

HTML5简介

学前准备

1. 学习要求

- 熟练掌握HTML、DIV、CSS
- 熟练掌握JavaScript

2. 准备工具

- 编辑器：Editplus / Dreamweaver / HBuilder / submitText 等都可以
- 浏览器：IE8、IE9、IE10、IE11、Chrome、Safari、Firefox和Opera

HTML5简介

1. HTML5语义

- 它是一个新版本的HTML语言
- 它是一个大的技术集，在使用它开发网站和应用程序时，时常也会被称为HTML5或者H5开发，H5约等于css+js+html

2. HTML5的与众不同

- 为什么单独列出HTML5，而不单独列出HTML4 / HTML3呢？重大版本更新、颠覆性的技术、更便捷的开发、更好的体验.....
- 示例

```
<form action="https://www.baidu.com" method="get">
  <input type="email" name="userEmail"
value="123456">    <!--设置邮箱验证类型-->
  <input type="submit">
</form>
```

3. HTML5的发展史

- HTML4.01在1999年12月就发布了，之后HTML5的发展非常的缓慢，2013年5月 才正式草案公布HTML5.1，2014年10月HTML5由万维网联盟（W3C）完成标准 制定并正式公布，之后的发展非常迅速。移动设备的普及对HTML5的发展起了推动性作用。

##DOCTYPE简介

1. DOCTYPE就是Document Type Declaration (文档类型声明,又简写DTD) , 一般DOCTYPE会声明到HTML文档的最前面, 因为浏览器需要知道这个文档的类型, 以便于采取什么渲染模式, 这将影响浏览器对css代码的解析, 甚至是js代码
2. document.compatMode这个属性可以查看当前的渲染模式, 它存在两种返回值BackCompat (怪异模式 ---不推荐使用) , CSS1Compat (标准模式---推荐使用)
 - 示例

```
<script>
    //查看浏览器当前的渲染模式, BackCompat(怪异模式),
    CSS1Compat(标准模式)
    console.log(document.compatMode)
</script>
```

3. 如果在文档最前面没有声明DOCTYPE, 则浏览器会默认为怪异模式, 在IE5中只有怪异模式, IE7,8,9中只在理论上存在怪异模式
4. 现在的主流浏览器中怪异模式 与标准模式渲染几乎没有太大的区别, 如 (火狐, 谷歌, IE9以上)

文档的申明

1. DOCTYPE

- HTML4.01需要DTD

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

- HTML5则只需要

```
<!DOCTYPE html>      <!-- 作用与上面一样, 只是简化了写法 -->
```

2. 标签规定编码

- 之前设置编码写法

```
<meta http-equiv=Content-Type
content="text/html; charset=utf-8">
```

- 现在H5页面中设置编码写法

```
<meta charset="UTF-8">
```

3. meta标签还有很多很棒的关于移动端的更新，后面再详细讨论

HTML5语义化介绍

1. 什么是标签语义化

- 根据内容的结构化（内容语义化），选择合适的标签（代码语义化）便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析,如之前的标题元素（h1-h6）

2. 之前布局头部与尾部div的id属性命名

◦ 示例

```
<div id="header">这里是头部信息</div> <!--id为header表示为头部-->
```

```
<div id="footer">这里是尾部信息</div> <!--id为footer表示为尾部-->
```

3. Opera对上百万个页面所用的id与class的统计,如其中头部标签id属性名为header所占比非常高

- <https://dev.opera.com/blog/presentation-html5-and-accessibility-sitting-in-a-tree-4/idlist-url.htm> 页面id统计
- <https://dev.opera.com/blog/presentation-html5-and-accessibility-sitting-in-a-tree-4/classlist-url.htm> 页面class统计

4. HTML5布局头部与尾部的标签,现在新标签header标签用来表示头部

◦ 示例

```
<!--标签语义化-->  
<header>这里是头部信息</header>  
  
<footer>这里是尾部信息</footer>
```

标签语义化的好处

1. 加强代码的可读性，有利于开发与维护
2. 有利于SEO，和搜索引擎建立良好的沟通
3. 有助于爬虫抓取更多的有效信息，爬虫依赖标签确定上下文与标签的权重
4. 利于其它设备的解析（如移动设备）以意义的方式来渲染网页等

新增结构性标签

1. header 定义文档的头部区域，一般用在顶部
2. footer 定义文档的尾部区域，一般用在尾部
3. article 定义页面独立的文章内容，帖子、博客文章、评论、新闻等可以使用
4. nav 定义导航，替代ul和li定义的导航
5. section 定义一块区域，一般用来写主体内容部分
6. aside 定义侧边栏，侧边信息可以使用
7. figure 定义一块独立的内容，通常用来展示图片及其描述
8. figcaption 定义figure的标题
9. iframe 框架， frameset已经被淘汰
10. main标签 规定文档的主要内容。 在一个文档中，不能出现一个以上的元素。

元素不能是以下元素的后代：

、
、
、
或
。

11. 示例

- html代码

```
<body>

    <!--定义头部信息-->
    <header>头部信息</header>

    <!--定义页面独立的文章内容，帖子、博客文章、评论、新闻等可以使用-->
    <article>
        <!--定义导航栏-->
        <nav>
            <ul>
                <li>第1</li>
                <li>第2</li>
                <li>第3</li>
                <li>第4</li>
                <li>第5</li>
            </ul>
        </nav>
    </article>

```

```

<!--定义一块区域，一般用来写主体内容部分-->
<section>
    <!--定义侧边栏，侧边信息可以使用-->
    <aside>侧边栏</aside>
</section>

<!--定义一块独立的内容，通常用来展示图片及其描述-->
<figure>
    <figcaption>这是一个侧边栏图片</figcaption>
    
</figure>

</article>

<!--尾部信息-->
<footer>尾部信息</footer>
</body>

```

- css代码

```

<style>
    *{
        margin: 0px;
        padding: 0px;
    }
    header{
        width: 900px;
        height: 60px;
        background: navajowhite;
        text-align: center;
        line-height: 60px;
        font-size: 24px;
        margin: 0px auto;
    }
    article{
        height: 576px;
        width: 900px;
        background: darkgrey;
        margin: 0px auto;
    }
    footer{
        height: 40px;
        width: 900px;
        background: pink;
    }

```

```
        line-height: 40px;
        text-align: center;
        font-size: 20px;
        margin: 0px auto;
    }
    ul li{
        list-style: none;
        float: left;
        background: dodgerblue;
        height: 40px;
        width: 179px;
        line-height: 40px;
        text-align: center;
        border-right: 1px solid black;
    }
    section{
        height: 400px;
        background: darkturquoise;
    }
    aside{
        width: 200px;
        height: 100%;
        background: darkorange;
    }
    figure img{
        height: 150px;
    }
</style>
```

新增功能性标签

1. mark 突出显示文字

◦ 示例

```
<!--mark 突出显示文字-->
<p>我是小明, <mark>小明爱小红</mark></p>
```

2. meter 刻度百分比

◦ 示例

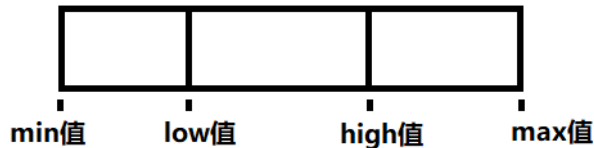
```

<!--刻度百分比-->
<meter value="90" optimum="50" min="0" max="100"
low="30" high="60">90%</meter>

<!--使用小数值，默认的min值为0，max值为1-->
<meter value="0.5">50%</meter>

```

- meter刻度颜色变化说明



low属性--表示被界定为低的价值(界线)

high属性--表示被界定为高的价值(界线)

optimum--定义什么样的度量值是最佳的价值。

如果该值高于 "high" 属性，则意味着值越高越好

如果该值低于 "low" 属性的值，则意味着值越低越好

min值, low值, high值, max值区域分为三个小区域

颜色变化

当value值与optimum值在同一个区域时--绿色

当value值与optimum值在相邻区域时--黄色

当value值与optimum值之间间隔一个区域时--红色

- 注意：在meter标签体中最好写当前的值或者百分比，因为在当浏览器不兼容meter标签时，则会把标签体内容显示给用户。meter标签目前不兼容IE

3. progress 进度条

- 示例

```

<!--进度条 IE9及以下不兼容-->
<progress value="50" min="0" max="100">50%如果不兼容这里
才会显示</progress>
<!--默认min为0，max为1-->
<progress value="0.6">60%</progress>

```

4. ruby 内容+注释 rt注释内容 rp不支持时显示

- 示例

```

<!--ruby内容+注释    rt注释内容    rp不支持时显示-->
<ruby>
  好<rt>hao</rt>
  <rp>当标签不兼容时，这里就会显示</rp>
</ruby>

```

5. wbr 长单词换行的位置

- 示例

```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    p{
      width: 370px;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <!--wbr 长单词换行的位置-->
  <p>如果小明爱小红，那么他必须要打败XML
  <wbr>Http</wbr>Request对象</p>
</body>
```

- 注意：br标签为强制换行，wbr为软换行，如果容纳内容的宽度足够长则不会换行。wbr目前不兼容IE

6. datalist 表示可显示数据的列表，要与input配合使用

- 示例

```
<!--datalist 表示可显示数据的列表，与input配合使用-->
<input type="text" list="list">      <!--设置list属性关联
datalist-->
<datalist id="list">
  <option value="java">java</option>
  <option value="javascript">javascript</option>
  <option value="ruby">ruby</option>
</datalist>
```

- 说明：datalist类似于普通的下拉列表，但它有一个内容检索查找功能

7. address 定义文档或文章的作者/拥有者的联系信息。

- 示例

```
<!--address 定义文档或文章的作者/拥有者的联系信息。-->
<address>
  作者： 小明与小红
  电话： 15552000025
</address>
```

- 说明：如果

元素位于

元素内部，则它表示该文章作者或拥有者的联系信息。address 元素一般添加到网页的头部或底部。

兼容性不是很好的标签

1. dialog 定义一个对话框
2. bdi 单独设置文字摆放方式 open属性
3. details 表示用户要求得到并且可以得到详细信息，要与summary标签一起使用
4. summary 提供标题或者图标，用户点击显示隐藏详细信息，要作为details标签子元素

删除掉的标签

1. acronym、applet、basefont、bgsound、big、blink、center、dir、font、frame、frameset、hgroup、isindex、listing、marquee、multicol、nextid、nobr、noembed、noframes、plaintext、spacer、strike、tt、xmp 等

兼容性处理

1. 兼容性代码 (header,footer等标签在IE8及以下存在兼容性问题)
2. 解决方案一
 - 通过createElement方法创建这些元素
 - 示例

```
<script>
    document.createElement("header");
    document.createElement("article");
    document.createElement("nav");
    document.createElement("section");
    document.createElement("aside");
    document.createElement("footer");
</script>
```

- 注意：元素自带的默认属性是不会被创建的，需要自己添加（如块级属性 display:block）
3. 解决方案二
 - 引用html5shiv.js文件，别人创建的元素封装好的方案一

- 示例

```
<script src="html5shiv.js"></script>
```

- html5shiv官方网址: <https://www.bootcdn.cn/html5shiv/>
- 注意: 对于功能性元素的不兼容, 还是需要通过js处理

兼容性问题说明

1. HTML5有部分内容兼容到IE9, IE8及以下对H5完全不兼容, 后面的内容不再考虑此类浏览器
2. 兼容性查询网站 <http://caniuse.com/>

HTML5 新增属性

1. contenteditable 规定是否允许用户编辑内容

- 示例

```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    div{
      width: 400px;
      height: 200px;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <!--contenteditable 规定是否允许用户编辑内容-->
  <div contenteditable="true"></div>
</body>
```

2. data- 创作者定义的属性。HTML 文档的创作者可以定义 他们自己的属性。必须以 "data-" 开头。

- 示例

```
<!--自定义data-属性-->
<div data-username="小明"></div>
```

- 注意: data-这种形式的属性在一些框架中比较常见

3. hidden 规定该元素是无关的。被隐藏的元素不会显示。

◦ 示例

```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <style>
    div{
      width: 400px;
      height: 200px;
      border: 1px solid black;
      background: red;
    }
  </style>
</head>
<body>
  <!--hidden 规定该元素是无关的。被隐藏的元素不会显示-->
  <div contenteditable="true" hidden></div>
</body>
```

4. 课堂案例一 (hidden属性切换)

5. 课堂案例二 (div双击可编辑)

课后作业

1. 利用课堂学习的结构性标签完成页面布局
2. 写一个动态的滚动条
3. 使用contenteditable属性写一个网页聊天效果

智能表单与拖拽

新增的input表单类型

1. email 表示需要输入邮箱类型

◦ 示例

```
<form action="https://www.baidu.com" method="get">
  <input type="email" name="userEmail"
value="123456">    <!--设置邮箱验证类型-->
  <input type="submit">
</form>
```

2. tel 表示需要输入电话号码（在移动有效果，PC端看不到效果）
3. url 表示需要输入URL地址类型
4. range 出现一个可调节范围值的控件
5. number 出现一个数字微调的控件
6. color 出现一个可选择颜色的控件
7. datetime-local 表示一个时间控件（年/月/日 小时:分钟）
8. date 表示一个时间控件（年/月/日）
9. time 表示一个时间控件（小时:分钟）
10. month 表示一个时间控件（月份）
11. week 表示一个时间控件（周期）

新增的表单属性

1. placeholder 设置输入框的提示信息

- 示例

```
<!--placeholder 设置输入框的提示信息-->
<form action="" method="get">
    邮箱: <input type="email" placeholder="请输入一个邮箱地址">
    <input type="submit">
</form>
```

2. autocomplete 是否让浏览器提示之前的缓存信息

- 示例

```
<!--autocomplete 是否让浏览器提示之前的缓存信息-->
<form action="" method="get">
    <input type="text" name="username"
autocomplete="off">
    <input type="submit">
</form>
```

- 注意autocomplete属性可以让浏览器不提示之前的表单缓存信息，值为off表示关闭提示。填写一些敏感数据时可以使用它
- 浏览器表单信息提示

897258976@qq.com

javascript

3. autofocus 让输入框自动获取焦点

◦ 示例

```
<!--autofocus 让输入框自动获取焦点-->
<form action="" method="get">
  <input type="text" name="username" autofocus>
  <input type="submit">
</form>
```

4. list和datalist 把输入框构造一个选择列表

◦ 示例

```
<!--list属性和datalist 把输入框构造一个选择列表-->
<input type="text" list="abc">
<datalist id="abc">
  <option value="java">java</option>
  <option value="javascript">javascript</option>
  <option value="c++">c++</option>
</datalist>
```

5. required 表示该字段为必填字段

◦ 示例

```
<!--required 表示该字段为必填字段-->
<form action="" method="get">
  邮箱: <input type="email" placeholder="请输入一个邮箱地址" required>
  <input type="submit">
</form>
```

6. Pattern 为该字段设置正则表达式验证

◦ 示例

```
<!--pattern 为该字段设置正则表达式验证-->
<form action="" method="get">
  <input type="text" name="username"
pattern="\w{4}">
  <input type="submit">
</form>
```

7. Formaction 可在表单中设置其它的提交路径

◦ 示例

```
<!--Formaction 可在表单中设置其它的提交路径-->
<form action="https://www.baidu.com" method="get">
  <input type="text" name="username"
pattern="\w{4}">
  <input type="submit">
  <input type="submit" value="提交到草稿箱"
Formaction="https://www.360.cn">
</form>
```

获取页面元素的新方法

1. querySelector() 通过css选择器获取页面元素，返回值是一个对象，IE7及以下不兼容

◦ 示例

```
//querySelector() 通过css选择器来获取页面元素，返回值是一个对象
//如果选择器匹配到了多个元素，也只会第一个元素节点对象
var oDiv = document.querySelector(".btn");
```

2. querySelectorAll(); 通过css选择器获取页面元素，返回值是一个对象集（伪数组），IE7及以下不兼容

◦ 示例

```
//querySelectorAll() 通过css选择器来获取页面元素，返回值是一个对象集（伪数组）
//需要通过下标获取里边的元素
var oElements =
document.querySelectorAll("#box");
console.log(oElements)
console.log(oElements[0]) //获取对象集中的第一个元素
```

3. `getElementsByClassName()`; 通过class名来获取页面元素，E18及以下不兼容

- 示例

```
<body>
  <p class="p1"></p>
  <p class="p1"></p>

  <script>
    //getElementsByClassName() 通过Class名来获取页面元素
    var obj = document.getElementsByClassName("p1");
    console.log(obj);
  </script>
</body>
```

class列表属性操作

1. `classList`属性 此属性可以返回一个对象,这个对象里包含了元素对象的class属性列表,以及操作class属性列表的一些方法

- `length`: class的长度
- `add()`: 添加新的class属性的方法
- `remove()` 删除class属性的方法
- `toggle()` 切换class属性，如果有则移除，没有则添加
- 示例

```
<div class="d1 d2 d3 d4"></div>
<script>
  var oDiv = document.querySelector("div");
  var oList = oDiv.classList; //classList属性返回一个对象
  包含了元素对象的class属性列表,以及操作class属性列表的一些方法

  console.log(oList)
  oList.add("d5"); //添加
  oList.remove("d2") //删除
  oList.toggle("d3") //切换
</script>
```

json数据的解析与序列化

1. 解析: 表示把json格式字符串数据转换成一个js对象
2. 序列化: 表示把一个js对象转换成一个json格式的字符串

3. JSON对象：可以通过JSON对象提供的方法对json数据进行操作

- parse(): 可以把json字符串转换成js对象,注意字符串的属性一定严格加上双引号

```
//解析, 把一个json格式的字符串, 解析成一个js对象
var str = '{"name":"小明"}';
var obj2 = JSON.parse(str);
console.log(obj2);
```

- stringify(): 可以把js对象转换一个标准格式json字符串

```
// 定义一个js对象
var obj = {
  name: '小明',
  girlfriend: {
    name: "小红"
  }
}
//序列化, 把js对象转换成一个json格式的字符串
var jsonStr = JSON.stringify(obj);
console.log(jsonStr)
```

- 4. JSON对象不兼容IE7及以下的浏览器, 可以引入json2.js解决兼容属性问题, 下载json2.js的网站: <http://www.json.org/>
- 5. 注意: 不管在写js对象或者json字符串时, 最后一个属性后面不要加逗号, 否则会出现一些糟糕的情况, 如导致引入的json2.js无效等情况

HTML5拖拽属性

- 1. draggable: 表示是否可以让元素进行拖拽,值类型为布尔值类型

- 示例

```
<div id="d1" draggable="true"></div>    <!-- 让div可拖拽 -->
```

- 2. 注意: 此拖拽属性在火狐中有兼容问题, 需要另做设置才能解决, 下面会提到

HTML5拖拽元素上的拖拽事件

- 1. ondragstart: 拖拽前触发

- 示例


```
<div id="d1" draggable="true"></div>
<script>
    var oD1 = document.querySelector("#d1");
    //ondragstart事件： 拖拽前触发
    oD1.ondragstart = function(){
        this.style.background = "red"
    }
</script>
```

2. ondrag: 拖拽前到 拖拽结束之间，连续触发

◦ 示例

```
<div id="d1" draggable="true"></div>
<script>
    var oD1 = document.querySelector("#d1");
    //ondrag: 拖拽前到 拖拽结束期间，连续触发
    var num = 0;
    oD1.ondrag = function () {
        num++;
        document.title = num;
    }
</script>
```

3. ondragend: 拖拽结束触发

◦ 示例

```
<div id="d1" draggable="true"></div>
<script>
    var oD1 = document.querySelector("#d1");
    //ondragend事件： 拖拽结束触发
    oD1.ondragend = function(){
        this.style.background = "navajowhite"
    }
</script>
```

HTML5拖拽目标元素中的对象（事件给目标元素添加）

1. ondragenter: 进入目标元素触发，类似于mouseover

◦ 示例

```

<!--拖拽元素-->
<div id="d1" draggable="true"></div>
<!--目标元素-->
<div id="box"></div>
<script>
    var oD1 = document.querySelector("#d1");
    var oBox = document.querySelector("#box");
    //Ondragenter: 拖拽元素进入目标元素触发，类似于mouseover
    oBox.ondragenter = function () {
        this.style.background="red"
    }
</script>

```

2. ondragover: 进入目标 到 离开目标期间，连续触发

- 示例

```

<!--拖拽元素-->
<div id="d1" draggable="true"></div>
<!--目标元素-->
<div id="box"></div>
<script>
    var oD1 = document.querySelector("#d1");
    var oBox = document.querySelector("#box");
    var num = 0;
    oBox.ondragover = function (ev) { //Ondragover:
拖拽元素进入目标 到 离开目标元素期间，连续触发
        num++;
        document.title = num;
    }
</script>

```

3. ondragleave: 离开目标元素触发，类似于mouseout

- 示例

```

<!--拖拽元素-->
<div id="d1" draggable="true"></div>
<!--目标元素-->
<div id="box"></div>
<script>
    var oD1 = document.querySelector("#d1");
    var oBox = document.querySelector("#box");
    //Ondragleave: 离开目标元素触发, 类似于mouseout
    oBox.ondragleave = function () {
        this.style.background="gainsboro"
    }
</script>

```

4. Ondrop: 在目标元素上释放鼠标触发

◦ 示例

```

<!--拖拽元素-->
<div id="d1" draggable="true"></div>
<!--目标元素-->
<div id="box"></div>
<script>
    var oD1 = document.querySelector("#d1");
    var oBox = document.querySelector("#box");
    var num = 0;
    oBox.ondragover = function (ev) { //Ondragover:
        进入目标 到 离开目标期间, 连续触发
        ev = ev || window.event; //做事件对象的兼容性处理
        ev.preventDefault(); //取消默认行为
    }
    oBox.ondrop = function () { //拖拽元素在目标
        元素上释放时触发
        alert("小红你好");
    }
</script>

```

- 注意: ondrop事件想要触发, 需要要ondragover中阻止默认行为

dataTransfer对象

1. 事件对象上的dataTransfer属性可以返回dataTransfer对象

◦ 示例

```

<div id="box" draggable="true"></div>
<script>
    var oBox = document.querySelector("#box");
    oBox.ondragstart = function(ev){
        ev = ev || window.event;           //事件对象兼容处理

        var oDataTransfer = ev.dataTransfer //获取dataTransfer对象
        console.log(oDataTransfer)           //在控制台打印dataTransfer对象，查看对象具体的属性
    }
</script>

```

- 注意：dataTransfer对象上有如下操作方法与属性

2. setData(): 可以设置数据

- 示例

```

<div id="box" draggable="true"></div>      <!--拖拽元素-->
<div id="d1"></div>                        <!--目标元素-->
<script>
    //获取到页面的#box元素对象
    var oBox = document.querySelector("#box");
    var oD1 = document.querySelector("#d1");
    oBox.ondragstart = function(ev){
        ev = ev || window.event;
        ev.dataTransfer.setData("name", oBox); //通过dataTransfer对象上setData方法可以设置数据
    }
</script>

```

- 注意：设置的数据一般会在目标对象的ondrop事件中接收

3. getData(): 可以通过key, 获取到value

- 示例

```

<div id="box" draggable="true"></div>      <!--拖拽元素-->
<div id="d1"></div>                        <!--目标元素-->
<script>
    //获取到页面的#box元素对象
    var oBox = document.querySelector("#box");
    var oD1 = document.querySelector("#d1");
    oBox.ondragstart = function(ev){
        ev = ev || window.event;

```

```

        ev.dataTransfer.setData("name", oBox); //通过
dataTransfer对象上setData方法可以设置数据
    }
    od1.ondragover = function(ev){
        ev = ev || window.event;
        ev.preventDefault();
    }
    od1.ondrop = function (ev) {
        ev = ev || window.event;
        var value = ev.dataTransfer.getData("name");//通过
dataTransfer对象上getData方法可以获取数据
        console.log(value);
    }
</script>

```

4. effectAllowed: 设置光标样式(none, copy, copyLink, copyMove, link, linkMove, move, all 和 uninitialized)

◦ 示例

```

<div id="box" draggable="true"></div>      <!--拖拽元素-->
<div id="d1"></div>                        <!--目标元素-->
<script>
    //获取到页面的#box元素对象
    var oBox = document.querySelector("#box");
    var od1 = document.querySelector("#d1");
    oBox.ondragstart = function(ev){
        ev = ev || window.event;
        ev.dataTransfer.effectAllowed = "link"; //设置
光标的样式
    }
    //需要给目标元素添加ondragover事件
    od1.ondragover = function(ev){
        ev = ev || window.event;
        ev.preventDefault();
    }
</script>

```

5. setDragImage(): 指定拖拽助手 (即设置拖拽时的元素样子)

◦ 示例

```

<div id="box" draggable="true"></div>
<div id="d1"></div>
<!--为了显示效果更好些，可以把图片位置到边界外隐藏掉-->

```

```



<script>
    var oBox = document.querySelector("#box");
    var oD1 = document.querySelector("#d1");
    var oImg = document.querySelector("img");
    oBox.ondragstart = function(ev){
        ev = ev || window.event;
        //setDragImage(指定拖拽助手, 相对拖拽助手的横向偏移, 相对
        拖拽助手的纵向偏移);
        ev.dataTransfer.setDragImage(oImg, 50, 50);
    }
</script>

```

6. Files: 此属性返回一个对象, 里面包含了放置外部文件的列表

- 示例

```

<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        #d1{
            width: 300px;
            height: 300px;
            background:#cccccc;
            margin: 100px auto;
        }
    </style>
</head>
<body>

<div id="d1"></div>
<script>
    var oD1 = document.querySelector("#d1");
    oD1.ondragover = function(ev){
        ev = ev || window.event;
        ev.preventDefault();
    }
    oD1.ondrop = function (ev) {
        ev = ev || window.event;
        ev.preventDefault();
    }
    //阻止默认行为,
    防止浏览器在新的窗口显示内容

```

```

        var oFiles = ev.dataTransfer.files; //获取到外部拖拽
进来的文件列表
        console.log(oFiles) //在控制台打印查
看文件列表
        var oFr = new FileReader(); //创建一个读取文
件的对象（FileReader对象）
        oFr.readAsDataURL(oFiles[0]); //读取外部文件列
表中的第1一个元素做为URL
        oFr.onload = function () { //当文件读取完成
时，则会触发onload事件
            var oImg = new Image(); //创建一个Image
对象
            oImg.src = this.result; //把读取的结果赋
值给Image对象的src属性
            document.body.appendChild(oImg); //把Image对
象添加到页面
        }
    }
</script>

```

FileReader对象

1. readAsDataURL 参数为要读取的文件对象，将文件读取为DataUrl

```

var oFiles = ev.dataTransfer.files; //获取到外部拖拽进来的文件列表
var fr = new FileReader(); //创建读取文件的对象
FileReader
fr.readAsDataURL(oFiles[0]); //读取文件列表第一个元素

```

2. onload 当读取文件成功完成时则会触发此事件

- this.result 可以获取读取的文件数据
- 如果文件是图片, 则会返回base64格式的图片数据
- 注意：图片拖放浏览器时，浏览器会新开一窗口来显示，可以在ondrop和ondragover事件中阻止添加ev.preventDefault();阻止默认行为
- 具体示例请参考上面

课后作业

1. 模拟回收站
2. 模拟购物车
3. 图片上传预览

CSS选择器

元素选择器

1. `*{ sRules }` 通配选择符，遍历并命中文档中所有的元素，出于性能考虑，需酌情使用

- 示例

```
<style>
  /*通配选择符，选择所有的元素，不太建议使用*/
  *{
    background: red;
  }
  /*匹配div下面的所有的元素*/
  div *{
    background: red;
  }
</style>
```

2. `E { sRules }` 元素选择符，以文档语言元素对象类型作为选择符
3. `E#myid { sRules }` ID选择符，通过ID匹配页面元素
4. `E.myclass { sRules }` class选择符，通过class名匹配页面中元素

层级(关系)选择器

1. `E F { sRules }` 包含(后代)选择符，匹配所有E元素下指定的后代元素F
2. `E>F { sRules }` 子代选择符，匹配所有E元素下指定的子代元素F，而不能命中孙辈
3. `E+F { sRules }` 相邻选择符，命中符合条件E元素的下一个相邻的兄弟元素 F
4. `E~F { sRules }` 兄弟选择符，命中所有符合条件的E元素的兄弟元素F，而不强制是紧邻的元素

属性选择器

1. `E[attr]` 选择具有att属性的E元素
2. `E[attr="value"]` 选择att属性=value的E元素
3. `E[attr^="value"]` 选择具有att属性且属性值以val开头的字符串的E元素。
4. `E[attr$="value"]` 选择具有att属性且属性值为以val结尾的字符串的E元素。
5. `E[attr*="value"]` 选择具有att属性且属性值为包含val的字符串的E元素

6. E[attr~="value"] 选择具有att属性且属性值是用空格分隔的字词列表，其中一个等于val的 E元素（包含只有一个值且该值等于val的情况）

```
<style>
  li[class ~= 'one']{           /*匹配结果是第一个和第二个li*/
    background:red;
  }
</style>

<ul id="list">
  <li class="one" name="hello">第1个</li>
  <li class="two one">第2个</li>
  <li class="three">第3个</li>
  <li class="four">第4个</li>
  <li class="five" name="yang">第5个</li>
</ul>
```

7. E[attr|=“value”] 选择具有att属性且属性值为以val开头并用连接符“-”分隔的字符串的E元素（如果属性值仅为val，也将被选择）

```
<style>
  li[class |= 'one']{ /*匹配的结果是第一个和第三个li*/
    background:red;
  }
</style>

<ul id="list">
  <li class="one" name="hello">第1个</li>
  <li class="two one">第2个</li>
  <li class="one-two">第2个</li>
  <li class="three">第3个</li>
  <li class="four">第4个</li>
  <li class="five" name="yang">第5个</li>
</ul>
```

结构性伪类选择器

1. E:nth-child(n) 匹配父元素的第n个子元素E，假设该子元素不是元素类型E，则选择无效

```
<style>
  /*匹配第1个子元素，且这个子元素的类型为p元素，此操作会限制选择子
  元素的类型只能为p元素*/
  #d1 p:nth-child(1){
```

```

        background: red;
    }
    /*匹配第1个子元素，不限制类型*/
    #d1 *:nth-child(1){
        background: red;
    }
    /*匹配第奇数个p子元素*/
    #d1 p:nth-child(odd){
        background: red;
    }
    /*匹配第偶数个p子元素*/
    #d1 p:nth-child(even){
        background: dodgerblue;
    }
    /*匹配第3n个元素，n从1开始数*/
    #d1 p:nth-child(3n){
        background: red;
    }
}
</style>

<body>
    <div id="d1">
        <h1>hhhhhhhhh</h1>
        <p>第1</p>
        <p>第2</p>
        <p>第3</p>
        <p>第4</p>
        <p>第5</p>
        <p>第6</p>
    </div>
</body>

```

2. E:nth-last-child(n) 与E:nth-child(n) 类似，但它是从后向前选择元素
3. E:nth-of-type(n) 匹配同类型中的第n个同级兄弟元素E

```

<style>
    p:nth-of-type(1){    /*匹配结果是标签体内容为ppppp和第一的p标
    签*/
        background: red;
    }
</style>

<body>
    <p>pppppppp</p>

```

```

<div id="d1">
  <h1>hhhhhhhhh</h1>
  <p>第1</p>
  <p>第2</p>
  <p>第3</p>
  <p>第4</p>
  <p>第5</p>
  <p>第6</p>
</div>
</body>

```

4. E:nth-last-of-type(n) 与E:nth-of-type(n)类似，但从后向前开始选择元素
5. E:first-child 匹配父元素的第一个子元素E，如果第一个子元素类型不是E，则匹配无效

```

<style>
  /*匹配父元素的第一个子元素,且第一个子元素类型需要为p*/
  #d1 p:first-child{          /*匹配结果是没有匹配到任何元素*/
    background: red;
  }
</style>

<div id="d1">
  <h1>hhhhhhhhh</h1>
  <p>第2</p>
  <p>第3</p>
  <p>第4</p>
  <p>第5</p>
  <p>第6</p>
</div>

```

6. E:last-child 匹配父元素的最后一个子元素E。与E:first-child类似，但从后向前开始选择元素
7. E:first-of-type 匹配同类型中的第一个同级兄弟元素E。

```

<style>
  /*匹配父元素#d1下的第一个p元素*/
  #d1 p:first-of-type{      /*匹配结果是 匹配到#d1下的第一个p标
  签*/
    background: red;
  }
</style>

<div id="d1">

```

```

<h1>hhhhhhhhh</h1>
<p>第1</p>
<p>第2</p>
<p>第3</p>
<p>第4</p>
<p>第5</p>
<p>第6</p>
</div>

```

8. E:last-of-type 匹配同类型中的最后一个同级兄弟元素E。与E:first-of-type类似，但从后向前开始选择
9. E:only-child 匹配父元素仅有的一个子元素E，表示父元素下面只有一个子元素且为E元素（子节点不包含文本节点）

```

<style>
  p span:only-child{ /*匹配结果是 只匹配到p2中的span元素*/
    background: red;
  }
</style>

<body>
  <p id="p1">
    <span>span</span>
    <strong>strong</strong>
  </p>
  <p id="p2">
    <span>span</span>
  </p>
</body>

```

10. E:only-of-type 表示在同级元素中，E元素是唯一的，选择E元素。表示父元素下面可以多个子元素，但只能有一个E元素（子节点不包含文本节点）

```

<style>
  p span:only-of-type{ /*匹配结果是 匹配p2与p1中的span元素*/
    background: red;
  }
</style>

<body>
  <p id="p1">
    <span>span</span>

```

```
        <strong>strong</strong>
    </p>
    <p id="p2">
        <span>span</span>
    </p>
</body>
```

伪类选择器(其它)

1. E:empty 匹配没有任何子元素（其中子元素包括text节点）的元素E

```
<style>
    li:empty{                               /*匹配结果：为第三个li元素*/
        background: red;
    }
</style>

<ul>
    <li><span>span</span></li>
    <li>12345</li>
    <li></li>
</ul>
```

2. E:target 表示当前的URL片段的元素类型，这个元素必须是E

```
<style>
    /*需要在URL地址后加上#box看到效果*/
    #box:target{
        width: 100px;
        height: 100px;
        background: red;
    }
</style>
<div id="box"></div>
```

3. E:disabled 表示不可点击的表单控件

```

<style>
    /*E:disabled      表示不可点击被禁用的表单控件*/
    input:disabled{
        background: red;
    }
</style>
<body>
    <input type="text" value="123456" disabled>
    <input type="text" value="1234">
</body>

```

4. E:enabled 表示可点击的表单控件

```

<style>
    /*E:enabled      表示可点击的表单控件*/
    input:enabled{
        background: red;
    }
</style>
<body>
    <input type="text" value="123456" disabled>
    <input type="text" value="1234">
</body>

```

5. E:checked 表示已选中的checkbox或radio

```

<style>
    /*E:checked      表示已选中的checkbox或radio*/
    input:checked{
        width: 100px;
        height: 100px;
    }
</style>
<body>
    <input type="checkbox" value="小明" checked>小明
</body>

```

6. E:first-line 表示E元素中的第一行

```

<style>
    div{
        width: 200px;
        height: 100px;
        border: 1px solid black;
    }

```

```

    }
    /*E:first-line      表示E元素中的第一行*/
    div:first-line{
        background: red;
    }
</style>
<div contenteditable="true">
    divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
    divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
    divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
</div>

```

7. E:first-letter 表示E元素中的第一个字符

```

<style>
    div{
        width: 200px;
        height: 100px;
        border: 1px solid black;
    }
    /*E:first-letter    表示E元素中的第一个字符*/
    div:first-letter{
        background: red;
    }
</style>
<div contenteditable="true">
    哈哈divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
    divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
    divdivdivdivdivdivdivdivdivdivdivdivdivdivdivdiv
</div>

```

8. E::selection 表示E元素在用户选中文字时

```

<style>
    div{
        width: 200px;
        height: 100px;
        border: 1px solid black;
    }
    /*E::selection      表示E元素在用户选中文字时*/
    div::selection{
        background: red;
    }
</style>
<div contenteditable="true">

```

[illegible]

9. E::before 生成内容在E元素前，需要与content属性一起使用

```
span::before{ /*在span内容前面添加一个内容，定义为一个  
宽高都为50px的盒子*/  
    content: "";  
    display: block;  
    width: 50px;  
    height: 50px;  
    background: red;  
}
```

10. `E::after` 生成内容在E元素后，与`E::before`类似，在元素内容后添加内容

11. not(s) 表示E元素不被匹配

```
<style>
    /*匹配ul下所有的li子元素，除了第三个子元素*/
    ul li:not( :nth-child(3) ){
        background:red;
    }
</style>
<ul>
    <li>第1</li>
    <li>第2</li>
    <li>第3</li>
    <li>第4</li>
    <li>第5</li>
</ul>
```

文字与边框

新增颜色模式(rgba)

1. `rgba(r,g,b,a)` 用于设置颜色

- o r Red 红 值范围 0-255
- o g Green 绿 值范围 0-255
- o b Blue 蓝 值范围 0-255
- o a Alpha 透明 值范围 0-1

2. 示例

```
<style>
/*背景透明，文字不透明*/
#box{
  width: 100px;
  height: 100px;
  background: rgba(251,4,29,0.1);
  font: 24px/100px 微软雅黑;
  text-align: center;
}
/*背景不透明，文字透明*/
#d1{
  width: 100px;
  height: 100px;
  background: rgba(251,4,29,1);
  font: 26px/100px 微软雅黑;
  text-align: center;
  color: rgba(0,0,0,0.4);
}
</style>
```

文字阴影

1. text-shadow(x y blur color, ...)

- x 横向偏移
- y 纵向偏移
- blur 模糊距离
- color 阴影颜色
- 示例

```
#d1{
  width: 500px;
  height: 500px;
  border: 1px solid black;
  margin: 50px auto;
  font: 90px/500px 微软雅黑;
  text-align: center;
}
#d1:hover{
  text-shadow: -3px -3px 10px red;
}
```

2. 阴影叠层：可以同时设置多个阴影，以逗号分开

```
#d1:hover{
    text-shadow: -3px -3px 10px red, -6px -6px 10px blue,
    -9px -9px 10px green;
}
```

3. 设置文字模糊

```
#d1:hover{
    color: rgba(0,0,0,0); /*把文字设置成透明*/
    text-shadow: 0px 0px 50px #000; /*用一个模糊的阴影代替文字*/
}
```

文字描边

1. -webkit-text-stroke (宽度 颜色)

2. 示例

```
#d1{
    width: 500px;
    height: 500px;
    border: 1px solid black;
    margin: 50px auto;
    font: 90px/500px 微软雅黑;
    text-align: center;
}
#d1:hover{
    -webkit-text-stroke: 2px red;
}
```

文字排列

1. direction 设置文字的排列方式

- rtl属性值 从右向左排列
- ltr属性值 从左向右排列
- 需要与unicode-bidi: bidi-override 一块使用，表示严格按照 <' [direction](#) '> 属性的值重排序。忽略隐式双向运算规则

2. 示例

```
/*设置文字从左到右排列*/
p{
  width: 200px;
  border: 1px solid black;
  direction: rtl;
  unicode-bidi: bidi-override;
}
```

文字超出显示省略号

1. text-overflow

- clip属性值 无省略号
- ellipsis属性值 超出显示省略号（需要与white-space:nowrap和overflow:hidden一起使用）

2. 示例

```
<style>
/*设置文字超出显示省略号*/
p{
  width: 200px;
  border: 1px solid black;
  white-space: nowrap;          /*强制上所有的文字在一行显示*/
  overflow: hidden;            /*隐藏掉超出的内容*/
  text-overflow: ellipsis;     /*显示省略号*/
}

/*
  设定在第三行出现省略号
  */
p{
  width: 200px;
  border: 1px solid black;
  overflow: hidden;            隐藏掉超出的内容
  text-overflow: ellipsis;     显示省略号

  display: -webkit-box;
  -webkit-line-clamp: 3;       块元素显示的文本行数
  -webkit-box-orient: vertical;
}
*/

</style>
<p>我是小明,我是小明,我是小明,我是小明,我是小明</p>
```

圆角 (border-radius)

1. 参数用法

- 只有一个参数值时，表示四个角一样
 - border-radius: 一样
- 只有二个参数值时，设置的对角
 - border-radius: 左上&右下 右上&左下
- 只有三个参数值时，依次设置的是左上 右上&左下 右下
 - border-radius: 左上 右上&左下 右下
- 有四个参数值时，以顺时针分别设置四个角
 - border-radius: 左上 右上 右下 左下

2. 长度单位：长度单位可以使用px或者%，

3. 示例

```
<style>
  #d1{
    width: 100px;
    height: 100px;
    background: red;
    /*border-radius: 50px;          /*表示全部的四个角*/
    /*border-radius: 40px 20px 10px 5px;  /*左上，右上，右
下，左下*/
    /*border-radius: 40px 20px 10px;  /*左上，右上与左下，右下
*/
    /*border-radius: 40px 10px;  /*左上与右下，右上与左下*/
    border-radius: 50%;          /*单位使用百分比*/
  }
</style>
<body>
  <div id="d1"></div>
</body>
```

兼容性前缀(需要记住)

1. -webkit-常用于兼容chrome浏览器
2. -moz-常用于兼容火狐
3. -o-常用于兼容opera
4. -ms-常用于兼容IE

渐变&背景&过渡

线性渐变（linear-gradient）

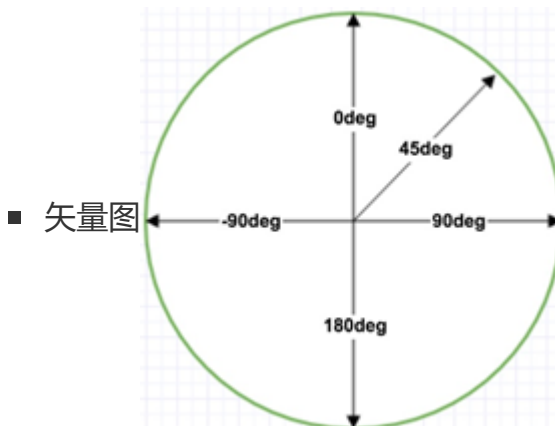
1. linear-gradient([<起点> || <角度>]? <点>, <点>...)

2. 参数说明

- 起点：从什么方向开始渐变 默认：top
 - 可以有的值 left, top, right, bottom, left top, right bottom等等
 - 示例

```
#box{
  width: 300px;
  height: 300px;
  /*向右下角的方向渐变,红色到绿色的渐变*/
  background-image: linear-gradient(to right bottom,
  red , green);
}
```

- 角度：从什么角度开始渐变
 - 属性值以xxx deg的形式，渐变的方向参考下图，箭头表示渐变的方向



- 示例

```
#box{
  width: 300px;
  height: 300px;
  /*使用角度设置渐变的方向，由下往上渐变*/
  background-image: linear-gradient(0deg, red,
  green);
}
```

- 点：渐变点的颜色和位置，其中位置为可选项，位置可以使用百分比或者像素

```
#box{
  width: 300px;
  height: 300px;
  /*从右向左渐变，从右往左50px开始black往white渐变，在250px渐变完成*/
  background-image: linear-gradient(270deg, black 50px,
  white 250px);
}
```

带有透明度的渐变

```
<style>
  #box{
    width: 300px;
    height: 300px;
    /*带透明度的渐变*/
    background-image: linear-gradient(270deg, rgba(255,0,0,1)
    , rgba(0,0,0,0.2));
  }
</style>
<div id="box"></div>
```

重复渐变 (repeating-linear-gradient)

1. 重复渐变使用的属性是 repeating-linear-gradient
2. 示例

```
<style>
  #box{
    width: 300px;
    height: 300px;
    /*重复的的渐变效果(repeating-linear-gradient)*/
    background-image: repeating-linear-gradient(270deg,
    black , white 20%);
  }
</style>
<div id="box"></div>
```

径向渐变 (radial-gradient)

1. radial-gradient([<起点>]? [<形状> || <大小>],)? <点>, <点>...);

2. 参数说明

- 起点：可以是具体的数值或者百分比，使用at操作符

```
#box{
  width: 300px;
  height: 300px;
  /*径向渐变，起点位置在横向80%，纵向20%的位置*/
  background-image: radial-gradient(at 80%
20%,red,black,white);
}
```

- 大小：可以设置整个渐变的大小

```
#box{
  width: 300px;
  height: 300px;
  /*径向渐变，横向与纵向半径大小都为50px*/
  background-image: radial-gradient(50px 50px at 80%
20%,red,black,blue);
}
```

- 形状： ellipse、 circle
 - ellipse 椭圆形（默认）
 - circle 圆形
 - 示例

```
#box{
  width: 300px;
  height: 350px;
  /*径向渐变，渐变形状为圆*/
  background-image: radial-gradient(circle at 50%
50%,red,black,blue);
}
```

重复径向渐变 (repeating-radial-gradient)

```
#box{
  width: 300px;
  height: 300px;
  background-image: repeating-radial-gradient(red,orange
2%,yellow 4%,green 6%, blue 8% ,indigo 10%,violet 12%);
  border-radius: 250px;
  margin: 2px auto;
}
```

图片背景

1. 背景图片回顾

```
#box{
  width: 500px;
  height: 500px;
  border: 1px solid black;

  /*background: 背景颜色 背景图片 背景坐标/大小 不平铺;*/
  background: #cccccc url("../imgs/Google.jpg") 0px
0px/100% 100% no-repeat;
}
```

2. 图片背景大小 (background-size: x y)

- background-size:100% 100%
- Cover: 保持图像的纵横比,将图像缩放到完全覆盖容器,背景图像有可能超出容器。



- Contain 保持图像的纵横比,并将图像缩放到宽度或高度与容器的宽度或高度相等,背景图像始终被包含在容器内。



3. 设置多图片背景

```
#box{
  width: 500px;
  height: 500px;
  border: 20px solid black;
  padding: 40px;
  background: #cccccc no-repeat;
  background-image: url("../imgs/Google.jpg"),url("../imgs/
图标1.jpg");
}
```

4. 背景图基点 (background-origin)

- border-box: 从border区域开始显示背景图片
- padding-box: 从padding区域开始显示背景图片
- content-box: 从content区域开始显示背景图片
- 示例

```
#box{
  width: 500px;
  height: 500px;
  border: 20px solid black;
  background: #cccccc url("../imgs/Google.jpg") no-
repeat;
  padding: 40px;
  /*background-origin: border-box;      !*从border区域开始
显示背景图片*!*/
  /*background-origin: padding-box;     !*从padding区域开
始显示背景图片*!*/
  background-origin: content-box;      /*从content区域开始
显示背景图片*/
}
```

5. 图片背景裁剪 (background-clip)

- border-box: 从border区域向外裁剪背景 (默认)
- padding-box: 从padding区域向外裁剪背景

```
#box{
  width: 500px;
  height: 500px;
  border: 20px solid transparent;    /*把边框设置成无色，方便看效果*/
  padding: 40px;
  background: #cccccc url("../imgs/Google.jpg") no-repeat;
  background-position: -20px -20px; /*设置背景图坐标超出内边距20px,从这里可以知道背景图可以扩展内边距里*/
  background-clip: padding-box;      /*切剪掉超出内边距部分的背景图*/
}
```

- content-box: 从content区域向外裁剪背景

```
#box{
  width: 500px;
  height: 500px;
  border: 20px solid black;
  padding: 40px;
  background: #cccccc url("../imgs/Google.jpg") no-repeat;
  background-clip: content-box;      /*剪切掉超出内容区域的背景图*/
}
```

- text: 从前景内容的形状 (比如文字) 作为裁剪区域向外裁剪

```
<style>
#box{
    width: 500px;
    height: 500px;
    border: 20px solid black;
    padding: 40px;
    font: 100px/500px "Agency FB";
    background: #cccccc url("../imgs/Google.jpg") 0 0/
100% 100% no-repeat;
    -webkit-background-clip: text; /*剪切内容形状*/
    color: transparent; /*把文字设置成无色*/
}
</style>
<div id="box">小明是谁啊</div>
```

遮罩图片 (mask)

1. 相关作用：可以让背景图片只在遮罩图片区域显示，用法与background样式属性类似

2. 相关样式属性

- mask-image 遮罩图片路径
- mask-position 位置
- mask-repeat 是否平铺
- mask-size 大小
- mask-clip 裁剪
- mask-origin 基点

3. 示例

```
#box{
    width: 600px;
    height: 600px;
    border: 1px solid black;
    background: url("imgs/2.jpg") 0px 0px/100% 100% no-
repeat;
    /*设置遮罩图片*/
    -webkit-mask: url("imgs/mask.png") 0px 0px/50px 50px no-
repeat;
    mask: url("imgs/mask.png") 0px 0px/50px 50px no-repeat;
}
```

4. 注意：在谷歌中需要加上兼容前缀-webkit-才能有效果。使用遮罩图片最好使用.png格式图片。目前还不能兼容IE浏览器

过渡

1. transition做为一个复合属性,包含有如下属性

- transition-property 过渡的样式
- transition-duration 过渡时间
- transition-delay 过渡延迟时间
- transition-timing-function 过渡的形式 (可以使用贝塞尔曲线<http://cubic-bezier.com/>)
- 示例

```
<style>
  #box{
    width: 100px;
    height: 100px;
    background: orange;
    /*transition: 0.1s;*/
    /*transition: width 2s;           **指定过渡的
样式, 默认值all**/
    /*transition: width 2s, height 5s; **指定多样式
过渡**/
    /*transition: width 2s, height 5s 10s; **设置过渡的
延迟,height过渡延迟10s**/
    /*transition: width 2s ease-out;   **设置过渡的
形式**/
    /*使用贝塞尔曲线*/
    transition: width 2s cubic-
bezier(.56,-0.54,.39,1.54);
  }
  #box:hover{
    width: 500px;
    height: 600px;
    background: red;
  }
</style>
<body>
  <div id="box"></div>
</body>
```

2. 过渡完成事件

- 当元素过渡完成时触发

- 谷歌浏览器中写法：
obj.addEventListener('webkitTransitionEnd',function(){},false);
- 火狐浏览器中写法： obj.addEventListener('transitionend',function(){},false);
- 示例

```
<style>
  #box{
    width: 100px;
    height: 100px;
    background: red;
    transition: width 1s, background 3s;
  }
  #box:hover {
    width: 600px;
    background: blue;
  }
</style>
<body>

  <div id="box"></div>
  <script>
    var oBox = document.querySelector("#box");

    // 在谷歌中的写法
    oBox.addEventListener('webkitTransitionEnd',
function () {
    alert(1);
  },false)

    // 在火狐中的写法
    oBox.addEventListener('transitionend',function ()
{
    alert(2)
  },false)
  </script>
</body>
```

- 注意：事件触发是针对样式属性的，比如元素中有两个样式属性完成过渡，则事件会触发两次

2D与3D转换

转换属性 (transform)

- transform 让元素应用 2D 或 3D 转换

```
div{
  transform: rotate(45deg);    /*让div旋转45度*/
}
```

平移 (translate)

1. translate(x, y)方法 根据X轴和Y轴位置给定的参数，从当前元素位置发生移动
2. 如果只有一个参数，则表示在x轴上平移，第二参数默认为0
3. 示例

```
<style>
  #box{
    width: 500px;
    height: 500px;
    border: 1px solid black;
    margin: 100px auto;
  }
  #box .d1{
    width: 100px;
    height: 100px;
    background: red;
    margin: 200px auto;
    transition: 1s;
  }

  #box: hover .d1{
    /*transform: translate(50px, 200px);  !*沿着x轴平移
    50px, 沿着y轴平移200px*!*/
    /*transform: translateX(200px);    !*沿着x轴平移
    200px*!*/
    transform: translateY(200px);    /*沿着y轴平移200px*/
  }
</style>
<body>
  <div id="box">
    <div class="d1"></div>
  </div>
</body>
```

旋转 (rotate)

1. rotate(xxxdeg)方法, 让元素顺时针旋转xxx度,如果参数xxxdeg为负数则逆时针旋转
2. rotateX() 绕着X轴旋转, 默认绕着中心点旋转
3. rotateY() 绕着y轴旋转
4. 示例

```
#box:hover .d1{  
    /*transform: rotate(90deg);    !*顺时针旋转90度, 如果为负数则是逆  
    时针旋转*!*/  
    /*transform: rotateX(60deg);    !*顺时针绕着x轴旋转60度*!*/  
    /*transform: rotateY(50deg);    !*顺时针绕着y轴旋转50度*!*/  
}
```

缩放倍数 (scale)

1. scale(x, y)方法, 使用元素缩放, 指定缩放倍数, 如果只一个参数时, 则同时表示宽高的缩放, 值可以是小数
2. scaleX()方法 宽度缩放
3. scaleY()方法 高度缩放
4. 示例



```
#box:hover .d1{  
    /*transform: scale(2,1.5);*/  
    /*transform: scale(2);          !*放大2倍*!*/  
    /*transform: scaleX(0.2);*/     /*宽度缩小到0.2倍*/  
    transform: scaleY(0.5);        /*高度缩小到0.5倍*/  
}
```

倾斜 (skew)

1. skew(x [,y])方法, 让元素倾斜, 两个参数分别表示x轴与y轴, 如果第二个参数没有则默认为0
2. skewX() 沿X轴倾斜
3. skewY() 沿Y轴倾斜
4. 示例

```
#box:hover .d1{
    transform: skew(30deg, 45deg);    /*绕着x轴倾斜30度，绕着y轴倾
斜45度*/
    /*transform: skewY(45deg);        !*绕着y轴倾斜45度!*/*
    /*transform: skewX(30deg);        !*绕着x轴倾斜30度!*/*
}
```

矩阵函数 (matrix)

1. matrix(a,b,c,d,e,f) 矩阵函数，使用matrix函数即可以实现上面平移，旋转，缩放，倾斜四个操作
2. matrix矩阵函数，2D矩阵形式为3*3，参数分别对应如下图，第一行(a c e)影响x轴，第二行(b d f)影响y轴，第三行表示z轴，这里2D, z轴默认用(0 0 1)
3.  
4. 平移 (在矩阵中影响平移的值为 e f)

```
#box:hover .d1{
    transform: matrix(1,0,0,1,50,200);    /*沿着x轴平移50px，沿着y
轴平移200px*/
}
```

5. 缩放 (在矩阵中影响缩放的值为 a d)

```
#box:hover .d1{
    transform: matrix(2,0,0,3,0,0);    /*宽度放大2倍，高度放大3倍
*/
}
```

6. 旋转 (在矩阵中影响旋转的值为a b c d -- matrix(cos(x), sin(x), -sin(x), cos(x), 0, 0))

```
<style>
#box{
    width: 500px;
    height: 500px;
    border: 1px solid black;
    margin: 100px auto;
}
#box .d1{
    width: 100px;
    height: 100px;
    background: red;
}
```



```

        margin: 200px auto;

        background-image: linear-gradient(red,blue);

        transition: 1s;
    }
</style>
<body>
    <div id="box">
        <div class="d1"></div>
    </div>
    <script>
        document.onclick = function () {
            var oD1 = document.querySelector(".d1");

            var _cos = Math.cos(Math.PI/180*60); /*计算
cos(60度)*/
            var _sin = Math.sin(Math.PI/180*60); /*计算
sin(60度)*/

            /*顺时针转换60度*/
            oD1.style.transform =
"matrix("+_cos+", "+_sin+", "+-_sin+", "+_cos+", 0, 0)";
        }
    </script>
</body>

```

7. 倾斜(拉伸) (在矩阵中影响倾斜的值为b c -- matrix(1, tan(x), tan(y), 1, 0, 0))

```

<style>
    #box{
        width: 500px;
        height: 500px;
        border: 1px solid black;
        margin: 100px auto;
    }
    #box .d1{
        width: 100px;
        height: 100px;
        background: red;
        margin: 200px auto;
        background-image: linear-gradient(red,blue);
        transition: 1s;
    }
</style>

```

```

<body>
  <div id="box">
    <div class="d1"></div>
  </div>
  <script>
    document.onclick = function () {
      var oD1 = document.querySelector(".d1");

      var tanX = Math.tan(Math.PI/180*30);    /*计算
tan(x)*/
      var tanY = Math.tan(Math.PI/180*45);    /*计算
tan(y)*/

      /*绕着x轴倾斜30度，绕着y轴倾斜45度*/
      oD1.style.transform =
"matrix(1,"+tanX+", "+tanY+", 1,0,0)";
    }
  </script>
</body>

```

3D空间

1. transform-style可以创建一个3D空间

```
transform-style : preserve-3d
```

2. perspective 景深（看事物的距离），建议给要观看元素父元素添加这个属性
3. perspective-origin 景深基点（看事物的角度），建议给要观看元素父元素添加这个属性
4. translateZ() 沿Z轴移动距离
5. rotateZ() 沿Z轴旋转

转换基点

1. transform-origin: x y z 可以设置元素转换的基点
2. 可以设置基点在x y z三轴上的位置
3. x y 的值可以使用left right center等关键字以及百分比与px，z的值则使用px
4. x与y这两个有默认值为center
5. 示例

```
#box .d1{
```

```

width: 100px;
height: 100px;
background: red;
margin: 200px auto;
background-image: linear-gradient(red,blue);
transition: 1s;

transform-origin: bottom right;      /*设置转换的基点为右下角*/
}

/*沿着右下角旋转180度*/
#box:hover .d1{
    transform: rotate(180deg);
    /*transform-origin: bottom right; 不要把基点设置在这里，因为这样
    在旋转过程中基点也会发生变化*/
}

```

css动画

创建动画

1. @keyframes 创建动画
2. 创建动画有两种形式，可以使用百分比或者关键字（from to）创建动画

```

<style>
#box{
    width: 100px;
    height: 100px;
    background: red;
}
/*使用百分比创建动画，宽度从100px开始变到500px，然后再从500px变到
1000px*/
@keyframes play1 {
    0%{
        width: 100px;
    }
    50%{
        width: 500px;
    }
    100%{
        width: 1000px;
    }
}

```

```

/*使用关键字创建动画*/
@keyframes play2 {
    from{          /*相当于0%，表示宽从100px开始*/
        width: 100px;
    }
    to{            /*相当于100%，表示宽到1000px时结束*/
        width: 1000px;
    }
}
</style>

```

3. 注意：创建的动画并不会自己执行，需要把动画绑定到元素上调用

```

#box{
    width: 100px;
    height: 100px;
    background: red;
    animation: 8s play2;    /*使用名为play1的动画，执行时间为8秒*/
}

```

动画属性

1. animation 所有动画属性的简写（复合属性）
2. animation-name 指定要执行动画名称
3. animation-duration 规定动画完成一个周期所花费的秒或毫秒。默认是 0。
4. animation-timing-function 规定动画的速度曲线。默认是 "ease"。
5. animation-fill-mode 规定当动画不播放时（当动画完成时，或当动画有一个延迟未开始播放时），要应用到元素的样式。
 - forwards值：在动画结束后（由 animation-iteration-count 决定），动画将应用该属性值。
 - backwards值：动画将应用在 animation-delay 定义期间启动动画的第一次迭代的关键帧中定义的属性值。
6. animation-delay 规定动画开始时间（延迟）默认是 0。
7. animation-iteration-count 规定动画执行的次数
 - 默认是 1
 - infinite值：表示无限循环执行动画
8. animation-direction 规定动画是否在下一个周期逆向运动，
 - reverse 动画反向播放。
 - alternate 动画在奇数次（1、3、5...）正向播放，在偶数次（2、4、6...）反向播放。

- alternate-reverse 动画在奇数次（1、3、5...）反向播放，在偶数次（2、4、6...）正向播放。

9. animation-play-state 规定动画是否暂停或者运行，

- paused: 暂停动画
- running: 运行动画

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="">
  <style>
    #box{
      width: 100px;
      height: 100px;
      background: red;
      animation: 1s          2s          play1          cubic-
bezier(1,.02,.1,.98) 2          backwards
alternate;
/*          设置时长,    设置延迟,    设置动画名,    设置运动曲
线,          设置运动次数,    设置动画开始样式    设置动
画方向*/
    }
    #box:hover{
      animation-play-state:paused; /*设置动画是否暂停或者运
行*/
    }
    @keyframes play1 {
      0%{width: 0px}
      100%{width: 600px}
    }
  </style>
</head>
<body>
  <div id="box"></div>
</body>
</html>
```

