wav 收费 支持的浏览器: F、O、S

<audio>的使用

<audio src="文件地址" controls="controls"></audio>

< audio src="文件地址" controls="controls">

您的浏览器暂不支持audio标签。播放视频

</ video >

< audio controls="controls" >

<source src="happy.MP3" type="video/mpeg" >

<source src="happy.ogg" type="video/ogg" >

您的浏览器暂不支持audio标签。播放视频

</ audio>

课后练习：

1. controls属性添加以后有什么效果
2. 可以通过Js暂停音频视频的播放么，关键字是什么？
3. Video、audio 哪个是音频哪个是视频
4. 怎么获取当前音频播放的时间
5. 怎么获取当前视频的总时间
6. 音频、视频的音量通过什么关键字来控制
7. Autopaly的作用是什么
8. Loop的作用是什么

第5单元：Web存储

5.1认识Web Storage

5.1.1什么是web存储 http://www.runoob.com/html/html5-webstorage.html

使用HTML5可以在本地存储用户的浏览数据。

早些时候,本地存储使用的是 cookie。但是Web 存储需要更加的安全与快速. 这些数据不会被保存在服务器上，但是这些数据只用于用户请求网站上.

它也可以存储大量的数据，而不影响网站的性能.

数据以 键/值 对存在, web网页的数据只允许该网页访问使用

5.1.2为什么 要用Web存储

降低延迟：缓存离客户端更近，因此，从缓存请求内容比从源服务器所用时间更少，呈现速度更快，网站就显得更灵敏

降低网络传输：副本被重复使用，大大降低了用户的带宽使用，其实也是一种变相的省钱（如果流量要付费的话），同时保证了带宽请求在一个低水平上，更容易维护了。

5.1.3Web存储的优缺点

web storage的概念和cookie相似，区别是它是为了更大的容量存储设计的。cookie大小受限，并且每次请求一个新的页面的时候后cookie都会被发送过去，另外cookie还需要指定作用域，不可以跨域调用。

cookie的作用是与服务器进行交互，作为HTTP规范的一部分存在，而web storage仅仅是为了在本地存储数据而生

5.2使用WebStorage(本地存储)

5.2.1 localStorage

5.2.2 sessionStorage

webStorage提供两种类型的API：localStorage和sessionStorage，两者的区别看名字就有大概了 解， localStorage在本地永久性存储数据，除非显式将其删除或清空，sessionStorage存储的数据只在会话 期间有效，关闭浏览器则自 动删除。两个对象都有共同的API

length:唯一的属性，只读，用来获取storage内的键值对数量。

key：根据index获取storage的键名

localStorage.key(index)

getItem：根据key获取storage内的对应value

setItem：为storage内添加键值对

removeItem：根据键名，删除键值对

clear：清空storage对象

课后练习：

1. 什么是WebStorage
2. WebStorage的优点在哪
3. localStorage是什么
4. sessionStorage是什么
5. localStorage和sessionStorage有什么不同，区别在哪
6. 使用什么方法存数据
7. 怎么样提取已经存在的数据

6.单元地理定位

http://www.w3school.com.cn/html5/html\_5\_geolocation.asp

什么是地理定位

Geolocation

定位用户当前的位置。

地理位置

经度 : 南北极的连接线

纬度 : 东西连接的线

位置信息从何而来

1. IP地址
2. GPS全球定位系统
3. Wi-Fi无线网络

基站

地理位置对象

navigator.geolocation

单次定位请求 ：getCurrentPosition(请求成功，请求失败，数据收集方式)

请求成功函数

经度 : coords.longitude

纬度 : coords.latitude

准确度 : coords.accuracy

海拔 : coords.altitude

海拔准确度 : coords.altitudeAccuracy

行进方向 : coords.heading

地面速度 : coords.speed

时间戳 : new Date(position.timestamp)

地理位置对象\_2

navigator.geolocation

请求失败函数

失败编号 ：code

0 : 不包括其他错误编号中的错误

1 : 用户拒绝浏览器获取位置信息

2 : 尝试获取用户信息，但失败了

3 : 设置了timeout值，获取位置超时了

数据收集 : json的形式

enableHighAcuracy : 更精确的查找，默认false

timeout : 获取位置允许最长时间，默认infinity

maximumAge : 位置可以缓存的最大时间，默认0

地理位置对象\_3

navigator.geolocation

多次定位请求 : watchPosition(像setInterval)

移动设备有用，位置改变才会触发

配置参数：frequency 更新的频率

关闭更新请求 : clearWatch(像clearInterval)

利用百度地图和谷歌地图接口获取用户地址

我们只需要将HTML5获取到的经纬度信息传给地图接口，则会返回用户所在的地理位置，包括省市区信息，甚至有街道、门牌号等详细的地理位置信息。

课后练习：

1. Geolocation是什么？
2. watchPosition是什么？
3. getCurrentPosition是什么？
4. watchPosition和getCurrentPosition有什么区别

5、longitude获取的什么？

6、speed 获取的什么？

7、altitudeAccuracy 获取的什么？

8、altitude 获取的什么？

9、accuracy 获取的什么？

10、latitude获取的什么？

7.单元LESS预处理语言

什么是Less

CSS（层叠样式表）是一门历史悠久的标记性语言，同 HTML 一道，被广泛应用于万维网（World Wide Web）中。HTML 主要负责文档结构的定义，CSS 负责文档表现形式或样式的定义。作为一门标记性语言，CSS 的语法相对简单，对使用者的要求较低，但同时也带来一些问题：CSS 需要书写大量看似没有逻辑的代码，不方便维护及扩展，不利于复用，尤其对于非前端开发工程师来讲，往往会因为缺少 CSS 编写经验而很难写出组织良好且易于维护的 CSS 代码，造成这些困难的很大原因源于 CSS 是一门非程序式语言，没有变量、函数、SCOPE（作用域）等概念。LESS 为 Web 开发者带来了福音，它在 CSS 的语法基础之上，引入了变量，Mixin（混入），运算以及函数等功能，大大简化了 CSS 的编写，并且降低了 CSS 的维护成本，就像它的名称所说的那样，LESS 可以让我们用更少的代码做更多的事情。

LESS 原理

LESS 包含一套自定义的语法及一个解析器，用户根据这些语法定义自己的样式规则，这些规则最终会通过解析器，编译生成对应的 CSS 文件。LESS 并没有裁剪 CSS 原有的特性，更不是用来取代 CSS 的，而是在现有 CSS 语法的基础上，为 CSS 加入程序式语言的特性。

使用方法

清单 1. LESS 文件

@color: #4D926F;

#header {

color: @color;

}

h2 {

color: @color;

}

经过编译生成的 CSS 文件如下：

清单 2. CSS 文件

#header {

color: #4D926F;

}

h2 {

color: #4D926F;

}

语法

变量

LESS 允许开发者自定义变量，变量可以在全局样式中使用，变量使得样式修改起来更加简单。

我们可以从下面的代码了解变量的使用及作用：

清单 3 LESS 文件

@border-color : #b5bcc7;

.mythemes tableBorder{

border : 1px solid @border-color;

}

经过编译生成的 CSS 文件如下：

清单 4. CSS 文件

.mythemes tableBorder {

border: 1px solid #b5bcc7;

}

从上面的代码中我们可以看出，变量是 VALUE（值）级别的复用，可以将相同的值定义成变量统一管理起来。

该特性适用于定义主题，我们可以将背景颜色、字体颜色、边框属性等常规样式进行统一定义，这样不同的主题只需要定义不同的变量文件就可以了。当然该特性也同样适用于 CSS RESET（重置样式表），在 Web 开发中，我们往往需要屏蔽浏览器默认的样式行为而需要重新定义样式表来覆盖浏览器的默认行为，这里可以使用 LESS 的变量特性，这样就可以在不同的项目间重用样式表，我们仅需要在不同的项目样式表中，根据需求重新给变量赋值即可。

LESS 中的变量和其他编程语言一样，可以实现值的复用，同样它也有生命周期，也就是 Scope（变量范围，开发人员惯称之为作用域），简单的讲就是局部变量还是全局变量的概念，查找变量的顺序是先在局部定义中找，如果找不到，则查找上级定义，直至全局。

单的例子来解释 Scope。

清单 5. LESS 文件

@width : 20px;

#homeDiv {

@width : 30px;

#centerDiv{

width : @width;// 此处应该取最近定义的变量 width 的值 30px

}

}

#leftDiv {

width : @width; // 此处应该取最上面定义的变量 width 的值 20px

}

经过编译生成的 CSS 文件如下：

清单 6. CSS 文件

#homeDiv #centerDiv {

width: 30px;

}

#leftDiv {

width: 20px;

}

Mixins（混入）

Mixins（混入）功能对用开发者来说并不陌生，很多动态语言都支持 Mixins（混入）特性，它是多重继承的一种实现，在 LESS 中，混入是指在一个 CLASS 中引入另外一个已经定义的 CLASS，就像在当前 CLASS 中增加一个属性一样。

清单 7. LESS 文件

// 定义一个样式选择器

.roundedCorners(@radius:5px) {

-moz-border-radius: @radius;

-webkit-border-radius: @radius;

border-radius: @radius;

}

// 在另外的样式选择器中使用

#header {

.roundedCorners;

}

#footer {

.roundedCorners(10px);

}

经过编译生成的 CSS 文件如下：

清单 8. CSS 文件

#header {

-moz-border-radius:5px;

-webkit-border-radius:5px;

border-radius:5px;

}

#footer {

-moz-border-radius:10px;

-webkit-border-radius:10px;

border-radius:10px;

}

Mixins 其实是一种嵌套，它允许将一个类嵌入到另外一个类中使用，被嵌入的类也可以称作变量，简单的讲，Mixins 其实是规则级别的复用。

Mixins 还有一种形式叫做 Parametric Mixins（混入参数），LESS 也支持这一特性：

清单 9. LESS 文件

// 定义一个样式选择器

.borderRadius(@radius){

-moz-border-radius: @radius;

-webkit-border-radius: @radius;

border-radius: @radius;

}

// 使用已定义的样式选择器

#header {

.borderRadius(10px); // 把 10px 作为参数传递给样式选择器

}

.btn {

.borderRadius(3px);// // 把 3px 作为参数传递给样式选择器

}

经过编译生成的 CSS 文件如下：

清单 10. CSS 文件

#header {

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

border-radius: 10px;

}

.btn {

-moz-border-radius: 3px;

-webkit-border-radius: 3px;

border-radius: 3px;

}

我们还可以给 Mixins 的参数定义一人默认值

清单 11. LESS 文件

.borderRadius(@radius:5px){

-moz-border-radius: @radius;

-webkit-border-radius: @radius;

border-radius: @radius;

}

.btn {

.borderRadius;

}

经过编译生成的 CSS 文件如下：

清单 12. CSS 文件

.btn {

-moz-border-radius: 5px;

-webkit-border-radius: 5px;

border-radius: 5px;

}

像 JavaScript 中 arguments一样，Mixins 也有这样一个变量：@arguments。@arguments 在 Mixins 中是一个很特别的参数，当 Mixins 引用这个参数时，该参数表示所有的变量，很多情况下，这个参数可以省去你很多代码。

清单 13. LESS 文件

.boxShadow(@x:0,@y:0,@blur:1px,@color:#000){

-moz-box-shadow: @arguments;

-webkit-box-shadow: @arguments;

box-shadow: @arguments;

}

#header {

.boxShadow(2px,2px,3px,#f36);

}

经过编译生成的 CSS 文件如下：

清单 14. CSS 文件

#header {

-moz-box-shadow: 2px 2px 3px #FF36;

-webkit-box-shadow: 2px 2px 3px #FF36;

box-shadow: 2px 2px 3px #FF36;

}

嵌套的规则

在我们书写标准 CSS 的时候，遇到多层的元素嵌套这种情况时，我们要么采用从外到内的选择器嵌套定义，要么采用给特定元素加 CLASS 或 ID 的方式。

清单 16. HTML 片段

<div id="home">

<div id="top">top</div>

<div id="center">

<div id="left">left</div>

<div id="right">right</div>

</div>

</div>

清单 17. LESS 文件

#home{

color : blue;

width : 600px;

height : 500px;

border:outset;

#top{

border:outset;

width : 90%;

}

#center{

border:outset;

height : 300px;

width : 90%;

#left{

border:outset;

float : left;

width : 40%;

}

#right{

border:outset;

float : left;

width : 40%;

}

}

}

经过编译生成的 CSS 文件如下：

清单 18. CSS 文件

#home {

color: blue;

width: 600px;

height: 500px;

border: outset;

}

#home #top {

border: outset;

width: 90%;

}

#home #center {

border: outset;

height: 300px;

width: 90%;

}

#home #center #left {

border: outset;

float: left;

width: 40%;

}

#home #center #right {

border: outset;

float: left;

width: 40%;

}

LESS 的嵌套规则的写法是 HTML 中的 DOM 结构相对应的，这样使我们的样式表书写更加简洁和更好的可读性。同时，嵌套规则使得对伪元素的操作更为方便。

有 & 时解析的是同一个元素或此元素的伪类，没有 & 解析是后代元素

清单 19. LESS 文件

a {

color: red;

text-decoration: none;

&:hover {// 有 & 时解析的是同一个元素或此元素的伪类，没有 & 解析是后代元素

color: black;

text-decoration: underline;

}

}

经过编译生成的 CSS 文件如下：

清单 20. CSS 文件

a {

color: red;

text-decoration: none;

}

a:hover {

color: black;

text-decoration: underline;

}

运算及函数

在我们的 CSS 中充斥着大量的数值型的 value，比如 color、padding、margin 等，这些数值之间在某些情况下是有着一定关系的，那么我们怎样利用 LESS 来组织我们这些数值之间的关系呢？

清单 21 . LESS 文件

@init: #111111;

@transition: @init\*2;

.switchColor {

color: @transition;

}

经过编译生成的 CSS 文件如下：

清单 22. CSS 文件

.switchColor {

color: #222222;

}

例子中使用 LESS 的 operation 是 特性，其实简单的讲，就是对数值型的 value（数字、颜色、变量等）进行加减乘除四则运算。同时 LESS 还有一个专门针对 color 的操作提供一组函数

Comments（注释）

适当的注释是保证代码可读性的必要手段，LESS 对注释也提供了支持，主要有两种方式：单行注释和多行注释，这与 JavaScript 中的注释方法一样，我们这里不做详细的说明，只强调一点：LESS 中单行注释 (// 单行注释 ) 是不能显示在编译后的 CSS 中，所以如果你的注释是针对样式说明的请使用多行注释。

第8单元：CSS3选择器(上)

属性选择器

CSS2中

E[foo]

选择匹配E的元素,且该元素定义了foo属性

E[foo=”bar”]

选择匹配E的元素,且改元素定义了foo属性值为bar

E[foo~=”bar”]

匹配E的元素且改元素定义了foo属性,属性值是一个以空格符分割的列表,其中一个列表值为bar

E[foo|=”bar”]

匹配E的元素且该元素定义了foo属性,属性值是一个以连字符分割的列表,其中一个列表值为bar

CSS3中：

E[foo^=”bar”]

匹配E的元素且改元素定义了foo属性,属性值是一个包含前缀为”bar”的子字符串

E[foo$=”bar”]

匹配E的元素且改元素定义了foo属性,属性值是一个包含后缀为”bar”的子字符串

E[foo\*=”bar”]

匹配E的元素且改元素定义了foo属性,属性值是一个包含”bar”的子字符串

结构伪类选择器(重点)

E:nth-child(n)

选择所有在其父元素中第n个位置的匹配E的子元素

E:nth-last-child(n)

E:nth-of-type(n)

E:nth-last-of-type(n)

第9单元：CSS3选择器(下)

使用选择器在页面上插入内容

在文本前插入东西

H2:before{content:”2016-5-24”；background：red;}在h2标签内容前添加时间并给他背景颜色

h2:nth-child(3):before{content:none;}如果none浏览器不认识用normal 后面插入用：after，当插入图片的时候用content:url(“./img/icon\_01.jpg”);就可以

插入计数器

Counter 计数器 counter-increment计数器的增量，让其自增长

H2:before{content:”第”counter(abc)”章”；}再写一个

H2{counter-increment:abc;}

这是阿拉伯数字的

要修改成其他的H2:before{content: counter(abc, 其他文字比分说罗马，英文字母啥的)；}

再加上H2{counter-increment:abc;}让它自增长

当想第一个大标题下面出现1-1 1-2 1-3 第二个大标题下面的是2-1 2-1时

li:before{content: "第"counter(z)"章";}

li{counter-reset: c; 这里是让p标签的自增量重置，重置后li下面的小标题就成了li的大标题----c（每次都从1开始）counter-increment: z;}

p:before{content: counter(z)"-"counter(c)".";}

p{counter-increment: c;}

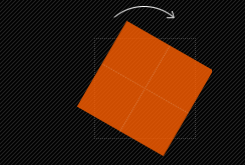
：first-line 第一行

：first-letter第一个字

第10单元：CSS3变形

在CSS3中，可以利用transform功能来实现文字或图像的旋转、缩放、倾斜、移动这四种类型的变形处理

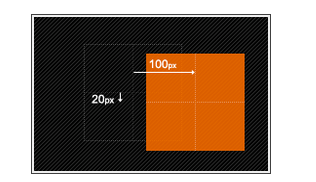
一、旋转rotate

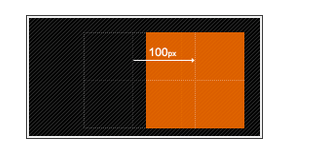
rotate(<angle>) ：通过指定的角度参数对原元素指定一个[2D rotation](http://www.w3.org/TR/SVG/coords.html%23RotationDefined)（2D 旋转），需先有transform-origin属性的定义。transform-origin定义的是旋转的基点，其中angle是指旋转角度，如果设置的值为正数表示顺时针旋转，如果设置的值为负数，则表示逆时针旋转。如：transform:rotate(30deg):

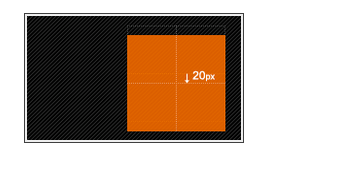
二、移动translate

移动translate我们分为三种情况：translate(x,y)水平方向和垂直方向同时移动（也就是X轴和Y轴同时移动）；translateX(x)仅水平方向移动（X轴移动）；translateY(Y)仅垂直方向移动（Y轴移动），具体使用方法如下：

1、translate(<translation-value>[, <translation-value>]) ：通过矢量[tx, ty]指定一个[2D translation](http://www.w3.org/TR/SVG/coords.html%23TranslationDefined)，tx 是第一个过渡值参数，ty 是第二个过渡值参数选项。如果 未被提供，则ty以 0 作为其值。也就是translate(x,y),它表示对象进行平移，按照设定的x,y参数值,当值为负数时，反方向移动物体，其基点默认为元素 中心点，也可以根据transform-origin进行改变基点。如transform:translate(100px,20px):



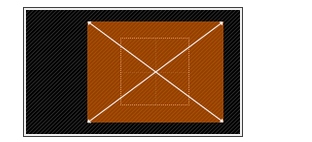
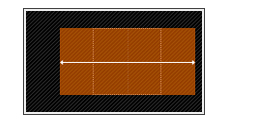
translateX(<translation-value>) ： 通过给定一个X方向上的数目指定一个[translation](http://www.w3.org/TR/SVG/coords.html%23TranslationDefined)。只向x轴进行移动元素，同样其基点是元素中心点，也可以根据transform-origin改变基点位置。如：transform:translateX(100px)

translateY(<translation-value>) ：通过给定Y方向的数目指定一个[translation](http://www.w3.org/TR/SVG/coords.html%23TranslationDefined)。只向Y轴进行移动，基点在元素心点，可以通过transform-origin改变基点位置。如：transform:translateY(20px)

三、缩放scale

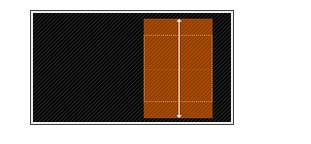
缩放scale和移动translate是极其相似，他也具有三种情况：scale(x,y)使元素水平方向和垂直方向同时缩放（也就是X轴和Y轴同时缩放）；scaleX(x)元素仅水平方向缩放（X轴缩放）；scaleY(y)元素仅垂直方向缩放（Y轴缩放），但它们具有相同的缩放中心点和基数，其中心点就是元素的中心位置，缩放基数为1，如果其值大于1元素就放大，反之其值小于1，元素缩小。下面我们具体来看看这三种情况具体使用方法：

1、scale(<number>[, <number>])：提供执行[sx,sy]缩放矢量的两个参数指定一个[2D scale](http://www.w3.org/TR/SVG/coords.html%23ScalingDefined)（2D缩放）。如果第二个参数未提供，则取与第一个参数一样的值。scale(X,Y)是用于对元素进行缩放，可以通过transform-origin对元素的基点进行设置，同样基点在元素中心位置；基中X表示水平方向缩放的倍数，Y表示垂直方向的缩放倍数，而Y是一个可选参数，如果没有设置Y值，则表示X，Y两个方向的缩放倍数是一样的。并以X为准。如：transform:scale(2,1.5):



2、scaleX(<number>) ： 使用 [sx,1] 缩放矢量执行缩放操作，sx为所需参数。scaleX表示元素只在X轴(水平方向)缩放元素，他的默认值是(1,1)，其基点一样是在元素的中心位置，我们同样是通过transform-origin来改变元素的基点。如：transform:scaleX(2)

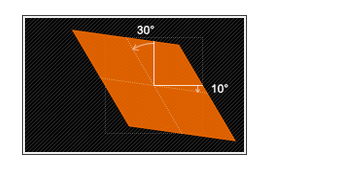
3、scaleY(<number>) ： 使用 [1,sy] 缩放矢量执行缩放操作，sy为所需参数。scaleY表示元素只在Y轴（垂直方向）缩放元素，其基点同样是在元素中心位置，可以通过transform-origin来改变元素的基点。如transform:scaleY(2):著作权归作者所有



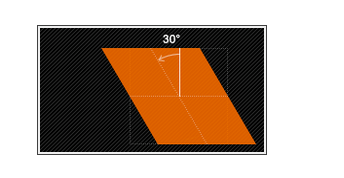
四、扭曲skew

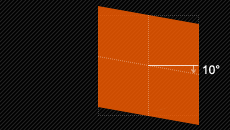
扭曲skew和translate、scale一样同样具有三种情况：skew(x,y)使元素在水平和垂直方向同时扭曲（X轴和Y轴同时按一定的角度值进行扭曲变形）；skewX(x)仅使元素在水平方向扭曲变形（X轴扭曲变形）；skewY(y)仅使元素在垂直方向扭曲变形（Y轴扭曲变形），具体使用如下：

1、skew(<angle> [, <angle>]) ：X轴Y轴上的[skew transformation](http://www.w3.org/TR/SVG/coords.html%23SkewXDefined)（斜切变换）。第一个参数对应X轴，第二个参数对应Y轴。如果第二个参数未提供，则值为0，也就是Y轴方向上无斜切。skew是用来对元素进行扭曲变行，第一个参数是水平方向扭曲角度，第二个参数是垂直方向扭曲角度。其中第二个参数是可选参数，如果没有设置第二个参数，那么Y轴为0deg。同样是以元素中心为基点，我们也可以通过transform-origin来改变元素的基点位置。如：transform:skew(30deg,10deg):



1. skewX(<angle>) ： 按给定的角度沿X轴指定一个[skew transformation](http://www.w3.org/TR/SVG/coords.html%23SkewXDefined)（斜切变换）。skewX是使元素以其中心为基点，并在水平方向（X轴）进行扭曲变行，同样可以通过transform-origin来改变元素的基点。如：transform:skewX(30deg)



3、skewY(<angle>) ： 按给定的角度沿Y轴指定一个[skew transformation](http://www.w3.org/TR/SVG/coords.html%23SkewYDefined)（斜切变换）。skewY是用来设置元素以其中心为基点并按给定的角度在垂直方向（Y轴）扭曲变形。同样我们可以通过transform-origin来改变元素的基点。如：transform:skewY（10deg）

五、矩阵matrix

matrix(<number>, <number>, <number>, <number>, <number>, <number>) ： 以一个含六值的(a,b,c,d,e,f)[变换矩阵](http://www.w3.org/TR/SVG/coords.html%23TransformMatrixDefined)的形式指定一个2D变换，相当于直接应用一个[a b c d e f]变换矩阵。就是基于水平方向（X轴）和垂直方向（Y轴）重新定位元素,此属性值使用涉及到数学中的矩阵，

六、改变元素基点transform-origin

前面我们多次提到transform-origin这个东东，他的主要作用就是让我们在进行transform动作之前可以改变元素的基点位置，因为我们元素默认基点就是其中心位置，换句话说我们没有使用transform-origin改变元素基点位置的情况下，transform进行的rotate,translate,scale,skew,matrix等操作都是以元素自己中心位置进行变化的。但有时候我们需要在不同的位置对元素进行这些操作，那么我们就可以使用transform-origin来对元素进行基点位置改变，使元素基点不在是中心位置，以达到你需要的基点位置。下面我们主要来看看其使用规则：

transform-origin(X,Y):用来设置元素的运动的基点（参照点）。默认点是元素的中心点。其中X和Y的值可以是百分值,em,px，其中X也可以是字符参数值left,center,right；Y和X一样除了百分值外还可以设置字符值top,center,bottom，这个看上去有点像我们background-position设置一样；下面我列出他们相对应的写法：

1、top left | left top 等价于 0 0 | 0% 0%

2、top | top center | center top 等价于 50% 0

3、right top | top right 等价于 100% 0

4、left | left center | center left 等价于 0 50% | 0% 50%

5、center | center center 等价于 50% 50%（默认值）

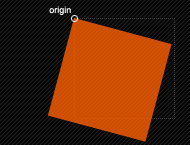
6、right | right center | center right 等价于 100% 50%

7、bottom left | left bottom 等价于 0 100% | 0% 100%

8、bottom | bottom center | center bottom 等价于 50% 100%

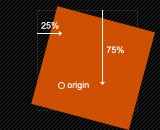
9、bottom right | right bottom 等价于 100% 100%

其中 left,center right是水平方向取值，对应的百分值为left=0%;center=50%;right=100%而top center bottom是垂直方向的取值，其中top=0%;center=50%;bottom=100%;如果只取一个值，表示垂直方向值不变，我们分别来看看以下几个实例

(1)transform-origin:(left,top):

(2)transform-origin:right

(3)transform-origin(25%,75%)



transform-origin并不是transform中的属性值，他具有自己的语法，前面我也说过了，说简单一点就是类似于我们的background-position的用法，但又有其不一样，因为我们background-position不需要区别浏览器内核不同的写法，但transform-origin跟其他的css3属性一样，我们需要在不同的浏览内核中加上相应的前缀，下面列出各种浏览器内核下的语法规则

//Mozilla内核浏览器：firefox3.5+

-moz-transform-origin: x y;

//Webkit内核浏览器：Safari and Chrome

-webkit-transform-origin: x y;

//Opera

-o-transform-origin: x y ;

//IE9

-ms-transform-origin: x y;

//W3C标准

transform-origin: x y ;

transform在不同浏览器内核下的书写规则

//Mozilla内核浏览器：firefox3.5+

-moz-transform: rotate | scale | skew | translate ;

//Webkit内核浏览器：Safari and Chrome

-webkit-transform: rotate | scale | skew | translate ;

//Opera

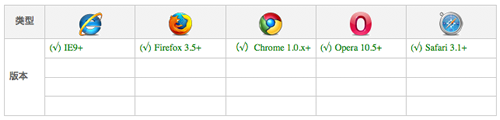
-o-transform: rotate | scale | skew | translate ;

//IE9

-ms-transform: rotate | scale | skew | translate ;

//W3C标准

transform: rotate | scale | skew | translate ;

支持transform浏览器

第11单元：CSS3动画

CSS Transition

1.1 基本用法

在CSS 3引入Transition（过渡）这个概念之前，CSS是没有时间轴的。也就是说，所有的状态变化，都是即时完成。

img{

height:15px;

width:15px;

}

img:hover{

height: 450px;

width: 450px;

}

当鼠标放置于缩略图之上，缩略图会迅速变大。注意，缩略图的变大是瞬间实现的。

transition的作用在于，指定状态变化所需要的时间。

img{

transition: 1s;

}

图片放大的过程需要1秒

我们还可以指定transition适用的属性，比如只适用于height。

img{

transition: 1s height;

}

这样一来，只有height的变化需要1秒实现，其他变化（主要是width）依然瞬间实现

1.2 transition-delay

在同一行transition语句中，可以分别指定多个属性

img{

transition: 1s height, 1s width;

}

但是，这样一来，height和width的变化是同时进行的，跟不指定它们没有差别，我们希望，让height先发生变化，等结束以后，再让width发生变化。实现这一点就是为width指定一个delay参数。

img{

transition: 1s height, 1s 1s width;

}

width在1秒之后，再开始变化，也就是延迟（delay）1秒

delay的真正意义在于，它指定了动画发生的顺序，使得多个不同的transition可以连在一起，形成复杂效果

1.3 transition-timing-function

transition的状态变化速度（又称timing function），默认不是匀速的，而是逐渐放慢，这叫做ease。

img{

transition: 1s ease;

}

除了ease以外，其他模式还包括

（1）linear：匀速

（2）ease-in：加速

（3）ease-out：减速

（4）cubic-bezier函数：自定义速度模式 (涉及贝塞尔曲线很难算)

1.4 transition的各项属性

transition的完整写法

img{

transition: 1s 1s height ease;

}

这是一个简写形式，可以单独定义成各个属性

img{

transition-property: height;

transition-duration: 1s;

transition-delay: 1s;

transition-timing-function: ease;

}

1.5 transition的使用注意

（1）目前，各大浏览器（包括IE 10）都已经支持无前缀的transition，所以transition已经可以很安全地不加浏览器前缀。

（2）不是所有的CSS属性都支持transition，完整的列表查看[这里](http://oli.jp/2010/css-animatable-properties/)，以及具体的[效果](http://leaverou.github.io/animatable/)。

（3）transition需要明确知道，开始状态和结束状态的具体数值，才能计算出中间状态。比如，height从0px变化到100px，transition可以算出中间状态。但是，transition没法算出0px到auto的中间状态，也就是说，如果开始或结束的设置是height: auto，那么就不会产生动画效果。类似的情况还有，display: none到block，background: url(foo.jpg)到url(bar.jpg)等等。

1.6 transition的局限

transition的优点在于简单易用，但是它有几个很大的局限。

（1）transition需要事件触发，所以没法在网页加载时自动发生。

（2）transition是一次性的，不能重复发生，除非一再触发。

（3）transition只能定义开始状态和结束状态，不能定义中间状态，也就是说只有两个状态。

（4）一条transition规则，只能定义一个属性的变化，不能涉及多个属性。

CSS Animation就是为了解决这些问题而提出的。

CSS Animation

基本用法

CSS Animation需要指定动画一个周期持续的时间，以及动画效果的名称。

div:hover {

animation: 1s rainbow;

}

当鼠标悬停在div元素上时，会产生名为rainbow的动画效果，持续时间为1秒。为此，我们还需要用keyframes关键字，定义rainbow效果。

@keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }

}

rainbow效果一共有三个状态，分别为起始（0%）、中点（50%）和结束（100%）。如果有需要，完全可以插入更多状态。

1.animation-iteration-count

默认情况下，动画只播放一次。加入infinite关键字，可以让动画无限次播放

div:hover {

animation: 1s rainbow infinite;

}

也可以指定动画具体播放的次数，比如3次

div:hover {

animation: 1s rainbow 3;

}

2 animation-fill-mode

动画结束以后，会立即从结束状态跳回到起始状态。如果想让动画保持在结束状态，需要使用animation-fill-mode属性。

div:hover {

animation: 1s rainbow forwards;

}

forwards表示让动画停留在结束状态

animation-fill-mode还可以使用下列值

（1）none：默认值，回到动画没开始时的状态。

（2）backwards：让动画回到第一帧的状态。

（3）both: 根据animation-direction（见后）轮流应用forwards和backwards规则。

3 animation-direction

动画循环播放时，每次都是从结束状态跳回到起始状态，再开始播放。animation-direction属性，可以改变这种行为。

@keyframes rainbow {

0% { background: yellow; }

100% { background: blue; }

}

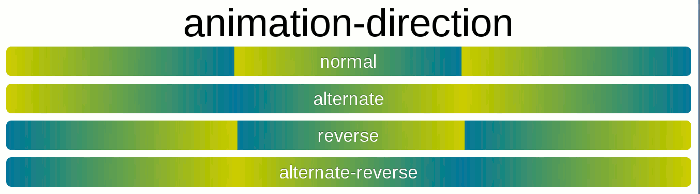
默认情况是，animation-direction等于normal。

div:hover {

animation: 1s rainbow 3 normal;

}

此外，还可以等于取alternate、reverse、alternate-reverse等值。它们的含义见下图（假定动画连续播放三次）。



nimation-direction指定了动画播放的方向，最常用的值是normal和reverse。

4 animation的各项属性

同transition一样，animation也是一个简写形式。

div:hover {

animation: 1s 1s rainbow linear 3 forwards normal;}

分解成各个单独的属性。

div:hover {

animation-name: rainbow;

animation-duration: 1s;

animation-timing-function: linear;

animation-delay: 1s;

animation-fill-mode:forwards;

animation-direction: normal;

animation-iteration-count: 3;

}

5 keyframes的写法

keyframes关键字用来定义动画的各个状态

@keyframes rainbow {

0% { background: #c00 }

50% { background: orange }

100% { background: yellowgreen }

}

0%可以用from代表，100%可以用to代表

@keyframes rainbow {

from { background: #c00 }

50% { background: orange }

to { background: yellowgreen }

}

如果省略某个状态，浏览器会自动推算中间状态

@keyframes rainbow {

50% { background: orange }

to { background: yellowgreen }

}

@keyframes rainbow {

to { background: yellowgreen }

}

可以把多个状态写在一行

@keyframes pound {

from，to { transform: none; }

50% { transform: scale(1.2); }

}

需要注意的是，浏览器从一个状态向另一个状态过渡，是平滑过渡。steps函数可以实现分步过渡。

div:hover {

animation: 1s rainbow infinite steps(10);}

6 animation-play-state

有时，动画播放过程中，会突然停止。这时，默认行为是跳回到动画的开始状态。

如果想让动画保持突然终止时的状态，就要使用animation-play-state属性。

div {

animation: spin 1s linear infinite;

animation-play-state: paused;}

div:hover {

animation-play-state: running;}

没有鼠标没有悬停时，动画状态是暂停；一旦悬停，动画状态改为继续播放。

7 浏览器前缀

目前，IE 10和Firefox（>= 16）支持没有前缀的animation所以必须使用webkit前缀

div:hover {

-webkit-animation: 1s rainbow;

animation: 1s rainbow; }

@-webkit-keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }}

@keyframes rainbow {

0% { background: #c00; }

50% { background: orange; }

100% { background: yellowgreen; }}

第12单元：媒体查询

https://blog.csdn.net/tomato\_\_/article/details/51306833

媒体查询的使用方法

@media [only] 媒体类型 and/or/not （媒体特性）{你的样式}

注意：使用Media Queries必须要使用“@media”开头，然后指定媒体类型（也可以称为设备类型），随后是指定媒体特性（也可以称之为设备特性）。媒体特性的书写方式和样式的书写方式非常相似，主要分为两个部分，第一个部分指的是媒体特性，第二部分为媒体特性所指定的值，而且这两个部分之间使用冒号分隔。例如：

(max-width: 480px)  
从前面表中可以得知，主要有十种媒体类型和13种媒体特性，将其组合就类似于不同的CSS集合。  
但与CSS属性不同的是，媒体特性是通过min/max来表示大于等于或小于做为逻辑判断，  
而不是使用小于（<）和大于（>）这样的符号来判断。接下来一起来看看Media Queries在实际项目中常用的方式。

**1. 最大宽度max-width**

“max-width”是媒体特性中最常用的一个特性，其意思是指媒体类型小于或等于指定的宽度时，样式生效。如：

@media screen and (max-width:480px){

.ads {

display:none;

}

}

上面表示的是：当屏幕小于或等于480px时,页面中的广告区块（.ads）都将被隐藏。

**2.最小宽度min-width**

“min-width”与“max-width”相反，指的是媒体类型大于或等于指定宽度时，样式生效。

@media screen and (min-width:900px){

.wrapper{width: 980px;}

}

上面表示的是：当屏幕大于或等于900px时，容器“.wrapper”的宽度为980px。

**3.多个媒体特性使用**

Media Queries可以使用关键词"and"将多个媒体特性结合在一起。也就是说，一个Media Query中可以包含0到多个表达式，表达式又可以包含0到多个关键字，以及一种媒体类型。

当屏幕在600px~900px之间时，body的背景色渲染为“#f5f5f5”，如下所示。

@media screen and (min-width:600px) and (max-width:900px){

body {background-color:#f5f5f5;}

}

**4.设备屏幕的输出宽度Device Width**

在智能设备上，例如iPhone、iPad等，还可以根据屏幕设备的尺寸来设置相应的样式（或者调用相应的样式文件）。同样的，对于屏幕设备同样可以使用“min/max”对应参数，如“min-device-width”或者“max-device-width”。

<link rel="stylesheet" media="screen and (max-device-width:480px)" href="iphone.css" />

上面的代码指的是“iphone.css”样式适用于最大设备宽度为480px，比如说iPhone上的显示，这里的“max-device-width”所指的是设备的实际分辨率，也就是指可视面积分辨率。

**5. not关键词**

使用关键词“not”是用来排除某种制定的媒体类型，也就是用来排除符合表达式的设备。换句话说，not关键词表示对后面的表达式执行取反操作，如：

@media not print and (max-width: 1200px){样式代码}

上面代码表示的是：样式代码将被使用在除打印设备和设备宽度小于1200px下所有设备中。

**6.only关键词 http://www.mamicode.com/info-detail-1726835.html**

only用来指定某种特定的媒体类型，可以用来排除不支持媒体查询的浏览器。其实only很多时候是用来对那些不支持Media Query但却支持Media Type的设备隐藏样式表的。其主要有：支持媒体特性的设备，正常调用样式，此时就当only不存在；表示不支持媒体特性但又支持媒体类型的设备，这样就会不读样式，因为其先会读取only而不是screen；另外不支持Media Queries的浏览器，不论是否支持only，样式都不会被采用。如

<linkrel="stylesheet" media="only screen and (max-device-width:240px)" href="android240.css" />

在Media Query中如果没有明确指定Media Type，那么其默认为all，如：

<linkrel="stylesheet" media="(min-width:701px) and (max-width:900px)" href="mediu.css" />

另外在样式中，还可以使用多条语句来将同一个样式应用于不同的媒体类型和媒体特性中，指定方式如下所示。

<linkrel="stylesheet" type="text/css" href="style.css" media="handheld and (max-width:480px), screen and (min-width:960px)" />

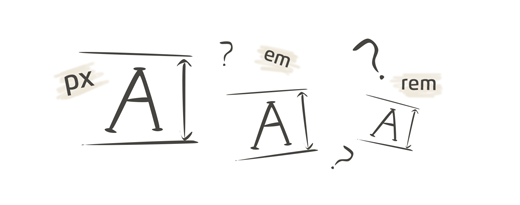
上面代码中style.css样式被用在宽度小于或等于480px的手持设备上，或者被用于屏幕宽度大于或等于960px的设备上。

到目前为止，CSS3 Media Queries得到了众多浏览器支持，除了IE6-8浏览器不支持之外，在所有现代浏览器中都可以完美支持。还有一个与众不同的是，Media Queries在其他浏览器中不要像其他CSS3属性一样在不同的浏览器中添加前缀。

Media所有参数汇总  
以上就是我们最常需要用到的媒体查询器的三个特性，大于，等于，小于的写法。媒体查询器的全部功能肯定不止这三个功能，下面是我总结的它的一些参数用法解释：  
width:浏览器可视宽度。  
height:浏览器可视高度。  
device-width:设备屏幕的宽度。  
device-height:设备屏幕的高度。  
orientation:检测设备目前处于横向还是纵向状态。  
aspect-ratio:检测浏览器可视宽度和高度的比例。(例如：aspect-ratio:16/9)  
device-aspect-ratio:检测设备的宽度和高度的比例。  
color:检测颜色的位数。（例如：min-color:32就会检测设备是否拥有32位颜色）  
color-index:检查设备颜色索引表中的颜色，他的值不能是负数。  
monochrome:检测单色楨缓冲区域中的每个像素的位数。（这个太高级，估计咱很少会用的到）  
resolution:检测屏幕或打印机的分辨率。(例如：min-resolution:300dpi或min-resolution:118dpcm)。  
grid:检测输出的设备是网格的还是位图设备。

px,em,rem使用

我们都知道基于像素的字体大小所用的单位是px，但是随着响应式设计的不断火热，基于相对字体大小的单位em变开始流行起来。当然，rem也在Web前端开发人员讨论如何更好设置字体大小的讨论话题之列。是不是需要用em或者是rem来替换掉px在前端开发中进行使用呢？



什么我们需要使用相对字体大小呢？

或许我们最能直接想到的一点就是它关于CSS3的一些东西。很好理解的是它是一种字号计算方式，尤其是当你在为多种分辨率进行字号计算的时候。想想要在手机上实现字体变大或者变小的时候，你不得不要反复重新计算对象的字体大小，这是相当耗时间的。这个时候相对字体大小的设置才凸显出了它的有用之处。当你要在不同的样式结构中进行设置字体大小的时候，使用em或者rem能够让你针对一个具体的视屏进行字体大小的增大或者减小。

px，em，rem的区别在哪？

Px是一个绝对字体大小，em则是基于基数（比如：1.5em）来计算出来的相对字体大小。这个基数是需要乘以当前对象从其父级遗传字体大小。

|  |  |
| --- | --- |
| 1 | 1.5em = 父级字体大小 \* 1.5 = ? |

不过，这有个问题就是在css中想要知道当前元素的字体大小所遗传的是哪个父级元素的字号。因此，rem的出现就很好的解决了这个问题。rem是基于根节点（比如html）的字体大小进行计算的。

|  |  |
| --- | --- |
| 1 | 1.5rem = 默认字体大小 (比如： 16px) \* 1.5 = 24px |

这个默认字体大小是依据你网站当前访问时所使用的浏览器或者其他设备来决定的，对于桌面浏览器默认是16px的字体大小。然后你现在要转换当前元素的字体大小为rem的时候你就可以这样做：

|  |  |
| --- | --- |
| 1 | 28px = 28/16 = 1.75rem |

为了更方便的进行计算转换，你可以把默认字体大小设置成62.5%或者是10px，这个时候你要计算当前元素字体大小的时候，你就可以这样：

|  |  |
| --- | --- |
| 1 | 28px = 28/10 = 2.8rem |

rem的浏览器支持

目前大多数浏览器都支持rem单位的字体大小，不过，要想兼容IE7和IE8，你还是需要使用px来做单位。写法如下：

|  |  |
| --- | --- |
| 1 | html { font-size: 62.5%; } |

|  |  |
| --- | --- |
| 1 | p { font-size: 15px; font-size: 1.5rem; } /\* =15px \*/ |

css中的px、em、rem 详解

概念介绍：

1、px (pixel，像素)：是一个虚拟长度单位，是计算机系统的数字化图像长度单位，如果px要换算成物理长度，需要指定精度DPI(Dots Per Inch，每英寸像素数)，在扫描打印时一般都有DPI可选。Windows系统默认是96dpi，Apple系统默认是72dpi。

2、em(相对长度单位，相对于当前对象内文本的字体尺寸)：是一个相对长度单位，最初是指字母M的宽度，故名em。现指的是字符宽度的倍数，用法类似百分比，如：0.8em, 1.2em,2em等。通常1em=16px。

3、pt (point，磅)：是一个物理长度单位，指的是72分之一英寸。pt=1/72(英寸),

4、rem（root em，根em）：是CSS3新增的一个相对单位，这个单位引起了广泛关注。这个单位与em有什么区别呢？区别在于使用rem为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8及更早版本外，所有浏览器均已支持rem。对于不支持它的浏览器，应对方法也很简单，就是多写一个绝对单位的声明。这些浏览器会忽略用rem设定的字体大小。

第13单元：弹性盒布局(上)

网页布局（layout）是 CSS 的一个重点应用。

布局的传统解决方案，基于[盒状模型](https://developer.mozilla.org/en-US/docs/Web/CSS/box_model)，依赖 [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display) 属性 + [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)属性 + [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float)属性。它对于那些特殊布局非常不方便，比如，[垂直居中](https://css-tricks.com/centering-css-complete-guide/)就不容易实现。

2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

浏览器支持

一、Flex 布局是什么？

Flex 是 Flexible Box 的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。

任何一个容器都可以指定为 Flex 布局。

.box{

display: flex;}

行内元素也可以使用 Flex 布局。

.box{

display: inline-flex;}

Webkit 内核的浏览器，必须加上-webkit前缀。

.box{

display: -webkit-flex; /\* Safari \*/

display: flex;}

注意，设为 Flex 布局以后，子元素的float、clear和vertical-align属性将失效。

二、基本概念

采用 Flex 布局的元素，称为 Flex 容器（flex container），简称"容器"。它的所有子元素自动成为容器成员，称为 Flex 项目（flex item），简称"项目"。



容器默认存在两根轴：水平的主轴（main axis）和垂直的交叉轴（cross axis）。主轴的开始位置（与边框的交叉点）叫做main start，结束位置叫做main end；交叉轴的开始位置叫做cross start，结束位置叫做cross end。

项目默认沿主轴排列。单个项目占据的主轴空间叫做main size，占据的交叉轴空间叫做cross size。

三、容器的属性

以下6个属性设置在容器上。

flex-direction

flex-wrap

flex-flow

justify-content

align-items

align-content

3.1 flex-direction属性

flex-direction属性决定主轴的方向（即项目的排列方向）。

.box {

flex-direction: row | row-reverse | column | column-reverse;}



它可能有4个值。

row（默认值）：主轴为水平方向，起点在左端。

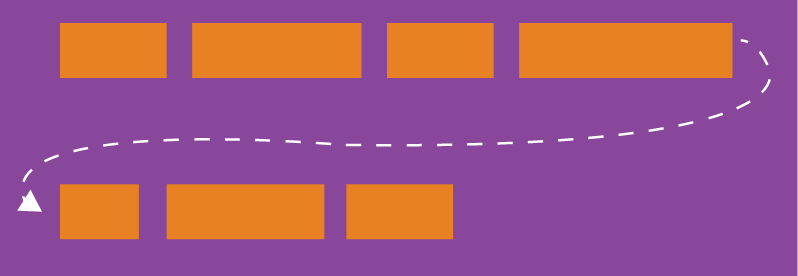
row-reverse：主轴为水平方向，起点在右端。

column：主轴为垂直方向，起点在上沿。

column-reverse：主轴为垂直方向，起点在下沿。

3.2 flex-wrap属性

默认情况下，项目都排在一条线（又称"轴线"）上。flex-wrap属性定义，如果一条轴线排不下，如何换行。



.box{

flex-wrap: nowrap | wrap | wrap-reverse;}

它可能取三个值。

（1）nowrap（默认）：不换行

（2）wrap：换行，第一行在上方。

（3）wrap-reverse：换行，第一行在下方。

3.3 flex-flow

flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。

.box {

flex-flow: <flex-direction> || <flex-wrap>;}

3.4 justify-content属性

justify-content属性定义了项目在主轴上的对齐方式。

.box {

justify-content: flex-start | flex-end | center | space-between | space-around;}

它可能取5个值，具体对齐方式与轴的方向有关。下面假设主轴为从左到右。

flex-start（默认值）：左对齐

flex-end：右对齐

center： 居中

space-between：两端对齐，项目之间的间隔都相等。

space-around：每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

3.5 align-items属性

align-items属性定义项目在交叉轴上如何对齐。

.box {

align-items: flex-start | flex-end | center | baseline | stretch;}

它可能取5个值。具体的对齐方式与交叉轴的方向有关，下面假设交叉轴从上到下。

flex-start：交叉轴的起点对齐。

flex-end：交叉轴的终点对齐。

center：交叉轴的中点对齐。

baseline: 项目的第一行文字的基线对齐。

stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。

3.6 align-content属性

align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

.box {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;}

该属性可能取6个值。

flex-start：与交叉轴的起点对齐。

flex-end：与交叉轴的终点对齐。

center：与交叉轴的中点对齐。

space-between：与交叉轴两端对齐，轴线之间的间隔平均分布。

space-around：每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。

stretch（默认值）：轴线占满整个交叉轴。

第14单元：弹性盒布局(下)

项目的属性

以下6个属性设置在项目上。

order

flex-grow

flex-shrink

flex-basis

flex

align-self

4.1 order属性

order属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。

.item {

order: <integer>;}



4.2 flex-grow属性

flex-grow属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。

.item {

flex-grow: <number>; /\* default 0 \*/}



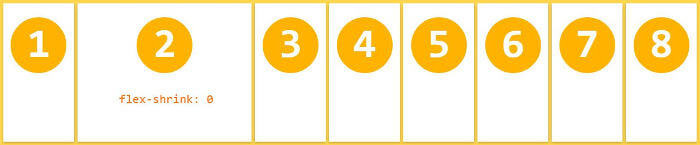
如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。

4.3 flex-shrink属性

flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。

.item {

flex-shrink: <number>; /\* default 1 \*/}



如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

负值对该属性无效。

4.4 flex-basis属性

flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。浏览器根据这个属性，计算主轴是否有多余空间。它的默认值为auto，即项目的本来大小。

.item {

flex-basis: <length> | auto; /\* default auto \*/}

它可以设为跟width或height属性一样的值（比如350px），则项目将占据固定空间。

4.5 flex属性

flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]}

该属性有两个快捷值：auto (1 1 auto) 和 none (0 0 auto)。

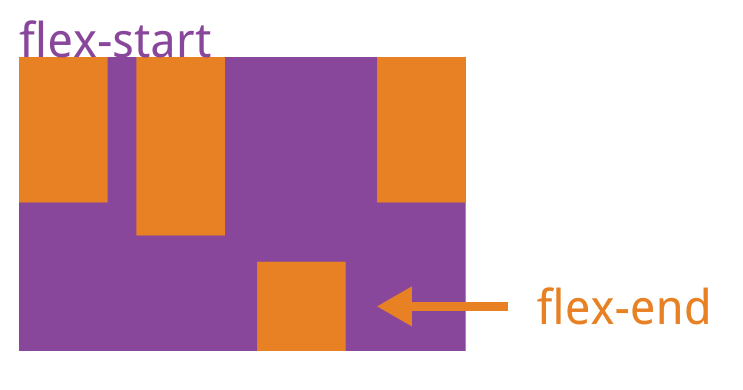
建议优先使用这个属性，而不是单独写三个分离的属性，因为浏览器会推算相关值。

4.6 align-self属性

align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

.item {

align-self: auto | flex-start | flex-end | center | baseline | stretch;}

该属性可能取6个值，除了auto，其他都与align-items属性完全一致。

# 第15单元：Bootstrap全局css样式

http://www.bootcss.com/

15.1 什么是 bootstrap？

Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目。

15.2 bootstrap的好处。

## 简洁、直观、强悍的前端开发框架，让web开发更迅速、简单。

Bootstrap 编码规范：编写灵活、稳定、高质量的 HTML 和 CSS 代码的规范。

15.3 bootstrap的版本。

[Bootstrap2中文文档(v2.3.2)](http://v2.bootcss.com/)

[Bootstrap3中文文档(v3.3.7)](http://v2.bootcss.com/)

v4.0.0

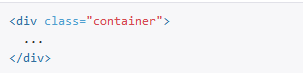
15.4全局的css样式

https://v3.bootcss.com/css/

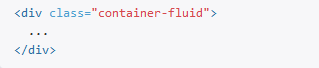
15.4.1布局容器

Bootstrap 需要为页面内容和栅格系统包裹一个 .container 容器。我们提供了两个作此用处的类。注意，由于 padding 等属性的原因，这两种 容器类不能互相嵌套。

.container 类用于固定宽度并支持响应式布局的容器。



.container-fluid 类用于 100% 宽度，占据全部视口（viewport）的容器。



15.4.2栅格系统

栅格系统用于通过一系列的行（row）与列（column）的组合来创建页面布局，你的内容就可以放入这些创建好的布局中。下面就介绍一下 Bootstrap 栅格系统的工作原理：

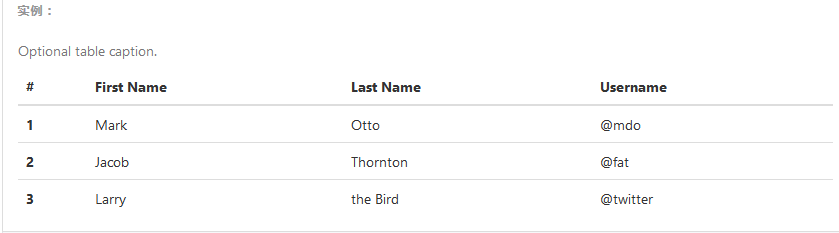
* “行（row）”必须包含在 .container （固定宽度）或 .container-fluid （100% 宽度）中，以便为其赋予合适的排列（aligment）和内补（padding）。
* 通过“行（row）”在水平方向创建一组“列（column）”。
* 你的内容应当放置于“列（column）”内，并且，只有“列（column）”可以作为行（row）”的直接子元素。
* 类似 .row 和 .col-xs-4 这种预定义的类，可以用来快速创建栅格布局。Bootstrap 源码中定义的 mixin 也可以用来创建语义化的布局。
* 通过为“列（column）”设置 padding 属性，从而创建列与列之间的间隔（gutter）。通过为 .row 元素设置负值 margin 从而抵消掉为 .container 元素设置的 padding，也就间接为“行（row）”所包含的“列（column）”抵消掉了padding。
* 栅格系统中的列是通过指定1到12的值来表示其跨越的范围。例如，三个等宽的列可以使用三个 .col-xs-4 来创建。
* 如果一“行（row）”中包含了的“列（column）”大于 12，多余的“列（column）”所在的元素将被作为一个整体另起一行排列。

## 栅格参数



15.4.3表格

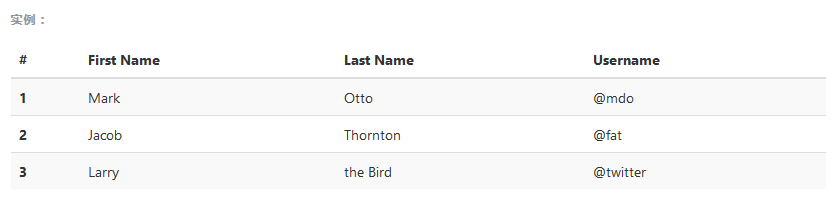
为任意 <table> 标签添加 .table 类可以为其赋予基本的样式 — 少量的内补（padding）和水平方向的分隔线。

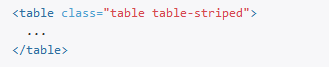




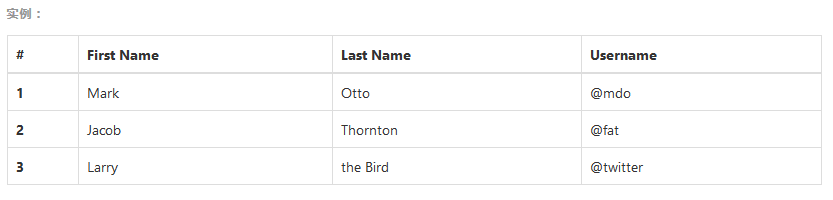
通过 .table-striped 类可以给 <tbody> 之内的每一行增加斑马条纹样式。通过 .table-striped 类可以给 <tbody> 之内的每一行增加斑马条纹样式。

条纹状表格是依赖 :nth-child CSS 选择器实现的，而这一功能不被 Internet Explorer 8 支持。



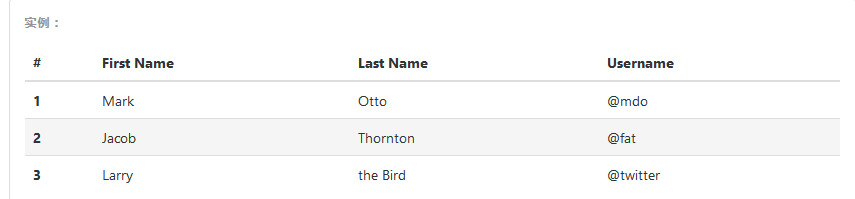


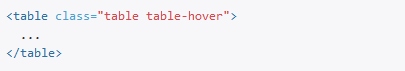
添加 .table-bordered 类为表格和其中的每个单元格增加边框。



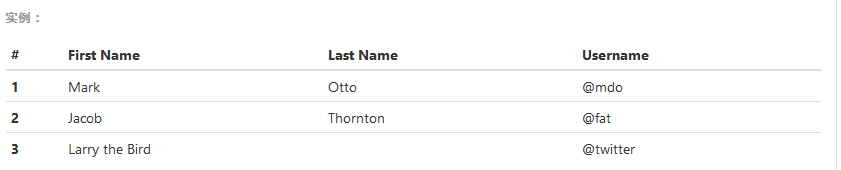
图片 52

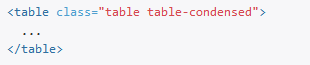
通过添加 .table-hover 类可以让 <tbody> 中的每一行对鼠标悬停状态作出响应。



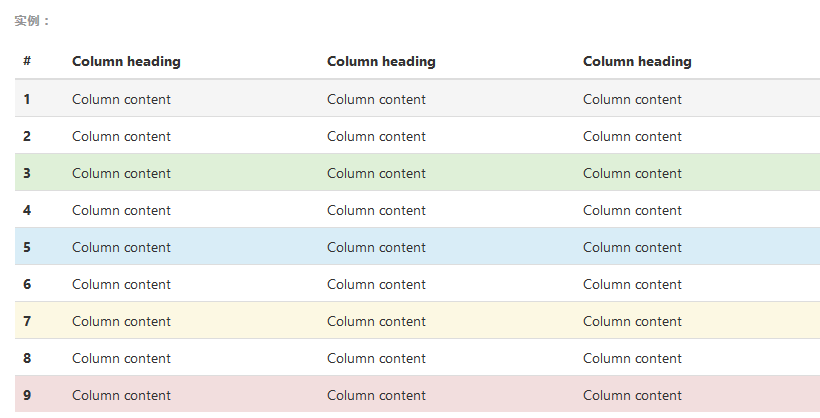


通过添加 .table-condensed 类可以让表格更加紧凑，单元格中的内补（padding）均会减半。



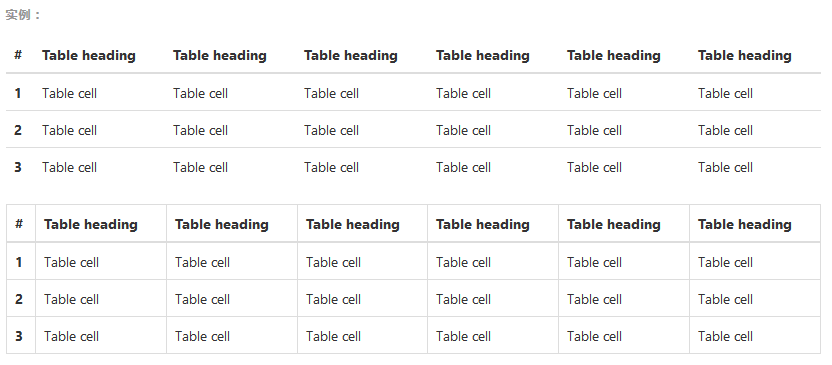


| **Class** | **描述** |
| --- | --- |
| **.active** | 鼠标悬停在行或单元格上时所设置的颜色 |
| **.success** | 标识成功或积极的动作 |
| **.info** | 标识普通的提示信息或动作 |
| **.warning** | 标识警告或需要用户注意 |
| **.danger** | 标识危险或潜在的带来负面影响的动作 |





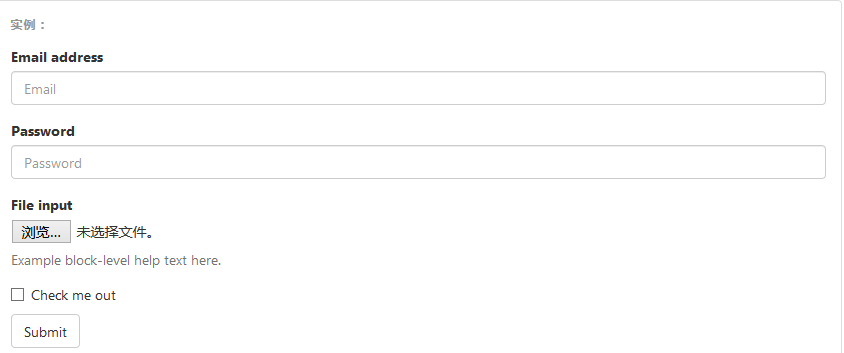
将任何 .table 元素包裹在 .table-responsive 元素内，即可创建响应式表格，其会在小屏幕设备上（小于768px）水平滚动。当屏幕大于 768px 宽度时，水平滚动条消失。





15.4.4表单

单独的表单控件会被自动赋予一些全局样式。所有设置了 .form-control 类的 <input>、<textarea> 和 <select> 元素都将被默认设置宽度属性为 width: 100%;。 将 label 元素和前面提到的控件包裹在 .form-group 中可以获得最好的排列。

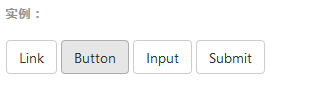


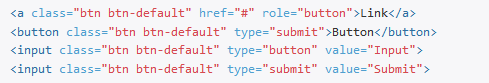


15.5按钮

15.5.1按钮样式

为 <a>、<button> 或 <input> 元素添加按钮类（button class）即可使用 Bootstrap 提供的样式。





虽然按钮类可以应用到 <a> 和 <button> 元素上，但是，导航和导航条组件只支持 <button> 元素。

如果 <a> 元素被作为按钮使用 -- 并用于在当前页面触发某些功能 -- 而不是用于链接其他页面或链接当前页面中的其他部分，那么，务必为其设置 role="button" 属性。

**强烈建议尽可能使用** <button> **元素**来获得在各个浏览器上获得相匹配的绘制效果。

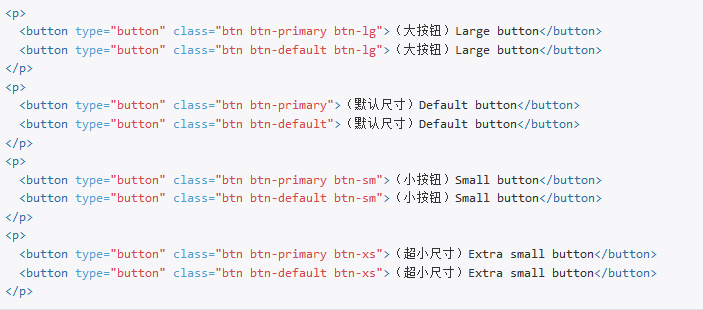
使用下面列出的类可以快速创建一个带有预定义样式的按钮。





需要让按钮具有不同尺寸吗？使用 .btn-lg、.btn-sm 或 .btn-xs 就可以获得不同尺寸的按钮。





通过给按钮添加 .btn-block 类可以将其拉伸至父元素100%的宽度，而且按钮也变为了块级（block）元素。



图片 70

## 激活状态

当按钮处于激活状态时，其表现为被按压下去（底色更深、边框夜色更深、向内投射阴影）。对于 <button> 元素，是通过 :active 状态实现的。对于 <a> 元素，是通过 .active 类实现的。然而，你还可以将 .active 应用到 <button> 上（包含 aria-pressed="true" 属性)），并通过编程的方式使其处于激活状态。

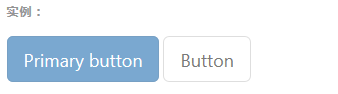
### button 元素

由于 :active 是伪状态，因此无需额外添加，但是在需要让其表现出同样外观的时候可以添加 .active 类。



图片 72

为 <button> 元素添加 disabled 属性，使其表现出禁用状态。



图片 75

如果为 <button> 元素添加 disabled 属性，Internet Explorer 9 及更低版本的浏览器将会把按钮中的文本绘制为灰色

15.6图片

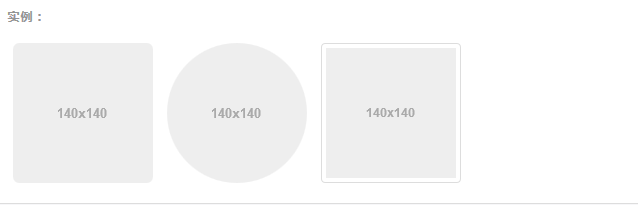
15.6.1响应式图片

在 Bootstrap 版本 3 中，通过为图片添加 .img-responsive 类可以让图片支持响应式布局。其实质是为图片设置了 max-width: 100%;、 height: auto; 和 display: block; 属性，从而让图片在其父元素中更好的缩放。

如果需要让使用了 .img-responsive 类的图片水平居中，请使用 .center-block 类，不要用 .text-center。

15.6.2图片的形状

通过为 <img> 元素添加以下相应的类，可以让图片呈现不同的形状。

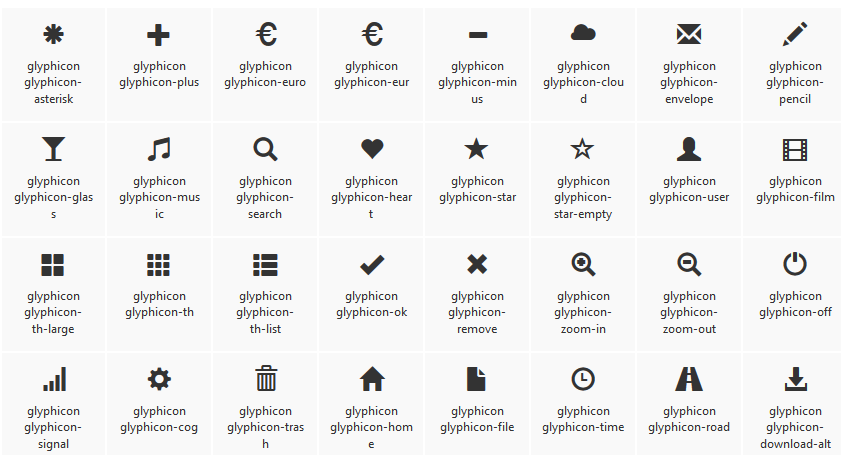


图片 77

# 第16单元：bootstrap组件

16.1 字体图标

# Glyphicons 字体图标



250多个来自 Glyphicon Halflings 的字体图标Bootstrap 免费使用

使用方法

出于性能的考虑，所有图标都需要一个基类和对应每个图标的类。把下面的代码放在任何地方都可以正常使用。注意，为了设置正确的内补（padding），务必在图标和文本之间添加一个空格。

#### 不要和其他组件混合使用

图标类不能和其它组件直接联合使用。它们不能在同一个元素上与其他类共同存在。应该创建一个嵌套的 <span> 标签，并将图标类应用到这个 <span> 标签上。

#### 只对内容为空的元素起作用

图标类只能应用在不包含任何文本内容或子元素的元素上。

#### 改变图标字体文件的位置

Bootstrap 假定所有的图标字体文件全部位于 ../fonts/ 目录内，相对于预编译版 CSS 文件的目录。如果你修改了图标字体文件的位置，那么，你需要通过下面列出的任何一种方式来更新 CSS 文件：

* 在 Less 源码文件中修改 @icon-font-path 和/或 @icon-font-name 变量。
* 利用 Less 编译器提供的 [相对 URL 地址选项](http://lesscss.org/usage/%23command-line-usage-relative-urls)。
* 修改预编译 CSS 文件中的 url() 地址。

#### 图标的可访问性

现代的辅助技术能够识别并朗读由 CSS 生成的内容和特定的 Unicode 字符。为了避免 屏幕识读设备抓取非故意的和可能产生混淆的输出内容（尤其是当图标纯粹作为装饰用途时），我们为这些图标设置了 aria-hidden="true" 属性。

如果你使用图标是为了表达某些含义（不仅仅是为了装饰用），请确保你所要表达的意思能够通过被辅助设备识别，例如，包含额外的内容并通过 .sr-only 类让其在视觉上表现出隐藏的效果。

如果你所创建的组件不包含任何文本内容（例如， <button> 内只包含了一个图标），你应当提供其他的内容来表示这个控件的意图，这样就能让使用辅助设备的用户知道其作用了。这种情况下，你可以为控件添加 aria-label 属性。

<span class="glyphicon glyphicon-search" aria-hidden="true"></span>



<button type="button" class="btn btn-default" aria-label="Left Align">

<span class="glyphicon glyphicon-align-left" aria-hidden="true"></span></button>

<button type="button" class="btn btn-default btn-lg">

<span class="glyphicon glyphicon-star" aria-hidden="true"></span> Star</button>

16.2 导航

16.2.1 导航条

导航条是在您的应用或网站中作为导航页头的响应式基础组件。它们在移动设备上可以折叠（并且可开可关），且在视口（viewport）宽度增加时逐渐变为水平展开模式。

#### 导航条内所包含元素溢出

由于 Bootstrap 并不知道你在导航条内放置的元素需要占据多宽的空间，你可能会遇到导航条中的内容折行的情况（也就是导航条占据两行）。解决办法如下：

1. 减少导航条内所有元素所占据的宽度。
2. 在某些尺寸的屏幕上（利用 [响应式工具类](http://v3.bootcss.com/css/%23responsive-utilities)）隐藏导航条内的一些元素。
3. 修改导航条在水平排列和折叠排列互相转化时，触发这个转化的最小屏幕宽度值。可以通过修改 @grid-float-breakpoint 变量实现，或者自己重写相关的媒体查询代码，覆盖 Bootstrap 的默认值。

#### 依赖 JavaScript 插件

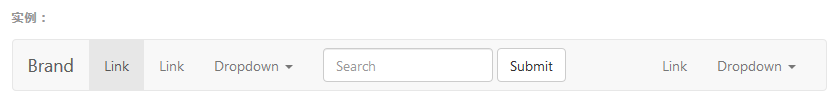
如果 JavaScript 被禁用，并且视口（viewport）足够窄，致使导航条折叠起来，导航条将不能被打开，.navbar-collapse 内所包含的内容也将不可见。

响应式导航条依赖 [collapse 插件](http://v3.bootcss.com/javascript/%23collapse)，定制 Bootstrap 的话时候必将其包含。

#### 修改视口的阈值，从而影响导航条的排列模式

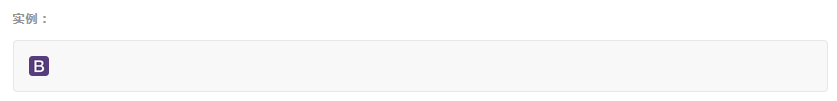
当浏览器视口（viewport）的宽度小于 @grid-float-breakpoint 值时，导航条内部的元素变为折叠排列，也就是变现为移动设备展现模式；当浏览器视口（viewport）的宽度大于 @grid-float-breakpoint 值时，导航条内部的元素变为水平排列，也就是变现为非移动设备展现模式。通过调整源码中的这个值，就可以控制导航条何时堆叠排列，何时水平排列。默认值是 768px （小屏幕 -- 或者说是平板 --的最小值，或者说是平板）。

#### 导航条的可访问性

务必使用 <nav> 元素，或者，如果使用的是通用的 <div> 元素的话，务必为导航条设置 role="navigation" 属性，这样能够让使用辅助设备的用户明确知道这是一个导航区域。



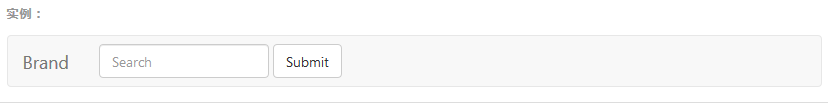
将导航条内放置品牌标志的地方替换为 <img> 元素即可展示自己的品牌图标。由于 .navbar-brand 已经被设置了内补（padding）和高度（height），你需要根据自己的情况添加一些 CSS 代码从而覆盖默认设置。

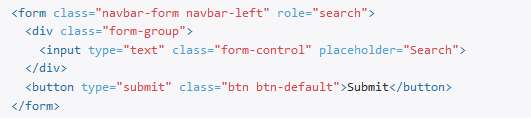




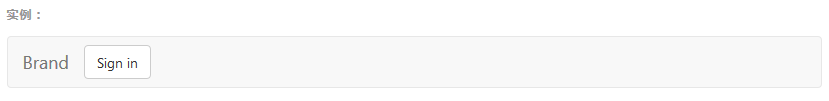
将表单放置于 .navbar-form 之内可以呈现很好的垂直对齐，并在较窄的视口（viewport）中呈现折叠状态。 使用对齐选项可以规定其在导航条上出现的位置。

注意，.navbar-form 和 .form-inline 的大部分代码都一样，内部实现使用了 mixin。 **某些表单组件，例如输入框组，可能需要设置一个固定宽度，从而在导航条内有合适的展现。**

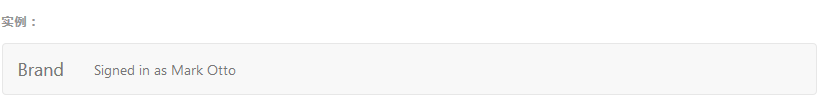




对于不包含在 <form> 中的 <button> 元素，加上 .navbar-btn 后，可以让它在导航条里垂直居中。有一些对于为辅助设备提供可识别标签的方法，例如， aria-label、aria-labelledby 或者 title 属性。如果这些方法都没有，屏幕阅读器将使用 placeholder 属性（如果这个属性存在的话），但是请注意，使用 placeholder 代替其他识别标签的方式是不推荐的。

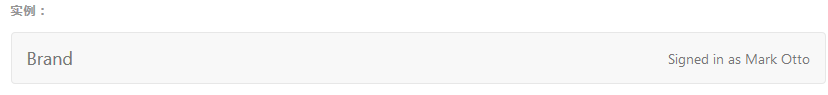


图片 15

把文本包裹在 .navbar-text中时，为了有正确的行距和颜色，通常使用 <p> 标签。

图片 17

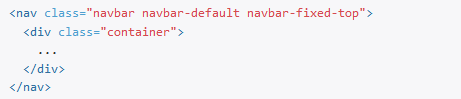
在标准的导航组件之外添加标准链接，那么，使用 .navbar-link 类可以让链接有正确的默认颜色和反色设置。



图片 19

## 固定在顶部

添加 .navbar-fixed-top 类可以让导航条固定在顶部，还可包含一个 .container 或 .container-fluid 容器，从而让导航条居中，并在两侧添加内补（padding）。



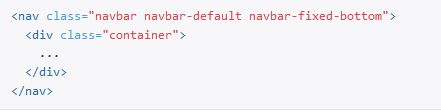
#### 需要为 body 元素设置内补（padding）

这个固定的导航条会遮住页面上的其它内容，除非你给 <body> 元素底部设置了 padding。用你自己的值，或用下面给出的代码都可以。提示：导航条的默认高度是 50px。

body { padding-top: 70px; }

## 固定在底部

添加 .navbar-fixed-bottom 类可以让导航条固定在底部，并且还可以包含一个 .container 或 .container-fluid 容器，从而让导航条居中，并在两侧添加内补（padding）。



## 静止在顶部

通过添加 .navbar-static-top 类即可创建一个与页面等宽度的导航条，它会随着页面向下滚动而消失。还可以包含一个 .container 或 .container-fluid 容器，用于将导航条居中对齐并在两侧添加内补（padding）。

与 .navbar-fixed-\* 类不同的是，你不用给 body 添加任何内补（padding）。



## 反色的导航条

通过添加 .navbar-inverse 类可以改变导航条的外观。



图片 24

16.2.2 路径导航

在一个带有层次的导航结构中标明当前页面的位置。

图片 25

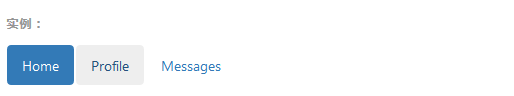
路径

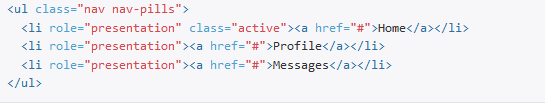


各路径间的分隔符已经自动通过 CSS 的 :before 和 content 属性添加了。

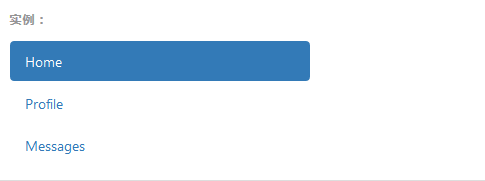


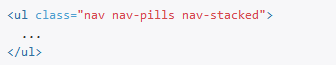
16.2.3 胶囊式导航





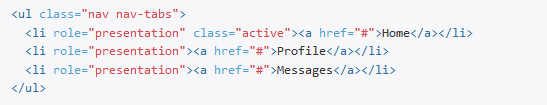
胶囊是标签页也是可以垂直方向堆叠排列的。只需添加 .nav-stacked 类。





16.2.4 标签导航

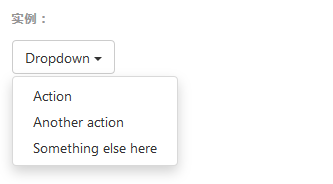




注意 .nav-tabs 类依赖 .nav 基类。

16.3 下拉菜单

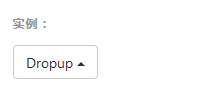
下拉菜单用于显示链接列表的可切换、有上下文的菜单



将下拉菜单触发器和下拉菜单都包裹在 .dropdown 里，或者另一个声明了 position: relative; 的元素。然后加入组成菜单的 HTML 代码。



通过为下拉菜单的父元素设置 .dropup 类，可以让菜单向上弹出（默认是向下弹出的）。





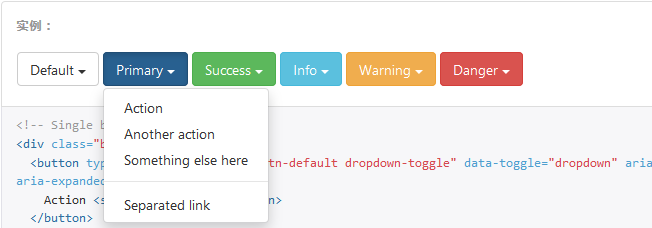
## 对齐

## 默认情况下，下拉菜单自动沿着父元素的上沿和左侧被定位为 100% 宽度。 为 .dropdown-menu 添加 .dropdown-menu-right 类可以让菜单右对齐。

16.3.1 按钮式下拉菜单

把任意一个按钮放入 .btn-group 中，然后加入适当的菜单标签，就可以让按钮作为菜单的触发器了。

只要改变一些基本的标记，就能把按钮变成下拉菜单的开关。





16.3.2 下拉菜单在导航条中的应用

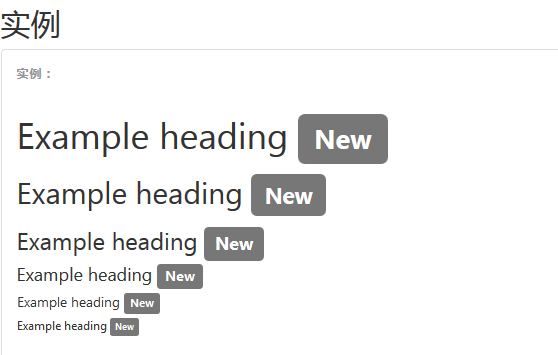
16.4 分页

为网站或应用提供带有展示页码的分页组件





16.5 标签



图片 43

#### 如果标签数量很多怎么办？

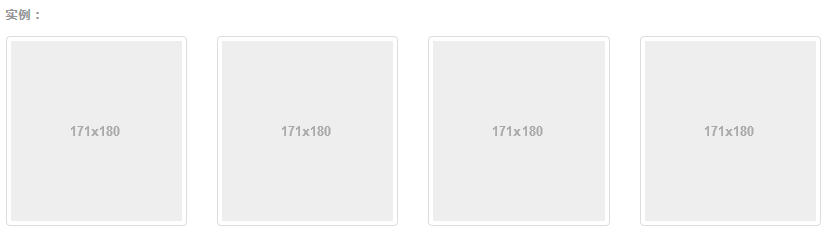
如果你有大量的设置为 inline 属性的标签全部放在一个较窄的容器元素内，在页面上展示这些标签就会出现问题，每个标签就会有自己的一个 inline-block 元素（就像图标一样）。解决的办法是为每个标签都设置为 display: inline-block; 属性

# 第17单元：bootstrap中的js插件（上）

17.1 缩略图

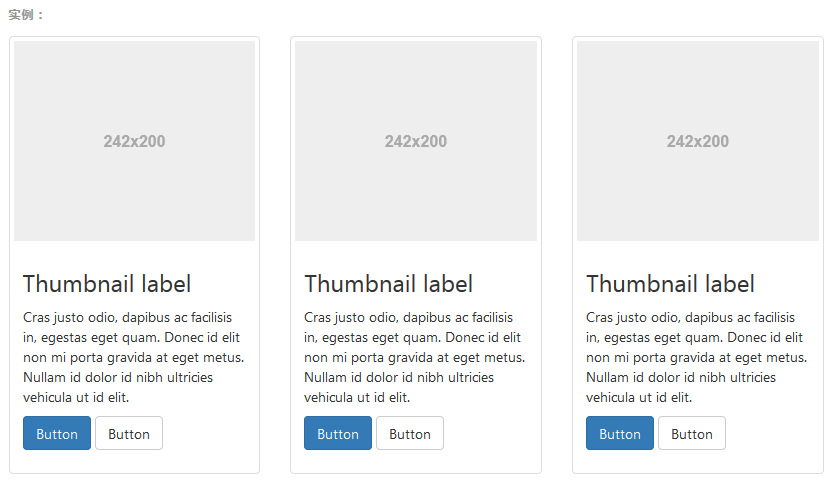
通过缩略图组件扩展 Bootstrap 的 [栅格系统](http://v3.bootcss.com/css/%23grid)，可以很容易地展示栅格样式的图像、视频、文本等内容。

Boostrap 缩略图的默认设计仅需最少的标签就能展示带链接的图片。





添加一点点额外的标签，就可以把任何类型的 HTML 内容，例如标题、段落或按钮，加入缩略图组件内。

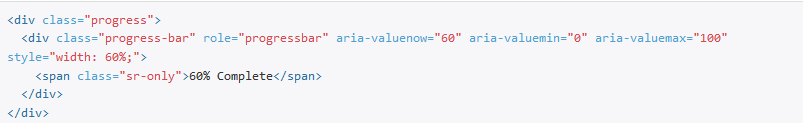


17.2 进度条

进度条组件使用了 CSS3 的 transition 和 animation 属性来完成一些特效。这些特性在 Internet Explorer 9 或以下版本中、Firefox 的老版本中没有被支持。Opera 12 不支持 animation 属性。

默认样式的进度条





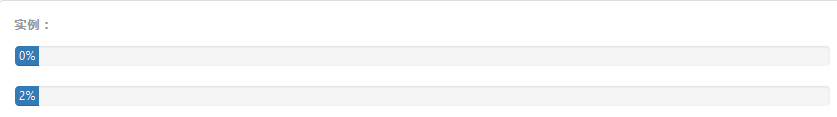
## 带有提示标签的进度条

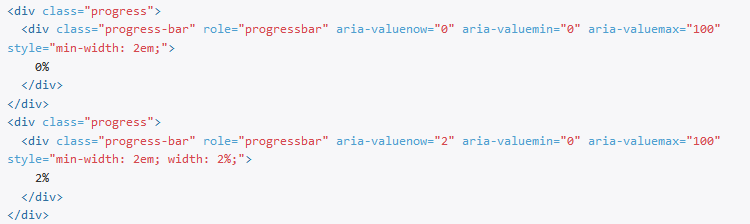
将设置了 .sr-only 类的 <span> 标签从进度条组件中移除 类，从而让当前进度显示出来。





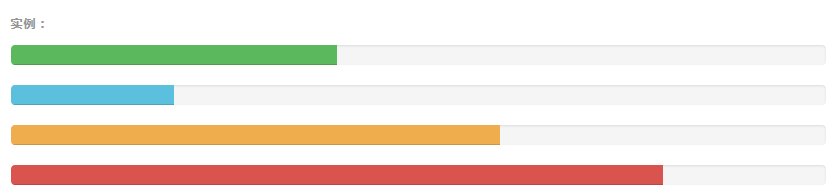
在展示很低的百分比时，如果需要让文本提示能够清晰可见，可以为进度条设置 min-width 属性。





## 根据情境变化效果

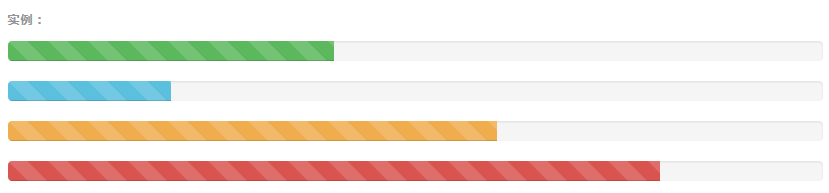
进度条组件使用与按钮和警告框相同的类，根据不同情境展现相应的效果。





## 条纹效果

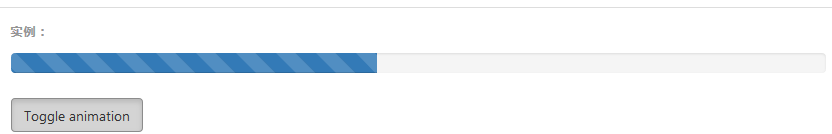
通过渐变可以为进度条创建条纹效果，IE9 及更低版本不支持。

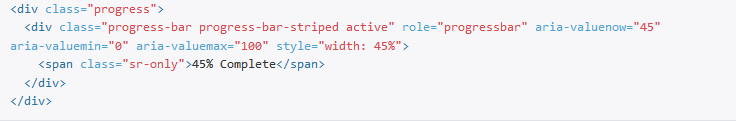




## 动画效果

为 .progress-bar-striped 添加 .active 类，使其呈现出由右向左运动的动画效果。IE9 及更低版本的浏览器不支持。





## 堆叠效果

把多个进度条放入同一个 .progress 中，使它们呈现堆叠的效果。



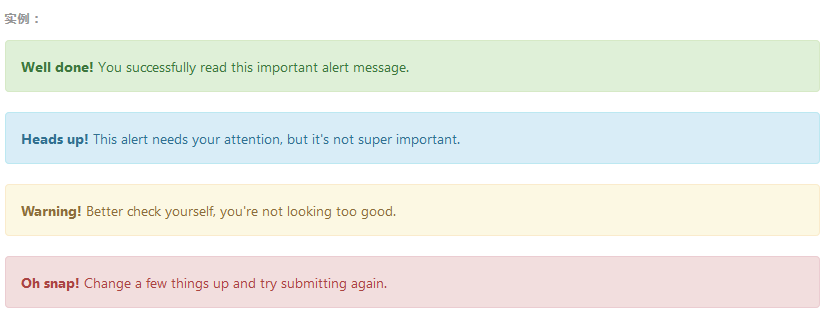


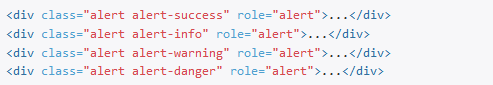
17.3 警告框

将任意文本和一个可选的关闭按钮组合在一起就能组成一个警告框，.alert 类是必须要设置的，另外我们还提供了有特殊意义的4个类（例如，.alert-success），代表不同的警告信息。

#### 没有默认类

警告框没有默认类，只有基类和修饰类。默认的灰色警告框并没有多少意义。所以您要使用一种有意义的警告类。目前提供了成功、消息、警告或危险。

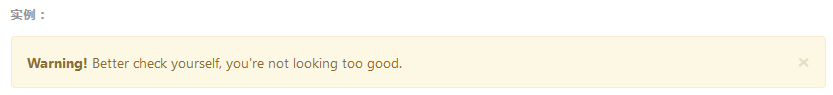


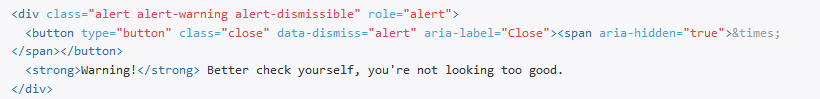


## 可关闭的警告框

为警告框添加一个可选的 .alert-dismissible 类和一个关闭按钮

如果需要为警告框组件提供关闭功能，请使用 [jQuery 警告框插件](http://v3.bootcss.com/javascript/%23alerts)。





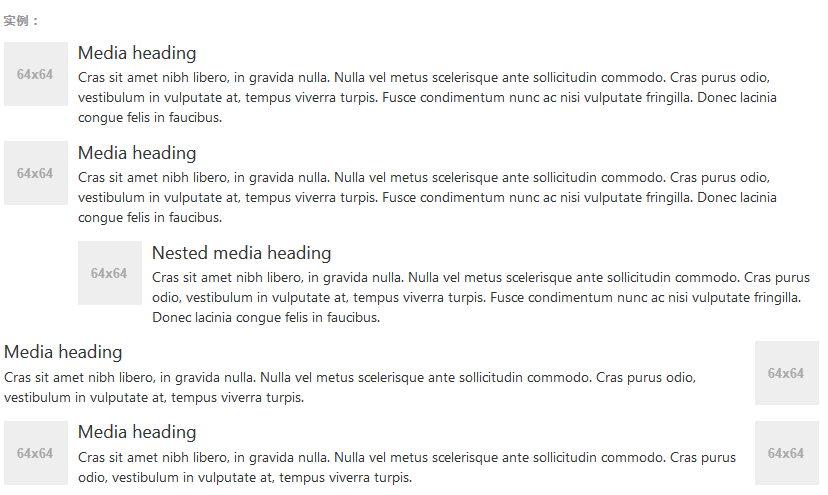
#### 确保在所有设备上的正确行为

务必给 <button> 元素添加 data-dismiss="alert" 属性。

17.4 媒体对象

这是一个抽象的样式，用以构建不同类型的组件，这些组件都具有在文本内容的左或右侧对齐的图片（就像博客评论或 Twitter 消息等）。

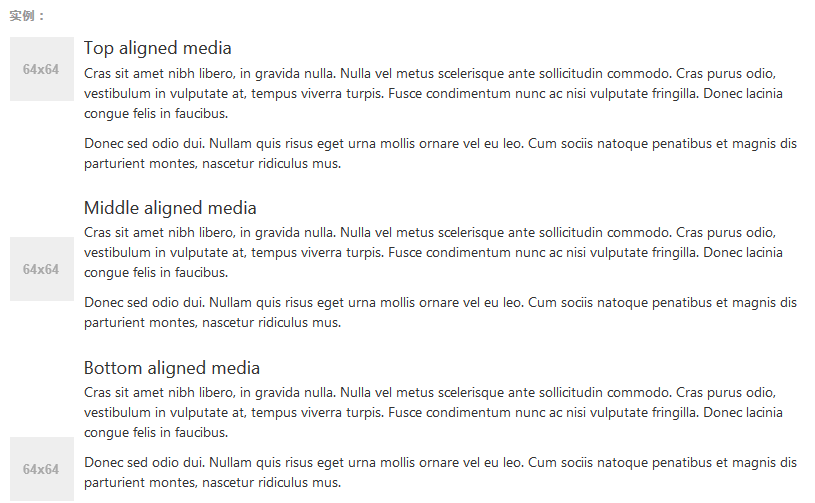
默认样式的媒体对象组件允许在一个内容块的左边或右边展示一个多媒体内容（图像、视频、音频）。





.pull-left 和 .pull-right 这两个类以前也曾经被用在了媒体组件上，但是，从 v3.3.0 版本开始，他们就不再被建议使用了。.media-left 和 .media-right 替代了他们，不同之处是，在 html 结构中， .media-right 应当放在 .media-body 的后面。

图片或其他媒体类型可以顶部、中部或底部对齐。默认是顶部对齐。

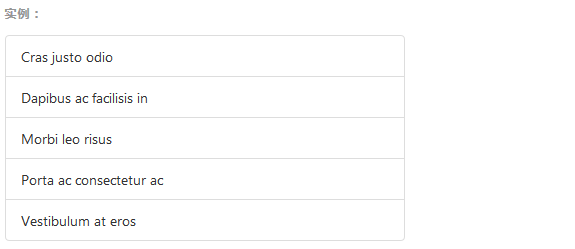


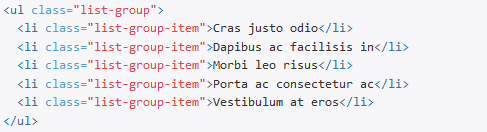


17.5 列表组与面板

列表组

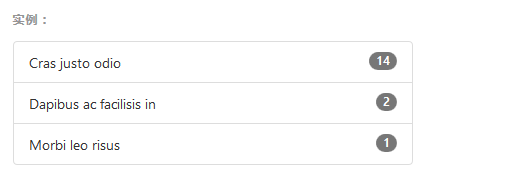
最简单的列表组仅仅是一个带有多个列表条目的无序列表，另外还需要设置适当的类。

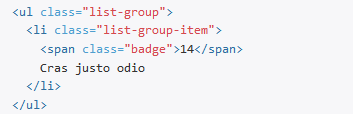




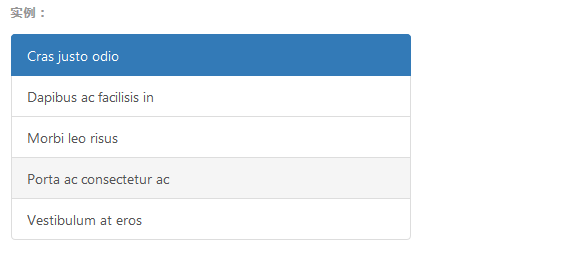
## 徽章

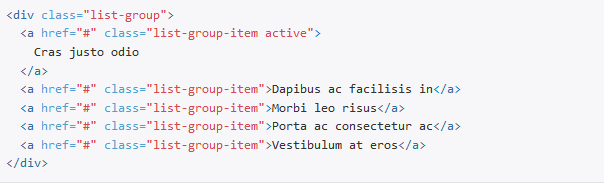
## 给列表组加入徽章组件，它会自动被放在右边。



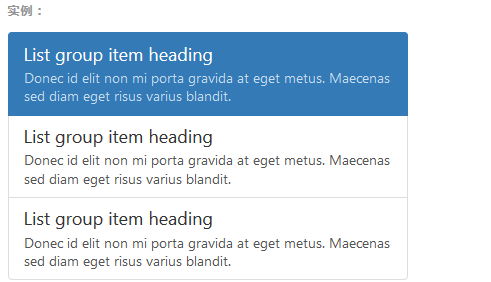


用 <a> 标签代替 <li> 标签可以组成一个全部是链接的列表组（还要注意的是，我们需要将 <ul> 标签替换为 <div> 标签）。没必要给列表组中的每个元素都加一个父元素。





列表组中的每个元素都可以是任何 HTML 内容



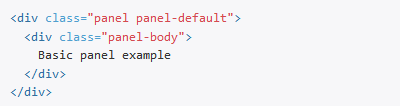


面板

虽然不总是必须，但是某些时候你可能需要将某些 DOM 内容放到一个盒子里。

默认的 .panel 组件所做的只是设置基本的边框（border）和内补（padding）来包含内容。





## 带标题的面版

通过 .panel-heading 可以很简单地为面板加入一个标题容器。你也可以通过添加设置了 .panel-title 类的 <h1>-<h6> 标签，添加一个预定义样式的标题。不过，<h1>-<h6> 标签的字体大小将被 .panel-heading 的样式所覆盖。

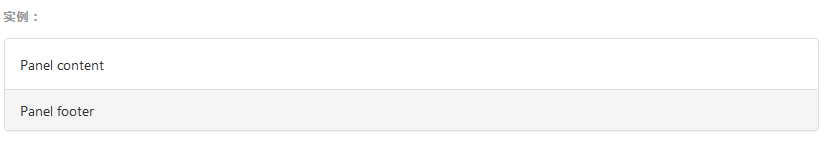
为了给链接设置合适的颜色，务必将链接放到带有 .panel-title 类的标题标签内。





## 带脚注的面版

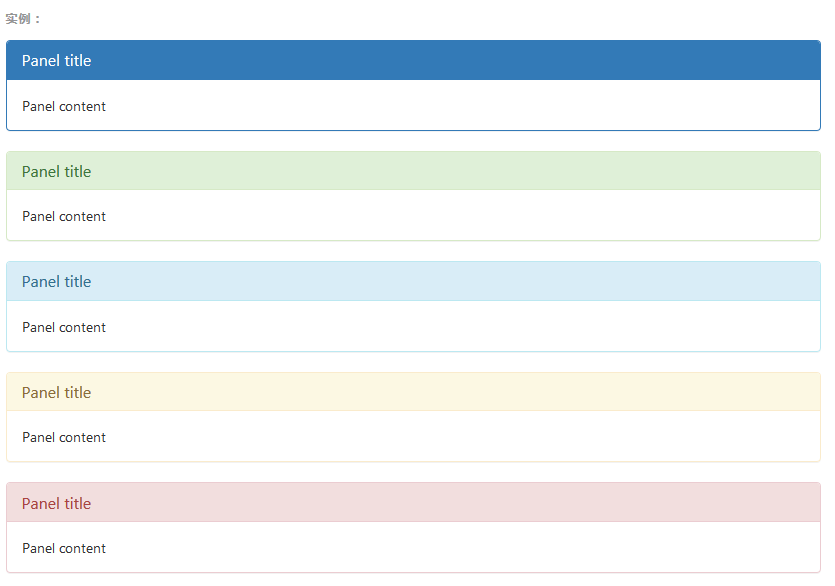
把按钮或次要的文本放入 .panel-footer 容器内。注意面版的脚注**不会**从情境效果中继承颜色，因为他们并不是主要内容。

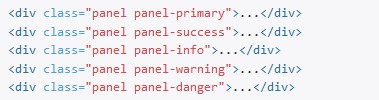




## 情境效果

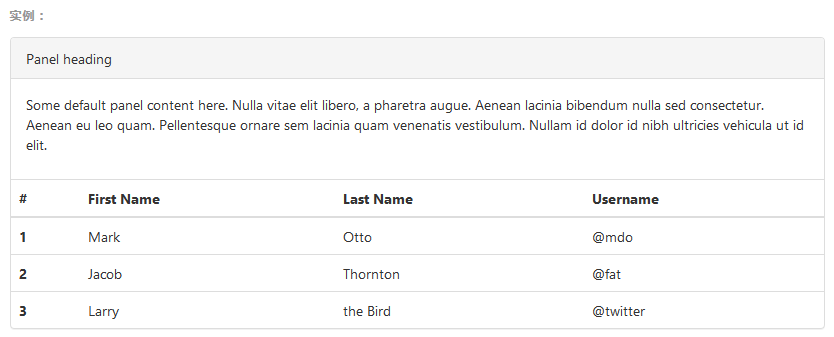
像其他组件一样，可以简单地通过加入有情境效果的状态类，给特定的内容使用更针对特定情境的面版。





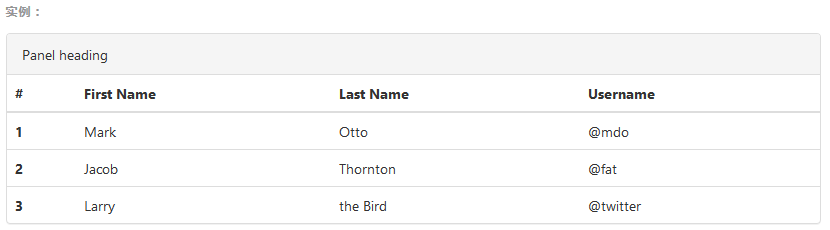
## 带表格的面版

为面板中不需要边框的表格添加 .table 类，是整个面板看上去更像是一个整体设计。如果是带有 .panel-body 的面板，我们为表格的上方添加一个边框，看上去有分隔效果。





如果没有 .panel-body ，面版标题会和表格连接起来，没有空隙。





# 第18单元 bootstrap中js插件 （下）

18.1 工具提示

18.2 弹出框

18.3 collapse

18.4 carousel