**04/03**

1. JavaScript基础介绍、实现、输出
2. 课程概要：JavaScript介绍、实现、输出、语法、注释、变量、数据类型。
3. 在HTML中，不限制脚本数量。
4. JavaScript代码可以放在<body>中、<head>中以及JavaScript文件中，当放在JavaScript文件中时，不需要<script>标签包裹。

<script>

document.write("jikexueyuan")

</script>

当JavaScript代码放在JavaScript文件中时，不需要<script>标签包裹。引入JavaScript文件的标签(<script src=”myjs.js”></script>)可以放在head中，也可以放在body中。

总结：①js代码有三个位置：html文档的head中、html文档的body中、js文件中。②js文件的引入有两个位置：html文档的head中、html文档的body中。

1. JS输出

①JS通常用来操作HTML（内容、属性、CSS样式、事件函数）；②文档输出：document.write(“<p>一段文字</p>”)

文档输出时，可输出字符串，甚至输出标签、变量。

注意：①document.write直接输出标签和字符串时需要加引号，否则出错；输出变量时不能加引号，否则按字符串处理。即：由于JavaScript不同数据类型的书写方式不同，标签和字符串类型需要加双引号，否则出错；其他数据类型不能加引号，否则按字符串处理。②document.write()括号不能省略。

1. 使用JS操作HTML

<p id="pid">原文字</p>

<script>

document.getElementById("pid").innerHTML="新文字";

</script>

注意：①document的首字母为小写。②输出的文字要加引号。

1. JS语法

①JS语句向浏览器发出命令，语句的作用是告诉浏览器该做什么。

②语句之间的分隔用分号（；）。

注意：分号是可选项，有时候看到不以分号隔开的。

③JS代码按照书写顺序依次执行。

④JS标识符必须以字母、下划线(\_)或美元符号($)开始。

⑤JS对大小写非常敏感，但会忽略多余的空格。

⑥保留字

1. 单行注释//代码 多行注释/\*代码\*/ 快捷键：Ctrl+/
2. 变量是用来储存信息的“容器”。

var i=10; //var：声明（标准） i：变量名称（属于标识符） 10：具体的值

//房子 宾馆 单人间；

var j=10; var m=i+j; document.write(m);

注意：document.write()括号中可以是字符串、标签、变量。

1. JS数据类型

字符串（String）；数字（Number）；布尔（Boolean）；数组（Array）；对象（Object）；空（null）；未定义；可以通过赋值为null的方式清除变量。

var str="hello"; //字符串

var i=5; //数字

var flag=true; //布尔

var n=null; //空

var r; //未定义

（1）var b=[2,6,1,”jike”,9];//三种数组的书写格式

（2）var b=new Array(2,6,1,”jike”,9); //中括号变小括号，加new Array。

（3）var b=new Array; b[0]=2;b[0]=2;b[1]=6;b[2]=1;b[3]=”jike”;b[4]=9;

document.write(b[3]);

注意：①数据类型是指变量值的数据类型，而不是变量名称的数据类型，变量名称不具有特殊性。②字符串放在双引号中，数组放在中括号中。③数组中不仅可以有数字，还可以有字符串。④可以通过赋值为null的方式清除变量。

1. JavaScript语法
2. 运算符、条件语句、循环语句、跳转语句。
3. 算数运算符：+、-、\*、/、%（取余）、++（自加）、--（自减）

方法：function 方法名称(){}

<p>25加16等于多少？</p>

<p id="pid"></p>

<button onclick="mySum()">计算</button>

<script>

function mySum(){

var i=25;

var j=16;

var s=i+j;

document.getElementById("pid").innerHTML=s

}

</script>

注意：①方法引入时不仅写方法名称，还要写()。②函数命名规则为动宾格式，并且第二个单词首字母大写。③var不要省。④i++意思为先赋值再加1，++i的意思是先加1再赋值；i--意思为先赋值再减1，--i的意思是先减1再赋值。

1. 赋值运算符：=、+=、-=、\*=、/=、%=；

var m=i+=j; // i+=j等价于i=i+j

i-=j //i=i-j i\*=j//i=i\*j i/=j//i=i/j

1. 字符串操作；

①被双引号所包裹的即是字符串，字符串可以是任何内容，包括数字等。②字符串相加是一个拼接的过程。③任何数据类型与字符串相加，都会被转化为字符串类型。字符串之前的数字相加以及字符串之后括号中的数字相加除外。

1. 比较运算符：==、===、!=（不等）、!==、>、<、>=、<=；

var a=10;

var b="10";

document.write(a==b); //返回true

注意：①比较运算符返回的是一个布尔（Boolean）类型数据true或者false。②==不识别数据类型，===识别数据类型。③!=不识别数据类型，!==识别数据类型。④>、<、>=、<=不识别数据类型（隐式转化）。⑤注意：=是赋值运算符，==才是比较运算符。

1. 逻辑运算符：&&（与）、||（或）、!（非）；

var a="10";

var b="7";

document.write(a>=20||!(b<5)); //返回true

注意：①逻辑运算符返回的也是一个布尔（Boolean）类型数据true或者false。②!取反运算符要与括号()结合使用。③&&（与）如果第一个比较运算符为false时，自动忽视第二个比较运算符。||（或）如果第一个比较运算符为true时，自动忽视第二个比较运算符。

var a=8;

var b=9;

alert(a<b||++b>=10); //弹出true

alert(b); //弹出9

var a=8;

var b=9;

alert(a>b||++b<10);//弹出false

alert(b);弹出10

7. 条件运算符（与if else类似）：

var a=10

document.write(a<11?"a小于11":"a大于等于11")

（条件？“满足条件输出”：“不满足条件输出”）

(表达式1)？(表达式2)：(表达式3)

注意：两个标点，第一个问号，第二个冒号。

1. if else条件语句

if(判断条件){若满足判断条件（返回true），则执行此大括号内的语句。}

else if(判断条件){若满足判断条件（返回true），则执行此大括号内的语句。}

……

else{若不满足以上所有判断条件（全返回false），则执行此大括号内的语句。}

注意：判断条件不是一个完整的语句，不加分号。

1. switch语句

switch(判断对象){case 数值:执行语句;break;case 数值:执行语句;break;……default:执行语句;break;}

var a=10;

switch(a){

case 1**:**

document.write("a=1");

break;

case 2**:**

document.write("a=2");

break;

default**:**

document.write("a不等于1也不等于2");

break;

注意：case后是一个具体的数值，具体的数值后为冒号，而不是分号。

1. 循环语句for循环

for(参数初始化表达式;条件表达式;更新循环变量表达式){循环体}

<script>

var i=[5,7,2,9,4,3]

for(var j=0;j<6;j++){

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

}

</script>

注意：①document.write()括号中可以使用加号+，以输出多项数据（字符串拼接用加号+）。②初始化表达式和变量表达式可以放在其他位置，但分号不可以省略。

var i=[5,7,2,9,4,3]

var j=0;

for(;j<6;){

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

j=j+1;

}

1. for in循环

<script>

var i=[5,7,2,9,4,3]

var j

for(j in i){ //如果j是i中的下标值（[]），则执行下列代码。

document.write(i[j]+"、");

}

</script>

1. while循环

while(条件表达式){循环体;变量表达式;}

<script>

var i=1;

while(i<10){

document.write("i="+i+"<br/>");

i=i+1;

}

</script>

注意：while循环与for循环的第二种写法的区别只有两点①单词for与while的区别；②条件表达式两边有无分号（；）的区别。

13. do while循环

do{循环体;变量表达式;}while(条件表达式)

<script>

var i=1;

do{document.write("i="+i+"<br/>");

i=i+1;}

while(i<10)

</script>

注意：do while与while的区别，do while循环是先执行第一次，然后再判断；while循环是先判断，然后再执行第一次。

1. break跳转语句与continue跳转语句

<script>

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

if(i==5){break;}

document.write("i="+i+"<br/>");

}

</script>

<script>

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

if(i==5){continue;}

document.write("i="+i+"<br/>");

}

</script>

注意：①break是结束当前循环，continue是跳过本次循环。②break跳转语句与continue跳转语句放在if条件语句中，if条件语句放在循环语句的循环体中。③break跳转语句与continue跳转语句放在循环体的起始位置，如果放在循环体的末尾位置，则执行循环体之后再判断，此时break跳转语句会多执行一次循环体之后结束循环，continu跳转语句执行当次循环之后跳出本次循环，执行下次循环，没有意义。

1. JavaScript函数
2. 了解函数、定义函数、函数调用、带参数的函数、带返回值的函数
3. 函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。函数即是方法。

例如：实现10组数字的和。

<script>

function demo(a,b){

var sum=a+b;

return sum;

}

alert(demo(13,32))

</script>

注意：①函数表达式：function 函数名(参数1，参数2){函数体;(代码块)}。②函数（方法）只有被调用才会生效，不被调用不生效，函数（方法）可以被多次调用。③return sum;不可省，因为函数必须返回函数值才有被调用的意义。④return sum;中间没有冒号。⑤函数被调用时，参数应赋具体值。

1. 函数表达式：function 函数名(参数1，参数2){函数体；(代码块)}，参数即是函数中的变量，函数体即是调用函数时所具体执行的代码块。

<script>

function demo(){

var a=10;

var b=5;

alert(a\*b)

}

demo()

</script>

注意：①JavaScript对大小写十分敏感，这里的function必须小写。在函数调用时，也必须按照函数的相同名称并大小写一致来调用函数。②函数采用驼峰式命名规则，第二个单词首字母应为大写。

1. 调用函数：函数在定义好之后，不能自动执行，需要进行调用。

调用方式：①在<script>标签内进行调用；②在HTML文件中进行调用。

<script>

function demo(){

var a=10;

var b=5;

var m=a+b

alert(m)

}

demo()

</script>

<script>

function demo(){

var a=10;

var b=5;

var m=a+b

alert(m)

}

</script>

<button onclick="demo()">按钮</button>

注意：①函数只能写在<script>标签内，但既可以在<script>标签内（function函数后）进行调用，又可以在HTML文件中进行调用。②函数在<script>标签内调用时，调用位置是在函数之后并且在<script>标签内。③函数不可能在函数内调用。④在HTML文件中进行调用时，需要将函数demo()设置为标签的onclick事件（可以是其他事件）的属性值。

1. 带参数的函数

在函数的调用中，也可以传递一些值，这些值被称为参数。例如：demo(a1,a2)

参数的个数可以为任意多，每个参数通过逗号”,”隔开。参数在传递时，其顺序必须一致。函数通过传递参数个数的不同以及参数类型的不同而完成不同的功能。

<script>

function demo(a,b){

var sum=a+b;

alert(sum);

}

demo(13,32);

demo(10,20)

</script>

<script>

function demo(name,age){

alert("我的名字是"+name+"，我的年龄是"+age);

}

</script>

<button onclick="demo('iwen','20')">按钮</button>

<button onclick="demo('lili','16')">按钮</button>

注意：①JS为弱类型语言，在JS函数中中传递参数不需要声明类型。（即：函数中的参数不需要声明。）②带参数的函数可以使函数在多次调用过程中给参数赋不同的值，以得到不同的结果。③与document.write()类似，alert()括号中可以使用加号+，以输出多项字符串（字符串拼接），或输出多项数字的和（算数运算符）。④双引号里包裹的字符串必须用单引号，不可再用双引号。

1. 带返回值的函数

返回值：有时，我们需要将函数的值返回给调用他的地方。通过return语句可以实现返回值。注意，在使用return语句时，函数会停止执行，同时返回值，return语句后不允许添加任何代码，因为return语句后的代码不会被执行。

<script>

function demo(){

alert("我会执行吗1？") //alert会执行，并执行2次。

return "hello";

alert("我会执行吗2？") //alert不会被执行。

}

var vl1=demo()+"：iwen";

var vl2=demo()+"：iwen";

alert(vl1);

alert(vl2);

</script>

注意：①给变量声明时，可以利用算数运算符“+”进行字符串拼接。②return后没有小括号，而alert后有小括号。③hello为字符串数据类型，必须加引号。

<p id="pid"></p>

<script>

function demo(a,b){

if(a>b){

return "a比较大";

}else{

return "b比较大";

}

}

document.getElementById("pid").innerHTML=demo(10,20);

</script>

注意：①getElementById("pid")双引号不可以省略。②此案例为在<script>标签内调用函数。③document.getElementById("")语句可以寻找（定位）到该html元素，进而可以通过js语句来修改元素内容及属性值。

1. 局部变量与全局变量

局部变量：声明在函数之内的，并且使用var进行声明的变量。

全局变量：声明在函数之外的变量是全局变量；声明在函数之内的，但没有使用var进行声明的变量，在函数被调用后是全局变量。

<script>

var n = 10; //全局变量，任何地方都可以使用。

m=10; //全局变量，任何地方都可以使用。

function demo(){

var a=10; //局部变量，只能在当前函数内使用。

x=10; //在当前函数被调用后是全局变量，任何地方都可以使用。

}

</script>

注意：①所有的变量都必须声明在<script>标签内。②所有变量都必须先声明后使用。

1. JavaScript异常捕获、事件
2. 异常

当JavaScript引擎执行JavaScript代码时，发生了错误，导致程序停止运行。

1. 异常抛出

当异常产生，并且将这个异常生成一个错误信息。

1. 异常捕获

try{

发生异常的代码块;

}catch(err){

错误信息处理;

}

例如： try{

alert(str);

}catch(err){

alert(err);

}

注意：try后大括号内的代码一定会被执行（一定会检查此代码有无错误），而catch(err)后大括号内的代码只有检查出错误时才会被执行。

4. Throw语句

通过throw语句创建一个自定义错误。

<form>

<input type="text" id="txt" />

<button onclick="demo()">按钮</button>

</form>

<script>

function demo(){

try{

var e=document.getElementById("txt").value

if(e==""){

throw "请输入";

}

}catch(err){

alert(err);

}

}

</script>

注意：throw必须与try、catch配合使用。如果将throw改为alert，可单独使用。

<form>

<input type="text" id="txt" />

<button onclick="demo()">按钮</button>

</form>

<script>

function demo(){

var e=document.getElementById("txt").value

if(e==""){

alert("请输入")

}

}

</script>

1. JavaScript事件

事件是可以被JavaScript侦测到的行为。

常见事件：onClick单击事件；onMouseOver鼠标经过事件；onMouseOut鼠标移出事件；onChange文本内容改变事件；onSelect文本框选中事件；onFocus光标聚集事件；onBlur移开光标事件；onLoad网页加载事件；onUnload关闭网页事件。

onClick单机事件；

<button onclick="demo()">按钮</button>

<script>

function demo(){

alert("hello")

}

</script>

注意：JavaScript事件的用法：在HTML文件中进行函数调用时，可以将函数demo()设置为标签的onclick事件（可以是其他事件）的属性值。即：事件相当于HTML元素的属性，函数（方法）在HTML文件中调用时（可以在<script>标签内进行调用）相当于HTML元素事件属性的属性值。即：HTML元素可以包含事件属性，事件属性可以有函数属性值。

onMouseOver鼠标经过事件；onMouseOut鼠标移出事件；

<div style="width: 100px;height: 100px;background-color: cadetblue;" id="div1" onmouseover="onover(this)" onmouseout="onout(this)">

</div>

<script>

function onover(a){

a.innerHTML="hello"

}

function onout(b){

b.innerHTML="word"

}

</script>

注意：**如果HTML标签的事件属性对应的函数作用于此标签本身时，调用函数时的实际参数为this。此时函数的参数为调用该函数的元素本身。**当用函数来创建对象时，函数即为所要创建的对象，此时在函数中通过this.a=”属性值”定义对象的属性值。（P17）即：当this为调用该函数的**实际参数**时，**指代该元素**；当this为function函数中的**形式参数**时，**指代调用该函数的对象。**

例子： <script>

function people(a,b){

this.name=a;

this.age=b;

}

son=new people("iwen",20)

document.write("姓名："+son.name+";年龄："+son.age)

</script>

注意：①创建具体对象时，new不可以省略。（P17自定义对象的第一种方式）

onChange文本内容改变事件；

第一种写法：

<input type="text" onchange="changedemo()"/>

<script>

function changedemo(){

alert("内容改变了！")

}

</script>

第二种写法：

<body>

<input type="text" onchange="alert('内容改变了!')"/>

</body>

注意：HTML元素事件属性的属性值不仅可以是JavaScript函数（方法），还可以是JavaScript语句。

例如：<div id="div"onmouseover="this.style.width='200px';this.style.height='200px';

this.style.backgroundColor='black';"onmouseout="document.getElementById('div').style.width='100px';document.getElementById('div').style.height='100px';document.getElementById('div').style.backgroundColor='red';"></div>

onSelect文本框选中事件；onFocus光标聚集事件；

<input type="text" **onselect="changedemo(this**)" onfocus="changedemo2(this)"/>

<script>

function changedemo(a){

a.style.backgroundColor="red";

}

function changedemo2(a){

a.style.backgroundColor="blue";

}

</script>

注意：①JavaScript函数可以用来修改HTML元素样式。②利用JavaScript函数修改HTML元素的CSS样式时，参数之后的单词是style而不是type。

onLoad网页加载事件；

<body onload="mag()">

<script>

function mag(){

alert("网页加载完成")

}

</script>

</body>

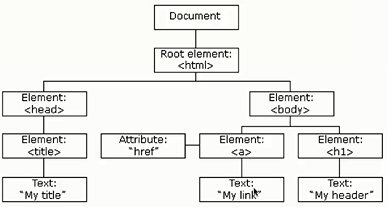
注意：并不是每一个HTML元素都拥有所有的事件属性，只有该HTML元素可以执行某事件时，才会匹配相对应的事件属性。例如：onChange文本内容改变事件只有包含文本内容的HTML元素才有此事件属性。

**04/04**

1. JavaScriptDOM对象

1. DOM简介、DOM操作HTML、DOM操作CSS、DOM EventListener（句柄）

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



注意：我们可以通过文档对象模型（DOM）来修改元素内容（innerHTML）以及属性值（style）。

1. DOM功能

a：JavaScript能够改变页面中所有的HTML元素；

b：JavaScript能够改变页面中所有的HTML属性；

c：JavaScript能够改变页面中所有的CSS样式；

d：JavaScript能够对页面中所有事件做出反应。

1. DOM操作HTML

(1)改变HTML输出流；

注意：绝对不要在文档加载完成之后使用document.write()。这会覆盖该文档

<p>hello</p>

<p>hello</p>

<p>hello</p>

<button onclick="demo()">按钮</button>

<script>

function demo(){

document.write("word");//点击按钮，”word”会覆盖所有内容。

}

</script>

注意：①function首字母小写；②function与demo之间没有点，只有空格。

(2)寻找元素

通过id找到HTML元素

通过标签名找到HTML元素

<p id="pid">hello</p>

<button onclick="demo()">按钮</button>

<script>

function demo(){

// document.getElementById("pid").innerHTML="<h1>word</h1>"

第二种写法： document.getElement**s**ByTagName("p")[0].innerHTML="<h1>word</h1>"

}

</script>

注意：①由于JavaScript不同数据类型的书写方式不同，标签和字符串类型需要加双引号，否则出错；其他数据类型不能加引号，否则按字符串处理。②通过标签名寻找HTML元素时，定位到是所有的p元素。

1. 改变HTML内容

使用属性：innerHTML

1. 改变HTML属性

<img id="imgid" src="img/009.jpg" />

<button onclick="demo()">按钮</button>

<script>

function demo(){

document.getElementById("imgid").src="img/010.jpg"

}

</script>

注意：使用JavaScript改变HTML元素属性与改变HTML元素CSS样式不同。在改变元素属性时，通过id或标签名定位到该元素之后，直接加点属性名即可；而改变元素的CSS样式时，通过id或标签名定位到该元素之后，需要先加点style之后，才可加点CSS样式属性名。

1. DOM操作CSS

通过DOM对象改变CSS语法： document.getElementById(“id”).style.property=new style

举例：<div id="div1" style="width: 100px;height: 100px;">hello</div>

<button onclick="demo()">按钮</button>

<script>

function demo(){

document.getElementById("div1").style.background="red";

}

</script>

1. DOM EventListener（给元素添加事件句柄）
2. DOM EventListener：

方法：addEventListener() removeEventListener():

1. addEventListener() 向指定元素添加事件句柄

removeEventListener() 移除添加的事件句柄

语法：寻找元素.addEventListener(“事件”，函数)

寻找元素.removeEventListener(“事件”，函数)

注意：①在添加句柄时，事件用双引号包裹并且省略on。②添加句柄时，函数不可用双引号包裹，并且或者写完整函数，或者只写函数名称，不加括号()。（DOM0级事件处理程序也一样，点事件=无括号函数名称或者函数全称。）

句柄的优点：①将元素、事件与函数在<script>标签内直接相连，减化了HTML元素调用函数的过程（如果将事件直接添加到HTML结构中，事件和函数名称分别为HTML元素的属性和属性值，函数的内容在<script>标签内。）并且当只添加一个事件时不必定义函数名称，避免了修改函数名称时的工作量。即：句柄减化了函数调用流程。②句柄添加多个事件不会覆盖。

例子1：<button id="btn">按钮</button>

<script>

document.getElementById("btn").addEventListener("click",function(){alert("hello")})

</script>

例子2： <script>

x=document.getElementById("btn")

//减化代码内容

x.addEventListener("click",hello);

//添加句柄

x.addEventListener("click",word);

//添加句柄

x.removeEventListener("click",hello);

//移除句柄

function hello(){

alert("hello")

}

function word(){

alert("word")

}

</script>

1. JavaScript事件详解
2. 事件流、事件处理、事件对象
3. 事件流：描述的是在页面中接收事件的顺序，分为事件冒泡和事件捕获。

事件冒泡：由最具体的元素接收，然后逐级向上传播至最不具体的元素的节点（文档）。

事件捕获：由最不具体的节点先接收事件，而最具体的节点应该是最后接收事件。

3. 事件处理

（1）HTML事件处理程序

直接添加到HTML结构中 例如：button的onclick属性

例子：<button onclick="demo()">按钮</button>

<script>

function demo(){

alert("html事件处理程序")

}

</script>

缺点：若修改一个函数的名称，则需要修改两处，HTML事件属性的函数名称属性值一处，<script>标签内的函数名称一处。

1. DOM0级事件处理程序

把一个函数赋值给一个事件处理程序属性。

语法：寻找元素.事件=函数

例子：<button id="btn">按钮</button>

<script>

document.getElementById("btn").onclick=function(){alert("DOM0级事件处理1")};

document.getElementById("btn").onclick=function(){alert("DOM0级事件处理2")};

document.getElementById("btn").onclick=null;

document.getElementById("btn").onclick=function(){alert("DOM0级事件处理3")};

</script>

优点：若修改一个函数的名称，只需修改一处，即<script>标签内的函数名称。

缺点：当添加多个事件处理程序时，后者会覆盖前者。

注意：DOM0级事件处理与DOM2级事件处理的区别：①DOM0级事件名称有on无引号，DOM2事件名称无on有引号。②DOM0级寻找到元素之后是点事件等于函数，DOM2寻找到函数之后是点添加句柄（addEventListener）括号（在DOM2中没有等号）。

（3）DOM2级事件处理程序

addEventListener(“事件名”，“事件处理函数”，“布尔值”)；

removeEventListener()

true：事件捕获

false：事件冒泡

语法：寻找元素.addEventListener(“事件名”，事件处理函数，布尔值)

<button id="btn">按钮</button>

<script>

var btn1=document.getElementById("btn");

btn1.addEventListener("click",demo1);

btn1.addEventListener("click",demo2);

btn1.addEventListener("click",demo3);

btn1.removeEventListener("click",demo3);

function demo1(){

alert("DOM2级事件处理程序1")

}

function demo2(){

alert("DOM2级事件处理程序2")

}

function demo3(){

alert("DOM2级事件处理程序3")

}

</script>

优点：①若修改一个函数名称，只需修改一处，即<script>标签内的函数名称。②当添加多个事件处理程序时，后者不会覆盖前者。

1. IE事件处理程序（IE≤8）

attachEvent

detachEvent

语法：与DOM2级事件处理程序只有两个区别：①单词不同；②DOM2事件名称没有on，IE事件处理程序中事件名称有on。

举例：<button id="btn">按钮</button>

<script>

var btn1=document.getElementById("btn");

if(btn1.addEventListener()){

btn1.addEventListener("click",demo)

}else if(btn1.attachEvent()){

btn1.attachEvent("**on**click",demo)

}else{

btn1.onclick=demo()

}

function demo(){

alert("hello")

}

</script>

1. 事件对象

在触发DOM事件的时候都会产生一个对象。

事件对象event：事件对象有很多属性和方法

属性：（1）type:获取事件类型；（2）target:获取事件目标；

方法：（1）stopPropagation()停止事件冒泡；（2）preventDefault()阻止事件默许行为。

属性：（1）type:获取事件类型；（2）target:获取事件目标；

举例：

<button id="btn">按钮</button>

<script>

var btn1=document.getElementById("btn");

btn1.addEventListener("click",showtype);

function showtype(event){

alert(event.type);

alert(event.target);

}

</script>

阻止事件冒泡举例：

<div id="div">

<button id="btn">按钮</button>

</div>

<script>

var btn1=document.getElementById("btn");

var div=document.getElementById("div")

btn1.addEventListener("click",showtype);

div.addEventListener("click",showdiv);

function showtype(){

alert("按钮");

event.stopPropagation(); //停止事件冒泡（传播）

}

function showdiv(){

alert("盒子");

}

</script>

阻止事件冒泡与阻止事件默许行为举例：

<div id="div">

<a id="aid" href="http://www.baidu.com">跳转</a>

</div>

<script>

document.getElementById("div").addEventListener("click",demo1) document.getElementById("aid").addEventListener("click",demo2)

function demo1(){

alert(event.type)

}

function demo2(){

event.stopPropagation();

event.preventDefault();

}

</script>

注意：①在JavaScript中，若变量有几个单词拼接而成，除第一个单词外，后面的单词首字母必须大写。②事件对象（属性、方法）在使用时，书写在事件对象点之后（event.type属性/target属性/stopPropagation()方法/preventDefault()方法）。③事件对象的两个属性和两个方法都书写在函数中，但针对的是函数所对应的事件。④书写函数时，function不要忘记。

1. JavaScript内置对象
2. 什么是对象、String字符串对象、Date日期对象、Array数组对象、Math对象
3. 什么是对象：JavaScript中的所有事物都是对象，字符串、数值、数组、函数…

每个对象都带有属性和方法：对象的属性即对象带有一些特定的性质，例如图像的长宽，a标签的外链接；方法是对象的使用方式，例如日期的方法让我们得到时间，包括年月份。

JavaScript允许自定义对象。（字符串、数值、数组、函数…为JavaScript内置对象）

1. 自定义对象有两种实现方式：
2. 定义并创建对象实例；

语法：变量=new Object();

变量.属性=“属性值”

例子：<script>

people=new Object();

people.name="iwen";

people.age="20";

document.write("姓名："+people.name+";年龄："+people.age)

</script>

语法：变量={属性：“属性值”，属性：“属性值”…}

例子： <script>

people={name:"iwen",age:"20"};

document.write("姓名："+people.name+";年龄："+people.age)

</script>

注意：①字符串连接用加号+。②对象属于JavaScript数据类型的一种（P2），声明对象属于声明变量中的一种。（var 变量=new Object()）。③声明变量时，属性值如果是字符串数据类型，用双引号包裹。④两种定义并创建对象对比：等分号，冒逗号。

1. 使用构造函数来定义**类**，然后创建新的对象实例。

例子：<script>

function people(a,b){

this.name=a;

this.age=b;

}

son=new people("iwen",20)

document.write("姓名："+son.name+";年龄："+son.age)

</script>

注意：①new不可省略。

②如果HTML标签的事件属性对应的函数作用于此标签本身时，调用函数时的实际参数为this。此时函数的参数为调用该函数的元素本身。（P10）当用函数来创建对象时，函数中通过this.a=”属性值”定义对象的属性值。即：this放在HTML元素函数属性值引用的参数中时（实参），指代该元素；this放在<script>标签中函数表达式中时（形参），指代实例对象。

例子： <div style="width: 100px;height: 100px;background-color: cadetblue;" id="div1" onmouseover="onover(this)" onmouseout="onout(this)">

</div>

<script>

function onover(a){

a.innerHTML="hello"

};

function onout(b){

b.innerHTML="word"

};

</script>

1. JavaScript内置对象——String字符串对象

String对象用于处理已有的字符串，字符串可以用双引号或单引号。

内置对象都带有属性和方法。获取属性不加括号，使用方法要加括号。

一切事物皆对象，对象都拥有属性和方法。属性即对象带有的一些**特定性质**，方法是对象的**使用方式**。获取属性（特定性质）不加括号，使用方法（使用方式）要加括号。

String字符串对象带有的属性和方法有：

字符串属性：length 、prototype、constructor

字符串方法：charAt()、charCodeAt()、concat()、formCharCode()、indexOf()、 lastIndexOf()、march()、replace()、search()、slice()、substring()、substr()、valueOf()、 toLowerCase()、toUpperCase()、split()。

（1）**length属性**；

例子： <script>

var a="hello world"

document.write("字符串长度为："+a.length)

</script>

1. 在字符串中查找字符串的方法：index**Of**(“索引的内容”)

在字符串中查找当前字符串是否存在，如果存在，则返回位置；如果不存在，则返回-1。 注：index 索引

例子：<script>

var a="hello world"

document.write(a.indexOf(“o w"))

</script>

1. 内容匹配的方法：match(“匹配的内容”)

如果字符串中有该内容，则返回该匹配内容，如果字符串中没有该内容，则返回null。

例子：<script>

var a="hello world"

document.write(a.match("word"))

</script> //返回null。

（4）替换内容的方法：replace(“原内容”，“新内容”)

例子： <script>

var a="hello world"

document.write(a.replace("world","世界"))

</script>

1. 字符串大小写转换：toUpperCase()/toLowerCase()

例子：<script>

var a="hello world"

document.write(a.toUpperCase())

</script>

注意：字符串大小写转换：toUpperCase()/toLowerCase()，括号中无内容。

1. 字符串转化为数组：split()

语法：字符串.split(“分隔符”)即：将字符串以分隔符分隔成为数组。

例子：<script>

var a="he llo wor ld"

var b=a.split(" ") //语法：b为字符串a以空格分隔而成的数组。

document.write(b[1]);

</script>

1. JavaScript内置对象——Date日期对象：日期对象用于处理日期和时间。

例子：<script>

var a=**new** Date();

document.write(a)

</script>

注意：new不可以省略。

Date对象的常用方法：getFullYear() 获取年份；**getTime()获取毫秒**；getDay()获取星期；setFullYear()设置具体的日期。

注意：括号中无内容。

例子： <script>

var a=new Date()

document.write(a.getFullYear())

</script>

例子：<script>

var a=new Date()

a.setFullYear(2016,7,15)

document.write(a)

</script>

时钟实例1：

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

<script>

function time(){

var a=new Date();

var h=a.getHours();

var m=a.getMinutes();

var s=a.getSeconds(); //t=d.toLocaleTimeString(); (p26对比)

var t=setTimeout(function(){

time()

},1000)

if(m<10){

m="0"+m

}

if(s<10){

s="0"+s

}

document.getElementById("div").innerHTML=h+":"+m+":"+s

}

time()

</script>

与P26案例的类似写法：

时钟实例2：

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

<script>

function time(){

var a=new Date;

var h=a.getHours();

var m=a.getMinutes();

var s=a.getSeconds(); //t=d.toLocaleTimeString(); (p26对比)

if(m<10){

m="0"+m

}

if(s<10){

s="0"+s

}

document.getElementById("div").innerHTML=h+":"+m+":"+s

}

t=setInterval(function(){

time()

},1000)

</script>

对比：这里的案例为计时方法写在时钟函数中（用setTimeout()），P26的案例为计时方法与时钟函数并列（用setInterval()）。两个方式的共同点：计时方法的调用函数为时钟函数。

当用setInterval()时，计时函数也可以写在时钟函数之中，但是此时无法通过clearInterval()停止时钟函数。

例如：<button id="btn" onclick="stop()">按钮</button>

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

<script>

function time(){

var a=new Date;

var h=a.getHours();

var m=a.getMinutes();

var s=a.getSeconds(); //t=d.toLocaleTimeString(); (p26对比)

if(m<10){

m="0"+m

}

if(s<10){

s="0"+s

}

document.getElementById("div").innerHTML=h+":"+m+":"+s

t=setInterval(function(){

time()

},1000)

}

time()

function stop(){

clearInterval(t)

}

</script>

此时点击按钮，时钟函数无法停止。解决方案：计时函数位置转移到时钟函数之外。

1. JavaScript内置对象——Array数组对象
2. Array对象：使用单独的变量名来存储一系列的值。
3. 数组的创建：例如var arr=[“Hello”,”iwen”,””ime]；
4. 数组的访问：通常指定数组名以及索引号码，可以访问特定元素。

注意：[0]是数组的第一个元素。[1]是数组的第二个元素。

1. 数组的常用方法：

concat() 合并数组；sort() 排序；push() 末尾追加元素；reverse()数组元素翻转。

例子： concat() 合并数组：

<script>

var a=["e","y","x"];

var b=[1,5,3];

var c=a.concat(b);

document.write(c);

</script>

sort() 排序：

<script>

var a=["e","y","x"];

document.write(a.sort());

</script>

倒序：

<script>

var a=[3,8,2];

document.write(a.sort(function(a,b){

return b-a;

}

));

</script>

注意：改变数组排序规则时，数组.sort()小括号之间必须写完整的函数。

push() 末尾追加元素：

<script>

var a=[3,8,2];

a.push(5);

document.write(a);

</script>

问题：若<script>

var a=[3,8,2];

document.write(a.push(5));//返回值为4

</script>

reverse()数组元素翻转：

<script>

var a=[3,8,2];

document.write(a.reverse());

</script>

1. JavaScript内置对象——Math对象

Math对象：执行常见的算数任务。

常用方法：round()四舍五入；random()返回0到1之间的随机数；max()返回最大值；min()返回最小值；abs()返回绝对值；floor()取整数。

举例：round()四舍五入：

<script>

document.write(Math.round(2.5));

</script>

random()返回0到1之间的随机数：

<script>

document.write(Math.random());

</script>

返回0到10之间的随机整数：

<script>

var a=Math.random()\*10

document.write(**Math.round(a)**);

</script>

max()返回最大值

<script>

document.write(Math.max(10,3,5,35,32));

</script>

abs()返回绝对值

<script>

var a=-4.5

document.write(Math.abs(a));

</script>

floor()取整数

<script>

var a=4.5

document.write(Math.floor(a));

</script>

注意：Math对象方法与其他对象方法的用法差异。Math对象方法从Math.开始；其他对象方法一般从已声明的变量加点开始。

1. JavaScript DOM对象控制HTML
2. DOM对象控制HTML
3. 方法：getElementsByName() ——获取name

getElementsByTagName() ——获取元素

getAttribute() ——获取元素属性

setAttribute() ——设置元素属性

childNodes() ——访问子节点

parentNode() ——访问父节点

creatElement() ——创建元素节点

creatTextNode() ——创建文本节点

insetBofore() ——插入节点

removeChild() ——删除节点

offsetHeight ——网页尺寸

scrollHeight ——网页尺寸

例子：（1）getElementsByName() ——获取name

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<script>

function getname(){

var a=document.getElementsByName("pn");

alert(a.length); //弹出5，说明获取到了所有name为“pn”的元素。通过一个集合来存储，我们可以通过下标来获取集合中某一个元素。

var b=a[2];

b.innerHTML="word";

}

getname()

</script>

注意：即使只有一个name为pn的p元素，a仍是一个数组（a不等于唯一的p元素）。如果想修改a中唯一的p元素，仍然需要通过下标值来寻找（定位）到该元素。

（2）getElementsByTagName() ——获取元素

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<p name="pn">hello</p>

<script>

function getname(){

var a=document.getElementsByTagName("p");

alert(a.length);//弹出5，说明获取到了所有p元素。通过一个集合来存储，我们可以通过下标值[0]来获取集合中某一个元素。

b=a[2]

b.innerHTML="word";

}

getname()

</script>

注意：即使只有一个p元素，a仍是一个数组（a不等于唯一的p元素）。如果想修改a中唯一的p元素，仍然需要通过下标值[0]来寻找（定位）到该元素。

（3）getAttribute() ——获取元素属性 attribute：属性；特质

<a href="#" id="aid"></a>

<script>

function getattr(){

var a=document.getElementById("aid");

var b=a.getAttribute("href");

alert(b);

}

getattr()

</script>

注意：获取元素属性时，需要已经获取该元素的前置条件，即元素属性只能从已锁定的元素中获取。

（4）setAttribute() ——设置元素属性

语法：setAttribute(“属性名称”，“新的属性值”)

注意：属性名和属性值都要加引号。

<a href="#" id="aid" title="这是a标签">hello</a>

<script>

function setattr(){

var a=document.getElementById("aid");

var b=a.setAttribute("title","这是a标签的动态属性值")

var c=a.getAttribute("title")

alert(c)

}

setattr()

</script>

方法2（P12）：

<a href="#" id="aid" title="这是a标签">hello</a>

<script>

function setattr(){

var a=document.getElementById("aid");

a.title="这是a标签的动态属性值";

var c=a.getAttribute("title");

alert(c);

}

setattr()

</script>

（5）childNodes() ——访问子节点（子元素）

<script>

function getchildnode(){

var a=document.getElementById("uid");

var b=a.childNodes;

alert(b.length);

alert(b[0].nodeName)

}

getchildnode();

</script>

注意：childNodes没有单词get。

1. parentNode() ——访问父节点

<div>

<p id="pid"></p>

</div>

<script>

var div=document.getElementById("pid").parentNode;

alert(div.nodeName); //弹出“DIV”

</script>

1. creatElement() ——创建元素节点

语法：document.createElement(“节点元素名称”)

注意：双引号不能省略。

两个例子：<body>

<div id="div" style="background-color: red;">

</div>

<script>

var div=document.getElementById("div")

// var body=document.body;

var ipt=document.createElement("input");

ipt.type="button";

ipt.value="按钮";

// body.appendChild(ipt);

div.appendChild(ipt)

</script>

</body>

注意：①在创建元素（节点）之后，输出此元素节点的固定语法为：父元素.appendChild(创建节点名称)。②父元素.appendChild()可以插入节点，但是插入的子节点在父元素的末尾位置。（append：附加、贴上）③获取body元素的简便方法document.body。

1. creatTextNode ——创建文本节点
2. insetBofore() ——插入节点

语法：父节点.insertBefore(新子节点,老子节点);

<div id="div">

<p id="pid">这是div的p元素</p>

</div>

<script>

var div=document.getElementById("div");

var p=document.getElementById("pid");

var newnode=document.createElement("p");

newnode.innerHTML="添加的p元素";

div.insertBefore(newnode,p);

</script>

1. removeChild() ——删除节点

语法：父元素.removeChild(父元素.childNodes[1])

<div id="div">

<p class="pid">这是div的p元素</p>

</div>

<script>

div=document.getElementById("div")

div.removeChild(div.childNodes[1])

</script>

1. offsetHeight ——网页尺寸（不包含滚动条）（P31）

<script>

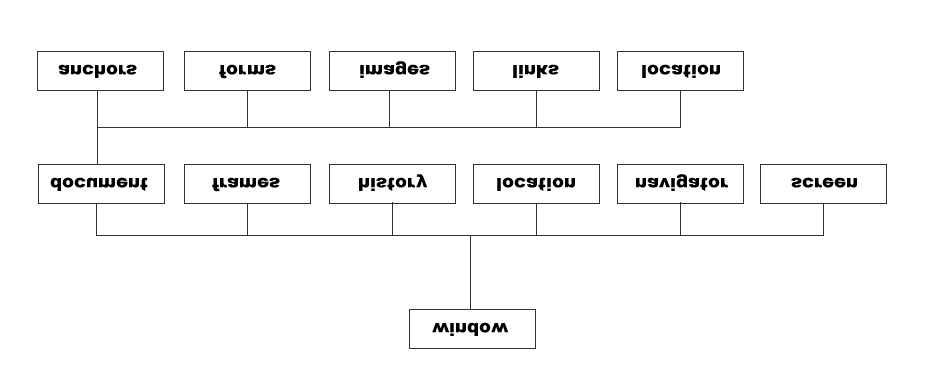
var width=document.documentElement.offsetWidth||document.body.offsetWidth;

var heitht=document.documentElement.offsetHeight||document.body.offsetHeight;

alert(width+","+heitht)

</script>

1. scrollHeight ——网页尺寸
2. JavaScript浏览器对象
3. Window对象、计时器、History对象、Location对象、Screen对象、Navigator对象、弹出窗口、Cookies。



2. window对象：

window对象是BOM的核心，window对象指当前的浏览器窗口。

所有JavaScript全局对象、函数以及变量均自动生成window对象的成员。

**全局变量是window对象的属性**

**全局函数是window对象的方法**

甚至HTML DOM的document也是window对象的属性之一

注：window.document widow一般省略不写。

注意：window对象是客户端JavaScript中最高层对象之一，**当我们打开浏览器窗口时，不管这个窗口是否打开了网页，都会创建一个window对象实例。**

3. window尺寸：

window.innerHeight - 浏览器窗口的内部高度（不包含滚动条）

window.innerWidth - 浏览器窗口的内部宽度（不包含滚动条）

注意：inner 不可省略。

例子：<script>

document.write("浏览器宽度为："+window.innerWidth+"，浏览器高度为： "+window.innerHeight)

</script>

4. window方法：

window.open() -打开新窗口

open()方法的语法：

window.open(url, “"**\_block/\_self**”, features, replace);

open函数参数说明：

url：要载入窗体的URL；

features:代表窗体特性的字符串，字符串中每个特性使用逗号分隔；

replace:一个布尔值，说明新载入的页面是否替换当前载入的页面，此参数通常不用指定。

例子：<button id="btn" onclick="demo()">按钮</button>

<script>

function demo(){

window.open("index.html","windowname","height=200,width=200px,top=500px,left=500px,toolbar=no,menubar=no") }

</script>

注意：引号之间表示一个完整的参数。

window.close()-关闭当前窗口

例子：<button id="btn" onclick="demo()">按钮</button>

<script>

function demo(){

window.close()

}

</script>

1. 计时器

计时事件：通过使用JavaScript，我们有能力作到一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行，我们称之为计时事件。

计时方法（函数）：

1. ①setInterval()-间隔指定的毫秒数不停地执行指定的代码

语法：setInterval(**function(){}**,1000);

写法1： <p id="ptime"></p>

<script>

setInterval(function(){

gettime()

},1000);

function gettime(){

var d=new Date();

var t=d.toLocaleTimeString(); //将时间转化为字符串输出(P19)

document.getElementById("ptime").innerHTML=t

}

</script>

注意：关键语法new Date().toLocaleTimeString(); //将时间转化为字符串输出。

数组的toLocaleTimeString()方法：把数组转换为本地数组，并返回结果。

计时方法写在时钟函数中的写法：

写法2： function gettime(){

var d=new Date();

var t=d.toLocaleTimeString(); //将时间转化为字符串输出

document.getElementById("ptime").innerHTML=t

setInterval(function(){

gettime()

},1000);

}

gettime()

注意：此时setInterval()计时函数写在时钟函数之中，无法通过clearInterval()停止时钟函数。（P21）

与P19案例类似写法：

写法3： <p id="ptime"></p>

<script>

function gettime(){

var d=new Date();

var t=d.toLocaleTimeString(); //将时间转化为字符串输出(P19)

document.getElementById("ptime").innerHTML=t

setTimeout(function(){

gettime()

},1000);

}

gettime()

</script>

对比：P19的案例为计时方法写在时钟函数中（用setTimeout()），这里的案例为计时方法与时钟函数并列（用setInterval()）。两个方式的共同点：计时方法的调用函数为时钟函数。

当用setInterval()时，计时函数也可以写在时钟函数之中，但是此时无法通过clearInterval()停止时钟函数。

另一种写法和效果（写法4）：

<script>

setInterval(function(){

document.write(new Date().toLocaleTimeString());

},1000)

</script>

②clearInterval()方法用于停止setInterval()方法执行的函数代码

例子：<button id="btn" onclick="stoptime()">按钮</button>

<p id="ptime"></p>

<script>

var mytime=setInterval(function(){

gettime()

},1000);

function gettime(){

var d=new Date();

var t=d.toLocaleTimeString(); //将时间转化为字符串输出(P19)

document.getElementById("ptime").innerHTML=t

}

function stoptime(){

clearInterval(mytime);

}

</script>

注意：clearInterval()括号中放置setInterval()的赋值变量。

1. setTimeout()-暂停指定的毫秒数后执行指定的代码

语法：①setTimeout(function(){},1000);

只执行一次：<script>

var win;

function mywin(){

setTimeout(function(){

alert("hello")

},3000);

}

mywin()

</script>

等价于：setTimeout(function(){

alert("hello")

},3000);

调用外层函数循环执行：<script>

var win;

function mywin(){

alert("hello");

setTimeout(function(){

mywin()

},3000);

}

mywin()

</script>

②clearTimeoutl()方法用于停止执行setTimeout()方法执行的函数代码

<button id="btn" onclick="stopwin()">按钮</button>

<script>

var win;

function mywin(){

alert("hello");

win=setTimeout(function(){

mywin()

},1000);

}

mywin()

function stopwin(){

clearTimeout(win);

}

注意：clearTimeoutl()括号中放置setTimeout()的赋值变量。

1. History对象

window.history对象包含浏览器的历史(url)的集合。

History方法：

history.back() -与在浏览器点击后退按钮相同。

history.forward()-与在浏览器中点击按钮向前相同。

history.go()-进入历史中的某个页面。

例子：

页面1 <body>

<button id="btn" onclick="gonext()">按钮</button>

<a href="new\_file.html">前进</a>

<script>

function gonext(){

history.forward()

}

</script>

</body>

页面2 <body>

<button id="btn" onclick="goback()">按钮</button>

<script>

function goback(){

history.back()

}

</script>

</body>

例子：

页面1 <body>

<a href="index.html">跳转</a>

</body>

页面2: <body>

<form>

<input type="text" id="usename" />

<button onclick="safe()">按钮</button>

</form>

<script>

function safe(){

if(document.getElementById("usename").value=="hello"){

history.go(-1);

}else{

alert("输入错误");

}

}

</script>

</body>

1. Location对象
2. Location对象：window.location对象用于获得当前页面的地址(URL)，并把浏览器重定向到新的页面。
3. Location对象的属性：

//页面地址组成：web协议+主机域名+端口+页面路径

location.hostname 返回web主机域名

location.pathname 返回当前页面的路径和文件名

location.port返回web主机的端口

location.protocol 返回所使用的web协议(http://或https://)

location.href 属性返回当前页面的URL

location.assign()方法加载新的文档

注：Protocol：协议；pathname：路径名；

例子：<button id="btn" onclick="getloc()">按钮</button>

<script>

function getloc(){

//页面地址组成：web协议+主机域名+端口+页面路径

//页面地址：http://127.0.0.1:8020/20160408/new\_file2.html

alert(window.location.hostname);//弹出127.0.0.1

alert(window.location.pathname);//弹出 /20160408/new\_file2.html

alert(window.location.port);//弹出8020

alert(window.location.protocol);//弹出http:

alert(window.location.href);

//弹出http://127.0.0.1:8020/20160408/new\_file2.html

window.location.assign("http//:www.baidu.com") //跳转到百度页面

}

</script>

1. Screen对象

window.screen对象包含有关屏幕的信息

属性：

screen.availWidth—可用的屏幕宽度

screen.availHeight —可用的屏幕高度

screen.height —屏幕高度

screen. width —屏幕宽度

注意：可用尺寸宽高单词的首字母大写，屏幕尺寸宽高单词的首字母小写。

例子：

<script>

document.write("可用的屏幕宽度:"+screen.availWidth+";可用的屏幕高度:"+screen.availHeight)

document.write("屏幕宽度:"+screen.width+";屏幕高度:"+screen.height)

</script>

1. JavaScript面向对象
2. 认识面向对象、基本面向对象、函数构造器构造对象、深入JavaScript面向对象。
3. 面向对象的概念：

（1）一切事物皆对象；（2）对象具有封装和继承特性；（3）信息隐藏；

1. 基本面向对象

<script>

var person={

name:"iwen",

age:30,

eat:function(){

alert("能吃")

}

}

alert(person.name)

</script>

**引入P17：自定义对象有两种实现方式：**

（1）定义并创建对象实例；

语法：变量=new Object();

变量.属性=“属性值”

例子：<script>

people=new Object();

people.name="iwen";

people.age="20";

document.write("姓名："+people.name+";年龄："+people.age)

</script>

语法：变量={属性：“属性值”，属性：“属性值”…}

例子： <script>

people={name:"iwen",age:"20"};

document.write("姓名："+people.name+";年龄："+people.age)

</script>

注意：①字符串连接用加号+。②对象属于JavaScript数据类型的一种（P2），声明对象属于声明变量中的一种。（var 变量=new Object()）。③声明变量时，属性值用双引号包裹。④两种定义并创建对象对比：等分号，冒逗号。⑤第二种对象创建对象方法中people后的等号不能省略。

1. 函数构造器构造对象

视频内容未理解（JS面向对象-认识面向对象）

例子：<script>

function people(name,age){

this.name=name;

this.age=age;

}

son=new people("iwen",20)

document.write("姓名："+son.name+";年龄："+son.age)

</script>

注意：①new不可省略。

②当函数的参数为将要调用该函数的元素本身时，该元素调用该函数时，参数赋值为this。（P10）当用函数来创建对象时，函数中通过this.a=”属性值”定义对象的属性值。即：this放在HTML元素函数属性值引用的参数中时，指代该元素；this放在<script>标签中函数表达式中时，指代实例对象。

例子： <div style="width: 100px;height: 100px;background-color: cadetblue;" id="div1" onmouseover="onover(this)" onmouseout="onout(this)">

</div>

<script>

function onover(a){

a.innerHTML="hello"

}

function onout(b){

b.innerHTML="word"

}

</script>

1. 深入JavaScript面向对象1（封装、继承特性、信息隐藏）

例子1：在JS文件中书写，并引入以下代码

function people(){

} //创建了一个函数，这个函数可以当做类来使用。

people.prototype.say=function(){

alert("hello")

} //指定类（people函数）的原型链的say方法为一个函数，此函数为打印一句话“hello”。

function student(){

} //创建另外一个类student。

student.prototype=new people() //指定student类的原型链继承自people原型链。

var s=new student() //创建一个student对象s。

s.say()  //调用对象s的say方法，弹出hello对话框。

例子2：

function people(){

} //创建父类people。

people.prototype.say=function(){

alert("hello")

} //指定父类people的原型链say方法。

function student(){

} //创建子类student。

student.prototype=new people() //指定子类student的原型链继承自父类people。

student.prototype.say=function(){

alert("word")

} //指定子类student的原型链say方法。

var s=new student() //创建一个子类student的对象s。

s.say() //调用对象s的say方法，弹出word对话框。

例子3：

function people(){

} //创建父类people。

people.prototype.say=function(){

alert("hello")

} //指定父类people的原型链say方法。

function student(){

} //创建子类student。

student.prototype=new people() //指定子类student的原型链继承自父类people。

var supersay=student.prototype.say

student.prototype.say=function(){

supersay.call(this);

alert("word")

} //指定子类student的原型链say方法。

var s=new student() //创建一个子类student的对象s。

s.say() //调用对象s的say方法，弹出hello对话框和 word对话框。

后面待学？？