# 计算机辅助机械设计

# 教案

# 2017-7

# 第一讲： CAD/CAM技术概述

## 教学目的

1. 简单介绍机械设计（以零件设计为主）的过程；
2. 简介CAD技术的过去现在和未来；
3. 介绍本课程的学习内容、考核方式等。

## 课型：新开课

## 课时：2学时

## 教学重点

1. 机械（零件）设计的过程和几个准则。

## 教学难点

1. 机械（零件）设计的过程和几个准则。

## 教学过程

### 机械零件设计概述

#### 机械设计的基本要求

##### 使用功能要求:

机器应具有预定的使用功能。这主要靠正确地选择机器的工作原理，正确地设计或选用能够全面实现功能要求的执行机构、传动机构和原动机，以及合理地配置必要的辅助系统来实现。

##### 经济性要求:

机器的经济性体现在设计、制造和使用的全过程中，设计机器时就要全面综合地进行考虑。设计制造的经济性表现为机器的成本低，使用经济性表现为高生产率，高效率，低能耗，以及低的管理和维护费用等。

*提高设计和制造经济性指标的主要途径有:*

* 采用先进的现代设计方法(CAD)，使设计参数最优化,达到尽可能精确的计算结果,保证机器足够的可靠性
* 最大限度地采用标准化、系列化及通用化的零、部件
* 尽可能采用新技术、新工艺、新结构和新材料
* 合理地组织设计和制造过程
* 力求改善零件的结构工艺性，使其用料少、易加工、易装配

提高使用经济性指标的主要途径有:

1. 合理地提高机器的机械化和自动化水平，以期提高机器的生产率
2. 选用高效率的传动系统，尽可能减少传动的中间环节，以期降低能源消耗
3. 适当地采用防护及润滑，以延长机器的使用寿命
4. 采用可靠的密封，减少或消除渗漏现象

##### 劳动保护要求:

* 要使机器的操作者方便和安全。因此设计时要按照人机工程学的观点布置各种按钮、手柄，使操作方式符合人们的 心理和习惯。同时，设置完善的安全装置、报警装置、显示装置等
* 改善操作者及机器的环境。所设计的机器应符合劳动保护法规的要求。降低机器运转时的噪声水平，防止有毒、有害介质的渗漏，对废水、废气和废液进行治理

##### 可靠性要求:

机器的可靠度——指在规定的使用时间内和预定的环境下机器能够正常工作的概率。机器由于某种故障而不能完成预定的功能称为失效，它是随机发生的，其原因是零件所受的载荷、环境温度、零件本身物理和机械性能等因素是随机变化的。为了提高零件的可靠性，就应当在工作条件和零件性能两个方面使其变化尽可能小

##### 其它专用要求:

* 对机床有长期保持精度的要求;
* 对飞机有质量小，飞行阻力小而运载能力大的要求；
* 对流动使用的机器有便于安装和拆卸的要求；
* 对大型机器有便于运输的要求等等;

#### 专用术语

##### 机械零件的失效：

机械零件由于某种原因不能正常工作时，称为失效。

零件的失效形式

* 整体断裂
* 过大的残余变形
* 工作表面的过度磨损或损伤
* 破坏正常的工作条件

**失效原因：**

* 强度，刚度，耐磨性，
* 振动稳定性，温度，等等

##### 工作能力(承载能力)

在不发生失效的条件下，零件所能安全工作的限度。通常此限度是对载荷而言，所以习惯上又称为：承载能力。对零件寿命、使用价值起决定性作用的是其中最小的一个工作能力

##### 设计准则:

为防止失效而制定的判定条件。

设计零件时，首先应根据零件的失效形式确定其设计准则以及相应的设计计算方法。一般来讲，有以下几种准则：

**强度准则**

确保零件不发生**断裂破坏**或*过大的塑性变形*，是最基本的设计准则，强度准则是指零件中的应力不得超过许用值。

**刚度准则**

确保零件不发生**过大的弹性变形**，零件在力作用下的弹性变形量不得超过许用值。

**寿命准则**

影响零件使用寿命的因素主要是**腐蚀、磨损和疲劳**，前两者还没有成熟的计算方法，工程中通常是求出使用寿命时的*疲劳极限*作为计算的依据。

**振动稳定性准则**

高速运转机械的设计应注重此项准则, 保证机器中*受激振零件*的固有频率f*f*与*激振源*的频率f\_p*fp*​错开

**可靠性准则**

**可靠度**的定义： 对件数为*N0*的一批零件进行试验，经过时间*t*之后有*N*件仍能正常工作，则该零件在该试验环境条件下工作时间*t*的可靠度定义为:

*R*=*N*0/​*N*​

##### 设计计算:

根据设计准则设计零件形状、尺寸的过程。

##### 校核计算:

先参照有关条件选定尺寸，再进行验算。

#### 机械零件的设计步骤

1. 选择零件类型、结构
2. 计算零件上的载荷
3. 确定计算准则
4. 选择零件的材料
5. 确定零件的基本尺寸
6. 结构设计
7. 校核计算
8. 画出零件工作图
9. 写出计算说明书

### CAD技术概述

#### 2.1 CAD的定义

计算机辅助设计（Computer Aided Design, CAD）技术是指利用电子计算机辅助设计人员处理设计过程中的数字、字符和图形等数据。由于计算机运算速度快、精度高，长于信息存储和记忆，可以进行枯燥无味的重复性工作，这些优点和设计人员的综合分析、形象思维和创造能力结合起来，形成最佳组合，使得产品的设计周期缩短，设计质量提高，设计成本降低，从而提高了产品的市场竞争能力。

#### 2.2 计算机在机械设计中的应用

在机械工程中进行机械设计是非常重要的环节，它是进行生产工作的前提，它也关系到机械性能问题，因为机械的性能关系到设计质量的问题。从狭义的角度分析，机械设计主要是进行技术设计的工作，在广义的角度来分析，机械设计就是按照现有的条件以及设计者的需求进行设计。所以，对机械相关零件形状、能量大小的传递方式、机械相关的工作原理、结构材料的质量等内容进行分析，然后将这些形成较为具体的描述当做进行机械制造的依据。该工作属于一种创造性的设计，同时也是在成功的经验上进行的，所以两种因素更好的结合，才能提高机器制造的质量。在进行机械制造时都是需要计算机进行辅助，比如:计算数据和编辑图片等内容都需要使用计算机中的软件进行。除了这些内容在计算机上完成，还可以现在计算机上把一些图纸数据模拟出来，这样可以利用计算机来分析产品的受力和性能，之后在根据表现的情况进行相应的改善，从而提高产品的质量。在机械设计加工中计算机技术越来越受欢迎了，主要作用包括但不限于下面的应用：

* 电子辅助绘图
* 辅助和自动计算
* 物理（装配，传动）仿真
* 可靠性分析

### 本课程的学习内容、考核方式等

本课程的主要学习内容包括：

1. 掌握计算机辅助和自动计算
2. 掌握参数化绘图和建模技术
3. 基本掌握物理装配和传动仿真
4. 了解其他计算机辅助机械设计技术

考核方式：

平时：40%

大作业：60%

## 作业

（本次无）

## 教学后记

# 第二讲： 计算机数表和线图处理

# ----使用的计算机语言

## 教学目的

1. 介绍本课程进行计算机编程所使用的语言：JavaScript，
2. 介绍进行用户界面交互所需要的技术：HTML+CSS

## 课型：新开课

## 课时：2学时

## 教学重点

1. HTML的标签（tag）和结构
2. CSS的定义和使用
3. HTML的DOM和JavaScript的使用
4. 事件驱动的概念和实践
5. JavaScript的几个基本结构和语句

## 教学难点

1. HTML的DOM
2. 事件驱动
3. JavaScript的动态语言特性

## 教学过程

### HTML

#### HTML标签和HTML版本（使用HTML5）

什么是 HTML？

HTML 是用来描述网页的一种语言。

HTML 指的是超文本标记语言 (Hyper Text Markup Language)

HTML 不是一种编程语言，而是一种标记语言 (markup language)

标记语言是一套标记标签 (markup tag)

HTML 使用标记标签来描述网页

HTML 标签

HTML 标记标签通常被称为 HTML 标签 (HTML tag)。

HTML 标签是由尖括号包围的关键词，比如 <html>

HTML 标签通常是成对出现的，比如 <b> 和 </b>

标签对中的第一个标签是开始标签，第二个标签是结束标签

开始和结束标签也被称为开放标签和闭合标签

HTML 文档 = 网页

HTML 文档描述网页

HTML 文档包含 HTML 标签和纯文本

HTML 文档也被称为网页

Web 浏览器的作用是读取 HTML 文档，并以网页的形式显示出它们。浏览器不会显示 HTML 标签，而是使用标签来解释页面的内容：

<!DOCTYPE HTML>

<html>

<head>

<title>Demo</title>

</head>

<body>

<h1>我的第一个标题</h1>

<p>我的第一个段落。</p>

</body>

</html>

#### HTML基础

HTML主要包括几个部分：HTML标签，head，body，

#### HTML元素

HTML 元素

HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

HTML 元素语法

HTML 元素以开始标签起始

HTML 元素以结束标签终止

元素的内容是开始标签与结束标签之间的内容

某些 HTML 元素具有空内容（empty content）

空元素在开始标签中进行关闭（以开始标签的结束而结束）

大多数 HTML 元素可拥有属性

嵌套的 HTML 元素

大多数 HTML 元素可以嵌套（可以包含其他 HTML 元素）。

HTML 文档由嵌套的 HTML 元素构成。

空的 HTML 元素

没有内容的 HTML 元素被称为空元素。空元素是在开始标签中关闭的。

<br> 就是没有关闭标签的空元素（<br> 标签定义换行）。

在 XHTML、XML 以及未来版本的 HTML 中，所有元素都必须被关闭。

在开始标签中添加斜杠，比如 <br />，是关闭空元素的正确方法，HTML、XHTML 和 XML 都接受这种方式。

即使 <br> 在所有浏览器中都是有效的，但使用 <br /> 其实是更长远的保障。

#### HTML属性

HTML 属性

HTML 标签可以拥有属性。属性提供了有关 HTML 元素的更多的信息。

属性总是以名称/值对的形式出现，比如：name="value"。

属性总是在 HTML 元素的开始标签中规定。

始终为属性值加引号

属性值应该始终被包括在引号内。双引号是最常用的，不过使用单引号也没有问题。

在某些个别的情况下，比如属性值本身就含有双引号，那么您必须使用单引号，例如：

name='Bill "HelloWorld" Gates'

下面列出了适用于大多数 HTML 元素的属性：

|  |  |  |
| --- | --- | --- |
| **属性** | **值** | **描述** |
| class | *classname* | 规定元素的类名（classname） |
| id | *id* | 规定元素的唯一 id |
| style | *style\_definition* | 规定元素的行内样式（inline style） |
| title | *text* | 规定元素的额外信息（可在工具提示中显示 |

#### HTML常见元素：标题，段落，链接，图像，表格

HTML 标题

标题（Heading）是通过 <h1> - <h6> 等标签进行定义的。

<h1> 定义最大的标题。<h6> 定义最小的标题。

HTML 水平线

<hr /> 标签在 HTML 页面中创建水平线。

hr 元素可用于分隔内容。

HTML 注释

可以将注释插入 HTML 代码中，这样可以提高其可读性，使代码更易被人理解。浏览器会忽略注释，也不会显示它们。

注释是这样写的：

实例

<!-- This is a comment -->

HTML 段落

段落是通过 <p> 标签定义的。

实例

<p>This is a paragraph</p>

<p>This is another paragraph</p>

HTML 超链接（链接）

超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，您可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当您把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：

通过使用 href 属性 - 创建指向另一个文档的链接

通过使用 name 属性 - 创建文档内的书签

HTML 链接语法

链接的 HTML 代码很简单。它类似这样：

<a href="url">Link text</a>

href 属性规定链接的目标。

开始标签和结束标签之间的文字被作为超级链接来显示。

HTML 链接 - target 属性

使用 Target 属性，你可以定义被链接的文档在何处显示。

下面的这行会在新窗口打开文档：

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

HTML 链接 - name 属性

name 属性规定锚（anchor）的名称。

您可以使用 name 属性创建 HTML 页面中的书签。

书签不会以任何特殊方式显示，它对读者是不可见的。

当使用命名锚（named anchors）时，我们可以创建直接跳至该命名锚（比如页面中某个小节）的链接，这样使用者就无需不停地滚动页面来寻找他们需要的信息了。

命名锚的语法：

<a name="label">锚（显示在页面上的文本）</a>

提示：锚的名称可以是任何你喜欢的名字。

提示：您可以使用 id 属性来替代 name 属性，命名锚同样有效。

图像标签（<img>）和源属性（Src）

在 HTML 中，图像由 <img> 标签定义。

<img> 是空标签，意思是说，它只包含属性，并且没有闭合标签。

要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

定义图像的语法是：

<img src="url" />

URL 指存储图像的位置。如果名为 "boat.gif" 的图像位于 www.w3school.com.cn 的 images 目录中，那么其 URL 为 http://www.w3school.com.cn/images/boat.gif。

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。

替换文本属性（Alt）

alt 属性用来为图像定义一串预备的可替换的文本。替换文本属性的值是用户定义的。

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

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。此时，浏览器将显示这个替代性的文本而不是图像。为页面上的图像都加上替换文本属性是个好习惯，这样有助于更好的显示信息，并且对于那些使用纯文本浏览器的人来说是非常有用的。

表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

<table border="1">

<tr>

<td>row 1, cell 1</td>

<td>row 1, cell 2</td>

</tr>

<tr>

<td>row 2, cell 1</td>

<td>row 2, cell 2</td>

</tr>

</table>

表格和边框属性

如果不定义边框属性，表格将不显示边框。有时这很有用，但是大多数时候，我们希望显示边框。

使用边框属性来显示一个带有边框的表格：

<table border="1">

<tr>

<td>Row 1, cell 1</td>

<td>Row 1, cell 2</td>

</tr>

</table>

#### HTML表单：输入标签

HTML 表单用于搜集不同类型的用户输入。

<form> 元素

HTML 表单用于收集用户输入。

<form> 元素定义 HTML 表单：

实例

<form>

.

form elements

.

</form>

HTML 表单包含表单元素。

表单元素指的是不同类型的 input 元素、复选框、单选按钮、提交按钮等等。

<input> 元素

<input> 元素是最重要的表单元素。

<input> 元素有很多形态，根据不同的 type 属性。

这是本章中使用的类型：

类型 描述

text 定义常规文本输入。

radio 定义单选按钮输入（选择多个选择之一）

submit 定义提交按钮（提交表单）

文本输入

<input type="text"> 定义用于文本输入的单行输入字段：

实例

<form>

First name:<br>

<input type="text" name="firstname">

<br>

Last name:<br>

<input type="text" name="lastname">

</form>

单选按钮输入

<input type="radio"> 定义单选按钮。

单选按钮允许用户在有限数量的选项中选择其中之一：

实例

<form>

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

<br>

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

</form>

用 <fieldset> 组合表单数据

<fieldset> 元素组合表单中的相关数据

<legend> 元素为 <fieldset> 元素定义标题。

实例

<form action="action\_page.php">

<fieldset>

<legend>Personal information:</legend>

First name:<br>

<input type="text" name="firstname" value="Mickey">

<br>

Last name:<br>

<input type="text" name="lastname" value="Mouse">

<br><br>

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

</form>

<select> 元素（下拉列表）

<select> 元素定义下拉列表：

实例

<select name="cars">

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

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

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

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

</select>

<textarea> 元素

<textarea> 元素定义多行输入字段（文本域）：

实例

<textarea name="message" rows="10" cols="30">

The cat was playing in the garden.

</textarea>

<button> 元素

<button> 元素定义可点击的按钮：

实例

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

#### HTML格式和样式

文本格式化标签

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<b>](http://www.w3school.com.cn/tags/tag_font_style.asp) | 定义粗体文本。 |
| [<big>](http://www.w3school.com.cn/tags/tag_font_style.asp) | 定义大号字。 |
| [<em>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义着重文字。 |
| [<i>](http://www.w3school.com.cn/tags/tag_font_style.asp) | 定义斜体字。 |
| [<small>](http://www.w3school.com.cn/tags/tag_font_style.asp) | 定义小号字。 |
| [<strong>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义加重语气。 |
| [<sub>](http://www.w3school.com.cn/tags/tag_sup.asp) | 定义下标字。 |
| [<sup>](http://www.w3school.com.cn/tags/tag_sup.asp) | 定义上标字。 |
| [<ins>](http://www.w3school.com.cn/tags/tag_ins.asp) | 定义插入字。 |
| [<del>](http://www.w3school.com.cn/tags/tag_del.asp) | 定义删除字。 |
| [<s>](http://www.w3school.com.cn/tags/tag_strike.asp) | 不赞成使用。使用 <del> 代替。 |
| [<strike>](http://www.w3school.com.cn/tags/tag_strike.asp) | 不赞成使用。使用 <del> 代替。 |
| [<u>](http://www.w3school.com.cn/tags/tag_u.asp) | 不赞成使用。使用样式（style）代替。 |

“计算机输出”标签

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<code>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义计算机代码。 |
| [<kbd>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义键盘码。 |
| [<samp>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义计算机代码样本。 |
| [<tt>](http://www.w3school.com.cn/tags/tag_font_style.asp) | 定义打字机代码。 |
| [<var>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义变量。 |
| [<pre>](http://www.w3school.com.cn/tags/tag_pre.asp) | 定义预格式文本。 |
| <listing> | 不赞成使用。使用 <pre> 代替。 |
| <plaintext> | 不赞成使用。使用 <pre> 代替。 |
| <xmp> | 不赞成使用。使用 <pre> 代替。 |

引用、引用和术语定义

|  |  |
| --- | --- |
| **标签** | **描述** |
| [<abbr>](http://www.w3school.com.cn/tags/tag_abbr.asp) | 定义缩写。 |
| [<acronym>](http://www.w3school.com.cn/tags/tag_acronym.asp) | 定义首字母缩写。 |
| [<address>](http://www.w3school.com.cn/tags/tag_address.asp) | 定义地址。 |
| [<bdo>](http://www.w3school.com.cn/tags/tag_bdo.asp) | 定义文字方向。 |
| [<blockquote>](http://www.w3school.com.cn/tags/tag_blockquote.asp) | 定义长的引用。 |
| [<q>](http://www.w3school.com.cn/tags/tag_q.asp) | 定义短的引用语。 |
| [<cite>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义引用、引证。 |
| [<dfn>](http://www.w3school.com.cn/tags/tag_phrase_elements.asp) | 定义一个定义项目。 |

HTML 的 style 属性

style 属性的作用：

提供了一种改变所有 HTML 元素的样式的通用方法。

样式是 HTML 4 引入的，它是一种新的首选的改变 HTML 元素样式的方式。通过 HTML 样式，能够通过使用 style 属性直接将样式添加到 HTML 元素，或者间接地在独立的样式表中（CSS 文件）进行定义。

HTML 样式实例 - 背景颜色

background-color 属性为元素定义了背景颜色：

<html>

<body style="background-color:yellow">

<h2 style="background-color:red">This is a heading</h2>

<p style="background-color:green">This is a paragraph.</p>

</body>

</html>

HTML 样式实例 - 字体、颜色和尺寸

font-family、color 以及 font-size 属性分别定义元素中文本的字体系列、颜色和字体尺寸：

<html>

<body>

<h1 style="font-family:verdana">A heading</h1>

<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>

</body>

</html>

HTML 样式实例 - 文本对齐

text-align 属性规定了元素中文本的水平对齐方式：

<html>

<body>

<h1 style="text-align:center">This is a heading</h1>

<p>The heading above is aligned to the center of this page.</p>

</body>

</html>

#### 颜色

W3C 的 HTML 4.0 标准仅支持 16 种颜色名，它们是：aqua、black、blue、fuchsia、gray、green、lime、maroon、navy、olive、purple、red、silver、teal、white、yellow

其他的颜色用16进制值来表示：[#000000](http://www.w3school.com.cn/tiy/color.asp?hex=000000)， #0000FF等

### CSS

#### 2.1、CSS的定义和基本概念

CSS 概述

CSS 指层叠样式表 (Cascading Style Sheets)

样式定义如何显示 HTML 元素

样式通常存储在样式表中

把样式添加到 HTML 4.0 中，是为了解决内容与表现分离的问题

外部样式表可以极大提高工作效率

外部样式表通常存储在 CSS 文件中

多个样式定义可层叠为一

样式解决了一个普遍的问题

HTML 标签原本被设计为用于定义文档内容。通过使用 <h1>、<p>、<table> 这样的标签，HTML 的初衷是表达“这是标题”、“这是段落”、“这是表格”之类的信息。同时文档布局由浏览器来完成，而不使用任何的格式化标签。

由于两种主要的浏览器（Netscape 和 Internet Explorer）不断地将新的 HTML 标签和属性（比如字体标签和颜色属性）添加到 HTML 规范中，创建文档内容清晰地独立于文档表现层的站点变得越来越困难。

为了解决这个问题，万维网联盟（W3C），这个非营利的标准化联盟，肩负起了 HTML 标准化的使命，并在 HTML 4.0 之外创造出样式（Style）。

所有的主流浏览器均支持层叠样式表。

样式表极大地提高了工作效率

样式表定义如何显示 HTML 元素，就像 HTML 3.2 的字体标签和颜色属性所起的作用那样。样式通常保存在外部的 .css 文件中。通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面的布局和外观。

由于允许同时控制多重页面的样式和布局，CSS 可以称得上 WEB 设计领域的一个突破。作为网站开发者，你能够为每个 HTML 元素定义样式，并将之应用于你希望的任意多的页面中。如需进行全局的更新，只需简单地改变样式，然后网站中的所有元素均会自动地更新。

多重样式将层叠为一个

样式表允许以多种方式规定样式信息。样式可以规定在单个的 HTML 元素中，在 HTML 页的头元素中，或在一个外部的 CSS 文件中。甚至可以在同一个 HTML 文档内部引用多个外部样式表。

层叠次序

当同一个 HTML 元素被不止一个样式定义时，会使用哪个样式呢？

一般而言，所有的样式会根据下面的规则层叠于一个新的虚拟样式表中，其中数字 4 拥有最高的优先权。

浏览器缺省设置

外部样式表

内部样式表（位于 <head> 标签内部）

内联样式（在 HTML 元素内部）

因此，内联样式（在 HTML 元素内部）拥有最高的优先权，这意味着它将优先于以下的样式声明：<head> 标签中的样式声明，外部样式表中的样式声明，或者浏览器中的样式声明（缺省值）。

#### 2.2、CSS的基础语法

CSS 语法

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明。

selector {declaration1; declaration2; ... declarationN }

选择器通常是您需要改变样式的 HTML 元素。

每条声明由一个属性和一个值组成。

属性（property）是您希望设置的样式属性（style attribute）。每个属性有一个值。属性和值被冒号分开。

selector {property: value}

下面这行代码的作用是将 h1 元素内的文字颜色定义为红色，同时将字体大小设置为 14 像素。

在这个例子中，h1 是选择器，color 和 font-size 是属性，red 和 14px 是值。

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

值的不同写法和单位

除了英文单词 red，我们还可以使用十六进制的颜色值 #ff0000：

p { color: #ff0000; }

为了节约字节，我们可以使用 CSS 的缩写形式：

p { color: #f00; }

我们还可以通过两种方法使用 RGB 值：

p { color: rgb(255,0,0); }

p { color: rgb(100%,0%,0%); }

请注意，当使用 RGB 百分比时，即使当值为 0 时也要写百分比符号。但是在其他的情况下就不需要这么做了。比如说，当尺寸为 0 像素时，0 之后不需要使用 px 单位，因为 0 就是 0，无论单位是什么。

记得写引号

提示：如果值为若干单词，则要给值加引号：

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

多重声明：

提示：如果要定义不止一个声明，则需要用分号将每个声明分开。下面的例子展示出如何定义一个红色文字的居中段落。最后一条规则是不需要加分号的，因为分号在英语中是一个分隔符号，不是结束符号。然而，大多数有经验的设计师会在每条声明的末尾都加上分号，这么做的好处是，当你从现有的规则中增减声明时，会尽可能地减少出错的可能性。就像这样：

p {text-align:center; color:red;}

你应该在每行只描述一个属性，这样可以增强样式定义的可读性，就像这样：

p {

text-align: center;

color: black;

font-family: arial;

}

空格和大小写

大多数样式表包含不止一条规则，而大多数规则包含不止一个声明。多重声明和空格的使用使得样式表更容易被编辑：

body {

color: #000;

background: #fff;

margin: 0;

padding: 0;

font-family: Georgia, Palatino, serif;

}

是否包含空格不会影响 CSS 在浏览器的工作效果，同样，与 XHTML 不同，CSS 对大小写不敏感。不过存在一个例外：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。

#### 2.3、CSS选择器

CSS 元素选择器

最常见的 CSS 选择器是元素选择器。换句话说，文档的元素就是最基本的选择器。

如果设置 HTML 的样式，选择器通常将是某个 HTML 元素，比如 p、h1、em、a，甚至可以是 html 本身：

html {color:black;}

h1 {color:blue;}

h2 {color:silver;}

类型选择器

在 W3C 标准中，元素选择器又称为类型选择器（type selector）。

“类型选择器匹配文档语言元素类型的名称。类型选择器匹配文档树中该元素类型的每一个实例。”

下面的规则匹配文档树中所有 h1 元素：

h1 {font-family: sans-serif;}

选择器分组

假设希望 h2 元素和段落都有灰色。为达到这个目的，最容易的做法是使用以下声明：

h2, p {color:gray;}

将 h2 和 p 选择器放在规则左边，然后用逗号分隔，就定义了一个规则。其右边的样式（color:gray;）将应用到这两个选择器所引用的元素。逗号告诉浏览器，规则中包含两个不同的选择器。如果没有这个逗号，那么规则的含义将完全不同。参见后代选择器。

可以将任意多个选择器分组在一起，对此没有任何限制。

CSS 类选择器

类选择器允许以一种独立于文档元素的方式来指定样式。

该选择器可以单独使用，也可以与其他元素结合使用。

提示：只有适当地标记文档后，才能使用这些选择器，所以使用这两种选择器通常需要先做一些构想和计划。

要应用样式而不考虑具体设计的元素，最常用的方法就是使用类选择器。

<h1 class="important">

This heading is very important.

</h1>

<p class="important">

This paragraph is very important.

</p>

CSS：

.important {color:red;}

结合元素选择器

类选择器可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本：

p.important {color:red;}

选择器现在会匹配 class 属性包含 important 的所有 p 元素，但是其他任何类型的元素都不匹配，不论是否有此 class 属性。选择器 p.important 解释为：“其 class 属性值为 important 的所有段落”。 因为 h1 元素不是段落，这个规则的选择器与之不匹配，因此 h1 元素不会变成红色文本。

如果你确实希望为 h1 元素指定不同的样式，可以使用选择器 h1.important：

p.important {color:red;}

h1.important {color:blue;}

CSS 多类选择器

在上一节中，我们处理了 class 值中包含一个词的情况。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。例如，如果希望将一个特定的元素同时标记为重要（important）和警告（warning），就可以写作：

<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;}

ID 选择器

<p id="intro">This is a paragraph of introduction.</p>

CSS：

#intro {font-weight:bold;}

属性选择器

a[href] {color:red;}

根据部分属性值选择

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;}

#### 2.4、CSS的框模型

#### 2.5、CSS的定位方法

CSS 定位和浮动

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。显然，这个功能非常强大，也很让人吃惊。要知道，用户代理对 CSS2 中定位的支持远胜于对其它方面的支持，对此不应感到奇怪。

另一方面，CSS1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

### JavaScript

#### 3.1、JavaScript 简介

JavaScript 是脚本语言

JavaScript 是一种轻量级的编程语言。

JavaScript 是可插入 HTML 页面的编程代码。

JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。

JavaScript 很容易学习。

#### 3.2、JavaScript 使用

HTML 中的脚本必须位于 <script> 与 </script> 标签之间。

脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。

外部的 JavaScript

也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

实例

<!DOCTYPE html>

<html>

<body>

<script src="myScript.js"></script>

</body>

</html>

#### 3.3、JavaScript 输出

操作 HTML 元素

如需从 JavaScript 访问某个 HTML 元素，您可以使用 document.getElementById(id) 方法。

请使用 "id" 属性来标识 HTML 元素：

例子

通过指定的 id 来访问 HTML 元素，并改变其内容：

<!DOCTYPE html>

<html>

<body>

<h1>我的第一张网页</h1>

<p id="demo">我的第一个段落</p>

<script>

document.getElementById("demo").innerHTML="我的第一段 JavaScript";

</script>

</body>

</html>

写到文档输出

下面的例子直接把 <p> 元素写到 HTML 文档输出中：

实例

<!DOCTYPE html>

<html>

<body>

<h1>我的第一张网页</h1>

<script>

document.write("<p>我的第一段 JavaScript</p>");

</script>

</body>

</html>

#### 3.4、JavaScript 语句

JavaScript 语句

JavaScript 语句向浏览器发出的命令。语句的作用是告诉浏览器该做什么。

下面的 JavaScript 语句向 id="demo" 的 HTML 元素输出文本 "Hello World"：

document.getElementById("demo").innerHTML="Hello World";

分号 ;

分号用于分隔 JavaScript 语句。

通常我们在每条可执行的语句结尾添加分号。

使用分号的另一用处是在一行中编写多条语句。

提示：您也可能看到不带有分号的案例。

在 JavaScript 中，用分号来结束语句是可选的。

JavaScript 代码

JavaScript 代码（或者只有 JavaScript）是 JavaScript 语句的序列。

浏览器会按照编写顺序来执行每条语句。

本例将操作两个 HTML 元素：

实例

document.getElementById("demo").innerHTML="Hello World";

document.getElementById("myDIV").innerHTML="How are you?";

JavaScript 代码块

JavaScript 语句通过代码块的形式进行组合。

块由左花括号开始，由右花括号结束。

块的作用是使语句序列一起执行。

JavaScript 函数是将语句组合在块中的典型例子。

下面的例子将运行可操作两个 HTML 元素的函数：

实例

function myFunction()

{

document.getElementById("demo").innerHTML="Hello World";

document.getElementById("myDIV").innerHTML="How are you?";

}

JavaScript 对大小写敏感。

JavaScript 对大小写是敏感的。

当编写 JavaScript 语句时，请留意是否关闭大小写切换键。

函数 getElementById 与 getElementbyID 是不同的。

同样，变量 myVariable 与 MyVariable 也是不同的。

空格

JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。下面的两行代码是等效的：

var name="Hello";

var name = "Hello";

对代码行进行折行

您可以在文本字符串中使用反斜杠对代码行进行换行。下面的例子会正确地显示：

document.write("Hello \

World!");

不过，您不能像这样折行：

document.write \

("Hello World!");

#### 3.5、JavaScript 注释

JavaScript 注释可用于提高代码的可读性。

JavaScript 注释

JavaScript 不会执行注释。

我们可以添加注释来对 JavaScript 进行解释，或者提高代码的可读性。

单行注释以 // 开头。

例子

下面的例子使用单行注释来解释代码：

// 输出标题：

document.getElementById("myH1").innerHTML="Welcome to my Homepage";

// 输出段落：

document.getElementById("myP").innerHTML="This is my first paragraph.";

JavaScript 多行注释

多行注释以 /\* 开始，以 \*/ 结尾。

下面的例子使用多行注释来解释代码：

例子

/\*

下面的这些代码会输出

一个标题和一个段落

并将代表主页的开始

\*/

document.getElementById("myH1").innerHTML="Welcome to my Homepage";

document.getElementById("myP").innerHTML="This is my first paragraph.";

#### 3.6、JavaScript 变量

变量是存储信息的容器。

实例

var x=2;

var y=3;

var z=x+y;

JavaScript 变量

与代数一样，JavaScript 变量可用于存放值（比如 x=2）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

变量必须以字母开头

变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）

变量名称对大小写敏感（y 和 Y 是不同的变量）

提示：JavaScript 语句和 JavaScript 变量都对大小写敏感。

JavaScript 数据类型

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。

在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

例子

var pi=3.14;

var name="Bill Gates";

var answer='Yes I am!';

一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

var name="Gates", age=56, job="CEO";

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

var carname;

#### 3.7、JavaScript 数据类型

JavaScript 拥有动态类型

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

实例

var x // x 为 undefined

var x = 6; // x 为数字

var x = "Bill"; // x 为字符串

JavaScript 字符串

字符串是存储字符（比如 "Bill Gates"）的变量。

字符串可以是引号中的任意文本。您可以使用单引号或双引号：

实例

var carname="Bill Gates";

var carname='Bill Gates';

您可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

实例

var answer="Nice to meet you!";

var answer="He is called 'Bill'";

var answer='He is called "Bill"';

JavaScript 数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

实例

var x1=34.00; //使用小数点来写

var x2=34; //不使用小数点来写

极大或极小的数字可以通过科学（指数）计数法来书写：

实例

var y=123e5; // 12300000

var z=123e-5; // 0.00123

JavaScript 布尔

布尔（逻辑）只能有两个值：true 或 false。

var x=true

var y=false

布尔常用在条件测试中。您将在本教程稍后的章节中学到更多关于条件测试的知识。

JavaScript 数组

下面的代码创建名为 cars 的数组：

var cars=new Array();

cars[0]="Audi";

cars[1]="BMW";

cars[2]="Volvo";

或者 (condensed array):

var cars=new Array("Audi","BMW","Volvo");

或者 (literal array):

实例

var cars=["Audi","BMW","Volvo"];

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

您将在本教程稍后的章节中学到更多关于数组的知识。

JavaScript 对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

var person={firstname:"Bill", lastname:"Gates", id:5566};

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

var person={

firstname : "Bill",

lastname : "Gates",

id : 5566

};

对象属性有两种寻址方式：

实例

name=person.lastname;

name=person["lastname"];

Undefined 和 Null

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

实例

cars=null;

person=null;

声明变量类型

当您声明新变量时，可以使用关键词 "new" 来声明其类型：

var carname=new String;

var x= new Number;

var y= new Boolean;

var cars= new Array;

var person= new Object;

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

#### 3.8、JavaScript 对象

JavaScript 中的所有事物都是对象：字符串、数字、数组、日期，等等。

在 JavaScript 中，对象是拥有属性和方法的数据。

属性和方法

属性是与对象相关的值。

方法是能够在对象上执行的动作。

举例：汽车就是现实生活中的对象。

汽车的属性：

car.name=Fiat

car.model=500

car.weight=850kg

car.color=white

汽车的方法：

car.start()

car.drive()

car.brake()

汽车的属性包括名称、型号、重量、颜色等。

所有汽车都有这些属性，但是每款车的属性都不尽相同。

汽车的方法可以是启动、驾驶、刹车等。

所有汽车都拥有这些方法，但是它们被执行的时间都不尽相同。

JavaScript 中的对象

在 JavaScript 中，对象是数据（变量），拥有属性和方法。

当您像这样声明一个 JavaScript 变量时：

var txt = "Hello";

您实际上已经创建了一个 JavaScript 字符串对象。字符串对象拥有内建的属性 length。对于上面的字符串来说，length 的值是 5。字符串对象同时拥有若干个内建的方法。

属性：

txt.length=5

方法：

txt.indexOf()

txt.replace()

txt.search()

提示：在面向对象的语言中，属性和方法常被称为对象的成员。

在本教程稍后的章节中，您将学到有关字符串对象的更多属性和方法。

创建 JavaScript 对象

JavaScript 中的几乎所有事务都是对象：字符串、数字、数组、日期、函数，等等。

你也可以创建自己的对象。

本例创建名为 "person" 的对象，并为其添加了四个属性：

实例

person=new Object();

person.firstname="Bill";

person.lastname="Gates";

person.age=56;

person.eyecolor="blue";

创建新 JavaScript 对象有很多不同的方法，并且您还可以向已存在的对象添加属性和方法。

您将在本教程稍后的章节学到更多相关的内容。

访问对象的属性

访问对象属性的语法是：

objectName.propertyName

本例使用 String 对象的 length 属性来查找字符串的长度：

var message="Hello World!";

var x=message.length;

在以上代码执行后，x 的值是：

12

访问对象的方法

您可以通过下面的语法调用方法：

objectName.methodName()

这个例子使用 String 对象的 toUpperCase() 方法来把文本转换为大写：

var message="Hello world!";

var x=message.toUpperCase();

在以上代码执行后，x 的值是：

HELLO WORLD!

#### 3.9、JavaScript 函数

JavaScript 函数语法

函数就是包裹在花括号中的代码块，前面使用了关键词 function：

function functionname()

{

这里是要执行的代码

}

当调用该函数时，会执行函数内的代码。

可以在某事件发生时直接调用函数（比如当用户点击按钮时），并且可由 JavaScript 在任何位置进行调用。

提示：JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

调用带参数的函数

在调用函数时，您可以向其传递值，这些值被称为参数。

这些参数可以在函数中使用。

您可以发送任意多的参数，由逗号 (,) 分隔：

myFunction(argument1,argument2)

当您声明函数时，请把参数作为变量来声明：

function myFunction(var1,var2)

{

这里是要执行的代码

}

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

实例

<button onclick="myFunction('Bill Gates','CEO')">点击这里</button>

<script>

function myFunction(name,job)

{

alert("Welcome " + name + ", the " + job);

}

</script>

带有返回值的函数

有时，我们会希望函数将值返回调用它的地方。

通过使用 return 语句就可以实现。

在使用 return 语句时，函数会停止执行，并返回指定的值。

语法

function myFunction()

{

var x=5;

return x;

}

上面的函数会返回值 5。

注释：整个 JavaScript 并不会停止执行，仅仅是函数。JavaScript 将继续执行代码，从调用函数的地方。

函数调用将被返回值取代：

var myVar=myFunction();

myVar 变量的值是 5，也就是函数 "myFunction()" 所返回的值。

即使不把它保存为变量，您也可以使用返回值：

document.getElementById("demo").innerHTML=myFunction();

"demo" 元素的 innerHTML 将成为 5，也就是函数 "myFunction()" 所返回的值。

您可以使返回值基于传递到函数中的参数：

实例

计算两个数字的乘积，并返回结果：

function myFunction(a,b)

{

return a\*b;

}

document.getElementById("demo").innerHTML=myFunction(4,3);

"demo" 元素的 innerHTML 将是：

12

在您仅仅希望退出函数时 ，也可使用 return 语句。返回值是可选的：

function myFunction(a,b)

{

if (a>b)

{

return;

}

x=a+b

}

如果 a 大于 b，则上面的代码将退出函数，并不会计算 a 和 b 的总和。

局部 JavaScript 变量

在 JavaScript 函数内部声明的变量（使用 var）是局部变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。

您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。

只要函数运行完毕，本地变量就会被删除。

全局 JavaScript 变量

在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它。

JavaScript 变量的生存期

JavaScript 变量的生命期从它们被声明的时间开始。

局部变量会在函数运行以后被删除。

全局变量会在页面关闭后被删除。

向未声明的 JavaScript 变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

这条语句：

carname="Volvo";

将声明一个全局变量 carname，即使它在函数内执行。

#### 3.10、JavaScript 运算符

给定 **y=5**，下面的表格解释了这些算术运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **描述** | **例子** | **结果** |
| + | 加 | x=y+2 | x=7 |
| - | 减 | x=y-2 | x=3 |
| \* | 乘 | x=y\*2 | x=10 |
| / | 除 | x=y/2 | x=2.5 |
| % | 求余数 (保留整数) | x=y%2 | x=1 |
| ++ | 累加 | x=++y | x=6 |
| -- | 递减 | x=--y | x=4 |

给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **例子** | **等价于** | **结果** |
| = | x=y |  | x=5 |
| += | x+=y | x=x+y | x=15 |
| -= | x-=y | x=x-y | x=5 |
| \*= | x\*=y | x=x\*y | x=50 |
| /= | x/=y | x=x/y | x=2 |
| %= | x%=y | x=x%y | x=0 |

+ 运算符用于把文本值或字符串变量加起来（连接起来）。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

txt1="What a very";

txt2="nice day";

txt3=txt1+txt2;

在以上语句执行后，变量 txt3 包含的值是 "What a verynice day"。

要想在两个字符串之间增加空格，需要把空格插入一个字符串之中：

txt1="What a very ";

txt2="nice day";

txt3=txt1+txt2;

或者把空格插入表达式中：

txt1="What a very";

txt2="nice day";

txt3=txt1+" "+txt2;

在以上语句执行后，变量 txt3 包含的值是：

"What a very nice day"

如果把数字与字符串相加，结果将成为字符串。

#### 3.11、JavaScript 比较和逻辑运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

给定 x=5，下面的表格解释了比较运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| == | 等于 | x==8 为 false |
| === | 全等（值和类型） | x===5 为 true；x==="5" 为 false |
| != | 不等于 | x!=8 为 true |
| > | 大于 | x>8 为 false |
| < | 小于 | x<8 为 true |
| >= | 大于或等于 | x>=8 为 false |
| <= | 小于或等于 | x<=8 为 true |

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| && | and | (x < 10 && y > 1) 为 true |
| || | or | (x==5 || y==5) 为 false |
| ! | not | !(x==y) 为 true |

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

语法

variablename=(condition)?value1:value2

例子

greeting=(visitor=="PRES")?"Dear President ":"Dear ";

如果变量 visitor 中的值是 "PRES"，则向变量 greeting 赋值 "Dear President "，否则赋值 "Dear"。

#### 3.12、条件跳转和循环

条件语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

if 语句 - 只有当指定条件为 true 时，使用该语句来执行代码

if...else 语句 - 当条件为 true 时执行代码，当条件为 false 时执行其他代码

if...else if....else 语句 - 使用该语句来选择多个代码块之一来执行

switch 语句 - 使用该语句来选择多个代码块之一来执行

JavaScript Switch 语句

请使用 switch 语句来选择要执行的多个代码块之一。

语法

switch(n)

{

case 1:

执行代码块 1

break;

case 2:

执行代码块 2

break;

default:

n 与 case 1 和 case 2 不同时执行的代码

}

工作原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 break 来阻止代码自动地向下一个 case 运行。

不同类型的循环

JavaScript 支持不同类型的循环：

for - 循环代码块一定的次数

for/in - 循环遍历对象的属性

while - 当指定的条件为 true 时循环指定的代码块

do/while - 同样当指定的条件为 true 时循环指定的代码块

例1：遍历数组

for (var i=0,len=cars.length; i<len; i++)

{

document.write(cars[i] + "<br>");

}

例2：遍历对象属性

var person={fname:"John", name:"Doe",age:25};

for (x in person)

{

txt=txt + person[x];

}

Break 语句

我们已经在本教程稍早的章节中见到过 break 语句。它用于跳出 switch() 语句。

break 语句可用于跳出循环。

break 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：

Continue 语句

continue 语句中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代。

JavaScript 标签

正如您在 switch 语句那一章中看到的，可以对 JavaScript 语句进行标记。

如需标记 JavaScript 语句，请在语句之前加上冒号：

label:

语句

break 和 continue 语句仅仅是能够跳出代码块的语句。

语法

break labelname;

continue labelname;

continue 语句（带有或不带标签引用）只能用在循环中。

break 语句（不带标签引用），只能用在循环或 switch 中。

通过标签引用，break 语句可用于跳出任何 JavaScript 代码块：

#### 3.13、错误处理

JavaScript 测试和捕捉

try 语句允许我们定义在执行时进行错误测试的代码块。

catch 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。

JavaScript 语句 try 和 catch 是成对出现的。

语法

try

{

//在这里运行代码

}

catch(err)

{

//在这里处理错误

}

实例

在下面的例子中，我们故意在 try 块的代码中写了一个错字。

catch 块会捕捉到 try 块中的错误，并执行代码来处理它。

<!DOCTYPE html>

<html>

<head>

<script>

var txt="";

function message()

{

try

{

adddlert("Welcome guest!");

}

catch(err)

{

txt="There was an error on this page.\n\n";

txt+="Error description: " + err.message + "\n\n";

txt+="Click OK to continue.\n\n";

alert(txt);

}

}

</script>

</head>

<body>

<input type="button" value="View message" onclick="message()">

</body>

</html>

Throw 语句

throw 语句允许我们创建自定义错误。

正确的技术术语是：创建或抛出异常（exception）。

如果把 throw 与 try 和 catch 一起使用，那么您能够控制程序流，并生成自定义的错误消息。

语法

throw exception

异常可以是 JavaScript 字符串、数字、逻辑值或对象。

实例

本例检测输入变量的值。如果值是错误的，会抛出一个异常（错误）。catch 会捕捉到这个错误，并显示一段自定义的错误消息：

<script>

function myFunction()

{

try

{

var x=document.getElementById("demo").value;

if(x=="") throw "empty";

if(isNaN(x)) throw "not a number";

if(x>10) throw "too high";

if(x<5) throw "too low";

}

catch(err)

{

var y=document.getElementById("mess");

y.innerHTML="Error: " + err + ".";

}

}

</script>

<h1>My First JavaScript</h1>

<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">

<button type="button" onclick="myFunction()">Test Input</button>

<p id="mess"></p>

#### 3.14、常见JavaScript对象：array, string, Math, Date, Timer

JavaScript 数字

JavaScript 数字可以使用也可以不使用小数点来书写：

实例

var pi=3.14; // 使用小数点

var x=34; // 不使用小数点

极大或极小的数字可通过科学（指数）计数法来写：

实例

var y=123e5; // 12300000

var z=123e-5; // 0.00123

所有 JavaScript 数字均为 64 位

JavaScript 不是类型语言。与许多其他编程语言不同，JavaScript 不定义不同类型的数字，比如整数、短、长、浮点等等。

JavaScript 中的所有数字都存储为根为 10 的 64 位（8 比特），浮点数。

精度

整数（不使用小数点或指数计数法）最多为 15 位。

小数的最大位数是 17，但是浮点运算并不总是 100% 准确：

实例

var x=0.2+0.1;

八进制和十六进制

如果前缀为 0，则 JavaScript 会把数值常量解释为八进制数，如果前缀为 0 和 "x"，则解释为十六进制数。

实例

var y=0377;

var z=0xFF;

提示：绝不要在数字前面写零，除非您需要进行八进制转换。

数字属性和方法

属性：

MAX\_VALUE

MIN\_VALUE

NEGATIVE\_INFINITIVE

POSITIVE\_INFINITIVE

NaN

方法：

toExponential()

toFixed()

toPrecision()

toString()

valueOf()

String 对象

String 对象用于处理文本（字符串）。

创建 String 对象的语法：

new String(s);

String(s);

参数

参数 s 是要存储在 String 对象中或转换成原始字符串的值。

返回值

当 String() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 s 或 s 的字符串表示。

当不用 new 运算符调用 String() 时，它只把 s 转换成原始的字符串，并返回转换后的值。

String 对象属性

属性 描述

constructor 对创建该对象的函数的引用

length 字符串的长度

prototype 允许您向对象添加属性和方法

String 对象方法

方法 描述

anchor() 创建 HTML 锚。

big() 用大号字体显示字符串。

blink() 显示闪动字符串。

bold() 使用粗体显示字符串。

charAt() 返回在指定位置的字符。

charCodeAt() 返回在指定的位置的字符的 Unicode 编码。

concat() 连接字符串。

fixed() 以打字机文本显示字符串。

fontcolor() 使用指定的颜色来显示字符串。

fontsize() 使用指定的尺寸来显示字符串。

fromCharCode() 从字符编码创建一个字符串。

indexOf() 检索字符串。

italics() 使用斜体显示字符串。

lastIndexOf() 从后向前搜索字符串。

link() 将字符串显示为链接。

localeCompare() 用本地特定的顺序来比较两个字符串。

match() 找到一个或多个正则表达式的匹配。

replace() 替换与正则表达式匹配的子串。

search() 检索与正则表达式相匹配的值。

slice() 提取字符串的片断，并在新的字符串中返回被提取的部分。

small() 使用小字号来显示字符串。

split() 把字符串分割为字符串数组。

strike() 使用删除线来显示字符串。

sub() 把字符串显示为下标。

substr() 从起始索引号提取字符串中指定数目的字符。

substring() 提取字符串中两个指定的索引号之间的字符。

sup() 把字符串显示为上标。

toLocaleLowerCase() 把字符串转换为小写。

toLocaleUpperCase() 把字符串转换为大写。

toLowerCase() 把字符串转换为小写。

toUpperCase() 把字符串转换为大写。

toSource() 代表对象的源代码。

toString() 返回字符串。

valueOf() 返回某个字符串对象的原始值。

String 对象描述

字符串是 JavaScript 的一种基本的数据类型。

String 对象的 length 属性声明了该字符串中的字符数。

String 类定义了大量操作字符串的方法，例如从字符串中提取字符或子串，或者检索字符或子串。

需要注意的是，JavaScript 的字符串是不可变的（immutable），String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase() 这样的方法，返回的是全新的字符串，而不是修改原始字符串。

在较早的 Netscape 代码基的 JavaScript 实现中（例如 Firefox 实现中），字符串的行为就像只读的字符数组。例如，从字符串 s 中提取第三个字符，可以用 s[2] 代替更加标准的 s.charAt(2)。此外，对字符串应用 for/in 循环时，它将枚举字符串中每个字符的数组下标（但要注意，ECMAScript 标准规定，不能枚举 length 属性）。因为字符串的数组行为不标准，所以应该避免使用它。

Math 对象

Math 对象用于执行数学任务。

使用 Math 的属性和方法的语法：

var pi\_value=Math.PI;

var sqrt\_value=Math.sqrt(15);

注释：Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

Math 对象属性

属性 描述

E 返回算术常量 e，即自然对数的底数（约等于2.718）。

LN2 返回 2 的自然对数（约等于0.693）。

LN10 返回 10 的自然对数（约等于2.302）。

LOG2E 返回以 2 为底的 e 的对数（约等于 1.414）。

LOG10E 返回以 10 为底的 e 的对数（约等于0.434）。

PI 返回圆周率（约等于3.14159）。

SQRT1\_2 返回返回 2 的平方根的倒数（约等于 0.707）。

SQRT2 返回 2 的平方根（约等于 1.414）。

Math 对象方法

方法 描述

abs(x) 返回数的绝对值。

acos(x) 返回数的反余弦值。

asin(x) 返回数的反正弦值。

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

atan2(y,x) 返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。

ceil(x) 对数进行上舍入。

cos(x) 返回数的余弦。

exp(x) 返回 e 的指数。

floor(x) 对数进行下舍入。

log(x) 返回数的自然对数（底为e）。

max(x,y) 返回 x 和 y 中的最高值。

min(x,y) 返回 x 和 y 中的最低值。

pow(x,y) 返回 x 的 y 次幂。

random() 返回 0 ~ 1 之间的随机数。

round(x) 把数四舍五入为最接近的整数。

sin(x) 返回数的正弦。

sqrt(x) 返回数的平方根。

tan(x) 返回角的正切。

toSource() 返回该对象的源代码。

valueOf() 返回 Math 对象的原始值。

\*\*注意，Math功能太弱，后面我们会用其他库取代。

Array 对象

Array 对象用于在单个的变量中存储多个值。

创建 Array 对象的语法：

new Array();

new Array(size);

new Array(element0, element1, ..., elementn);

参数

参数 size 是期望的数组元素个数。返回的数组，length 字段将被设为 size 的值。

参数 element ..., elementn 是参数列表。当使用这些参数来调用构造函数 Array() 时，新创建的数组的元素就会被初始化为这些值。它的 length 字段也会被设置为参数的个数。

返回值

返回新创建并被初始化了的数组。

如果调用构造函数 Array() 时没有使用参数，那么返回的数组为空，length 字段为 0。

当调用构造函数时只传递给它一个数字参数，该构造函数将返回具有指定个数、元素为 undefined 的数组。

当其他参数调用 Array() 时，该构造函数将用参数指定的值初始化数组。

当把构造函数作为函数调用，不使用 new 运算符时，它的行为与使用 new 运算符调用它时的行为完全一样。

Array 对象属性

属性 描述

constructor 返回对创建此对象的数组函数的引用。

length 设置或返回数组中元素的数目。

prototype 使您有能力向对象添加属性和方法。

Array 对象方法

方法 描述

concat() 连接两个或更多的数组，并返回结果。

join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

pop() 删除并返回数组的最后一个元素

push() 向数组的末尾添加一个或更多元素，并返回新的长度。

reverse() 颠倒数组中元素的顺序。

shift() 删除并返回数组的第一个元素

slice() 从某个已有的数组返回选定的元素

sort() 对数组的元素进行排序

splice() 删除元素，并向数组添加新元素。

toSource() 返回该对象的源代码。

toString() 把数组转换为字符串，并返回结果。

toLocaleString() 把数组转换为本地数组，并返回结果。

unshift() 向数组的开头添加一个或更多元素，并返回新的长度。

valueOf() 返回数组对象的原始值

Date 对象

Date 对象用于处理日期和时间。

创建 Date 对象的语法：

var myDate=new Date()

注释：Date 对象会自动把当前日期和时间保存为其初始值。

Date 对象属性

属性 描述

constructor 返回对创建此对象的 Date 函数的引用。

prototype 使您有能力向对象添加属性和方法。

Date 对象方法

方法 描述

Date() 返回当日的日期和时间。

getDate() 从 Date 对象返回一个月中的某一天 (1 ~ 31)。

getDay() 从 Date 对象返回一周中的某一天 (0 ~ 6)。

getMonth() 从 Date 对象返回月份 (0 ~ 11)。

getFullYear() 从 Date 对象以四位数字返回年份。

getYear() 请使用 getFullYear() 方法代替。

getHours() 返回 Date 对象的小时 (0 ~ 23)。

getMinutes() 返回 Date 对象的分钟 (0 ~ 59)。

getSeconds() 返回 Date 对象的秒数 (0 ~ 59)。

getMilliseconds() 返回 Date 对象的毫秒(0 ~ 999)。

getTime() 返回 1970 年 1 月 1 日至今的毫秒数。

getTimezoneOffset() 返回本地时间与格林威治标准时间 (GMT) 的分钟差。

getUTCDate() 根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。

getUTCDay() 根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。

getUTCMonth() 根据世界时从 Date 对象返回月份 (0 ~ 11)。

getUTCFullYear() 根据世界时从 Date 对象返回四位数的年份。

getUTCHours() 根据世界时返回 Date 对象的小时 (0 ~ 23)。

getUTCMinutes() 根据世界时返回 Date 对象的分钟 (0 ~ 59)。

getUTCSeconds() 根据世界时返回 Date 对象的秒钟 (0 ~ 59)。

getUTCMilliseconds() 根据世界时返回 Date 对象的毫秒(0 ~ 999)。

parse() 返回1970年1月1日午夜到指定日期（字符串）的毫秒数。

setDate() 设置 Date 对象中月的某一天 (1 ~ 31)。

setMonth() 设置 Date 对象中月份 (0 ~ 11)。

setFullYear() 设置 Date 对象中的年份（四位数字）。

setYear() 请使用 setFullYear() 方法代替。

setHours() 设置 Date 对象中的小时 (0 ~ 23)。

setMinutes() 设置 Date 对象中的分钟 (0 ~ 59)。

setSeconds() 设置 Date 对象中的秒钟 (0 ~ 59)。

setMilliseconds() 设置 Date 对象中的毫秒 (0 ~ 999)。

setTime() 以毫秒设置 Date 对象。

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

setUTCMonth() 根据世界时设置 Date 对象中的月份 (0 ~ 11)。

setUTCFullYear() 根据世界时设置 Date 对象中的年份（四位数字）。

setUTCHours() 根据世界时设置 Date 对象中的小时 (0 ~ 23)。

setUTCMinutes() 根据世界时设置 Date 对象中的分钟 (0 ~ 59)。

setUTCSeconds() 根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。

setUTCMilliseconds() 根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。

toSource() 返回该对象的源代码。

toString() 把 Date 对象转换为字符串。

toTimeString() 把 Date 对象的时间部分转换为字符串。

toDateString() 把 Date 对象的日期部分转换为字符串。

toGMTString() 请使用 toUTCString() 方法代替。

toUTCString() 根据世界时，把 Date 对象转换为字符串。

toLocaleString() 根据本地时间格式，把 Date 对象转换为字符串。

toLocaleTimeString() 根据本地时间格式，把 Date 对象的时间部分转换为字符串。

toLocaleDateString() 根据本地时间格式，把 Date 对象的日期部分转换为字符串。

UTC() 根据世界时返回 1970 年 1 月 1 日 到指定日期的毫秒数。

valueOf() 返回 Date 对象的原始值。

顶层函数（全局函数）

函数 描述

decodeURI() 解码某个编码的 URI。

decodeURIComponent() 解码一个编码的 URI 组件。

encodeURI() 把字符串编码为 URI。

encodeURIComponent() 把字符串编码为 URI 组件。

escape() 对字符串进行编码。

eval() 计算 JavaScript 字符串，并把它作为脚本代码来执行。

getClass() 返回一个 JavaObject 的 JavaClass。

isFinite() 检查某个值是否为有穷大的数。

isNaN() 检查某个值是否是数字。

Number() 把对象的值转换为数字。

parseFloat() 解析一个字符串并返回一个浮点数。

parseInt() 解析一个字符串并返回一个整数。

String() 把对象的值转换为字符串。

unescape() 对由 escape() 编码的字符串进行解码。

顶层属性（全局属性）

方法 描述

Infinity 代表正的无穷大的数值。

java 代表 java.\* 包层级的一个 JavaPackage。

NaN 指示某个值是不是数字值。

Packages 根 JavaPackage 对象。

undefined 指示未定义的值。

### DOM

#### 4.1 JavaScript HTML DOM

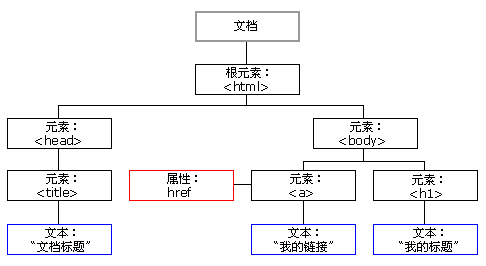
通过 HTML DOM，可访问 JavaScript HTML 文档的所有元素。

HTML DOM （文档对象模型）

当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。

HTML DOM 模型被构造为对象的树。

**HTML DOM 树**



通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。

* JavaScript 能够改变页面中的所有 HTML 元素
* JavaScript 能够改变页面中的所有 HTML 属性
* JavaScript 能够改变页面中的所有 CSS 样式
* JavaScript 能够对页面中的所有事件做出反应

查找 HTML 元素

通常，通过 JavaScript，您需要操作 HTML 元素。

为了做到这件事情，您必须首先找到该元素。有三种方法来做这件事：

通过 id 找到 HTML 元素

通过标签名找到 HTML 元素

通过类名找到 HTML 元素

通过 id 查找 HTML 元素

在 DOM 中查找 HTML 元素的最简单的方法，是通过使用元素的 id。

实例

本例查找 id="intro" 元素：

var x=document.getElementById("intro");

如果找到该元素，则该方法将以对象（在 x 中）的形式返回该元素。

如果未找到该元素，则 x 将包含 null。

通过标签名查找 HTML 元素

实例

本例查找 id="main" 的元素，然后查找 "main" 中的所有 <p> 元素：

var x=document.getElementById("main");

var y=x.getElementsByTagName("p");

#### 4.2 改变 HTML 元素的内容

HTML DOM 允许 JavaScript 改变 HTML 元素的内容。

改变 HTML 输出流

JavaScript 能够创建动态的 HTML 内容：

今天的日期是： Mon Sep 04 2017 12:52:36 GMT+0800 (中国标准时间)

在 JavaScript 中，document.write() 可用于直接向 HTML 输出流写内容。

实例

<!DOCTYPE html>

<html>

<body>

<script>

document.write(Date());

</script>

</body>

</html>

改变 HTML 内容

修改 HTML 内容的最简单的方法时使用 innerHTML 属性。

如需改变 HTML 元素的内容，请使用这个语法：

document.getElementById(id).innerHTML=new HTML

实例

本例改变了 <p> 元素的内容：

<html>

<body>

<p id="p1">Hello World!</p>

<script>

document.getElementById("p1").innerHTML="New text!";

</script>

</body>

</html>

改变 HTML 属性

如需改变 HTML 元素的属性，请使用这个语法：

document.getElementById(id).attribute=new value

实例

本例改变了 <img> 元素的 src 属性：

<!DOCTYPE html>

<html>

<body>

<img id="image" src="smiley.gif">

<script>

document.getElementById("image").src="landscape.jpg";

</script>

</body>

</html>

#### 4.3 改变 CSS

HTML DOM 允许 JavaScript 改变 HTML 元素的样式。

改变 HTML 样式

如需改变 HTML 元素的样式，请使用这个语法：

document.getElementById(id).style.property=new style

例子 1

下面的例子会改变 <p> 元素的样式：

<p id="p2">Hello World!</p>

<script>

document.getElementById("p2").style.color="blue";

</script>

#### 4.4 DOM 事件

对事件做出反应

我们可以在事件发生时执行 JavaScript，比如当用户在 HTML 元素上点击时。

如需在用户点击某个元素时执行代码，请向一个 HTML 事件属性添加 JavaScript 代码：

onclick=JavaScript

HTML 事件的例子：

当用户点击鼠标时

当网页已加载时

当图像已加载时

当鼠标移动到元素上时

当输入字段被改变时

当提交 HTML 表单时

当用户触发按键时

例子 1

在本例中，当用户在 <h1> 元素上点击时，会改变其内容：

<h1 onclick="this.innerHTML='谢谢!'">请点击该文本</h1>

HTML 事件属性

如需向 HTML 元素分配 事件，您可以使用事件属性。

实例

向 button 元素分配 onclick 事件：

<button onclick="displayDate()">点击这里</button>

使用 HTML DOM 来分配事件

HTML DOM 允许您通过使用 JavaScript 来向 HTML 元素分配事件：

实例

向 button 元素分配 onclick 事件：

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

</script>

onmousedown 和 onmouseup

当用户按下鼠标按钮时，更换一幅图像。

onload

当页面完成加载时，显示一个提示框。

onfocus

当输入字段获得焦点时，改变其背景色。

鼠标事件

当指针移动到元素上方时，改变其颜色；当指针移出文本后，会再次改变其颜色。

## 作业

1. 编写一个完整的HTML5的网页，内包含h1~h6, p, span, b, i, a, table, img 等元素，并通过设置CSS来为不同的元素设置不同的颜色；
2. 编写一个HTML5网页，包含一个完整的form（表单），里面有一个文本输入框（input type=text），一个选择框（select，可选项：男，女），和一个添加按钮，用户在文本框中输入名字，选择男女，按按钮即可将该用户添加到数组中，并动态加入html中的表格中显示出来。编写相对应的HTML文件和JavaScript程序。

## 教学后记