1、HTML 即超文本标记语言，是一种解释型文档，不做任何的编译处理
2、HTML5 是 HTML 的第五次产品更新，能够提高新元素的互操作性，解决了浏览器之间的不兼容性问题，目标就是将 HTML5 网页上的音视频、图像、动画等都带入一个国际标准化
3、HTML5 结构清晰，增加了很多主体元素，比如 article、section、nav、time 等；
增加了很多标签及功能，包括：
（1）**canvas 元素**：用来绘制矩形、圆形、文本、动画等
（2）**Web Storage 本地存储**：新增 sessionStorage 对象临时保存、localStorage 对象永久保存，可以实现简单的留言板功能
（3）**video 元素与 audio 元素**：专门用来播放音频和视频，不需要第三方插件，只要浏览器支持 HTML5 语法即可
（4）**HTML5 的拖放 API**：新增很多与拖放过程相关的事件，比如 dragstart、dragover、drop 等等，同时拖放过程中还可以利用 dataTransfer 对象传输数据

4、HTML5 新的网页结构

```
<!DOCTYPE html>      <!--文档声明格式，比 HTML4 更简化-->
<html>
<head lang='en'>
   <meta charset='UTF-8'>     <!--网页字符编码格式，比 HTML4 更简化-->
   <title>网页名称</title>
</head>
<body>
   <article>页面中一块和上下文不相关的独立内容，比如页面中调用一篇新闻、博客</article>
   <header>页面中一个 article 元素内容或者整个页面的标题</header>
   <section>页面中的一块内容区块，将文章内容分块，也可以和 hn 配合使用</section>
   <aside>表示 article 元素内容之外的，和内容相关的辅助信息</aside>
   <footer>页面中一个 article 元素内容或者整个页面的页脚，比如作者、联系方式等</footer>
   <nav>页面中的导航链接部分</nav>
</body>
</html>
```

5、HTML5 新增的主体结构元素

（1）article 元素：表示页面中一块和上下文不相关的独立内容，内容的标题和页脚分别放在<header>和<footer>元素中，demo 如下：

```
<body>
<article>
    <header>
        <h1>我是 article 标题</h1>
        <p>创建时间：<time pubdate='pubdate'>2018/3/7</time></p>
    </header>
    <p>
        <b>Article</b>是一个独立的区域，表示正文内容
    </p>
    <section><!--这个内容区块表示读者评论相关信息，将 article 正文与评论分开-->
        <h2>读者评论</h2>
```

```
            <article>
                <header>
                    <h3>读者：张三</h3>
                    <p><time pubdate datetime='2018/03/07T10:00'>2 小时前</time></p>
                </header>
                <p>文章很好！</p>
            </acticle>
            <article>
                <header>
                    <h3>读者：李四</h3>
                    <p><time pubdate datetime='2018/03/07T11:00'>2 小时前</time></p>
                </header>
                <p>文章非常好！</p>
            </acticle>
    </section>
    <footer>
        <p><small>麦子学院版权所有</small></p>
    </footer>
</article>
</body>
```

（2）section 元素：对网页内容进行分块，每一个 section 都由标题和内容组成，和 div 元素一样也是容器元素，但如果涉及 CSS 样式的设置，还是推荐使用 div 元素，不要用 section

section 元素自身嵌套使用的 demo：

```
<body>
<section>
    <h1>这是一个 section 元素</h1>
    <p>这里是一个内容区块</p>
    <section>
        <h2>A</h2>
        <p>A 的内容</p>
    </section>
    <section>
        <h2>B</h2>
        <p>B 的内容</p>
    </section>
</section>
</body>
```

section 元素和 article 元素互相嵌套使用的 demo：

```
<body>
<article>
    <h1>产品</h1>
    <p>产品详细列表</p>
    <section>
        <h2>产品 A</h2>
```

```
            <p>产品 A 的介绍</p>
        </section>
        <section>
            <h2>产品 B</h2>
            <p>产品 B 的介绍</p>
        </section>
</article>
</body>
```

注意：

1、如果要求文章内容是独立的，使用 article 元素；如果要求对内容分块，使用 section 元素

2、如果要对元素内容进行 CSS 样式设置，不要使用 section 元素，而是使用 div 容器元素

3、如果元素内容区块没有标题，则不要使用 section 元素

（3）aside 元素：使用方式有两种，一种是包含在 article 元素中作为主要内容的附属信息，比如当前文章的参考资料，名词解释等等；另一种是在 article 元素之外作为整个页面全局站点的附属信息

方式一的 demo：

```
<body>
    <header>
        <h1>国庆节去成都看熊猫</h1>
    </header>
    <article>
        <h2>看熊猫要去大熊猫基地</h2>
        <p>那里有很多的大熊猫</p>
        <aside>
        <h3>名词解释</h3>
        <dl>
            <dt>熊猫基地</dt>
            <dd>在四川卧龙</dd>
        </dl>
        </aside>
    </article>
</body>
```

方式二的 demo：

```
<body>
<aside>
        <h1>网站公告</h1>
        <p>国庆节放假通知</p>
</aside>
</body>
```

（4）nav 元素：表示页面导航的链接组，包含很多重要链接，可以链接到其他页面或者页面其他部分，一个页面可以嵌套很多 nav 标签，demo 如下：

```
<body>
    <h1>nav 的使用方法</h1>
    <nav>
```

```
            <ul>
                <li><a href='nav 标签'>首页</a></li>
                <li><a href='section 标签'>section 页面</a></li>
                <li><a href='article 标签'>article 页面</a></li>
            </ul>
        </nav>
        <article>
            <h2>nav 的嵌套使用</h2>
            <nav>
                <ul>
                    <li><a href='nav 标签'>首页</a></li>
                    <li><a href='section 标签'>section 页面</a></li>
                    <li><a href='article 标签'>article 页面</a></li>
                </ul>
            </nav>
        </article>
        <footer>
            <p>
                    <a href='section 标签'>section 页面</a>
                    <a href='article 标签'>article 页面</a>
            </p>
        </footer>
</body>
```

注意：

1、不是所有的导航链接<a>都要添加到 nav 标签中，比如页脚中的版权信息、站点信息、联系我们等链接放到 footer 元素中即可

2、nav 标签的使用场合有：传统的上边导航条、侧边栏导航、内页导航（比如单击百度百科的内部词条可以跳转到本页的词条解释）、翻页操作

（5）time 元素：表示某个日期或者 24 小时中的某一个时刻，允许带时差，定义的格式有：

```
<time datetime='2014-9-27'>2014 年 9 月 27</time>
<time datetime='2014-9-27'>9 月 27</time>
<time datetime='2014-9-27'>今天的时间</time>
<time datetime='2014-9-28T22:30'>2014 年 9 月 28 晚上 10 点</time>
<time datetime='2014-9-28T22:30Z'>UTC 标准时间 2014 年 9 月 28 晚上 10 点</time>
<time datetime='2014-9-2T22:30+8:00'>中国时间 2014 年 9 月 28 晚上 10 点</time>
```

（6）pubdate 元素：用在 time 元素里面，作为 time 元素的属性，表示当前文章或页面的发布时间，只有 time 中带有 pubdate 属性的时间才是文章的发布时间

```
<body>
<header>
<h1>你好 pubdate</h1>
<p>发布时间：<time datetime='2018-03-07' pubdate='pubdate'>2018 年 3 月 7 日</time></p>
</header>
<p>国庆节<time datetime='2017-10-01'>10 月 1 日</time>开始放假</p>
</body>
```

6、HTML5 新增的非主体结构元素

（1）header 元素：通常用来放置整个页面或一个内容区块的标题，也可以包含 logo 图片、搜索表单等，一个页面中可以出现多次 header 元素，一个 header 元素至少包含一个 heading 元素（h1-h6），另外还可以包含 hgroup 元素，nav 元素等，具体 demo 如下：

```
<body>
<header>
    <h1>网页标题</h1>
</header>
<article>
    <header>
        <h1>文章标题</h1>
    </header>
    <p>文章正文部分</p>
</article>
</body>
```

（2）hgroup 元素：将标题和他的子标题进行分组，一般会把 h1-h6 的元素进行分组，如果文章只有一个主标题，不用使用 hgroup 元素，demo 如下：

```
<body>
<!--只有一个主标题时不用 hgroup-->
<article>
  <header>
    <h1>文章标题</h1>
    <p><time datetime='2018-03-08'>2018 年 3 月 8 日</time></p>
  </header>
  <p>文章正文</p>
</article>
<!--需要对主标题和子标题分组时，要用到 hgroup-->
<article>
  <header>
    <hgroup>
      <h1>文章主标题</h1>
      <h2>文章子标题</h2>
    </hgroup>
    <p><time datetime='2018-03-08'>2018 年 3 月 8 日</time></p>
  </header>
  <p>文章正文</p>
</article>
</body>
```

（3）footer 元素：包含与它相关的内容区块的注脚信息，比如作者、版权信息等，一个页面中可以出现多次 footer 元素，demo 如下：

```
<body>
<footer>
  <p>
    <a href='/'>版权信息</a>
```

```
    <a href='/'>关于我们</a>
    <a href='/'>联系我们</a>
    <a href='/'>站点信息</a>
  </p>
  <p>麦子学院版权所有</p>
</footer>
</body>
```

（4）address 元素：用来呈现联系信息，包括文档的作者、邮箱、地址、电话信息等，还可以用来展示文章中相关联系人的所有信息，demo 如下：

```
<body>
<header>
    <h1>HTML5+CSS3 视频教程</h1>
</header>
<p>这里是文章正文</p>
<footer>
    <address>
        <a href='/'>张三</a>
        <a href='/'>大连理工大学管理学院</a>
        地址：大连市甘井子区
    </address>
    时间：<time datetime='2018-03-08'>2018 年 3 月 8 日</time>
</footer>
</body>
```

7、HTML5 新增的其他元素

（1）figure 元素与 figcaption 元素：

figure 元素表示页面中一块独立的内容，比如图片、音频、视频、统计表格等插件，带有可选标题，如果删除该元素也不会给页面造成影响；

figcaption 元素表示 figure 元素的标题，必须放在 figure 元素的内部，注意一个 figure 元素内最多只允许放置一个 figcaption 元素，具体 demo 如下：

```
<body>
<figure>
    <img src='1.jpg' title='风景 1'></img>
    <img src='2.jpg' title='风景 2'></img>
    <img src='3.jpg' title='风景 3'></img>
    <figcaption>风景</figcaption>
</figure>
</body>
```

（2）details 元素与 summary 元素：（细节与概括）

details 元素用于标识该元素内部的子元素可以被展开或收缩显示。该元素具有一个布尔类型的 open 属性（默认值为 false），值为 true 时表示该元素内部的子元素会被展开，值为 false 时表示该元素内部的子元素会被收缩。

summary 元素从属于 details 元素，用鼠标单击 summary 元素中的内容文字时，details 元素中的其他子元素会被展开或收缩，如果 details 中没有 summary 元素，浏览器会提供默认的文字以供点击，具体 demo 如下：

```
<body>
<details>
    <summary>HTML5+CSS3 视频教程</summary>
    <p>该视频教程由麦子学院提供！</p>
</details>
</body>
```

（3）mark 元素：表示页面中需要突出显示或高亮显示的内容，通常在引用原文时使用 mark 元素，目的是吸引读者的注意，具体 demo 如下：

```
<body>
<!--对页面中出现的所有 HTML5 进行高亮显示-->
<h2>以下是关于<mark>HTML5</mark>的搜索结果</h2>
<section>
    <article>
        <h2>
            <a href='/index.html'><mark>HTML5</mark>_百度百科</a>
        </h2>
        <p><mark>HTML5</mark>是万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言(HTML)的第五次重大修改</p>
    </article>
    <article>
        <h2>
            <a href='/index.html'><mark>HTML5</mark> 教程| 菜鸟教程</a>
        </h2>
        <p><mark>HTML5</mark> 简介 <mark>HTML5</mark>是 HTML 最新的修订版本,2014 年 10 月由万维网联盟(W3C)完成标准制定。</p>
    </article>
</section>
</body>
```

注意：mark 元素和 HTML4 中的 em 元素（表示强调，通常为斜体）、strong 元素（表示强调，通常为粗体）的作用是有区别的：一般 mark 元素高亮的内容与原文作者的意图无关，主要是目的是吸引读者的注意

（4）progress 元素：表示一个任务的完成进度，这个进度可以是不确定的。该元素具有两个属性，value 属性表示已经完成了多少工作量，max 属性表示总共有多少工作量，工作量的单位可以任意指定，比如任意数字或者百分比，只要 value 属性值小于等于 max 属性值，并且二者的值都大于 0 即可，具体 demo 如下：

```
<body>
<p>
    当前任务完成进度：
    <progress max='100' value='80'></progress>      <!--最终呈现效果是显示一个进度条-->
</p>
</body>
```

（5）meter 元素：表示规定范围内的数量值。有 6 个属性，value 属性表示元素的实际值，max 属性指定范围允许的最大值，min 属性指定范围允许的最小值，low 属性指定安全范围的下限值，high 属性指定安全范围的上限值，optimum 指定最佳值，必须在 min 和 max 之

间，demo 如下：

```
<body>
<p>
    硬盘实际使用情况：
    <meter value='40' max='120' min='0' low='50' high='70' optimum='60'>40/120</meter>GB
</p>
</body>
```

注意：当 optimum 值大于 high 值，可能会出现红色警告

8、HTML5 废除的其他元素

（1）能使用 CSS 设置样式代替的元素：big　center　font　s　strike　tt　u

（2）不支持 frame、frameset、noframes 元素，只支持 iframe 框架

（3）其他废除元素可以见转载的博客

9、HTML5 的大纲：

（1）HTML5 大纲分析工具网址：https://gsnedders.html5.org/outliner/（会呈现最终页面效果）

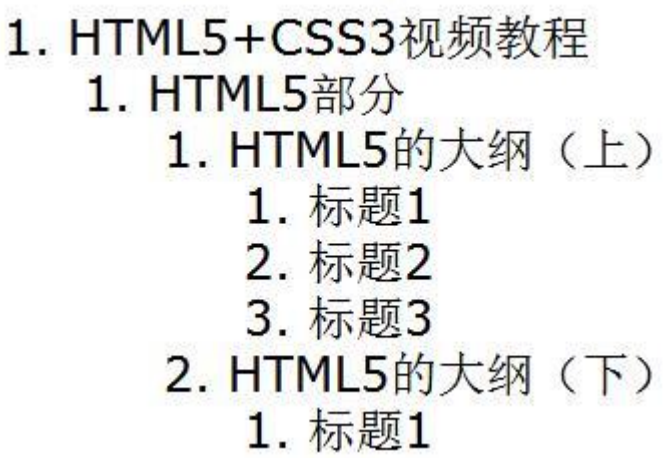
（2）HTML5 大纲的编排规则：分为‘显示编排’和‘隐式编排’两种方式。

‘显示编排’是指明确使用 section 元素进行分块，每个内容区块内使用标题元素 h1-h6；

‘隐式编排’是指不使用 section 元素进行分块，而是根据标题元素 h1-h6 的级别进行自动创建区块，标题分级的规则是：如果新出现的标题比上一个标题级别低，那么将生成下级内容区块；如果级别高或者级别相同，那么将生成新的内容区块！

（3）“显示编排”demo 如下，将代码复制到上面链接页面中，即可看到右边的效果图：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>题目</title>
</head>
<body>
<h1>HTML5 大纲显示编排</h1>
<section>
    <h1>HTML5 部分</h1>
    <section>
        <h2>HTML5 的大纲（上）</h2>
        <section>
            <h3>标题 1</h3>
            <p>内容 1</p>
        </section>
        <section>
            <h3>标题 2</h3>
            <p>内容 2</p>
        </section>
        <section>
            <h3>标题 3</h3>
            <p>内容 3</p>
        </section>
```



```
    </section>
    <section>
        <h2>HTML5 的大纲（下）</h2>
        <section>
            <h3>标题 1</h3>
            <p>内容 1</p>
        </section>
    </section>
</section>
</body>
</html>
```

注意：单独的 header 元素、footer 元素不可以做大纲，因为大纲是根据标题元素 h1-h6 生成的，但可以在 header 元素中包含标题子元素，这样标题子元素中包含的内容或图片就可以生成大纲了

（4）“隐式编排”demo 如下，将代码复制到上面链接页面中，即可看到右边的效果图：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>题目</title>
</head>
<body>
<h1>HTML5 大纲隐式编排</h1>
<p>隐式编排的内容</p>
<h2>子标题 1</h2>
<p>子标题 1 的内容</p>
<h3>子子标题 1</h3>
<p>子子标题 1 的内容</p>
<h2>子标题 2</h2>
<p>子标题 2 的内容</p>
<h1>HTML5 大纲隐式编排的例子</h1>
<p>隐式编排的内容</p>
</body>
</html>
```

1. HTML5大纲隐式编排
    1. 子标题1
        1. 子子标题1
    2. 子标题2
2. HTML5大纲隐式编排的例子

10、加强版的 ol 元素及 dl 元素

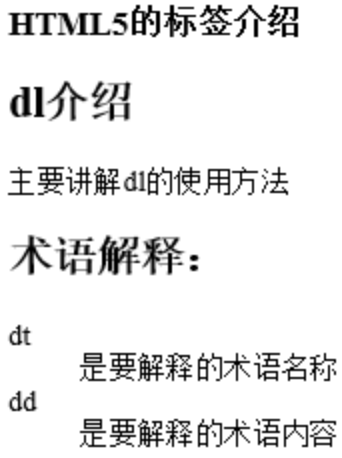
（1）ol 元素已有 type 属性表示标号类型，HTML5 新增两个属性 start 和 reversed，start 属性定义标号的开始值，reversed 属性进行反向编号，具体 demo 如下：

```
<!--列表标号结合 type 属性值从第二个开始，分别是 bcde-->
<ol start='2' type='a'>
    <li><a href="index.html">有序列表 1</a></li>
    <li><a href="index.html">有序列表 2</a></li>
    <li><a href="index.html">有序列表 3</a></li>
    <li><a href="index.html">有序列表 4</a></li>
```

```
</ol>
<!--列表标号结合 type 属性值进行反向编号，分别是 4321-->
<ol    type='1' reversed="reversed">
      <li><a href="index.html">有序列表 1</a></li>
      <li><a href="index.html">有序列表 2</a></li>
      <li><a href="index.html">有序列表 3</a></li>
      <li><a href="index.html">有序列表 4</a></li>
</ol>
```

（2）dl 元素是一个专门用来定义术语的列表元素，每个 dl 元素都包含一个或多个 dt 元素，表示术语名称，dt 元素后面紧跟一个或多个 dd 元素，表示术语解释内容，具体 demo 如下：

```
<body>
<h3>HTML5 的标签介绍</h3>
<article>
      <h1>dl 介绍</h1>
      <p>主要讲解 dl 的使用方法</p>
      <aside>
            <h2>术语解释：</h2>
            <dl>
                  <dt>dt</dt>
                  <dd>是要解释的术语名称</dd>
                  <dt>dd</dt>
                  <dd>是要解释的术语内容</dd>
            </dl>
      </aside>
</article>
</body>
```



11、利用 canvas 元素创建画布

（1）创建画布：指定 canvas 元素的三个属性 id、width（宽度）、height（高度）

（2）引入 JS 脚本：利用 script 标签引入一个外部 JS 脚本文件，该文件中定义一个 draw 函数，包括绘制各种图形的逻辑实现过程

（3）使用 draw 函数进行绘画：在 body 元素中加入 onload="draw('canvas')"属性，调用脚本文件中的 draw 函数，canvas 的基本语法如下：

```
<!DOCTYPE html>
<html>
<head>
      <meta charset='utf-8'>
      <title>canvas 元素</title>
      <script type="text/javascript" src='canvas.js'></script>
</head>
<body onload="draw('canvas')">
<canvas id='canvas' width='500' height="350"></canvas>
</body>
</html>
```

12、利用 canvas 元素及 JS 脚本绘制矩形（下面步骤要放在一个 JS 文件中）

（1）获取 canvas 元素：利用 document.getElementById()方法
（2）获取图形上下文对象（封装了很多绘图功能）：利用 canvas 对象的 getContext()方法，并传入参数‘2d’
（3）设置绘制样式：利用上下文对象的 fillStyle 属性设置填充颜色值，strokeStyle 属性设置边框颜色值
（4）设置颜色值：有多种方式①颜色名（black）②16 进制表示法（#000000）③rgb（rgba(0,0,0)）④rgba（rgba(0,0,0,1)）
（5）设置边框宽度：利用上下文对象的 lineWidth 属性，可以设置任何直线的宽度
（6）绘制最终矩形：使用上下文对象的 fillRect(x,y,width,height)来填充矩形，strokeRect(x,y,width,height)来绘制矩形边框，参数中的 x,y 分别表示矩形的起点横坐标和纵坐标，坐标原点是画布的左上角，width、height 分别表示矩形的宽度和高度，demo 及呈现效果如下：

Index.html 文件中的代码：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>canvas 元素绘制矩形</title>
    <script type="text/javascript" src='canvas.js'></script>
    <style type="text/css">
        body{
            margin: 0;
            padding: 0;
        }
    </style>
</head>
<body onload="draw('canvas')">
<canvas id='canvas' width='400' height="300"></canvas>
</body>
</html>
```

canvas.js 文件中的代码：

```
function draw(id){
    var canvas=document.getElementById(id);
    var context=canvas.getContext('2d');
    context.fillStyle='#000';//填充颜色为黑色
    context.strokeStyle='#f60';//边框颜色为橙色
    context.lineWidth=5;
    context.fillRect(0,0,400,300);//绘制填充矩形
    context.strokeRect(50,50,180,120);//绘制矩形边框
}
```

12、利用 canvas 元素及 JS 脚本绘制圆形

（1）表明开始绘制新路径：利用上下文对象的 beginPath()方法
（2）绘制圆形路径：利用上下文对象的 arc()方法，
语法是：arc(x,y,radius,startAngle,endAngle,anticlockwise),其中（x,y）是圆的圆心，radius 是圆

的半径，startAngle 是起始角度，endAngle 是结束角度，anticlockwise 表示是否按照顺时针方向绘制，如果要表示 360 度，可以使用 Math.PI*2

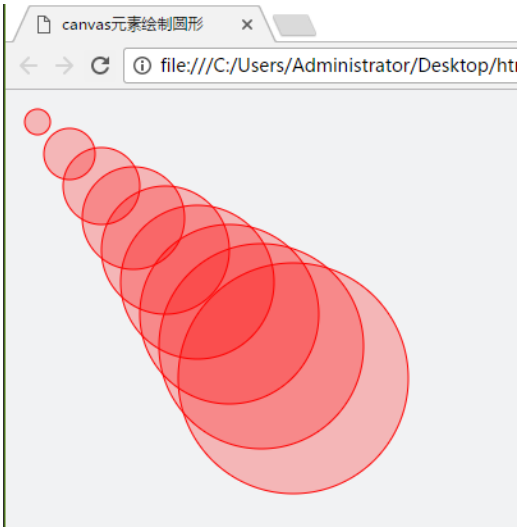
（3）创建完成闭合路径：利用 closePath()方法

（4）设置绘制的样式：调用 fillStyle 属性及 fill()方法进行填充，或者调用 strokeStyle 及 stroke()方法绘制边框，demo 及呈现效果如下：

Index.html 文件中的代码同上

canvas.js 文件中的代码如下：

```
function draw(id){
    var canvas=document.getElementById(id);
    var context=canvas.getContext('2d');
    context.fillStyle='#f1f2f3';//填充颜色为黑色
    context.fillRect(0,0,400,400);//绘制填充矩形
    for(var i=1;i<10;i++){
        context.beginPath();//每一次循环都重新开始绘制
        context.arc(i*25,i*25,i*10,0,Math.PI*2,true);
        context.closePath();//每一次循环都要闭合路径
        context.fillStyle='rgba(255,0,0,0.25)';
        context.fill();//设置填充样式
        context.strokeStyle='red';
        context.stroke();//设置边框样式
    }
}
```



注意：绘制的路径如果不设置样式，在画布上是看不到的，所以最后一步必须调用 fill 或 stroke 方法进行填充或描边才能在画布上显示出来

13、利用 canvas 元素及 JS 脚本绘制文字

（1）设置文本的三个属性：

a、文本样式 font=”font-weight font-size font-family”（字体加粗 字体大小 字体类型）

b、文本水平对齐方式 textAlign，属性值有：start（默认值）、end、left、right、center

c、文本垂直对齐方式 textBaseline，属性值有：top（顶部对齐，但会留点距离）、middle（中间对齐）、bottom（底部对齐）、hanging（悬挂，紧贴顶部）、alphabetic（默认值、字母写法）

（2）绘制文字使用上下文对象的 fillText(text,x,y,[maxwidth])或 strokeText(text,x,y,[maxwidth])方法进行填充或描边，其中参数 text 表示要绘制的文字，x,y 表示文字的起始坐标，maxwidth 是可选参数，表示显示文字的最大像素宽度，防止溢出，demo 及呈现效果如下：

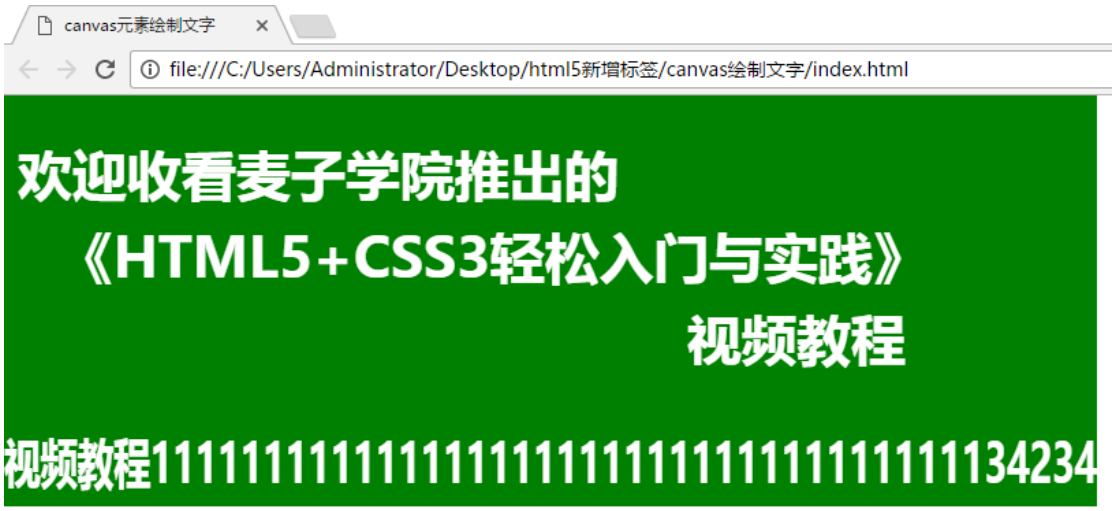
Index.html 文件中的代码同上

canvas.js 文件中的代码如下

```
function draw(id){
    var canvas=document.getElementById(id);
    var context=canvas.getContext('2d');
    context.fillStyle='green';//填充颜色为绿色
    context.fillRect(0,0,800,300);//绘制填充矩形，矩形的长宽最好和画布大小一致
    context.fillStyle='#fff';//填充颜色为白色
    //在绘制文本前先设置字体的属性 font、textBaseline、textAlign
    context.font="bold 40px '微软雅黑','宋体'";//设置多个字体时要用逗号隔开，浏览器会从头匹配合适的字体
```

```
    context.textBaseline='hanging';//紧贴画布的最上面
    context.fillText('欢迎收看麦子学院推出的',10,40);
    context.fillText('《HTML5+CSS3 轻松入门与实践》',40,100);
    context.fillText('视频教程',500,160);
    //最后一个参数会防止绘制的文字溢出，将文字压缩在最大像素宽度之内
    context.fillText('视频教程 1111111111111111111111111111111111111134234',0,250,800);

}
```



14、canvas 保存文件

如果绘制完成的图片需要保存，可以使用 Canvas API 中的 toDataURL()方法，将绘画状态输出到一个 data URL 中重新装载，然后重新保存。

data URL 实际上就是 base64 位编码的 URL，主要用于小型的可以在页面中直接嵌入的数据，如 img 元素里面的图像文件。

toDataURL()的使用方法：canvas 对象.toDataURL(type)，其中参数 type 表示要输出数据的 MIME 类型，比如 image/jpeg　　image/png

canvas.js 文件中的代码如下：

```
function draw(id){
    var canvas=document.getElementById(id);
    var context=canvas.getContext('2d');
    context.fillStyle='green';//填充颜色为绿色
    context.fillRect(0,0,400,300);//绘制填充矩形，矩形的长宽最好和画布大小一致
    //将图像导出到一个 data URL 中，并改变浏览器的位置
    window.location=canvas.toDataURL('image/jpeg');
}
```

15、利用 canvas 元素绘制动画

在 canvas 画布中制作动画实际上就是不断变化坐标、擦除、重绘的过程

（1）设置动画的间隔时间：使用 setInterval（执行动画的函数，间隔时间）全局方法

（2）设置绘图函数的方式一：通过不断变换 x，y 的坐标来实现

Index1.html 文件中的代码如下：（Index2.html 文件中把'canvas1.js'换成'canvas2.js'即可）

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset='utf-8'>
    <title>canvas 绘制动画</title>
    <script type="text/javascript" src='canvas1.js'></script>
</head>
<body onload="draw('canvas')">
<canvas id='canvas' width='400' height="400"></canvas>
</body>
</html>
```

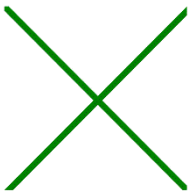
canvas1.js 文件中的代码如下：

//实现的动画效果：相同样式的矩形每隔 100 秒不断出现（可以模仿贪吃蛇，进度条的移动）

```
function draw(id){
    var canvas=document.getElementById(id);
    context=canvas.getContext('2d');
    setInterval(painting,100);//每隔 100 毫秒执行一次 painting 函数，自带循环功能
    i=0;
}
//动画的实现以画布大小为准，所以 setInterval 函数不用调用 clearInterval 就可以停止
function painting(){
    context.fillStyle='green';//填充颜色为绿色
    context.fillRect(i,0,10,10);//绘制起点的横坐标每次都是变化的
    i=i+20;//横坐标每次都加 20，矩形之间有间隔，一直到画布的边缘停止循环
    //i++;横坐标每次只加 1，矩形连起来在水平方向上慢慢移动
}
```

canvas2.js 文件中的代码如下：

//实现的动画效果：从四个角开始以渐变的方式画一个叉号



```
function draw(id){
    var canvas=document.getElementById(id);
    context=canvas.getContext('2d');
    setInterval(painting,10);//每隔 10 毫秒执行一次 painting 函数，自带循环功能
    i=0;
}
//动画的实现以画布大小为准，所以 setInterval 函数不用调用 clearInterval 就可以停止
function painting(){
    context.fillStyle='green';//填充颜色为绿色
    context.fillRect(i,i,10,10);//横纵坐标从对角线的左上角方向开始移动
    context.fillRect(400-i,400-i,10,10);//横纵坐标从对角线的右下角方向开始移动
    context.fillRect(i,400-i,10,10);//横纵坐标从对角线的左下角方向开始移动
    context.fillRect(400-i,i,10,10);//横纵坐标从对角线的右上角方向开始移动
    i++;//横纵坐标每次只加 1，矩形连起来在对角线的方向上移动
}
```

（3）设置绘图函数的方式二：用 clearRect(x,y,width,height)方法将画布整体或者局部擦除，其中 x,y 是擦除的起点坐标，width、height 是擦除的宽度和高度

canvas3.js 文件中的代码如下：

```
//实现的动画效果：每次只有一个矩形不断在水平方向上移动
var context;
var width,height;
var i;
function draw(id){
    var canvas=document.getElementById(id);
    context=canvas.getContext('2d');
    width=canvas.width;
    height=canvas.height;
    setInterval(painting,100);//每隔 10 毫秒执行一次 painting 函数，自带循环功能
    i=0;
}
//动画的实现以画布大小为准，所以 setInterval 函数不用调用 clearInterval 就可以停止
function painting(){
    context.fillStyle='#fff';//擦除颜色为白色
    context.clearRect(0,0,width,height);//每次循环都擦除整个画布的图形，重新绘制新的矩形
    context.fillStyle='green';//填充颜色为绿色
    context.fillRect(i,20,10,10);//新绘制的矩形在水平方向上不断移动
    i=i+20;
}
```

16、Web Storage 本地存储

在 HTML4 中使用 cookie 在客户端保存用户名等一些简单的用户信息，但 cookie 的缺点是：难以操作，长度只能限制在 4KB，每个 cookie 都会被添加到 http 请求头部中发送给服务器，容易造成宽带浪费等，为了解决这些问题，在 HTML5 中提供了在本地客户端存储数据的功能 Web Storage，其中包括 **sessionStorage 和 localStorage** 两种对象的定义

（1）**sessionStorage 临时保存**：就是将数据保存在 session 对象中，session 对象用来存储特定用户会话所需的属性及配置信息。在浏览器关闭后，数据就会被删除

保存数据的方法：sessionStorage.setItem(‘key’,’value’) 或者 sessionStorage.key=’ value’

读取数据的方法：变量名=sessionStorage.getItem(‘key’) 或者 变量名=sessionStorage.key

index.html 文件中的代码如下：

```
 <!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Web Storage 本地存储</title>
    <script type="text/javascript" src='sessionStorage.js'></script>
</head>
<body >
<h1>Web Storage 本地存储</h1>
<p id='msg'></p>
```

```
<input type="text" id="input">
<input type="button" value="保存数据" onclick="saveData('input')">
<input type="button" value="读取数据" onclick="readData('msg')">
</body>
</html>
```

sessionStorage.js 文件中的代码如下：

```
//把文本框中输入的数据保存到 session 对象中
function saveData(id){
    var target=document.getElementById(id);
    var str=target.value;
    sessionStorage.setItem('msg',str);
}
//读取 session 对象中的数据并显示到 p 元素中
function readData(id){
    var p=document.getElementById(id);
    var message=sessionStorage.getItem('msg');
    p.innerHTML=message;
}
```

（2）**localStorage 永久保存**：就是将数据保存在客户端本地的硬件设备上，可以跨越会话持久化保存，即使关闭浏览器，数据也不会丢失，下次打开浏览器数据仍然可以读取到，除非删除 js 代码或者清除浏览器缓存

保存数据的方法：localStorage.setItem(‘key’,’value’) 或者 localStorage.key=’ value’

读取数据的方法：变量名=localStorage.getItem(‘key’) 或者 变量名=localStorage.key

Index.html 文件中的代码把'sessionStorage.js'换成'localStorage.js'即可

localStorage.js 文件中的代码如下：

```
//把文本框中输入的数据永久保存到本地磁盘中
function saveData(id){
    var target=document.getElementById(id);
    var str=target.value;
    localStorage.setItem('msg',str);
}
//读取本地磁盘中的数据并显示到 p 元素中
function readData(id){
    var p=document.getElementById(id);
    var message=localStorage.getItem('msg');
    p.innerHTML=message;
}
```

17、利用 Web Storage 制作简单的网页留言板

需要自定义三个函数，分别实现不同的功能：

（1）saveData()：将留言板中输入的内容保存到 localStorage 对象中。首先利用 new Date().getTime()获取留言时间表示的毫秒数，然后使用 localStorage.setItem(‘key’,’value’)将留言板中的数据作为键值，将毫秒数作为键名保存下来

（2）readData()：取得保存后的所有数据，然后以表格的形式进行显示。利用 localStorage 的两个重要属性：localStorage.length 表示保存在 localStorage 对象中的所有键值对的数目，

localStorage.key(index)取得 index 索引号对应的键名

（3）clearData()：将保存在 localStorage 对象中的数据全部删除。利用 localStorage.clear()方法，具体实现 demo 如下：

index.html 文件中的代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>简单的网页留言板</title>
    <script type="text/javascript" src='liuyanban.js'></script>
</head>
<body >
<h1>简单的网页留言板</h1>
<textarea id='textarea' cols="60" rows="10"></textarea>
<br/>
<input type="button" value="保存" onclick="saveData('textarea')">
<input type="button" value="读取" onclick="readData('msg')">
<input type="button" value="删除" onclick="clearData('msg')">
<hr><!--加一条水平分割线-->
<p id='msg'></p>
</body>
</html>
```

liuyanban.js 文件中的代码如下：

```
//把留言板区域输入的数据保存到本地磁盘上
function saveData(id){
    //获取留言内容数据
    var target=document.getElementById(id);
    var str=target.value;
    //获取当前日期的毫秒数,将留言时间保存下来
    var time=new Date().getTime();
    //保存时的键名是毫秒数，键值是留言内容
    localStorage.setItem(time,str);
    alert('数据已经被保存');
    readData('msg');
}
//读取本地磁盘中的数据并以表格的形式显示到 p 元素的位置
function readData(id){
    //创建一个空表格
    var table="<table border='1'>";
    //读取磁盘中的所有数据，并将其添加到表格中显示出来
    for(var i=0;i<localStorage.length;i++){
        //获取留言时间表示的毫秒数
        var key=localStorage.key(i);
        //获取留言内容
```

```
            var value=localStorage.getItem(key);
            //重新设置当前时间为留言时间
            var date=new Date();
            date.setTime(key);
            //将留言时间设置成 GMT 格林威治时间形式的字符串
            var dateStr=date.toGMTString();
            //将序号、留言内容、留言时间添加到表格中
            table += '<tr><td>' + '这是第' + i + '条数据' + '</td><td>'
            +    value + '</td><td>' + dateStr +    '</td></tr>';
        }
        table += '</table>';
        var p=document.getElementById(id);
        p.innerHTML=table;
}
//将磁盘中的数据全部删除
function clearData(id){
        localStorage.clear();
        alert('数据已经被成功删除');
        readData('msg');
}
```

18、video 元素与 audio 元素

HTML4 中如果要在网页上播放一段音频或视频，必须使用第三方插件比如 flash，HTML4 代码量太大，容易给服务器增加负担，造成宽带浪费，为了解决这些问题，HTML5 中增加了 video 视频元素和 audio 音频元素功能

（1）基础用法：

①video 元素专门用来播放网络上的视频，audio 元素专门用来播放网络上的音频；使用 video 和 audio 元素进行播放时不需要使用其他的插件，只要浏览器支持 HTML5 即可；

②audio 元素必须指定 src 属性和 controls 属性（必需的），比如：<audio src="MP3.mp3" controls="controls"></audio>，对于不支持 HTML5 的浏览器可以在 audio 元素之间加入提示语句即可

③video 元素除了指定 src 属性和 controls 属性（可选的），还要指定 width 和 height，比如：<video src="time.mp4" width="700" height="400"></video>，对于不支持 HTML5 的浏览器可以在 video 元素之间加入提示语句即可

④source 元素作为 audio 和 video 元素的子元素，可以为同一个媒体数据指定多种播放格式与编码方式，以确保浏览器可以从中选择一种自己支持的播放格式进行播放，选择顺序自上而下，直到选到支持的格式为止（音频后缀名有 mp3，视频后缀名有 mp4/m4v/ogv/webm）比如：

```
<video>
        <source src="video.m4v" type="video/mp4">
        <source src="video.mp4" type="video/mp4">
        <source src="video.webm" type="video/webm">
        <source src="video.ogv" type="video/ogg">
</video>
```

支持 audio 元素和 video 元素的 demo：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>video 元素与 audio 元素</title>
</head>
<body >
<h1>audio 元素的使用示例</h1>
<audio src="MP3.mp3" controls="controls"></audio>
<br/>
<h1>video 元素的使用示例</h1>
<video src="video.mp4" width="600" height="280"></video>
</body>
</html>
```

（2）video 和 audio 常用的属性：

1）src 属性：指定媒体数据的 URL 地址（播放源）

2）controls 属性：指定是否为媒体数据添加浏览器自带的播放控制条

3）width 和 height 属性（video 元素独有）：指定视频的宽度和高度

4）autoplay 属性：指定当页面加载完成后是否自动开始播放

5）preload 属性：指定是否对数据进行预加载，如果是的话，浏览器会对媒体数据进行缓冲，加快播放速度，包括三个属性值：none 表示不进行预加载，metadata 表示只预加载媒体的元数据，auto（默认值）表示预加载全部的视频或音频，对比演示：

```
<audio src="MP3.mp3" controls="controls"></audio>
<br/>
<audio src="MP3.mp3" controls="controls" preload="none"></audio>
```

6）poster 属性（video 元素独有）：当视频一开始没有播放或者无法播放时，使用 poster 元素指定一张图片代替视频，使用方法：<video src="video.mp4" poster="img.jpg"></video>

7）loop 属性：指定是否循环播放音视频文件

8）error 属性：读取过程发生错误时，返回一个 Media Error 对象，该对象的 code 属性返回一个数字，表示对应的错误状态，默认值为 null，数字及对应的错误状态分别为：

1=MEDIA_ERR_ABORTED（读取过程中被用户中止）

2=MEDIA_ERR_NETWORK（下载时发生错误）

3=MEDIA_ERR_DECODE（解码时发生错误）

4=MEDIA_ERR_SRC_NOT_SUPPORTED（媒体不可用或者不支持音视频）

具体 demo 如下：（video1.mp4 文件不存在，所以可能出现错误 4）

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>video 元素与 audio 元素</title>
</head>
<body >
```

```
<h1>video 和 audio 元素的属性 error</h1>
<video src="video1.mp4" id='video'></video>
<script type="text/javascript">
    var video=document.getElementById('video');
    video.addEventListener('error',function(){
        var error=video.error;//返回一个错误对象
        switch(error.code){   //根据错误代码执行不同的错误状态
            case 1:alert('读取过程中被用户中止');
                break;
            case 2:alert('下载时发生错误');
                break;
            case 2:alert('解码时发生错误');
                break;
            case 2:alert('媒体不可用或者不支持音视频');
                break;
        }
    },false);
</script>
</body>
</html>
```

19、HTML5 的拖放 API

（1）要使元素成为可拖动元素，首先要设置元素的 **draggable 属性为 true**，但默认情况下，图像 img 元素、链接 a 元素都是可拖动的，即 draggable 属性默认为 true

（2）与拖放有关的事件类型有：

dragstart（拖放操作开始时触发）：产生事件的元素是被拖放的元素

drag（拖放过程中持续触发）：产生事件的元素是被拖放的元素

dragenter（被拖放的元素开始进入放置目标元素时触发）：产生事件的元素是目标元素

dragover（被拖放的元素在目标元素范围内移动时持续触发）：产生事件的元素是目标元素

dragleave（被拖放的元素离开放置目标元素时触发）：产生事件的元素是目标元素

drop（被拖放的元素放置到目标元素时触发）：产生事件的元素是目标元素

dragend（整个拖放操作结束时触发）：产生事件的元素是被拖放的元素

（3）拖放过程中的数据传输方法：利用 **event.dataTransfer 对象**

该对象的属性有：（注意：dropEffect 必须搭配 effectAllowed 才能使用）

1）dropEffect：表示实际拖放时的视觉效果，一般在 ondragover 事件处理程序中设定，属性值有：none（不允许放置）、copy（复制到目标元素）、move（移动到目标元素）、link（目标元素打开被拖动的元素），但必须在 effectAllowed 属性允许的视觉效果范围内设置属性值

2）effectAllowed：指定元素被拖放时所允许的视觉效果，一般在 ondragstart 事件处理程序中设定，属性值有：copy（只允许值为“copy”的 dropEffect）、move、link、copyMove、copyLink、linkMove、all（允许所有拖动操作）、none（不允许执行任何拖动操作）、uninitialized（不指定属性值，将执行浏览器中默认允许的拖动操作）

3）types：表示存入数据的类型

该对象的方法有：

1）setData（数据格式，数据值）：向 dataTransfer 对象中存入数据

2）getData（数据格式）：从 dataTransfer 对象中读取数据

3）clearData（数据格式）：清除 dataTransfer 对象中保存的指定格式的数据，如果不指定格式，则清除全部数据

4）setDragImage（图像元素，光标在图像中的 x 坐标，y 坐标）：用 img 元素设置拖放图标

具体 demo 如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>HTML5 的拖放功能</title>
    <style type="text/css">
        #dropelement{
            width:500px;
            height: 300px;
            border:solid 2px #d2d2d3;
            box-shadow: 1px 4px 8px #646464;
        }
        #dragelement{
            width: 500px;
            height: 300px;
            background: #e54d26;
        }
    </style>
</head>
<body >
<h1>拖动图片到指定位置</h1>
<div id='dropelement' ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br/>
<img src="HTML5.png" id='dragelement' width="400" height="280" draggable='true' ondragstart="drag(event)">
<script type="text/javascript">
    function drag(event){
        //设置元素被拖放时所允许的视觉效果
        event.dataTransfer.effectAllowed='all';
        //将被拖动元素的 id 存入 dataTransfer 对象
        event.dataTransfer.setData('Text',event.target.id);
    }
    function allowDrop(event){
        //所有元素默认是不允许放置其他元素的，所以要阻止默认行为，使其成为可放置元素
        event.preventDefault();
        //设置实际拖放时的视觉效果，拖放鼠标会带个"+"
        event.dataTransfer.dropEffect='copy';
    }
```

```
    function drop(event){
        //所有元素默认是不允许放置其他元素的，所以要阻止默认行为，使其成为可放置元素
        event.preventDefault();
        //从 dataTransfer 对象中获取保存的被拖动元素的 id
        var data=event.dataTransfer.getData('Text');
        //将被拖动元素作为子节点添加到放置目标元素中
        event.target.appendChild(document.getElementById(data));
    }
</script>
</body>
</html>
```

效果图为：