儒

释

道

·

随缘

之前端基础

目录

第一部分 HTML

一、HTML 的语法

1、什么是 HTML 标记语言

描述网页信息的超文本标记语言（HyperText Markup Language）。

特点： 1.可以设置文本的格式，比如标题、字号、文本颜色、段落等等

2.可以创建列表

3.可以插入图像和媒体

4.可以建立表格

5.超链接，可以使用鼠标点击超链接来实现页面之间的跳转

2、HTML 的标记和他的属性

1.HTML 文档的保存格式：.html、.htm、.xhtml；

2.标记和被标记的内容构建出 HTML 文档

格式： 这是一个段落。

<标记>内容</标记>

3.标记的属性

标记的属性，就是用来控制我们的内容（图像、文本等的）如何的显示。

格式：<标记 属性 1=属性值 属性 2=属性值 ... ... >内容</标记>

例如：<body bgcolor="red">内容</body>

以上的属性是用来控制我们的网页的背景颜色，bgcolor 就是 body 的属性，他的值是 red（红色）

3、语法不区分字母大小写

<HTML>、<Html>、<html>都是定义相同的标记，但是在编写网页的时候尽量使用小写

4、文档注释

注释一段内容时使用“<!--”开始，以"-->"结束

例如<!--我是 sidney-->

5、代码格式

空格键和回车键在网页中都不会起到任何作用，我们为了让代码清晰易读，可以使用空格和回车键进行编排。

注意：缩进时保持严格的规则，以“Tab”键进行缩进！

6、字符实体

1.什么是字符实体？

比如我们想在网页上面显示一个“<”小于符号，但是“<”在 HTML 中是文档标记的开始语言，如果我们直接使用“<”会出差错，所以我们就会一些实体名称来代替！

2.常见的字符实体

显示结果	描述	实体名称	实体编号
	空格		
<	小于号	&lt;	&#60;
>	大于号	&gt;	&#60;
&	和号	&amp;	&#38;
"	引号	&quot;	&#34;
'	引号	' (IE 不支持)	&#39;
¢	分	&cent;	&#162;

£	镑	£	£
¥	日圆	¥	¥
€	欧元	€	€
§	小节	§	§
©	版权	©	©
®	注册商标	®	®
™	商标	™	™
×	乘号	×	×
÷	除号	÷	÷

二、HTML 的基本结构

1、<!DOCTYPE> 声明

```
<!DOCTYPE html>
<html>
  <head>
      <title>文档标题</title>
  </head>
  <body>
      可见文本...
  </body>
</html>
```

HTML5

```
<!DOCTYPE html>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2、<html>内容</html>

解释：HTML 文档的文档标记，也称为 HTML 开始标记。

功能：这对标记分别位于网页的最前端和最后端。

<html>在最前端表示网页的开始，</html>在最后端表示网页的结束。

3、<head>内容</head>

解释：HTML 文件头标记，也称为 HTML 头信息开始标记。

功能：用来包含文件的基本信息，比如网页的标题、关键字。

在<head></head>内可以放<title></title>、<meta></meta>、<style></style>等等标记。

注意：在<head></head>标记内的内容不会在浏览器中显示。

4、<title>内容</title>

解释：HTML 文件标题标记。

功能：网页的“主题”，显示在浏览器的窗口的左上边。

注意：网页的标题不能太长，要短小精悍，能具体反应页面的内容。

<title></title>标记中不能包含其他标记

5、<body>内容</body>

解释：HTML 文档的主体标记。

功能：<body>...</body>是网页的主体部分，在此标记之间可以包含如 、<h1></h1>、
、<hr>等等标记，正是由这些内容组成了我们所看见的网页。

body 标记的常见属性：

bgcolor：设置背景颜色 ，<body bgcolor="red"></body>

text：设置文本颜色，<body text="green"></body>

link：设置链接颜色，<body link="blue"></body>

vlink：已经访问了的链接颜色，<body vlink="yellow"></body>

alink：正在被点击的链接颜色，<body alink="red"></body>

6、<meta>内容</meta>

解释：页面的元信息（meta-information）。

功能：提供有关页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词。

必须的属性：content

值：some_text

定义与 http-equiv 或 name 属性相关的元信息

常见的属性： name 属性

1.author：规定文档的作者的名字。实例： <meta name="author" content="Hege Refsnes">

2.keywords：规定一个逗号分隔的关键词列表 - 相关的网页（告诉搜索引擎页面是与什么相关的）。

提示：总是规定关键词（对于搜索引擎进行页面分类是必要的）。

实例： <meta name="keywords" content="HTML, meta tag, tag reference">

3.description：规定页面的描述。搜索引擎会把这个描述显示在搜索结果中。

实例： <meta name="description" content="Free web tutorials">

4.application-name：规定页面所代表的 Web 应用程序的名称。

注意：meta 标记必须放在 head 元素里面

三、文档设置标记

1、格式标记

1.
：强制换行标记，让后面的文字、图片、表格等等，显示在下一行。

2.<p>：换段落标记，换段落，由于多个空格和回车在 HTML 中会被等效为一个空格，所以 HTML 中要换段落就要用<p>,<p>段落中也可以包含<p>段落！

3. ：居中对齐标记（html5 不支持），让段落或者是文字相对于父标记居中显示。

4.<pre>：预格式化标记，保留预先编排好的格式。

5.<li>：列表项目标记，可用在有序列表（<ol>）、无序列表（<ul>）和菜单列表（<menu>）。

6.<ul>：无序列表标记，声明这个列表没有序号

7.<ol>：有序列表标记，可以显示特定的一些顺序。

1)格式：规定列表的类型。（不赞成使用，使用样式代替）

<ol type=""符号类型"">

<li type=""符号类型"">

<li type=""符号类型"">

2)有序列表的 type 属性值

1：阿拉伯数字 1.2.3 等等，默认 type 属性值

A：大写字母 A、B、C 等等

a：小写字母 a、b、c 等等

Ⅰ：大写罗马数字Ⅰ、Ⅱ、Ⅲ、Ⅳ等等

ⅰ：小写罗马数字ⅰ、ⅱ、ⅲ、ⅳ等等

3)value：指定一个新的序列数字起始值

4)列表可以进行嵌套

8.<dl><dt><dd>：定义型列表。

使用场合：对列表条目进行简短的说明。

格式：

<dl>

<dt>软件说明：</dt>

<dd>简单介绍软件的功能及基本应用</dd>

<dt>软件界面</dt>

<dd>用于选择软件的外观</dd>

</dl>：水平分割线标记，段落之间的分割线。

10. ：分区显示标记，也称之为层标记，常用来编排一大段的 HTML 段落，也可以用于格式化表，和<p>很相似，可以多层嵌套使用

2、<!DOCTYPE> 声明

1.hn：标题标记，共有 6 个级别，n 的范围 1~6，不同级别对应显示大小不同的标题，h1 最大，h6 最小。

2.font：字体设置标记，设置字体的格式。

三个常用属性：

1)size（字体大小）：<font size=""14px"">

2)color（颜色）：<font color=""red"">

3)face（字体）：<font face=""微软雅黑"">

3.b：粗字体标记

4.i：斜字体标记

5. u：下划线字体标记

6.sub：文字下标字体标记

7.sup：文字上标字体标记

8.tt：打印机字体标记

9.cite：引用方式的字体，通常是斜体

10.em：表示强调，通常显示为斜体字

11.strong：表示强调，通常显示为粗体字

12.small：小型字体标记

13.big：大型字体标记

四、图像标记

1、<img>图像标记

1.使用方法：<img src="路径/文件名.图片格式" width="属性值" height="属性值" border="属性值" alt="属性值">

2.<img>标记的属性：

1) src 属性：指定我们要加载的图片的路径和图片的名称以及图片格式。

2)width 属性：指定图片的宽度，单位 px、em、cm、mm。

3)height 属性：指定图片的高度，单位 px、em、cm、mm。

4)border 属性：指定图标的边框宽度，单位 px、em、cm、mm。

5)alt 属性：当网页上的图片被加载完成后，鼠标移动到上面去，会显示这个图片指定的属性文字。

如果图像没有下载或者加载失败，会用文字来代替图像显示。

搜索引擎可以通过这个属性的文字来抓取图片。

6)usemap：将图像定义为客户端图像映射（图像映射指的是带有可点击区域的图像）。

<img usemap=""#mapname"">

一个 hash 字符 (""#"") 加上要使用的 <map> 元素的 name 或 id。

3.注意：

1)<img>为单标记，不需要使用</img>闭合

2)在加载图像文件的时候，文件的路径或者文件名文件格式错误，将无法加载图片

2、<map><area>图像映射

1.<map>标签：标签用于客户端图像映射

<map name=""planetmap""> ：name 属性必需。为 image-map 规定的名称。

2.<area>标签

1)alt 属性-text 值：规定区域的替代文本。如果使用 href 属性，则该属性是必需的。

2)coords 属性-coordinates：规定区域的坐标。

3)href 属性：规定区域的目标 URL。

4)shape 属性：规定区域的形状。

default	规定全部区域。
rect	定义矩形区域。
circ	定义圆形。
poly	定义多边形区域。

5)target 属性：规定在何处打开目标 URL。

_blank	在新窗口中打开被链接文档。
_self	默认。在相同的框架中打开被链接文档。
_parent	在父框架集中打开被链接文档。
_top	在整个窗口中打开被链接文档。
framename	在指定的框架中打开被链接文档。

3.图像映射例子

```
<!DOCTYPE html>
<html>
<body>
  <p>点击太阳或其他行星，注意变化：</p>
  <img src=""planets.gif"" width=""145"" height=""126"" alt=""Planets"" usemap=""#planetmap"">
  <map name=""planetmap"">
      <area shape=""rect"" coords=""0,0,82,126"" alt=""Sun"" href=""sun.htm"">
      <area shape=""circle"" coords=""90,58,3"" alt=""Mercury"" href=""mercur.htm"">
      <area shape=""circle"" coords=""124,58,8"" alt=""Venus"" href=""venus.htm"">
  </map>
</body>
</html>
```

五、链接

<a>超链接

基本语法：链接文字或者图片

基本属性：

1.href 属性：链接的地址，链接的地址可以是一个网页，也可以是一个视频、图片、音乐等等。

2.target 属性：定义超链接的打开方式。

_blank	在一个新的窗口中打开链接。
_self	默认。在当前窗口中打开链接
_parent	在父窗口中打开页面（框架中使用较多）

_top 在顶层窗口中打开文件（框架中使用较多）

framename 在指定的框架中打开被链接文档。

3.name 属性：指定页面的锚点名称。

六、表格

1、<table>标记

1.基本格式 <table 属性 1=""属性值 1"" 属性 2=""属性值 2"" ... ... >表格内容</table>

2.table 标记的属性

1)width 属性：表示表格的宽度，他的值可以是像素（px）也可以是父级元素的百分百（%）。

2)height 属性：表示表格的高度，他的值可以是像素（px）也可以是父级元素的百分百（%）。

3)border 属性：表示表格外边框的宽度

4)align 属性：表格的显示位置

值：

1.left 居左显示（默认值）

2.center 居中显示

3.right 居右显示

5)cellspacing 属性：单元格之间的间距，默认是 2px，单位像素

6)cellpadding 属性：单元格内容与单元格边框的显示距离，单位像素

7)frame 属性：控制表格边框最外层的四条线框

属性值：

1.void(默认值)：表示无边框

2.above：表示仅顶部有边框

3.below：表示仅有底部边框

4.hsides：表示仅有顶部边框和底部边框

5.lhs：表示仅有左侧边框

6.rhs：表示仅有右侧边框

7.vsides：表示仅有左右侧边框

8.box：包含全部 4 个边框

9.border：包含全部 4 个边框

8)rules 属性：控制是否显示以及如何显示单元格之间的分割线

属性值：

1.none(默认值)：表示无分割线

2.all：表示包括所有分割线

3.rows：表示仅有行分割线

. 4.clos：表示仅有列分割线

5.groups：表示仅在行组和列组之间有分割线

2、<caption>标记

1.什么时候使用

1)什么时候使用如果表格需要使用标题，那么就可以使用<caption>标记

`2.如何正确使用：<caption>属性的插入位置，直接位于<table>属性之后，<tr>表格行之前

align 属性

1)top：标题放在表格的上部

2)bottom：标题放在表格的下部

3.)left.：标题放在表格的左部

4)right 标题放在表格的右部

3、<tr>标记

定义表格的一行，对于每一个表格行，都是由一对<tr>...</tr>标记表示，每一行<tr>标记内可以嵌套多个<td>或者<th>标记。

可选属性

1.bgcolor 属性：设置背景颜色

格式：bgcolor=""颜色值""

2.align 属性：设置垂直方向对齐方式

格式：align=""值""

值：

bottom：靠顶端对齐

top：靠底部对齐

middle：居中对齐

3.valign 属性：设置水平方向对齐方式

格式:：valign=""值""

值：

left：靠左对齐

right：靠右对齐

center：居中对齐

4.<td>和<th>

1)<td>和<th>都是单元格的标记，其必须嵌套在<tr>标签内，是成对出现

2)两者的区别:

<th>是表头标记，通常位于首行或者首列，<th>中的文字默认会被加粗，而<td>不会

<td>是数据标记，表示该单元格的具体数据

3)共同之处:

两者的标记属性都是一样的

4)属性

1)bgcolor：设置单元格背景

2)align：设置单元格对齐方式

3)valign：设置单元格垂直对齐方式

4)width：设置单元格宽度

5)height：设置单元格高度

6)rowspan：设置单元格所占行数

7)colspan：设置单元格所占列数

七、HTML 框架

1、什么是框架？

框架将浏览器划分成不同的部分，每一部分加载不同的网页，实现在同一浏览器窗口中加载多个页面的效果

2、<frameset>划分框架标记

1.语法格式：<frameset>....</frameset>

2.属性：

1)cols：

使用“像素数”和%分割左右窗口，“*”表示剩余部分

如果使用“*”，“*”表示框架平均分成 2 个

如果使用“*”，“*”，“*”表示框架平均分成 3 个

2)rows：使用“像素数”和%分割上下窗口，“*”表示剩余部分

3)frameborder：指定是否显示边框，0 不显示，1 显示

4)border：设置边框的大小，默认值 5 像素

3.<frame>子窗口标记

<frame>标记是一个单标记，该标记必须放在<frameset>中使用，在<frameset>中设置了几个窗口，就必须对应使用几个<frame>框架，而且还必须使用 src 属性指定一个网页

属性：

1)src：加载网页文件的 URL 地址

2)name：框架名称，是链接标记的 target 所要参数

3)noresize：表示不能调整框架大小，没有设置时就可以调整

4)scrolling：是否需要滚动条

值：

auto：根据需要自动出现

yes：有

no：无

5)frameborder：是否需要边框

值：

1（1）：显示边框

2（0）：不显示边框

八、表单设计

1、表单标记

<form>...</form>：定义表单的开始位置和结束位置，表单提交时的内容就是<form>表单中的内容

基本格式：

<form action=""服务器端地址（接受表单内容的地址）"" name=""表单名称"" method=""post|get"">...</form>

常用属性：

1.name：表单名称

2.method：传送数据的方式，分为 post 和 get 两种方式

get 方式：get 方式提交时，会将表单的内容附加在 URL 地址的后面，所以限制了提交的内容的长度，不超过 8192 个字符，且不具备保密性。

post 方式：post 方式提交时，将表单中的数据一并包含在表单主体中，一起传送到服务器中处理，没有数据大小限制

3.action：表单数据的处理程序的 URL 地址，如果为空则使用当前文档的 URL 地址，如果表单中不需要使用 action 属性也要指定其属性为”no “

4.enctype：设置表单的资料的编码方式

5.target：和超链接的属于类似，用来指定目标窗口

2、文本域和密码

1.<input>标记：<input>标记没有结束标记

基本语法： <input type="""" name="""" value="""" size="""" maxlength="""">

属性介绍：

1)type 属性：type 属性有两个值

text：当 type=""text""时，<input>表示一个文本输入域

password：当 type=""password""时，<input>表示一个密码输入域

2)name 属性：定义控件的名称

3)value 属性：初始化值，打开浏览器时，文本框中的内容

4)size 属性：设置控件的长度

5)maxlength 属性：输入框中最大允许输入的字符数

3、提交、重置、普通按钮

1.提交按钮：当<input type=""submit"">时，为提交按钮

2.重置按钮：当<input type=""reset"">时，为重置按钮

3.普通按钮：当<input type=""button"">时，为普通按钮

4、单选框和复选框

单选按钮：当<input type=""radio"">时，为单选按钮

复选框：当<input type=""checkbox"">时，为复选框

注意：单选框和复选框都可以使用”cheked“属性来设置默认选中项

5、隐藏域

当<input type=""hidden"">时，为隐藏表单域

6、多行文本域

用法：使用<textarea>标记可以实现一个，能够输入多行文本的区域

语法格式：<textarea name=""name"" rows=""value"" cols=""value"" value=""value""> … … <textarea>

rows 属性和 cols 属性分别用来指定，显示的行数和列数，单位是字符个数

7、菜单下拉列表域

<select>标记

1.语法格式：

```
<select name="""" size=""value"" multiple>
    <option value=""value"" selected>选项 1</option>
    <option value=""value"">选项 2</option>
    <option value=""value"">选项 3</option>
    … … …
</select>
```

2.属性：

multiple 属性：multiple 属性不用赋值，其作用是，表示用可以多选的下来列表，如果没有这个属性就只能单选

size 属性：设置列表的高度

name 属性：定义这个列表的名称

3.option 标记：<option>标记用来指定列表中的一个选项，需要放在<select></select>之间

值：

value：给选项赋值，指定传送到服务器上面的值

selected：指定默认的选项

第二部分 HTML5

一、什么是 HTML5

HTML 即超文本标记语言（HyperText Markup language）,这是一种语法简单、结构清晰的语解释型文档,他不同于其他的编程语言。

HTML5 就是 HTML 网页标记语言的第五次重大更新产品，在这个版本中，新功能不断推出，以帮助 Web 应用程序的作者，

努力提高新元素互操作性。

二、HTML5 的优点

用更简洁的 HTML 代码创建更多功能的网页程序！并且 HTML5 让网页结构变得更加的清楚明了，增加了更加语义化的结构标签，这样一个网页的结构就非常清晰，那个部位显示的什么内容，让人一目了然！

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

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

兼容性：HTML5 在老版本的浏览器上面也可以完美运行。

实用性：HTML5 抛弃了不切实际的功能，一切按照实用性的线路出发！

三、最基本的 HTML5 网页骨架

1、新的网页结构

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
  <header>...</header>
  <nav>...</nav>
  <article>...</article>
  <section>...</section>
  <aside>...</aside>
  <footer>...</footer>
</body>
</html>
```

1、DOCTYPE 声明

HTML4 中的 DOCTYPE 声明格式

```
<!DOCTYPE html PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
```

HTML5 中的 DOCTYPE 声明格式

```
<!DOCTYPE html>
```

2、网页字符编码

HTML4 中是指定<meta>元素的形式来制定网页的字符编码，如下：

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

而在 HTML5 中是对<meta>元素直接进行追加 sharset 属性来指定字符，如下：

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

3、<header>元素

header 元素表示页面中的一个内容区块或者整个页面的标题！

HTML5 中使用方法：

```
<header></header>
```

4、<nav>元素

nav 元素表示页面中的导航链接部分。

HTML5 中使用方法：

<nav>...</nav>

5、<article>元素

article 元素表示页面中的一块玉上下文不相关的的独立内容，比如一篇文章中的文章。

HTML5 的使用方法：

<article></article>

6、<section>元素

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

<section></section>

7、<aside>元素

aside 元素表示 article 元素的内容之外的，和内容相关的辅助信息！

<aside><aside>

8、<footer>元素

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

<footer></footer>

2、新增的主体结构元素

1、article 元素

2、section 元素

3、aside 元素

4、nav 元素

使用场合：

传统的导航条

侧边栏导航

内页导航

翻页操作

5、time 元素

Time 元素代表 24 小时中的某个时间或者是日期，表示时间时允许带时差。

定义的格式如下：

<time datetime=""2014-9-28"">2014 年 9 月 28<time>

<time datetime=""2014-9-28"">9 月 28<time>

<time datetime=""2014-9-28"">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=""2014-9-28T22:30+8:00"">中国时间 2014 年 9 月 28 晚上 10 点<time>

6、pubdate 属性

3、新增的非主体结构元素

1、header 元素

header 元素是一种具有引导和导航作用的结构元素，通常用来放置整个页面或页面内的一个内容区块的标题，但也可以包含其他的内容，比如在 header 里面放置 logo 图片、搜索表单等等。

注意：一个页面内并没有限制 header 的出现次数，也就是说我们可以在同一页面内，不同的内容区块上分别加上一个 header 元素。

在 HTML5 中，一个 header 元素至少可以包含一个 heading 元素（h1-h6），也可以包括我们下节课将要学习的 hgroup 元素，还可以包括其他的元素，在新的 W3C HTML5 标准，我们还可以把 NAV 元素包括进去。

2、hgroup 元素

hgroup 元素是将标题和他的子标题进行分组的元素。hgroup 元素一般会把 h1-h6 的元素进行分组，比如在一个内容区块的标题和他的子标题算是一组。

通常情况下，文章只有一个主标题时，是不需要 hgroup 元素的。

3、footer 元素

footer 元素可以作为他的上层父级内容区块或是一个根区块的注脚。footer 元素一般情况下包括与它相关的区块的注脚信息，比如作者、版权信息等。

注意：footer 元素和我们前面所讲的 header 元素一样，并没有限制他的个数。并且 footer 元素可以为 article 元素或者 section 元素添加 footer 元素。

4、address 元素

address 元素用来在页面中呈现联系信息，包括文档的作者、邮箱、地址、电话信息等。

address 元素还用来展示和文章中的相关的联系人的所有联系信息。

4、新增和废除元素的认识

1、其他新增元素

1.新增的 figure 元素与 figcaption 元素

figure 元素是一种元素的组合，带有可选标题。figure 元素用来表示页面上一块独立的内容，如果将他从网页上删除不给我们的网页造成影响。

figcaption 元素表示 figure 元素的标题，它属于 figure 元素，figcaption 元素必须书写在 figure 元素内部，可以写在 figure 元素内的其他从属元素前面或后面。一个 figure 元素内最多只允许放置一个 figcaption 元素。

2.新增的 details 元素与 summary 元素

details 元素是一种用于标识该元素内部的子元素可以被展开、收缩显示的元素。details 元素具有一个布尔类型的 open 属性，当该值为 true 时，该元素内部的子元素应该被展开显示，当该属性值为 false 时，其内部的子元素应该被收缩起来不现实。该属性的默认值为 false，当页面打开时其内部的子元素应该处于收缩状态。

summary 元素从属于 details 元素，用鼠标点击 summary 元素中的内容文字时，details 元素中的其他所有从属元素将会展开或者收缩。如果 details 元素内没有 summary 元素，浏览器那你会提供默认的文字以供点击，例如“details”

目前只有谷歌的 Chrome 浏览器支持！

3.新增的 mark 元素

mark 元素表示页面中需要突出显示或高亮显示的，对于当前用户具有参考作用的一段文字。通常在引用原文时使用 mark 元素，目的是引起读者的注意。mark 元素是对原文内容有补充作用的一个元素，他应该用在一段原文作者不认为重要的，但是现在为了与原文作者不相关的其他目的而需要突出显示或者高亮显示的文字上面。

mark 元素最主要的目的就是吸引当前用户的注意。

mark 元素除了高亮显示之外，还有一个作用就是在引用原文时，为了某种特殊的目的而把作者没有表示出来的内容重点表示出来。

4.新增的 progress 元素

progress 元素代表一个任务的完成进度，这个进度可以使不确定的，表示进度 z 正在进行，但不清楚这个还有多少工作量没有完成，也可以用 0 到某个最大数字（比如 100）之间的 s 数字来表示准确的进度情况（比如百分比）

该元素具有两个表示当前任务完成情路昂的属性，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。工作量的单位是随意的，不指定的。

在设定属性点时候，value 属性和 max 属性只能指定为有效的浮点数，value 属性必须大于 0，且小于或等于 max 的属性，max 的属性必须大于 0。

5.新增的 meter 元素

meter 元素表示规定范围内的数量值。

meter 元素有 6 个属性！

1)value 属性：在元素中特地地表示出来的实际值。该属性值默认为 0，可以为该属性制定一个浮点小数值。

2)min 属性：指定规定范围时允许实用的最小值，默认 0，在设定该属性时所设定的值不能小于 0。

3)max 属性：指定规定的范围时允许使用的最大值，如果设定时该属性值小于 min 的值，那么把 min 属性的值视为最大值。max 属性的默认值 1。

4)low 属性：规定范围的下限值，必须小于或者等于 high 的值。

5)high 属性：规定范围的上限值。如果该属性值小于 low 属性的值，那么把 low 属性的值视为 high 属性的值，同样如果该属性的值大于 max 属性的值，那么把 max 属性的值视为 high 的值。

6)optimum 属性：最佳值属性值必须在 min 属性值与 max 属性值之间，可以大于 high 属性值。

2、废除元素

1.能使用 CSS 代替的元素：

basefont、big、center、font、s、strike、tt、u

2.不在使用 frame 框架

对于 frameset 元素、frame 元素与 noframes 元素，由于 frame 框架对页面可用性存在负面影响，在 html5 里面已经不支持 frame 框架，只支持 iframe 框架，同时废除以上这三个元素。

3.只有部分浏览器支持的元素

对于 applet 元素、bgsound、blink、marquee 元素，由于只有部分浏览器支持这些元素，特别是 bgsound 元素以及 marquee 元素，只被 IE 浏览器支持，所以在 HTML5 里面被废除！而 applet 元素可以由 embed 元素或者 object 元素代替，bgsound 元素由 audio 元素代替，marquee 可以使用 javascript 来代替！

4.其他被废除的元素

A、废除 rb 元素，使用 ruby 元素代替

B、废除 acronym 元素，使用 abbr 元素代替

C、废除 dir 元素，使用 ul 元素代替

D、废除 inindex 元素，使用 form 元素与 input 元素相结合的方式代替

E、废除 listing 元素，使用 pre 元素代替

F、废除 xmp 元素，使用 code 元素代替

G、废除 nextid 元素，使用 GUIDS 代替

H、废除 plaintext 元素，使用“text/plian” MIME 类型代替

3、Html5 大纲

1.在 HTML5 中，可以使用之前我们所学习的结构元素来编排一份网页大纲，那样我们就可以通过这个网页大纲来了解网页中具有那些内容，网页中以什么样的形式来组织这些内容有更清楚的认识。

2.分析一个网页的大纲！

找到出现 Untitled Section 的原因并解决！

3.header 元素可以做大纲吗？

4.在 header 元素中用图片来生成大纲的方法！

5.关于 nav 元素和 aside 元素。

大纲的编排规则

关于 HTML 的大纲编排，我们可以划分为“显式编排”和“隐式编排”两种方式。

显式编排：显式编排是指明确使用 section 元素进行分块来创建网页文档，每个内容区块内使用标题（h1~h6）

隐式编排：所谓的隐式编排，就是我们不使用 section 元素进行明确的区块划分，而是根据我们写的（h1-h6、hgroup 等）元素进行自动创建区块，因为 HTML5 的分析器可以根据不同级别的标题判断出对应的内容区块。

标题分级：不同的标题有不同的级别，在（h1-h6）中 h1 的级别最高，而 h6 的级别最低。所以在隐式编排的情况下就会按照以下规则生成！

1)如果出现新的标题比上一个标题级别低，那么将生成下级内容区块！

2)如果新出现的标题比上面出现的一个标题级别高，或者两者级别相同，那么就生成新的内容区块！

不同的内容区块可以使用相同级别的标题，父内容区块与子内容区块都可以使用相同级别的 H1 标题。

5、加强版的 ol 与 dl

1、ol 的 start 属性和 reversed 属性

可以通过 start 属性来定义标号的开始值。

可以通过 reversed 属性来进行反向编号。

2、重新定义含义的 dl 元素

dl 元素在 HTML4 中：

dl 元素在 HTML4 中，是一个专门用来定义术语的列表。

dl 元素在 HTML5 中：

dl 元素在 HTML5 中，把 dl 元素进行了重定义。每一项包含一条或者多条带名字的 dt 元素，用来表示术语，dt 元素后面紧跟一个或者多个 dd 元素，用来表示定义。重定义后的 dl 列表包含多个带名字的列表项。

6、初识 CANVAS

1、简单了解 canvas

什么是 canvsa？ canvas 是在 html5 中新增的一个重要元素，专门用来绘制图形。

①创建一个 canvas 画布

画布的创建方法：指定 id、width（画布宽度）、height（画布高度）。

例如：<canvas id=""canvas"" width=""500"" height=""350""></canvas>

上面这个实例的意思就是，创建一个画布，长度为 500，高度为 350.

②引入绘画脚本

引入脚本的方法：

比如：<script type=""text/javascript"" src=""canvas.js"" charset=""utf-8""></script>

上面这个实例的意思就是：引入一个名为 canvas 的 JS 脚本，JS 脚本的语言编码是 utf-8

③使用 draw 函数进行绘画

在 body 的属性里面，使用 onload=""draw('canvas');""语句。调用脚本文件中的 draw 函数进行图形绘画。

2、绘制一个矩形

①canvas 元素

用 getElementById 方法获取到 canvas 对象。

②取得上下文

在绘制图形的时候要用到图形上下文，图形上下文是一个封装了很多绘图功能的对象。要使用 canvas 对象的 getContext() 方法获得图形上下文。在 draw 函数中把参数设置为“2d”。

③填充与会绘制边框

canvas 绘制有两种方法：

填充（fill）：填充是将图形内部填满。

绘制边框（stroke）：绘制边框是不把图形内部填满，只是绘制图形的外框。

④设置绘制样式

当我们在绘制图形的时候，首先要设定好绘制的样式，然后我们就可以调用有关的方法进行绘制。

fillStyle 属性：填充的样式，在这个属性里面设置填入的填充颜色值。

strokeStyle 属性：图形边框的样式，在这个属性里面设置填入边框的填充颜色。

⑤指定画笔宽度

通过对上下文对象（context）的 lineWidth 属性来设置图形边框的宽度，任何直线的宽度都是可以通过 lineWidth 来设置直线的宽度的。

⑥设置颜色值

绘制图形的时候要填充的颜色或者边框分别可以通过 fillStyle 属性和 strokeStyle 属性来指定。

颜色的值可以使用

16 进制的颜色值（#000000）

颜色名（black）

rgb（rgba(0,0,0)）

rgba（rgba(0,0,0,0.1)）

⑦绘制矩形

使用 fillRect 方法和 strokeRect 方法来填充矩形和绘制矩形的边框。

context.fillRect（x,y,width,height）

context.strokeRect（x,y,width,height）

这两种方法的参数都是一样的，X 是指矩形的起点横坐标，y 是指矩形的纵坐标，坐标的原点是 canvas 画布的最左上角，width 是指矩形的长度，height 是指矩形的高度。

3、绘制一个圆形

①开始创建路径

使用图形上下文对象的 beginPath 方法。

context.beginPath（）;

②创建图形路径

创建圆形路径时，需要使用图形上下文对象的 arc 方法。

context.arc（x,y,radius,starAngle,endAngle,anticlockwise）

x 是绘制圆形的起点横坐标，y 是绘制圆形起点的纵坐标，radius 是圆形的半径，starAngle 是开始的角度，endAngle 是结束的角度，anticlockwise 是否按顺时针方向绘制。

绘制半径与圆弧时指定参数为开始弧度与结束弧度，如果你喜欢使用角度，可以使用以下这个方法，把角度换成弧度。

var radius = degrees*Math.PI/180

这个里面的 Math.PI 表示的角度是 180 度，Math.PI*2 的角度是 360 度。

③创建完成关闭路径

使用图形上下文对象的 closePath 方法将路径关闭。

context.closepath();

④设置绘制样式然后调用绘制方法进行绘制

context.fillStyle = 'rgba(255,0,0,0.25)';

context.fill();

4、绘制文字

绘制字体时可以使用 fillText 方法或者 strokeText 方法。

fillText 方法用填充的方式来绘制字符串；

context.fillText（text，x,y,[maxwidth]）;

strokeText 方法用轮廓的方式来绘制字符串；

context.strokeText（text，x，y,[maxwidth]）;

第一个参数 text 表示要绘制的文字，第二个参数 X 表示要绘制的文字的起点横坐标，第三个参数 Y 表示要绘制的文字的起点纵坐标，第 4 个参数 maxwidth 为可选参数，表示显示文字的时候最大的宽度，可以防止文字溢出。

①设置文字字体

context.font = ""font-weight font-size font-family "";

context.font 有三个参数：

第一个参数 font-weight

第二个参数 font-size 规定文本的字体尺寸

第三个参数 font-family 规定文本的字体系列

font-family 可以的值是'ariSDal','arial','宋体','微软雅黑'...等等任何字体。

②设置文字垂直对齐方式

context.textBaseline = 'alphabetic';

属性值可以是 top（顶部对齐）、hanging（悬挂）、middle（中间对齐）、、bottom（底部对齐）、alphabetic 是默认值。

③设置文字水平对齐方式

conText.textAlign = ""start""

属性值可以设置为：start、end、left、right、center。

5、保存文件

很多时候绘制完成的图片需要保存，那么我们就可以使用到 Canvas API 来完成这最后一步！

Canvas API 使用 toDataURL 方法把绘画的状态输出到一个 data URL 中然后重新装载，然后我们就可以把重新装载后的文件直接保存。

Canvas API 保存文件的原理实际上就是把我们绘画的状态动态输出到一个 data URL 地址所指向的数据中的过程。

什么是 data URL？

data URL 实际上就是 base64 位编码的 URL,主要用于小型的，可以在网页中直接嵌入，而不需要从外部嵌入数据，比如 img 元素里面的图像文件。

data URL 的格式 “...”

toDataURL 的使用方法

canvas.toDataURL（type）;

这个方法使用一个参数 type，表述输出数据的 MIME 类型。

什么是 MIME 类型：

jpg image/jpeg

6、canvas 小示例

在 canvas 画布中制作动画相对来说很简单，实际上就是不断变化坐标、擦除、重绘、的过程。

①使用 setInterval 方法设置动画的间隔时间。

setInterval(code,millisec)

setInterval 方法 html 中固有方法，这个方法接受两个参数，第一个函数表示执行动画的函数，第二个参数为间隔时间，单位是（毫秒）。

②用来绘图的函数

通过不断的变换 X 和 Y 的坐标来实现动画效果。

在该函数中先用 clearRect 方法将画布整体或者是局部擦除。

擦除图像 clearRect 方法：

context.fillRect(x,y,width,height);

x 是指我们起点的横坐标，y 是指我们起点的纵坐标，width 是指擦子的长度，height 是指擦子的高度。

7、本地储存

1、Web Storsge

在 HTML5 中，除了 CANVAS 元素，另外一个新增的非常重要功能就是可以在本地客户端储存数据的 Web Storage 功能。

在 HTML4 我们是使用的 cookies 在客户端保存用户名等等一些简单的用户信息。

cookies 的缺点：

1）、大小限制在 4KB；

2）、带宽浪费；

3）、难以操作；

为了解决这样的一些问题，在 HTML5 中重新提供了在客户端保存数据的功能，他就是我们的“Web Storage”。

这个小章节我们来简单了解，sessionStorage 和 localStorage 之间的区别，掌握两者的基本用法。

1.sessionStorage 临时保存

就是把数据保存在 session 对象之中。

session 就是在打开网站到关闭网站之间要求进行保存的数据。

sessionStorage

临时保存的用法：

sessionStorage.setitme('key','value');

//或者是 sessionStorage.key = 'value';

临时数据读取的方法：

变量=sessionStorage.getItem（'key'）

//或者是 sessionStorage.key;

2.localStorage 永久保存

就是将数据保存在客户端本地的硬件设备至上面，如果浏览器被关闭，这个数据不会丢失，下次打开可以继续使用。这个功能就是我们的 localStorage 永久保存功能。

localStorage

永久保存数据的方法：

localStorage.setItem（'key','value'）;

//或者

localstorage.key;

读取的方法：

变量=localStorage.getItem（'key'）;

//或者

变量=localStorage.key;

2、实战简单的访客留言板

制作留言板需要使用到的函数有 3 个

1.saveStorage 函数

使用“new Date().getTime()”语句来获取当前的日期和时间戳，然后使用 localStorage.setItem 将获取到的时间戳作为键值，并将文本框中的数据作为键名进行保存。保存后使用 loadStorage 函数在页面上显示保存后的数据。

2.loadStorage 函数

这个函数取得保存后的所有数据，然后以表格的形式进行显示。

两个重要的属性：

1）、loadStorage.length

所有保存在 localStorage 中的数据条数。

2）、localStorage.key（index）

想要得到的数据的索引号作为 index 参数传入，可以得到得到 localStorage 中与这个索引号对应的数据。

3）、clearStorage 函数

将保存在 localStorage 中的数据全部清除。

用法：localStorage.clear();

8、影音多媒体

video 元素与 audio 元素

1、video 元素与 audio 元素的基础知识

video 元素：在 HTML5 中专门用来播放网络上的视频或者电影。

audio 元素：在 HTML5 中专门用来播放网络上的音频。

使用 video 和 audio 元素进行播放时就不在需要使用其他的插件了，只要我们的浏览器支持 HTML5 就可以了！

浏览器的支持：

Safari3 以上、Firefox4 以上、0pera10 以上、chrome3.0 以上版本都对 audio 元素和 video 元素支持！

使用方法：

audio 元素只需要给他指定一个 src 属性：

```
<audio src=""MP3.mp3"" controls=""controls""></audio>
```

对于不支持的浏览器我们可以在这对元素之间加入提示语句来代替

```
<audio src=""MP3.mp3"" controls=""controls"">您的浏览器不支持 Audio 元素</audio>
```

video 元素要设定好长宽和 src 属性就可以了：

```
<video width=""750"" height=""400"" src=""time.mp4""></video>
```

同样对于不支持 video 的浏览器可以在中间加入替换文字：

```
<video width=""750"" height=""400"" src=""time.mp4"">您的浏览器不支持 video 元素</video>
```

source 元素指定多个播放格式与编码：

source 元素可以为同一个媒体数据指定多个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，直到选择到所支持的格式为止。

使用方法：

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

各种设备对编码格式的支持情况：

webm(.webm)格式的视频 火狐 4.0+、chrome6.0+、opera10.6+

mp4(.m4v)格式的视频 IE9.0+ 、Safari3.1+ 、iso5.0 、Android4.0+

ogg(.ogv)格式的视频 火狐 3.5+、chrome3.0+、opera10.5+

mp4(.mp4)格式的视频 IE9.0+ 、Safari3.1+ 、iso3.0 、Android2.3+

viedo 与 audio 的常用属性

src 属性：在这个属性里面指定媒体数据的 URL 地址。

controls 属性：指定是否为视频或者音频数据添加浏览器自带的播放控制条，控制条中有播放按钮、暂停等按钮。

使用方法：

```
<video src=""video.mp4"" controls=""controls""></video>
```

width 和 height 属性（video 独有）：指定视频的宽度与高度。

使用方法：

```
<video src=""video.mp4"" width=""650"" height=""450""></video>
```

autoplay 属性：这个属性指定是否当我们网页加载完成之后就开始自动播放。

preload 属性：这个属性指定是否对数据进行预加载，如果是的话，浏览器会将视频数据或者音频数据进行缓冲，这样做可以加快播放的速度。

preload 属性的三个值：

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者音频。

使用方法：

```
<video src=""video.mp4"" preload=""auto"" ></video>
```

poster 属性（video 独有）：当视频不可以播放的时候，使用 poster 元素向用户展示一张图片代替视频。

使用方法：

```
<video src=""video.mp4"" poster=""video.jpg""></video>
```

loop 属性：指定是否循环播放视频或者音频数据。

使用方法：

```
<video src=""video.mp4"" autoplay=""auto"" loop=""loop""></video>
```

error 属性：读取过程中一旦发生错误，返回一个 Media Error 对象，这个对象的 code 返回对应的错误状态，默认情况下 video 和 audio 的 error 属性都是 null。

4 种错误状态，返回一个数字值，它表示音频/视频的错误状态：

1 = MEDIA_ERR_ABORTED - 取回过程被用户中止

2 = MEDIA_ERR_NETWORK - 当下载时发生错误

3 = MEDIA_ERR_DECODE - 当解码时发生错误

4 = MEDIA_ERR_SRC_NOT_SUPPORTED - 媒体不可用或者不支持音频/视频

读取错误状态示例

```
<video id=""video"" src=""video.mp4""></video>
    <script type=""text/javascript"">
        var video = document.getElementById('video');
        video.addEventListener(""error"",function(){
        var error = video.error;
        switch (error.code){
        case 1:
          alert('取回过程被用户中止。');
          break;
        case 2:
          alert('当下载时发生错误。');
          break;
        case 3:
          alert('当解码时发生错误。');
          break;
        case 4:
          alert('媒体不可用或者不支持音频/视频。');
          break;
        }
      },false);
  </script>
```

networkState 属性： networkState 属性返回音频/视频的当前网络状态（activity）

4 种错误状态，表示音频/视频元素的当前网络状态：

0 = NETWORK_EMPTY - 音频/视频尚未初始化

1 = NETWORK_IDLE - 音频/视频是活动的且已选取资源，但并未使用网络

2 = NETWORK_LOADING - 浏览器正在下载数据

3 = NETWORK_NO_SOURCE - 未找到音频/视频来源

networkState 属性：networkState 属性返回音频/视频的当前网络状态（activity）

3、video 与 audio 的 4 种方法

video 元素和 audio 元素的 4 种方法

play 方法：使用 play 方法来播放媒体，自动将元素的 paused 属性的值变成 false。

pause 方法：使用 pause 方法来暂停播放，自动将元素的 paused 属性的值变成 true。

load 方法：使用 load 方法来重新载入媒体进行播放，自动将元素的 playbackRate 属性的值变成 defaultPlaybackRate 属性的值，自动把 error 的值变成 null。

canPlayType 方法：使用 canPlayType 方法来测试浏览器是否支持指定的媒体类型。

使用方法如下：

var support = videoElment.canPlayType(type);

canPlayType() 方法可返回下列值之一：

""probably"" - 浏览器最可能支持该音频/视频类型

""maybe"" - 浏览器也许支持该音频/视频类型

"""" - （空字符串）浏览器不支持该音频/视频类型

9、HTML5 拖放 API

1、拖放的步骤

①对象元素的 draggable 属性设置为 true（draggable=""true""）。还需要注意的是 a 元素和 img 元素必须指定 href。

②编写与拖放有关的事件处理代码。事件产生事件的元素描述 dragstart 被拖放的元素开始拖放操作 drag 被拖放的元素拖放过程中 dragenter 拖放过程中鼠标经过的元素被拖放的元素开始进入本元素的范围之内 dragover 拖放过程中鼠标经过的元素被拖放的元素正在本元素范围内移动 dragleave 拖放过程中鼠标经过的元素被拖放的元素离开本元素的范围 drop 拖放的目标元素其他元素被拖放到了本元素中 dragend 拖放的对象元素拖放操作结束

2、DataTransfer 对象的属性与方法

DataTransfer 对象的属性和方法属性/方法描述 dropEffect 属性表示拖放操作的视觉效果，允许设置其值，这个效果必须用在 effectAllowed 属性指定的允许的视觉效果的范围内，允许指定的值 none、copy、link、moveeffectAllowed 属性 用来指定当元素被拖放时所允许的视觉效果，可以指定的值 copy、link 、move、copylink 、linkmove、all、none、uninitializedtypes 属性存入数据的种类，字符串的伪数组 void clearData（DOMString format）方法清除 DataTransfer 对象中存放的数据，如果省略掉参数 format 就会清除全部数据。void steData （DOMString format， DOMStrong data ）向 DataTransfer 对象中存入数据 DOMString getData(DOMString format)从 DataTransfer 对象中读取数据 void setDragImage(Element image，long x， long y)用 img 元素来设置拖放图标

clearData 方法可以用于清除 DataTransfer 对象中的数据。

3、拖放的时的效果

设置拖放时的视觉效果

dropEffect 属性与 effectAllowed 属性结合起来可以设定拖放时的视觉效果。effectAllowed 属性表示一个元素被拖放时所允许的视觉效果，一般在 ondragstart 事件中设定，他的值如下：属性值说明 copy 允许被拖动的元素被复制到项目中 move 允许将被拖动元素移动到被拖动的目标元素中 link 通过拖放操作，被拖动的元素会连接到拖到的目标元素上 copylink 被拖动元素被复制或链接到拖动的目标元素中，根据拖动的目标元素来决定执行复制操作还是链接操作 copyMove 被拖动元素复制或移动到拖到的目标元素中，根据被拖动的目标元素来决定复制操作还是移动操作 linkmove 被拖动元素被连接或移动到拖动的目标元素中，根据拖动的目标元素来决定执行链接操作还是移动操作 all 允许执行所有拖动操作 none 不允许执行任何拖动操作 uninyialize 不指定 effectAllowed 属性值。这是将执行浏览器中默认允许的拖动操作，但是这个操作不能通过 effectAllowed 属性值来获取

dropEffect 属性表示实际拖放时的视觉效果，一般在 ondragover 事件中指定，允许的值 none、copy、link、move。

四、IE 兼容

兼容问题：

<!--[if lt IE 9]>

<script src=""http://libs.useso.com/js/html5shiv/3.7/html5shiv.min.js""></script>

<script src=""http://libs.useso.com/js/respond.js/1.4.2/respond.min.js""></script>

<![endif]-->

html5shiv.min.js

可以在老式 IE 里创建 main，header，footer 等 HTML5 元素

respond.min.js

让 IE6-8 支持 CSS3 Media Query

selectivizr.js

提供大量 IE 不支持的 CSS 选择器和属性，包括所有的 last-child 选择器。

判断 IE 的方法

<!--[if lt IE 7]>IE6<![endif]-->

<!--[if IE 7]>IE7<![endif]-->

<!--[if IE 8]>IE8<![endif]-->

<!--[if IE 9]>IE9<![endif]-->

第三部分 CSS

一、使用 CSS 样式的方式

1、HTML <!DOCTYPE> 声明标签

1.定义和用法

<!DOCTYPE> 声明必须是 HTML 文档的第一行，位于 <html> 标签之前。

<!DOCTYPE> 声明不是 HTML 标签；它是指示 web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。

在 HTML 4.01 中，<!DOCTYPE> 声明引用 DTD，因为 HTML 4.01 基于 SGML。DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。

HTML5 不基于 SGML，所以不需要引用 DTD。

2.各版本的声明

HTML5：<!DOCTYPE html>、<meta charset=""utf-8"">

HTML 4.01：<!DOCTYPE HTML PUBLIC ""-//W3C//DTD HTML 4.01 Transitional//EN""

""http://www.w3.org/TR/html4/loose.dtd"">

该 DTD 包含所有 HTML 元素和属性，包括展示性的和弃用的元素（比如 font）。不允许框架集（Framesets）。

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

注意的规则：

单标记必须闭合

比如
必须写为
 <input />

单属性必须添加属性值

<input typr=""radio"" checked >必须写为<input typr=""radio"" checked=""checked"" />

标记和属性必须使用小写

<Body><BODY>是错的必须写为<body>

属性的属性值必须使用""""

在 HTML4.01 之前可以使用<body bgcolor=red>,HTML4.01 必须写为<body bgcolor=""red"">

2、内链样式表

<body style=""background-color:green; margin:0; padding:0;""></body>

3、嵌入式样式表

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

需要将样式放在<head></head>中

4、引入式样式表

<link rel=""StyleSheet"" type=""text/css"" href=""style.css"">

二、定义样式表

1、HTML 标记定义

...

p { 属性:属性值 ； 属性 1:属性值 1}

p 可以叫做选择器，定义那个标记中的内容执行其中的样式

一个选择器可以控制若干个样式属性，他们之间需要用英语的“;”隔开，最后一个可以不应加“;”

2、Class 定义

<p class=""p"">...</p>

class 定义是以“.”开始

.p { 属性:属性值 ； 属性 1:属性值 1}

3、ID 定义

<p id=""p"">...</p>

ID 定义是“#”开始的

#p { 属性:属性值 ； 属性 1:属性值 1}

4、优先级问题

1.ID>Class>HTML

2.同级时选择离元素最近的一个的

#p { color: red }

#p { color: #f60 }

执行颜色为#f60 的

5、组合选择器（同时控制多个元素）

h1,h2,h3 { font-size: 14px; color: red; }

选择器组合可以使用“,”隔开

6、伪元素选择器

.a:link：正常连接的样式

.a:hover：鼠标放上去的样式

.a:active：选择链接时的样式

.a:visited：已经访问过的链接的样式

三、常见属性

1、颜色属性

color 属性定义文本的颜色

color：green color：#ff6600 color：#f60

简写式：

color：rgb（255,255,255）

红(R)、绿(G)、蓝(B) 每个的取值范围 0~255

color：rgba（255,255,255,1）

RGBA 是代表 Red（红色） Green（绿色） Blue（蓝色）和 Alpha 的(色彩空间)透明度

2、字体属性

1.font-size 字体大小

font-size:14px

2.font-family 定义字体

font-family：微软雅黑，serif；

可以使用“,”隔开，以确保当字体不存在的时候直接使用下一个

3.font-weight 字体加粗

normal(默认值)、bold(粗)、bolder(更粗)、lighter(更细)

100、200、300~900

400 = normal，而 700 = bold

3、背景属性

1.背景颜色 background-color

2.背景图片 background-image

background-image:url(图片路径)

background-image:none

3.背景重复 background-repeat

repeat：重复平铺满

repeat-x：向 Y 轴重复

repeat-y：向 Y 轴重复

no-repeat：不重复

4.背景位置 background-position

横向（left,center,right）

纵向（top,center,bottom）

5.简写方式

background:背景颜色 url(图像) 重复 位置

background:#f60 url(images/bg,jpg) no-repeat top center

4、文本属性

1.text-align 横向排列

left，center，right

2.line-height 文本行高

%：基于字体大小的百分比行高

数值：来设置固定值

3.text-indent 首行缩进

4.letter-spacing 字符间距

normal：默认

length：设置具体的数值（可以设置负值）

inherit 继承

5、边框属性

1.边框风格 border-style

统一风格（简写风格）

border-style：单独定义某一方向的边框样式

border-bottom-style：下边边框样式

border-top-style：上边边框样式

border-left-style：左边边框样式

border-right-style：右边边框样式

边框风格样式的属性值

1)none 无边框

2)solid 直线边框

3)dashed 虚线边框

4)dotted 点状边框

5)double 双线边框

6)groove 凸槽边框

7)ridge 垄状边框

8)inset inset 边框

9)outset outset 边框

10)inherit 继承

(6~9)依托 border-color 的属性值

2.边框宽度 border-width

统一风格

border-width：单独风格

border-top-width：上边边框宽度

border-bottom-width：下边边框宽度

border-left-width：左边边框宽度

border-right-width：右边边框宽度

边框宽度的属性值：

1)thin：细边框

2)medium：中等边框

3)thick：粗边框

4)px：固定值的边框

5)inherit：继承

3.边框颜色 border-color

统一风格

border-color：单独风格

border-top-color：上边边框颜色

border-bottom-color：下边边框颜色

border-left-color：左边边框颜色

border-right-color：右边边框颜色

属性值

颜色值的名称：red、green 等

RGB：rgb(255,255,0)

RGBA：rgba(255,255,0,0.1)

十六位进制：#ff0、#ff0000

继承：inherit

属性值的四种情况

一个值：border-color:（上、下、左、右）;

两个值：border-color:（上下） （左右）;

三个值：border-color:（上） （左、右） （下）;

四个值：border-color:（上）（右）（下）（左）;

简写方式

border：solid 2px #f60

6、列表属性

1.标记的类型

list-style-type

none 无标记。

disc 默认。标记是实心圆。

circle 标记是空心圆。

square 标记是实心方块。

decimal 标记是数字。

decimal-leading-zero 0 开头的数字标记。(01, 02, 03, 等。)

lower-roman 小写罗马数字(i, ii, iii, iv, v, 等。)

upper-roman 大写罗马数字(I, II, III, IV, V, 等。)

lower-alpha 小写英文字母 The marker is lower-alpha (a, b, c, d, e, 等。)

upper-alpha 大写英文字母 The marker is upper-alpha (A, B, C, D, E, 等。)

lower-greek 小写希腊字母(alpha, beta, gamma, 等。)

lower-latin 小写拉丁字母(a, b, c, d, e, 等。)

upper-latin 大写拉丁字母(A, B, C, D, E, 等。)

hebrew 传统的希伯来编号方式

armenian 传统的亚美尼亚编号方式

georgian 传统的乔治亚编号方式(an, ban, gan, 等。)

cjk-ideographic 简单的表意数字

hiragana 标记是：a, i, u, e, o, ka, ki, 等。(日文片假名)

katakana 标记是：A, I, U, E, O, KA, KI, 等。(日文片假名)

hiragana-iroha 标记是：i, ro, ha, ni, ho, he, to, 等。(日文片假名)

katakana-iroha 标记是：I, RO, HA, NI, HO, HE, TO, 等。(日文片假名)

2.标记的位置

list-style-position

inside：列表项目标记放置在文本以内，且环绕文本根据标记对齐。

outside：默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。

inherit：规定应该从父元素继承 list-style-position 属性的值。

3.设置图像列表标记

list-style-image

URL：图像的路径。

none：默认。无图形被显示。

inherit：规定应该从父元素继承 list-style-image 属性的值。

4.简写方式

list-style

list-style:square inside url('/i/arrow.jpg');

四、DIV+CSS 布局

1.div 和 span

DIV 和 SPAN 在整个 HTML 标记中，没有任何意义，他们的存在就是为了应用 CSS 样式

DIV 和 span 的区别在与，span 是内联元素，div 是块级元素

2.盒模型

margin：盒子外边距

padding：盒子内边距

border：盒子边框宽度

width：盒子宽度

height：盒子高度

3.布局相关的属性

①position 定位方式

1)正常定位：relative

2)根据父元素进行定位：absolute

3)根据浏览器窗口进行定位：fixed

4)没有定位：static

5)继承：inherit

②定位

left（左），right（右），top（上），bottom（下）离页面顶点的距离

③z-index 层覆盖先后顺序（优先级）

④display 显示属性

display：none：层不显示

display：block：块状显示，在元素后面换行，显示下一个块元素

display：inline：内联显示，多个块可以显示在一行内

⑤float 浮动属性

left：左浮动

right：右浮动

⑥clear 清除浮动

clear：both

⑦overflow 溢出处理

hidden：隐藏超出层大小的内容

scroll：无论内容是否超出层大小都添加滚动条

auto：超出时自动添加滚动条

4.兼容问题及高效开发工具

1.兼容性测试工具：IE Tester、Multibrowser

2.常用的浏览器：Google chrome、Firefox、opera

3.高效的开发工具

轻量级的：Notepad++、sbulime Text、记事本

重量级的：WebStorm、Dreamweaver

根据自己的需要来选择(轻量级的，重量级的)

4.网页设计工具：fireworks、photoshop

5.判断 IE 的方法

条件判断格式：<!--[if 条件 版本]> 那么显示 <![endif]-->

不等于：[if !IE 8] 除了 IE8 都可以显示

小于：[if lt IE 5.5] 如果 IE 浏览器版本小于 5.5 的显示

大于：[if gt IE 5] 如果 IE 浏览器版本大于 5 的显示

小于或者等于：[if lte IE 6] 如果 IE 浏览器版本小于 6 的显示

大于或等于：[if gte IE 7] 如果 IE 浏览器版本小于 7 的显示

大于和小于之间：[if (gt IE 5)&(lt IE 7)] 如果 IE 浏览器版本大于 IE5 小于 7 的显示

或：[if (IE 6)|(IE 7)] 如果是 IE6 或者 IE7 显示

仅：<!--[if IE 8]> 如果是 IE8 显示

注意：条件注释只有在 IE 浏览器下才能执行，这样就达到了我们的效果！

6.《DIV+CSS 网页布局实战》

分析+切图+搭建框架

循序渐进>解决兼容>大功告成

第四部分 CSS3

一、CSS3 简介

1、什么是 CSS3

CSS3 是 CSS 技术的升级版本，CSS 即层叠样式表（Cascading StyleSheet）。 在网页制作时采用层叠样式表技术，可以有效地对页面的布局、字体、颜色、背景和其它效果实现更加精确的控制。 只要对相应的代码做一些简单的修改，就可以改变同一页面的不同部分，或者页数不同的网页的外观和格式。

2、css3 可以做什么

CSS3 是 CSS 技术的升级版本，CSS3 语言开发是朝着模块化发展的。以前的规范作为一个模块实在是太庞大而且比较复杂，所以，把它分解为一些小的模块，更多新的模块也被加入进来。

这些模块包括：盒子模型、文字特效、边框圆角、盒阴影、旋转、渐变

二、超级选择器

什么是选择器：

使用 css 对 HTML 页面中的元素实现一对一，一对多或者多对一的控制，这就需要用到 CSS 选择器。选择器是 CSS3 中一个重要的内容，使用 CSS 选择器可以大幅度提高开发人员的编写或者修改页面样式的时候的工作效率。每一条 css 样式定义由两部分组成，形式如：选择器{样式}

在{}之前的部分就是“选择器”。 “选择器”指明了{}中的“样式”的作用对象，也就是“样式”作用于网页中的哪些元素。

1、属性选择器

CSS3 中的属性选择器：

在 CSS 中追加了三个属性选择器：[att*=val]、[att^=val]和[att$=val]

[att*=val]属性选择器：如果元素用 att 表示的属性的值中包含用 val 指定的字符，那么该元素使用这个样式。

[att^=val]属性选择器：如果用 att 表示的属性的属性值的开头字符为用 val 指定的字符的话，那么该元素使用这个值。

[att$=val]属性选择器：如果元素用 att 表示的属性的属性值的结尾字符为用 val 指定的字符，那么该元素使用这个样式。

2、结构性伪类选择器

伪类选择器以及伪元素：

1.类选择器

在 css 中可以使用类选择器把相同的元素定义成不同的样式。比如：

```
p.left{text-align: left}
p.rigth{text-align: right}
```

2.伪类选择器

类选择器和伪类选择器的区别在于，类选择器我们可以随意起名，而伪类选择器是 CSS 中已经定义好的选择器，不可以随意起名。

最常见的伪类选择器

```
a:link{ color: #ff6600 } /* 未被访问的链接 */
a:visited{ color: #87b291 } /* 已被访问的链接 */
a:hover{ color: #6535b2 } /* 鼠标指针移动到链接上 */
a:active{ color: #55b28e } /* 正在被点击的链接 */
```

3.伪元素选择器

伪元素选择器，针对于 CSS 中已经定义好的伪元素使用的选择器。

使用方法：

选择器：伪元素{属性：值}，与类配合使用

选择器.类名：伪元素{属性：值}

4.在 CSS 中，主要有四个伪元素选择器

1)、first-line 伪元素选择器

first-line 伪元素选择器用于向某个元素中的第一行文字使用样式。

2)、first-letter 伪元素选择器

first-letter 伪元素选择器用于向某个元素中的文字的首字母（欧美文字）或第一个字（中文或者是日文等汉字）使用样式。

3)、before 伪元素选择器

before 伪元素选择器用于在某个元素之前插入一些内容。

4)、after 伪元素选择器

after 伪元素选择器用于在某个元素之后插入内容。

5.结构性伪类选择器 root、not、empty 和 target

1)、root 选择器

root 选择器将样式绑定到页面的根元素中。

2)、not 选择器

想对某个结构元素使用样式，但是想排除这个结构元素下面的子结构元素，让它不使用这个样式时，我们就可以使用 not 选择器。

3)、empty 选择器

empty 选择器指定当元素中内容为空白时使用的样式。

4)、target 选择器

target 选择器对页面中某个 target 元素指定样式，该样式只在用户点击了页面中的超链接，并且跳转到 target 元素后起作用。

6.选择器 first-child、last-child、nth-child 和 nth-last-child

1)、first-child 选择器

first-child 单独指定第一个子元素的的样式。

2)、last-child 选择器

last-child 单独指定最后一个子元素的的样式。

3)、nth-child 选择器

nth-child(n) 选择器匹配正数下来第 N 个子元素

nth-child(odd)选择器匹配正数下来第奇数个子元素

nth-child(even)选择器匹配正数下来第偶数个子元素

4)、nht-last-child 选择器

nth-last-child(n) 选择器匹配倒数数下来第 N 个子元素

nth-last-child(odd)选择器匹配倒数数下来第奇数个子元素

nth-last-child(even)选择器匹配倒数下来第偶数个子元素

7.选择器 nth-of-type 和 nth-last-of-type

1)、在使用 nth-child 和 nth-last-child 时产生的问题

在案例中指定奇数文章的标题背景为黄色，偶数文章的标题为绿色。 <h2>标题</h2> 内容 <h2>标题</h2> 内容 <h2>标题</h2> 内容 <h2>标题</h2> 内容 ... ...

2)、使用 nth-of-type 和 nth-last-of-type

nth-of-type 和 nth-last-of-type 在 css3 中就是用来避免上面这类问题的发生，在统计的时候就只针对同类型的子元素进行计算。

nth-of-type 正数

nth-last-of-type 倒数

兼容性：

nth-of-type 和 nth-last-of-type 都是 CSS3 开始提供需要在 IE8 以上的浏览器才会被支持，Chrome 浏览器、Firefox 浏览器、Opera 浏览器、Safari 浏览器都支持！

8.循环使用样式

nth-child(An+B)A 表示每次循环中共包括几种样式，B 表示指定的样式在循环中所处的位置。

9.only-child 选择器

only-child 选择器，只对唯一的子元素起作用。

3、UI 元素状态伪类选择器

UI 元素状态伪类选择器

什么是 UI 选择器？

UI 选择器的特征就是指定的样式只有当元素处于某种状态下时，才起作用，在默认状态下不起作用！

E:hover 支持 firefox、safari、Opera、ie8、chrome

E:active 支持 firefox、safari、Opera、chrome 不支持 ie8

E:focus 支持 firefox、safari、Opera、ie8、chrome

E:enabled 支持 firefox、safari、Opera、chrome 不支持 ie8

E:disabled 支持 firefox、safari、Opera、chrome 不支持 ie8

E:read-only 支持 firefox、Opera 不支持 ie8、safari、chrome

E:read-write 支持 firefox、Opera 不支持 ie8、safari、chrome

E:checked 支持 firefox、safari、Opera、chrome 不支持 ie8

E::selection 支持 firefox、safari、Opera、chrome 不支持 ie8

E:default 只支持 firefoxE:indeterminate 只支持 chrome

E:invalid 支持 firefox、safari、Opera、chrome 不支持 ie8

E:valid 支持 firefox、safari、Opera、chrome 不支持 ie8

E:required 支持 firefox、safari、Opera、chrome 不支持 ie8

E:optional 支持 firefox、safari、Opera、chrome 不支持 ie8

E:in-range 支持 firefox、safari、Opera、chrome 不支持 ie8

E:out-of-rang 支持 firefox、safari、Opera、chrome 不支持 ie8

1.选择器 E:hover、E:active 和 E:focus

1)、E:hover 选择器被用来指定当鼠标指针移动到元素上时元素所使用的样式

使用方法： <元素>:hover{ CSS 样式 }

我们可以在“<元素>”中添加元素的 type 属性。

例：

input[type=""text""]:hover{ CSS 样式 }

2)、E:active 选择器被用来指定元素被激活时使用的样式

3)、E:focus 选择器被用来指定元素获得光标聚焦点使用的样式，主要是在文本框控件获得聚焦点并进行文字输入时使用。

4)、小案例：

结合上诉实现以下效果，当鼠标移动上去是文本框变成绿色，当鼠标点击时（点击并未谈起）文本框处于蓝色，当处于可输入状态时文本框变成橙色。

2.E:enabled 伪类选择器与 E:disabled 伪类选择器

1)、E:enabled 选择器被用来指定当元素处于可用状态时的样式。

2)、E:disabled 选择器被用来指定当元素处于不可用状态时的样式。

3)、小案例：

结合上诉实现以下效果，文本框可用时背景颜色绿色，不可用时背景变成浅灰色。

3.E:read-only 伪类选择器与 E:read-write 伪类选择器

1)、E:read-only 选择器被用来指定当元素处于只读状态时的样式。

2)、E:read-write 选择器被用来指定当元素处于非只读状态时的样式。

3)、小案例：

结合上诉实现以下效果，文本框可用时输入文字变成红色，不可用时，设置里面的背景为黑色，文字为绿色。

4.伪类选择器 E:checked、E:default 和 indeterminate

1)、E:cehcked 伪类选择器用来指定当表单中的 radio 单选框或者是 checkbox 复选框处于选取状态时的样式。

2)、E:default 选择器用来指定当页面打开时默认处于选取状态的单选框或复选框的控件的样式。

3)、E:indeterminate 选择器用来指定当页面打开时，一组单选框中没有任何一个单选框被设定为选中状态时，整组单选框的样式。

4)、小案例：

A、模拟一个发布房产信息的选框控件，当选择以后，会给他一个 2px 实线 绿色的边框！

效果：

B、实现一个默认选择中的控件，默认给他一个 2px 实线 绿色的边框！

效果：

当他去掉勾以后还会有样式

C、实现一个在默认打开都没有选中时，给他们一个 2px 实线 绿色的边框！

效果：

5.伪类选择器 E::selection

1)、E:selection 伪类选择器用来指定当元素处于选中状态时的样式。

2)、小案例：

选择网页中 p 段落中的文字和文本框中的文字，改变他的背景颜色为绿色！

效果：

选择网页中 p 段落中的文字

选择文本框中的文字

6.E:invalid 伪类选择器与 E:valid 伪类选择器

1)、E:invalid 伪类选择器用来指定，当元素内容不能通过 HTML5 通过使用的元素的诸如 requirde 等属性所指定的检查或元素内容不符合元素规定的格式时的样式。

2)、E:valid 伪类选择器用来指定，当元素内容能通过 HTML5 通过使用的元素的诸如 requirde 等属性所指定的检查或元素内容符合元素规定的格式时的样式。

3)、小案例：

通过 Email 属性的表单判断输入是否正确，正确输入的内容为绿色，输入错误就是红色！

效果：

错误

正确

7.E:required 伪类选择器与 E:optional 伪类选择器

1)、E:required 伪类选择器用来指定允许使用 required 属性，而且已经指定了 required 属性的 input 元素、select 元素以及 textarea 元素的样式。

2)、E:optional 伪类选择器用来指定允许使用 required 属性，而且未指定了 required 属性的 input 元素、select 元素以及 textarea 元素的样式。

3)、小案例：

姓名必填表单背景红色，年龄可选填表单背景绿色

效果：

8.E:in-range 伪类选择器与 E:out-of-range 伪类选择器

1)、E:in-range 伪类选择器用来指定当元素的有效值被限定在一段范围之内，且实际的输入值在该范围之内时的样式。

2)、E:out-of-range 伪类选择器用来指定当元素的有效值被限定在一段范围之内，但实际输入值在超过时使用的样式。

3)、小案例：

0-100 以内的数字，小于 0 或超过 100 字体变成红色，否则绿色。

效果：

100 时：

1000 时：

4、通用兄弟元素选择器

通用兄弟元素选择器

他用来指定位于同一个父元素之中的某个元素之后的所有其他某个种类的兄弟元素所使用的样式。

使用方法：

```
<子元素>~<之元素之后的同级兄弟元素>{
  CSS 样式
}
```

三、使用选择器在页面中插入内容

在前面使用 before 伪元素和 after 伪元素在页面中的元素前或者后加入内容，而插入的内容是以 content 属性来定义的。

before 伪元素和 after 伪元素在 CSS2.0 的时候就已经被添加，从 CSS2 到 CSS3 一直都在对这两个选择器进行改良和扩展，从而使得我们的 before 伪元素和 after 伪元素的功能越来越强大。

目标：能够熟练的使用 before 伪元素和 after 伪元素在页面中插入文字、图像、项目编号等等。

1、插入文字

使用选择器插入内容

在 CSS2 中，使用 before 选择器在元素前面插入内容，使用 after 在元素后面插入内容，在选择器 content 属性中定义要插入的内容。

例如对 H2 使用 before 选择器在 H2 的前面插入文字“Title”。

```
<style type=""text/css"">
h2:before{
content:""Title"";
}
</style>
```

并且我们还可以给他定义样式，进行美化操作。

比如我们给“Title”的文字设置为白色，加上绿色的背景，内边距上下 1 像素左右 5 像素，外边距右边 5 像素，当然还可以设置他的字体等等。

1-2、排除一些不需要插入内容的元素

使用 content 属性的追加一个 none 属性值。

比如：

```
<style type=""text/css"">
 h2.nocontent:before{
 content:none;
 }
 </style>
```

2、插入图片

1、插入图片文件

使用 before 或者 after 除了可以在元素前后插入文字之外还可以插入图片。

在插入图片是需要使用 URL 指定图片的路径，比如在标题前插入一张图片！

比如：

```
<style type=""text/css"">
 h2:before{
 content:url(1.gif);
 }
 </style>
```

2、插入图片文件的好处

节省开发人员的工作量，比如可以通过类的方式来进行不同标题图片的追加！

比如我们给标题定义一个“hot”“digest”分别来调用一张站的图标个顶的小图标！

```
<style type=""text/css"">
 h2.hot:before{ content:url(hot.gif); }
 h2.digest:before{ content:url(digest.gif); }
 </style>
<h2 class=""hot"">这是标题测试</h2>
<h2 class=""digest"">这是标题测试</h2>
<h2 class=""hot"">这是标题测试</h2>
<h2>这是标题测试</h2>
<h2>这是标题测试</h2>
```

3、插入项目编号

1、在多个标题前加上连续编号

在 content 属性中使用 counter 属性来正对对个项目追加连续的编号。

使用方法：

```
 元素:before{
 content:counter(计数器);
 }
```

使用计数器来计算编号，计数器可以任意命名。

除了使用计数器，还需要在元素的样式中追加对元素的（counter-increment）属性的指定为 content 属性值中所指定的计数器名称。

```
 元素{
 counter-increment:content 属性值中所指定的计数器名称
 }
```

2、在项目编号中追加文字

```
h1:berore{
content：'第'counter（content 属性值中所指定的计数器名称）'章';
}
```

变成另一种效果！“第一章、第二章...”

3、指定编号的样式

比如给他在编号后面带一个“.”文字，并且设置编号的颜色为绿色，字体大小 42 像素。

```
h1:before{
content：counter（content 属性值中所指定的计数器名称）'.';
```

color:green;

font-size:42px;

}

变成另一种效果！“第一章、第二章...”

4、指定编号的种类

before 和 after 不仅可以追加数字编号，还可以追加字母编号或者罗马数字编号。

content：counter（计数器名,编号种类）

可以使用 list-style-type 属性的值来指定编号的种类。比如指定大写字母编号时，使用“upper-alpha”属性，指定大写罗马字母时使用“upper-roman”属性。

```
h1:before{
content：counter（计数器名,编号种类）'.';
color:green;
font-size:42px;
}
```

5、编号嵌套、重置编号

我们可以在大编号中华嵌套中编号，在中编号中呢又嵌入小编号！

1)、大编号中嵌入中编号

2)、counter-reset 属性重置编号

6、中编号中嵌入大编号

```
h2:before{
content:counter(大编号的计数器)'-'counter(中编号的计数器)
}
```

7、在字符串两边嵌套文字符号

可以使用 content 属性的 open-quote 属性值与 close-quote 属性值在字符串的两边添加诸如单引号、双引号之类的文字字符。

open-quote 开始符号

close-quote 结尾字符

quote 字符类型（使用双引号（""""）的时候需要使用转义字符“\”）

示例：

```
<style>
h1:before{
content:open-quote;
}
h1:after{
content:close-quote;
}
h1{
quote:""("" "")"";
}
</style>
<h1>标题</h1>
```

四、文字阴影与自动换行

CSS3 中与文字字体相关的一些属性做详细的介绍，比如 text-shadow 属性、word-break 属性、word-wrap 属性等等。

目标：

掌握 text-shadow 属性给页面的文字加阴影效果

掌握 word-break 属性让页面文字自动换行

掌握 word-wrap 属性让浏览器在长单词或很长的 URL 地址的中间进行换行。

掌握 如何使用浏览器在显示文字的时候使用服务器端的字体。

1、text-shadow

1.text-shadow 属性的使用方法

在 CSS3 我们可以是用 text-shadow 属性给页面上的文字添加阴影效果，text-shadow 在 CSS2.1 的时候是被删除了的一个属性，但是呢在 3.0 的 CSS 中又恢复了使用。

text-shadow 的使用方法：

text-shadow：length length length color

第一个 length 表示的是阴影离开文字的横方向距离；

第二个 length 表示的是阴影离开文字的纵方向的距离；

第三个 length 表示的是阴影模糊半径；

color 表示的是阴影的颜色。

2.位移距离

text-shadow 所使用的参数中，前两个参数是阴影离开文字的横方向和纵方向的位移距离，使用的时候必须指定这两个参数

3.阴影的模糊半径

text-shadow 属性的第三个参数就是阴影模糊半径，代表阴影向外模糊时的模糊范围。

4.阴影的颜色

text-shadow 属性的第四个参数就是绘制阴影时所使用的颜色，可以放在三个参数之前，也可以放在之后。当没有指定颜色值的时候，会使用 Color 的颜色值。

5.指定多个阴影

我们可以使用 text-shadow 属性来给文字指定多个阴影，并且针对每个阴影使用不同的颜色。指定多个阴影的时候使用逗号“,”将多个阴影进行分割。

2、word-break

1.浏览器文本自动换行

学习 word-break 之前我们先来看看浏览器对默认文本的换行处理方法。

2.指定自动换行的处理方法

在 CSS3 中可以使用 word-break 属性来设置自动换行的处理方法。值换行规则 IE5+Safari、chromenormla 使用浏览器默认的规则支持支持 keep-all 只能在半角空格或连字符处理换行支持不支持 break-all 允许在单词内换行支持支持

使用示例：

```
<style>
div{
  word-break:keep-all
}
</style>
```

3、word-wrap

长单词与 URL 地址自动换行

在 CSS3 中，使用 word-wrap 属性来实现长单词和 URL 的自动换行。

使用方法：

```
div{
```

```
    word-wrap:break-word;
}
```

word-wrap 属性的属性值有两个

第一个：normal 浏览器保持默认处理方式，只在半角空格或者是连字符的地方换行

第二个：break-word 浏览器可以在长单词或 URL 地址内部进行换行。

4、服务器端字体和@font-face 属性

1.在页面上显示服务器端的字体

在 CSS3 中可以使用@font-face 属性来利用服务器端字体。

@font-face 属性的使用方法：

```
@font-face{
font-family:webFont;
src:url('font/字体名称.otf')format(""opentype"");
}
```

font-family 属性值中使用 webfont 来声明使用的是服务器端字体。

src 属性值中首先指定了字体文件所在的路径，format 声明字体文件的格式，可以省略文件格式的声明，单独使用 src 属性值。

可以使用的字体文件格式：字体格式字体属性 otfopentypettftruetypeeotembedded-opentype

2.定义斜体或粗体字体

在定义字体的时候，可以把字体定义成斜体或者粗体，在使用服务器服务器端字体的时候，需要根据是斜体还是粗体，使用不同的文字 。

使用方法如下：

```
@font-face {
    font-family: webfont;
    src: url(字体 1.otf);
}
@font-face {
    font-family: webfont;
    font-style: italic;
    src: url(字体 2.otf);
}
h1{
    font-family: webfont;
}
h2{
    font-family: webfont;
    font-style: italic;
}
```

3.显示客户端本地的字体

@font-face 除了可以显示服务器端的字体还可以显示本地字体。

首先将 font-family 设置为本地的字体名，然后将 src 属性设置为 local('字体')。

例如：

```
@font-face{
```

font-family:Arial;

src：local('Arial');

}

4.属性值的指定：

1)、font-family：设置文本的字体名称。

2)、font-style：设置文本样式。

取值：normal 不使用斜体

italic 使用斜体

oblique 使用倾斜体

inherit 从父元素继承

3)、font-variant：设置文本是否大小写。

取值：

normal 使用浏览器默认值

small-caps 使用小型大写字母

inherit 从父元素继承

4)、font-weight：设置文本的粗细。

取值：

normal 使用浏览器默认值

bold 使用粗体

bolder 使用更粗的字体

lighter 使用更细的字体

100-900 从细字体到粗，值必须是 100 的倍数，其中 400 等于 normal，700 等同于 bold。

5)、font-stretch：设置文本是否横向的拉伸变形。（IE 及 Firefox 已支持 font-stretch，但显示效果与正常文字并无不同。）

取值：

normal：正常文字宽度

wider 把伸展比例设置为更进一步的伸展值 ；

narrower：把收缩比例设置为更进一步的收缩值；

ultra-condensed：比正常文字宽度窄 4 个基数；

extra-condensed：比正常文字宽度窄 3 个基数；

condensed：比正常文字宽度窄 2 个基数；

semi-condensed：比正常文字宽度窄 1 个基数；

semi-expanded：比正常文字宽度宽 1 个基数；

expanded：比正常文字宽度宽 2 个基数；

extra-expanded：比正常文字宽度宽 3 个基数；

ultra-expanded：比正常文字宽度宽 4 个基数；

6)、font-size：设置文本字体大小。

7)、src：设置自定义字体的相对路径或者绝对路径，注意，此属性只能在@font-face 规则里使用。

五、盒模型

CSS3 中各种盒模型、概念、使用方法和浏览器的支持情况，以及当内容溢出是的操作办法，给盒子添加阴影效果，和如何使用 CSS3 中的属性来定义元素的宽度值和高度值中是否包含内补白区域，以及边框的宽度和高度。

目标：

1、掌握 CSS3 中各种盒的类型、概念、使用方法以及浏览器的支持情况。

2、当内容溢出时，如何按照自己的想法来让浏览器进行显示

3、掌握 CSS3 加阴影的方法，可以给盒加阴影。

1、各种盒模型

1、盒的基本类型：

在 css3 中使用 display 属性来定义盒的类型，总体来说盒分为 block 类型和 inline 类型。

比如之前我们所学的 div 元素和 p 元素就是属于 block 类型，span 和 a 属于 inline 类型。

我们可以通过一个小小的实验来对比两种盒。

inline-block 类型

1.inline-block 类型

inline-block 类型是 css2.1 中追加的一个盒类型。

inline-block 类型盒属于 block 盒的一种，但是在显示时具有 inline 类型盒的特点。

例如：在 DIV 元素中分别将 display 设置为 inline-block 和 inline 后他们的效果一样。

如果我们给他们都指定宽度和高度结果会是怎样？

2.使用 inline-block 类型来执行分列显示

在 css2.1 之前，如果需要在一行中并列显示多个 block 类型的元素，需要用 float 属性或者 position 属性，但是这样会让我们的样式变的复杂，所以在 CSS2.1 中就追加了 inline-block 类型，使得并列显示多个 block 类型的元素操作变的很简单。

示例对比：

block 类型

首先新建三个 div 元素，给前两个 DIV 使用 float 让前两个 div 元素并列显示，第三个 div 的话就会显示在前两个 div 元素的下部，但是因为前两个的高度不一样所以我们要使用 clear 属性清除浮动。

inline-block 类型

使用 inline-block 类型可以直接将两个 div 元素进行并列显示，不需要使用 float 属性，也不需要使用 clear 属性了。

默认情况下使用 inline-block 类型时并列显示的元素的垂直对齐方式是底部对齐，为了让对齐方式改成顶部对齐，还要给 DIV 元素的样式中加入 vertical-align 属性。

3.使用 inline-block 类型来显示水平菜单

在 css2.1 之前，如果要实现水平菜单，那么我们需要使用 float 属性，大多数菜单是利用 ul 列表和 li 列表项目来显示的，li 元素隶属于 block 类型下的 list-item 类型，所以要并列显示的话就要使用 float 属性。

inline-table 类型

1.inline-table 类型

inline-table 类型是 css2 中新增的盒类型 -- inline-table

实例讲解：

在 css 中使用 table 元素的示例，一个表格，前后都有文字将其围绕。

list-item 类型

list-item 类型，可以将多个元素作为列表来显示，同时在元素的开头加上列表的标记。

run-in 类型与 compact 类型

将元素指定为 run-in 类型或 compact 类型时，如果元素后面还有 block 类型的元素，run-in 类型那个的元素将被包括在后面的 block 类型的元素的内部，而 compact 类型的元素将被放置在 block 类型的元素左边。

表格相关类型

在 CSS3 中所有与表格相关的元素及其所属类型表：元素所属类型说明 tabletable 代表整个表格 tableinline-table 代表整个表格可以被指定为 table 类型也可以是 inline-table 类型 trtable-row 代表表格中的一行 tdtable-cell 代表表格中的单元格 thtable-cell 代表但单元格中的列标题 tbodytable-row-group 代表表格中行的所有行 theadtable-header-group 代表表格中的表头部分 tfoottable-footer-group 代表表格中的脚注部分 coltable-columm 代表表格中的一列 colgrouptable-column-group 代表表格中的所有列 captiontable-caption 代表整个表格的标题

通过一个小示例，来指定各种 div 的类型为各种表格相关的类型，让这些 DIV 构成一个完整的表格

none 类型

当元素被指定了 none 类型后，这个元素将不会被显示

小示例使用前面所学的伪类选择器来实现鼠标移动上去隐藏，移开后显示。

各种浏览器对各种盒模型的支持情况

盒类型

	Firefox	Safari	Opera	IE8	Chrome
inline					
inline-block					

blockrun-innonetabletable-celltable-columntable-colgrouptable-header-grouptable-row-grouptable-footer-grouptable-rowtable-captionYESYESYESNOYESYESYESYESYESYESYESYESYES YESYESYESYESYESYESYESYESYESYESYESYES

YESYESYESYESYESYESNONOYESYESYESYESYES YESYESYESNOYESYESYESYESYESYESYESYES

YESYESYESNOYESYESYESNONOYESYESYESYESYES

2、显示不下的内容

当我们在样式中设置了宽度和高度之后，就有可能出现内容溢出的情况，就可以使用 overflow 属性来指定如何显示盒子内显示不下的内容。

1、overflow 属性

overflow 属性用来指定对盒中容纳不下的内容的显示办法。

取值：

hidden 超出容纳范围的文字将被隐藏起来 scroll 在 div 元素中出现水平与垂直的滚动条，超出的内容被滚动显示 auto 当文字超出 div 元素的容纳范围时，根据需要出现水平滚动条或者是垂直滚动条，滚动显示超出的内容。visible 和不使用 overflow 时候的显示效果一样，超出容纳范围的文字按原样显示

2、overflow-x 属性与 overflow-y 属性

使用 overflow-x 属性或者 overflow-y 属性，可以单独指定可以在水平或者垂直方向上如果内容超出盒的容纳范围时的显示方法。

小示例：

将 overflow-x 属性设置为 hidden，将 overflow-y 属性设置为 scroll，那么超出之后只会在垂直方向上显示滚动条。

3、text-overflow 属性

通过使用 text-overflow 属性，可以在盒的末尾显示一个代表省略的符号“...”，但是 text-overflow 属性只在当盒中的内容在水平方向上超出盒的容纳范围时有效

小示例：

通过将 white-space 属性的属性值设定为 nowrap，使得盒的右端内容不能换行显示，这样就达到了水平方向溢出的效果了。

3、盒阴影

1、box-shadow 属性的使用方法

在 CSS3 中，可以使用 box-shadow 属性让盒在显示的时候产生阴影效果。

使用方法：

box-shadow：length length length color

前三个 length 表示文字阴影离开文字的横方向距离，阴影离开文字的纵方向距离和阴影的模糊半径，color 制定阴影的颜色。

2、将参数设置为 0

1、当阴影的模糊半径设定为 0 的时候，将绘制不向外模糊的阴影。

2、将阴影离开文字的横方向距离与阴影离开文字的纵方向距离都设置为 0 的时候，会在盒子的周围绘制阴影。

3、将阴影离开文字的横方向距离设定为负数值的时候，向左绘制阴影。

4、将阴影离开文字的纵方向距离设定为负数值的时候，向上绘制阴影。

3、对盒内子元素使用阴影

可以单独对盒内的子元素使用阴影。

小示例：

有一个 DIV 元素，div 元素内部有一个 span 子元素，使用 box-shadow 属性让 span 子元素具有阴影效果。

4、对第一个文字或第一行使用阴影。

通过使用 first-letter 选择器或 frist-line 选择器可以只让第一个字或第一行具有阴影效果。

5、对表格及单元格使用阴影

可以使用 box-shadow 属性让表格及表格内的单元格产生阴影效果。

4、box-sizing 宽高计算

1、box-sizing 属性

在 CSS3 中，使用 box-sizing 属性来指定针对元素的宽度与高度的计算方法。

box-sizing 可以指定用 width 属性与 height 属性分别指定的宽度只与高度值是否包含元素的内补白区域以及边框的宽度和高度。

可以给 box-sizing 属性指定的属性值为 content-box 属性值与 border-box 属性值。

content-box 属性值表示元素的宽度与高度不包括内补白区域及边框的宽度高度；

border-box 属性值表示元素的宽度与高度包括内补白区域及边框的宽度与高度，在没有使用 box-sizing 属性的时候，默认值是 content-box 属性值。

2、为什么要使用 box-sizing 属性

使用 box-sizing 属性的目的是对元素的总宽度做一个控制，如果不适应这个属性的话，样式中默认使用的是 content-box 属性值，他只对内容做一个指定，却没有队员说的总宽度进行指定，有些场合合理利用 border-box 属性值会是页面的布局很方便。

六、背景和边框的相关样式

主要讲解 CSS 中与背景和边框相关的一些样式，包括背景相关的几个属性，如何在一个背景中使用多个图像文件、如何绘制圆角边框、如何给元素添加图像边框。

目标：

1、掌握 CSS 3 中新增的与背景相关的属性的概念、使用方法以及各种浏览器的支持情况。

2、学会如何使用 CSS 3 中的 border-radius 属性给元素添加一个圆角边框。

3、学会如何使用 CSS 3 中的 border-images 属性给元素添加一个可以随元素尺寸的变化而自动伸缩的图像边框。

1、新增的背景属性

属性功能 background-clip 指定背景的显示范围 background-origin 指定绘制背景图象时的起点 background-size 指定背景中图像的尺寸 background-break 指定内联元素的背景图像进行平铺时的循环方式

1、background-clip 指定背景的显示范围

css3 中的 background-clip 属性，它主要是用来控制我们的背景显示区域。

首先呢，我们来看一下它的用法

background-clip ： border-box || padding-box || content-box

1、border-box:此值为默认值，背景从 border 区域向外裁剪，也就是超出部分将被裁剪掉；

2、padding-box：背景从 padding 区域向外裁剪，超过 padding 区域的背景将被裁剪掉；

3、content-box：背景从 content 区域向外裁剪，超过 context 区域的背景将被裁剪掉；

浏览器支持情况：

background-clip 在各种浏览器内核下，都具有自己的私有前缀：

Firefox3.6 版本以下（包含 3.6 版本）：

-moz-background-clip: border || padding

Firefox4.0 版本以上：

Firefox4.0 版本以上，支持 border-box,padding-box,content-box 并且无需带上其前缀，如果你一不小心带上了“-moz-”，那么在 Firefox4.0+版本反而是一种错误的写法

background-clip: border-box || padding-box || content-box

background-clip 兼容各浏览器的正确写法应该如下：

/*Firefox3.6-*/

-moz-background-clip: border || padding;

/*Webkit*/

-webkit-background-clip: border-box || padding-box || context-box;

/*W3C 标准 IE9+ and Firefox4.0+*/

background-clip: border-box || padding-box || context-box;

2、background-origin 属性指定绘制背景图像的绘制起点

在绘制背景图象时，默认是从内补白（padding）区域的左上角开始绘制的，但是可以利用 background-origin 属性来指定绘制时从边框的左上角开始绘制，或者是从内容的左上角开始绘制。

padding-box(padding):此值为 background-origin 的默认值，决定 background-position 起始位置从 padding 的外边缘（border 的内边缘）开始显示背景图片；

border-box(border):此值决定 background-position 起始位置从 border 的外边缘开始显示背景图片；

content-box(content):此值决定 background-position 起始位置从 content 的外边缘（padding 的内边缘）开始显示背景图片；

语法：

background-origin 在早期的 Webkit 和 Gecko 内核浏览器（Firefox3.6-,Safari 和 Chrome 代版本）他们都支持一种老式的语法规则，类似于 background-clip 在 Firefox3.6 以下的版本一样

background-origin: padding || border || content

那么在那些现代浏览器都支持的是一种新的语法

background-origin: padding-box || border-box || content-box

为了兼容新老版本的浏览器，在使用 background-origin 改变 background-position 的原点位置时，最好老旧语法一起加上，并且新语法放在老语法后面，这样只要是支持新语法规则的浏览器就自动会识别 background-origin 的最新语法

background-origin: padding || border || content

background-origin: padding-box || border-box || content-box

取值说明：

1、padding-box(padding):此值为 background-origin 的默认值，决定 background-position 起始位置从 padding 的外边缘（border 的内边缘）开始显示背景图片；

2、border-box(border):此值决定 background-position 起始位置从 border 的外边缘开始显示背景图片；

3、content-box(content):此值决定 background-position 起始位置从 content 的外边缘（padding 的内边缘）开始显示背景图片；

有一点需要提出，在ＩＥ8 以下版本解析是不一样的，在 IE7 以下版本 background-origin 默认是从 border 开始显示背景图片。

兼容浏览器：

background-origin 虽然现代浏览器都支持，但在不同内核浏览器下还是需要带上其各自的前缀，这样在实际应用中最好按下面的语法规则书写，以保证只要支持 background-origin 的都能正常运行：

/*Old Webkit and Gecko*/

-moz-background-origin: padding || border || content;

-webkit-background-origin: padding || border || content;

/*New Webkit and Gecko*/

-moz-background-origin: padding-box || border-box || content-box;

-webkit-background-origin: padding-box || border-box || content-box;

/*Presto*/

-o-background-origin: padding-box || border-box || content-box;

/*W3c 标准*/

background-origin: padding-box || border-box || content-box;

3、background-size 属性指定背景图像的尺寸

在 CSS3 中，可以使用 background-size 属性来指定背景图像的尺寸。

使用方法：

background-size: auto || <length> || <percentage> || cover || contain

1、auto:此值为默认值，保持背景图片的原始高度和宽度；

2、length：设置背景图像的高度和宽度,第一个值设置宽度，第二个值设置高度,如果只设置一个值，则第二个值会被设置为""auto""。

3、percentage：以父元素的百分比来设置背景图像的宽度和高度，第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 ""auto""。

4、cover：此值是将图片放大，以适合铺满整个容器，这个主要运用在，当图片小于容器时，又无法使用 background-repeat 来实现时，我们就可以采用 cover;将背景图片放大到适合容器的大小，但这种方法会使用背景图片失真；

5、contain:此值刚好与 cover 相反，其主要是将背景图片缩小，以适合铺满整个容器，这个主要运用在，当背景图片大于元素容器时，而又需要将背景图片全部显示出来，此时我们就可以使用 contain 将图片缩小到适合容器大小为止，这种方法同样会使用图片失真。

兼容的浏览器

background-size 和其他的一些 CSS3 属性一样，需要加上自己的别名，

/*Mozilla*/

-moz-background-size: auto || <length> || <percentage> || cover || contain

/*Webkit*/

-webkit-background-size: auto || <length> || <percentage> || cover || contain

/*Presto*/

-o-background-size: auto || <length> || <percentage> || cover || contain

/*W3c 标准*/

background-size: auto || <length> || <percentage> || cover || contain

4、Background-break 属性指定背景图像的尺寸

css3 里标签元素能被分在不同区域（如：让内联元素 span 跨多行），background-break 属性能够控制背景在不同区域显示。

属性值：

(1)Background-break: continuous;

此属性是默认值，忽视区域之间的间隔空隙（给它们应用图片就好像把它们看成一个区域一样）

(2)Background-break: bounding-box;

重新考虑区域之间的间隔

(3)Background-break: each-box;

对每一个独立的标签区域进行背景的重新划分。

2、显示多个背景图片

在一个元素中显示多个背景图像

在 CSS3 中一个元素可以显示多个背景图像，还可以将多个背景图像进行重叠显示，这样可以让我们对背景中所用素材的调整变的更加容易。

使用方法：

background-image:url(1.png),url(2.png),url(3.png);

图层的排序方法：浏览器中显示时叠放的顺序从上往下指定的，第一个图像文件是放在最上面的，最后指定的文件是放在最下面的。

3、圆角边框

border-radius 属性

在 css3 中可以使用 border-radius 进行圆角边框的绘制，在网页中呢，我们经常使用圆角边框来美化我们的页面，在 css3 出现之前呢！我们如果要在网页上展示一个圆角边框的一个效果,那么我们需要绘制图形，在 css3 出现之后呢我们就只需要一段简单的代码就可以实现圆角边框的效果。

使用方法：border-radius:半径

border-radius 属性使用的时候，我们只需要定义好圆角的半径就可以绘制圆角边框了。

兼容性：

IE9+、Firefox 4+、Chrome、Safari 5+ 以及 Opera 支持 border-radius 属性。

border-radius 属性中指定两个半径

在 border-radius 属性中，可以指定两个半径，制定方法如下：

border-radius:40px 20px;

第一个半径作为边框左上角与边框右下角的圆半径来绘制。

第二个半径作为边框右上角与边框左下角的圆半径来绘制。

不显示边框时

在 CSS3 中，如果使用了 border-radius 属性但是把边框设置为不显示时，浏览器会把背景的四个角绘制成圆角。

修改边框种类时

使用了 border-radius 属性后，不管边框是什么种类，都会将边框沿着圆角曲线进行绘制。

绘制 4 个不同半径的圆角边框

border-top-left-radius:左上角半径

border-bottom-left-radius:左下角半径

border-top-right-radius：右上角半径

border-bottom-right-radius:右下角半径

4、图像边框

使用图像边框

在 css3 中增加了一个 border-image 属性,可以让元素的长度或宽度处于随时变化时，变化状态的边框统一使用一个图像文件来进行绘制.

1、border-image 属性最简单的使用方法：

-webkit-border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距 ；

-moz-border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距 ；

-o-border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距 ；

border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距 ；

上面的参数中，图像的路径、上边距、右边距、下边距、左边距必须进行指定，但是如果上边距、右边距、下边距、左边距得值完全一样，那么就可以缩写为一个。

写法如下：

border-image: url(""边框图像的路径"") 边距 ；

当我们指定了上边距右边距下边距左边距之后呢！浏览器就会对我们的这个背景图像进行切割，那么他是怎样去对我们的这个图像进行切割的呢？

他会把图像切割成九个部分。

border-top-left-imageborder-top-imageborder-top-right-imageborder-left-imageborder-right-imageborder-bottom-left-imageborder-bottom-imageborder-bottom-right-image

在浏览器中显示的时候，border-top-left-image、border-top-right-image、border-bottom-right-image、border-bottom-left-image 不会进行任何的拉伸。

而 border-top-image、border-right-image、border-bottom-image、border-left-image 会分别作为，上边框、右边框、下边框、左边框的这个背景图像来进行显示，必要的时候呢，还可以将这四个图像进行这个平铺或者伸缩。

2、border-image 属性指定边框的宽度

在 css3 中除了可以使用 border 属性和 border-width 属性来指定宽度，我们还可以使用 border-image 属性进行指定边框的宽度。

指定方法：border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距/border 宽度 ；

另外 border 的宽度也可以对四条边进行分别设置

设置的方法：

border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距/border 上宽度 border 右宽度 border 下宽度 border 左宽度;

3、指定四条边的背景的显示方法

在 CSS3 中可以在 border-image 属性中指定元素四条边中的图像，是以拉伸的方式显示，还是平铺的方式进行显示。

指定方法如下：

border-image: url(""边框图像的路径"") 上边距 右边距 下边距 左边距/border 宽度 topbottom leftright;

topbottom 表示元素的上下两条边中图像的显示方法，leftright 表示元素的左右两条边中的显示方式，在显示方法中可以指定的值有 repeat、stretch 和 round。

repeat：将图像以平铺的方式进行显示。

stretch：将图像以拉伸的方式进行显示。

round：将图像进行平铺显示，但是如果最后一幅图不能被完全显示时，就不显示图像，把之前的图像扩大。

七、CSS3 的变形功能

这 css3 中，可以利用 transform 功能来实现文字或图像的旋转、缩放、倾斜、移动这四种类型的变形处理。

目标：

第一、掌握 css3 中 transform 功能的使用方法，能够使用 transform 功能来实现文字或图像的旋转、缩放、倾斜、移动的变形效果。

第二、能够将旋转、缩放、倾斜和移动，这四种变形效果进行结合使用，并且知道如果使用的先后顺序不同，页面显示的结果会是怎样的一个区别。

1、如何使用 transform 功能

在 css3 中通过 transform 属性，来实现 transform 功能。

```
transform:功能;
-ms-transform:功能; /* IE 9 */
-moz-transform:功能; /* Firefox */
-webkit-transform:功能; /* Safari 和 Chrome */
-o-transform:功能; /* Opera */
```

2、rotate 旋转，在参数中规定角度

使用方法：

```
-ms-transform:rotate(角度); /* IE 9 */
-moz-transform:rotate(角度); /* Firefox */
-webkit-transform:rotate(角度); /* Safari 和 Chrome */
-o-transform:rotate(角度); /* Opera */
```

rotate 表示的是顺时针旋转，deg 是 CSS3 中的角度单位。

3、scale 缩放转换

使用方法：

transform:scale(值)，他的值是指定的缩放倍率，比如 0.5 就是缩放到 50%，1 就是 100%，1.5 就是放大点 150%。

可能的值：

1)、scale(x,y)使元素 X 轴和 Y 轴同时缩放

2)、scaleX(x)元素仅 X 轴缩放

3)、scaleY(y)元素仅 Y 轴缩放

4、倾斜 skew

使用方法：transform:skew(值)，他的值是角度。

可能的值：

1)、skew(x,y)使元素在水平和垂直方向同时扭曲（X 轴和 Y 轴同时按一定的角度值进行扭曲变形）只有一个参数时，只在水平方向上倾斜。

2)、skewX(x)仅使元素在水平方向扭曲变形（X 轴扭曲变形）

3)、skewY(y)仅使元素在垂直方向扭曲变形（Y 轴扭曲变形）

5、移动 translate

使用方法：transform:translate(值)，他的值是指定移动的距离。

可能的值：

1)、translate(x,y)水平方向和垂直方向同时移动（也就是 X 轴和 Y 轴同时移动）只有一个参数时，只在水平方向上移动；

2)、translateX(x)仅水平方向移动（X 轴移动）

3)、translateY(Y)仅垂直方向移动（Y 轴移动）

6、对一个元素使用多种变形的方法

使用方法：transform:方法 1 方法 2 方法 3 方法 4；

7、改变元素基点 transform-origin

可能的值 top left top right topleft center right bottom leftbottom bottom right

浏览器兼容情况：

八、CSS3 的动画功能

在 CSS3 中，如果使用动画功能可以使网页上文字或者图像具有动画效果,可以使背景颜色从一种颜色平滑过渡到另外一种颜色,Transition 功能支持从一个属性值平滑到另外一个属性值,Animations 功能支持通过关键帧的指定来在页面上产生更复杂的动画效果.

目标：

1、掌握 CSS3 中 Transition 功能的使用方法，以及能够使用 Transition 功能来实现属性值的开始值与属性得结束值之间的平滑过渡的动画。

2、掌握 CSS3 中 Animations 功能的使用方法，能够在样式中创建多个关键帧，在这些关键帧之中，编写样式，并且能够在页面中创造结合这些关键帧所运行的较为复杂的动画。

1、Transition 功能

css3 中 transition 允许 css 的属性值在一定的时间区间内平滑地过渡。这种效果可以在鼠标单击、获得焦点、被点击或对元素任何改变中触发，并圆滑地以动画效果改变 CSS 的属性值。

transition 属性的使用方法：

transition:语法;

-moz-transition: 语法; /* Firefox 4 */

-webkit-transition:语法; /* Safari 和 Chrome */

-o-transition:语法; /* Opera */

语法：

transition: property duration timing-function delay;

transition 主要包含四个属性值：

1)、执行变换的属性：transition-property，属性规定应用过渡效果的 CSS 属性的名称。（当指定的 CSS 属性改变时，过渡效果将开始）

值有三个类型：

A、none 没有属性会获得过渡效果。

B、all 所有属性都将获得过渡效果。

C、property 定义应用过渡效果的 CSS 属性名称列表，列表以逗号分隔。

2)、变换延续的时间：transition-duration；

规定完成过渡效果需要花费的时间（以秒或毫秒计），默认值 0 没有效果

3)、在延续时间段，变换的速率变化 transition-timing-function；

值：

A、ease：（逐渐变慢）默认值，ease 函数等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0).

B、linear：（匀速），linear 函数等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0).

C、ease-in：(加速)，ease-in 函数等同于贝塞尔曲线(0.42, 0, 1.0, 1.0).

D、ease-out：（减速），ease-out 函数等同于贝塞尔曲线(0, 0, 0.58, 1.0).

E、ease-in-out：（加速然后减速），ease-in-out 函数等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)

F、cubic-bezier(n,n,n,n) 在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值。

4)、变换延迟时间 transition-delay；

transition-delay 是用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行 transition 效果，其取值：<time>为数值，单位为 s（秒）或者 ms(毫秒)

2、Animations 功能

在 CSS3 之中我们除了可以使用 transition 实现动画效果之外呢，我们还可以使用 Animations 来实现动画效果。

1、使用 transition 和 Animations 的区别：

transition 和 Animations 的区别在于，transition 只能通过指定属性的开始值与结束值，然后通过两属性值之间进行平滑过渡的方式来实现动画效果，所以 transition 不能实现复杂的动画效果，而 Animations 功能是是通过关键帧以及每个关键帧中的属性值来实现更为复杂的动画效果。

2、Animations 的使用方法：

@-webkit-keyframes 关键帧合集名称{创建关键帧的代码}

0%~100%{

本关键帧中的样式

}

关键帧创建好了之后，还要在元素的样式中使用该关键帧。

方法如下：

元素{

-webkit-animation-name:关键帧合集名称;

-webkit-animation-duration:5s;

-webkit-animation-timing-function:linear;

-webkit-animation-iteration-count:infinite;

}

-webkit-animation-name 指定合集名称，-webkit-animation-duration 整个动画执行完成所需的时间、需要的时间，-webkit-animation-timing-function 实现动画的方法,-webkit-animation-iteration-count 属性的属性值设定为某个整数值，那么这个动

画播放的次数就等于这个整数值（infinite 是无限循环播放）。

3、实现动画的方法：

A、linear：从开始到结束都是以同样的速度进行.

B、ease-in：开始速度很慢，然后沿着曲线进行加快.

C、ease-out：开始速度很快，然后沿着曲线进行减速.

D、ease：开始时速度很快，然后沿着曲线进行减速，然后再沿着曲线加速.

E、ease-in-out：开始时速度很慢，然后沿着曲线进行加速，然后再沿着曲线减速.

九、布局相关样式

什么是网页的布局？网页布局呢是指在页面中如何对标题、导航栏、主要内容、页脚、表单等各种构成要素进行一个合理的编排。在使用啊 CSS3 之前呢,主要使用 float 属性或 position 属性进行页面的简单布局，但是呢他们也有他们的缺点，比如在两栏或者多栏布局的时候，会出现元素的内容高度不一致，那么导致他们的底部很难对齐，在 CSS3 中追加了一些新的布局方式，使用这些新的布局方式，除了可以修改之前存在的问题之外，还可以进行更为便捷更为复杂的页面布局。

目标：

第一点、掌握 CSS3 中多栏布局的使用方法，知道为什么要想多栏布局。

第二点、掌握 CSS3 中盒布局的方法，知道为什么要使用盒布局。

第三点、掌握 CSS 中弹性盒布局的基本概念及使用方法。

1、多栏布局

1、column-count 属性

在 CSS3 中可以通过，column-count 属性来进行多栏布局，这个属性的含义是将一个元素中的内容分成多栏进行显示。

写法：

column-count: 栏目数;

兼容性写法：

-webkit-column-count:3;

-moz-column-count:3;

需要注意的是，在使用多栏布局的时候，要将元素的宽度设置成多个栏目的总宽度。

2、column-width 属性：指定栏目的宽度来生成分栏。

兼容性写法： -webkit-column-width、-moz-column-width

3、column-gap 属性：指定栏目与栏目之间的距离

兼容性写法： -webkit-column-gap、-moz-column-gap

4、column-rule 属性：栏目与栏目之间增加一条分割线

兼容性写法： -webkit-column-rule、-moz-column-rule

2、盒布局

盒布局的基础知识

1、使用 float 属性和 position 属性时的缺点

在 CSS3 中，除了多栏布局之外，还可以使用盒布局来解决使用 float 属性和 position 属性时多栏底部不能对齐的缺点。

首先来实现一个 float 属性布局的案例，并且找到缺点。

2、使用盒布局

在CSS3中,使用box属性来使用盒布局,在火狐中要把box写成“-moz-box”的形式,在谷歌浏览器中呢需要写成“-webkit-box”的形式。

综合上一个案例，将他修改成盒布局。

3、盒布局与多栏布局的区别

多栏布局的栏目宽度必须相等，指定栏目的宽度的时候也只能统一指定，栏目的宽度不可能全都一样，所以多栏布局比较

适合在文字内容页面使用，并不适合整个网页编排时使用。

3、弹性盒布局

1、使用自适应窗口的弹性盒布局

如何才能让 DIV 的宽度跟随浏览器窗口变化而变换呢？在 CSS3 中我们只要使用一个 box-flex 属性，使得我们的盒布局变成弹性盒布局就可以了。

兼容性写法：

-webkit-box-flex (Safari 浏览器、Chrome 浏览器时前面加-webkit-)

-moz-box-flex (Firefox 浏览器时前面加-moz-)

2、改变元素的显示顺序

使用弹性盒布局的时候，可以通过 box-ordinal-group 属性来改变各个元素的显示顺序，在每个元素中加入 box-ordinal-group 属性，这个属性使用一个表示序号的正数属性值，浏览器在显示的时候根据序号从小到大来显示这些元素。

兼容性写法：

-webkit-box-ordinal-group (Safari 浏览器、Chrome 浏览器时前面加-webkit-)

-moz-box-ordinal-group (Firefox 浏览器时前面加-moz-)

3、改变元素的排列方向

在使用弹性盒布局的时候，可以通过 box-orient 来指定多个元素的排列方向。

值：

horizontal 在水平行中从左向右排列子元素。

vertical 从上向下垂直排列子元素。

兼容性写法：

-webkit-box-orient:vertical (Safari 浏览器、Chrome 浏览器时前面加-webkit-)

-moz-box-orient:vertical (Firefox 浏览器时前面加-moz-)

4、元素的宽度与高度自适应

在使用盒布局的时候，元素的宽度与高度具有自适应性，就是元素的宽度与高度可以根据排列方向的改变而改变。

5、使用弹性盒布局来消除空白

方法就是给子 DIV 中加入一个 box-flex 属性

6、对多个元素使用 box-flex 属性

让浏览器或者容器中的元素的总宽度或者是总高度都等于浏览器或者是容器的高度。

方法对多个元素使用 box-flex 属性

7、指定水平方向与垂直方向的对齐方式

使用盒布局的时候，可以使用 box-pack 属性及 box-align 属性来指定元素中的文字、图像、以及子元素的水平方向或者是垂直方向上的对齐方式。

兼容性写法：

-webkit-box-pack:值 (Safari 浏览器、Chrome 浏览器时前面加-webkit-)

-moz-box-pack:值 (Firefox 浏览器时前面加-moz-)

-webkit-box-align:值 (Safari 浏览器、Chrome 浏览器时前面加-webkit-)

-moz-box-align:值 (Firefox 浏览器时前面加-moz-)

属性值 box-pack 属性值的含义 box-align 属性值的含义 start 左对齐，文字、图像或子元素被放置在元素最左边顶部对齐，文字.图像或子元素被放置在元素最顶部 center 中对齐，文字、图像或子元素被放置在元素中部中部对齐，文字、图像或子元素被放置在元素中部 end 右对齐，文字、图像或子元素被放置在元素最右边底部对齐，文字、图像或子元素被放置在元素最底部 start 顶部对齐，文字、图像或子元素被放置在元素最顶部左对齐，文字、图像或子元素被放置在元素最左边 center 中部对齐，

文字、图像或子元素被放置在元素最中部中对齐，文字、图像或子元素被放置在元素最中部 end 底部对齐，文字、图像或子元素被放置在元素最底部右对齐，文字、图像或子元素被放置在元素最右边

十、不同浏览器窗口大小加载不同 CSS 样式

1、Media Queries

Media Queries 模块 CSS3 中的一个和各种媒体相关的重要模块。

目标：

目标一：掌握 CSS3 中的 Media Queries 模块的基本概念，以及使用 Media Queries 模块 可以实现的功能。

目标二:掌握如何编写媒体表达式来让浏览器根据当前窗口尺寸自动在样式表中挑选一种样式并使用。了解 iPhone 和 iPad touch 设备在支持 Media Queries 时，有何特殊之处，以及应该用何种方法来制定 iPhone 或者 iPad touch 设备中的 Safari 浏览器在处理页面是根据多少像素的窗口来处理。

目标三：掌握媒体查询表达式编写方法，掌握 css3 中定义所有设备类型及设备特性、媒体查询表达式中各种关键字的含义，以及结合使用设备类型设备特征和各种关键字来正确的编写媒体查询表达式。

2、使用方法

Media Queries 的使用方法：

@media 设备类型 and （设备特征）{样式代码}

在样式的代码开头必须要写@media，然后制定设备的类型（媒体类型）

可以指定的值与其所代表的设备类型 all 所有设备 screen 电脑显示器 print 打印用纸或者打印预览图 handheld 便携设备 tv 电视机类型设备 speech 语音和音频合成器 braille 盲人用点字法触觉回馈设备 embossed 盲文打印机 projection 各种投影设备 tty 使用固定密度字母栅格的媒介，比如电传打字机和终端。

设备特性的书写方式与样式的书写方式呢很相似，分为两个部分，当中有冒号分隔，冒号前书写设备的某种特性，冒号后书写该特性的具体值。例如，需要指定浏览器的窗口宽度大于 400px 时,我们可以这样书写(min-width:400px),大部分设备特性的指定值接受 min/max 的前缀,用来表示大于等于或小于等于的逻辑,以此来避免使用<和>这些字符。13 种设备的特性说明特效可指定的值是否允许使用 min/max 前缀特性说明 width 带单位的长度数值允许浏览器的窗口宽度 height 带单位的长度数值允许浏览器窗口的高度 device-width 带单位的长度数值允许设备屏幕的分辨值 device-height 带单位的长度数值允许设备屏幕分辨率的最高值 orientation 只能指定 2 个值 padding 或 landscape 不允许浏览器的方向是纵向还是横向，当浏览器的高度值大于等于浏览器的宽度值的时候，这个特性的 portrait 否则为 landscapeaspect-ratio 比例值允许浏览器窗口的纵横比，比例值为浏览器窗口的宽度值/高度值 device-aspect-ratio 比例值允许屏幕分辨率的纵横比，比例值为设备屏幕分辨率的宽度值/高度值 color 整数值允许设备使用多少位的颜色值，如果不是彩色设备，该值为 0color-index 整数值允许色彩表中的色彩数 monochrome 整数值允许单色帧缓冲器中每像素的字节数 resolution 分辨率值允许设备的分辨率 scan 只能指定两个值 progressive 或 interlace 不允许电视机类型设备的扫描方式。progressive 表示逐行扫描，interlace 表示隔行扫描 grid 只能指定两个值 0 或 1 不允许设备是基于栅格还是基于位图。基于栅格时该值为 1，否则该值为 0

使用 and 关键字来指定某种设备类型的某种特性的值满足某个条件时所使用样式， 比如以下语句指定了，当设备窗口宽度小于 640px 时所使用的样式。

```
@media screen and (max-width：639px){
 样式代码
}
```

可以使用多条语句来将同一个样式应用于不同的设备类型的设备特性中，指定方式如下：

@media handheld and (min-width：360px),screen and (min-width：480px){

样式代码

}

可以在样式中加入 not 关键字和 only 关键字，not 表示去反，only 是让那些不支持@Media Queries 但是可以读取 Media Type 的设备的浏览器，将表达式中的样式隐藏起来。

not 示例：

@media not handheld and (color){样式代码}

/*样式将被使用在除开便携设备和非色彩携带设备的所有设备中*/

@media all and （not color）

/*样式将被使用在所有非彩色设备中*/

其他的引入方式：

<link rel=""stylesheet"" media=""screen and (max-width: 600px)"" href=""600.css"" />

/*当页页宽度小于或等于 600px,调用 600.css 样式表来渲染你的 Web 页面。*/

或者：

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

@import url(""css/style.css"");

</style>

第五部分 JAVASCRIPT

一、JavaScript 简介

1、JavaScript 简介

JavaScript 是 NetScape 公司为 Navigator 浏览器开发的，是卸载 HTML 文件中的一种脚本语言，能实现网页内容的交互显示。当用户在客户端显示该网页时，浏览器就会执行 JavaScript 程序，用户通过交互的操作来改变网页的内容，来实现 HTML 语言无法实现的效果。

2、如何使用 JavaScript

通过<script></script>中直接编写

通过<script src='目标文档的 URL'></script>链接外部的 Js 文件

作为某个元素的事件属性值或者是超链接的 href 属性值

3、代码屏蔽

<script type='text/javascript'>

<!--

Js 代码;

//-->

</script>

如果浏览器不支持 Js，可以使用<noscript></noscript>标签，显示 noscript 中的内容

4、JavaScript 的基本语法

JavaScript 的执行顺序：按照在 HTML 文件中出现的顺序依次执行

如果需要在 HTML 文件执行函数或者全局变量，最好将其放在 HTML 的头部中。

大小写敏感：JavaScript 严格区分大小写

忽略空白符和换行符：JavaScript 会忽略关键字、变量名、数字、函数名或其它各种元素之间的空格、制表符或换行符

我们可以使用缩进、换行来使代码整齐，提高可读性

语句分隔符：使用;结束语句

可以把多个语句写在一行：最后一个语句的分号可以省略，但尽量不要省略

可以使用{}括成一个语句组，形成一个块 block：通过\对代码进行折行操作

```
document.write(' hello\
 world');
```

注释：

单行注释：//

多行注释：/*注释内容*/

JavaScript 的保留字：

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

通过 document.write()向文档书写内容

通过 console.log()向控制台写入内容

avaScript 中的错误：

语法错误：通过控制台进行调试

逻辑错误：通过 alert()进行调试

二、Javascript 数据类型与变量

1、原始数据类型

1.数值型

JavaScript 中的数值包含整数和浮点数，所有数值都以双精度浮点型来表示。

双精度浮点数可以表示-2 的 53 次方到 2 的 53 次方的整数，也可以表示为正负 1.7976 的 10 的 308 次方的最大值和正负 2.2250 乘以 10 的-308 次方的浮点数

十进制数：12、1.2、-23、.222e33、-1.3e3、3.E-2、12e+20

十六进制：0x0、0XABCDEF、0x1a2b3c4d

八进制数：00、0123、0241234

特殊值：Infinity 无穷大、NaN

当一个数值或数值表达式的值超出了可表示的最大值的范围，将被赋值为 Infinity。可以有无穷大 Infinity，也可以有无穷小 -Infinity。

Infinity 值：1.79e309、-1.79e309。

NaN 代表 Not a Number。当一个 Undefined 表达式的结果为数值型数据时，该数值型就是 NaN 值。

NaN 是唯一一个不能和自身做比较的值

NaN 值：0/0

可以通过 isNaN()检测值是否为 NaN

2.字符串型

定界符：""|''

转义符

\n->回车换行

\r->换行

\t->水平制表符

\""->""

\'->'

2.布尔类型：true|false

2、复合数据类型

对象(object)、数组(array)、函数(function)

3、特殊数据类型

1.无定义数据类型 undefined

undefined 用来表示不存在的值或者尚未赋值的变量。对一个变量只声明不赋值或者赋予一个不存在的属性值，都会使这个变量的值为 Undefined

2.空值 null

null 表示空值，表示什么都没有，相当于一个占位符。

null 和 undefined 的区别就是 undefined 表示变量未被赋值，而 null 表示变量被赋予了一个空值。

4、变量

声明变量：通过 var 关键字声明变量

可以声明变量的同时给变量赋值

可以一次声明一个变量也可以一次声明多个看变量

如果只声明变量未对其赋值，默认值为 undefined

如果变量重名产生覆盖

注意：

变量严格区分大小写

变量名称不要包含特殊字符

变量名称最好遵循驼峰标记法或者下划线法

变量名称最好含义明确

变量在内存中的存储与释放：收集方式、收集内容、回收算法

5、类型转换

1.隐式转换

①转换成布尔类型：

undefined->false

null->false

数值型 0 或 0.0 或 NaN->false

字符串长度为 0->false

其它对象->true

②转换为数值型数据：

undefined->NaN

null->0

true->1|false->0

内容为数字->数字，否则转换成 NaN

其它对象->NaN

③转换为字符串型数据

undefined->""undefined""

null->""NaN""

true->""true"" false->""false""

数值型->NaN、0 或者与数值对应的字符串

其它对象->如果存在这个对象则转换为 toString()方法的值，否则转换为 Undefined

2.显示转换

①转换成数值：

Number 函数强制转换成数值

数值->转换成原来的值

字符串->如果可以解析为数值，则转换成数值；否则转换成 NaN 或者 0

true->1,false->0

undefined->NaN

null->0

转换成整型：praseInt()

转换成浮点型：praseFloat()

注意： Number 函数将字符串转换为数值比 praseInt 函数严格很多。基本上只要有一个字符无法转换成数值，整个字符串就会被转换成 NaN

②转换成字符串

通过 String 函数转换成字符串

数值->数值本身

字符串->字符串本身

true->""true"",false->""false""

undefined->""undefined""

null->""null""

转换成字符串型：toString()

③转换成布尔类型：通过 Boolean 函数强制转换成布尔值

0、-0->false

NaN->false

空字符串->false

undefined->false

null->false

三、Javascript 运算符与表达式

1、表达式

表达式是用于 JavaScript 脚本运行时进行计算的式子，可以包含常量、变量、运算符

2、运算符

1.算术运算符

+、-、*、/、%、++、--

++、--分为前缀形式和后缀形式：前缀形式先加减 1 在执行、后缀形式先执行再加减 1

注意：

+号用来连接两个字符串

只要+连接的操作数中有一个是字符串型，Js 就会自动把非字符串型数据作为字符串型数据来处理

Js 代码的执行顺序是从左到右，所以在+连接的表达式中，遇到字符串型数据之前，所有出现的数值型数据(或者可以自动转换为数值型的数据)仍被作为数值来处理。为了避免这种情况，我们可以在表达式前拼一个空字符串

2.字符连接符

通过+连接字符串

3.赋值运算符

=、+=、-=、*=、/=、%=、.=

4.比较运算符

>、>=、<、<=、==、!=、===、!==

注意：

比较运算符的结果为布尔类型

==只比较值是否相等、===比较值和类型

5.逻辑运算符

&&、||、！

注意：

逻辑运算符的结果为布尔类型

&&如果第一个表达式为 false，造成短路

||如果第一个表达式为 true,造成短路

6.三元运算符

exp1?exp2:exp3

7.其它运算符

逗号运算符：逗号用来将多个表达式连接为一个表达式，新表达式的值为最后一个表达式的值，多用在变量声明处

void 运算符：void 运算符用来指明一个表达式无返回结果

typeof 运算符：typeof 运算符用来返回一个字符串，返回的是操作数的数据类型

3、运算符的优先级

运算符	描述
. [] ()	字段访问、数组下标、函数调用以及表达式分组
++ -- - ~ ! delete new typeof void	一元运算符、返回数据类型、对象创建、未定义值
* / %	乘法、除法、取模
+ - +	加法、减法、字符串连接
<< >> >>>	移位
< <= > >= instanceof	小于、小于等于、大于、大于等于、instanceof
== != === !==	等于、不等于、严格相等、非严格相等
&	按位与
^	按位异或
|	按位或
&&	逻辑与
||	逻辑或
?:	条件
= oP=	赋值、运算赋值
,	多重求值

通过()改变优先级

四、Javascript 流程控制

1、条件语句

if(exp)执行一句代码

if(exp){执行代码段;}

if(exp){exp 为 true 执行代码段}else{exp 为 false 执行代码段;}

if...elseif...

switch...case

if 嵌套

2、循环语句

for 循环

while 循环

do...while 循环

3、特殊循环控制

break 终止循环

continue 跳过当次循环，进入下次循环

五、Javascript 函数

1、什么是函数?

函数是完成某一功能的代码段

函数是可重复执行的代码段

函数方便管理和维护

2、自定义函数

1.通过 function 关键字

function 函数名称([参数,...]){

代码段;

return 返回值;

}

注意：

函数名称不要包含特殊字符

函数名称最好含义明确

函数名称最好遵循驼峰标记法或者下划线法

函数名称严格区分大小写

函数名称如果重复会产生覆盖

函数可以有参数也可以没有参数，可以有一个参数也可以有多个参数

函数通过 return 加返回值，如果没有 return 默认返回 undefined

函数不调用不执行

2.匿名函数

函数表达式可以存储在变量中，变量也可以作为一个函数使用

可以将匿名函数作为参数传递给其它函数，接收方函数就可以通过所传递进来的函数完成某些功能

可以通过匿名函数来执行某些一次性的任务

3.通过 Function()构造函数

通过内置 JavaScript 函数构造器(Function())定义

var myFunction=new Function('a','b','return a+b');

var myFunction=function(a,b){return a+b;};

注意：以上两种方式是等价的、尽量避免使用 new 关键字。

3、调用函数

1.作为一个函数调用

通过函数名称()进行调用，如果有参数传递相应参数即可

在 HTML 中默认的全局对象是 HTML 页面本身，所以函数是属于 HTML 页面。在浏览器中的页面对象是浏览器窗口(window 对象)，所以函数会自动变为 window 对象的函数。也可以通过 window.函数名称()进行调用。

2.全局对象

当函数没有被自身的对象调用时，this 的值就会变成全局对象。在 web 浏览器中全局对象是浏览器窗口 window 对。

函数作为全局对象调用，会使 this 的值称为全局对象。使用 window 对象作为一个变量容易造成程序崩溃

3.函数作为方法调用

可以将函数定义为对象的方法进行调用

4.使用构造函数调用函数

如果在函数调用前使用了 new 关键字，则调用了构造函数

5.作为回调函数调用函数

call()、apply()

4、参数

函数可以有参数也可以没有参数，如果定义了参数，在调用函数的时候没有传值，默认设置为 undefined

在调用函数时如果传递参数超过了定义时参数，js 会忽略掉多余参数

js 中不能直接写默认值，可以通过 arguments 对象来实现默认值效果

可以通过 arguments 对象实现可变参数的函数

通过值传递参数在函数体内对变量做修改不会影响变量本身

通过对象传递参数在函数体内对变量做更改会影响变量本身

5、变量的作用域

局部变量：在函数体内声明的变量，仅在函数体内可以使用

全局变量：函数体外声明的变量，在变量声明开始到脚本结束都可以是使用

注意：尽量控制全局变量的数量容易引发 bug、最好总是使用 var 语句来声明变量。

6、JavaScript 全局函数

1.parseInt(string,radix)

返回转换成整数的值

注意：

当参数 radix 的值为 0，或者没有设置这个参数，parseInt()会根据 string 来判断数字的基数。

当忽略 radix，JavaScript 默认数字的基数规则为：

如果 string 以 0x 开头，parseInt()会把 string 的其余部分解析为十六进制的整数

如果 string 以 0 开头，那么 ECMAScript v3 允许 parseInt()的一个实现把其后的字符解析为八进制或十六进制的数字

如果 string 以 1~9 的数字开头，parseInt()将把它解析为十进制的整数

字符串如果以合法字符开始，截取合法字符

开头和结尾的空格是允许的

如果字符串的第一个字符不能被转换为数字，parseInt()会返回 NaN

在字符串以""0""为开始时旧的浏览器默认使用八进制基数。ECMAScript 5，默认的是十进制的基数。

2.parseFloat(string)

返回转换成浮点型的值

该函数指定字符串中的首个字符是否是数字。如果是，则对字符串进行解析，直到到达数字的末端为止，然后以数字返回该数字，而不是作为字符串。

3.isFinite(value)

检测某个是是否是无穷值

如果 number 是 NaN 或者+Infinity 或者-Infinity 的数，isFinite()返回 false

4.isNaN(value)

检测某个值是否是 NaN

isNaN()检测是否是非数字值，如果值为 NaN 返回 true，否则返回 false

5.encodeURI(uri)

将字符串编码为 URI

注意：

使用 decodeURI() 方法可以编码 URI（通用资源标识符:UniformResourceIdentifier,简称""URI""）。

对以下在 URI 中具有特殊含义的 ASCII 标点符号，encodeURI() 函数是不会进行转义的： , / ? : @ & = + $ # (可以使用 encodeURIComponent() 方法分别对特殊含义的 ASCII 标点符号进行编码。).

6.decodeURI(uri)

解码某个个编码的 URI

7.encodeURIComponent(uri)

将字符串编码为 URI 组件

注意：

该方法不会对 ASCII 字母和数字进行编码，也不会对这些 ASCII 标点符号进行编码： - _ . ! ~ * ' ( )

其他字符（比如 ：;/?:@&=+$,# 这些用于分隔 URI 组件的标点符号），都是由一个或多个十六进制的转义序列替换的。

8.decodeURIComponent()

解码一个编码的 URI 组件

9.escape()

对字符串进行编码

注意：

escape() 函数可对字符串进行编码，这样就可以在所有的计算机上读取该字符串。

该方法不会对 ASCII 字母和数字进行编码，也不会对下面这些 ASCII 标点符号进行编码： * @ - _ + . / 。其他所有的字符都会被转义序列替换。

escape()函数不能用于编码 URIs(通用资源标识符（UniformResourceIdentifier,简称""URI""))

10.unescape()

解码由 escape()编码的字符串

11.eval()

将 JavaScript 字符串当作脚本来执行

注意：

如果参数是一个表达式，eval() 函数将执行表达式。如果参数是 Javascript 语句，eval()将执行 Javascript 语句。

eval()函数是一种由函数执行的动态代码，比直接执行脚本慢很多

慎重使用 eval()函数,尽量不用，保证程序的安全性

12.Number(object)

把对象的值转换为数字

注意：

如果对象的值无法转换为数字，Number()函数返回 NaN

如果参数的 Date 对象，Number()返回从 1970 年 1 月 1 日到现在所经历的毫秒数

13.String()

把对象的值转换为字符串

String() 函数返回与字符串对象的 toString()方法值一样

7、特殊形式的函数

函数也是数据

JavaScript 中的函数也是一种数据类型，只不过这种特殊类型有两个重要的特性

他们所包含的是代码、他们是可以执行的

六、Javascript 对象

1、JavaScript 对象

Javascript 对象是基本数据类型之一，是复合类型

Javascript 中几乎所有事物都是做对象

Javascript 的对象是拥有属性和方法的数据。

JavaScript 中的对象可以简单理解成""名称:值""对(name:value)

JavaScript 中的对象与这些概念类似：

Python 中的字典

Perl 和 Ruby 中的散列（哈希）

C/C++ 中的散列表

Java 中的 HashMap

PHP 中的关联数组

名称："名称"部分是一个 JavaScript 字符串

注意：

可以将属性名括在引号之间

①这三种形式一样

```
var obj={prop:1}
var obj={""prop"":1}
```

var obj={'prop':1}

②必须放到引号之间

如果属性名是 Javascript 的保留字之一

如果属性名种包含特殊字符(除字母、数字、下划线以外的字符)

如果属性名以数字开头

在 ECMAScript5 中，保留字可以用作不带引号的属性名，但对于 ECMAScript3 中必须用引号括起来

在 ECMAScript5 中对象直接量中的最后一个属性后的逗号将被忽略，在 ECMAScript 3 的大部分实现中也可以忽略这个逗号，但在 IE 中报错

值："值"部分可以是任何 JavaScript 的数据类型——包括对象

2、JavaScript 中得对象分类

1.内建对象

①数据封装对象

Object 对象

定义：Object 是 Javascript 中所有对象的父级对象，我们创建的所有对象都继承于此，包括内建对象

语法： new Object([value])、{name:value,name:value...}

描述： Object 构造函数为给定的值创建一个对象包装。如果给定值是 null or undefined，将会创建并返回一个空对象，否则，将返回一个与给定值对应类型的对象。

当以非构造函数形式被调用时，Object 等同于 new Object()。

属性：Object.prototype：可以为所有 Object 类型的对象添加属性

方法：

Object.create():指定原型对象和属性创建一个对象。

语法：Object.create(proto,[propertiesObject])

参数：

proto:一个对象，作为新创建对象的原型

propertiesObject:一个对象值，可以包含若干个属性，属性名称为新建对象的属性名，属性值为那个属性的属性描述对象。

Object.defineProperty():给对象添加/修改一个属性并指定该属性的配置

语法：Object.defineProperty(obj,prop,descriptor)

参数：

obj：需要定义的对象

prop：需要定义或修改的属性名

descriptor：属性定义或修改的属性的描述

描述：

该方法允许精确添加或修改对象的属性。正常的属性添加通过赋值来创建并显示在属性枚举中（for...in 循环 或 Object.keys 方法）， 这种方式添加的属性值可能被改变，也可能会被删除。该方法允许改变这些额外细节的默认设置。

对象里目前存在的属性描述符有两种主要形式：数据描述符和存取描述符。数据描述符是一个拥有可写或不可写值的属性。存取描述符是由一对 getter-setter 函数功能来描述的属性。描述符必须是两种形式之一。

数据描述符和存取描述符均具有以下可选键值：

configureable：当且仅当这个属性描述符值为 true 时，该属性可能会改变，也可能会被从相应的对象删除。默认为 false。

enumerable：true 当且仅当该属性出现在相应的对象枚举属性中。默认为 false。

value：与属性有关的值。可以是任何有效的 Javascript 值。默认为 undefined

writable：true 当且仅当可能用赋值运算符改变与属性相关的值。默认为 false

存取描述同时具有以下可选键值：

get：一个给属性提供 getter 的方法，如果没有 getter 则为 undefined。方法将返回作用属性的值，默认为 undefined

set：一个给属性提供 setter 的方法，如果没有 setter 则为 undefined。该方法将受到作为唯一参数的新值分配给属性。默认为 undefined

注意：这些选项不一定是自身属性，如果是继承来的也要考虑。为了确认保留这些默认值，你可能要在这之前冻结 Object.prototype，明确指定所有的选项，或者将 __proto__ 属性指向空。

	configurable:true writable:true	configurable:true writable:false	configurable:false writable:true	configurable:false writable:false
修改属性的值	√	√* 重设value标签修改	√	×
通过属性赋值修改属性的值	√	×	√	×
delete该属性返回true	√	√	×	×
修改getter/setter方法	√	√	×	×
修改属性标签* (除了writable从true修改为false总是允许)	√	√	×	×

Object.defineProperties():在一个对象上添加或修改一个或者多个自有属性，并返回该对象。

语法：Object.defineProperities(obj,props)

参数：

obj:将要被添加属性或修改属性的对象

props:该对象的一个或多个键值对定义了将要为对象添加或修改的属性的具体配置

Object.keys():方法会返回一个由给定对象的所有可枚举自身属性的属性名组成的数组，数组中属性名的排列顺序和使用 for-in 循环遍历该对象时返回的顺序一致(两者的主要区别是 for-in 还会遍历除一个对象从其原型链上继承到得可枚举的属性)。

语法：Object.keys(obj)

参数：返回该对象的所有可枚举自身属性的属性名

描述：

Object.keys 返回一个所有元素为字符串的数组，其元素来自于从给定的对象上面可直接枚举的属性。

这些属性的顺序与手动遍历该对象属性时的一致。

如果你想获取一个对象的所有属性,，甚至包括不可枚举的，可以通过 Object.getOwnPropertyNames()实现。

Object.getOwnPropertyNames():返回一个由指定对象的所有自身属性的属性名（包括不可枚举属性）组成的数组。

语法： Object.getOwnPropertyNames(obj)

参数：obj:要查看的对象

描述：

Object.getOwnPropertyNames 返回一个数组，该数组对元素是 obj 自身拥有的枚举或不可枚举

属性名称字符串。

数组中枚举属性的顺序与通过 for...in loop（或 Object.keys)）迭代该对象属性时一致。 数组中不可枚举属性的顺序未定义。

Object.getOwnPropertyDescriptor()：返回指定对象上一个自有属性对应的属性描述符。(自有属性指的是直接赋予该对象的属性，不需要从原型链上进行查找的属性）)

语法： Object.getOwnPropertyDescriptor(obj,prop)

参数：

obj:在该对象上查看属性

prop:一个属性名称，该属性的属性描述被返回

返回值：如果指定的属性存在于对象上，则返回其属性描述符（property descriptor），否则返回 undefined

描述：该方法允许对一个属性的描述进行检索。在 Javascript 中， 属性 由一个字符串类型的“名字”（name）和一个“属性描述符”（property descriptor）对象构成

一个属性描述符是一个记录

Object.getPrototypeOf():返回指定对象的原型(也就是该对象内部属性[[Prototype]]的值)

语法：Object.getPrototypeOf(obj)

参数：要返回的对象

描述：如果参数不是一个对象类型，将跑出 TypeError 异常

Object.freeze():冻结一个对象。冻结对象是指那些不能添加新的属性，不能修改已有属性的值，不能删除已有属性，以及不能修改已有属性的可枚举性、可配置性、可写性的对象。也就是说这个对象永远不能改变的。

语法： Object.freeze(obj)

参数：obj:要被冻结的对象

描述：

冻结对象的所有自身属性都不可能以任何方式被修改。任何尝试修改该对象的操作都会失败，可能是静默失败，也可能会抛出异常（严格模式中）

数据属性的值不可更改，访问器属性（有 getter 和 setter）也同样（但由于是函数调用，给人的错觉是还是可以修改这个属性）。如果一个属性的值是个对象，则这个对象中的属性是可以修改的，除非它也是个冻结对象。

Object.isFrozen():判断对象是否已经被冻结

语法： Object.isFrozen(obj)

参数： obj:被检测的对象

描述：

一个对象是冻结的（frozen）是指它不可扩展，所有属性都是不可配置的（non-configurable），且所有数据属性（data properties）都是不可写的（non-writable）

数据属性是值那些没有取值器（getter）或赋值器（setter）的属性。

Object.preventExtensions():阻止对象扩展

语法： Object.preventExtensions(obj)

参数：obj:将要变得不可扩展的对象

描述：

如果一个对象可以添加新的属性，则这个对象是可扩展的。preventExtensions 可以让这个对象变的不可扩展，也就是不能再有新的属性。

需要注意的是不可扩展的对象的属性通常仍然可以被删除

尝试给一个不可扩展对象添加新属性的操作将会失败，不过可能是静默失败，也可能会抛出 TypeError 异常（严格模式）。

Object.isExtensible():检测一个对象是否可扩展(是否可以在它上面添加新的属性)

语法：Object.isExtensible(obj)

参数：obj：需要检测的对象

描述：默认情况下，对象是可扩展的：即可以为他们添加新的属性。以及它们的 __proto__ 属性可以被更改。

Object.preventExtensions，Object.seal 或 Object.freeze 方法都可以标记一个对象为不可扩展（non-extensible）。

Object.seal():可以让一个对象密封，并返回被密封之后的对象。密封对象是指那些不能添加新的属性、不能删除已有属性，以及不能修改已有属性的可枚举性、可配置性、可写性，但可能可以修改已有属性的值的对象

语法：Object.seal(obj)

参数： obj:要被密封的对象

描述：

通常情况下，一个对象是可扩展的（可以添加新的属性）

密封一个对象会让这个对象变的不能添加新属性，且所有已有属性会变的不可配置。

属性不可配置的效果就是属性变的不可删除，以及一个数据属性不能被重新定义成为访问器属性，或者反之。

但属性的值仍然可以修改。尝试删除一个密封对象的属性或者将某个密封对象的属性从数据属性转换成访问器属性，结果会静默失败或抛出 TypeError 异常（严格模式）。

不会影响从原型链上继承的属性。但 __proto__ ( ) 属性的值也会不能修改。

Object.isSealed():检测一个对象是否被密封 sealed

语法：Object.isSealed(obj)

参数：obj:要被检测的对象

描述：

如果这个对象是密封的，则返回 true，否则返回 false。

密封对象是指那些不可 扩展 的，且所有自身属性都不可配置的（non-configurable）对象。

Object.prototype

JavaScript 语言的所有对象都是由 Object 衍生的对象

所有对象都继承了 Object.prototype 的方法和属性，尽管它们可能被覆盖

属性： Object.prototype.constructor:返回一个指向创建了该对象原型的函数引用

注意：该属性的值是那个函数本身，而不是一个包含函数名称的字符串。对于原始值（如 1，true 或 ""test""），该属性为只读。

所有对象都会从它的原型上继承一个 constructor 属性

方法：

Object.prototype.hasOwnProperty():检测某个对象是否含有指定的自身属性

语法：obj.hasOwnProperty(prop)

参数：要检测的属性名称

描述：

所有继承了 Object.prototype 的对象都会从原型链上继承到 hasOwnProperty 方法

这个方法可以用来检测一个对象是否含有特定的自身属性，和 in 运算符不同，该方法会忽略掉那些从原型链上继承到的属性

Object.prototype.isPrototypeOf():检测一个对象是否存在于另一个对象的原型链上

语法：prototype.isPrototypeOf(object)

参数：

prototype:检测该对象是否在参数 object 的原型链上

object:在该对象的原型链上搜寻

描述： isPrototypeOf 方法允许你检测一个对象是否存在于另一个对象的原型链上

Object.prototype.propertyIsEnumerable():检测指定的属性名是否是当前对象可枚举的自身属性

语法：obj.propertyIsEnumerable(prop)

参数：prop:需要检测的属性名

描述：

每个对象都有 propertyIsEnumerable 方法。该方法可以判断出指定的属性是否是自身的可枚举属性，也就是说该属性是否可以通过 for...in 循环等遍历到

有些属性虽然可以通过 for...in 循环遍历到，但因为它们不是自身属性，而是从原型链上继承的属性,所以该方法也会返回 false。如果对象没有指定的属性，该方法返回 false。

Object.prototype.toString():返回一个代表该对象的字符串

语法：object.toString()

描述：

当对象需要转换为字符串时，会调用它的 toString()方法.

默认情况下，每个对象都会从 Object 上继承到 toString()方法，如果这个方法没有被这个对象自身或者更接近的上层原型上的同名方法覆盖(遮蔽)，则调用该对象的 toString()方法时会返回""[object type]""，这里的字符串 type 表示了一个对象类型

Object.prototype.valueOf():返回的诗 this 值，即对象本身

语法 object.valueOf()

返回值：在其他类型的对象中，valueOf 有可能返回一个不同的值

Number 对象

定义：Number JavaScript 对象是一个允许你处理数字值的包装对象。Number 对象使用 Number() 构造器创建。

语法：new Number(value)

描述：

如果参数无法被转换为数字，则返回 NaN。

在非构造器上下文中 (如：没有 new 操作符)，Number 能被用来执行类型转换。

属性：

MAX_VALUE：最大的正数

MIN_VALUE：最小的正数

NaN：特殊的非数字值

NEGATIVE_INFINITY：特殊的负无穷大值，在溢出时返回

POSITIVE_INFINITY：特殊的正无穷大值，在溢出时返回

prototype：Number 对象上允许的额外属性

方法：

toFixed():返回一个字符串，以定点数的形式来表示某一个数字，并进行四舍五入

toExponential():返回一个字符串，以指数形式来表示某一数字

toPrecision():返回一个字符串，既可以是指数型，也可以是小数型

Boolean 对象

定义：Boolean 对象是一个包装了布尔值的对象

语法：new Boolean([value])

描述：

如果 Boolean 构造函数的参数不是一个布尔值,则该参数会被转换成一个布尔值.

如果参数是 0, -0, null, false, NaN, undefined, 或者空字符串 (""""),生成的 Boolean 对象的值为 false

其他任何值,包括任何对象或者字符串""false"", 都会创建一个值为 true 的 Boolean 对象.

注意：

原始的布尔值和 Boolean 对象是不一样的

任何值为 undefined 或者 null 的对象，包括值为 false 的 Boolean 对象，在条件语句中,其值都将作为 true 来判断。

不要通过新建 Boolean 对象的方法将一个非布尔值转换成布尔值。直接使用 Boolean 函数即可

```
var x=Boolean(exp);//这样使用
var x=new Boolean(exp);//不要这样使用
```

如过你用一个对象作为 Boolean 对象的初始化值,则即使该对象是个值为 false 的 Boolean 对象,生成的 Boolean 对象的值也是 true.

不要在该使用 Boolean 原始值的地方使用 Boolean 对象.

String 对象

定义：new String(s)构造函数、function String(s)转换函数。

参数：存储到一个 String 对象中或转换为一个原始字符串的值

描述：

当使用 new 操作符将 String()作为一个构造函数使用时，它将返回一个 String 对象，内容为字符串 s 或 s 的字符串表示

当不带 new 操作符调用 String()构造函数时，只是简单地将 s 转换为原始字符串并返回转换后的值

属性：

length：得到字符串的长度

方法：

String.charAt(n)

描述：取得一个字符串中的第 n 个字符

参数：希望返回的字符在字符串 string 中的索引

返回值：字符串 string 的第 n 个字符

注意：

返回字符串中的第 n 个字符

字符串的起始点为 0

如果不再字符串长度之内返回空字符串

String.charCodeAt(n)

描述：取得字符串中第 n 个字符的编码

参数：返回编码字符的索引

返回值：string 中第 n 个字符的 Unicode 编码。返回的值为一个 16 位的整数，值在 0~5535 之间

注意：如果 n 为负数或大于等于字符串的长度，则返回 NaN

String.fromCharCode(c1,c2,...)

描述：从字符编码创建一个字符串

参数：指定待创建字符串中的字符的 Unicode 编码，一个或多个整数

返回值：一个新的字符串，内容为指定编码对应的字符

注意：这是一个静态方法，是 String()构造函数的一个属性，而不是 String 对象的方法

String.concat(value,...)

描述：连接字符串

参数：一个或多个待连接的字符串

返回值：返回连接之后的字符串

注意：和+作用一样，连接字符串的

String.indexOf(substring[,start])

描述：搜索一个字符串

参数：在 string 中搜索的字符串，一个可选的整数参数 start。指定 string 中起始点。

返回值：在字符串 string 中 start 位置之后，substring 第一次出现的位置，如果没找到返回-1

注意：

String.indexOf()搜索指定的字符串，从前到后搜索

找到返回字符串第一次出现的位置

没找到返回-1

搜索的字符串按照字符串整体搜索

String.lastIndexOf(substring[,start])

描述：从后面搜索

参数：一个可选的整数参数 start。指定 string 中起始点。在 string 中搜索的字符串。

返回值：返回字符串最后一次出现的位置

String.localeCompare(target)

描述：使用本地特定的顺序比较两个字符串

参数：要与 string 使用区分地区设置的方式比较的字符串

返回值

string<target，比 0 小的数

string>target，比 0 大的数

string=target，返回 0

注意

当使用>或者<操作符比较字符串的时候，是按照字符的 Unicode 编码进行比较的，不考虑本地的顺序，这种方式不准确

使用 localeCompare()提供了一个根据默认的本地排序来比较字符串的方法，这个函数利用底层的操作系统提供的排序进行

String.match(regexp)

描述：找到一个或多个正则表达式匹配结果

参数：一个指定要匹配的模式的 RegExp 对象。如果如果这个参数不是一个 RegExp 对象，则它将先被传入 RegExp()构造函数，后转换为 RegExp 对象

返回：一个包含撇皮结果的数组

String.repalce(regexp,replacement)

描述：替换匹配给定正则表达式的一个或多个子串

参数：

指定了要替换的模式的 RegExp 对象。如果这个参数是一个字符串，它将用作一个要搜索的直接量文本模式；它将不会先转化为 RegExp 对象

replacement 为替换文本的字符串，或者一个函数，用在替换时对应的替换文本

返回值：返回替换之后的内容

String.search(regexp)

描述：根据一个正则表达式查找

参数：一个 RegExp 对象，指定要在字符串 string 中查找的模式。如果这个参数不是一个 RegExp，它将先传入 RegExp()构造函数，后转换为一个 RegExp 对象

返回 string 中第一个匹配 regexp 的子串的开始位置，如果没有找到匹配则返回-1

String.slice(start,end)

描述：截取字符串

参数

start 起始点

end 结束的位置

返回值：返回截取之后的字符串

String.substr(start,length)

描述：截取字符串

参数：

start 起始点

length 截取字符串的长度

返回值：返回截取之后的字符串

String.substring(from,to)

描述：截取字符串

参数：起始点、结束点

返回值：返回截取之后的字符串

String.split(delimiter,limit)

描述：将一个字符串切分为一个由字符串组成的数组

参数

delimiter 分隔符

limit 指定已返回数组的最大长度。如果指定，则最多返回数量为这个数字的子串。如果没有指定，则将切分整个字符串，无论结果数组有多长

返回值：返回拆分之后的数组

String.toLowerCase()

描述：返回小写之后的字符串

返回值：返回小写之后的字符串

String.toUperCase()

描述：返回大写之后的字符串

返回值：返回大写之后的字符串

String.toLocaleLowerCase()

描述：返回小写之后的字符串

返回值：返回小写之后的字符串

String.toLocaleUpperCase()

描述：返回大写之后的字符串

返回值：返回大写之后的字符串

String.toString()

描述：返回对象的字符串

返回值：返回字符串

String.trim()

描述：去掉字符串两端的空白字符

返回值：返回去除字符串两端空白的字符串

String.valueOf()

描述：返回对应的字符串

返回值：返回 String 的原始字符串值

String.link(url)

描述：link() 方法创建一个 <a> HTML 元素，用该字符串作为超链接的显示文本，参数作为指向另一个 URL 的超链接。

参数：任何能够指定 a 标签的 href 属性的字符串；它应当是有效的 URL（相对或绝对），任何 & 字符将会被转义为 &，任何 "" 字符将会被转义为 "。

返回值：返回创建好的链接

String.anchor(name)

描述：用 anchor 方法差 un 关键一个<<a>HTML 锚元素

参数：超链接的 name 属性

返回值：返回创建的锚点元素

Array 对象

见七、JavaScript 数组

Function 对象

定义：JavaScript 中的 Function 对象，就是我们常说的函数对象。在 JS 中，所有的函数也是以对象的形式存在的。

语法：

充当 Function 对象的构造函数使用，用于结合 new 关键字构造一个新的 Function 对象。

new Function([argName1 [, argName1 [, argNameN... [, funcBody]]]])

当作普通函数使用，其行为与用法一(使用 new 关键字)完全一致，相当于用法一省略了 new 关键字。

Function([argName1 [, argName1 [, argNameN... [, funcBody]]]])

返回值：Function()的返回值是 Function 类型，返回一个函数对象

注意：

多数时候，你无需显示地通过 new 关键字来构造一个 Function 对象，因为在 JavaScript 中，你可以直接以 function 关键字形式声明函数

JavaScript 在调用 Function()构造函数时编译由该构造函数创建的对象。虽然它使您的脚本在运行时重新定义函数的情况下具有更大的灵活性，但它也会减慢代码的执行速度。为了避免减慢脚本速度，应尽可能少地使用 Function()构造函数。建议优先考虑以 function 关键字的形式声明函数。

Function()会把传入的最后一个参数作为函数定义的执行代码，之前的所有参数均依次作为函数定义的参数。

如果没有指定任何参数，则表示该函数没有定义参数列表，函数的执行代码也为空

如果只指定了一个参数，则该参数将被视作函数的执行代码。如果你想定义一个参数、执行代码为空，请传入两个参数，第二个参数为空字符串即可：new Function(""argName1"", """")。

属性：

arguments：返回该函数执行时内置的 arguments 对象。

caller：返回调用当前函数的函数。

constructor：返回创建该对象的构造函数。

length：返回函数定义的参数个数。

方法：

call()：调用当前 Function 对象，可同时改变函数内的 this 指针引用，函数参数一个个分别传入。

apply()：调用当前 Function 对象，可同时改变函数内的 this 指针引用，函数参数以数组或 arguments 对象的形式传入。

toString()：返回定义该 Function 对象的字符串。

valueOf()：返回 Function 对象本身。

②工具类对象

Math 对象

定义：

Math 是一个内置对象， 为数学常量和数学函数提供了属性和方法。

Math 不是一个函数对象

Math 不是一个构造器. Math 的所有属性和方法都是静态的

属性：

Math.E：欧拉常数，也是自然对数的底数，约等于 2.718.

Math.LN2：2 的自然对数，约等于 0.693

Math.LN10：10 的自然对数，约等于 2.303

Math.LOG2E：以 2 为底 E 的对数，约等于 1.443.

Math.LOG10E：以 10 为底 E 的对数，约等于 0.434

Math.PI：圆周率，一个圆的周长和直径之比，约等于 3.14159.

Math.SQRT1_2：1/2 的平方根，约等于 0.707.

Math.SQRT2：2 的平方根,约等于 1.414.

方法：

Math.abs(x)：返回 x 的绝对值.

Math.ceil(x)：返回 x 向上取整后的值.

Math.floor(x)：返回小于 x 的最大整数。

Math.round(x)：返回四舍五入后的整数.

Math.pow(x,y)：返回 x 的 y 次幂.

Math.sqrt(x)：返回 x 的平方根.

Math.random()：返回 0 到 1 之间的伪随机数.

Math.max([x[,y[,...]]])：返回 0 个到多个数值中最大值

Math.min([x[,y[,...]]])：返回 0 个到多个数值中最小值.

Math.acos(x)：返回 x 的反余弦值.

Math.asin(x)：返回 x 的反正弦值.

Math.atan(x)：以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值.

Math.atan2(x, y)：返回 y/x 的反正切值.

Math.cos(x)：返回 x 的余弦值.

Math.exp(x)：返回 Ex, 当 x 为参数， E 是欧拉常数 (2.718...), 自然对数的底.

Math.log(x)：返回对数

Math.sin(x)：返回正弦值

Date 对象

定义：创建 Date 实例用来处理日期和时间。Date 对象基于 1970 年 1 月 1 日世界协调时起的毫秒数

语法：

构造函数

new Date()

new Date(value)

value 代表自世界协调时 1970 年 1 月 1 日 00:00:00 经过的毫秒数。

new Date(dateString)

dateString 表示日期的字符串值。该字符串应该能被 Date.parse() 方法识别

new Date(year,month,day,hour,minute,second,millisecond)

注意：

需要注意的是只能通过调用 Date 构造函数来实例化日期对象：以常规函数调用它（即不加 new 操作符）将会返回一个字符串，而不是一个日期对象。另外，不像其他 JavaScript 类型，Date 对象没有字面量格式。

Date 需要调用多个参数的构造函数，当数值大于合理范围时（如月份为 13 或者分钟数为 70），会被调整为相邻值。比如 new Date(2013, 13, 1)会等于 new Date(2014, 1, 1)，还会新建一个 2014-02-01 的日期（注意月份是有补 0 的）

如果没有输入任何参数，则 Date 的构造器会依据系统设置的当前时间来创建一个 Date 对象。

如果提供了至少两个参数，其余的参数均会默认设置为 1（如果没有提供 day 参数）或者 0

JavaScript 的时间是由世界标准时间（UTC）1970 年 1 月 1 日开始，用毫秒计时，一天由 86,400,000 毫秒组成。

Date 对象的范围是-100,000,000 天至 100,000,000 天（等效的毫秒值）。

JavaScript 的 Date 对象为跨平台提供了统一的行为。时间属性可以在不同的系统中表示相同的时刻，而如果使用了本地时间对象，则反映当地的时间

JavaScript 的 Date 对象提供了数个 UTC 时间的方法，也相应提供了当地时间的方法。UTC，也就是我们所说的格林威治时间，指的是 time 中的世界时间标准。而当地时间则是指执行 JavaScript 的客户端电脑所设置的时间。

以一个函数的形式来调用 JavaScript 的 Date 对象（i.e., 不使用 new 操作符）会返回一个代表当前日期和时间的字符串。

方法：

Date.now()：返回自 1970-1-1 00:00:00 UTC 至今所经过的毫秒数。

Date.parse()：解析一个表示日期的字符串，并返回从 1970-1-1 00:00:00 所经过的毫秒数。

Date.UTC()：接受和构造函数最长形式的参数相同的参数（从 2 到 7），并返回从 1970-01-01 00:00:00 UTC 开始所经过的毫秒数。

dateObj.getDate()：根据本地时间，返回一个指定的日期对象为一个月中的第几天，返回一个 1 到 31 的整数值。

dateObj.getDay()：根据本地时间返回指定日期对象的星期中的第几天（0-6

dateObj.getFullYear()：根据本地时间返回指定日期对象的年份（四位数年份时返回四位数字）。

dateObj.getHours()：根据本地时间返回指定日期对象的小时（0-23）

dateObj.getMilliseconds()：根据本地时间返回指定日期对象的微秒（0-999）

dateObj.getMinutes()：根据本地时间返回指定日期对象的分钟（0-59）

dateObj.getMonth()：根据本地时间返回指定日期对象的月份（0-11）

dateObj.getSeconds()：根据本地时间返回指定日期对象的秒数（0-59）

dateObj.getTime()：返回从 1970-1-1 00:00:00 UTC（协调世界时）到该日期经过的毫秒数，对于 1970-1-1 00:00:00 UTC 之前的时间返回负值。

dateObj.getTimezoneOffset()：返回当前时区的时区偏移

dateObj.setDate()：根据本地时间为指定的日期对象设置月份中的第几天。

dateObj.setFullYear()：根据本地时间为指定日期对象设置完整年份（四位数年份是四个数字）

dateObj.setHours()：根据本地时间为指定日期对象设置小时数

dateObj.setMilliseconds()：根据本地时间为指定日期对象设置毫秒数

dateObj.setMinutes()：根据本地时间为指定日期对象设置分钟数

dateObj.setMonth()：根据本地时间为指定日期对象设置月份

dateObj.setSeconds()：根据本地时间为指定日期对象设置秒数

dateObj.setTime()：通过指定从 1970-1-1 00:00:00 UTC 开始经过的毫秒数来设置日期对象的时间，对于早于 1970-1-1 00:00:00 UTC 的时间可使用负值。

dateUTCDate()：根据世界时设置 Date 对象中月份的一天 (1 ~ 31)

dateObj.toDateString()：toDateString() 方法以美式英语和人类易读的形式返回一个日期对象日期部分的字符串

dateObj.toJSON()：toJSON() 方法返回 Date 对象的字符串形式

dateObj.toString()：toString() 方法返回一个字符串，表示该日期对象。

dateObj.valueOf()：valueOf() 方法返回一个日期对象的原始值。

valueOf 方法返回以数值格式表示的一个 Date 对象的原始值，从 1970 年 1 月 1 日 0 时 0 分 0 秒（UTC，即协调世界时）到该日期对象所代表时间的毫秒数

RegExp 对象

JavaScript 中的正则表达式

1.什么是正则表达式(regular expression)：

正则表达式是一个描述字符模式的对象。

可以处理更复杂的字符串

JavaScript 中的正则表达式使用 RegExp 对象表示

正则表达式用于对字符串模式匹配及检索替换，是对字符串执行模式匹配的强大工具。

2.如何使用 JavaScript 中的正则表达式

语法：

var patt=new RegExp(pattern,modifiers)

动态创建正则表达式

var patt=/pattern/modifiers;

参数：

pattern(模式)描述了表达式的模式

modifiers(修饰符)用于指定全局匹配、区分大小写的匹配和多行匹配

常用修饰符：修饰符用于执行区分大小写和全局匹配

修饰符	描述
i	执行对大小写不敏感的匹配。
g	执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。
m	执行多行匹配。

方括号：方括号用于查找某个范围内的字符

表达式	描述
[abc]	查找方括号之间的任何字符。
[^abc]	查找任何不在方括号之间的字符。
[0-9]	查找任何从 0 至 9 的数字。
[a-z]	查找任何从小写 a 到小写 z 的字符。
[A-Z]	查找任何从大写 A 到大写 Z 的字符。
[A-z]	查找任何从大写 A 到小写 z 的字符。
[adgk]	查找给定集合内的任何字符。
[^adgk]	查找给定集合外的任何字符。
(red|blue|green)	查找任何指定的选项。

元字符：元字符（Metacharacter）是拥有特殊含义的字符

元字符	描述
.	查找单个字符，除了换行和行结束符。
\w	查找单词字符。
\W	查找非单词字符。
\d	查找数字。
\D	查找非数字字符。
\s	查找空白字符。
\S	查找非空白字符。
\b	匹配单词边界。
\B	匹配非单词边界。
\0	查找 NUL 字符。
\n	查找换行符。
\f	查找换页符。
\r	查找回车符。
\t	查找制表符。
\v	查找垂直制表符。
\xxx	查找以八进制数 xxx 规定的字符。
\xdd	查找以十六进制数 dd 规定的字符。
\uxxxx	查找以十六进制数 xxxx 规定的 Unicode 字符。

量词：

量词	描述
n+	匹配任何包含至少一个 n 的字符串。
n*	匹配任何包含零个或多个 n 的字符串。
n?	匹配任何包含零个或一个 n 的字符串。
n{X}	匹配包含 X 个 n 的序列的字符串。
n{X,Y}	匹配包含 X 或 Y 个 n 的序列的字符串。
n{X,}	匹配包含至少 X 个 n 的序列的字符串。
n$	匹配任何结尾为 n 的字符串。
^n	匹配任何开头为 n 的字符串。
?=n	匹配任何其后紧接指定字符串 n 的字符串。
?!n	匹配任何其后没有紧接指定字符串 n 的字符串。

3.RegExp 对象方法

compile

语法：RegExpObject.compile(regexp,modifier)

参数：

regexp 正则表达式

modifier 模式修饰符

描述：compile 方法用于在脚本执行过程中编译正则表达式或者改变和重新编译正则表达式

exec

语法：RegExpObjec.exec(string)

参数：string 指定的字符串

描述：在目标字符串中执行一次正则匹配操作，并将结果以数组的形式返回

注意：

每次执行 exec()函数都只查找最多一个匹配并返回

如果为正则表达式设置了全局标志(g)，exec()函数仍然只返回最多一个匹配，不过我们再次调用该对象的 exec()函数就可以查找并返回下一个匹配

如果为正则表达式设置了全局标识 g，test 函数仍然只查找最多一个匹配，不过我们再次嗲用该对象的 test 函数就可以查找下一个匹配

如果 regExpObject 带有全局标志 g，test()函数不是从字符串的开头开始查找，而是从属性 regExpObject.lastIndex 所指定的索引处开始查找该属性值默认为 0，所以第一次仍然是从字符串的开头查找。当找到一个匹配时，test()函数会将 regExpObject.lastIndex 的值改为字符串中本次匹配内容的最后一个字符的下一个索引位置。当再次执行 test()函数时，将会从该索引位置处开始查找，从而找到下一个匹配。

当我们使用 test()函数执行了一次匹配之后，如果想要重新使用 test()函数从头开始查找，则需要手动将 regExpObject.lastIndex 的值重置为 0。如果 test()函数再也找不到可以匹配的文本时，该函数会自动把 regExpObject.lastIndex 属性重置为 0

test

语法：regExpObject.test(str)

参数：string 指定的字符串

描述：检测字符串中是否存在正则表达式模式对应的匹配,返回值布尔类型

注意：每次执行 test 函数都只查找最多一个匹配，找到返回 true，否则 false

4.支持正则表达式的 String 对象的方法

search

语法：str.search(regexp)

参数：regexp 一个正则表达式对象。如果传入一个非正则表达式对象，则会使用 new RexExp(obj)隐式地将其转换为正则表达式

描述：如果匹配成功，则 search() 返回正则表达式在字符串中首次匹配项的索引。否则，返回 -1

注意：

当你想要知道字符串中是否存在某个模式（pattern）时可使用 search，类似于正则表达式的 test 方法。

当要了解更多匹配信息时，可使用 match（会更慢），该方法类似于正则表达式的 exec 方法。

match

语法：str.match(regexp)

参数：regexp 一个正则表达式对象。如果传入一个非正则表达式对象，则会隐式地使用 new RegExp(obj)将其转换为正则表达式对象

描述：当字符串匹配到正则表达式（regular expression）时，match() 方法会提取匹配项，如果找到返回匹配结果的数组，如果没有匹配项，则返回 null

注意：

如果正则表达式没有 g 标志，返回和 RegExp.exec(str) 相同的结果

而且返回的数组拥有一个额外的 input 属性，该属性包含原始字符串

还拥有一个 index 属性，该属性表示匹配结果在原字符串中的索引（以 0 开始）

如果正则表达式包含 g 标志，则该方法返回一个包含所有匹配结果的数组。如果没有匹配到，则返回 null。

replace

语法：str.replace(regexp|substr, newSubStr|function)

参数：

regexp，一个 RegExp 对象。该正则所匹配的内容会被第二个参数的返回值替换掉

substr,一个要被 newSubStr 替换的字符串

newSubStr,替换掉第一个参数在原字符串中的匹配部分。该字符串中可以内插一些特殊的变量名。

function,一个用来创建新子字符串的函数，该函数的返回值将替换掉第一个参数匹配到的结果。

描述：replace() 方法使用一个替换值（replacement）替换掉一个匹配模式（pattern）在原字符串中某些或所有的匹配项，并返回替换后的字符串。这个替换模式可以是字符串或者 RegExp（正则表达式），替换值可以是一个字符串或者一个函数。

注意：

该方法并不改变调用它的字符串本身，而只是返回替换后的字符串

在进行全局的搜索替换时，第一个参数要么是包含 g 标志的正则表达式，要么是包含 g 标志的字符串。

split

语法：str.split([separator][, limit])

参数：

separator,指定用来分割字符串的字符（串）。separator 可以是一个字符串或正则表达式。 如果忽略 separator，则返回的数组包含一个由原字符串组成的元素。如果 separator 是一个空字符串，则 str 将会转换成一个由原字符串中字符组成的数组。

limit,一个整数，限定返回的分割片段数量。split 方法仍然分割每一个匹配的 separator，但是返回的数组只会截取最多 limit 个元素。

描述：split() 方法通过把字符串分割成子字符串来把一个 String 对象分割成一个字符串数组。

注意：

当找到一个 seperator 时，separator 会从字符串中被移除，返回存进一个数组当中的子字符串。如果忽略 separator 参数，则返回的数组包含一个元素，该元素是原字符串。如果 separator 是一个空字符串，则 str 将被转换为由字符串中字符组成的一个数组。

如果 separator 是一个正则表达式，且包含捕获括号（capturing parentheses），则每次匹配到 separator 时，捕获括号匹配的结果将会插入到返回的数组中

③错误对象

Error 对象

定义：创建一个 error 对象。当发生运行时异常时,其实例会被抛出.Error 对象可作为用户自定义异常的基对象.下面是关于标准内置 error 类型的说明.

语法：new Error([message[, fileName[,lineNumber]]])

参数：

message 可阅读的错误描述信息

filename 创建 Error 对象的 fileName 属性的值。默认是包含异常代码的文件名

lineNumber 创建 Error 对象的 lineNumber 属性的值。默认是构造 Error 对象的行数

描述：当代码运行时的发生错误，会创建新的 Error 对象，并将其抛出

Error 类型：

EvalError：创建一个 error 实例，表示错误的原因：与 eval() 有关

InternalError：创建一个代表 Javascript 引擎内部错误的异常抛出的实例. 如: ""递归太多"".

RangeError：创建一个 error 实例，表示错误的原因：数值变量或参数超出其有效范围。

ReferenceError：创建一个 error 实例，表示错误的原因：无效引用

SyntaxError：创建一个 error 实例，表示错误的原因：eval()在解析代码的过程中发生的语法错误

TypeError：创建一个 error 实例，表示错误的原因：变量或参数不属于有效类型

URIError：创建一个 error 实例，表示错误的原因：给 encodeURI()或 decodeURI()传递的参数无效

2.自定义对象

在经典的面向对象语言中，对象是指数据和在这些数据上进行的操作的集合。与 C++ 和 Java 不同，JavaScript 是一种基于原型的编程语言，并没有 class 语句，而是把函数用作类。

①通过 var obj={} ------ 对象字面量（object literal）法

对象字面量是一个表达式，这个表达式的每次运算都创建并初始化一个新对象。每次计算对象字面量的时候，也都会计算他的每个属性的值。也就是说，如果在一个重复调用的函数中的循环体内使用了对象直接量，它将创建很多新对象，并且每次创建的对象的属性值也有可能不同。

②通过 var obj=new Object()创建

③通过构造函数创建对象

function Person(){}

var Person=function(){}

构造器属性(constructor property)

当我们创建对象的时候，实际上同时也赋予了该对象一种特殊的属性，就是构造器属性

这个构造器属性实际上是一个指向用于创建该对象的构造器函数的引用

通过 instanceof 操作符可以检测一个对象是否由某个指定的构造器函数创建的

注意：

使用的时候通过 new 操作符得到对象

var person1=new Person()

用构造器创建对象的时候可以接收参数

构造器函数的首字母最好大写，区别其他的一般函数

④通过 Object.create 创建对象

3、访问对象的属性和方法

1.属性

①数据属性

数据属性包含一个数据值的位置，在这个位置可以读取和写入值

4 个描述行为的特性

[[writable]]：表示能否修改属性的值。默认值为 true

[[Enumerable]]：表示能否通过 for in 循环返回属性。代表属性是否可以枚举。直接在对象上定义的属性默认值为 true。

[[configurable]]：表示是否能通过 delete 删除属性从而重新定义属性，能否修改属性的特性，或者能否把属性修改为访问器属性。直接在对象上定义的属性，他们的默认值为 true

[[value]]：包含这个属性的数据值。读取属性值的时候，从这个位置读取。写入属性值的时候，把新值保存在这个位置。这个特性的默认值为 undefined

②存取器属性

get：获取属性的值

set：设置属性的值

2.三个相关的对象特性(Object attribute)

①对象的原型(prototype)指向另外一个对象，本对象的属性继承自它的原型对象

通过对象字面量创建的对象使用 Object.prototype 作为它们的原型

通过 new 创建的对象使用构造函数的 prototype 属性作为他们的原型

通过 Object.create()创建的对象使用第一个参数(也可以是 null)作为它们的原型

②对象的类(class)是一个标识对象类型的字符串

ECMAScript3 和 ECMAScript5 都未提供这个属性的方法，可以通过对象的 toString()方法间接查询

③对象的扩展标记(extensible flag)指明了(在 ECMAScript5 中)是否可以向该对象添加新属性

所有内置对象和自定义对象都是显示可扩展的，宿主对象的可扩展性由 JavaScript 引擎定义的。

可以通过 Object.preventExtensions()将对象设置为不可扩展的，而且不能再转换成可扩展的了，可以通过 Object.isExtensible()检测对象是否是可扩展的

preventExtensions()只影响到对象本身的可扩展性，如果给一个不可扩展的对象的原型添加属性，这个不可扩展的对象同样会继承这些新属性

可扩展性的目的是将对象锁定，防止外接干扰，通常和对象的属性的可配置行与可写性配合使用

Object.seal()和 Object.preventExtensions()类似，除了能够将对象设置为不可扩展的，还可以将对象的所有自身属性都设置为不可配置的。也就是说不能给这个对象添加新属性，而且它已有的属性也不能删除或配置，不过它已有的可写属性依然可以设置。可以通过 Object.isSealed()检测对象是否封闭

Object.freeze()将更严格地锁定对象--冻结(frozen).除了对象设置为不可扩展的和将其属性设置为不可配置的之外，还可以将它自身的所有数据属性设置为只读(如果对象的存储器属性具有 setter 方法，存取器属性将不受影响，仍可以通过给属性赋值调用它们)。可以使用 Object.isFroze()来检测对象是否冻结。

3.属性操作

①访问属性

对象名.属性名

对象名[属性名]

当处于某个对象方法内部的时候，可以通过 this 来访问同一对象的属性

②添加属性

对象名.属性名=值

对象名[属性名]=值

③修改属性

对象名.属性名=值

对象名[属性名]=值

④删除属性

delete 对象名.属性名

delete 对象名[属性名]

注意：

delete 只能删除自身属性，不能删除继承属性

要删除继承属性，只能从定义它属性的原型对象上删除它，而且这会影响到所有继承自这个原型的对象。

delete 只是断开属性和宿主对象的联系，而不会去操作属性的属性

delete 不能删除哪些可配制性为 false 的属性

⑤遍历属性

for in 遍历属性

⑥访问方法

对象名.方法名()

注意：如果对象名.方法名的形式，将返回定义函数的字符串

4.传递对象和对象比较

传递对象：

对象传递是通过引用进行传递的

在引用上做的改动会影响原对象

对象比较：当对两个对象做比较的时候，只有当两个引用指向同一个对象时为 true；否则都为 false

七、JavaScript 数组

1、数组定义

数组是值的有序集合

2、创建数组

1.字面量形式

①空数组：var arr=[];

②带有元素的数组：var arr=[1,2,3,1,2]

③数组值可以是任意类型：var arr=[1,2.3,'king',true,null,undefined,[1,2,3],{name:'king',age:12}]

④省略数组中的某个值，默认值为 undefined：var arr=[1,,3]

2.通过构造函数 Array()创建数组

①调用时没有参数：var arr=new Array()

注意：空数组，等同于 arr=[]；

②调用时传递一个参数：var arr=new Array(5)

注意：代表指定了数组的长度

如果预先知道数组元素个数，可以通过参数指定，这种形式的 Array()构造函数可以用来分配一个数组空间。

③超过两个参数：var arr=new Array(1,2,3,true,'king')

注意：显示指定了数组中的值

3、使用数组

1.根据下标找到对应的值

2.向数组中添加元素

arr[下标]=值;

arr.push(值,...):数组末尾添加元素

arr.shift(值,...):数组开始添加元素

3.读取数组中元素

arr[下标]

4.修改数组中的元素

arr[下标]=值;

5.删除数组中元素

delete arr[下标]

使用 delete 删除数组元素不会改变数组的 length 属性

arr.pop()：删除数组的元素

arr.unshift()：删除数组开头的元素

可以通过设置 length 属性，删除数组后面的元素

4、遍历数组

1.for 循环遍历下标连续的数组

2.for-in 遍历数组

3.通过 forEach()遍历数组

语法：Array.forEach(function(value[,index[,array]]){函数体})

5、常用方法

Array.join([delimiter])

描述：将数组中的值连接成字符串

参数：以指定分隔符连接，如果不指定，默认以,连接

返回值：返回连接之后的字符串

注意：Array.join()是 string.split()方法的逆向操作

Array.reverse()

描述：数组倒置

返回值：返回倒置之后的数组

Array.sort()

描述：数组排序函数

参数：如果不带参数调用 sort，数组以字母顺序进行排序，升序

返回值：返回排序之后的数组

注意：如果数组中包含 undefined 元素，它们会被排到数组的尾部

Array.concat(value,...)

描述：创建并返回一个新数组，数组中包含调用 concat 的原始数组的元素和 concat 的每个参数。

返回值：返回连接之后的新数组

注意：如果这些参数中的任何一个自身是数组，则连接的是数组的元素，而非数组本身

Array.slice(start[,end])

描述：返回数组中的部分

参数：start 起始点、end 结束位置

返回值：返回数组中的部分

Array.splice(index , howMany[, element1[, ...[, elementN]]])

描述：添加或删除数组中的一个或多个元素

参数：

index 从数组的哪一位开始修改内容。如果超出了数组的长度，则自动从数组末尾开始添加内容；如果是负值，则表示从数组末位开始的第几位。

howMany 是整数，表示要移除的数组元素的个数。如果 howmany 是 0，则不移除元素。这种情况下，至少应添加一个新元素。如果 howmany 超出了 index 位之后的元素的总数，则从 index 向后至数组末尾的全部元素都将被删除（含第 index 位）。如果没有指定 howmany 参数（如上第二种语法，是 SpiderMonkey 的扩展功能），将会删除第 index 位之后的所有元素（不含第 index 位）

element1...要添加进数组的元素。如果不指定，则 splice 只删除数组元素。

返回值：由被删除的元素组成的一个数组。如果只删除了一个元素，则返回只包含一个元素的数组。如果没有删除元素，则返回空数组。

注意：如果添加进数组的元素个数不等于被删除的元素个数，数组的长度会发生相应的改变。

Array.push(value,...)

描述：像数组末尾添加一个或者多个元素

参数：添加的值

返回值：返回数组的长度

Array.pop()

描述：弹出数组的最后的元素

返回值：返回弹出的元素

Array.unshift(value,...)

描述：向数组开始添加一个或者多个元素

参数：添加的值

返回值：返回数组的长度

Array.shift()

描述：弹出数组开始的元素

返回值：返回弹出之后的值

Array.map()

描述：返回一个由原数组中的每个元素调用一个指定方法后的返回值组成的新数组

语法：Array.map(callback)

参数：

callback 回调函数

currentValue,callback 的第一个参数,数组中当前被传递的元素

index，callback 的第二个参数，数组中当前被传递的元素的索引

array，callback 的第三个参数，调用 map 方法的数组

注意

map 方法会给原数组中的每个元素都按顺序调用一次 callback 函数

callback 每次执行后的返回值组合起来形成一个新数组

callback 函数只会在有值的索引上被调用，那些从来没被赋过值或者使用 delete 删除的索引则不会被调用

Array.filter()

描述：方法使用指定的函数测试所有元素，并创建一个包含所有通过测试的元素的新数组

语法：arr.filter(callback)

注意：filter 为数组中的每个元素调用一次 callback 函数，并利用所有使得 callback 返回 true 或 等价于 true 的值 的元素创建一个新数组

callback 只会在已经赋值的索引上被调用，对于那些已经被删除或者从未被赋值的索引不会被调用。那些没有通过 callback 测试的元素会被跳过，不会被包含在新数组中。

Array.reduce()

描述：reduce 方法接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终为一个值

语法：arr.reduce(callback,[initialValue])

参数：

callback

previousValue，上一次调用返回的值或者是提供的初始值(initialValue)

currentValue,数组中当前被处理的元素

index,当前元素在数组中得索引

array，调用 reduce 的数组

initialValue：作为第一次调用 callback 的第一个参数

注意：

reduce 为数组中的每一个元素依次执行回调函数，不包括数组中被删除或从未被赋值的元素，接受四个参数：初始值（或者上一次回调函数的返回值），当前元素值，当前索引，调用 reduce 的数组

回调函数第一次执行时，previousValue 和 currentValue 可以是一个值，如果 initialValue 在调用 reduce 时被提供，那么第一个 previousValue 等于 initialValue ，并且 currentValue 等于数组中的第一个值；如果 initialValue

未被提供，那么 previousValue 等于数组中的第一个值，currentValue 等于数组中的第二个值。

Array.reduceRight()

描述：与 reduce()方法的执行方向相反

Array.some()

描述：测试数组中的某些元素是否通过了指定函数的测试

语法：arr.some(callback)

注意：some 为数组中的每一个元素执行一次 callback 函数，直到找到一个使得 callback 返回一个真值（即可转换为布尔值 true 的值）

Array.every()

描述：测试数组的所有元素是否都通过了指定函数的测试。

语法：arr.every(callback)

注意：

every 方法为数组中的每个元素执行一次 callback 函数，直到它找到一个使 callback 返回 falsy（表示可转换为布尔值 false 的值）的元素。如果发现了一个这样的元素，every 方法将会立即返回 false。否则，callback 为每一个元素返回 true，every 就会返回 true。callback 只会为那些已经被赋值的索引调用。不会为那些被删除或从来没被赋值的索引调用。

Array.indexOf()

Array.lastIndexOf()

Array.isArray()

描述：检测某个值是否为数组

Array.toString()

描述：返回一个字符串，表示指定的数组及其元素。

返回值：返回一个字符串，表示指定的数组及其元素。

八、JavaScript 事件

1、事件类型

1.定义

指的是文档或者浏览器窗口中发生的一些特定交互瞬间。我们可以通过侦听器（或者处理程序）来预定事件，以便事件发生的时候执行相应的代码。

①事件类型(event type)

事件类型是一个用来说明发生什么类型事件的字符串。像鼠标悬浮，按下键盘等。我们也可以把事件类型叫做事件名字，用特定的名字来标识所谈论的特定类型的事件。

②事件目标(event target)

事件目标是发生的事件或与之相关的对象。当讲事件时，我们必须同时指定类型和目标。像 window 上的 load 事件或者链接的 click 事件。在客户端 js 的应用程序中，Window、Document、和 Element 对象是最常见的事件目标，但是某些事件也是由其它类型的对象触发的。

③事件处理程序(event handler)或事件监听程序(event listener)

我们用户在页面中进行的点击这个动作，鼠标移动的动作，网页页面加载完成的动作等，都可以称之为事件名称，即：click、mousemove、load 等都是事件的名称。响应某个事件的函数则称为事件处理程序，或者叫做事件侦听器。

④事件对象(event object)

事件对象是与特定事件相关且包含有关该事件详细信息的对象。事件对象作为参数传递给事件处理程序函数。所有的事件对象都有用来指定事件类型的 type 属性和指定事件目标的 target 属性。每个事件类型都为其相关的事件对象定义一组属性。

⑥事件传播(event propagation)

事件传播是浏览器决定那个对象触发其事件处理程序的过程。

2.事件模型

①内联模型

这种模型是最传统接单的一种处理事件的方法。在内联模型中，事件处理函数是 HTML 标签的一个属性，用于处理指定事件。虽然内联在早期使用较多，但它是和 HTML 混写的，并没有与 HTML 分离

在 HTML 中把事件处理函数作为属性执行 js 代码

在 HTML 中把事件处理函数作为属性执行 js 函数

②脚本模型

由于内联模型违反了 HTML 与 JavaScript 代码层次分离的原则。为了解决这个问题，我们可以在 JavaScript 中处理事件。这种处理方式就是脚本模型。

③DOM2 模型

“DOM2 级事件”定义了两个方法，用于添加事件和删除事件处理程序的操作：

addEventListener()和 removeEventListener()。所有 DOM 节点中都包含这两个方法，并且它们都接受 3 个参数；事件名、函数、冒泡或捕获的布尔值(true 表示捕获，false 表示冒泡)。IE 事件处理程序，IE 中实现了 DOM 中类似的两个方法：attachEvent()和 detachEvent().这两个方法接受相关的两个参数：事件处理程序名称和事件处理程序函数。在 IE8 及 IE8 之间版本中只支持事件冒泡，所以通过 attachEvent()添加的事件处理都会被添加到冒泡阶段。

3.传统事件类型

鼠标事件

属性	描述	DOM
onclick	当用户点击某个对象时调用的事件句柄。	2
oncontextmenu	在用户点击鼠标右键打开上下文菜单时触发	
ondblclick	当用户双击某个对象时调用的事件句柄。	2
onmousedown	鼠标按钮被按下。	2
onmouseenter	当鼠标指针移动到元素上时触发。	2
onmouseleave	当鼠标指针移出元素时触发	2
onmousemove	鼠标被移动。	2
onmouseover	鼠标移到某元素之上。	2
onmouseout	鼠标从某元素移开。	2
onmouseup	鼠标按键被松开。	2

键盘事件

属性	描述	DOM
onkeydown	某个键盘按键被按下。	2
onkeypress	某个键盘按键被按下并松开。	2
onkeyup	某个键盘按键被松开。	2

事件对象 Event

常量

静态变量	描述	DOM
CAPTURING-PHASE	当前事件阶段为捕获阶段(3)	1
AT-TARGET	当前事件是目标阶段,在评估目标事件(1)	2
BUBBLING-PHASE	当前的事件为冒泡阶段 (2)	3

属性

属性	描述	DOM
bubbles	返回布尔值，指示事件是否是起泡事件类型。	2
cancelable	返回布尔值，指示事件是否可拥可取消的默认动作。	2
currentTarget	返回其事件监听器触发该事件的元素。	2
eventPhase	返回事件传播的当前阶段。	2
target	返回触发此事件的元素（事件的目标节点）。	2
timeStamp	返回事件生成的日期和时间。	2
type	返回当前 Event 对象表示的事件的名称。	2

方法

方法	描述	DOM
initEvent()	初始化新创建的 Event 对象的属性。	2
preventDefault()	通知浏览器不要执行与事件关联的默认动作。	2
stopPropagation()	不再派发事件。	2

目标事件对象

方法

方法	描述	DOM
addEventListener()	允许在目标事件中注册监听事件(IE8 = attachEvent())	2
dispatchEvent()	允许发送事件到监听器上 (IE8 = fireEvent())	2
removeEventListener()	运行一次注册在事件目标上的监听事件(IE8 = detachEvent())	2

框架/对象(Frame/Object)事件

属性	描述	DOM
onabort	图像的加载被中断。（<object>）	2
onbeforeunload	该事件在即将离开页面（刷新或关闭）时触发	2
onerror	在加载文档或图像时发生错误。（<object>, <body>和 <frameset>）	
onhashchange	该事件在当前 URL 的锚部分发生修改时触发。	
onload	一张页面或一幅图像完成加载。	2
onpageshow	该事件在用户访问页面时触发	
onpagehide	该事件在用户离开当前网页跳转到另外一个页面时触发	
onresize	窗口或框架被重新调整大小。	2
onscroll	当文档被滚动时发生的事件。	2
onunload	用户退出页面。（<body> 和 <frameset>）	2

表单事件

属性	描述	DOM
onblur	元素失去焦点时触发	2
onchange	该事件在表单元素的内容改变时触发(<input>, <keygen>, <select>, 和 <textarea>)	2
onfocus	元素获取焦点时触发	2
onfocusin	元素即将获取焦点是触发	2
onfocusout	元素即将失去焦点是触发	2
oninput	元素获取用户输入是触发	3
onreset	表单重置时触发	2
onsearch	用户向搜索域输入文本时触发（<input="search">）	
onselect	用户选取文本时触发（<input> 和 <textarea>）	2
onsubmit	表单提交时触发	2

剪贴板事件

属性	描述	DOM
oncopy	该事件在用户拷贝元素内容时触发	
oncut	该事件在用户剪切元素内容时触发	
onpaste	该事件在用户粘贴元素内容时触发	

打印事件

属性	描述	DOM
onafterprint	该事件在页面已经开始打印，或者打印窗口已经关闭时触发	
onbeforeprint	该事件在页面即将开始打印时触发	

拖动事件

事件	描述	DOM
ondrag	该事件在元素正在拖动时触发	
ondragend	该事件在用户完成元素的拖动时触发	
ondragenter	该事件在拖动的元素进入放置目标时触发	
ondragleave	该事件在拖动元素离开放置目标时触发	
ondragover	该事件在拖动元素在放置目标上时触发	
ondragstart	该事件在用户开始拖动元素时触发	
ondrop	该事件在拖动元素放置在目标区域时触发	

动画事件

事件	描述	DOM
animationend	该事件在 CSS 动画结束播放时触发	
animationiteration	该事件在 CSS 动画重复播放时触发	
animationstart	该事件在 CSS 动画开始播放时触发	

过渡事件

事件	描述	DOM
transitionend	该事件在 CSS 完成过渡后触发。	

其它事件

事件	描述	DOM
onmessage	该事件通过或者从对象(WebSocket, Web Worker, Event Source 或者子 frame 或父窗口)接收到消息时触发	
onmousewheel	已废弃。 使用 onwheel 事件替代	
ononline	该事件在浏览器开始在线工作时触发。	
onoffline	该事件在浏览器开始离线工作时触发。	
onpopstate	该事件在窗口的浏览历史（history 对象）发生改变时触发。 event occurs when the window's history changes	
onshow	该事件当 <menu> 元素在上下文菜单显示时触发	
onstorage	该事件在 Web Storage(HTML 5 Web 存储)更新时触发	
ontoggle	该事件在用户打开或关闭 <details> 元素时触发	
onwheel	该事件在鼠标滚轮在元素上下滚动时触发	

事件监听对象

方法

方法	描述	DOM
handleEvent()	把任意对象注册为事件处理程序	2

文档事件对象

方法

方法	描述	DOM
createEvent()		2

鼠标/键盘事件对象

属性

属性	描述	DOM
altKey	返回当事件被触发时，"ALT" 是否被按下。	2
button	返回当事件被触发时，哪个鼠标按钮被点击。	2
clientX	返回当事件被触发时，鼠标指针的水平坐标。	2
clientY	返回当事件被触发时，鼠标指针的垂直坐标。	2
ctrlKey	返回当事件被触发时，"CTRL" 键是否被按下。	2
Location	返回按键在设备上的位置	3
charCode	返回onkeypress事件触发键值的字母代码。	2
key	在按下按键时返回按键的标识符。	3
keyCode	返回onkeypress事件触发的键的值的字符代码，或者 onkeydown 或 onkeyup 事件的键的代码。	2
which	返回onkeypress事件触发的键的值的字符代码，或者 onkeydown 或 onkeyup 事件的键的代码。	2
metaKey	返回当事件被触发时，"meta" 键是否被按下。	2
relatedTarget	返回与事件的目标节点相关的节点。	2
screenX	返回当某个事件被触发时，鼠标指针的水平坐标。	2
screenY	返回当某个事件被触发时，鼠标指针的垂直坐标。	2
shiftKey	返回当事件被触发时，"SHIFT" 键是否被按下。	2

方法

方法	描述	W3C
initMouseEvent()	初始化鼠标事件对象的值	2
initKeyboardEvent()	初始化键盘事件对象的值	3

注意：

所有的事件处理函数都会都有两个部分组成，on + 事件名称，例如 click 事的事件处理函数就是：onclick。在这里，我们主要谈论脚本模型的方式来构建事件，违反分离原则的内联模式，我们忽略掉。

对于每一个事件，它都有自己的触发范围和方式，如果超出了触发范围和方式，事件处理将失效

2、事件流

事件流是描述的从页面接受事件的顺序，当几个都具有事件的元素层叠在一起的时候，那么你点击其中一个元素，并不是只有当前被点击的元素会触发事件，而层叠在你点击范围的所有元素都会触发事件。事件流包括两种模式：冒泡和捕获。

事件冒泡，是从里往外逐个触发。

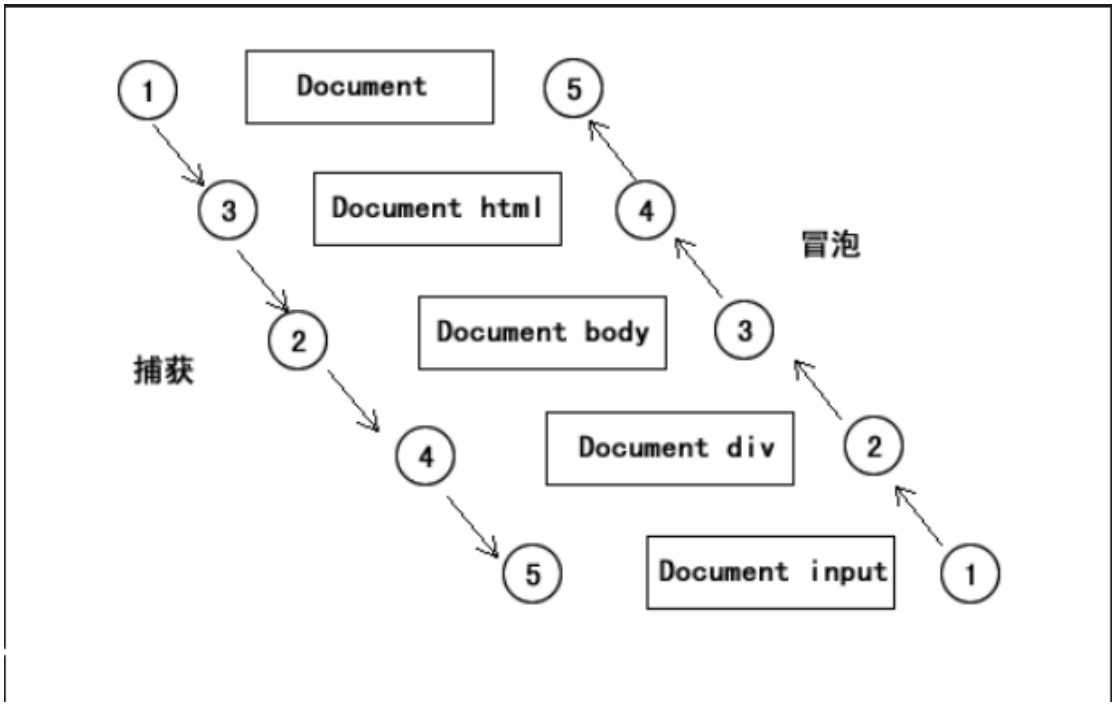
事件捕获，是从外往里逐个触发。

那么现代的浏览器默认情况下都是冒泡模型，而捕获模式则是早期的 Netscape 默认情况。而现在的浏览器要使用 DOM2

级模型的事件绑定机制才能手动定义事件流模式。

IE 中的事件流是事件冒泡(event bubbling)

Netscape Communicator 的事件流是事件捕获流



3、事件兼容

事件绑定:addEventListener 与 attachEvent

事件移除:removeEventListener 与 detachEvent

获取事件对象:e.target 与 window.event.srcElement

阻止冒泡:e.stopPropagation 与 window.event.cancelBubble

阻止默认:e.preventDefault 与 window.event.returnValue