

HTML5 Technology

HTML5 技术

JavaScript和XML

内容安排

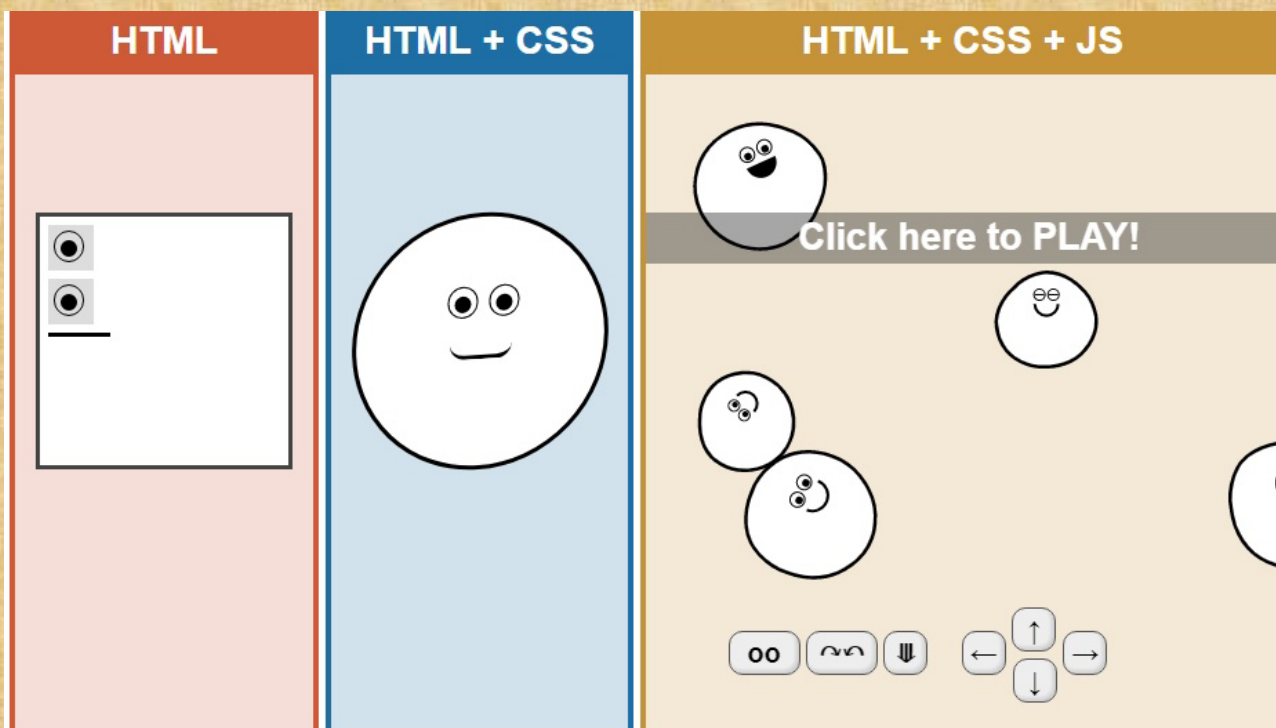
- ❑ 1.1 JavaScript 简述
- ❑ 1.2 JavaScript 在HTML中的使用
- ❑ 1.3 JavaScript 词法结构
- ❑ 1.4 JavaScript 语言基础
- ❑ 1.5 JavaScript 基本语句
- ❑ 1.6 函数
- ❑ 1.7 对象
- ❑ 1.8 XML 基础知识
- ❑ 课后思考
- ❑ 1.9 小结

1.1 JavaScript简述

- JavaScript的起源
- JavaScript的主要特点
- Javascript的应用

导学

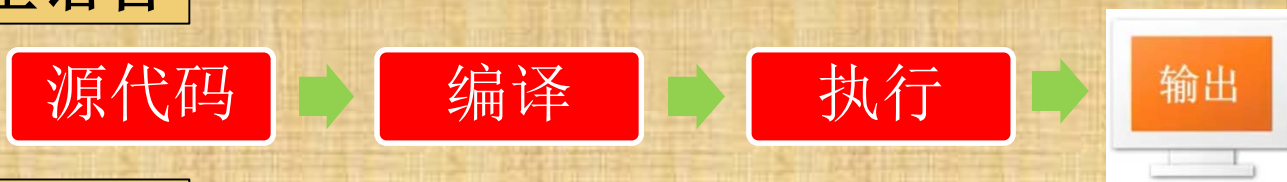
前面我们已经系统学习了HTML和CSS，其中**HTML**是基础**架构**，**CSS**用来**美化**页面，而**JavaScript**是实现网页**动态效果**的基石 在Web开发中扮演重要的角色。



导学

JavaScript是Web页面中的一种脚本编程语言，也是一种通用的、跨平台的、基于对象和事件驱动并具有安全性的脚本语言。它不需要进行编译，而是直接嵌入在HTML页面中，把静态页面转换成支持用户交互并响应相应事件的动态页面。

编译型语言



解释型语言



□ JavaScript的起源

LiveScript



JavaScript



Netscape®



Sun
microsystems

Microsoft®



JScript

□ JavaScript的主要特点

JavaScript特点

解释性

由浏览器执行、不需要编译。

基于对象

能运用自己已经创建的对象。

事件驱动

直接对用户或客户输入进行响应，无须经过Web服务程序。

跨平台

依赖浏览器本身，与操作环境无关。

安全性

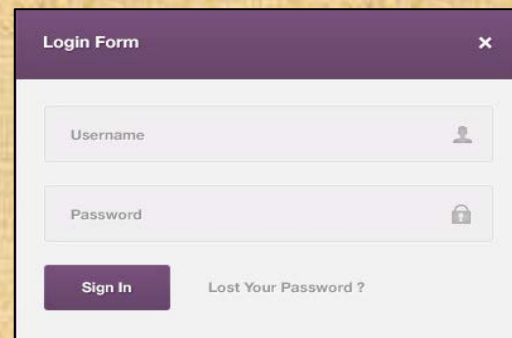
不能访问本地硬盘，不能对网络文档进行修改和删除等。

□ JavaScript的应用

JavaScript被广泛的应用到各个领域:



① 地图搜索



② 用户注册、验证



③ 网上购物



④ web 聊天

.....

1.2 JavaScript在HTML中的使用

- JavaScript的基本语法
- 在页面中直接嵌入JavaScript代码
- 链接外部JavaScript文件

□ JavaScript的基本语法

```
<script type="text/javascript" [src= "外部js文件"]>
```

```
...
```

```
</script>
```

语法说明：

（1）**script**：脚本标记。必须以**<script>...</script>**形式成对出现。

（2）**script**在页面中的位置决定了什么时候装载它们，如果希望在其他所有内容之前装载脚本，就要确保脚本在**head**部分。

□ JavaScript的基本语法

```
<script type="text/javascript" [src= "外部js文件"]>
```

...

```
</script>
```

语法说明：

(3) **src**属性不是必要的。它指定了一个要加载的外部js代码文件，一旦应用了这个属性，则**<script>**标签中的任何内容都将被忽略。

(4) **type**:指定HTML文档中使用的脚本语言及其脚本。

PS: JavaScript程序本身不能独立存在，它是依附于某个HTML页面，在浏览器端运行的。

□ 在页面中直接嵌入JavaScript代码

1、位于head部分的脚本

如果把脚本放置到head部分，在页面载入的时候，就同时载入了代码。通常这个区域的JavaScript代码是为body区域程序代码所调用的事件处理函数。如示例：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>这是我的第一个JavaScript程序</title>
  <script type="text/javascript">
    function show() {
      alert("欢迎加入JavaScript学习")
    }
  </script>
</head>
<body onload="show()">
</body>
</html>
```

显示结果：

此网页显示

欢迎加入JavaScript学习

确定

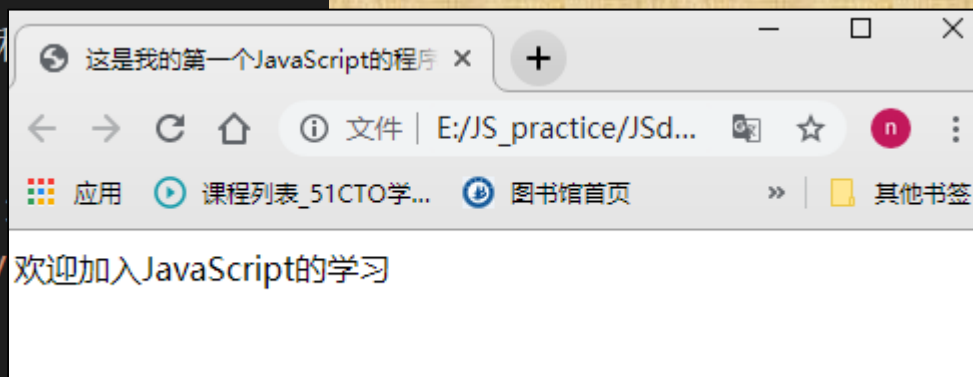
□ 在页面中直接嵌入JavaScript代码

2、位于**body**部分的脚本

放置于body中的脚本，通常是一些在页面载入时需要同时执行的一些脚本，这些代码执行后的输出就成为页面的内容，在浏览器中可以即时看到。如示例。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>这是我的第一个JavaScript程序
</head>
<body>
  <script type="text/javascript">
    document.write("欢迎加入JavaScript的学习")
  </script>
</body>
</html>
```

显示结果：



□ 链接外部JavaScript文件

如果JS代码比较复杂或是同一段代码可以被多个页面所使用，则可以将这些带代码放置在一个单独的文件中（扩展名为.js），然后在需要使用改代码的Web页面中链接该JavaScript文件即可。

链接外部JavaScript文件的语法格式如下：

```
<script type="text/javascript" src="script.js"></script>
```

□ 链接外部JavaScript文件

示例：先建立一个名为`script.js`的文件，内容如下：

```
function show() {  
    alert("欢迎加入JavaScript的学习")  
}
```

然后再建立一个HTML文件，内容如下：

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>这是我的第一个  
    <script src="script.js">  
</head>  
<body onload="show()"></body>  
</html>
```

显示结果：

此网页显示

欢迎加入JavaScript学习

确定

1.3 JavaScript 词法结构

- ❑ 字符集
- ❑ 注释
- ❑ 直接量
- ❑ 标识符和保留字
- ❑ 可选的分号

□ 字符集

JavaScript大小写敏感。 JavaScript中的关键字、变量、函数名和所有的标识符都必须采取一致的大小写形式。（HTML不区分大小写。）

如：“online”、“Online”、“OnLine”和
“ONLINE”是四个不同的变量名。

□ 注释

JavaScript支持两种格式的注释。

单行注释：// 或 /* */

```
//一个弹窗提示  
function show() {  
    alert("欢迎加入JavaScript的学习")  
}
```

```
/*一个弹窗提示*/  
function show() {  
    alert("欢迎加入JavaScript的学习")  
}
```

多行注释：

```
/**  
 * function show() {  
 *     alert("欢迎加入JavaScript的学习")  
 * }  
 */
```

□ 直接量

直接量(literal),即程序中直接使用的数据值。如:

1.2	//小数
12	//数字
“hello world”	//字符串文本
‘Hi’	//另一个字符串
true	//布尔值
false	//另一个布尔值
null	//空

□ 标识符和保留字

标识符就是一个名字。关于标识符的规定有：

(1) 必须使用字母、下划线或美元（\$）开始。

(2) 必须使用英文字母、数字、下划线组成，不能出现空格或制表符。

i

✓

my_variable_name

✓

V13

✓

_dummy

✓

\$str

✓

1he

✗

□ 标识符和保留字

- (3) 不能使用JavaScript关键字与JavaScript保留字。
- (4) 不能使用JavaScript语言内部的单词，比如：Infinity, NaN, undefined等。
- (5) 大小写敏感。

作为命名的一种规定，如同起名一样，也是要很慎重的。总的来说，标识符的确定应该做到“见名知意”。

□ 标识符和保留字

JavaScript保留了一部分关键字，它们标识了程序的结构和功能，所以在编写代码时，不能用它们作为自定义的变量名或函数名。

表 JavaScript的关键字

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

□ 标识符和保留字

除了关键字，JavaScript还有一些可能未来扩展时使用的保留字，同样不能用于标识符的定义。

表 JavaScript的保留字

abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	throws	transient
volatile				

□ 可选的分号

JavaScript使用分号（;）将语句分隔开，对增强代码的可读性和整洁性非常重要。有些地方可以省略分号，有些地方则不能省略分号。

考虑如下代码，因为两条语句用两行书写，第一个分号是**可以省略**的：

```
a=3;  
b=4;
```

如果按照如下格式书写，第一个分号则**不能省略**：

```
a=3; b=4;
```

1.4 JavaScript 语言基础

- 数据类型
- 常量和变量
- 运算符和表达式

□ 数据类型

1、while语句

2、do...while语句

3、for语句

4、循环语句的嵌套

□ 数据类型

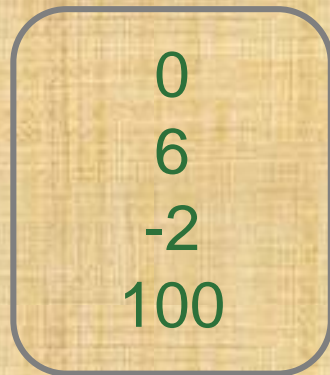
JavaScript基本数据类型有数值型、字符串型、布尔型以及两个特殊的数据类型。

1、数值型

是JavaScript种最基本的数据类型。与其他程序设计语言不同的是，它**不区分整型和浮点型数值**。

(1) 十进制

由0~9组成的数字序列，如：



0
6
-2
100

□ 数据类型

(2) 八进制

八进制数据以数字0开头，其后跟谁一个数字序列，每个数字在0~7之间。如：

07
0366

PS:JS对八进制数据支持性不是很好，尽量不要使用。

(3) 十六进制

以“0X”或“0x”开头，可以是0~9中的某个数字，也可以是A（a）~F（f）中的某个字母,来表示0到15的某个值。如：

0xff // 255
0x66 // 102

□ 数据类型

2、字符串型（string）

由0个或多个字符组成的序列，它可以包含大小写字母、数字、标点符号和汉字等其他符号，其使用 ‘ ’ 或者 “ ” 表示。如：

‘你好，JavaScript’
“你好，JavaScript”
‘你好， “JavaScript” ’
“你好， ‘Javascript’ ”

□ 数据类型

表 JavaScript常用的转义字符

转义字符	描述	转义字符	描述
\b	退格	\/	斜线
\n	换行符	\r	回车符
\t	水平制表符, Tab空格	\\	反斜杠
\f	换页	\u	编码转换
\'	单引号	\"	双引号

数据类型

示例:在外部.js文件上写入如下代码。

```
//用 (\") 表示 (")
document.write("\JavaScript\"真强大");
document.write("<hr>");
//用(\\)表示(\\)
document.write("文件在C:\\Windows\\下");
document.write("JavaScript"真强大);
//用(\n)表示换行
document.write("文件在C:\Windows\下");
document.write("JavaScript"真强大);
```

运行结果：

"JavaScript"真强大

文件在C:\Windows\下

hello
JavaScript

□ 数据类型

3、布尔型 (boolean)

布尔值指代真或假、开或关、是或否。这个类型只有两个值，保留字true和false。

JavaScript程序中的比较语句的结果通常都是布尔值，例如：

```
a==4
```

布尔值通常用于JavaScript中的控制结构中。如：

```
if (a==4)
  b=b+1;
else
  a=a+1;
```

□ 数据类型

4、特殊数据类型

(1) 未定义值

未定义值`undefined`,表示变量还没有赋值(如`var a;`)

(2) 空值(`null`)

关键字`null`是一个特殊的值，它表示空值，用于定义空的或不存在的引用。

□ 常量和变量

1、常量

常量是指在程序运行过程中**保持不变**的数据。如：

123

//数值型常量

“JavaScript”

//字符型常量

true/false

//布尔型常量

2、变量

可以保存执行时变化的值的名字，被称为“**变量**”，变量相当于是值的容器。**var**的作用就是声明(创建)变量。

□ 常量和变量

变量使用方法：先定义，后使用。

变量声明的实例：

```
var account;  
var area=0;  
var name="张三";  
var status=tur;  
var a,b,c;
```


□ 运算符和表达式

JavaScript算术运算符负责算术运算。

运算符	描述	例子(令a=2)	结果
+	加	$b = a + 2$	$b = 4$
-	减	$b = a - 1$	$b = 1$
*	乘	$b = a * 2$	$b = 4$
/	除	$b = a / 2$	$b = 1$
%	取模	$b = a \% 2$	$b = 0$
++	自增	$b = a++$	$b = 2$
--	自减	$b = --a$	$b = 1$

□ 运算符和表达式

简单的赋值运算符由等号(=)实现，复合赋值运算是由算术运算符或位移运算符加等号实现

运算符	描述	例子(令a=2)	结果
=	赋值	a = 2	a=2
+=	加法赋值	a += 1	a = 3
-=	减法赋值	a -= 1	a = 1
*=	乘法赋值	a *= 2	a = 4
/=	除法赋值	a /= 2	a = 1
%=	取模赋值	a %= 2	a = 0

□ 运算符和表达式

关系运算符负责判断两个值是否符合给定的条件，关系表达式返回的结果为“true”或“false”，分别代表符合给定的条件或者不符合。

运算符	描述	例子	结果	判断内容
>	大于	6>5	true	数值
<	小于	6<5	false	数值
>=	大于或等于	6>=5	true	数值
<=	小于或等于	6<=5	false	数值
!=	不等于	6!=5	true	数值
==	相等	6==5	false	数值
===	恒等于	5===“5”	false	数值与类型
!==	不恒等于	5!==“5”	true	数值与类型

□ 运算符和表达式

逻辑运算符用于对一个或多个布尔值进行逻辑运算。

JavaScript中有三个逻辑运算符：`&&`(与)、`||`(或)、`!`(非)。

a	b	!a	a b	a&&b
true	true	false	true	true
true	false	false	true	false
false	true	true	true	false
false	false	true	false	false

□ 运算符和表达式

条件运算符

基本语法: 表达式? 结果1:结果2

语法说明: 如果“表达式”的值为true, 则整个表达式的结果为“结果1”, 否则为“结果2”。如下示例:

```
<script type="text/javascript">  
  var a=10;  
  var b=10;  
  alert(a==b?"相等":"不相等");  
</script>
```

结果显示:

此网页显示
相等

确定

□ 运算符和表达式

其他运算符

1、逗号运算符

逗号运算符用于将多个表达式排在一起，而整个表达式的值为最后一个表达式的值。如：

```
<script type="text/javascript">  
    var a,b,c,d;  
    a=(b=3,c=5,d=6);  
    alert("a的值为"+a);  
</script>
```

结果显示：

此网页显示
a的值为6

确定

□ 运算符和表达式

其他运算符

2、typeof运算符

typeof运算符用于判断操作数的数据类型。

语法格式： `typeof 操作数`

表 不同类型数据使用typeof运算符的返回值

数据类型	返回值	数据类型	返回值
数值	number	null	object
字符串	string	对象	object
布尔值	boolean	函数	function
undefined	undefined		

□ 运算符和表达式

typeof运算符实例：

```
var a,b,c,d;  
a=3;  
b="name";  
c=true;  
d=null;  
/* 输出变量的类型 */  
alert("a的类型为"+(typeof a)+"\nb的类型为"+(typeof b)  
      +"\nc的类型为"+(typeof c)+"\nd的类型为"+(typeof d));
```

运行结果：

此网页显示

a的类型为number
b的类型为string
c的类型为boolean
d的类型为object

确定

□ 运算符和表达式

其他运算符

3、new运算符

在JavaScript中有很多内置对象，如字符串对象、日期对象和数值对象等，通过new运算符可以用来创建一个新的内置对象实例。如：

```
object1 = new object;           // 创建自定义对象  
Array2 = new Arrat();           // 创建数组对象  
Date3 = new Date("2019/9/6");   // 创建日期对象
```

1.5 JavaScript 基本语句

- 条件判断语句
- 循环语句
- 跳转语句

□ 条件判断语句

1、if语句

简单if语句

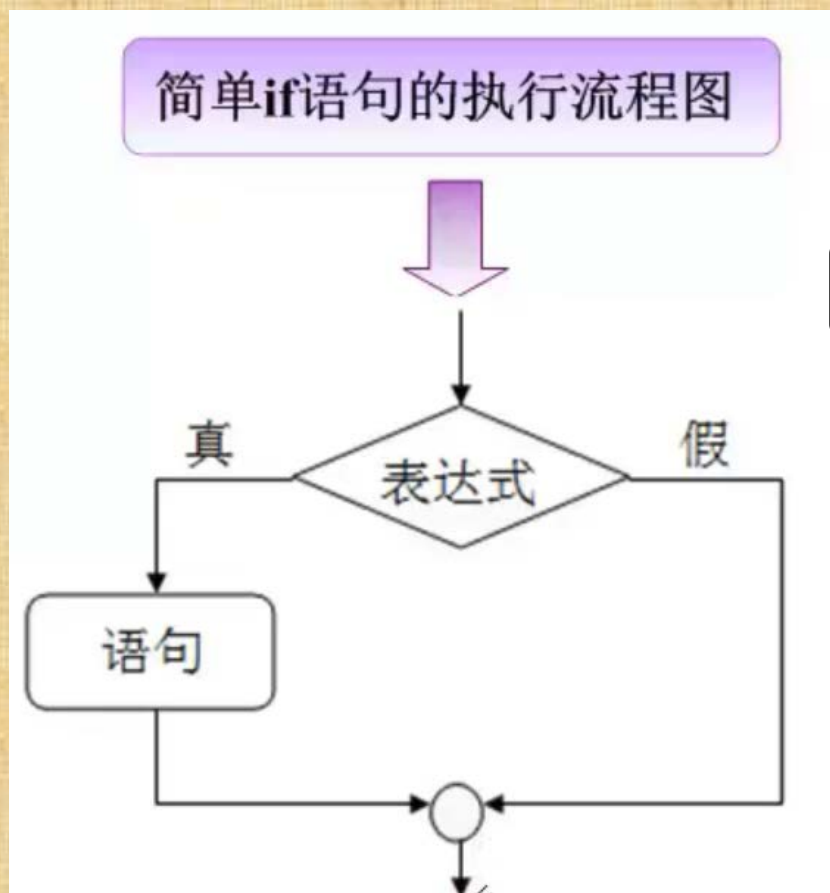
if...else语句

if...else if语句

if语句的嵌套

2、switch语句

□ 条件判断语句—简单if语句



语法规式

条件表达式

```
if(表达式){  
    语句  
}
```

条件为真时
执行的语句

□ 条件判断语句—简单if语句

实例 求三个数中的最大值。

```
var a,b,c,maxValue;  
a=10;  
b=20;  
c=30;  
maxValue=a;  
if(maxValue<b){  
    maxValue=b;  
}  
if(maxValue<c){  
    maxValue=c;  
}  
alert(a+"、"+b+"、"+c+"三个数中的最大值为："+maxValue);
```

显示结果：

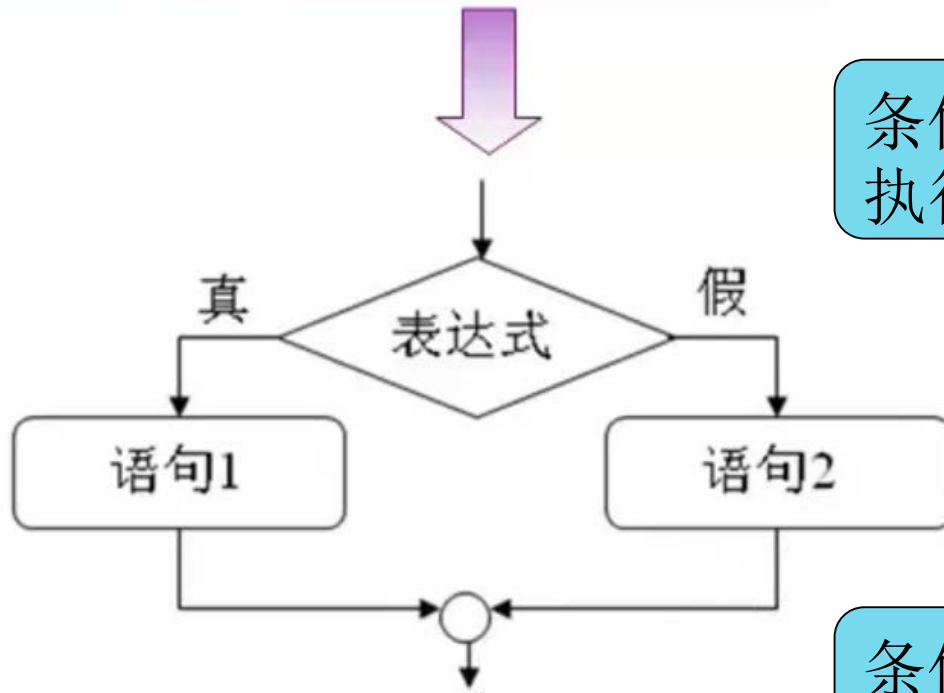
此网页显示

10、20、30三个数中最大值为：30

确定

□ 条件判断语句— if...else语句

if...else语句的执行流程图



语法格式

条件为真时
执行的语句1

```
if(表达式){  
    语句1  
}else{  
    语句2  
}
```

条件为假时
执行的语句2

□ 条件判断语句— if...else语句

实例 求三个数中的最大值。

```
var a,b,c,maxValue;  
a=10;  
b=20;  
c=30;  
maxValue=a;  
if(maxValue<b){  
    maxValue=b;  
}  
else{  
    maxValue=c;  
}  
alert(a+"、"+b+"、"+c+"中的最大值为: "+maxValue);
```

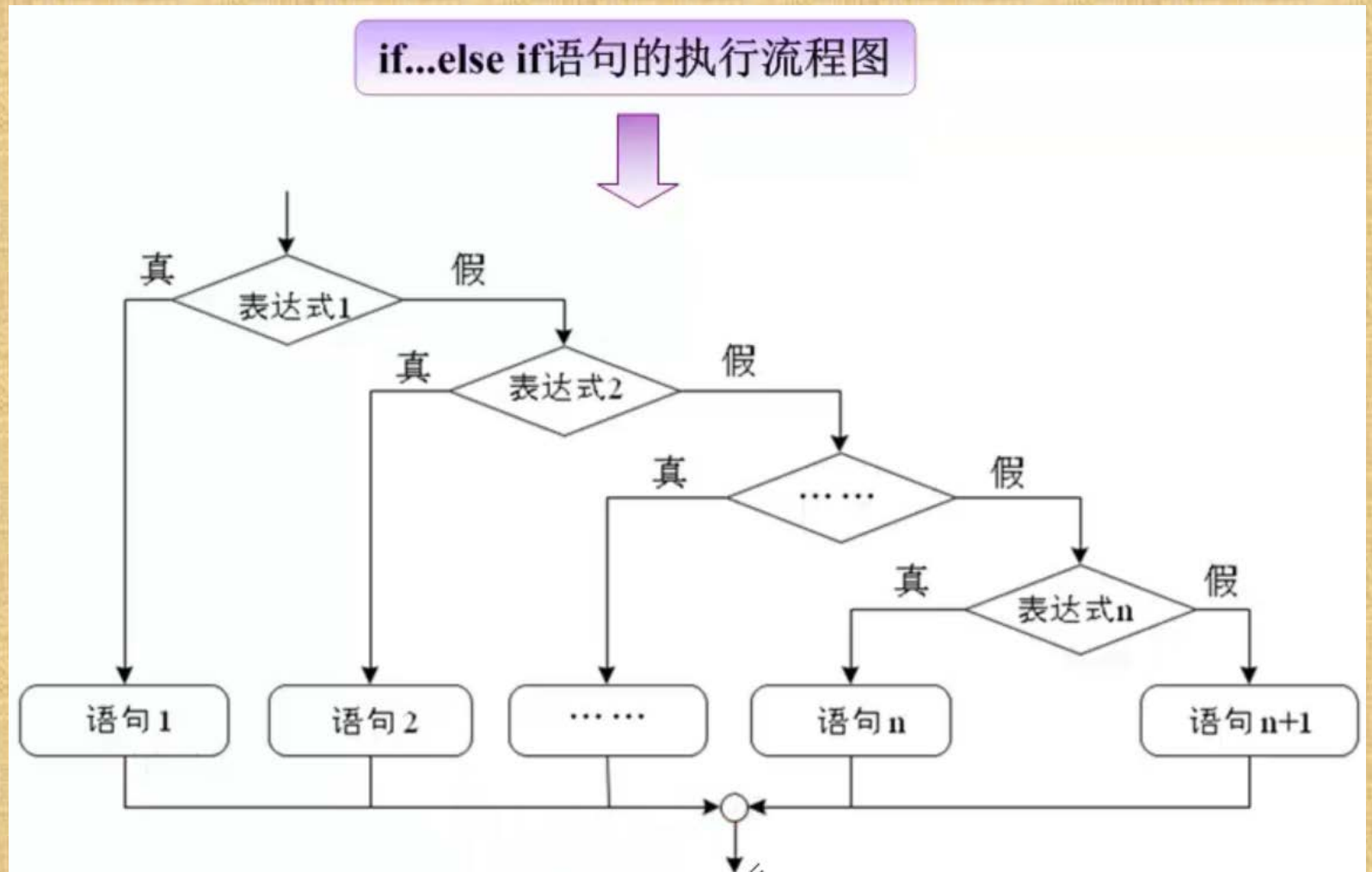
显示结果:

此网页显示

10、20、30三个数中最大值为: 30

确定

□ 条件判断语句— if...else if语句



□ 条件判断语句— **if...else if** 语句

语法格式



```
if(表达式1){  
    语句1  
}else if(表达式2){  
    语句2  
}  
...  
else if(表达式n){  
    语句n  
}else{  
    语句n+1  
}
```

□ if...else语句

实例

某校的学生成绩转化不同等级，划分标准如下：

- ① “优秀”，大于等于90分；
- ② “良好”，大于等于75分；
- ③ “及格”，大于等于60分；
- ④ “不及格”，小于60分。

假设小明的成绩是85分，输出改成绩对应的等级。

显示效果如下：

此网页显示

小明的考试成为良好

确定

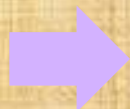
□ 条件判断语句— if...else语句

具体代码如下所示：

```
var grade="";  
var score=85;  
if(score>=90){  
    grade="优秀";  
}else if(score>=75){  
    grade="良好";  
}else if(score>=60){  
    grade="及格";  
}else if(score<60){  
    grade="不及格";  
}  
alert("小明的考试成为"+grade);
```

□ 条件判断语句— if语句的嵌套

语法格式



```
if(表达式1){  
    if(表达式2){  
        语句1  
    } else {  
        语句2  
    }  
}
```

Case 1

```
if(表达式1){  
    if(表达式2){  
        语句1  
    } else {  
        语句2  
    }  
} else {  
    if(表达式3){  
        语句3  
    } else {  
        语句4  
    }  
}
```

Case 2

□ 条件判断语句— if...else语句

实例

假设某工种的男职工60岁退休，女职工55岁退休，应用if语句的嵌套来判断一个58岁的女职工是否已经退休。

显示效果如下：

此网页显示

该女职工已经退休3年

确定

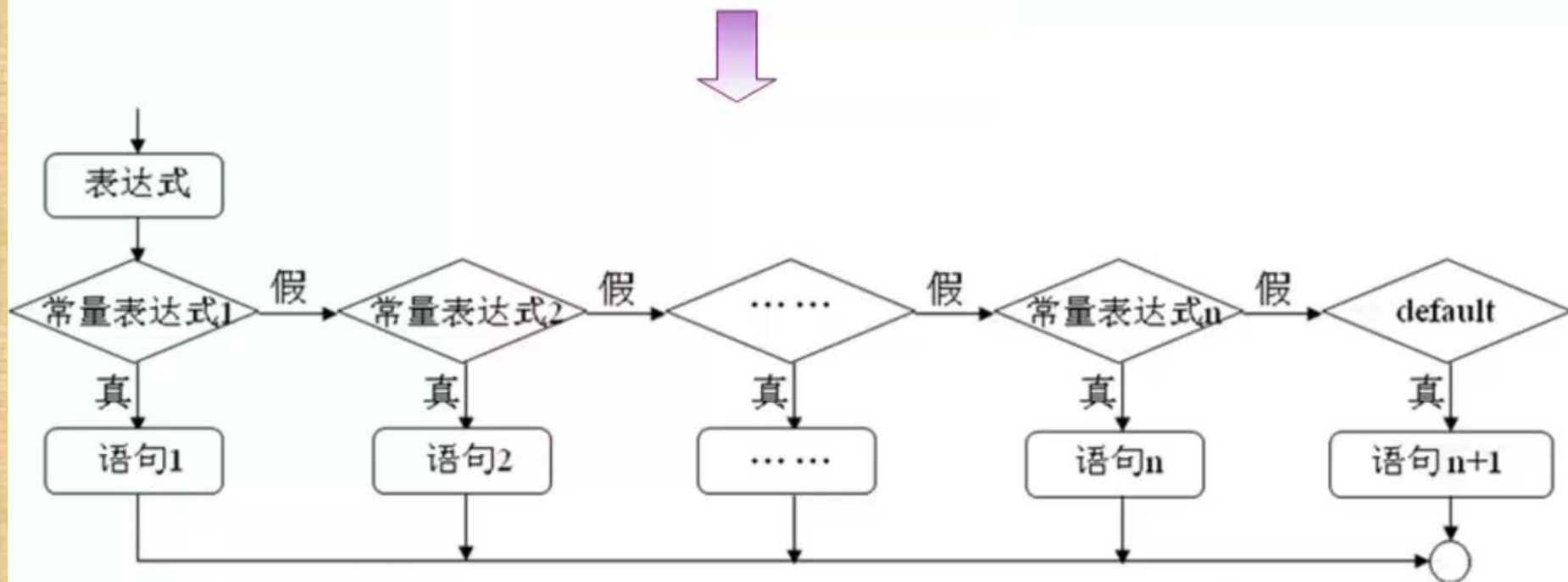
□ 条件判断语句— if...else语句

具体代码如下所示：

```
var sex="女";
var age=58;
if(sex=="男"){
    if(age>=60){
        alert("该男职工已经退休"+(age-60)+"年");
    }else{
        alert("该男职工并未退休");
    }
} else{
    if(age>=55){
        alert("该女职工已经退休"+(age-55)+"年");
    }else{
        alert("该女职工并未退休");
    }
}
```

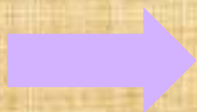
□ 条件判断语句— switch语句

switch语句的执行流程图



□ 条件判断语句

语法格式



用于结束switch语句，从而使JavaScript只执行匹配的分支。不能省略。

```
switch(表达式){  
    case 常量表达式1:  
        语句1;  
        break;  
    case 常量表达式2:  
        语句2;  
        break;  
    ...  
    case 常量表达式n:  
        语句n;  
        break;  
    default:  
        语句n+1;  
        break;  
}
```

冒号，不能写成分号。

□ 条件判断语句— switch语句

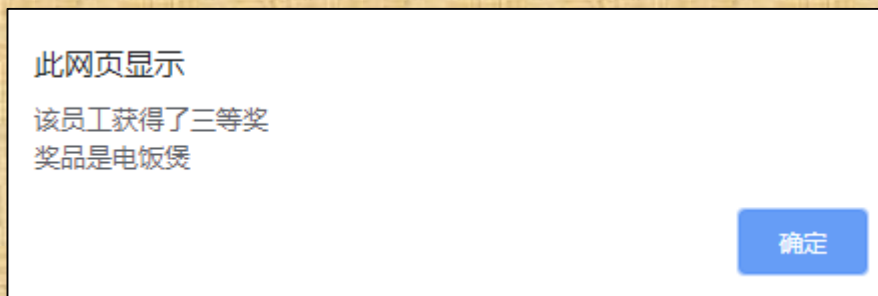
实例

某公司年会抽奖活动，中将号码及其对应的奖品设置如下：

- ① “1” 代表 “一等奖”，奖品是 “华为手机”；
- ② “2” 代表 “二等奖”，奖品是 “微波炉”；
- ③ “3” 代表 “三等奖”，奖品是 “电饭煲”；
- ④ 其他号码代表 “安慰奖”，奖品是 “16G-U盘”。

假设某员工抽中的号码为3。输出该员工的中奖级别以及获得的奖品。

显示效果如下：



□ 条件判断语句— switch语句

具体代码如图所示：

```
var grade;
var prize;
var code=3;
switch(code){
    case 1:
        grade="一等奖";
        prize="华为手机";
        break;
    case 2:
        grade="二等奖";
        prize="微波炉";
        break;
    case 3:
        grade="三等奖";
        prize="电饭煲";
        break;
    default:
        grade="安慰奖";
        prize="16G-U盘";
        break;
}
alert("该员工获得了"+grade+"\n奖品是"+prize);
```

□ 循环语句

1、while语句

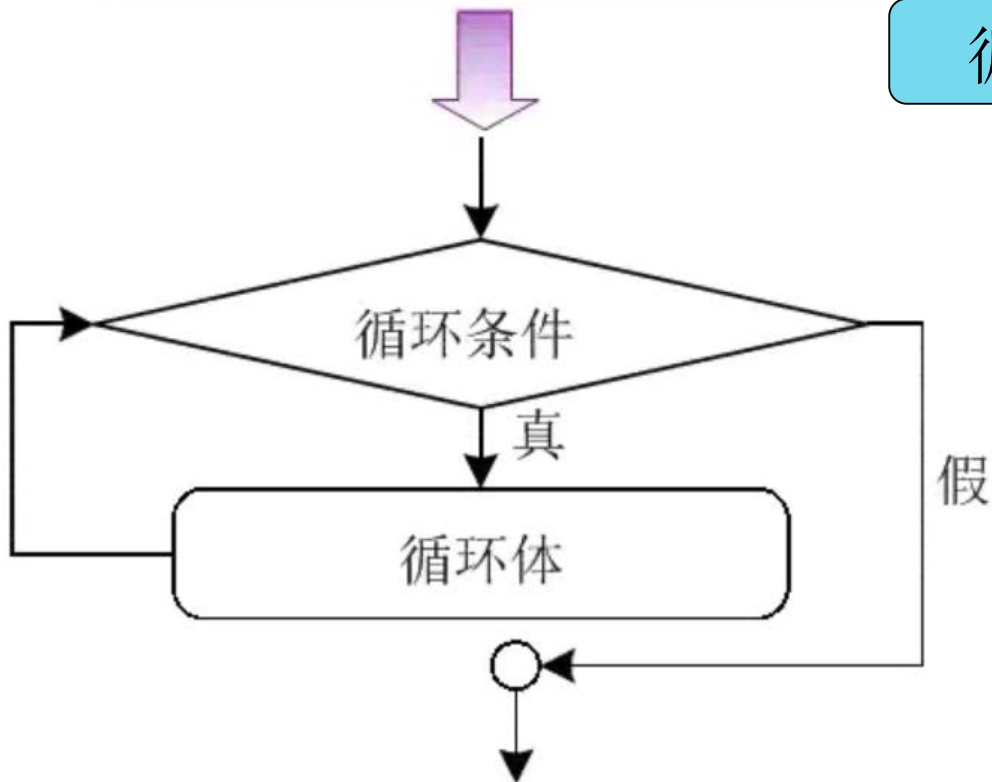
2、do...while语句

3、for语句

4、循环语句的嵌套

□ 循环语句—while语句

while循环语句的执行流程图



语法格式

循环条件

```
while(表达式) {  
    语句  
}
```

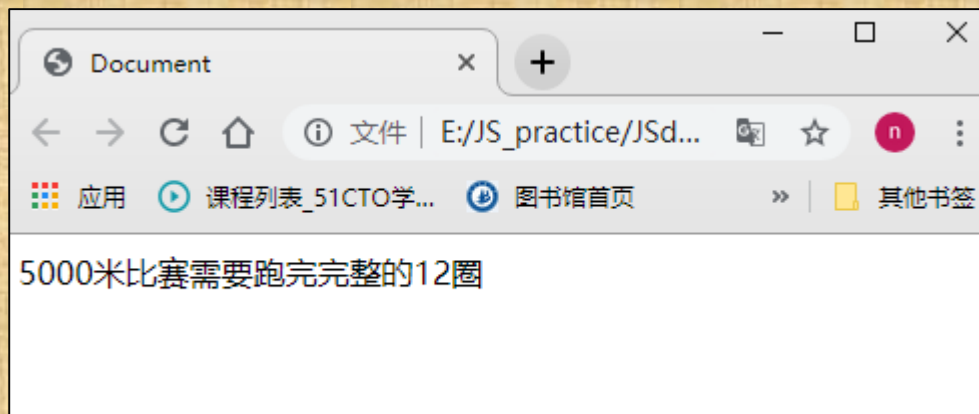
循环体

□ 循环语句—while语句

实例

运动员参加5000米比赛，已知标准的体育场跑道一圈是400米，应用while语句计算出在标准的体育场跑道上完成比赛需要跑完的完整圈数。

显示效果如图：



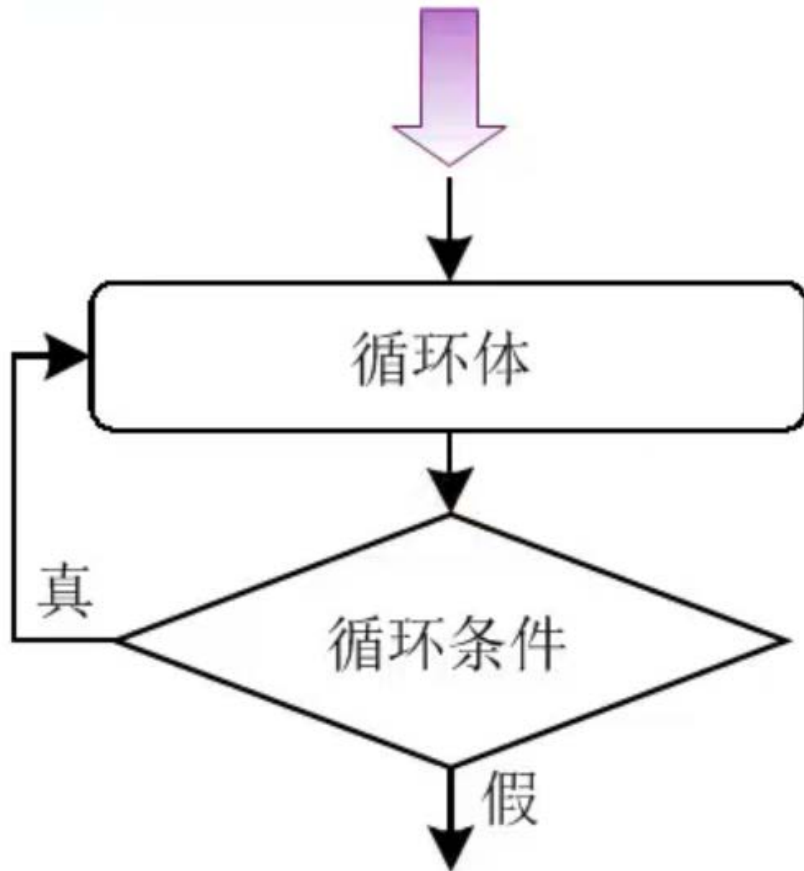
□ 循环语句— while语句

具体代码如下所示：

```
var distance=400;  
var count=0;  
while(distance<=5000){  
    count++;  
    distance=(count+1)*400;  
}  
document.write("5000米比赛需要跑完完整的"+count+"圈");
```

□ 循环语句—do...while语句

do...while语句的执行流程图



语法格式

循环体

```
do{  
    语句  
} while(表达式);
```

循环条件

□ 循环语句—do...while语句

实例 计算 $1+2+3+\dots+100$ 的和。

```
var i=1;
var sum=0;
do{
    sum+=i;
    i++;
}while(i<=100);
alert("1+2+3+...+100="+sum);
```

显示结果如下：

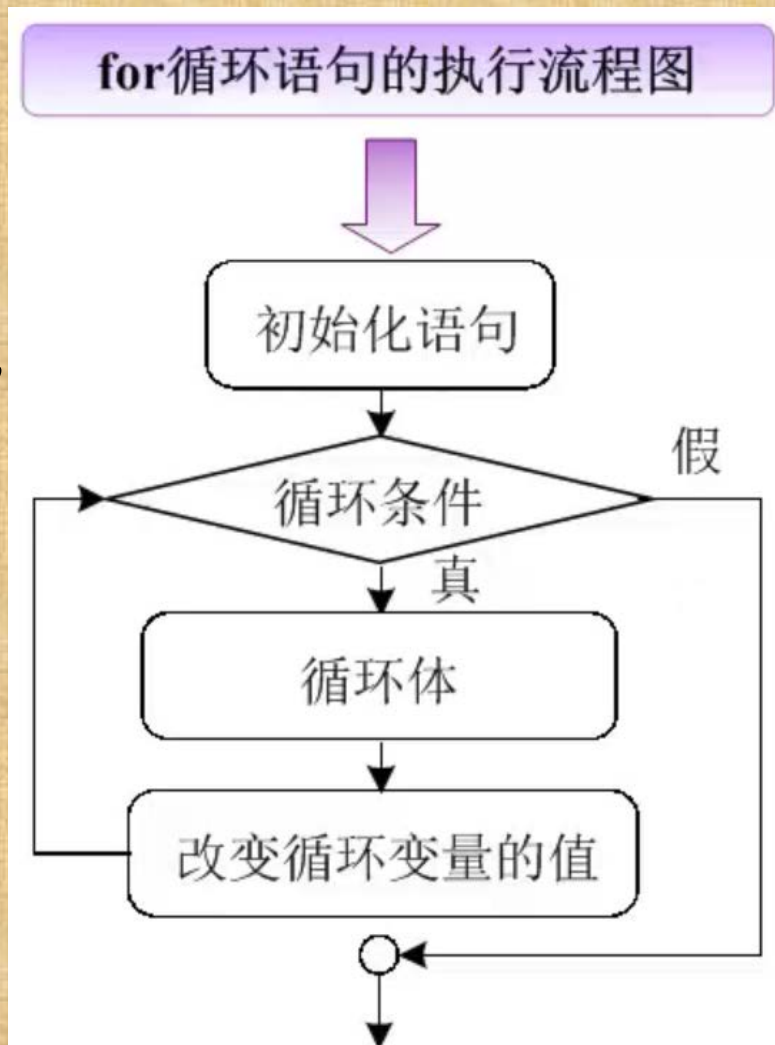
此网页显示

1+2+3+...+100+5050

确定

□ 循环语句— for语句

for循环语句也称为**计次循环语句**，一般用于循环次数已知的情況，在JavaScript中应用比较广泛。



□ 循环语句— for语句

初始化语句

循环条件

改变循环变量的值

语法格式



```
for(初始化表达式; 条件表达式; 迭代表达式){  
    语句  
}
```

循环体

□ 循环语句— for语句

实例 计算 $1+2+3+\dots+100$ 的和。

```
var i,sum;  
sum=0;  
for(i=1;i<=100;i++){  
    sum+=i;  
}  
alert("1+2+3+...+100="+sum);
```

显示结果如下：

此网页显示

1+2+3+...+100+5050

确定

□ 循环语句的嵌套

在一个循环语句的循环体中可以包含其他的循环语句，这称为循环语句的嵌套。上述三种循环语句(**while**语句、**do...while**语句和**for**语句)都是可以相互嵌套的。

练一练：运用上述知识输出乘法口诀表。

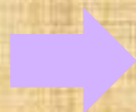
□ 跳转语句

有时在循环中，可能碰到一些需要提前中止循环的情况，或者放弃某次循环的情况。主要用到了**continue**语句和**break**语句。

□ 跳转语句—continue语句

continue语句用于跳过本次循环，并开始下一次循环。

语法格式



```
continue;
```

说明：continue语句只能**应用在循环语句**中，即while语句、for语句、do...while语句中。

注意：当使用continue语句跳过本次循环后，如果循环条件的结果为false，则退出循环，否则继续下一次循环。

□ 跳转语句—continue语句

实例

万达影城7好影厅的观众席有4排，每排有10个座位。其中，1排6座和3排9座已经出售，在页面中输出该影厅当前的作为图。

结果如图所示：



□ 跳转语句—continue语句

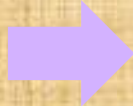
关键代码如图所示：

```
document.write("<table align='center'>");           // 输出表格标签
for(var i = 1; i <= 4; i++){                         // 定义外层for循环语句
    document.write("<tr height=70>");               // 输出表格行标签
    for(var j = 1; j <= 10; j++){                     // 定义内层for循环语句
        if(i == 1 && j == 6){                         // 如果当前是1排6座
            document.write("<td align='center' width=80 background=yes.png>已售</td>"); // 居中显示并将座位标记为“已售”
            continue;                                 // 应用continue语句跳过本次循环
        }
        if(i == 3 && j == 9){                         // 如果当前是3排9座
            document.write("<td align='center' width=80 background=yes.png>已售</td>"); // 将座位标记为“已售”
            continue;                                 // 应用continue语句跳过本次循环
        }
        document.write("<td align='center' width=80 background=no.png>"+i+"排"+j+"座"+"</td>"); // 输出排号和座位号
    }
    document.write("</tr>");                         // 输出表格行结束标签
}
document.write("</table>");                         // 输出表格结束标签
```

□ 跳转语句—break语句

在循环语句中，**break**语句用于跳出循环，即结束循环。

语法格式



break;

说明： **break**语句通常用在**for**语句、**while**语句、**do...while**语句或**switch**语句中。

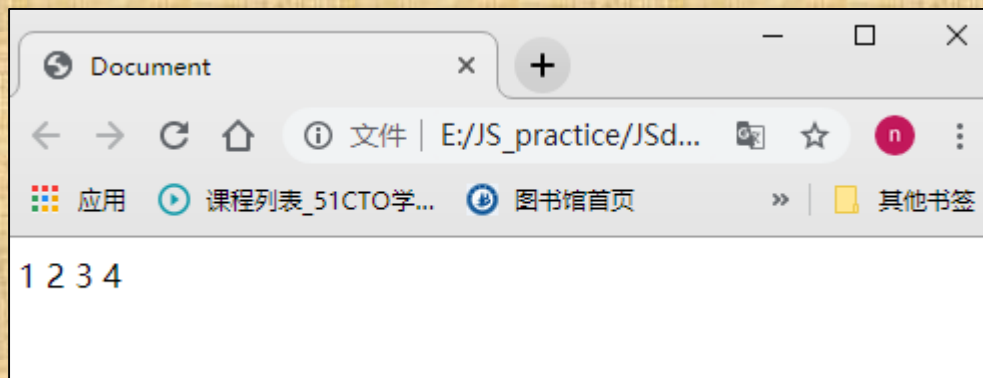
注意： 在嵌套的循环语句中，**break**语句只能跳出当前这一层的循环，而不能跳出所有的循环语句。

□ 跳转语句—break语句

示例

```
for(i=1;i<+10;i++){  
    if(i==5)  
        break;           // 如果i等于5就跳出整个循环  
    document.write(i+"\n"); //输出变量i的值  
}
```

结果如图所示：



1.6 函数

- 函数的定义和调用
- 函数的返回值

□ 函数的定义和调用

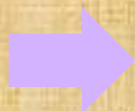
作用：通过函数可以封装任意多条语句，而且可以在任何地方，任何时间调用执行。

1、函数的定义

函数使用`function`声明，后跟一组参数以及置于大括号中需要执行的一段代码。

□ 函数的定义和调用

语法格式



```
function 函数名([参数1],[参数2]...){  
    语句  
    [return 返回值]  
}
```

说明：

函数名：必选，用于指定函数名，具有唯一性。

参数：可选，用于指定参数列表，最多可以有**255**个参数。

语句：必选，是函数体，用于实现函数功能的语句。

返回值：可选，用于返回数值。

□ 函数的定义和调用

2、函数的调用

函数定义后并不会自动执行，要执行一个函数需要在特定的位置调用函数。

```
//声明一个函数
function myFun(){
    alert("我是一个函数");
}
//函数调用
myFun();
```



此网页显示
我是一个函数

确定

```
//声明一个有参数函数
function add(num1,num2){
    var sum=num1+num2;
    alert(num1+"和"+num2+"的和是"+sum);
}
add (3,5);
add (10,30);
```



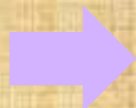
此网页显示
10和30的和是40

确定

□ 函数的返回值

对于函数调用，既可以通过参数向函数传递数据，也可以从函数中获取数据，也就是说函数可以返回值。

语法规式



return 表达式;

说明:

- 1、函数会在执行完**return**语句之后停止并立即退出。
- 2、**return**语句也可以不带有任何返回值，一般用于需要提前停止函数执行而又不需要返回值的情况下。

□ 函数的返回值

示例 模拟淘宝网计算购物车中商品总价的功能。假设购物车中有如下商品信息：

- ① 手机：单价2500元，数量2台；
- ② 笔记本：单价5000元，数量1台。

```
function price(unitPrice,number) {  
    var totalPrice = unitPrice*number;  
    return totalPrice;  
}  
  
var phone = price(2500,2);  
var computer = price(5000,1);  
var total = phone + computer;  
alert("购物车中商品总价: "+total+"元");
```

显示结果：

此网页显示

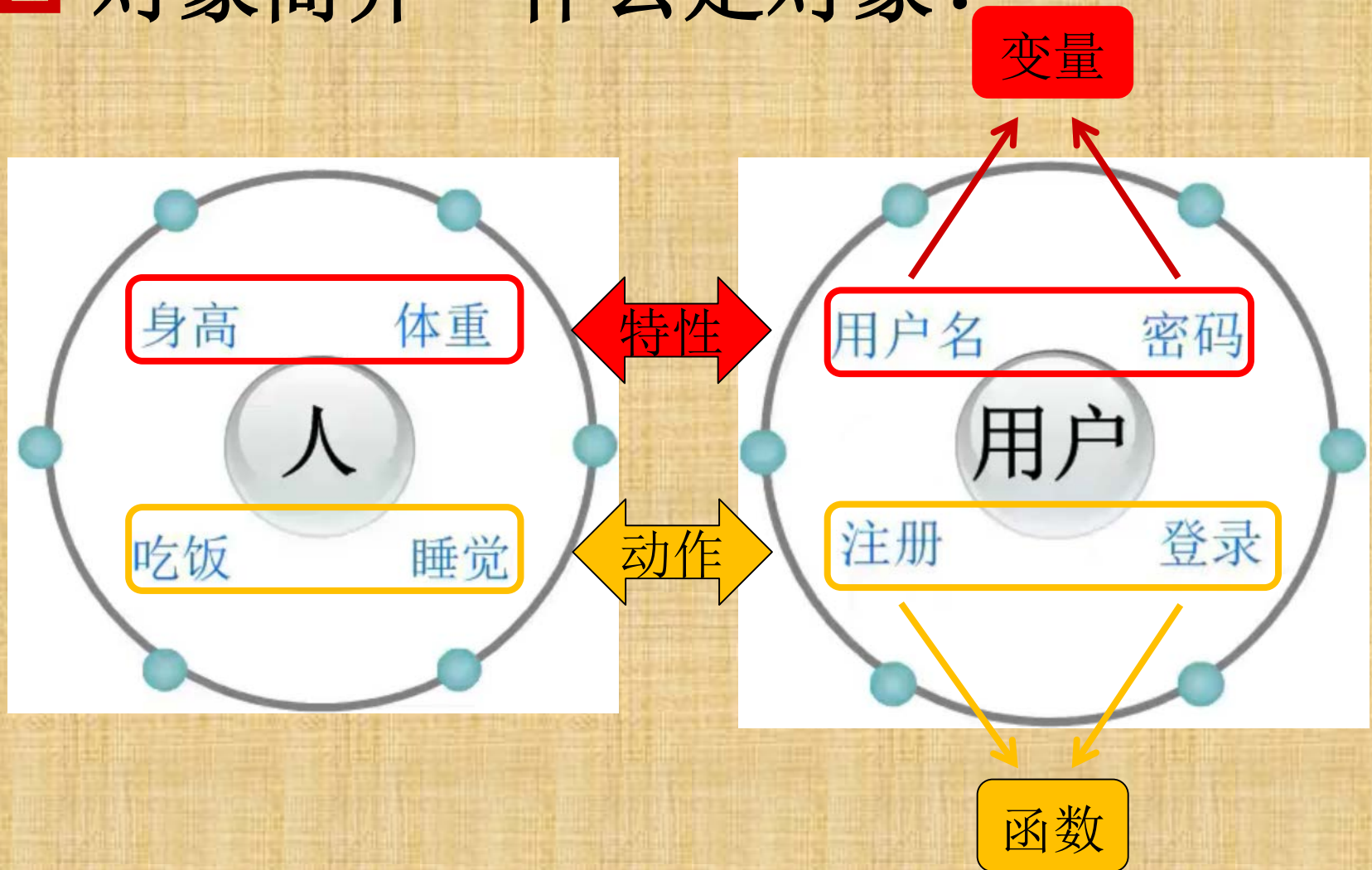
购物车中商品总价：10000元

确定

1.7 对象

- 对象简介
- 对象分类
- 创建对象

□ 对象简介—什么是对象？



□ 对象简介—属性和方法



1、对象的属性

包含在对象内部的变量称为对象的属性，用来描述对象特性的一组数据，表明对象的**状态**。

获取或设置属性语法

①对象名.属性名;

②对象名["属性名"]

如：var name = 用户.用户名

□ 对象简介—属性和方法

2、对象的方法

包含在对象内部的函数称为对象的方法，它可以用来实现某个功能，表明对象所具有的**行为**。

获取或设置属性语法



对象名.方法名(参数);

如：用户.注册();

因此，在JavaScript中，对象就是属性和方法的集合，这些属性和方法也叫做对象的成员。

□ 对象分类

1、内置对象

由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现。

-如：Math、String、Number、Date、Function

2、宿主对象

由 ECMAScript 实现的宿主环境提供的对象，可以理解为：浏览器提供的对象。

-如：BOM、DOM

3、自定义对象

开发人员自己定义的对象。

□ 对象分类

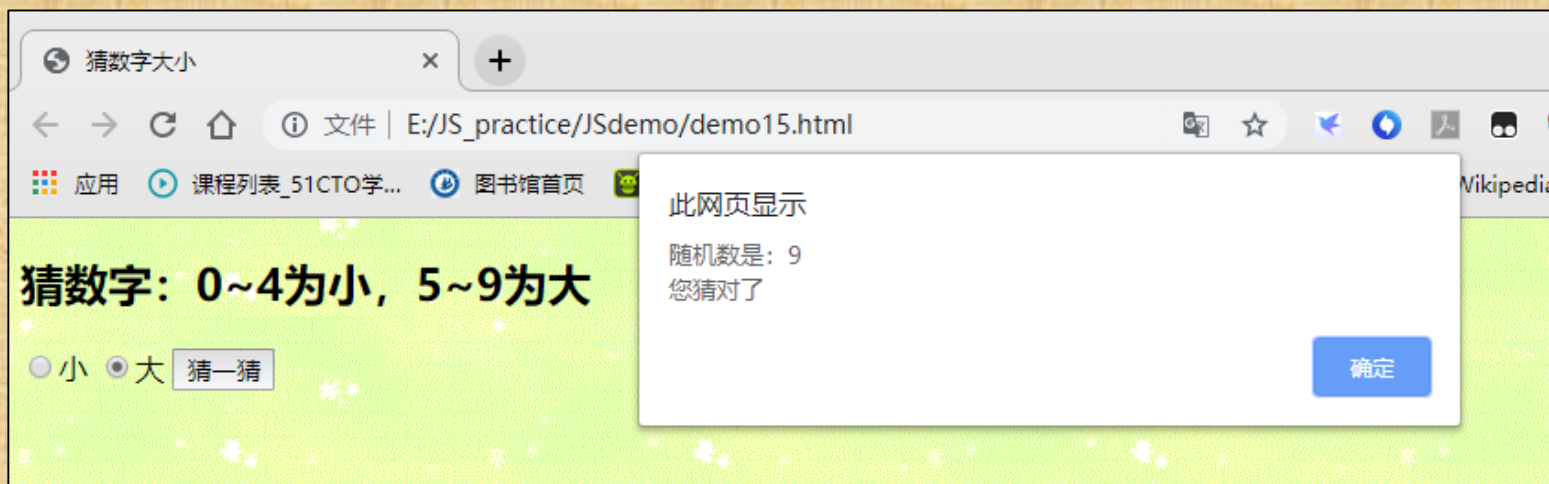
Example 1: **math**对象。

math对象提供了大量的数学常量和数学函数。更多详细的math对象的属性和方法请参考：

https://www.w3school.com.cn/jsref/jsref_obj_math.asp。

示例

做一个简单的猜数字大小游戏。结果如下所示：



□ 对象分类

Example 1: **math**对象。

`math.random()`方法可返回0~1之间的一个随机数。

`math.floor(x)`方法对一个数经行舍入，其返回值小于等于x，且与x最接近的整数。

如下为HTML代码：

```
<h2>猜数字：0~4为小，5~9为大</h2>
<form name="form">
  <!-- 设置单选 -->
  <input type="radio" name="num" value="small" checked="checked">小
  <input type="radio" name="num" value="big">大
  <!-- 设置按钮 -->
  <input type="button" name="but" value="猜一猜" onclick="guess()">
</form>
```

□ 对象分类

如图为JavaScript代码：

```
function guest(){  
    // 生成一个0~9之间的随机整数  
    var number = Math.floor(Math.random()*10);  
    // 如果选择第一个单选按钮，即选小  
    if(form.num[0].checked){  
        if(number >= 0 && number <= 4){  
            alert("随机数是: "+number+"\n您猜对了");  
        }else{  
            alert("随机数是: "+number+"\n您猜错了");  
        }  
    }  
    // 如果选择第二个单选按钮，即选大  
    if(form.num[1].checked){  
        if(number >= 5 && number <= 9){  
            alert("随机数是: "+number+"\n您猜对了");  
        }else{  
            alert("随机数是: "+number+"\n您猜错了");  
        }  
    }  
}
```

□ 对象分类

Example 2: **Data**对象。

Data对象来实现对日期和时间的控制。

常用的方法：

<code>getFullYear()</code>	// 获取年份
<code>getTime()</code>	// 获取时间
<code>setFullYear()</code>	// 设置具体的日期
<code>getDay()</code>	// 获取星期

更多知识参考：

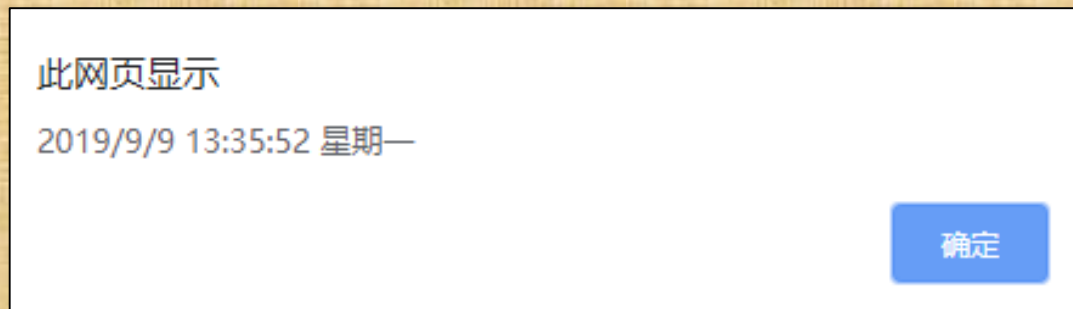
https://www.w3school.com.cn/js/js_date_methods.asp。

□ 对象分类

Example 2: **Data**对象。

示例 应用**Data**对象中的方法获取当前的完整年、月、日、星期、小时、分钟和秒数。

显示结果如下所示：



□ 对象分类

实现代码为：

```
var now = new Date();           //创建日期对象
var year = now.getFullYear();   // 获取当年年份
var month = now.getMonth()+1;  // 获取当前月份
var date = now.getDate();       // 获取当前日期
var day = now.getDay();         // 获取当前星期
var week;                       // 初始化变量
switch(day){
    case 1:                     // 判断变量day的值为1
        week = "星期一";      // 为变量week赋值
        break;
    case 2:
        week = "星期二";
        break;
    case 3:
        week = "星期三";
        break;
    case 4:
```


□ 对象分类

```
    week = "星期四";  
    break;  
    case 5:  
    week = "星期五";  
    break;  
    case 6:  
    week = "星期六";  
    break;  
    default:  
    week = "星期日";  
    break;  
}  
var hour = now.getHours();      // 获取当前的小时数  
var minute = now.getMinutes();  // 获取当前的分钟数  
var second = now.getSeconds();  // 获取当前的秒数  
// 输出年、月、日、时、分、秒、星期时间  
alert(year+"\\"+month+"\\"+date+" "+hour+": "  
      +minute+": "+second+" "+week);
```

□ 创建对象—对象直接量

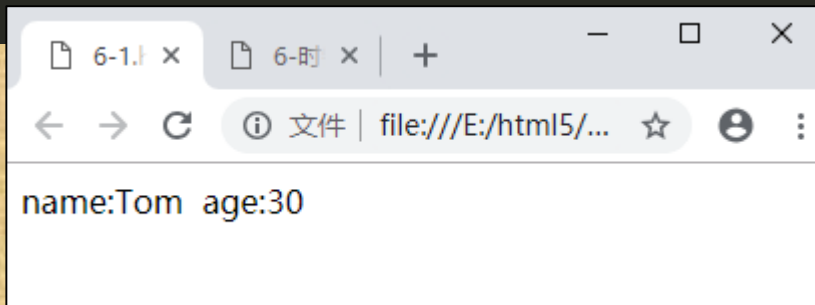
创建对象最简单的方式就是在JavaScript代码中使用对象直接量。

语法格式

```
var 对象名 = {  
    属性名1:属性值1,  
    属性名2:属性值2  
};
```

```
var people={name:"Tom",age:"30"};
document.write("name:"+people.name+"    "+ "age:"+people.
age);
```

结果如图：



□ 创建对象—通过new创建对象

new运算符创建并初始化一个新对象。关键字new后跟随一个函数调用。

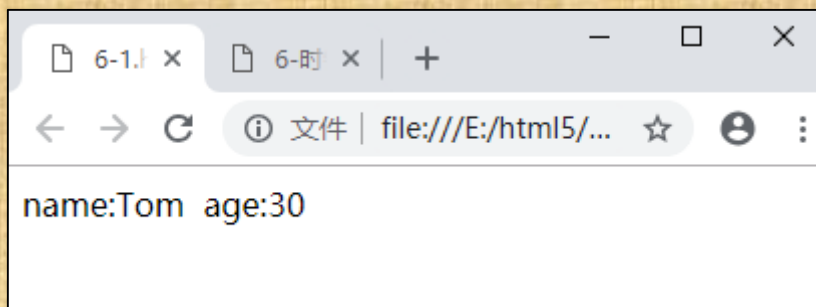
语法格式



```
var obj = new Object();
```

```
var people= new Object();  
people.name="Tom";  
people.age="30";  
document.write("name:"+people.name+"&nbsp;&nbsp; "+age:"+people.  
age);
```

结果如图:

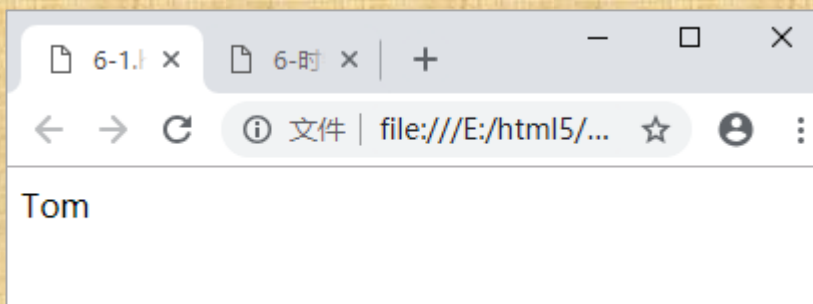


□ 创建对象—Object.create()

ECMAScript 5定义了一个名为Object.create()的方法，它创建一个新对象，其中第一个参数是这个对象的原型。Object.create()提供了第二个可选参数，用以对对象的属性进行进一步描述。

```
var obj={name:"Tom"};  
var newObj=Object.create(obj);  
document.write(newObj.name);
```

结果如图：

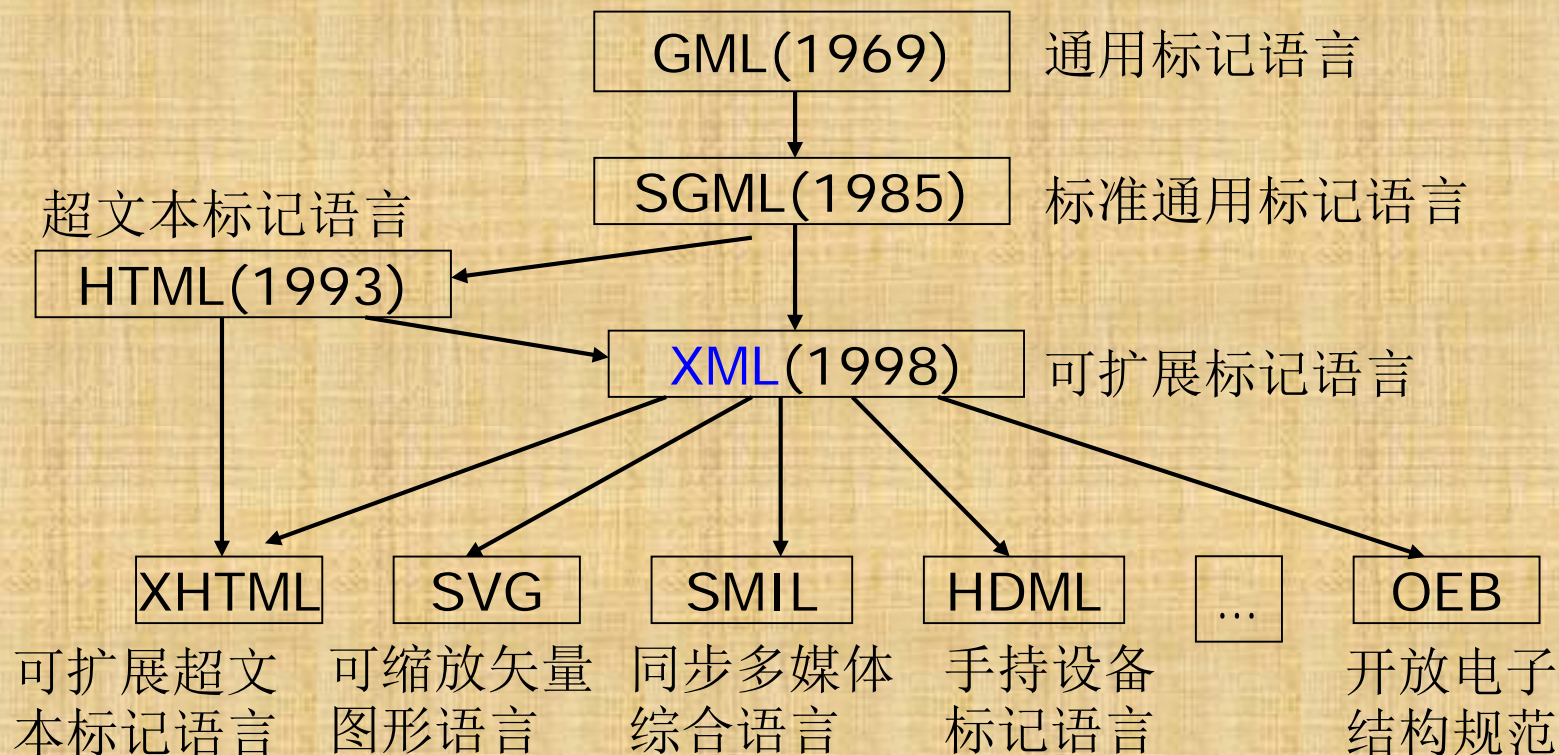


1.8 XML基础知识

- XML简介
- XML文档的组成
- XML文档的有效性
- XML的基本语法
- 元素的语法

□ XML简介—标记语言的发展

标记语言家谱表



□ XML简介—标记语言的发展

标记：为了处理的目的，在数据中加入的附加信息，又称置标。

标记语言：运用置标法描述结构化数据的形式语言，又称置标语言。

■ SGML (标准通用标准语言)

SGML实际上是一种通用的文档结构描述符号化语言，主要用来定义文献模型的逻辑和物理类结构。

■ HTML

HTML是设计用来做人机交流用的，HTML对外观、布局很擅长，但缺乏对内容，即咨询内涵表达的能力。

□ XML简介—标记语言的发展

■ XML

XML是Extensible Markup Language的缩写, 是一种可扩展性标记语言。XML是由World Wide Web Consortiu(W3C)的XML工作组定义。

扩展标记语言（XML）是SGML的子集，其目标是允许普通的SGML在Web上以目前HTML的方式被服务、接收和处理。XML被设计成易于实现，且可在SGML和HTML之间互相操作。

□ XML简介

例1 飞机票订单的HTML程序与XML

<!--显示飞机票订单的HTML程序-->

<h1>TICKET_ORDER</h1>

<p>ID:SD1803</p>

<p>FROM:北京</p>

<p>TO: 上海</p>

<p>DATE: 2016年5月20日</p>

<p>TIME: 8:30AM </p>

<p>PRICE:700.00 </p>

<p>AMOUNT:2 </p>

<p>TOTAL:1400.00 </p>

<!--保存飞机票订单的XML程序-->

<TICKET_ORDER>

<ID>SD1803</ID>

<FROM>北京</FROM>

<TO>上海</TO>

<DATE>

<YEAR>2005</YEAR>

<MONTH>5</MONTH>

<DAY>20</DAY>

</DATE>

<TIME>

<HOUR>8</HOUR>

<MINUTE>30</MINUTE>

</TIME>

<PRICE>700.00</PRICE>

<AMOUNT>2</AMOUNT>

<TOTAL>1400.00</TOTAL>

</TICKET_ORDER>

例2 客户信息的HTML程序与

<!--显示客户信息的HTML程序-->

张三

用户ID: 001

公司: A公司

EMAIL: zhang@aaa. com

电话: (010)62345678

地址: 五街1234号

城市: 北京市

省份: 北京

...

<!--显示客户信息的XML程序-->

<联系人列表>

<联系人>

<姓名>张三</姓名>

<ID>001</ID>

<公司>A公司</公司>

<EMAIL>zhang@aaa. com<
</EMAIL>

<电话>(010)62345678</电话>

<地址>

<街道>五街1234号</街道>

<城市>北京市</城市>

<省份>北京</省份>

</地址>

</联系人>

<联系人>

...

</联系人>

</联系人列表>

□ XML简介

表 HTML与XML的不同点比较

比较内容	HTML	XML
可扩展性	不具有扩展性	是元标记语言, 可用于定义新的标记语言
侧重点	侧重于如何表现信息	侧重于如何结构化地描述信息
语法要求	不要求标记的嵌套, 配对等, 不要求标识之间具有一定的顺序	严格要求嵌套, 配对, 并遵循DTD的树形结构
可读性及可维护性	难于阅读, 维护	结构清晰, 便于阅读, 维护
数据和显示的关系	内容描述与显示方式整合为一体	内容描述与显示方式相分离
与数据库的关系	没有直接联系	与关系型和层状数据库均可对应和转换
大小写敏感性	大小写不区分	大小写区分

□ XML简介—XML特点

每种语言的产生都能完成某些特定的功能，XML作为一种标记语言也不例外。XML最大的优势在于它能对各种编程语言编写的数据进行管理，使得在任何平台下都能通过解析器来读取XML数据。它的优势可归纳为以下几点：

数据的搜索

在XML中可以提取文档中任何位置的数据。

数据的显示

XML将数据的结构和数据的显示形式分开，根据需要使数据呈现出多种显示方式。如HTML、PDF等格式。

数据的交换

XML语法简单，可以通过解析器在任何机器上解读。并可以在各种计算机平台上使用。逐渐成为一种数据交换的语言。

□ XML简介—XML的应用前景

1.网络服务领域

2.EDI（电子数据交换）

XML数据接口会成为商业软件的标准配置。

3. 电子商务领域

XML推动EDI（Electronic Data Interchange）技术在电子商务领域的大规模应用。

4.数据库领域

XML—数据库—网页或文档中的表格这三者可以互相转换

□ XML简介—XML的应用前景

5.Agent（智能体）

XML能够更准确地表达信息的真实内容，其严格的语法降低了应用程序的负担，也使智能工具的开发更为便捷。

6. 软件设计元素的交换

XML也可以用来描述软件设计中有关的设计元素。

□ XML文档的组成

XML文档也属于纯文本文件，该文档一般如下四部分组成：

XML文档的声明

示例如下所示：

XML文档类型定义

XML文档注释

前三部分都是可选的

XML标识及其内容

```
<?xml version="1.0" encoding="UTF-8"?>
  <!--XML文档注释-->
  <?xml:stylesheet type="text/xsl"
    href="stu.xsl"?>
  <!--班级中学生的信息-->
  <class>
    <student>
      <name>Jone</name>
      <age>20</age>
    </student>
  </class>
```


□ XML文档有效性

结构良好的XML文档

如果某个文档符合XML语法规则，那么我们就说这个文档是“结构良好”的文档。

有效的XML文档

所谓有效的XML文档是指通过了DTD的验证的，具有良好结构的XML文档，在此大家要明白XML文档可分为结构良好的XML文档和有效的XML文档，以及他们之间的关系。即具有结构良好的XML文档并不一定就是有效的XML文档，反之一个有效的XML文档必定是一个结构良好的XML文档。

□ XML的基本语法

XML的语法规则

XML的语法规则既简单又严格，非常容易学习，在使用过程中只需认真仔细。一般 XML的语法规则大致可归纳为以下几点：

结束标记不可忽略

在HTML中某个标记有起始标记，却可以没有结束标记，但在XML文档中却不可以。

区分大小写

在XML中严格区分大小写，主要表现在开始标记和结束标记的大小写必须相同。还包括文档的声明部分和文档类型定义部分的大小写区分。

正确的嵌套包含

□ 元素

元素是XML文档的重要组成部分，在XML文档中必须存在元素。XML文档的元素一般是由标记头、标记末和标记间的字符串数据构成，如下代码所示：

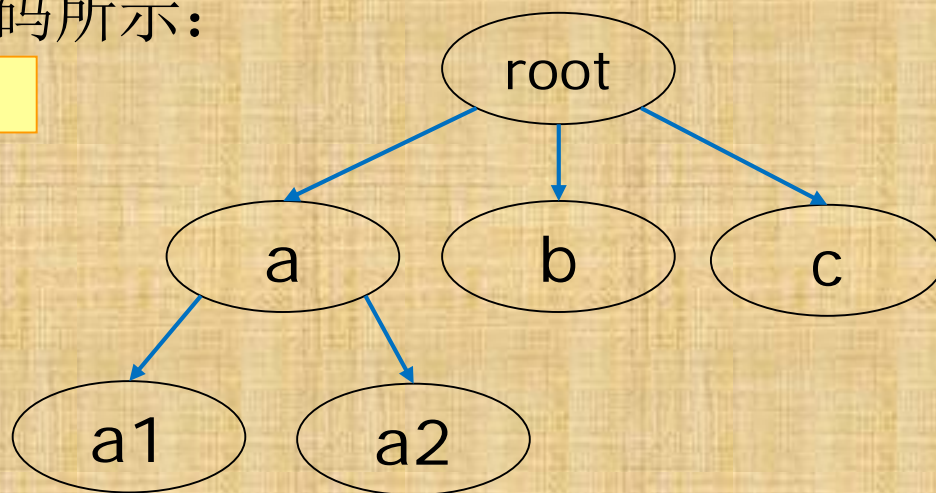
`<root>`

`<a>this is test`

`</root>`

元素a的值

元素a的元素名或标签名



XML文档中有且仅有一个根元素。标记间的字符串数据就是该元素的值，在XML中，如果元素的值中存在空格，那么这些空格将按原样解析出来

□ 实体

预定义实体表如下所示：

实体名	引用格式	表示的符号
lt	<	<
gt	>	>
amp	&	&
apos	'	'
quot	"	"

实体在XML文档中的一般引用格式如下：

&实体名;

□ 属性

属性是用来修饰某个元素的，如：

```
<root>
```

```
  <a attribute="aa">this is test</a>
```

```
</root>
```

属性名

属性值

关于元素的属性需注意如下几个问题：

- ① 属性的值必须用引号括起来，如： `attribute1="aa"` 或 `attribute3='aa'` ；
- ② 元素的属性以名和值成对出现；
- ③ 用来修饰同一个元素的属性的属性名不能相同 ；
- ④ 属性值不能包含 “&”、“'”、“<”等字符。

□ CDATA节

通过CDATA节可以通知分析器，在CDATA节包含的字符中没有标记。如果文档包含可能会出现标记字符，但我们又不是把它当作标记来使用，而只是属于文本字符，那么使用CDATA节来创建这样的文档就容易得多。CDATA节主要用于脚本语言内容、示例XML文档内容和HTML内容。如下所示：

```
<?xml version="1.0" encoding="gb2312"?>  
  <程序>
```

```
    <title>test</title>  
    <内容>  
    <![CDATA[  
        if(20<10){  
            return "你好";  
        }else{  
            return "hello";  
        }  
    ]]>  
    </内容>
```

注意：在“<![CDATA[”和“]]>”之间不能再加入CDATA节 或“]]>”。

```
</程序>
```

课后思考

- ❑ JS中空值(null)与字符串("")或0的区别?
- ❑ JS中空值(null)与undefined的区别?
- ❑ 常用的XML编辑器有哪些以及如何使用?
- ❑ XML与JSON的区别与优势?

1.9 小结

- ❑ JavaScript在HTML中的使用。
- ❑ JavaScript的数据类型种类。
- ❑ JavaScript常用的分支语句。
- ❑ 函数的定义与调用
- ❑ 对象的调用与创建
- ❑ XML基础知识