# HTML

**一个完整的HTML文件，应该至少包含html元素，body元素，以及里面的内容**

中文乱码问题，在html的最前面加上编码设置

<head>

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

</head>

注释格式用<! This is a comment >//即括号加一个感叹号

## 标签

#### <!DOCTYPE>

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

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

提示：请始终向 HTML 文档添加 <!DOCTYPE> 声明，这样浏览器才能获知文档类型。

#### <a>

属性：**最重要的属性是 href 属性，它指示链接的目标。**

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。

rel text 规定当前文档与被链接文档之间的关系。

target属性规定在何处打开被链接文档：

\_self：默认值，在当前窗口打开URL

\_blank：在一个新的窗口中打开URL

\_parent：在父窗口中打开URL

\_top：在顶层窗口中打开URL

某个iframe的name值：在某个frame中打开URL

去掉下划线可用css里的text-decoration:none;

## 框架

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面。每份HTML文档称为一个框架，并且每个框架都独立于其他的框架。每一个window对应一个框架

fra框架结构标签（<frameset>）定义如何将窗口分割为框架

每个 frameset 定义了一系列行或列

### frame

Frame 标签定义了放置在每个框架中的 HTML 文档。

不能将 <body></body> 标签与 <frameset></frameset> 标签同时使用

html5不支持该功能

### iframe

iframe 元素会创建包含另外一个文档的内联框架（即行内框架）。

必须放在body中间

## 元素

<html> HTML <html> 元素 表示一个HTML文档的根（顶级元素），所以它也被称为根元素。所有其他元素必须是此元素的后代。

<head> HTML head 元素 规定文档相关的配置信息（元数据），包括文档的标题，引用的文档样式和脚本等。

<body> 元素定义了 HTML 文档的主体。这个元素拥有一个开始标签 <body>，以及一个结束标签 </body>。

<link> HTML外部资源链接元素 (<link>) 规定了当前文档与外部资源的关系。该元素最常用于链接样式表，此外也可以被用来创建站点图标(比如PC端的“favicon”图标和移动设备上用以显示在主屏幕的图标) 。

<meta> HTML <meta> 元素表示那些不能由其它 HTML 元相关（meta-related）元素（(base、link, script、style 或 title）之一表示的任何Metadata信息。

<style> HTML的<style>元素包含文档的样式信息或者文档的部分内容。默认情况下，该标签的样式信息通常是CSS的格式。

<title> HTML <title> 元素 定义文档的标题，显示在Browser的标题栏或标签页上。它只应该包含文本，若是包含有标签，则它包含的任何标签都将被忽略。

<nav> HTML <nav>元素表示页面的一部分，其目的是在当前文档或其他文档中提供导航链接。导航部分的常见示例是菜单，目录和索引。

### 文本内容

<li> HTML <li> 元素 （或称 HTML 列表条目元素） 用于表示列表里的条目。它必须包含在一个父元素里：一个有序列表(ol)，一个无序列表(ul)，或者一个菜单 (menu)。在菜单或者无序列表里，列表条目通常用点排列显示；在有序列表里，列表条目通常在左边显示按升序排列的计数，例如数字或者字母。

<div> HTML <div> 元素 (或 HTML 文档分区元素) 是一个通用型的流内容容器，在不使用CSS的情况下，其对内容或布局没有任何影响。

<ol> HTML <ol> 元素表示有序列表，通常渲染为一个带编号的列表。

### 内联文本语义

**使用 HTML 内联文本语义（Inline text semantics）定义一个单词、一行内容，或任意文字的语义、结构或样式。**

<a> HTML <a> 元素（或称锚元素）可以创建通向其他网页、文件、同一页面内的位置、电子邮件地址或任何其他 URL 的超链接。

<b> HTML提醒注意（Bring Attention To）元素（<b>）用于吸引读者的注意到该元素的内容上（如果没有另加特别强调）。这个元素过去被认为是粗体（Boldface）元素，并且大多数浏览器仍然将文字显示为粗体。尽管如此，你不应将 <b> 元素用于显示粗体文字；替代方案是使用 CSS font-weight 属性来创建粗体文字。

<br> HTML <br> 元素在文本中生成一个换行（回车）符号。此元素在写诗和地址时很有用，这些地方的换行都非常重要。

<code> HTML <code> 元素呈现一段计算机代码. 默认情况下, 它以浏览器的默认等宽字体显示.

<i> HTML元素 <i> 用于表现因某些原因需要区分普通文本的一系列文本。例如技术术语、外文短语或是小说中人物的思想活动等，它的内容通常以斜体显示。

<span> HTML <span> 元素是短语内容的通用行内容器，并没有任何特殊语义。可以使用它来编组元素以达到某种样式意图（通过使用类或者Id属性），或者这些元素有着共同的属性，比如lang。应该在没有其他合适的语义元素时才使用它。<span> 与 div 元素很相似，但 div 是一个 块元素 而 <span> 则是 行内元素 .

<strong> Strong 元素 (<strong>)表示文本十分重要，一般用粗体显示。

### 图片和多媒体

<img> HTML <img> 元素将一份图像嵌入文档

<video> HTML <video> 元素 用于在HTML或者XHTML文档中嵌入媒体播放器，用于支持文档内的视频播放。

<audio> HTML <audio> 元素用于在文档中嵌入音频内容。 <audio> 元素可以包含一个或多个音频资源， 这些音频资源可以使用 src 属性或者source 元素来进行描述：浏览器将会选择最合适的一个来使用。也可以使用 MediaStream 将这个元素用于流式媒体。

### 内嵌内容

### frame框架

### iframe框架

### 表格内容

### 表单

<button> HTML <button> 元素表示一个可点击的按钮，可以用在表单或文档其它需要使用简单标准按钮的地方。

<form> HTML <form> 元素表示文档中的一个区域，此区域包含交互控件，用于向 Web 服务器提交信息。

<input> HTML <input> 元素用于为基于Web的表单创建交互式控件，以便接受来自用户的数据; 可以使用各种类型的输入数据和控件小部件，具体取决于设备和user agent。

<label> HTML <label> 元素（标签）表示用户界面中某个元素的说明。

<output> HTML <output> 标签表示计算或用户操作的结果。

HTML 提示：使用小写标签

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。

## 旧约

属性用来修饰标签的，比如要设置一个标题居中

<h1 align="center">居中标题</h1>

写在开始标签里的 align="center" 就叫属性，align 是属性名，center 是属性值

属性值应该使用双引号括起来

html使用<!--输入内容 --> 进行注释

标题会自动粗体，大写其中的内容，并且带有换行效果

一般会使用<h1> 到 <h6> 分别表示不同大小的标题

<h1>标题1</h1>

<h2>标题2</h2>

<h3>标题3</h3>

<h4>标题4</h4>

<h5>标题5</h5>

<h6>标题5</h6>

<p>标签即表示段落是paragraph的缩写，自带换行效果.。在p标签中的文本会自动换行，不在p标签中的，不会自动换行

<b><strong>都可以用来实现粗体的效果

区别：

b是bold的缩写，仅仅表示该文本是粗体的，并不暗示这段文字的重要性

strong虽然也是粗体，但是更多的是强调语义上的加重，提醒用户该文本的重要性。 在SEO（搜索引擎优化）的时候，也更加容易帮助用户找到重点的内容

推荐使用strong

特殊元素：

<**br**/>表示换行

<**hr**/>表示一行水平线

<i>和<em>都可以表示斜体效果

区别：

i是italic的缩写，仅仅表示该文本是斜体的，并不暗示这段文字的重要性

em 是 Emphasized的缩写，虽然也是斜体，但是更多的是强调语义上的加重，提醒用户该文本的重要性。 常常用于引入新的术语的时候使用。

通过嵌套实现多种效果

嵌套即标签中有标签

<**strong**><**i**>同时有粗体和斜体</**i**></**strong**>

有时候，需要在网页上显示代码，比如java代码就需要用到pre

<del>即删除标签delete的缩写

注：<s>也有删除效果，但很多浏览器不支持。

<ins>即下划线标签

**图像**引用

若是同一目录下图片<**img** src="example.gif"/>

若是上上一级目录下图片<**img** src="../../example.gif"/>

若使用绝对路径在前面加上file://<**img** src="file://c:/example.gif"/>

图像大小

如果设置的大小比原图片大，则会产生失真效果

<**img** width="200" height="200" src="https://how2j.cn/example.gif"/>

图像居中

img不能够自己居中，需要放在其他能够居中的标签中实现这个效果，比如h1标签,p标签.  
经常采用的手段是放在div中居中

<**div** align="center">

 <**img** src="example.gif"/>

如果图片不存在，默认会显示一个缺失图片，这是不友好的。所以可以加上alt属性。

当图片存在的时候，alt是不会显示的，当图片不存在的时候，alt就会出现

<**img** src="example\_not\_exist.gif" alt="这个是一个图片" />

**超链**

<a>标签即用来显示超链

完整元素是<a href="跳转到的页面地址">超链显示文本</a>

在新的页面打开超链

新增属性target

<**a** href="http://www.12306.cn" target="\_blank">http://www.12306.cn</**a**>

超链上的提示文字

当鼠标放在超链上的时候，就会弹出提示文字

<**a** href="http://www.12306.com"

 title="跳转到http://www.12306.com">www.12306.com</**a**>

使用图片作为超链

<**a** href="http://www.12306.com">

<**img** src="https://how2j.cn/example.gif"/>

</**a**>

**表格**

**<table>标签用于显示一个表格  
<tr> 表示行  
<td> 表示列又叫单元格**

例：两行两列

<**table**>

  <**tr**>

      <**td**>1</**td**>

      <**td**>2</**td**>

  </**tr**>

  <**tr**>

      <**td**>3</**td**>

      <**td**>4</**td**>

  </**tr**>

 </**table**>

带边框的表格，设置table的属性border

<**table** border="1">

设置table宽度通过设置table的属性 width，px即像素的意思。

<**table** border="1" width="200px">

单元格宽度绝对值通过设置td的属性width，同一列自动继承该属性，另一列由table宽度和1单元格的宽度之差的平均值决定

 <**td** width="50px">1</**td**>

单元格宽度相对值通过设置td的属性width为百分数，该百分数为td占table的百分数

 <**td** width="80%">1</**td**>

单元格水平对齐用设置td的属性align

<**td** width="50%" align="lenter/right">1</**td**>

单元格垂直对齐用设置td的属性valign

<**td** width="50%" valign="top/middle/bottom" >1</**td**>

水平合并用设置td的属性colspan

横跨两行, 垂直合并用设置td的属性rowspan

<**td** rowspan="2">1,3</**td**>

背景色用设置tr或者td的属性bgcolor，以td属性决定最后效果

<**tr** bgcolor="gray">

<**td**  bgcolor="pink">b</**td**>

列表

**列表分无序列表和有序列表**

分别用<ul>标签和<ol>标签表示，有序列表自带顺序编号

<**ul**>

<**li**>DOTA</**li**>

<**li**>LOL</**li**>

</**ul**>

**<div>，<span>这两种标签都是布局用的，**这种标签本身没有任何显示效果，通常是用来结合css进行页面布局

div和span的区别

div是块元素，即自动换行。常见的块元素还有h1,table,p

span是内联元素，即不会换行。常见的内联元素还有img,a,b,strong

**使用<font>标签表示字体**

font常用的属性有 color和size, 分别表示颜色和大小

<**font** color="green">绿色默认大小字体</**font**>

<**br**>

<**font** color="blue" size="+2">蓝色大2号字体</**font**>

<**br**>

<**font** color="red" size="-2">红色小2号字体</**font**>

在html中，颜色通常使用两种方式来表示：

1. 颜色名，比如red, blue

2. 颜色对应的16进制，比如#ff0000 就表示红色

**<iframe> 即内联框架，通过内联框架 可以实现在网页中“插入”网页**

相当于浏览器里面有个小浏览器，在这个小浏览器中，打开另一个网页

**<input type="text"> 即表示文本框**

并且只能够输入一行

如果要输入多行

使用文本域<textarea>

用size属性设置文本框长度

用属性value设置文本框初始文字

**用属性placeholder设置文本框背景文字**

<input type="password"> 即表示密码框

输入的数据会自动显示为星号

**表单form**

action="/study/login.jsp" 表示把账号和密码提交到login.jsp这个页面去

例：

<**form** action="https://how2j.cn/study/login.jsp">

账号：<**input** type="text" name="name"> <**br**/>

密码：<**input** type="password" name="password" > <**br**/>

<**input** type="submit" value="登陆">

</**form**>

**使用method="get" 提交数据 是常用的提交数据的方式**

如果form元素没有提供method属性，**默认就是get方式提交数据**

get方式的一个特点就是，可以在浏览器的地址栏看到提交的参数，即便是密码也看得到

<**form** method="get" action="https://how2j.cn/study/login.jsp">

使用method="post" 也可以提交数据

post不会在地址栏显示提交的参数

如果要提交二进制数据，比如上传文件，必须采用post方式

get和post的区别  
get是form默认的提交方式  
如果通过一个超链访问某个地址，是get方式；如果在地址栏直接输入某个地址，是get方式。提交数据会在浏览器显示出来，不可以用于提交二进制数据，比如上传文件  
post必须在form上通过 method="post" 显示指定  
提交数据不会在浏览器显示出来，可以用于提交二进制数据，比如上传文件

**<input type="radio" > 表示单选框**

//此时两选项可以同时选，但选上还需设置可取消

单选1 <input type="radio" >

单选2 <input type="radio" >

设置默认选中<**input** type="radio" checked="checked" >

分组即，多个单选框，都在一个分组里，**同一时间，只能选中一个单选框**

将多个单选框的name属性设置相同即可

<**input** type="radio" name="sth" checked="checked" >

**<input type="checkbox"> 即表示复选框**

//即使name属性相同也不影响多选，可直接取消

**下拉列表**

<select> 即下拉列表，需要配合<option>使用

<**select** >

 <**option** >苍老师</**option**>

 <**option** >高树玛利亚</**option**>

 <**option** >遥美</**option**>

</**select**>

属性size设置下拉列表高度

<**select** size="3" multiple="multiple">

使用多选，用ctrl或者shift进行多选

对option元素设置selected="selected" 属性，默认选中该option

用style属性设置选项宽度style="width: 60px;"

**<textarea> 即文本域**

与文本框不同的是，文本域可以有多行，并且可以有滚动条

<**textarea** cols="30" rows="8">

使用属性cols(每行可容纳的字符)和rows设置宽度和行数

**按钮**

<input type="button"> 即普通按钮，普通按钮不具备提交form的效果

<input type="submit"> 即为提交按钮，用于提交form，把数据提交到服务端

<input type="reset"> 重置按钮 可以把输入框的改动复原

<input type="image" > 即使用图像作为按钮进行form的提交

<button></button>即按钮标签

与<input type="button">不同的是，<button>标签功能更为丰富；按钮标签里的内容可以是文字也可以是图像；如果button的type=“submit” ，那么它就具备提交form的功能。此时用图片提交则内容为图片，type为submit。

IE下button的type的默认值为button不具备提交功能，其他浏览器type的默认值是submit

## 新约

data属性

data-\* 属性用于存储页面或应用程序的私有自定义数据。

data-\* 属性赋予我们在所有 HTML 元素上嵌入自定义 data 属性的能力。

存储的（自定义）数据能够被页面的 JavaScript 中利用，以创建更好的用户体验（不进行 Ajax 调用或服务器端数据库查询）。

data-\* 属性包括两部分：

属性名不应该包含任何大写字母，并且在前缀 "data-" 之后必须有至少一个字符

属性值可以是任意字符串

<element data-\*="somevalue">

# CSS

css把和颜色，大小位置等信息剥离到<style>中，而html只用关心提供什么样的内容就行了。

**css的语法**

**selector {property: value}**

**即 选择器{属性:值}**

学习css即学习有哪些选择器，哪些属性以及可以使用什么样的值

<**style**>

p{

   color:red;

}

</**style**>

//设置所有p元素为红色字体

<**p** style="color:red">这是style为红色的</**p**>

//也可以对单独的p元素的style属性赋值

**选择器主要分3种：元素选择器；id选择器；类选择器**

元素选择器通过标签名选择元素

在例子中，所有的p都被设置成红色

通过id选择元素(给不同元素赋予单独的id属性定位)

注： 一个元素的id应该是唯一的。另一个元素不应该重复使用

<style>

p{

color:red;

}

**#p1{**

**color:blue;**

**}**

#p2{

color:green;

}

</style>

<p>没有id的p</p>

<p **id="p1"**>id=p1的p</p>

<p id="p2">id=p2的p</p>

**类选择器**

当需要多个元素，都使用同样的css的时候，就会使用类选择器

对类进行css处理，再对每个元素赋予相应的类属性

.pre{

  color:blue;

}

<**p** class="pre">css处理过的蓝色类字体</**p**>

**更准确的选择**

同时根据元素名和class来选择  
<**style**>

**p.blue**{

  color:blue;

}

</style>

//类为blue且元素为p的才变为蓝色

**优先级：元素内style > 类选择器 > ID选择器 > 元素选择器**

**style样式中的注释**

以/\* 开始，以\*/结束，被注释掉的文字会自动隐藏

## 背景及尺寸

**大小属性**：width，height；值：可以是百分比或者像素

**背景色**：background-color；值有预定义的颜色名字，rgb格式，16进制三种方式

**背景图片**： background-image:url(/study/background.jpg);、

**背景重复**：background-repeat属性，将设置好的部分背景一直重复直至完全占满背景

其值有repeat; 水平垂直方向都重复  
repeat-x; 只有水平方向重复  
repeat-y; 只有垂直方向重复  
no-repeat; 无重复

**背景平铺**：属性：background-size；值：contain

、

### 文本

**文字颜色**：属性名color

**文字对齐**：属性:text-align；值：left,right,center

**div是块级元素，其默认宽度是100%，所以文本有对齐的空间前提。**

**span是内联元素其默认宽度就是其文本内容的宽度，文本已经占满空间**

**文本修饰：**属性：text-decoration；

值： overline（上划线）；line-through（中划线）；underline（下划线）；blink（闪烁效果，大部分浏览器已取消）;none（去掉划线，对超链默认的划线有用）

**行间距**：属性：line-height；值：数字或者百分比

**字符间距**：属性：letter-spacing；值： 数字

**单词间距**：属性：word-spacing；值： 数字//即改变单个空格长短实现

**首行缩进**：属性：text-indent值： 数字

**大小写**：属性：text-transform

值：uppercase 全部大写；capitalize 首字母大写；lowercase 全部小写

**空白格**：属性：white-space  
值：  
normal 默认。多个空白格或者换行符会被合并成一个空白格  
pre 有多少空白格，显示多少空白格，相当于[pre标签](https://how2j.cn/k/html/html-pre/182.html),如果长度超出父容器也不会换行。  
pre-wrap 有多少空白格，显示多少空白格，相当于[pre标签](https://how2j.cn/k/html/html-pre/182.html),如果长度超出父容器，会换行。  
nowrap 一直不换行，直到使用<br/>。

### 字体

**字体尺寸**：属性：font-size值：数字或者百分比或数字+em（1em为标准大小）

**风格**：属性font-style:值：normal 标准字体；italic 斜体

**字体粗细**：属性 font-weight值：normal 标准粗细；bold 粗一点

**字体种类**：属性font-family

默认字库 font family:default

设置字库 font-family: Times New Roman

设置字库 font-family: Arial

设置字库 font-family: 宋体, 这种字体是IE默认字体

设置字库 font-family: 黑体

设置字库 font-family: 楷体

设置字库 font-family: 微软雅黑, 这个字体是火狐默认字体

**把大小，风格，粗细，字库都写在一行里面**：顺序依次为：1.风格(normal italic) 2.粗细(normal bold) 3.尺寸 4.字库

不按照这个顺序就没办法显示想要的效果

**font:**italic bold 30px "Times New Roman"

**鼠标样式**：元素span，属性cursor

值与样式：default,默认鼠标样式；auto，文本框选样式；crosshair，十字准星样式；

Pointer，点击样式；text，文本框选样式；wait，加载样式；help，疑问样式；not-allowed，禁止样式；

**表格**

表格布局：属性：table-layout

值：automatic，单元格的大小由td的内容宽度决定；fixed，单元格的大小由td上设置的宽度决定。

注：只对连续的英文字母起作用，如果使用中文就看不到效果

即前者单元格最小恰好能框住td内容，后者单元格过小td内容会超出单元格。

表格边框：

属性：border-collapse

值：separate:边框分隔；collapse:边框合并

## 边框：

**边框风格**：属性： border-style  
值：solid: 实线  
dotted:点状  
dashed:虚线  
double:双线

**边框颜色**：属性：border-color值：red,#ff0000,rgb(255,0,0)

**边框宽度**：属性：border-width；值：数字

**边框总和**：属性：border；值：颜色 风格 宽度

通过制定位置，可以只给一个方向设置边框风格，颜色和宽度

border**-top**-style:solid;

border**-top**-color:red;

border**-top**-width: 50px;

top,bottom,left,right 分别对应上下左右

它们的交界同时出现边框的时候，就会以倾斜的形式表现交界线。

可以看到，块级元素div默认是占用100%的宽度  
**常见的块级元素有div h1~h5 p li ol ul** 等  
而内联元素span的宽度由其内容的宽度决定  
**常见的内联元素有 a b strong i input** 等

**元素内边距**

指的是元素里的内容与边框之间的距离

属性：

padding-left: 左内边距

padding-right: 右内边距

padding-top: 上内边距

padding-bottom: 下内边距

padding: 上 右 下 左

//写1个值时默认上下左右全为该值，否则按照上 右 下 左,依顺时针的方向依次赋值。如果缺少左内边距的值，则使用右内边距的值。

如果缺少下内边距的值，则使用上内边距的值。

如果缺少右内边距的值，则使用上内边距的值。

值：数字

**元素外边距**

指的是元素边框和元素边框之间的距离

属性：

margin-left: 左外边距

margin-right: 右外边距

margin-top: 上外边距

margin-bottom: 下外边距

注：像span这样的内联元素，默认情况下，只有左右外边距，没有上下外边距。 为了观察上下外边距的效果，可以采用块级元素，比如div.

**边框模型打包**

.box{

width:70px;

padding:5px;

margin: 10px;

}

**超链状态**

伪类，所谓的伪类即被选中的元素处于某种状态的时候

超链状态有4种

link - 初始状态，从未被访问过

visited - 已访问过

hover - 鼠标悬停于超链的上方

active - 鼠标左键点击下去，但是尚未弹起的时候

<**style**>

a:link {color: #FF0000}

a:visited {color: #00FF00}

a:hover {color: #FF00FF}

a:active {color: #0000FF}

</**style**>

**去除超链的下划线**

默认状态下，超链是有下划线的，但是现在网站上的超链普遍采用无下划线风格。

使用 text-decoration: none 文本修饰的样式来解决

**隐藏元素**

隐藏元素有两种方式

使用display:none; 隐藏一个元素，这个元素将不再占有原空间 “坑” 让出来了

使用 visibility:hidden;隐藏一个元素，这个元素继续占有原空间，只是“看不见”

## CSS文件

如果把所有的css都写在html文件里面，一旦样式比较多的时候，就会显得不易维护

这个时候就会选择把所有的css内容，放在一个独立文件里

然后在html中引用该文件

通常这个文件会被命名为style.css

并在html中引用该文件

<**link** rel="stylesheet" type="text/css" href="https://how2j.cn/study/style.css" />

//引用how2j的style.css文件

注：style.css文件里，就不要再使用style标签了

当html内部style标签与外部文件style.css样式重复属性矛盾时，内容由最后出现的决定

如果样式上增加了**!important**，则**优先级最高**，甚至高于style属性

<**style**>

.p1{

  color:green !important;

}

</**style**>

## 定位

### 绝对定位

属性：position；值： absolute

通过指定left,top绝对定位一个元素

设置了绝对定位的元素，相当于该元素被从原文档中删除了然后新建了一个有该元素的图层。

**绝对定位定位的是父容器的位置，若没有则默认body为父容器**

通过绝对定位可以把一个元素放在另一个元素上，这样位置就重复了。

此时用z-index属性， 当z-index的值越大，就表示图层在越上面，z-index:越小就表示图层在越下面。Body主体z-index值默认为0。

### 相对定位

属性：position；值： relative

与绝对定位不同的是，相对定位不会把该元素从原文档删除掉，而是在原文档的位置的基础上，移动一定的距离。原文档位置保留为空白

**若把空白填上则对下一元素进行绝对引用，并把left赋值auto。**

fixed

生成绝对定位的元素，相对于浏览器窗口进行定位。

元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

inherit

规定应该从父元素继承 position 属性的值。

static

**默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。**

## 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。

属性：float；值： left,right

**浮动后，原来的“坑”就让出来了**。并且是在原来的高度的基础上，向右浮动

但是，文字向左浮动时“浮动的文字”并没有走，还在原位置，结果，后一个元素就只好接在它后面。

**因此，当图片设为浮动却没有离开原位置时，下一个元素就围着图片摆放。**

**不允许出现浮动元素：**若不想下一个元素接着浮动的对象，需设置属性:clear；值: left right both none。此时该值前面达到不允许浮动元素出现的效果，就会继续向下排。

默认的div排列是会换行的

如果连续使用float就可以达到水平排列的效果，通常会用在菜单，导航栏等地方

如果超出了父容器，还会有自动换行的效果

## 清除浮动

1. 使用伪元素来清除浮动(::after,注意：作用于浮动元素的父亲）
2. 添加新的元素对其应用 clear:both
3. 父级div定义overflow:hidden

//在添加overflow属性后，浮动元素又回到了容器层，把容器高度撑起，达到了清理浮动的效果。当元素设置了overflow样式且值部位visible时，该元素就构建了一个BFC，BFC在计算高度时，内部浮动元素的高度也要计算在内，也就是说技术BFC区域内只有一个浮动元素，BFC的高度也不会发生塌缩，所以达到了清除浮动的目的，

## overflow

overflow 属性规定当内容溢出元素框时发生的事情。

值：

**visible 默认值。内容不会被修剪，会呈现在元素框之外。**

hidden 内容会被修剪，并且其余内容是不可见的。

scroll 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。

auto 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

inherit 规定应该从父元素继承 overflow 属性的值。

overflow-x

规定当内容溢出水平元素框时发生的事情。

overflow-y

当内容溢出垂直元素框时发生的事情。

## 元素的显示方式

元素的display显示方式有多种，隐藏、块级、内联、内联-块级

**display:none 隐藏**

display:none; 使得被选择的元素隐藏，并且不占用原来的位置

**display:block 块级**

display:block; 表示块级元素

块级元素会自动在前面和后面加上换行，并且在其上的width和height也能够生效

**display:inline 内联**

display:inline; 表示内联元素

内联元素前后没有换行，并且在其上的width和height也无效。 其大小由其中的内容决定

**display:inline-block 内联-块**级

内联是不换行，但是不能指定大小

块级时能指定大小，但是会换行

有时候，需要元素处于同一行，同时还能指定大小，这个时候，就需要用到内联-块级 inline-block

## CSS盒模型

包括IE盒子模型和标准的W3C盒子模型。

box-sizing：border-box,content-box.

IE盒子模型：width表示content+padding+border这三个部分的宽度

标准W3C盒子模型： width只表示content宽度

## 水平居中

内容居中

通过设置属性align="center" 居中的内容

通过样式style="text-align:center" 居中的内容

元素居中

margin:0 auto

默认情况下div会占用100%的宽度,所以无法观察元素是否居中

设置本div的宽度，然后再使用样式 margin: 0 auto来使得元素居中

span 是内联元素，无法设置宽度，所以不能通过margin:0 auto居中

span的居中可以通过放置在div中，然后让div text-align实现居中

左侧固定，右边自动占满

<**style**>

 .left{

   width:200px;

   float:left;

   background-color:pink

  }

  .right{

    overflow:hidden;

    background-color:lightskyblue;

  }

</**style**>

<**div** class="left">左边固定宽度</**div**>

<**div** class="right">右边自动填满</**div**>

垂直居中

**line-height方式**

line-height：数字+px//适合单独一行垂直居中

**内边距方式**

用padding实现；借助设置相同的上下内边距，实现垂直居中效果，可以用在多行文本上

**Table方式**

首先通过display: table-cell;把div用单元格的形式显示，然后借用单元格的垂直居中vertical-align: middle; 来达到效果。  
这样对图片也可以居中，上一步 line-height就不能对图片居中。

左右固定，中间自适应的布局

 .left{

   width:200px;

**float:left;**

   background-color:pink

  }

  .right{

   width:200px;

**float:right;**

   background-color:pink

  }

  .center {margin:0 200px;   background-color:lightblue}

一个div始终贴在下方的布局

首先把蓝色div设置为相对定位

然后把内部的绿色div设置为绝对定位， bottom: 0表示贴在下面

**块之间有空格**

span之间有回车换行导致显示效果出现时span间有空格

使用float来解决。将它们打包在同一个块下  
float使用完毕之后，记得在下面加上 <div style="clear:both"></div> 用于使得后续的元素，不会和这些span重复在一起

注：一个空白的（没有内容）div块不会占据空间

## 弹性布局

display:flex 布局

Flex是Flexible Box的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

### flex的六个属性

**flex-direction //容器内元素的排列方向(默认横向排列)**

flex-direction:row; 沿水平主轴让元素从左向右排列

flex-direction:column; 让元素沿垂直主轴从上到下垂直排列

flex-direction:row-reverse;沿水平主轴让元素从右向左排列

**flex-wrap //容器内元素的换行(默认不换行)**  
1：flex-wrap: nowrap; (默认)元素不换行,比如：一个div宽度100%，设置此属性，2个div宽度就自动变成各50%；

2：flex-wrap: wrap; 元素换行,比如：一个div宽度100%，设置此属性，第二个div就在第二行了；

**justify-content //元素在主轴（页面）上的排列**

1：justify-content : center;元素在主轴（页面）上居中排列

2：justify-content : flex-start;元素在主轴（页面）上由左或者上开始排列

3：justify-content : flex-end;元素在主轴（页面）上由右或者下开始排列

4：justify-content : space-between;元素在主轴（页面）上左右两端或者上下两端开始排列

5：justify-content : space-around;每个元素两侧的间隔相等。所以，元素之间的间隔比元素与边框的间隔大一倍。

**align-items 元素在主轴（页面）当前行的横轴（纵轴）方向上的对齐方式**

## CSS transform

语法transform: none|transform-functions;

transform 属性向元素应用 2D 或 3D 转换。该属性允许我们对元素进行旋转、缩放、移动或倾斜。

translate()

translate() 方法从其当前位置移动元素（根据为 X 轴和 Y 轴指定的参数）。

例：transform: translate(50px, 100px);//向右移50px,向下移100px.

rotate()

rotate() 方法根据给定的角度顺时针或逆时针旋转元素。

例：transform: rotate(20deg);//顺时针旋转20°

scale()

scale() 方法增加或减少元素的大小（根据本元素给定的宽度和高度参数）。

例：transform: scale(2, 3);// 元素增大为其原始宽度的两倍和其原始高度的三倍。

scaleY()

scaleX()

skewX()//沿x轴旋转度数

skewY()//沿y轴旋转度数

skew()

matrix()

## CSS 动画

@keyframes

animation-name

animation-duration

animation-delay

animation-iteration-count

animation-direction

animation-timing-function

animation-fill-mode

animation

CSS 可实现 HTML 元素的动画效果

如需使用 CSS 动画，您必须首先为动画指定一些关键帧。

关键帧包含元素在特定时间所拥有的样式。

@keyframes

在 @keyframes 规则中指定 CSS 样式，动画将在特定时间逐渐从当前样式更改为新样式。@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

div {

width: 100px;

height: 100px;

background-color: red;

**animation-name: example;**

animation-duration: 4s;

}

在上面的例子中，通过使用关键字 "from" 和 "to"（代表 0％（开始）和 100％（完成）），设置了样式何时改变。也可以使用百分比值。通过使用百分比，可以根据需要添加任意多个样式更改。

例：@keyframes example {

0% {background-color:red; left:0px; top:0px;}

25% {background-color:yellow; left:200px; top:100px;}

50% {background-color:blue; left:200px; top:200px;}

75% {background-color:green; left:0px; top:200px;}

100% {background-color:red; left:0px; top:0px;}

}//将在duration的25%时变成相应样式

animation-duration

规定动画的持续时间，默认为0s,即不发生动画。

animation-delay规定动画开始的延迟时间。

animation-iteration-count

指定动画应运行的次数。infinite表示无限持续

animation-direction

指定是向前播放、向后播放还是交替播放动画。

normal - 动画正常播放（向前）。默认值

reverse - 动画以反方向播放（向后）

alternate - 动画先向前播放，然后向后

alternate-reverse - 动画先向后播放，然后向前

animation-timing-function

规定动画的速度曲线。

ease - 指定从慢速开始，然后加快，然后缓慢结束的动画（默认）

linear - 规定从开始到结束的速度相同的动画

ease-in - 规定慢速开始的动画

ease-out - 规定慢速结束的动画

ease-in-out - 指定开始和结束较慢的动画

cubic-bezier(n,n,n,n) - 运行您在三次贝塞尔函数中定义自己的值

# JavaScript用于网页和用户之间的交互

**完整的javascript由语言基础，BOM和DOM组成。**

document是javascript的内置对象，代表浏览器的文档部分

document.write("Hello Javascript"); 向文档写入字符串

这是一个javascript和DOM结合的一个简单例子

onclick="..." 表示点击button之后做的事情

document.getElementById 根据id获取指定的元素

.style.display='none' 隐藏

.style.display='block' 显示(准确的讲，以block的形式显示)

onclick，getElementById，style.display 这些内容，是HTML DOM 应该到才会用到的知识和概念

<button onclick="document.getElementById('text') .style.display='none'">隐藏文本</button>

<button onclick="document.getElementById('text') .style.display='block'">显示文本</button>

<p id="text"> 这个是一段可以被javascript隐藏的文本</p>

**javascript代码必须放在script标签中**

script标签可以放在html的任何地方，一般建议放在head标签里

**如果有多段script代码，会按照从上到下，顺序执行**

**使用外部js文件**

和css一样，当javascript代码特别多，并且都写在html里的时候，会显得比较繁杂，难以维护。

这个时候可以采用和css一样的手段，把javascript代码剥离出来，单独放在一个文件里，在html中引用该文件即可。

注释：//单行注释

/\*

多行注释

\*/

**使用var声明一个变量**

关键字var 可有可无，本实例演示不使用var声明的变量

命名规则和java差不多；开头可以用 \_，$和字母；其他部分可以用 $ \_ 字母或者数字。

**调试(debug)**

1. 使用alert()进行调试

会弹出一个对话框，里面的内容是()的内容。换句话说，如果弹出了该内容,这个位置以上的代码，都是可以运行的。在此基础上不断将其往下移动，直到某一行之后不弹出，则该行出错

1. 用浏览器的F12打开界面，其中console是控制台的意思，用于输出一些错误和调试信息。
2. console.log()

类似于 alert进行调试， firebug有一个日志输出工具console.log().

**不同于alert的缺点( 弹出屏幕，阻挡所有其他操作)，console.log() 只会把信息输出在console里，而不会影响用户的使用。 用户甚至意识不到console.log()的存在。**

## JavaScript 嵌套函数//套娃函数

**所有函数都有权访问全局作用域。**

**事实上，在 JavaScript 中，所有函数都有权访问它们“上面”的作用域。**

**JavaScript 支持嵌套函数。嵌套函数可以访问其上的作用域。**

## JavaScript基本数据类型

undefined，Boolean, Number, String，null，symbol

**undefined 声明了但未赋值**

Boolean 布尔

Number 数字

**String 字符串**

**var 动态类型**

**typeof 变量类型判断**

**null 空对象/对象不存在**

**NaN是JavaScript的一个特殊的值，并不属于任何类型，并且不等于任何类型，包括它自身**

**undefined**

当一个变量被声明了，却没有赋值的时候，叫做 undefined

例：var x;输出x时输出undefined

**Boolean 布尔**

var y=false;

**Number 数字**

javascript中的Number可以表示十进制，八进制，十六进制整数，浮点数，科学记数法

var a=10; //十进制

var b=012;//第一位是0，表示八进制

var c=0xA;//0x开头表示十六进制

var d=3.14;//有小数点表示浮点数

var e=3.14e2;//使用e的幂表示科学计数法

**String 字符串**

与java不同的是，javascript中没有字符的概念，只有字符串，所以单引号和双引号，都用 来表示字符串。

**var 动态类型**

变量的类型是动态的，当值是整数的时候，就是Number类型，当值是字符串的时候，就是String类型

## 变量类型判断

使用typeof来进行判断数据类型

typeof可以识别出基本类型boolean,number,undefined,string,symbol，但是不能识别null。不能识别引用数据类型，会把null、array、object统一归为object类型,但是可以识别出function。

例：console.log(typeof bool); //Boolean

instanceof

instanceof不能识别出基本的数据类型 number、boolean、string、undefined、null、symbol。

但是可以检测出引用类型，如array、object、function，同时对于是使用new声明的类型，它还可以检测出多层继承关系。

instanceof一般用来检测对象类型，以及继承关系。

arrname instanceof Array;// return true

constructor

null、undefined没有construstor属性，因此constructor不能判断undefined和null。

console.log(arr.constructor === Array);// true

Object.prototype.toString.call

console.log(Object.prototype.toString.call(num));//[object Number]

该方法直接返回对应类型的字符串

## 类型转换

伪对象

**伪对象概念：**javascript是一门很有意思的语言，**即便是基本类型，也是伪对象，所以他们都有属性和方法。**

变量a的类型是字符串，通过调用其为伪对象的属性length获取其长度

**转换为字符串**

无论是Number,Boolean还是String都有一个toString方法，用于转换为字符串

**数字转换为字符串**

默认模式下，数字10转换为十进制的10

基模式下，数字10转换为二进制的1010

基模式下，数字10转换为八进制的12

基模式下，数字10转换为十六进制的a

a.toString()); //默认模式

a.toString(2)；//将数字a转换为基模式下二进制。

**转换为数字**

javascript分别提供内置函数 parseInt()和parseFloat()，转换为数字

注：如果被转换的字符串，同时由数字和字符构成，那么parseInt会一直定位数字，直到出现非字符。 所以"10abc" 会被转换为 10，而”abc10”为NaN

**//判断每一位，直到发现不是数字的那一位就break;返回NaN**

**转换为Boolean**

使用内置函数Boolean() 转换为Boolean值

当转换字符串时：

非空即为true

当转换数字时：

非0即为true

当转换对象时：

非null即为true

**Number()和parseInt()的区别**

Number()和parseInt()一样，都可以用来进行数字的转换

区别在于，当转换的内容包含非数字的时候，Number() 会返回NaN(Not a Number)

parseInt() 要看情况，如果以数字开头，就会返回开头的合法数字部分，如果以非数字开头，则返回NaN

**String()和toString()的区别**

String()和toString()一样都会返回字符串，区别在于对null的处理

String()会返回字符串"null"

toString() 就会报错，无法执行

## JavaScript函数

**function**关键字用于定义一个函数

**print**即函数的名称

()表示参数列表

{ 表示函数开始

} 表示函数结束

光有函数的定义，还不够，它不会自动执行，还需要进行调用

**带返回值的函数，创建函数时不需要声明返回什么，直接在最后一行return + 返回值。**

**文本框获取的值都视作字符串**

**作用域：**一个参数的作用域就在这个函数内部，超出函数就看不见该参数了

**全局变量**：定义在函数之外，即全局变量，所有函数都可以访问

**事件：**事件是javascript允许html与用户交互的行为。用户任何对网页的操作，都会产生一个事件。事件有很多种，比如鼠标移动，鼠标点击，键盘点击等等。

**JavaScript运算符：**

**绝对等，绝对不等于**

与==进行值是否相等的判断不同 ，绝对等 ===还会进行 类型的判断

比如 数字1和 字符串'1'比较，值是相等的，但是类型不同

所以==会返回true,但是===会返回false

绝对不等于!== 与上是一个道理

**三目运算符**

三目运算符 ?: 有三个操作数

如果第一个返回true,就返回第二个操作符

否则就返回第三个操作符。

var movie = age<18?"卡通":"你懂的";

**注意： 进行循环的时候，一定要设置好自增和结束条件，否则会导致无限循环，浏览器直接无法响应。**

## Javascript对象

常见的对象有数字Number，字符串String，日期Date，数组Array等。

**Number**

注: 这里讲的Number是对象Number,和基本数据类型中的基本类型Number是不一样的Number。

new Number 创建一个数字对象

属性MIN\_VALUE

属性MAX\_VALUE 最小值 最大值

属性NaN 表示不是数字

方法toFixed 返回一个数字的小数表达

方法toExponential 返回一个数字的科学计数法表达

方法valueOf 返回一个数字对象的基本数字类型

**与基本类型的数字不同，对象类型的数字，拥有更多的属性和方法**

**new Number 创建一个数字对象**

var x = new Number(123);//此时它的值虽然是123，但它的类型是object。

**Number边界**

Number.MIN\_VALUE//Number对象的最小值:5e-324

Number.MAX\_VALUE//Number对象的最大值:1.7976931348623157e+308

NaN(Not a Number),表示不是一个数字  
当通过非数字创建Number的时候，就会得到NaN.  
注意： 不能通过 是否等于Number.NaN来判断 是否 “不是一个数字”，应该使用函数 **isNaN()**

**返回一个数字的小数表达**

1. toFixed(2)//返回保留2位小数的number a

**返回一个Number对象的科学计数法表达**

a.toExponential ()

**返回一个数字对象的基本数字类型**

var b = a.valueOf()//返回a的number类并赋值给b

#### 数组

**javascript中的数组是动态的，即长度是可以发生变化的。**

new Array 创建数组对象

属性 length 数组长度

for

for in 遍历一个数组

方法 concat 连接数组

方法 join 通过指定分隔符，返回一个数组的字符串表达

方法 push pop 分别在最后的位置插入数据和获取数据(获取后删除)

方法 unshift shift 分别在最开始的位置插入数据和获取数据(获取后删除)

方法 sort 对数组的内容进行排序

方法 sort(comparator) 自定义排序算法

方法 reverse 对数组的内容进行反转

方法 slice 获取子数组

方法 splice 删除和插入元素

**连接数组**

var z = x.concat(y);//x必须是数组

**方法 join 通过指定分隔符，返回一个数组的字符串表达**

数组x是:3,1,4  
y = x.join() 得到的是数组x的字符串表达，其值是3,1,4 其类型是 :string  
z = x.join("@");是x的字符串表达，不过分隔符不是默认的"," 而是"@" : 3@1@4

**分别在最后的位置插入数据和获取数据(获取后删除)**

方法 push pop,分别在最后的位置插入数据和获取数据(获取后删除)  
就像先入后出的栈一样

x.push(5);

**分别在最开始的位置插入数据和获取数据(获取后删除)**

方法 unshift shift ,分别在最开始的位置插入数据和获取数据(获取后删除)

x.unshift (5);

**对数组的内容进行排序**

x.sort();//默认从小到大排序

**//sort调用了toString()方法，所以排序10及以上的数字需要自定义排序函数**

**自定义排序算法**

如果需要采用自定义排序的算法，就把比较器函数作为参数传递给sort()。

调用sort函数的时候，把这个比较器函数comparator作为参数传递进去即可

x.sort(comparator);

**对数组的内容进行反转**

方法 **reverse**，对数组的内容进行反转

x.reverse();

**获取子数组**

var z = x.slice(1,3);

左闭右开

**删除和插入元素**

方法 splice (不是 slice) 用于删除数组中的元素，奇葩的是 ，**它还能用于向数组中插入元素** 例如：数组x是:3,1,4,1,5,9,2,6  
x.splice (3,2) 表示从位置3开始 ，删除2个元素:3,1,4,9,2,6  
x.splice(3,0,1,5) 从位置3开始，删除0个元素，但是插入1和5,最后得到:3,1,4,1,5,9,2,6

#### JavaScript 日期

new Date 创建日期对象

getFullYear

getMonth

getDate 年/月/日

getHours

getMinutes

getSeconds

getMilliseconds 时:分:秒:毫秒

getDay 一周的第几天

getTime 经历的毫秒数

setFullYear

setMonth

setDate

setHours

setMinutes

setSeconds 修改日期和时间

需要注意的是，getMonth()返回的月数，是基零的，0代表1月份

所以需要+1

通过getDay()获取，今天是本周的第几天。与getMonth()一样，返回值是基0的。

#### JavaScript Math

Math是JavaScript的工具对象，用于常见的数学运算

属性E PI 自然对数和圆周率

方法 abs 绝对值

方法 min max 最小最大

方法 pow 求幂

方法 round 四舍五入

方法 random 随机数//取0-1之间的随机数

**JavaScript 自定义对象**

**通过new Object()创建一个对象**

var hero = new Object();

hero.name = "盖伦"; //定义一个属性name，并且赋值

hero.kill = function(){

  document.write(hero.name + " 正在杀敌" ); //定义一个函数kill

}

hero.kill(); //调用函数kill

**通过function设计一个对象**

通过new Object创建对象有个问题，就是每创建一个对象，都得重新定义属性和函数。那么，采用另一种方式，通过function设计一种对象。 然后实例化它。这种思路很像Java里的设计一种类，但是 javascript没有类，只有对象，所以我们叫设计一种对象

//虚拟对象？

function Hero(name){

  this.name = name;

  this.kill = function(){

     document.write(this.name + "正在杀敌<**br**>");

  }

}

var gareen = new Hero("盖伦");

**为已经存在的对象，增加新的方法**

Hero.prototype.keng = function(){

  document.write(this.name + "正在坑队友<**br**>");

//继承虚拟对象？只要在调用新方法或属性之前添加新方法或属性就行。

### 闭包

一个函数和对其周围状态（lexical environment，词法环境）的引用捆绑在一起（或者说函数被引用包围），这样的组合就是闭包（closure）。也就是说，**闭包让你可以在一个内层函数中访问到其外层函数的作用域。**在 JavaScript 中，每当创建一个函数，闭包就会在函数创建的同时被创建出来。

# BOM

**BOM即 浏览器对象模型(Browser Object Model)**

**浏览器对象包括  
Window(窗口)**[**Navigator(浏览器)**](https://how2j.cn/k/javascript/javascript-bom-navigator/453.html)[**Screen (客户端屏幕)**](https://how2j.cn/k/javascript/javascript-bom-screen/450.html)[**History(访问历史)**](https://how2j.cn/k/javascript/javascript-bom-history/452.html)[**Location(浏览器地址)**](https://how2j.cn/k/javascript/javascript-bom-location/451.html)

**Window(窗口)**

**获取文档显示区域的高度和宽度.**

一旦页面加载，就会自动创建window对象，所以无需手动创建window对象。

通过window对象的属性可以获取文档显示区域的高度和宽度.

window.innerWidth；window.innerHeight；

**获取外部窗体的宽度和高度**

所谓的外部窗体即浏览器

window.outerWidth；window.outerHeight；

**打开一个新的窗口**

通过window的open方法做到的

不建议使用，如果需要打开一个新的网站，应该通过超级链接等方式让用户主动打开，在没有告知用户的前提下就打开一个新的网站会影响用户的体验

window.open("/");//打开目前网站主页，即根目录

[**Navigator(浏览器)**](https://how2j.cn/k/javascript/javascript-bom-navigator/453.html)

Navigator即浏览器对象，提供浏览器相关的信息

浏览器产品名称:navigator.appName

浏览器版本号：navigator.appVersion

浏览器内部代码：navigator.appCodeName

操作系统：navigator.platform

是否启用Cookies：navigator.cookieEnabled

浏览器的用户代理报头：navigator.userAgent

[**Screen (客户端屏幕)**](https://how2j.cn/k/javascript/javascript-bom-screen/450.html)

Screen对象表示用户的屏幕相关信息

用户的屏幕分辨率:screen.width + "\*" + screen.height

可用区域大小:screen.availWidth + "\*" + screen.availHeight

//如果是在台式电脑上，通常看到的可用区域的高度会比屏幕高度小一点，因为有任务栏的存在。

[**History(访问历史)**](https://how2j.cn/k/javascript/javascript-bom-history/452.html)

History用于记录访问历史

history.back();//返回上次访问

history.go(-2); //-1表示上次，-2表示上上次，以次类推

[**Location(浏览器地址)**](https://how2j.cn/k/javascript/javascript-bom-location/451.html)

Location表示浏览器中的地址栏

location.reload();//**刷新当前页面**

**跳转页面**：方法1：location="/";方法2：location.assign("/");

**Location的其他属性**

协议 location.protocol  
主机名 location.hostname  
端口号 (默认是80，没有即表示80端口)location.port  
主机加端口号 location.host  
访问的路径 location.pathname  
锚点 location.hash  
参数列表 location.search

**弹出框**

浏览器上常见的弹出框有

警告框，确认框，提示框 这些都是通过调用window的方法实现的。

比如警告框用的是window.alert("警告内容")，因为很常用，所以就把window省略掉，直接使用alert

**Alert() 警告框** //警告框 alert，常用于消息提示，比如注册成功等等

**Confirm() 确认框** //常用于危险性操作的确认提示。 比如删除一条记录的时候，弹出确认框  
confirm返回基本类型的Boolean true或者false

**prompt 输入框**//输入框 prompt，用于弹出一个输入框，供用户输入相关信息。 因为弹出的界面并不好看，很有可能和网站的风格不一致，所以很少会在实际工作中用到。

**计时器**

setTimeout 只执行一次

setInterval 不停地重复执行

clearInterval 终止重复执行

**document.write()** 不要在setInterval调用的函数中使用document.write();

**只执行一次**

函数setTimeout(functionname, 距离开始时间毫秒数 );  
通过setTimeout在制定的毫秒数时间后，执行一次 函数functionname  
本例在3秒钟后，打印当前时间。  
document.getElementById 获取id=time的div元素  
.innerHTML 修改该元素的内容

**不停地重复执行**

函数setInterval(函数名, 重复执行的时间间隔毫秒数 );

通过setInterval重复执行同一个函数，重复的时间间隔由第二个参数指定

**终止重复执行**

通过clearInterval()终止一个不断重复的任务

# JavaScript

## javascript放置位置

**在 HTML 中，JavaScript 代码必须位于 <script> 与 </script> 标签之间。**

**head**

**body**

把脚本置于 <body> 元素的底部，可改善显示速度，因为脚本编译会拖慢显示。

**外置**

外部 JavaScript 的优势

在外部文件中放置脚本有如下优势：

分离了 HTML 和代码

使 HTML 和 JavaScript 更易于阅读和维护

已缓存的 JavaScript 文件可加速页面加载

## JavaScript输出

使用 window.alert() 写入警告框

使用 document.write() 写入 HTML 输出

使用 innerHTML 写入 HTML 元素

使用 console.log() 写入浏览器控制台

注意：在 HTML 文档完全加载后使用 document.write() 将删除所有已有的 HTML

## JavaScript赋值

用var再次声明时，若第二次未赋值，其值将仍为之前的值。

js中\*\*为幂运算，var x = x \*\* 2; 令x=x平方。结果与 Math.pow(x,2)相同

运算符优先级

9 & 按位与 x & y

8 ^ 按位 XOR x ^ y

7 | 按位或 x | y

6 && 逻辑与 x && y

5 || 逻辑否 x || y

4 ? : 条件 ? "Yes" : "No"

## JavaScript数据类型

数字会省略可省略的小数，即34.00变成34。

任何变量均可通过设置值为 undefined 进行清空。其类型也将是 undefined。

person = undefined;

在 JavaScript 中，null 的数据类型是对象，可以通过设置值为 null 清空对象

**Undefined 与 null 的值相等，但类型不相等**

#### JavaScript object

以属性:值的方式来书写，JavaScript 对象是被命名值的容器。

例：var person = {firstName:"Bill", lastName:"Gates", age:62, eyeColor:"blue"};

对象也可以有方法，方法是在对象上执行的动作。方法以函数定义被存储在属性中

例：var person = {

firstName: "Bill",

lastName : "Gates",

id : 678,

fullName : function() {

return this.firstName + " " + this.lastName;

}

};

**this** 引用该函数的“拥有者”,此处为fullname的拥有者person对象

有两种访问属性的方式；objectName.propertyName或者objectName["propertyName"]

访问方法则用objectName.methodName()

**注：如果通过关键词 "new" 来声明 JavaScript 变量，则该变量会被创建为对象**

## JavaScript字符串

转义字符：为了防止字符串中的’,”,\等特殊符号被误解

\' ' 单引号

\" " 双引号

\\ \ 反斜杠

尽量别用new 构造字符串，它会拖慢执行速度甚至

var x = new String("Bill");

var y = new String("Bill");

// (x == y) 为 false，因为 x 和 y 是不同的对象，而JavaScript 对象无法进行对比，比较两个对象将始终返回 false。

## JavaScript字符串方法

属性 length 字符串长度

方法 indexOf lastIndexOf 子字符串出现的位置 未找到返回-1

而且接受第二个参数作为查找的起始位置

方法 search()搜索特定字符串的位置

**提取字符串**

slice(start, end) 若参数为负代表从最后一位往前计数

substring(start, end)

substr(start, length) 第二个参数代表截取的长度，如果首个参数为负，则从字符串的结尾计算位置。

**替换子字符串**

方法 replace() //**方法不会改变调用它的字符串。它返回的是新字符串**。

**字符串拼接**

方法 concat() 字符串拼接,可拼接多个字符串

trim() 方法删除字符串两端的空白符：

方法 localeCompare 比较两段字符串是否相同

方法 split 根据分隔符，把字符串转换为数组

**返回指定位置的字符**

charAt 返回指定位置的字符，找不到返回空字符串  
charCodeAt 返回指定位置的字符对应的Unicode码

**比较两段字符串是否相同**

x.localeCompare(y)

//返回0 表示相等

1 表示字母顺序靠后

-1 表示字母顺序靠前

**根据分隔符，把字符串转换为数组**

var y =  x.split(" ");//通过空格分隔split(" "),得到数组

var z =  x.split(" ",2);//通过空格分隔split(" ",2),得到数组，并且只保留前两个

注： 第二个参数可选，表示返回数组的长度

## JavaScript数字

JavaScript 数值始终是 64 位的浮点数，此格式用 64 位存储数值，其中 0 到 51 存储数字（片段），52 到 62 存储指数，63 位存储符号。

在所有数字运算中，JavaScript 会尝试将字符串转换为数字。

NaN的含义是not a number,但它的type是number，它参与的数学运算结果也是NaN.

isNaN() 来确定某个值是否是数.

Infinity的含义是超出最大值时返回的值。也是number型。

使用 toString() 方法把数输出为十六进制、八进制或二进制。

myNumber.toString(16)//返回myNumber的16进制值80

## JavaScript数字方法

toFixed() 返回字符串值，指定保留几位小数。

* 1. toFixed(1); //返回3.6

toPrecision() 返回字符串值，指定保留几位有效数字。

## JavaScript数组

使用[]创建数组或new Array();的方式创建数组。

数组类型是array,属于引用类型。

数组元素可以是对象，函数，甚至数组。

向数组添加新元素的最佳方法是使用 push() 方法。

也可以直接arr[index],但在0到index之间未被定义的区域为undefined。

数组和对象的区别

在 JavaScript 中，数组使用数字索引。

在 JavaScript 中，对象使用命名索引。

数组是特殊类型的对象，具有数字索引。

## JavaScript数组方法

数组转字符串

方法 toString() 把数组转换为数组值（逗号分隔）的字符串。

join() 方法也可将所有数组元素结合为一个字符串并且还可以规定分隔符。

例：var fruits = ["Banana", "Orange","Apple", "Mango"];

document.getElementById("demo").innerHTML = fruits.join(" \* ");

//Banana \* Orange \* Apple \* Mango

添加和删除数组元素

pop():删除末尾并返回最后一个元素

push()：添加到末尾并返回该数组的长度。

shift():删除开头并返回第一个元素

unshift():添加到开头并返回该数组长度

delete，将元素修改为undefined

例：delete fruits[0]; // 把 fruits 中的首个元素改为 undefined

拼接数组

splice() 方法返回一个包含已删除项的数组。

splice(fromindex,deletenum,newelement,..).

例：fruits.splice(2, 0, "Lemon", "Kiwi");

从fruits的第三个元素开始拼接，删除之后的0个元素，接上“Lemon”，“Kiwi”新元素。

可通过设置不加入新元素达到删除原有元素的效果。

合并（连接）数组

concat() 方法通过合并（连接）现有数组来创建一个新数组。

concat() 方法**不会更改现有数组。它总是返回一个新数组**。

concat() 方法可以使用任意数量的数组参数。

var myChildren = arr1.concat(arr2, arr3); // 将arr1、arr2 与 arr3 连接在一起

也可以直接带入数组值为参数。[]

裁剪数组

slice() 方法用数组的某个片段切出新数组。

slice() 可接受两个参数，比如 (1, 3)。

该方法会从开始参数选取元素，直到结束参数（不包括）为止。

如果需要原始值，则 JavaScript 会自动把数组转换为字符串。

## JavaScript数组排序

sort() 方法照字符串顺序对值进行排序，因此25>100。只适用于字符串排序。

可通过定义比值函数进行排序，如function(a, b){return a-b}。返回负，0，正值。

当 sort() 函数比较两个值时，会将值发送到比较函数，并根据所返回的值（负、零或正值）对这些值进行排序。

例：points.sort(function(a, b){return a - b});

reverse()方法反转数组中的元素,也是按照字符串顺序对值排序，但不能通过定义比值函数排序。

还可以随机排序

例：points.sort(function(a, b){return 0.5 - Math.random()});

## Javascript for-each循环

for-in

使用for-in可以遍历数组，但是会存在以下问题：

1.index索引为字符串型数字（注意，非数字），**不能直接进行几何运算**。

2.遍历顺序有可能不是按照实际数组的内部顺序（可能按照随机顺序）。

3.使用for-in会遍历数组所有的可枚举属性，包括原型。

所以for-in更适合遍历对象，通常是建议不要使用for-in遍历数组。

for-of

for-of可以简单、正确地遍历数组（不遍历原型method和name）。

1.这是最简洁、最直接的遍历数组元素的语法。

2.这个方法避开了for-in循环的所有缺陷。

3.与forEach()不同的是，它可以正确响应break、continue和return语句。

for-in是ES5标准，遍历的是key（可遍历对象、数组或字符串的key）；for-of是ES6标准，遍历的是value（可遍历对象、数组或字符串的value）。

因此建议是使用for-of遍历数组，因为for-of遍历的只是数组内的元素，而不包括数组的原型属性method和索引name。

## …

## JavaScript 正则表达式

正则表达式是构成搜索模式（search pattern）的字符序列。正则表达式可用于执行所有类型的文本搜索和文本替换操作。

语法/pattern/modifiers;//pattern是搜索内容，modifiers是修饰符

**在 JavaScript 中，正则表达式常用于两个字符串方法：search() 和 replace()。**

search() 方法使用表达式来搜索匹配，然后返回匹配的位置。

var str = "Visit W3School!";

var n = str.search(/W3School/i);//返回n=6，即在str中出现的位置。

replace() 方法返回模式被替换处修改后的字符串。

var res = str.replace("Microsoft", "W3School");//将Microsoft替换为W3School

修饰符

i 执行对大小写不敏感的匹配。

g 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

m 执行多行匹配。

## JavaScript作用域

JavaScript 拥有函数作用域：每个函数创建一个新的作用域。

作用域决定了这些变量的可访问性（可见性）。

函数内部定义的变量从函数外部是不可访问的（不可见的）。

在函数开始时会创建局部变量，在函数完成时会删除它们。

自动全局

如果您为尚未声明的变量赋值，此变量会自动成为全局变量。

比如在函数内部直接给carname赋值，会默认为carname在外部定义为全局变量。

function myFunction() {

carName = "porsche";

}

在 HTML 中，全局作用域是 window。所有全局变量均属于 window 对象。

## JavaScript 提升（Hoisting）

提升（Hoisting）是 JavaScript 将声明移至顶部的默认行为。

即可以**先使用再声明**（先上车后买票？）

为了避免 bug，请始终在每个作用域的开头声明所有变量。

## JavaScript严格模式

通过在脚本或函数的开头添加 "use strict"; 来声明严格模式。

"use strict"; 的作用是指示 JavaScript 代码应该以“严格模式”执行。

在严格模式中，您无法，例如，使用未声明的变量。

严格模式中不允许的事项

在不声明变量的情况下使用变量，是不允许的

对象也是变量，在不声明对象的情况下使用对象也是不允许的

删除变量（或对象）是不允许的：

删除函数是不允许的：

重复参数名是不允许的

八进制数值文本是不允许的：

转义字符是不允许的：

写入只读属性是不允许的：

写入只能获取的属性是不允许的：

删除不可删除的属性是不允许的：

字符串 "eval" 不可用作变量

字符串 "arguments" 不可用作变量：

with 语句是不允许的：

处于安全考虑，不允许 eval() 在其被调用的作用域中创建变量：

在类似 f() 的函数调用中，this 的值是全局对象。在严格模式中，现在它成为了 undefined。

## JavaScript this

JavaScript this 关键词指的是它所属的对象。

它拥有不同的值，具体取决于它的使用位置：

在方法中，this 指的是所有者对象。

单独的情况下，this 指的是全局对象。

在函数中，this 指的是全局对象。

在函数中，严格模式下，this 是 undefined。

在事件中，this 指的是接收事件的元素。

像 call() 和 apply() 这样的方法可以将 this 引用到任何对象。

## javascript版本

|  |  |  |
| --- | --- | --- |
| 版本 | 官方名称 | 描述 |
| 1 | ECMAScript 1 (1997) | 第一版。 |
| 2 | ECMAScript 2 (1998) | 只改变编辑方式。 |
| 3 | ECMAScript 3 (1999) | * 添加了正则表达式。 * 添加了 try/catch。 |
| 4 | ECMAScript 4 | 从未发布过。 |
| 5 | ECMAScript 5 (2009)  [阅读更多：JS ES5](https://www.w3school.com.cn/js/js_es5.asp) | * 添加了“严格模式”。 * 添加了 JSON 支持。 * 添加了 String.trim()。 * 添加了 Array.isArray()。 * 添加了数组迭代方法。 |
| 5.1 | ECMAScript 5.1 (2011) | 编辑改变。 |
| 6 | ECMAScript 2015  [阅读更多：JS ES6](https://www.w3school.com.cn/js/js_es6.asp) | * 添加了 let 和 const * 添加了默认参数值 * 添加了 Array.find() * 添加了 Array.findIndex() |
| 7 | ECMAScript 2016 | * 添加了指数运算符（\*\*）。 * 添加了 Array.prototype.includes。 |
| 8 | ECMAScript 2017 | * 添加了字符串填充。 * 添加了新的 Object 属性。 * 添加了异步功能。 * 添加了共享内存。 |
| 9 | ECMAScript 2018 | * 添加了 rest / spread 属性。 * 添加了异步迭代。 * 添加了 Promise.finally()。 * 增加 RegExp。 |

ECMAScript 通常缩写为 ES。

所有浏览器都完全支持 ECMAScript 3。

所有现代浏览器都完全支持 ECMAScript 5。

## JavaScript版本ES5

新特性

* "use strict" 指令
* String.trim()
* Array.isArray()
* Array.forEach()
* Array.map()
* Array.filter()
* Array.reduce()
* Array.reduceRight()
* Array.every()
* Array.some()
* Array.indexOf()
* Array.lastIndexOf()
* JSON.parse()
* JSON.stringify()
* Date.now()
* 属性 Getter 和 Setter
* 新的对象属性和方法

# JavaScript版本ES6

新特性

* let
* const
* 幂 (\*\*)
* 默认参数值
* Array.find()
* Array.findIndex()
* set和map数据结构
* promise对象
* Generator 函数
* symbol
* 箭头函数
* Class（类）
* module（模块）

箭头函数

箭头函数表达式的语法比函数表达式更简洁，并且没有自己的this，arguments，super或new.target。箭头函数表达式更适用于那些本来需要匿名函数的地方，并且它不能用作构造函数。

**基础语法**：(param1, param2, …, paramN) => { statements }

(param1, param2, …, paramN) => expression

//相当于：(param1, param2, …, paramN) =>{ return expression; }

当只有一个参数时（）可省略。

**高级语法：**//加括号的函数体返回对象字面量表达式：params => ({foo: bar})

**this：**箭头函数不会创建自己的this,它只会从自己的作用域链的上一层继承this。

promise

Promise 对象用于表示一个异步操作的最终完成 (或失败)及其结果值。

本质上 Promise 是一个函数返回的对象，我们可以在它上面绑定回调函数，这样我们就不需要在一开始把回调函数作为参数传入这个函数了。

var promise = new Promise(function(resolve, reject) {

// 异步处理

// 处理结束后、调用resolve 或 reject

});

**promise.then()**

即使异步操作已经完成（成功或失败），在这之后通过 then() 添加的回调函数也会被调用。

如果之前的promise返回的结果是一个新的promise，会等到新的promise结束后才开始调用then内的函数。//传递到 then() 中的函数被置入到一个微任务队列中，而不是立即执行，这意味着它是在 JavaScript 事件队列的所有运行时结束了，且事件队列被清空之后，才开始执行。

并且可通过后接then形成链式调用

**Promise.catch方法：捕捉错误**

Promise.prototype.catch 方法是 Promise.prototype.then(null, rejection) 的别名，用于指定发生错误时的回调函数。

**Promise.all方法**

var p = Promise.all([p1,p2,p3]);

Promise.all 方法用于接收多个promise的iterable类型，包装成一个新的 Promise 实例。这样可以进行多个异步操作，

只要p1、p2、p3之中有一个被rejected，p的状态就变成rejected，此时第一个被reject的实例的返回值，会传递给p的回调函数。

Promise.race 方法同样是将多个 Promise 实例，包装成一个新的 Promise 实例。

**promise.finally()**

无论如何最终都会执行的语句，类似Java多线程

## JavaScript对象

# JavaScript函数

**匿名函数（没有名称的函数）**

存放在变量中的函数不需要函数名。他们总是使用变量名调用。

使用表达式定义的函数不会被提升。即var x = function(){};用等号定义的函数。

自调用函数

函数表达式会自动执行，假如表达式后面跟着 ()。

(function () {

var x = "Hello!!"; //我会调用我自己

})();

函数参数

**JavaScript 函数不会对参数值进行任何检查。**

**当参数不足时默认传递的undefined**

**当参数过多时可使用arguments对象**

JavaScript 函数有一个名为 arguments 对象的内置对象。

arguments 对象将传入的参数视为一个数组。

**参数的改变在函数之外是不可见的。**

**对象属性的改变在函数之外是可见的。**

函数call

call()函数能够调用另一个函数的方法，并应用到作为参数传入的对象上。

**object1.fuctionName.call(object2);//将object1中的函数调用到object2上。**

例：var person = {

fullName: function() {

return this.firstName + " " + this.lastName;

}

}

var person1 = {

firstName:"Bill",

lastName: "Gates",

}

person.fullName.call(person1);

call的参数还需要根据调用的函数所需参数决定。

函数apply

类似call, 用于不同对象的方法调用，但call() 方法分别接受参数。**apply() 方法接受数组形式的参数**。如果要使用数组而不是参数列表，则 apply() 方法非常方便。

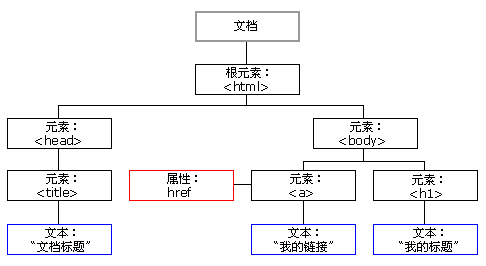
注：call 和apply存在于每一个方法中。

# HTML DOM

DOM 是Document Object Model( 文档对象模型 )的缩写。

DOM是**把html里面的各种数据当作对象**进行操作的一种思路。

比如一个超链，作为一个DOM对象，就可以使其隐藏，修改其href指向的地址。



### 获取节点

document.getElementById 通过id获取元素节点

document.getElementsByTagName 通过标签名称获取元素节点

document.getElementsByClassName 通过类名获取元素节点

document.getElementsByName 通过表单元素的name获取元素节点

**表单元素都有name属性，通过getElementsByName可以根据name属性的值，获取元素节点。**

### 获取属性节点

获取元素节点，然后通过元素节点的attributes获取其下所有的属性节点。

因为属性节点是多个，所以是以数组的形式返回出来的，接着通过for循环遍历，查看每个节点的nodeName和nodeValue

如果要获取一个指定属性的值，可以采用如下风格，**as表示所有的属性，as["id"]取出名称是id的属性**

**as["id"].nodeValue**

<html>

<div id="d1" align="center" class="abc">hello HTML DOM</div>

<script>

var div1 = document.getElementById("d1");

var as = div1.attributes;

document.write("div总共有"+as.length +" 个属性");

document.write("分别是:");

for(i = 0; i< as.length; i++){

document.write("<br>");

document.write(as[i].nodeName);

document.write(":");

document.write(as[i].nodeValue);

}

document.write("<br>");

document.write("div的id属性值是："+ as["id"].nodeValue);

</script>

</html>

### 获取内容节点

获取元素节点，然后通过childNodes获取其所有的子节点。 其中第一个子节点，就是其内容节点。然后借助nodeName和nodeValue把内容节点的名称和值打印出来。

setTimeout()

setTimeout() 方法用于在指定的毫秒数后调用函数或计算表达式。

setTimeout(code,millisec)

code 必需。要调用的函数后要执行的 JavaScript 代码串。

millisec 必需。在执行代码前需等待的毫秒数。

调用该方法后，code里的内容进入任务列队，会等到millisec之后再一起执行。

clearTimeout()

用于取消setTimeout()

使用方法：clearTimeout(id\_of\_settimeout)；

setinterval()

setinterval()是定时调用的函数，可按照指定的周期（以毫秒计）来调用函数或计算表达式。

setInterval() 方法会不停地调用函数，直到 clearInterval() 被调用或窗口被关闭。

setInterval(function,interval,[arg1,arg2,......argn])//默认语法，interval设置间隔时间。之后的参数为传入function的值。

## 节点属性

nodeName 节点名称

nodeValue 节点值

nodeType 节点类型

innerHTML 元素的文本内容

id

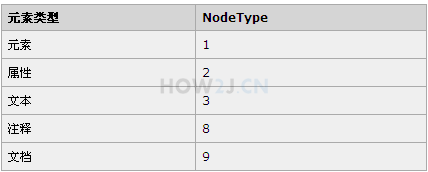
value

className 元素上的属性

### 节点类型

nodeType表示一个节点的类型

不同的节点类型，对应的节点类型值是不一样的



### 元素文本内容

修改与获取内容的值可以通过 childNodes[0].nodeValue进行；还有个简便办法就是通过innerHTML进行。 效果是一样的。

### 元素属性

元素上的属性，比如id,value 可以通过 . 直接访问

如果是自定义属性，那么可以通过如下两种方式来获取

getAttribute("test")

attributes["test"].nodeValue

注: class需要通过className获取

## DOM元素样式

**1.一个元素节点的style属性即对应的css**，通过获得一个DOM，并用DOM.style.~的格式修改对应CSS属性

例：d.style.display="none";//隐藏代号为d的DOM

**2.或者直接修改DOM的CSS，**

例：d1.css("background-color","green");//修改代号为d1的DOM的背景颜色

**注：Javascript并不提供这样的解决方案，但是到了JQuery就提供了这样的解决方案**

## DOM 事件

onfocus 获取焦点事件

onblur 失去焦点事件

onmousedown 鼠标按下事件

onmouseup 鼠标弹起事件

onmousemove 鼠标经过事件 注：每次移动都会触发

onmouseover 鼠标进入事件

onmouseout 鼠标离开事件

onkeydown 键盘按下事件

onkeypress 键盘按下事件

onkeyup 键盘弹起事件

onclick 单击事件 注1：在组件上，按下空格或则回车键也可以造成单击的效果，但是却不能造成双击的效果

ondblclick 双击事件

onchange 变化事件 注：对于输入框而言，只有在失去焦点的时候，才会触发

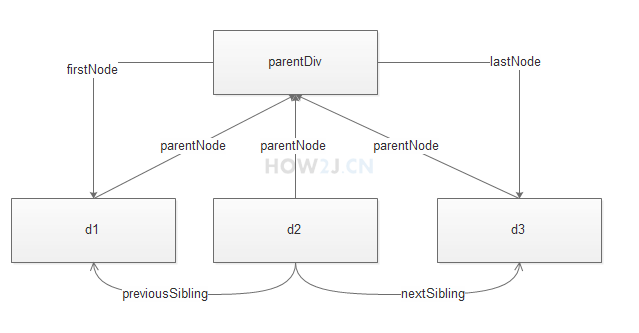
onsubmit 提交事件

onload 加载事件

**this 当前组件 this表示触发事件的组件，可以在调用函数的时候，作为参数传进去**

return false 阻止事件的发生

## DOM 节点关系



<div id="parentDiv">

<div id="d1">第一个div</div>

<div id="d2">第二个div</div>

<div id="d3">第三个div</div>

</div>

假设html代码如实例中，那么各个元素节点的关系如下:

d1 d2 d3 的parentNode是parentDiv，再往上是body，再往上是html，再往上是document

parentDiv的firstNode是 d1

parentDiv的lastNode是d3

d2的previousSibling是d1

d2的nextSibling是d3

parentDiv的children是 d1 d2 d3

**元素节点的属性：parentNode父节点、previousSibling，nextSibling同胞节点、childNodes子节点**

**注意不是紧挨着的标签之间有任何字符、空白、换行都会产生文本元素。 所以获取到的节点名称是#text.**

### childNodes和children的区别

childNodes和children都可以获取一个元素节点的子节点。

childNodes 会包含文本节点；children 会排除文本节点

## 创建节点

createElement 创建元素节点 例：var hr=document.createElement("hr");注：标签类型用双引号

appendChild 添加子节点 div1.appendChild(hr);

createTextNode 创建文本节点 可加入到文本型节点如<p>里作为值

createAttribute 创建属性节点 然后用setAttributeNode把该属性设置到元素节点a上

var href = document.createAttribute("href");

href.nodeValue="http://12306.com";

a.setAttributeNode(href);

## 删除节点

removeChild 删除元素节点

removeAttribute 删除属性节点

removeChild 删除文本节点

要删除某个元素节点有两步

第一：先获取该元素的父节点

第二：通过父节点，调用removeChild 删除该节点

要删除某个属性节点有两步

第一：先获取该元素节点

第二：元素节点，调用removeAttribute删除指定属性节点

删除文本节点

1. 通过childNodes[0] 获取文本节点

注:children[0] 只能获取第一个子元素节点，不能获取文本节点

2. 通过removeChild删除该文本节点

但是这种方式比较麻烦，一般都是直接通过innerHTML设置为空即可。

注: 通过innerHTML=""的方式，同样会导致文本子节点被删除。

parentDiv.innerHTML="";

## 替换节点

替换节点也需要先获取父节点，然后通过父节点替换子节点。

1. 获取父节点

2. 创建子节点

3. 获取被替换子节点

4. 通过replaceChild进行替换

注: replaceChild 第一个参数是保留的节点，第二个参数是被替换的节点

parentNode.replaceChild(kept,replaced);

## 插入节点

appendChild 追加节点 只能加在最后面

insertBefore 在前方插入节点 在指定位置插入节点

parentNode.insertBefore(d25,d3); //将d25节点插入到d3节点前

# JSON

**JSON JavaScript 对象表示法（JavaScript Object Notation） 是一种存储数据的方式。JSON 格式是纯文本**

var gareen = {"name":"盖伦","hp":616};

这样就创建了一个JSON 对象

**JSON对象由 名称/值对组成 名称和值之间用冒号:隔开**

**名称必须用双引号" 包含起来**

值可以是任意javascript数据类型，字符串，布尔，数字 ，数组甚至是对象

不同的名称/值对之间用 逗号 , 隔开

## JSON数组

JSON 数组在方括号中书写。

var array = ‘{“arrayname”:[‘+'{"firstName":"Bill","lastName":"Gates" },' +

'{"firstName":"Steve","lastName":"Jobs" },' +

'{"firstName":"Elon","lastName":"Musk" }]}';