笔 记

-------------------------------**北京训练营**--------------------------------

**行业介绍：**

**1、什么是 WEB前端----WEB前端开发工程师**

a、软件：设计、开发、测试、运维

----软件：受限于操作系统等相关影响

b、开发：细分多种技术

c、从应用的角度来细分

C/S类型：桌面应用程序、exe--单机

B/S类型：browser浏览器 server---WEB类型的应用程序

硬件相关：嵌入式

移动端开发

d、WEB前端究竟做什么？

一、web程序的运行过程

浏览器--url--浏览器发送请求到服务器

---服务器端返回响应（页面代码）---浏览器负责运行显示代码

二、WEB程序分前后

运行在客户端浏览器里

---WEB前端代码---决定了用户体验度

运行在服务器端

---后端代码---决定的数据和业务的正确

三、早期 WEB程序开发--重点在后端

各种后端开发技术

JAVA、PHP、.NET

四、由于互联网行业的发展，体验度高

----WEB前端代码变得很重要

页面

页面的动效

页面的功能

和服务器端的交互

**2、移动端开发**

a、开发一个大型的 WEB 应用：三个版本

---开发成本高

b、flash---封杀PC端

移动端

c、H5-----隶属于 WEB前端技术范围

d、web前端技术范围

HTML---写页面

CSS---设置样式外观

Javascript----编写页面功能

经过 W3C 规范

H5：google 和 apple

页面复杂动效、绘图、地理定位。。

web前端技术可以应用于移动端

e、移动端APP

原生APP(**Native App**)：Android和IOS

轻量级 WEB APP：web前端技术，不能和底层交互

混合开发(**Hybrid App)**：使用 js 调用底层接口

**3、就业**

设计型WEB前端：会web前端技术、WEB UI（只WEB相关的设计）

开发型WEB前端：会web前端技术、了解服务器端，和服务器端交互

大前端：前后端均会

**课程介绍：**

1、4个月，三个月相同（HTML/CSS/JS/H5/Jquery/WEB APP/Bootstrap/weixin）

设计型: 1个月的 web UI设计及实现

开发型：一些服务器端相关内容，服务器交互，和服务器交互的js框架

2、因材施教、分级教学

设计型：难度稍低，进度稍慢

开发型：难度稍高，进度稍快

3、学1个月

4、学习安排

训练营课程

1、HTML/CSS/JS

2、正式课程中：JS框架/库、CSS库

**HTML 课程**

**一：HTML 概述**

**1、什么是 html( hyperlink text markup language)**

超文本标记语言---网页的编写语言

一种解释性的纯文本标记语言，用于编写网页

-----解释性，相对于“编译性”而言

如果代码有错误，显示效果不确定

**2、编写**

创建代码文件：改后缀（扩展名）

编写代码：编辑工具打开它，写内容，保存

运行代码：使用浏览器打开它

**3、浏览器：运行网页及网页上的程序**

主流浏览器：IE、firefox、chrome、safari、opera

**4、标记语法：不同的标记带来不同的效果**

a、标记分为：

封闭类型的标记，也叫双标记

<b>aaaa</b>

<h1>ccc</h1>

-----标记必须成对

非封闭类型的标记，也叫单标记

<br /> <br>

b、标记可以有属性：为标记添加一些额外的描述

<h1 align="center">ssss</h1>

属性定义在开始标记里

属性和标记名称间用空格隔开

属性名 = "值" 属性名 = '值'

多个属性之间用空格隔开

c、标记可以嵌套：注意嵌套顺序

d、不区分大小写：为了阅读方便，建议使用小写

文档类型声明除外

**5、html文档的标准结构**

<!DOCTYPE HTML> 文档类型声明

<html> 整个文档

<head></head> 头信息：文档相关的整体定义

<body></body> 文档主体：页面显示的内容

</html>

**二、head里面的标记（元素）**

**1、title：为网页定义标题，显示在浏览器顶端**

**2、meta：元数据，定义各种相关信息，取决于属性**

<meta name="keywords" content="html,css" />

----为网页声明关键字，提高网页给搜索到的概率

<meta name="description" content="html,css" />

----为网页声明描述

<meta http-equiv="refresh" content="10" />

----为网页定义刷新频率

<meta http-equiv="content-type"

content="text/html;charset=utf-8"/>

----为网页定义内容的类型：文本类型的html文件

且字符集为 utf-8

<meta charset="utf-8" />

----为网页编码格式

**3、link：为网页引入各种资源文件**

**4、script：为网页引入脚本代码，比如 js**

**三、文本标记**

**1、页面上的普通文本：**直接显示，空格折叠的效果（多个换行、多个空格，折叠 为一个空格）

**2、特殊文本**

已经被系统征用，赋予其特定意义

键盘上无法录入

----用 字符实体 代替它

**3、注释：**为代码添加一些说明，增加代码的可阅读性

<!-- 文本描述 -->

**4、文本标题**

<h1>。。。<h6>---独占一行、加粗、段落间距

**5、段落**

<p></p>---自成一段落：独占一行、段落间距

**6、换行**

<br />---插入回车的效果

**7、文本样式**

<b></b>---粗体的效果 bold

<i></i>---斜体 italic

<u></u>---下划线 underline

**8、为标记分类：**

块级元素（block）：独占一行，有高度、有宽度

h1>p

行内元素(inline)：可以和其他行内元素位于同一行，没有尺寸

b、i、u

行内块级元素：可以和其他行内元素位于同一行，有大小

img

**9、<span>元素：**

行内元素，标记出来一行中的某些文本，用于做动态的修改、以及额外的样式 定义

**10、<div>元素**

块级元素，用于实现页面的基本构成块，实现页面元素的分组

**11、嵌套规则：**大的包含小的

块级元素包含行内元素

块级元素：div用最外层

**四、图像和链接**

**1、URL---用于 WEB程序，资源位于服务器，浏览器访问时候，需要把资源下载 存放缓存**

相对路径：资源相对于当前文件来存放

绝对路径：单独存放

完整的 URL：统一资源定位器

协议名称 主机名称 路径 文件名

http:// www.jd.com /images /a.jpg

smtp://

ftp://

telnet

**2、<img src="url" />**

<img src="d:/a.jpg" />---error

<img src="a.jpg" />---相对路径

<img src="http://www.jd.com/images/a.jpg" />

---绝对路径，用于资源文件单独存放的情况

**3、超级链接**

<a href="url" target="\_blank">被点击的</a>

**五、列表**

**1、有序列表 ol：ordered list**

<ol>

<li>aaa</li> ---list item

<li>ccc</li>

<li>bbb</li>

</ol>

**2、无序列表**

<ul>

<li><a>aaa</a></li> ---list item

<li><a><img /></a></li>

<li>bbb</li>

</ul>

------块级元素：用于列表的直接使用，用于一系列相同选项的组件，实现树 形结构

**3、列表嵌套：列表中嵌套列表元素---实现多级菜单和导航**

ul和ol：只能直接包含 li 元素

li元素还可以包含其他元素

-----相对于大量重复使用 div 元素而言，使用列表元素构建一些结构，提高 了代码的可阅读性

**六、结构**

**1、相对于原有的那种大量使用 div 元素的方式：标签重复，id和class 复杂---- 代码难以维护**

**2、建议尽量使用 语义化 的标签构建页面**

header：表示页头部分

nav： navigator的简写，表示导航部分

section：表示组成部分

aside：边栏、侧栏

footer：页脚

article：文章

**七、表格**

**八、表单**

**CSS**

**一、概述**

**1、在 html标记，也定义了一些和样式相关的属性**，比如 align、bgcolor、width、 height...

隐含的问题：不通用----用 CSS 来解决

**2、CSS 的作用：**用统一的方式为页面元素定义样式，提供优秀的样式解决方案

**3、CSS 的写法**

a、内联方式：将 CSS 代码写在 元素的 style 属性里---仅限于当前元素可 用，无法重用

b、外部样式表

将 CSS 代码写在单独的 .css 的文件里

页面在 head 里 用 link 引入

-----可以实现内容和表现的分离（HTML代码和CSS代码分开写），提 高代码的可重用度和可维护性

**4、CSS 层叠样式表**

用继承、集合、优先级，决定样式

继承：子元素可以继承使用父元素的某些样式

集合：一个元素使用多次样式定义，不重复的地方，均起效

优先级：如果重复定义，按照优先级

---就近优先（影响的范围越小，优先级越高）

优先级一样时，按照顺序---后定义的优先

**二、CSS 的语法**

**1、最基本的构成** 样式名称：样式值；

**2、样式表**

选择器 {

样式名称：样式值；

样式名称：样式值；

}

**3、英文符号、不区分大小写（样式代码）**

对于自定义的名称（id），严格区分

-----建议统一使用小写

**4、CSS 代码添加注释**

/\* 注释的文本内容 \*/

**三、样式选择器**

**1、元素选择器：用 标签 的名称，作为选择器的名称**

p{ } h1{ }

------为某一种元素定义统一的样式：范围广

**2、ID 选择器：用元素的 id 属性的值，作为选择器的名称**

HTML文件： <p id="p2" ></p>

CSS文件： #p2 { }

-----id 的取值：在当前页面上唯一，子母和数字的组合但是不能以数字开头 ----为某一个

**3、类选择器：用元素的 class 属性的值，作为选择器的名称**

HTML 文件： <元素 class="name" />

CSS文件： .name { }

-----适合于某种范围内的元素

**4、后代选择器：以元素的层次关系作为选择器，综合运用前面的那几种选择器**

p span {color:red;}---被p元素包含的span（子元素、孙子元素）

#d1 span { }

.s1 span { }

ul li a { }

**5、伪类选择器：根据元素的各种细节来定义样式**

结构伪类

状态伪类：结合其他选择器，选中某些/个元素的某种状态

:link ----未访问过的链接--a

:hover ----悬停，鼠标移入--适用于所有元素

:active ----激活的--单击按下，比如a、按钮

:visited ----访问过的链接--a

:focus ----获得焦点，常用于文本框，可以录入数据

a:hover

#p1:hover

.s1:hover

#p1 a:hover

link 和 visitied 状态：取决于 URL 是否被成功访问过

**6、选择器分组：将各种选择器作为一组，统一定义他们相同的部分**

p,#a1,.s1,div { }

p #a1,.s1,div { }

**四：样式的单位**

**1、尺寸相关的单位**

相对度量

px：像素（屏幕上的一个点）

%、em....

绝对度量

pt：磅

cm、mm、in。。。

**2、颜色**

color:red;

color:#ffffff;---代表三个颜色的色值，使用十六进制的单位

color:#fff;---白色

#000黑色

某种灰色 #333 #666---文字

#ccc #ddd---边框、背景色

**五、尺寸和边框**

**1、尺寸**

width:;

height:;

<img width="200" />----html的属性

<img class="m1" />

.m1{width:200px;} ----CSS

**2、边框：表示出来元素的边界**

a、四个边

border:1px solid red;---简写方式，优先建议

宽度 线型 颜色

border-width:10px;

border-style:dotted;

border-color:red;

b、单边定义

border-left:2px solid black;

border-right/top/bottom;

c、边框倒圆角

border-radius:5px;

**3、是否所有的元素都可以设置边框：是**

**4、是否所有的元素都可以设置尺寸**

行内元素：尺寸(width和height)无效，尺寸由内容决定---其他行内元素位 于同一行

块级元素

行内块级元素：img

**六、框模型（box modal）：解决的就是边距问题**

**1、外边距：自己和周围其他元素（邻居、父元素）之间的距离，注意不要造成元 素的互相重叠**

a、margin:10px;---10像素的外边距，四个方向

b、margin-left/top/bottom/right:;---单方向设置

c、margin:10px 20px 30px 40px; ----顺时针

上 右 下 左

d、margin：10px 20px;

上下 左右

e、margin:0px auto;

---auto：自动，由浏览器计算元素的左右外边距（可以实现元素在当前 父元素的范围里居中显示）

问：是否所有的元素都可以设置外边距？

对于行内元素而言：左右外边距有效，上下外边距无效

既然行内元素的左右外边距有效，问，是否可以设置为 auto，实现居中？ ---不能

**2、内边距：自己和子元素之间的距离**

padding:10px;

padding:10px 20px 30px 40px;

padding:10px 20px;

a、设置了元素的内边距后，会导致元素框撑开变大

-----width和height：设置的是内容区域的大小

默认情况下，边框和内容区域之间没有距离

如果设置了内边距（边框和内容之间的距离），内容大小不变，导致框撑 开

b、实际使用时

计算元素的占地面积的时候：外边距 + 边框 + 内边距 +　ｗｉｄｔｈ ／ｈｅｉｇｈｔ

问：是否对于所有的元素都可以设置内边距：可以设置

可能会导致元素边框撑开，导致堆叠

**七、浮动**

**1、页面默认布局方式：流模式布局**（块级元素从上向下，一行里，行内元素从左 向右）---位于同一层

**2、需要复杂布局的时候：改变默认的方式**

---浮动、相对定位、决定定位、固定定位

**3、浮动定位**

float:left/right;

----元素会脱离原有的层，停靠在包含框或者其他浮动元素的左侧或者右侧， 而该元素原有的空间不再保留

------适合于将多个块级元素位于同一行

**4、清除浮动**

clear:left/right/both;

设置了某元素浮动后，会影响后续的元素

如果不希望受到影响（不补位）

**八、背景**

**1、background-color：；设置背景色，纯色**

**2、background-image:url("images/a b.jpg");**

**3、如果框大图小，就会出现平铺的效果**

background-repeat:repeat/repeat-x/repeat-y/no-repeat;

repeat-x/repeat-y：经常用于实现渐变色

no-repeat:;---出现一次，就会有位置的问题

**4、background-position:x y;**---设置背景图片的位置

10px 20px

10% 50%

left center

a、图像小，且出现一次：正值

b、图像大，只需要显示背景图像中的某一块：负值

**5、框小图大**

background-size:50%;

**6、background-attachment:fixed；**

---可以设置背景图片固定，不随着页面滚动，实现类似于水印的效果

**7、应用技巧**

设置背景图片的同时，设置背景色--色和图尽量接近

**九、文本**

**1、字体相关**

font-size:10pt;---文本的大小

font-family:XXXX，aaa，bbb，ccc；

---字型，设置多个，替换字型，尽量使用同一个 字型系列

font-weight:normal/bold;---加粗

font-style:italic; ---倾斜

letter-spacing ; -letter-spacing 属性增加或减少字符间的空白（字符间距）。

**2、**

color:;---文本的颜色

text-decoration:none/underline;---是否有下划线

text-align:left/right/center;---文本在水平方向上排列对齐（对于块级元素）

<p align="center"></p>---HTML，不要使用

<p style="text-align:center;"></p>---CSS

line-height:20px;---行高，设置文本所在行的高度，文本的中心和行的中心 重合，如果文本比较小，可以实现文本在当前行，垂直方向上居中显示

--------所有浏览器默认加入的样式，都可以被 CSS 代码改写（CSS RESET）

margin:0px auto;---元素本身在父元素的范围里居中，仅对于块级元素

text-align:center;---元素里的内容在元素本身的范围里居中

**十、列表**---经常用于一个元素下面包含多个子元素的结构

list-style-type:none/circle/disc/...;

**十一：定位**

**1、页面默认情况下，使用的 流模式 布局**

margin：可以通过设置外边距，改变元素的位置，同时会影响其他元素的位 置----所有的元素默认都在同一层

**2、复杂情况下，流模式 所解决不了的**---页面分层问题

a、浮动：----多个块级元素位于同一行，解决不了精确定位的问题

b、相对定位

c、绝对定位

d、固定定位

**3、相对定位**

a、position：relative；

---设置了元素为相对定位（脱离了原有的层，原有的位置依然保留）

b、位置偏移属性

top/left/right/bottom:;----相对于元素原来的位置发生偏移

c、层叠顺序

z-index:;---层的顺序，正数，层向上

------适用于移动范围较小的情况，配合其他定位方式

**4、固定定位**

a、position:fixed;---设置了元素为固定定位（脱离原有的层，原有的位置不 再保留，其他元素补位）

b、位置偏移属性

top/left/right/bottom:;----相对于页面发生偏移，固定在页面的某个位 置，不随着页面的滚动而移动

c、层叠顺序

z-index:;---层的顺序，正数，层向上

------适用于为页面添加导航效果

**十二、显示**

**1、透明度：元素设置了背景色，不透明的**

opacity:;---0到1之间的数据

**filter:alpha(opacity=70); (IE)**

**2、光标（鼠标）：默认情况下，不同类型的元素，光标形状不同**

文本： I

图片：箭头

a：小手

cursor:default/pointer/help/wait/...;

**3、display:none/block/inline;---设置元素的显示方式**

**JavaScript**

**1、是一种基于对象和事件驱动的解释性语言**

window window.alert()

Math Math.sin(124)

Date Date.xxxx

。。。

**2、如何书写**

a、写在 <script type="text/javascript"></script>

b、将代码直接书写在 <script> 块

-----页面加载，马上执行

c、将代码封装为方法，在事件中，调用他

**3、如何书写调用方法**

a、调用 JS 提供好的某个方法，实现某种功能

alert是一个方法，实现弹出一个对话框

调用方法的语法：

alert();

b、自定义一个方法

function 方法名称(){

}

调用方法

方法名称();

**4、JS 的基础语法**

a、数据类型

b、变量

字面值：写在代码里，用完就丢弃

var aa = 123;---内存里开辟一个空间，把数字123存起来，用 aa 指向 它；当我需要使用123的时候，用 aa

c、运算符

d、流程控制

e、自定义方法

**5、JS 各种内置对象**

**6、面向对象编程**

**DOM：基于 JS ，实现对于页面上各种元素的动态处理**

1. 分析布局

2、使用合理的标签写出来（尽量使用语义化的标签）

3、决定了选择器

4、样式

-------------------------------**HTML5BASIC**--------------------------------

**Day01**

赵旭

zhaoxu@tedu.cn

QQ:948227958

**WEB前端 ：**

WEB : 网页

前端 ：界面、速度

第一阶段 ：基础

HTML + CSS + Javascript

HTML5Basic（基础）

CSS3Basic（基础）

WEB UI

CSS3Core（核心）

JavascriptBasic（基础）

**开发型WEB前端 ：**

第二阶段：

Javascript

第三阶段：

jQuery 框架

HTML5Core(核心)

服务器端

Ajax

第四阶段：

Bootstrap ：ＣＳＳ框架

AngularJS : JS框架

微信开发

**设计型WEB前端：**

第二阶段：

WEB UI :设计网页 + 实现

第三阶段：

Javascript

第四阶段：

HTML5Core

Ajax

jQuery

Bootstrap

微信开发

**学习方法：**

练：写代码

背：

背效果

背单词

**1、WEB 与 Internet**

**1、Internet**

因特网、互联网

**主要服务:**

WWW,BBS(电子公告板，论坛),Email...Telnet,FTP

**目标：**信息共享

**实现技术：**

分组交换原理：

TCP/IP协议

**2、WEB**

**1、什么是WEB**

WEB就是 Internet 上面的一种 网页 应用程序

将 各种服务 与 信息进行连接，并且提供了生动的图形用户界面

**2、WEB的工作原理**

**1、基于 B/S 结构的模式**

B/S :

B : Browser , 浏览器

S : Server , 服务器

通过 浏览器去访问服务器的一种模式，都称为BS结构模式

C/S :

C ：Client , 客户端

S ：Server , 服务器

必须通过 指定的客户端 才能访问的服务器的一种程序或模式，统称为 CS结构模式

代表：QQ

**2、WEB的组成**

由 WEB服务器(Server)、浏览器(Browser)、通信协议(HTTP协议)

HTTP:Hyper Text Transfer Protocol

超级 文本 传输 协议

规定了数据是如何打包的以及数据在互联网上是如何传递的。

服务器：

PC ：不是服务器，Personal Computer(个人计算机)

Server : 硬件强悍

**3、浏览器与服务器**

**1、服务器**

主要功能：

存储web页面，提供托管环境

响应浏览器的请求，执行服务器端程序

主流的服务器软件：

TOMCAT : Apache

IIS ：微软

APACHE : Apache

**2、浏览器**

主要功能：

User Agent : UA,用户代理

提交用户请求

解释 HTML代码 和 脚本程序

使用图形化方式显示内容

浏览器产品：

Chrome : Google

IE : Microsoft

Opera : 欧朋

Safari ： Apple

Firefox : Mizilla

**4、WEB相关技术**

**1、服务器端技术**

只能运行在服务器端

所有的服务器端技术都提供了数据库访问能力

php

jsp : Java Server Page

asp : Active Server Page

aspx : ASP.NET

**2、客户端(浏览器)技术**

运行与客户端，由浏览器解释运行

HTML

CSS

Javascript

**2、HTML快速入门**

**1、HTML概述**

HTML：Hyper Text Markup Language

超级 文本 标记 语言

html文件 以 .html 或 .htm 作为结尾的

**2、HTML 基本语法**

**1、标记语法**

**1、什么是标记**

在HTML中，用于描述功能的符号，称为标记,不同标记具备不同的功能

**2、标记的组成**

由 < > 组成

<标记名称> : <a> <b> <br>

**3、标记的类型**

标记分成2大类：

**1、封闭类型标记**

有开始，也有显示的结束的标记，称为封闭类型标记，也叫 "双标记"

语法：

<标记名称>内容</标记名称>

<a></a> : 超链接

<b></b> : 加粗

<div></div>

注意：内容， 是受标记的样式控制的。

**2、非封闭类型标记**

有开始，可以没有结束。自己既可以表示开始也可以表示结束。也称为 空 标记 或 单标记

语法：<标记名称> 或 <标记名称/>

注意：单标记 本身 就是一段独立的功能

<br> : 换行

<hr> : 水平线

<img> : 图片

**2、元素**

就是标记

从开始标记 开始 到 结束标记 之间的所有内容，都统称为元素

<a href="http://www.baidu.com">百度</a>

元素：标记、属性、内容

**1、元素嵌套**

在一个元素内部 又增加了另外一个元素

一个html页面 就是由各种各样的 嵌套元素所组成的

**语法：**

<标记1>

<标记2></标记2>

</标记1>

**注意：**

**1、嵌套顺序**

一定要完整嵌套

<div>

<p></p>

</div> 正确

<div>

<p>

fdsafdsafdsa

</div>

</p> 错误

**2、代码缩进**

嵌套代码中，子级元素前，要通过几个空格来表示层级关系

<div>

<p></p>

</div>

嵌套标记写法：

1、开始 结束 一起写

<div>

<p>

<a></a>

</p>

</div>

**3、属性 和 值**

属性 ：修饰元素不同显示效果的内容

值：最终，确定显示的效果的内容

属性 只能 声明在开始标记中 ，并且 属性 与 标记名称 中间用 空格分开

语法：<标记 属性></标记>

在一个元素中，属性可以有过个，多个属性间用空格区分，并且排名不分先后

语法：<标记 属性1 属性2 属性3></标记>

属性与值：

属性 和 值 用 = 关联

语法 <标记 属性1="值" 属性2='值'></标记>

<p id="p1" align="center"></p>

标准属性：

所有的元素都具备的属性

id : 标识元素在页面中的唯一名称

title : 鼠标移入到元素上时，所提示的文本内容

class : 定义元素的 类样式(与CSS相关)

style : 定义元素的 内联样式 (与CSS相关)

**4、html 注释**

标识 内容 不会被浏览器解释运行。

通常会将 对代码的解释说明 放在注释中

语法：<!-- 被注释的内容 -->

注意：

1、注释不能嵌套注释

<!--

<!-- -->

--> 错误

2、注释不能写在标签内

<div<!-- -->></div>错误

**5、HTML版本**

HTML2.0

HTML3.0

HTML4.01 1999 12 24 发布

XHTML1.0 2000 1 26 发布

HTML4.01

<div align=center></div>

XHTML1.0

<div align="center"></div>

HTML4.01

<br>

XHTML1.0

<br/>

**HTML5 : H5**

目标：让代码更简洁

1、所有的单标记可以不加 /

<br> <br/>

2、某些特定的属性 可以不给值

disabled

checked

selected

Readonly

**3、html文档结构**

**1、文档类型声明**

用于指定 html 文档的风格和版本

html5 :

<!doctype html>

出现在整个 html 文档 最顶层

**2、html页面**

**位置：<!doctype html>之下**

**数量：有且仅有一个**

**标签：<html></html>**

html页面包含两大部分：

**1、网页头部**

负责定义网页的全局信息

标签：<head></head>

网页头部允许包含以下内容：

1、<title></title> --标题

2、<meta/> --定义页面元数据(编码、关键字、描述 ...)

3、<script></script> -- 定义/引入Javascript

4、<link /> -- 引入外部 CSS 样式表

5、<style></style> -- 定义内部样式表

**2、网页主体**

标签：<body></body>

作用：定义网页的主体，所有的显示内容，都要放在body中

**4、文本标记**

**1、特殊字符**

需求： 页面中打印输出 <p></p>

注意：

1、html中 ， <> 不能随意输出

2、页面中无论出现多少个 回车 和 空格 ，最终都被解释成 1 个空格

解决方案：

靠 "转义字符" 解决这些特殊的效果

转义字符：

&lt; <

&gt; >

&nbsp; 空格

&copy; ©

&yen; ￥

**2、文本标记**

**1、文本样式**

能够改变文本的显示效果的标记

<b></b> : 加粗

<i></i> : 斜体

<u></u> : 下划线

<s></s> : 删除线

<sub></sub> : 下标

<sup></sup> : 上标

**2、标题元素**

通过改变文字的 大小 和 加粗 方式 来呈现文本的显示效果，主要作用为 突出 显示文字

<hn></hn> n:1~6

一级标题 字最大，六级标题 字最小

**3、段落元素**

突出显示文本 为段落，表现形式为 上下会具备 垂直的空白

<p></p>

**4、换行元素**

效果等同于 回车的效果

<br> 或 <br/>

CV大法:

Ctrl + C : 复制

Ctrl + V : 粘贴

**5、分区元素**

1、行内分区元素

<span></span>

作用：自定义显示效果

2、块分区元素

<div></div>

作用：布局

**6、分割线元素**

作用：在页面 绘制一条 水平线

<hr>

注意：水平线的前后内容会换行显示

属性：

size：高度

width：宽度

color : 颜色

align : 线条的水平对齐方式

取值：left(左) center(中) right(右)

**7、预格式化**

作用：保留 源文档中的格式（回车 和 空格）

<pre></pre>

**8、行内元素 与 块级元素**

1、行内元素：

多个元素会在同一行内显示

<b></b>

<s></s>

<u></u>

....

<span></span>

作用：修饰文本的不同表现形式

2、块级元素

默认情况下，块级元素会独占一行(宽度 100% )，块级元素的前后 都会自动 换行。

作用：布局

**Day02**

**1、图像和链接**

**1、URL**

**1、目录**

文件夹

所谓的目录 就是 web 站点中的文件夹名称

**2、URL**

**1、什么是URL**

Uniform Resource Locator

统一 资源 定位器

俗称：路径，

从一个位置到目标文件位置所经过的路线描述

URL的表现形式：

1、绝对路径

2、相对路径

3、根相对路径

以上三种形式的本质区别：查找资源文件的起始位置不同

**2、绝对路径**

从 资源文件 所在的最高级目录下开始查找

1、获取网络资源

1、协议名称

2、主机名称(域名、IP地址)

3、各级目录路径

4、文件名称

http://www.baidu.com/img/270\_36cdbb596e619fcc8548d28d950 6ad3d.gif

2、获取本机资源

从 文件所在 的盘符 开始查找

C:/Users/Public/Pictures/Sample Pictures/Koala-1.jpg

弊端：不能更换计算机,不适合程序的移植

**3、相对路径**

是通过当前文件 与 资源文件 之间的位置关系进行查找的

从当前文件位置出发 找到 资源文件所在位置所经过的路径 就是相对路径

同级别 ：直接找

子目录 ：先进入

父目录 ：先返回

**4、根相对路径(了解)**

永远都是从 WEB 站点的 根目录处开始查找

语法：/

/images/login.jpg

注意：根相对路径，只有将web项目部署到服务器时才可用

注意：路径要严格区分大小写。尤其是在服务器上。

**2、图像**

**1、图像格式**

jpeg ：\*.jpg \*.jpeg (有损压缩，压缩较大，文件小)

gif：\*.gif (动画)

png：\*.png (透明)

**2、图像元素**

将图像显示在页面中

标签：<img> 或 <img/>

属性：

src : 要显示图像的URL(绝对、相对、根相对)

width : 宽度

height：高度

数字作为值，像素(px)作为单位

注意：width 或 height 如果只给定一个属性的话，那么图片会按照给定的 值进行等比缩放

title : 鼠标移入到图片时或者图片不显示时，所提示的文字

alt : 作用与title一样，老版本浏览器支持，现在主流浏览器支持性很差。

1. ****Filter :****  grayscale灰度.......(严重兼容性问题)

**3、注意问题**

1、align

尽量加在 块级元素中

行内元素没有作用

2、标签嵌套问题

1、p 标记中只能嵌套 文本、图像、行内元素，绝对不能嵌套块级元素，否 则的话块级元素会将 p 元素拆开

2、hn 标题，一般只嵌套 文本 或 行内元素

3、div 允许嵌套任何元素(除 html 、body、head之外)

4、行内元素不许嵌套块级元素

**3、链接**

**1、允许用户 进行点击 的操作，可以通过链接来表示**

典型代表：用户点击 文字 ， 跳转到另外一个页面

**2、标签**

标签：<a>内容</a>

属性：

href : 链接URL

注意：添加了 href 属性后，链接才可以被点击，否则，没有效果

target : 目标

打开新页面的方式（在当前标签页打开还是 在新标签页中打开）

取值：

\_self : 在当前标签页中打开(默认值)

\_blank : 新标签页中打开

**3、链接的表现形式**

1、目标文档为下载资源

<a href="01.rar">下载</a>

<a href="01.zip">下载</a>

以上两种格式的文件出现在 href 中，不是跳转，而是 下载

2、目标 为发送电子邮件

<a href="mailto:zhaoxu@tedu.cn">联系我们</a>

3、返回页面顶部的空链接

<a href="#">返回顶部</a>

4、链接到Javascript

<a href="javascript:js代码"></a>

**4、锚点**

在文档的某个位置处，做一个记号.随时随地可以跳转到该记号的位置处。

锚点的使用方式：

1、定义锚点

1、<a name="锚点名称"></a>

2、<标签 id="值"></标签>

2、链接到锚点

<a href="#锚点名称"></a> --跳转到本页的锚点处

<a href="页面url#锚点名称"></a> --跳转到其他页面的指定锚点处

http://www.tmooc.cn

**2、表格**

**1、什么是表格**

用来组织结构化信息的一个"容器"

表格是由 一系列的”单元格“按照从左到右从上到下的顺序排列组成的。表格的最终 使用目的是将数据保存在单元格中

html中的表格 主要由三大部分组成

1、表格 : <table></table>

2、在表格的内部，由若干行 组成的 <tr></tr>

3、在行的内部，由 单元格(列)组成的 <td></td>

注意：默认情况下，所有行的列数都是统一化的。即每行的列数都一样。

练习：在html中，创建一个 5行4列的表格

4、表格标题

出现的位置在表格之上，水平居中的内容

<caption></caption>

注意：

1、caption的位置 位于 所有tr 之上，紧跟着table之下

2、一个table最多只能有一个 caption

**2、属性**

**1、table 属性**

width : 宽度

height : 高度

align : 表格在其容器内的水平对齐方式,取值：left(默认),center,right

border : 边框,边框宽度 以px为单位的数值

bgcolor : 表格的背景颜色

cellpadding ： 内边距,即 单元格与内容之间的距离

cellspacing : 外边距,即 单元格与单元格之间的距离

**2、tr 属性**

align : 当前行数据的水平对齐方式

valign: 当期行数据的垂直对齐方式(top,middle,bottom)

**3、td属性**

width

height

align

valign

bgcolor

colspan : 设置单元格跨列

rowspan : 设置单元格跨行

**3、th 标记**

th 与 td 是同级别的

th ：标题，当前列的内容，以标题的样式呈现出来的（加粗显示，水平居中）

**4、表格的复杂应用**

**1、行分组**

table 可以将 tr 划分为 3大部分：表头、表主体、表尾

表头 ：<thead></thead>

表主体：<tbody></tbody>

表尾：<tfoot></tfoot>

**2、不规则的表格**

改变传统表格的模式，每行的列数可以不同

实现方式：设置 td 的跨行 和 跨列 属性

跨行 ：rowspan

跨列 ：colspan

跨列：从某个单元格位置处 横向向右 ，合并几个单元格(包含自己)

注意：被 合并 掉的单元格，一定要从代码中删除出去，否则当前行会多出几 个单元格

跨行：从某个单元格位置处 纵向向下 ， 合并几个单元格(包含自己)

注意：被 合并 掉的单元格，一定要从代码中删除出去，否则当前行会多出几 个单元格

**3、表格嵌套**

在单元格中，允许放置另一个表格

在 <td> 中再包含一个 <table>

**3、列表**

**1、什么是列表**

列表 也是 按照一定的格式来显示数据的

格式：默认情况，从上到下的显示数据

**2、什么样的数据 适合 放在列表中**

会将 具有相似特征 或 明显具备先后顺序的内容放在列表中

**3、列表的组成**

列表类型：有序列表 和 无序列表

列表项：列表中所显示的内容

**4、使用列表**

**1、有序列表**

可以将列表项的"顺序"罗列出来的列表

标记:<ol></ol> --> Order List

列表项：<li></li>

<ol>

<li>数据1</li>

<li>数据2</li>

<li>数据3</li>

</ol>

属性:

1、type

列表项前面的标识类型 是什么

取值：

1 ：数字 （默认值）

a ：小写字母

A ：大写字母

i ：小写罗马数字 i ii iii iv v vi ...

I ：大写罗马数字

2、start

规定 在 type 值中的第几个位置处开始

**2、无序列表**

通过统一的固定标识，显示出列表项

标签：<ul></ul> --> Unorder List

列表项：<li></li> --> List Item

属性：

1、type

列表项标识类型

取值：

disc : 实心圆(默认值)

circle : 空心圆

square : 实心矩形

**3、使用场合**

1、数据 按照 从上到下的顺序来排列的时候

2、数据 按照 从左到右的顺序来排列的时候 - 网页导航

**5、定义列表**

给出一类事物定义的情形时使用

张无忌

英俊潇洒，玉树临风... 有好几个对象 ....

语法：

<dl>

<dt>描述的术语或名词</dt>

<dd>对上述名词或术语的解释</dd>

</dl>

使用场合：图文混排

<dl>

<dt>

<img src="xxx.jpg" >

</dt>

<dd>

Price : 125.55

xxxxx

</dd>

</dl>

**Day03**

**1、结构标记**

<div id="a"></div>

<div id="b"></div>

<div id="c"></div>

...

<div id="aa"></div>

...

<nav id="a"></nav>

**1、作用**

为了提升标记的语义性

**2、结构标记**

**1、<header></header>**

作用：用于定义文档的页眉，就是html文档中最上面的内容

等同于 ：<div id="header"></div>

<header>

<h1>Welcome</h1>

<img src="..."/>

</header>

**2、<nav></nav>**

作用：用来定义页面的导航链接部分

<nav>

<ul>

<li></li>

</ul>

</nav>

等同于：<div id="nav"></div>

**3、<section></section>**

作用：文档主体内容中的小节,现在，section可以表示整个页面中的主体内容

等同于：<div id="main"></div>

**4、<article></article>**

作用：用于描述 文本性 较强，或艺术气息较强的

一般情况下，论坛中的帖子信息、报纸中的文章信息、博客或微博中的条目信息、 用户回复信息 优先考虑放在 article 中

**5、<footer></footer>**

作用：定义页面中或某个区域中的脚注信息（位置靠下的信息）

**6、<aside></aside>**

作用：多数情况用于定义页面中的 侧边栏 信息

**2、表单**

**1、表单作用**

用于显示、收集信息 并且将信息提交个服务器

表单中主要包含两组内容：

**1、表单元素**

**2、表单控件**

**2、表单元素 <form>**

标签：<form></form>

注意：

1、使用表单控件提交数据时，表单不能省略

属性：

**1、action :** 动作，后台处理程序的地址(服务器端工程师提供)。默认 提交给本页。

**2、method :** 方法、方式，表单的提交方式。不同的提交方式，约束的内容不一 样。

常用取值：2个

get ：

1、显示提交数据，会将提交信息显示在地址栏上，安全性不够高

2、大小限制，最大支持到 2KB 的提交

3、如果 不设置 method 属性的话，默认就会按照 get 的方式进 行提交。

4、使用场合：向服务器索取数据时，优先使用get。

post :

1、隐式提交数据，不会将提交的信息显示在地址栏上，安全性较高， 所有有关密码的信息提交时，必须用post

2、post 没有大小限制(大量数据提交时，可以使用)

3、使用场合：

1、安全性要求较高的页面(如包含敏感信息)

2、要求服务器 去处理数据时， 使用post

3、传递数据量较大时使用

**3、enctype(了解)**

功能：设置 表单中提交的数据的 编码方式

规范哪些数据可以提交给服务器...

取值：

1、applicatioin/x-www-form-urlencoded

默认值，

可以将普通的文本，特殊的字符，一起提交给服务器

2、multipart/form-data

允许将表单中的文件，传递给服务器，普通文本不能直接传递

3、text/plain

只能将普通文本(英文、数字)传递给服务器，特殊字符(=,& ...)不允 许

4、id

5、name

**3、表单控件**

**1、什么是表单控件**

包含在 表单 中的元素 ，具备可视化外观，并且可以接受用户输入的数据

**2、分类**

**1、input 元素**

**2、textarea 文本域**

**3、select 和 option 选项框**

**1、input 元素**

作用：收集用户信息

语法：<input>

属性：

**1、type ：**根据不同的类型值，可以创建不同的输入控件

**2、value ：**控件的值，提交给服务器的数据

**3、name ：**控件的名称，必须设置，否则无法提交。服务器主要根据 name的值，来获取value。

**4、disabled ：**禁用控件,该属性可以没有值。

**具体的 input 类型的控件**

**1、文本框 与 密码框**

**文本框: type="text"**

<input type="text">

作用：允许用户输入任意字符的数据，明文显示。

**密码框：type="password"**

<input type="password">

作用：允许用户输入任意字符的数据，密文显示。

**属性：**

1、maxlength ：限制输入的字符数

2、readonly ：只读,无需给值，只有有该属性即可

3、value ：控件值，同时也可以设置控件的默认值

注意：input元素下，如果不写 type值，或者 type值写错时，都 默认为文本框。

**2、单选框 和 复选框**

**单选框：type="radio"**

<input type="radio">

**属性：**

1、name ：定义名称 并且 实现控件分组。只有一组内的元素 才能实现单选

2、value

3、checked ：设置默认被选中,该属性不需要值

**复选框：type="checkbox"**

<input type="checkbox">

**属性：**

name ：定义名称 并且 分组

value :

checked :

**3、按钮**

**1、提交按钮**

功能固定化，负责将数据提交给服务器

<input type="submit">

**2、重置按钮**

功能固定化，负责将表单控件恢复到初始化的状态

<input type="reset">

**3、普通按钮**

没有固定功能，由 开发者 通过 JS 来设置

<input type="button">

**属性：**

value : 显示在按钮上的文本

name : 名称

**4、非 input 标签的按钮**

<button></button>

属性：type

**4、隐藏域 和 文件选择框**

**1、隐藏域**

不想让用户看见，但是又要提交给服务器的数据，可以放在 隐 藏域中

eg ：用户 id

type="hidden"

<input type="hidden">

属性：

name

value

https://www.baidu.com/?username=admin&userpass=ad min&repass=admin&gender=Male&hobby=drink&userI D=334455

**2、文件选择框**

作用：提供一个基础控件，允许用户选择本机的文件上传到服务 器

type="file"

<input type="file">

属性：

name

注意：

1、表单的method 属性值必须为post

2、表单的enctype属性值必须为 multipart/form-data

**2、textarea**

文本域，能够接收用户录入的多行数据

标签：<textarea>默认文本</textarea>

属性：

name ：名称

readonly ：只读

cols : 指定文本区域的列数,变相的设置控件的宽度

rows : 指定文本区域的行数,变相的设置控件的高度，数据如果超出高度 的话，则出现滚动条

**3、选项框**

**1、下拉选项框**

**2、滚动列表**

**标签：**

**1、选项框**

<select></select>

属性：

name：

size：默认显示的选项数量是几个

如果值大于1则为滚动列表，否则就是下拉选项框

multiple : 设置多选，可以没有值，配合着键盘上的ctrl或 shift键位一起

**2、选项**

<option>显示的值</option>

属性：

value ：提交给后台服务器的值

selected : 设置默认被选中，可以没有值

**4、其他标记**

1、<label><label>

作用：关联文本和表单控件，关联后，点击文本就如同是点击控件一样。

语法：<label>文本<label>

属性：

for : 指定 与当前 label 相关联的控件的 id 值

2、为控件分组

分组：<fieldset></fieldset>

标题：<legend></legend>

**3、其他标记**

**1、浮动框架**

可以将多个页面放到一个页面中去显示，就是页面间的嵌套

标签：<iframe></iframe>

属性：

1、src

要引入的网页url

2、width

3、height

4、frameborder

浮动框架的边框，可以手动设置为 0

**2、摘要与细节**

简单模拟 展开 与 收缩的动作的标记

标记：

<details></details> 用于定义细节信息

<summary></summary> 定义细节内容的标题，允许被用户点击的部分，是 <details>的子标记

**3、度量元素**

可以在页面中定义一个度量衡,可以描述有关 比例 信息

标记：

<meter>文本</meter>

属性：

min : 定义范围的最小值,默认值为0

max : 定义范围的最大值,默认值为1

value : 度量值，当前显示的值，默认为0

**4、时间元素**

用于定义 显示的时间 与 具体时间值的一个关联操作

标记:

<time>显示的文本</time>

属性：

datetime : 设置关联的日期 和 时间,日期与时间之间用T来分割

**5、高亮文本显示**

标记：

<mark>高亮显示的文本</mark>

-------------------------------**CSS3BASIC**--------------------------------

**Day01**

**1、css概述**

**1、html 控制样式的弊端**

**1、相同的样式效果，通过不同的属性 来完成的**

<body text="">

<font color="">

**2、程序 可维护性 不高**

**2、通过 CSS 解决上述问题**

可以让 各个元素 都使用统一的 样式声明 从而提高程序的可重用性和可为维护性

**3、什么是CSS**

CSS ：Cascading Style Sheet ,层叠样式表、级联样式表，简称为样式表

作用：为 html 元素 去定义样式

能够实现 内容与表现相分离

提高程序的可重用性 和 可维护性

重用性：可以让多个元素 使用相同的样式

维护性：一个元素的样式改变，其他元素也可以跟着变

**4、HTML 与 CSS 之间的关系**

HTML 主要负责显示内容

CSS 主要负责样式的定义

一个完整的页面 = HTML + CSS

W3C建议尽量使用CSS样式 取代 HTML 属性

**5、CSS样式表的使用**

**1、使用方式**

**1、内联方式**

也称为 ：内联样式 、行内样式

将样式定义在HTML元素中

特点：只能控制某一个元素的显示效果

语法:<标记 style="样式属性:值;样式属性:值;"></标记>

常用属性：

color : 文本颜色，取值 表示颜色的英文单词

style="color:red;"

background-color : 背景颜色

style="background-color:green;"

style="color:red;background-color:green;"

font-size : 字体大小，取值 以px为单位

style="font-size:20px"

**2、内部样式表**

将所有的样式内容 放在 <head></head>中的 <style></style>元素内

<style></style>中 添加 若干 "样式规则"

**样式规则：**对一套 样式的 完整描述,主要包含两部分

1、哪些标记允许使用定义好的样式 - 选择器

2、样式内容 - 若干样式声明

**样式规则语法：**

选择器{

//样式声明

样式属性:值;

样式属性:值;

...........

}

**选择器：**通过 标记 的名称来表示选择器，相当于定义该标记的样式

div{

color:red;

background-color:green;

font-size:48px;

}

p{

color:blue;

font-size:24px;

}

**特点：**

1、能够实现 内容与表现相分离

2、提升了 样式的可重用性和可维护性

3、只能在一个页面中做通用样式定义

**3、外部样式表**

将样式规则 保存在独立的 样式文件中，由页面对样式文件进行引用

**特点：**

1、实现 内容与表现相分离

2、将可重用性 和 可维护性 体现的 最完美(所有的页面都能用)

3、可以在整个网站乃至于互联网中做通用的样式定义

**步骤：**

1、创建一个 样式表文件

纯文本文件，以 .css 结尾

该文件中，只能包含样式规则

2、在页面中，对样式表文件进行引用

在 <head></head>中 增加以下代码

<link rel="stylesheet" type="text/css" href="链接的css文件地 址" />

**2、CSS语法规范**

**1、基本语法**

样式声明： 属性:值;

样式规则： 选择器 和 样式声明

**2、CSS样式表特征**

**1、继承性**

大部分的CSS样式属性，是可以被继承的

继承：子级不用单独定义样式，可以直接父级的东西

**2、层叠性**

为一个元素定义多个样式的时候

样式不冲突时，多个样式表中的样式可以层叠(合并)为一个

**3、优先级**

样式冲突时，按照不同样式规则的优先级来应用样式

优先级：

1、浏览器默认设置 (User Agent Style) ,级别最低

2、内部样式表 或 外部样式表,级别 处于中间位置

就近原则

3、内联样式 ,级别 最高

Unknown property name : 样式属性名称写错了

Invalid property value : 样式属性值写错了

**4、!important规则**

可以通过 !important 显示调整样式优先级，!important优先级最高

语法： 样式属性:属性值 !important;

谨慎使用

**3、CSS基础选择器**

**1、选择器的作用**

规范了页面中的 哪些元素 能够使用定义好的样式

p{

color:red;

}

div{

font-size:12px;

}

**2、选择器(重点)**

**1、通用选择器**

**作用：匹配页面中所有的元素**

**语法：\***

**eg:**

\*{

font-size:12px;

}

在页面会会匹配多次

body{

font-size:12px;

}

在页面上只匹配一次，其余都是继承，效率较高

注意：能不用就不用

**2、元素选择器**

**作用：定义/修改特定 "标记" 的默认样式**

**语法：标记名称作为选择器**

**别名：标签选择器、标签样式**

**eg:**

div{

....

}

p{

....

}

ul{

....

}

**3、类选择器**

**作用：定义公共样式，可以由其他标记灵活引用**

**语法：.className**

**使用：通过标签的class属性对className进行引用**

**别名：类样式**

**注意：**

1、className 不能以数字开头

2、如果一个标记向同时引用多个类的话，中间用 空格 区分

**eg:**

.redBack{

background-color:red;

}

<div class="redBack"></div>

<div></div>

**4、类选择器 - 分类选择器**

**作用：将类选择器 和 元素选择器结合到一起，从而实现对某种元素中不同样 式的细分控制,其目的是缩小范围**

**语法：元素选择器.类选择器{}**

**eg:**

.redBack{background-color:red;}

div.redBack{background-color:pink;}

**5、id选择器**

**作用：匹配 页面中指定id值得元素的样式**

.bea{

background-color:blue;

color:red;

}

<div id="menu" class="bea"> </div>

**语法：#idValue{}**

**eg:**

#menu{

color:red;

background-color:blue;

}

<div> id="menu">...</div>

**6、群组选择器**

以 , 隔开的选择器列表

**作用：定义一组选择器的公共样式**

**语法：选择器1,选择器2,选择器3 ...{}**

**eg:**

div,p,span,#menu,.big{

font-size:24px;

}

**7、后代选择器**

通过标签的 上下文关系(父子级关系) 来声明定义样式

爹的儿子 可以称为 子代 或 后代

爹的孙子以及以下辈分 只能叫 后代

**语法：选择器1 选择器2{}**

**eg:**

/\*所有 div 元素 里 所有的p元素\*/

div p{

}

/\*id为message的元素 里 所有的 p元素 \*/

#message p{

}

#message .important{

}

**注意：不强调层级关系**

**8、子代选择器**

要求选择器之间只能存在 父子关系

**语法：选择器1>选择器2**

**注意：子代选择器，只向下找一级，多了不管**

**eg:**

/\*匹配 id 为message 元素 下一级的 所有div元素\*/

#message>div{

}

/\*匹配 id 为 message 的元素中 下一级 class为 important的元素中 下 一级的 div 元素\*/

#message>.important>div{

}

**9、伪类选择器**

**1、作用：匹配元素 不同的状态**

**2、伪类选择器 分类**

1、链接伪类

2、动态伪类

3、目标伪类

4、元素状态伪类

5、结构伪类

6、否定伪类

**3、语法规范**

:伪类选择器{}

**1、链接伪类**

:link 匹配 尚未访问的 超链接

:visited 匹配 访问过的 超链接

**2、动态伪类**

:hover 匹配 鼠标悬停在 html元素上的状态

:active 匹配 元素被激活时的状态

:focus 匹配 元素获取 焦点时的状态(text,password)

**注意：**

对于超链接的伪类设置 ，一般情况下只设置 两种选择器即可

1、该标记的默认样式

2、鼠标移入到该标记的样式

**3、选择器优先级**

选择器 权值

标签选择器 1

类选择器 10

伪类选择器 10

ID选择器 100

内联样式 1000

如果选择器冲突的时候， 最终以 权值相加 大者为主

**3、尺寸与边框**

**1、单位**

**2、尺寸**

多表示元素的宽和高，取值为数字，单位为 px 或 %

**1、宽度**

**属性：width**

min-width

max-width

**2、高度**

**属性：height**

min-height

max-height

**3、溢出**

设置元素尺寸时，并且 元素中的内容 的所占空间超出元素本身的话，会导致内容 溢出。

**属性：overflow**

overflow-x : 横向溢出处理

overflow-y : 纵向溢出处理

**取值：**

**1、visible**

默认值，溢出可见

**2、hidden**

隐藏

**3、scroll**

滚动条，在元素中增加滚动条，溢出时可用，不溢出，则不可用

4、auto

自动，溢出时显示滚动条，不溢出是不显示滚动条

**注意：**

1、块级元素 允许设置 尺寸属性

2、具备 width height 属性的 html 元素

img,table,input

3、行内元素 不允许设置 尺寸属性

**3、边框**

**1、边框**

**1、简写方式**

border:width style color;

width:边框线条的尺寸，值越大越明显 ，以px为单位

style:边框线条样式，取值:solid(实线),dotted(虚线),dashed(虚线)

color:颜色

该属性 设置 四个边框的 宽度 样式 颜色

**2、单边定义**

border-left/right/top/bottom:width style color;

border-方向:width style color;

方向：left/right/top/bottom

**3、单属性定义**

border-width:四条边宽度

border-style:四条边样式

border-color:四条边颜色

**4、单边单属性**

border-方向-属性 :

方向：top/bottom/left/right

属性：width/style/color

注意：

1、边框颜色 ，可取值为 transparent(透明)

边框的组成 是由 四个 三角形 或 梯形组成的

**2、边框倒角**

**圆角矩形**

属性：border-radius

取值：以px为单位的数字 或 %

**单独定义：**

border-top-left-radius:左上

border-top-right-radius:右上

border-bottom-left-radius:左下

border-bottom-right-radius:右下

**3、边框阴影**

**属性：box-shadow**

**取值：由多个值组成的值列表**

box-shadow:h-shadow v-shadow blur spread color inset;

h-shadow:阴影水平投射距离，值为正 向右偏，值为负 向左偏(必须)

v-shadow:阴影垂直投射距离，值为正 向下偏，值为负 向上偏(必须)

blur : 可选 ，模糊距离

spread : 可选，阴影尺寸

color ：可选，颜色

inset ：可选，将默认的外部阴影 改为 内部阴影

常用场合：模拟边框发光效果

box-shadow:0px 0px 1px color;

**4、轮廓**

绘制于元素边框之外的一条线

属性:outline

取值：width style color;

outline:width style color;

outline:0px;

outline:none;

**Day02**

**1、框模型**

**1、框**

框就是个容器，目的为了承装其他的东西

html中所有的元素都可以称之为 框

**2、框模型**

Box Model : 也称为盒子模型

定义了元素框 处理元素内容尺寸、内边距、边框 和 外边距的方式

**元素内容尺寸 ： width , height**

div{

width:100px;

height:100px;

}

**边框：border**

div{

border:1px solid #ddd;

}

**注意：**当框模型(内边距，外边距，边框)介入到元素后，元素的 内容区域 尺寸是不变 的，但是会增加元素框的总尺寸

**元素总宽度 = 左右外边距 + 左右边框 + 左右内边距 + width;**

**元素总高度 = 上下外边距 + 上下边框 + 上下内边距 + height;**

**eg :**

div{

width : 200px ;

border: 2px solid #ddd;

左右外边距 : 15px;

左内边距: 8px;

右内边距: 4px;

}

总宽度：200 + 2\*2 + 15\*2 + 8 + 4 =

200 + 4 + 30 + 12 = 246px

**3、外边距**

**1、什么是外边距**

在元素边框周围的一些空白区域，这些区域是透明的，默认情况下，是不允许被其 他元素所占据的。也可以表示为 当前元素 与 其他元素之间的距离

**2、语法：**

**margin:value; (四个方向的外边距)**

**单边设置：**

margin-top : 上外边距

margin-bottom:下外边距

margin-left:左外边距

margin-right:右外边距

**取值：**

**单位:**

px

%

auto : 自动，只对 左右外边距有效,设置水平居中

负值 : 目的 是为了移动元素

margin-left:取负值，为了向左移动元素

margin-top :取负值，为了向上移动元素

**取值为 auto ：**由浏览器自己计算外边距。左右方向取值为 auto 时，会呈 现出 元素水平居中对齐显示方式（仅仅局限于块级元素，而且必须要设 置宽度属性）

**取值数量：**

margin:value; 四个方向外边距

margin:v1 v2; 上下 左右

margin:v1 v2 v3; 上 左右 下

margin:v1 v2 v3 v4; 上 右 下 左

**3、默认 外边距**

html元素里， 有些元素默认具备 外边距的

h1~h6

p

ul

body

dl,dd

如果程序中，不想使用默认的外边距，可以设置为 0

**4、外边距问题**

**1、(垂直)外边距合并**

当两个垂直外边距相遇时，它们将形成一个外边距，称为 外边距合并

合并后的外边距的高度值为两个元素中 外边距较大者的值

**2、外边距溢出**

父子元素中，如果设置 子元素 的上下外边距时，默认情况 会 被作用到父元素的 外边距上

**解决方式：**

1、可以为 父元素 增加边框

2、增加 父元素 的 padding-top

最好将 父元素的 高度 相应的 减少一部分数值

**4、内边距**

**1、什么是内边距**

内容区域 与 边框之间的距离控件

注意：内边距 会将元素边框 撑大

**2、属性**

padding : value;

padding-top:

padding-bottom:

padding-right:

padding-left:

**取值：**

**单位：**

px

%

**数量：**

padding:value ; 上下左右

padding:v1 v2 ; 上下 左右

padding:v1 v2 v3;上 左右 下

padding:v1 v2 v3 v4;上 右 下 左

**2、背景**

**1、背景颜色**

**属性：background-color**

**取值：**可设置为合法的值 或 transparent

**注意：**默认情况下，背景(颜色、图像)是从边框位置处就开始填充。

**2、背景图片**

**属性：background-image**

**取值：**url("图像URL")

background-image:url("xxxx/xxx.jpg");

background-image:url('xxxx/xxx.jpg');

background-image:url(xxxx/xxx.jpg);

**3、背景平铺**

**属性：background-repeat**

**取值：**

repeat : 即水平方向又垂直方向平铺(默认值)

no-repeat : 没有平铺

repeat-x : 仅水平方向平铺

repeat-y : 仅垂直方向平铺

**4、背景图片尺寸**

**属性：background-size**

**取值：**

value1 value2 : 宽度 高度

value1% value2% : 父容器的宽和高的占比

cover : 覆盖，将背景图像扩展至足够大，直到背景图已经覆盖了元素的所有区域. 有可能导致背景图不能完整的显示在元素中。

contain: 包含，将背景图扩大，直到背景图已经碰都某一个边缘位置（右、下）, 背景图能够完完整整的显示在元素中，不一定能覆盖到所有的区域。

**5、背景图片固定**

**属性：background-attachment**

**取值：**

scroll ：背景 会随文档发生滚动

fixed : 固定 ， 背景不会随着内容而发生滚动

**6、背景定位**

改变背景图片在元素中的位置

**属性：background-position**

**取值：**

**x y :** x水平偏移位置,值为正 向右移动，值为负，向左移动

y垂直偏移位置,值为正 向下移动，值为负，向上移动

**x% y% :**

左上角 : 0% 0%

右下角 : 100% 100%

居中显示 : 50% 50%

**关键字：**

x : left,center,right

y : top,center,bottom

**CSS Sprite :** 将多个背景图合并到一张里面去，根据需求情况，通过 背景定位 对背 景图片进行偏移，从而找到要显示的图像

**7、背景属性**

将 以上 几个属性全部综合到一起,通过一个属性描述所有的值

**属性:background**

**取值：color url( ) repeat attachment position;**

**常用值：**

background:url( ) reapeat position;

background-image:url(Images/user.png);

background-repeat:no-repeat;

background-position:right center;

background:url(Images/user.png) no-repeat right center;

**3、渐变**

**1、渐变语法：**

**属性：background-image**

**取值：**

linear-gradient() -- 线性渐变

radial-gradient() -- 径向渐变

repeating-linear-gradient() -- 重复线性渐变

repeating-radial-gradient() -- 重复径向渐变

background-image:url(); -- 显示背景图像

background-image:linear-gradient(); -- 显示线性渐变颜色

**2、线性渐变**

**属性：background-image**

**取值:linear-gradient(angle,color-point1,color-point2,...)**

**angle:方向(角度)**

to top : 向上 --> 0deg

to right : 向右 --> 90deg

to bottom : 向下 --> 180deg

to left : 向左 --> 270deg

**color-point :**

表示颜色的数值 和 位置

red 10px

red 0%

background-image:linear-gradient(to top,red 0%,green 50%,blue 100%);

**3、径向渐变**

**属性：background-image**

**取值: radial-gradient();**

radial-gradient([size at position],color-point1,color-point2,....);

size at postion :

可以省略不写

size : 标识的圆的半径 以 px 为单位

position : 圆心的位置

top,left,right,bottom,center

具体数值

百分比

**4、重复渐变**

repeating-linear-gradient(angle,color-point);

color-point:

位置 取具体的数值或者 不到 100% 的 数字

repeating-radial-gradient();

**Day03**

**1、文本格式化(重要)**

**1、字体属性**

**1、指定字体(非重要)**

font-family:value1,value2,value3....;

font-family:"微软雅黑","宋体","Arial";

**2、字体大小(重要)**

font-size:

取值 ： px 或 pt 作为单位的值

**3、字体加粗(非重要)**

相当于 : <b></b>

font-weight:normal/bold/无单位数字;

无单位数字：400 - normal

900 - bold

**4、字体样式(非重要)**

相当于 ：<i></i>

功能：斜体 显示文字

font-style : normal / italic;

**5、小型大写字母(非重要)**

font-variant:normal/small-caps;

**6、字体属性**

font:font-style font-variant font-weight font-size font-family;

**常用设置方式：**

font:12px "微软雅黑","Arial";

font:12px/24px "microsoft yahei";

font:font-size/line-height "微软雅黑";

行高

**2、文本属性**

**1、文本颜色 (重要)**

color:value;

**2、文本水平排列方式(重要)**

相当于：html 属性中的 align

text-align:left / right / center

功能：能够控制 当前元素内 所有的文本、行内元素、行内块、水平对齐方式。

**3、文字线条修饰**

**属性：text-decoration**

**取值：**

none 无线条(重要)

underline 下划线(重要)

overline 上划线(非重点)

line-through 删除线(非重点) --> <s></s>

**4、行高(重要)**

一行文本所占据的高度是多少

如果 行高 大于 文本大小的话，那么这行文字将呈现出垂直居中的显示方式

属性：line-height

取值：以 px 为单位的值

**5、首行文本缩进(非重要)**

属性：text-indent

取值：以 px 为单位的值

**6、文本阴影(非重要)**

text-shadow:h-shadow v-shadow blur color;

h-shadow:水平投射距离

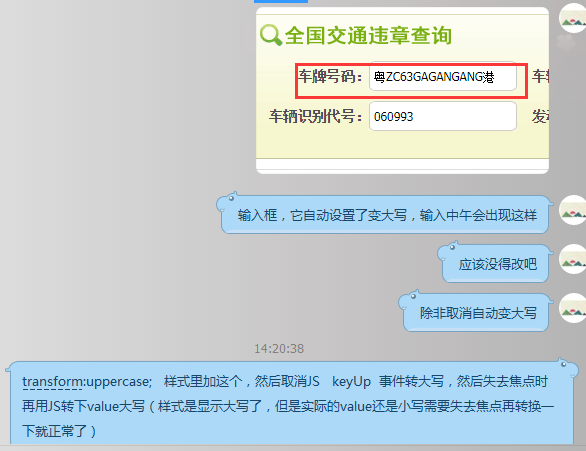
v-shadow:垂直投射距离

blur:模糊距离

color:颜色

**Input属性，输入直接变大写**

**text-transform:uppercase;**



**2、表格 属性**

**1、表格常用属性(重点)**

1、内边距 : padding

2、尺寸属性 ：width,height

3、文本属性 : font-\* , text-\*

4、背景属性 : background-\*

5、边框 : border

6、vertical-align

功能：控制单元格内容的垂直对齐方式

取值：top / middle / bottom

**2、表格特有属性**

**注意：该组属性只能在 table 中使用**

**1、边框合并(重要)**

**属性：border-collapse**

**功能：合并两个相邻的边框**

**取值：**

separate : 默认值，分离边框

collapse : 合并

**2、边框边距(非重点)**

两个边框间的 上下 和 左右的距离

**注意：只有在 boder-collapse:separate;时才有效r**

**属性：border-spacing**

**取值：**

取一个值：每两个单元格之间的垂直和水平间距是一致的。

取两个值：第一个值，指定的是水平间距，第二个值，指定的是垂直间距。两 个值用 空格 隔开。

**3、显示规则(非重点)**

**属性：table-layout**

**取值：**

auto : 列宽度由内容来决定，默认值。自动表格布局

fixed: 列宽度由设置的值来决定。固定表格布局

**自动表格布局：**

单元格的大小会适应内容的大小

加载时较慢

适用于不确定每列大小时使用

**固定表格布局：**

列宽度取决于设置好的相关属性值，与内容无关

会加速显示表格，每次加载表格时不用计算

**固定表格布局虽然速度较快，但是不够灵活，不能体现出表格的特点。**

**自动表格布局，虽然算法较慢，但是是传统表格的体现。**

**2、浮动定位**

**1、定位**

元素框 相对于其正常位置 应该出现的 位置在哪

**分类：**

**1、普通流定位**

流：排列元素的一种方式

又称为 文档流定位 ，是页面默认排列元素的一种方式。

页面中的块级元素：元素是从上到下的方式排列

页面中的行内元素：元素是从左到右的方式排列

弊端：块级元素 无法 一行内显示多个，布局不好做。

**2、浮动定位**

解决问题：让多个块级元素能够在一行内显示

**3、相对定位**

**4、绝对定位**

**5、固定定位**

**2、浮动定位**

**解决问题：让多个块级元素能够在一行内显示**

**1、什么是浮动定位**

1、将元素排除在文档流之外，即元素脱离文档流。不受 默认排列方式(从上到下 或 从左到右)的控制

2、元素将不再占用页面的空间

有可能会压住其他的元素

3、元素 会停靠在包含框的左边 或 右边，或者停靠在已经浮动的元素的左边 或 右边

4、元素无论怎么浮动，最终还是在包含框之内

**2、浮动属性**

**属性：float**

**取值：none / left / right;**

**注意：**

1、块级元素浮动后 ，宽度会变成自适应

2、行内元素 浮动起来后，除具备以上特点外，它将变成块级元素

**3、清除浮动**

**功能：清除当前元素的左边(上边) 或 右边 已浮动元素对它所带来的影响**

**属性：clear**

**取值：left / right / both / none**

**4、float 与 overflow**

**原因：**浮动元素 会对 父元素的高度带来影响，父元素的高度，最终以没有浮动元 素的高度为准。

**解决方案：**

1、手动设置元素的高度

弊端：自适应高度时无法使用

2、通过 overflow 属性改变

overflow:hidden;放在 包含浮动元素的父元素中，它会撑起当前元素的 高度，变得自适应

弊端：能够隐藏超出范围的元素

3、在父元素内最后位置处，追加一个子元素 div，增加 clear:both;属性即 可。

**Day04**

**1、显示**

**1、显示方式(重点)**

**功能：可以通过 display 属性 修改元素框的显示方式**

**属性：display**

**取值：**

**none : 让生成的元素没有框**

让元素脱离文档流

不显示元素(隐藏)

隐藏元素，并且不占据页面空间

**block : 让元素 像块级一样显示**

使用场合：将行内元素 改变成块级元素

**inline : 让元素 像行内元素一样去显示**

注意：不要将块级元素 改成 行内元素

**inline-block : 让元素显示的向 行内块 一样**

本身是行内元素：一行内能够显示多个

具备块级元素特点：允许改宽和高

**table : 让元素 显示的和 table 一样**

**总结：display 常用方式**

**1、隐藏元素 以及 显示元素**

可以通过 display:none的方式隐藏

如果需要显示的话，按照自己默认的方式显示出来即可

比如：div : display:block , span : display:inline;

**2、如果程序中 ，想修改 行内元素的 尺寸时使用**

float:left / right;

display:block / inline-block;

**2、显示效果**

**1、可见性(非重点)**

**特点：元素可以隐藏，但是空间会保留，不允许被其他元素占据**

**属性：visibility**

**取值：**

visible : 默认值，可见的

hidden : 元素不可见，但保留空间

collapse : 用在表格元素上，删除一行或一列时，不影响表格整体布局

**2、透明度(非重点)**

**属性：opacity**

**IE: filter:alpha(opacity=70);**

**取值：**0 - 1 之间的小数(能取0 也可以取1)

值越小，越看不清

**3、垂直对齐方式**

**使用场合：**

1、在td中使用 ：

2、在 img 中使用 : 控制图像两端的文本 相对于图像的垂直对齐方式

**属性：vertical-align**

**取值：**

top:

middle:

bottom:

baseline : 默认值，基线对齐，默认会将文本放在元素的基线上

**注意：**通过 修改 图像的vertical-align 为 非 baseline的值，取消默认底部的 3px空白距离

**3、光标(非重点)**

**属性：cursor**

**取值：**

default 默认值

pointer : 小手(重点)

crosshair : +

text : I

wait : 等待

help : 帮助

**2、列表(非重点)**

**功能：灵活控制或修改 列表项的显示标识**

**1、list-style-type**

**修改列表项前面的标识**

**属性：list-style-type**

**取值：**

none ：无标记（重点）

disc : 实心圆

circle : 空心圆

square :实心方块

decimal : 数字

... ...

**2、列表项图像(非重点)**

**功能：使用自定义图像，作为列表项的显示标识**

**属性：list-style-image(非重点)**

**取值：url(图像路径);**

**3、列表项位置(非重点)**

**属性:list-style-position**

**取值：**

outside ：列表项标识位于 文本之外，默认值

inside ：列表项标识更改在文本范围之内

**4、列表属性**

**功能：将以上三个属性值 综合到一起**

**属性：list-style**

**取值：type url() position;**

**常用写法：list-style:none;(重点)**

**3、定位(重点)**

**1、什么是定位**

相对于其本身位置，应该出现的位置在哪

**2、定位分类**

1、普通流定位 / 文档流定位

2、浮动定位

3、相对定位

4、绝对定位

5、固定定位

**3、定位属性**

**1、定位属性**

**属性：position**

**取值：static / relative / absolute / fixed**

**2、偏移属性**

top ：

bottom ：

left ：

right ：

该组属性 能够帮助我们完成元素的位置移动

**3、堆叠顺序**

属性：z-index

取值：无单位的数字

**4、定位方式**

**1、相对定位（重点）**

**1、what**

元素框会相对于它原来的位置偏移某个距离

要参考 元素本身的位置 然后去实现新的位置移动

**2、属性 和 值**

position:relative;

通过 top / left / bottom / right 属性值实现位置的微调

**3、注意：**

1、相对定位不会改变元素的形状

2、元素原本所占的控件仍然会保留

**4、使用场合**

1、元素本身要进行位置的微调时，优先选择相对定位

2、配合绝对定位一起使用(弹出菜单)

3、要实现堆叠顺序的调整

**2、绝对定位（重点）**

**1、什么是绝对定位**

1、绝对定位的元素会脱离文档流，不占据页面空间

2、相对于 最近的 已定位的祖先元素 来实现位置的初始化,如果元素没有已 定位的祖先元素，那么就相对于最初的包含框body 实现位置的初始化

**2、属性 和 值**

position:absolute;

通过 top / left / bottom / right 实现位置调整

**3、使用场合**

1、想实现元素堆叠的效果时使用(一个元素压住另外一个元素)

2、弹出菜单

**3、堆叠顺序（重点）**

已经定位的元素（尤其是绝对定位）是允许出现堆叠效果的。

**默认的堆叠顺序**

1、按元素出现的顺序，后来者居上(同级别)

2、子元素永远都会压在父元素的上面(父子级,不可变)

**修改堆叠顺序:**

属性：z-index

取值：无单位的数字，数字大者靠上

注意：无法修改 父子级元素的堆叠顺序

页面中哪些元素能够使用堆叠顺序？

已定位的元素可以使用(relative,absolute,fixed)

**4、固定定位(非重点)**

1、什么是固定定位

将元素固定在页面的某个位置处不动

会将元素脱离文档流，不占页面空间

2、属性 和 值

position:fixed;

通过 left / bottom / right / top 实现位置调整

**5、静态定位(了解)**

position:static;

默认定位方式。

-----------------------------**CSS3CORE**----------------------------

**Day01**

**1、复杂选择器**

**1、兄弟选择器**

**兄弟：具备相同父元素 的元素可称为兄弟元素**

**1、相邻兄弟选择器**

**相邻兄弟：**当前元素 后面的 紧紧挨着的元素可称为相邻元素

<div>

<a></a>

<b></b>

<span></span>

</div>

**作用：**获取 当前元素 后面的 紧紧挨着当前元素 并且具备指定选择器的元素

**语法：** + 作为结合符

selector1+selector2

eg:

div+span : 获取 紧挨在div后面的 span元素

#section+.message : 获取 紧挨在 id为section元素 后面的 class 为 message的元素。

**2、通用兄弟选择器**

**作用：**匹配某元素后所有的并且满足指定选择器的兄弟元素

**语法：**~作为选择器结合符

selector1~selector2

**2、属性选择器**

**作用：**能够将元素所附带的属性用于选择器当中

<div id="main" class="" style=""></div>

**语法：**[ ]

**1、 [attribute]**

[id] : 匹配所有 附带 id 属性的元素

[class] : 匹配所有 附带 class属性的元素

div[style] : 匹配所有附带 style 属性的div元素

**问题：**获取页面所有的文本框元素

input[type="text"]

**2、element[attr1][attr2]**

匹配同时附带 attr1 属性以及attr2属性的 element元素

div[id][class] : 即附带 id属性又附带class属性的div元素

**3、element[attr=val]**

element : 元素

attr:属性

val:值

匹配attr属性值为val的element元素

input[type=password]

匹配type属性值为password的input元素

**4、element[class~=val]**

**作用：** 匹配 class属性值是以空格隔开的单词列表，并且val是单词列表中 的一个单词

<div class="lf border"></div>

div[class~=border]

**5、element[attr^=val]**

匹配 element元素 attr 属性以val开始的元素

div[class^=user\_]

<div class="user\_name" ></div>

<div class="user\_age"></div>

<div class="teacher\_salary"></div>

**6、element[attr$=val]**

匹配 的是 attr属性以val作为结尾的element元素

<input class="userable" />

input[class$=able]

**7、element[attr\*=val]**

attr属性之中 包含 val 字符的 element元素

**3、伪类选择器**

**1、目标伪类**

**作用：**突出显示活动的html锚元素。

**语法：** :target

**2、元素状态伪类**

匹配 元素 启用、禁用、被选中的状态

**使用场合：**表单元素上

**语法：**

1、:enabled

匹配每个已启用元素

2、:disabled

匹配每个被禁用的元素

以上两个选择器，用于表单元素中

3、:checked

匹配每个被选中的input元素（限于 radio 和 checkbox）

**3、结构伪类（重点）**

**1、:first-child**

匹配属于其父元素的首个子元素

table td:first-child

table tr td:first-child

**2、:last-child**

匹配属于其父元素的最后一个子元素

**3、:empty**

匹配没有子元素的每个元素

<div>

<span></span>

</div> 不是 empty

<div>太阳的后裔</div> 不是empty

<div>

</div> 不是empty

<div></div> 标准empty

**4、:only-child**

匹配属于其父元素的唯一子元素

**5、:nth-child(num)**

匹配是其父元素中的第 num 个子元素

**4、否定伪类**

:not(selector)

在一组元素中将 满足 selector 选择器的元素排除出去

table td:not(:first-child)

**4、伪元素选择器**

**伪类：匹配元素**

**伪元素：匹配 元素中的内容**

**1、:first-letter 或 ::first-letter**

匹配指定元素的第一个字符

**2、:first-line 或 ::first-line**

匹配指定元素的首行

**3、::selection**

**（选择器匹配被用户选取的部分）**

: 与 ::的区别

CSS2.0 中 伪元素选择器和伪类选择器全部都使用相同的标识 :

CSS3.0 中 伪类选择器使用 : ,而伪元素选择器一律使用 ::

**2、内容生成**

作用：在现有的标记中，通过css的方式，追加新的内容进来

**1、伪元素选择器**

:before

定位到元素的内容区域之前

:after

定位到元素的内容区域之后

**2、属性**

**属性：content**

**注意：配合 :before 和 :after 一起使用**

**取值：**

**1、字符串 ，纯文本**

**1、解决问题**

**1、包含浮动元素的父元素的高度问题**

1、指定元素高度

2、overflow:hidden;

3、追加空子元素，clear

4、 div:after{

content:"";

display:block;

clear:both;

}

**2、上外边距溢出**

#d2:before{

content:"";

display:table;

}

**2、图像，url( )**

**3、计数器**

**1、什么是计数器**

通过css的方式生成一个数字，作为内容添加到元素中去

**2、属性 和 函数**

**1、counter-reset 属性**

**功能：声明(复位)一个计数器，并且可以赋初始值**

**注意：**

1、 不能放在使用计数器的元素中去声明

2、 如果页面中多个元素要使用计数器的话，可以放在 body或这些元素的父元素中去声明

**语法：**

counter-reset:名称 初始值;

counter-reset:名称1 初始值1 名称2 初始值2 ...;

初始值可以省略，不写默认为0

**2、counter-increment属性**

**功能：设置计数器出现时的增量，默认为1。**

**语法：**

counter-increment:名称 增量值;

**注意：** 哪个元素使用计数器，counter-increment就放在哪 个元素中声明。

**3、counter() 函数**

作为 content属性的值出现的。

**功能：使用指定计数器生成的数值。**

**语法：**

content:counter(名称);

**3、多列**

**1、column-count 属性**

指定元素被拆分成几列

**eg:**

column-count:3;

**2、column-gap 属性**

指定列与列之间的间隔距离

**eg:**

column-gap:25px;

**3、column-rule 属性**

指定两个列之间的线条 宽度 样式 颜色

**eg:**

column-rule:width style color;

**4、浏览器兼容性**

IE10 + , Opera 支持多列属性

Firefox : -moz-

Chrome , Safari : -webkit-

**4、CSS Hack**

**解决的问题：**浏览器兼容性(IE)

cursor:hand;

**CSS Hack的原理 ：**通过选择器和样式属性的优先级来解决问题

**CSSHack方式：**

1、CSS类内部Hack

在样式属性名前加浏览器前缀或在属性值后加浏览器后缀即可

2、选择器Hack

在选择器前加浏览器的识别前缀即可

3、HTML头部引用Hack

根据条件注释 判断浏览器类型，再决定做哪些操作

条件注释：

语法结构

<!--[if 条件 IE 版本]>

<![endif]-->

问题1:在ie8浏览器中 打印输出 一对h1

<!--[if IE 8]>

<h1></h1>

<![endif]-->

**Day02**

**1、转换**

**1、什么是转换**

转换指的是 使元素 改变 形状、尺寸、位置的一种效果

在CSS中，允许元素实现 2D 和 3D的转换效果,主要包含 ：旋转，缩放，移动，倾斜

**2D ：**元素只能在X轴和Y轴平面上发生变化

**3D ：**元素还可以在 Z轴上发生变化

**2、转换属性**

**1、transform 属性**

**-webkit-transition: all 2s;**

**-ms-transform: translateX(100px);**

**-o-transition: all 2s;**

**transition:**

**作用：**规范元素使用2D或3D的转换效果

**取值：** 可以指定一组转换"函数"

none : 默认值，元素不进行转换

transform-function : 表示一个或一组转换函数，多于一个的话，中间 用 空格 分开

**2、转换的原点**

**注意：**会对 旋转操作 有很大的影响

**属性：**transform-origin

默认位置，原点是在元素的中心位置处

**取值：**数值 / 百分比 / 关键字

两个值：表示x轴 和 y轴的位置

三个值：表示x轴 y轴 z轴

**3、2D转换**

**1、 什么是2D转换**

使元素在x轴 和 y轴上所进行的转换效果可以橙汁为2D转换

2D转换包含：位移 缩放 旋转 倾斜

**2、2D转换-位移**

**1、 位移**

让元素产生一个位置的移动变化效果

**2、属性和函数**

**属性：transform**

**函数：translate()**

translate(x) : 在x轴的位移距离

translate(x,y) ：在x轴和y轴的位移距离

**取值：数值、百分比**

可以取负值：

x为正 则向右移动

x为负 则向左移动

y为正 则向下移动

y为负 则向上移动

单向位移函数：

translateX(x):只在x轴进行位移操作 -->translate(x)

translateY(y):只在y轴进行位移操作

**3、2D转换-缩放**

**1、缩放**

改变元素的大小 称之为缩放

**2、属性和函数**

**属性：**transform

**函数：**scale()

scale(x)

scale(x,y)

**注意：**如果只给一个值，那么第二个默认与第一个值相等

**取值：**

默认值为1

**缩小：**0 到 1之间的数值

**放大：**大于1的数字

**单向缩放函数：**

scaleX(x) : 只在横向(宽度)产生变化效果

scaleY(y) : 只在纵向(高度)产生变化效果

**4、2D转换-旋转**

**1、旋转**

使元素围绕着一个点（转换原点）实现角度的变化

**2、属性与函数**

属性：transform

函数：rotate(ndeg)

n取值为正 ：则按顺时针旋转

n取值为负 ：则按逆时针旋转

**5、2D转换-倾斜**

**1、倾斜**

能够改变元素的形状

以原点位置，让元素围绕着x轴或y轴按照一定的角度倾斜

**2、属性 和 函数**

**属性：**transform

**函数：**skew(x)

skew(x,y)

取值为角度

**单向倾斜函数：**

skewX(x) --> skew(x)

skewY(y)

**4、3D转换**

**1、什么是3D转换**

在x轴和y轴的基础上，增加对z轴(空间轴)的转换效果

**2、perspective 属性**

假定 人眼 到 投射平面的距离

注意：

1、使用perspective属性元素的本身不会得到3d转换效果，其子元素 会有3d转换效果

2、浏览器兼容性

Chrome 和 Safari 需要使用 -webkit-perspective属性

**3、3D转换-位移**

**1、什么是3d位移**

在2d基础上，增加了对z轴上的位移距离

**2、属性 和 函数**

属性：transform

函数：

translateZ(z)

取值为正：向着人眼方向移动，物体越大

取值为负：原理人眼方向，物体越小

translate3d(x,y,z)

**4、3D转换-旋转**

1、属性 和 函数

属性：transform

函数：

1、rotatex(xdeg)

2、rotatey(ydeg)

3、rotatez(zdeg)

4、rotate3d(x,y,z,deg)

rotate3d(1,1,1,45deg);

-->rotatex(45deg) rotatey(45deg) rotatez(45deg)

rotate3d(1,0,0,45deg) -->

rotatex(45deg)

**5、transform-style 属性**

**作用：**如何在3D空间中，呈现被嵌套的元素

规范了当前元素的子元素 呈现什么样的位置显示

**取值：**

flat ：子元素将不保留其3d位置，呈2d位置显示

preserve-3d ：子元素将保留其3d位置

**2、过渡**

**1、什么是过渡**

使CSS属性值，在一段时间内的平缓过渡

过渡 强调的是中间的过程，而不是最后的结果

**2、过渡4要素**

1、过渡属性

2、过渡时间

3、过渡函数：过渡效果在完成过程的变化速率

4、过渡延迟时间：当用户激发操作后，等待多长时间开始执行过渡效果

**3、激发过渡效果**

现阶段，只能通过 鼠标移入 时进行激发

**4、过渡属性**

**1、过渡属性(必要)**

规定使用css过渡效果的属性是谁

**属性：**transition-property

**取值：**none / all / property

**以下属性是允许使用过渡效果的：**

**1、颜色属性**

color

background-color

border-color

**2、取值为数值的属性**

**3、转换属性**

transform

transition-property:transform;正确

transition-property:rotatex();错误

**4、渐变属性**

background-image:

background:

**5、visibility**

**6、阴影属性**

**注意：** 如果想一次性定义多个属性使用过渡效果的话，那么多个属性名称之间， 用，区分即可。

1、元素宽度发生变化的时候，使用过渡效果

transition-property:width;

2、元素宽度和高度发生变化是使用过渡效果

transition-property:width,height;

**2、过渡时间(必要)**

一个完整的过渡效果需要花费的时间是多少

属性：transition-duration

取值：以 s | ms 作为单位

1000ms = 1s

默认值为0，即不采用过渡的效果。在设置过渡效果中，该属性必须设置，而且要 大于0。

**3、过渡的时间曲线函数**

规定了过渡效果的速度曲线

属性：transition-timing-function

取值：预定义的值或贝塞尔曲线

ease:默认值，慢速开始，快速变快，再慢速结束

linear:匀速

ease-in:慢速开始，加速效果

ease-out:快速开始，减速效果

ease-in-out:慢速开始和结束，中间先加后减

**4、过渡延迟**

规定了，激发操作后，等待多长时间再显示过渡效果

属性：transition-delay

取值：以 s | ms作为单位的数字

**5、过渡简写属性**

transition:property duration timing-function delay;

transition:transform 2s,border-radius 2s;

**3、动画**

**1、创建动画的步骤**

**1、声明动画**

创建一个 已命名 动画

指定 关键帧 上的行为动作

**2、为元素调用动画**

调用定义好的动画，以实现动画效果

animation 属性 可以调用动画

**2、声明动画**

**语法：**

**@keyframes name{}**

**兼容性:**

@-webkit-keyframes name{}

@-o-keyframes name{}

@-moz-keyframes name{}

**关键帧：表示的是动画的分解动作**

1、每个关键帧都代表着动画执行的一个时间点

2、定义每个关键帧的动作(元素的样式状态)

@keyframes name{

/\*若干关键帧\*/

0% | from{

CSS样式;

}

1%{}

...

100% | to{

CSS样式;

}

}

**案例：**

1、动画开始时 (0%)，width=100，height=100，blue的背景颜色

2、动画进行到一半的时候(50%)，元素变为圆形，背景颜色变为 green 色

3、动画结束时(100%)，元素变为 初始的状态,矩形，背景颜色变为 blue 色

**3、调用动画 - 动画属性**

**1、animation-name**

需要调用动画的名称

**2、animation-duration**

动画完成一个周期所需要的时间，取值单位为 s | ms

**3、animation-timing-function**

规定动画的速度曲线

ease

linear

ease-in

ease-out

ease-in-out

**4、animation-delay**

播放动画之前的延迟时间

取值单位 s | ms;

**5、animation-iteration-count**

取值：具体数值 或 infinite(无限次播放)

**6、animation-direction**

作用：指定动画的播放方向

取值：

normal : 默认值，正常播放 0% - 100%

reverse : 反向播放，从 100% - 0%

alternate : 轮流播放，奇数次数正向播放(0% ~ 100%),偶数此时，反向播放 (100% ~ 0%)。

**7、animation 属性**

简写属性，定义动画的各个子属性

animation:name duration timing-function delay iteration-count direction;

**8、animation-fill-mode**

**作用：**规定动画在播放前以及播放后，其动画效果是否可见

**none :** 默认值，不改变默认行为

**forwards :** 当动画完成后，保持在最后一个帧的状态上

**backwards:**在动画播放前，应用第一帧的状态，注意,只有在播放延迟时才会体现

**both :** 相当于同时设置 forwards 以及 backwards

**9、animation-play-state**

规定动画的播放状态(暂停、播放)

取值：

paused ：动画暂停

running ：动画播放

-----------------------------**JAVASCRIPTBASIC**----------------------------

**Day01**

**Javascript基础**

**1、Javascript 概述**

**1、什么是Javascript**

是一种运行与 Javascript解析器 / 引擎中的解释型脚本语言

编译型：运行之前会做检查语法结构是否有问题，如果有问题的话，程序是不会被 运行的。

解释型：不会检查错误，直接运行，直到碰见错误再停止运行

**2、运行环境**

1、独立安装的Javascript解析器

NodeJS

2、嵌入在浏览器内核中的Javascript解析器

**3、Javascript 发展史**

1、1992年 Nombas公司 开发了一款 叫做 ScriptEase 脚本语言

2、1995年 Netscape公司为Navigator2.0浏览器开发了一种脚本语言 LiveScript,为了赶时髦，从此改名为 Javascript。

3、1996年 Microsoft 在IE3.0 发布了一款Javascript克隆版 JScript

4、1997年 Javascript1.1 作为草案提交给了 ECMA(欧洲计算机制造商协会)

ECMA-262标准

ECMAScript是全新的标准的脚本语言

**完整的Javascript 包含三部分：**

**1、核心-ECMAScript**

**2、文档对象模型(DOM,Document Object Model)**

允许我们操作页面的元素

**3、浏览器对象模型(BOM,Browser Object Model)**

允许我们与浏览器进行对话，比如：操作地址栏，历史记录

**4、Javascript特点**

**1、代码可以使用任何文本编辑工具编写**

语法结构类似于C，Java，C#

**2、无需编译，由Javascirpt的引擎负责执行**

**3、弱类型语言**

由数据 决定了 变量的数据类型

对应 强类型语言

由数据类型决定数据

**4、基于对象 ， 面向对象(OO)**

**5、Javascript 用途**

**1、客户端Javascript**

1、客户端数据计算

2、表单验证

3、浏览器事件的触发和处理

4、网页特效/动效制作

5、异步数据提交(Ajax)

**2、服务器端Javascript**

1、分布式运算

2、实时服务器

3、窗口应用

4、网络应用

**2、使用Javascript**

**1、浏览器内核**

**作用：负责页面内容渲染,主要有两部分组成**

**1、内容排版引擎**

负责解析 HTML / CSS

**2、脚本解释引擎**

负责解析 Javascript

**浏览器 内核 排版引擎 脚本引擎**

IE Trident - Chakra

Firefox Gecko - 猴子系列

Safari Webkit Webcore Nitro

Chrome Webkit Webcore V8

Opera Presto - Carakan

Opera(2013)Webkit Webcore V8

**2、第一个Javascript程序**

**1、使用独立的Javascript解释器(了解)**

console.log("Hello Javascript");

**2、使用浏览器内嵌的js引擎执行Javascript**

1、在控制台(console)上输入脚本并执行

2、将JS脚本嵌入在HTML页面中去执行

1、将脚本写在html元素的事件里

onclick事件:当用户点击元素时要执行的操作

\" : "

2、通过<script>元素 执行JS脚本

3、使用外部脚本文件

1、创建 以.js为后缀的文件

2、直接编写JS代码

3、页面通过<script src=""></script>标记，进行外部JS文件的引 入

**3、语法规范**

Javascript 代码组成 - 语句

语句：会被JS引擎所执行的代码

由表达式、关键字、运算符

严格区分大小写

console.log();

Console.log(); /\*错误，C应小写\*/

username

Username

通常情况下，语句都用；表示结束

**3、变量 和 常量**

**1、变量**

**1、什么是变量**

**一个可以变化的量(数据)**

**内存：**保存程序运行过程中所需要用到的数据。

**变量：**就是一段内存的空间，目的为了保存数据二用。

**变量名：**变量的名称。就是内存地址的别名

**2、变量的声明**

**声明：**在内存中，开辟一段空间

**语法：**var 变量名;

**保存数据：**使用 = 为变量赋值

**= :** 赋值运算符

var 变量名; //声明

变量名=值;//赋值

var 变量名=值;//声明变量并且初始化值

**初始化：**为变量第一次赋值的过程可称为初始化

**注意：**

如果声明了变量，但是没有初始化，那么变量将自动取值为 "undefined"

**问题：**声明一个变量，用来保存学员的名称

var name;

问题：声明三个变量，保存三个学员的名称

var name1;

var name2;

var name3;

**一条语句中声明多个变量：**

多个变量名中间，用 , 隔开即可

var name1,name2,name3;

var name1="aaa",name2,name3;

**3、变量名的命名规范**

**1、要求**

1、不允许关键字以及保留关键字左变量名称

比如：var,if,break ...

**2、命名规范**

**1、可以包含字母、数字、下划线以及$**

var name;

var name\*;//错误，不允许包含\*

var name$;

**2、不能以数字开头**

var name1,name2,name3;//正确

var 1name,2name,3name;//错误

var $name;//正确

var \_name;//正确

**3、尽量见名知意**

var age;

var xingming;

**4、变量名不能重**复

**5、可以采用 “匈牙利命名法”,“下划线命名法”,“驼峰命名法”**

**匈牙利命名法：控件缩写+功能名称;**

text : txt

password : txt,pwd

radio : rdo

checkbox : chk

select : sel

submit

button

reset : btn

**eg :** 声明变量 用于保存 从文本框中得到用户姓名的 信息

var txt\_username;

var txtUsername;

**下划线命名法：**

var user\_name;

var \_name;

**驼峰命名法：**

var studentName;

**4、变量的使用**

**1、未经初始化的变量**

1、未初始化指的是，只声明了，但是没有负值的变量,系统会自动赋值为 undefined。

2、变量未声明直接使用

语法错误

**2、声明直接初始化**

将声明变量与赋值操作放在一条语句内完成

var 变量名=值;

var name="张三丰";

var gender="Male";

var age=80;

**3、对变量中的数值进行操作**

1、取值 - GET操作

在程序中直接打印输出变量值

console.log(变量名);

document.write(变量名);

window.alert(变量名);

直接使用变量名即可

2、赋值 - SET操作

通过 = 完成

=的左边 必须是变量

=的右边 可以是值，也可以是变量

var name = "张三丰";

var teaName="张无忌";

name = teaName;//将teaName的值 赋值给name变量

**2、运算符**

**1、算术运算符**

+,-,\*,/,%

- :

\* :

/ :

**% : 俗称 模 ，取余，获取两个数字相除之后的余数**

5 % 2

**使用场合：**

1、判断奇偶性

2、判断 x 数字的倍数

**+ :**

1、计算两个数字相加的结果

2、连接变量 与 普通数据

var name = "张三丰";

console.log("学员姓名:"+name);

由算术运算符组成的表达式称之为 算术表达式

8341

8341 / 10 = 834 ... 1

8341 / 100 = 83 ... 41

**3、常量**

**1、什么是常量**

常量也是内存中的一段存储空间，但是一旦确定值后，就不允许再修改

**2、目的**

保证程序中的数据的准确性

**3、语法**

使用 const 关键字 声明常量

const 常量名 = 值;

习惯上，常量名采用全部大写

eg :

1、声明PI的值为3.14

const PI = 3.14;

练习：定义一个变量，来表示 一个圆的半径，数值自定。计算出该半径的圆的周 长以及面积分别是多少，在控制台上打印输出。

周长：2\*PI\*r;

面积：PI\*r\*r

打印结果 :

半径为 xx 的圆的周长为 xxx

半径为 xx 的圆的面积为 xxx

**4、数据类型**

**作用：决定了内存中开辟多大的空间来保存数据**

**Javascript数据类型：**

**1、原始类型(基本类型)**

**1、number 类型**

数字类型

表示32位(4字节)的整数，，也可以表示64位(8字节)的浮点数

**浮点数：小数**

3M :

3\*1024 = 3072KB \* 1024 = 3145728字节

**整数：**

十进制数：

八进制：0作为开始

var num = 010;

十六进制：0x

var num = 0x00000abc;

**浮点数：**

95.5,3.14,100.00

4.5e10 ...

**2、string 类型**

字符串

在JS中，一个字符或多个字符都可以称之为字符串。

字符串在使用的时候必须用 '' 或 "" 括起来

var name="张三丰";

var age="24"; //字符串

注意：只要字符串参与到 "+" 操作，起到的全部都是连接作用

转移字符:

\" : "

\' : '

\n : 换行

\t : 制表符(英文状态8个空格)

**3、boolean 类型**

布尔类型

取值：true 和 false

使用场合：条件判断的结果

参与+-\*/%运算时，true=1，false=0

1+true

var b=true;

**4、undefined**

数据不存在，未定义值

**5、null**

引用类型使用 ...

**2、引用类型**

**Day02**

**1、数据类型转换**

**1、JS是一款弱类型语言**

var a="Hello"; //string

a=15;//number

**2、typeof函数(运算符)**

**作用：查询指定变量的数据类型**

语法:

typeof(变量)

typeof 变量

**3、数据类型的转换 - 隐式转换**

不同数据类型之间的数据再进行 + 操作时，最终的结果的数据类型是什么

**隐式转换：由Javascript 自动完成的。**

string

number

boolean

undefined

**4、数据类型转换 - 显示转换**

**由 "转换函数" 指定转换成的数据类型，从而改变默认的数据类型转换策略**

var num = "42";

var n = 42;

**转换函数:**

**1、toString( ) : 将任何数据类型的数据转换成字符串**

**2、parseInt( )**

解析出string 或 number 类型的整数部分

parseInt("35"); 可以转换

parseInt("HelloWord");

**注意：**

1、parseInt 在转换时，碰到第一个 非数字的字符时，则停止转换

parseInt("35你好");

parseInt("你好35");

2、如果待转换的数据解析不出数字部分或首字符也不是数字的话，最后 的结果为NaN

**3、parseFloat( )**

在parseInt的基础上，允许转换小数点 .

弹出提示框，输入数据

var salary = window.prompt("提示的内容","输入框的默认值");

**2、函数**

**1、什么是函数**

console.log( );

window.alert( );

document.write( );

parseInt( );

parseFloat( );

函数(function),可以称之为方法(Method),或过程(Procudure)。是一段预定义好的， 独立的功能体，可以被反复的调用。

**预定义：**提前声明，声明后才能使用

**独立功能体：**可以包含若干条的可执行语句

**反复调用：**谁想用谁就过来调用，相当于把该功能体中的所有的语句都执行一遍

**注意：**函数不是在加载时就执行，而是在被调用时才会执行的。

函数的本质是功能完整的"对象"

**2、定义函数(方法，过程)**

**语法：**

function 函数名(){

//若干条可执行操作语句

}

**3、调用函数**

直接通过函数名称进行调用

函数名称( );

<script>

function print(){...}

print();

</script>

<button onclick="print();"></button>

**4、定义参数**

函数在运行过程中，需要使用外部的数据进行辅助操作，那么这个外部数据就是参数

**语法：**

function 函数名(参数列表){

//若干条执行操作语句

}

function 函数名称(参数名称1,参数名称2){

}

定义函数时所指定的参数名称，可称之为 "形参(形式上的参数)"

**调用带参函数：**

函数名称(参数值列表);

调用函数值所传递的参数值，可称之为 "实参(实际参数)"

**3、分支结构**

**1、运算符**

>,<,>=,<=,==,!=

== : 判断两个数据是否相等，值相等，则为true，值不等则为false

!= : 判断两个数据是否不相等,值不相等，则为true，值相等时，则为false

以上运算符运算的结果 一定是 boolean 类型

**2、if语句**

作用：当"条件"满足时，执行某些语句，当条件不满足时，则不运行这些语句

**语法：**

if(条件){

若干条待执行语句

}

条件：结果为boolean类型的值或表达式

若条件为true，则执行if语句块中的语句

若条件为false，则不执行if语句块中的语句

**注意：**

**1、if( )中 ，必须是 boolean 类型的值**

若出现的不是 boolean 类型 ， 则会自动转换

**以下情况，都会自动转换成false**

if(0){ }

if(0.0){ }

if(null){ }

if(undefined){ }

if(""){ }

if(NaN){ }

**2、if语句后 { }**

if后的{ }可以省略，但不推荐

如果 if 语句块中，只有一条待执行语句时，那么就可以省略{}

**3、if...else...语句**

语法：

if(条件){

//语句块1

}else{

//语句块2

}

先判断条件，如果条件为真，则执行语句块1的代码内容，如果条件为假(否则)，则执 行语句块2的代码内容。

**4、else if**

接收一个考试成绩

如果 成绩为100分 则奖励一台兰博基尼

如果 大于等于90分 则奖励一台 法拉利

如果 大于等于80分 则奖励一台 宝马

如果 大于等于70分 则奖励一台 捷达

如果 大于等于60分 则奖励一台 奥拓

否则 奖励一台 永久牌自行车

**多重if结构**

语法:

if(条件1){

//语句块1

}else if(条件2){

//语句块2

}else if(条件3){

//语句块3

} ... ...

else{

//语句块n

}

满足哪个条件，则执行哪个条件后对应的语句块，执行完成后跳出整个结构，不会 进行其他语句块的判断。如果所有的条件都不满足，则执行else语句块的内容。

**Day03**

**1 、循环结构**

**1、什么是循环**

问题：

1、在控制台上输出1句 "好好学习"

console.log("好好学习 ");

2、在控制台上输出10句 "好好学习"

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

console.log("好好学习 ");

3、在控制台上输出 1000句 "好好学习"

... ...

4、将 1000句 "好好学习" 更改为 "Good Good Study"

5、

第1遍 Good Good Study

第2遍 Good Good Study

......

**循环：重复的一遍又一遍的执行相同或相似的操作**

**生活中的循环：**

1、活着

2、上学

**循环的特点：**

1、循环条件

规范了 循环的开始 与 结束

2、循环体

循环过程中，要做的事情

**2、while 循环**

while : 当 ... 的时候

**语法：**

while(循环条件){

循环操作(循环体)

}

如果 循环条件为 true，则执行一遍循环体中的操作。然后再判断一次循环条件， 若为true，则再执行一次循环体 .... .... 直到 循环条件为 false 时，则结束循环。

**注意：**正常处理过程中，不要让条件成为 "永真",否则的话将出现 “死循环” 的 效果。

在控制台上打印10遍 好好学习

1、循环条件

从 第 1 遍 开始

到 第 10 遍 结束

2、循环操作

控制台上 输出 好好学习

**跳出循环结构：**

**break : 在循环体中，用于退出循环**

**使用场合：**

1、在满足某种条件下，提前退出循环

2、在不确定循环次数时，break可以作为结束循环的语句

练习：

1、打印 1-100之间数字，如果碰到第一个 是3的倍数的数字的话，则停止打印

2、要求用户循环录入学员姓名，并打印在控制台上，直到 输入 0 ， 则停止录 入。

条件：输入 0 为止

操作：

1、录入数据

2、打印录入的内容

while(true){

1、录入数据

2、打印录入的内容

//判断条件

if(用户输入 == "0"){

break;

}

}

**3、for 循环**

**1、确定循环次数条件下，while的语句结构**

var i=1;//1、初始化循环条件

while(i<=10){ //2、判断循环条件

console.log("Hello World");//3、循环操作 ..

i++;//4、更新循环变量

}

**2、for语法**

for(表达式1;表达式2;表达式3){

循环操作

}

表达式1：初始化循环条件

表达式2：判断循环条件

表达式3：更新循环变量

**for循环执行流程：**

1、计算表达式1的值

2、计算表达式2的值(boolean类型)，如果为true，则进行下一步，否则， 退出循环

3、执行循环体

4、执行表达式3

5、计算表达式2的值 ，如步骤2

**3、for语句三个表达式的特殊用法**

在for循环中，可以省略任何一个或几个表达式，但分号 不能省略

**1、省略第一个表达式**

var i=1;//条件声明不能省略

for(;i<=10;i++){

console.log(...);

}

**2、省略第二个表达式**

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

console.log(...);

}

条件的表达式可以省略，但条件不能省略

**3、省略第三个表达式**

for(var i=1;i<=10;){

console.log( );

}

表达式3可以省略，如果一旦确定循环次数的话，更新循环变量的操作不能省

**4、for( ; ; ){ }**

**5、表达式1 和 表达式3的位置处，允许出现多个表达式**

**2 、数组**

**1、什么是数组**

一组数据 保存在 一个变量里

可以在一个变量中 存储多个数据

数组中所存放的数据 称之为 数组元素

数组元素在内存中按线性顺序排列.

**线性顺序：**

除第一个元素外，每个元素都有一个唯一的前驱元素

除最后一个元素外，每个元素都有唯一的一个后继元素

数组是按照顺序来排列数据的。

数组中，可以通过元素所在的位置顺序来访问元素

位置顺序：下标 或 索引,是从0开始，到元素个数-1（数组长度-1）

**2、数组的定义和初始化**

**1、var 数组名称=[ ];**

定义一个不包含任何元素的数组(空数组)

**2、var 数组名称=[值1,值2,...值n];**

定义一个包含n个值的数组

**3、var 数组名称=new Array( );**

定义一个空数组

**4、var 数组名称=new Array(值1,值2,...值n);**

**3、访问数组元素**

**1、设置数组元素的值 - SET**

数组名称[索引] = 值;

**2、获取数组元素的值 - GET**

数组名称[索引];

**4、属性：获取数组长度**

**获取数组中元素的个数，即数组长度**

**属性：**length

**使用方式：**数组名称.length;

数组的长度值 就是 数组中新元素位置的索引

**5、循环遍历**

**1、属性**

作用：获取数组元素个数，即数组长度

属性：length

用法：数组名称.length

**2、循环遍历**

for(var i=0;i<数组名称.length;i++){

数组名称[i]

}

**Day04**

**1、DOM概述**

**1、DHTML**

Dynamic HTML 动态的HTML

是 HTML + CSS + Javascript 整合体

**功能：页面在下载后，仍然能实时变化页面元素效果的设计。**

1、动态改变页面元素

2、与用户进行交互

**主要包含 ：BOM ，DOM**

**BOM :**

Browser Object Model

浏览器对象模型

提供了一组访问 浏览器 的方法和属性，使得Javascript有能力与浏览器进行对话

暂时 没有相关标准和规范，但是却被广泛支持。

**DOM :**

Document Object Model

文档对象模型

**文档：**html文档

**作用：**操作html文档及其内容

定义了访问和操作HTML文档的标准方法

W3C组织定义了相关的操作规范和标准

**2、DOM概述**

**W3C DOM标准的3部分：**

1、核心DOM : DOM Core ,针对任何一个结构化(HTML,XML)文档的标准模型

2、XML DOM ：针对XML文档的标准模型

3、HTML DOM ：针对HTML文档的标准模型

1、网页加载时，浏览器会创建文档对象模型

2、文档中所有的节点组成了一个文档树(节点树，DOM树)

document 对象是文档树的根

**发展：**

从发展至今，共三级

DOM 1级规范：1998年，定义最底层机构

DOM 2级规范：在一级上增加了许多交互模块

Core : 扩展了更多的基本方法和属性

Style : 专门操作html元素的样式

Traversal and Range: 专门遍历DOM树

Event : 标准化事件。

DOM 3级规范：扩展了方法和属性，增加新类型

**3、节点树(DOM树)**

HTML DOM 将HTML文档 当做 一个树状结构看待

文档中 元素、属性、文本、注释 都被当做一个节点

树根：document 对象,可以通过document对象操作整个节点树

HTML文档中的每个元素，属性，文本，注释，都代表着树中的一个节点

整个文档是一个文档节点 ：document node

每个HTML元素是一个元素节点：element node

每个HTML中的文本是一个文本节点：text node

每个HTML元素中的属性，是一个属性节点：attribute node

注释是一个注释节点:comment node

**document对象：**

浏览器中的JS解释器会为每个载入的HTML文档创建一个对应的document对象

可以通过document对象，访问页面中所有的元素

**2、选取元素**

**1、通过ID选取元素**

作用：通过 元素的 id属性值，得到页面的某一元素对象

方法：document.getElementById("id属性值");

通过该方法，会得到一个 元素对象(element node)

**2、通过标签名获取元素**

node.getElementsByTagName("标签名");

node : 元素对象 或 文档对象

返回值：调用该函数所得到的东西

会返回 符合指定标签名称的元素节点数组(动态数组)

document.getElementsByTagName("div");

**3、通过class属性值获取元素**

node.getElementsByClassName("class值");

返回值：节点数组

**3、节点间关系**

**1、属性**

parentNode : 获取父节点对象

childNodes : 获取子节点"数组",包含文本节点

children : 获取子节点数组，只包含元素节点，不包含文本节点对象

firstChild : 获取第一个子节点

lastChild : 获取最后一个子节点

previousSibling : 获取上一个兄弟节点

previousElementSibling : 获取上一个元素兄弟节点

nextSibling : 获取下一个兄弟节点

nextElementSibling : 获取下一个元素兄弟节点

**4、修改和读取节点对象**

**1、元素的内容**

**1、HTML内容**

获取或设置的内容，会被解析成HTML内容

属性：elementNode.innerHTML

作用：读取或设置元素节点中的HTML内容

**2、文本内容**

属性：textContent

作用：读取或设置元素节点中的文本内容

属性：innerText

作用：同上，火狐不支持

**2、元素的属性**

**1、elementNode.属性名称**

anchor.href="http://www.jd.com";

**2、元素节点的 属性集合 获取属性**

**属性：**attributes

**用法：**elementNode.attributes;

**返回值：**属性集合对象 NamedNodeMap

**1、读取属性**

1、elem.attributes[下标].value;

2、elem.attributes["属性名"].value;

3、elem.getAttributeNode("属性名").value;

4、elem.getAttribute("属性名");

**2、修改属性**

1、elem.setAttribute(name,value);

2、elem.setAttributeNode(attrNode);

**3、元素的样式**

**1、className**

属性：className

作用：更改 元素的 class 值

**2、style 属性**

DOM 针对不同的样式编写方式，提供了不同的操作方法

**1、内联样式(行内样式)**

**属性：**style

**作用：**获取 或 设置 元素的 行内样式

**语法：**elem.style.属性名

elem.style.color=red;

属性名要经过变化才能用：

background-color

text-align

变化方式：去掉 - ，将-后的首字符变大写

elem.style.backgroundColor="red";

(border-right-color)

elem.style.borderRightColor="#000"

elem.style.border="1px solid #ddd";

注意：如果想设置元素的css属性的话，一律采用 该方式实现

**3、获取计算的样式**

主要针对 内部 和 外部样式表

1、DOM 标准

document.defaultView.getComputedStyle(元素对象).属性名;

2、IE低版本(8,7,6 ...)

元素对象.currentStyle.属性名;

**4、修改样式表中的样式(谨慎使用)**

1、获取 样式表 对象

主要 获取 包含要修改样式的 样式表

var sheet = document.styleSheets[index];

2、获取样式表中所有的样式规则集合

属性：

cssRules : 获取指定样式表中所有的样式规则集合

rules : 同上

var rules = sheet.rules ;

或

var rules = sheet.cssRules;

3、获取规则集合中包含目标属性的规则

var rule = rules[index];

4、获取或设置规则中的目标属性

rule.style.属性名

-----------------------------**JAVASCRIPTCORE**----------------------------

**Day01**

张东

zhangdong@tedu.cn

**1. 控制台(console)的使用:**

1. 单行执行: 输入一行语句，直接回车

2. 多行执行: 输入一行语句后, shift+回车，换行

3. 清屏: 左上角圆形叉号

4. 切换已经执行过的旧代码: 向上/向下

5. 控制台中，可直接输出表达式的结果或变量的值

**2. 数据类型转换:**

**2大类：**

1. 隐式转换: 不需要程序员干预，即可自动完成的转换

2. 强制转换: 由程序员主动调用专门函数执行的类型转换

何时: 如果隐式转换的结果不是想要的

**包含:**

**一切 to string:**

x.toString(): 无法转换null和undefined

String(x): 万能

原理: function String(x){

if(x===undefined){

return "undefined";

}else if(x===null){

return "null";

}else{

return x.toString();

}

}

**一切 to number:**

非字符串 to number: Number(x);

特殊:

Number("")->0

Number(null)->0

Number([])->0

Number(undefined)->NaN

bool to number: true->1 false->0

如果转不了,返回NaN: Not a Number

**2大特点:**  1.参与任何算术计算，结果永远为NaN

2.不大于，不小于，不等于任何值

**字符串 to number:**

**parseInt(str)原理:** 从头开始，跳过开头的空字符

依次读取数字字符

碰到第一个非数字字符退出

**固定套路:** 去掉字符串中数字之后的单位

**问题：**不认识小数点，结果，去掉小数部分

**解决:**

parseFloat(str)原理: 同parseInt

只不过，认识第一个小数点

**特点:** 如果可转为整数，则优先转为整数

无法转为整数，才转为浮点数

**选择:** 除非确实要去掉小数部分，采用parseInt

否则，一律使用parseFloat

**一切 to boolean: Boolean(x);**

**\*\*\*规则:** 只有0,undefined,null,NaN,"",转为false

其余都转为true

**3. 运算符和表达式:**

**程序:** 人的想法，在计算机中的执行

**运算符:** 专门模拟人类想法的特殊符号

**表达式:**  完成一项专门任务的公式: 由变量，运算符，数据组成

所有表达式都有返回值。

**算术运算:** + - \* / %

**%:** 模运算/取余数

比如: m%n: m除以n，取除不尽的余数部分

**何时使用:**  1. 判断能否整除

2. 限制一个运算结果不能超过指定上限

**隐式转换:** 默认: 都转为数字类型(Number(x))，再算术计算

**返回值:** 一定是数字

**特殊:** +运算中，只要碰上字符串，一切都转为字符串。+法计算，变为字符串

拼接。

**固定套路:** x+"";

**返回值:** 一定是字符串

**关系运算:** 作比较

**返回值:** true/false

**包括:** > < >= <= == !=

**隐式转换:**  默认都转为数字(Number(x))，再比较大小

null或引用该类型的对象，则不隐式转换而比较地址值。

**特殊:** 参与比较的两个值都是字符串，不再转数字

而是逐位PK每个字符的unicode号

**NaN: NaN不等于一切，所以普通关系运算无法判断NaN**

**isNaN(num):** 专门判断num是不是NaN

**何时使用:** 只要检查一个数是否是NaN

**null和undefined: 普通的==会将undefined自动转为null**

**=== 全等: 不带隐式转换的==比较**

首先要求类型相同，然后数值相等

**!== 不全等: 不带隐式转换的!=比较**

**何时使用:** 只有不希望隐式转换时

**逻辑运算: 将多个关系运算综合得出最终结论**

**包括:** &&(且) ||(或) !(非)

**条件1&&条件2:** 只有两个条件都满足，才返回true

只要有任意一个条件不满足，就返回false

**条件1||条件2:**  只要任意一个条件满足，就返回true

除非所有条件都不满足才返回false

**!条件:** 颠倒条件的bool结果

**隐式转换:** 每个条件默认都被转为bool做判断

**固定套路:** !!x

**短路逻辑:** 如果前一个条件已经可以得出最终结论，其余条件不再执行。

**&&:** 如果前一个条件为false，则整个结论为false

**||:** 如果前一个条件为true，则整个结论为true

**利用短路逻辑:** 忽略逻辑运算本身的返回结果

**&&:** 简单分支结构: 一个条件，一件事，满足就执行不满足就什么也不做。 **条件&&操作:** 如果满足"条件"，才执行"操作"

否则，什么也不做

**||:** 两个值二选一使用，（默认值）

值1||值2: 如果值1可被转为true(有效),就返回值1

否则返回值2

**位运算: 简化代码量**

左移: m<<n : m\* 2的n次方 m\*Math.pow(2,n)

右移: m>>n : m/ 2的n次方 m/Math.pow(2,n)

取整: m>>>0 m^0 m|0 parseInt(m)

**赋值运算: 将等号右边的值，保存在等号左边的变量中**

**返回值:** 等号右侧表达式的计算结果

也是要存入左侧变量中的值

**简写: 扩展赋值运算:**

一句话做两件事: 1. 取出变量中的值做计算

2. 将结果再存回变量中

m+=n; m=m+n; 累加

m-=n; m=m-n;

m\*=n; m=m\*n;

m/=n; m=m/n;

m%=n; m=m%n;

**简写: 递增递减:**

n++ 递增1 : n+=1 n=n+1

n-- 递减1 : n-=1 n=n-1

++n vs n++

**相同:** n中的值都被+1

**不同:** 返回值: 前++，返回新值

后++，返回旧值

**Day02**

1. \*\*\*函数

2. 分支结构

**1. \*\*\*函数**

**什么是函数:** 封装一项任务的步骤清单的代码段，再起一个名字

**为什么:** 代码重用

**何时使用函数:** 如果一项任务可能被反复调用，都要先封装在一个函数中，再反复调 用函数。

**如何使用: 2步：**

**1. 创建一个函数: 声明方式**

function 函数名(参数变量列表){

函数体;（步骤清单）

return 返回值;

}

**其中:**

**参数变量:** 专门在调用函数时，接收函数必须的数据的变量

何时定义: 只要函数必须某些数据才能正常执行时

**返回值:** 函数执行的结果

何时定义: 只有调用者需要获得执行结果时，才定义

**2. 调用函数:**

var 返回值=函数名(参数值列表);

**其中:** ()是调动函数的标识，不能省

**原则上:** 参数值列表的个数和顺序应和函数定义时的参数变量列表一致。

**函数的本质:**

function： 创建一个函数对象的意思

什么是函数对象: 专门封装一个函数定义的存储空间

其实，函数是一个引用类型的对象

函数名，其实是一个引用函数对象的变量

函数只有在调用时才执行，不调用不执行！

**\*\*\*\*变量作用域(scope):**

**什么是作用域:** 变量的可用范围

**全局作用域:** 凡是直接定义在window范围内的变量都属于全局作用域。全局作用域 （window）中的变量也叫全局变量

**特点:** 随处可用，可反复使用

**函数作用域:** 凡是在函数内部定义的变量都属于函数作用域。函数作用域中的变量也称 为局部变量。

**特点:**  只有在函数调用时可用

不可反复使用

**原因:** 局部变量只有在调用时才创建，每次调用后都清除

**变量使用的规律:** 调用函数时,优先使用局部变量。

只有局部没有时，才去全局找。

如果全局也找不到，才报错

**注意:** 局部变量包括: 在函数内var的变量

参数变量也是局部变量，只不过不用var创建

**\*\*\*\*声明提前（hoist）:** 在正式执行程序前，都会将所有var声明的变量和function 声明的函数提前到\*当前作用域\*的顶部集中创建。

但是，赋值留在原地。

**\*\*\*\*按值传递(byValue):** 两个变量间赋值或将变量作为参数传入函数时，其实都是将 原变量中的值，赋值一份给对方(新变量)。

**对原始类型的值:** 修改新变量，不会影响原变量。

**对引用类型的对象:** 通过新变量修改对象，等效于直接修改原对象。

**2. 全局函数:** \*ES标准规定的\*，浏览器厂商已经实现的不用任何对象，就可直接访问的现 成函数。

**手册:** JavaScript->JavaScript 对象->JS Functions->全局函数列表

**编码和解码:**

**编码:** 将url中的多字节字符，编码为单字节字符

**为什么:** url标准中，不允许出现多字节字符，否则乱码

**比如:** Search?keyword=犀 牛 书

Search?keyword=%E7%8A%80%E7%89%9B%E4%B9%A6

宋 仲 基

%E5%AE%8B%E4%BB%B2%E5%9F%BA

**如何编码:** var code=encodeURI(str)

**解码:** 将编码后的单字节字符，解码回多字节字符原文

**如何解码:** var str=decodeURI(code);

**问题:** url标准中也不允许出现保留字符: 比如: ":" "/"

但encodeURI和decodeURI不能对保留字符编码解码

比如: http://tedu.cn

wd=http%3A%2F%2Ftedu.cn

**解决:** 编码：var code=encodeURIComponent(str)

解码: var str=decodeURIComponent(code);

**eval:** 可执行字符串格式的表达式

注意: 如果传入的字符串语法错误，或无法执行，则报错

**无穷大:** js中，如果除数为0，返回无穷大(infinity)

检测一个数字是否在有效范围内: isFinite(num)

如果返回true，说明num是有效范围内的数字

如果返回false，说明num超出了最大数字的范围

**3. 分支结构:** 让程序根据不同的条件，执行不同的任务

**一个条件，一件事，满足就做，不满足就不做:**

**如果操作简单: 短路: 条件&&(操作1,操作2,...)**

**如果操作复杂: if结构: if(条件){**

操作...

}

**一个条件，两件事，二选一执行:**

**如果操作简单: 三目(三元，条件)运算**

条件?操作1:操作2

**如果根据不同条件，使用不同的值时:**

条件?值1:值2 -> 返回值1或值2

**如果操作复杂: if...else结构:if(条件){**

操作1

}else{

操作2

}

**多个条件，多件事，多选一执行:**

**如果操作简单: 三目运算:**

条件1?操作1:

条件2?操作2:

...?...:

默认操作;

**如果根据不同条件，选择不同的值:**

条件1?值1:

条件2?值2:

...?...:

默认值 -> 最终返回的是一个值

**强调:** 默认操作或默认值不能省略！

**如果操作复杂: else if**

if(条件1){

操作1;

}else if(条件2){

操作2;

}else if(...){

...

}else{

默认操作;

}//最后的else可省略

**如果所有条件都是等值比较: switch case**

switch(表达式){//入口，开关

case 值1:

操作1;

case 值2:

操作2;

case ...:

...;

default:

默认操作;

}//最后一个default可省略

**执行原理:** 先计算表达式的值

再用表达式的值依次和每个case后的值做全等比较

如果表达式的值全等于某个case的值，则进入case中执行操 作。

**问题：** 执行完当前case后，会陆续触发之后所有case中的操作。

**解决:** break 专门中止当前结构的执行

通常用于每个case之间

return, break, 不能嵌入到三目或短路结构中

必须独立一句话使用

如果前后两个case使用相同的操作时，两个case间不用加break;

**switch case vs if else**

如果所有条件都是等值比较，首选switch case

其余都选if else

**分支结构中的隐式转换:** 默认将所有条件都转为bool，再判断。

**补:**

**创建一个函数: 2种:**

**1. 声明方式:** 可被声明提前

function 函数名(参数列表){函数体; return 返回值}

**2. 以函数直接量方式创建：**不会被提前

var 函数名=function(参数列表){函数体; return 返回值

说明: 函数名其实就是一个变量

函数其实就是一个对象

函数名变量保存了函数对象的地址，是引用的关系

**作用域:**

**js中:** 只有2级作用域: 全局作用域和函数作用域

没有块作用域

**Java中:** 有3级作用域: 全局作用域，函数作用域，块作用域{ }

**函数 vs 方法**

**相同:** 本质都是function

**差别:** 函数不属于任何对象

方法指特定对象内的函数

**Day03**

1. 循环:

2. 数组

**1. 循环: 程序反复执行同一代码段**

**何时使用:** 反复执行同一代码段

**如何使用:**

**三要素: 1. 循环条件:** 允许再次\*进入\*循环的条件

**2. 循环变量:** 循环条件中用作比较的变量

从?开始，到?结束，每次递增/减?

**3. 循环体:** 要反复执行的同一代码段

**如果循环变量的变化没有规律: while**

var 循环变量=初始值;

while(循环条件){

循环体;

迭代变化循环变量;

}

**do-while: 先执行一次循环体，再判断条件**

**何时使用:** 如果第一次条件都不满足，也希望至少执行一次时

**如何使用:**

var 循环变量=初始值;

do{

循环体;

迭代变化循环变量;

}while(循环条件);

**鄙视:** vs while： 只看第一次条件是否满足:

如果第一次条件满足，则两者效果完全相同！

如果第一次条件不满足，

do while至少可以执行一次

while一次都不执行

**如果循环变量的变化有规律: for**

for(var 循环变量=初始值;循环条件;迭代变化循环变量){

循环体;

}

**for的特殊用法:**

1. 第一部分可同时声明并初始化多个循环变量,也可省略或移到循环之前外 部。

2. 第三部分可同时执行多个短小的操作，用逗号分隔，也可省略，还可以放 在循环内部循环体之后

3. 死循环: for( ; ; ) while(true)

**退出循环: 2种办法：**

1. 使用循环条件: 优雅，要求高——建议

2. 使用break，随时手动退出循环: 野蛮，简单

**continue:** 跳过本轮循环，继续下一轮。

**1. \*\*\*数组:**

**什么是:** 连续存储多个数据的存储空间，再起一个统一的名字

多个变量的集合，起一个统一的变量名

**为什么:** 程序=数据结构+算法

**算法:** 解决问题的步骤

**数据结构:** 数据在内存中的存储结构

好的数据结构，可极大提高程序的执行效率

**何时使用:** 今后只要同时存储多个相关数据，就要用数组

**如何使用:**

**创建数组: 3种情况:**

**1. 创建一个空数组:**

1. 数组直接量: var arr=[];//创建新数组

2. 使用new: var arr=new Array();

**2. 创建数组同时，初始化元素:**

1. 数组直接量: var arr=[值1,值2,...]

2. 使用new: var arr=new Array(值1,值2,...);

**3. 创建指定个数个空元素的数组:**

1. 使用new: var arr=new Array(n)

**数组的存储方式:**

每个元素都是连续存储

每个元素都有唯一的序号——下标

数组下标都是从0开始

只要访问数组中指定位置的元素，都必须用下标

**将数据存入数组: arr[i]=值**

将"值"存入数组arr的下标为i的元素中

**特殊:** 如果i越界，js会自动在i位置创建新元素，保存值

会造成下标不再连续——稀疏数组

数组中的元素不限制存入的数据类型

**获取数组元素值: arr[i] —— 用法和普通变量完全一样**

**特殊:** 如果i越界，则返回undefined

**数组的length属性:** 记录了理论上的元素个数，但无法反应实际的元素个数。

**强调:** length属性永远等于最大下标+1

**固定套路:** 1.最后一个元素: arr[arr.length-1]

2.在末尾追加一个新元素: arr[arr.length]=值

3.缩容: arr.length=更小的值

**数组是引用类型的对象。**

**垃圾回收:** 释放不再被任何变量引用的对象

**垃圾回收器:** 专门记录对象的引用次数，并回收不再被引用的对象的程序。

垃圾回收器和主程序并行在后台执行

垃圾回收器会为每个对象创建一个引用计数器(counter)

只要多一个变量引用对象，counter就会被+1

只要一个变量释放了对对象的引用，counter就会被-1

不再有任何变量引用该对象时(counter=0)，对象被垃圾回收器释放。

**null专用于主动释放一个对象。**

建议，在使用完一个较大的对象后，都要主动释放

**遍历数组:** for(var i=0;i<arr.length;i++){

arr[i]//当前元素

}

**数组分类：**2大类:

**索引数组:** 下标都是数字的数组

**关联(hash)数组:** 可自定义下标名称的数组

[杨幂, 81, 59, 89];

name math chs eng

**如何创建关联数组: 2步:**

1. 先创建空数组:

2. 再向数组中添加新元素，同时自定义下标名称

arr["下标名"]=值;

key value

**如何访问数组元素:** arr["下标名"]

关联数组length属性失效，永远为0

**遍历关联数组: for in**

for(var key in arr){

key//当前元素的下标

arr[key]//当前元素

}

**其中in:** 依次获取关联数组中每个下标，保存到左边的变量中。

**key:** 保存的是当前元素的下标

**关联(hash)数组的原理:**

**hash算法:** 接收一个字符串，计算出一个尽量不重复的序号

不同的字符串，计算出的序号尽量不同

相同的字符串，计算出的序号一定是相同

**存入数据时:** 将自定义下标名称交给hash算法，计算出一个尽量不重复的序 号。将要保存的数据，存入序号对应的位置。

**获取数据时:** 将要查找的下标名称交给hash算法，再计算出和存入时一模一 样的序号，直接去数组指定位置获取元素内容。

**优点:** 查找速度极快，与数据量和存储位置无关！

**数组API:**

**1. 将数组转化为字符串:2种:**

1. var str=String(arr): 将数组转为字符串并用逗号分隔每个元素

2. var str=arr.join("自定义分隔符"): 将数组转为字符串,可自定义分隔符

**强调：**如果join省略""，就等效于String了

**固定套路:** 1. 将单词数组，拼接成句子: arr.join(" ")

2. 无缝拼接: arr.join("")

3. 将数组转化为页面元素: 2步:

Step1: 用join拼接</结束标签><开始标签>

Step2: 再拼接开头和结尾

**2. 连接和获取子数组:**

**连接:** var newArr=arr1.concat(值1,值2,arr2,...)

将concat后的内容， 和arr1拼接，组成新数组返回

**强调:** concat不修改原数组，而是返回新数组

如果concat的参数中包含数组，则打散数组，以单个元素拼接

**获取子数组:** var subArr=arr.slice(starti,endi+1);

获取arr中starti位置开始，到endi位置的所有元素组成新数组

**强调:** 含头不含尾

其实，第二个参数可省略，表示从starti一直获取到结尾

**可支持负数参数:** -n表示倒数第n个元素，相当于length-n

**注意:** 即使使用负数参数，也必须遵守含头不含尾

**3. splice: 删除，插入，替换**

**强调:** 直接修改原数组

**删除:** var deletes=arr.splice(starti,n)

删除arr中starti位置开始的n个元素

返回被删除的元素组成的临时新数组

**插入:** arr.splice(starti,0,值1,值2,...)

在arr中starti位置，插入新值。旧值被向后顺移

**强调:** 要插入的值，只能以单独的参数传入，不支持打散数组参数

**替换:** var deletes=arr.splice(starti,n,值1,值2,...)

删除arr中starti位置开始的n个元素，再在starti位置插入新元素。

删除的元素个数和插入的新元素个数不必相等

1. **反转数组元素: arr.reverse( );**

**Day04**

**1. \*\*\*数组:**

**\*\*\*排序:**

**手动排序：**冒泡排序

**使用API：**arr.sort(); //直接修改原数组

**特点:** 将arr中的元素，以字符串方式\*升序\*排列

**问题:** 只能对字符串排序

如果元素不是字符串，会被隐式转为字符串

**解决:** 自定义排序规则:

何时自定义排序？只要对非字符串类型的元素排序

如何自定义排序规则: 2步:

**1. 自定义比较器函数:**

比较器函数: 专门比较任意两值大小的函数

要求: 2个参数: a,b

返回值: 如果a>b，就返回正数

否则如果a<b，就返回负数

否则，就返回0

function cmp(a,b){return a-b;}

**2. 将比较器函数传入sort方法作为参数:**

arr.sort(cmp)

**\*\*\*强调:** 将函数作为参数传入另一个函数时不加( )。

**其它时候:** 如果另一个函数，需要补充一段逻辑才能正常执行时， 就要将逻辑封装在一个函数中，再作为参数传入。

**降序排列:**

1. 自定义比较器: 只要颠倒比较器函数的比较逻辑

2. 默认字符串比较: 先sort升序，再reverse反转

**栈(stack)和队列(queue):**

**什么是栈:** 一端封闭，只能从另一端进出的数组

**特点:** FILO

**何时使用:** 如果希望总是使用最新进入数组的元素时

**如何使用:**

**1. 结尾出入栈：**

**入栈:** arr.push(值)

将值压入数组结尾

**出栈:** var last=arr.pop();

弹出数组最后一个元素

**优:** 每次出入栈，不影响其余元素的位置

**2. 开头出入栈:**

**入栈:** arr.unshift(值);

将值插入数组开头

**出栈:** var first=arr.shift();

取出数组第一个元素

**缺:** 每次出入栈，其余元素的位置都要顺移1

**队列: 只能从一端进入，从另一端出的数组**

FIFO

**何时使用:** 只要希望按照进入数组的顺序使用元素时

**如何使用:**

**入:** arr.push(值)

**出:** var first=arr.shift();

**二维数组:** 数组中的元素又引用了另一个子数组

**何时使用:**  1. 专门存储横行竖列的二维数据

2. 将一个大数组的元素，再进行分类时

**如何使用:**

**创建: 2种:**

**1. 如果创建时，就初始化元素:**

var arr=[

//0 1

[值1,值2,...], //0

[值1,值2,...], //1

... ...

]

**2. 先创建空数组，再添加子数组:**

var arr=[ ];

arr[0]=[值1,值2,...];

arr[1]=[值1,值2,...];

**访问元素: arr[r][c]**

**强调: c越界:** 返回undefined

**r越界:** 报错！

**遍历:** 外层循环遍历行，内层循环遍历列

for(var r=0;r<arr.length;r++){

arr[r]->当前行(子数组)

for(var c=0;c<arr[r].length;c++){

arr[r][c]//当前元素?

}

}

**2. \*\*\*String:**

**内置对象:** ES标准中规定的，浏览器厂商已经实现的对象

**11个:** （String Number Boolean） 包装类型

Array Date Math RegExp

Error

Function Object

Global (在浏览器中被window对象替代)

**包装类型:** 专门封装原始类型的值，并提供对原始类型值进行操作的方法的对象。

**为什么包装类型:** 原始类型的值本身不包含任何属性和方法

但是，经常需要对原始类型的值调用方法或访问属性

**何时使用包装类型:** 不用主动使用

**生命周期:** 当试图用原始类型的值调用方法时，会自动创建包装类型的对象。封 装原始类型的值。

调用的方法，其实是包装类型对象的

当调用完方法后，包装类型对象自动释放

**为什么null和undefined不能调用任何方法:** 因为没有对应的包装类型。

**什么是字符串:** 由多个字符组成的一个只读字符数组

**特点:** 一旦创建不允许修改

**vs 数组:** 1. 下标访问每个字符

2. 都有length属性记录字符个数

3. 不直接修改原数组的API都可用

concat slice

**String API:**

\*\*\*所有字符串API都无法直接修改原字符串，只能返回新字符串

**1. 大小写转换:**

**何时使用:** 只要不区分大小写时，都要先转换为一致的大小写，再比较

**如何转换:**

var newStr=str.toUpperCase();

var newStr=str.toLowerCase();

**2. 获取指定位置的字符: 2种:**

str[i] 或 str.charAt(i);

获得指定位置字符的unicode号:

var num=str.charCodeAt(i);

将unicode号，反向还原为原字符

var char=String.fromCharCode(num);

**Day05**

正课:

1. \*\*\*String API:

查找关键字

替换

切割

2. \*\*\*\*\*正则表达式

**1. \*\*\*String API:**

**1. 获取子字符串:**

var subStr=str.slice(starti,endi+1);

str.substring(starti,endi+1)

substring不支持负数参数

解决: -n换位str.length-n

str.substr(starti,n);

特点: 不考虑含头不含尾

**2. 查找关键词:**

**1. 查找一个固定的关键词的位置:**

var i=str.indexOf("关键词"[,fromi]);

从fromi位置开始，查找下\*一个\*关键词的位置

如果省略fromi，表示从头开始找(默认为0)

返回值: 如果找到，返回关键词第一个字所在的位置下标

如果没找到，返回-1

var i=str.lastIndexOf("关键词"[,fromi])

从fromi位置开始，查找前\*一个\*关键词的位置

如果省略fromi，表示从结尾开始找(默认为str.length-1)

返回值:都一样

**缺点: 只能查找固定关键词**

**2. 利用正则表达式查找关键词:**

**2. \*\*\*\*\*正则表达式:**

**什么是:** 规定字符串中字符出现规律的一套规则

**何时使用:** 查找: 只要按照字符出现的规律，模糊匹配关键词时

验证:格式验证

**如何使用:**

**1. 最简单的正则就是关键词的原文**

**2. 字符集: 规定一位字符可选的备选字符列表**

**何时使用:** 只要字符串中一位字符，有多种情况可选时

**如何使用:** [备选字符列表]

**强调:** 一个字符集，只匹配一位字符

**简写:** 如果备选字符列表中部分字符unicode连续

可用-省略中间的字符:

**固定套路:** 1. 一位数字: [0-9]

2. 一位小写字母: [a-z]

3. 一位字母数字或下划线:

[0-9A-Za-z\_]

4. 汉字: [\u4e00-\u9fa5]

**排除:** 除了制定的字符，都行: [^排除的字符]

比如: 除了4和7: [^47]

**慎用:** 因为排除的范围，太大

**3. 预定义字符集: 对常用字符集的简写方式**

1. 一位数字: [0-9] 简写为\d

2. 一位字母数字或下划线: [0-9A-Za-z\_] 简写为 \w

3. 空字符: 空格，换行，制表符... 简写为 \s

4. 除换行外的所有字符: .

**缺点: 固定的。**

**今后:** 如果需求和预定义字符集完全匹配，就优先使用预定义字符集。

如果需求和预定义字符集不匹配，指定自定义字符集

**4. 数量词: 规定一个字符集出现次数的规则**

默认仅修饰相邻的前\*一个\*字符集

有明确数量: {n,m} 规定一个字符集最少出现n次，最多m次

{n,} 规定至少n次，多了不限

{n} 规定必须n次

没有明确数量: ? 可有可无 最多一次

\* 可有可无，多了不限

+ 至少一次，多了不限

**5. 选择和分组:**

**分组:** 将多个字符集，分为一组

**为什么:**  1. 希望一个数量词，同时修饰多个字符集时

2. 分段获取关键词的子内容

**选择:** 或 | 满足前后两个规则之一即可

**强调:** |的优先级，最低

手机号:

+86或0086 可有可无，最多1次 (\+86|0086)?

空字符，可有可无，多了不限 \s\*

1

在3,4,5,7,8中选一个 [34578]

9位数字 \d{9}

**6.匹配特殊位置:**

**匹配字符串中的特殊位置:**

**^ 指代字符串的开始位置: 以xxx开头**

几乎总是写在正则的开始位置

**$ 指代字符串的结尾位置: 以xxx结尾**

几乎总写在正则的结尾位置

**如果同时前加^，后加$——要求从头到尾必须完整匹配**

凡是验证，都要加^和$

**\b 单词边界: 开头和结尾，标点，空字符**

**7. 预判: 提前扫描整个字符串，判断是否满足部分规则**

密码强度: 6-8位，字母，数字的组合。要求至少包含一个大写字母和一位数字

1. 预判不全由字母组成: 说明可能包含数字或特殊符号

(?![A-Za-z]+$)

2. 预判不全由小写字母和数字组成:

说明可能包含大写字母或特殊符号

(?![a-z0-9]+$)

3. [A-Za-z0-9]{6,8}

完整: (?![A-Za-z]+$)(?![a-z0-9]+$)[A-Za-z0-9]{6,8}

**查找关键词: 4种:**

**1. 查找一个固定的关键词位置:**

var i=str.indexOf("关键词",fromi);

var i=str.lastIndexOf("关键词",fromi);

**缺点: 只能查找固定的一个关键词**

**不支持正则**

**补: 在js中创建正则表达式:**

var reg=/正则表达式/[ig];

创建一个正则表达式对象。

其中，i表示忽略大小写，g表示查找全部

**2. 判断是否包含关键词: 判断有没有**

var i=str.search(reg);

在str中查找第一个符合正则表达式要求的关键词的位置

没找到，返回-1。

**缺点:** 1. 只返回位置，无法返回关键词内容

2. 只能找第一个，无法查找所有，不认g

**3. 查找所有关键词的内容:**

**var kwords=str.match(reg);**

**1. reg不加g，match也只找第一个**

**何时使用:** 希望获得第一个关键词的内容和位置时

**返回值:** 数组，其中包含找到的关键词内容

kwords:["关键词"]

kwords.index->关键词的位置

如果没找到，返回null！

**2. reg加g，match会找到所有关键词的内容**

**何时使用:** 希望获得所有关键词的内容时

**返回值:**  数组，包含了所有关键词的内容

数组的length属性，记录了所有关键词个数

**缺点:** 只能找到所有关键词的内容，无法找到每个关键词的位置。

**强调:** 只要使用match，都可能返回null，都要先判断不是null，再执行后续操作！

**4. reg.exec(str)->即查找每个关键词的内容，又查找每个关键词的位置。**

**替换关键词:**

**基本:**

str=str.replace(reg,"替换内容")

**强调:** replace如果不加g，也只能替换第一个关键词

通常都加g

**高级:**

str=str.replace(reg,function(kword,$1,$2,...){

//kword获得当前找到的一个关键词

//$n，获得正则中第n个()分组的子内容

...

return 替换内容;

});

如果今后希望对找到的关键词分段处理时，都要先将正则表达式用()分组。

**删除: 替换为""**

**格式化: 将字符串替换为具有指定格式的内容**

2步: 1. 将正则表达式，按格式要求分组

2. 在replace时，使用$n指代第n个分组的子内容

**切割: 将字符串按照指定的分隔符切割为子字符串的数组**

**何时切割:** 只要希望分段获得字符串的子内容时

**如何切割:** var subStrs=str.split("分隔符"/reg);

**返回值:** 包含切割后所有子字符串的数组

**强调:** 结果数组中不再包含分隔符

**固定套路:** 将str打散成单个字符的数组: str.split("");

**Day06**

**\*\*\*\*\*正则表达式:**

1. 最简单的正则表达式: 关键词原文

2. 字符集: [备选字符列表]

如果字符列表部分连续: 可用-省略中间字符

如果表示除了个别字符: [^xxx]

**3. 备选字符集:**

一位数字 \d 一位字母数组或下划线: \w

一位空字符 \s 任意一个字符: .

**4. 量词:**

**有明确数量:**

字符集{m,n} 至少m次，最多n次

字符集{m,} 至少m次，多了不限

字符集{m} 必须m次，不能多不能少

**没有明确数量:**

字符集? 可有可无，最多一次

字符集\* 可有可无，多了不限

字符集+ 至少一次，多了不限

**5. 选择和分组:**

**选择:** 或 |

**分组:** ( ) 每个分组在匹配时，自动获得一个序号，从1开始

2种用途: 1. 在正则内部\n，匹配和第n个分组相同的内容

2. js中使用$n获得关键词的第n个分组子关键词

**6. 指定匹配位置:**

**^:** 字符串的开始位置

**$:** 字符串的结束位置

**\b:** 单词边界, 默认仅匹配一边。

如果匹配独立单词，要两边同时加\b

**密码强度:** 8位字母数字的组合，至少包含一位大写字母和一位数字

1. 不都是字母: 至少有一个数字或特殊符号

(?![a-zA-Z]+$)

2. 不都是小写子母和数字: 至少有一个大写字母或特殊符号

(?![a-z0-9]+$)

3. 只能由字母和数字组成:

[a-zA-Z0-9]{8}

**结果：** ^(?![a-zA-Z]+$)(?![a-z0-9]+$)[a-zA-Z0-9]{8}$

**\*\*\*RegExp:**

专门封装一个正则表达式，提供用正则表达式执行查找和验证的API 的对象

**1. 创建:**

**1. 直接量:** var reg=/正则/ig;

**何时使用:** 如果正则是固定的，不需要动态拼接

**转义:** 如果包含/,都要转为\/

**2. 使用new关键字:** var reg=new RegExp("正则"[,"ig"]);

**何时使用:** 如果正则不是现成的，需要运行时拼接

**转义:** 如果包含"或'，都要转为\"和\'

如果包含\d,\w,\s，必须写成\\d,\\w,\\s

**2. API:**

**1. 查找关键词：**查找每个关键词的内容和位置

var arr=reg.exec(str);

**意为:** 在str中查找下\*一个\*和reg匹配的关键词

**返回值: arr:** [kword,$1,$2,...]

0

**arr.index:** 保存了本次找到的关键词的位置

如果没找到返回null

**原理:** 默认从开头查找下一个关键词

如果找到，就将将关键词保存在结果数组的第一个元素

将关键词位置保存在结果数组的index属性中

exec自动将reg的lastIndex调整到本次找到的关键词之后

reg.lastIndex： 标识正则下次匹配的开始位置

贪婪模式: 正则表达式默认匹配最长的符合条件的字符串

比如: .\* .+

懒惰模式: 让正则仅匹配最短的符合条件的字符串

贪婪->懒惰: (.\*?) (.+?)

**1. \*\*\*RegExp:**

**什么是:** 封装一条正则表达式，提供用正则表达式执行查找和验证的功能

**何时使用:** 按规则，查找，替换，切割，验证字符串时

**如何使用:**

**创建:** 1. 直接量: var reg=/正则表达式/ig;

**何时使用:** 如果正则表达式是提前固定好的

**缺点:** 无法在运行时动态生成正则表达式

**转义:** 如果正则中包含/，需要换为\/

2. 用new: var reg=new RegExp("正则表达式"[,"ig"]);

**何时使用:** 如果正则表达式需要在运行时，动态拼接

**转义:** " -> \"

\d,\s,\w -> \\d \\s \\w

**API:**

查找: 4种:

**String: 3种:**

**1. 查找固定的一个关键词的位置:** 不支持正则

var i=str.indexOf("关键词",fromi);

var i=str.lastIndexOf("关键词",fromi);

**2. 只判断有没有关键词:** 只能返回第一个关键词的位置

无法返回内容

var i=str.search(reg); //不支持g

**3. match:**

**1. 不加g:** 查找第一个关键词的位置和内容

**缺:** 只能找第一个

var arr=str.match(reg);

arr: [kword,$1,$2,...].index

**2. 加g:** 查找所有关键词的内容: 无法获得每个位置

var arr=str.match(reg);

arr: [kword1,kword2,...]

**4. RegExp:**

**即可查找每个关键词的内容，又可获得每个位置**

var arr=reg.exec(str)

**强调:** 一般都要加g

如果不加g，等效于str.match(reg)

依次查找str中符合reg要求的一个关键词

**3件事:**

**1. 将本次找到的关键词保存在数组第一个元素**

arr:[kword,$1,$2,...]

**2. 将本次找到的关键词的位置保存在数组的index**

arr.index

**\*3. 将reg对象的lastIndex属性设置为index之后**

reg.lastIndex: 规定下次从哪个位置开始

**正则有两种匹配模式:**

**1. 贪婪模式:** 默认情况，

正则表达式匹配最长的符合条件的子内容

**2. 懒惰模式:** 让正则表达式仅匹配最短的符合条件的子内容

贪婪改懒惰: .\*? .+?

**验证:** 只要用正则表达式验证字符串的格式，都用reg.test

**如何使用:** var bool=reg.test(str);

str符合reg的规则，就返回true，否则返回false

**强调： 问题:** 默认只要部分子字符串符合规则，就通过！

**解决:** 同时前加^,后加$ 要求从头到尾必须完全匹配

今后凡是验证格式，都要加^和$，同时不加g 。

**1. Math对象:**

**什么是:** 封装数学计算所需的常量，提供数学计算的函数

**何时使用:** 只要进行数学计算，都要用Math对象

**如何使用:** Math对象不能new！而是直接Math.属性/方法()

**API:**

**1. 取整:**

**上取整:** 只要超过当前整除，就取下一个整数

Math.ceil(num)

**下取整:** 舍弃小数部分，只保留整数部分

Math.floor(num);

**四舍五入取整:** 够5进位，不够5舍弃

Math.round(num);

vs num.toFixed(d)

**不同:** Math.round(num) 只能取整！

num.toFixed(d) 可按任意小数位数四舍五入

**强调:** d的范围是0~20之间

**返回值:** Math.round(num)->number

num.toFixed(d)->string

自定义一个按任意小数位数四舍五入，且返回number的函数

**2. 乘方和开平方**

**Math.pow(底数,幂):** 5的3次方: Math.pow(5,3)

**Math.sqrt(num):** 将num开平方

**3. 最大值和最小值:**

**Math.max(a,b,c,...);** 获得参数中的最大值

**Math.min(a,b,c,...);** 获得参数中的最小值

**问题:** max和min不支持获取数组中的最大值

**解决:** Math.max.apply(null,[a,b,c,...]);

**apply:**调用函数，并。。。

同时将数组参数，打散后传递给之前的函数

**运行时:** Math.max(a,b,c,...)

**4. 随机数:** Math.random(): 0<=r<1

**在min~max之间取随机整数:**

Math.floor(Math.random()\*(max-min+1)+min);

60 100

**简写:** 在0~max之间取随机整数

Math.floor(Math.random()\*(max+1));

**2. Date对象:**

**什么是:** 封装一个时间点，提供对时间计算的API

**何时使用:** 只要操作日期时，都要用Date对象

**创建:** 3种

**1. 创建一个日期对象，同时自动获得当前客户端系统时间:**

var now=new Date();

**2. 创建一个日期对象，同时封装自定义时间点**

var date=new Date("yyyy/MM/dd hh:mm:ss");

var date=new Date(yyyy,MM-1,dd,hh,mm,ss);

**3. 用毫秒数创建一个日期对象:**

var date=new Date(ms);

**复制一个日期对象:**

**为什么:** 日期的计算会直接修改原日期对象

**何时复制:** 只要希望同时保存计算前和计算后的两个时间

都要先复制一份，再计算

**如何复制:**  var copy=new Date(date);

**原理:** 其实一个日期对象中保存的是一个巨大的毫秒数

从1970年1月1日0点至今的毫秒数

**API:**

**8个分量:**

FullYear Month Date Day 都没有s结尾

Hours Minutes Seconds Milliseconds 都有s结尾

**1. 每个分量都要一对儿get/set方法:**

其中date.getXXX()获得指定分量的值

date.setXXX(n)设置指定分量的值

但是,Day只有getDay()，没有setDay()

**2. 返回值:**

Date:月中的日，从1~31

其余都是从0~进制-1结束

\*Month: 0~11 -> 计算机中的月份值比现实小1，需要修正

Day: 0~6

Hours: 0~23

Minutes/Seconds: 0~59

**计算:**

1. 两日期对象可相减，结果是毫秒差

2. 基于一个日期，推导另一个日期:

**3步:** 1. 取分量 2. 做加减 3. 放回去（自动换算进制）

**简版:** date.setXXX(date.getXXX()+/-n)

**输出:**

**API:**

date.toString(); //将date输出为国际标准时间格式

date.toLocaleString();//输出为当地完整标准时间格式

date.toLocaleDateString();//当地时间格式,仅包含日期

date.toLocaleTimeString();//当地时间格式,仅包含时间

**问题:**以上API可能造成浏览器不一致

**解决:**自定义format方法，格式化任意一个日期对象

**Day07**

**1. 错误处理:**

**什么是错误:** 程序在执行过程中发生的异常中断

**错误对象:** 专门封装错误信息的对象

**创建:** 在发生错误时，自动创建

**类型:** 6种:

SyntaxError: 语法错误:

ReferenceError: 引用错误: 找不到变量

TypeError: 类型错误: 错误的使用了类型的API

RangeError: 范围错误: 函数的参数超范围

EvalError: Eval函数使用错误

URIError: 不符合URI的使用标准

**错误处理:** 当程序发生错误时，保证程序不异常中断的机制

**程序结构:**

try{

有可能发生错误的代码段

}catch(error){

只有出错时才执行的代码

}[finally{

无论是否出错都一定执行的代码

}]

**鄙视题:** try catch中的return

**前提:** finally中的代码，一定会被执行

1. 如果finally中有return，则try catch中的return失效

2. 如果finally中没有return，finally中的代码一定也会执行，但是不会影响之前 已经确定的返回值

**关于效率:**

**问题:** 只要放入try catch结构中的代码，即使正常执行，效率也比普通代码 低。

**解决:** 绝大多数try catch其实都可提前用if else预防

**何时使用:** 可提前预判错误的原因，就用if...else预防

如果无法预判可能发生的错误，就用try catch

**还可解决浏览器兼容性问题:** 见课堂练习

**主动抛出异常:**

**何时主动抛出:** 在团队合作中，函数的定义者会向调用者抛出自定义错误， 来提醒调用者如何正确使用函数。

**如何抛出:** throw new Error("错误信息")

**2. \*\*\*\*\*Function:**

**什么是:** 封装一个函数定义，并提供对函数操作的API

**创建函数:** 3种:

**1. 声明:** 只有声明方式创建的函数才能被声明提前

function fun(arg1,arg2,...){statements; return value}

**2. 函数直接量:** 无法被声明提前

var fun=function(arg1,arg2,...){statements; return value}

**揭示:** 函数其实是一个对象，函数名仅是一个引用函数对象的普通变量。

**3. 用new:**

var fun=new Function(

"arg1","arg2",...,"statements; return value");

**比如:** 定义比较器函数:

**\*\*\*重载(overload):**

**什么是重载:** 相同函数名，不同参数列表的多个函数

在调用时，可根据传入参数的不同，自动选择匹配的函数执行

**为什么:** 减少API的数量，减轻调用者的负担

**何时使用:** 如果多个函数只是流程不同，其实都是一类事儿时

就可将多个函数明明为同名函数，定义不同的参数

**如何使用:** js的语法不支持重载！

**为什么:** js中后定义的同名函数会覆盖先定义的

**变通:** arguments:

**什么:** 函数中自动接收所有传入函数的参数值的类数组对象

**什么是类数组对象:** 长的像数组的\*对象\*!

**vs 数组:** **相同:** 1. 都用[i]方式访问每个元素

2. 都有length属性记录元素个数

3. 都可用for循环遍历每个元素

**不同:** 类型不同，无法使用数组原生API

**创建:** 不用创建，本来就有！

**匿名函数:** 创建函数时，不指定函数名的函数

**为什么:** 节约内存

**何时使用:** 如果一个函数只使用一次，不会重用时，就必须用匿名函数

**如何使用:** 2种:

**1. 回调:** 将一个函数，传递给另一个函数去调用

**2. 自调:** 创建一个函数后，立刻调用，调用完，立刻释放！

**\*\*\*\*\*函数的生命周期:**

**程序开始执行时:**

1. 创建执行环境栈: ECS

2. 创建全局作用域对象: window

3. 在ECS中压入第一个元素:全局EC 引用 window

**定义函数时:**

1. 创建函数对象，封装函数定义

2. 声明全局变量，应用函数对象

3. 在函数对象中，添加scope属性指向创建函数的作用域

**调用前:**

1. 向ECS中压入一个本次函数调用的EC

2. 创建一个活动对象AO，封装所有局部变量

3. 让本次函数调用的EC引用活动对象AO

4. 让AO再指向函数的scope属性引用的对象

**调用中:**

**变量使用规则:** 只要AO中有，就不用window的！

除非AO中没有，才去window找

**调用后:**

1. 本次函数的EC出栈

2. 本次函数的活动对象AO释放

3. 同时AO中的局部变量一起释放

**作用域和作用域链:**

**作用域:** 变量的可用范围，其实就变量的存储位置

**只有2种:** **全局作用域：**window对象

**函数作用域：**AO对象

**作用域链:** 当前函数的EC中的scope chain属性应用AO对象

AO对象再逐级引用，直到window结束

形成的链式结构

**用途:** 保证如果AO中没有变量，可继续延作用域链继续到window找

**\*\*\*\*\*闭包(closure):**

**问题: 全局变量: 优:** 可反复使用，随处可用

**缺:** 易被全局污染

**局部变量: 优:** 不会被污染

**缺:** 不可重用！

**解决:** **闭包:** 既重用，又保护局部变量的机制

**创建:3步:**

1. 用外层函数封装受保护的局部变量和操作局部变量的内层函数。

2. 外层函数将内层函数对象返回到外部

3. 全局范围，调用外层函数，获得并保存返回的内层函数对象

**鄙视:** 1. 先找受保护的局部变量，

\*判断外层函数调用后变量值是几

2. 同一次外层函数调用返回的多个内层函数，共享同一个变量。

返回内层函数的多种方式:

1. 直接return

2. 直接给全局变量赋值

3. 将函数装在数组或对象中返回

3. 每多调用一次外层函数，都会再创建一个闭包和受保护变量的副本。

**Day08**

**\*\*\*\*\*面向对象: OOP**

**什么是对象:** 内存中同时存储多个数据的存储空间，再起个名字

程序中专门描述现实中一个具体事物的结构

**名称:** 属性 : 值

...:...

功能:...

**什么是面向对象:** 一切皆对象！

先用对象描述出现实中的具体实物

再根据需要调用对象中的属性和功能

**为什么面向对象:** 更符合现实中的思考方式。

**\*\*\*\*\*面向对象: 封装 继承 多态**

**如何使用面向对象:**

**1. 封装:** 将现实中一个具体实物的属性和功能集中定义在一个对象中。—— 创建一个对象！

**如何创建:** 事物的属性，会成为对象中的属性变量

事物的功能，会成为对象中的方法函数

**3种:**

**1. 仅创建一个单独的对象**

**1. 对象直接量:** var obj={//创建新对象

**属性名:**属性值,

...:...,

**方法名:**function(){...},

...:...

}

**强调:** 每个属性和方法之间必须用逗号分隔！

每个属性和方法名不用加引号

但是底层都是用字符串存储

**如何访问对象中的属性和方法:** 都用.访问！读作"的"

**比如:** 对象.属性 -> 和普通变量完全一样

对象.方法() -> 和普通函数完全一样

**\*\*\*\*\*this: 引用当前正在调用函数的对象！**

**简单说，就是.前的对象**

**何时使用:** **对象自己的\*方法中\*，想访问对象自己的属性**

**必须用this.属性名**

**2. 使用new: var obj=new Object();//{}**

//创建一个Object类型的对象(空对象)

obj.属性名=值;//等效于obj["属性名"]

......

obj.方法名=function(){...this.属性...}

**何时使用:** 如果在创建对象时，暂时不知道都包含哪些属性， 就先创建一个对象，再添加属性。

**揭示:** js中的对象，可随时添加新属性

只要给不存在的属性赋值，都自动添加。

**\*\*\*\*\*其实，js中一切对象都是hash数组:**

克隆一个对象: 见课堂练习

**2. 反复创建多个相同类型和结构的对象:**

**2步:**

**1. 定义构造函数，描述一类对象的统一结构**

**什么是构造函数:** 描述一类对象的统一结构的函数

**何时使用:** 只要反复创建相同结构的多个对象前，都需要先定义构造 函数，再创建对象。

**如何定义:**

function 类型名/构造函数名(属性参数列表){

this.属性名=属性参数值;

...=...;

this.方法名=function(){

...this.属性名...

}

}

**2. 使用new调用构造函数创建对象并添加统一的属性**

var obj=new 构造函数名(属性值列表);

构造函数中的this，指new正在创建的对象

**new: 4件事:**

1. 创建一个新的空对象

2. ......

3. 调用构造函数，为新对象添加规定的属性

4. 将新对象地址返给变量保存

**优:** 代码重用！

**缺:** 并没有节约内存！

**\*\*\*\*\*原型和原型链:**

**什么是原型:** 保存所有子对象的共有成员的父级对象

成员=属性+方法

**如何访问:**

每个构造函数，都有一个prototype属性，引用自己的原型对象

每个子对象都有一个\_\_proto\_\_属性，继承自构造函数的原型对象

**2、继承:** 父对象中的成员，子对象无需创建，可直接使用！

**回顾: new的4件事:**

1. 创建空对象

2. 设置新对象的\_\_proto\_\_属性继承构造函数的原型对象

3. 调用构造函数，添加成员

4. 返回地址

**何时使用:** 同一类型的子对象共享的属性值和方法,都要放在构造函数的原型对 象中。

**优点:** 代码重用! 节约内存!

**缺点:** 易被污染

**为对象扩展属性:**

**1. 自有属性:** 直接保存在对象本地的属性

**扩展:** 对象.属性名=值

**2. 共有属性:** 只要保存在原型对象中的，子对象共享的成员

**扩展共有属性：**只能用构造函数的原型对象

**构造函数.prototype.属性名=值**

**修改属性值:**

**1. 自有属性:** 对象.属性名=值

**2. 共有属性:** \*只能通过构造函数的原型对象修改共有属性\*

构造函数.prototype.属性名=值

如果强行使用子对象.共有属性=值;

**后果:** 在子对象添加同名的自有属性;

**删除属性:**

**1. 自有属性:**  delete 对象.属性名;

**2. 共有属性:** 只能通过构造函数的原型对象删除

delete 构造函数.prototype.属性名

**判断自有属性和共有属性:**

**1. 判断自有属性: 判断指定属性是否直接保存在当前对象本地**

var bool=obj.hasOwnProperty("属性名")

如果返回true，说明"属性名"是obj的自有属性

返回false: 无法确定是否为共有！

可能是共有，也可能是没有

**2. 判断共有属性: 同时满足2个条件:**

**1. 不是自有:** obj.hasOwnProperty("属性名")==false

**2.而且在原型中有:** obj.属性名!==undefined

if(!obj.hasOwnProperty("属性名")

&&obj.属性名!==undefined){

说明是共有属性

}

**原型链:** 由各级子对象的\_\_proto\_\_属性，逐级继承，形成的链式结构。

**控制:** 对象成员的使用规则:

只要自己有，就不用父级的

只有自己没有，才延原型链向父级找

直到Object.prototype都没找到，才返回undefined

**总结:** 不使用任何对象.方式访问的变量，都去作用域链找

只要使用对象.方式访问的成员，都去原型链找

**this:** 和定义在哪个对象中无关！

只和\*调用时\*.前的对象有关！

**原型相关API:**

**1. 获取原型对象: 2种:**

构造函数.prototype

**子对象.\_\_proto\_\_:** 内部属性，可能被禁用

**尽量使用:** Object.getPrototypeOf(子对象)

**2. 判断对象间是否有继承关系:**

var bool=父对象.isPrototypeOf(子对象)

如果返回true，说明子对象确实继承自父对象

否则，两个对象间没有继承关系。

**3. instanceof: 判断一个对象是否是指定构造函数的实例**

function Student(){} 构造函数

Student：类型名/构造函数名

**实例化:** 用一个构造函数，创建一个对象

**结果:** 新创建的对象称为 类型的 实例

var lilei=new Student();

lilei instanceof Student =>true

lilei就是Student类型中的一个实例对象

**3、多态: 同一个事物，在不同情况下表现出不同的状态。**

**js中的多态只支持重写。**

**重写: override :** 如果子对象觉得父对象的成员不好用

可在本地定义同名成员，覆盖父对象的