



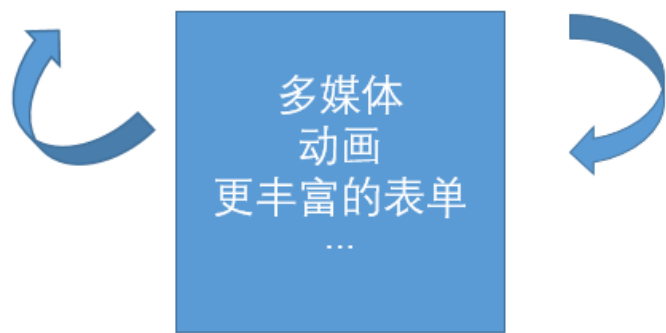
HTML5简介

● 什么是HTML5？

生态/应用 → web标准

H5页面是一种生态或者说是应用，而HTML5是一种web标准。

生态/应用 → web标准



● HTML5标准

丰富以及语义化的新标签

1. 新的区块和段落: `<section>` | `<article>` | `<nav>` | `<header>` | `<footer>` | `<aside>`
2. 全新的元素: `<main>` | `<figure>` | `<figcaption>` | `<time>` | `<progress>` | `<mark>` 等
3. 多媒体标签: `<video>` | `<audio>`
4. 更丰富的表单: 更多类型的 `<input>` | `<output>`

关于输入类型: color、date、email、number、search、tel、time、url、week

HTML5标准

HTML 语义化的建议：

1. 少使用无意义的 `<div>`、`` 标签；
2. 在 `<label>` 标签中设置 `for` 属性和对应的 `<input>` 关联起来；
3. 设置 `` 标签的 `alt` 属性，给 `<a>` 标签设置 `title` 属性，利于 SEO；
4. 在页面的标题部分使用 `<h1>`~`<h6>` 标签，不需要给它们加多余的样式；
5. 与表单、有序列表、无序列表相关的标签不要单独使用。

HTML5 也新增了一些语义化的元素，通过标签名就能判断标签内容。

HTML5标准

1. `<header>`标签通常放在页面或页面某个区域的顶部，用来设置页眉；
2. `<nav>`标签可以用来定义导航链接的集合，点击链接可以跳转到其他页面；
3. `<article>`标签中的内容比较独立，可以是一篇新闻报道，一篇博客，它可以独立于页面的其他内容进行阅读；
4. `<section>`标签表示页面中的一个区域，通常对页面进行分块或对内容进行分段，`<section>`标签和 `<article>`标签可以互相嵌套；
5. `<aside>`标签用来表示除页面主要内容之外的内容，比如侧边栏；
6. `<footer>`标签位于页面或页面某个区域的底部，用来设置页脚，通常包含版权信息，联系方式等。



HTML5标准

在HTML5前的表单标签中
验、日期选择控件、颜色
些常用的基本功能直接力

输入类型	描述
color	主要用于选取颜色
date	从一个日期选择器选择一个日期
datetime	选择一个日期 (UTC 时间)
datetime-local	选择一个日期和时间 (无时区)
email	包含 e-mail 地址的输入域
month	选择一个月份
number	数值的输入域
range	一定范围内数字值的输入域
search	用于搜索域
tel	定义输入电话号码字段
time	选择一个时间
url	URL 地址的输入域
week	选择周和年

文本框提示信息、表单校
跨表单等功能，HTML5中这
输入控制和验证。

```

<ul>
  <!-- 输入格式也限制为不同模式 -->
  <!-- 输入必须为邮箱类型 -->
  <li>email邮箱: <input type="email"></li>
  <!-- 输入必须为网址类型 -->
  <li>url网络地址: <input type="url"></li>
  <!-- 输入必须为日期类型 -->
  <li>date日期: <input type="date"></li>
  <!-- 输入必须为时间类型 -->
  <li>time时间: <input type="time"></li>
  <!-- 输入必须为数值类型 -->
  <li>number数量: <input type="number" min="5" max="20" step="2"></li>
  <!-- 输入必须为数值类型 -->
  <li>tel电话号码: <input type="tel"></li>
  <li>color颜色: <input type="color"></li>
  <li>周/年: <input type="week"></li>
  <li>月/年: <input type="month"></li>
  <li>范围: <input type="range"></li>
</ul>

```

- email邮箱:
- url网络地址:
- date日期:
- time时间:
- number数量:
- tel电话号码:
- color颜色:
- 周/年:
- 月/年:
- 范围:

HTML5标准

同时，还添加了placeholder、required、pattern、min、max、height、width等表单属性。

- placeholder 提供对输入域的提示值
- required 规定表单提交前输入域是否必填
- autofocus 页面加载完成自动聚焦输入框
- autocomplete 当用户输入信息时显示历史输入信息on/off
- pattern 规定用于验证input域的正则表达式
- min 规定输入域允许的最小值
- max 规定输入域允许的最大值
- step 属性，为输入域规定合法的数字间隔
- multiple 输入域可以选择多个值，适用于email和file类型

HTML5标准

多媒体以及图像——给Web应用带来了更多的可能性

➤ 多媒体： video、 audio、 WebRTC

使原生web页面对音视频的播放提供了支持，不需要借助flash插件。还允许通过WebRTC相关的一些API获取客户端的音视频流来建立一个端到端的音视频的通信。

➤ 图像： svg、 canvas、 WebGL

svg可缩放矢量图形， jpg、 png等格式的图片在缩放一定比例的时候会有一定的失真，但svg在缩放时可以保证不失真。

WebGL： 是一个JavaScript API，能够在任何一个支持它的浏览器中实现一个渲染性能很高的2D或者3D图形，通常我们用它绘制一些2D或3D图形。

HTML5 音频 audio

HTML5音频和视频的添加可以直接采用标签，不需要借助第三方插件就可以完成。

在HTML5中音频的添加方式是采用<audio></audio>标签，它支持三种音频格式的文件。

.mp3 .wav .ogg格式的音频文件

IE8及以下浏览器不支持audio标签。

若浏览器不支持video元素或者无法播放音频，则会显示替代文本（开始和结束标签之间的内容）。

<audio src="music.mp3">当前浏览器不支持audio标签</audio>

标签属性：

- ◆ autoplay: 音频会尽快自动播放，不会等待整个音频下载完成
- ◆ controls: 浏览器提供包括声音、播放进度、播放暂停的控制面板（不同浏览器不一致），用户可以控制音频播放
- ◆ loop: 循环播放音频
- ◆ muted: 是否静音，默认值为false，表示有声音
- ◆ preload: 预加载，包括auto、metadata和none三个参数值，auto表示加载音频，metadata表示不加载音频，但是需要获取音频元数据（如音频长度），none表示不加载音频。若指定为空字符串，则等效于auto。注意autoplay属性优先级高于preload，若autoplay被指定，则会忽略此属性，浏览器将加载音频以供播放
- ◆ src: 嵌入的音频URL

HTML5 音频 audio

```
<audio controls= “controls” >
```

```
  <source src="horse.ogg" type="audio/ogg">
```

```
  <source src="horse.mp3" type="audio/mpeg">
```

```
  <source src="horse.wav" type="audio/wav">
```

您的浏览器不支持 HTML5: audio 元素。

```
</audio>
```



由于各家浏览器制造商未能在对标准音频和视频解码器支持上达成一致，因此需要使用<source>元素来指定不同格式的媒体源。

<audio> 标签允许使用多个 <source> 元素。<source> 元素可以链接不同的音频格式文件，浏览器将使用第一个支持的音频格式文件。

HTML5 视频 video

在HTML5中视频的添加方式是采用<video></video>标签，它支持三种视频格式的文件。

.mp4 .webm .ogg格式的视频文件

```
<video width="320" height="240" controls>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogv" type="video/ogg">  
  <source src="movie.webm" type="video/webm">  
  您的浏览器不支持 HTML5: video 标签。
```

```
</video>
```



注意：video标签内我们添加的多个source表示的是当前视频文件的多个格式，而不是多个视频文件。

HTML5 视频 video

在HTML5中视频的添加方式是采用<video></video>标签，它支持三种视频格式的文件。

.mp4 **.webm** **.ogg**格式的视频文件

```
<video id="myVideo" width="320" height="240">  
  <source src="resources/movie.mp4" type="video/mp4" />  
  <source src="resources/movie.webm" type="video/webm" />  
  <source src="resources/movie.ogg" type="video/ogg" />  
  您的浏览器不支持 HTML5 video 标签  
</video>  
<button onclick="playPause()">暂停/播放</button>
```



暂停/播放

- 在HTML4的标准中，按钮我们是通过form表单以及input标签中type设置为button来完成一个按钮。在HTML5中，仅通过button标签就可以完成按钮的设置。

HTML5 视频 video

在HTML5中视频的添加方式是采用<video></video>标签，它支持三种视频格式的文件。

.mp4 .webm .ogg格式的视频文件

```
<script>
    let v = document.getElementById('myVideo')
    function playPause() {
        if (v.paused)
            v.play()
        else
            v.pause()
    }
</script>
```

video元素提供的API能够让脚本控制媒体



暂停/播放

HTML5 canvas

HTML5 `<canvas>` 元素用于图形的绘制，通过脚本（通常是JavaScript）来完成。

注意`<canvas>` 标签只是图形容器，必须使用脚本来绘制图形。

在HTML网页上定义canvas元素后，它只是一张空白的画布，为了在canvas上绘图，必须经过三步：

- ① 获取canvas元素对应的DOM对象，这是一个Canvas对象。
- ② 调用Canvas对象的`getContext()`方法，返回一个`CanvasRenderingContext2D`对象（称作绘制上下文对象）。
- ③ 调用`CanvasRenderingContext2D`对象的方法（一般也称作画布API）绘图。

HTML5 canvas

HTML5 <canvas> 元素用于图形的绘制，通过脚本（通常是JavaScript）来完成。

注意<canvas> 标签只是图形容器，必须使用脚本来绘制图形。



绘制直线(0,0)-(180,50)

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
```

您的浏览器不支持 HTML5 canvas 标签。

```
</canvas>
```

```
<script>
```

```
    window.onload = draw
```

```
    function draw(e) {
```

```
        let c = document.getElementById('myCanvas')
```

```
        let ctx = c.getContext('2d')
```

```
        ctx.moveTo(0, 0) //绘图的起始点
```

```
        ctx.lineTo(180, 50) //绘图的终止点
```

```
        ctx.lineWidth = 5 //画笔的粗细
```

```
        ctx.strokeStyle = '#FF0000' //画笔的颜色
```

```
        ctx.stroke() //真正的绘图
```

```
    }
```

```
</script>
```

2d表示当前我们的绘图是一个二维平面绘图。

HTML5 canvas

HTML5 <canvas> 元素用于图形的绘制，通过脚本（通常是JavaScript）来完成。

注意<canvas> 标签只是图形容器，必须使用脚本来绘制图形。

```
<body>
```

```
This is a red square: <canvas id="square" width=10 height=10></canvas>.
```

```
This is a blue circle: <canvas id="circle" width=10 height=10></canvas>.
```

```
<script>
```

```
var canvas = document.getElementById("square"); // 获取第一个画布元素
```

```
var context = canvas.getContext("2d"); // 获取2D绘制上下文
```

```
context.fillStyle = "#f00"; // 设置填充色为红色
```

```
context.fillRect(0,0,10,10); // 填充一个正方形
```

```
canvas = document.getElementById("circle"); // 第二个画布元素
```

```
context = canvas.getContext("2d"); // 获取它的绘制上下文
```

```
context.beginPath(); // 开始一条新的路径
```

```
context.arc(5, 5, 5, 0, 2*Math.PI, true); // 将圆形添加到该路径中
```

```
context.fillStyle = "#00f"; // 设置填充色为蓝色
```

```
context.fill(); // 填充路径
```

```
</script>
```

```
</body>
```

HTML5 canvas

使用canvas把一幅图像放置到画布上，使用以下方法drawImage(image, x, y)。

```

```

```
<canvas id="myCanvas" width="250" height="300" style="border:1px solid #d3d3d3;">
```

您的浏览器不支持 HTML5 canvas 标签。</canvas>

```
<script>
```

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

```
    var ctx=c.getContext("2d");
```

```
    var img=document.getElementById("scream");
```

```
    img.onload = function() {
```

```
        ctx.drawImage(img,10,10);
```

```
    }
```

```
</script>
```



什么是矢量图形？在网络中，大家经常会和两种类型的图片打交道 —— **位图和矢量图**

1. 位图使用**像素网格**来定义 — 一个位图文件精确的包含了每个像素的位置和它的色彩信息。流行的位图格式包括 Bitmap (.bmp), PNG (.png), JPEG (.jpg), and GIF (.gif)。
2. 矢量图使用**算法**来定义 — 一个矢量图文件包含了图形和路径的定义，电脑可以根据这些定义计算出当它们在屏幕上渲染时应该呈现的样子。 SVG格式可以让我们创造用于Web的精彩矢量图形。

将SVG添加到页面

1、快捷方式：

```

```

好处：

快速，熟悉的图像语法。

可以通过在<a>元素嵌套，使图像轻松地成为超链接。

缺点：

如果要使用 CSS 控制 SVG 内容，则必须在SVG代码中包含内联 CSS 样式；

不能用CSS伪类来重设图像样式（如:focus）

2、在 HTML 中引入 SVG 代码

```
<svg width="300" height="200">  
  <rect width="100%" height="100%" fill="green" />  
</svg>
```

在文本编辑器中打开 SVG 文件，复制 SVG 代码，并将其粘贴到 HTML 文档中。--内联 SVG

优点：

- SVG 内联可以减少 HTTP 请求，可以减少加载时间。
- 可以为 SVG 元素分配class和id，并使用 CSS 修改样式，无论是在 SVG 中，还是在 HTML 文档中的 CSS 样式规则。实际上，可以使用任何 SVG 外观属性作为 CSS 属性。
- 内联 SVG 是唯一可以在 SVG 图像上使用 CSS 交互（如:focus）和 CSS 动画的方法。
- 可以通过将 SVG 标记包在<a>元素中，使其成为超链接。

● 将SVG添加到页面

3、使用 <iframe> 嵌入 SVG

```
<iframe src="triangle.svg" width="500" height="500" sandbox>
  
</iframe>
```

关于SVG的更多，请参考链接

[https://developer.mozilla.org/zh-](https://developer.mozilla.org/zh-CN/docs/Learn/HTML/Multimedia_and_embedding/Adding_vector_graphics_to_the_Web)

[CN/docs/Learn/HTML/Multimedia_and_embedding/Adding_vector_graphics_to_the_Web](https://developer.mozilla.org/zh-CN/docs/Learn/HTML/Multimedia_and_embedding/Adding_vector_graphics_to_the_Web)

● HTML5标准

强大的连通性

早期服务器在接收到客户端的访问请求后，依据客户端请求类型，服务器把请求内容返回给客户端页面。这样子的请求方式是一种双向的顺序的过程。这种方式就有两个局限，一是服务端不能主动发送资源给客户端，必须得有客户端发起请求才可以；二是这是一种有顺序的通信，通信效率低，页面和服务端要进行通信，需要先建立连接才可以。

HTML5带来了上述两种局限的解决方案。

- 1、Server-sent Event
- 2、Web Socket
- 3、WebRTC

HTML5标准

1、Server-sent Event

允许服务器通过EventSource的方式来向客户端推送内容，这就解决了刚才前边提到的第一个局限。

2、Web Socket

HTML5还带来了Web Socket，允许页面和服务器之间建立一个永久的链接，页面和服务器之间可以进行一个永久的通信，这种通信过程可以不是顺序的，可以由服务端主动进行的，也可以是由客户端主动进行的，解决了前边说的第二个问题，也可以不用频繁的去建立连接。

3、WebRTC

RTC是实时通信的简称，webRTC是实时通信在web上的应用，允许我们在不同的浏览器客户端之间建立一个语音、视频的交流和数据分享的技术，并且不用安装任何第三方的工具或软件。这个技术给web应用带来了更多的可能，比如说我们可以实现一个在线的音视频会议或者在线的聊天室。

HTML5标准

离线和缓存

- **Application Cache:** 应用级缓存，能够在客户端有选择的缓存一些资源，通过这样的方式，用户也可以在离线的环境下浏览一些内容，毕竟由于一些资源缓存在了本地，浏览速度也会更快。这样的方式也能一定程度减轻服务器的负担，因为浏览器只会在资源发生改变的情况下向服务器端发送请求。不过被Service Worker取代。
- **在线离线状态的判断:** 通过在Windows上监听online或者offline事件的状态判断在线或离线。
- **Index DB:** 使用其将一些结构化的数据持久的存储在浏览器中，这样不用考虑网络，创建出具有查询能力的离线流。用户没有网络的情况下也能很好的使用，等有网络的时候可以将用户产生的数据进行在线同步。
- **localStorage/sessionStorage:** 除了结构化的数据还有一些比较简单的数据并不需要存储在indexDB中，可以借助localStorage和sessionStorage去存储，早期的话存储数据可以借助cookie，但cookie的话其实是用来做信息验证或者登录信息存储，支持长度小，有长度限制。localStorage/sessionStorage两个都是将信息存储在客户端，区别在于localStorage在页面关闭再次打开时存储的信息依然存在，sessionStorage在页面关闭后再次打开信息删除，它只是存储的是一个会话的信息。

● HTML5标准

性能

- JavaScript是一个单线程的语言，所有任务都会通过顺序的方式执行，所有的任务都只能依次执行。这样的执行过程其实原本没有问题，后来由于多核CPU的产生，电脑的计算能力增强。这种单线程模式不能很好应用计算机的计算能力。所以HTML5引入了Web Worker， Web Worker为JS创建多线程环境，它允许主线程在执行任务的过程中创建worker进程，将一些耗时或者高延时的任务放在worker中运行，这样这些任务在后台运行，与主线程互不干扰。
- HTML5中引入了动画效果，保证动画效果的流畅，其实是需要借助requestAnimationFrame这个API，即允许API通过一定的频率来刷新页面从而保证动画的流畅性。

◆ HTML5标准

其他

- **Drag and Drop**, 就是拖放, 这个API的引入, 允许我们在页面中去进行元素的拖拽以及释放。
- **History API**, 给予浏览器JS脚本对于历史记录操作的一个能力, VUE和React两个框架, 他们中的路由的history模式就是基于这个API来实现的。
- **XMLHttpRequest Level 2**, 相较于level 1, 做了以下3种更新, 比如说以前只支持文本格式的传送, 无法读取和上传二进制文件, 在level2版本中得到了支持。Level1中传送和接收数据中无法知道进度, 只能知晓完成或失败, 在2中能够通过事件获取到传送进度; 1中不能跨域去请求数据, 2中可以通过一定手段进行跨域资源的请求。
- ...



CSS3特性

概述

前面课程讲解的CSS属性大部分是CSS2标准，本章将介绍CSS3新增加的属性和规则。CSS3会让网页更美观，例如对元素设置圆角边框和阴影，可以取得原来图片处理软件才可以达到的效果；利用过渡、2D变换、3D变换、动画等功能，可以取得JavaScript才能实现的动态效果。

由于这些特性是逐渐获得浏览器支持的，所以不同浏览器或浏览器的不同版本对新特性的支持度各不相同，有时需要在属性名前面添加浏览器前缀，以便得到正确解析。

浏览器	CSS3前缀	浏览器	CSS3前缀
Safari、Chrome	-webkit-	Opera	-o-
Firefox	-moz-	IE	-ms-





浏览器私有前缀

浏览器私有前缀是为了兼容老版本的写法，比较新版本的浏览器无须添加。

1. 私有前缀

- `-moz-`: 代表 `firefox` 浏览器私有属性
- `-ms-`: 代表 `ie` 浏览器私有属性
- `-webkit-`: 代表 `safari`、`chrome` 私有属性
- `-o-`: 代表 `Opera` 私有属性

2. 提倡的写法

```
-moz-border-radius: 10px;  
-webkit-border-radius: 10px;  
-o-border-radius: 10px;  
border-radius: 10px;
```

圆角边框与阴影

圆角边框**border-radius** 用于添加元素的圆角边框

border-top-left-radius 左上角的形状

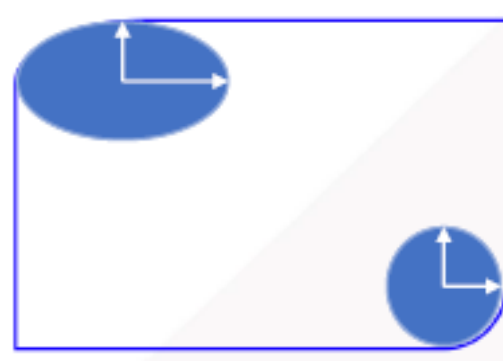
border-top-right-radius 右上角的形状

border-bottom-left-radius 左下角的形状

border-bottom-right-radius 右下角的形状



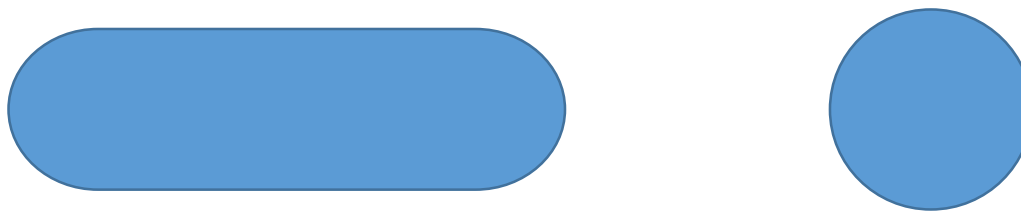
```
div {  
  height: 100px;  
  width: 150px;  
  border: 1px solid blue;  
  border-top-left-radius : 40px 20px;  
  border-bottom-right-radius: 20px;  
}
```



圆角边框与阴影

圆角边框border-radius

```
<div> </div>
div
{
    width:350px;
    height:50px;
    border:2px solid #a1a1a1;
    background:#ddd;
    border-radius:25px;
}
```



border-radius扩展

-webkit-border-radius:

-moz-border-radius:

border-radius:

圆角边框与阴影

阴影box-shadow

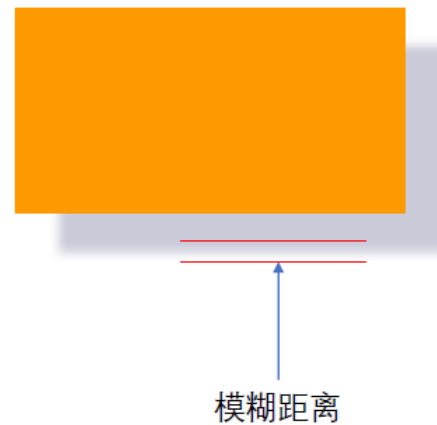
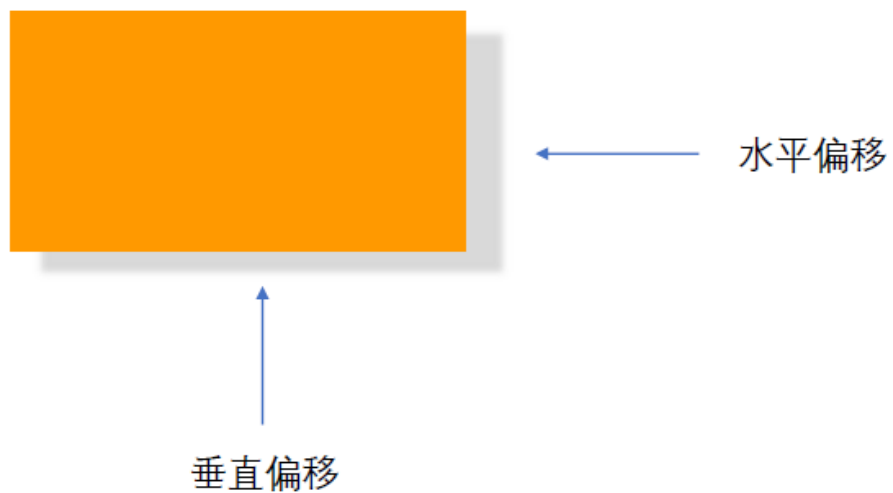
取值：inset|outset 水平偏移 垂直偏移 模糊距离 颜色

- inset可选，内部阴影，outset默认值，外部阴影
- 模糊半径决定了阴影沿偏移边界每侧的模糊范围，值越大，模糊的范围越大
- box-shadow取值可以为多个，即添加多重效果，用逗号隔开即可。



圆角边框与阴影

阴影box-shadow





圆角边框与阴影

阴影box-shadow

```
<div> </div>
div
{
    width:300px;
    height:100px;
    background-color:#f90;
    box-shadow: 10px 10px 5px #888;
}
```

box-shadow扩展

```
-webkit-border-shadow:inset hoff voff blur color
-moz-border-shadow:inset hoff voff blur color
border-shadow:inset hoff voff blur color
```

● 文本与文字

text-shadow属性 用于添加文本阴影

text-shadow: 水平偏移 垂直偏移 阴影大小 颜色

```
h1 {  
  text-shadow: 2px 2px red;  
}
```

text-shadow

<h1>text-shadow</h1>

```
h1 {  
  text-shadow: 2px 2px 8px blueviolet;  
}
```

text-shadow

<h1>text-shadow</h1>

● 文本与文字

text-shadow属性：水平偏移 垂直偏移 阴影大小 颜色

```
h1 {  
  text-shadow: 0 0 3px #f00;  
}
```

<h1>text-shadow</h1>

text-shadow

```
h1 {  
  color: lightpink;  
  text-shadow: 2px 2px 4px lightgreen;  
}
```

<h1>text-shadow</h1>

text-shadow

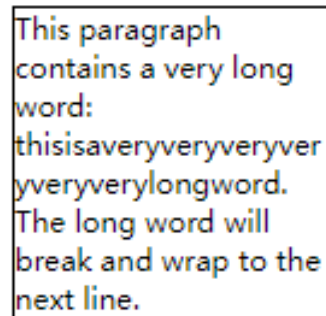
文本与文字

word-wrap属性：允许长单词、URL强制进行换行

取值： normal （默认值）
 break-word

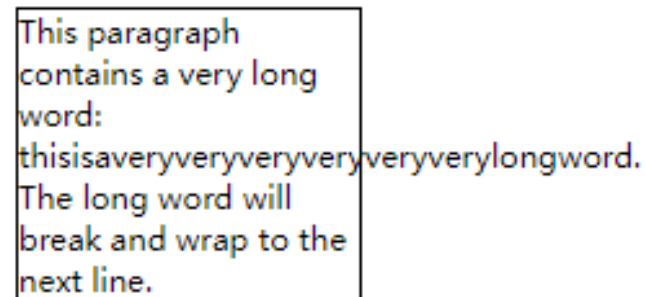
```
p.test
{
  width:11em;
  border:1px solid #000000;
  word-wrap:break-word;
}
```

```
<p class="test">
  This paragraph contains a very long word:
  thisisaveryveryveryveryveryverylongword. The
  long word will break and wrap to the next
  line.
</p>
```



This paragraph
contains a very long
word:
thisisaveryveryveryver
yveryverylongword.
The long word will
break and wrap to the
next line.

break-word



This paragraph
contains a very long
word:
thisisaveryveryveryveryveryverylongword.
The long word will
break and wrap to the
next line.

normal

文本与文字

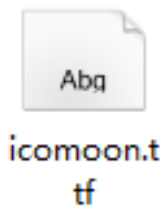
@font-face规则

字体文件后缀	适用于浏览器
.TTF或.OTF	Firefox、Safari、Opera
.EOT	Internet Explorer 4.0+
.SVG	Chrome、IPhone
.WOFF	Chrome、Firefox

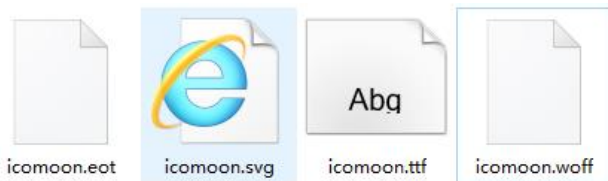
不同浏览器支持不同字体文件格式，在应用web字体前，需要生成同一种字体的不同格式文件。

使用

1、下载字体 只有一种.ttf格式



2、需要生成其他格式字体文件



● 文本与文字

@font-face规则

Web字体首先用@font-face规则定义字体名称和来源，然后再用font-family引用该字体。语法格式如下：

```
@font-face{  
    font-family:字体名字;  
    src:url(字体来源文件1),  
        ...  
        url(字体来源文件4);  
}
```

文本与文字

@font-face规则

```
...<style>
...  @font-face {
...    font-family: firasans font;
...    src: url(fonts/firasans-light-webfont.eot),
...         url(fonts/firasans-light-webfont.svg),
...         url(fonts/firasans-light-webfont.ttf),
...         url(fonts/firasans-light-webfont.woff);
...  }
...  p {
...    margin: 100px auto;
...    width: 200px;
...    font-family: 'firasans font';
...    font-size: 20px;
...    color: blueviolet;
...    font-weight: 700;
...  }
...</style>
```

定义字体名称

定义字体来源

引用字体

● 2D变换

变换 (transform) 是CSS3中具有颠覆性的特征之一，可以实现元素的位移、旋转、缩放等效果。

变换 (transform) 可以简单理解为变形。

移动: translate

旋转: rotate

缩放: scale

倾斜: skew



2D变换

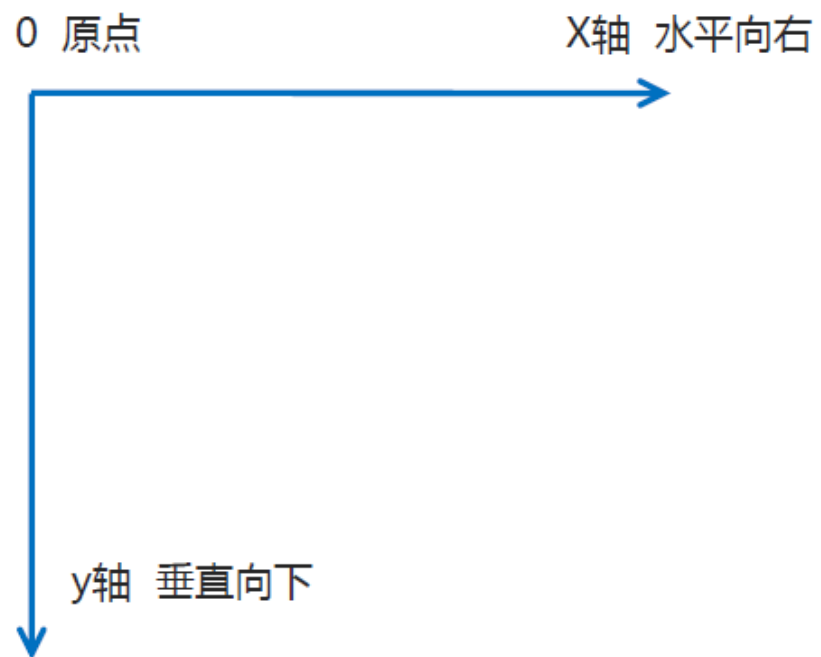
transform属性的常用方法

transform属性常用方法	说明
translate(x,y)	移动：沿X和Y轴
translateX(n)	移动：沿X轴
translateY(n)	移动：沿Y轴
rotate(angle)	旋转：单位是deg（角度），正数为顺时针，负数为逆时针
scale(x,y)	缩放：改变元素的宽度和高度，<1为缩小，>1为放大
scaleX(n)	缩放：改变元素的宽度
scaleY(n)	缩放：改变元素的高度
skew(x-angle,y-angle)	倾斜：沿X轴和Y轴，单位是deg（角度），一个值表示Y轴不倾斜
skewX(angle)	倾斜：沿X轴倾斜，与Y轴夹角为angle
skewY(angle)	倾斜：沿Y轴倾斜，与X轴夹角为angle

2D变换

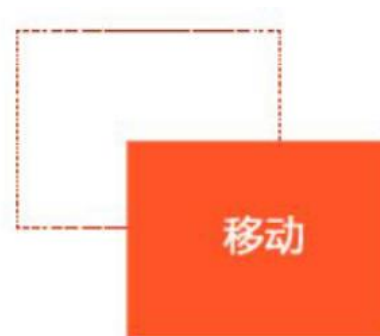
二维坐标系

2D变换是改变标签在二维平面上的位置和形状的一种技术，先来学习二维坐标系



● 2D变换

2D变换之移动 translate



2D移动是2D转换里面的一种功能，可以改变元素在页面中的位置，类似定位。

1、语法

```
transform: translate(x,y); 或者分开写  
transform: translateX(n);  
transform: translateY(n);
```

2、重点

定义 2D 转换中的移动，沿着 X 和 Y 轴移动元素；

translate最大的优点：不会影响到其他元素的位置；

translate中的百分比单位是相对于自身元素的 translate:(50%,50%);

对行内元素标签没有效果；

2D变换

2D变换之旋转rotate

2D旋转指的是让元素在2维平面内顺时针旋转或者逆时针旋转。

1、语法

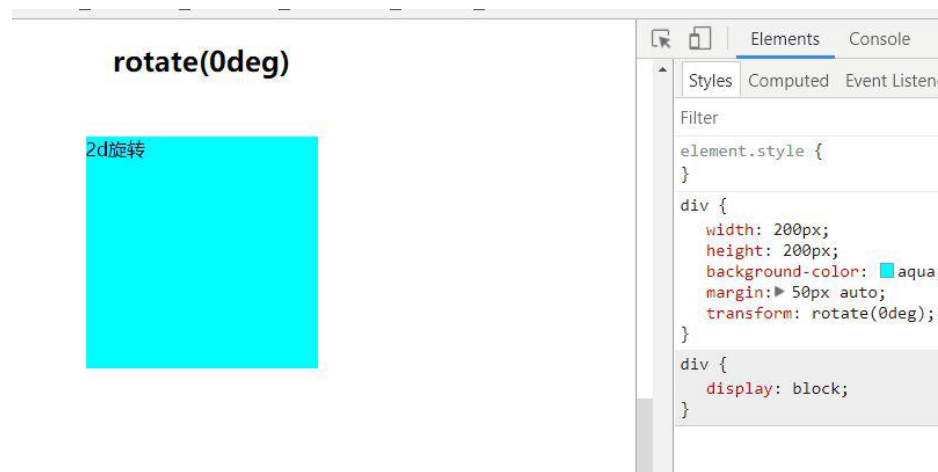
`transform:rotate(度数)`

2、重点

rotate里面跟度数，单位是 deg 比如 rotate(45deg)

角度为正时，顺时针，负时，为逆时针

默认旋转的中心点是元素的中心点





2D变换

案例：三角形

```
.outer {  
    position: relative;  
    width: 200px;  
    height: 30px;  
    border: 1px solid #000;  
}  
.outer::after {  
    content: '';  
    position: absolute;  
    top: 10px;  
    right: 15px;  
    width: 8px;  
    height: 8px;  
    border-right: 1px solid #000;  
    border-bottom: 1px solid #000;  
    transform: rotate(45deg);  
    transition: all 0.5s;  
}  
.outer:hover::after {  
    transform: rotate(225deg);  
}
```



2D变换

2D变换中心点transform-origin

1、语法

`transform-origin: x y;`

2、重点

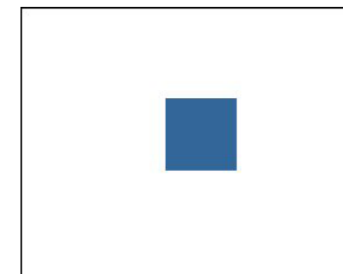
注意后面的参数 `x` 和 `y` 用空格隔开

`x y` 默认转换的中心点是元素的中心点 (`50% 50%`), 相当于 (`center center`)

可以设置9个位置值: 水平方向: `left center right`

`0 50% 100%`

垂直方向: `top center bottom`



2D变换

2D变换之缩放scale

缩放，顾名思义，可以放大和缩小。只要给元素添加上了这个属性就能控制它放大还是缩小。

1、语法

`transform: scale(x,y);`

2、注意

注意其中的x和y用逗号分隔

`transform:scale(1,1)`：宽和高都放大一倍，相对于没有放大

`transform:scale(2,2)`：宽和高都放大了2倍

`transform:scale(2)`：只写一个参数，第二个参数则和第一个参数一样，相当于 `scale(2,2)`

`transform:scale(0.5,0.5)`：缩小

scale缩放最大的优势：可以设置变换中心点缩放。默认以中心点缩放的，不影响其他盒子

2D变换

```
<div class="box"></div>
```

```
<style>
  .box {
    width: 100px;
    height: 100px;
    border: 1px solid blue;
    transform: translate(50px, 50px); /* 沿X轴移动50px, Y轴移动50px */
    transform: rotate(30deg); /* 顺时针旋转30度 */
    transform: scale(0.5, 2); /* 宽度缩小为原来的一半, 高度放大为原来的2倍 */
    transform: skew(10deg, 30deg); /* 沿X轴倾斜10度, 沿Y轴倾斜30度 */
  }
</style>
```

2D变换

2D 变换综合写法

注意：

1. 同时使用多个转换，其格式为：`transform: translate() rotate() scale() ...`等。
2. 其顺序会影响转换的效果。（先旋转会改变坐标轴方向）
3. 当我们同时有位移和其他属性的时候，记得要将位移放到最前，顺序不能随意更改否则效果无法展示。

2D变换

2D变换总结

- transform, 简单理解就是变形, 有2D和3D之分;
- 2D 移动, `translate(x, y)` 最大的优势是不影响其他盒子, 里面参数如果用%, 是相对于自身宽度和高度来计算的, 可以分开写比如 `translateX(x)` 和 `translateY(y)`;
- 2D 旋转, `rotate(度数)` 可以实现旋转元素, 度数的单位是deg;
- 2D 缩放, `scale(x,y)` 里面参数是数字, 不跟单位, 可以是小数, 最大的优势是不影响其他盒子;
- 设置转换中心点 `transform-origin: x y`; 参数可以百分比、像素或者是方位名词;
- 当我们进行综合写法, 同时有位移和其他属性的时候, 记得要将位移放到最前边;

◆ 3D 变换

2D变换能够改变元素在X轴和Y轴方向上的特性，3D变换能改变元素在Z轴方向上的特性，并可以通过视角设置透视关系，使元素具有透视效果。

3D -> transform-style: preserve-3d;

旋转: transform属性

- transform: rotateX(deg);
- transform: rotateY(deg);
- transform: rotateZ(deg);

透视: perspective属性

3D 变换

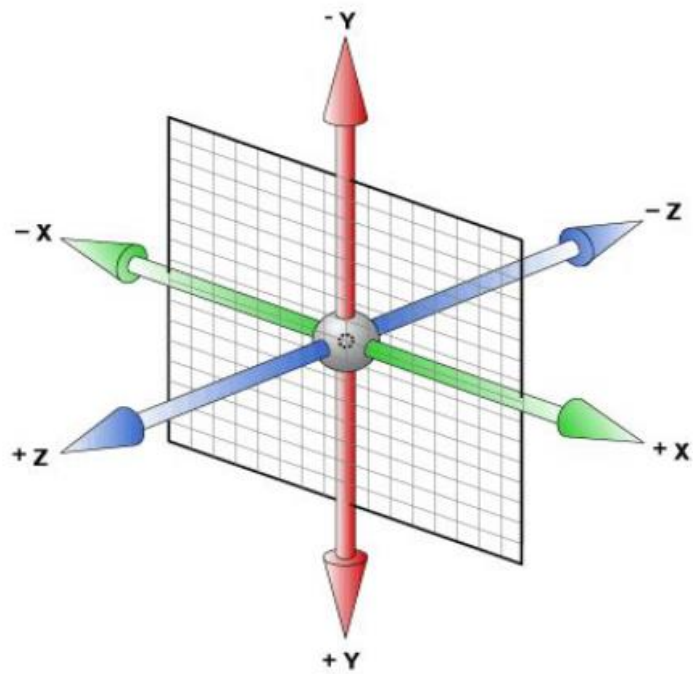
1、三维坐标系

三维坐标系其实就是指立体空间，立体空间是由3个轴共同组成的。

x轴：水平向右 注意：x 右边是正值，左边是负值

y轴：垂直向下 注意：y 下面是正值，上面是负值

z轴：垂直屏幕 注意：往外面是正值，往里面是负值



3D 变换

3D变换常用属性

属性	说明
transform	2D或3D变换，取值为变换方法
transform-origin	变换的起始点，默认为元素中心点 transform-origin: x y x和y可以取值px, %, 方位名词 (left、right、top、bottom、center)
transform-style	取值为preserve-3d时，则可以产生3D变换效果
perspective	眼睛和舞台间的距离，像素为单位 值越小，透视效果越明显
perspective-origin	视线关注的原点，默认为元素中心点
backface-visibility	元素背对屏幕时是否可见，取值为hidden表示隐藏

3D 变换

3D变换transform属性常用方法

旋转	移动	缩放
rotate3d(x,y,z,angle)	translate3d(x,y,z)	scale3d(x,y,z)
rotateX(angle)	translateX(x)	scaleX(x)
rotateY(angle)	translateY(y)	scaleY(y)
rotateZ(angle)	translateZ(z)	scaleZ(z)

3D变换

2、3D移动 `translate3d`

3D移动在2D移动的基础上多加了一个可以移动的方向，就是 z 轴方向。

`transform:translateX(100px)`: 仅仅是在 x 轴上移动

`transform:translateY(100px)`: 仅仅是在 Y 轴上移动

`transform:translateZ(100px)`: 仅仅是在 Z 轴上移动（注意：`translateZ`一般用`px`单位）

`transform:translate3d(x,y,z)`: 其中 x 、 y 、 z 分别指要移动的轴的方向的距离

因为 z 轴是垂直屏幕，由里指向外面，所以默认是看不到元素在 z 轴的方向上移动

◆ 3D变换

3、透视 perspective

电脑显示屏是个2D的平面，在2D平面要产生近大远小的立体视觉，但是效果只是二维的。

- 如果想要在网页上产生3D效果需要加透视（透视原理：近大远小）。
- 模拟人类的视觉位置，可认为安排一只眼睛去看；
- 透视我们也称为视距：视距就是人的眼睛到屏幕的距离；
- 距离视觉点越近的在电脑平面成像越大，越远成像越小；
- 透视的单位是像素；

透视属性写在被观察元素的父盒子上面的

3D变换

4、translateZ

`transform:translateZ(100px)`: 仅仅是在Z轴上移动。

有了透视（`perspective`属性），就能看到`translateZ` 引起的变化了

`translateZ`: 近大远小

`translateZ`: 往外是正值

`translateZ`: 往里是负值

演示



3D 变换

5、3D旋转

对于元素旋转的方向的判断 我们需要先学习一个左手准则。

左手准则

左手的大拇指指向 x 轴的正方向

其余手指的弯曲方向就是该元素沿着 x 轴旋转的方



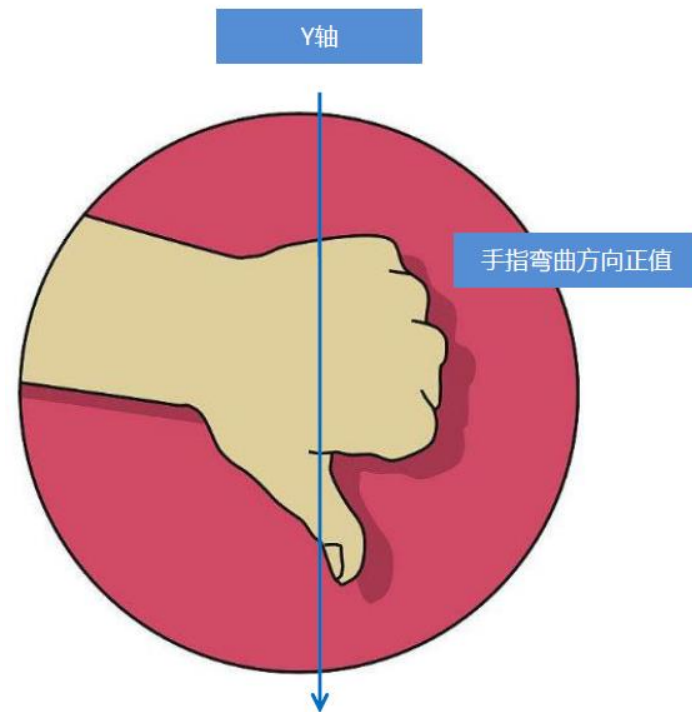
3D 变换

5、3D旋转

左手准则

左手的大拇指指向 y轴的正方向

其余手指的弯曲方向就是该元素沿着y轴旋转的方向（正值）



3D变换

5、3D旋转

3D旋转指可以让元素在三维平面内沿着x轴，y轴，z轴或自定义轴

语法

`transform: rotateX(45deg)`: 沿着x轴正方向旋转 45度

`transform: rotateY(45deg)` : 沿着y轴正方向旋转 45度

`transform: rotateZ(45deg)` : 沿着Z轴正方向旋转 45度

`transform: rotate3d(x,y,z,deg)`: 沿着自定义轴旋转 deg为角度（了解即可）

初始状态

`rotateX()`

`rotateY()`

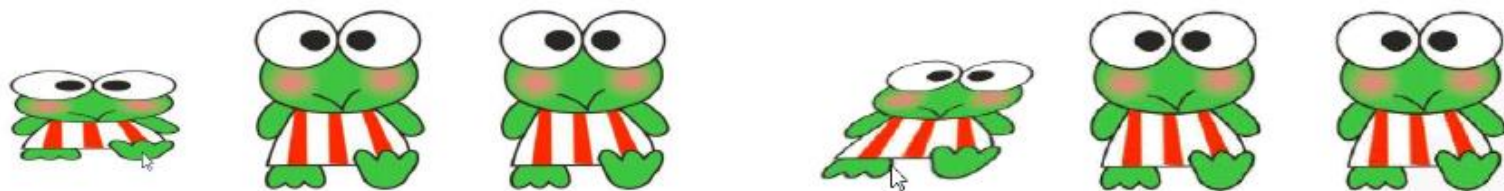
`rotateZ()`

没加透视

3D变换

加透视

rotateX()



rotateY()



3D变换



父容器:



旋转

```
transform-style: preserve-3d;  
transform: rotateY(60deg);
```

舞台:



透视关系

```
眼睛到舞台的距离  
perspective: 100px;
```



3D变换



3D变换

```
<body>
  <div class="stage">
    <div class="box x">
      
    </div>
    <div class="box y">
      
    </div>
    <div class="box z">
      
    </div>
  </div>
</body>
```

```
.stage {
  width: 900px;
  margin: 100px auto;
  perspective: 500px;
}

.box {
  width: 300px;
  height: 300px;
  float: left;
  transition: all 1s linear;
  transform-style: preserve-3d;
}
```

```
img {
  width: 100%;
  height: 100%;
  border: 1px solid #fff;
}

.x:hover {
  transform: rotateX(60deg);
}

.y:hover {
  transform: rotateY(60deg);
}

.z:hover {
  transform: rotateZ(60deg);
}
```

3D 变换

5、3D旋转

`transform: rotate3d(x,y,z,deg)`: 沿着自定义轴旋转deg为角度（了解即可）

xyz是表示旋转轴的矢量，是标示你是否希望沿着该轴旋转，最后一个标示旋转的角度。

- `transform: rotate3d(1,0,0,45deg)` 就是沿着x轴旋转 45deg
- `transform: rotate3d(1,1,0,45deg)` 就是沿着对角线旋转 45deg

3D 变换

6、3D呈现 transform-style

控制子元素是否开启三维立体环境。

`transform-style: flat;` 子元素不开启3d立体空间，默认的取值。

`transform-style: preserve-3d;` 子元素开启立体空间。

注意`transform-style: preserve-3d;` 这段代码写给父级容器，但是它影响的是子盒子。

这个属性很重要。



3D变换

应用3D变换，当鼠标指针悬停时，完成沿Y轴旋转的效果。

```
.box {  
  position: relative;  
  box-sizing: border-box;  
  width: 200px;  
  height: 200px;  
  margin: 100px auto;  
  transform-style: preserve-3d;  
  border: 1px dashed green;  
  perspective: 300px;  
}  
  
img {  
  position: absolute;  
  width: 190px;  
  margin: 3px 3px;  
  border: 1px solid green;  
  /* 过渡效果 */  
  transition: all 0.8s;  
  /* 变换原点位于中心 */  
  /* transform-origin: 50%; */  
}  
  
.box:hover img {  
  /* 鼠标指针悬停，图片沿Y轴方向旋转45度 */  
  transform: rotateY(45deg);  
}
```



```
<div class="box">  
    
</div>
```

过渡

CSS3过渡可以使元素在规定时间内，从一种样式逐渐改变为另一种样式。常见的过渡属性如下：

属性	说明
transition	简写属性
transition-property	哪些CSS属性产生变化，多个属性用逗号隔开，所有属性用all表示
transition-duration	时长
transition-delay	过渡效果何时开始
transition-timing-function	过渡效果的时间曲线

transition-timing-function属性值：

- linear: 匀速
- ease: 慢-快-慢
- ease-in: 慢-快
- ease-out: 快-慢
- ease-in-out: 慢-快-慢

过渡

例1：应用过渡效果，当鼠标指针悬停时，完成开关圆钮滑动和背景色变换效果：



```
<div id="btn">  
  <i id="btn_circle"></i>  
</div>
```

```
* {  
  margin: 0;  
  padding: 0;  
}  
#btn {  
  position: relative; /*父层相对定位*/  
  width: 48px;  
  height: 20px;  
  margin: 100px auto;  
  border: 1px solid #ccc;  
  border-radius: 20px; /*圆角边框*/  
  background-color: #fff;  
  transition: all 1s ease; /*1s过渡所有变化的属性，包括背景颜色*/  
}
```

```
#btn_circle {  
  display: inline-block;  
  position: absolute; /*子层绝对定位*/  
  top: 0;  
  right: 28px; /*圆钮位于开关右侧28像素距离*/  
  height: 18px;  
  width: 18px;  
  border-radius: 9px;  
  border: 1px solid #ddd;  
  background-color: #4cb946;  
  transition: all 0.6s ease; /*0.6s过渡所有变化的属性，包括背景颜色、位置*/  
}  
#btn:hover {  
  background-color: #4cb946; /*开关背景由白变绿*/  
}  
#btn:hover #btn_circle {  
  right: 0; /*变换右侧距离，使圆钮有滑动效果*/  
  background-color: #fff; /*圆钮背景由绿变白*/  
}
```

过渡

例2：应用过渡效果，当鼠标指针悬停时，文字部分与父元素重合



动画

动画 (animation) 是CSS3中具有颠覆性的特征之一，是由静态图片连续播放而成的，静态图片称为关键帧 (Key Frame)。CSS3支持动画创建，可以代替动画图片、Flash以及部分JavaScript动态效果。相比较过渡，动画可以实现更多变化，更多控制，连续自动播放等效果。

动画

1、动画的基本使用

制作动画分为两步：

1. 先定义动画
2. 再使用（调用）动画

① 用keyframes 定义动画（类似定义类选择器）

```
@keyframes 动画名称 {  
    0%{  
        width:100px;  
    }  
    100%{  
        width:200px;  
    }  
}
```

动画

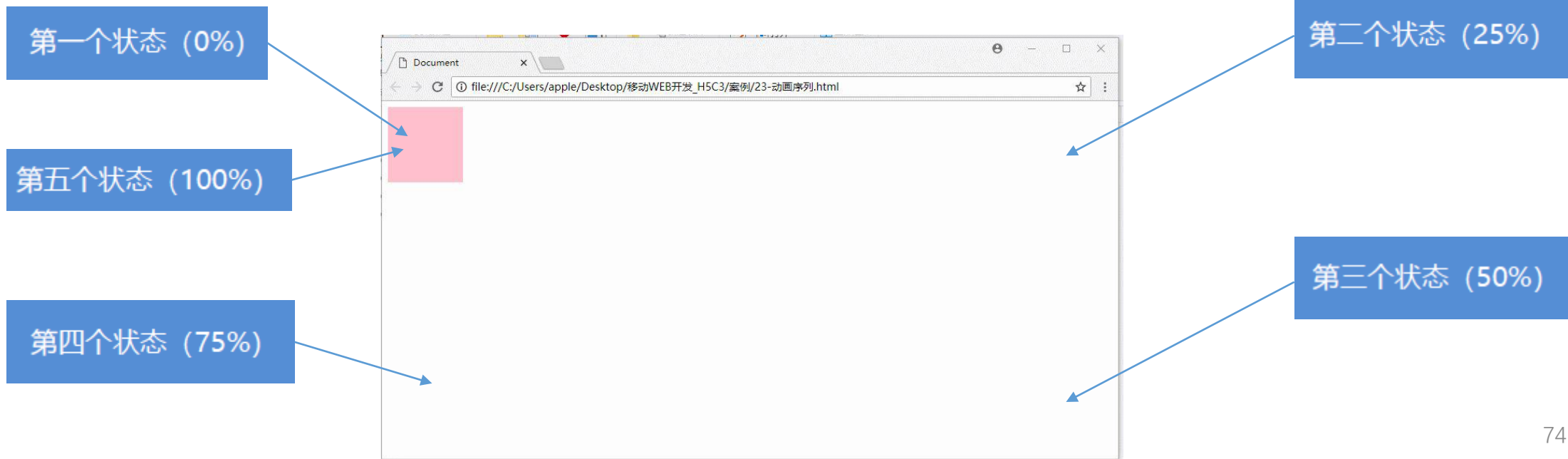
动画序列

0% 是动画的**开始**，100% 是动画的**完成**。这样的规则就是动画序列。

在 **@keyframes** 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

动画是使元素从一种样式逐渐变化为另一种样式的效果。您可以改变任意多的样式任意多的**次数**。

请用百分比来规定变化发生的时间，或用关键词 "**from**" 和 "**to**"，等同于 **0%** 和 **100%**。



动画

② 元素使用动画

```
div {  
    width: 200px;  
    height: 200px;  
    background-color: aqua;  
    margin: 100px auto;  
    /* 调用动画 */  
    animation-name: 动画名称;  
    /* 持续时间 */  
    animation-duration: 持续时间;  
}
```



动画

2、动画常用属性

属性	描述
@keyframes	规定动画。
animation	所有动画属性的简写属性，除了animation-play-state属性。
animation-name	规定@keyframes动画的名称。（必须的）
animation-duration	规定动画完成一个周期所花费的秒或毫秒，默认是0。（必须的）
animation-timing-function	规定动画的速度曲线，默认是“ease”。
animation-delay	规定动画何时开始，默认是0。
animation-iteration-count	规定动画被播放的次数，默认是1，还有infinite
animation-direction	规定动画是否在下一周期逆向播放，默认是“normal”，alternate逆播放
animation-play-state	规定动画是否正在运行或暂停。默认是“running”，还有“paused”。
animation-fill-mode	规定动画结束后状态，保持forwards回到起始backwards

动画

3、动画简写属性

animation: 动画名称 持续时间 运动曲线 何时开始 播放次数 是否反方向 动画起始或者结束的状态;

```
animation: myfirst 5s linear 2s infinite alternate;
```

简写属性里面不包含 animation-play-state

暂停动画: animation-play-state: paused; 经常和鼠标经过等其他配合使用

想要动画走回来，而不是直接跳回来: animation-direction : alternate

想要动画结束后，停在结束位置: animation-fill-mode : forwards

动画

4、速度曲线细节

animation-timing-function: 规定动画的速度曲线，默认是“ease”

值	描述
linear	动画从头到尾的速度是相同的。匀速
ease	默认。动画以低速开始，然后加快，在结束前变慢。
ease-in	动画以低速开始。
ease-out	动画以低速结束。
ease-in-out	动画以低速开始和结束。
steps()	指定了时间函数中的间隔数量（步长）

● 总结

利用CSS3可以实现很多图像处理软件、Flash和JavaScript才能完成的效果。

可以通过border-radius设置圆角边框；box-shadow设置阴影效果；text-shadow设置文字阴影效果。@font-face规则可以实现Web字体的添加，使客户端可以不必预先安装而是从服务器端下载对应的字体文件，完成字体显示。

CSS3支持2D和3D变换，可完成缩放、移动、旋转、倾斜、透视等效果。还可以应用过渡和动画，使元素在两个或多个状态间切换。