#### 一些基础

Win快捷键

**常用的快捷键、**ctrl+c 复制

ctrl+v 粘贴

ctrl+x 剪切（就是移动文件，在原来的文件夹ctrl+x一个文件，然后在新文件夹中ctrl+v粘贴）

ctrl+tab 切换（具体切换什么，要看是什么软件）

alt+F4 关闭程序

F2 重命名

F5 刷新，比如看网页的时候，想刷新网页，按f5

ctrl+z 撤销，就是这一步干错了，就ctrl+z撤销

**windows+E**  打开资源管理器

windows+D 显示桌面

ctrl+空格 **切换中英文，严禁用shift键切换，所有同学现在调整搜狗拼音输入法：**

常见的联网文件传输

C:\Users\Danny\AppData\Local\Microsoft\Windows\Temporary Internet Files

这个文件夹叫做临时文件夹**（文件夹知道存在就可以了，不用自己试着去找）**

**服务器和客户端的传输**

服务器上存放着网页的相关文件，包括html文件、css文件、js文件、图片等。当我们打开浏览器，输入网址，我们的计算机就会对这些文件发出HTTP请求。

服务器收到请求之后，会把这些文件通过HTTP协议，传输到我们的计算机中（保存到了刚才那个临时文件夹中）。这些文件，将在我们计算机本地的浏览器中，进行渲染、呈递

|  |
| --- |
| 1. http://www.icdjs.com/1.html |

就是服务器上面的1.html文件

|  |
| --- |
| 1. http://www.icdjs.com/ |

看似没有精确到一个文件，但是有一个规定，就是**index.html是默认的首页文件**。

index就是英语“目录”的意思。

|  |
| --- |
| 1. http://www.icdjs.com/aaa/b.html |

服务器上面有一个aaa文件夹，这个文件夹里面有一个b.html文件。

本质

**网页是真实物理的文件。并且一个网页是很多的物理文件组成的：html文件、图片文件、js文件、css文件。这些文件要通过特殊软件才能上传到服务器上。然后就能让用户看了。用户通过浏览器，访问网址，服务器上面的文件就会通过http请求悄悄地传输到用户的电脑中的临时文件夹中，在用户的电脑中执行、渲染、呈递**

Html 超文本标记语言

只能增加文本的语义 、结构

可以压缩网页的尺寸的

#### Html部分

Html文档类型

<!DOCTYPE html>

|  |  |
| --- | --- |
| HTML4.01 | **Strict** 严格的，体现在一些标签不能使用，比如u  **Transitional** 普通的  **Frameset** 带有框架的页面 |
| XHTML1.0  严格体现在小写标签、闭合、引号 | **Strict** 严格的，体现在一些标签不能使用，比如u  **Transitional** 普通的**（我们学习的版本）**  **Frameset** 带有框架的页面 |

**Html：xt 用写u一类的标签当做css勾来做**

**Html：5 h5**

Html <head>

head> 元素

<head> 元素是所有头部元素的容器。<head> 内的元素可包含脚本，指示浏览器在何处可以找到样式表，提供元信息，等等。

以下标签都可以添加到 head 部分：<title>、<base>、<link>、<meta>、<script> 以及 <style>

<title> 标签定义文档的标题。

title 元素在所有 HTML/XHTML 文档中都是必需的。

title 元素能够：

* 定义浏览器**工具栏中的标题**
* 提供页面被添加到**收藏夹时**显示的标题
* 显示在搜索**引擎结果中的页面标题**

<base> 元素

<base> 标签为页面上的所有链接规定默认地址或默认目标（target）：

<head>

<base href="http://www.w3school.com.cn/images/" />

<base target="\_blank" />

</head>

<link> 元素

<link> 标签定义文档与外部资源之间的关系。

<link> 标签最常用于连接样式表：

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css" />

</head>

Media 属性

所有浏览器都支持值为 "screen"、"print" 以及 "all" 的 media 属性。

如果您打印本页面，或者在打印预览模式中打开它，此页面会通过 media="print" 样式表进行格式化。"print" 样式表会把本页面格式化为白底黑字。

rel属性

rel 属性的 "stylesheet 所有浏览器支持

type属性

type 属性规定被链接文档的 MIME 类型。

该属性最常见的 MIME 类型是 "text/css"，该类型描述样式表

Href 属性

超链接的 URL。可能的值：

* 绝对 URL - 指向另一个站点（比如 href="http://www.example.com/theme.css"）
* 相对 URL - 指向站点内的某个文件（href="/themes/theme.css"）

<style> 元素

<style> 标签用于为 HTML 文档定义样式信息。

您可以在 style 元素内规定 HTML 元素在浏览器中呈现的样式：

<head>

<style type="text/css">

body {background-color:yellow}

p {color:blue}

</style>

</head>

 <meta> 元素

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

针对搜索引擎的关键词

一些搜索引擎会利用 meta 元素的 name 和 content 属性来索引您的页面。

描述：

<meta name="**description**" content="Free Web tutorials on HTML, CSS, XML" />

关键词：

<meta name="**keywords**" content="HTML, CSS, XML" />

name 和 content 属性的作用是描述页面的内容。

5s延迟后重定向

<head>

<meta **http-equiv="Refresh" content="5;url=http://www.w3school.com.cn**" />

</head>

字符集

第一种：UTF-8

|  |
| --- |
| 1. <meta http-equiv="Content-Type" content="text/html;charset=**UTF-8**"> |

第二种：gb2312

|  |
| --- |
| 1. <meta http-equiv="Content-Type" content="text/html;charset=**gb2312**"> |

也可以写成gbk

|  |
| --- |
| 1. <meta http-equiv="Content-Type" content="text/html;charset=**gbk**"> |

**保存类型要与声明的东西相同**

**utf-8 内容多，文件大**

**gb2312 内容小，文件小加载快**

<meta http-equiv="charset" content="iso-8859-1">

<meta http-equiv="expires" content="31 Dec 2008">

###### <body>

标题 <h1> - <h5>

主标题 一级标签

段落 <p>

HTML标签是分等级的，HTML将所有的标签分为两种：容器级、文本级。

顾名思义，**容器级的标签，里面可以放置任何东西；文本级的标签里面，只能放置文字、图片、表单元素**。

p标签是一个文本级标签。从学习p的第一天开始，就要死死记住：p里面只能放文字、图片、表单元素。其他的一律不能放。’

正确的：

|  |
| --- |
| 1. <h3>个人经历</h3> 2. <p> 3. Angelababy从小怎么怎么样 4. </p> |

错误的：

|  |
| --- |
| 1. <p> 2. **<h3>个人经历</h3>** 3. Angelababy从小怎么怎么样 4. </p> |

里面的话会截断p ，然后后面的就轮空了

连接 <a href=[www.baidu.com](http://www.baidu.com)”> 百度一下</a>

**注释：**在 href 属性中指定链接的地址。

图像

<img src="w3school.jpg" width="104" height="142" />

HTML 标签对大小写不敏感：<P> 等同于 <p>。许多网站都使用大写的 HTML 标签。但是推荐小写

<p> 浏览器会自动地在段落的前后添加空行。（<p> 是块级元素）

**HTML 折行**

如果您希望在不产生一个新段落的情况下了\* 进行换行（新行），请使用 <br /> 标签：

当显示页面时，浏览器会移除**源代码中**多余的**空格和空行**。所有连续的空格或空行都会被算作一个空格。需要注意的是，HTML 代码中的所有**连续的空行（换行）也被显示为一个空格。**

属性

href a标签的href中互联网资源要加上 http:// 也可以加载本地资源

align 剧中什么的

bgcolor 背景色

background 背景图 文件夹下面路径直接上就行了 在body里面

属性是 key value类型 value一般放在双引号里面，如果自带双引号必须放在单引号厘

标题

H1 –H5

浏览器会自动在标题前后添加空行

Html会自动在块级元素前后添加一个空行，比如段落和标题元素前后

搜索引擎使用标题为您的网页的结构和内容编制索引

标题就最好就是标题的属性

水平线

<hr/>

换行：

|  |
| --- |
| 1. <br /> |

不另起一个段落，进行换行。

|  |
| --- |
| 1. <p> 2. 哈<br />哈哈 3. </p> |

网页中99.9999%需要换行的时候，是因为另起了一个段落，所以要用p，不要用<br />

br是英语break打断的意思。

段落

<

<pre>

这是

预格式文本。

它保留了 空格

和换行。

</pre>

html文本格式化（可以认为是内容样式）

**物理样式 丢弃了**

~~<~~**~~b~~**~~>bond加粗</~~**~~b~~**~~>  
<~~**~~i~~**~~>italic 斜体</~~**~~i~~**~~>  
<~~**~~small~~**~~>小一号的</~~**~~small~~**~~>  
<~~**~~big~~**~~>大一号的字</~~**~~big~~**~~>  
下标<~~**~~sub~~**~~>下标subscript</~~**~~sub~~**~~>  
上标<~~**~~sup~~**~~>supscript上标</~~**~~sup~~**~~>~~

~~<ins>下划线，插入~~

~~<del> 文字中画线 删除~~

~~<tt> 标签呈现类似打字机或者等宽的文本效果。~~

语意标签

|  |  |
| --- | --- |
| [<em>](http://www.w3school.com.cn/tags/tag_em.asp) | 把文本定义为强调的内容。 |
| [<strong>](http://www.w3school.com.cn/tags/tag_strong.asp) | 把文本定义为语气更强的强调的内容。 |
| [<dfn>](http://www.w3school.com.cn/tags/tag_dfn.asp) | 定义一个定义项目。 |
| [<code>](http://www.w3school.com.cn/tags/tag_code.asp) | 定义计算机代码文本。 |
| [<samp>](http://www.w3school.com.cn/tags/tag_samp.asp) | 定义样本文本。 |
| [<kbd>](http://www.w3school.com.cn/tags/tag_kbd.asp) | 定义键盘文本。它表示文本是从键盘上键入的。它经常用在与计算机相关的文档或手册中。 |
| [<var>](http://www.w3school.com.cn/tags/tag_var.asp) | 定义变量。您可以将此标签与 <pre> 及 <code> 标签配合使用。 |
| [<cite>](http://www.w3school.com.cn/tags/tag_cite.asp) | 定义引用。可使用该标签对参考文献的引用进行定义，比如书籍或杂志的标题。 |

**地址**

<address>

Written by <a href="mailto:webmaster@example.com">Donald Duck</a>.<br>

Visit us at:<br>

Example.com<br>

Box 564, Disneyland<br>

USA

</address>

定义缩写

缩写，鼠标移动到上面显示全称abbreviation， title显示全部的内容

<**abbr title="etcetera"**> etc.</**abbr**>

**定义首字母缩写。**首字母缩略词

<acronym title="World Wide Web">WWW</acronym>

Bdo 定义文字的方向 rtl从右到左，ltr从左到右

<bdo dir="rtl">Here is some text</bdo>

<blockquote>块引用

**缩进换行加外边距**

**提示：**请使用 [q 元素](http://www.w3school.com.cn/tags/tag_q.asp)来标记短的引用。

**注释：**如需把页面作为 strict XHTML 进行验证，那么 <blockquote> 元素必须包含块级元素，比如这样：

<blockquote>

<p>here is a long quotation here is a long quotation</p>

</blockquote>

短引用

<q>Here is a short quotation here is a short quotation</q>

浏览器经常在引用的内容周围添加引号。

<q> 与 <blockquote> 的区别

<q> 标签在本质上与 [<blockquote>](http://www.w3school.com.cn/tags/tag_blockquote.asp) 是一样的。不同之处在于它们的显示和应用。<q> 标签用于简短的行内引用。如果需要从周围内容分离出来比较长的部分（通常显示为缩进的块），请使用 <blockquote> 标签。

Html样式

**所有的格式化代码均可移出 HTML 文档，然后移入一个独立的样式表**

**样式的使用方法： style定义样式或者rel=“stylesheet”**

**type=”text/css” 在外链和内部的时候指出规则是什么，内嵌不用**

**外部连接样式 表**

<head>

<link rel="stylesheet" type="text/css" href="mystyle.css">

</head>

**内部样式表**

<head>

<style type="text/css">

body {background-color: red}

p {margin-left: 20px}

</style>

</head>

Body 和 后面的p都是样式名

**内嵌样式**

<p style="color: red; margin-left: 20px">

This is a paragraph

</p>

Div

是一个**块级元素**。**换行**是 <div> 固有的唯一格式表现。可以通过 <div> 的 class 或 id 应用额外的样式。

不必为每一个 <div> 都加上类或 id，虽然这样做也有一定的好处。

**可以对同一个 <div> 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种**。这两者的主要差异是，c**lass 用于元素组**（类似的元素，或者可以理解为某一类元素），而 **id 用于标识单独的唯一的元素。**

<span>

标签被用来组合文档中的**行内元素。没有特殊效应但可以定义样式**

可以对同一个 <span> 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。

<font>

规定文本的字体、字体尺寸、字体颜色。

<font size="3" color="red">This is some text!</font>

**建议用样式代替**

Html链接

超文本传输

**文字链接**

<a href="http://www.microsoft.com/">本文本</a>

**图像连接**

<a href="/example/html/lastpage.html">

<img border="0" src="/i/eg\_buttonnext.gif" />

</a>

这样的链接会在图像前加一个空格的，因为有换行，图像属于文本，则前后都会出现短线，正确的做法是 <a><img src=”></a>

A标签属性介绍

**href** 给出网址hypertext reference 超文本引用；

万维网得加上<http://www.w3school.com.cn/> 本网站地址直接鞋/index.html

**target** 给出新标签还是自己页面 \_blank 新页面 \_top 跳出原来的框架？？？？？？

Download 要下载的文件名

<a href="http://www.w3school.com.cn/" target="\_blank">Visit W3School!</a>

<a href="/images/myw3schoolimage.jpg" download="w3logo">

锚

**也可以用id代替name**

首先，我们在 HTML 文档中对锚进行命名（创建一个书签）：

<a name="tips">基本的注意事项 - 有用的提示</a>

然后，我们在同一个文档中创建指向该锚的链接：

<a href="#tips">有用的提示</a>

您也可以在其他页面中创建指向该锚的链接：

<a href="http://www.w3school.com.cn/html/html\_links.asp#tips">有用的提示</a>

在上面的代码中，我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了

**命名锚经常用于在大型文档开始位置上创建目录。**

邮件 ？？？？

<p>

这是邮件链接：

<a href="mailto:someone@microsoft.com?subject=Hello%20again">发送邮件</a>

</p>

**提示和注释**

**提示：**如果不使用 href 属性，则不可以使用如下属性：download, hreflang, media, rel, target 以及 type 属性。

**提示：**被链接页面通常显示在当前浏览器窗口中，除**非您规定了另一个目标（target 属性）**。

**提示：**请使用 CSS 来设置链接的样式。

**A是一个文本级的标签**

A能放在p里面，但p不能放在里面，可以把a直接当作文本处理

Html图像

<img src=””>

**图片来自本地文件**

<img src="/i/eg\_mouse.jpg" width="128" height="128" />

<**img src=".imag/girl.jpg"**/>

本地的网站，路径是从该html的所在目录开始的， ../上一级的

当图片在文件夹里面的时候：

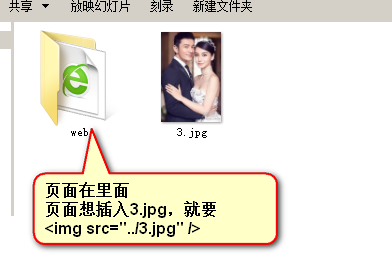


|  |
| --- |
| 1. <img src="**images/**baby.jpg" alt="巴黎结婚照" /> |

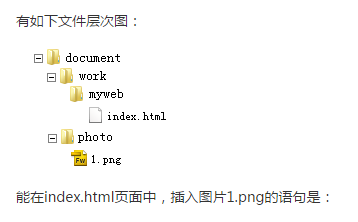
如果在很深的文件夹中，也不怕，可以一直罗列下去：

|  |
| --- |
| 1. <img src="**images/jiehunzhao/**baby.jpg" alt="巴黎结婚照" /> |

如果情况变得复杂，我们的图片在浅一层的文件夹中：



面试题



标准答案：

|  |
| --- |
| 1. <img src="../../photo/1.png" /> |

**来自万维网**

<img src="http://www.w3school.com.cn/i/w3school\_logo\_white.gif" />

**动态图和静态图插入一样的**

浏览器将图像显示在文档中图像标签出现的地方。如果你将图像标签置于两个段落之间，那么浏览器会首先显示第一个段落，然后显示图片，最后显示第二段。

文件无法加载的时候，alt 的值显示出来

<img src="boat.gif" alt="Big Boat">

**图像的排列位置**

<p>图像 <img src="/i/eg\_cute.gif" align="bottom"> 在文本</p>

图像底部和文本框的底部对齐，

<p>图像 <img src ="/i/eg\_cute.gif" align="middle"> 在文本</p>

图像中部和文本框的底部对齐，

<p>图像 <img src ="/i/eg\_cute.gif" align="top"> 在文本</p>

图像的顶部和文本狂对其

<p>请注意，bottom 对齐方式是默认的对齐方式。</p>

<p>

<img src ="/i/eg\_cute.gif" align ="left">

带有图像的一个段落。图像的 align 属性设置为 "left"。图像将浮动到文本的左侧。

</p>

<p>

<img src ="/i/eg\_cute.gif" align ="right">

带有图像的一个段落。图像的 align 属性设置为 "right"。图像将浮动到文本的右侧。

</p>

**图片的大小**

通过改变 img 标签的 "height" 和 "width" 属性的值，您可以放大或缩小图像。

背景图片

<**body background=".idea/imag/girl.jpg"**>

**Gif 和jpg均可以作为html的背景，**

如果图像小于页面，图像就会重复

**图像映射（图片内部有各种可以点的超链接）**

<img src="planets.jpg" border="0" usemap="#planetmap" alt="Planets" />

<map name="planetmap" id="planetmap">

<area shape="circle" coords="180,139,14" href ="venus.html" alt="Venus" />

<area shape="circle" coords="129,161,10" href ="mercur.html" alt="Mercury" />

<area shape="rect" coords="0,0,110,260" href ="sun.html" alt="Sun" />

</map>

**注意在 area中可以定义 target ="\_blank"**

注释：</b>img 元素中的 "usemap" 属性引用 map 元素中的 "id" 或 "name" 属性（根据浏览器），所以我们同时向 map 元素添加了 "id" 和 "name" 属性**。**

nohref 属性规定该区域没有相关的链接。 Href 指出链接位置

|  |
| --- |
| **coords 描述 坐标** |
| *x1,y1,x2,y2* | 如果 shape 属性设置为 "rect"，则该值规定矩形左上角和右下角的坐标。 |
| *x,y,radius* | 如果 shape 属性设置为 "circ"，则该值规定圆心的坐标和半径。 |
| *x1,y1,x2,y2,..,xn,yn* | 如果 shape 属性设置为 "poly"，则该值规定多边形各边的坐标。如果第一个坐标和最后一个坐标不一致，那么为了关闭多边形，浏览器必须添加最后一对坐标 |

Html表格

表格 table

表格的行 tr （table row）

表格数据td （table date）

文本、图片、列表、段落、表单、水平线、表格

表头 th table head

空单元格里面写 &nbsp;

**标题**<caption>我的标题</caption> caption 在tr上 在table中

**跨两列** 不写第三个td就行了

<tr>

<th>姓名</th>

<th **colspan="2"**>电话</th>

</tr>

**跨两行 被夸的部分对应的td不写**

<tr>

<th rowspan="2">电话</th>

<td>555 77 854</td>

</tr>

<tr>

<td>555 77 855</td>

</tr>

Table的属性 <table>

Border 边框粗细 0 无 越大月粗

Cellpadding 内容距离边框的距离

Cellspacing 各个单元格之间的间距

Bgcolor 背景色

Background 背景图片

Frame 一些方向上的边框显示

Td属性

Align 内容的位置

Bgcolor

Background

细分的table内的标签，三个必须一起出现

Thead <thead> 内部必须拥有 <tr> 标签！

Tbody

Tfoot

这种划分使浏览器有能力支持独立于表格标题和页脚的表格正文滚动。当长的表格被打印时，表格的表头和页脚可被打印在包含表格数据的每张页面上

内部以char 的字幕对其时，没有支持的浏览器

<thead align="char" char="M" charoff="2">

属性

[valign](http://www.w3school.com.cn/tags/att_tbody_valign.asp) 垂直方向上的对齐方式 top middle bottom baseline

align 水平内容对其方式 right left center justify整理版面 char

**colgroup** 可以跨列定义样式用

<table width="100%" border="1">

<colgroup span="2" align="left"></colgroup>

<colgroup align="right" style="color:#0000FF;"></colgroup>

<tr

Span colgrpoup 跨的元素

**Width**

|  |  |
| --- | --- |
| *pixels* | 设置以像素计的宽度值（例子：width="50"）。 |
| *percent* | 设置以包围元素的百分比计的宽度值（例子：width="50%"）。 |

假如表格的宽度是 100 像素，第一个列组是 20 像素，第二个列组是 50%，那么剩余的可用像素是 30 像素。如果可用像素是 30px，那么可以设置部分一为 "1\*"，部分二为 "2\*"

**Col** 定义每一列的样式

<table width="100%" border="1">

<col align="left" />

<col align="left" />

<col align="right" />

<tr>

Html列表

**无序列表 前面的点的样式 type 定**

<ul type="circle"> difined 设置的前面的点或者什么的

<li>咖啡</li>

<li>茶</li>

<li>牛奶</li>

</ul>

**li是一个容器级标签，li里面什么都能放**

**有序列表** order list 用的少，多数是用ul在前面加上标签即可

<ol start="50">

<li>咖啡</li>

<li>牛奶</li>

<li>茶</li>

</ol>

默认是数字序列的

type="A" 或者 type=”a” 序列变成 字母的

可以嵌套定义

**自定义列表**

dl表示definition list 定义列表

dt表示definition title 定义标题

dd表示definition description 定义表述词儿

dt、dd都是容器级标签，想放什么都可以。所以，现在就应该更加清晰的知道：

**用什么标签，不是根据样子来决定，而是语义**

**<dl>**

<**dt**>计算机</dt>

**<dd**>用来计算的仪器 ... ...</dd>

<dt>显示器</dt>

<dd>以视觉方式显示信息的装置 ... ...</dd>

</dl>

Html的块

<div>

可以通过div 和 span 将html元素组合起来

<div>用于文档布局 和css一起使用，加上id或者class属性

**div 元素的 "align" 属性不被支持。**

div在浏览器中，**默认是不会增加任何的效果改变的**，但是语义变了，div中的所有元素是一个小区域。**div标签是一个容器级标签，**里面什么都能放，甚至可以放div自己

<span>

内联元素，用于**文本的容器**，与css一起使用时，可用于**部分文本设置样式属性**

**span里面只能放置文字、图片、表单元素。 span里面不能放p、h、ul、dl、ol、div。**

Html的布局 ？？

没看懂div如何变化的

<**html**>  
<**head**>  
 <**style type="text/css"**>  
 **div#container**{**width**:500**px**}  
 **div#header** {**background-color**:**#99bbbb**;}  
 **div#menu** {**background-color**:**#ffff99**;**height**:200**px**;**width**:150**px**;**float**:**left**;}  
 **div#content** {**background-color**:**#EEEEEE**;**height**:200**px**;**width**:350**px**;**float**:**left**;}  
 **div#footer** {**background-color**:**#99bbbb**;**clear**:**both**;**text-align**:**center**;}  
 **h1** {**margin-bottom**:0;}  
 **h2** {**margin-bottom**:0;**font-size**:18**px**;}  
 **ul** {**margin**:0;}  
 **li** {**list-style**:**none**;}  
 </**style**>  
</**head**>  
<**body**>  
<**div id="container"**>  
 <**div id="header"**>  
 <**h1**>Main Title of Web Page</**h1**>  
 </**div**>  
 <**div id="menu"**>  
 <**h2**>Menu</**h2**>  
 <**ul**>  
 <**li**>HTML</**li**>  
 <**li**>CSS</**li**>  
 <**li**>JavaScript</**li**>  
 </**ul**>  
 </**div**>  
 <**div id="content"**>Content goes here</**div**>  
 <**div id="footer"**>Copyright W3School.com.cn</**div**>  
</**div**>  
</**body**>  
</**html**>

html表单

action 提交的地址 method 提交方式

<form name="input" action="html\_form\_action.asp" method="get">

**Enctype encoding type**

enctype="text/plain"

enctype 属性规定在发送到服务器之前应该如何对表单数据进行编码。

默认地，表单数据会编码为 "application/x-www-form-urlencoded"。就是说，在发送到服务器之前，所有字符都会进行编码（空格转换为 "+" 加号，特殊符号转换为 ASCII HEX 值）

|  |  |
| --- | --- |
| **值** | **描述** |
| application/x-www-form-urlencoded | 在发送**前编码所有字符（默认）** |
| multipart/form-data | 不对字符编码。  在使用**包含文件上传控件的表单时**，必须使用该值。 |
| text/plain | 空格转换为 "+" 加号，但不对特殊字符编码。 |

**Novalidate**

novalidate="novalidate" 提交表单的时候不进行认证

输入

Type

属性值

|  |  |
| --- | --- |
| **值** | **描述** |
| button | 定义可点击按钮（多数情况下，用于通过 JavaScript 启动脚本）。 |
| checkbox | 定义复选框。 |
| file | **定义输入字段和 "浏览"按钮，供文件上传**。 |
| hidden | 定义隐藏的输入字段。 |
| image | 定义图像形式的提交按钮。 |
| password | 定义密码字段。该字段中的字符被掩码。 |
| radio | 定义单选按钮。 |
| reset | 定义重置按钮。重置按钮会清除表单中的所有数据。 |
| submit | 定义提交按钮。提交按钮会把表单数据发送到服务器。 |
| text | 定义单行的输入字段，用户可在其中输入文本。默认宽度为 20 个字符。 |

文本域

Name:

<input type=”text” name=”name” value="yourmail" size="20"/> 可以添加值类似于hint

Password:

<input type=”password” name=”password”/>

文本域默认缺省宽度 20 字符

单选按钮

input radio 相同的name 不同的 value

<input type="radio" name="sex" value="male" /> Male

<input type="radio" name="sex" value="female" checked="checked"/> Female

下拉框

**select name option value**

<select name="cars">

<option value="volvo" **selected**=”selected”>Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

**Optgroup**

<select>

<**optgroup** label="Swedish Cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

</optgroup>

<optgroup label="German Cars">

<option value="mercedes">Mercedes</option>

<option value="audi">Audi</option>

</optgroup>

</select>

复选框

input **checkbox** **不同的name**

<input type="checkbox" name="bike" />

I have a bike

<br />

<input type="checkbox" name="car" />

I have a car

<lebel>

**在单选和复选框中必须定义id ，然后在 lable中for属性指向该id ，并且必须定义value**

<**input type="radio" name="sex" id="famale" value="famale"**/><**label for="famale"**>女</**label**>

<**input type="radio" name="sex" ID="male" value="male"** /><**label for="male"**>男</**label**>

任何表单都有label

也可以

<lable>姓名

<input type="text">

</lable>

提交

**input submit value**

<input type="submit" value="Submit" />

文本域

Textarea rows cols

**<textarea rows="10" cols="30" name=“”” resize=”none”></textarea>里面的文字是默认的文字**

**Resize=none 防止用户拖动**

按钮

<**input type="button" value="什么东西"**>

表单周围的框

**<fieldset>**

本例演示如何在数据周围绘制一个带标题的框。

**<fieldset>**

**<legend>**健康信息</legend>

身高：<input type="text" />

体重：<input type="text" />

</fieldset>

**电子邮件**

<form action="MAILTO:someone@w3school.com.cn" method="post" **enctype**="text/plain">

上传文件

**file name id accept （上传文件的属性）**

<input type="file" name="pic" id="pic" accept="image/gif, image/jpeg" />

如果不限制图像的格式，可以写为：accept="image/\*"。 记得用逗号隔开文件内容

autocomplete

自动完成（补全，记录历史）功能

自动补全 on 在第二次输入的时候会有下拉选项的

form 和 <input> 类型：text, search, url, telephone, email, password, datepickers, range 以及 color。 属性中可以使用这个

<**form action="/example/html5/demo\_form.asp" method="get" autocomplete="on"**>  
 First name:<**input type="text" name="fname"** /><**br** />  
 Last name: <**input type="text" name="lname"** /><**br** />  
 E-mail: <**input type="email" name="email" autocomplete="off"** /><**br** />  
 <**input type="submit"** />  
</**form**>

autofocus

自动获取焦点

文本输入字段被设置为当页面加载时获得焦点

<input type="text" name="fname" autofocus="autofocus" />

Disabled属性

被禁用的 input 元素既不可用，也不可点击。可以设置 disabled 属性，直到满足某些其他的条件为止（比如选择了一个复选框等等）。然后，就需要通过 JavaScript 来删除 disabled 值，将 input 元素的值切换为可用。

disabled 属性无法与 <input type="hidden"> 一起使用。

<input type="text" name="lname" disabled="disabled" />

Form属性 form="form1"

指向自己应该属于的表单id，可以多个

<form action="/example/html5/demo\_form.asp" method="get" id="form1">

First name: <input type="text" name="fname" /><br />

<input type="submit" value="提交" />

</form>

<p>下面的 "Last name" 字段位于 form 元素之外，但仍然是表单的一部分。</p>

Last name: <input type="text" name="lname" form="form1" />

Formaction

该属性适用于 type="submit" 以及 type="image"。的 input标签，目的是在标签内部提交到不同的网址， 造成权限什么的

Formenctype

formenctype="multipart/form-data"

formmethod

formmethod 属性覆盖 form 元素的 [method](http://www.w3school.com.cn/tags/att_form_method.asp) 属性。

可以通过以下方式发送 form-data ：

* 以 URL 变量 (使用 method="get") 的形式来发送
* 以 HTTP post (使用 method="post") 的形式来发送

该属性与 type="submit" 以及 type="image" 配合使用。

formnovalidate

formnovalidate 属性覆盖 form 元素的 [novalidate](http://www.w3school.com.cn/tags/att_form_novalidate.asp) 属性。

如果使用该属性，则提交表单时按钮不会执行验证过程。

该属性适用于 <form> 以及以下类型的 <input>：text, search, url, telephone, email, password, date pickers, range 以及 color。

Formtaget

formtarget 属性覆盖 form 元素的 [target](http://www.w3school.com.cn/tags/att_form_target.asp) 属性。

该属性与 type="submit" 以及 type="image" 配合使用。

**注释：**HTML5 不支持框架和框架集。现在，parent, top 和 *framename* 值大多用于 iframe。

Type= imae 可以做submmit使用，

并且默认地，image 输入类型会发生点击图像按钮时的 X 和 Y 坐标，可以定义width和heigth

<form action="/example/html5/demo\_form.asp" method="get">

First name: <input type="text" name="fname" /><br />

Last name: <input type="text" name="lname" /><br />

<input type="image" src="/i/eg\_submit.jpg" alt="Submit" width="128" height="128"/>

</form>

<p><b>注释：</b>默认地，image 输入类型会发生点击图像按钮时的 X 和 Y 坐标。</p>

List datalist

**list=“datalist的id“ datalist标签（和select有点象） 属性lable（位于列表右边显示是什么的书属性）**

网页：

<input type="url" list="url\_list" name="link" />

<**datalist** id="url\_list">

<option **label="W3School"** value="http://www.w3school.com.cn" />

<option label="Google" value="http://www.google.com" />

<option label="Microsoft" value="http://www.microsoft.com" />

</datalist>

Max和min

max 属性规定输入字段所允许的最大值。

**提示：**max 属性与 [min](http://www.w3school.com.cn/tags/att_input_min.asp) 属性配合使用，可创建合法值范围。

**注释：**max 和 min 属性适用于以下 <input> 类型：number, range, date, datetime, datetime-local, month, time 以及 week

可选属性值为 number date

Maxlenth

maxlength 属性规定输入字段的最大长度，以字符个数计。

maxlength 属性与 <input type="text"> 或 <input type="password"> 配合使用。

Panttern

定义和用法

pattern 属性规定用于验证输入字段的模式。

模式指的是[正则表达式](http://www.w3school.com.cn/js/js_obj_regexp.asp)。您可以在我们的 JavaScript 教程中阅读到这方面的内容。

**注释：**pattern 属性适用于以下 <input> 类型：text, search, url, telephone, email 以及 password 。

**提示：**请使用标准的 "title" 属性来描述模式。

[placeholder](http://www.w3school.com.cn/tags/att_input_placeholder.asp)

hint

**readonly**

只读字段是不能修改的。不过，用户仍然可以使用 tab 键切换到该字段，还可以选中或拷贝其文本。

readonly 属性可以防止用户对值进行修改，直到满足某些条件为止（比如选中了一个复选框）。然后，需要使用 JavaScript 消除 readonly 值，将输入字段切换到可编辑状态。

readonly 属性可与 <input type="text"> 或 <input type="password"> 配合使用。

Size

对于 <input type="text"> 和 <input type="password">，size 属性定义的是可见的字符数。而对于其他类型，size 属性定义的是以像素为单位的输入字段宽度。

由于 size 属性是一个可视化的设计属性，我们推荐您使用 CSS 来代替它。

CSS 语法：<input style="width:100px" /

Step

step 属性规定输入字段的合法数字间隔（假如 step="3"，则合法数字应该是 -3、0、3、6，以此类推）。

**提示：**step 属性可以与 max 以及 min 属性配合使用，以创建合法值的范围。

**注释：**step、max 以及 min 属性适用于以下 <input> 类型：number, range, date, datetime, datetime-local, month, time 以及 week。

会出现上下的箭头

**Required**

required="required"

required 属性规定必需在提交之前填写输入字段。没填就不能提交

如果使用该属性，则字段是必填（或必选）的。

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

|  |  |  |
| --- | --- | --- |
| [name](http://www.w3school.com.cn/tags/att_input_name.asp) | field\_name | 定义 input 元素的名称。 |
| [pattern](http://www.w3school.com.cn/tags/att_input_pattern.asp) | *regexp\_pattern* | 规定输入字段的值的模式或格式。  例如 pattern="[0-9]" 表示输入值必须是 0 与 9 之间的数字。 |
| [placeholder](http://www.w3school.com.cn/tags/att_input_placeholder.asp) | *text* | 规定帮助用户填写输入字段的提示。 Hint |
| [readonly](http://www.w3school.com.cn/tags/att_input_readonly.asp) | readonly | 规定输入字段为只读。 |
| [required](http://www.w3school.com.cn/tags/att_input_required.asp) | required | 指示输入字段的值是必需的。 |
| [size](http://www.w3school.com.cn/tags/att_input_size.asp) | number\_of\_char | 定义输入字段的宽度。 |
| [src](http://www.w3school.com.cn/tags/att_input_src.asp) | URL | 定义以提交按钮形式显示的图像的 URL。 |
| [step](http://www.w3school.com.cn/tags/att_input_step.asp) | *number* | 规定输入字的的合法数字间隔。 |
| [type](http://www.w3school.com.cn/tags/att_input_type.asp) | * button * checkbox * file * hidden * image * password * radio * reset * submit * text | 规定 input 元素的类型。 |
| [value](http://www.w3school.com.cn/tags/att_input_value.asp) | *value* | 规定 input 元素的值。 在text中是直接填写的的，不是hint |

Html框架

不能将 <body></body> 标签与 <frameset></frameset> 标签同时使用！不过，假如你添加包含一段文本的 <noframes> 标签，就必须将这段文字嵌套于 <body></body> 标签内。（在下面的第一个实例中，可以查看它是如何实现的。

**Frameset 框架结合标签**

里面有rows分成几行 cols 分成几列

cols="25%,50%,25%" cols="120,\* “ 定义宽度的方法

**frame 框架标签**

src=”“ 指出里面内联的网址 name 属性定义一个name属性， 在其他的跳转 target选项

noresize="noresize" 不允许手动调节大小

**frameset中的noframe标签 浏览器无法显示frame的时候显示，必须加上<body>标签**

<frameset cols="25%,50%,25%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

<noframes>

<body>您的浏览器无法处理框架！</body>

</noframes>

</frameset>

**分成三列得到框架**

<frameset cols="25%,50%,25%">

<frame src="/example/html/frame\_a.html">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

**混合框架**

<frameset rows="50%,50%">

<frame src="/example/html/frame\_a.html">

<frameset cols="25%,75%">

<frame src="/example/html/frame\_b.html">

<frame src="/example/html/frame\_c.html">

</frameset>

</frameset>

html内联框架

**iframe 标签 在任意位置开辟单独的空间来显示网页**

<iframe src="demo\_iframe.htm" width="200" height="200"></iframe>

**height 和 width** 属性用于规定 iframe 的高度和宽度。

属性值的默认单位是像素，但也可以用百分比来设定（比如 "80%"）。

删除边框

frameborder="0"

name属性作为外链显示的target

<iframe src="demo\_iframe.htm" name="iframe\_a"></iframe>

<p><a href="http://www.w3school.com.cn" target="iframe\_a">W3School.com.cn</a></p>

Html背景

背景颜色（Bgcolor）

<body bgcolor="#000000">

背景（Background）

<body background="clouds.gif">

<body background="http://www.w3school.com.cn/clouds.gif">

**Bacground 会导致尺寸不够的时候红蝶显示**

**提示：**如果你打算使用背景图片，你需要紧记一下几点：

* 背景图像是否增加了页面的加载时间。小贴士：图像文件不应超过 10k。
* 背景图像是否与页面中的其他图象搭配良好。
* 背景图像是否与页面中的文字颜色搭配良好。
* 图像在页面中平铺后，看上去还可以吗？
* 对文字的注意力被背景图像喧宾夺主了吗

<script> 元素

<script> 标签用于定义客户端脚本，比如 JavaScript。

html脚本

**script**

<script type="text/javascript">

document.write("Hello World!")

</script>

<script> 标签用于定义客户端脚本，比如 JavaScript。

script 元素既可以包**含脚本语句**，也可以通过 **src 属性指向外部脚本文件。**

必需的 **type 属性规定脚本的 MIME 类型。**

JavaScript 的常见应用时**图像操作、表单验证以及动态内容更新。**

假如此元素内部的**代码没有位于某个函数中，那么这些代码会在页面被加载时被立即执行。<frameset> 标签之后的脚本会被忽略。**

**HTML 4 和 XHTML 在处理脚本中的内容方面有所不同：**

在 HTML 4 中，内容类型声明为 CDATA，就是说不会对实体进行解析。

在 XHTML 中，内容类型声明为 (#PCDATA)，也就是说会对实体进行解析。

这意味着，在 XHTML 中，应该编码所有特殊的字符，或者把所有内容嵌套在 CDATA 部分中。

为了确保在 XHTML 文档中脚本正确进行解析，

**noscript**

<noscript>Sorry, your browser does not support JavaScript!</noscript>

**async="async" 外部脚本异步进行**

<script type="text/javascript" **src**="demo\_async.js" **async="async"**></script>

**Charset**

<script type="text/javascript" src="myscripts.js" charset="UTF-8"></script>

* UTF-8 - Unicode 字符编码
* ISO-8859-1 - 拉丁字母表的字符编码

**Defer**  是否加载结束后才加载

Html实体

|  |  |  |  |
| --- | --- | --- | --- |
| **显示结果** | **描述** | **实体名称** | **实体编号** |
|  | 空格 | &nbsp; | &#160; |
| < | 小于号 | &lt; | &#60; |
| > | 大于号 | &gt; | &#62; |
| & | 和号 | &amp; | &#38; |
| " | 引号 | &quot; | &#34; |
| ' | 撇号 | &apos; (IE不支持) | &#39; |
| ￠ | 分 | &cent; | &#162; |
| £ | 镑 | &pound; | &#163; |
| ¥ | 日圆 | &yen; | &#165; |
| € | 欧元 | &euro; | &#8364; |
| § | 小节 | &sect; | &#167; |
| © | 版权 | &copy; | &#169; |
| ® | 注册商标 | &reg; | &#174; |
| ™ | 商标 | &trade; | &#8482; |
| × | 乘号 | &times; | &#215; |
| ÷ | 除号 | &divide; | &#247; |

Html统一资源定位器 URL 网址

http 超文本传输协议 以 http:// 开头的普通网页。不加密。

https 安全超文本传输协议 安全网页。加密所有信息交换。

ftp 文件传输协议 用于将文件下载或上传至网站。

file 您计算机上的文件。

**scheme://host.domain:port/path/filename**

scheme - 定义因特网服务的类型。最常见的类型是 http

host - 定义域主机（http 的默认主机是 www）

domain - 定义因特网域名，比如 w3school.com.cn

:port - 定义主机上的端口号（http 的默认端口号是 80）

path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。

filename - 定义文档/资源的名

**URL 编码**

URL 只能使用 ASCII 字符集来通过因特网进行发送。

由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格

**Html Web Serve**

<http://www.w3school.com.cn/hosting/index.asp>

网站主机教程

Html 多媒体

多媒体值得是 音效 音乐 视频 动画

**视频格式**

.avi window常用，非win不好用

.wmv . windows常用，非win不能用

.mpg .mpeg 跨平台支持所有浏览器

.mov ios常用，win要装插件

.ram .rm **低宽带**优选，质量差

.swf .flv flash 插件

**.mp4**

**声音格式**

.mid .midi 小巧的数字方法的，广泛支持

Rm ram 低宽带

.wav 除了谷歌都支持,**最受欢迎，**无压缩的音乐和演讲用

.mp3 .mpga 通那个用，**压缩**录制音乐格式

Html对象

**<object>作用是支持html助手（插件或者辅助应用程序,连入）,**

**Quicktime播放 wave，播放mp4**

<object width="420" height="360"

classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"

codebase="http://www.apple.com/qtactivex/qtplugin.cab">

<param name="src" **value="bird.wav"** />

<param name="controller" value="true" />

</object>

**Flash播放 .swf**

<object width="400" height="40"

classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/

pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">

<param name="src" **value="bookmark.swf"**>

<embed **src="bookmark.swf"** width="400" height="40"></embed>

</object>

**Window Media Player 播放WMV 影片**

<object width="100%" height="100%"

type="video/x-ms-asf" url=**"3d.wmv"** data=**"3d.wmv"**

classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">

<param name="url" value="3d.wmv">

<param name="filename" value="3d.wmv">

<param name="autostart" value="1">

<param name="uiMode" value="full" />

<param name="autosize" value="1">

<param name="playcount" value="1">

<embed type="application/x-mplayer2" src="3d.wmv" width="100%"

height="100%" autostart="true" showcontrols="true"

pluginspage="http://www.microsoft.com/Windows/MediaPlayer/"></embed>

</object>

HRML 音频

**<!DOCTYPE html> 一定开头声明要使用**

使用 <embed> 元素，定义外部（非 HTML）内容的容器。

<**embed** height="100" width="100" **src="song.mp3"** />

<Object > 元素

<object height="100" width="100" **data="song.mp3**"></object>

使用<audio>元素 ,只支持3种列表 MP3 ogg wav

<audio controls="controls">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<source src="song.wave" type="audio/wav" />

</audio>

<audio> 标签的属性

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| [autoplay](http://www.w3school.com.cn/tags/att_audio_autoplay.asp) | autoplay | 如果出现该属性，则音频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/tags/att_audio_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [loop](http://www.w3school.com.cn/tags/att_audio_loop.asp) | loop | 如果出现该属性，则每当音频结束时重新开始播放。 |
| [preload](http://www.w3school.com.cn/tags/att_audio_preload.asp) | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/tags/att_audio_src.asp) | *url* | 要播放的音频的 URL。 |

最好的html解决方法

audio embed

MP3 ogg wav

<audio controls="controls" height="100" width="100">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<embed height="100" width="100" src="song.mp3" />

</audio>

2.雅虎媒体播放器

直接链接就能播放但是要保证script标签内的内容存在

<a href="song.mp3">Play Sound</a>

下面是要保证的标签

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js">

</script>

3.使用超链接（播放器会自动用插件播放）

<a href="song.mp3">Play the sound</a>

Html视频

**<!DOCTYPE html>**

**<embed> 标签**

<embed src="movie.swf" height="200" width="200"/>

**<object>标签**

<object width="320" height="240" data="/i/bookmark.swf"></object>

**<video>标签** Vadio支持ogg，mp4,webm 三种格式

<video width="320" height="240" controls="controls" autoplay="autoplay">

<source src="/i/movie.ogg" type="video/ogg" />

<source src="/i/movie.mp4" type="video/mp4" />

<source src="/i/movie.webm" type="video/webm" />

</video>

Vadio支持ogg，mp4,webm 三种格式

Ogg = 带有 Theora 视频编码和 Vorbis 音频编码的 Ogg 文件

MPEG4 = 带有 H.264 视频编码和 AAC 音频编码的 MPEG 4 文件

WebM = 带有 VP8 视频编码和 Vorbis 音频编码的 WebM 文件

<video> 与 </video> 之间插入的内容非<source>是供不支持 video 元素的浏览器显示的

Vadio的属性

|  |  |  |  |
| --- | --- | --- | --- |
| [autoplay](http://www.w3school.com.cn/tags/att_video_autoplay.asp) | autoplay | 如果出现该属性，则视频在就绪后马上播放。 | |
| [controls](http://www.w3school.com.cn/tags/att_video_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 | |
| [height](http://www.w3school.com.cn/tags/att_video_height.asp) | *pixels* | 设置视频播放器的高度。 | |
| [loop](http://www.w3school.com.cn/tags/att_video_loop.asp) | loop | 如果出现该属性，则当媒介文件完成播放后再次开始播放。 | |
| [preload](http://www.w3school.com.cn/tags/att_video_preload.asp) | preload | 如果出现该属性，则视频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 | |
| [src](http://www.w3school.com.cn/tags/att_video_src.asp) | *url* | 要播放的视频的 URL。 | |
| [width](http://www.w3school.com.cn/tags/att_video_width.asp) | *pixels* | 设置视频播放器的宽度。 | |
| **方法** | | **属性** | **事件** |
| play() | | currentSrc | play |
| pause() | | currentTime | pause |
| load() | | videoWidth | progress |
| canPlayType | | videoHeight | error |
|  | | duration | timeupdate |
|  | | ended | ended |
|  | | error | abort |
|  | | paused | empty |
|  | | muted | emptied |
|  | | seeking | waiting |
|  | | volume | loadedmetadata |
|  | | height |  |
|  | | width |  |

**注释：**在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

最好的html解决方法

1.组合框架

<**video** width="320" height="240" controls="controls">

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

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

<**source** **src**="movie.webm" type="video/webm" />

<**object** **data**="movie.mp4" width="320" height="240">

<**embed** **src**="movie.swf" width="320" height="240" />

</object>

</video

2.优酷解决方案

<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v.swf"

width="480" height="400"

type="application/x-shockwave-flash">

</embed>

Xhtml

**Xhtml 可扩展超文本标签语言**

语法：

属性名必须小写（所有的东西都是小写的）

属性值必须加引号

属性不能简写

**用id 属性代替那么属性**

**DTD定义了强制使用的html元素**

**HTML 4.01 针对下列元素定义 name 属性：a, applet, frame, iframe, img, 和map。**

**这些东西没有name属性**

**重要的兼容性提示：**

你应该在 "/" 符号前添加一个额外的空格，以使你的 XHTML 与当今的浏览器相兼容。

**强制使用的xhtml 元素**

强制使用的 XHTML 元素

所有 XHTML 文档必须进行文件类型声明（DOCTYPE declaration）。在 XHTML 文档中必须存在html、head、body元素，而 title 元素必须位于在 head 元素中。

下面是一个最小化的 XHTML 文件模板：

<!DOCTYPE Doctype goes here>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Title goes here</title>

</head>

<body>

</body>

</html>

xHtml文档声明类型

**Strict**

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

在此情况下使用：需**要干净的标记，避免表现上的混乱。请与层叠样式表配合使用。**

**Transitional**

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

在此情况下使用：**当需要利用 HTML 在表现上的特性时，并且当需要为那些不支持层叠样式表的浏览器编写 XHTML 时。**

**Frameset**

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

在此的情况下使用：**需要使用HTML框架将浏览器窗口分割为两部分或更多框架时**。

升级到xhtml

1.添加文档类型声明

## CSS

**解决内容与表现分离的问题**

选择器相关

各种样式定义的优先级

1. 浏览器缺省设置
2. 外部样式表
3. 内部样式表（位于 <head> 标签内部）
4. 内联样式（在 HTML 元素内部）

Css基础语法

一些规范

4个空格作为一个缩进层级

空格

1.选择器 与 { 之间必须包含空格

2.属性名 与之后的 : 之间不允许包含空格， : 与 属性值 之间必须包含空格。

3.列表型属性值 书写在单行时，, 后必须跟一个空格

选择器

1.当集体声明多个 selector 时，每个选择器声明必须独占一行。

2. >、+、~ 选择器的两边各保留一个空格

1.选择器的嵌套层级应不大于 3 级，位置靠后的限定条件应尽可能精确。

2. 在可以使用缩写的情况下，尽量使用属性缩写。

3. 使用 border / margin / padding 等缩写时，应注意隐含值对实际数值的影响，确实需要设置多个方向的值时才使用缩写

4.文本内容必须用双引号包围。

5.当数值为 0 - 1 之间的小数时，省略整数部分的 0。

6.长度为 0 时须省略单位。 (也只有长度单位可省)

7.url() 函数中的路径不加引号。

8.RGB颜色值必须使用十六进制记号形式 #rrggbb。不允许使用 rgb()

9.颜色值可以缩写时，必须使用缩写形式。

10. 颜色值不允许使用命名色值。

11. 颜色值中的英文字符采用小写。如不用小写也需要保证同一项目内保持大小写一致。

12. font-weight 属性必须使用数值方式描述。

13. 层级(z-index)必须清晰明确，页面弹窗、气泡为最高级（最高级为999），不同弹窗气泡之间可在三位数之间调整

Css检测地址

http://jigsaw.w3.org/css-validator/

**选择器+声明**

p { color: #f00; }

**值为若干个单词的时候要加引号**

p {font-family: "sans serif";}

**多重声明（每一行结尾都加上;）**

p {

text-align: center;

color: black;

font-family: arial;

}

**空格和大小写**

如果涉及到与 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的

**选择器的分组**

你可以对选择器进行分组，这样，被分组的选择器就可以分享相同的声明。用**逗号**将需要分组的选择器分开。在下面的例子中，我们对所有的标题元素进行了分组。所有的标题元素都是绿色的。

h1,h2,h3,h4,h5,h6 {

color: green;

}

选择器

后代选择器

<li><strong>我是斜体字。这是因为 strong 元素位于 li 元素内。</strong></li>

li strong {

font-style: italic;

font-weight: normal;

}

具体应用

后代选择器的功能极其强大。有了它，可以使 HTML 中不可能实现的任务成为可能。

假设有一个文档，其中有一个边栏，还有一个主区。边栏的背景为蓝色，主区的背景为白色，这两个区都包含链接列表。不能把所有链接都设置为蓝色，因为这样一来边栏中的蓝色链接都无法看到。

解决方法是使用后代选择器。在这种情况下，可以为包含边栏的 div 指定值为 sidebar 的 class 属性，并把主区的 class 属性值设置为 maincontent。然后编写以下样式：

div.sidebar {background:blue;}

div.maincontent {background:white;}

div.sidebar a:link {color:white;}

div.maincontent a:link {color:blue;}

有关**后代选择器**有一个易被忽视的方面，**即两个元素之间的层次间隔可以是无限的。**

例如，如果写作 ul em，这个语法就会选择从 ul 元素继承的所有 em 元素，而不论 em 的嵌套层次多深。

因此，ul em 将会选择以下标记中的所有 em 元素：

<ul>

<li>List item 1

<ol>

<li>List item 1-1</li>

<li>List item 1-2</li>

<li>List item 1-3

<ol>

<li>List item 1-3-1</li>

<li>List item <em>1-3-2</em></li>

<li>List item 1-3-3</li>

</ol>

</li>

<li>List item 1-4</li>

</ol>

</li>

<li>List item 2</li>

<li>List item 3</li>

</ul>

**>子元素选择器（Child selectors）**

**只能选择作为某元素子元素的元素，直接子元素**

如果您希望选择只作为 h1 元素子元素的 strong 元素，可以这样写：

**h1 > strong** {color:red;}

这个规则会把第一个 h1 下面的两个 strong 元素变为红色，但是第二个 h1 中的 strong 不受影响：

<h1>This is <strong>very</strong> <strong>very</strong> important.</h1>

<h1>This is <em>really <strong>very</strong></em> important.</h1>

+相邻兄弟

如果需要选择紧接在另一个元素后的元素，而且二**者有相同的父元素，**可以使用相邻兄弟选择器（Adjacent sibling selector）。

例如，如果要增加紧接在 h1 元素后出现的段落的上边距，可以这样写：

h1 + p {margin-top:50px;}

这个选择器读作：“选择紧接在 h1 元素后出现的段落，h1 和 p 元素拥有共同的父元素”。

相邻兄弟选择器能作用于后面的标签

Id选择器 #id

**id 选择器即使不被用来创建派生选择器，它也可以独立发挥作用：**

#sidebar {

border: 1px dotted #000;

padding: 10px;

}

**有 id=”sidebar”的标签**

**列表**

#mostImportant {color:red; background:yellow;}

<ul id="mostImportant">

<li>item 1</li>

<li>item 2</li>

<li>item 3</li>

一个选择器，多种用法

**即使被标注为 sidebar 的元素只能在文档中出现一次，这个 id 选择器作为派生选择器也可以被使用很多次：**

#sidebar p {

font-style: italic;

text-align: right;

margin-top: 0.5em;

}

#sidebar h2 {

font-size: 1em;

font-weight: normal;

font-style: italic;

margin: 0;

line-height: 1.5;

text-align: right;

}

Css 类选择器 .class

**注意：**类名的第一个字符不能使用数字！它无法在 Mozilla 或 Firefox 中起作用。

**元素也可以基于它们的类而被选择：**

td.fancy {

color: #f60;

background: #666;

}

在上面的例子中，类名为 fancy 的表格单元将是带有灰色背景的橙色。

<td class="fancy">

你可以将类 fancy 分配给任何一个表格元素任意多的次数。那些以 fancy 标注的单元格都会是带有灰色背景的橙色。那些没有被分配名为 fancy 的类的单元格不会受这条规则的影响。还有一点值得注意，class 为 fancy 的段落也不会是带有灰色背景的橙色，当然，任何其他被标注为 fancy 的元素也不会受这条规则的影响。这都是由于我们书写这条规则的方式，这个效果被限制于被标注为 fancy 的表格单元（即使用 td 元素来选择 fancy 类）。

多类选择器

在 HTML 中，一个 class 值中可能包含一个词列表，**各个词之间用空格分隔**。

<p class="important warning">

This paragraph is a very important warning.

</p>

这**两个词的顺序无关紧要**，写成 warning important 也可以。

我们假设 **class 为 important 的所有元素都是粗体，而 class 为 warning 的所有元素为斜体，class 中同时包含 important 和 warning 的所有元素还有一个银色的背景** 。就可以写作：

.important {font-weight:bold;}

.warning {font-style:italic;}

**.important.warning** {background:silver;}

这里不能有空格。

也就是说：

1） 不要去试图用一个类名，把某个标签的所有样式写完。这个标签要多携带几个类，共同造成这个标签的样式。

2） 每一个类要尽可能小，有“公共”的概念，能够让更多的标签使用。

到底用id还是用class？

答案：尽可能的用class，除非极特殊的情况可以用id。

原因：id是js用的。也就是说，js要通过id属性得到标签，所以我们css层面尽量不用id，要不然js就很别扭。另一层面，我们会认为一个有id的元素，有动态效果。

Css属性选择器 []属性放在中括号中

可以为拥有指定属性的 HTML 元素设置样式，而不仅限于 class 和 id 属性。

注释：只有在规定了 !DOCTYPE 时，IE7 和 IE8 才支持属性选择器。在 IE6 及更低的版本中，不支持属性选择、

[title]

{ //属性选择器

color:red;

}

[title=W3School]

{ //属性和值选择器

border:5px solid blue;

}

[title~=hello] { color:red; }

多个属性的时候i

例如，为了将同时有 href 和 title 属性的 HTML 超链接的文本设置为红色，可以这样写：

a[href][title] {color:red;}

**例子 5：为 XML 文档使用属性选择器**

属性选择器在 XML 文档中相当有用，因为 XML 语言主张要针对元素和属性的用途指定元素名和属性名。

假设我们为描述太阳系行星设计了一个 XML 文档。如果我们想选择有 moons 属性的所有 planet 元素，使之显示为红色，以便能更关注有 moons 的行星，就可以这样写：

planet[moons] {color:red;}

根据具体的属性值

a[href="http://www.w3school.com.cn/about\_us.asp"] {color: red;}

**上面的都是严格的相同才行**

~根据部分属性值选择

如果需要根据属性值中的词列表的某个词进行选择，则需要使用波浪号（~）。

假设您想选择 class 属性中包含 important 的元素，可以用下面这个选择器做到这一点：

p[class~="important"] {color: red;}

子串匹配属性选择器

|  |  |
| --- | --- |
| **类型** | **描述** |
| [abc^="def"] | 选择 abc 属性值以 "def" 开头的所有元素 |
| [abc$="def"] | 选择 abc 属性值以 "def" 结尾的所有元素 |
| [abc\*="def"] | 选择 abc 属性值中包含子串 "def" 的所有元素 |

a[href\*="w3school.com.cn"] {color: red;}

特定属性选择类型

\*[lang|="en"] {color: red;}

**lang**属性值以en开头的

属性选择器参考手册

|  |  |
| --- | --- |
| **选择器** | **描述** |
| [[*attribute*]](http://www.w3school.com.cn/cssref/selector_attribute.asp) | 用于选取带有指定属性的元素。 |
| [[*attribute*=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value.asp) | 用于选取带有指定属性和值的元素。 |
| [[*attribute*~=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp) | 用于选取属性值中包含指定词汇的元素。 |
| [[*attribute*|=*value*]](http://www.w3school.com.cn/cssref/selector_attribute_value_start.asp) | 用于选取带有以指定值开头的属性值的元素，该值必须是整个单词。 |
| [[*attribute*^=*value*]](http://www.w3school.com.cn/cssref/selector_attr_begin.asp) | 匹配属性值以指定值开头的每个元素。 |
| [[*attribute*$=*value*]](http://www.w3school.com.cn/cssref/selector_attr_end.asp) | 匹配属性值以指定值结尾的每个元素。 |
| [[*attribute*\*=*value*]](http://www.w3school.com.cn/cssref/selector_attr_contain.asp) | 匹配属性值中包含指定值的每个元素。 |

交集选择器 ie7开始支持

**同一个标签里标签名和类名都都有，两个必须紧挨着，不能有空格，有空格就变成后代选择器了，且多数情况下是不对的 .既是也是 。一般以标签开头类名结束**

**并集选择器**

选择器之间用逗号隔开

序选择器

**first-child 自己是第一个子标签**

p:first-child {font-weight: bold;}

last-chilid

**lang**

**q元素的lang属性为no的，标签，定义引号 quote(引号)的样式**

q:lang(no)

{

quotes: "~" "~"

}

**p:first-line 找到首行**

{ color:#ff0000;

font-variant:small-caps;

}

**First-letter 首字母的样式**

p:first-letter

{

color:#ff0000;

font-size:xx-large;

}

：before 在该标签前

**在标签前插入图片**

h1:before {

content:url(logo.gif);

}

：after 在该标签后

h1:after

{ content:url(logo.gif); }

选择器总结

说IE6层面兼容的： 标签选择器、id选择器、类选择器、后代、交集选择器、并集选择器、通配符。

|  |
| --- |
| 1. p 2. #box 3. .spec 4. div p 5. div.spec ie7开始兼容 6. div,p 7. \* |

IE7能够兼容的：儿子选择器、下一个兄弟选择器

|  |
| --- |
| 1. div>p 2. h3+p |

IE8能够兼容的：

|  |
| --- |
| 1. ul li:first-child 2. ul li:last-child |

属性选择器

After before

Css样式

继承性

哪些属性能继承？

color、 text-开头的、line-开头的、font-开头的。

这些关于文字样式的，都能够继承； 所有关于盒子的、定位的、布局的属性都不能继承。

层叠性

**层叠性：就是css处理冲突的能力。 所有的权重计算，没有任何兼容问题！**

**CSS像艺术家一样优雅，像工程师一样严谨。**

**总结**

**选中的时候**

**!important; >行内样式>id样式>类样式>标签样式**

不过如果你想无视麻烦的优先级问题，想把一个属性始终应用在元素上。  
可以在样式后面加个 !important;

**都选中的情况下**

**统计权重 id：class：leble 前面的一定大于后面的**

**不进位 一个class>255leble**

**权重相同的谁在后面看谁的**

**没选中的情况下，谁接近听谁的**

**都选中的情况!important 提升该属性的权重**

div{

color:red **!important**;

}

p{

color:blue;

}

由于div是通过继承性来影响文字颜色的，所以!important无法提升它的权重，权重依然是0。

干不过p标签，因为p标签是实实在在选中了，所以字是蓝色的（以p为准）。

背景

背景色

设置为灰色从元素中的文本向外少有延伸，只需增加一些内边距：

p {**background-color**: gray; padding: 20px;}

background-color 不能继承，其默认值是 transparent2。transparent 有“透明”之意，透明的时候会显示父元素的背景

颜色的表示方法

1. 单词
2. Rgb（x,x,x） 红绿蓝发光合成的颜色0-255个颜色

绿：background-color: rgb(0,255,0); 黑：background-color: rgb(0,0,0)

白: background-color: rgb(255,255,255);

1. 16进制 把rgb的十进制转换成了16进制
2. ff

**十六进制可以简化为3位，所有#aabbcc的形式，能够简化为#abc;**

常见的颜色代码：

#000黑 #fff白 #f00 红 #333 灰 #222深灰 #ccc浅灰

背景图像

Background—image

如果需要设置一个背景图像，**必须为这个属性设置一个 URL 值：**

body {background-image: url(/i/eg\_bg\_04.gif);}

**背景重复**

如果需要在页面上对背景图像进行平铺，可以使用 **background-repeat** 属性。

属性值 repeat 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。**repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复**，**no-repeat** **则不允许图像在任何方向上平铺**。默认地，背景图像将从一个元素的左上角开始。请看下面的例子：

body

{

**background-image: url(/i/eg\_bg\_03.gif);**

**background-repeat: repeat-y;**

}



背景定位

**background-position**:center

先，可以使用一些**关键字：top、bottom、left、right 和 center**。通常，这些关键字会**成对出现，另一个默认为ceter**，不过也不总是这样。

**暗黑破坏神的背景图**

**通栏banner**

背景溢出就直接溢出了没什么滚动条

百分数

**换句话说，百分数值同时应用于元素和图像。**也就是说，图像中描述为 50% 50% 的点（中心点）与元素中描述为 50% 50% 的点（中心点）对齐。

如果图像位于 0% 0%，其左上角将放在元素内边距区的左上角。如果图像位置是 100% 100%，会使图像的右下角放在右边距的右下角。

如果你想把一个图像放在水平方向 2/3、垂直方向 1/3 处，可以这样

background-position:66% 33%;

如果只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 50%。这一点与关键字类似。

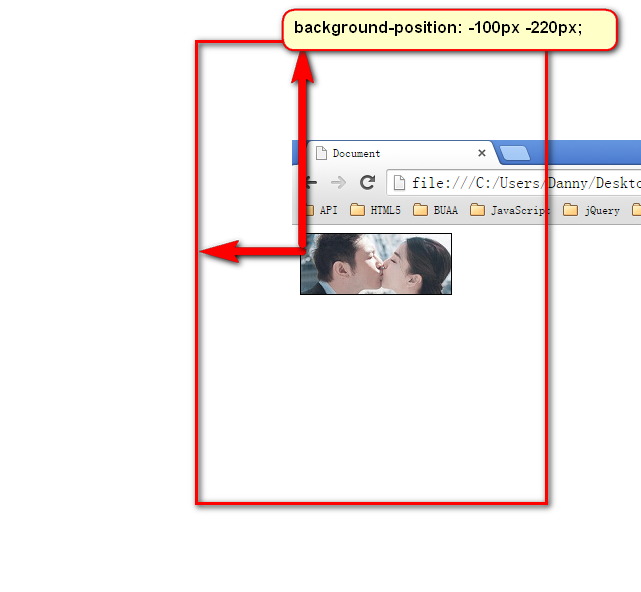
background-position 的默认值是 0% 0%，在功能上相当于 top left。这就解释了背景图像为什么总是从元素内边距区的左上角开始平铺，除非您设置了不同的位置值。

**长度值 精灵用**

长度值解释的是元素内边**距区左上角的偏移。偏移点是图像的左上角。**

比如，如果设置值为 50px 100px，图像的左上角将在元素内边距区左上角向右 50 像素、向下 100 像素的位置上：

原理：

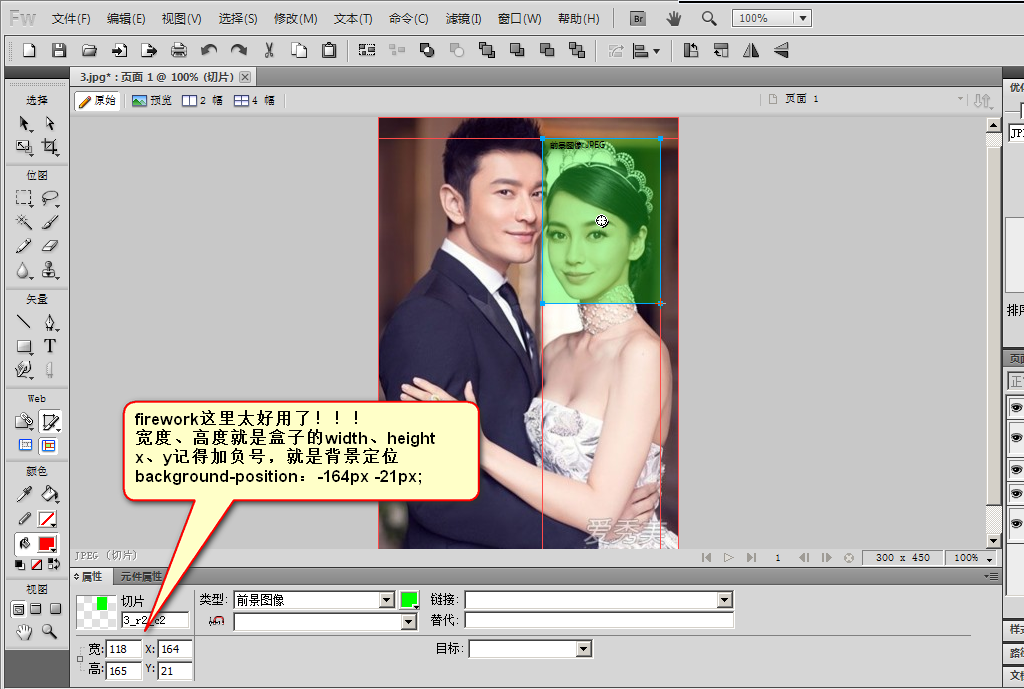


“css精灵”，英语css sprite，所以也叫做“css雪碧”技术。是一种CSS图像合并技术，该方法是将小图标和背景图像合并到一张图片上，然后利用css的背景定位来显示需要显示的图片部分。

css精灵有什么优点，就是减少了http请求。比如4张小图片，原本需要4个http请求。但是用了css精灵，小图片变为了一张图，http请求只有1个了。

淘宝网的精灵图：

 用fireworks精确控制精灵：



**背景关联**

**Background-attachment** 附属物

Fixed 固定在屏幕上，不随滚动滚动 scroll 固定在文档上，随滚动而滚动

**background属性和border一样，是一个综合属性：**

background:red url(1.jpg) no-repeat 100px 100px fixed;

预告一下，CSS3课程中，将学习更多background属性：

background-origin、background-clip、background-size（在CSS2.1背景图片是不能调整尺寸，IE9开始兼容）、

多背景。

**可以任意省略部分**

background: blue url(images/wuyifan.jpg) no-repeat 100px 100px;

Css文本

Text-indent 首行缩进

p {text-indent: 5em;}

**提示：**如果想把一个行内元素的第一行“缩进”，可以用左内边距或外边距创造这种效果。

em 一个汉子的宽度

**使用负值**

如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距

p {text-indent: -5em; padding-left: 5em;}

**使用百分比**

百分数要相对于缩进元素父元素的宽度

**继承**

text-indent 属性可以继承，请考虑如下标记：

水平对齐

**Text-align**

eft 把文本排列到左边。默认值：由浏览器决定。

right 把文本排列到右边。

center 把文本排列到中间。

justify 实现两端对齐文本效果。自动设置字与字间隔

inherit 规定应该从父元素继承 text-align 属性的值。

字间隔 word-spacing

如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把它拉近：

字母间隔letter-spacing

默认关键字是 normal（这与 letter-spacing:0 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量：

字符转换

**Text—transform**

默认值 none 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。最后，capitalize 只对每个单词的首字母大写。

文本装饰decoration

**underline** 会对元素加下划线，就像 HTML 中的 U 元素一样。

overline 的作用恰好相反，会在文本的顶端画一个上划线。

**line-through** 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 元素。

blink 会让文本闪烁，类似于 Netscape 支持的颇招非议的 blink 标记。

**none** 值会关闭原本应用到一个元素上的所有装饰，应用：去除超链接的下划线

不过要注意的是，如果两个不同的装饰都与同一元素匹配，胜出规则的值会完全取代另一个值。请考虑以下的规则：

处理空白符 white-space 属性

p {white-space: normal;}

**normal**它会把所有空白符合并为一个空格

**white-space** 属性的值为 pre，浏览器将会注意额外的空格，甚至回车。在这个方面，而且仅在这个方面，任何元素都可以相当于一个 pre 元素。

**注意：**经测试，IE 7 以及更早版本的浏览器不支持该值，因此请使用非 IE 的浏览器来查看上面的实例。

**nowrap**  它会防止元素中的文本换行，除非使用了一个 br 元素。在 CSS 中使用 nowrap 非常类似于 HTML 4 中用 <td nowrap> 将一个表单元格设置为不能换行，不过 white-space 值可以应用到任何元素

**pre-wrap**： 那么该元素中的文本会保留空白符序列，但是文本行会正常地换行

**pre-line**： 会像正常文本中一样合并空白符序列，但保留换行符。

**Pre-wap** 和 pre-line

下面的表格总结了 white-space 属性的行为：

|  |  |  |  |
| --- | --- | --- | --- |
| **值** | **空白符** | **换行符** | **自动换行** |
| pre-line | 合并 | 保留 | 允许 |
| normal | 合并 | 忽略 | 允许 |
| nowrap | 合并 | 忽略 | 不允许 |
| pre | 保留 | 保留 | 不允许 |
| pre-wrap | 保留 | 保留 | 允许 |

文字方向

Direction ltr rtl

Font字体属性

**● 使用font属性，能够将字号、行高、字体，能够一起设置。**

font: 14px/24px “宋体”;

等价于三行语句：

font-size:14px;

line-height:24px;

font-family:"宋体";

**指定字体系列 font-family**

p {font-family: Times, TimesNR, 'New Century Schoolbook',

Georgia, 'New York', serif;} 逗号隔开

**微软雅黑、宋体、黑体。 如果页面中，需要其他的字体，那么需要切图。**

**英语：Arial 、 Times New Roman**

● **我们要将英语字体，放在最前面，这样所有的中文，就不能匹配英语字体，就自动的变为后面的中文字体：后面逗号隔开无数个备选字体**

1 font-family: "Times New Roman","微软雅黑","宋体";

● 所有的中文字体，都有**英语别名，**我们也要知道：

微软雅黑的英语别名：

1 font-family: "**Microsoft YaHei**";

宋体的英语别名：

1 font-family: "**SimSun**";

font属性能够将font-size、line-height、font-family合三为一：

font:12px/30px "Times New Roman","Microsoft YaHei","SimSun";

● 行高可以用**百分比，表示字号的百分之多少**。一般来说，都是大于100%的，因为行高一定要大于字号。

font:12px/200% “宋体” 等价于 font:12px/24px “宋体”;

**使用引号**

只有当字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号，多为单引号

**字体风格 font-style 斜体的使用**

Normal 正常字体

Italic 斜体显示

Oblique 文本倾斜显示

字体变形 font-variant

p {font-variant:small-caps;}

**字体加粗 font-weight**

100-900 数值或者文字

100最细 数字 400 等价于 normal，而 700 等价于 bold，900最粗

Bolder 比继承的字体粗一些，Lighter 比继承的细一些

**字体大小 font-size**

如果您没有规定字体大小，普通文本（比如段落）的默认大小是 16 像素 (16px=1em)。

**使用像素来设置字体大小**  p {font-size:14px;}

**W3C 推荐使用 em 尺寸单位设置大小**

em 的值会相对于父元素的字体大小改变。

**结合使用百分比和 EM**

在所有浏览器中均有效的方案是为 body 元素（父元素）以百分比设置默认的 font-size 值：

body {font-size:100%;}

h1 {font-size:3.75em;}

h2 {font-size:2.5em;}

p {font-size:0.875em;}

Line-height:设置行间距

**Padding 作用在行上，行高和文字字号不同的行高、字号，一般都是偶数，为了居中**

**百分比设置**

默认为110%-120% 115%

小点的设置为90% 大点的设置为200%

**像素设置**

默认约**20px**

小点的设置为1px。大点的设置为

**数值设置**

默认>1

小于1的就会重合

**综合**

**font:italic bold 12px/30px arial,sans-serif;**

**设置文字在垂直方向上的位置：**

**单行居中：行高设置为height**

**多行居中：padding 设置为1/2 的剩余高度**

css链接 中间用： 应用

**改变链接颜色**

a:link {color:#FF0000;} /\* 未被访问的链接 \*/

a:visited {color:#00FF00;} /\* 已被访问的链接 \*/

a:hover {color:#FF00FF;} /\* 鼠标指针移动到链接上 \*/

a:active {color:#0000FF;} /\* 正在被点击的链接 \*/

hover ai’huai

**文本修饰**

a:link {text-decoration:none;}

a:visited {text-decoration:none;}

a:hover {text-decoration:underline;}

a:active {text-decoration:underline;}

**背景色**

background-color 属性规定链接的背景色：

**方框的链接**

display:block;

font-weight:bold;

font-size:14px;

font-family:Verdana, Arial, Helvetica, sans-serif;

color:#FFFFFF;

background-color:#98bf21;

width:120px;

text-align:center;

padding:4px;

text-decoration:none

css列表

**改变（无序）列表的标志**

ul {**list-style-type** : square}

square 方块

disc 实心点

circle 空心圈

none 没有

**有序列表的标志**

ol.decimal {list-style-type: **decimal**} 十进制的数

ol.lroman {list-style-type: lower-roman} 小号罗马字母

ol.uroman {list-style-type: upper-roman} 大写罗马字母

ol.lalpha {list-style-type: lower-alpha} abcde

ol.ualpha {list-style-type: upper-alpha} ABCDE

cjk-ideographic 一二三四五

**将标志改为图像**

ul li {**list-style-image** : url(xxx.gif)}

**列表的放置位置 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 |

**可以直接写list-style 然后写出各个默认值**

list-style: square inside url('/i/eg\_arrow.gif')

css表格

**表格边框**

table

{ border-collapse:collapse; } //记得加上html5的开头

table,th, td

{ border: 1px solid black; }、

**table、th 以及 td 元素都有独立的边框。需要单独定义**

**宽度和高度**

**Width:100%**

**Height：50px；**

**表格文本对齐**

**Text-align 水平的对齐方式 left right center**

**Vertical-align top bottom center**

**表格内边距**

**Td 和 th 设置padding**

**padding:15px;**

**背景和字体**

**th{**

**background-color:green;**

**color:white;}**

**border-spacing**

规定相邻单元的边框之间的距离。使用 px、cm 等单位。不允许使用负值。

如果定义一个 length 参数，那么定义的是水平和垂直间距。

如果定义两个 length 参数，那么第一个设置水平间距，而第二个设置垂直间距。

border-collapse:separate;

border-spacing:10px 50px;

**empty-cells （仅分离的可用）**

border-collapse:separate;

empty-cells:hide;

hide 隐藏空的td

**caption-side**

caption

{ caption-side:bottom; }

Bottom top inherit（继承父类） 位置

**Table-layout**

Fixed 宽高是固定的

Automatic 宽度随着内容变化

外部轮廓outline

Outline：color style width

**Outline-color**

**Outline-style**

|  |  |
| --- | --- |
| none | 默认。定义无轮廓。 |
| dotted | 定义点状的轮廓。 |
| dashed | 定义虚线轮廓。 |
| solid | 定义实线轮廓。 |
| double | 定义双线轮廓。双线的宽度等同于 outline-width 的值。 |
| groove | 定义 3D 凹槽轮廓。此效果取决于 outline-color 值。 |
| ridge | 定义 3D 凸槽轮廓。此效果取决于 outline-color 值。 |
| inset | 定义 3D 凹边轮廓。此效果取决于 outline-color 值。 |
| outset | 定义 3D 凸边轮廓。此效果取决于 outline-color 值。 |
| inherit | 规定应该从父元素继承轮廓样式的设置。 |

**Outline-width**

thin规定细轮廓。

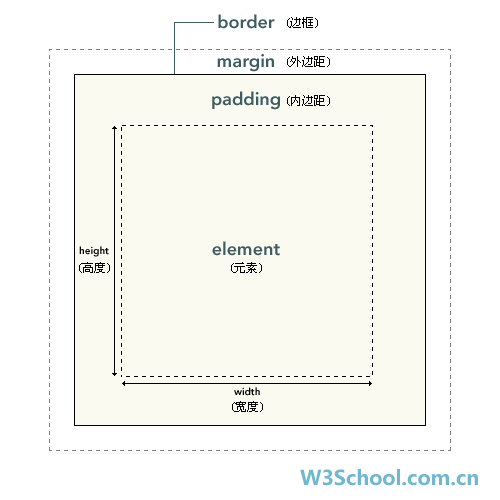
medium默认。规定中等的轮廓。

thick规定粗的轮廓。

*length*允许您规定轮廓粗细的值。

inherit规定应该从父元素继承轮廓宽度的设置。

Css盒子模型



width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。 开发的时候要用**专业的清除默认样式的样式报表，去除影响**

**要保持盒子占有空间的大小，width和padding的增减是是相反的**

Padding 属性

Padding区域有background的

**可以直接padding：** 上右下左的顺序来

1个对应4个 2个对应上下、左右 3个 对应上右下左=右

也可以padding-top： Padding-right: Padding-bottom Padding-left:

如果只用小属性写的时候，没有写哪个，哪个方向上padding就是0

**要懂得，用小属性层叠大属性：**

|  |
| --- |
| padding: 20px;  padding-left: 30px; |

这个顺序不能改变,否则前面的小属性会被后面的大属性层叠，覆盖

最终结果 left:30px top right buttom 20px

一些属性默认带有padding;

Padding对盒子大小的硬性

内边距 会影响盒子大小

行内元素 尽量不用 上下的padding和margin

**继承的宽度 padding不会挤开** 。子盒子内不给宽度的时候默认父亲的，但是，加了左右padding不会撑开父亲，总宽度仍然是这些

同时高度必须自己定义不然无法显示宽度

**Border 属性**

width style color

1px solid black

色如果不写，默认是黑色。另外两个属性不写，要命了，显示不出来边框。

如果公司里面的设计师，处女座的，追**求极高的页面还原度，那么不能使用css来制作边框。**

就要用到图片，**就要切图了。所以，比较稳定的就几个：solid、dashed 虚线、dotted 点**

**border-style 也可以分开定义 border-top-style border-right-style border-bottom-s30ty**

|  |  |
| --- | --- |
| n1one | 定义无边框。 |
| hidden | 与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。 |
| dotted | 定义点状边框。在大多数浏览器中呈现为实线。 |
| dashed | 定义虚线。在大多数浏览器中呈现为实线。 |
| solid | 定义实线。 |
| double | 定义双线。双线的宽度等于 border-width 的值。 |
| groove | 定义 3D 凹槽边框。其效果取决于 border-color 的值。 |
| ridge | 定义 3D 垄状边框。其效果取决于 border-color 的值。有阴影的样子的东西 |
| inset | 定义 3D inset 边框。其效果取决于 border-color 的值。 |
| outset | 定义 3D outset 边框。其效果取决于 border-color 的值。 |
| inherit | 规定应该从父元素继承边框样式。 |

**Border-width: border-top-width**

可以指定长度值，比如 2px 或 0.1em；

或者使用 3 个关键字 thin 、medium（默认值） 和 thick。

**因此，如果您希望边框出现，就必须声明一个边框样式。**

**Border-color: border-top-color**

border是一个大综合属性，

|  |
| --- |
| 1. border:1px solid red; |

就是把4个边框，都设置为1px宽度、线型实线、red颜色。

border属性能够被拆开，有两大种拆开的方式：

1） 按3要素:border-width、border-style、border-color

2） 按方向：border-top、border-right、border-bottom、border-left

**按3要素拆开：**

|  |
| --- |
| 1. **border-width**:10px; → 边框宽度 2. **border-style**:solid; → 线型 3. **border-color**:red; → 颜色。 |

等价于：

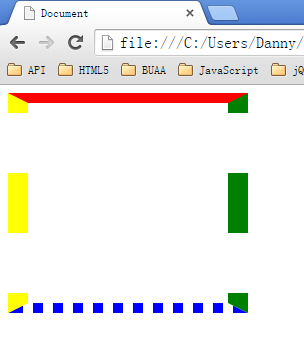
|  |
| --- |
| 1. border:10px solid red; |

现在心里要明白，**原来一个border是由三个小属性综合而成：**

**border-width border-style border-color。**

如果某一个小要素后面是空格隔开的多个值，那么就是**上右下左**的顺序：

|  |
| --- |
| 1. border-width:10px 20px; 2. border-style:**solid dashed dotted**; 3. border-color:**red green blue yellow**; |



**按方向来拆**

|  |
| --- |
| 1. border-top:10px solid red; 2. border-right:10px solid red; 3. border-bottom:10px solid red; 4. border-left:10px solid red; |

等价于

|  |
| --- |
| 1. border:10px solid red; |

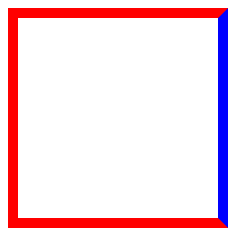
**按方向还能再拆一层，就是把每个方向的，每个要素拆开，一共12条语句：**

|  |
| --- |
| 1. border-top-width:10px; 2. border-top-style:solid; 3. border-top-color:red; 4. border-right-width:10px; 5. border-right-style:solid; 6. border-right-color:red; 7. border-bottom-width:10px; 8. border-bottom-style:solid; 9. border-bottom-color:red; 10. border-left-width:10px; 11. border-left-style:solid; 12. border-left-color:red; |

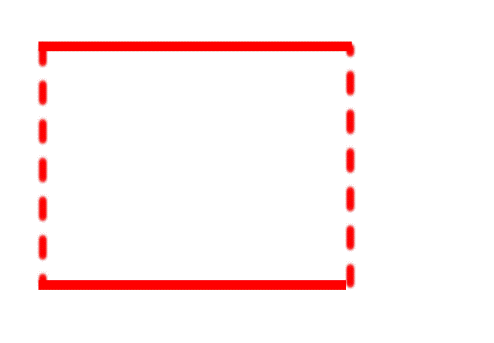
等价于

|  |
| --- |
| 1. border:10px solid red; |

工作中到底用什么？很简答：什么简单用什么？

 写法：

|  |
| --- |
| 1. border:10px solid red; 2. border-right-color:blue; |

写法：

|  |
| --- |
| 1. border:10px solid red; 2. border-style:solid dashed; |

border可以没有，

|  |
| --- |
| 1. border:none; |

某一条边没有：

|  |
| --- |
| 1. border-left: none; |

也可以调整左边边框的宽度为0：

|  |
| --- |
| 1. border-left-width: 0; |

**黑科技：可控三角形**

div{

width: 0px;

height: 0px;

border: 20px solid white;

border-top-color: green;

border-bottom: none;

}

四、标准文档流

宏观的讲，我们的web页面和photoshop等设计软件有本质的区别：web页面的制作，是个“流”，必须从上而下，像“织毛衣”。而设计软件，想往哪里画个东西，都能画。

我们要看看标准流有哪些微观现象：

1. **空白折叠现象：** 两个**img之间的空隙**是因为换行被转换为空格造成的

图片和表单都是纯文本，遵循**多个空格合成一个，并且换行也是空格的设计**

比如，如果我们想让img标签之间没有空隙，必须紧密连接：

|  |
| --- |
| 1. <img src="images/0.jpg" /><img src="images/1.jpg" /><img src="images/2.jpg" /> |

2） **高矮不齐，底边对齐**：



3） **自动换行，一行写不满，换行写。**

4.1 块级元素和行内元素

学习的初期，你就要知道，标准文档流等级森严。标签分为两种等级：

1） 块级元素

● 霸占一行，不能与其他任何元素并列

● 能接受宽、高

● 如果不设置宽度，那么宽度将默认变为父亲的100%。

2） 行内元素

● 与其他行内元素并排

● 不能设置宽、高。默认的宽度，就是文字的宽度。可以设置行高

在HTML中，我们已经将标签分过类，当时分为了：文本级、容器级。

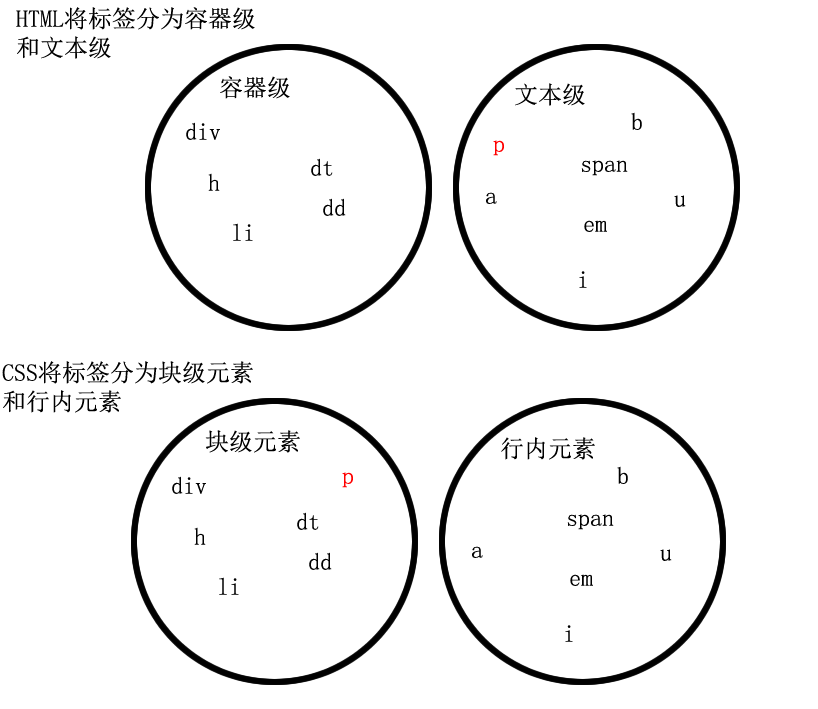
文本级：p、span、a、b、i、u、em、img

容器级：div、h系列、li、dt、dd

CSS的分类和上面的很像，就p不一样：

所有的文本级标签，都是行内元素，除了p，p是个文本级，但是是个块级元素。

所有的容器级标签都是块级元素。



4.2 块级元素和行内元素的相互转换

块级元素可以设置为行内元素

行内元素可以设置为块级元素

|  |
| --- |
| 1. div{ 2. **display: inline;** 3. background-color: pink; 4. width: 500px; 5. height: 500px; 6. } |

display是“显示模式”的意思，用来改变元素的行内、块级性质

inline就是“行内”。

一旦，给一个标签设置

|  |
| --- |
| 1. **display: inline;** |

那么，这个标签将立即变为行内元素。此时它和一个span无异：

● 此时这个div不能设置宽度、高度；

● 此时这个div可以和别人并排了

同样的道理，

|  |
| --- |
| 1. span{ 2. **display: block;** 3. width: 200px; 4. height: 200px; 5. background-color: pink; 6. } |

“block”是“块”的意思

让标签变为块级元素。此时这个标签，和一个div无异：

● 此时这个span能够设置宽度、高度

● 此时这个span必须霸占一行了，别人无法和他并排

● 如果不设置宽度，将撑满父亲

标准流里面限制非常多，标签的性质恶心。比如，我们现在就要并排、并且就要设置宽高。

所以，移民！脱离标准流！

css中一共有三种手段，使一个元素脱离标准文档流：

1） 浮动

2） 绝对定位

3） 固定定位

浮动

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

Float属性

Float：浮动的时候要一起浮动

**脱标、贴边、字围、收缩。**

脱标：

可以设置宽高，脱离了标准文档流，

下面的标准流会成为第一个元素，渲染到了第一个元素位置，部分被float的盖住

不需要displlay ，可以并排并且可以设宽高

贴边

浮动的元素互相贴靠，不会忘往里钻，盒子内部的元素也遵循这个原则，

|  |  |  |
| --- | --- | --- |
|  |  |  |

字围：

可以挡住盒子挡不住盒子里面的字

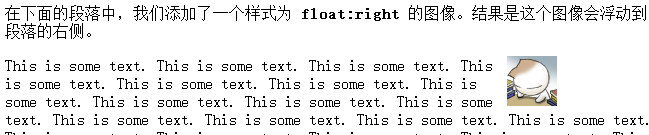
**第一个例子**

img

{ float:right;

margin-right:100px;

}



**第二个例子**

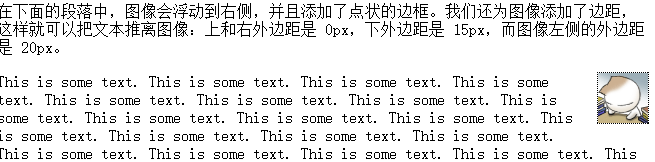
img {

**float:right;**

**border:1px dotted black;**

**margin:0px 0px 15px 20px;**

**}**



**带标题的图像浮于右侧**

div{

float:right;

width:120px;

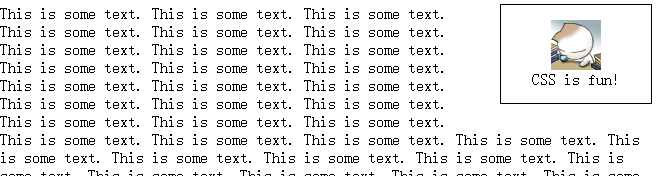
margin:0 0 15px 20px;

padding:15px;

border:1px solid black;

text-align:center;

}



**首字母浮于左侧**

span{

float:left;

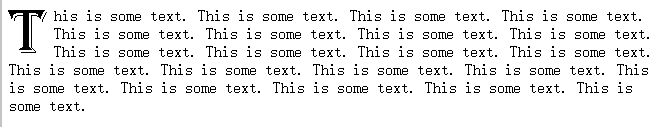
width:0.7em;

font-size:400%;

font-family:algerian,courier;

line-height:80%;

}



**创建水平菜单（导航列表）**

<style type="text/css">

ui{

**float:left;**

width:100%;

padding:0;

margin:0;

**list-style-type:none;**

}

a{

float:left;

width:7em;

text-decoration:none;

color:white;

background-color:purple;

padding:0.2em 0.6em;

border-right:1px solid white;

}

a:hover {background-color:#ff3300}

**li {display:inline} //uxia 段内元素**

</style>

控制浮动的办法让浮动可以分开(给父标签高)

**1.给浮动的元素的祖先元素加高度。**

如果一个元素要浮动，那么它的祖先元素一定要有高度。**高度的盒子，才能关住浮动。**

高度height很少出现。因为能被内容撑高！

**2.Clear**

浮动元素哪侧没有其他的元素

|  |  |  |
| --- | --- | --- |
| **值** | **描述** | |
| left | 在左侧不允许浮动元素。 | |
| right | 在右侧不允许浮动元素。 | |
| both | 在左右两侧均不允许浮动元素。 | |
| none | 默认值。允许浮动元素出现在两侧。 | |
| inherit | 规定应该从父元素继承 clear 属性的值。 | |
| **clear:both;** | | |

**clear就是清除，both指的是左浮动、右浮动都要清除。意思就是：清除别人对我的影响。写在后面的上**

**这种方法有一个非常大的、致命的问题，margin失效了。**

**3.隔墙法 clear:both;**

**外墙法：在两个div间新建一个新空的div，携带两个属性，但是没有高度**

**clear:both;**

**height：10px； margin依然不可用，但是自己高度当组margin**

如果我们现在想让第一个div，自动的根据自己的儿子，撑出高度，我们就要想一些“小伎俩”，“奇淫技

.

**4.内枪法： 在div内部新建一个空div 外部的div被内部的儿子撑出高了**

**5.overflow:hidden; 在父标签加上属性**

一个父亲不能被自己浮动的儿子，撑出高度。但是，只要给父亲加上overflow:hidden; 那么，父亲就能被儿子撑出高了。这是一个偏方。

6：伪元素 盒子的开头和结尾加上：：before ：：after

.clearfix:after {

content:””;

Visibility:hidden;

Display:block;

Height:0;

Clear:both;

}

.clearfix{

Zoom:1;

}

清除浮动： 真正的叫法 闭合浮动

4. 双伪元素

.clearfix:before,.clearfix:after{  
 display: table;  
 content: "";  
}  
.clearfix:after {  
 clear: both;  
}  
.clearfix {  
 zoom: 1;  
}

3.6 浏览器兼容问题

**上述知识点遇见的浏览器兼容问题**

第一，IE6，不支持小于12px的盒子，任何小于12px的盒子，在IE6中看都大

解决办法很简单，就是将盒子的字号，设置小（小于盒子的高），比如0px。 装饰用

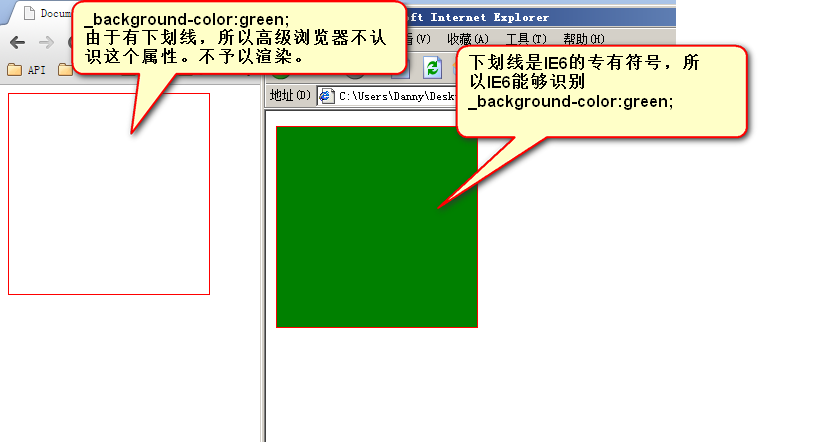
|  |
| --- |
| 1. height: 4px; 2. \_font-size: 0px; |

我们现在介绍一下浏览器hack。hack就是“黑客”，就是使用浏览器提供的后门，针对某一种浏览器做兼容。

**IE6留了一个后门，就是只要给css属性之前，加上下划线，这个属性就是IE6认识的专有属性。**

比如：

|  |
| --- |
| 1. \_background-color: green; |



解决微型盒子，正确写法：

|  |
| --- |
| 1. height: 10px; 2. **\_font-size:0;** |

第二，IE6不支持用overflow:hidden;来清除浮动的

解决办法，以毒攻毒。追加一条

|  |
| --- |
| 1. \_zoom:1; |

完整写法：

|  |
| --- |
| 1. overflow: hidden; 2. **\_zoom:1;** |

实际上，\_zoom:1;能够触发浏览器hasLayout机制。这个机制，不要深究了，因为就IE6有。我们只需要让IE6好用，具体的实现机制，有兴趣的同学，自行百度。

强调一点， overflow:hidden;的本意，就是溢出盒子的border的东西隐藏，这个功能是IE6兼容的。不兼容的是overflow:hidden;清除浮动的时候。

我们刚才学习了两个IE6的兼容问题，这两个IE6的兼容问题，都是通过多写一条hack来解决的。

这个我们称为伴生属性。

|  |
| --- |
| 1. height:6px; 2. **\_font-size:0;** |

|  |
| --- |
| 1. overflow:hidden; 2. **\_zoom:1;** |

3. 伪元素

.clearfix:after {

content:””;

Visibility:hidden;

Display:block;

Height:0;

Clear:both;

}

.clearfix{

Zoom:1;

}

清除浮动： 真正的叫法 闭合浮动

4. 双伪元素

.clearfix:before,.clearfix:after{  
 display: table;  
 content: "";  
}  
.clearfix:after {  
 clear: both;  
}  
.clearfix {  
 zoom: 1;  
}

Css定位 position

**Position**

**Relative 相对定位**

h2{

position:relative;

**left:-20px**  //相对哪个边移动多少 }

**不脱标，形影分离，所有的功能都在自己原来的位置**

↘：

|  |
| --- |
| 1. position: relative; 2. top: 10px; 3. left: 40px; |

↙：

|  |
| --- |
| 1. position: relative; 2. **right: 100px; → 往左边移动** 3. top: 100px; |

**作用**

相对定位有坑，所以一般不用于做“压盖”效果。页面中，效果极小。就两个作用：

1） **微调**元素

2） 做绝对定位的参考，子绝父相（讲绝对定位的时候说）

Absolute 绝对定位

绝对定位之后，标签就不区分所谓的行内元素、块级元素了，不需要display:block;就可以设置宽、高了：

{ //绝对坐标，相对于页面的左方和上方

position:absolute;

left:100px;

top:150px

}

**参考点**

绝对定位的参考点，如果用**top**描述，那么定位参考点就是**页面的左上角**，而不是浏览器的左上角：

如果用**bottom描**述，那么就是**浏览器首屏窗口尺寸**，对应的页面的左下角：

一个绝对定位的元素，如果**父辈元素中出现了也定位了的元素**，那么将**以父辈这个元素**，为参考点。

● 要听最近的已经定位的祖先元素的，不一定是父亲，可能是爷爷：

|  |
| --- |
| 1. <div class="box1"> → 相对定位 2. <div class="box2"> → 没有定位 3. <p></p> → 绝对定位，将以box1为参考，因为box2没有定位，box1就是最近的父辈元素 4. </div> 5. </div> |

|  |
| --- |
| 1. <div class="box1"> → 相对定位 2. <div class="box2"> → 相对定位 3. <p></p> → 绝对定位，将以box2为参考，因为box2是自己最近的父辈元素 4. </div> 5. </div> |

● 不一定是相对定位，任何定位，都可以作为参考点

|  |
| --- |
| 1. <div> → 绝对定位 2. <p></p> → 绝对定位，将以div作为参考点。因为父亲定位了。 3. </div> |

子绝父绝、子绝父相、子绝父固，都是可以给儿子定位的。但是，工程上子绝、父绝，没有一个盒子在标准流里面了，所以页面就不稳固，没有任何实战用途。工程上，“子绝父相”有意义，父亲没有脱标，儿子脱标在父亲的范围里面移动。

|  |
| --- |
| 1. <div class=”box1”> → 绝对定位 2. <div class=”box2”> → 相对定位 3. <div class=”box3”> → 没有定位 4. <p></p> → 绝对定位，以box2为参考定位。 5. </div> 6. </div> 7. </div> |

● 绝对定位的儿子，无视参考的那个盒子的padding。

下图中，绿色部分是div的padding，蓝色部分是div的内容区域。那么此时，div相对定位，p绝对定位。

p将无视父亲的padding，在border内侧为参考点，进行定位：

**绝对定位的盒子居中**

非常简单，当做公式记忆下来。就是left:50%; margin-left:负的宽度的一半。

Fixed 固定定位

**脱标的，**就是相对浏览器窗口定位。页面如何滚动，这个盒子显示的位置不变。

**fixed**{ //相对于窗口位置固定

position:fixed;

left:5px;

top:5px;

}

z-index

**数值大的压着数值小的**

**只有定位了的元素才能有z-index ，绝对/相对/固定定位，而浮动的东西不能用**

是否在同一个垂直维度上，决定是否遮挡，如果不在一个维度上就会有前后显示，决定那个在最前面，有透明属性的时候可以当背景用，可以为正负int值

Z-index 仅能在定位元素上奏效（例如 position:absolute;）！

**● 从父现象：父亲怂了，儿子再牛逼也没用。**

**滚动显示**

Background-color:#00FFFF;

width:150px;

height:150px;

**overflow 溢出**

元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚动条等行scroll 滚动条显示内容

visible 可见（默认的，会超出和尺寸显示）

hidden（隐藏多余的部分）

auto 系统自己判定，没有超出的时候就是visible

**clip 剪裁**

属性剪裁**绝对定位元素。**唯一合法的形状值是：rect (*top*, *right*, *bottom*, *left*)，相对于左和上的方向画出显示窗口，**rect 矫正**

img

{ position:absolute;

clip:rect(10px 500px 200px 10px) }

**vertical-align**

图像和文字的上部还是下部对齐

|  |  |
| --- | --- |
| baseline | 默认。元素放置在父元素的基线上。 |
| sub | 垂直对齐文本的下标。 下边界和文字的下标对齐 |
| super | 垂直对齐文本的上标 下边界和文字的上标对齐 |
| top | 把元素的顶端与行中最高元素的顶端对齐 上边界对齐 |
| text-top | 把元素的顶端与父元素字体的顶端对齐 上边界对齐 |
| middle | 把此元素放置在父元素的中部。 中间对齐 |
| bottom | 把元素的顶端与行中最低的元素的顶端对齐。 |
| text-bottom | 把元素的底端与父元素字体的底端对齐。 |
| length | 将下表对齐设置为0；向上移动为正，向下移动为负 |
| % | 使用 "line-height" 属性的百分比值来排列此元素。 和上面的相似相对于文本的宽度上下调 |
| inherit | 规定应该从父元素继承 vertical-align 属性的值。 |

相对定位

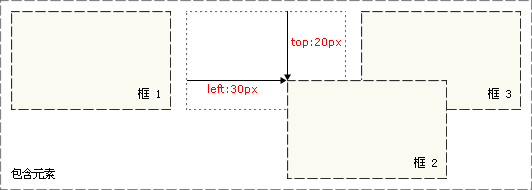
#box\_relative {

position: relative;

left: 30px;

top: 20px;

}



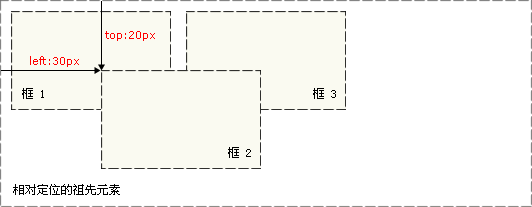
注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

绝对定位

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其它元素的布局就像绝对定位的元素不存在一样：

如下图所示：



绝对定位的元素的位置相对于**最近的已定位祖先元素**，如果元素没有已定位的祖先元素，那么它的位置相对于**最初的包含块**。

**提示：**因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 [z-index 属性](http://www.w3school.com.cn/cssref/pr_pos_z-index.asp)来控制这些框的堆放次序。

浮动定位

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

Float属性

Float：rigth 浮动在文本外，文本遇到了会自动换行

脱标、贴边、字围、收缩。

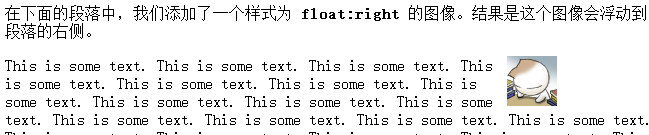
**第一个例子**

img

{ float:right;

margin-right:100px;

}



**第二个例子**

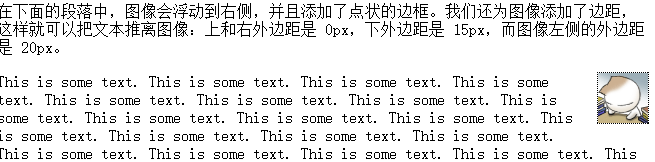
img {

**float:right;**

**border:1px dotted black;**

**margin:0px 0px 15px 20px;**

**}**



**带标题的图像浮于右侧**

div{

float:right;

width:120px;

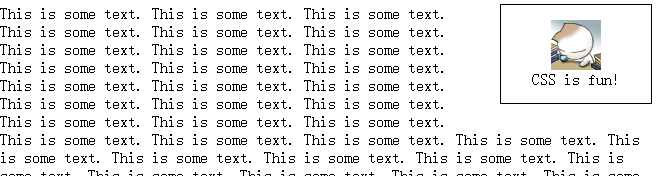
margin:0 0 15px 20px;

padding:15px;

border:1px solid black;

text-align:center;

}



**首字母浮于左侧**

span{

float:left;

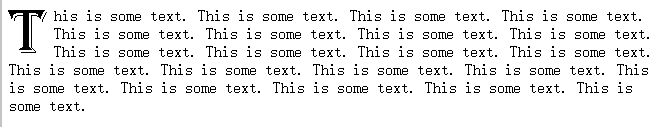
width:0.7em;

font-size:400%;

font-family:algerian,courier;

line-height:80%;

}



**创建水平菜单（导航列表）**

<style type="text/css">

ui{

**float:left;**

width:100%;

padding:0;

margin:0;

**list-style-type:none;**

}

a{

float:left;

width:7em;

text-decoration:none;

color:white;

background-color:purple;

padding:0.2em 0.6em;

border-right:1px solid white;

}

a:hover {background-color:#ff3300}

**li {display:inline} //uxia 段内元素**

</style>

控制浮动的办法让浮动可以分开(给父标签高)

**1.给浮动的元素的祖先元素加高度。**

如果一个元素要浮动，那么它的祖先元素一定要有高度。**高度的盒子，才能关住浮动。**

高度height很少出现。因为能被内容撑高！

**2.Clear**

浮动元素哪侧没有其他的元素

|  |  |  |
| --- | --- | --- |
| **值** | **描述** | |
| left | 在左侧不允许浮动元素。 | |
| right | 在右侧不允许浮动元素。 | |
| both | 在左右两侧均不允许浮动元素。 | |
| none | 默认值。允许浮动元素出现在两侧。 | |
| inherit | 规定应该从父元素继承 clear 属性的值。 | |
| **clear:both;** | | |

**clear就是清除，both指的是左浮动、右浮动都要清除。意思就是：清除别人对我的影响。**

**这种方法有一个非常大的、致命的问题，margin失效了。**

**3.隔墙法**

**外墙法：在两个div间新建一个新空的div，隔开两个**

如果我们现在想让第一个div，自动的根据自己的儿子，撑出高度，我们就要想一些“小伎俩”，“奇淫技巧

**4.内枪法： 在div内部新建一个空div 外部的div被内部的儿子撑出高了**

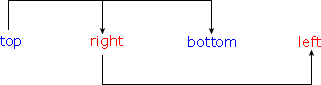
**5.overflow:hidden; 在父标签加上属性**

一个父亲不能被自己浮动的儿子，撑出高度。但是，只要给父亲加上overflow:hidden; 那么，父亲就能被儿子撑出高了。这是一个偏方。

Margin

接受任何长度单位，像素px、 英寸 in、毫米mm、em、百分数（相对于父元素）

**值复制**



换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到

**提示：**Netscape 和 IE 对 body 标签定义的默认边距（margin）值是 8px。而 Opera 不是这样。相反地，Opera 将内部填充（padding）的默认值定义为 8px，因此如果希望对整个网站的边缘部分进行调整，并将之正确显示于 Opera 中，那么必须对 body 的 padding 进行自定义。

外边距合并（塌陷现象）

**标准文档流中**，**竖直方向的margin不叠加，以较大的为**准，两个相邻的边距会合并留下较大的边距

**浮动之后两个盒子之间是margin叠加，**不塌陷

水平方向没有叠加

**盒子居中**

Margin-left：auto

Margin-right:auto

简写

Margin:0 auto;

1)使用margin:0 auto; 的盒子，**必须有width**，有明确的width

2） 只有**标准流的盒子**，才能使用**margin:0 auto**; 居中。

也就是说，当一个盒子浮动了、绝对定位了、固定定位了，都不能使用margin:0 auto;

3） **margin:0 auto;是在居中盒子，不是居中文本**。 P是块级元素

**文本的居中**

text-align:center;

text-align:right; 文本居右

默认居中的

**要善于使用父亲的padding 而不是儿子的margin**

如果**父亲没有border，那么儿子的margin实际上踹的是“流”**，踹的是这“行”。所以，父亲整体也掉下来了

这个p有一个margin-top踹父亲，试图将自己下移

|  |
| --- |
| 1. <div> 2. <p></p> 3. </div> |

结果：

|  |  |
| --- | --- |
|  |  |

**margin这个属性，本质上描述的是兄弟和兄弟之间的距离； 最好不要用这个marign表达父子之间的距离。**

所以，我们一定要善于使用父亲的padding，而不是儿子的margin。

**IE BUG**

**连续浮动会出现双倍浮动**

解决方案：

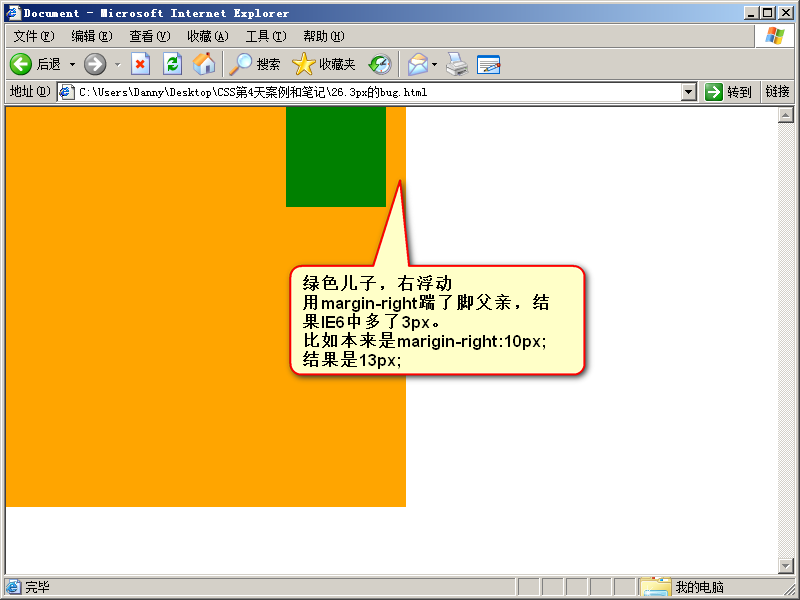
1）使**浮动的方向和margin的方向，相反。**

所以，你就会发现，我们特别喜欢，浮动的方向和margin的方向相反。并且，前端开发工程师，把这个当做习惯了。

|  |
| --- |
| 1. float: left; 2. margin-right: 40px; |

2) 也可以单独手动第一个元素一般的边界

IE6的3px bug



解决办法：

不用管，因为根本就不允许用儿子踹父亲。所以，如果你出现了3px bug，说明你的代码不标准。

**IE6，千万不要跟他死坑、较劲，它不配。 格调要高，老师给你讲，就是为了增加面试的成功率。不是为了让你成为IE6的专家。**

关于margin的IE6兼容问题

IE6双倍margin bug

**当出现连续浮动的元素，携带和浮动方向相同的margin时，队首的元素，会双倍marign。**

|  |
| --- |
| 1. <ul> 2. <li></li> 3. <li></li> 4. <li></li> 5. </ul> |



解决方案：

1）使浮动的方向和margin的方向，相反。

所以，你就会发现，我们特别喜欢，浮动的方向和margin的方向相反。并且，前端开发工程师，把这个当做习惯了。

|  |
| --- |
| 1. float: left; 2. margin-right: 40px; |

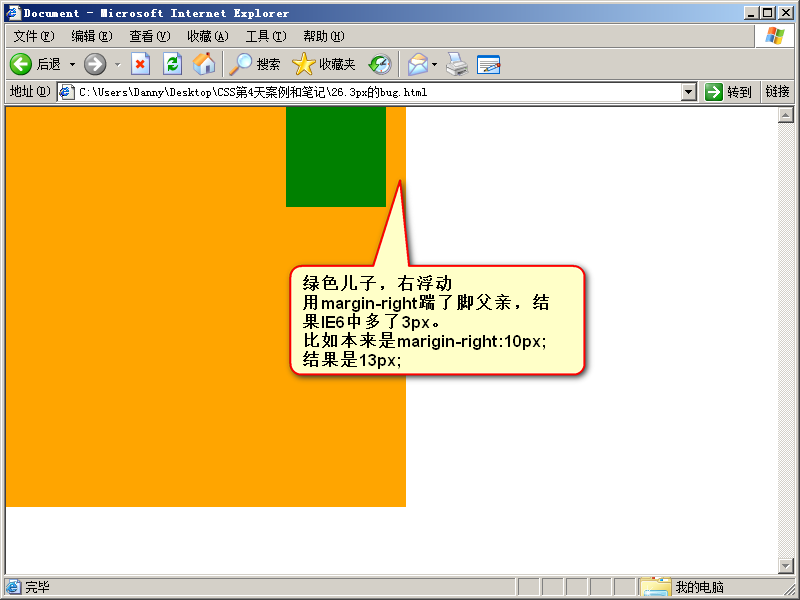
2）使用hack（没必要，别惯着这个IE6）

单独给队首的元素，写一个一半的margin

|  |
| --- |
| 1. <li class="no1"></li> |

|  |
| --- |
| 1. ul li.no1{ 2. \_margin-left:20px; 3. } |

IE6的3px bug



解决办法：

不用管，因为根本就不允许用儿子踹父亲。所以，如果你出现了3px bug，说明你的代码不标准。

**IE6，千万不要跟他死坑、较劲，它不配。 格调要高，老师给你讲，就是为了增加面试的成功率。不是为了让你成为IE6的专家。**

**创建无表格的首页 float创建**

**浮动不会**

css伪类

**同一个标签，根据用户的某种状态不同，有不同的样式。这就叫做“伪类”。**

**用法**

selector : pseudo-class {property: value}

标签名 : 伪类 {name:value}

**超链接**

a:link {color: #FF0000} 链接

a:visited {color: #00FF00} 访问过

a:hover {color: #FF00FF} 悬停，在上上面

a:active {color: #0000FF} 激活

最标准的，就是把link、visited、hover都要写。但是前端开发工程师在大量的实践中，发现不写link、visited浏览器也挺兼容。所以这些“老油条”们，就把a标签简化了：

a:link、a:visited都是可以省略的，简写在a标签里面。也就是说，a标签涵盖了link、visited的状态。

|  |
| --- |
| .nav ul li a{  **display: block;**  width: 120px;  height: 50px;  **text-decoration: none;**  background-color: purple;  color:white;  }  .nav ul li a:hover{  background-color: orange;  } |

**focus使用鼠标焦点选中了**

input:focus {background-color:yellow;}

**first-child 自己是第一个子标签**

p:first-child {font-weight: bold;}

**lang**

**q元素的lang属性为no的，标签，定义引号 quote(引号)的样式**

q:lang(no)

{

quotes: "~" "~"

}

伪元素

**p:first-line 找到首行**

{ color:#ff0000;

font-variant:small-caps;

}

**First-letter 首字母的样式**

p:first-letter

{

color:#ff0000;

font-size:xx-large;

}

**：before 在该标签前**

**在标签前插入图片**

h1:before {

content:url(logo.gif);

}

**：after 在该标签后**

h1:after

{ content:url(logo.gif); }

对齐

**块级元素的对齐 !DOCTYPE 水平对齐**

**.center{**

Margin-left:auto;

margin-right:auto;

width:70%;

background-color:#b0e0e6;

**}**

**绝对定位**

Display属性

**块级元素和行内元素**

1） 块级元素

● 霸占一行，不能与其他任何元素并列

● 能接受宽、高

● 如果不设置宽度，那么宽度将默认变为父亲的100%。

2） 行内元素

● 与其他行内元素并排

● 不能设置宽、高。默认的宽度，就是文字的宽度。

|  |  |
| --- | --- |
| none | 此元素不会被显示。 |
| block | 此元素将显示为块级元素，此元素前后会带有换行符。链接显示为块元素可使整个链接区域可点击（不仅仅是文本），同时也允许我们规定宽度。 |
| inline | 默认。此元素会被显示为内联元素，元素前后没有换行符。 |
| inline-block | 行内块元素。（CSS2.1 新增的值） |
| list-item | 此元素会作为列表显示。 |
| run-in | 此元素会根据上下文作为块级元素或内联元素显示。 |
| table | 此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。 |
| inline-table | 此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。 |
| table-row-group | 此元素会作为一个或多个行的分组来显示（类似 <tbody>）。 |
| table-header-group | 此元素会作为一个或多个行的分组来显示（类似 <thead>）。 |
| table-footer-group | 此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。 |
| table-row | 此元素会作为一个表格行显示（类似 <tr>）。 |
| table-column-group | 此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。 |
| table-column | 此元素会作为一个单元格列显示（类似 <col>） |
| table-cell | 此元素会作为一个表格单元格显示（类似 <td> 和 <th>） |
| table-caption | 此元素会作为一个表格标题显示（类似 <caption>） |
| inherit | 规定应该从父元素继承 display 属性的值 |

标准流里面限制非常多，标签的性质恶心。比如，我们现在就要并排、并且就要设置宽高移民！脱离标准流！

css中一共有三种手段，使一个元素脱离标准文档流：

1） 浮动

2） 绝对定位

3） 固定定位

Css定位

**Position**

**Relative**

h2{

position:relative;

**left:-20px**  //元素的原始左侧位置减去 20 像素

}

**Absolute** { //绝对坐标，相对于页面的左方和上方

position:absolute;

left:100px;

top:150px

}

**fixed**{ //相对于窗口位置固定

position:fixed;

left:5px;

top:5px;

}

**滚动显示**

background-color:#00FFFF;

width:150px;

height:150px;

**overflow 溢出**

元素中的内容超出了给定的宽度和高度属性，overflow 属性可以确定是否显示滚动条等行scroll 滚动条显示内容

visible 可见（默认的，会超出和尺寸显示）

hidden（隐藏多余的部分）

auto 系统自己判定，没有超出的时候就是visible

**clip 剪裁**

属性剪裁**绝对定位元素。**唯一合法的形状值是：rect (*top*, *right*, *bottom*, *left*)，相对于左和上的方向画出显示窗口，**rect 矫正**

img

{ position:absolute;

clip:rect(10px 500px 200px 10px) }

**vertical-align**

图像和文字的上部还是下部对齐

|  |  |
| --- | --- |
| baseline | 默认。元素放置在父元素的基线上。 |
| sub | 垂直对齐文本的下标。 下边界和文字的下标对齐 |
| super | 垂直对齐文本的上标 下边界和文字的上标对齐 |
| top | 把元素的顶端与行中最高元素的顶端对齐 上边界对齐 |
| text-top | 把元素的顶端与父元素字体的顶端对齐 上边界对齐 |
| middle | 把此元素放置在父元素的中部。 中间对齐 |
| bottom | 把元素的顶端与行中最低的元素的顶端对齐。 |
| text-bottom | 把元素的底端与父元素字体的底端对齐。 |
| length | 将下表对齐设置为0；向上移动为正，向下移动为负 |
| % | 使用 "line-height" 属性的百分比值来排列此元素。 和上面的相似相对于文本的宽度上下调 |
| inherit | 规定应该从父元素继承 vertical-align 属性的值。 |

**z-index**

是否在同一个垂直维度上，决定是否遮挡，如果不在一个维度上就会有前后显示，决定那个在最前面，有透明属性的时候可以当背景用，可以为正负int值

Z-index 仅能在定位元素上奏效（例如 position:absolute;）！

相对定位

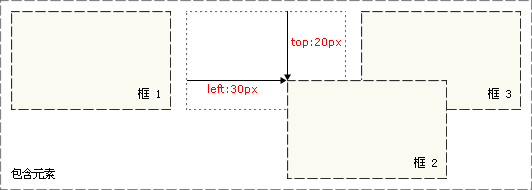
#box\_relative {

position: relative;

left: 30px;

top: 20px;

}



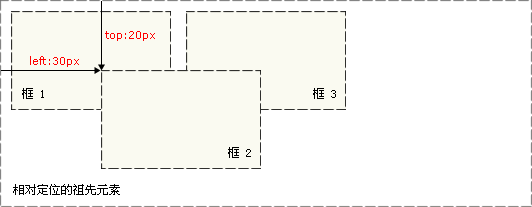
注意，在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。

绝对定位

绝对定位使元素的位置与文档流无关，因此不占据空间。这一点与相对定位不同，相对定位实际上被看作普通流定位模型的一部分，因为元素的位置相对于它在普通流中的位置。

普通流中其它元素的布局就像绝对定位的元素不存在一样：

如下图所示：

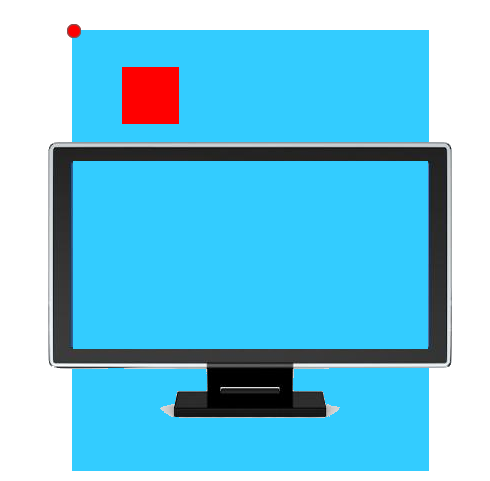


绝对定位的元素的位置相对于**最近的已定位祖先元素**，如果元素没有已定位的祖先元素，那么它的位置相对于**最初的包含块**。

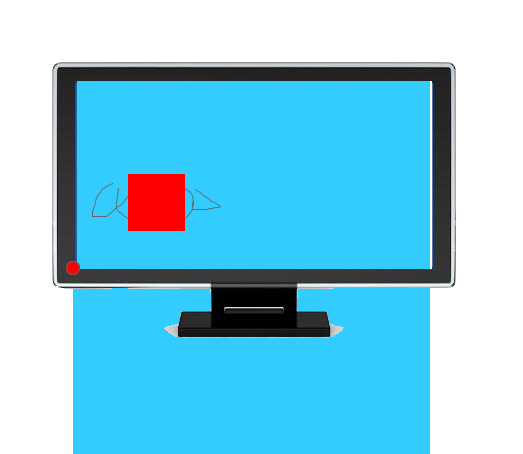
**提示：**因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其它元素。可以通过设置 [z-index 属性](http://www.w3school.com.cn/cssref/pr_pos_z-index.asp)来控制这些框的堆放次序。

参考点

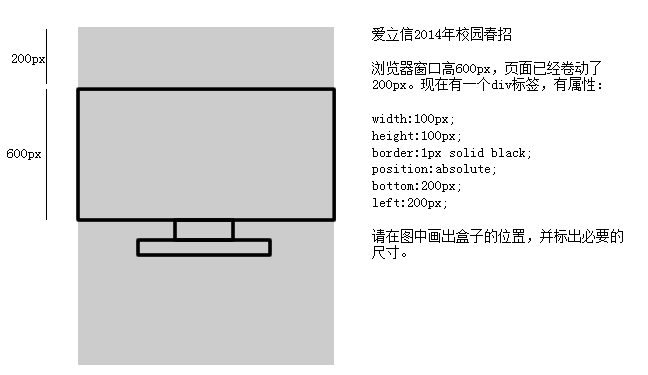
**绝对定位的参考点，如果用top描述，那么定位参考点就是页面的左上角，而不是浏览器的左上角：**



**如果用bottom描述，那么就是浏览器首屏窗口尺寸，对应的页面的左下角**：

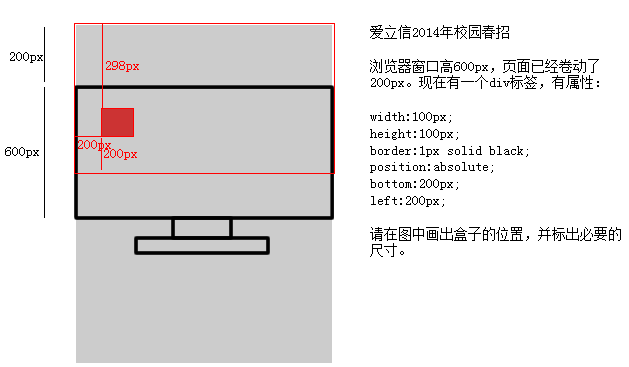


面试题：



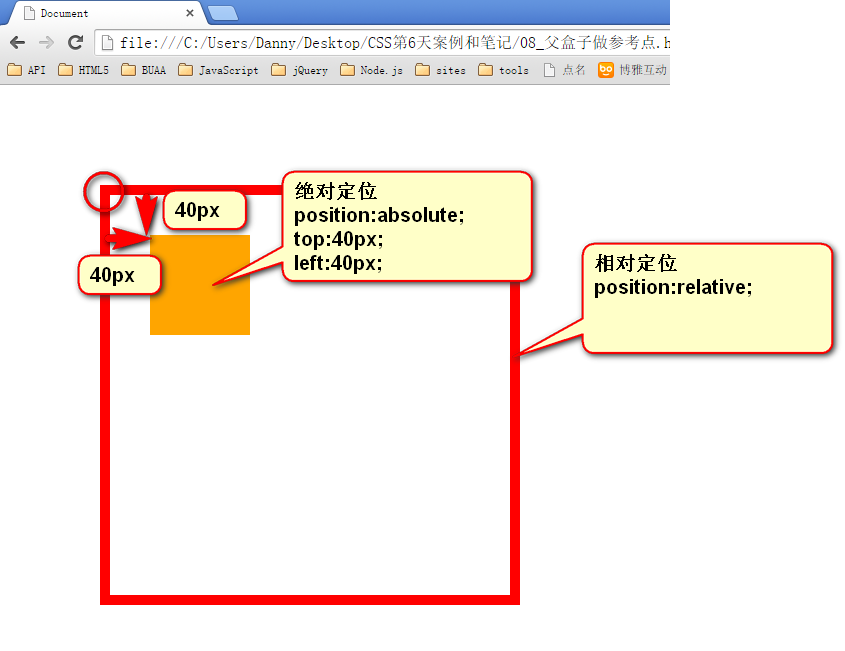
答案：

用bottom的定位的时候，参考的是浏览器首屏大小对应的页面左下角。



3.3 以盒子为参考点

一个绝对定位的元素，如果父辈元素中出现了也定位了的元素，那么将以父辈这个元素，为参考点。



● 要听最近的已经定位的祖先元素的，不一定是父亲，可能是爷爷：

|  |
| --- |
| 1. <div class="box1"> → 相对定位 2. <div class="box2"> → 没有定位 3. <p></p> → 绝对定位，将以box1为参考，因为box2没有定位，box1就是最近的父辈元素 4. </div> 5. </div> |

|  |
| --- |
| 1. <div class="box1"> → 相对定位 2. <div class="box2"> → 相对定位 3. <p></p> → 绝对定位，将以box2为参考，因为box2是自己最近的父辈元素 4. </div> 5. </div> |

● 不一定是相对定位，任何定位，都可以作为参考点

|  |
| --- |
| 1. <div> → 绝对定位 2. <p></p> → 绝对定位，将以div作为参考点。因为父亲定位了。 3. </div> |

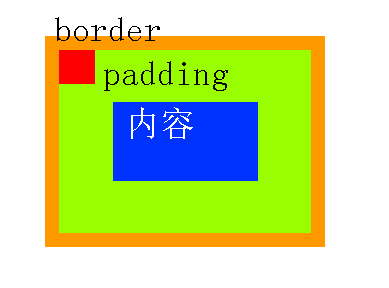
**子绝父绝、子绝父相、子绝父固**，都是可以给儿子定位的。但是，工程上子绝、父绝，没有一个盒子在标准流里面了，所以页面就不稳固，没有任何实战用途。工程上，“子绝父相”有意义，父亲没有脱标，儿子脱标在父亲的范围里面移动。

|  |
| --- |
| 1. <div class=”box1”> → 绝对定位 2. <div class=”box2”> → 相对定位 3. <div class=”box3”> → 没有定位 4. <p></p> → 绝对定位，以box2为参考定位。 5. </div> 6. </div> 7. </div> |

● 绝对定位的儿子，无视参考的那个盒子的padding。

下图中，绿色部分是div的padding，蓝色部分是div的内容区域。那么此时，div相对定位，p绝对定位。

p将无视父**亲的padding，在border内侧为参考点，**进行定位：



3.4 绝对定位的盒子居中

绝对定位之后，所有标准流的规则，都不适用了。所以margin:0 auto;失效。

|  |  |
| --- | --- |
|  |  |

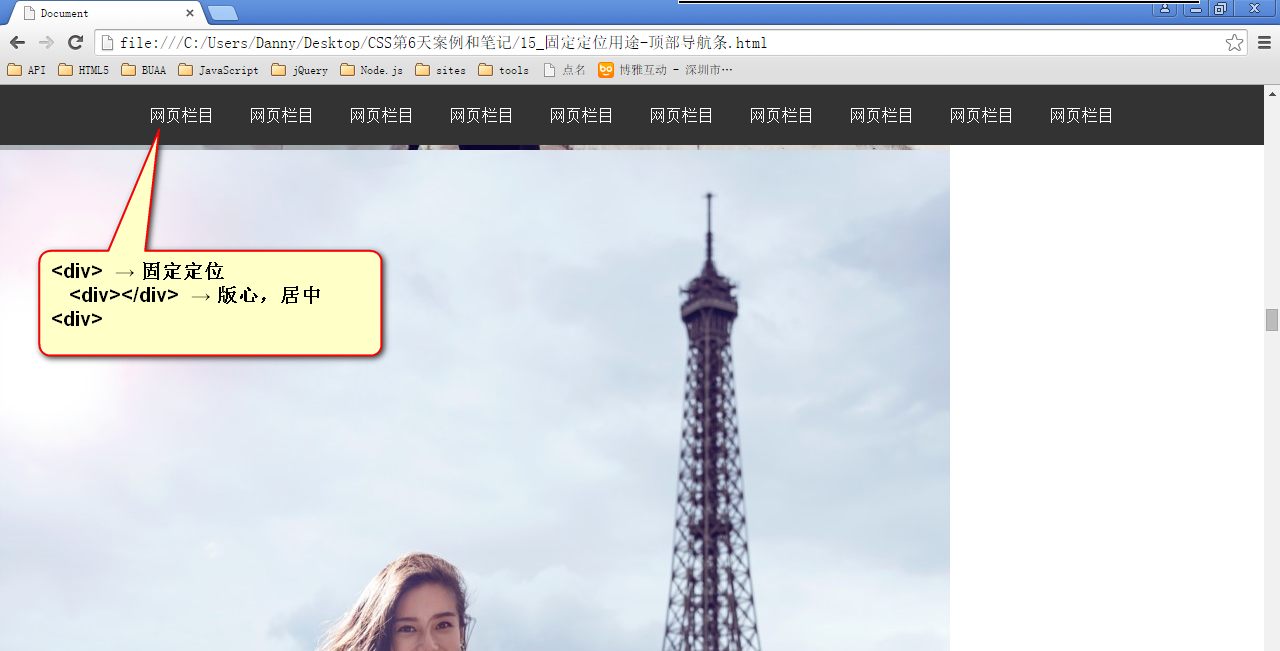
|  |
| --- |
| 1. width: 600px; 2. height: 60px;   **position: absolute;**   1. **left: 50%;** 2. top: 0; 3. **margin-left: -300px; → 宽度的一半** |

非常简单，当做公式记忆下来。**就是left:50%; margin-left:负的宽度的一半。**

固定定位

固定定位，就是相对浏览器窗口定位。页面如何滚动，这个盒子显示的位置不变。

固定定位脱标！



**IE6不兼容。**

z-index

● z-index值表示谁压着谁。数值大的压盖住数值小的。

● 只有定位了的元素，才能有z-index值。也就是说，不管相对定位、绝对定位、固定定位，都可以使用z-index值。而浮动的东西不能用。

● z-index值没有单位，就是一个正整数。默认的z-index值是0。

● 如果大家都没有z-index值，或者z-index值一样，那么谁写在HTML后面，谁在上面能压住别人。定位了的元素，永远能够压住没有定位的元素。

**● 从父现象：父亲怂了，儿子再牛逼也没用。**

**没有单位：**

|  |
| --- |
| 1. z-index: 988; |

**在做通栏的时候回出现好多错误**

浮动定位

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

Float属性

Float：rigth 浮动在文本外，文本遇到了会自动换行

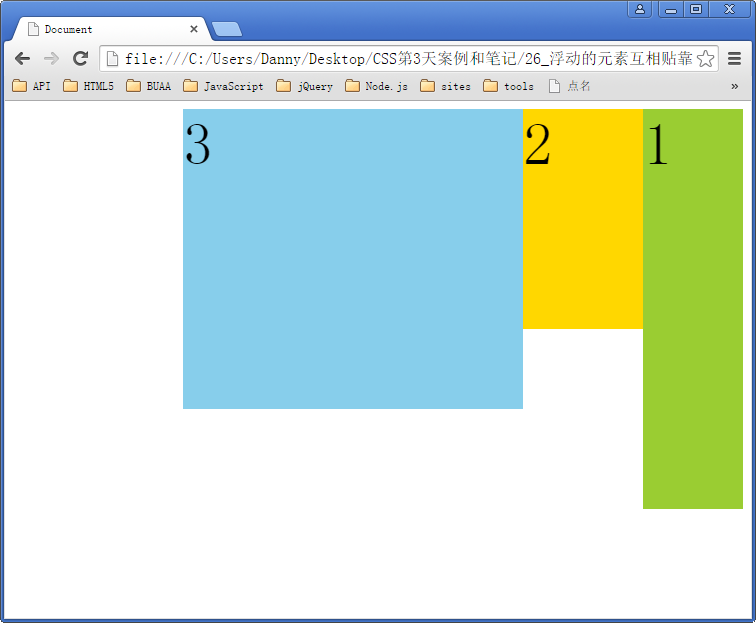
1. **旦一个元素浮动了，那么，将能够并排了，并且能够设置宽高了。无论它原来是个div还是个span。**
2. **5.2 浮动的元素互相贴靠**

**如果有足够空间，那么就会靠着2哥。如果没有足够的空间，那么会靠着1号大哥。**

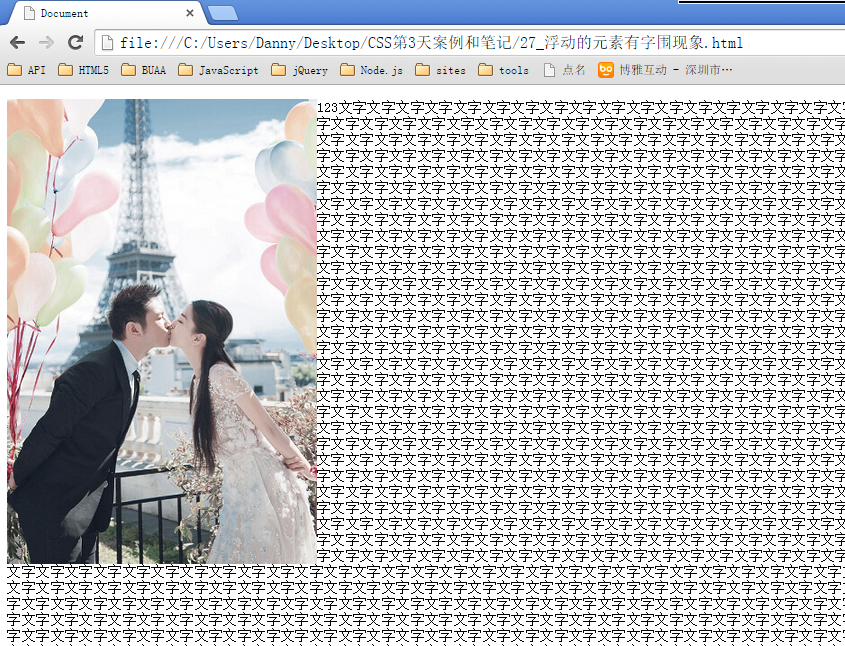
**如果没有足够的空间靠着1号大哥，自己去贴左墙。**

|  |  |  |
| --- | --- | --- |
|  |  |  |

右浮动： float:right;



1. 字围效果



div挡住了p，但是p中的文字不会被挡住，形成“字围”效果。

关于浮动我们要强调一点，浮动这个东西，我们在初期一定要遵循一个原则：

**永远不是一个东西单独浮动，浮动都是一起浮动，要浮动，大家都浮动。**

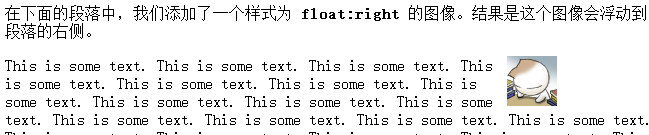
**第一个例子**

img

{ float:right;

margin-right:100px;

}



**第二个例子**

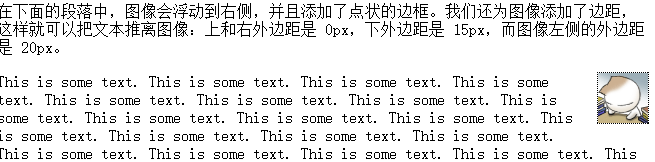
img {

**float:right;**

**border:1px dotted black;**

**margin:0px 0px 15px 20px;**

**}**



**带标题的图像浮于右侧**

div{

float:right;

width:120px;

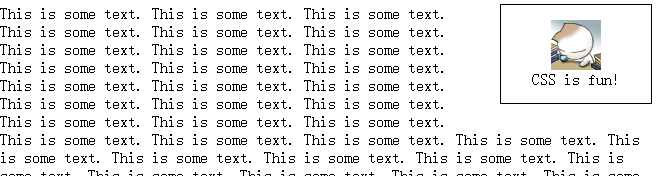
margin:0 0 15px 20px;

padding:15px;

border:1px solid black;

text-align:center;

}



**首字母浮于左侧**

span{

float:left;

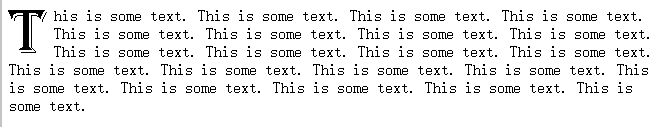
width:0.7em;

font-size:400%;

font-family:algerian,courier;

line-height:80%;

}



**创建水平菜单（导航列表）**

<style type="text/css">

ui{

**float:left;**

width:100%;

padding:0;

margin:0;

**list-style-type:none;**

}

a{

float:left;

width:7em;

text-decoration:none;

color:white;

background-color:purple;

padding:0.2em 0.6em;

border-right:1px solid white;

}

a:hover {background-color:#ff3300}

**li {display:inline} //uxia 段内元素**

</style>

**Clear**

浮动元素哪侧没有其他的元素

|  |  |
| --- | --- |
| **值** | **描述** |
| left | 在左侧不允许浮动元素。 |
| right | 在右侧不允许浮动元素。 |
| both | 在左右两侧均不允许浮动元素。 |
| none | 默认值。允许浮动元素出现在两侧。 |
| inherit | 规定应该从父元素继承 clear 属性的值。 |

**创建无表格的首页 float创建**

<head>

<style type="text/css">

div.container

{

width:100%;

margin:0px;

border:1px solid gray;

line-height:150%;

}

div.header,div.footer

{

padding:0.5em;

color:white;

background-color:gray;

clear:left;

}

h1.header

{

padding:0;

margin:0;

}

div.left

{

float:left;

width:160px;

margin:0;

padding:1em;

}

div.content

{

margin-left:190px;

border-left:1px solid gray;

padding:1em;

}

</style>

</head>

<body>

<div class="container">

<div class="header"><h1 class="header">W3School.com.cn</h1></div>

<div class="left"><p>"Never increase, beyond what is necessary, the number of entities required to explain anything." William of Ockham (1285-1349)</p></div>

<div class="content">

<h2>Free Web Building Tutorials</h2>

<p>At W3School.com.cn you will find all the Web-building tutorials you need,

from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>

<p>W3School.com.cn - The Largest Web Developers Site On The Net!</p></div>

<div class="footer">Copyright 2008 by YingKe Investment.</div>

</div>

</body>

Css分类

**显示为内联元素**

Display:none

**显示为块级元素**

**隐藏元素**

**Visibility：**

|  |  |
| --- | --- |
| visible | 默认值。元素是可见的。 |
| hidden | 元素是不可见的。 |
| collapse | 当在表格元素中使用时，此值可删除一行或一列，但是它不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。 |
| inherit | 规定应该从父元素继承 visibility 属性的值。 |

**光标属性**

**<span style="cursor:move">**

|  |  |
| --- | --- |
| *url* | 需使用的自定义光标的 URL。  注释：请在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标。 |
| default | 默认光标（通常是一个箭头） |
| auto | 默认。浏览器设置的光标。 |
| crosshair | 光标呈现为十字线。 |
| pointer | 光标呈现为指示链接的指针（一只手） |
| move | 此光标指示某对象可被移动。 |
| e-resize | 此光标指示矩形框的边缘可被向右（东）移动。 |
| ne-resize | 此光标指示矩形框的边缘可被向上及向右移动（北/东）。 |
| nw-resize | 此光标指示矩形框的边缘可被向上及向左移动（北/西）。 |
| n-resize | 此光标指示矩形框的边缘可被向上（北）移动。 |
| se-resize | 此光标指示矩形框的边缘可被向下及向右移动（南/东）。 |
| sw-resize | 此光标指示矩形框的边缘可被向下及向左移动（南/西）。 |
| s-resize | 此光标指示矩形框的边缘可被向下移动（南）。 |
| w-resize | 此光标指示矩形框的边缘可被向左移动（西）。 |
| text | 此光标指示文本。 |
| wait | 此光标指示程序正忙（通常是一只表或沙漏）。 |
| help | 此光标指示可用的帮助（通常是一个问号或一个气球） |

FW使用

新建时 默认72 的分辨率

Ctrl|+alt+r 标尺的快捷键 ，在标尺上拉出参考线，按住shift显示像素，用切图工具也可以实现

测上下的padding的时候，要先测出行高，做切片得到padding，

测左右padding的时候，文字一般直接测出左边的padding就行了，右侧可能换行

这时height为内容高，加上padding为盒子高

项目创建

规划好页面

项目文件夹

1.首页 index.html

2.样式 css文件夹 css文件 相同样式 （全局样式 公共样式）

Base.css（基本样式） global.css (全局样式)

3.图片 images 文件

4.特效 js文件夹 js 文件

Css 初始化

**@charset** "UTF-8";  
/\*css 初始化 \*/  
html, body, ul, li, ol, dl, dd, dt, p, h1, h2, h3, h4, h5, h6, form, fieldset, legend, img { margin:0; padding:0; }  
fieldset, img,input,button { border:none; padding:0;margin:0;outline-style:none; }  
ul, ol { list-style:none; }  
input { padding-top:0; padding-bottom:0; font-family: "SimSun","宋体";}  
select, input { vertical-align:middle; }  
select, input, textarea { font-size:12px; margin:0; }  
textarea { resize:none; } /\*防止拖动\*/  
img {border:0; vertical-align:middle; } /\* 去掉图片低测默认的3像素空白缝隙\*/  
table { border-collapse:collapse; }  
body {  
 font:12px/150% Arial,Verdana,"\5b8b\4f53";  
 color:#666;  
 background:#fff  
}  
.clearfix:before,.clearfix:after{  
 content:"";  
 display:table;  
}  
.clearfix:after{clear:both;}  
.clearfix{  
 \*zoom:1;/\*IE/7/6\*/  
}  
a{color:#666; text-decoration:none; }  
a:hover{color:#C81623;}  
h1,h2,h3,h4,h5,h6{text-decoration:none;font-weight:normal;font-size:100%;}  
s,i,em{font-style:normal;text-decoration:none;}  
.col-red{color: #C81623**!important**;}  
  
/\*公共类\*/  
.w { /\*版心 提取 \*/  
 width: 1210px;margin:0 auto;  
}  
.fl {  
 float:left  
}  
.fr {  
 float:right  
}  
.al {  
 text-align:left  
}  
.ac {  
 text-align:center  
}  
.ar {  
 text-align:right  
}  
.hide {  
 display:none  
}

Shortcut和结尾公共则在bass.css

Favicon

<link rel="shortcut icon" href="[favicon.ico](view-source:file:///C:/Users/andy/Desktop/%E4%BA%AC%E4%B8%9C%E7%BD%91%E7%AB%99/favicon.ico)" />

常见的标签一些性质

S del 删除线

I em 倾斜

U ins 下划线

字体加粗 font-weight: 700;

让字体不加粗: font-weight:normal;

字体倾斜: font-style:italic; 不用

字体不倾斜： font-style:normal;

不下划线 不删除线: text-decoration: none;

定位： position：static; 静态定位 约等于标准流

浮动的不浮动: float:none; none left right

定位的不定位: position: static; absolute relative fixed

标签 1 类 10 id 100 行内 1000

网页稳定：

Width 和height 最稳定

其次 padding

最后才考虑margin

正常流 normal flow

浮动 定位 脱标 out of flow

浮动目的：

可以让多个块级 元素 放到一行上。

Float: left right none;

搜索栏



搜索栏和下面的文字是两个不同的块

**搜索栏由input和button组成 ，button和img属于行内块元素。和其他元素之间总有间隔，因此链接在一起的方法是都浮动**

**Ctrl+l 定位行号**

淘宝多盒子并排显示边框和选中

**多个盒子并排的时候**：边界宽， margin-left：-1px；由于浮动的作用才会盖住

Div和其他标签都可以用hover了

Div：hover{

Border:1px solid red;

Position:relative; //防止遮挡

}

当并排的都有定位的时候

Div：hover{

z-index=1；

}

绝对定位的遮盖方法

当需要绝对定位来遮盖的时候，可以将这个绝对定位放在一个空的相对定位里，然后位于前面也采用相对定位，后来的在上面就可以制作定位了

定位同事出现left right

**定位的时候同时有left、right、top、buttom的时候，执行left 和top**

背景半透明

Css3

Background：rgba（0,0,0,0.5）； 红绿蓝 透明度

Opacity：0.5 盒子透明度 ，里面的文字什么的也就透明了

行内元素没有上下的padding margin 但是有左右的

Alert(“”);

两个相同的alert 就会依次弹出两个，程序是 要一步一步的执行的

虽然； 是不必须的，但我们要加上 ；，压缩就不能使用了

循环的话就可以轰炸浏览器，但是谷歌浏览器可以防轰炸

**括号里显示内容** 数字直接写、 字符串要加上 “”

控制台输出

Console .log()

Console.warn();

Console.error();

一行是一行

注释：

//注释 ctrel /

/\* \*/ 多行注释 ctrl shift /

Console 控制台 ，浏览器f12 显示出错误

字符串和数字 ：

+ 数字是数字相加 字符串是 连接符

变量 var variant

**只能由英语字母、数字、下划线、美元符号$构成，且不能以数字开头，并且不能是JavaScript保留字**。**支持汉字**

全局变量和局部变量

**直接使用也是可以的，并不一定需要var声明，没有声明直接使用的话会自动提升为全局变量**

**全局变量：（一个页面里面的script是相通的）**

1. 在往外面声明过的
2. 在函数里面出现的并且没有var声明的，函数调用过之后才是全局变量了

**局部变量**：

在函数内部声明的变量，带有var的，并且，声明之后如果和全局变量冲突的话会覆盖全局变量，但是仍然不会改变全局变量的数值

**大小写敏感**

**Undefined 定义**

Var a； 定义

A= 100; 赋值

Number

Boolean 直接用 true false

**typeof表示“某某的类型”**

JavaScript种，只要是个数，那么就是数值型的，**无论整浮、无论大小、无论正负，都是number类型的**。“浮”就是浮点数，就是“小数”，术语叫做“浮点数”。

运算符

算数运算符

**+— \* 和java一样 /是直接的 除并不会取整 % 取余**

**加号+有字符串就是连字符**

**但是 — \* / 会将字符串自动转换成数字 然后在进行计算**

3\*“5” 15 “3”\*“5”=15

乘方 Math.paw(a,b); ab

开方 Math.sqrt() 直接乘方 0.5

星期 （a+b）%7 由于初始没有0

弹出的输入框

var a = **prompt(**"请输入一个数字"**);** // 必须用一个变量，来接收用户输入的值，无论输入什么都是字符串类型的

字符串转成数字

**parseInt("365天每天都爱你10000次") 结果是 365** 后面的中文自动消失，只保留最开头的数字。 **parseInt()**

并且自动截断为整数 parseInt(5.6) 5 取整 不四舍五入

**用户输入一个三位数，用程序计算三位数字的和。**

比如：用户输入155，就弹出11

**var *c***=parseInt(prompt(**"请输入一行文字"**));  
***c***=(***c***-***c***%100)/100+(***c***%100-***c***%10)/10+***c***%10;

傻逼了吧

Var baiwei=parseInt(c/100);

Var shiwei=parseInt(c%100/10);

Var gewei=c%10;

关系运算符

**< >= <=** 和java一样

==验证字符串是否相同

“我爱北京天安门”==”我爱北京天安门” true 很奇怪吧

不严谨。会将不同类型的东西转换为相同类型进行比较

“5”==5 true

6==“6” true

因此衍生出 === 全等于三个等号

56===“56” false

！= 是 ==的反面 ！==是===的反面

**逻辑运算符**

&& || ！ 没有短路不短路的问题

连比 3<2 && 3>5

**if else if 选择语句 下一楼层暗含上面的楼层都不满足**

if else if else if 终止于第一个满足的条件

**则实际应用中连续的条件就可以不写出上半部分了**

**循环语句**

**for（var i=1;1<=100;i++）{**

**}**

**好像在这里没有局部变量的概念**

**<script** type=**"text/javascript"**>  
 **for**(**var *i***=1;***i***<=100;***i***++){  
 **console**.log(***i***);

Var sum=9；  
 }  
 **console**.log(***i***);  **101**

**Console.log(sum) 9**</**script**>

**I++ 和 ++I 与java相同**

例子：

被5 6 整除

if(num % 5 == 0 && num % 6 == 0){

alert("能被5、6同时整除");

}else if(num % 5 == 0){

alert("能被5整除，但6不");

}else if(num % 6 == 0){

alert("能被6整除，但5不");

}else{

alert("被5、6都不能");

}

**求因子**

可以先使用在赋值之后直接放在条件语句中

**var *num***=parseFloat(prompt(**"请输入一个数字"**));  
**var *yinshu***=**""**;  
 **for**(**var *i***=1;***i***<=***num***;***i***++){  
 **if**(***num***%***i***==0){  
 ***yinshu***=***yinshu***+**" "**+ ***i***;  
 }  
 }  
 alert(***yinshu***);

赋值

可以a=b=1；

这样就a=1；b=1；了

函数中

Var a=b=1;等价于var a=1；b=1； a是局部变量，b在函数调用之后是全局变量

Var a = 10, b = 20;

等价的

Var a = 10;

Var b = 20;

函数内部的问题

function f1(){

console.log(a); undifine 声明提升，但是没赋值

console.log(b); is not difined 报错

var a=b=1; 这里b是全局但是也得等调用之后

}

// console.log(b); is not difined 报错

f1();

console.log(b); 1

当声明和赋值在外面的时候，不论何时，都会被直接提到最前面

函数

**函数，是一种封装。就是将一些语句，封装到函数里面。通过调用的形式，执行这些语句。**

函数的使用，是两个步骤，**第一步，函数的定义**：

语法：

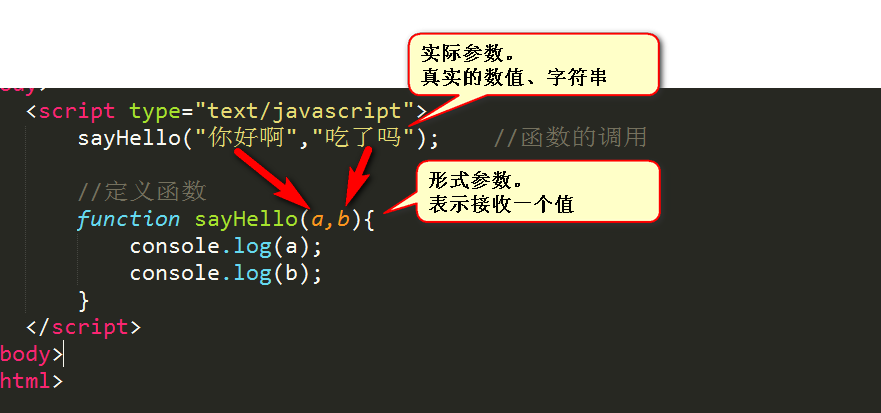
|  |
| --- |
| 1. function 函数名字(){ 3. } |

Function后面要加上 空格

调用

函数名字（）

**带参数的函数，定义函数的时候不能写var**



函数的使用

function fun() { 语句 }

fun 是函数体的意思 不会立即执行

fun() 调用函数 的意思，会立即执行

function fun() {  
 return 3;  
}  
**console**.log(fun); // 返回函数体 function fun() { retrun 3}  
**console**.log(fun()); // 调用函数 3 返回的是结果  
fun();  
**window**.onresize = 3  
**window**.onresize = function fun() { retrun 3}

**返回值**

1. <script type="text/javascript">
2. console.log(**qiuhe(3,4)**);
3. function qiuhe(a,b){
4. **return** a + b;
5. }

</script>

没有类型，直接返回就可以了

Return 后面必须有东西， 如

Return{

A:10，

B:00

}

不能写成

Return

{

A:10，

B:00

}

**//斐波那契数列**

//1、1、2、3、5、8、13、21、34……

//输出前100项的斐波那契数列

for(var i = 1 ; i <= 100 ; i++){

console.log(fib(i));

}

//传入n，返回第n位的数。

//比如传入6，返回8

//传入7，返回13

//传入3，返回2

function fib(n){

if(n == 1){

return 1;

}else if(n == 2){

return 1;

}else{

return fib(n-1) + fib(n-2);

}

}

**获取元素**

**Getelimentbbyid**

事件

事件源

要触发的对象 手 用手去触发的。 谁触发了

一般情况下 是 个名词

发起者

被触发者 开关按钮

事件

怎么触发的这个事情 按

|  |  |
| --- | --- |
| 事件名 | 说明 |
| onclick | 鼠标单击 |
| ondblclick | 鼠标双击 |
| onkeyup | 按下并释放键盘上的一个键时触发 |
| onchange | 文本内容或下拉菜单中的选项发生改变 |
| onfocus | 获得焦点，表示文本框等获得鼠标光标。 |
| onblur | 失去焦点，表示文本框等失去鼠标光标。 |
| onmouseover | 鼠标悬停，即鼠标停留在图片等的上方 |
| onmouseout | 鼠标移出，即离开图片等所在的区域 |
| onload | 网页文档加载事件 |
| onunload | 关闭网页时 |
| onsubmit | 表单提交事件 |
| onreset | 重置表单时 |

Oninput

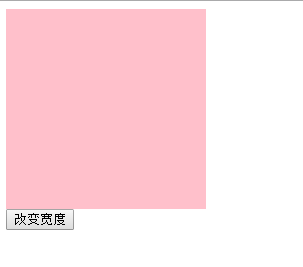
事件处理程序

发生了什么事 灯亮了

= function(){ }

总结

事件源.事件 = function(){ 事件处理函数 }



<script>

var box=document.getElementById("box");

**var cw=document.getElementById("cw"); 必须用“”包围**

**cw.onclick=function(){**

box.style.width="1000px";

}

</script>

**cw.onclick=function(){**

box.style.width="1000px";

}

直接 **事件源.事件=function（）{ 这尼玛是匿名函数啊**

box.style.width="1000px"; 属性值要用“”包裹

}

必须这么写，后面还不能直接用一个函数代替，函数只能放在事件处理程序里面了

关闭广告

<**a** href=**"javascript:;"** class=**"close-banner"** id=**"jd\_close"**></**a**>

A里面必须写 href=“javascipt：；”

鼠标经过切换图片 onmouseover onmouseout

<img src="images/jd1.png" alt="" id="jd\_icon"/>

<script>

var jd\_icon=document.getElementById("jd\_icon");

jd\_icon.**onmouseover**=function(){

jd\_icon.src="images/jd2.png";

}

jd\_icon.onmouseout=function(){

jd\_icon.src="images/jd1.png";

}

修改背景

<script>

**window.onload = function(){ 入口函数**

/\*要做事，先找人\*/

var pic1 = document.getElementById("pic1");

var pic2 = document.getElementById("pic2");

var pic3 = document.getElementById("pic3");

pic1.onclick = function(){

document.body.style.backgroundImage = "url(images/1.jpg)";

}

pic2.onclick = function(){

document.body.style.backgroundImage = "url(images/2.jpg)";

}

pic3.onclick = function(){

document.body.style.backgroundImage = "url(images/3.jpg)";

}

}

</script>

模态框 两个fixed盒子 一个999 一个1000



<script type="text/javascript">  
 **window**.onload = function(){  
 //事件源: 登录  
 var login = **document**.getElementById("login");  
 var mask = **document**.getElementById("mask");  
 var box = **document**.getElementById("box");  
 login.onclick = function(){  
 // 当我们点击登录，会弹出灰色的大盒子和白色的小盒子  
 mask.**style**.**display** = "block";  
 box.**style**.**display** = "block";  
 }  
 // 事件源 span x  
 var close\_all = **document**.getElementById("close\_all");  
 close\_all.onclick = function(){  
 mask.**style**.**display** = 'none';  
 box.**style**.**display** = "none";  
 }  
 }  
 </script>  
</head>

隐藏样式

Display: none display: block ; 显示的意思

Visibility: hidden; visibility: visible 显示的意思

Display 隐藏不占位置

Visibility:hidden 隐藏占有位置 停职留心

Overflow:hidden; 隐藏超出的部分。

Js 的书写位置

Js 的书写位置非常的自由。

也可以参照 css 的位置来分类。

行内式

<button onclick="alert('你好吗')">点击我</button>

一般情况，单双引号是一样 的 但是出现 了包裹的情况。

我们一般采取的是 外双内单的格式。

<a href=”javascript:;”></a>

<a href=”javascript:void(0);”></a>

内嵌式

<script type=”text/javascript”> </script> 任何一个地方

外链式

<script type=”text/javascript” src=”xx.js”></script>

这对标记之间不能写任何的东西。

数据类型

Undifine

String 字符型

**转换成字符型：**

直接加“”

String() 大写的

+””

Boolean

就两个值 正确的和错误的 true 和 false

!!转换成布尔型

Boolean（；）

false、undefined 、null、0、”” 为 false

true、1、”somestring”、[Object] 为 true

Number

数值的前面带 0 表示 八进制

Var num = 020;

0\*80+2\*81 = 16

数值的前面带 0x 表示 十六进制

var result = 0xb; 11

转换为数值型:

1. 利用 - \* / 都可以转换

2 利用Number( )

运算符

算数运算符

**+— \* 和java一样 /是直接的 除并不会取整 % 取余**

**不能小数相加，会溢出**

**加号+有字符串就是连字符**

**但是 — \* / 会将字符串自动转换成数字 然后在进行计算**

3\*“5” 15 “3”\*“5”=15

乘方 Math.paw(a,b); ab

开方 Math.sqrt() 直接乘方 0.5

星期 （a+b）%7 由于初始没有0

弹出的输入框

var a = **prompt(**"请输入一个数字"**);** // 必须用一个变量，来接收用户输入的值，无论输入什么都是字符串类型的

字符串转成数字

**parseInt("365天每天都爱你10000次") 结果是 365** 后面的中文自动消失，只保留最开头的数字。 **parseInt()**

并且自动截断为整数 parseInt(5.6) 5 取整 不四舍五入

**用户输入一个三位数，用程序计算三位数字的和。**

比如：用户输入155，就弹出11

**var *c***=parseInt(prompt(**"请输入一行文字"**));  
***c***=(***c***-***c***%100)/100+(***c***%100-***c***%10)/10+***c***%10;

傻逼了吧

Var baiwei=parseInt(c/100);

Var shiwei=parseInt(c%100/10);

Var gewei=c%10;

关系运算符

**< >= <=** 和java一样

==验证字符串是否相同

“我爱北京天安门”==”我爱北京天安门” true 很奇怪吧

不严谨。会将不同类型的东西转换为相同类型进行比较

“5”==5 true

6==“6” true

因此衍生出 === 全等于三个等号

56===“56” false

！= 是 ==的反面 ！==是===的反面

**逻辑运算符**

&& || ！ 没有短路不短路的问题

连比 3<2 && 3>5

**if else if 选择语句 下一楼层暗含上面的楼层都不满足**

if else if else if 终止于第一个满足的条件

**则实际应用中连续的条件就可以不写出上半部分了**

ParseInt() parseFloat()

parseInt(值, 进制);

NOT a number

MMD

BBD

**parseInt(110,2)**

表示2进制 吧10 这个2进制转换为 10进制

0\*20+1\*21  + 1\*22 = 6

1.var a="15.15abc" , b='10.15' , c='10.0abc';

alert(parseInt(a)+Number(b)+parseFloat(c));

**parseInt("365天每天都爱你10000次") 结果是 365** 后面的中文自动消失，只保留最开头的数字。 **parseInt()只有开头是数字才行；否则返回NAN**

并且自动截断为整数 parseInt(5.6) 5 取整 不四舍五入

**用户输入一个三位数，用程序计算三位数字的和。**

比如：用户输入155，就弹出11

**var *c***=parseInt(prompt(**"请输入一行文字"**));  
***c***=(***c***-***c***%100)/100+(***c***%100-***c***%10)/10+***c***%10;

傻逼了吧

Var baiwei=parseInt(c/100);

Var shiwei=parseInt(c%100/10);

Var gewei=c%10;

Null undefined

Null 空的 没有值 。

Undefined 未定义的 应该有值，但是没有给。

Null

函数(function)

函数： 函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

四大发明 造纸术 火药 指南针 印刷术

函数的声明

函数使用跟变量一样，需要 声明

自定义函数

function fun(){  
 alert("我是自定义函数")  
}  
fun(); // 函数不调用，自己不执行

函数直接量声明

var fun1 = function(){  
 alert("直接量声明")  
}  
fun1(); 也需要调用

利用Function 关键字声明

var fun2 = new **Function**("var a = 10; var b = 20; alert(a+b)");  
fun2();

变量声明提升函数（函数内部存在）

什么是变量提升

1. function fun(){
2. console.log(num);
3. var num = 20;
4. }

相当于 ---

1. function fun(){
2. var num;
3. console.log(num);
4. Num = 20;
5. }

在函数体内部，声明变量，会把该声明提升到函数体的最顶端。 只提升变量声明，不赋值。

var a = 18;  
f1();  
function f1(){  
 var b=9;  
 **console**.log(a);  
 **console**.log(b);  
 var a = '123';  
}

结果是： undefined 9

Var a = 10, b = 20;

等价的

Var a = 10;

Var b = 20;

函数内部的问题

function f1(){

console.log(a); undifine 声明提升，但是没赋值

console.log(b); is not difined 报错

var a=b=1; 这里b是全局但是也得等调用之后

}

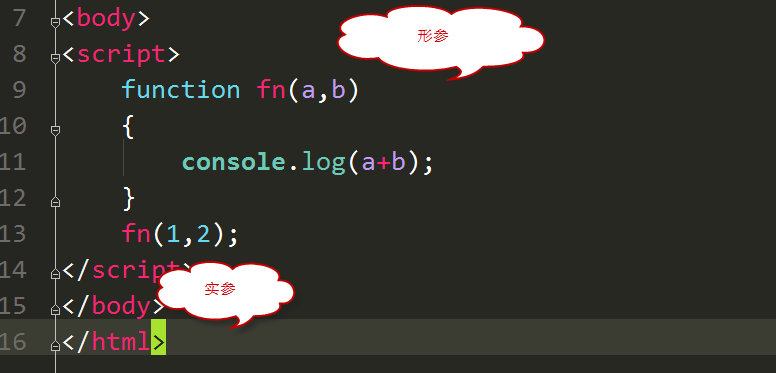
// console.log(b); is not difined 报错

f1();

console.log(b); 1

当声明和赋值在外面的时候，不论何时，都会被直接提到最前面

函数参数 arguments



arguments是存储了函数传送过过来实参 ， 上面例子直接输出的话是log（argument）；【1,2】；argument.Length 为实参的个数 fun1.length为形参的个数

Javascript在创建函数的同时，会在函数内部创建一个arguments对象实例.

arguments对象只有函数开始时才可用。函数的 arguments 对象并不是一个数组，访问单个参数的方式与访问数组元素的方式相同

arguments对象的长度是由实参个数而不是形参个数决定的

<script>  
 function fn(a,b)  
 {  
 **console**.log(fn.**length**); //得到是 函数的形参的个数  
 //console.log(arguments);  
 **console**.log(arguments.**length**); // 得到的是实参的个数  
 if(fn.**length** == arguments.**length**)  
 {  
 **console**.log(a+b);  
 }  
 else  
 {  
 **console**.error("对不起，您的参数不匹配，正确的参数个数为:" + fn.**length**);  
 }  
 //console.log(a+b);  
 }  
 fn(1,2);  
 fn(1,2,3);  
</script>

Var a = 10, b = 20;

等价的

Var a = 10;

Var b = 20;

函数内部的问题

function f1(){

console.log(a); undifine 声明提升，但是没赋值

console.log(b); is not difined 报错

var a=b=1; 这里b是全局但是也得等调用之后

}

// console.log(b); is not difined 报错

f1();

console.log(b); 1

当声明和赋值在外面的时候，不论何时，都会被直接提到最前面

切换背景图





上面的代码出问题了

For的拓展

当页面加载完的时候，for其实已经遍历完了，此时的i只是当前存在的最大的那个不满足的数字，

**解决方案**

添加一个记录i的变量；

闭包一下？？？？？鬼知道怎么弄的

Input对话框的操作

获得焦点 失去焦点 事件

我们前面学过了 onclick 点击 onmouseover onmouseout’

获得焦点: **onfocus**  fao ~克死

失去焦点: **onblur**  不len ~~

自动获取焦点

<script>  
 **window**.onload = function(){  
 var txt = **document**.getElementById("txt");  
 txt.focus(); //自动获得焦点  
 }  
 </script>



<script>  
 **window**.onload = function(){  
  
 var txt = **document**.getElementById("txt");  
  
 txt.onfocus = function(){ //得到焦点  
 //alert("得到了焦点");  
 //什么时候该清空呢  
 //用户没有输入的时候，用户第一次使用的时候  
 // 如果 这里input 里面的文字 是 请输入... 说明用户没有用过，就应该清空  
 if(txt.**value** == "请输入...")  
 {  
 txt.**value** = "";  
 txt.**style**.**color** = "#333";  
 }  
 }  
  
 txt.onblur = function(){ //失去焦点  
 //alert("失去了焦点");  
 //什么时候再还原呢？  
 //input的值是 空的时候，我们再复原  
 if(txt.**value** == "")  
 {  
 txt.**value** = "请输入...";  
 txt.**style**.**color** = "#ccc";  
 }  
 }  
 }  
</script>

This

指的是本身

this 主要是指**事件的调用者** 。 一般只用在事件的内部

**className 类名**

$("result").**className** ="wrong";

**innerHTML Input.value**

更换 盒子里面的内容 文字 标签都换.

表单更换内容

Button 和submit

<button value="TTT">FFF</button>

<input type="submit" value="fff">

IsNaN()

isNaN 如果里面的不是个数字 返回 true 否则返回false

isNaN(“12”) F

isNaN(12) F

isNaN(“HAH”) T

布尔值也是数字 只是parseint无法转换出来

很弱会自动转化

鼠标经过选择表单

sele.onmouseover = function(){  
 this.select(); //选择  
}

方法 select() 选择功能

getElementsByTagName() 获取某类标签

前面我们可以得到一个盒子 通过 id 获得

getElementById() 只得到一个 盒子

我们想要获取某类标签 比如说所有 的div li span

getElementsByTagName(); 很多个所以是复数 很多个

得到的是一个伪数组。

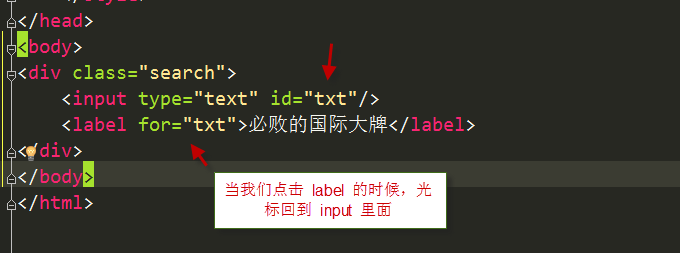
Lis 数组

Lis[索引号] 一个

用户输入事件，高端的使用搜索栏中的东西

正常浏览器 : oninput

Ie 678 支持的 ： onpropertychange



**Label for id 之后点击label可以进入input，并且 cusor：text；**

**script**>  
 **window**.onload = **function** () {  
 **function** *$id*(id) {  
 **return document**.getElementById(id);  
 }  
 **var** input = *$id*(**"input"**);  
 **var** label = *$id*(**"label"**)  
 *$id*(**"input"**).focus(); *//自动获取焦点* input.oninput = input.onpropertychange=**function** () {  
 **if** (input.**value** == **""**) {  
 label.**style**.**display** = **"block"**;  
 } **else** {  
 label.**style**.**display** = **"none"**;  
 }  
 }  
 }  
</**script**>

数组

声明数组

var arr = [1,3,5,7,9];

var arr = new Array(1,3,5);

使用数组

使用的方法: **数组名[索引值];**  函数名();

索引号是从0开始的。 0 1 2 3 4 5 ...

var textArr = ["刘备","诸葛亮","赵云","关羽"];  
**console**.log(textArr[3])

数组的长度

数组名.length;

遍历数组

for(var i = 0，len=lis.length; i<len; i++)   
 长度单独弄出来定义会节省资源

想到了for 遍历 数组

getElementsByTagName() 得到一个伪数组

案例 隔行变色

<script>  
 **window**.onload = function(){  
 var lis = **document**.getElementsByTagName("li"); // 得到所有的li  
 //alert(lis.length);  
 for(var i = 0; i<lis.**length**; i++)  
 {  
 if(i%2 == 0) // 只有偶数能被2整除  
 {  
 lis[i].**style**.**backgroundColor** = "#eee";  
 }  
 //鼠标经过li 的时候， 当前的底色变亮  
 lis[i].onmouseover = function(){  
 this.**className** = "current";  
 }  
 lis[i].onmouseout = function(){  
 this.**className** = "";  
 }  
 }  
 }  
</script>

遍历京东的小图标

获取某个标签里的其他标签

Var var=**document.getElementById(“id”).getElementsByTagName(“name”;**

Input传递的值

所有的input 取过来的值 是 字符型。

先要算数运算要先 Number（）

**全选和反选**

/\*全选和取消 函数\*/

**全选**

function all(flag){

for(var i=0;i<inputs.length;i++)

{

inputs[i].checked = flag;

}

}

btns[0].onclick = function(){

all(true);

};

btns[1].onclick = function(){

all(false);

};

/**/反选**

btns[2].onclick = function(){

for(var i=0;i<inputs.length;i++)

{

inputs[i].checked == true ? inputs[i].checked = false : inputs[i].checked = true;

}

}

排他思想

排他思想:

首先干掉所有人， 重铸我自己

<script>  
 **window**.onload = function(){  
 var btns = **document**.getElementsByTagName("button");  
 for(var i=0;i<btns.**length**;i++)  
 {  
 btns[i].onclick = function(){  
 //清除所有人的 类名 只能用 for 遍历  
 for(var j=0;j<btns.**length**;j++)  
 {  
 btns[j].**className** = "";  
 }  
 //就剩下自己 就是一个 而且是点击那个  
 this.**className** = "damao";  
 }  
 }  
 }  
</script>



修改选中的东西

属性

为数组的元素添加一个属性，就可以记录自己的顺序了，然后就可以使用

**This.属性**

**获取当前的引用**

Tab栏切换

设计页面的时候。要遍历的部分就不要使用类了，不同的部分在类里面显示出来，记得提升该类的样式的权重；

默认的初始化标签在里面直接添加上面的类就行了；

Js部分；

大循环：定义次序自身属性 ，加载事件

小循环：清楚全部，并重铸自己，由自身属性确定目标位置

<style>  
 \*{  
 margin: 0;  
 padding: 0;  
 }  
 #first{  
 height: 300px;  
 width:300px;;  
 margin:100px auto;  
 }  
 .spanbox{  
 overflow: hidden;  
  
 }  
 .spanbox span{  
 float:left;  
 height: 50px;  
 width: 99px;  
 background-color: #a3a3a3;  
 font:25px/50px "simsun","microsoft yahei";  
 color: #fff;  
 border-left:1px solid #fff;  
 }  
 #first .choose{  
 background-color: #f00;  
 }  
 #first .show{  
 display: block;  
 }  
 #first ul{  
 list-style-type:none;  
 }  
 #first li{  
 height: 250px;  
 width:300px;  
 background-color: #000;  
 font:25px/50px "simsun","microsoft yahei";  
 color: #fff;  
 display: none;  
 }  
  
  
 </style>  
 <script src="tab标签用到的.js"></script>  
</head>  
<body>  
<div id="first">  
 <div class="spanbox"><span class="choose">1</span><span>2</span><span>3</span>  
 </div>  
 <ul>  
 <li class="show">1个盒子显示</li>  
 <li>2个盒子显示</li>  
 <li>3个盒子显示</li>  
 </ul>  
</div>

window.onload= **function** () {  
 show("first");  
 **function** show(id){  
 **var** box=document.getElementById(id);  
 **var** span=box.getElementsByTagName("span");  
 **var** li=box.getElementsByTagName("li");  
 **for**(i=0,len=span.length;i<len;i++){  
 span[i].index=i;  
 span[i].onmouseover=**function**(){  
 **for**(j=0;j<span.length;j++){  
 span[j].className="";  
 li[j].className="";  
 }  
 **this**.className="choose";  
 li[**this**.index].className="show";  
  
 }  
 }  
  
 }  
}

两个小循环

循环 for(初始化; 条件; 增量){}

while() 当 do {} while()

while(条件) { 语句 }

var j = 1;  
while(j<=100)  
{  
 sum1+=j;  
 j++;  
}  
**console**.log(sum1);

do while 至少执行一次 while 不一定

switch

switch 跟 if else if else if else 几乎一样的 但是switch效率更好。

作用其实就是 ： 多选1 从多个里面选1个 。

语法格式：

switch(参数)

{

case 参数1：

语句;

break; 退出的意思

case 参数2：

语句;

break; 退出的意思

........

default: 默认的

语句;

}

下拉菜单选择背景 onchange

<select name="" id="select">  
 <option value="1">红色</option>  
 <option value="2">黄色</option>  
 <option value="3">蓝色</option>  
 <option value="4">绿色</option>  
</select>

Onchange

window.onload=**function**(){  
 **var** select=document.getElementById("select");  
 **var** mybody=document.getElementsByTagName("body")[0];  
 select.onchange=**function**(){  
 **switch** (**this**.value){  
 **case** "1":  
 console.log(1);  
 mybody.style.backgroundColor="#f00";  
 **break**;  
 **case** "2":  
 console.log(2);  
 mybody.style.backgroundColor="#0f0";  
 **break**;  
 **case** "3":  
 console.log(3);  
 mybody.style.backgroundColor="#00f";  
 **break**;  
 **case** "4":  
 console.log(4);  
 mybody.backgroundColor="#000";  
 **break**;  
 }  
 }  
}

case 里面赢得value必须是加上 ““ 的”

获取body标签

Document.body

背景色

**Document.body.style.backgroundColor="#00f";**

背景图

**Document.body.backgroundImge=“url（）”;**

**表单提供的值只有字符串，因此在switch和if中都是字符串选项**

星座（下拉条改变大量的东西）

那堆星星；

两个放在一起的图片。压着放，通过改变上面的宽度来显示完整的信息

数组常用的方法

添加：

Var arr=[1,2,34,4]

**Arr.push(7); 并且返回现在数组的长度**

**Arr.unshift(8);从数组的前面加入**

取出删除

arr.pop() → 结果 [1,3] 删除最后一个。返回的最后一个

arr.shift() 删除第一个元素， 并返回第一个值

**var** arr=[1,2,3];  
arr.push(4); //返回4，数列的长度  
console.log(arr); //[1,2,3,4]  
arr.unshift(0); //返回5  
console.log(arr); //[0,1,2,3,4]  
console.log(arr.pop()); //返回最后一个元素并删除  
console.log(arr); //[0,1,2,3,]  
console.log(arr.shift()); //返回第一个元素并删除  
console.log(arr); //[1,2,3]

连接两个数组

concat()该方法用于连接两个或多个数组。它不会改变现有的数组，而仅仅会返回被连接数组的一个副本

var aa = [1,3,5]; var bb = [“a”,”b”,”c”];

var c= aa.concat(bb);

把数组转换成字符串

var arr = [1,2,3];

console.log(arr.join(“-”)) 结果就是： 1-2-3 字符串

把字符串转换成数组

**var** a="a-b-c";  
**var** b= a.split("-");  
console.log(b);

D0M

我们js 有三部分组成的？

ECMAscript DOM BOM

核心（ECMAScript）欧洲计算机制造商协会

描述了JS的语法和基本对象。 var aa var AA 不一样

**文档对象模型（DOM）** 学习的重点

处理网页内容的方法和接口

浏览器对象模型（BOM） 了解一下

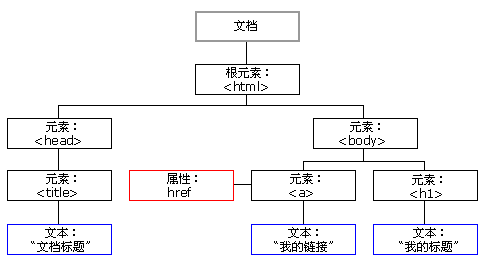
与浏览器交互的方法和接口

window.alert() 很大的兼容问题

DOM 为文档提供了结构化表示，并**定义了如何通过脚本来访问文档结构。**

目的其实就是为了能让js操作html元素而制定的一个规范。

DOM 树



访问节点（标签 属性 文本）

getElementById() id 访问节点

getElementsByTagName() 标签访问节点

直接document. getElementsByTagName() 获取的是整个页面的额】

Id. getElementsByTagName() id 下面的

getElementsByClassName() 类名 有兼容性问题

主流浏览器支持 ie 67 8 不认识

怎么办？ 我们自己封装自己的 类 。

封装自己的getByClass方法、

<script>  
 **window**.onload = function(){  
 //封装自己class类名  
 function getClass(classname){  
 //如果浏览器支持，则直接返回  
 if(**document**.getElementsByClassName)  
 {  
 return **document**.getElementsByClassName(classname);  
 }  
 // 不支持的 浏览器  
 var arr = []; // 用于存放满足的数组  
 var dom = **document**.getElementsByTagName("\*");  
 //alert(dom.length);  
 for(var i=0;i<dom.**length**;i++)  
 {  
 if(dom[i].**className** == classname)  
 {  
 arr.push(dom[i]);  
 }  
 }  
 return arr;  
 }  
 **console**.log(getClass("demo").**length**);  
 }  
</script>

a+b+c=120；

1. b=30；

a+b-c=100;

2a-c=130;

2a+c=150;

C=10;

判断真假

|  |  |
| --- | --- |
| 我们用条件语句来判断5大数据类型中的真假； | |
| 数据 | 结论 |
| 数字类型 | 所有数字都是真，0是假 |
| 字符串 | 所有字符串都是真，’ ’串是假 |
| 对象 | 所有对象都是真，null是假 |
| 未定义 | undefined是假，没有真 |

访问关系



父节点 小关大

父 ： parentNode 亲的 一层

<script>  
 **window**.onload = function(){  
 var x = **document**.getElementById("x");  
 x.onclick = function(){  
 this.parentNode.**style**.**display** = "none";  
 // 关掉的是他的 父亲  
 }  
 }  
</script>

兄弟节点

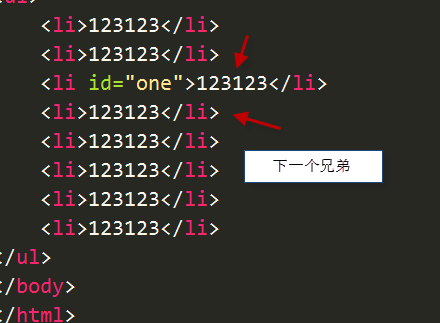
nextSibling 下一个兄弟 亲的 ie 678 认识

nextElementSibling 其他浏览器认识的

previousSibling 同理 上一个兄弟

previousElementSibling

我们想要兼容 我们可以合写 || 或者



var div = one.**nextElementSibling** || one.**nextSibling**;  
div.**style**.**backgroundColor** = "red";

必须先写 **正常浏览器 后写 ie678**

子节点

firstChild 第一个孩子 ie678

firstElementChild 第一个孩子 正常浏览器

var one.firstElementChild || one.firstChild;

lastChild

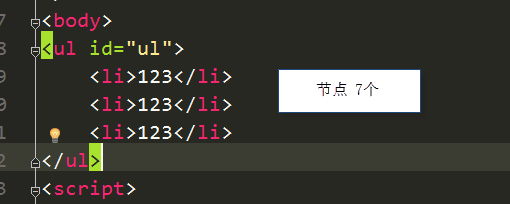
lastElementChild

孩子节点

childNodes 选出全部的孩子，亲的

childNodes：它是标准属性，它返回指定元素的子元素集合，包括HTML节点，所有属性，文本节点 （嫡出）

**火狐 谷歌等高本版会把换行也看做是子节点**



利用 nodeType == 1 时才是元素节点 来获取元素节点

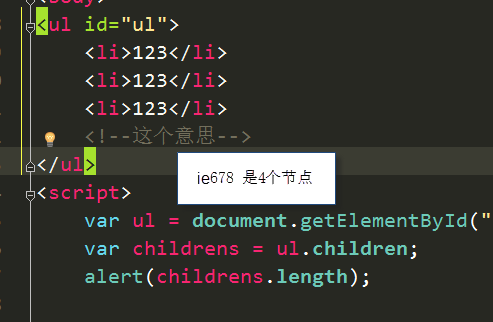
标签 ===元素

**var** ul=document.getElementById("ul");  
**var** li=ul.childNodes;  
console.log(li.length);  
**for**(**var** i=0;i<li.length;i++){  
 **if**(li[i].nodeType==1){  
 li[i].style.backgroundColor="#e3e3e3";  
 }  
}

**children 重要 选取所有的孩子 （只有元素节点）**

这个更好 跟喜欢它 。 （庶出）

ie 678 包含 注释节点 这个要避免开。删掉注释就行了



dom 的节点操作

document.createElement（）;

parentNode.qppendChild( childnode);

parentNode.insertBefore( childnode,beforewho);

parentNode.removeChild( childnode);

demo.cloneNode(ture||false)

创建节点

var div = document.creatElement(“li”);

上面的意思就是 生成一个新的 li 标签

插入节点

1. appendChild(); 添加孩子 append 添加的意思

意思： 添加孩子 放到盒子的 最后面。

Dome。Appendchild（div）；

2. insertBefore(插入的节点，参照节点) 子节点 添加孩子

写满两个参数

demo.insertBefore(test,childrens[0]);

放到了第一个孩子的前面

如果第二个参数 为 null 则 默认这新生成的盒子放到最后面。

demo.insertBefore(test,null);

移除孩子节点

removeChild() 孩子节点

var da = **document**.getElementById("xiongda");  
demo.removeChild(da);

克隆节点

cloneNode();

复制节点

**括号里面可以跟参数** ， 如果 **里面是 true**  深层复制， 除了复制本盒子，还复制子节点

如果为 false 浅层复制 只复制 本节点 不复制 子节点。

Demo.parentnode.appendchild(demo.cloneNode(demo));

设置节点属性

1. 获取节点属性

**getAttribute(属性) 获取属性**

通过这个方法，可以得到 某些元素的 某些属性 。

alert(demo.getAttribute("title"));

弹出对话框： 弹出title里面的内容

2. 设置节点属性

**setAttribute(“属性”,”值”);**

d**iv.setAttribute(“class”,”demo”);**

3. 删除某个属性

removeAttribute(“属性”);

**demo.removeAttribute(“title”)**

这个盒子就没有title 属性 给删掉了 。

A.appendChild(B);

B 一定是 A 孩子 同时 b 放到了a 的里面 装到里面去了 最后面。 b 放到 a 里面

A.insertBefore（B，C）

B C 都是 A 的孩子

把 b 放到 a 里面 ，但是 是 c 的前面

内置对象

内置对象： 内置对象就是指这个语言自带的一些对象，供开发者使用，这些对象提供了一些常用的或是最基本而必要的功能。

手机买来就能发短信 就能打电话

日期函数 ( Date() )

这个函数 （对象） 可以设置我们本地 日期。 年月日 时分秒

声明日期

var date = new Date();

创造声明一个新的日期函数 赋值给了 date

var arr = new Array();

使用函数

得到 毫秒数 ms s m h

从 1970 年1 月1号 unix 32位 68年

2038 千年虫 64 位

var date = new Date();

date.getTime();

date.valueOf(); 得到 距离 1970年的毫秒数，更新到

var date = new Date(); // 声明  
**console**.log(date.getTime()); // 提倡使用的  
**console**.log(date.valueOf());  
  
// 直接使用  
**console**.log(Date.now());  
**console**.log(+new Date());

常用的日期的方法

获取日期和时间

getDate() 获取日 1-31

getDay () 获取星期 0-6

getMonth () 获取月 0-11

getFullYear () 获取完整年份（浏览器都支持）

getHours () 获取小时 0-23

getMinutes () 获取分钟 0-59

getSeconds () 获取秒 0-59

getMilliseconds () 获取当前的毫秒

getTime () 返回累计毫秒数(从1970/1/1午夜)

如果是上午 ，我打开页面 页面中显示的是

上午好，好好学习 显示的是上午的图片

如果是下午 我打开页面 页面中显示的是

下午好，天天向上 显示的是下午的图片

根据当前的小时来判断 if

定时器



很多情况下，一些操作不需要人工干预， 代码自己会不断的去执行 。

而且 会有 时间的绑定 。 比如每隔 5秒钟就去执行一次事件。

我们可以设定时间 让某个动作不断的去执行 。 这个我们再js 里面用定时器来表示。

window.setInterval(“执行的函数”,间隔时间)

正确的写法:

setInterval(fun, 1000); 1000 ms 毫秒

每隔1秒钟，就去执行一次 fun 这个函数.

setInterval(“fun()”,1000) 可以用

setInterval( function(){} , 1000 )

~~setInterval(fun(),1000) 错误的~~

定时器 特别的像 for 循环，但是我 的定时器最大的特点在于， 自动，可以设定时间。

倒计时

好比，今年你多大了 ？

2015 - 1990 25

我们要计算的 倒计时

有一个最终时间 12月12日

有一个现在时间 11月 13日

倒计时 = 用 将来的时间 - 现在的时间

问题： 用 毫秒减去 现在距离 1970年1

将来时间 距离 1970 毫秒数

用将来的毫秒数 - 现在的毫秒数 不断转换就可以了

<!DOCTYPE html>  
<html>  
<head lang="en">  
 <meta charset="UTF-8">  
 <title></title>  
 <style>  
 body{  
 font-size:30px;  
 text-align: center;  
 color:red;  
 }  
 </style>  
 <script>  
 **window**.onload = function(){  
 var demo = **document**.getElementById("demo");  
 var endTime = new Date("2015/12/12 17:30:00"); // 最终时间  
 setInterval(clock,1000); // 开启定时器  
 function clock(){  
 var nowTime = new Date(); // 一定是要获取最新的时间  
 // console.log(nowTime.getTime()); 获得自己的毫秒  
 var second = parseInt((endTime.getTime() - nowTime.getTime()) / 1000);  
 // 用将来的时间毫秒 - 现在的毫秒 / 1000 得到的 还剩下的秒 可能处不断 取整  
 // console.log(second);  
 // 一小时 3600 秒  
 // second / 3600 一共的小时数 /24 天数  
 var d = parseInt(second / 3600 / 24); //天数  
 //console.log(d);  
 var h = parseInt(second / 3600 % 24) // 小时  
 // console.log(h);  
 var m = parseInt(second / 60 % 60);  
 //console.log(m);  
 var s = parseInt(second % 60); // 当前的秒  
 **console**.log(s);  
 /\* if(d<10)  
 {  
 d = "0" + d;  
 }\*/  
 d<10 ? d="0"+d : d;  
 h<10 ? h="0"+h : h;  
 m<10 ? m="0"+m : m;  
 s<10 ? s="0"+s : s;  
 demo.**innerHTML** = "距离抢购时间还剩: "+d+"天 "+h+"小时 "+m+"分钟 "+s+"秒";  
 }  
 }  
 </script>  
</head>  
<body>  
<div id="demo"></div>  
</body>

定义自己的日子

var endTime = new Date(“2015/12/12”);

如果date 括号里面写日期 就是 自己定义的时间

如果 date括号里面不写日期 ， 就是当前时间 。

new Date(“2015/12/12 17:30:00”);

日期和时分秒中间 有空格隔开

时钟案例



1. minute.style.MozTransform = "rotate("+ m\*6 +"deg)";
2. hour.style.MozTransform = "rotate("+ h\*30 +"deg)";

分两步进行的。

第一步： 要得到现在的 时 分 秒

但是这里面有一个小玄机 。

比如现在是 9点整 时针指向 9 是没错的

但是如果现在是 9点半 时针应该指向的是 9到10 之间 才对

所以，我们不但要得到现在的小时 ，还要得到 已经过去了多少分

ms = date.getMilliseconds(); // 现在的毫秒数  
s = date.getSeconds() + ms / 1000; // 得到秒 1.3 s  
m = date.getMinutes() + s / 60; // 得到的是分数 45.6分钟  
h = date.getHours() % 12 + m / 60 ;

旋转角度原理

秒针 一秒 走多少度呢 ？

一圈 360 ° 一共有 60 秒 每秒 6 °

分针 一圈 360 一圈走 60次 每次 6° 每分钟 6°

时针 一圈 360 一共 12 个 表盘没有24小时 每个小时 走 30°

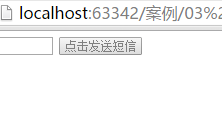
完整代码：

1. <script>
2. var hour = document.getElementById("hour");
3. var minute = document.getElementById("minute");
4. var second = document.getElementById("second");
5. // 开始定时器
6. var s = 0,m = 0, h = 0, ms = 0;
7. setInterval(function() {
8. // 内容就可以了
9. var date = new Date(); // 得到最新的时间
10. ms = date.getMilliseconds(); // 现在的毫秒数
11. s = date.getSeconds() + ms / 1000; // 得到秒 1.3 s
12. m = date.getMinutes() + s / 60; // 得到的是分数 45.6分钟
13. h = date.getHours() % 12 + m / 60 ;
14. // console.log(h);
15. // 旋转角度
16. // 一圈 360 ° 一共有 60 秒 每秒 6 ° 现在是 s秒
17. second.style.WebkitTransform = "rotate("+ s\*6 +"deg)";
18. // 变化 旋转 deg 度
19. minute.style.WebkitTransform = "rotate("+ m\*6 +"deg)";
20. hour.style.WebkitTransform = "rotate("+ h\*30 +"deg)";
21. second.style.MozTransform = "rotate("+ s\*6 +"deg)";
22. // 变化 旋转 deg 度
23. minute.style.MozTransform = "rotate("+ m\*6 +"deg)";
24. hour.style.MozTransform = "rotate("+ h\*30 +"deg)";
25. },30);
26. </script>

按钮不可用

button 不可以用 disabled 不可用的意思

btn.disabled = “disabled” || btn.disabled = true; 选任意一个都行

 灰色的

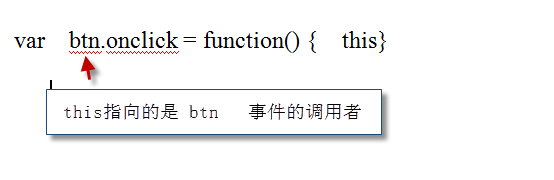
**注意：**

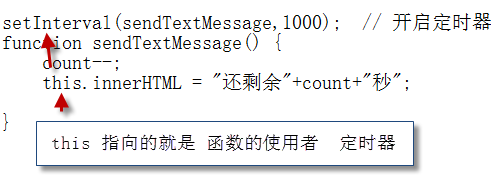
1. 因为 button是个双标签 所以要更改他的值， 使用 innerHTML 的，不是value。
2. 关闭定时器 clearInterval(定时器名称); 定时器不再进行

this

this 指向的是 事件的调用者 ，或者是函数的使用者。

var btn.onclick = function() { this}





**好多时候不能使用but，一般情况下，我们喜欢 var that = this;就可以吧上面的传递下去**

var that = this; // 把 btn 对象 给 that var \_this = this;

获取验证码

<script>  
 var sendMessage=**document**.getElementById("sendMessage");  
 var input=sendMessage.**children**[0];  
 var btn=sendMessage.**children**[1];  
 btn.onclick=function() {  
 this.**disabled**=true; //标签不可用，disabled=true  
 var count=5;  
 var this\_btn=this; //为了使用本对象，给他加个句柄，以后弄成函数好弄  
 var timer=**window**.setInterval(settimer,1000); //设置定时循环器，注意只写函数名就行了，并定义定时器  
 function settimer(){ //执行的函数，必须写在事件函数内部  
 count--;  
 var innerhtml="";  
 if(count>0){  
 innerhtml=count+"秒后重新发送";  
 this\_btn.**innerHTML**=innerhtml;  
 }else{  
 innerhtml="获取验证码";  
 this\_btn.**innerHTML**=innerhtml;  
 this\_btn.**disabled**=false; //设置为可用  
 count=5; //重置计数  
 clearInterval(timer); //关闭计数器  
  
 }  
 }  
 }  
  
</script>

setTimeout()

时间去哪儿了 类似于定时炸弹 。。

setTimeout(“函数”, 时间 )

setInterval(fn,5000); 每隔 5秒钟，就去执行函数fn一次

setTimeout(fn,5000); 5秒钟之后，去执行 fn 函数， 只执行一次，可以递归调用完成循环

**自动关闭广告，自动跳转什么得**

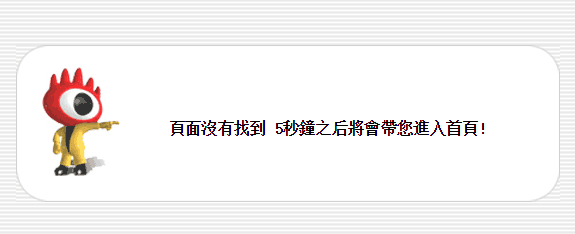
深层次的看待定时器区别

**setInterval是排队执行的**

假如 间隔时间是1秒， 而执行的程序的时间是2秒 上次还没执行完的代码会排队, 上一次执行完下一次的就立即执行, 这样实际执行的间隔时间为2秒

setTimeout延迟时间为1秒执行, 要执行的代码需要2秒来执行,那这段代码上一次与下一次的执行时间为3秒.

5秒钟自动跳转页面



JS **页面跳转: window.location.href = ”http://www.itcast.cn” ;**  BOM

**函数自己调用自己的过程** 我们称之为 ： **递归调用** 自残



但是这样用，一定要加一个退出 if 的条件，不然成为死循环了。

目的就是为了，**模拟使用 settimeout 来实现setinterval 的效果**。**不存在队列问题了就**

<script>  
 var demo = **document**.getElementById("demo");  
 var count = 5;  
 var speed = 1000;  
 setTimeout(goIndexPage,speed); // 1秒钟之后去执行 goIndexPage这个函数  
 function goIndexPage() {  
 count--;  
 demo.**innerHTML** = "<a href='http://www.baidu.com'>本页面将在第"+count+"秒钟之后跳转页面</a>";  
 if(count <= 0)  
 {  
 // 如果 count 小于 0 就到了时间了 我们应该跳转页面  
 **window**.**location**.**href** = "http://www.baidu.com";  
 }  
 else  
 {  
 setTimeout(goIndexPage,speed); // 递归调用 自己调用自己  
 }  
 }

辞海 10万字 2500 汉字 1000次常用汉字

arguments 对象

function fn(a,b,c) { console.log(a+b+c); alert(arguments.length;)}

fn(1,3,4,6);

**只能用在函数内部的**

**arguments.length; 返回的是 实参的个数。**

但是这个对象有讲究，他只在正在使用的函数内使用。

arguments.callee;

返回的是正在执行的函数。 也是在函数体内使用。 在使用函数递归调用时推荐使用**arguments.callee代替函数名本身。**

用来在递归调用中使用

function fn() { console.log(arguments.callee); }

这个callee 就是 ： function fn() { console.log(arguments.callee); }

小米图片移动

<script>

function $(id) { return document.getElementById(id);}

var num = 0; // 控制图片的top值

var timer = null; // 定时器名称

$("picUp").onmouseover = function(){

// alert(11);

clearInterval(timer);

timer = setInterval(function(){

num -= 3;

num >= -1070 ? $("pic").style.top = num + "px" : clearInterval(timer);

},30);

}

$("picDown").onmouseover = function(){

clearInterval(timer);

timer = setInterval(function(){

num++;

num < 0 ? $("pic").style.top = num + "px" : clearInterval(timer);

},30);

}

$("picUp").parentNode.onmouseout = function() {

clearInterval(timer);

}

</script>

定时器的滞后性

Settimeout 阻塞型的语言，优先执行先执行的，最后才执行这里的代码

运算符

一元操作符 ++， -- + - +5 -6

逻辑操作符 ! && ||

基本运算符 +, -, \*, /, %

关系操作符 >, <, >=, <=, ===, ==, !=, !==

= 赋值 == 判断 === 全等

条件操作符 （三元运算符） ? :

赋值运算符 +=, -=, \*=, /=, %=

a+=5 a= a + 5

逗号运算符 , var a=0,b=0;

运算符顺序

1 ()

2 !、-、++、-- (-10) 负号 正号

3 \*、/、%

4 +、- 10-5

5 <、<=、<、>=

6 ==、!=、===、!==、

7 &&

8 ||

9?:

10 =、+=、-=、\*=、/=、%= 赋值

1+2\*3

几个面试题

1. a&&b 结果是什么？ 短位与

如果a 为假 ，则返回 a

如果a 为真 ，则返回 b

var aa = 0&&1;

alert(aa) // 0

var bb = 1&&0;

alert(bb); //0

var cc = 1&&10;

alert(cc); // 10

1. a||b

如果 a 为假 则返回b

如果 a 为真 则返回a

**console**.log(0||1); 1  
**console**.log(1||0); 1  
**console**.log(1||5); 1  
**console**.log(5||1); 5

var a = 1 && 2 && 3;

console.log(a); 3

var b = 0 && 1 && 2;

console.log(b); 0

var c = 1 && 0 && 2;

console.log(c); 0

%=

a+=3

a = a % 3;

字符串对象常用方法

我们工作中经常进行字符串操作。

转换为字符串

1. + “” 2+ “” = “2” 2+”ab” = “2ab”

2. String() 转换为字符串 String（3）；

new string（） 的话就是一个对象了 object

3. toString（基数） ; 基数就是进制

var txt = 10;

txt.toString(2) 转换进制，还是数字型？？？

获取字符位置方法

charAt，获取相应位置字符（参数： 字符位置） 0开始的

charCodeAt获取相应位置字符unicode编码（参数： 字符位置）

var txt = “abcedf”;

比如，  **txt.charAt(4);**  索引号一定**是从0开始** 返回的结果是 d，汉字相同

我们根据我们输入的 位数 返回相应的 字符 。

unicode编码是我们字符的字符的唯一表示 。

字符串长度

Length 是数量 不是长度，汉字长度是2需要重新弄

是字符的话charcodeat 返回值是0<x<127;

<script>  
 function strLength(str){  
 var strlen=0;//记录字符串的实际长度  
 var len=str.**length**;  
 for(var i=0;i<len;i++){  
 var c=str.charCodeAt(i);  
 if(0<c&&c<127){  
 strlen++;  
 }else{  
 strlen+=2;  
 }  
 }  
 return strlen;  
 }  
 **console**.log(strLength("we are 中国人"));  
</script>

根据字符返回位置

跟 charAt() 相反的 根据 字符 返回的是 位置

返回前面起第一个字符的位置

indexOf(“字符”);

它是从 前面开始数（从左边开始数）， 而且只找第一个， 然后返回改字符的位置， **索引号都是从0开始的**。 返回的是个数值。

var txt = “abcdef”;

alert(txt.indexOf(“d”)) 结果就是 3

**如果找不到该字符 返回 -1**

返回后面起第一个字符的位置

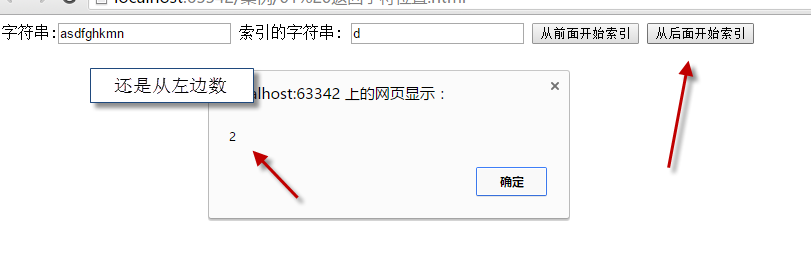
**lastIndexOf(参数：索引字符串)**

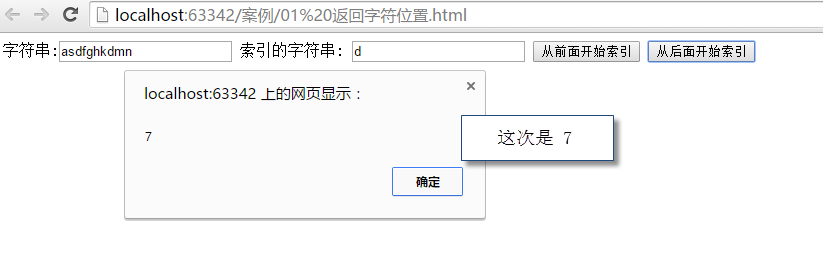
**最后一次出现该字符的索引号**

var txt = “abcdef” ;

txt.lastIndexOf(“d”); 3

**返回的值，还是从 左边开始 数的 索引号 。**





网址编码

我们知道一个网址 自己的网址， 不同页面也有自己id网址， 我们经常会做一些， **把网址送入到后台**。 但是后台再处理的 不认识比如 换行啊 等特殊符号的 ？

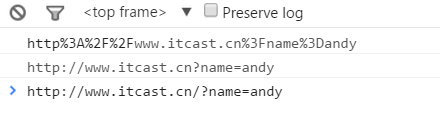
var url = “http://www.itast.cn?name=cz”

**所以我们要实现编码，然后再传到后台。**

**encodeURIComponent()** 函数可把字符串作为 URI 组件进行编码

**decodeURIComponent()** 函数可把字符串作为 URI 组件进行解码

var url = "http://www.itcast.cn?name=andy";  
**console**.log(encodeURIComponent(url)); // 编码  
var afterUrl = encodeURIComponent(url);  
**console**.log(decodeURIComponent(afterUrl)); // 解码

结果：

操作字符串

合并字符串 取字符串

concat() 连接字符串

var txt1 = “abc”;

var txt2 = ”123”;

console.log(txt1.concat(txt2)); “abc123”;

slice()

slice(“取字符串的起始位置”, [结束位置]) ; [] 可选的

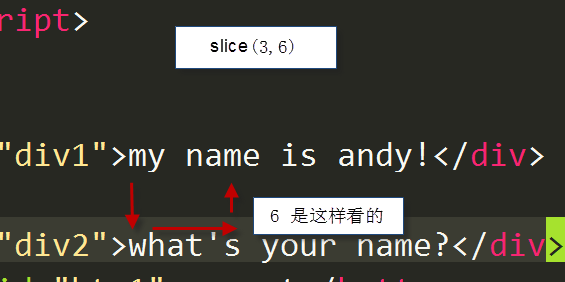
**起始位置一定要有 结束位置可以省略**

var txt = “abcedf”;

txt.slice(3) ; 从 txt 里面字符的 第 3（索引号）个开始取 结**束位置省略， 一直取到最后一个**。

总结 slice(3) 从第3个开始取，一直取到最后

**slice(3,6) 3 从 第3个开始 取 6 取到第6索引号的位置，还是从左边的第0个开始数。 但是不包 6 。**



**起始位置可以是负数 ， 如果是负数，则是从 右边往左边开始取。**

var txt =”asdf”;

**txt.slice(-1) 结果是 f**

**txt.slice(-2) 结果是 df**

**txt.slice(-3) 结果是 sdf**

substr()

**substr(起始位置,[取的个数])**

同上。

不写取的个数， 默认从起始位置一直取到最后 。

取的个数： 是指从起始位置开始，往后面数几个。

var txt = “abcdefghijk”;

txt.substr(3,4);

从第3个 （d） 开始 数 4个 defg

**substr(-1) 少用 ie678 报错 。 尽量少用 ，不认识的话就会从头到尾**

兼容性的写法 ：

onBtnClick("btn7",div1.substr(div1.**length**-1,1)); // 兼容的写法

substring 同slice 一样的 但是有一点不同

substring(3,6)

**substring 始终会把 小的值作为 起始位置 大的值作为结束位置**

**例如： substring(6,3) 实际中 自动变成 substring(3,6);**

保留小数位数

122340.12345 保留两位有效小数 122340.12

首先要转换成字符串

**substr(0, .+3)； 不能用的，注意indexof是从0开始的 用下面的**

1.**console**.log(str.substr(0,str.indexOf(".")+3));

通过 indexOf 返回小数点的位置 截取字符串

2 **console**.log(parseInt(PI\*100) /100);

先乘以100 取整 然后 除以100

3 **console**.log(PI.toFixed(2));

pi.toFixed(2) 保留 2位 小数 必须是数字

大小写转换

asdf = ASDF

ASDF -- asdf

1 toUpperCase，转换为大写（参数： 无）

2 toLowerCase，转换为小写（参数：无）

asdf.toUpperCase() 结果就是： ASDF

$("txt").**value**.toUpperCase();

把txt 的值 转换为大写

图片底部的3px的间距

img{  
 vertical-align: top;  
}

或者

Display：block

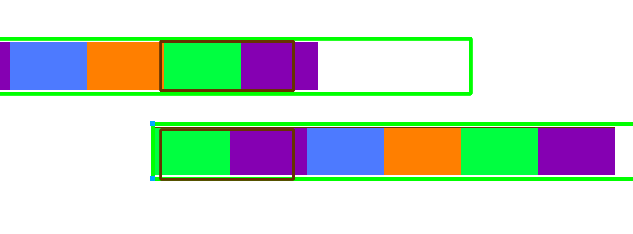
验证文件是否正确

var lastname= str.substr(str.lastIndexOf(".")).toUpperCase();  
 if(lastname=="JPG"){}

无缝滚动

原理：

首先先复制 两张图片 （第一张和第二张） 放到最后面 （本质上是第 5.6张）



ul 是盒子移动的， 如果ul 的left 值 大于等于 4张图片的宽度，就应该快速复原为0 。

完整代码：

1. <script>
2. var scroll = document.getElementById("scroll");
3. var ul = scroll.children[0];
4. var num = 0; //控制左侧值 left
5. var timer = null; // 定时器
6. timer = setInterval(autoPlay,20);
7. function autoPlay() {
8. num--;
9. //console.log(num);
10. num<=-1200 ? num = 0 : num;
11. ul.style.left = num + "px";
12. }
13. scroll.onmouseover = function() { // 鼠标经过大盒子 停止定时器
14. clearInterval(timer);
15. }
16. scroll.onmouseout = function() {
17. timer = setInterval(autoPlay,20); // 开启定时器
18. }
19. </script>

缓动动画

匀速动画，看起来，效果太简单。

等差序列 : 2 4 6 8 10

缓动动画公式:

一个盒子初始值 是 0 要走到 400 px 的位置

假如说，初始值 leader 0 target 400

box.style.left = xxxx + “px”

leader = leader + (target - leader ) /10 ;

**步长不断减小的过程**

1. btn.onclick = function() {
2. setInterval(function(){
3. leader = leader + (target - leader )/10;
4. box.style.left = leader + "px";
5. },30)
6. }

自制banner

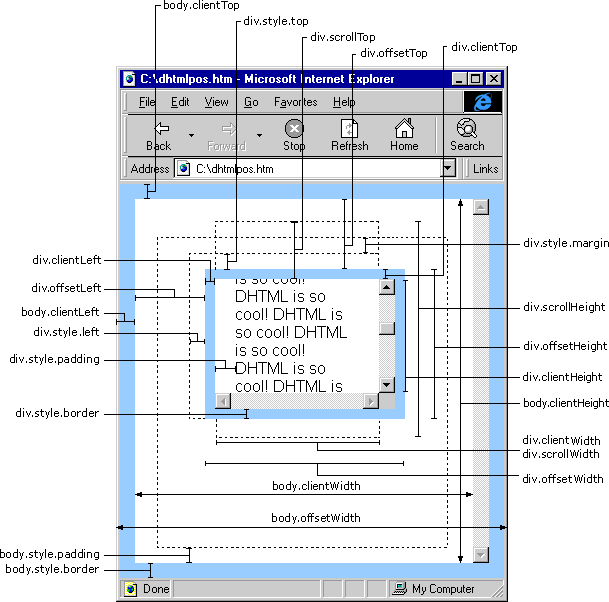
/\*\*  
 \* Created by hahaha on 2016/9/4.  
 \*/  
**window**.onload=function(){  
 function $(id){  
 return **document**.getElementById(id);  
 }  
 var ul\_img=$("ul\_img");  
 var li\_img=ul\_img.**children**;  
 var ul\_point=$("ul\_point");  
 var li\_point=ul\_point.**children**;  
 var banner=$("banner");  
 var leader= 0,target=0;  
 var arr=$("arr");  
 var left=$("left");  
 var right=$("right");  
 //实现点击改变目标地址  
 for(var i=0;i<li\_img.**length**;i++){  
 li\_point[i].**index**=i;  
 li\_point[i].**onmouseover**=function(){  
 for(var j=0;j<li\_point.**length**;j++){  
 li\_point[j].**className**="";  
 }  
 li\_point[this.index].**className**="choose";  
 target=-520\*(this.index);  
 }  
  
 }  
 banner.onmouseover=function() {  
 arr.**style**.**display**="block";  
  
 }  
 banner.onmouseout= function () {  
 arr.**style**.**display**="none";  
 }  
  
  
 //箭头的时候运动的值  
 left.onclick=function(){  
 target+=-520;  
 if(target<-2080){  
 target=-2080;  
 }  
 }  
 right.onclick=function() {  
 target += 520;  
 if (target > 0) {  
 target = 0;  
 }  
 }  
 //点的随动  
 function lrchoose(target){  
 var i=-target/520;  
 choose(i);  
  
 }  
 function choose(i) {  
 for (var j = 0; j < li\_point.**length**; j++) {  
 li\_point[j].**className** = "";  
 }  
 li\_point[i].**className** = "choose";  
 }  
 //自动动  
 setInterval(function(){  
 target+=-520;  
 if(target<-2080){  
 target=0;  
 }  
 },10000);  
 //lrchoose(target); //点随动调用，不知道为什么放在这不能用？？？？  
 setInterval(function(){  
 leader=leader+(target-leader)/10;  
 ul\_img.**style**.**left**=leader+"px";  
 lrchoose(target); //点随动调用  
 },30);  
}

Offset

offset家族

offset 自己的

目的： js中有一套方便的**获取元素尺寸**的办法就是offset家族；



offsetWidth offsetHeight 动态获取宽高

**得到对象的宽度和高度**(自己的，与他人无关)

**offsetWidth = width + border + padding**

div { width:220px; border-left:2px solid red; padding:10px;}

**div.offsetWidth = 220 + 2 + 20**

**div.style.width 因为东西 只能得到行内的数值,就像下面的**

**< div width=”80px>**

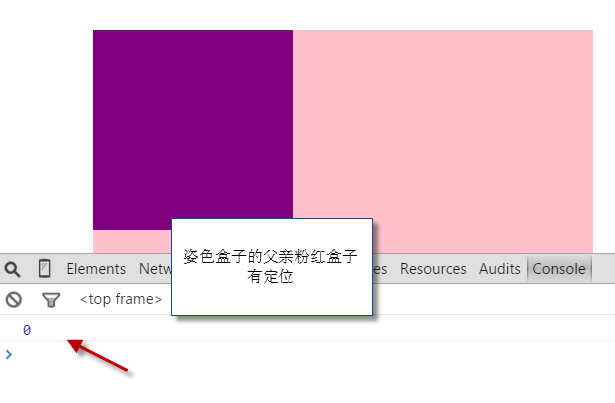
**div.style.heigth 永远的不到**

offsetLeft offsetTop

返回距离上级盒子（**最近的带有定位**）左边的位置

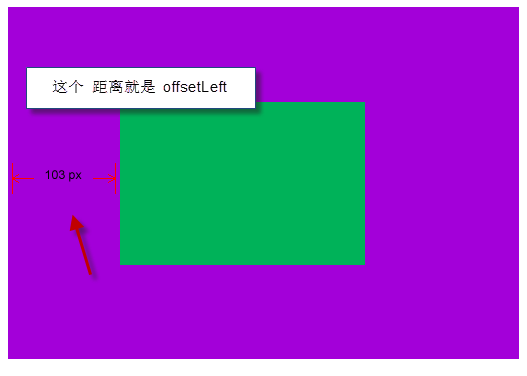
如果父级都没有定位则以body 为准

这里的父级指的是所有上一级 不仅仅指的是 父亲 还可以是 爷爷 曾爷爷 曾曾爷爷。。。。



offsetLeft **从父级的padding 开始算**  父亲的border 不算

总结一下： **就是子盒子到定位的父盒子边框到边框的距离**



offsetParent

**返回改对象的父级 （带有定位） 不一定是亲的爸爸**

前面学过一个返回父亲(亲的) parentNode 有所区别

如果当前元素的父级元素没有进行CSS定位（position为absolute或relative），offsetParent为body。

2、如果当前元素的父级元素中有CSS定位（position为absolute或relative），offsetParent取最近的那个父级元素。

var son = **document**.getElementById("son");  
alert(son.parentNode.id);  
alert(son.**offsetParent**.**tagName**); // tagName标签的名字

返回父标签的id或name

alert(son.parentNode.id);

alert(son.**offsetParent**.**tagName**);

offsetTop style.top 的区别

一、最大区别在于 **offsetLeft 可以返回没有定位盒子**的距离左侧的位置。 而 **style.top** **只有定位的盒子 才有 left top right**

二、**offsetTop 返回的是数字，而 style.top 返回的是字符串，除了数字外还带有单位：px。**

style.left = 300px parseInt（300px） 结果 300

parseInt（style.left） + parseInt（style.left）

三、**offsetTop 只读，而 style.top 可读写。**

1. **如果没有给 HTML 元素指定过 top 样式，则 style.top 返回的是空字符串。**
2. 最重要的区别  **style.left 只能得到 行内样式 offsetLeft 随便**

事件对象

我们学过一些事件 : onmouseover onmouseout onclick .....

**btn.onclick = function(event) { 语句 }**

event 就是事件的对象 指向的是 事件 是 onclick

**再触发DOM上的某个事件时，会产生一个事件对象event，这个对象中包含着所有与事件有关的信息。所有浏览器都支持event对象，但支持的方式不同。**

**Event不是必须的命名**

比如鼠标操作时候，会添加鼠标位置的相关信息到事件对象中。

**普通浏览器支持 event**

**ie 678 支持 window.event**

所以我们 采取兼容性的写法 ： 在函数内部的写这个的，虽然没出现window.Event

var event = event || **window**.**event**;

**Document.onclick=function(event){**

**Var event=evemt||window.event;**

**}**

**也可以写成**

**Document.onclick=function(aa){**

**Var event=aa||window.event;**

**}**

在任何地方点击触发的事件

文档中点击

**Document.onclick=function(event){**

**Var event=evemt||window.event;**

**}**

Document是整个页面的大小； body可以有大小的

event 常见属性

**clientX 以当前窗口，可视区，位置 类似于 fixed**

**pageX 以文档边界 x 位置 ie678不认识 absolute**

**screenX 相对屏幕的x位置**

|  |  |
| --- | --- |
| **属性** | **作用** |
| data | 返回拖拽对象的URL字符串（dragDrop） |
| width | 该窗口或框架的高度 |
| height | 该窗口或框架的高度 |
| pageX | 光标相对于该网页的水平位置（ie无） |
| pageY | 光标相对于该网页的垂直位置（ie无） |
| screenX | 光标相对于该屏幕的水平位置 |
| screenY | 光标相对于该屏幕的垂直位置 |
| target | 该事件被传送到的对象 |
| type | 事件的类型 |
| clientX | 光标相对于该网页的水平位置 （当前可见区域） |
| clientY | 光标相对于该网页的水平位置 |

获取在盒子中的位置

event.cleintX-$(“id’).offsetlef

常用事件

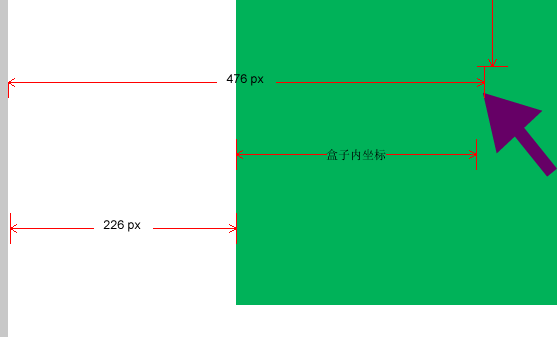
onmouseover onmouseout onclick

**onmousemove**  当鼠标移动的时候 就是说，**鼠标移动一像素就会执行的事件**

**div.onmousemove = function() { 语句 }**

**当鼠标再div 身上移动的时候，就会执行**。

得到在某个盒子内的坐标：



div.onmouseover 和 div.onmousemove 区别

他们相同点 都是 经过 div 才会触发

div.onmouseover 只触发一次

div.onmousemove 每移动一像素，就会触发一次

淘宝商品放大镜

<div class="box" id="fdj">  
 <!--小盒子-->  
 <div class="small">  
 <img src="images/001.jpg" alt=""/>  
 <div class="mask"></div>  
 </div>  
 <div class="big">  
 <img src="images/0001.jpg" alt=""/>  
 </div>  
</div>

<style>  
 img {  
 vertical-align: top;  
 }  
 .box {  
 width: 350px;  
 height: 350px;   
 position: relative;  
 }  
 .big {  
 width: 450px;  
 height: 450px;  
 position: absolute;  
 top: 0;  
 left: 360px;  
 overflow: hidden;  
 display: none;  
 }  
 .mask {  
 width: 100px;  
 height: 100px;  
 background: rgba(255, 255, 0, 0.4);  
 position: absolute;  
 top: 0;  
 left: 0;  
 cursor: move;  
 display: none;  
 }  
 .small {  
 position: relative;  
 }  
 .big img {  
 position: absolute;  
 top: 0;  
 left: 0;  
 }  
</style>

Box>big,small 方便大盒子定位，可以显示在图片的旁边

Small》img，mask 鼠标移动事件由small触发，因为big属于box，鼠标在big中图不会消失，mask也放在盒子中是因为如果放在外面的话，独立的个体，鼠标在小盒子上的时候，属于mouseout ，mask消失，但由于有接触了small，所以又显示了，会不停的闪，定义在里面的话就没影响了

<script>  
 var fdj = **document**.getElementById("fdj"); // 获得最大的盒子  
 var small = fdj.**children**[0]; // 获得small 小图片 350盒子  
 var big = fdj.**children**[1]; // 获得 大图片 800 盒子  
 var mask = small.**children**[1]; // 小的黄色盒子  
 var bigImage = big.**children**[0]; // 大盒子里面的图片  
 var smallImage=small.**children**[0];//小盒子里面图片  
 small.onmouseover = function() { // 鼠标经过显示出他们  
 mask.**style**.**display** = "block";  
 big.**style**.**display** = "block";  
 }  
 small.onmouseout = function() {  
 mask.**style**.**display** = "none";  
 big.**style**.**display** = "none";  
 }  
 // 鼠标在small 内移动  
 var x = 0;  
 var y = 0;  
 small.onmousemove = function(event) {  
 var event = event || **window**.**event**;  
 x = event.**clientX** - this.offsetParent.**offsetLeft** - mask.**offsetWidth** /2; // 再某个盒子内的坐标  
 //alert(this.offsetLeft);  
 y = event.**clientY** - this.offsetParent.**offsetTop** - mask.**offsetHeight** /2;  
 if(x < 0)  
 {  
 x = 0;  
 }  
 else if(x > small.**offsetWidth** - mask.**offsetWidth**)  
 {  
 x = small.**offsetWidth** - mask.**offsetWidth**;  
 }  
 if(y<0)  
 {  
 y = 0;  
 }  
 else if(y > small.**offsetHeight** - mask.**offsetHeight**)  
 {  
 y = small.**offsetHeight** - mask.**offsetHeight**;  
 }  
 mask.**style**.**left** = x + "px";  
 mask.**style**.**top** = y + "px";  
 /\*计算 ： 夫子 一顿吃 2个馒头 娇子 一顿 4个馒头  
 问 夫子今天吃了 3个馒头 娇子应该吃几个？ \*/  
 /\*计算出他们的倍数 4 / 2 2倍  
 3 \* 2 == 6个 \*/  
 /\* 大图盒子 / 小图盒子 倍数  
 我们 再小图移动的距离 \* 倍数 == 大图的位置\*/  
// bigImage.style.left = -x \* big.offsetWidth /small.offsetWidth + "px";  
// bigImage.style.top = -y \* big.offsetHeight /small.offsetHeight + "px";  
 //上面会导致显示不全的情况出现，因此正确的应该是  
  
 bigImage.**style**.**top** = -y \* bigImage.**offsetHeight** /smallImage.**offsetHeight** + "px";  
 bigImage.**style**.**left** = -x \* bigImage.**offsetWidth** /smallImage.**offsetWidth** + "px";  
 //上面会导致显示不全的情况出现，因此正确的应该是  
  
  
 }

拖动

onmouseup 当鼠标弹起

onmousedown 当鼠标按下的时候

1.拖动 原理 == 鼠标按下 接着 移动鼠标 。

bar.onmousedown = function(){

document.onmousemove = function(){

}

}

1. 当我们按下鼠标的时候，就要记录当前 鼠标的位置 - 大盒子的位置   
    算出 bar 当前 在 大盒子内的距离 。
2. <!DOCTYPE html>  
   <html>  
   <head lang="en">  
    <meta charset="UTF-8">  
    <title></title>  
    <style>  
    \* {margin: 0;padding: 0;}  
    img {  
    vertical-align: top;  
    }  
    .box {  
    width: 350px;  
    height: 350px;  
    margin:100px;  
    border: 1px solid #ccc;  
    position: relative;  
    }  
    .big {  
    width: 450px;  
    height: 450px;  
    position: absolute;  
    top: 0;  
    left: 360px;  
    border: 1px solid #ccc;  
    overflow: hidden;  
    display: none;  
    }  
    .mask {  
    width: 100px;  
    height: 100px;  
    background: rgba(255, 255, 0, 0.4);  
    position: absolute;  
    top: 0;  
    left: 0;  
    cursor: move;  
    display: none;  
    }  
    .small {  
    position: relative;  
    }  
    .big img {  
    position: absolute;  
    top: 0;  
    left: 0;  
    }  
    </style>  
   </head>  
   <body>  
   <div class="box" id="fdj">  
    <!--小盒子-->  
    <div class="small">  
    <img src="images/001.jpg" alt=""/>  
    <div class="mask"></div>  
    </div>  
    <div class="big">  
    <img src="images/0001.jpg" alt=""/>  
    </div>  
   </div>  
   </body>  
   </html>  
   <script>  
    var fdj = **document**.getElementById("fdj"); // 获得最大的盒子  
    var small = fdj.**children**[0]; // 获得small 小图片 350盒子  
    var big = fdj.**children**[1]; // 获得 大图片 800 盒子  
    var mask = small.**children**[1]; // 小的黄色盒子  
    var bigImage = big.**children**[0]; // 大盒子里面的图片  
    var smallImage=small.**children**[0];//小盒子里面图片  
    small.onmouseover = function() { // 鼠标经过显示出他们  
    mask.**style**.**display** = "block";  
    big.**style**.**display** = "block";  
    }  
    small.onmouseout = function() {  
    mask.**style**.**display** = "none";  
    big.**style**.**display** = "none";  
    }  
    // 鼠标在small 内移动  
    var x = 0;  
    var y = 0;  
    small.onmousemove = function(event) {  
    var event = event || **window**.**event**;  
    x = event.**clientX** - this.offsetParent.**offsetLeft** - mask.**offsetWidth** /2; // 再某个盒子内的坐标  
    //alert(this.offsetLeft);  
    y = event.**clientY** - this.offsetParent.**offsetTop** - mask.**offsetHeight** /2;  
    if(x < 0)  
    {  
    x = 0;  
    }  
    else if(x > small.**offsetWidth** - mask.**offsetWidth**)  
    {  
    x = small.**offsetWidth** - mask.**offsetWidth**;  
    }  
    if(y<0)  
    {  
    y = 0;  
    }  
    else if(y > small.**offsetHeight** - mask.**offsetHeight**)  
    {  
    y = small.**offsetHeight** - mask.**offsetHeight**;  
    }  
    mask.**style**.**left** = x + "px";  
    mask.**style**.**top** = y + "px";  
    /\*计算 ： 夫子 一顿吃 2个馒头 娇子 一顿 4个馒头  
    问 夫子今天吃了 3个馒头 娇子应该吃几个？ \*/  
    /\*计算出他们的倍数 4 / 2 2倍  
    3 \* 2 == 6个 \*/  
    /\* 大图盒子 / 小图盒子 倍数  
    我们 再小图移动的距离 \* 倍数 == 大图的位置\*/  
   // bigImage.style.left = -x \* big.offsetWidth /small.offsetWidth + "px";  
   // bigImage.style.top = -y \* big.offsetHeight /small.offsetHeight + "px";  
    //上面会导致显示不全的情况出现，因此正确的应该是  
     
    bigImage.**style**.**top** = -y \* bigImage.**offsetHeight** /smallImage.**offsetHeight** + "px";  
    bigImage.**style**.**left** = -x \* bigImage.**offsetWidth** /smallImage.**offsetWidth** + "px";  
    //上面会导致显示不全的情况出现，因此正确的应该是  
     
     
    }  
   </script>

防止选择拖动

我们知道 按下鼠标然后拖拽可以选择文字 的。

清除选中的内容

window.getSelection ? window.getSelection().removeAllRanges() : document.selection.empty();

拖动选择窗口

<script>

var box = document.getElementById("d\_box");

var drop = document.getElementById("drop");

startDrop(drop,box); // 鼠标放到 drop 但是移动 是 box

function startDrop(current,move) {

current.onmousedown = function(event) {

var event = event || window.event;

var x = event.clientX - move.offsetLeft - 205;

// 记录点击点到盒子边框的距离x，当这 clientx-x 就是大盒子到最外层的距离

var y = event.clientY - move.offsetTop - 155; // // 记录当前盒子的y位置

document.onmousemove = function(event) {

var event = event || window.event;

move.style.left = event.clientX - x + "px";

move.style.top = event.clientY - y + "px";

window.getSelection ? window.getSelection().removeAllRanges() : document.selection.empty();

}

}

document.onmouseup = function() { // 鼠标弹起之后， 鼠标继续移动不应该操作

document.onmousemove = null;

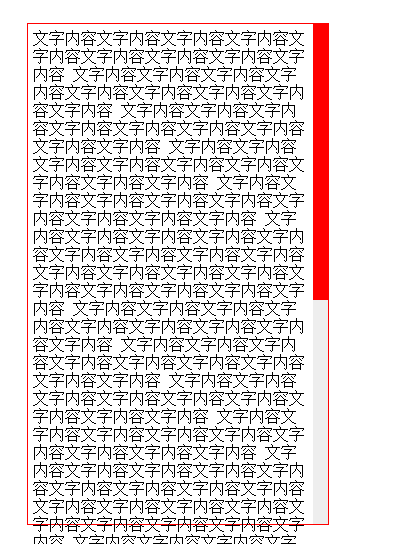
}

}

</script>

模拟垂直滚动条

有问题，当滚动条显示一半的时候回出现一些问题



// **滚动条的长度计算公式**: 容器的高度 / 内容的高度 \* 容器的高度

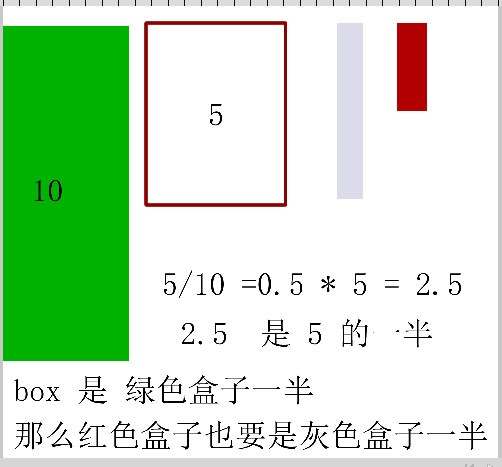
**//内容盒子要移动距离**// （内容盒子高度 - 大盒子高度） / (大盒子高度 - 红色盒子的高度) \* 红色盒子移动的数值

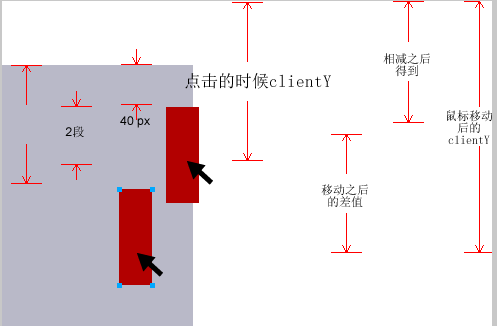
**.box>.contant+.scroll>.bar**

<style>  
 .box {  
 width: 300px;  
 height: 500px;  
 border: 1px solid red;  
 margin:100px;  
 position: relative;  
 }  
 .content {

//这里不能给宽度，因为给了之后padding会撑出  
 height: auto;  
 padding: 5px 18px 5px 5px;  
 position: absolute;  
 top: 0;  
 left: 0;  
 }  
 .scroll {  
 width: 18px;  
 height: 100%;  
 position: absolute;  
 top: 0;  
 right: 0;  
 background-color: #eee;  
 }  
 .bar {  
 width: 100%;  
 height: 100px;  
 background-color: red;  
 cursor: pointer;  
 border-radius: 10px;  
 position: absolute;  
 top: 0;  
 left: 0;  
 }  
</style>

红色盒子高度计算公式： 容器的高度 / 内容的高度 \* 容器的高度





红色方块移动一像素 ，我们的内容盒子移动多少呢？

（内容盒子高度 - 大盒子高度） / (大盒子高度 - 红色盒子的高度) 计算倍数

（内容盒子高度 - 大盒子高度） / (大盒子高度 - 红色盒子的高度) \* 红色盒子移动的数值

<script>  
 var box = **document**.getElementById("box"); // 最大的盒子  
 var content = box.**children**[0]; // 内容盒子  
 var scroll = box.**children**[1]; // 右边盒子  
 var bar = scroll.**children**[0];  
 // 1. 首先先要计算红色滚动条的高度 内容越多，滚动条越短 反之 反之  
 // 滚动条的长度计算公式: 容器的高度 / 内容的高度 \* 容器的高度  
 // box 是 内容盒子一半 那么红色盒子也要是box盒子的一半  
 var barHeight = box.**offsetHeight** / content.**offsetHeight** \* box.**offsetHeight**;  
 bar.**style**.**height** = barHeight + "px";  
 // 下面开始 拖动 红色盒子  
 startScroll(bar,content); // 第一次参数 拖动的 第二个参数 内容的盒子  
 function startScroll(obj,target) {  
 obj.onmousedown = function(event) {  
 // alert(11);  
 var event = event || **window**.**event**;  
 var t = event.**clientY** - this.**offsetTop** ; // 红色盒子距离 父亲 盒子顶部距离//  
 //这里的client- offsettop得到的是鼠标点到红色盒子上部+文章内部距页面上部的距离  
 var that = this; // 把 bar 对象给 that 对象  
 **document**.onmousemove = function(event) {  
 var event = event || **window**.**event**;  
 var barTop = event.**clientY** - t ; //=offset的  
 //内容盒子要移动距离  
 // （内容盒子高度 - 大盒子高度） / (大盒子高度 - 红色盒子的高度) \* 红色盒子移动的数值  
 var contentTop = (target.**offsetHeight** - target.parentNode.**offsetHeight**) / (target.parentNode.**offsetHeight** - that.**offsetHeight**) \* barTop;  
 // 内容盒子移动的距离  
 if(barTop < 0)  
 {  
 barTop = 0;  
 }  
 else if(barTop > target.parentNode.**offsetHeight** - that.**offsetHeight**)  
 // 大于 大盒子的高度 - 红色盒子的高度  
 {  
 barTop = target.parentNode.**offsetHeight** - that.**offsetHeight** ;  
 }  
 else  
 {  
 target.**style**.**top** = -contentTop + "px"; // 往上走是负值，只有滑块在滚动条里面的时候才能动  
 }  
 that.**style**.**top** = barTop + "px";  
 **window**.getSelection ? **window**.getSelection().removeAllRanges() : **document**.**selection**.empty(); // 防止拖动滑块的时候， 选中文字  
 }  
 }  
 **document**.onmouseup = function() {  
 **document**.**onmousemove** = null;  
 }  
 }  
</script>

Html基本结构访问方法

文档是 document

html body head

**document.head**

**document.body**

**document.title**

没有 document.html

取而代之的是 **document.documentElement;**

scroll家族

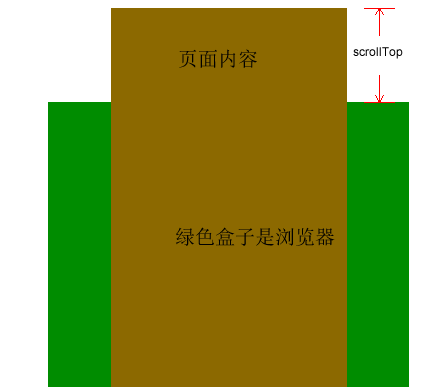
Offset 自己的 偏移

scroll 滚动的

scrollTop scrollLeft

scrollTop 被卷去的头部 ，用于将定位换成fixed

它就是当你滑动滚轮浏览网页的时候**网页隐藏在屏幕上方的距离**



怎么得到scrollTop

我们学习一个事件 ： 页面滚动效果

window.onscroll = function() { 页面滚动语句 }

谷歌浏览器 和没有声明 DTD <DOCTYPE > ：

document.body.scrollTop;

火狐 和其他浏览器 ie 678

document.documentElement.scrollTop;

ie9+ 和 最新浏览器 都认识

window.pageXOffset; pageYOffset （scrollTop）

兼容性写法：

1. var scrollTop = window.pageYOffset || document.documentElement.scrollTop
2. || document.body.scrollTop || 0;

JSON

var myjson={k:v,k:v,k:v...} 键值对 key: value color: red;

Json一般就是被当做一个配置单用；有逗号

var json = { key: value， key1:value }

var json1 = {**name** :"刘德华",**age**: 55};  
**console**.log(json1.**name**); // 输出名字 刘德华  
**console**.log(json1.**age**); // 输出年龄 55

判断是不是怪异模式的浏览器（没有声明头的）

**document**.**compatMode** == "CSS1Compat"

检测是不是， 不是为true

document.compatMode === "BackCompat"

BackCompat 未声明

CSS1Compat 已经声明

注意大小写

封装好的scroll

function scroll() {

if(window.pageYOffset != null) // ie9+ 和其他浏览器

{

return {

left: window.pageXOffset,

top: window.pageYOffset

}

}

else if(document.compatMode == "CSS1Compat") // 声明的了 DTD

// 检测是不是怪异模式的浏览器 -- 就是没有 声明<!DOCTYPE html>

{

return {

left: document.documentElement.scrollLeft,

top: document.documentElement.scrollTop

}

}

return { // 剩下的肯定是怪异模式的

left: document.body.scrollLeft,

top: document.body.scrollTop

}

}

Window.onscroll

移动到某些位置出现或消失什么

<script>  
 var nav = $("Q-nav");  
 var navTop = nav.**offsetTop**; // 得到导航栏距离顶部的距离 168  
 **console**.log(navTop);  
 **window**.onscroll = function() {  
 // console.log(nav.offsetTop);  
 if(scroll().**top** >= navTop)  
 {  
 //alert("到位置了");  
 nav.**className** = "nav fixed";  
 //因为涉及的项目比较多所以样式写在class里面  
 }  
 else  
 {  
 nav.**className** = "nav";  
 }  
 }  
</script>

使用

window.onscroll = function() {

console.log(scroll().top);

}

侧边栏的广告随滚动显示：不采用fixed

<script>  
 **window**.onload = function() {  
 var pic = $("pic");  
 var leader = 0;  
 var target = 0;  
 var timer = null; // 定时器  
 var top = pic.**offsetTop**; // 50  
 **window**.onscroll = function() {  
 clearInterval(timer);  
 target = scroll().**top** + top; // 把最新的 scrolltop 给 target  
 timer = setInterval(function() {  
 leader = leader + (target - leader ) / 10;  
 pic.**style**.**top** = leader + 'px';  
 },30)  
 }  
 }  
</script>

scrollTo(x,y)

**window.scrollTo(15,15); //不加单位**

**方法可把内容滚动到指定的坐标。**

格式：

scrollTo(xpos,ypos)

xpos 必需。要在窗口文档显示区左上角显示的文档的 x 坐标。

ypos必需。要在窗口文档显示区左上角显示的文档的 y 坐标

因为我们的网页大部分都没有水平滚动条，所以，这个x 不太常用。

返回顶部

方法1：

直接给图片加上一个

<a href=”#”> </a>

方法二：

<script>  
 **window**.onload = function() {  
 var goTop = $("gotop");  
 **window**.onscroll = function() {  
 scroll().**top** > 0 ? show(goTop) : hide(goTop); // 如果大于0 就显示 否则隐藏  
 leader = scroll().**top**; // 把 卷去的头部 给 起始位置  
 **console**.log(scroll().**top**);  
 }  
 var leader = 0,target = 0,timer = null;  
 // leader 起始位置 target 目标位置  
 goTop.onclick = function() {  
 target = 0; // 点击完毕之后 奔向0 去的 不写也可以  
 timer = setInterval(function() {  
 leader = leader + (target - leader ) / 10;  
 **window**.scrollTo(0,leader); // 去往页面中的某个位置  
 if(leader == target)  
 {  
 clearInterval(timer);  
 }  
 },20);  
  
 }  
  
 }  
</script>

通过边框浏览各个版面

Ul>li\*5

Ol>li\*5

每个ul中的li占领整个屏幕

html,body {

width: 100%;

height: 100%;

}

#ul {

width: 100%;

height: 100%;

}

ul li{

width: 100%;

height: 100%;

}

**Js**

var leader = 0,target = 0,timer = null;

for(var i= 0; i<ulBoxLi.length; i++)

{

ulBoxLi[i].style.backgroundColor = bgColor[i];

olBoxLi[i].style.backgroundColor = bgColor[i];

olBoxLi[i].index = i; // 记录当前的索引号

olBoxLi[i].onclick = function() {

clearInterval(timer);

target = ulBoxLi[this.index].offsetTop; // 核心语句

timer = setInterval(function() {

leader = leader + (target - leader ) /10;

window.scrollTo(0,leader); // 屏幕滑动

//pic.style.left = leader + 'px';

},30)

}

}

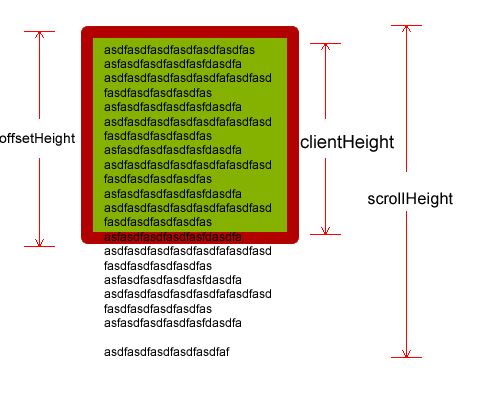
#### client 家族

client 可视区域

offsetWidth: width + padding + border （披着羊皮的狼）

clientWidth： width + padding 不包含border

scrollWidth: 大小是内容的大小



检测屏幕宽度(可视区域)

ie9及其以上的版本

window.innerWidth

标准模式

document.documentElement.clientWidth

怪异模式

document.body.clientWidth

自己封装一个 返回可视区宽度和高度的函数。

封装可视区宽度

function client() {  
 if(**window**.**innerWidth** != null) // ie9 + 最新浏览器  
 {  
 return {  
 **width**: **window**.**innerWidth**,  
 **height**: **window**.**innerHeight** }  
 }  
 else if(**document**.**compatMode** === "CSS1Compat") // 标准浏览器  
 {  
 return {  
 **width**: **document**.**documentElement**.**clientWidth**,  
 **height**: **document**.**documentElement**.**clientHeight** }  
 }  
 return { // 怪异浏览器  
 **width**: **document**.**body**.**clientWidth**,  
 **height**: **document**.**body**.**clientHeight** }  
}

window.onresize 改变窗口事件

昨天 window.onscroll = function() {} 屏幕滚动事件

今天 window.onresize = function() {} 窗口改变事件

onresize 事件会在窗口或框架被调整大小时发生

响应式

要求：

当我们的页面宽度大于 960 像素的时候 页面颜色是红色

当我们的页面宽度 大于 640 小于 960 页面的颜色是 绿色

剩下的颜色是 蓝色

**window**.onresize = change;  
 function change() {  
 var width = client().**width**;//返回的是数值没有单位  
 if (width > 900) {  
 **document**.**body**.**style**.**backgroundColor** = "green";  
 } else if (width > 600) {  
 **document**.**body**.**style**.**backgroundColor** = "red";  
 } else {  
 **document**.**body**.**style**.**backgroundColor** = "#ff0";  
 }  
 }  
</script>

检测屏幕宽度(分辨率)

clientWidth 返回的是 可视区 大小 浏览器内部的大小

**window.screen.width**  返回的是我们电脑的 分辨率 跟浏览器没有关系

简单冒泡机制

事件冒泡: 当一个元素上的事件被触发的时候，比如说鼠标点击了一个按钮，同样的事件将会在那个元素的所有祖先元素中被触发。这一过程被称为事件冒泡；这个事件从原始元素开始一直冒泡到DOM树的最上层。

顺序

E 6.0:

div -> body -> html -> document

其他浏览器:

div -> body -> html -> document -> window

不是所有的事件都能冒泡。以下事件不冒泡：blur、focus、load、unload

阻止冒泡的方法

标准浏览器 和 ie浏览器 propagation传播

w3c的方法是**event.stopPropagation()**  proPagation 传播 传递

IE则是使用**event.cancelBubble = true**  bubble 冒泡 泡泡 cancel 取

**兼容的写法：**

1. if(event && event.stopPropagation)
2. {
3. event.stopPropagation(); // w3c 标准
4. }
5. else
6. {
7. event.cancelBubble = true; // ie 678 ie浏览器
8. }

小案例 点击空白处隐藏盒子

这个案例就是说，一个盒子，点击除了自己之外的任何一个地方，就会隐藏。

原理：

点击自己不算 ( 怎么证明我是我 点击的这个对象id 正好和自己一样 )

点击空白处 就是点击 document

判断当前对象

火狐 谷歌 等 event.target.id

ie 678 event.srcElement.id

兼容性写法：

var targetId = event.**target** ? event.**target**.id : event.**srcElement**.id;

targetId != "show"

代码：

1. <!DOCTYPE html>
2. <html>
3. <head lang="en">
4. <meta charset="UTF-8">
5. <title></title>
6. <style>
7. body {
8. height:2000px;
9. }
10. #mask {
11. width: 100%;
12. height: 100%;
13. opacity: 0.4; /\*半透明\*/
14. filter: alpha(opacity = 40); /\*ie 6半透明\*/
15. background-color: black;
16. position: fixed;
17. top: 0;
18. left: 0;
19. display: none;
20. }
21. #show {
22. width: 300px;
23. height: 300px;
24. background-color: #fff;
25. position: fixed;
26. left: 50%;
27. top: 50%;
28. margin: -150px 0 0 -150px;
29. display: none;
30. }
31. </style>
32. </head>
33. <body>
34. <a href="javascript:;" id="login">注册</a>
35. <a href="javascript:;">登录</a>
36. <div id="mask"></div>
37. <div id="show"></div>
38. </body>
39. </html>
40. <script>
41. function $(id) { return document.getElementById(id);}
42. var login = document.getElementById("login");
43. login.onclick = function(event) {
44. $("mask").style.display = "block";
45. $("show").style.display = "block";
46. document.body.style.overflow = "hidden"; // 不显示滚动条
47. //取消冒泡
48. var event = event || window.event;
49. if(event && event.stopPropagation)
50. {
51. event.stopPropagation();
52. }
53. else
54. {
55. event.cancelBubble = true;
56. }
57. }
58. document.onclick = function(event) {
59. var event = event || window.event;
60. // alert(event.target.id); // 返回的是点击的某个对象的id 名字
61. // alert(event.srcElement.id);
62. var targetId = event.target ? event.target.id : event.srcElement.id;
63. // 看明白这个写法
64. if(targetId != "show") // 不等于当前点点击的名字
65. {
66. $("mask").style.display = "none";
67. $("show").style.display = "none";
68. document.body.style.overflow = "visible"; // 显示滚动条
69. }
70. }
71. </script>

选中之后，弹出层

我们想，选中某些文字之后，会弹出一个弹出框

这个和 我们前面讲过的拖拽有点不一样。

拖拽 是拖着走。 拉着鼠标走 。

选择文字： 这个弹出的层 选中的时候不出来，弹起鼠标的时候才出现 。

所以这个的事件一定是 onmouseup . 盒子显示而且盒子的位置 再 鼠标的 clientX 和 clientY 一模一样

用来判断选择的文字：

获得用户选择内容

window.getSelection() 标准浏览器

document.selection.createRange().text; ie 获得选择的文字

兼容性的写法：

if(**window**.getSelection)  
{  
 txt = **window**.getSelection().toString(); // 转换为字符串  
}  
else  
{  
 txt = **document**.**selection**.createRange().**text**; // ie 的写法  
}

综合代码:

1. <!DOCTYPE html>
2. <html>
3. <head lang="en">
4. <meta charset="UTF-8">
5. <title></title>
6. <style>
7. div {
8. width: 400px;
9. margin:50px;
10. }
11. #demo {
12. width: 100px;
13. height: 100px;
14. background-color: pink;
15. position: absolute;
16. top: 0;
17. left: 0;
18. display: none;
19. }
20. </style>
21. </head>
22. <body>
23. <span id="demo"></span>
24. <div id="test">我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字我要复制的文字</div>
25. <div id="another">
26. 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字 我不要的文字
27. </div>
28. </body>
29. </html>
30. <script>  
     function $(id) {return **document**.getElementById(id)}  
     $("test").**onmouseup** = function(event) {  
     var event = event || **window**.**event**;  
     var x = event.**clientX**; // 鼠标的x坐标  
     var y = event.**clientY**; // 同理  
     var txt; // 用于存贮文字的变量  
     if(**window**.getSelection) // 获取我们选中的文字  
     {  
     txt = **window**.getSelection().toString(); // 转换为字符串  
     }  
     else  
     {  
     txt = **document**.**selection**.createRange().**text**; // ie 的写法  
     }  
     if(txt) // 所有的字符串都为真 "" 是假 所有的数字为真 0 为假  
     {  
     //看看有没有选中的文字，没有选中文字为空的，就不应该执行 点击一下鼠标 就是空的  
     showBox(x,y,txt); // 调用函数  
     }  
     }  
     **document**.onmousedown = function(event) { // 点击空白处隐藏  
     var event = event || **window**.**event**;  
     var targetId = event.**target** ? event.**target**.**id** : event.**srcElement**.**id**;  
     if(targetId != "demo"){  
     $("demo").**style**.**display** = "none";  
     }  
     }  
     function showBox(mousex,mousey,contentText) { // 相关操作  
     setTimeout(function() {  
     $("demo").**style**.**display** = "block";  
     $("demo").**style**.**left** = mousex + "px";  
     $("demo").**style**.**top** = mousey + "px";  
     $("demo").**innerHTML** = contentText;  
     },200)  
      
      
     }

动画原理

人走路的时候， 步长

动画的基本原理 ： 让盒子的 offsetLeft + 步长

盒子 原来的位置 0 + 10 盒子现在的offsetLeft 10

10 + 10 = 20 + 10

原理：

<script>  
 //动画的基本原理 盒子的 offsetLeft + 步长  
 var btn = **document**.getElementsByTagName("button")[0];  
 var div = **document**.getElementsByTagName("div")[0];  
 var timer = null;  
 btn.onclick = function() {  
 timer = setInterval(function() {  
 if(div.**offsetLeft** > 400)  
 {  
 clearInterval(timer);  
 }  
 div.style.**left** = div.**offsetLeft** + 10 + "px";  
 },20);  
 }  
</script>

|-5| = 5

Math.abs(-5) 取绝对值函数 js 就是 数学计算

匀速运动封装函数

function animate(obj,target){  
 clearInterval(obj.**timer**); // 先清除定时器  
 var speed = obj.**offsetLeft** < target ? 5 : -5; // 用来判断 应该 + 还是 -  
 obj.**timer** = setInterval(function() {  
 var result = target - obj.**offsetLeft**; // 因为他们的差值不会超过5  
 obj.**style**.**left** = obj.**offsetLeft** + speed + "px";  
 if(**Math**.abs(result)<=5) // 如果差值不小于 5 说明到位置了  
 {  
 clearInterval(obj.**timer**);  
 obj.**style**.**left** = target + "px"; // 有5像素差距 我们直接跳转目标位置  
 }  
 },30)  
}

三个取整函数

Math.ceil() Math.floor() Math.round()

Math

Math.ceil() 向上取整 天花板

比如说 console.log(Math.ceil(1.01)) 结果 是 2

console.log(Math.ceil(1.9)) 结果 2

console.log(Math.ceil(-1.3)) 结果 是 -1

Math.floor() 向下取整 地板

比如说 console.log(Math.floor(1.01)) 结果 是 1

console.log(Math.floor(1.9)) 结果 1

console.log(Math.floor(-1.3)) 结果 是 -2

Math.round() 四舍五入函数

console.log(Math.round(1.01)) 结果 是 1

console.log(Math.round(1.9)) 结果 是 2

缓动动画原理

匀速动画的原理： 盒子本身的位置 + 步长

缓动动画的原理: 盒子本身的位置 + 步长 (不断变化的)

封装代码：

1. function animate(obj,target){ // 第一个参数 动谁 第二个参数 动多少
2. clearInterval(obj.timer);
3. obj.timer = setInterval(function() {
4. // 计算步长 动画的原理 盒子本身的位置 + 步长
5. var step = (target - obj.offsetLeft) / 10; // 步长
6. step = step > 0 ? Math.ceil(step) : Math.floor(step); // 取整步长
7. // obj.style.left = 盒子本身的位置 + 步长
8. obj.style.left = obj.offsetLeft + step + "px";
9. if(obj.offsetLeft == target){
10. clearInterval(obj.timer);
11. }
12. },30)
13. }

js 常用 访问 CSS 属性

我们访问得到css 属性，比较常用的有两种：

1. 利用点语法

box.style.width box.style.top

点语法可以得到 width 属性 和 top属性 带有单位的。 100px

但是这个语法有非常大的缺陷， 不变的。

后面的width 和 top 没有办法传递参数的。

var w = width;

~~box.style.w~~

2. 利用 [] 访问属性

语法格式: **box.style[“width”]**

元素.style[“属性”];

**console**.log(box.**style**["left"]);

最大的优点 ： 可以给属性传递参数

得到css 样式

核心： 我们怎么才能得到**内嵌或者外链的样式**呢？

1. obj.currentStyle ie opera 常用

obj.currentStyle.left

外部（使用<link>）和内嵌（使用<style>）样式表中的样式（ie和opera）

2 .window.getComputedStyle("元素", "伪类") w3c

两个选项是必须的， 没有伪类 用 null 替代

3 兼容写法 ：

我们这个元素里面的属性很多， left top width ===

我们想要某个属性， 就应该 返回改属性，所有继续封装 返回当前样式的 函数。

/\*\*  
获取css当前样式 带单位的字符串  
 \*/  
function getStyle(obj,attr){  
 if(obj.**currentStyle**){  
 return obj.**currentStyle**[attr];  
 }else {  
 return **window**.getComputedStyle(obj,null)[attr];  
 }  
}

动画

/\*\*  
 动画  
 \*/  
function animate(obj,attr,target){  
 clearInterval(obj.**timer**);  
  
 obj.**timer**=setInterval(function () {  
 var current=parseInt(getStyle(obj,attr));  
 var step=(target-current)/10;  
 step=step>0?**Math**.ceil(step):**Math**.floor(step);  
 obj.**style**[attr]=current+step+"px";  
 if(current==target){  
 clearInterval(obj.**timer**);  
 }  
  
 },30)  
}

JSON 遍历

**for in 关键字**

**for ( 变量 in 对象)**

**{ 执行语句; }**

var json = {**width**:200,**height**:300,**left**:50}  
**console**.log(json.**width**);  
for(var k in json)  
{  
 **console**.log(k); // k 遍历的是json 可以得到的是 属性  
 **console**.log(json[k]); // json[k] 得到 是属性的 值  
}

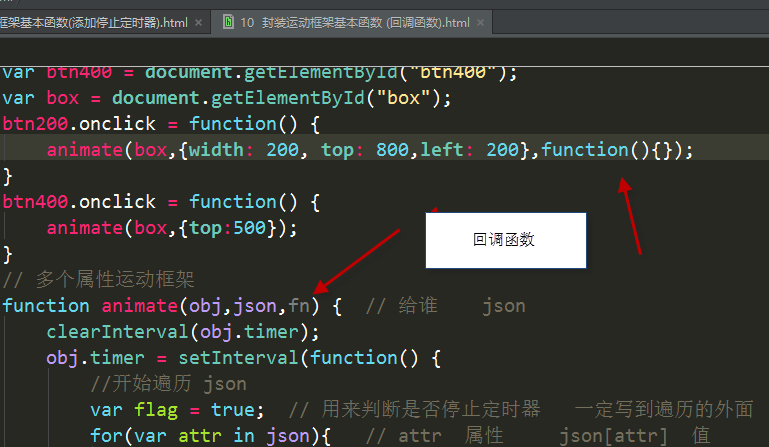
千万要记得 每个 的意思 ： 那是相当重要

**k 是 属性**

json[k] 得到的是属性值

多个条件同时满足

回调函数

**等动画执行完毕再去执行的函数** 回调函 数 

**很简单 当定时器停止了。 动画就结束了**

**上面直接写函数体就行，可以在函数里面在写一个动画**

in 运算符

in运算符**要求第1个（左边的）操作数必须是字符串类型**或可以**转换为字符串类型的其他类型**，**而第2个（右边的）操作数必须是数组或对象**。 左边是否存在右边里

// in 可以用用来判断 json 里面有没有某个属性

var json = {**name**: "刘德华",**age** : 55};  
// in 可以用用来判断 json 里面有没有某个属性  
if("andy" in json)  
{  
 **console**.log("yes"); // 返回的是 yes   
}  
else  
{  
 **console**.log("no");  
}

手风琴

同时减小，自己增大

<script>  
 var box = **document**.getElementsByTagName("div")[0];  
 var lis = box.children[0].**children**;  
 for(var i=0;i<lis.**length**;i++)  
 {  
 lis[i].**style**.**backgroundImage** = "url(images/"+(i+1)+".jpg)";  
 lis[i].**onmouseover** = function() {  
 for(var i=0;i<lis.**length**;i++)  
 {  
 animate(lis[i],{**width**:100});  
 }  
 animate(this,{**width**:800});  
 }  
 lis[i].**onmouseout** = function(){  
 for(var i=0;i<lis.**length**;i++)  
 {  
 animate(lis[i],{**width**:240});  
 }  
 }  
 }  
</script>

Js中一些css属性的写法

Border-left borderLeft

z-index zIndex

网易轮播图

跑马图

瀑布图

闭包

**一次调用外部函数，外部执行域的非持久型变量神奇地保留他们在闭包最初定义(或创建)时的值**。 **保留闭包的初始值，使用函数内部的数据**

函数使用该函数外的变量，也就是外函数的变量变成了全局变量

白话： 我**们可以用一个函数 去访问 另外一个函数的内部变量的方式就是闭包。**

**测试题**

1. function outerFun()
2. {
3. var a=0;
4. function innerFun()
5. {
6. a++;
7. alert(a);
8. }
9. return innerFun; //注意这里
10. }
11. var obj=outerFun();
12. obj(); obj();
13. var obj2=outerFun();
14. obj2(); obj2();
15. 结果是 ： 1 2 1 2

闭包的优点 ：

优点：不产生全局变量，实现属性私有化。

缺点：闭包中的数据会常驻内存，在不用的时候要删掉否则会导致内存溢出。

tab栏切换

**最初的**

<div id="first">  
 <div class="spanbox"><span class="choose">1</span><span>2</span><span>3</span>  
 </div>  
 <ul>  
 <li class="show">1个盒子显示</li>  
 <li>2个盒子显示</li>  
 <li>3个盒子显示</li>  
 </ul>  
</div>

window.onload= **function** () {  
 show("first");  
 **function** show(id){  
 **var** box=document.getElementById(id);  
 **var** span=box.getElementsByTagName("span");  
 **var** li=box.getElementsByTagName("li");  
 **for**(i=0,len=span.length;i<len;i++){  
 span[i].index=i;  
 span[i].onmouseover=**function**(){  
 **for**(j=0;j<span.length;j++){  
 span[j].className="";  
 li[j].className="";  
 }  
 **this**.className="choose";  
 li[**this**.index].className="show";  
 }  
 }  
 }  
}

**改进1**

<script>  
 **window**.onload = function(){  
 //要想多个盒子不相互影响 ，我们可以通过id 给他们分开  
 //封装tab栏切换函数  
 function tab(obj){  
 var target = **document**.getElementById(obj);  
 var spans = target.getElementsByTagName("span");  
 var lis = target.getElementsByTagName("li");  
 for(var i=0;i<spans.**length**;i++)  
 {  
 var timer = null;  
 spans[i].**onmouseover** = function (num) {  
 return function(){  
 clearTimeout(timer);  
 timer = setTimeout(function(){  
 for(var j=0; j<spans.**length**;j++)  
 {  
 spans[j].**className** = "";  
 lis[j].**className** = "";  
 }  
 spans[num].**className** = "current";  
 lis[num].**className** = "show";  
 },300)  
 }  
 }(i);  
 spans[i].onmouseout = function() {  
 clearTimeout(timer);  
 }  
 }  
 }  
 tab("one");  
  
 }  
</script>

**对比**

for(**i**=0,**len**=span.**length**;**i**<**len**;**i**++){  
 span[**i**].**onmouseover**=move(**i**);  
}  
function move(i){  
 return function(){  
 for(**j**=0;**j**<**len**;**j**++){  
 span[**j**].**className**="";  
 li[**j**].**className**="";  
 }  
 span[i].**className**="choose";  
 li[i].**className**="show";  
 }  
  
}

for(var i=0;i<spans.**length**;i++) {  
 spans[i].**onmouseover** = function (num) {  
 return function(){  
 for(var j=0; j<spans.**length**;j++)  
 {  
 spans[j].**className** = "";  
 lis[j].**className** = "";  
 }  
 spans[num].**className** = "current";  
 lis[num].**className** = "show";  
 }  
 }(i);  
}

**带闭包节流的tab栏**

function tab(obj){  
 var target = **document**.getElementById(obj);  
 var spans = target.getElementsByTagName("span");  
 var lis = target.getElementsByTagName("li");  
 for(var i=0;i<spans.**length**;i++)  
 {  
 // spans[i].index = i;  
 var timer = null;  
 spans[i].**onmouseover** = function (num) {  
 return function(){  
 clearTimeout(timer);  
 timer = setTimeout(function(){  
 for(var j=0; j<spans.**length**;j++)  
 {  
 spans[j].**className** = "";  
 lis[j].**className** = "";  
 }  
 spans[num].**className** = "current";  
 lis[num].**className** = "show";  
 },300)  
  
 }  
 }(i);  
 spans[i].onmouseout = function() {  
 clearTimeout(timer);  
 }  
  
  
 }  
}

屏幕缩放

<script>  
 var num = 0;  
 var demo = **document**.getElementById("demo")  
 **window**.onresize = function() { // ie8 等浏览器 每触发一次 ，执行两次  
 num++;  
 demo.**innerHTML** = **window**.**innerWidth** || **document**.**documentElement**.**clientWidth**; // 得到当前的屏幕宽度  
 **console**.log(num);  
 }  
</script>

闭包的屏幕缩放

var num = 0;  
var demo = **document**.getElementById("demo")  
**window**.onresize = throttle(function(){  
 demo.**innerHTML** = **window**.**innerWidth** || **document**.**documentElement**.**clientWidth**;  
 num++;  
 **console**.log(num);  
},300);  
function throttle(fn,delay) { // 闭包 节流  
 var timer = null;  
 return function() {  
 clearTimeout(timer);  
 timer = setTimeout(fn,delay);  
 }  
}

对象（object）

基本数据类型 string number boolean null undefined

Array 对象

对象数据类型： 对象就是带有属性和方法的 数据类型

var num = 10; // 变量  
var arr = []; // 数组  
arr.**index** = 10; // 数组arr 的 一个 index 属性

但是有个问题， 我们想要某些属性或者方法的时候，用数组不合适。

arr.lenght .

我们想要自己id属性和方法 。 要求这个一定是个对象才行。

声明对象

var obj = new Object();

var obj = {};

var obj = {}; // 声明对象  
obj.**name** = "刘德华"; // 属性  
obj.**age** = 55;  
obj.showName = function() { // 声明方法 方法一定带有 ()  
 alert("俺是刘德华");  
}  
obj.showAge = function() {  
 alert("俺今年18岁");  
}

使用对象

**console**.log(obj.**name**); // 调用属性  
**console**.log(obj.**age**);  
obj.showName(); // 调用方法  
obj.showAge();

函数型对象

function Person(name,age) {  
 var obj = {}; // 声明对象  
 obj.**name** = name;  
 obj.**age** = age;  
 obj.showName = function() {  
 alert("我的名字是" + name);  
 }  
 obj.showAge = function() {  
 alert("我的年龄是" + age);  
 }  
 return obj; // 返回对象 this  
}  
var xiaoming = Person("小明1",12);

//xiaoming=  
**console**.log(xiaoming.**name**);  
**console**.log(xiaoming.**age**);

This

function fun() { // this 指向的是 函数的调用者 或者是 事件的调用者  
 alert(this);  
}  
fun(); // 这个的this 指向的是 window  
  
function fn() {  
 alert(this);  
}  
new fn(); // 添加了 new 的函数 this 指向的是 obj 新的对象  
  
function again() {  
 this.**x** = 10; // x window  
}  
again();  
**console**.log(x); // **window**.x  
function person() {  
 this.**x** = 20;  
}  
var demo = new person();  
var demo1 = new person();  
 // 实例 对象 吃的苹果 类  
**console**.log(demo.**x**);

面向对象

1. 抽象

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。

2. 封装

封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面

3. 继承

4. 多态

多态是指两个或多个属于不同类的对象，对于同一个消息（方法调用）作出不同响应的方式



new

新的 我们经常利用new 关键字 去声明新的对象

javascript

new运算符的作用是创建一个对象实例。这个对象可以是用户自定义的，也可以是带构造函数的一些系统自带的对象。

new 关键字可以让 this 指向新的对象

所谓"构造函数"，其实就是一个普通函数，但是内部使用了this变量。对构造函数使用new运算符，就能生成实例，并且this变量会绑定在实例对象上。

构造函数

==类

function Person(name,age) { // 构造函数 就是一个普通函数 为了和普通函数区别 第一个字母大写  
 this.**name** = name;  
 this.**age** = age;  
 this.showName = function() {  
 alert("我的名字是" + name);  
 }  
 this.showAge = function() {  
 alert("我的年龄是"+ age);  
 }  
}  
var demo = new Person("刘德华",18);  
var demo1 = new Person("刘德华",18);

alert(demo1.showName);  
alert(demo.**name** == demo1.**name**); //true  
alert(demo.showName == demo1.showName); //false

prototype 原型

共同的 相同的 部分

主要解决：函数因为使用非常非常多，重复执行效率太低。

Person.**prototype**.showName = function() { // 用的共同的父亲  
 alert("我的名字是"+ this.**name**);  
 }

类.prototype.方法 = function() {} 具体格式

可以把那些不变的属性和方法，直接定义在prototype对象上

使用方法：

类名.prototype.方法

原型形对象

function Person(name,age) { // 构造函数  
 this.**name** = name; // 只写属性  
 this.**age** = age;  
}  
 Person.**prototype**.showName = function() { // 用的共同的父亲  
 alert("我的名字是"+ this.**name**);  
 }  
 Person.**prototype**.showAge = function() {  
 alert("我的名字是"+ this.**age**);  
 }  
 var demo = new Person("刘德华",15);  
 var demo1 = new Person("刘德华",15);  
 demo.showName();  
 alert(demo.showName === demo1.showName); //true

列表显示

JQuary

jQuery的基本使用



min：它是压缩过的版本

区别：我们开发过程中，会用未压缩的版本，或者压缩的。

项目上线的时候，我们要用压缩过的版本。

版本问题：

1. 1.xxx 版本 jQuery-1.11.1.js
2. 2.xxx 版本 不再支持IE6、7、8

入口函数

$(document).ready(function(){});

$(function(){}); 只是是文档加载完成

相当于

Window.onload=function(){}

js入口函数跟jQuery入口函数的区别：

1. Js的window.onload事件是等到所有内容，以及我们的**外部图片之类的文件加载完了之后，才回去执行 所有资源加载完成之后才能运行**
2. jQuery的入口函数 是在 **html所有标签都加载之后**，**就回去执行。**

jQuery详细介绍

$问题

* 1. Js命名归法：下划线、字母、$、数字
  2. 但是不能以数字作为开头

var $ = “我是$符号”;

jQUery的两个变量：$ 和 jQuery

jQuery占用了我们两个变量：$ 和 jQuery

Js创建对象

三种方式：

1. var obj = {};
2. var obj1 = new Object(); 效率低点 ，存在原型的问题
3. var obj2 = Object.create();

1跟2的区别：

推荐使用第一个方式

第二种方式存在效率问题，因为要**new对象，会涉及到原型查找的问题**。

Css（）

$(“#id”).css(“background”); //获得属性的值 带单位的字符串

$(“#id”).css(“background”,”red”); //设置对象的值 三种

$(“tagname”).css(“background”,”red”); //所有的对应标签都应用 ，**隐式迭代**

// 设置样式  
$("li").css("background-color","red");  
  
// 设置多个样式  
$("li").css({  
 **"background-color"**: "red",  
 **"font-size"**: "20px",  
 **"color"**: "white"  
});  
  
// 设置样式，可以根据我们自己的意愿来设置  
$("li").css("font-size", function (i,v) {  
 // index: 指定当前元素的索引号  
 // value: 当前元素，前面属性的值

//通过return 字符串或者数字来改变当前元素的属性值  
 //return (i + 1) \* parseInt(v) + "px"  
 // 只设置第三个  
 if(i === 2) {  
 return (i + 1) \* parseInt(v) + "px";  
 }

//不做操作的保持了原有值  
});

和上面一样多个属性设置多个值得时候的用法

$("div").click(function() {

$(this).css({

width: function(index, value) {

return parseFloat(value) \* 1.2;

},

height: function(index, value) {

return parseFloat(value) \* 1.2;

}

});

});

增删类样式

$(“#id”).addClass(“className”);//没有 .

$(“#id”).removeClass(“className”);//没有 . 不给参数就删除全部class，制定的为不存在的样式不起作用

$(“#id”).hasClass(“className”); . 返回布尔值

$(“#id”).toggleClass(“className”);//切换 没有的加有的删

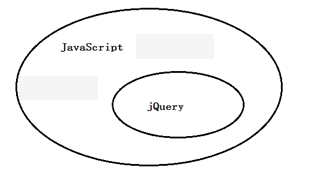
jQuery是什么

jQuery就是javascript的一个库，把我们常用的一些功能进行了封装，方便我们来调用，提高我们的开发效率。

Javascipt跟jQuery的区别：

Javascript是一门编程语言，我们用它来编写客户端浏览器脚本。

jQuery是javascript的一个库，包含多个可重用的函数，用来辅助我们简化javascript开发



jQuery能做的javascipt都能做到，而javascript能做的事情，jQuery不一定能做到。

选择器

|  |  |  |
| --- | --- | --- |
| **$(“#demo”)** | 选择id为demo的第一个元素，一般用J\_开头 | $(“#demo”).css(“background”,”red”) |
| **$(“.liItem”)** | 选择所有类名（样式名）为liItem的元素 | $(“.liItem”). css(“background”,”red”); |
| **$(“div”)** | 选择所有标签名字为div的元素 | $(“div”). css(“background”,”red”); |
| **$(“.liItem,div”)** | 选择多个指定的元素，这个地方是选择出了 .liItem元素和div元素 | $(“.liItem,div”). css(“background”,”red”) |
| **空格** | 后代选择器  选择所有的后代元素 | $(“div span”). css(“background”,”red”); |
| * **a>b也可以**   **a>b>c** | 子代选择器  选择所有的子代元素 | $(“div > span”). css(“background”,”red”) |
| **+** | 紧邻选择器 同级  选择紧挨着的**兄弟**下一个元素 | $(“div + p”). css(“background”,”red”) |
| **~** | 兄弟选择器  选择后面的所有的兄弟元素 | $(“div ~ p”). css(“background”,”red”) |
| **:eq(index)** | index是从0开始的一个数字，选择序号为index的元素。选择第一个匹配的元素。 | $(“li:eq(1)”). css(“background”,”red”) |
| **$(“a[href]”)** | 选择所有包含href属性的元素 | $(“a[href]”). css(“background”,”red”) |
| **$(“a[href=’itcast’]”)** | 选择href属性值为itcast的所有a标签 | $(“a[href=’itcast’]”). css(“background”,”red”) |
|  |  |  |

**// .eq(index)**  
// 类比 ":eq(index)" 选择器  
$("span").eq(1).css("font-size","40px");  
**// .first() .last()**// 类比 ":first" 选择器  
$("span").first().css("font-size", "40px");  
/**/ .parent([exp])**  
// 选择父亲元素  
$("#sp").parent().css("background", "gold");  
//里面可以传入一个选择器  
$("p").parent("#div").css("background", "gold");  
  
**// .siblings()**  
// 选择所有的兄弟元素  
$("#li04").siblings().css("background", "gold");  
$("#li04").siblings(".liItem").css("background", "gold");  
**.children（[exp]）**  
**// .find()// 查找所有后代元素**  
$("div").find("#li04").css("background", "gold");

**//.next([exp])紧邻**

下一个，可以传入一个参数

**Each（）**

**//数组里面的每一个，以每一个匹配的元素作为上下文来执行一个函数。**

**Index是索引，ele为当前元素==this 但是需要加$才能使用**

.each(function(index,ele){  
 var y = index \* 25;  
 $(ele).css("background-position","5px -" + y + "px");  
})

$.each(array,function(key,value){})

Array 要遍历的json key对应的key value：key对应的value

数组的话 key对应索引值 value对应 元素

var CITYS = { **"北京"**: ["北京"], **"上海"**: ["上海"], **"天津"**: ["天津", "塘沽"]}

**$**.each(CITYS,function(key, value){  
 var $option = **$**("<option></option>");  
 $option.text(key);//获取当前省的名字  
 **$**("#province").append($option);  
});

全局的

$.each(array, function(index, object){})

普通jQuery对象的each方法

$("li").each(function(index, element){} )

参数的顺序是一致的。

jQuery选择器

|  |  |  |
| --- | --- | --- |
| **基本选择器** | | |
| **符号** | 说明 | 用法 |
| **$(“#demo”)** | 选择id为demo的第一个元素，一般用J\_开头 | $(“#demo”).css(“background”,”red”) |
| **$(“.liItem”)** | 选择所有类名（样式名）为liItem的元素 | $(“.liItem”). css(“background”,”red”); |
| **$(“div”)** | 选择所有标签名字为div的元素 | $(“div”). css(“background”,”red”); |
| **$(“\*”)** | 选择所有元素  少用或配合其他选择器来使用 | $(“\*”). css(“background”,”red”) |
| **$(“.liItem,div”)** | 选择多个指定的元素，这个地方是选择出了 .liItem元素和div元素 | $(“.liItem,div”). css(“background”,”red”) |

规律：$(selector).css(“background”,”red”);

jQuery其他选择器

层级选择器

|  |  |  |
| --- | --- | --- |
|  | | |
| **符号** | 说明 | 用法 |
| **空格** | 后代选择器  选择所有的后代元素 | $(“div span”). css(“background”,”red”); |
| * **a>b也可以**   **a>b>c** | 子代选择器  选择所有的子代元素 | $(“div > span”). css(“background”,”red”) |
| **+** | 紧邻选择器 同级  选择紧挨着的**兄弟**下一个元素 | $(“div + p”). css(“background”,”red”) |
| **~** | 兄弟选择器  选择后面的所有的兄弟元素 | $(“div ~ p”). css(“background”,”red”) |

层级选择器选择了选择符 后面那个元素，比如，div > p，是选择>后面的p元素。

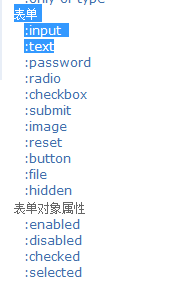
过滤选择器

|  |  |  |
| --- | --- | --- |
| **伪类选择选择器** | | |
| **符号** | 说明 | 用法 |
| **:eq(index)** | index是从0开始的一个数字，选择序号为index的元素。选择第一个匹配的元素。 | $(“li:eq(1)”). css(“background”,”red”) |
| **:gt(index)**  **Greater than** | Index 是从0开始的一个数字，选择序号大于index的元素  下面存在的也会被悬赏  前面最好带上类 | $(“li:gt(2)”). css(“background”,”red”) |
| **:lt(index)**  **Little than** | Index是从0开始的一个数字，选择小于index 的元素 | $(“li:lt(2)”). css(“background”,”red”)  0和1 |
| **:odd** | 选择所有序号为奇数行的元素 | $(“li:odd”). css(“background”,”red”) |
| **:even** | 选择所有序号为偶数的元素 | $(“li:even”). css(“background”,”red”) |
| **:first** | 选择匹配第一个元素 | $(“li:first”). css(“background”,”red”) |
| **:last** | 选择匹配的最后一个元素 | $(“li:last”). css(“background”,”red”) |

|  |  |  |
| --- | --- | --- |
| **属性选择器** | | |
| **符号** | 说明 | 用法 |
| **$(“a[href]”)** | 选择所有包含href属性的元素 | $(“a[href]”). css(“background”,”red”) |
| **$(“a[href=’itcast’]”)** | 选择href属性值为itcast的所有a标签 | $(“a[href=’itcast’]”). css(“background”,”red”) |
| **$(“a[href!=’baidu’]”)** | 选择所有href属性不等baidu的所有元素，包括没有href的元素 | $(“a[href!=’baidu’]”). css(“background”,”red”) |
| **$(“a[href^=’web’]”)** | 选择所有以web开头的元素 | $(“a[href^=’web’]”). css(“background”,”red”) |
| **$(“a[href$=’cn’]”)** | 选择所有以cn结尾的元素 | $(“a[href$=’cn’]”). css(“background”,”red”) |
| **$(“a[href\*=’i’]”)** | 选择所有包含i这个字符的元素，可以是中英文 | $(“a[href\*=’i’]”). css(“background”,”red”) |
| **$(“a[href][title=’我’]”)** | 选择所有符合指定属性规则的元素，都符合才会被选中。 | $(“a[href][title=’我’]”). css(“background”,”red”) |

筛选选择器参考上课讲的 11 筛选选择器.html

表单选择器



筛选选择器

**// .eq(index)**  
// 类比 ":eq(index)" 选择器  
$("span").eq(1).css("font-size","40px");  
**// .first() .last()**// 类比 ":first" 选择器  
$("span").first().css("font-size", "40px");  
/**/ .parent([exp])**  
// 选择父亲元素

* $("p").filter(".intro"); //过滤，选择所有p标签带有 .intro类 $('p.intro')
* $("p").not(".intro"); //去除，跟上面的filetr正好相反

$("#sp").parent().css("background", "gold");  
//里面可以传入一个选择器  
$("p").parent("#div").css("background", "gold");  
  
**// .siblings()**  
// 选择所有的兄弟元素  
$("#li04").siblings().css("background", "gold");  
$("#li04").siblings(".liItem").css("background", "gold");

//  
**.children（[exp]）**  
// .find()// 查找所有后代元素  
$("div").find("#li04").css("background", "gold");

**//.next([exp])紧邻**

下一个，可以传入一个参数

Each

each(function(index,ele){  
 var y = index \* 25;  
 $(ele).css("background-position","5px -" + y + "px");  
})

下拉菜单

$(**document**).ready(function () {  
 alert($(".wrap li").**length**);  
 // mouseenter事件 是 鼠标经过时触发的事件  
 $(".wrap li").**mouseenter**(function () {  
 $(this).children("ul").show();  
 });  
  
 $(".wrap li").**mouseleave**(function () {  
 $(this).children("ul").hide();  
 });  
  
  
// hover方法来做  
 $(".wrap li").hover(function () {  
 //alert("1");  
 $(this).children("ul").show();  
 }, function () {  
 $(this).children("ul").hide();  
 });  
  
// 一个方法的时候  
 $(".wrap li").hover(function () {  
 /\*alert("1");\*/  
 var $this = $(this).children("ul");  
 var isShow = $this.css("display");  
 if(isShow === "block") {  
 $this.hide();  
 } else {  
 $this.show();  
 }  
 });  
// 最简单的  
 $(".wrap li").hover(function () {  
 $(this).children("ul").**slideToggle**();  
 });  
  
 });

DOM对象跟jQuery对象相互转换

jQuery对象转换成DOM对象:

方式一：$(“#btn”)[0]

方式二：$(“#btn”).get(0)

DOM对象转换成jQuery对象：

$(document) -> 把DOM对象转成了jQuery对象

var btn = document.getElementById(“bt n”);

btn -> $(btn);

mouseover事件跟mouseenter事件的区别：

mouseover/mouseout事件，鼠标经过的时候会触发多次，每遇到一个**子元素就会触发一次。**

mouseenter/mouseleave事件，鼠标经过的时候只会触发一次

上午提到的两个面试题：

1. js对象创建的有几种方式（三种）
   1. 1.var obj = {};
   2. 2. var obj = new Object();
   3. 3. ECMAScript5里面提供的一个方法：Object.create(参数);
2. window.onload 跟jQuery的ready函数区别：
   1. 参考 1.4.2

今天的内容，大家有不明白的地方，或者其他问题，可以晚上的时候，总结一下，然后，班长统计下明天咱们集中解答。

动画

**显示**

show([speed,[easing],[fn]])

**speed**:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**easing**:(Optional) 用来指定切换效果，默认是"swing"先慢后快，可用参数"linear" 线性

**fn:**在动画完成时执行的函数，每个元素执行一次。 show(1000,hide(1000))

hide([speed,[easing],[fn]])

slidedDown() 下拉显示

slideUp() 上拉隐藏

slideToggle([speed],[easing],[fn])

fadeIn() 淡入

fadeOut 淡出

fadeToggle([speed,[easing],[fn]])

[fadeTo([[s],o,[e],[fn]])](mk:@MSITStore:E:\小黄片\jQuery帮助文档及源文件\jQuery1.11.0_20140330.chm::/fadeTo.html) o 是0-1之间的小树

自定义动画

[animate(p,[s],[e],[fn])](mk:@MSITStore:E:\小黄片\jQuery帮助文档及源文件\jQuery1.11.0_20140330.chm::/animate.html)1.8\*

**params**:一组包含作为动画属性和终值的样式属性和及其值的集合{json类似}

**speed**:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)

**easing**:要使用的擦除效果的名称(需要插件支持).默认jQuery提供"linear" 和 "swing".

**fn**:在动画完成时执行的函数，每个元素执行一次

**params,options** String,StringV1.0

**params**::一组包含作为动画属性和终值的样式属性和及其值的集合

**options**:动画的额外选项。如：speed - 设置动画的速度,easing - 规定要使用的 easing 函数,callback - 规定动画完成之后要执行的函数,step - 规定动画的每一步完成之后要执行的函数,queue - 布尔值。指示是否在效果队列中放置动画。如果为 false，则动画将立即开始,specialEasing - 来自 styles 参数的一个或多个 CSS 属性的映射，以及它们的对应 easing 函数

[stop([c],[j])](mk:@MSITStore:E:\小黄片\jQuery帮助文档及源文件\jQuery1.11.0_20140330.chm::/stop.html)1.7\*

**[clearQueue],[gotoEnd]** Boolean,BooleanV1.2

**clearQueue**:true 停止当前动画并清空队列 默认为false 不清空队列，只停止当前动画

**gotoEnd**: true 立即执行完该步动画 false 不执行完概不动画，听到哪算哪

**[queue],[clearQueue],[jumpToEnd]**BooleanV1.7

**queue**:用来停止动画的队列名称

**clearQueue**:如果设置成true，则清空队列。可以立即结束动画。

**jumpToEnd**:如果设置成true，则完成队列。可以立即完成动画。

常用的dom操作

动态添加属性

$(this).attr(“key”,”value”);

获取属性

$(this).attr(“key”);

移除属性

$(this).removeAttr(“key”);

值 val

有些表单标签有这个value属性

$(this).val() 获取value

$(this).val(“str”); 设置value

内容 html

$(“div”).html(“str”); str可以是标签可以是别的

获取内容

$(“div”).html();

文本

text([val|fn])

valString V1.0用于设置元素内容的文本

function(index, text)

此函数返回一个字符串。接受两个参数，index为元素在集合中的索引位置，text为原先的text值。

增删类

$(“#id”).addClass(“className”);//没有 .

$(“#id”).removeClass(“className”);//没有 . 不给参数就删除全部class，制定的为不存在的样式不起作用

$(“#id”).hasClass(“className”); . 返回布尔值

$(“#id”).toggleClass(“className”);//切换 没有的加有的删

元素的操作

css基本的用法

$(“#id”).css(“background”); //返回对象对应属性的值

$(“#id”).css(“background”,”red”); //设置对象的值

$(“tagname”).css(“background”,”red”); //所有的对应标签都应用 ，隐式迭代

// 设置样式  
$("li").css("background-color","red");  
  
// 设置多个样式  
$("li").css({  
 **"background-color"**: "red",  
 **"font-size"**: "20px",  
 **"color"**: "white"  
});  
  
// 设置样式，可以根据我们自己的意愿来设置  
$("li").css("font-size", function (i,v) {  
 // index: 指定当前元素的索引号  
 // value: 表示当前元素要设置的样式，此处指：background-color,为已经设置的值，可能为默认  
 /\*console.log(index);  
 console.log(value);\*/  
 //return (i + 1) \* parseInt(v) + "px";  
  
 // 只设置第三个  
 if(i === 2) {  
 return (i + 1) \* parseInt(v) + "px";  
 }  
});

.height([value]) .width([value])

不带value 获取数据 。

带了value为设置数据 ，不用加单位

Css（“heigth”） 获取的是带单位的字符串

。height（） 没单位的数字

并且可以和css一样传入一个函数那种

可以获取document 和window的宽高

.offset()

不带参数获取的是 obj{“top”：10px， “left”:10px}

.offset().top

.offset().left

每单位的数字

设置

.offset({

Top：100，

Left：100 //不带单位并且可以只设置其中一个

}) //设置之后会自动升级为relative

.position()

相对父元素的位置 top left

.scrollTop([val])

被卷起的高度,没参数的为获取，有参数为设置，且不带单位

操作文档

**不是动态创建的元素的话就会被移动了**

Append prepend

**$(this).append(“<p></p>”); 添加到后面 str||node**

$(“<p></p>”).appendTo($(this)); 添加到后面

$(this).prepend(“<p></p>”); 前面添加

$(“<p></p>”).prependTo($(this)); 添加到后面

After before

$(this).after(“node”) node添加到this的后面

$(this).before(“node”) node 添加到this的前面

$(“node”).insertBefore($(this));

$(“<p></p>”).insertAfter($(this)); 添加到后面

Remove html（）

**删除节点**

$(this).remove() 干掉自己

$(this).empty() 干掉自己的所有子标签

$(this).html(“”) 推荐，干掉所有的子节点

Clone

$(this).clone();

$(this).clone(true); 里面的事件也会被复制

Wrap

**包裹节点**

$(this).wrap(“node”) 用node包裹this

$(this).wrapAll(“node”) 用node包裹所有的this ，用一个

Replacewith

$("p").replaceWith("<b>Paragraph. </b>");

事件

Bind（）

**传统方式 单独的写**

$(“#id”).mouseenter(fuction(){})

.click(fuction(){})

**Bind同时绑定多个事件类型/处理程序**

$(“#id”).bind({

“click”:function(){},

“mouseenter”:function(){}

})

**Bind传递数据**

function handler(event) {

alert(event.data.foo);

}

$("p").bind("click", {foo: "bar"}, handler)

**通过返回false来取消默认的行为并阻止事件起泡。**

$("form").bind("submit", function() { return false; })

**通过使用 preventDefault() 方法只取消默认的行为。**

$("form").bind("submit", function(event){

event.preventDefault();

});

**通过使用 stopPropagation() 方法只阻止一个事件起泡。**

$("form").bind("submit", function(event){

event.stopPropagation();

});

one（）

只能执行一次

.delegate()

多个绑定耗时较大，委托给父元素，冒泡机制

delegate(sel,[type],[data],fn)

. on(events,[selector],[data],fn)

记住这个就行了，例如： selector 可以任意层级，绑定到自己就不写selector

$(function(){  
 $("div").on("click","li",function(){  
 alert($(this).index());  
 alert($(this).html());  
 });  
})

.off(events,[selector],[fn])

$(“div”).off(“click”);

可以在on中加上此属性，实现one

**通过返回false来取消默认的行为并阻止事件起泡。**

$("form").bind("submit", function() { return false; })

**通过使用 preventDefault() 方法只取消默认的行为。**

$("form").bind("submit", function(event){

**event.preventDefault();**

});

**通过使用 stopPropagation() 方法只阻止一个事件起泡。**

$("form").bind("submit", function(event){

**event.stopPropagation();**

});

触发trigger 可以出发多个

$(this).trigger(“focus”);

$(this).triggerHander(“focus”);触 发事件但是浏览器默认操作没有执行

$(function(){  
 $("#old").click(function(){  
 $("input").trigger("focus");//input获得焦点并执行了获得焦点函数  
 });   
 $("#new").click(function(){ //input没获得焦点单执行了获得焦点函数  
 $("input").triggerHandler("focus");  
 });  
 $("input").**focus**(function(){  
 $("<span>Focused!</span>").**appendTo**("body").**fadeOut**(2000);  
 });  
 });  
</script>

事件源

on(events,[selector],[data],fn) 例：鼠标跟随

$(**document**).on("mousemove",function(e){  
 $("img").css({  
 **"position"**:"absolute",  
 **"top"**: e.**pageY**,  
 **"left"**:e.**pageX** })  
})

用一个e来记录事件源

常见功能实现

弹出的广告

$(function(){  
 $(".ad").show(1000);  
 $(".ad span").click(function(){  
 $(this).parent(".ad").hide(1000);  
 })  
})

下拉菜单

$(".wrap li").hover(function () {  
 $(this).children("ul").stop().**slideToggle**();  
});

Tab栏切换

$(function(){  
 $(".tab>li").**mouseenter**(function(){  
 $(this).addClass("active").siblings().removeClass("active");  
// 获取当前被点击的对象的index，index（）  
 var index=$(this).index();  
 $(".products .main")  
 .eq(index).addClass("selected")  
 .siblings().removeClass("selected");  
 });  
 })

360图片隐藏和显示标题

<script type="text/javascript">  
 $(function(){  
 var div = $("<div></div>"),speed = 388,$p = $(".wrapper p");  
 $p.before(div);  
 $(".wrapper div").fadeTo(0,0.5);  
 $p.each(function(index,ele){  
 var y = index \* 25;  
 $(ele).css("background-position","5px -" + y + "px");  
 });  
  
 $(".wrapper li").hover(function(){  
 $(this).children("div,p").stop().animate({**"bottom"**:0},speed);  
 },function(){  
 $(this).children("div,p").stop().animate({**"bottom"**:-26},speed);  
 });  
 });  
</script>

<script>  
 var json={  
 **'01.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'02.JPG'**:['www.sina.com','新郎微博'],  
 **'03.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'04.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'05.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'06.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'07.JPG'**:['www.baidu.com','百度一下你就知道'],  
 **'08.JPG'**:['www.baidu.com','百度一下你就知道'],  
 }  
 $(function(){  
 //动态创建360的图片和内容  
 for(var index in json){  
 var html="";  
 html+="<li>";  
 html+= "<a href='"+json[index][0]+"'>";  
 html+= "<img src='images/"+index+"' alt='"+json[index][1]+"'/>"  
 html+= "<p>"+json[index][1]+"</p>"  
 html+= "</a>"  
 html+="</li>";  
 $(".wrapper ul").append(html);  
 }

//给每个元素添加不同的属性的时候用each（function（i，v））  
 $(".wrapper p").each(function(index,ele){  
 $(ele).css("background-position","5px -"+index\*25+"px")  
 })

//这里没能正确的定位到元素是我的失策  
 $(".wrapper li").hover(function(){  
 $(this).find("p").css("bottom","0");  
 },function(){  
 $(this).find("p").css("bottom","-25px");  
 })

固定导航栏

$(function () {  
 var H = $(".top").height();  
 $(**window**).scroll(function () {  
 var docSccrollTop = $(**document**).**scrollTop**();  
 if(docSccrollTop > H){  
 $(".nav").css({**"position"**:"fixed",**"top"**:0});  
 // 此时 nav的位置固定，如果不设置 main部分的margin-top的话，将有一部分内容被挡住 nav的高度+开始设置的20  
 $(".main").css("margin-top",106);  
 }else{  
 $(".nav").css({**"position"**:"static"}); /\*静态定位\*/  
 $(".main").css("margin-top",20);  
 }  
 });  
});

侧边栏随动广告

$(function(){  
 $(**window**).scroll(function () {  
 var scrollTop = $(**document**).**scrollTop**();  
  
 $(".left,.right").stop().animate({**"top"** : scrollTop + 50},600);  
 });  
});

动态创建表格

$(function(){  
 var data = [{  
 **name**: "传智播客",  
 **url**: "http://www.itcast.cn",  
 **type**: "IT最强培训机构"  
 },{  
 **name**: "黑马程序员",  
 **url**: "http://www.itheima.com",  
 **type**: "大学生IT培训机构"  
 },{  
 **name**: "传智前端学院",  
 **url**: "http://web.itcast.cn",  
 **type**: "前端的黄埔军校"  
 }];  
 var inHtml="";  
 /\*for(var temp in data){  
 inHtml+="<tr>";  
 inHtml+= "<td>"+data[temp].name+"</td>";  
 inHtml+= "<td>"+data[temp].url+"</td>";  
 inHtml+= "<td>"+data[temp].type+"</td>";  
 inHtml+="</tr>";  
 }\*/  
 //注意 for(var temp in data){} data为数组的时候temp为序列号。获取对象为data[tmp]  
 // data为对象的时候，k为属性名， 属性值为data[temp]  
  
 for(var i = 0;i < data.**length**;i++ ) {  
 inHtml+="<tr>";  
 inHtml+= "<td>"+data[i].**name**+"</td>";  
 inHtml+= "<td>"+data[i].**url**+"</td>";  
 inHtml+= "<td>"+data[i].**type**+"</td>";  
 inHtml+="</tr>";  
 }  
 $("#J\_tbody").html(inHtml);  
  
});

选择省市

var CITYS = { **"北京"**: ["北京"], **"上海"**: ["上海"], **"天津"**: ["天津", "塘沽"]}

**$**(function(){  
 //初始化 所有的省的数据  
 //.each(arr,funtion(key,value){})  
 **$**.each(CITYS,function(key, value){  
 var $option = **$**("<option></option>");  
 $option.text(key);//获取当前省的名字  
 **$**("#province").append($option);  
 });  
  
 //当省的数据发生改变的时候执行 加载市的数据  
 **$**("#province").on("change",function(){  
 //拿到当前省的名字  
 var pro = **$**("#province option:selected").text();  
 //清空原有的city数据  
 **$**("#city").empty()  
 //根据省名，拿到所有的城 市，然后根据城市创建option添加到cityselect中  
 **$**.each(CITYS[pro],function(i, value){  
 **$**("#city").append("<option>"+ value +"</option>");  
 });  
 });  
  
 //上来后初始化一下城市的数据  
 **$**("#province").triggerHandler('change');

### H5

常见标签

Datalist

不常用

Progress

<!--默认max为1 ，但是可以设置，这样的话就可以有value/max得到显示百分比-->  
 <progress value="0.8"></progress>   
 <progress value="40" max="100"></progress>

链接关系

Rel=”noflow”

Html5 标签不支持的话需要在js部分声明一下,或者html5-shiv

Document.creatElement(“header”);

Itemprop=”人” 抓取重要信息的关键字,没推广

Aria 属性

无障碍应用

自定义属性

data-\*

this.data

data-\*属性

动态创建表格

1 <script>

2 // 键是ID 值是信息

3 var data = {

4 01: {

5 name: "张三1",

6 age: 18

7 },

8 02: {

9 name: "张三2",

10 age: 19

11 },

12 03: {

13 name: "张三3",

14 age: 20

15 }

16 };

17 // jQuery操作一定要做变量本地化

18 var list = document.getElementById('list');

19 for (var id in data) {

20 var item = data[id];

21 var liElement = document.createElement('li');

22 // liElement.innerHTML = item.name;

23 liElement.appendChild(document.createTextNode(item.name));

24 liElement.setAttribute('data-age', item.age);

25 liElement.setAttribute('data-id', item.id);

26 list.appendChild(liElement); // 变量本地化

27 // 此处才将元素加到界面上

28 liElement.addEventListener('click', function() {

29 // alert(this.name);

30 // this 是当前点击的元素

31 // alert(this.getAttribute('data-age'));

32 console.log(this.dataset['age']);

33 });

34 }

35

Tab标签

<script>

(function() {

// 写这个是为了有一个单独作用域，避免污染

// api 是应用程序编程接口

var $lis = $('.tabs>ul>li');

$lis.on('click', function() {

// 获取目标对象的选择器

var targetSelector = $(this).data('target');

var $target = $(targetSelector);

//

});

})();

</script>

Sublime使用

P+ tab 自动补全

Ctrl+shift+d复制向下

Ctrl+shift+k 删除当前

C a n

Ctrl+shift+上 向上移动当前行

Li\*10 tab 产生10行

点击鼠标中键 多行编辑

Ctrl +左键 选中多行进行编辑 end 到行尾部 home 移动到行头

Ctrl + 滚动 放大缩小

分屏 查看

Ctrl + d 选择下一个相同的元素

Ctrl+k 跳过下一个相同的元素

C s p 控制台

智能表单

<form action="#" method="">

<input type="text">

<input type="image" src="toy.png"> <!-- 大于submit，会传送点击的坐标也 -->

<input type="submit">

<!-- file text password radio checkbox button image submit reset select button -->

<!-- range可以拉动的条 有最大max最小值min和当前的value， step使拉动不流畅，跳动拉 -->

<input type="range" min="0" max="10" step="5" value="5">

<!-- 多用于手机端 --> <!-- input email date tel number 键盘弹出不同 -->

<input type="text" required> <!-- 移动端用 -->

<!-- email用于 手机端 -->

<input type="email" value="some@email.com">

<input type="date" min="2010-08-14" max="2011-08-14" value="2010-08-14">

<input type="range" min="0" max="50" value="10">

<input type="search" results="10" placeholder="Search...">

<!-- pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.\*?$"正则表达式 -->

<input type="tel" placeholder="(555) 555-5555" pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.\*?$">

<input type="color" placeholder="e.g. #bbbbbb">

Html 多媒体

多媒体值得是 音效 音乐 视频 动画

**视频格式**

.avi window常用，非win不好用

.wmv . windows常用，非win不能用

.mpg .mpeg 跨平台支持所有浏览器

.mov ios常用，win要装插件

.ram .rm **低宽带**优选，质量差

.swf .flv flash 插件

**.mp4**

**声音格式**

.mid .midi 小巧的数字方法的，广泛支持

Rm ram 低宽带

.wav 除了谷歌都支持,**最受欢迎，**无压缩的音乐和演讲用

.mp3 .mpga 通那个用，**压缩**录制音乐格式

Html对象

**<object>作用是支持html助手（插件或者辅助应用程序,连入）,**

**Quicktime播放 wave，播放mp4**

<object width="420" height="360"

classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"

codebase="http://www.apple.com/qtactivex/qtplugin.cab">

<param name="src" **value="bird.wav"** />

<param name="controller" value="true" />

</object>

**Flash播放 .swf**

<object width="400" height="40"

classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/

pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">

<param name="src" **value="bookmark.swf"**>

<embed **src="bookmark.swf"** width="400" height="40"></embed>

</object>

**Window Media Player 播放WMV 影片**

<object width="100%" height="100%"

type="video/x-ms-asf" url=**"3d.wmv"** data=**"3d.wmv"**

classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">

<param name="url" value="3d.wmv">

<param name="filename" value="3d.wmv">

<param name="autostart" value="1">

<param name="uiMode" value="full" />

<param name="autosize" value="1">

<param name="playcount" value="1">

<embed type="application/x-mplayer2" src="3d.wmv" width="100%"

height="100%" autostart="true" showcontrols="true"

pluginspage="http://www.microsoft.com/Windows/MediaPlayer/"></embed>

</object>

HRML 音频

**<!DOCTYPE html> 一定开头声明要使用**

使用 <embed> 元素，定义外部（非 HTML）内容的容器。

<**embed** height="100" width="100" **src="song.mp3"** />

<Object > 元素

<object height="100" width="100" **data="song.mp3**"></object>

使用<audio>元素 ,只支持3种列表 MP3 ogg wav

<audio controls="controls">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<source src="song.wave" type="audio/wav" />

</audio>

<audio> 标签的属性

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| [autoplay](http://www.w3school.com.cn/tags/att_audio_autoplay.asp) | autoplay | 如果出现该属性，则音频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/tags/att_audio_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [loop](http://www.w3school.com.cn/tags/att_audio_loop.asp) | loop | 如果出现该属性，则每当音频结束时重新开始播放。 |
| [preload](http://www.w3school.com.cn/tags/att_audio_preload.asp) | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/tags/att_audio_src.asp) | *url* | 要播放的音频的 URL。 |

最好的html解决方法

audio embed

MP3 ogg wav

<audio controls="controls" height="100" width="100">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<embed height="100" width="100" src="song.mp3" />

</audio>

2.雅虎媒体播放器

直接链接就能播放但是要保证script标签内的内容存在

<a href="song.mp3">Play Sound</a>

下面是要保证的标签

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js">

</script>

3.使用超链接（播放器会自动用插件播放）

<a href="song.mp3">Play the sound</a>

Html视频

**<!DOCTYPE html>**

**<embed> 标签**

<embed src="movie.swf" height="200" width="200"/>

**<object>标签**

<object width="320" height="240" data="/i/bookmark.swf"></object>

**<video>标签** Vadio支持ogg，mp4,webm 三种格式

<video width="320" height="240" controls="controls" autoplay="autoplay">

<source src="/i/movie.ogg" type="video/ogg" />

<source src="/i/movie.mp4" type="video/mp4" />

<source src="/i/movie.webm" type="video/webm" />

</video>

<video> 与 </video> 之间插入的内容非<source>是供不支持 video 元素的浏览器显示的

兼容性

Vadio的属性

**poster=“a.jpg” 没加载的时候出现的图片**

|  |  |  |  |
| --- | --- | --- | --- |
| [autoplay](http://www.w3school.com.cn/tags/att_video_autoplay.asp) | autoplay | 如果出现该属性，则视频在就绪后马上播放。 | |
| [controls](http://www.w3school.com.cn/tags/att_video_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 | |
| [height](http://www.w3school.com.cn/tags/att_video_height.asp) | *pixels* | 设置视频播放器的高度。 | |
| [loop](http://www.w3school.com.cn/tags/att_video_loop.asp) | loop | 如果出现该属性，则当媒介文件完成播放后再次开始播放。 | |
| [preload](http://www.w3school.com.cn/tags/att_video_preload.asp) | preload | 没点击播放的时候自动加载  如果使用 "autoplay"，则忽略该属性。 | |
| [src](http://www.w3school.com.cn/tags/att_video_src.asp) | *url* | 要播放的视频的 URL。 | |
| [width](http://www.w3school.com.cn/tags/att_video_width.asp) | *pixels* | 设置视频播放器的宽度。 | |
| **方法** | | **属性** | **事件** |
| play() | | currentSrc | play |
| pause() | | currentTime | pause |
| load() | | videoWidth | progress |
| canPlayType | | videoHeight | error |
|  | | duration | timeupdate |
|  | | ended | ended |
|  | | error | abort |
|  | | paused | empty |
|  | | muted | emptied |
|  | | seeking | waiting |
|  | | volume | loadedmetadata |
|  | | height |  |
|  | | width |  |

**注释：**在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

最好的html解决方法

1.组合框架

<**video** width="320" height="240" controls="controls">

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

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

<**source** **src**="movie.webm" type="video/webm" />

<**object** **data**="movie.mp4" width="320" height="240">

<**embed** **src**="movie.swf" width="320" height="240" />

</object>

</video

2.优酷解决方案

<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v.swf"

width="480" height="400"

type="application/x-shockwave-flash">

</embed>

3html5midia组件

Svg

矢量图 svg 体积小质量高 可控图标

我们

**<iframe src="demo.svg" frameborder="0"></iframe>**

**<object data="demo.svg" type=""></object>**

**<embed src="demo.svg" type="">**

<script>

window.addEventListener('load', function() {

var svgs = document.getElementsByTagName('svg');

for (var i = 0; i < svgs.length; i++) {

// ???

// console.log(svgs[i].dataset);

var src = $(svgs[i]).data('src');

// 向服务器发送请求 得到svg

$.get(src, function(data) {

var el = data.documentElement;

$(document.body).append($(el));

});

}

});

</script>

H5 api

qureySelector

JS多了一个原始支持，类似jqueryDOM选择器

document.getElementById() 需要给DOM元素设置ID  
document.querySelector(selector) 可以通过CSS选择器的语法找到DOM元素

document.getElementsByTagName()  
document.querySelectorAll(‘.item’)

$(‘.item’).on

**document.querySelector(selector); qs**

返回第一个满足选择器条件的元素第一个dom对象

**document.querySelectorAll(‘.item’); qsa**

返回所有满足该条件的元素 一个元素类型是dom类型的数组

**addeventListener()**

**DOM数组的每一个成员注册事件不能像jquery一样直接注册， 必须分别给每个元素注册**

**h5就是将我们经常需要的操作又包装一层**

元素.classList

新H5中DOM对象多了一个classList属性，是一个数组

add 添加一个新的类名

remove 删除一个的类名

contains 判断是否包含一个指定的类名

toggle 切换一个class element.toggle(‘class-name’,[add\_or\_remove])

toggle函数的第二个参数true为添加 false删除

**lis[i].classList.remove**('active');

**this.classList.add('active');**

var exist = this.classList.contains('active');

// toggle函数的第二个参数true为添加 false删除

**this.classList.toggle('active', !exist);**

**e.preventDefault();**

标签栏

<script>

// 1. 获取所有LI元素

var lis = document.querySelectorAll('.nav > li');

// 2. 遍历 给每个元素注册事件 click

for (var i = 0; i < lis.length; i++) {

var item = lis[i];

item.addEventListener('mouseover', function(e) {

// 3. 重置

for (var i = 0; i < lis.length; i++) {

**lis[i].classList.remove**('active');

}

// 4. 事件执行中我们将当期触发事件的元素active

**this.classList.add('active');**

//阻止浏览器操作

**e.preventDefault();**

});

}

var link = document.querySelector('.navbar-brand');

link.addEventListener('click', function(e) {

// 判断当前ClassList中有没有active

var exist = this.classList.contains('active');

// toggle函数的第二个参数true为添加 false删除

this.classList.toggle('active', !exist);

e.preventDefault();

});

2.1.3 访问历史 API

界面上的所有JS操作不会被浏览器记住，就无法回到之前的状态  
在HTML5中可以通过window.history操作访问历史状态，让一个页面可以有多个历史状态

window.history.forward(); // 前进

window.history.back(); // 后退

window.history.go([url]); // 刷新 不给参数是刷新本页面，给了就跳转了

直接使用就行了

通过JS可以加入一个访问状态

history.pushState(放入历史中的状态数据, 设置title(现在浏览器不支持)， 改变历史状态)

### 2.1.4 全屏 API

JavaScript中可以通过调用requestFullScreen()方式实现指定元素的全屏显示  
var element = document.querySelector(‘…’);  
element.requestFullScreen();

# H5

常见标签

Datalist

不常用

Progress

<!--默认max为1 ，但是可以设置，这样的话就可以有value/max得到显示百分比-->  
 <progress value="0.8"></progress>   
 <progress value="40" max="100"></progress>

链接关系

Rel=”noflow”

Html5 标签不支持的话需要在js部分声明一下,或者html5-shiv

Document.creatElement(“header”);

Itemprop=”人” 抓取重要信息的关键字,没推广

Aria 属性

无障碍应用

自定义属性

data-\*

this.data

data-\*属性

动态创建表格

1 <script>

2 // 键是ID 值是信息

3 var data = {

4 01: {

5 name: "张三1",

6 age: 18

7 },

8 02: {

9 name: "张三2",

10 age: 19

11 },

12 03: {

13 name: "张三3",

14 age: 20

15 }

16 };

17 // jQuery操作一定要做变量本地化

18 var list = document.getElementById('list');

19 for (var id in data) {

20 var item = data[id];

21 var liElement = document.createElement('li');

22 // liElement.innerHTML = item.name;

23 liElement.appendChild(document.createTextNode(item.name));

24 liElement.setAttribute('data-age', item.age);

25 liElement.setAttribute('data-id', item.id);

26 list.appendChild(liElement); // 变量本地化

27 // 此处才将元素加到界面上

28 liElement.addEventListener('click', function() {

29 // alert(this.name);

30 // this 是当前点击的元素

31 // alert(this.getAttribute('data-age'));

32 console.log(this.dataset['age']);

33 });

34 }

35

Tab标签

<script>

(function() {

// 写这个是为了有一个单独作用域，避免污染

// api 是应用程序编程接口

var $lis = $('.tabs>ul>li');

$lis.on('click', function() {

// 获取目标对象的选择器

var targetSelector = $(this).data('target');

var $target = $(targetSelector);

//

});

})();

</script>

Sublime使用

P+ tab 自动补全

Ctrl+shift+d复制向下

Ctrl+shift+k 删除当前

C a n

Ctrl+shift+上 向上移动当前行

Li\*10 tab 产生10行

点击鼠标中键 多行编辑

Ctrl +左键 选中多行进行编辑 end 到行尾部 home 移动到行头

Ctrl + 滚动 放大缩小

分屏 查看

Ctrl + d 选择下一个相同的元素

Ctrl+k 跳过下一个相同的元素

C s p 控制台

智能表单

<form action="#" method="">

<input type="text">

<input type="image" src="toy.png"> <!-- 大于submit，会传送点击的坐标也 -->

<input type="submit">

<!-- file text password radio checkbox button image submit reset select button -->

<!-- range可以拉动的条 有最大max最小值min和当前的value， step使拉动不流畅，跳动拉 -->

<input type="range" min="0" max="10" step="5" value="5">

<!-- 多用于手机端 --> <!-- input email date tel number 键盘弹出不同 -->

<input type="text" required> <!-- 移动端用 -->

<!-- email用于 手机端 -->

<input type="email" value="some@email.com">

<input type="date" min="2010-08-14" max="2011-08-14" value="2010-08-14">

<input type="range" min="0" max="50" value="10">

<input type="search" results="10" placeholder="Search...">

<!-- pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.\*?$"正则表达式 -->

<input type="tel" placeholder="(555) 555-5555" pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.\*?$">

<input type="color" placeholder="e.g. #bbbbbb">

Html 多媒体

多媒体值得是 音效 音乐 视频 动画

**视频格式**

.avi window常用，非win不好用

.wmv . windows常用，非win不能用

.mpg .mpeg 跨平台支持所有浏览器

.mov ios常用，win要装插件

.ram .rm **低宽带**优选，质量差

.swf .flv flash 插件

**.mp4**

**声音格式**

.mid .midi 小巧的数字方法的，广泛支持

Rm ram 低宽带

.wav 除了谷歌都支持,**最受欢迎，**无压缩的音乐和演讲用

.mp3 .mpga 通那个用，**压缩**录制音乐格式

Html对象

**<object>作用是支持html助手（插件或者辅助应用程序,连入）,**

**Quicktime播放 wave，播放mp4**

<object width="420" height="360"

classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"

codebase="http://www.apple.com/qtactivex/qtplugin.cab">

<param name="src" **value="bird.wav"** />

<param name="controller" value="true" />

</object>

**Flash播放 .swf**

<object width="400" height="40"

classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase="http://fpdownload.macromedia.com/

pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">

<param name="src" **value="bookmark.swf"**>

<embed **src="bookmark.swf"** width="400" height="40"></embed>

</object>

**Window Media Player 播放WMV 影片**

<object width="100%" height="100%"

type="video/x-ms-asf" url=**"3d.wmv"** data=**"3d.wmv"**

classid="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6">

<param name="url" value="3d.wmv">

<param name="filename" value="3d.wmv">

<param name="autostart" value="1">

<param name="uiMode" value="full" />

<param name="autosize" value="1">

<param name="playcount" value="1">

<embed type="application/x-mplayer2" src="3d.wmv" width="100%"

height="100%" autostart="true" showcontrols="true"

pluginspage="http://www.microsoft.com/Windows/MediaPlayer/"></embed>

</object>

HRML 音频

**<!DOCTYPE html> 一定开头声明要使用**

使用 <embed> 元素，定义外部（非 HTML）内容的容器。

<**embed** height="100" width="100" **src="song.mp3"** />

<Object > 元素

<object height="100" width="100" **data="song.mp3**"></object>

使用<audio>元素 ,只支持3种列表 MP3 ogg wav

<audio controls="controls">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<source src="song.wave" type="audio/wav" />

</audio>

<audio> 标签的属性

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| [autoplay](http://www.w3school.com.cn/tags/att_audio_autoplay.asp) | autoplay | 如果出现该属性，则音频在就绪后马上播放。 |
| [controls](http://www.w3school.com.cn/tags/att_audio_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 |
| [loop](http://www.w3school.com.cn/tags/att_audio_loop.asp) | loop | 如果出现该属性，则每当音频结束时重新开始播放。 |
| [preload](http://www.w3school.com.cn/tags/att_audio_preload.asp) | preload | 如果出现该属性，则音频在页面加载时进行加载，并预备播放。  如果使用 "autoplay"，则忽略该属性。 |
| [src](http://www.w3school.com.cn/tags/att_audio_src.asp) | *url* | 要播放的音频的 URL。 |

最好的html解决方法

audio embed

MP3 ogg wav

<audio controls="controls" height="100" width="100">

<source src="song.mp3" type="audio/mp3" />

<source src="song.ogg" type="audio/ogg" />

<embed height="100" width="100" src="song.mp3" />

</audio>

2.雅虎媒体播放器

直接链接就能播放但是要保证script标签内的内容存在

<a href="song.mp3">Play Sound</a>

下面是要保证的标签

<script type="text/javascript" src="http://mediaplayer.yahoo.com/js">

</script>

3.使用超链接（播放器会自动用插件播放）

<a href="song.mp3">Play the sound</a>

Html视频

**<!DOCTYPE html>**

**<embed> 标签**

<embed src="movie.swf" height="200" width="200"/>

**<object>标签**

<object width="320" height="240" data="/i/bookmark.swf"></object>

**<video>标签** Vadio支持ogg，mp4,webm 三种格式

<video width="320" height="240" controls="controls" autoplay="autoplay">

<source src="/i/movie.ogg" type="video/ogg" />

<source src="/i/movie.mp4" type="video/mp4" />

<source src="/i/movie.webm" type="video/webm" />

</video>

<video> 与 </video> 之间插入的内容非<source>是供不支持 video 元素的浏览器显示的

兼容性

Vadio的属性

**poster=“a.jpg” 没加载的时候出现的图片**

|  |  |  |  |
| --- | --- | --- | --- |
| [autoplay](http://www.w3school.com.cn/tags/att_video_autoplay.asp) | autoplay | 如果出现该属性，则视频在就绪后马上播放。 | |
| [controls](http://www.w3school.com.cn/tags/att_video_controls.asp) | controls | 如果出现该属性，则向用户显示控件，比如播放按钮。 | |
| [height](http://www.w3school.com.cn/tags/att_video_height.asp) | *pixels* | 设置视频播放器的高度。 | |
| [loop](http://www.w3school.com.cn/tags/att_video_loop.asp) | loop | 如果出现该属性，则当媒介文件完成播放后再次开始播放。 | |
| [preload](http://www.w3school.com.cn/tags/att_video_preload.asp) | preload | 没点击播放的时候自动加载  如果使用 "autoplay"，则忽略该属性。 | |
| [src](http://www.w3school.com.cn/tags/att_video_src.asp) | *url* | 要播放的视频的 URL。 | |
| [width](http://www.w3school.com.cn/tags/att_video_width.asp) | *pixels* | 设置视频播放器的宽度。 | |
| **方法** | | **属性** | **事件** |
| play() | | currentSrc | play |
| pause() | | currentTime | pause |
| load() | | videoWidth | progress |
| canPlayType | | videoHeight | error |
|  | | duration | timeupdate |
|  | | ended | ended |
|  | | error | abort |
|  | | paused | empty |
|  | | muted | emptied |
|  | | seeking | waiting |
|  | | volume | loadedmetadata |
|  | | height |  |
|  | | width |  |

**注释：**在所有属性中，只有 videoWidth 和 videoHeight 属性是立即可用的。在视频的元数据已加载后，其他属性才可用。

最好的html解决方法

1.组合框架

<**video** width="320" height="240" controls="controls">

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

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

<**source** **src**="movie.webm" type="video/webm" />

<**object** **data**="movie.mp4" width="320" height="240">

<**embed** **src**="movie.swf" width="320" height="240" />

</object>

</video

2.优酷解决方案

<embed src="http://player.youku.com/player.php/sid/XMzI2NTc4NTMy/v.swf"

width="480" height="400"

type="application/x-shockwave-flash">

</embed>

3html5midia组件

Svg

矢量图 svg 体积小质量高 可控图标

我们

**<iframe src="demo.svg" frameborder="0"></iframe>**

**<object data="demo.svg" type=""></object>**

**<embed src="demo.svg" type="">**

<script>

window.addEventListener('load', function() {

var svgs = document.getElementsByTagName('svg');

for (var i = 0; i < svgs.length; i++) {

// ???

// console.log(svgs[i].dataset);

var src = $(svgs[i]).data('src');

// 向服务器发送请求 得到svg

$.get(src, function(data) {

var el = data.documentElement;

$(document.body).append($(el));

});

}

});

</script>

H5 api

qureySelector

JS多了一个原始支持，类似jqueryDOM选择器

document.getElementById() 需要给DOM元素设置ID  
document.querySelector(selector) 可以通过CSS选择器的语法找到DOM元素

document.getElementsByTagName()  
document.querySelectorAll(‘.item’)

$(‘.item’).on

**document.querySelector(selector); qs**

返回第一个满足选择器条件的元素第一个dom对象

**document.querySelectorAll(‘.item’); qsa**

返回所有满足该条件的元素 一个元素类型是dom类型的数组

**addeventListener()**

**DOM数组的每一个成员注册事件不能像jquery一样直接注册， 必须分别给每个元素注册**

**h5就是将我们经常需要的操作又包装一层**

元素.classList

新H5中DOM对象多了一个classList属性，是一个数组

add 添加一个新的类名

remove 删除一个的类名

contains 判断是否包含一个指定的类名

toggle 切换一个class element.toggle(‘class-name’,[add\_or\_remove])

toggle函数的第二个参数true为添加 false删除

**lis[i].classList.remove**('active');

**this.classList.add('active');**

var exist = this.classList.contains('active');

// toggle函数的第二个参数true为添加 false删除

**this.classList.toggle('active', !exist);**

**e.preventDefault();**

标签栏

<script>

// 1. 获取所有LI元素

var lis = document.querySelectorAll('.nav > li');

// 2. 遍历 给每个元素注册事件 click

for (var i = 0; i < lis.length; i++) {

var item = lis[i];

item.addEventListener('mouseover', function(e) {

// 3. 重置

for (var i = 0; i < lis.length; i++) {

**lis[i].classList.remove**('active');

}

// 4. 事件执行中我们将当期触发事件的元素active

**this.classList.add('active');**

//阻止浏览器操作

**e.preventDefault();**

});

}

var link = document.querySelector('.navbar-brand');

link.addEventListener('click', function(e) {

// 判断当前ClassList中有没有active

var exist = this.classList.contains('active');

// toggle函数的第二个参数true为添加 false删除

this.classList.toggle('active', !exist);

e.preventDefault();

});

访问历史 API js操作无法被浏览器记住

界面上的所有JS操作不会被浏览器记住，就无法回到之前的状态  
在HTML5中可以通过window.history操作访问历史状态，让一个页面可以有多个历史状态

window.history.forward(); // 前进

window.history.back(); // 后退

window.history.go([url]); // 刷新 不给参数是刷新本页面，给了就跳转了

直接使用就行了

通过JS可以加入一个访问状态

history.pushState(state, 设置title(现在浏览器不支持)， 改变历史状态显示在地址栏的)

**加入历史中**

**if (window.history && history.pushState) {**

// 添加一个新的历史记录

**history.pushState(title, 'title没有任何浏览器支持', '?t=' + title)**;

} else {

console.log('不支持');

}

**前进后退内容变化**

// 当我们在伪造的访问历史中前进或后退时会执行一个popstate事件

window.addEventListener('popstate', function(e) {

content.innerHTML = **data[e.state];**

});

**地址作用**

单页面地址作用

// window.location = "https://www.baidu.com";

// 第一次请求过来 获取地址栏中的t参数

// window.location可以拿到当前网页中跟地址相关的信息

**var search = window.location.search;** // ?t=jkaljdksfla

// 如果地址栏中的地址有中文，会以URL编码方式呈现

// decodeURI 可以转换到之前中文

**var title = search.split('=')[1]; // ['?t','jkaljdksfla']**

**if (title) {**

**// 有值 decodeURI作用就是从URL编码转换到之前的状态**

**console.log(decodeURI(title));**

**content.innerHTML = data[decodeURI(title)];**

**}**

}

全屏 requestFullScreen()

JavaScript中可以通过调用requestFullScreen()方式实现指定元素的全屏显示  
var element = document.querySelector(‘…’);  
element.requestFullScreen();

网页全屏需要每个组件都设置背景且谷歌需要加上wibkid前缀

var img = document.querySelector('#img\_full');

img.addEventListener('click', function(e) {

if (document.body.webkitRequestFullScreen)

document.body.webkitRequestFullScreen();

else

console.log('不支持');

e.preventDefault();

});

模拟不同网络访问网页

F12 》network>nothrotlling

Application cacha

<html lang="en" manifest="cache.manifest"> //里面写上路径

创建 cache.manifest 文件 记得换行和tab键

CACHE MANIFEST

# version 1.0.1

CACHE:

index.html

script.js

style.css

NETWORK:

\*

Cache 下面为需要缓存的文件，当需要更新缓存的时候 增大版本就可以了

Webstore

F12>application>storage>localstorage

书签或者本地化

btnGet.addEventListener('click', function() {

// txtValue.value = sessionStorage.getItem('key1');

if (window.sessionStorage)

txtValue.value = sessionStorage.getItem('key1')

// txtValue.value = sessionStorage['key1'];

});

btnSet.addEventListener('click', function() {

// sessionStorage.setItem('key1', txtValue.value);

sessionStorage['key1'] = txtValue.value;

});

**Sessionstorage**

**取出**

if (window.sessionStorage)

**txtValue.value = sessionStorage.getItem('key1');**

**// txtValue.value = sessionStorage['key1'];**

**存储**

// **sessionStorage.setItem('key1', txtValue.value);**

**sessionStorage['key1'] = txtValue.value;**

**localstorage 和上面相同**

file

对于表单里的input可以通过name属性找到标签

1.先创建file空间

**<input id="input\_file" class="form-control hidden" type="file" name="input\_file" multiple>**

Multiple 可以选择多个文件 hidden隐藏该空间

2.太丑，隐藏了 重新创建一个按钮，通过该按钮触发其他按钮

var btnSelect = document.querySelector('#btn\_select');

btnSelect.addEventListener('click', function() {

// 在按钮点击时调用input的点击

inputFile.click();

});

3.显示上传文件的信息，change事件，name time storage

inputFile.addEventListener('change', function() {

var files = **inputFile.files;**

for (var i = 0; i < files.length; i++) {

var li = document.createElement('li');

li.setAttribute('class', 'list-group-item');

// 创建信息的子节点

li.**innerHTML** = '<h5 class="list-group-item-heading">' + **files[i].name** + '</h5><p class="list-group-item-text">' + **files[i].lastModifiedDate.toLocaleDateString()** + ' ' + **files[i].lastModifiedDate.toLocaleTimeString()** + ' ' + **(files[i].size / 1024).toFixed(2)** + 'KB</p>';

fileList.appendChild(li);

}

});

Drag&drop

**e.datatransfer.getdata(“text/plain”);**

<script>

(function() {

// 找到目标位置框框

var target = document.querySelector('#target');

var fileList = document.querySelector('#result');

// 注册拖拽进入

target.addEventListener('dragenter', function() {

this.classList.add('actived');

});

// 离开

target.addEventListener('dragleave', function() {

this.classList.remove('actived');

});

// 如果想要捕获drop事件 就一定得在该事件中阻止默认事件

target.addEventListener('dragover', function(e) {

e.preventDefault();

e.stopPropagation();

e.stopPropagation();

});

// 当元素放到该对象上

target.addEventListener('drop', function(e) {

if (e.dataTransfer.files.length) {

var files = e.dataTransfer.files;

for (var i = 0; i < files.length; i++) {

var li = document.createElement('li');

li.setAttribute('class', 'list-group-item');

// 创建信息的子节点

li.innerHTML = '<h5 class="list-group-item-heading">' + files[i].name + '</h5><p class="list-group-item-text">' + files[i].lastModifiedDate.toLocaleDateString() + ' ' + files[i].lastModifiedDate.toLocaleTimeString() + ' ' + (files[i].size / 1024).toFixed(2) + 'KB</p>';

fileList.appendChild(li);

}

} else {

// 短路运算

// var data = e.dataTransfer.getData('text/plain') || e.dataTransfer.getData('text/uri-list');

var data = e.dataTransfer.getData('text/plain');

if (data) {

// 拖入的是文本

target.innerHTML = data;

} else {

var img = document.createElement('img');

img.src = e.dataTransfer.getData('text/uri-list');

target.appendChild(img);

}

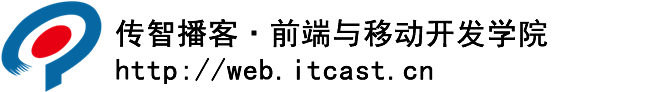
}

this.classList.remove('actived');

e.preventDefault();

e.stopPropagation();

});



**CSS3**

目录 css3

[目录 2](file:///E:\xxoo\h5和之后的.docx#_Toc437115385)

[第1章 CSS3简介 4](file:///E:\xxoo\h5和之后的.docx#_Toc437115386)

[1.1 CSS3的现状 4](file:///E:\xxoo\h5和之后的.docx#_Toc437115387)

[1.2 如何对待 4](file:///E:\xxoo\h5和之后的.docx#_Toc437115388)

[第2章 准备工作 4](file:///E:\xxoo\h5和之后的.docx#_Toc437115389)

[2.1 统一环境 4](file:///E:\xxoo\h5和之后的.docx#_Toc437115390)

[2.1 如何使用手册 5](file:///E:\xxoo\h5和之后的.docx#_Toc437115391)

[第3章 基础知识 5](file:///E:\xxoo\h5和之后的.docx#_Toc437115392)

[3.1 选择器 5](file:///E:\xxoo\h5和之后的.docx#_Toc437115393)

[3.1.1 属性选择器 5](file:///E:\xxoo\h5和之后的.docx#_Toc437115394)

[3.1.2 伪类选择器 6](file:///E:\xxoo\h5和之后的.docx#_Toc437115395)

[3.1.3 伪元素选择器 6](file:///E:\xxoo\h5和之后的.docx#_Toc437115396)

[3.2 颜色 6](file:///E:\xxoo\h5和之后的.docx#_Toc437115397)

[3.3 文本 7](file:///E:\xxoo\h5和之后的.docx#_Toc437115398)

[3.4 边框 7](file:///E:\xxoo\h5和之后的.docx#_Toc437115399)

[3.4.1 边框圆角 8](file:///E:\xxoo\h5和之后的.docx#_Toc437115400)

[3.4.2 边框图片 8](file:///E:\xxoo\h5和之后的.docx#_Toc437115401)

[3.4.3 边框阴影 9](file:///E:\xxoo\h5和之后的.docx#_Toc437115402)

[3.5 盒模型 9](file:///E:\xxoo\h5和之后的.docx#_Toc437115403)

[3.6 背景 10](file:///E:\xxoo\h5和之后的.docx#_Toc437115404)

[3.7 渐变 11](file:///E:\xxoo\h5和之后的.docx#_Toc437115405)

[3.8 伸缩布局 11](file:///E:\xxoo\h5和之后的.docx#_Toc437115406)

[3.8.1 新版伸缩布局 11](file:///E:\xxoo\h5和之后的.docx#_Toc437115407)

[3.9 多列布局 12](file:///E:\xxoo\h5和之后的.docx#_Toc437115408)

[3.10 过渡 12](file:///E:\xxoo\h5和之后的.docx#_Toc437115409)

[3.11 动画 13](file:///E:\xxoo\h5和之后的.docx#_Toc437115410)

[3.12 转换 13](file:///E:\xxoo\h5和之后的.docx#_Toc437115411)

[3.13 媒体查询 15](file:///E:\xxoo\h5和之后的.docx#_Toc437115412)

[第4章 Web字体 16](file:///E:\xxoo\h5和之后的.docx#_Toc437115413)

[4.1 字体格式 16](file:///E:\xxoo\h5和之后的.docx#_Toc437115414)

[4.2 字体图标 17](file:///E:\xxoo\h5和之后的.docx#_Toc437115415)

[第5章 兼容性 18](file:///E:\xxoo\h5和之后的.docx#_Toc437115416)

[第6章 高级应用 18](file:///E:\xxoo\h5和之后的.docx#_Toc437115417)

# CSS3简介

如同人类的的进化一样，CSS3是CSS2的“进化”版本，在CSS2基础上，增强或新增了许多特性， 弥补了CSS2的众多不足之处，使得Web开发变得更为高效和便捷。

## CSS3的现状

1、浏览器支持程度差，需要添加私有前缀

2、移动端支持优于PC端

3、不断改进中

4、应用相对广泛

## 如何对待

1、坚持渐进增强原则

2、考虑用户群体

3、遵照产品的方案

4、听Boss的

# 准备工作

## 统一环境

由于CSS3兼容性问题的普遍存在，为了避免因兼容性带来的干扰，我们约定统一的环境，以保证学习的效率，在最后会单独说明兼容性的问题。

1、Chrome浏览器 version 46+

2、Firefox浏览器 firefox 42+

3、PhotoShop CS6（建议）

## 如何使用手册

学会使用工具，可以让我们事半功倍。

[] 表示全部可选项

|| 表示或者

| 表示多选一

？ 表示0个或者1个

\* 表示0个或者多个

{} 表示范围

# 基础知识

## 选择器

CSS3新增了许多灵活查找元素的方法，极大的提高了查找元素的效率和精准度。CSS3选择器与jQuery中所提供的绝大部分选择器兼容。

### 属性选择器

Attr attr=val attr~=val attr|=val attr\*=val attr^=val attr$=val

参考手册

1、E[attr] 表示存在attr属性即可；

2、E[attr=val] 表示属性值完全等于val

3、E[attr~=val] contain val 这个val是一个独立的属性

4、E[attr|=val] contain 独立属性值 要么这个属性值是以“-”分隔的

5、E[attr\*=val] 表示的属性值里包含val字符并且在“任意”位置

6、E[attr^=val] 表示的属性值里包含val字符并且在“开始”位置

7、E[attr$=val] 表示的属性值里包含val字符并且在“结束”位置

### 伪类选择器

E:first-child 自己是当前元素的父级所有子元素的第一个子元素

E:nth-child(n) 第n个子元素，计算方法是E元素的全部兄弟元素 ，从1开始

E:nth-last-child(n) 同E:nth-child(n) 计算顺序相反。

E:only-child 自己是唯的子元素

E:last-child 自己是当前元素的父级所有子元素的最后一个子元素

E:first-of-type 自己是当前元素的父级所有该类子元素的第一个子元素

E:nth-of-type(n) 第n个子元素，计算方法只是E元素，会忽略其子元素的

E:nth-last-of-type(n) 同E:nth-of-type(n) 计算顺序相反。

E:only-of-child 自己是该类型的唯一子元素

E:last-of-type

注意：

1、当n做为一个独立值时，n取值为n>=1，例如nth-child(n)

2、当n做一个系数时，n取值为n>=0~~者n<0~~，例如，作为表达式匹配序号

nth-child(2n+1)、nth-child(-n+5) 此处需要理解2n+1或者-n+5做为一个整体不能小于1； 奇数 前5个

E:empty 元素没有任何内容才能被选中

E:target 自己必须是锚点，当跳转到自己的时候生效

E:before {

Content：“”

}

E:after 类似于增加一个东西空间，内容为content 里的，可以设置其他属性

### 伪元素选择器

E::selection 可改变选中文本的样式，被选中的情况下

E::placeholder 可改变placeholder默认样式，这个存在明显的兼容问题，比如::-webkit-input-placeholder，具体参考手册进行对比。

在定义placeholder的input中有用，input：placeholder{}

E:after、E:before 在旧版本里是伪类，在新版本里是伪元素，新版本下E:after、E:before会被自动识别为E::after、E::before，按伪元素来对待。

":" 与 "::" 区别在于区分伪类和伪元素

nav a span::after {  
 content: attr(data-hover); /\* attribute 自定义属性 的值\*/  
 获取了span的父元素的属性值（datahover的），可以设置为别的  
}

## 颜色

新增了RGBA、HSLA模式，其中的A 表示透明度通道，即可以设置颜色值的透明度，相较opacity，不具有继承性，即不会影响子元素的透明度。

Red、Green、Blue、Alpha即RGBA rgba(25,23,23,0.4)

R、G、B 取值范围0~255

Hue、Saturation、Lightness、Alpha即HSLA

H 取值范围0~360，0/360表示黑色、120表示绿色、240表示蓝色

S 取值范围0%~100%

L 取值范围0%~100%

A 取值范围0~1

关于透明度：

1、opacity子元素会继承父元素的透明度，在实际开发中会带来干扰；

2 、transparent 设置透明度时完全类似于“玻璃”一样的透明；

Background: transparen

文本

参考手册

文字阴影

与边框阴影相似，可分别设置偏移量、模糊度、颜色（可设透明度）。

**text-shadow**: 2px 2px 8px #000上下偏移 模糊度 颜色

打断文本并换行

**word-break: break-all;** 整个当做一个单词打断换行

**word-wrap: break-word;** 分别开左单词然后换行

保留原样式

**White-space：pre； 和pre标签相同**空白会被浏览器保留。

**White-space：pre—wrap** 保留空白符序列，但是正常地进行换行。

**White-space：pre—line** 合并空白符序列，但是保留换行符。

white-space: no-wrap 文本不会换行，文本会在在同一行上继续，直到遇到 <br> 标签

单行文本溢出，

**text-overflow: ellipsis;**

overflow:hidden;

white-space: nowrap;

多行省略溢出（手机用）

**text-overflow: ellipsis;**

**overflow: hidden;**

**display: -webkit-box;**

**-webkit-line-clamp: 5;**

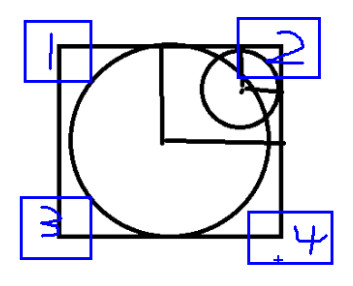
**-webkit-box-orient: vertical;**

边框

Radio是基于cleintwidth来的设置长短轴

百分比来设置的话，基数为对应方向上的尺度

其边框圆角、边框阴影属性，应用十分广泛，兼容性也相对较好，具有符合渐进增强原则的特征，我们需要重点掌握。



圆和椭圆

**半个椭圆 前面为水平方向上的长，右边为水平方向上的**

**够成椭圆的长短轴**

**width: 45px;**

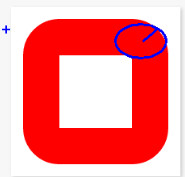
**border-radius: 45px 0 0 45px/90px 0 0 90px;**

表格

**表格运用圆角需要要 border-collapse: separate;**

内部直角

**当圆角半径小于或等于边框宽度时，元素内角是直角**



如何在PS中查看圆角半径？

圆饼：环

Height==width

R=height/2+border

实心圆：

Border宽=radius

四边不同色



width: 0px;

height: 0px;

border-width: 90px;

border-style: solid;

border-color: red green yellow blue;

border-right-color: transparent;

四分园



width: 0px;

height: 0px;

border-width: 90px;

border-style: solid;

border-color: red;

border-right-color: transparent;

border-radius: 90px;

上下不同色的圆



上下边设置宽， 实体宽width=2\*border

Radius=border、

width: 180px;

height: 0px;

border-top-width: 90px;

border-bottom-width: 90px;

border-style: solid;

border-top-color: red;

border-bottom-color: green;

/\*border-right-color: red;\*/

border-radius: 90px;

设置一半的内容一半的边框

width: 180px;

height: 90px;

border-bottom-width: 90px;

border-style: solid;

border-bottom-color: green;

background-color: red;

/\*border-right-color: red;\*/

border-radius: 90px;

阴阳图

前后加上元素，然后定位到改变位置

.item:nth-child(21) .border-radius::after,

.item:nth-child(21) .border-radius::before {

content: '';

position: absolute;

top: 50%;

width: 20px;

height: 20px;

/\*margin: -10px 0 0 0;\*/

border-width: 35px;

border-style: solid;

border-radius: 45px;

}

/\*左阴阳\*/

.item:nth-child(21) .border-radius::after {

background-color: red;

border-color: green;

right: 0;

}

/\*右阴阳\*/

.item:nth-child(21) .border-radius::before {

background-color: green;

border-color: red;

/\* right: 0; \*/

}



/\*奇怪的图形\*/

.item:nth-child(24) .border-radius {

width: 40px;

height: 40px;

border-width: 45px 0 45px 70px;

border-style: solid;

border-radius: 0 0 60px 0;

border-color: red;

}



.item:nth-child(26) .border-radius {

width: 160px;

height: 80px;

background-color: red;

border-radius: 6px;

position: relative;

}

.item:nth-child(26) .border-radius::after {

content: '';

position: absolute;

top: 0;

right: -20px;

width: 30px;

height: 30px;

border-width: 0 0 30px 30px;

border-style: solid;

border-bottom-color: red;

border-left-color: transparent;

border-radius: 0 0 60px 0;

}

边框图片

border-image-source: url(images/border.png);//源

border-image-width: 10px; //边框宽度

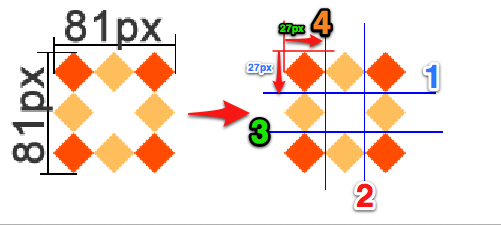
border-image-slice: 27; //怎么切，{1,4}参数

border-image-repeat: round;//round直接拉伸用，stretch 会缩放的布满

//repeat 不缩放，满了直接砍掉多余的

设置的图片将会被“切割”成九宫格形式，然后进行设置。如下图

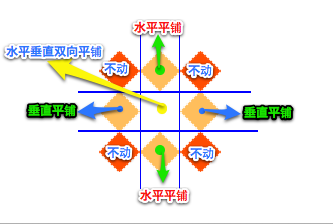
Border-img-slice：27 27 27 27;



Border-img-slice

“切割”完成后生成虚拟的9块图形，然后按对应位置设置背景，

其中四个角位置、形状保持不变，中心位置水平垂直两个方向平铺。如下图



round 会自动调整尺寸，完整显示边框图片。

repeat 单纯平铺多余部分，会被“裁切”而不显示。

.border-image {

margin: auto;

border-image-source: url(images/border.png);

border-image-width: 10px;

}

/\*标准写法\*/

.item:nth-child(1) .border-image {

width: 200px;

height: 100px;

border-image-slice: 27; //直接等距离切

}

/\*round平铺 不会截断\*/

.item:nth-child(2) .border-image {

width: 180px;

height: 90px;

border-image-slice: 27;

border-image-width: 15px;

border-image-repeat: round;

}

/\*stretch平铺\*/

.item:nth-child(3) .border-image {

width: 200px;

height: 100px;

border-image-slice: 27;

border-image-repeat: stretch;

}

/\*repeat平铺 会截断背景图片\*/

.item:nth-child(4) .border-image {

width: 180px;

height: 90px;

border-image-slice: 27;

border-image-width: 15px;

border-image-repeat: repeat;

}

/\*平铺效果不作用于四个角，只适应于四边\*/

.item:nth-child(5) .border-image {

width: 90px;

height: 90px;

border-image-slice: 27;

display: block;

border-image-width: 15px;

/\*border-image-repeat: repeat;\*/

}

/\*分别设置水平和垂直边的平铺效果\*/

.item:nth-child(6) .border-image {

width: 180px;

height: 90px;

border-image-slice: 27;

border-image-width: 15px;

border-image-repeat: round stretch;

}

/\*slice 参数\*/

.item:nth-child(7) .border-image {

width: 180px;

height: 180px;

border-image-slice: 27 40 40 27;

border-image-width: 60px;

}

边框阴影

/\*标准写法 左右偏移量 上下偏移量 模糊度 颜色\*/

.item:nth-child(1) .box-shadow {

box-shadow: 2px 2px 5px #000;

}

/\*水平偏移量 正值向右 负值向左 垂直正值向下 负值向上,仅代表方向,总偏移量为+模糊度\*/

.item:nth-child(2) .box-shadow {

box-shadow: -2px -2px 5px #000;

}

/\*上面有阴影\*/

.item:nth-child(3) .box-shadow {

box-shadow: 0 -2px 5px #000;

}

/\*下面有阴影\*/

.item:nth-child(4) .box-shadow {

box-shadow: 0 2px 5px #000;

}

/\*模糊度是不能为负 自行设一下\*/

/\*偏移量和模糊度是可以相加计算 负号只代表方向\*/

.item:nth-child(5) .box-shadow {

box-shadow: -5px -5px 0px #000;

}

/\*偏移量和扩展也是相加的 ，扩展是4周都有

但是会和偏移量相互作用，就是真的偏移了，上增下减\*/

.item:nth-child(6) .box-shadow {

box-shadow: 15px 15px 0 #000;

}

/\*偏移量和扩展也是相加的 需要对比第六个\*/

.item:nth-child(7) .box-shadow {

box-shadow: 0 0 0 10px #000;

}

/\*移量和扩展也是数学运算的\*/

.item:nth-child(8) .box-shadow {

内阴影

box-shadow: **inset** 20px 20px 20px -10px #000;

设置边框阴影不会影响盒子的布局，即不会影响其兄弟元素的布局

spread可以与blur、h-shadow、v-shadow相互抵消，blur不可为负值

可以设置多重边框阴影，实现更好的效果，增强立体感。

/\*内外阴影\*/

.item:nth-child(10) .box-shadow {

box-shadow: inset 20px 20px 20px -10px #000, 20px 20px 10px red;

}

/\*多个内外阴影，谁在前谁在上\*/

.item:nth-child(11) .box-shadow {

box-shadow: inset 20px 20px 0px -10px green, inset 20px 20px 20px -10px #000, 20px 20px 10px red, 20px 20px 10px blue;

}

/\*是否会影响盒子布局\*/

.item:nth-child(12) .box-shadow {

width: 90px;

height: 90px;

margin-right: 10px;

background-color: red;

}

/\*不会影盒子的布局\*/

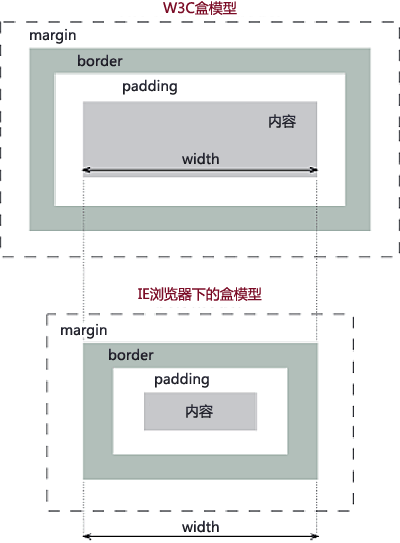
.item:nth-child(12) .left {

box-shadow: 50px 50px 50px 20px blue;

}

盒模型

关于盒模型存在两种形式，分别是W3C标准盒模型和IE盒模型，如下图所示，其区别主要在于宽度和高度的计算方式，CSS3对盒模型做出了新的定义，即允许开发人员指定盒子宽度和高度的计算方式。



IE盒模型只会出现在IE5版本和IE6+的怪异模式中。

//设置成怪异模式了，默认的content-box

box-sizing: border-box;

/\*box-sizing: content-box;\*/

border: 10px solid red;

/\*类似 IE低版本 怪异模式\*/

font-size: 20px;

背景

**背景区域**

background-clip: border-box;

/\*默认北景填充区域\*/

background-clip: content-box;

/\*可以修改我的背景区域\*/

background-clip: padding-box;

/\*从padding开始\*/

**定位原点**

background-origin: border-box;

background-origin: content-box;

background-origin: padding-box; //默认

/\*修改的是定位原点\*/

缩放

**Background-size：40px 40px ; //将图片缩放到 40x40里面**

background-size: auto 100%; //百分比相对于父元素的大小

background-size: 100% 100%; //auto指，一个边定了之后，里一个边按比例来缩放

background-size: cover;

/\***cover** 安最大边会使“**最大”边，进行缩放，**另一边同比缩放，铺满容器，超出部分会溢出。\*/

/\***contain** 会**使“最小”边，进行缩放**，另一边同比缩放，不一定铺满容器，会完整显示图片。按盒子的最小边\*/

/**\*background-size 与 background-clip无关\*/**

**/\*background-size 与 background-origin 保持一致\*/**

background-size会以background-clip设定的盒模型计算

背景图片尺寸在实际开发中应用十分广泛。

多图背景

background: url(images/bg1.png) left top no-repeat,

url(images/bg2.png) right top no-repeat,

url(images/bg3.png) right bottom no-repeat,

url(images/bg4.png) left bottom no-repeat,

url(images/bg5.png) center center no-repeat;

这里left top 指的是背景图位于盒子的哪部分 ，数字的话是 00 的表示，改变可以使用就精灵吐了

渐变

线性渐变

/\*标准写法\*/

.item:nth-child(1) .linear-gradient {

background-image: linear-gradient(yellow, green);

}

/\*多个颜色渐变\*/

.item:nth-child(2) .linear-gradient {

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

}

/\*用角度确定方向\*/

.item:nth-child(3) .linear-gradient {

background-image: linear-gradient(0, yellow, green);

}

/\*用角度确定方向 正上方为0度 顺时针旋转\*/

.item:nth-child(4) .linear-gradient {

background-image: linear-gradient(90deg, yellow, green);

}

/\*用角度确定方向 180deg 正上方为0度 顺时针旋/默认为180deg\*/

.item:nth-child(5) .linear-gradient {

background-image: linear-gradient(180deg, yellow, green);

}

/\*用关键字来确定方向\*/

.item:nth-child(6) .linear-gradient {

background-image: linear-gradient(to top, yellow, green);

}

/\*用关键字来确定方向\*/

.item:nth-child(7) .linear-gradient {

background-image: linear-gradient(to left top, yellow, green);

}

/\*用关键字来确定方向\*/

.item:nth-child(8) .linear-gradient {

background-image: linear-gradient(135deg, yellow, green);

}

/\*第一个后面的数字是到哪结束纯色，后面的每一个都是到哪开始纯色，并且不

可被越过，后面部分保持到结束,没标注位置的直接从上个达到纯色之后开会渐变\*/

.item:nth-child(9) .linear-gradient {

background-image: linear-gradient(to left, yellow 20%, green 40%, blue);

}

.item:nth-child(11) .linear-gradient {

background-image: linear-gradient(to left, yellow 50%, green 40%, blue 50%);

}

径向渐变

/\*从一个中心点开始沿着四周方向进行渐变\*/  
.item:nth-child(1) .radial-gradient {  
 background-image: radial-gradient(yellow, green);  
}  
/\*1、辐射范围到这个之后便纯色 2、中心点，以左上角为基点 3、颜色的起止\*/  
.item:nth-child(2) .radial-gradient {  
 background-image: radial-gradient(120px at center center,yellow, green);  
}  
  
/\*辐射范围的半径可以不等即可以是椭圆\*/  
.item:nth-child(4) .radial-gradient {  
 background-image: radial-gradient(120px 80px at center center,yellow, green);  
}  
  
/\*不写辐射范围写成circle 或者\*/  
.item:nth-child(5) .radial-gradient {  
 background-image: radial-gradient(circle at center center,yellow, green);  
}  
 /\*椭圆\*/  
.item:nth-child(6) .radial-gradient {  
 background-image: radial-gradient( ellipse at center center,yellow, green);  
}

多重渐变

repeating-linear-gradient(#f00, #ff0 10%, #f00 15%);

repeating-linear-gradient(to bottom, #f00, #ff0 10%, #f00 15%);

repeating-linear-gradient(180deg, #f00, #ff0 10%, #f00 15%);

repeating-linear-gradient(to top, #f00, #ff0 10%, #f00 15%);

水平开始，顺时针为正，

repeating-linear-gradient(54deg, #f00, #ff0 10%, #f00 15%)

伸缩布局

新版伸缩布局

这里我们需要引入一些新的概念：

主轴：Flex容器的主轴主要用来配置Flex项目。

侧轴：与主轴垂直的轴称作侧轴，是侧轴方向的延伸。

主轴和侧轴并不是固定不变的，通过flex-direction可以调整。



另个两个版本伸缩布局其实现思路与新版基本一致，区别在于其属性及属性值不同，熟练掌握新版伸缩布局后，要参照对比另外两个版本的不同。

**在父标签中定义**

Display：flex;

//收缩布局

Flex-direction:

主轴方向，切换主轴

调整主轴方向（默认为水平方向）包括row、column、row-reverse、column-reverse

justify-content:

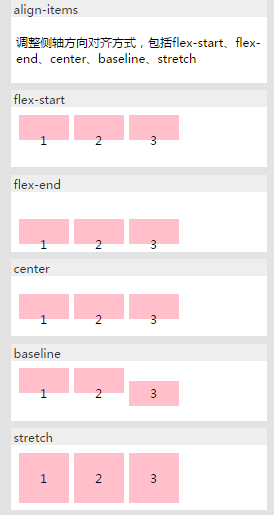
space-between;//主轴方向上的对其

flex-start、flex-end、center、space-around、space-between



align-items:

调整侧轴方向对齐方式，包括flex-start、flex-end、center、baseline（基于基线）、stretch（不设置高（或者auto）的话自动拉伸）



flex-wrap:

**wrap;**

**换行 ，no‑wrap的话会忽略子元素的宽，都凑到一行上**

Align-content

堆栈排列换行后，可对应用flex-wrap: wrap后产生的换行进行控制，包括flex-start、flex-end、center、space-between、space-around、stretch

flex-flow

flex-direction和flex-wrap的简写形式

子元素

Flex：numberr

在没指定宽度的时候，通过数字占比例来分配父盒子剩余的空间

align-self

同align-items可覆盖父元素设置的algin-items，包括flex-start、flex-end、center、baseline、stretch

Order

order: 21;

控制子元素的顺序

**越大越靠后**

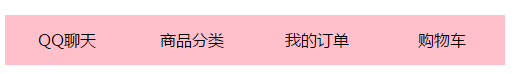
所有方向上水平垂直居中

Css 3

Margin：auto;本来是只支持水平方向上的，现在都支持了

例子

1.自适应导航



section:nth-child(2) ul {  
 width: 500px;  
 height: 50px;  
 display: flex;  
}  
  
section:nth-child(2) li {  
 flex: 1;  
 text-align: center;  
 line-height: 50px;  
 background-color: pink;  
}

2.经典布局

/\*经典网页布局\*/  
.layout {  
 width: 400px;  
 height: 600px;  
 display: flex;  
 /\*默认是不换行的，所以要用wrap\*/  
 flex-wrap: wrap;  
 /\*外面盒子需要垂直排列\*/  
 flex-direction: column;  
 /\*头部和底部贴边\*/  
 justify-content: space-between;  
 /\*整体的部分设置overflow会导致两边出现下拉条\*/  
 /\*overflow: scroll;\*/  
}  
 /\*头部和底部设置高，之后中间设置flex=1，等分中间的时候讲父元素设置成display=flex，里面在分别设置flex值\*/

多列布局

/\*设置分成几列\*/  
-webkit-column-count: 2;  
  
/\*当列宽\*列数乘积大于盒子总宽，会自动调整列数\*/  
-webkit-column-width: 400px;  
  
/\*调整列之间的宽度\*/  
-webkit-column-gap: 60px;  
中间线  
-webkit-column-rule: 2px dashed #CCC;

h4 {

font-size: 24px;

/\*all代表所有列\*/

-webkit-column-span: all;

}

过渡 transition

Transparent透明的

设置transition之后，属性变化时的动画变成可控的了

transition: all 1s;  
/\*当前元素只要有“属性”发生变化时，可以平滑的进行过渡\*/

transition: height 1s, width 0.5s;

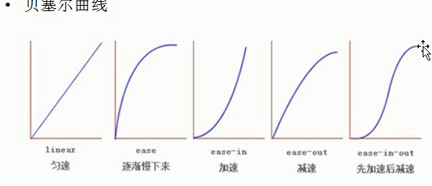
属性可以设置成all也可以设置单独的，不是字符串

/\*transition: left 2s;\*/  
transition-property: left;  
transition-duration: 2s;

控制速度

Linear 线性

/\*transition-timing-function: linear;\*/  
/\*控制过渡的速度\*/  
transition-timing-function: ease; 先快后慢



延迟后再动

transition-delay: 5s;

transform: rotate(45deg);

动画animation

帧动画：通过一帧一帧的关键画面按照固定顺序和速度播放。如电影胶片

补间动画：自动完成从起始状到终止状的的过度

写在一起或者每个属性分开写

animation: change 5s infinite;

/\*其中有3个参数与过渡一致即 transition-delay、transition-timing-function、transition-duration\*/  
  
.animation {  
   
 /\*动画名称\*/  
 animation-name: change;  
 /\*动画持续时间\*/  
 animation-duration: 1s;  
 /\*动画结束后的状态\*/  
 animation-fill-mode: forwards;  
 /\*无限次播放\*/  
 animation-iteration-count: infinite;  
 /\*动画延时\*/  
 animation-delay: 1s;  
 /\*动画暂停\*/  
 animation-play-state: paused;  
 /\*动画反方向\*/  
 animation-direction: alternate;  
 /\*动画速度\*/  
 animation-timing-function: linear;  
}  
  
/\*动画序列\*/  
**@keyframes** change {  
 0% {  
 width: 40px;  
 height: 40px;  
 }  
  
 100% {  
 width: 200px;  
 height: 200px;  
 }  
}

转换 transform

**Transform-origin：top left；原点**

transform: rotate(45deg); //里面是角度，可正可负

transform: scale(x,y); // 里面是倍数

transform: translate(x,y);//移动的距离

**transform：rotatex(60deg)**

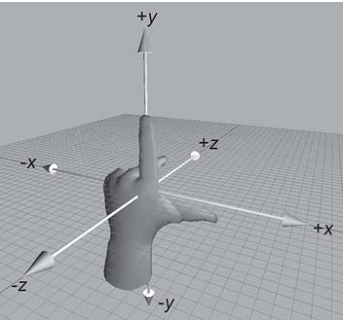
transform：rotatex(60deg)

2D 转换

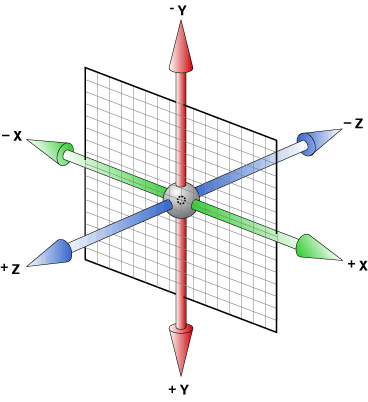
translate(x,y) x、y 可为负值，相对自身移动，并未脱离文档流。

左手坐标系：伸出左手，让拇指和食指成“L”形，大拇指向右，食指向上，中指指向前方。这样我们就建立了一个左手坐标系，拇指、食指和中指分别代表X、Y、Z轴的正方向。

左手法则：左手握住旋转轴，竖起拇指指向旋转轴正方向，正向旋转方向就是其余手指卷曲的方向。



3D坐标轴，用X、Y、Z分别表示空间的3个维度，三条轴上互相垂直。



3d旋转

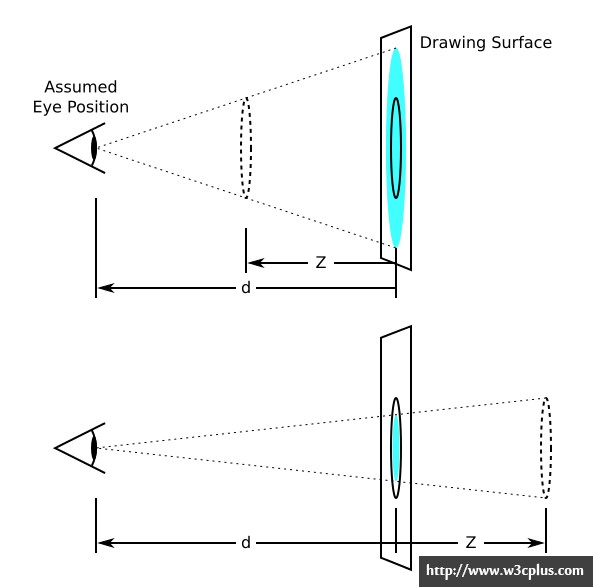
RotateX(60deg)

视距perspective

定义在父盒子上；数值越小，效果越明显（三角形的高越小，边的斜率越大），

只对3d效果有用； 无穷大或者0 或者不设置或者none都是没效果的

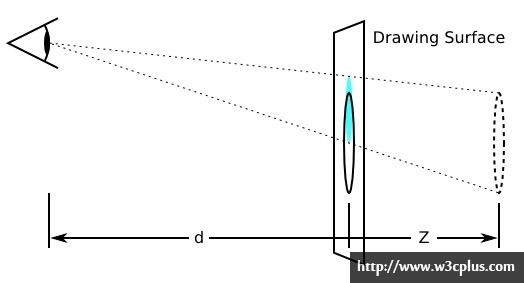
perspective属性，经常和translateZ的结合起来使用。



perspective-origin属性

边线焦点在 **X轴和Y轴的坐标**

**默认50% 50% ，定义在父元素上**



transform-style属性（重点）

**preserve-3d flat**

transform-style属性是3D空间一个重要属性，

其中flat值为默认值，表示所有子元素在2D平面呈现。preserve-3d表示所有子元素在3D空间中呈现。

Preserve-3d 保持3d

颜色： span #2195de span:after #0965a0

媒体查询

由于网页呈现终端设备越来越趋向于多样化，尤其是移动终端（手机），具有不同屏幕尽寸、不同分辨率，为了保证网页能十分友好的呈现，CSS3为开发人员提供了可以识别呈现终端的方法，这样便可以有针对性的为不同的呈现终端分别进行处理，被广泛应用于响应式开发中。

html 标签方式：

css 属性方式：

常使用的是检测设备宽度

参考手册

Web字体

开发人员可以为自已的网页指定特殊的字体，无需考虑用户电脑上是否安装了此特殊字体，从此把特殊字体处理成图片的时代便成为了过去。

支持程度比较好，甚至IE低版本浏览器也能支持。

字体格式

不同浏览器所支持的字体格式是不一样的，我们有必要了解一下有关字体格式的知识。

1、TureTpe(.ttf)格式

.ttf字体是Windows和Mac的最常见的字体，是一种RAW格式，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome4+、Safari3+、Opera10+、iOS Mobile、Safari4.2+；

2、OpenType(.otf)格式

.otf字体被认为是一种原始的字体格式，其内置在TureType的基础上，支持这种字体的浏览器有Firefox3.5+、Chrome4.0+、Safari3.1+、Opera10.0+、iOS Mobile、Safari4.2+；

3、Web Open Font Format(.woff)格式

woff字体是Web字体中最佳格式，他是一个开放的TrueType/OpenType的压缩版本，同时也支持元数据包的分离，支持这种字体的浏览器有IE9+、Firefox3.5+、Chrome6+、Safari3.6+、Opera11.1+；

4、Embedded Open Type(.eot)格式

.eot字体是IE专用字体，可以从TrueType创建此格式字体，支持这种字体的浏览器有IE4+；

5、SVG(.svg)格式

.svg字体是基于SVG字体渲染的一种格式，支持这种字体的浏览器有Chrome4+、Safari3.1+、Opera10.0+、iOS Mobile Safari3.2+；

关于字体介绍摘自http://www.w3cplus.com/content/css3-font-face/

了解了上面的知识后，我们就需要为不同的浏览器准备不同格式的字体，通常我们会通过字体生成工具帮我们生成各种格式的字体，因此无需过于在意字体格式间的区别差异。

推荐http://www.zhaozi.cn/、http://www.youziku.com/ 查找更多中文字体

字体图标

其实我们可以把文字理解成是一种特殊形状的图片，反之我们是不是也可以把图片制作成字体呢？

不同的浏览器支持不同的字体图标，需要**引用**多个

@font-face {

font-family: 'itcast';

src: url('../font/MiFie-Web-Font.eot') format('embedded-opentype'),

url('../font/MiFie-Web-Font.svg') format('svg'),

url('../font/MiFie-Web-Font.ttf') format('truetype'),

url('../font/MiFie-Web-Font.woff') format('woff');

}

将所有的用到字体图标的**class都定义为 icon-** 的形式，然后匹配所有的字体图标的类用于**初始化**，一定要把他们 放到开始，

[class^="icon-"],

[class\*=" icon-"] {

font-family: itcast;

}

兼容性

通过http://caniuse.com/ 可查询CSS3各特性的支持程度，一般兼容性处理的常见方法是为属性添加私有前缀，如不能解决，应避免使用，无需刻意去处理CSS3的兼容性问题。

###### 高级应用

360 nav 的音乐导航鼠标按下事件

**Keydown**

**$**(**window**).keydown(function(event) { // 事件对象  
 // e.keycode 返回该键盘的 unicode 编码,1对应的49  
 var num = 0;  
 if (event.**keyCode** >= 49 && event.**keyCode** <= 56) {  
 /\*比如 我按了 1 键盘 应该 第 0 个li操作  
 比如 我按了 2 键盘 应该 第 1 个li操作 \*/  
 num = event.**keyCode** - 49;  
  
 **$**("nav li").eq(num).trigger("mouseenter"); // 事件触发  
 setTimeout(function() {  
 **$**("nav li").eq(num).trigger("mouseleave"); // 事件触发  
 }, 300)  
  
 }  
})

**$**("nav li").**mouseenter**(function() {  
 **$**(this).children("ins").stop().animate({  
 **top**: 0  
 }, 200);  
 **$**("audio").get(**$**(this).index()).load(); // 加载，变成dom对象  
 **$**("audio").get(**$**(this).index()).play(); // 播放  
 }).**mouseleave**(function() {  
 **$**(this).children("ins").stop().animate({  
 **top**: 35  
 }, 200);  
 })

动态的图片transition

url(images/paopao.png) no-repeat right bottom #369;  
border-radius: 15px;  
transition: all 1s;

.pop:hover {  
 background-position: right bottom, top left;  
}

手风琴效果 transition

每个下面第一一个高度为0的盒子，点击的时候把高度弄成大的

Z

正反向旋转

Display:inline; //自动适应大小

div {  
 display: inline-block;  
 transition: all 0.5s;  
}  
.rotateDiv {  
 transform: rotate(360deg);  
}

//js中，添加和删除都会转，一个正一个父

**$**(this).toggleClass("rotateDiv");

判断鼠标进入方向

**$**(".list").on("mouseenter mouseleave",function(e) {  
 var that = **$**(this); // 先把当前元素存起来  
 var w = **$**(this).width(); // 得到盒子宽度  
 var h = **$**(this).height();// 得到盒子高度  
 var x = (e.**pageX** - this.**offsetLeft** - (w / 2)) \* (w > h ? (h / w) : 1);  
 // 获取x值  
 var y = (e.**pageY** - this.**offsetTop** - (h / 2)) \* (h > w ? (w / h) : 1);  
 // 获取y值  
 var direction = **Math**.round((((**Math**.atan2(y, x) \* (180 / **Math**.**PI**)) + 180) / 90) + 3) % 4;  
 //direction的值为“0,1,2,3”分别对应着“上，右，下，左”

这里多个事件的处理办法

if(e.**type** == 'mouseenter'){  
 **$**(this).html(dirName[direction]+'进入');  
}else{  
 **$**(this).html(dirName[direction]+'离开');  
}

不同的方向不同的动画

.list {  
 border-radius: 15px;  
 overflow: hidden;  
 width: 300px;  
 height: 200px;  
 float: left;  
 margin: 20px;  
 position: relative;  
}  
.list div {  
 position: absolute;  
 top: 0;  
 left: 0;  
 transform: rotate(-100deg);  
 transform-origin: top left; /\* 旋转中心点\*/  
 transition: transform .8s;  
}  
.current div {  
 transform: rotate(0deg);  
}  
.list div.top {  
 transform-origin: top left;  
}  
.list div.right {  
 transform-origin: top right;  
}  
.list div.bottom {  
 transform-origin: bottom right;  
}  
.list div.left {  
 transform-origin: bottom left;  
}

**$**(function() {  
 **$**(".list").on("mouseenter mouseleave",function(e) {  
 var that = **$**(this); // 先把当前元素存起来  
 var w = **$**(this).width(); // 得到盒子宽度  
 var h = **$**(this).height();// 得到盒子高度  
 var x = (e.**pageX** - this.**offsetLeft** - (w / 2)) \* (w > h ? (h / w) : 1);  
 // 获取x值  
 var y = (e.**pageY** - this.**offsetTop** - (h / 2)) \* (h > w ? (w / h) : 1);  
 // 获取y值  
 var direction = **Math**.round((((**Math**.atan2(y, x) \* (180 / **Math**.**PI**)) + 180) / 90) + 3) % 4;  
 //direction的值为“0,1,2,3”分别对应着“上，右，下，左”  
 // 将点的坐标对应的弧度值换算成角度度数值 0 1 2 3  
 if(e.**type** == 'mouseenter'){  
 switch(direction) {  
 case 0: // 鼠标从上面进来  
 comeOn("top");  
 break;  
 case 1: // 鼠标从上面进来  
 comeOn("right");  
 break;  
 case 2: // 鼠标从上面进来  
 comeOn("bottom");  
 break;  
 case 3: // 鼠标从上面进来  
 comeOn("left");  
 break;  
 }  
 }else{  
  
 switch(direction) {  
 case 0: // 鼠标从上面进来  
 getOut("top");  
 break;  
 case 1: // 鼠标从上面进来  
 getOut("right");  
 break;  
 case 2: // 鼠标从上面进来  
 getOut("bottom");  
 break;  
 case 3: // 鼠标从上面进来  
 getOut("left");  
 break;  
 }  
 }  
 function comeOn(str) {  
 that.addClass("current");  
 that.children("div").removeClass().addClass(str);  
 // 先清除前的类， 留下现在的  
 }  
 function getOut(str) {  
 that.removeClass("current");  
 that.children("div").removeClass().addClass(str);  
 }  
 });  
})

###### HTTP服务和AJAX编程

服务器类型

按服务类型可分为：文件服务器、数据库服务器、邮件服务器、Web 服务器等；

按应用软件可分为 Apache服务器、Nginx 服务器、IIS服务器、Tomcat服务器、Node服务器等。

服务器软件

HTTP 服务器：Apache、Nginx、IIS、Tomcat、NodeJS等；

通过安装相应的服务软件，然后进行配置后就可以使计算具务了提供某种服务的能力

HTTP服务器

即网站服务器，主要提供文档(文本、图片、视频、音频)浏览服务。

服务端开发

HTTP服务器可以结合某一编程语言处理业务逻辑，由此进行的开发，通常称之为服务端开发；

常见的运行在服务端的编程语言包括 PHP、Jsp、Asp、Python、Ruby、Perl等；

前端开发

以浏览器为宿主环境，结合 HTML、CSS、Javascript等技术，而进行的一系列开发，通常称之为前端开发。

注：这里会涉及到 B/S、C/S 结构(了解)

网络基础

IP地址

所谓IP地址就是给每个连接在互联网上的主机分配的一个32位地址。(就像每部手机能正常通话需要一个号码一样)

查看本机IP地址 ipconfig、ifconfig

域名

域名是一个IP地址的“面具”，由于IP地址基于数字，不方便记忆，于是便用域名来代替IP地址。

查看域名对应的IP地址 ping、tracert

DNS

DNS记录了 IP 地址和域名的映射（对应）关系；

查找优先级 本机hosts文件、DNS服务器

端口

端口号是计算机与外界通讯交流的出口，每个端口对应不同的服务

查看端口占用情况 netstat -an

常见端口号 80、8080、3306、21、22

搭建HTTP服务

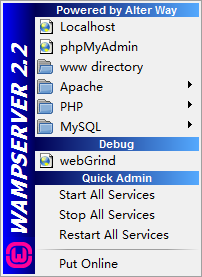
Windows + Apache + Mysql + PHP，首字母组合。

安装WampServer

安装wampserver，和普通软件安装无差别，除指定安装路径外，其它默认安装

管理HTTP服务

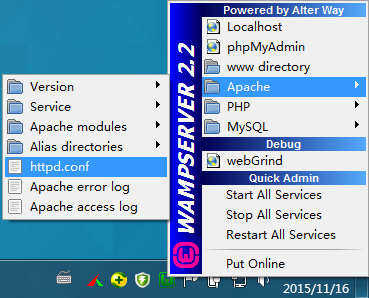
任务图标绿色为正常启动状态



配置网站根目录

网站根目录是Web服务器上存放网站程序的空间。

a) 控制台选择



或者 wampserver安装目录下

bin\apache\Apache2.2.21\conf\httpd.conf

b) 查找并修改



例如：



这样就指定了 "D:/www/"为存放网站的根目录。

c)配置根目录，查找



修改成



d)修改完后，并不能立即生效，需要重启Apache

注：可以指定任意目录为根目录

网站部署

配置虚拟主机

可以在一台Web服务器上指定多个虚拟服务器，可分别为每个虚拟服务器指定根目录。

a)在httpd.conf 中找到



去掉前面的#号注释，开启虚拟主机配置

b)打开conf/extra/httpd-vhosts.conf



DocumentRoot "D:/www/example"

ServerName "example.com "

ServerAlias "www.example.com"

其它项无需指定

c)修改DNS（hosts）文件

打开C:\Windows\System32\drivers\etc\hosts



注：修改hosts文件权限

d)重启Apache

异步：

浏览器发送请求后，页面不白屏，后面步骤可以继续进行

网络传输协议

指服务器和客户端间进行通信时的约束和规范。

常见协议

1、HTTP 超文本传输协议

2、FTP 文件传输协议

3、SMTP 简单邮件传输协议等。

HTTP协议

即超文本传输协议

一次HTTP连接由从客户机到服务器的请求(Request)和从服务器到客户机的响应(Response)构成。

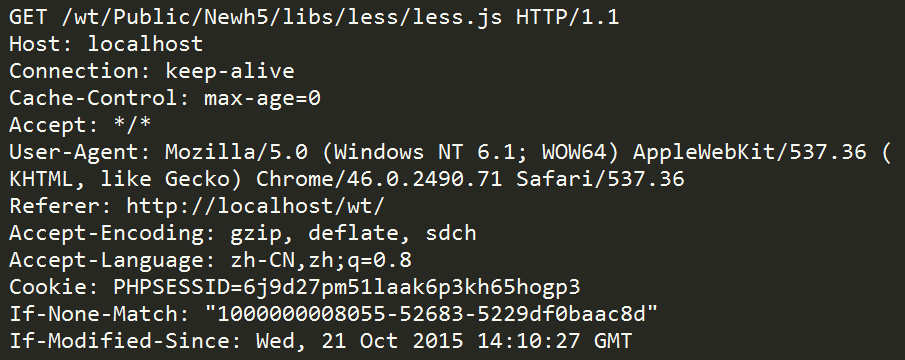


常用请求方法 POST、GET、PUT、DELETE

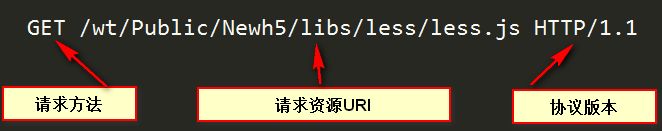
请求

a)请求由客户端发起

格式组成：请求行、请求头、请求主体



aa)请求行



ab)请求头

Host：localhost请求的主机

Connection：Keep-alive 持久连接

Cache-Control：max-age=0控制缓存

Accept：\*/\* 接受的文档MIME类型

User-Agent：很重要

Referer：“我从哪里来的”

Accept-Encoding：可接受的压缩格式

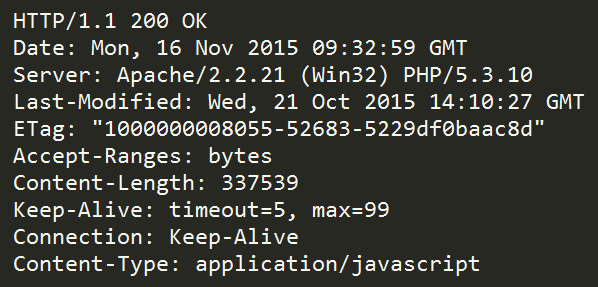
If-None-Match：记录服务器响应的ETag值，用于控制缓存

If-Modified-Since：记录服务器响应的Last-Modified值

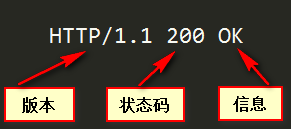
响应

b)响应由服务器发出

格式组成：状态行、响应头、响应主体



ba)状态行



bb)响应头

Date：响应时间

Server：服务器信息

Last-Modified：资源最后修改时间

ETag：资源修改后生成的唯一标识

Content-Length：响应主体长度

Keep-Alive：持久连接的时间

Content-Type：响应资源的MIME类型

调试工具

利用HTTP抓包工具在开发中可以帮我们进行调试，常用抓包工具HttpWatch、Fiddler、Charles、FireBug等

浏览器插件

Firebug、HttpWatch、chrome dev tools

代理软件

Charles、Fiddler

协商缓存

利用浏览器的缓存机制，可以有效的减少HTTP的请求，提高页面加载速度，增强用户体验，同时也能极大的减轻服务器的负担，结合HTTP协议，缓存协商就是根据HTTP协议实现缓存控制的一种机制。

PHP 基础

变量

变量以$开头 字母/数字/下划线 不能以数字开头

大小写敏感

变量作用域 global local \*static

内容输出

echo：输出简单数据类型，如字符串、数值

print\_r()：输出复杂数据类型，如数组

var\_dump()：输出详细信息，如对象、数组

数据类型

字符型、整型、浮点型、布尔型、数组、对象、NULL

gettype() 检测数据类型

is\_string() 是否是字符

is\_array() 是否是数组

运算符

基本与Javascript语法一致

. 号表示链接符

分支、循环语句

与Javascript基本一致

foreach()

switch()

函数

与Javascript基本一致

函数名对大小写不敏感

默认参数

文件引入

include 引入失败后程序继续执行

require 引入失败后程序终止执行

超全局变量

$\_GLOBALS

$\_SERVER

$\_GET

$\_POST

$\_REQUEST

$\_FILES

$\_COOKIE

$\_SESSION

$\_ENV

表单处理

表单name属性的是用来提供给服务端接收所传递数据而设置的

表单action属性设置接收数据的处理程序

表单method属性设置发送数据的方式

当上传文件是需要设置 enctype="multipart/form-data"

$\_GET接收 get 传值

$\_POST接收 post 传值

$\_FILES接收文件上传

常用PHP函数

AJAX编程

**Ajax**

**局部更新内容**

**可以用iframe ajax**

本质是在HTTP协议的基础上通过异步的方式与服务器进行通信

传统的通信

Php中

header("location:./01form.html?flag=1");

js中

var str = **window.location.search; 获取头文件？及其之后的东西**

if(str){

var flag = str.split('=');

if(flag[1] == 1){

document.getElementById('showInfo').innerHTML = '注册成功';

}

}

ajax详解

request.readystate

创建 0

Open 1

Send 1

等待：完全发送 2，得到响应3，可以使用4

代码解析

如果用ajax，则默认的提交按钮是submit的话需要阻止默认行为，一般写成button，然后在点击事件中，定义ajax，

Ajax实际是某行为触发了网络交互，将a和submit等会刷新页面的事件给换成自己用request来发送请求，然后处理数据，然后更新页面信息

**请求行用**

**request.open(‘get’,url,true)**

Get方式提交的时候，url是拼接的字符串，

传送的数据通过 uri？name=value&name2=value2 来传递，请求方式异步为true

var url = '05open.php?username='+ encodeURIComponet(username)+'&password='+password;

**var url = '05open.php?username='+ encodeURIComponet(username)+'&password='+password;**

**解决中文乱码**

**request.open(‘post’,uri,true)**

url不带数据，并且，如果在form中没有写enctypy的话需要在请求提发送前，写请求头



**请求体**

**Get方式 request.send(null);**

**Post**



**Param=’name=’+name+’&password=’+password;**

**request.send(param);**

**接口：url可以获取数据的**

**响应部分**

request.onreadystatechange = function(){

if(request.readystate==4&&request.status==200){

//已经获取到数据并且可使用呢

Var content= request.responseXML;//xml文件

//Var conten= request.responseText;//json和纯文本

//两种数据文件的内容对象呢

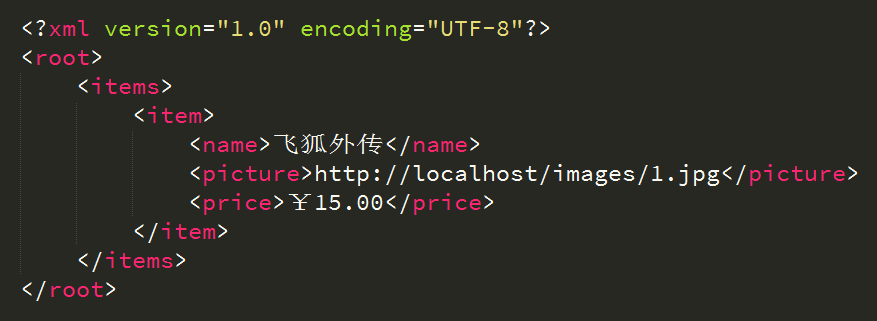
}

}

XML

XML是一种标记语言，很类似HTML，其宗旨是用来传输数据，具有自我描述性。

a)标准XML文档



必须有一个根元素

不可有空格、不可以数字或.开头、大小写敏感

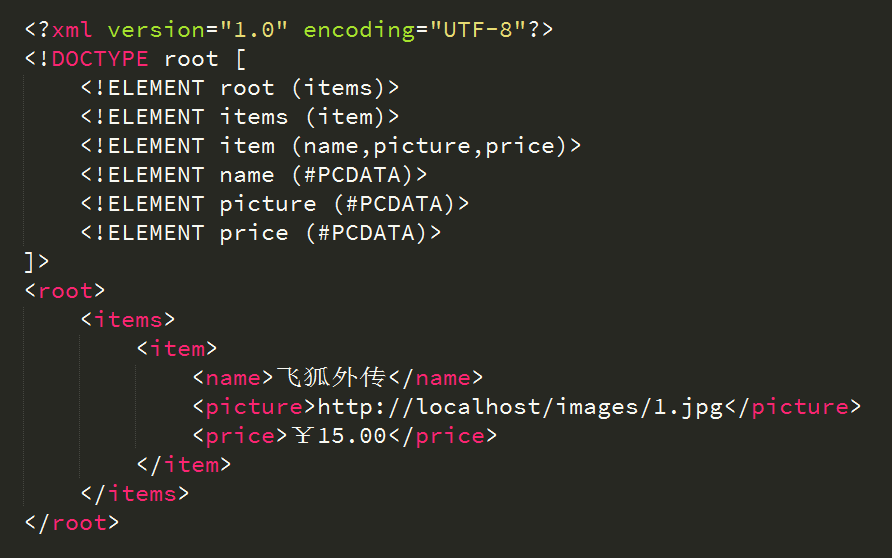
不可交叉嵌套

属性双引号

特殊符号要使用实体

注释和HTML一样

**b) XML验证 (DTD)**



JSON

即 JavaScript Object Notation，另一种轻量级的文本数据交换格式

独立于语言

语法规则

数据在名称/值对中

数据由逗号分隔

花括号保存对象方括号保存数组

使用双引号

JSON解析

eavl()、JSON对象 JSON.parse()、JSON.stringify()

归纳总结XML和JSON的差异以及优缺点

**Responsetext返回的是字符串，需要操作**

**Var json=JSON.parse(request.responseText);转化成json对象**

var str = '{"name":"三国演义","category":"文学"}';

// 字符串转换成json

var data = JSON.parse(str);

var s = JSON.stringify(data);

// 把json转化成字符串

var str=JSON.stringify(data);

var data = eval("("+ str +")

**window**.onload = function(){  
 var btn = **document**.getElementById('btn');  
 btn.onclick = function(){  
 var code = **document**.getElementById('code').**value**;  
 var xhr = null;  
 try{  
 xhr = new XMLHttpRequest();  
 }catch(e){  
 xhr = new ActiveXObject('Microsoft.XMLHTTP');  
 }  
 xhr.open('get','07ems.php?code='+code);  
  
 xhr.send(null);  
  
 xhr.**onreadystatechange** = function(){  
  
 if(this.**readyState** == 4 && this.**status** == 200){  
 var data = **JSON**.parse(this.**responseText**);  
 if(data.**status** == 0){  
 var list = data.data.info.**context**;  
 var tag = '';  
 for(var i=0;i<list.**length**;i++){  
 var item = list[i];  
 var desc = item.desc;  
 var t = new Date();  
 t.setTime(item.time+'000');  
 var str = t.getFullYear()+'年'+(t.getMonth()+1)+'月'+t.getDay()+'日'+(t.getHours()>12?' 下午':' 上午')+t.getHours()%12+':'+t.getMinutes()+':'+t.getSeconds();  
  
 tag += '<li><div>'+str+'</div><div>'+item.desc+'</div></li>';  
  
 }  
  
 var ul = **document**.createElement('ul');  
 ul.**innerHTML** = tag;  
 **document**.getElementById('list').appendChild(ul);  
  
 }else{  
 var info = data.msg;  
 **document**.getElementById('list').**innerHTML** = info;  
 }  
  
  
 }  
   
  
  
 }  
  
 //这里的代码依然执行  
  
  
  
  
  
 }  
}

原装ajax封装

function ajax(data){  
 //data={data:"",dataType:"xml/json",type:"get/post"，url:"",asyn:"true/false",success:function(){},failure:function(){}}  
 //data:{username:123,password:456}  
 //data = 'username=123&password=456';  
 //第一步：创建xhr对象  
 var xhr = null;  
 if(**window**.XMLHttpRequest){//标准的浏览器  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject('Microsoft.XMLHTTP');  
 }  
 //第二步：准备发送前的一些配置参数  
 var type = data.**type** == 'get'?'get':'post';  
 var url = '';  
 if(data.**url**){  
 url = data.**url**;  
 if(type == 'get'){  
// 分装原装的ajax，get需要加上不同的数据来清理缓存，或者是通过不同的请求来重新向客户端发送  
 url += "?" + data.data + "&\_t="+new Date().getTime();  
 }  
 }  
 var flag = data.asyn == 'true'?'true':'false';  
 xhr.open(type,url,flag);  
  
 //第三步：执行发送的动作  
 if(type == 'get'){  
 xhr.send(null);  
 }else if(type == 'post'){  
 xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xhr.send(data.data);  
 }  
  
 //第四步：指定回调函数  
 xhr.onreadystatechange = function(){  
 if(this.readyState == 4){  
 if(this.status == 200){  
 if(typeof data.success == 'function'){  
 var d = data.**dataType** == 'xml'?xhr.**responseXML**:xhr.**responseText**;  
 data.success(d);  
 }  
 }else{  
 if(typeof data.failure == 'function'){  
 data.failure();  
 }  
 }  
 }  
 }  
  
**}**

兼容性

IE5、IE6中使用 ActiveObject("Microsoft.XMLHTTP")

如下图



封装Ajax工具函数

函数

//javascript 代码运行分连个阶段：  
// 1、预解析 --- 把所有的函数定义提前，所有的变量声明提前，变量的赋值不提前  
// 2、执行 --- 从上到下执行 （setTimeout，setInterval,ajax中的回调函数，事件中的函数需要触发执行）

//函数的嵌套形成闭包  
var i=3;  
var data = [];  
for(i=0;i<3;i++){  
 data[i] = (function(k){  
 return function(){  
 **console**.log(k);  
 }  
 })(i);

**构造函数和对象**

function JuiceMachine(fruit){  
 this.**fruit** = fruit;  
 this.produceJuice = function(){  
 **console**.log(this.**fruit** + '果汁');  
 }  
}  
  
//不可以直接调用  
JuiceMachine.produceJuice();  
  
//这里的juiceMachine是一个对象了，可以有商标一类的  
 JuiceMachine.**brand** = '九阳';  
 JuiceMachine.open = function(){  
 **console**.log('open');  
 }  
  
// new 之后形成的对象与本身无关，  
 var apple = new JuiceMachine("苹果");  
 apple.produceJuice();  
 var brand = apple.brand;//这里是错的  
  
  
//真确的方法  
 var orange = new JuiceMachine("橘子");  
 orange.produceJuice();  
  
  
//结论：function有双重身份：  
//1、对象  
//2、构造函数

jQuery中的Ajax

jQuery中的Ajax是基于XMLHttpRequest进行的封装，通过灵活的参数配置，可以极大的提高开发效率。

一个典型jQuery Ajax实例

jQuery Ajax API



例子

Php操作

数组转化成json datatype:”json”,

$arr = array("username"=>"zhangsan","password"=>"456");  
echo json\_encode($arr);

处理

//data.name

直接返回tag ，这是datatype：”html”

$tag = '<div>水果</div><div>水果</div><div>水果</div><div>水果</div><div>水果</div><div>水果</div><div>水果</div><div>水果</div>';  
echo $tag;

处理

/$("#showInfo").html(data);

返回script

**dataType**:"script",

$script = 'alert(123);';  
echo $script;

处理

Eval（data）

返回文本 datatype：’text’

$text = 'hello world';  
echo $text;

Ajax部分

**$**(function(){  
  
 **$**("#btn").click(function(){  
 **$**.ajax({  
 **url**:"04data.php",  
 **dataType**:"script",  
 **type**:"get",  
 success:function(data){  
 },  
 error:function(e){  
 **console**.log(e);  
 }  
 });  
 });  
});

常见Ajax插件

Nprogress.js

同源策略与跨域

**所谓同源是指，域名，协议，端口完全相同**

例如<http://www.example.com/> 前面的东西完全一样

|  |  |  |
| --- | --- | --- |
| http://api.example.com/detail.html | 不同源 | 域名不同 |
| https//www.example.com/detail.html | 不同源 | 协议不同 |
| http://www.example.com:8080/detail.html | 不同源 | 端口不同 |
| http://api.example.com:8080/detail.html | 不同源 | 域名、端口不同 |
| https://api.example.com/detail.html | 不同源 | 协议、域名不同 |
| https://www.example.com:8080/detail.html | 不同源 | 端口、协议不同 |
| http://www.example.com/detail/index.html | 同源 | 只是目录不同 |

http://api.example.com/detail.html 和 http://www.example.com/index.html 通过domain.name来解决，即同时设置 domain.name = example.com

跨域解决方案

document.domain + **iframe**

**jsonp**

**服务器响应头：添加属性（允许访问的域名）**

JSONP

跨域问题的解决方案，不能范访问不同域名

**由事件触发创建一个script。其中src中定义要访问的地址，并且添加属性键值对，**

**&\_jsonp=functionName 来触发本地的函数解析结果（返回的数据自动放到函数中了）**

**window**.onload = function(){  
 var city = **document**.getElementById('city');  
 city.onchange = function(){  
 **document**.getElementById('info').**innerHTML** = '';  
 }  
 var btn = **document**.getElementById('btn');  
   
 btn.onclick = function(){  
 var cityCode = city.**value**;  
 var url = 'http://cdn.weather.hao.360.cn/api\_weather\_info.php?app=hao360&\_jsonp=abc&code='+city.**value**;  
 var script = **document**.createElement('script');  
 script.**src** = url;  
 **document**.**body**.appendChild(script);  
 }  
  
}

function abc(data){  
 var d = data.weather;  
 var info = **document**.getElementById('info');  
 info.**innerHTML** = '';  
   
 for(var i=0;i<d.**length**;i++){  
 var date = d[i].date;  
 var day = d[i].info.day;  
 var night = d[i].info.night;  
 var tag = '';  
 tag += '<span>日期：'+date+'</sapn><ul>';  
 tag += '<li>白天天气：'+day[1]+'</li>'  
 tag += '<li>白天温度：'+day[2]+'</li>'  
 tag += '<li>白天风向：'+day[3]+'</li>'  
 tag += '<li>白天风速：'+day[4]+'</li>'  
 tag += '</ul>';  
  
 tag += '<ul>';  
 tag += '<li>夜间天气：'+night[1]+'</li>'  
 tag += '<li>夜间温度：'+night[2]+'</li>'  
 tag += '<li>夜间风向：'+night[3]+'</li>'  
 tag += '<li>夜间风速：'+night[4]+'</li>'  
 tag += '</ul>';  
 var div = **document**.createElement('div');  
 div.**innerHTML** = tag;  
 info.appendChild(div);  
   
 }  
**}**

对应的php程序

<?php   
$callback = $\_GET['\_jsonp'];  
$arr = array("zhangsan","lisi","zhaoliu");  
echo $callback."(".json\_encode($arr).")";  
 ?>

Access-Control-Allow-Origin

服务器代理

jQuery中对JSONP的使用

**$**(function(){  
  
 **$**.ajax({  
 **type** : "get",  
 **async**: false,  
 **url** : "./jsonp.php",  
 **dataType** : "jsonp",  
 //jsonp: "qwe",//传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(默认为:callback)  
 //jsonpCallback:"liudehua",//自定义的jsonp回调函数名称，默认为jQuery自动生成的随机函数名(类似：jQuery1102016820125747472048\_1450147653563(["zhangsan", "lisi", "wangwu"]);)

//只制定datatype；不写jsonp 和jasoncallback，可以直接用success就可以了  
 success : function(data){  
 **console**.log(data);  
 },  
 error:function(){  
 **console**.log('fail');  
 }  
 });  
  
});

模板引擎

Lacal软件

原理剖析

流行模板引擎

artTemplate、JsRender、baiduTemplate、Mustache、Hanldebars

artTemplate

查询天气，需要先查询城市编码一个ajax，网络查编码（）

然后在 由城市编码查另外的城市天气（另外的编码）

用id标注编码的位置，数据在后面传入

例子

**数组**

var data = {  
 **title**: '基本例子',  
 **isAdmin**: true,  
 **list**: ['文艺', '博客', '摄影', '电影', '民谣', '旅行', '吉他']  
};  
var html = template('test', data);  
**document**.getElementById('content').**innerHTML** = html;

<script id="test" type="text/html">  
{{if isAdmin}}  
  
<h1>{{title}}</h1>  
<ul>  
 {{each list as value i}}  
 <li>索引 {{i + 1}} ：{{value}}</li>  
 {{/each}}  
</ul>  
{{/if}}  
</script>

**处理时间**

var data = {  
 **time**: 1408536771253,  
};  
var html = template('test', data);  
**document**.getElementById('content').**innerHTML** = html;

<script id="test" type="text/html">  
{{time | dateFormat:'yyyy年 MM月 dd日 hh:mm:ss'}}

</script>

没有数据的时候讲返回渲染函数

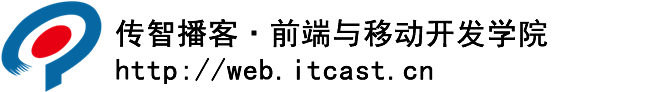
var source = '<ul>'  
 + '{{each list as value i}}'  
 + '<li>索引 {{i + 1}} ：{{value}}</li>'  
 + '{{/each}}'  
 + '</ul>';  
var render = template.compile(source);  
var html = render({  
list: ['摄影', '电影', '民谣', '旅行', '吉他']  
});  
document.getElementById('content').innerHTML = html;

百度搜索

<script type="text/javascript">  
 **$**(function(){  
 /\*输入完成注册keyup事件\*/  
 **$**('#keyWord').keyup(function(e){  
 var kd = **$**('#keyWord').val();  
 /\*百度搜索用的借口地址\*/  
 var url ='https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd='+kd;  
 querySUG(url);  
 });  
 });  
  
 function querySUG(url){  
 **document**.getElementById('list').**innerHTML** = '';  
// 异步的jsonp  
 **$**.ajax({  
 **type** : "get",  
 **async**: true,  
 **url** : url,  
 **dataType** : "jsonp",  
 **jsonp**: "cb",//传递给请求处理程序或页面的，用以获得jsonp回调函数名的参数名(默认为:callback)  
 // **jsonpCallback**:"callback",//自定义的jsonp回调函数名称，默认为jQuery自动生成的随机函数名(类似：//jsonp一般不可省略，callback一般可以省略

jQuery1102016820125747472048\_1450147653563(["zhangsan", "lisi", "wangwu"]);)  
 success : function(data){  
 var tag = '<ul>';  
 for(var i=0;i<data.**s**.**length**;i++){  
 tag += '<li>'+data.**s**[i]+'</li>';  
 }  
 tag += '</ul>';  
 **$**('#list').html(tag).show();  
 **$**('#list').find('li').hover(function(){  
 **$**(this).css('background','lightGreen');  
 },function(){  
 **$**(this).css('background','lightGray');  
 });  
 },  
 error:function(){  
 **console**.log('fail');  
 }  
 });  
 }  
</script>

移动端开发



**移动web开发《京东商城》**

### 淘宝的viewport设置

window.devicePixelRatio

屏幕像素比

物理像素 最小的像素点，最小的颗粒度，

屏幕像素比：表示1px长度需要多少个物理点

物理点显示1px长度需要，2个像素点，

物理像素 iphone 4 像素比是2 640 \* 960 >> 320\*480

### 京东移动站构建

## 首页

至少要设置宽度是640px以上，

### 顶部搜索



<**form action="#"**>  
 *<!--定义能调出搜索按钮的方式-->* <**input type="search"**/>  
</**form**>

设置搜索按钮

### 复习

1 非主流的一个 viewport 设置

淘宝 根据屏幕像素比

2 设计图 最少是640px

3. base。Css

Box-sizing 在百分比布局

4 300-640

640 主流的企业 640px 设计图

300 考虑有一些老设备 比300

5 搜索

6 banner

### 导航栏

导航栏 分为 8个区域 分别左浮动在 nav 容器内部。

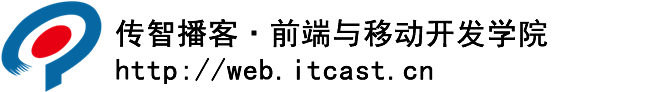
### 秒杀区块





用伪类来写





**移动web开发《京东商城》**

## 复习

### 三种布局

**有最大、最小宽度的百分比自适应布局**

适用场景：门户网站首页，图片较多的首页。

**百分比自适应布局**

适用场景：信息文字较多的网页，内容较多网页。

**全屏自适应布局**

适用场景：单页面网页，移动web app 页面。

### Box-sizing在移动端的使用

在百分比定宽的页面经常使用。

\*,  
::before,  
::after{-webkit-box-sizing: border-box;

*/\*以你的border开始计算你的宽度\*/*}

## 移动端事件

Touch

touchstart：当手指触碰屏幕时候发生。不管当前有多少只手指

touchmove：当手指在屏幕上滑动时连续触发。通常我们再滑屏页面，

会调用event的preventDefault()可以阻止默认情况的发生：阻止页面滚动

touchend：当手指离开屏幕时触发

touchcancel：系统停止跟踪触摸时候会触发。

例如在触摸过程中突然页面alert()一个提示框，此时会触发该事件，这个事件比较少用

**触摸事件的响应顺序**

1、ontouchstart

2、ontouchmove

3、ontouchend

4、onclick 300ms延时

**Event**

originalEvent （原生事件） 是jquery 封装的事件。

targetTouches 目标元素的所有当前触摸

changedTouches 页面上最新更改的所有触摸

touches 页面上的所有触摸

clientX、clientY 相对于当前屏幕的X或Y位置

pageX、pageY 相对于整体页面的X或Y位置

transitionEnd 过渡结束事件。

animationEnd 动画结束事件

移动端事件框架

例如 zepto touch.js

'swipe',

'swipeLeft',

'swipeRight',

'swipeUp',

'swipeDown',  
 'doubleTap',

'tap',

'singleTap',

'longTap'

都是由我们的原生touch事件封装的。

## 常见的移动端问题

### 什么是Retina 显示屏，带来了什么问题

retina：一种具备超高像素密度的液晶屏，同样大小的屏幕上显示的像素点由1个变为多个，如在同样带下的屏幕上，苹果设备的retina显示屏中，像素点1个变为4个

在高清显示屏中的位图被放大，图片会变得模糊，**因此移动端的视觉稿通常会设计为传统PC的2倍**。

那么，前端的应对方案是：

设计稿切出来的图片长宽保证为偶数，并使用backgroud-size把图片缩小为原来的1/2

//例如图片宽高为：200px\*200px，那么写法如下

.css{width:100px;height:100px;background-size:100px 100px;}

其它元素的取值为原来的1/2，例如视觉稿40px的字体，使用样式的写法为20px

.css{font-size:20px}

### 百度禁止转码

通过百度手机打开网页时，百度可能会对你的网页进行转码，往你页面贴上它的广告，非常之恶心。不过我们可以通过这个meta标签来禁止它：

<meta http-equiv="Cache-Control" content="no-siteapp" />

### 移动端手机号码识别（IOS）

在 iOS Safari （其他浏览器和Android均不会）上会对那些看起来像是电话号码的数字处理为电话链接，比如：

* 7位数字，形如：1234567
* 带括号及加号的数字，形如：(+86)123456789
* 双连接线的数字，形如：00-00-00111
* 11位数字，形如：13800138000

**可能还有其他类型的数字也会被识别。我们可以通过如下的meta来关闭电话号码的自动识别：**

<meta name="format-detection" content="telephone=no" />

**开启电话功能**

<a href="tel:123456">123456</a>

**开启短信功能：**

<a href="sms:123456">123456</a>

### 移动端邮箱识别（Android）

与电话号码的识别一样，在安卓上会对符合邮箱格式的字符串进行识别，我们可以通过如下的meta来管别邮箱的自动识别：

<meta content="email=no" name="format-detection" />

同样地，我们也可以通过标签属性来开启长按邮箱地址弹出邮件发送的功能：

<a mailto:dooyoe@gmail.com">dooyoe@gmail.com</a>

### ios系统中元素被触摸时产生的半透明灰色遮罩怎么去掉

ios用户点击一个链接，会出现一个半透明灰色遮罩, 如果想要禁用，可设置-webkit-tap-highlight-color的alpha值为0，也就是属性值的最后一位设置为0就可以去除半透明灰色遮罩。

{-webkit-tap-highlight-color: rgba(0,0,0,0;)}

### webkit表单元素的默认外观怎么重置

.css{-webkit-appearance:none;}

### 禁止文本缩放

当移动设备横竖屏切换时，文本的大小会重新计算（或者在内嵌浏览器中设置字体的大小），进行相应的缩放，当我们不需要这种情况时，可以选择禁止：

html {

　　-webkit-text-size-adjust: 100%;

}

### 移动端禁止选中内容

如果你不想用户可以选中页面中的内容，那么你可以在css中禁掉：

.user-select-none {

-webkit-user-select: none;

｝

### 如何禁止保存或拷贝图像（IOS）

通常当你在手机或者pad上长按图像 img ，会弹出选项存储图像 或者拷贝图像，如果你不想让用户这么操作，那么你可以通过以下方法来禁止：

img { -webkit-touch-callout: none; }

### 摇一摇功能

HTML5 deviceMotion：封装了运动传感器数据的事件，可以获取手机运动状态下的运动加速度等数据。

### android 上去掉语音输入按钮

input::-webkit-input-speech-button {display: none}

### 移动端基础框架

[**zepto.js**](http://zeptojs.com/)语法与jquery几乎一样，会jquery基本会zepto；

搭配

backbone underscore

<http://daneden.github.io/animate.css/> 动画css