[一、课程简介 4](#_Toc456636671)

[1.软件的架构： 4](#_Toc456636672)

[2.网页的组成：结构、表现、行为 4](#_Toc456636673)

[二、HTML & CSS 5](#_Toc456636674)

[1.HTML和CSS简介 5](#_Toc456636675)

[2、进制 5](#_Toc456636676)

[3、编码和解码 5](#_Toc456636677)

[4、常用标签 5](#_Toc456636678)

[5、HTML实体 6](#_Toc456636679)

[6、图片标签 6](#_Toc456636680)

[7、meta标签 6](#_Toc456636681)

[8、HTML基本语法（严格版规范） 7](#_Toc456636682)

[9、内联框架 7](#_Toc456636683)

[10、<a>标签 7](#_Toc456636684)

[11、CSS 层叠样式表 Cascading Style Sheets 7](#_Toc456636685)

[11.1 使用CSS设置元素样式： 7](#_Toc456636686)

[11.2 css的基本语法： 8](#_Toc456636687)

[12、块元素（block）和内联元素（行内元素）（inline） 8](#_Toc456636688)

[13、id和class 8](#_Toc456636689)

[14、元素之间的关系 8](#_Toc456636690)

[15、CSS选择器selector 9](#_Toc456636691)

[16、css选择器2 9](#_Toc456636692)

[17、a的伪类（补充） 11](#_Toc456636693)

[18、文本标签及样式 11](#_Toc456636694)

[19、列表： 11](#_Toc456636695)

[20、文本格式 12](#_Toc456636696)

[20.1单位 12](#_Toc456636697)

[20.2颜色 12](#_Toc456636698)

[20.3文字样式 12](#_Toc456636699)

[20.4 字体的简写属性font 13](#_Toc456636700)

[20.5英文字母大写化：text-transform 13](#_Toc456636701)

[20.6 设置文本修饰：text-decoration 13](#_Toc456636702)

[20.7 字符间距和单词间距 13](#_Toc456636703)

[20.8 对齐文本 14](#_Toc456636704)

[20.9 首行缩进 14](#_Toc456636705)

[20.10 块和行内元素的转换 display 14](#_Toc456636706)

[21、盒子模型 boxmodel 15](#_Toc456636707)

[21.1 边框样式 15](#_Toc456636708)

[21.2 padding 内边距 15](#_Toc456636709)

[21.3 margin外边距 15](#_Toc456636710)

[21.4 外边距重叠 16](#_Toc456636711)

[21.5 内联元素的盒子模型 16](#_Toc456636712)

[21.6 内联元素和块元素的相互转换 16](#_Toc456636713)

[21.7 设置元素的可见性 16](#_Toc456636714)

[21.8 元素的溢出处理overflow 16](#_Toc456636715)

[21.9文本的溢出处理：text-overflow：clip|ellipsis|string; 17](#_Toc456636716)

[21.10处理元素内的空白:white-space 17](#_Toc456636717)

[21.11例子：实现文字溢出隐藏，鼠标滑过显示效果： 17](#_Toc456636718)

[21.12 文档流 17](#_Toc456636719)

[21.13浏览器默认样式的处理 18](#_Toc456636720)

[22、错误检查 18](#_Toc456636721)

[23、浮动： 18](#_Toc456636722)

[24、定位 20](#_Toc456636723)

[24.1相对定位 20](#_Toc456636724)

[24.2绝对定位： 21](#_Toc456636725)

[24.3固定定位： 21](#_Toc456636726)

[25、元素透明度 21](#_Toc456636727)

[26、背景 21](#_Toc456636728)

[26.2图片整合技术 （CSS-Sprite） 22](#_Toc456636729)

[27、表格 table 22](#_Toc456636730)

[27.2 长表格： 23](#_Toc456636731)

[27.3表格嵌套： 23](#_Toc456636732)

[28、表单： 23](#_Toc456636733)

[29、框架集 <frameset> 24](#_Toc456636734)

[30、CSS-Hack技术 25](#_Toc456636735)

[31、对于一个新项目的准备工作： 25](#_Toc456636736)

[32、浏览器兼容中的问题 25](#_Toc456636737)

[三、JavaScript 26](#_Toc456636738)

[33、JavaScript 简介 26](#_Toc456636739)

[34、JavaScript基本语法： 27](#_Toc456636740)

[34.1 JavaScript基本语法 27](#_Toc456636741)

[34.2 JavaScript编写位置 27](#_Toc456636742)

[34.3 常量和变量 28](#_Toc456636743)

[34.4 标识符 28](#_Toc456636744)

[34.5 数据类型： 28](#_Toc456636745)

[34.6 类型转换 29](#_Toc456636746)

[34.7隐式的类型转换： 30](#_Toc456636747)

[35、 运算符： 31](#_Toc456636748)

[35.1 一元运算符： 31](#_Toc456636749)

[35.2 自增和自减 32](#_Toc456636750)

[35.3 逻辑运算符 32](#_Toc456636751)

[35.4 赋值运算符： 32](#_Toc456636752)

[35.5 关系运算符： 33](#_Toc456636753)

[35.6 条件运算符： 33](#_Toc456636754)

[35.7 运算优先级： 34](#_Toc456636755)

[36语句 34](#_Toc456636756)

[36.1条件判断语句： 34](#_Toc456636757)

[36.2 输入弹窗prompt(); 35](#_Toc456636758)

[36.3 switch...case 35](#_Toc456636759)

[36.4 while循环 35](#_Toc456636760)

[36.5 do ... while 循环 36](#_Toc456636761)

[36.6 循环的四个要素： 36](#_Toc456636762)

[36.7 for循环 36](#_Toc456636763)

[36.8 break和continue 36](#_Toc456636764)

[37、对象： 37](#_Toc456636765)

[37.1 对象的分类： 37](#_Toc456636766)

[37.2 创建一个对象： 37](#_Toc456636767)

[37.3对象的字面量 38](#_Toc456636768)

[37.4枚举对象中的属性： for..in 39](#_Toc456636769)

[37.5 创建对象的方法 39](#_Toc456636770)

[37.6 创建函数的方法 39](#_Toc456636771)

[38、函数： 40](#_Toc456636772)

[38.1方法： 41](#_Toc456636773)

[38.2 执行环境 和 作用域 41](#_Toc456636774)

[38.3函数总结： 42](#_Toc456636775)

[38.4执行环境总结： 43](#_Toc456636776)

[38.5、作用域、作用域链 43](#_Toc456636777)

[38.6、this 函数执行的上下文对象 44](#_Toc456636778)

[38.7、对象的创建 44](#_Toc456636779)

[38.8、类 45](#_Toc456636780)

[38.9、原型对象：prototype 46](#_Toc456636781)

[39、数组 Array 46](#_Toc456636782)

[39.1、数组 的 常用 方法 47](#_Toc456636783)

[40、Math 对象 48](#_Toc456636784)

[41、Date对象 49](#_Toc456636785)

# 一、课程简介

## 1.软件的架构：

- 我们的软件一般分成两种架构：C/S 和 B/S

- C/S架构

- C/S指的就是客户端/服务器的架构

- 我们在计算机中使用的大部分软件都是C/S架构的软件

- 我们用户通过客户端来使用软件

- 服务器用来处理软件的业务逻辑

- 一般我们说的C/S架构的软件，指的都是图形化界面的软件

- 特点：

1、C/S架构软件都需要安装客户端。

2、C/S架构软件更新时服务器和客户端都要更新。

3、C/S架构软件不能跨平台使用。

4、C/S架构软件客户端和服务器端之间的通信采用“自有协议”，比较安全。

- B/S架构

- B/S指的是浏览器/服务器的架构

- B/S本质上也是C/S，只不过B/S架构的软件使用的是浏览器作为客户端

- B/S架构的软件是通过访问一个网页的形式来使用一个软件，比如：京东、淘宝、12306

- 特点：

1、B/S架构软件不需要安装，可以直接使用。（需要浏览器）

2、B/S架构软件更新时客户端不需要更新。

3、B/S架构软件可以跨平台使用。

4、B/S架构软件客户端和服务器之间通信采用的是HTTP协议，而HTTP协议是公开的协议，安全性较差。

## 2.网页的组成：结构、表现、行为

根据W3C标准：一个网页主要由三部分组成： 结构、表现、行为。

1、结构： HTML 用于描述页面的结构。

2、表现： CSS 用于控制页面中元素的样式。

3、行为： JavaScript 用于响应用户操作。

结构HTML、表现CSS、行为JavaScript 应该分离，便于后期维护。

一个设计优良的网页，要求结构、表现、行为三者分离！

第一阶段学习：两周：

HTML、CSS、网页布局

# 二、HTML & CSS

## 1.HTML和CSS简介

HTML(hypertext markup language) 超文本标记语言

在纯文本编辑器（如记事本）中写的内容叫做纯文本。纯文本不能保存格式（大小、颜色、字体等）。

<html><head><title></title></head><body></body></html> 根标签,子标签。祖先、后代。

<head>标签中的内容不会直接在浏览器中显示。<body>中包含所有可见内容。

<font color="red" size="7"></font>定义文本 字体、大小、颜色 size=1~7

纯样式标签（如font）不赞成使用。

## 2、进制

常用的进制有：十进制、二进制、八进制、十六进制

十进制：满十进一，共有10个数字。

二进制：满二进一，共有2个数字。

十六进制：满16进1，共有16个数字。

## 3、编码和解码

计算机实际上是一个特别特别笨的机器，它只认识0和1,。

计算机底层在保存一个文件时都会将这个文件保存成2进制。

所有文字在计算机底层都会转换成二进制编码。

计算机在保存字符时需要将字符全都转换成二进制编码，这个过程我们称为 编码。

计算机在读取文件时需要将二进制编码转换成文字，这个过程我们称为 解码。

编码和解码时所采用的规则我们称为 字符集。

常见的字符集： ASCII （7位）、ISO-8859-1 （8位）、GBK、GB2312、UTF-8（万国码）

乱码产生的根本原因：编码和解码所采用的字符集不同。

解决乱码：统一编码和解码字符集。<meta charset="utf-8" />自结束标签

使用 “UTF-8 无BOM格式”编码

## 4、常用标签

搜索引擎在检索一个网页时，将会最先检索title中的内容，但内容不要太长，尽量在25个文字左右。

即使网页编写的不符合规范，浏览器也会尽最大的努力正确的解析页面。

h1`h6是语义化标签，h1表示一级标题，它的重要性仅次于title，对于搜索引擎的检索是很重要的。一般情况下，一个页面中只能有一个h1。

h1以后的标题标签重要性越来越低，一般只用到h3，h3以后很少用到。

<hr />水平线标签，可以再指定位置输出一条横线。

## 5、HTML实体

在HTML中一些特殊字符不能直接编写。如：大于号、小于号不能和字母一起写。我们需要用转义字符（实体）来表示这些特殊符号。

实体的语法： &实体的名字;

如：小于号 &lt; 大于号 &gt; 空格(无换行空格) &nbsp; 版权声明 &copy;

## 6、图片标签

图片标签：<img src="" alt=""/>

src属性指向的是一个外部图片的地址；

alt属性 在图片不能显示的时候对图片的描述，主要作用是被搜索引擎检索的。

搜索引擎会根据alt属性为图片进行分类。如果希望搜索引擎检索到图片，就一定要写alt属性。

src属性使用相对路径时，相对路径是相对于当前文件所在的目录的路径。使用 ../ 来返回上一层。

一般不推荐使用图片的width和height改变图片宽高，最好使用PS等剪切成合适大小。

图片格式的选择：

web中常见的图片格式有：JPEG、GIF、PNG

JPEG：支持的颜色比较多，可以显示一些鲜艳的图片，一般情况下照片、颜色复杂的图片都使用jpeg。

jpeg 不支持透明。

GIF： 支持的颜色比较少，可以显示一些颜色单一的图片，比如网站单色的logo。

只支持简单的透明，对复杂的图片透明支持效果不好。（支持全透明，不支持半透明）

png： 支持的颜色比较多，功能比较强大。

完全支持透明。

图片的选用原则： 显示效果一致用最小的图片。显示效果不一致，使用效果合适的。

## 7、meta标签

<!-- meta用来设置网页的元数据，比如可以用来设置网页的编码 -->

<meta charset="utf-8" />

<!-- 使用meta来设置网页的关键字 -->

<!-- name属性用来设置属性的名字。 content用来设置属性的值 -->

<!-- 我们可以同时为一个网站设置多个关键字，多个关键字之间使用,隔开 -->

<meta name="keywords" content="美女,帅哥,HTML5,Java" />

<!-- meta还可以用来设置网页的描述（简介） -->

<!-- 搜索引擎在检索页面时，会检索keywords和description中的内容

但是这些内容不会影响网站在搜索引擎中的排名

-->

<meta name="description" content="专注于搜集各种美女帅哥的图片" />

<!-- meta还可以用来做请求的重定向 -->

<!-- http-equiv="refresh" 固定值

content="秒数;url=目标地址"

-->

<meta http-equiv="refresh" content="5;url=demo8.html" />

<meta http-equiv="refresh" content="5;url=http://www.baidu.com" />

## 8、HTML基本语法（严格版规范）

1.注释：注释不能嵌套注释。

2.标签：标签必须正确结束。一般标签成对出现，自结束标签后面加" /".

3.属性：属性必须有值，且值必须加引号，单引号或双引号。

4.HTML不区分大小写，规范推荐使用小写。

5.标签可以嵌套，但是不能交叉嵌套。

！！！！如果出现错误，检查是否有 **单词错误** 和 **中英标点符号错误** 和 **误删除某些文字符号** 和 **标签是否成对** 和 **使用了错误的属性**和 **数字单位不对**和 **注释文字没有正确注释**！！！！

## 9、内联框架

内联框架：<iframe></iframe> : 用于向一个网页中引用另一个页面。

实际开发中不建议使用内联框架，因为内联框架中的内容不会被搜索引擎检索。

<iframe src="demo.html" width="500px" height="500px" name="ifr01"></iframe>

属性：

src：用来指定一个外部网页的地址。

width、height 定义宽高。

neme: 为内联框架设置一个名字。

## 10、<a>标签

超链接：<a></a> 可以使浏览器跳转到其他页面。

<a href="http://www.qq.com">跳转到QQ</a>

<a>标签属性：

href：定义连接地址。可以是相对路径、完整网址、

target：设置打开目标网页方式。\_blank在新标签页打开、\_self在当前页打开、framename在指定内联框架打开、 \_parent \_top

href="#" 回到当前页顶部；href="#id\_name" 跳转到指定id名称的位置。

href="mailto:123@qq.com" 向该邮箱发送邮件。

## 11、CSS 层叠样式表 Cascading Style Sheets

### 11.1 使用CSS设置元素样式：

1.在标签内部使用style属性设置： <p style="color:red; "><p/>;

缺点：样式不能复用，维护性差，一般开发中不会使用内联样式（行内样式）。

2.在head中的style标签中使用CSS： <style type="text/css"> p{ color:red; }</style>

优点：使CSS和HTML代码进一步分离，使CSS可以在当前页面中复用。缺点：不可以在其它页面中复用。

1. 将CSS样式表编写到一个外部的CSS文件中，通过link标签将css文件引入到当前文件。

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

rel="stylesheet" type="text/css"是固定值。 href定义地址。

优点：可以使CSS与HTML 完全分离。CSS文件可以在不同页面中引用。将样式写到一个外部的css文件中，可以使用浏览器的缓存，提高用户的访问速度。

### 11.2 css的基本语法：

/\* CSS中的注释方法 \*/

css样式由两部分组成：

第一部分：选择器： 选择器的作用是选中页面中需要设置样式的元素

第二部分：声明块： 声明块里面是由一个一个的样式的键值对结构组成的，

每一个样式名和样式值之间使用：连接，多个样式之间使用;隔开，每个样式我们又称之为一个声明。

选择器{ 样式名:样式值; }

## 12、块元素（block）和内联元素（行内元素）（inline）

块元素独占一行，即使长度很短。如：div。

内联元素：只会占用自身大小。如：a标签。span。

div不会对内容产生任何效果，它也没有任何语义。div主要用来对页面进行布局。

span不会对内容产生影响，没有任何语义。span主要用来选中页面中的文字，为其设置样式。

块元素主要作用是用来在页面中布局。内联元素主要是在页面中引入内容。

一般情况下都是块元素包含 块元素或内联元素，但是内联元素不能包含块元素。

但是a元素可以包含任何元素除了它本身。

p元素不能包含任何块元素。

## 13、id和class

div.classname + Tab 创建一个带有class的div

div#idname + Tab 创建一个带有id的div

id是一个元素的唯一标识，同一个页面中不能出现重复的id属性值。

class属性用来为一个元素分组，页面中相同class属性值可以出现多个，并且一个元素可以拥有多个class。

id和class属性值必须以字母开头。

在前端开发中，编写css时，一般我们选择使用class选择器，尽量避免使用id选择器。

标签属性 title="描述文字" ，当鼠标指向该标签时，显示title属性值。

## 14、元素之间的关系

祖先元素、后代元素：直接或间接包含关系。

直接或间接 包含 后代元素 的元素称为该 后代元素 的 祖先元素 ；

直接或间接被 祖先元素 包含的元素称为该元素的 后代元素。

父元素、子元素：直接包含关系。

直接包含子元素的元素称为该子元素的父元素；

直接被父元素包含的元素称为该元素的子元素。

父元素也是祖先元素，子元素也是后代元素。

兄弟元素：拥有相同父元素的元素互为兄弟元素。

## 15、CSS选择器selector

1. 元素选择器：语法：标签名。 可以选中页面中所有指定的元素。

2. id选择器：语法：#id属性的值{ }。可以根据元素的id属性值选中元素。

3. 类选择器：语法：.class属性的值{ }。 可以根据元素的class属性选中一组拥有相同class属性值的元素。

可以为一个元素指定多个class值，多个class值之间使用空格分开。

4. 选择器分组：语法：选择器1，选择器2{ }。 我们可以同时选中符合多个选择器的元素，多个选择器之间使用逗号隔开。如： p,span,a{}

5. 通配符选择器：语法：\*{ } 。 选中所有元素。\*性能较差，不推荐使用。

6. 复合选择器：语法：选择器1选择器2{ } 。 可以为一个选择器设置多个条件，只会选中“同时满足多个条件” 的元素

6.27讲课

## 16、css选择器2

1.后代元素选择器：选中指定元素的后代元素。 语法： 祖先 后代{ } 。

2.子元素选择器： 选中指定元素的指定子元素。语法： 父元素>子元素{ } 。

注意：子元素选择器在IE6中不支持。如果页面需要兼容IE6就不要使用这种选择器。 关系比较严格，少用。

注意：开发项目时 确认 项目需要支持哪些浏览器。chrome、IE8/IE6、Firefox

3.伪类选择器：

对于超链接a元素：

用户有没有访问过指定连接，是通过用户的历史记录识别的。

【顺序必须为:link :visited :hover :active】

正常链接：a:link{ } 。

访问过的链接： a:visited{ } 。【只能定义字体样式和颜色。为了保护用户隐私】

鼠标滑过的链接：a:hover{ } 。

正在点击的链接：a:active{ } 。

4.获取焦点

获取焦点 :focus{ } 。

指定元素前 :before{ } 。 可以在指定元素内部前面添加内容。如：p:before{content:"添加的文字无法被选中"；background-color:red;}

指定元素后 :after{ } 。 可以在指定元素内部后面添加内容。如：p:after {content:url(图片.jpg);}

元素选中效果 ::selection 可以设置文字的选中的效果p::selection{background-color: orange;}

火狐中不支持::selection支持::-moz-selection。p::-moz-selection{background-color: orange;}

5.给段落定义样式

首字母 :first-letter{ };

首行 :first-line{ };

6.属性选择器：

[属性名]{ }; 选中含有指定属性的元素。

[属性名=属性值]{ ｝; 选中指定属性名等于指定值的元素。

[属性名^属性值开始字符]{ }; 选属性值中指定属性的以指定字符开始的元素。

[属性名$属性值结尾字符]{ }; 选中指定属性的属性值以指定字符结尾的元素。

[属性名\*属性值包含字符]{ }; 选中指定属性的属性值含有指定字符的元素。

7.其他子元素选择器

第一个子元素 :first-child{ };

p:first-child{ }; 选中是 第一个子元素的p元素。 是在所有的子标签中排序。包括<br />。

最后一个子元素 :last-child{ };

p:last-child{ }; 选中是最后一个子元素的p元素。 是在所有的子标签中排序。

指定位置的子元素 :nth-child(n);

p:nth-child(3){ }; 选中是第三个子元素的p元素。 是在所有的子标签中排序。

选中奇数子元素 :nth-child(odd);

选中偶数子元素 :nth-child(even);

选中指定类型中的第一个元素 :first-of-type{ };

p:first-of-type{ }; 选中p元素中第一个p元素。只在p元素中排序。

选中指定类型中的最后一个元素 :last-of-type{ };

p:last-of-type{ }; 选中p元素中最后一个p元素。只在p元素中排序。

选中指定类型中的指定位置元素 :nth-of-type(n){ };

8.兄弟选择器

前一个元素1+后一个元素2{ }; 选中元素的后一个兄弟元素

前一个元素1~后一个元素2{ }； 选中元素的后面【所有的】指定兄弟选择器

9.否定伪类

:not(选择器){ }; 除了not（选择器）中的元素外的其他元素。

如：p:not(.classname){ }; 选中除了包含.classname的p元素。

10.所有的选择器我们都可以组合使用。但是要注意：我们在使用选择器时越简单越好。

浏览器在解析CSS选择器时，它是按照从右向左的顺序去解析的。

尽量避免使用 \* 。

11.CSS的样式设置具有继承性。 祖先元素的样式会自动应用到他的后代元素。

但是，并不是所有的样式都有继承性。比如：背景颜色不能继承。

特殊情况：如果给body设置背景颜色，它会默认将背景颜色应用到其父元素<html>上。

12.CSS选择器优先级（权重）

当我们使用不同的选择器为同一个元素设置同一个样式时：由选择器的优先级（权重）来决定应用哪个样式。

当一个元素上有相同的样式产生冲突时，会根据选择器优先级来决定显示哪个样式。优先级高的优先显示。

优先级的计算： 需要将所有的选择器的优先级相加计算。

优先级的累加，不能超过选择器的最高数量级。阶级不可逾越。

我们可以在一个样式后面(分号之前)添加 !important 来将该样式设置为优先级最高的样式，这样他会优先于所有的样式显示。但是在开发中和内联样式一样，尽量避免使用 !important ..

当优先级相同时，使用后一个样式。

!important 样式是最最优先的样式。

内联样式的优先级最大。内联样式会优先于所有的选择器显示。优先级是 1000 。

id选择器的优先级是 100 。

类和伪类、属性选择器的优先级是 10 。

元素选择器的优先级是 1 。

通配符选择器的优先级是0 。

继承来的样式没有优先级 ，最低。

6.28讲课

## 17、a的伪类（补充）

:link :visited :hover :active 有顺序限制 lvha

当我们通过a的伪类来为超链接设置样式时，这些选择器的优先级都是一样的。优先显示靠下的样式。

:link 状态： "未访问" 的链接

:visited 状态： "已访问" 的链接

:hover 状态： "鼠标滑过的" "未访问/已访问" 的链接

:active 状态： "鼠标点击的" "鼠标正在滑过的" "未访问/已访问" 的链接

## 18、文本标签及样式

em标签 用于表示一段内容中的着重点。是语气上的强调。。

strong标签 用于表示一个内容的重要性。【更重要】

通常 em显示为 斜体， strong 显示为 粗体。

b标签 加粗，显示效果与strong一样。没有语义。

i标签 斜体，显示效果与em一样。没有语义。

根据Html5 的标准，如果需要单纯的为一段文字设置加粗或斜体，可以使用 b和i。

small标签 会使里面的文字略小于其父元素中的文字大小。small可以嵌套small。

html5中规定：small标签用来表示 细则（小字）一类的信息。

big标签 会使里面的文字略大于其父元素中的文字大小。但是在H5中没有任何语义。一般不用。

cite标签 对某内容的引用或参考（需要加《》的东西），显示为斜体。

blockquote标签和q标签 都表示引用的内容，

q表示一个短引用，（行内元素）浏览器会在q标签的前后各加一个引号。

blockquote表示长引用（块引用）（块元素）浏览器会给一个缩进。

sup 上标 如 平方米 m<sup>2</sup>

sub 下标 如 氧气分子 O<sub>2</sub>

ins 插入的内容， 加下划线。

del 删除的内容， 加删除线。

pre标签：格式标签，（无语义）可以将代码中的格式保留。 常用于在页面中添加程序代码。

code标签： 代码标签，（有语义）

一般将pre和code标签一起使用来表示一段程序。<pre><code>程序代码</code></pre>

注意：不能直接书写html代码，需要将<>转义成实体才能显示。

## 19、列表：

无序列表 ul ；列表项 li ；

ul标签 type属性：修改项目符号。 disc、square、circle

一般不使用默认项目符号，因为不同浏览器显示效果不同。

去掉无序列表的项目符号：css中： ul{list-style:none;}

有序列表 ol ；列表项 li ；

有序列表使用有序的序号作为项目符号。

ol标签 type属性修改项目符号。a、A、i、I、1

ol标签 start属性修改起始数字。

有序列表用的比较少。无序列表可用来做导航条等。

ul、ol、li 都是块元素。

列表之间可以互相嵌套。

定义列表 dl：

dl 定义列表，用于为一些定义做一些解释

dl 定义列表、 dt 定义的内容、 dd 定义的内容的描述（dd有缩进效果）

dl dt dd 都是块元素

列表元素ul、ol、li、dl、dd、dt都是块元素

列表之间可以互相嵌套。

## 20、文本格式

### 20.1单位

px像素：最常用单位，表示一个像素。屏幕是有一个一个像素点构成的，一个像素点我们称为一个像素。不同的屏幕一个像素的大小不同。

百分比：使用百分比做单位时是相对于它的父元素的属性来计算大小。优点：可以随父元素属性按比例改变。

使用百分比作单位的属性 对应 其父元素的属性 ，没有固定值。

em：em和百分比很像，1em = 父元素的字体的大小。em永远相对于父元素的字体的大小。em主要用来为文字设置效果

。

### 20.2颜色

在css中设置颜色有多种方式。

1.使用颜色对应颜色的英文单词来表示一个颜色。局限：需要记住大量颜色的单词，而且使用英文单词也不能描绘五彩缤纷的世界。

2.使用RGB值的形式来设置颜色。Red Green Blue。rgb(红色浓度，绿色浓度，蓝色浓度)浓度0~255.

除了使用数值，也可以用百分比来表示rgb值。rgb（R%,G%B%）。

3.使用16进制的RGB值，使用 三组 两位 的十六进制来表示一个颜色。 # 红色浓度 绿色浓度 蓝色浓度。00-ff

像#ffff00这样两两重复的颜色，可以简写成#ff0。

4.RGBA： rgb(255,255,255,0.5)

### 20.3文字样式

浏览器默认文字大小16px；可以使用css的font-size来修改文字大小。

一般页面中使用的文字大小是12px，所以需要将文字大小重置为12px。

应该修改body元素的文字大小。 body{font-size:12px;}

文字颜色：color：#f00;

文字斜体：font-style:italic;

文字加粗：font-weight:bold;

小型大写字母：font-variant:small-caps;

文字大小：font-size:12px;

文字字体：font-family:微软雅黑;

arial是英文字体，不支持中文。

当设置了一个字体时，如果浏览器能找到该字体，就会显示该字体，如果找不到该字体就会显示默认字体。

font-family可以指定多个字体。多个字体以逗号分开。 优先使用第一个字体，如果不支持则使用下一个。

字体大分类：serif(衬线字体)、sans-serif(非衬线字体)、monospace（等宽字体）、 cursive（草书字体）、fantasy（虚幻字体）

宋体属于衬线字体，微软雅黑属于非衬线字体。

这些大的分类一般指定在所有的字体的最后，入过所有的字体都不支持，则从大的分类中选择一种。

斜体：font-style:italic; italic斜体；oblique倾斜；对大部分浏览器这两种显示效果一样。所以斜体使用italic。

加粗：font-weight:bold; 一般只使用bold来设置加粗。

也可以使用100-900九个数值由细到粗，但是由于系统里的字体没有那么多粗细级别，所以往往200和100的粗细可能是一样的，所以如果级别大于前一个级别，有可能一样粗，也有可能更粗，但是一般不用。

小型大写字母：font-variant:small-caps; 当设置了小型大写字母时，小写字母会显示为大写，并且比大写字母要小一点。

### 20.4 字体的简写属性font

通过font属性可以同时设置所有的字体相关样式。

font：bold italic small-caps 20px/40px "微软雅黑";

顺序：“斜体、加粗、小型大写字母”没有顺序要求，不写则应用默认值。

文字大小/行高、文字字体 有顺序且必须写，而且必须是后两个。

行高：line-height。页面在显示文字时，都有一个默认行高。文字默认在行高中垂直居中。

行间距 = 行高 - 字体大小

单行文本的情况下，我们可以设置行高和元素的高度一致，可以使文本在元素中垂直居中。

### 20.5英文字母大写化：text-transform

可以修改英文字母大小写。

值：uppercase将所有字母转换为大写。 lowercase将所有字母转换为小写。capitalize将所有的单词的首字母大写。通过空格识别单词。

### 20.6 设置文本修饰：text-decoration

值：underline添加下划线，overline在文本上方添加一条线，line-through添加删除线，blink闪烁文字（所有浏览器都不支持） ，none去除下划线。

6.29讲课：：：：

### 20.7 字符间距和单词间距

letter-spacing: 字符间距，支持中文，值可以使用10px、1em 等

word-spacing: 单词间距，浏览器通过空格来判断是不是一个单词的，实际上是设置【空格】的大小。

### 20.8 对齐文本

text-align:

left：左对齐（默认）

right：右对齐

center：居中

justify：两端对齐

### 20.9 首行缩进

text-indent: 【用于定义“块级元素”中“第一个内容行”的缩进】

缩进时一般使用em，这样可以根据文字大小来自动适应。

如果设置一个[正值] ，则向右移动相应长度，

如果设置一个[负值] ，则向左移动相应长度。

当页面中有一些内容不希望被用户看见时可以使用负的缩进值。如网站的logo相关设置。

### 20.10 块和行内元素的转换 display

元素的分类

\* 标签，标记，元素，对象----都是指的一个对象！

块元素 div,p,h1-h6,body

特点：支持宽高；

独占一行；

不受空格影响。

display:block;

用途：搭架子

行内元素 a,em,i,b,strong,span

特点：不支持宽高；

共处一行；

受空格和回车的影响。

display:inline;

用途：修饰细节

行内块 img，input

特点： 支持宽高；

共处一行；

受空格和回车的影响。

display:inline-block;

元素的转换：

转成块： display:block;

转成行内元素： display:inline;

转成行内块： display：inline-block;

元素的嵌套:--谁包谁的问题

1）块元素可以嵌套块和行内元素，但是p例外，p不能包含块元素！

2）行内元素只可以嵌套行内元素，但是a例外，a既可以包块，也可以包行内！

一般在实际项目中，我们经常把a转成块再去包块！

## 21、盒子模型 boxmodel

内容区 content

内边距 padding

边框 border

外边距 margin

内容区：宽、高 只是内容区的宽高。

width:200px;

height:200px;

background-color:red;

### 21.1 边框样式

border不可继承。

宽度 border-width:10px; 一个值时是四个边框的宽度；2个值：上下 右左 ；3个值：上 右左 下 ；4个值：上右下左 【奇数位是上下，偶数位是右左】

样式 border-style:solid; solid实线，dotted点状，dashed虚线，double双线，none无边框。 4个值：上右下左

颜色 border-color:red;

边框简写： border：宽度 样式 颜色；无顺序。 同时设置4个边样式。

border-top|right|bottom|left : 10px solid red ; 设置一个边。

边框style默认是none，其他属性都有默认值。

### 21.2 padding 内边距

可见框：元素的盒子的可见部分，边框是可见框最外层。

可见框大小由内容区、内边距、边框共同决定。

内边距：内容区和边框的距离我们称为盒子的内边距。 有上、右、下、左 四个方向的内边距。

内边距会影响可见框大小。

在内容区设置的背景会延伸到内边距。

子元素会在父元素的内容区显示。

要使内容区颜色和内边距区分开，可以添加一个div，设置样式

.div{width:100%;height:100%;background-color:green;}

简写： padding:10px 20px 30px 40px; 上右下左。

### 21.3 margin外边距

外边距：盒子的边框和其他元素的距离

每个盒子都有4个方向的外边距

margin-top|right|bottom|left：100px；

修改元素的外边距不会影响元素的可见框大小，但是会影响元素的位置。

若外边距设置成 负值，元素会向反方向移动。

简写： margin：上 右 下 左 ;

若margin的值有矛盾（不能同时满足）时，优先设置左外边距和上外边距。

margin值可以设置成自动。margin:auto；这样浏览器会自动调整外边距。

当左右外边距设置成自动时，浏览器会将左右外边距设置为相等的值（水平居中）。

上下外边距设置成自动时，不会设置上下外边距。基于该特点，使用如下代码来设置子元素在父元素中水平居中： margin:0 auto;

### 21.4 外边距重叠

垂直方向的相邻的外边距会发生 外边距重叠现象。外边距重叠以后，会取两外边距中较大值作为两个元素的外边距。

水平方向的外边距不会重叠，距离是两个外边距之和。

### 21.5 内联元素的盒子模型

1.内联元素可以设置边框，

2.内联元素不能设置 宽、高。

3.内联元素支持水平方向外边距，不支持垂直方向外边距。

4.内联元素支持水平方向内边距，也可以设置垂直方向内边距，垂直内边距会影响到内联元素的显示大小，但是不会影响到布局。

### 21.6 内联元素和块元素的相互转换

display:; 可以设置元素的显示类型。 block块元素。inline行内元素。inline-block行内块元素（既可以设置宽高，又不独占一行）。 none不显示该元素，同时该元素不在页面中占用位置。

### 21.7 设置元素的可见性

visibility:visible;可以显示（默认值）

visibility:hidden; 元素隐藏显示，但是元素的位置依然保留。区别于display:none;

### 21.8 元素的溢出处理overflow

当子元素的高度和宽度超出父元素时，不会将父元素撑开，而是使子元素溢出父元素。

但是在IE6中子元素会将父元素撑开，因为在IE6中width和height相当于min-width和min-height。

overflow:;用来设置当子元素大小超出父元素时如何处理。

visible（默认值）内容不会被修剪，会呈现在元素框之外；hidden 溢出内容被隐藏。scroll在元素中添加水平和垂直滚动条，（不论是否溢出）可以通过滚动条查看完整内容。

auto 当元素溢出时根据需要添加滚动条。

### 21.9文本的溢出处理：text-overflow：clip|ellipsis|string;

clip 修剪文本；

ellipsis 显示省略号来代表被修剪的文本;

string 使用给定的字符串来代表被修剪的文本;

### 21.10处理元素内的空白:white-space

normal 默认。空白会被浏览器忽略。

pre 空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。

nowrap 文本不会换行，文本会在在同一行上继续，直到遇到 <br> 标签为止。

pre-wrap 保留空白符序列，但是正常地进行换行。

pre-line 合并空白符序列，但是保留换行符。

### 21.11例子：实现文字溢出隐藏，鼠标滑过显示效果：

<style>

.over{ 【.over是这个元素的类名】

white-space:nowrap; 【设置文字不换行】

width:8em;

overflow:hidden; 【设置元素溢出隐藏】【如果不写则”文字溢出隐藏”无效】

text-overflow:ellipsis; 【设置文字溢出隐藏方式】

}

.over:hover{ 【鼠标划过时】

text-overflow:inherit; 【取消文字溢出隐藏（可以省略）】

overflow:visible; 【取消元素溢出隐藏】

}

</style>

<div class="over" >这段文字超出了div的宽度</div>

### 21.12 文档流

所谓的文档流指的是所有的元素，在网页中默认处在文档流当中的。

而文档流就是用来规范元素在页面中显示和排列的方式的。

文档流中的元素默认按照自左向右自上向下的顺序写去排列。

块元素：

块元素在文档流中默认按照从上到下顺序排列。

块元素在文档流中宽度默认是其父元素的宽度的100%；高度默认是被其内容撑开（子元素有多高，块元素就有多高。）

内联元素：

内联元素在文档流中默认按照从左到右顺序排列。当一行容不下所有元素时，会换到下一行继续从左到右排列。

内联元素在文档流中宽度和高度都是被内容撑开的。

【浮动和定位会使元素脱离文档流。】

### 21.13浏览器默认样式的处理

在浏览器中，实际上为我们设置了很多默认样式，但是这些样式在不同的浏览器中设置效果不同。开发中需要清除掉浏览器默认样式。

如：body{margin:0;}

清除掉所有元素的默认的margin和padding。

\*{margin:0;padding:0;}

## 22、错误检查

！！！！如果出现错误，检查是否有 **单词错误** 和 **中英标点符号错误** 和 **误删除某些文字符号** 和 **标签是否成对** 和 **使用了错误的属性**和 **数字单位不对**和 **注释文字没有正确注释**！！！！

7.01:讲课：：：：

## 23、浮动：

元素默认是在文档流中的，在文档流中元素会按照从左到右、从上到下的顺序排列。

当为一个元素设置浮动以后，元素将会完全脱离文档流，不受文档流的约束，

当元素设置浮动以后，元素将会最大程度漂浮到父元素的最上边，直到遇到其父元素的边框或其他的浮动元素。

【如果浮动元素前边有一个没有浮动的块元素，浮动元素不会超过块元素。】

如果父元素的宽度不足以装下所有的浮动子元素，子元素将会换行，并从左到右排列。

【元素再怎么浮动也不会超过它上面浮动的兄弟元素的高度。】

当元素完全脱离文档流后，它在文档流中将不再占用位置，它下边的元素将会占用原来的位置。

块元素脱离文档流后，它的宽度和高度都会由子元素撑开。

浮动会使内联元素变成块元素，可以设置宽高。

文字不会被浮动元素遮挡，文字会环绕浮动元素。可以使用浮动来设置文字环绕图像的效果。

float属性：

left：向左浮动； right：向右浮动。

清除浮动：

在文档流中块元素的宽度占父元素的全部，而高度是被子元素撑开的。

当元素浮动时，它会完全脱离文档流，便不能撑开父元素的高度。

这样会导致父元素的高度塌陷。会导致页面中整个布局变得混乱。

1.给父元素指定一个高度。

如果给父元素指定一个固定的高度，父元素的高度将不会改变， 这样父元素便不能去适应子元素的高度。（不灵活）

2.设置父元素浮动。

如果给父元素设置浮动，可以使父元素包裹子元素，但是将导致父元素的宽度变成自适应，并且父元素后部的元素也必须设置浮动。

3.开启父元素的BFC，使父元素可以包含浮动子元素。

：开启BFC方法：

1.开启浮动。

2.开启元素的绝对定位。

3.设置元素的类型为inline-block。

4.设置元素的overflow属性为非默认值。 overflow:hidden

使用overflow:hidden;用来处理父元素的高度塌陷问题。

原理：设置overflow:hidden；后会默认开启BFC。

5.兼容IE6：

在IE6中不支持BFC，这种方法不能解决高度塌陷问题。

在IE6中有另一个隐藏属性： has layout 。和BFC功能类型，开启 has layout 就可以使元素包含浮动。

zoom:1; 这个属性只有IE认识，不会影响其他浏览器。

;

根据w3c标准，网页中每个元素都有一个隐藏属性“BFC（Block Formatting Context）块级布局环境”。

该属性可以设置开启或不开起。

当元素开启BFC以后，会有三个作用：

1.父元素的垂直外边距不会和子元素的外边距重叠。

2.元素不会与浮动元素重叠。

3.元素可以包含浮动元素。

使用小数作为百分比时，不同的浏览器对于小数处理方式不同，

有的浏览器会将小数部分省略，有的会将小数部分进位。尽量避免使用小数。

清除浮动元素对当前元素（浮动元素的兄弟元素）的影响：clear：

left： 清除左侧浮动元素对当前元素的影响。

right：清除右侧浮动元素对当前元素的影响。

both： 清除两侧浮动元素对当前元素的影响。

清除浮动的第二种方法：

在父元素的最后添加一个div，并且让该div清除两侧浮动 div{clear:both;} 这样div不再受浮动元素影响，便可以撑开整个父元素的高度。

使用这种方式虽然可以解决问题，但是会在页面中增加没用HTML代码，我们不使用这种方法。

清除浮动的第三种方法：

选中父元素的最后部分，使用伪类:after{;}

使用这种方法，不需要再页面中添加多余的元素，想给谁清浮动就将这个伪类.clearfix 加到那个元素上。

IE6/IE7不支持 伪类:after ，不能用这种方法。可以使用代码 zoom:1;

.clearfix:after{

content:"";

display:block;

clear:both;

zoom:1; /\*IE6/IE7使用\*/

}

/\* .clearfix:before用来解决垂直外边距重叠

\* .clearfix:after用来清除浮动\*/

.clearfix:before,

.clearfix:after{

content: "";

display: table;

clear: both;

zoom: 1;

}

PS 简单介绍：

1、修改单位：

2、Ctrl+R 出现标尺：可以拖动一些辅助线。

3、矩形选框工具：选取选区后，按 F8出现选区信息，

4、alt+滚轮： 放大缩小图片。Ctrl+1:100% ；Ctrl+0：合适大小。

5、取色：

## 24、定位

定位可以使元素定位到页面指定位置。 默认不开启。

position 规定元素的定位类型：

static 默认值，不开启定位

relative 相对定位

fixed 固定定位

absolute 绝对定位

### 24.1相对定位

开启元素的相对定位： position：relative；

1.当开启了元素的相对定位后，如果不设置偏移量，元素不会发生任何变化。

2.相对定位是相对于元素自身的位置。

3.元素开启相对定位后不会脱离文档流。

4.元素开启相对定位后，会提升一个层级（可以覆盖没有定位的元素）。

5.设置相对定位不会改变元素的性质（块元素/内联元素）

6.元素开启定位以后，可以通过一个z-index来设置元素的层级

z-index需要一个整数作为参数，值越大，层级越高，层级越高越优先显示。如：z-index:10;

当元素的position属性是一个非static的值时，可以设置4个偏移量来设置元素的位置：

top | right | bottom | left ： 元素相对于 顶部|右边|底部|左边 的偏移量；

相对定位元素比浮动元素高一个层级。

### 24.2绝对定位：

position：absolute；

1.绝对定位的元素会完全脱离文档流。

2.绝对定位的元素会相对于离他最近的开启了定位的祖先元素定位。

如果没有开启定位的祖先元素，则相对浏览器窗口定位。

3.开启绝对定位的元素会提升一个层级。

4.当元素设置绝对定位时它的宽度由内容撑开。

5.内联元素设置绝对定位后会变成一个块元素。

### 24.3固定定位：

position：fixed；

固定定位也是一种绝对定位。比较特殊的绝对定位。

固定定位大部分特点都和绝对定位相同。

固定定位永远相对于浏览器窗口定位。

固定元素一旦定位，就会固定在窗口的某一位置，不会随着滚动条滚动。

## 25、元素透明度

opacity：设置元素的透明度。

值为 0 - 1； 0是完全透明，1是完全不透明。

opacity不支持IE8之前的浏览器。

IE8中使用（过滤器）： filter：alpha（opacity=50）；opacity值是0 - 100 。 0是全透明，100是不透明。

## 26、背景

背景颜色：background-color：red;

背景图片：background-image：url(img/01.jpg);

背景图片重复：background-repeat：repeat；

背景图片开始的位置：background-position：x y;

背景图片是否可滚动：background-attachment：scroll默认滚动 | fixed不滚动

如果背景图片比元素大，则会显示左上角部分。

背景图片的路径可以使用相对路径，

样式写在哪个文件里，就要按照那个文件计算相对路径。

如果图片比元素小，则图片默认在元素中平铺；

background-repeat： 用来设置背景图片的重复方式：

repeat：默认值：水平和垂直方向重复。

repeat-x： 水平方向重复。

repeat-y： 垂直方向重复。

no-repeat： 不重复。

可以同时指定背景颜色和背景图片，这样背景颜色将会作为图片的底色。

background-position: 布局图片位置

top left | top right | top center | bottom left | bottom right |　bottom center |　left center | right center | center center

从 top left bottom right center 中选中两个值来指定一个位置，共9个位置。如果只写一个值，第二个值默认为center。

指定两个长度单位，（x,y）背景图片在水平方向和垂直方向位移。 x y 可以指定负值。

background-position： x y ;

背景图片是否可随页面滚动：

background-attachment：

scroll；默认值；

fixed：图片固定在页面中，不随页面滚动，

当设置fixed后，背景图片会相对于浏览器的窗口进行定位，

设置完成后，虽然背景图片不会随页面滚动，但是元素可以随页面滚动，当页面从浏览器窗口消失时，图片也会消失。。

一般将该样式应用给body

浏览器加载图片是在图片第一次被使用时才加载。

没有使用的图片会在使用时才加载，加载需要花费一定的时间，这样在显示的过程中会有一小段时间图片无法显示，会影响用户的体验

### 26.2图片整合技术 （CSS-Sprite）

将多个图片整合成一张图片，然后通过background-position来调整显示图片。

将多个图片整合到一起，浏览器只需要发送一次请求便可以加载所有图片，提高了访问速度，提升了用户的体验。

将多个图片整合到一起，图片占用空间更小，加快加载速度。

缺点:

1、图片整合只能用于背景图片。

2、图片整合中的内容不会被搜索引擎检索。

背景图片属性简写

background:可以设置所有背景相关的样式并且没有顺序要求，没有数量要求。不写的属性会使用默认值。

背景颜色、背景图片、背景图片重复、背景图片位置、背景图片滚动。

7.04 讲课：：：：：

## 27、表格 table

表格一般用来表示一些结构化的数据

在html中我们同样也可以创建表格。

在html中使用<table></table>标签创建表格。<th></th>表示表头 ,<tr></tr>表示一行,<td></td>表示一个单元格。

有几行就用几个tr，有几个单元格就有几个td；

横向合并单元格 <td colspan="2">

纵向合并单元格 <td rowspan="2">

表格样式

table边框是最外部边框；

tr不能设置边框；

td边框是单元格边框。

边框之间的距离：table{ border-spacing:0px;}

合并边框线： table{ border-collapse:collapse;}

设置了border-collapse:collapse;后 边框之间的距离变为0 ，即 无需设置 border-spacing:0px;

table是块元素。

隔行变色： tr:nth-of-type(even){ background-color:#fba; } 偶数even，奇数行odd。

鼠标滑过变色： tr:hover{ background-color:#bfc;}

### 27.2 长表格：

有时我们去创建一个表格时，表格中的内容较多，我们需要将表格分为几部分：

表格的标题：<caption></caption>

表格的头部：<thead></thead> 永远显示在表格的上部

表格的主体：<tbody></tbody> 永远显示在表格的中部

表格的底部：<tfoot></tfoot> 永远显示在表格的底部

这三个标签用来将表格分成几部分，我们需要将相应的 tr 编写到表格中。

如果在表格中没有写tbody，则浏览器在渲染页面时，会自动的在table中添加tbody，并且将tr放到tbody中。

所以，tr实际上是tbody的子元素，而不是table的子元素。

我们创建表格时应该把tbody写上。然后将所有内容放到tbody中。

### 27.3表格嵌套：

我们可以在表格中嵌套其他的表格。

很久以前，我们通过表格之间的嵌套来对网页进行布局。

没有CSS时，布局主要由两个标签：table 、font；

表格的列数由单元格最多的行决定。

使用表格布局结构不清晰，搜索引擎不能理解表格中的内容。不便于SEO。

使用表格布局嵌套过深，不利于后期的维护。

## 28、表单：

表单用来向服务器提交信息。

form：创建表单

属性：action：需要一个服务器地址，指定地址以后，表单将会把数据提交给该地址。

method：指定请求的类型。get：默认值，发送get请求；post：发送post请求。

表单项：

<input type="" name="" value="" />

根据type的不同，input的显示方式也不同。

：text ：文本框

：password：密码框

：radio：单选按钮：单选按钮和多选框都是通过name属性进行分组，相同的name属性是一组。这种选择框不需要用户填写内容，必须制定value值。

：checkbox ：多选框： 单选框和多选框中可以添加 checked="checked" 来将该选项设置成默认选中。

：submit：提交按钮： 通过value属性值来制定提交按钮上的文字。

：reset：重置按钮： 可以将表单中的内容重置到默认值。

下拉列表：

<select name=""> --> 用来创建一个下拉列表

<optgroup label=""> --> 指定选项的分组，label用来指定分组的名字

<option value=""></option> ---> 用来创建下拉列表中的选项

</optgroup>

</select>

- 可以通过在option中添加selected="selected"来设置选项为默认选中

多行文本域

<textarea name="" rows="" cols=""></textarea>

rows:指定行数，可以使用css中height代替

cols:指定列数，可以使用css中width来代替

按钮

<button type="">按钮上的文字</button>

- type的可选值：

submit : 提交按钮

reset : 重置按钮

button ：普通按钮

- fieldset

- 为表单项进行分组

- legend

- 指定分组的名字

- label

- 指定表单项的提示文字

- 属性：for，可以指定一个表单项的id

这样当点击label时，指定的表单项将会获取焦点

## 29、框架集 <frameset>

<frameset>

- 框架集和iframe（内联框架）的作用类似，都可以向网页引入页面

但是不同的是iframe只引入一个，frameset可以引入多个

- 例子：

<frameset rows="20%,\*,50%">

<frame src="" />

<frame src="" />

<frameset rows="20%,\*,50%">

<frame src="" />

<frame src="" />

<frame src="" />

</frameset>

</frameset>

- 属性：

rows --> 内联框架中所有页面，按照一行一行的排列，可以在rows指定每个框架的高度

cols --> 内联框架中的所有页面，按照一列一列的排列，可以在属性中指定每个框架的宽度

- 在实际开发中并不推荐使用内联框架和框架集，但是如果非得用，建议使用框架集。

7.05讲课：：：：：

## 30、CSS-Hack技术

1、CSS-Hack技术：

可以用来专门为某种或某些浏览器设置单独的样式。而不影响其他的浏览器。

不到万不得已的时候不用。

这种注释只有IE认识，其他浏览器会当成普通注释，而IE会显示。(IE10以上版本不支持）

这种注释应该写在style标签外面。 alt+/

<!--[if IE 8]>

<![endif]-->

2、在属性名前面加了 下划线\_ 的样式 ，只有IE6认识。 如： \_background:red;

在属性名前加了 星号\* 的样式，只有IE7和IE6认识。如： \*background: red;

在样式后面添加 \9，只有IE6-10认识。 如：background: red\9;

在选择器前添加 “\* html ”，只有IE6认识该选择器。

3、IE6不支持透明背景png。 png格式的图片的透明背景在IE6中不能正常显示。

4、带有-min.js的js文件，是经过压缩过的代码。

引入外部的js文件: <script src="" type="text/javascript" charset="utf-8"></script>

编写js代码来处理透明图片问题：

## 31、对于一个新项目的准备工作：

准备工作：

1.新建项目

2.清除默认样式

3.创建CSS文件

4.引入CSS文件

5.分析页面，规划架构。

命名习惯

1.驼峰命名法：

首字母小写，每个单词的首字母大写。helloWorld

2.使用\_或-连接单词，

所有字母都小写，单词之间用 \_或-连接。hello\_world

chrome浏览器字体最小为12px；

火狐浏览器background no-repeat 不起作用？

按alt+左键点击图层，只显示该图层； 再用ctrl+左键点击图层，加选一个图层。

选择工具 - 自动选择+图层 --> 点击鼠标可以选中该图片所在图层。 图像-裁切 即可剪切出这个图片。

## 32、浏览器兼容中的问题

1、对于IE6的png透明背景的解决

添加的js代码应该放到body的后面。这样可以等浏览器加载完成所有元素后再执行，减少出错。

2、在IE6和IE8中文本域textarea默认有个滚动条，

去掉滚动条的办法：overflow：auto；

3、设置文本域不能修改大小：resize：none；

4、文本框input 在浏览器中有默认的内边距，但在不同浏览器中效果不同。

应设置样式padding：1px 10px；然后在input的宽高中分别减去2px和20px。

再设置行高使文字垂直居中： line-height：31px；

5、文本域在浏览器中默认有内边距和外边距，但是在不同浏览器中效果不同。

应设置样式，清除margin和padding。

6、按钮button在浏览器中默认有内边距。但在不同的浏览器中效果不同。

应注意清除。

7、input 输入框 属性 placeholder="水印内容" ；

当没有输入内容时，显示水印内容，输入文字时 提示文字消失。

这个属性IE8及以下浏览器不支持。

8、内联元素中间有空格时 ，会在浏览器中体现出来。

可以使其 浮动 即可。

9、行高在不同的浏览器中默认效果不同。

可以设置line-height：1；（1倍行高）

10、IE6只支持a标签的hover

11、设置光标样式：cursor：

porintr:表示小手

defaul：默认的箭头

cursor这个样式不是必须放到hover中的。

# 三、JavaScript

## 33、JavaScript 简介

JavaScript是编程语言。

机器语言==> 汇编语言==> 高级语言

JavaScript诞生于1995年，它的出现主要是用于处理“网页”中的“前端验证”。

网景公司JavaScript、微软公司JScript、标准版ECMAScript 标准ECMA-262

ECMAScript是一个标准，而这个标准需要由各个厂商去实现。

不同的浏览器厂商对该标准会有不同的实现。

chrome的JavaScript实现方式 v8

一个完整的JavaScript实现应该由以下三个部分构成：

ECMAScript（标准） 、DOM（文档对象模型）、 BOM（浏览器对象模型）

JavaScript的语言可以在两个地方使用：浏览器端、服务器端

js的特点：

解释性语言

类似于C和Java的语法结构

动态语言

基于原型的面向对象

面向对象：1.找对象：2.搞对象：

JS代码需要编写到网页的script标签中。script标签习惯于写到head标签中。

script需要设置一个type属性，type="text/javascript"，但是这个值不是必须的，默认值也是它。

//JS注释

常见的三个输出语法：

1、在浏览器中弹出一个警告框

alert（"警告框中的内容"）;

2、在页面中打印一段内容

document.write("打印的内容");

document.write("<h1>h1中的内容</h1>");

3、cansole.log可以向控制台输出内容

//控制台中的内容不会在页面中显示，普通用户看不见

//我们可以通过cansole.log 调试程序。

cansole.log("在控制台中出现的内容");

//单行注释，我们可以使用单行注释对程序进行说明，一定要养成一个良好的编写注释的习惯。

//编写注释时尽量简单明了。

/\* 多行注释，不能嵌套 \*/

1.代码注释内容：

代码作用、编写时间、编写人。

2.可以通过注释对程序进行调试。

确定错误行。

## 34、JavaScript基本语法：

### 34.1 JavaScript基本语法

1.在JS中 严格区分 大小写 ！！！

2.在JS中 每个语句的结束都要以 分号; 结尾。

如果语句不以分号结尾，浏览器会自动为语句添加分号。

JS代码都是由浏览器解析的，如果不加分号浏览器需要花费一定的资源去自动添加分号，多少会影响一些性能，而且有时候会加错分号。

所以在编写代码时，一定要在语句结束时添加分号。

3.在JS中 多个空格和多个换行会被解析器（浏览器）所忽略。

所以我们可以使用换行、空格来调整代码的格式，使其方便阅读。

### 34.2 JavaScript编写位置

1.编写到 外部js文件中。

引用方法：<script src="js/script.js"></script>

用来引用外部js文件的script标签，不能在其中编写代码，编写了也不会执行，可以再写一个script标签。

2.编写到<script></script>标签中

//以下方式我们称为结构与行为耦合，不推荐使用。

3.编写到标签的属性中：

<button onclick="alert('内容');">按钮</button>

4.编写到a标签的href属性中

当href标签中写了js代码以后再去点击超链接，超链接不会跳转页面而是执行js代码。

<a href="javascript:alert('内容');">a标签</a>

### 34.3 常量和变量

在所有的编程语言中，都有两种量，

字面量（常量）: 表示一个固定值， 如1 2 3 ，字面量可以在程序中直接使用。

字面量的值是不可变的。

变量： 变量可以用来保存字面量。变量的值不是固定的。使用变量可以明确字面量的意义，方便修改。

使用变量先 声明变量：

在js中使用 var关键字来声明一个变量。

var x; //声明一个变量x。

没有赋值的变量，默认值是undefined

x = 123; //为x赋值。

一般同时进行声明和赋值

var x = 123;

可以同时声明多个变量：使用逗号分隔。

var a,b=2,c=3;

### 34.4 标识符

在语言中，所有可以由我们自主命名的内容都是标识符。

比如：变量名、函数名、对象的属性名。

标识符的限制：

1.标识符必须以 字母 、 \_ 、 $ 开头 。

2.标识符中可以使用字母、数字、 \_ 、$ 。

3.标识符不能是 关键字或保留字。

关键字就是在ES标准中已经使用的内容，ES标准赋予了这些关键字特殊的意义，我们不能使用。

保留字是以后可能会用到的内容。

命名习惯：

js中命名时采用驼峰命名法。首字母小写，每个单词的开头大写。

### 34.5 数据类型：

JavaScript中一共有 5种基本数据类型、1种引用数据类型

基本数据类型：

字符串型 String

数值型 Number

布尔型 Boolean

null型 Null

undefined型 Undefined

引用数据类型：

对象 Object

使用typeof运算符来检查数据的类型。

语法： typeof 数据 ；

使用typeof得到该数据类型的描述的字符串。

1.字符串：表示的是由一个个字符构成的内容

在js中所有的字符串都需要使用 "双引号" 或者 '单引号' 包裹，来标识这个内容是字符串。如果不加 引号 会被认为是一个变量名。

使用双引号或者单引号没有任何区别，但是必须成对出现。同一类型引号不能嵌套。

转义字符：表示特殊字符：以 \ 开头

换行 \n ； 斜杠 \\ ； 制表符Tab \t ； 空格 \b ； 引号 \" ；

任何数据类型和字符串做加法都会变成字符串。

使用 + 可以将多个字符串拼接成一个字符串。

var str = "<p>" + "这是一个p标签的格式" + "</p>" ;

2.数值型 Number

在js中所有的数字都是Number类型。包括整数和小数（浮点数）

在js中数字最大范围是 正负308位 。如果超过该范围，将会返回 Infinity 。

正无穷 返回 Infinity， 负无穷 返回 -Infinity。 这两个值的类型也是Number。

数值类型中还有一个特殊值 NaN （Not a Number）。任何数值和NaN计算时都是NaN。

在js中对数值类型进行计算时，浮点数计算时可能会出现意想不到的结果，它对浮点数的计算不够精确。

使用js做小数计算时，它都是返回一个近似值，所以尽量避免使用小数计算。

不能用js做 涉及到 钱 的 小数计算。

js中50位以内的整数，可以保证精确计算。超过50位的数值结果不精确。

3.布尔型Boolean

布尔值就是用来表示真和假。

布尔值主要用来进行逻辑判断的。

布尔值只有两个值 ： true 真， false 假。

4.Null ：

Null类型的值只有一个： null。

使用typeof来检查null值的类型会返回object。

Null值专门用来表示一个 "空的对象"。

当我们创建一个变量，用于保存一个对象时，但是又不想马上为该变量赋值，可以给该变量赋值Null。

5.Undefined

Undefined类型的值只有一个： undefined

当声明了一个变量但是又没有为变量赋值时，这个变量就是undefined。

### 34.6 类型转换

基本数据类型之间可以互相转换。

String可以转换为Boolean

Number也可以转换为String。

a.将其他数据类型转换为String：

1.调用指定类型的 toString() 函数。

调用数据的toString()函数，它会将数据转换为String类型并返回，并不会影响原来的数据。

这种方式不能应用于null值和undefined值。

我们不能调用一个 null或undefined 的函数。

var num = 123; var str=num.toString();

2.使用String（）；函数。

将要转换的变量，作为参数传递给String（）函数，它会自动将其转换为String类型。

在这个函数内部，如果是 非Null或Udefined的值，则会默认调用其toString（）方法进行转换。如果是Null或Udefined值，则会转换为null或undefined

这种方式可以转换null和undefined。

var num=123; str=String(num);

b.将其他的数据类型转换为Number，有三种方法：

1.使用Number（）函数

num= Number(str);

使用Number（）函数转换含有非数字内容的字符串，将会转换为NaN。

只有空格的字符串或者是空的字符串都会转换成0 。

true转换成1，false转换成0； null转换成 0；undefined转换成 NaN。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number() | | | | |
|  | 普通数字 | 1 | 0 | NaN |
| String | 纯数字字符串 |  | 只有空格的字符串 ” “  空的字符串 ”” | 含有非数字内容的字符串 |
| Boolean |  | True | False |  |
| Null |  |  | null |  |
| Undefined |  |  |  | undefined |

2.使用parseInt()函数

只可以把字符串转换为数字。只能转换成整数。自动忽略小数点以后的内容。

parseInt在转换时，会从字符串的左侧开始读取，当读到一个非数字的内容时，会停止解析。如果第一个字符不是数字则返回NaN。

3.使用parseFloat（）函数。

parseFloat功能和parseInt类似，只可以把字符串转换为数字，但是可以处理小数。可以获取字符串中合法的小数部分。

在ES3中和ES5中处理的方式不同，像 070 在ES5中会当成10进制数处理，而ES3中会将070当成8进制看。

我们可以在parseInt中传递一个参数，来说明当前进制。 str= parseInt(str,10);

为了兼容ES3和ES5，最好每次都指定进制。

c.将其他类型转换为布尔型

Boolean（）；

将一个Number类型转换为Boolean时，除了 0 和 NaN 全都是true。

将一个String类型转换为Boolean时，只有 空字符串"" 是false ，其余全是true。

null 和undefined 都会转换为 false。

除了值为null的对象，对象都会转化为true。

|  |  |  |
| --- | --- | --- |
| 其他类型 ▶▶▶ 布尔类型 Boolean(); | | |
|  | 转换为false | 转换为true |
| Number | 0 、 NaN | 其他非0数字 |
| String | 空字符串 “” | 其他非空字符串 |
| Null | null | 无 |
| Undefined | undefined | 无 |
| Object | 值为null的对象 | 其他任何对象 |

### 34.7隐式的类型转换：

其他类型转换为String：

变量 = 变量+"" --> String

任何数据和字符串进行加法运算，都会转换为字符串，然后再做拼串操作。

底层使用String();

其他类型转换为Number：

变量 = +变量 --> Number

使用一元的+会使变量自动转型为Number类型

底层使用的是Number();

其他类型转换为布尔：

变量 = !!变量 --> Boolean

我们可以为一个变量做两次取反操作，来使其变为Boolean

底层使用的是Boolean();

## 35、 运算符：

运算符：也叫操作符。

运算符可以对JS中的数据进行各种各样的运算，比如 + - \* / %

typeof 也是一个运算符，用来检测变量的数据类型。

二元运算符：

+ 可以对其左右两侧的值进行加法运算。

使用加法运算时不会对原来的值产生影响。

对字符串进行加法运算时，他会将左右两个字符串拼接成一个字符串并返回。

任何数据类型和字符串进行相加，都会转化为字符串，并做拼串操作。

它在转换时，底层也是使用的String（）函数。

我们经常为一个数据加上一个空串，以使其变成字符串。

除了字符串，其余数据类型之间做加法运算，会将其转换为Number类型以后再计算。 NaN和任何数据做运算都是NaN。

加法运算是从左到右依次计算。

- 减法：

减法和加法类似，

但是任何类型的数据做减法时都会转换成数字。

除法、乘法、取模 也是这样。

可以使用任意数据类型减0 ，使其转换为Number类型。

注意：其底层使用的是Number（）函数。

\* 乘法：

/ 除法：

% 取模：取余

### 35.1 一元运算符：

一元运算符只需要一个变量即可。

一元 + ： 正号；

一元 - ： 负号。

+ 和 - 不会对原值产生影响。

给一个变量添加 正号+ 不会发生变化。

给一个变量添加 负号- 时， 原正值变成负值，原负值变成正值。

当为非数字类型使用 +或- 时，会自动将其转换为Number类型，然后再运算。

一般可以给一个变量添加一个 + 来将其转换为一个Number类型，其底层使用的是Number()函数。

### 35.2 自增和自减

1.自增：++ ：在原变量基础上 加1； var a = 1; a++;

无论是 前++ 还是 后++ ，执行完了以后，原变量都会立即自增1；

区别： 前++ 和 后++ 的值不一样。

var a; 当调用 a++时，a的值立刻自增1，而 a++的值和 a的值不一样， a++ 的值是a自增以前的值。

var a; 当调用 ++a时，a的值立刻自增1，而 ++a的值和 a的值一样， ++a的值是a 自增以后的值。

2.自减：-- ：在原变量基础上 减1；

无论是 前-- 还是 后-- ，执行完了以后，原变量都会立即自减1；

a-- 的值是变量自减前的原值。

--a 的值是变量自减以后的值，

自增、自减以后会修改原变量的值。

a++ 、a-- 是a原值

++a 、--a 是变化后的a值，当前a的值。

### 35.3 逻辑运算符

1.非 ! ：取反，可以对一个布尔值进行取反操作。! true = false。 非 不会对原值产生影响。

对一个值两次取反，该值不变。 对非布尔类型的值取反操作，则会将其转换为布尔类型再取反。转换方式底层使用Boolean();

可以使用!!将一个非布尔类型转换为一个布尔类型。

2.与 && ：&&可以将符号两侧数据进行与运算。两个值都是true则返回true；只要有一个false则返回false。

3.或 || ：||可以将符号两侧数据进行或运算。两个值都是false则返回false；只要有一个true则返回true。

JS中的或和与都是短路的或和与：

即：如果可以通过第一个值来判定结果，则不会再去检查第二个值。

如：var flag = false && alert("判断是否运行")；【alert()-->false。】

与或非运算均可以使用非布尔值，它们在处理非布尔值时需要将其转换为布尔值，然后在做相应的操作。。

对非布尔类型的数据进行与或运算时，它不会返回布尔值，而是返回原值。

与运算 var a,b; a && b;如果两个值转换完都为true，则返回后面的值；如果两个值中有false，则返回靠前的false。

或运算 var a,b; a || b;如果两个值转换完都为false，则返回后面的值；如果两个值中有true，则返回靠前的true。

由于各个浏览器对ES的标准实现的方式不同，有一些对象可能在不同的浏览器中不同。

var a = a || b ; 可以判断a和b谁存在。返回存在且靠前的值。

### 35.4 赋值运算符：

赋值运算符可以用来作为一个变量进行赋值。

= ：就是一个赋值运算符。可以将右边的值赋值给左边的变量。

+= ：可以使原变量自增指定的数值。a += b; 相当于 a=a+b;

-+ ：a -= b; 相当于 a=a-b;

\*= ：a \*= b; 相当于 a=a\*b;

/= ：a /= b; 相当于 a=a/b;

%= ：a %= b; 相当于 a=a%b;

### 35.5 关系运算符：

用来比较两个值之间的大小关系。会返回一个布尔值来说明结果。

小于 < :

小于等于 <= ：

大于 > ：

大于等于 >= ：

当使用一个关系运算符对非数字的内容进行比较时，会将其转换为Number类型。

当为 两个字符串 比较大小时，不是将其转换为数字，而是将其转换为了 Unicode编码来进行比较。

注意：比较两个字符串型的数字时，往往不能得到正确的结果，所以注意比较字符串数字的时候需要转型。（如：在前面加一个 + ）

：比较两个字符串时是从左到右一个字符一个字符的进行比较，然后取第一个能比出大小的字符作为结果。

相等运算符： == ：判断两个值是否相等。

可以判断 == 两侧的值是否相等，相等返回true。不相等返回false。

相等运算符可以自动把值进行类型转换，类型转换后的值相等，它也返回true。

全等运算符 ： === ：

全等和相等类似，但是不会自动转型。所以即使转换后相等，也会返回false。不同类型的数据不相等。

NaN 不和任何值相等，包括它本身。

我们不能通过相等或全等来判断一个值是否为NaN。

在JS中提供了一个函数： isNaN();这个函数可以用来检测一个值是否是NaN，，如果是则返回true，否则返回false。

这个函数会自动将非数字进行转换，转换为Number类型，然后判断。

undefined值衍生自null值。所以 undefined == null 返回true。但是 undefined === null 返回false。

不等运算符： != ：

用来判断两个运算符是否不相等，如果不相等则返回true，如果相等则返回false。

转型以后相等，不等运算 也认为相等。它会自动为值转型。

!== 全不等运算符 ：： 和不等类似，但是它不会自动转换类型。 类型不同也被认为不等。

### 35.6 条件运算符：

条件运算符也叫三元运算符、三目运算符

可以根据一定的条件来决定返回哪个值。

表达式1 ? 表达式2 : 表达式3；

如果表达式1的结果是true，则执行表达式2，并返回结果；

如果表达式1的结果是false，则执行表达式3，并返回结果。

如果三目运算符编写的太长的话，不方便阅读，尽量不要写太长。如果非得写这么长，可以给表达式添加括号来说明结构。

比较两个数的大小：

var a = 234,b = 456;

a>b ? alert("a>b"):alert("a<b");

求三个数中的最大值：

var num1 = 423,num2 = 314,num3 = 45;

var max = num1>num2 ? num1>num3 ? num1:num3 :num2>num3 ? num2:num3;

可以使用括号来说明结构：

var max = num1>num2 ? (num1>num3 ? num1:num3) :(num2>num3 ? num2:num3);

console.log(max);

### 35.7 运算优先级：

在JS中运算符的优先级：先乘除后加减。 '与&&' 运算 比 '或||' 运算 优先级高。

优先级的顺序按照优先级表 的从上到下排列，位置越靠上越高。

优先级越高，越优先计算，优先级相同按照编写的顺序从左到右的顺序计算。

可以使用小括号()来改变运算顺序。

## 36语句

语句： statement

语句是程序的最小单位，每一个语句都是向计算机发送一条指令。

JS中每个语句都需要以分号；结尾，以标识语句的结束。

代码块：

在JS中使用大括号{ } 来为代码分组，在同一个大括号中的语句我们称之为一个代码块。

代码块的作用就是为语句进行分组。代码块后面不用写分号。

注意：JS 中的代码块和其他语言中的有区别，只用于分组，没有其他任何作用。在JS中没有块级作用域。

流程控制

我们的程序默认是顺序执行的，按照从上到下的顺序执行的。从第一条语句开始，直到执行到最后一条语句结束。

流程控制就是指控制计算机执行程序的流程。

流程控制语句主要分为两种：

条件分支语句；循环语句。

### 36.1条件判断语句：

根据一个条件来决定是否执行某条语句。如果判断结果为真则执行，如果判断为假，则不执行。

if ... else switch ... case

if语句（条件判断语句）

语法1：

If (条件表达式) { 语句；}

如果if中的条件表达式执行结果为true，则执行if后的语句，如果条件表达式执行结果为false则不执行if后的语句。

if语句，只能控制紧挨着它的语句，不能控制其他语句。

我们可以在if后面跟着一个代码块，则当判断结果为true时，代码块中的代码会全部执行，判断结果为false时，代码块中的代码都不执行。

如果if后只有一条语句，并不需要编写代码块，但是根据开发的习惯，即使只有一条，也需要写上大括号。

if 语句可以嵌套。

当条件满足时执行一些语句，当条件不满足时执行其他语句。

if（条件表达式）{ 语句1 ；}

else { 语句2 ；}

如果if中的条件表达式成立则执行语句1，否则执行语句2。

if...else 可以嵌套

If (条件表达式1){语句1；}

else if （条件表达式2）{语句2；}

else {语句3；}

这样可以依次判断if后的条件表达式，如果满足则执行该if后的代码并结束，如果表达式不成立，则继续向下判断。

这个语句只会执行一个if后的代码，一旦遇到满足条件，后续的不再进行判断，如果条件都不满足，则执行最后的else语句。

### 36.2 输入弹窗prompt();

prompt("提示文字");

prompt()；函数可以用来在浏览器中弹出一个带有提示文字和输入框的提示框，它需要一个字符串作为参数，这个字符串将会作为弹出框的提示文字。

这个函数会将用户输入的内容作为返回值返回。不输入则返回 空字符串。

prompt()；返回值是String。如果要用数字，则 var a = +prompt ("TIP");

var input = +prompt("提示文字");

document.write();是在浏览器body中输出，换行需使用<br />换行

### 36.3 switch...case

switch(表达式){

case 表达式：

语句；break；

case 表达式：

语句；break；

default：

语句；break；

}

如果switch后的语句和某个case后的语句执行结果全等（ === ），则执行相应case后的语句；如果均不等则执行default后的语句。

break可以中断switch语句执行。如果不加break；当指定的case中的条件满足时，case后的【所有】代码都会执行。执行break后，break后面所有的代码都不会执行。

switch使用的是全等=== ，不会转换数据类型。

switch和if的功能是一样的，所有switch语句均可用if语句替换，但是switch结构更清晰，两者没有本质区别。

### 36.4 while循环

循环语句可以使一组代码多次重复执行。

while(条件表达式){

语句；

}

while循环 每次循环先判断while后的条件表达式的真假，如果为真则执行循环中的语句，如果为假则停止执行。

如果把while中循环条件表达式设置为true时，则while循环变成一个死循环。

while循环也可以使用break来结束循环。

1.定义一个初始化变量

2.指定一个循环执行的条件

3.每次循环修改变量。

var i = 0 ; while ( i<10 ) { alert(i); i++; }

### 36.5 do ... while 循环

do{ 语句； } while（条件）；

do...while 和while 基本相同，只不过while是先判断再执行循环，do while 是先执行循环再判断条件表达式。 do while可以保证循环至少执行一次，其他的没有区别。

### 36.6 循环的四个要素：

1.初始化变量

2.循环执行的条件

3.更新表达式（变量）

4.要执行的语句。

### 36.7 for循环

for(初始化表达式; 条件表达式; 更新表达式){ 语句； }

for循环和while循环实际上是一致的，只不过for循环可以将 初始化表达式、条件表达式、更新表达式 都统一写在一起。

for循环执行顺序：

1.初始化表达式； 对变量进行初始化

2.判断条件表达式； 判断循环是否执行，

3.如果结果为true，则执行循环中的语句； 语句执行完成，则执行 更新表达式； 然后跳转至2.判断条件表达式

如果结果为false，则跳出循环

如果for循环中的三个表达式都不写，则变成了一个死循环。

将变量的初始化定义在循环内部，会导致每次循环都重新初始化变量。

计数 count

获取个位数字: var a= i%10;

获取十位数字：var b= parseInt(i%100 / 10);

获取百位数字：var c= parseInt(i%1000 /100);

获取千位数字：var d= parseInt(i%10000 /1000);

### 36.8 break和continue

break可以用来退出循环。可以在循环或switch语句中使用，当break执行以后，循环或者switch将立即停止。

break在处理循环时，只会跳出离它最近的一层循环，不会影响其他循环。

我们可以给一个循环指定一个标签label，在break时可以直接指定跳出哪个循环。

continue 和break类似。

continue可以用来跳过当次循环。只会影响离它最近的一层循环，也可以使用label标签来指定影响哪个循环。

## 37、对象：

JS中一共有6种数据类型：5种基本数据类型和1种Object类型。

Object我们称为对象，它是一种引用数据类型。

5种基本数据类型使用过程中有时候有一些局限。基本数据类型是独立的。

实际上我们的对象就相当于一个容器，可以将多个数据统一放到容器中管理。

所以，对象也可以称为是一种复合数据类型。

### 37.1 对象的分类：

1.内建对象：由ES标准定义的对象，在所有ES标准的浏览器都可以直接使用。

如：String Object Function Math RegExp

2.宿主对象：由各个浏览器厂商定义的对象，也可以在浏览器中直接使用。

如：console document window

3.自定义对象：由开发人员自己创建的对象。

### 37.2 创建一个对象：

1.使用new关键字来创建一个新的对象。new关键字后面会跟着一个函数，这个函数专门用来创建对象。

像这种专门用来创建函数的对象我们称为构造函数（constructor）

var obj = new Object();

2.向对象中添加属性:

对象.属性名=属性值；

obj.name="孙悟空"；

3.读取对象中的属性。

对象.属性名

alert(obj.name);

对象的属性名并不受标识符规则约束，但是尽量避免是用。

当命名一些特殊的名字时，比如有空格，数字开头，特殊符号等，可以采用一种特殊语法

obj[属性名字符串] = 属性值；

读取特殊属性名时，也需要使用特殊语法：

对象[属性名字符串]

虽然对象的属性名没有约束，但是一般不使用稀奇古怪的名字。

如果读取对象中没有的属性时，它不会报错，而是返回undefined

使用[]方式更加灵活，因为[]中可以直接传递一个变量，可以动态获取属性值。

4.修改属性：

对象.属性名= 新的属性值。

对象[属性名]=新的属性值。

5.删除属性：

使用delete删除对象中指定属性

delete 对象.属性名;

delete obj.name;

对象的属性值也是没有任何限制，可以赋值为任何类型，甚至可以赋值一个对象。

基本数据类型和引用数据类型的区别：

基本数据类型的修改，修改一个变量不会影响其他变量。

引用数据类型的修改，会影响其他引用该数据的变量。

内存分为栈内存和堆内存

栈内存用来保存变量；堆内存用来保存对象。

基本数据类型的值全都是直接保存到栈内存中的。当我们使用var c=a 时，它会将变量a的值赋值一份，赋值给c。所以修改a的值时，不会影响c的值。

当使用new关键字调用一个构造函数时，浏览器会立即在堆内存中开辟一个空间，保存对象。每个对象在堆内存中都有一个唯一的地址。

变量中保存的实际上是内存的地址，再通过变量操作对象时，实际上是操作栈内存中的对象。

对象是在第4步给属性定义及赋值的。读取未声明的obj属性返回undefined.

6、内存分为两种：栈内存和堆内存

栈内存用来储存变量，堆内存用来储存对象。

当创建一个基本数据类型的变量时，变量会保存到栈内存中，同时变量会直接保存基本数据类型的值；

当修改一个基本数据类型时，由于每一个变量都是独立的，不会影响其他的变量。

当创建一个引用类型（对象）的变量时，变量会保存到栈内存中，对象会保存到堆内存；

而在栈内存的变量中保存的是堆内存的地址，当我们对变量进行操作时，实际上是在操作堆内存中的对象。

可以有多个变量同时指向一个对象，这样当我们修改一个变量时，其他的变量也会受到影响！

如果有多个变量同时指向一个对象，将其中任意一个变量置为null均不会删除该对象。

7、比较两个对象

当比较两个对象是否相等时，比较的是对象的内存地址，如果内存的地址相等则返回true，否则返回false。

我们一般使用null来表示一个空的对象。所以我们使用typeof检查null类型时，返回object；

2、创建对象的两种方法

1.使用构造函数创建对象。

var obj = new Object();

2.使用对象字面量的方式

var obj = { } ;

这两种方法创建的对象是一样的。没有区别。

通过对象字面量的形式创建对象，我们可以在创建对象时直接添加属性,多个属性之间用逗号隔开，【最后一个属性不能加逗号】。

var obj = { 属性名:属性值 , 属性名:属性值 };

var obj = { name:"孙悟空" , age:18 , gender:female};

相比构造函数，我们使用对象字面量的方式来创建对象的情况会更多些。

### 37.3对象的字面量

var obj = { };

使用这种方式也可以创建一个对象，原理跟new Object()是一样的。

创建对象时直接添加属性：

var obj = {

属性名：属性值,

属性名：属性值,

"特殊字符属性名".属性值

}

alert(obj); ==> 弹出[object Object]

属性名可以用 "双引号" 包裹，但是性能不好。 调用时可以将属性名用 [中括号] 包裹.

对象的属性值没有任何类型限制，也可以是对象、函数。

### 37.4枚举对象中的属性： for..in

for(var n in obj){

document.write("属性名："+n+"&nbsp;"+"属性值："+obj[n]); //输出名值对

}

当我们使用for..in枚举一个对象中的属性时，对象中有几个属性，循环就执行几次。每次都会将对象中的一个属性的名字赋值给 变量n ，调用属性使用 obj[n]。

### 37.5 创建对象的方法

1.调用对象的构造函数（constructor）创建对象

var obj = new Object();

obj.name = “jay”;

obj[“age”]=18; 不加"引号"会被认为是一个变量。

delete obj.age;

1. 使用对象字面量的方式

var obj = { 属性名:属性值 , 属性名:属性值 } ; 【最后一个属性不能加逗号】

var obj = { } ;

var obj = { name:"孙悟空" , age:18 , gender:female};

obj.name;

### 37.6 创建函数的方法

1.使用new Function();来创建一个函数对象 [不用]

var fun = new Function();

var fun = new Function("alert('hello');alert('hello2');");

2.使用函数创建的表达式

function 函数名(形参1 , 形参2){ 代码; }

function fun1(){

alert("1");

alert("2");

return “任意类型的返回值” ;

}

3.匿名函数

var 函数名 = function(形参){ 语句; };

var sum = function(a,b){

var result = a+b;

return result;

};

sum(123,456);

## 38、函数：

函数的作用：函数可以将一些功能性的代码封装到一起，当调用函数时，函数中封装的这些代码将会一起执行。

在JS中，函数就是一个对象。里面也可以执行添加属性删除属性等操作，只不过这个对象中可以封装一些可以执行的代码。

函数的类型是function ， typeof fun 显示为function。

创建一个函数对象：

1.使用new Function();来创建一个函数对象 [不用]

var fun = new Function();

在创建函数时，可以将代码以字符串的形式传递到Function()中。这样这些代码就会在函数中保存，这些代码不会立即执行，只有在调用函数时才会执行。

var fun = new Function("alert('hello');alert('hello2');");

调用函数：

调用函数的本质就是 函数的引用 函数名()；当调用函数时，函数对象中保存的代码将会一起执行。

fun();

只要 fun() 被执行（函数被调用），函数就会被执行， fun() 的值是 函数的返回值，没有返回值则是undefined。

2.使用函数创建的表达式

function 函数名(形参){ 代码; }

function fun1(a,b){

alert("a");

alert("b");

}

调用函数方法：

fun1();

函数中可以指定形式参数。声明了形式参数以后，就相当于在函数中声明了两个变量，此时变量并没有赋值，调用时可以传递实际参数并赋值给两个变量。

调用函数时，函数不会检查实参的 类型 和 数量 。多余的实参不会使用。实参的个数小于形参的个数，未赋值的形参值默认为undefined。

函数可以接收任意类型的实参。实参可以是对象、函数。

当使用一个函数作为实参，传参时使用 函数名() 则传递的是该函数的返回值；如果是使用 函数名 则传递的是函数对象本身。

函数中可以使用return关键字来指定一个返回值。函数执行完毕可以使用一个变量去接收该返回值。

如果一个函数中没有返回值或return后面没有任何值，会默认返回undefined。

在函数中使用return，return后面的代码全都不执行。

如果直接打印 调用函数 函数对象() 则会输出 函数的返回值。

如果直接打印 函数名 函数对象 则会输出 函数对象本身。

function fun2(a,b){

//alert(a+b);

var result = a+b;

return result;

}

fun2(3,5);

3.匿名函数

var 函数名 = function(形参){ 语句; };

var sum = function(a,b){

alert(a+b)

};

sum(123,456);

### 38.1方法：

函数也可以作为一个对象的属性，并且也可以通过一个对象去调用函数，

当我们通过对象去调用一个函数时，我们就称这个函数是这个对象的方法，我们调用了某个对象的方法。

函数的返回值也可以是对象、函数。

### 38.2 执行环境 和 作用域

执行环境就是指js代码执行的区域。js代码要执行需要先有一个执行环境。

执行环境分两种，全局执行环境、函数执行环境。

**JS的执行顺序**：

**1.浏览器开辟空间作为js的执行环境：**

**2.创建一个变量对象。**这个变量对象用来保存js执行过程中所有的变量和函数。

除了全局执行环境的变量对象，变量对象都是隐藏的，不可见的。

全局执行环境的变量对象是window。

在执行环境中创建的所有的变量都是变量对象的属性。

所有的函数都是变量对象的方法。

**3.预解析**：浏览器不会立即执行代码，而是去代码中找 var 和 function 。

将var声明的变量在变量的对象中声明（创建为对象的属性），但并不赋值。

将function创建的函数直接创建为对象的方法。

**4.逐行执行代码**。

函数声明表达式不会再执行。

把没写var关键字的变量赋值语句设置成window对象的属性。

所有不写var的变量如果被使用，最终都会声明为window对象的属性保存，可以在全局执行中被访问。

函数声明表达式在第三步预解析阶段完成创建，所以可以在函数声明前调用函数表达式；

使用匿名函数的方式创建函数，函数会在第四步逐行执行代码时才赋值，所以不能在匿名函数声明前调用函数。

**函数执行环境**

当一个函数被调用时，浏览器会做如下四步操作。

**1.开辟一个函数执行环境**，（每次函数调用都会重新开辟一个函数执行环境）

**2.创建函数执行环境的变量对象**（这个变量对象不可见）

这个变量对象用于保存函数执行过程中的变量和函数

函数内部声明的变量会保存在函数执行环境的变量对象中，在全局环境中不能访问。

当一个函数执行时，函数中需要使用变量时，它会先去自身的变量对象中去寻找，如果没有则去上一级的执行环境中去找，如果没有找到则继续向上找，直到找到window对象中，如果仍然没有，则报错。

**3.预解析**：找var 和 function

形参就相当于var声明。

所有不写var的变量最终都会声明为window对象的属性保存，可以在全局执行中被访问。

**4.逐行执行代码**：

传递实参（变量赋值），

**5.函数执行完毕，销毁执行环境。**

**debugger；**

当浏览器执行到debugger时，会自动停止程序的运行，等待我们的指令。必须在浏览器中打开控制台。

### 38.3函数总结：

**函数是对象，函数名是指针**

函数是一个特殊的对象，它里面除了可以封装属性以外，还可以封装一些可以执行的代码。

将代码封装到函数对象中不会立即执行。他会在函数调用以后再执行。

函数的创建方式：

第一种：

var fun = new Function ("代码的字符串"); //不用

第二种：**函数声明表达式** ：

function 函数名(形参){

语句代码；

}

第三种：**匿名函数**

var sum = function(形参1,形参2){

语句；

};

：区别：第二种写法在 第三步预解析 时加载，第三种写法在 第四步逐步执行 时才加载。

调用函数时浏览器不会检查实参的类型和个数，只会根据函数名去调用。

调用函数的本质：函数的引用 +（）；

**匿名立即执行函数**：函数声明以后会马上执行，而且只执行一次。

( function(形参){ 语句;} ) (实参);

(function(){ 语句1； return function(){ 语句2 };} ) () ();

fun()();

先调用外层函数，返回值为内层函数，再调用内层函数。

**返回值：**

在函数执行完毕以后，可以通过return返回一个值，作为函数的执行结果。

return后面可以跟着一个任意的值：基本数据类型或对象、函数。

如果return后面没有任何内容或不写return 均会返回undefined

当函数中编写return以后，return后面的所有代码均不会执行。

函数也可以作为一个对象的属性：

当我们通过一个对象去调用一个函数时，我们称 调用了对象的一个方法。

全局函数时window的方法。可以使用 window.fun();来调用。

### 38.4执行环境总结：

JS引擎（浏览器） 在运行JS代码的时候，需要准备一个环境用来执行JS 代码，这个环境就是我们说的执行环境。

执行环境分为两种： 全局执行环境、函数执行环境。

浏览器执行代码的流程：

**1.开辟一块内存作为 执行环境。**

**2.创建一个"变量对象"用来保存程序执行过程当中的所有的变量。**

变量对象是不可访问的，除了全局执行环境变量对象window

程序运行过程中所有的变量都会作为 变量对象 的属性 保存；

而所有的函数都会作为 变量对象 的方法 保存。

我们在全局执行环境中的所有的变量和函数都会设置为window对象的属性和方法。

**3.预解析**

这时代码并不会由上至下执行，而是找 var 和 function（函数声明表达式）

所有的var 都会被设置为变量对象的属性，但是并不会赋值，也就是undefined

所有的函数声明表达式中的函数 都会被设置为变量对象的方法，并且赋值完毕。

函数声明表达式在第三步预解析执行的，第四步不会再执行了。

没写var声明的变量都会在第四步进行赋值，最终作为全局环境window的属性保存。也就是全局变量。

**4.逐行执行代码**

一行一行的执行程序代码，但是函数声明表达式不会再执行。

**5.代码执行完毕**

销毁执行环境（清理内存空间）

注意：全局执行环境不会在代码执行完毕后立即销毁，而是在浏览器关闭后才销毁。

函数执行环境和全局执行环境步骤是一样的，只不过它范围要小一些。

### 38.5、作用域、作用域链

作用域指的是我们在某个执行环境只能访问作用域以内的变量，而超出这个作用域的变量，我们将不能访问。

**1、作用域链：**

当在一个函数执行环境中使用一个变量时，它会先在自身的执行环境变量对象中寻找变量，如果找到了则执行使用，不再寻找，

如果没找到，则向上一级的变量对象中寻找，如果找到了直接使用，

如果没找到，则继续向上一级寻找，直到找到window对象为止，如果window中也没有，则报错。

这些函数作用域和全局作用域就构成一条作用域链。

函数作用域链不是在调用时决定的，而是在声明时决定的。

**2、闭包：**

闭包指的是内部声明的函数，这个函数可以访问到外部函数中的变量。

闭包是指有权访问另一个函数作用域中的变量的函数。

特点：当外部函数执行完毕以后，使用内部函数依然可以访问到外部函数中的变量。

**3、垃圾回收**

垃圾回收又称为 GC（garbage collection）

在程序的运行过程中也会产生一些垃圾，程序运行时间越长，垃圾越多，垃圾过多以后会大量的占用系统内存，影响程序的运行性能。

当对象在栈内存中没有任何变量引用它，那么这个对象我们将不能再次使用。这时这个对象在内存中依然占用空间，这种对象多了以后，内存会消耗很多，这种对象我们就称为垃圾。

这种垃圾我们需要采用一定手段进行回收，腾出内存空间以更好的运行程序。

但是垃圾回收的工作不用我们手动操作，浏览器会自动将这些垃圾回收。

浏览器会自动将没有被引用的对象进行回收。至于什么时间回收不一定。

我们要做的只是将这些不再使用的对象置空即可。obj = null;

### 38.6、this 函数执行的上下文对象

我们可以通过window对象来访问全局属性。 window.变量名

调用函数的对象是在函数调用时才知道的，浏览器知道函数的调用者。

在函数执行时，浏览器每次都传递了一个隐藏的参数，这个参数就是当前函数执行的上下文对象 this。

这个上下文对象就是当前函数的调用对象。

这个上下文对象this可以在函数中直接使用。

this 是当前调用函数的 对象。

this 是 对象 ===全等

1. 当以一个方法的形式调用一个函数，则谁调用的方法，this就是谁。

obj.fun() --> this 就是obj

1. 当以一个函数的形式调用一个函数，则调用者是window。

fun() --> this 就是 window ,相当于 window.fun()

1. 当以一个构造函数的形式调用一个函数，则this就是新创建的那个Object。

var per = new Person() --> this 就是 per

//1.能否使用this来显示fun中的 以var形式定义的 变量。

答：不能。因为代码只是封装在函数中，而不是这个函数的属性。

//2.函数中如何增加属性 。

答：最好在外面添加。

//3.怎么查看一个函数执行时的 变量对象。

答：除了window都不能查看

//4.将一个函数fun1赋值给fun的属性fun2, fun.fun2=fun1; 执行 fun.fun2(); 则fun1中的this是fun，this.name是fun。

【对象没有单独的作用域】 在Object对象中定义函数和在window（Object所在作用域）中定义是一样的。

如果一个函数fun1中包含另一个函数fun2，在fun1中调用fun2(),则fun2的this 是 window。如果想让fun2使用fun1的this，可以在fun1种创建一个变量var \_this=this ，把fun1的this值传递给fun2.

### 38.7、对象的创建

**1.工厂方法**

当遇到需要创建很多 结构、属性名、方法名 相同的对象时，需要编写大量重复代码。

我们可以将对象创建的代码封装到一个函数中，通过传递实参，然后使用函数来统一的创建对象。这种方式叫做工厂方法。

函数作用：创建指定格式的对象。

function fun(a,b,c){ var obj=new Object(); }

**2.使用构造函数**

我们希望每个类型的对象都有他自己的构造函数。

定义一个构建函数 （constructor）

定义构造函数和普通函数没有什么区别，只是构造函数的名字我们一般首字母大写。

使用构造函数必须使用new关键字。

每个对象中都有一个隐藏 属性 .constructor，它保存着创建这个对象的构造函数。

我们可以通过constructor属性来判断对象时通过哪个构造函数创建的。

当我们使用new关键字调用一个函数时，这个函数就会作为一个构造函数被使用。

一个函数作为构造函数调用时，它有以下几步：

1.创建一个新的对象。

2.并且将这个 新对象 作为我们执行的 上下文对象this 然后调用函数。

3.将新创建的对象返回（自动返回，return无效。）

当以一个构造函数的形式调用一个函数，则this就是新创建的那个Object。

function Person（a,b）{ this.name = a ; this.age = b; }

var obj = new Person(123,456);

如果不写new，则所有使用this设置的属性都会设置给window。

创建自定义的构造函数意味着可以将它的实例标识为一种特定的类型；而这正是构造函数模式胜过工厂模式的地方。

以这种方式定义的构造函数是定义在 Global 对象（在浏览器中是 window 对象）中的

构造函数与其他函数的唯一区别，就在于调用它们的方式不同。

任何函数，只要通过 new 操作符来调用，那它就可以作为构造函数；而

任何函数，如果不通过 new 操作符来调用，那它跟普通函数也不会有什么两样

**constructor属性**

每一个对象都有一个 constructor属性 ，用来保存它自己的构造函数。

**instanceof**

可以通过instanceof来判断一个对象是否属于某个构造函数的实例。 语法： 对象 instanceof 构造函数名。 是返回true，否返回false。

由于Object是所有对象的祖先，所以使用instanceof来判断 对象 instanceof Object 均返回true。

**in运算符**

- 使用in运算符来检查对象中是否包含某个属性。

- 语法："属性名" in 对象名。 如果对象中包含指定属性则返回true，否则返回false。

- alert（"age" in obj）;

### 38.8、类

使用相同构造函数创建的对象，我们称为为一类对象。

这样的每一个对象都叫做该构造函数的实例。

使用instanceof来检查一个对象是否由某个构造函数创建。

如果对象是由构造函数创建的，则返回true；

如果对象不是由构造函数创建的，则返回false。

语法： 对象名 instanceof 构造函数名 。

Object构造器是所有的对象的祖先。所有的对象都继承自Object。

任何对象和Object做 instanceof 检查都会返回true。

## 38.9、原型对象：prototype

**1. prototype**

我们创建的每一个函数对象（function），创建以后都有一个隐藏的属性：prototype。

这个属性指向了一个对象，这个对象就是 原型对象 。

每一个函数都有一个自己的原型对象，除非手动修改。

当函数作为普通的函数被调用时，它没有任何作用。它只会在作为构造函数调用时才会起作用。

**2. \_\_proto\_\_**

每一个通过该构造函数创建的对象中，都有一个隐藏属性\_\_proto\_\_,它指向构造函数的原型对象。

但这个隐藏属性在ES标准中没有提供访问的方式，但是火狐谷歌等浏览器为我们提供了访问方式：这个属性名 是 \_\_proto\_\_ 。

向构造函数的原型对象中添加属性：

构造函数名.prototype.添加的属性名 = 属性值;

当我们读取一个对象中的属性时，如果对象中有该属性，则使用对象中的属性，

如果对象中没有该属性，则会去它的原型对象中寻找属性，

如果原型对象中有该属性，则使用它，

如果原型对象中也没有，则返回undefined。

1. **obj.hasOwnProperty("属性名");**

使用 in 运算符检查一个对象中是否包含某个属性时，如果原型中有该属性也会返回true。

可以使用“原型对象的原型对象”中的方法 obj.hasOwnProperty("属性名"); 检查对象本身中是否含有某个属性，不会检查原型。

## 39、数组 Array

数组也是一个对象。

普通对象的属性名是字符串。而 数组也可以用来保存属性，它使用索引index来获取属性。

索引就是一个非负整数 0,1,2,3,4，。。。

数组存钱数据的性能比较高，所以存储较多数据时，我们一般使用数组。

var arr = new Array();

语法： 数组对象[索引] = 值 ;

打印数组对象时，默认将对象中的所有的元素都会输出

读取： 数组对象[索引]

如果读取的索引值超过数组的长度，则返回undefined

数组索引最大值是 大概是40多亿。

**length**

获取数组长度： 使用数组对象属性 length。 arr.length ;

一般情况下连续的数组的长度就是数组中元素的个数，但length属性的值实际上是最大的 index+1，此时length不能判断实际元素个数。

数组的长度可以修改 arr.length = 23; 当修改以后的长度小于原长度时，多余的部分会被删除，我们可以通过修改length的形式删除数组中的元素。

向最后一个元素后面添加一个元素： 可以使用 arr[arr.length]= 23;

创建数组：var arr = [元素1,元素2,元素3,元素4];

var arr = new Array(元素1,元素2,元素3,元素4);

var arr = new Array(长度); 当为Array传递一个整数型参数时，会变成长度。

JS中的数组可以保存任意数据类型，也可以是对象、函数、数组。

当数组也用来存放数组时，我们称这个数组是二维数组。

将数组中的每个元素都获取出来叫做 遍历 数组。

for(var i = 0; i<arr.length; i++){ console.log(arr[i]; }

**forEach()**

ESS标准中提供了一个forEach()方法，专门用来遍历数组。

forEach() 需要传递一个函数作为参数。

当forEach在遍历数组时，数组中有几个元素它就会调用函数几次。

每次调用都会遍历数组中的一个元素，然后通过参数的形式将遍历的信息传递给函数。

第一个参数是当前遍历到的元素，第二个参数是当前元素的索引，第三个参数 是数组对象本身。

forEach只支持IE9及以上的浏览器。

arr.forEach(function(value,index,arr){ console.log(value); });

由我们声明但是 不是由我们调用的函数叫做回调函数。

### 39.1、数组 的 常用 方法

**1.push**

使用push方法向数组的最后边添加元素,返回值是最新的length。

arr.push(元素，元素，元素);

**2.pop**

使用pop方法删除并返回数组元素的 最后一个元素

arr.pop();

**3.unshift**

使用unshift方法向数组的最前边按顺序添加元素,返回值是最新的length。

arr.unshift(元素，元素，元素）;

**4.shift**

使用shift方法删除并返回数组元素的 第一个元素

arr.shift();

使用shift和unshift会改变数组中原本数据的索引值

**5.slice**

使用slice方法从某个已有数组中返回选定个数的元素: 参数1：开始的index（包括） ；参数2：结束的index（不包括）；

如果只传一个参数，则返回从该位置开始直到数组结束的所有元素。

如果第二个参数是负数，则结束位置从数组的尾部开始计数。

slice 方法不会修改原数组，只是将取出的元素封装为一个新数组并返回。

arr.slice(2,4); 返回2、3位置的元素

**6.splice**

使用splice方法删除元素，并向数组的被删除元素位置添加新元素。参数1：删除的起始位置的index ；参数2：删除的个数；参数3开始均为添加的元素

splice方法是修改原数组。并将删除的元素返回。

splice删除元素后会自动调整index。

如果第二个参数为0则不会删除元素。

**7.concat**

concat方法连接两个或更多的数组，并返回。不会影响原数组。 参数可以写多个数组，也可以写一个要添加的值。

arr4 = arr.concat(arr1,arr2); 将arr1,arr2添加到arr后面，并返回结果给arr4

**8.join**

join方法 可以将数组中的所有元素拼接一个字符串并返回，不会改变原数组。

如果不传参数默认是逗号, 可以使用join传递的字符串参数，将所有的元素连接为一个字符串。

var str = arr.join();

**9.reverse**

reverse方法 反转数组中的数据顺序，会改变原数组。

arr.reverse();

**10.sort**

arr.sort();

sort方法 对数组中的数据进行排序，会改变原数组。

sort方法默认按照Unicode编码进行排序，所以对英文排序还凑合，但是对数字排序会出现问题。

也可以自己制定sort方法的排序规则，需要传递一个函数作为参数。这个函数是回调函数，由浏览器自己调用。

函数需要指定两个形参，这个函数会被浏览器多次调用，每次调用都会传递数组中的两个元素作为参数，通过该函数进行比较，函数需要返回一个值，如果返回值大于0则交换两个元素的位置，如果返回值小于等于0 则不交换两个元素的位置。

数字升序排序： arr.sort(function(a,b){ return a-b;}）；

数字降序排序： arr.sort(function(a,b){ return b-a;}）；

## 40、Math 对象

Math对象专门用来处理数学相关运算的对象。

Math对象不能作为一个构造函数调用。

Math这种对象，它里面封装了一些常用的属性和方法，这种对象我们成为工具类，首字母也大写。

Math里面封装了数学相关的常量、方法等。

对象中的常量我们一般全用大写的字母来表示。

**1、Math.PI 圆周率π。**

**2、Math.abs(num)用来获取一个数的绝对值。**

**3、ceil(num) 对小数进行上舍入**。只要小数点后有非零的数，就向上进1.

**4、floor(num) 对小数进行下舍入**。舍去小数位。

**5、round(num) 对小数进行四舍五入**。

**6、max(a,b,c,d)** 返回多个值中的**最大值**。

**7、min(a,b,c,d)** 返回多个值中的**最小值**。

**8、pow(a,n) 返回 a 的 n 次幂**。

**9、sqrt(a) 返回 a 的平方根**。

**10、random() 返回一个 0 - 1 之间的随机数**，不包括 0 和 1。

生成一个 0 - 10 之间的随机数(包括0和10)， Math.round(Math.random()\*10)

生成一个 0 - y 之间的随机数（包括0和y）， Math.round(Math.random()\*y)

生成一个 x - y 之间的随机数（包括x和y）， Math.round(Math.random()\*(y-x) + x)

生成一个0 - 6 之间的随机数（包括0和6），Math.floor(Math.random()\*(6+1));向下取整

生成一个x - y 直接的随机数（包括x和y），Math.floor(Math.random()\*(y+1-x)+x);

## 41、Date对象

Date对象在JS中用来表示一个时间

当创建一个对象时，它会自动将“对象创建的时间”封装到对象中,即代码执行时间。

var dt = new Date();

dt.getDate(); 几号：1-31

dt.getDay(); 周几：0-6 0是周日，6是周六。

dt.getMonth(); 月份： 0-11 ， 0是1月份，11是12月。

dt.getFullYear();年份： 4位年份

dt.getHours() 时

dt.getMinutes() 分

dt.getSeconds() 秒

dt.getTime(); 获取时间戳：从格林威治时间1970年1月1日0时0分0秒到现在的时间所花费的毫秒数。北京时间比格林威治时间多8小时。

通过传递一个日期的字符串来创建一个指定的date对象："月/日/年 小时:分钟:秒数"

var dt2 = new Date("7/18/2015 16:36:00")

获取现在的时间戳：

Date.now();

可以在一段代码前后分别获取一次时间戳，求差 可得这段代码执行所花费的时间。

dt2.toLocaleString(); 可以将时间转换为中国格式。不同浏览器格式不同

开启计时器:console.time("计时器名称");

停止计时器:console.timeEnd("计时器名称");

## 42、函数的补充

### 42.1、函数调用时的this

函数也是对象

以函数形式调用函数，this永远是window ：fun();

以方法形式调用函数，this永远是对象 ：obj.fun();

以构造函数形式调用函数时，this是新创建的对象

使用call和apply调用函数时，this是指定的参数。

### 42.2、函数对象中的两个方法： call apply

fun.call();

fun.call()方法可以传递一个对象作为参数 fun.call(obj) ，这个对象将会变成函数执行的上下文对象this

fun.call()第二个及其以后的参数，将会作为实参传递进函数里。

fun.call(obj,123,456); this 是obj;

obj.fun.call(window); this 是 window;

fun.apply(obj); 功能和call基本一致，只不过传递实参时需要使用一个数组。

fun.apply(obj,[123,345]);fun.apply(obj,arr);

### 42.3、函数的隐藏参数arguments

当一个函数被调用时，浏览器会隐式的传递两个参数，一个是this，一个是arguments

arguments 类数组对象，用来封装函数执行时的实参。即使不定义形参，也可以通过它来获取实参。

arguments[index];读取元素

arguments中有一个属性callee，保存着当前函数的引用。

arguments.callee == fun

递归调用就是在函数里调用函数本身。JS中不推荐使用，容易导致内存溢出。

## 43、包装类

JS中的数据分两大种：基本数据类型，引用数据类型。

基本数据类型就是一个简单的值，不能添加属性和方法。

引用数据类型是一个符合数据类型，里面可以添加属性和方法。

在JS中提供了三个包装类：

String() ,Number(), Boolean()

它们可以用来创建基本数据类型的对象，

当我们调用一个的基本数据类型的方法时或者向基本数据类型中添加属性时，

浏览器会默认将这些基本数据类型包装为对象，然后再调用方法。方法调用完成以后再转换为基本数据类型。

## 44、String 对象

String对象中为我们提供了很多字符串相关的方法，由于基本数据类型会自动转型为对象，这些属性和方法每一个字符串都可以直接调用。

JS底层处理字符串时是将其转换为一个字符数组来处理的。

var str = "一个任意字符串";

**str.length**;获取字符串长度

**str[index]**;通过索引来查看某个字符。

**str.charAt(index)**;根据索引获取字符串中指定位置的字符。

**str.charCodeAt(index)**;根据索引获取字符串中指定位置的字符的Unicode编码。

**String.fromCharCode(字符编码);**将字符编码转换为字符。

**str.cancat(str1,str2,"字符串");**连接多个字符串，并返回。【不会影响原字符串】

**str.indexOf("A",0);**返回某个字符在字符串中首次出现的位置。（从前往后找） 参数1：要查找的字符、字符串；参数2：在字符串中查找的起始位置index

**str.lastIndexOf("A",0);**返回某个字符在字符串中最后一次出现的位置。（从后往前找）参数1：要查找的字符、字符串；参数2：在字符串中查找的起始位置index

如果找到了返回索引，如果没找到，返回 -1；

**str.slice(startIndex,end);**截取字符串并返回。end可以是负值

**str.subString(startIndex，endIndex);**用法同slice，只是不能接收负值。如果设置负值，则默认为0。两个值可以调换位置。

**str.split("分割b标识")**;把字符串分割成字符串数组，参数：根据哪个字符去拆分，该字符会被删除，如果参数是"",则每个字符作为一个元素。

## 45、异常处理

一旦程序出现异常，异常以后的代码都将会终止运行。

我们可以将可能出现异常的代码放到 try{ } 语句中，它将会尝试执行该代码，如果可以执行则执行，如果不能执行就不执行。

如果代码执行时出现异常，则执行catch语句后的内容，来对异常进行捕获。

try{

//有可能出现异常的代码;

}catch(e){

//出现异常以后处理的语句

console.log(e);

}

## 46、正则表达式 RegExp对象

对用户输入的内容进行检查

正则表达式就是用来指定一些规则，计算机可以根据这些规则来识别字符串。可以识别字符串是否符合规则，也可以将规则的内容提取出来。

在JS中使用RegExp()来创建一个正则表达式。

RegExp()构造函数中需要 传递一个字符串，这个字符串就是我们的正则表达式。

使用正则表达式来检查一个 字符串 是否符合规则。

使用正则表达式的test()方法来检查一个字符串，参数是要检查的字符串。

如果符合要求返回true，否则返回false。

正则表达式中严格区分大小写。

正则表达式可以传递 第二个字符串参数，用来指定正则表达式的匹配模式，一般有两个值可=可选， i忽略大小写，g全局匹配模式。

1.使用构造函数创建正则表达式

var reg = new RegExp("正则表达式","匹配模式");

2. var reg =/正则表达式 / 匹配模式 ;

reg.test("要检查的字符串");

第一种方法更灵活，我们可以根据需要随时修改。第二种编写方便，但是不便于修改。

检查是否包含指定字符串,并忽略大小写： var reg = /abc/i;

检查是否包含指定的字符中的某一个： var reg = /a|b|c/; | 表示或者

检查是否包含指定的字符中的某一个： var reg = /[abc]/; [] 中的字符都是或的关系

检查是否包含任意的小写字母： var reg = /[a-z]/; [a-z]所有的小写字母； [a-zA-Z]所有大写小写；[A-z]所有大写和小写

[^a-z]除了小写字母。

/ a[bdf]c / abc或adc或afc

console.log(reg.test("hjlacliu"))

**string的方法**

**serch()**可以用来搜索一个字符串的内容个，如果有则返回内容的索引，如果没有则返回-1；可以使用一个正则表达式作为参数。只能搜索第一个，不支持全局匹配。

str.search(/a/);

**match()**可以从一个字符串中找到符合要求的内容,默认只找到并返回第一个符合要求的内容。可以使用g参数全局匹配，这样可以找到并返回所有的符合要求的内容。返回值是个数组。

str.match(/a[bcd]e/);

**replace();**可以用来替换一个字符串中的内容，参数1：被替换的内容；参数2：新的内容；返回值是新的字符串。默认只替换第一个。

str.replace(/a/gi,"新内容")

**split();**可以将一个字符串拆分为一个数组并返回。参数1：分割位置；参数2：最大数组length（可不写）。参数可以是正则表达式。默认全局拆分。不支持全局匹配。

str.split();

**正则表达式的量词：**

在正则表达式中使用量词来描述一个内容的数量，量词使用{}来表示。量词只对大括号前面的一个字符起作用。可以使用（）对字符分组。

/a{3}/ 3个连续的a.

/(ab){3}/ 3个ab.

/a{3,5}/ 3到5个a;

/a{3,}/ 最少3个，最多不限

/^a/ 以a开头

/a$/ 以a结束

/^a$/ 只有一个a

/^a{3,5}$/ 只能有3-5个a.

/a+/ 至少有一个a,相当于{1，}

/a\*/ 任意个a，相当于{0，}；有没有均可

/a?/ 0个或1个 a,相当于{0,1}

\在正则表达式中是转义字符串

\w 任意的字母或数字

\W 除了字母和数字

\d 任意数字

\D 除了数字

\s 空格

\S 除了空格

\b 匹配单词边界 / \bchild\b / 前后均加\b才是一个单词

\B 除了单词边界

. 点 是任意单个字符，除了空格和换行

对特殊字符需要转义。如小数点应写为 \.

\ 在字符串中是转义， reg = new RegExp("\\.");

当使用构造函数去创建一个正则表达式时，需要使用两个 \\ 来代替一个 \ 。

去除字符串中前后两边的空格 /^\s\*|\s\*$/g

## 47、DOM DocumentObjectModel文档对象模型

JS中通过DOM来对HTML文档进行操作。

HTML中一切都是对象。一切都是节点。

节点：Node 网页的最基本的组成部分，网页是由一个一个节点构成的。

节点分类：

1.文档节点：整个HTML文档

2.元素节点：HTML标签

3.属性节点：元素的属性

4.文本节点：标签中的文字

节点属性：nodeName，nodeType，nodeValue

document对象代表整个网页的文档。它是window对象的属性。我们可以在页面中直接使用。

**getElementById()** 方法返回带有指定 ID 的元素：

**document.getElementById("id");**  通过id属性来获取元素节点对象。

### 47.1、事件：Event

事件Event:

文档或浏览器窗口中发生的一些特定的交互瞬间。用户和浏览器之间的交互行为。

响应：

我们需要为事件绑定一个处理函数，当事件触发时，处理函数便会执行。

为元素的指定的事件属性来设置代码：

1.获取指定ID对象： var btn = document.getElementById("btn");

2.为对象事件绑定一个响应函数： btn.onclick = function(){};

为对象的指定 事件属性 绑定一个 回调函数，这个函数将会在指定事件被触发时才会执行。

为单击事件绑定的处理函数，我们称为单击响应函数。

浏览器在加载网页时是从上到下的顺序加载代码。

为window对象绑定一个onload事件，window的onload对应的函数，将会在整个页面加载完成之后执行。

在头部编写js代码时，应将代码编写到 window.onload =function(){ };中。

Event属性：

onload事件：一个页面或一张图片加载完成之后触发响应。

通过innerHTML属性可以修改 “元素内部” 的 “HTML代码” 。

innerHTML会获取到标签内部的所有内容，包括标签。

innerHTML对自结束标签没有意义。

innerText会获取到标签内部所有的文字内容，自动去除html标签。

读取元素的属性: 语法： 对象.属性名

class属性是JS中的保留字，不能直接使用，使用className代替。

获取元素节点的 document方法：

1. document.**getElementById()**;通过id属性获取一个元素节点对象

2. document.**getElementsByTagName();**通过标签名获取一组元素节点对象

3. document.**getElementsByName();**通过name属性获取一组元素节点对象（常用于表单项）

4. **document.body;** 获取body元素

5. **document.all;** 获取所有的元素节点，相当于document.getElementsByTagName("\*");

获取元素节点的子节点：

1.**getElementsByTagName();** 元素方法，返回当前节点下的所有指定标签名的子节点（后代元素）

2.**childNodes** ：属性 ，获取当前节点的所有子节点。 根据DOM标准，空白内容也会当成节点。IE8以下的浏览器不会将空白内容当成节点。

3.**children** ：属性 ， 获取当前节点的所有元素子节点

4**.firstChild** ：属性 ，获取当前节点的第一个子节点。 会获取到空白的文本节点。

5.**firstElementChild** ：属性，获取当前节点的第一个元素子节点。不会获取空白的文本节点。IE8以下浏览器不支持该属性。

6.**lastChild** ：属性 ， 获取当前节点的最后一个子节点。

7.**lastElementChild** ：属性 ，获取当前节点的最后一个元素子节点。

获取父节点和兄弟节点

1.**parentNode**： 属性，当前节点的父节点

2.**previousSibling**：属性，当前节点的前一个兄弟节点

3.**nextSibling**:属性，当前节点的后一个兄弟节点

DOM查询方法：

### 47.2 自定义一个通用的getElementsByClassName

**document.getElementsByClassName("className");**根据className属性值获取一组元素。**【IE8及以下浏览器不支持】**

自定义一个通用的getElementsByClassName：

function getElesByClass(className){

//创建一个数组保存获取到的元素

var arr = [];

//获取所有的元素

car allEle = document.all;

//从所有的元素中找到拥有指定class属性值的元素

//遍历allEle

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

//判断每个className是否等于参数值

//定义一个正则表达式,匹配className，并有单词边界

var reg = new RegExp("\\b"+className+"\\b");

if(reg.test(allEle[i].className)){

//将指定className的元素放到arr数组中。

arr.push(allEle[i]);

}

}

//返回数组

return arr;

}

querySelector可以通过CSS选择器来查询单个元素【只支持IE8以上的浏览器】

querySelector如果出现多个符合条件的元素，只返回第一个元素。

**document.querySelector("选择器字符串");**

document.querySelectorAll可以查找到所有符合条件的元素。返回的是一个数组。

**document.querySelectorAll("选择器字符串");**

DOM方法：

**document.createElement("标签名");** 创建一个元素节点对象。

**document.createTextNode("文本内容字符串")**;创建一个文本节点对象

**父节点.appendChild(子节点对象)**;向父节点中添加一个子节点。

**父节点.insertBefore(新子节点对象，老子节点对象);**在指定的子节点前面插入新的子节点。

**父节点.replaceChild(新子节点对象，老子节点对象);**替换指定的子节点。

**父节点.removeChild(子节点对象);**删除指定的子节点。

**子节点.parentNode.removeChild(子节点);**删除指定的子节点。

**父元素.innerHTML +="<li>添加的元素</li>";**在父元素后面添加一个子元素。

取消超链接的默认跳转页面行为：在超链接的onclick函数的最后添加 return false；

**document.confirm("确认XXX吗？");**confirm可以弹出一个带有确认和取消按钮的提示框。 返回值：确认-true；取消-false。

可以在a标签的href属性中添加JavaScript代码取消a标签的跳转行为。

<a href="javascript:;">不会跳转的连接</a> <a href="javascript:void(0);">不会跳转的连接</a>

### 47.3、通过JS操作元素的css样式

1.元素中有个属性： style ，所有的样式都可以通过该属性设置。

修改元素样式： 元素对象.style.样式名 = "样式值"；

像background-color这样中间包含 - 的样式名不能直接使用，应该转换为驼峰命名法，转换为 backgroundColor；

使用style设置的样式是内联样式，而内联样式的优先级最高，所以我们通过js设置的样式都会优先显示。除非样式中设置了 “!important”。

2.读取元素的样式：

元素.style.样式名。

但是这种方式只能读取内联样式。

3.读取CSS样式表

元素.currentStyle.样式名

可以获取元素当前的样式，包括样式表中的样式。但是currentStyle只能读取属性，不能修改。【这个属性只有IE支持】

正常浏览器支持的方式：

getComputedStyle();

这个方法是window的方法，可以直接使用。【IE8及以下不支持】

这个方法需要两个参数，参数1：要查找的元素 ； 参数2：可以设置一个伪元素，一般写成null。

它会将样式对象返回。然后 该对象.样式名。

使用这种方式获取没有设置的样式，将得到默认值。

4.自定义一个方法用来获取元素中指定的样式。

//参数：元素对象，样式字符串。

function getStyle(obj,str){

return window.getComputedStyle?getComputedStyle(obj,null)[str]:obj.currentStyle[str];

/\*try{

//支持正常的浏览器

return getComputedStyle(obj,null)[str];

}catch(e){

//如果出现异常，证明不支持getComputedStyle方法，则使用currentStyle。

return obj.currentStyle[str];

}\*/

/\*if(window.getComputedStyle){

return getComputedStyle(obj,null)[str];

}else{

return obj.currentStyle[str];

}\*/

}

5.查看元素的可见高度、可见宽度

**clientWidth**可以返回元素的可见宽度，包括width和padding，不包括border。

元素.clientWidth；

**clientHeight**可以返回元素的可见宽度，包括height和padding，不包括border。

元素.clientHeight；

这两个值是数字，不带px，可以直接参与计算。但这些值是只读的，不可修改。

6.返回元素实际可见宽度高度，包括width/height、padding、border.

**元素.offsetWidth;**

**元素.offsetHeight;**

值是数字，不带px。

7.查看元素的定位元素。离当前元素最近的开启定位的祖先元素。

**元素.offsetParent；**

8.查看定位偏移量。相对其定位元素的偏移量。

**元素.offsetLeft;**

**元素.offsetTop;**

9.

**元素.scrollHeight;**获取到包含滚动条可滚动高度在内的总高度。

**元素.scrollWidth;**获取到包含滚动条可滚动宽度在内的总宽度。

**元素.scrollTop;**获取滚动条垂直滚动距离（垂直偏移量）。

**元素.scrollLeft;**获取滚动条水平滚动距离（水平偏移量）。

csrollHeight - scrollTop == clientHeight;

如果该等式为true，则证明滚动条已经滚动到头。

3、表单项的disabled属性

disabled = "disabled" 可以禁用一个表单项。

使用JS禁用： 元素.disabled = true ；

使用JS解禁： 元素.disabled = false ；

4、滚动条滚动事件

获取含有滚动条的元素，为该元素绑定滚动事件。

像onscroll这种事件，它会频繁触发，所以尽量避免使用，如果要使用，事件的响应函数中尽量不要写性能太差的代码。

元素.onscroll = function(){

//判断滚动条是否滚到底了。

if(this.scrollHeight - this.scrollTop === this.clientHeight){ };

};

}

5、事件对象

获取鼠标的坐标

element.onmousemove ；鼠标移动事件，当鼠标在元素上移动时触发。

1.鼠标相对于可见区的坐标：

x坐标：event.clientX; y坐标： event.clientY;

2.鼠标相对于当前页面的坐标【IE8及以下浏览器不支持】

event.pageX ; event.pageY;

3.document.body.scrollTop 只能获取到chrome的滚动条的高度。

4.document.documentElement 代表整个页面的根元素 <html>

document.documentElement.scrollTop 只能获取到火狐和IE的滚动条高度。

5.鼠标滚动条兼容解决：

var scrolltop = document.documentElement.scrollTop || document.body.scrollTop ;

在JS中每一个事件在触发时，浏览器都会调用我们为事件绑定的响应函数。

浏览器每次调用响应函数时都会传递一个事件对象作为参数，这个事件对象中封装了当前事件相关的一些信息，

比如：鼠标的坐标、键盘的哪个按键被按下等。

我们可以给响应函数设置一个形参event，来获取该对象。

则 如鼠标事件 [object MouseEvent] x坐标：event.clientX; y坐标： event.clientY;

【IE8及以下的浏览器中它不会传递该事件对象】在IE8中该事件对象是作为window对象属性保存。

在IE中可以使用window.event获取。但是火狐中不支持该属性。

兼容性处理： 使event等于第一个true。

event = event || window.event;

6、冒泡

冒泡指的是后代元素上的事件被触发时，其祖先元素上的相同事件也会同时触发。事件的向上传递。

有的时候需要利用冒泡，有的时候需要禁止冒泡。

利用事件对象取消冒泡：

event = event || window.event;

当将event.cancelBubble属性设置为true时，事件便会停止向上传递。这个属性是IE首先使用的，所有浏览器都支持。

event.cancelBubble = true;

鼠标移入事件: element.onmouseenter

鼠标移出事件：element.onmouseleave

7、为同一个元素的同一个事件绑定多个响应函数：

监听器：

监听对象：

监听事件：

响应函数：

element.addEventListener(); 可以为一个元素绑定多个响应函数【不支持IE8及以下】

参数1：监听事件的字符串；如："click" ，不用写on；

参数2：响应函数；

参数3：布尔值：是否在捕获阶段触发事件。一般都用false。

使用addEventListener()每次可以为一个元素的同一个事件绑定一个响应函数；绑定多个需要写多次。先绑的定先执行。

[在IE5-10中]

element.attachEvent();【只支持IE5-10】

参数1：事件的字符串，如"onclick",需要写on

参数2：响应函数。

element.attachEvent();绑定的响应函数是 后绑定的先执行；

element.attachEvent();绑定的响应函数中this是window！！。

兼容性解决方法：

定义一个函数，解决这两个方法的兼容性：：

bind();

同时解决this不统一的问题：

function bind(obj,eventStr,callBack){ };

需要三个参数：要绑定的对象，要绑定的事件字符串，回调函数。

call和apply可以解决this不统一问题；

使用判断浏览器是否支持该方法来解决兼容性。

function bind(ele,eventStr,callBack){

function fun(){

// 设置原函数的this，当事件被触发时，调用fun函数，从而间接调用了原函数的call方法 指定this。

callBack.call(ele);

}

ele.addEventListener?ele.addEventListener(eventStr,fun,false) : ele.attachEvent("on"+eventStr,fun);

/\*if(ele.addEventListener){

ele.addEventListener(eventStr,fun,false);

// alert("first");//这句可以测试浏览器调用了哪个方法

}else{

ele.attachEvent("on"+eventStr,fun);

// alert("second");

}\*/

}