-------------------------------------------------------------------------------------------

重要：<script type="text/JavaScript"> 中间写 </script>

注意每个语句后面加分号 " ; "

Ctrl + / = 快速单行注掉

Ctrl + Shift + / = 快速多行注掉

注意大小写，注意大小写，注意大小写

-----------------------------------------------------------------------------

# 第一课

1.console.log（“-----”） //管理器那有

2.document.write(“------”) //左上角打的字

3.alert(“-----”) //警告框

4.confirm(“-----”) //带确认和取消的对话框

-----------------------------------------------------------------------------

第二课 "不长用"

<body>

//一个按钮（左上角）

<button onclick="第1课任意内容"> 按钮中显示的文字 <button>

//一个超链接（左上角）

<a href="JavaScript:第1课任意内容"> 超链接中显示的文字 </a>

</body>

分支：

<script type="text/JavaScript” src=“引入的文件”></script>

//算是当做外置东西了 "常用"

-----------------------------------------------------------------------------

# 第三课

赋值：var

用法：var 任意 = 任意

（ 后赋前 ）

-----------------------------------------------------------------------------

# 第四课

String 字符串

Number 数值

Boolean 布尔值

Null 空值

Undefined 未定义

Object 对象

其中 String Number Boolean Null Undefined 属于基本数据类型

Object 属于引用数据类型

第五课 | 输出：

\n 换行 1 | 1 今天

例：alert( "今天\n天气真不错 " ) | 天气真不错

\" 表示 " | --------------------------------

\' 表示 ' | 2 今天 天气真不错

\t 表示制表符 2 | --------------------------------

\\ 表示 \ |

\\\\ 表示 \\ | ====================

注：反斜杠在键盘上是“Enter”上方 | 英文输入

"<br />" 在网页中换行

-----------------------------------------------------------------------------

# 第六课

使用一个运算符 typeof 来检查一个变量的类型

语法： typeof 变量

例:console.log( typeof a );

检查字符串时会返回string

检查数值时会返回number

附赠：Numbe.MAX\_VALUE 表示在JS中的最大值

结果：1.7976931348623157e+308

如果超过了会表示" Infinity " 表示无穷

倘若要表示在JS中的最小数 Number 前加负号

----------------------------------

附赠：Numbe.MIN\_VALUE 表示在JS中的正小数（零以上的最小值）

结果：5e-324

如果低过了会表示" Infinity " 表示最小的正小数

----------------------------------

NaN 也表示一个数字，表示 Not A Number

意： 不是一个数字

注：使用 tapeof 检查一个 NaN 也会返回 number

----------------------------------

倘若使用JS进行浮点元素，可能得到一个不精确的数值

例：var x = 0.3 + 0.4

console.log (x)

输出：0.30000000000000004

# 第七课

布尔值 Boolean 只有两个

用来做逻辑判断

1.true

-表示逻辑上的真

2.false

-表示逻辑上的假

使用 typeof 检查一个布尔值时，会返回 boolean

-----------------------------------------------------------------------------

# 第八课

Null 的类型值只有一个，就是 null

null 这个值专门用来表示一个值为空的对象

使用 typeof 检查一个空值时，会返回 object （上文的对象）

Undefined 的类型值只有一个，就是 Undefined

（typeof 使用的方法在第6课）

-----------------------------------------------------------------------------

# 第九课 **{强制类型转换}**

1.调用toString（） 方法使上面赋值的东西转换为 String

var a = 123;

调用 a 的 toString( ) 方法

调用 xxx 的 yyy( ) 方法，就是 xxx.yyy( )

----------------------------------

例：var a = 123;

console.log( typeof a );

输出：number

----------------------------------

例：var a = 123;

var a = a.toString( )

console.log( typeof a )

输出：string

但是注意：null 与 undefined 这 2 个值没有 toString（） 方法

如果调用会抱错

----------------------------------

2.调用 String（） 函数使上面赋值的东西转换为 String

1.概念：被转换的数据作为参数传递给函数

例：var a = 123; 应当为 number

此时 alert（ typeof a ）;

输出：number

----------------------------------

例 : var a = 123 应当为 number

a = String(a)

alert（ typeof a ）;

输出：string

可以无视 null 与 undefined

-----------------------------------------------------------------------------

使用 Number( ) 函数将上面赋值的东西转换为 Number

例：var a = 123;

a = number( a );

document.write( typeof a );

字符串 --》 数字

1.如果是纯数字的字符串，则直接转换为数字

2.如果字符串中有非数字的内容，则值接转换为 NaN

（ NaN 也表示一个数字，表示 Not A Number ）

（意： 不是一个数字）

（注：使用 tapeof 检查一个 NaN 也会返回 number ）

3.如果字符串是一个空串或者是一个全是空格的字符串，则转换为0

4. 布尔值（ Boolean ）

true = 1

false = 0

5. Null 转数字 为 0

6. undefined 转数字为 NaN , 注意上文的返回 Number

-----------------------------------------------------------------------------

以下方式专门用来对付字符串

1. parseInt( )

parse 解析 Int 整形

用法：把一个字符串转换为整数

注意：由左到右，所以说 123abc456 输出为 123 | 123.456 输出为 123

2. parseFloat( ) 把一个字符串转换为一个浮点数

parseFloat( ) 作用和 parseInt( )类似，不同的是它可以获得有效的小数

注意：由左到右， 所以说 123.456 输出为 123.456

123.456.789 输出为 123.456

注意：如果对非 String（字符串）使用 parseInt( ) 或 parseFloat( )

它会先将其转换为 String 然后再操作

----------------------------------

例：var a = "123zbc" var a = "123.456"

a = parseInt( a ) a = parseFloat( a )

alert( a ) alert( a )

-----------------------------------------------------------------------------

第十课 （其他进制的数字）

0x = 表示16进制的数字(里面有些数字可以用英文代替）

0 = 表示8进制的数字

0b = 表示2进制数字（不是所有浏览器都支持）

（这个计算机每次将任何进制转换成10进制）

例：var a = 0x426 （十六进制）

换：2 1 0

算：4 2 6

2 1 0

4\*16 + 2\*16 + 6\*16 = 1062（换算成10进制）

----------------------------------

var a = 0426 （八进制）

换：2 1 0

算：4 2 6

2 1 0

4\*8 + 2\*8 + 6\*8 = 278（换算成10进制）

----------------------------------

向"070"这种字符串，有些浏览器当8进制转，有些当10进制转。（撒比IE）

解救方法：

可以在parseInt( )中传递一个的2个参数

var a ="070";

a = parseInt(a,10);

document.write(a );

document.write(typeof a);

-----------------------------------------------------------------------------

、

第十一课 （将其他数值转换为Boolean(布尔值)）

分为true与false

使用Boolean（）函数

例：

var a = 123 var a = 123

a = 0 a = 123或465或-5641416

a = Boolean(a) a = Boolean(a)

alert(typeof a) alert(typeof a)

输出：false 输出：true

//数字 -------》布尔值

//非 0，NaN 为true

//字符串 -------》布尔值

//除了空串其他都是true

// null 和 undefined 都是 false

-----------------------------------------------------------------------------

第十二课 (二元运算符）

（运算符也叫操作符）

（typeof就是运算符，可以获得一个值得类型)

算数运算符： + ; - ; \* ; / ; % ;

任何值和NaN做运算，结果都是Nan

如果对俩字符串进行相加结果会是拼串 “ a ”+ “ a ” = aa

任何值和字符串做加法运算，都会先转换为字符串拼接的操作

隐式转换法：

例：

var a = 123

a = a + ""

document.write(typeof a)

结果“ string ”

var a = 1 + +"2" + 3

document.write(typeof a)

可以将任何数值变为Number

----------------------------------

document.write( "a"+a )

在想知道a数值的情况下用

% : 取余

例：var a = 14 % 3

document.write( a )

显示：2

-----------------------------------------------------------------------------

第十三课 (一元运算符）

+ 与 -

var a = -123;

a = -a

document.write(a)

（能负负得正）

----------------------------------

var a ="123"

a = +a

document.write(typeof a)

输出：number

-----------------------------------------------------------------------------

# 第十四课

a++ 与 ++a

有个优先级的区分

-----------------------------------------------------------------------------

第十五课 （逻辑运算符）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

！ 非

可以对一个值进行非运算，对一个布尔值进行取反操作（可多次使用）

可以对一个值先转换为布尔值，再取反

例： var a = true var a = false var a = false

a = !a a = !a a = ！!a

alert( a ) alert( a ) alert( a )

结果：false 结果：true 结果：false

======================================

&& 与

可以对符号两侧的值进行 与 运算并返回结果

只要有一个 false ，就返回 false

如果第一个值为 false ，则不会看第二个值

例：true && alert( "Hello" )

======================================

|| 或

可以对符号两侧的值进行 与 运算并返回结果

只要有一个 true ，就返会 true

如果第一个值为 true ，则不会看第二个值

例：true || alert("Hello")

则不会执行

======================================

|| 与 && (非布尔值的情况）

对一个值先转换为布尔值，再运算，再返回原值

如果两个值都为 true ,就返回后面的

“ ”为 false

-----------------------------------------------------------------------------

第十六课 （赋值运算符）

======================================

= 可以将右侧数值赋值给左边

======================================

+= 用法：a = a + 5 可以写为 a += 5

======================================

-= 用法一致

======================================

\*= 用法一致

======================================

/= 用法一致

======================================

%= 用法一致

======================================

-----------------------------------------------------------------------------

第十七课 （关系运算符）

======================================

> 大于号 var a = 10 > 3

alert(a)

结果：返回true

======================================

< 小于号 用法一致

======================================

>= 大于等于 用法一致

======================================

<= 小于等于 用法一致

======================================

《非数值的情况》

对一个非数值先转换为数值，再比较，再返回原值

任何值对NaN进行比较，返回结果都是false

如果符号两侧都是字符串时，不会将其进行数字比较，

而会分别比较字符串中的Unicode编码

双字符串只比第一位，倘若第一位一致则比下一位

例：console.log( "abc" < "b") //true

console.log( "bbc" < "b") //false

可以用它对英文进行排序

======================================

-----------------------------------------------------------------------------

第十八课 编码表问题

写法：document.write("\u2620")

输出：?

平常：16进制

网页使用Unicode编码（10进制）

写法：<h1>&#9760</h1>

输出：?

辅助：电脑计算器中“查看”

<h1 style="font-size: 像素px">中间编码</h1>

-----------------------------------------------------------------------------

第十八课 （相等运算符）

使用 == 来进行相等运算

如果值类型不同，会先将其转换成相同的类型，再进行转换

大部分情况会转成 Number

Null 和 undefined 相等

NaN 不和任何值相等

判断一个数是否为 NaN 的方法

document.write( isNaN（ ） )

例： var b = NaN

alert( isNaN( b ) )

======================================

使用 ！= 来进行不相等运算

如果值类型不同，会先将其转换成相同的类型，再进行转换

大部分情况会转成 Number

======================================

=== 全等，与 == 区别为无法将数值自动转换

======================================

！== 不全等，与 ！= 区别为无法将数值自动转换

======================================

-----------------------------------------------------------------------------

第十九课 （条件运算符） （三元运算符）

？：

用法： 条件表达式 ？语句1 ：语句2；

如果 条件表达式 为 true ,则执行语句1，并返回结果

如果 条件表达式 为 false ,则执行语句2，并返回结果

对一个非布尔值先转换为布尔值，再比较，再返回原值

回顾：第十一课

//数字 -------》布尔值

//非 0，NaN 为true

//字符串 -------》布尔值

//除了空串其他都是true

// null 和 undefined 都是 false

例子：

var a = 10 , b = 20

a > b ? alert( a大 ) : alert( b大 )

输出：b大

一件有趣的事

var a = 10 , b = 900

setInterval(function(){

a++

a>b?document.write("字符"):document.write("字符")

},1000)；

----------------------------------

由于会返回结果，所以可以

var a =10 , b = 20 , c = 30

var x = a > b? a : b

var y = x > c? x : c

console.log( y )

-----------------------------------------------------------------------------

第二十课 （运算符的优先级）

跟数学差不多，假设符号优先级一致，则从左往右

改变优先级方法：套个（ ）

例子：已知 && 比 || 等级高

则：var a = (1 || 2) && 3

alert( a )

就行了

（相权与皇权）

-----------------------------------------------------------------------------

第二十一课 （代码块）

{

语句

语句

语句

}

一个代码块，唯一作用就是分组，非隔离

-----------------------------------------------------------------------------

第二十二课 （if语句）

例子：

var a = 10

if(a>10 && a<20){

alert( 比10大，比20小)

}；

----------------------------------

if( ){ }else{ }

----------------------------------

if( ){ }else if( ){ }else if( ){ }else if( ){ }

----------------------------------

更高级的对话框

prompt(" ")

----------------------------------

练习：

var a = prompt("你有多帅（1--100）"),b = prompt("你的财富水平（1--100）"),c = prompt("你的才华水平（1--100）")

if(a>100 || b>100 || c>100){

alert("宁太高贵了")

}else if(a>=90 && b>=90 && c>=90){

alert("ok")

}else if(a>=50 || b>=50 || c>=50){

alert("宁也就勉勉强强")

}else{alert("no")}

----------------------------------

prompt( )返回值是字符串转数字方法为+prompt( )

-----------------------------------------------------------------------------

第二十三课 （switch语句）

语法：

switch(条件表达式){

case: 表达式

语句......

break;

case: 表达式

语句......

break;

case: 表达式

语句......

break;

default: --------------------------------------- 除此之外的数，全为此单词之下的语句执行

语句...

break

}

----------------------------------

var a = +prompt("给爷来个数字")

switch( a>60 ){ case true:alert("踩踩踩踩踩") }

-----------------------------------------------------------------------------

第二十四课 （while循环）

while循环 语法：

while（条件表达式）{

语句.........

}

来回判断

（第三十五讲）

break 退出循环

例子：var i = 0

while(i < 10){

document.while(i++ +"<br />")

}

while 先判断，后执行

do......while 先执行，后判断（至少执行一次）

例子：var i = 0

do{document.white(i++ + "<br />"

}while(i <= 10)

无 while do 形式

--------------------------------------使用循环往往需要三个条件

1.初始化一个变量

2.在循环中设置一个条件表达式

3.定义一个更新表达式，每次更新初始化变量

--------------整合与复习-------------

while(true){

var a = +prompt()

if(a<=10 || a>=0 && isNaN(a)){

break

}

document.write("666")

}

结合：

|| 遇true则停止,只要有一个 true ，就返会 true

&& 遇false则停止,只要有一个 false ，就返会 false

-----------------------------------------------------------------------------

第二十五课 （For循环）

用法：

for( 初始化表达式;条件表达式;更新表达式){

语句。。。。。。

}

例子：

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

document.write(a)

}

--------------整合与复习-------------

for(var a = 1;a < 100;a++){

if(a%2 !=0){

document.write(a + "<br/>")

}

列出100内的所有单数

var s = 0

for(var a = 1;a < 100;a++){

if(a%7 ==0){

s = s + a

}

}

document.write(s)

列出7内的所有单数并加在一起

-----------------------------------------------------------------------------

第二十六课 （For循环的嵌套使用）

&nbsp 在网页中的空格

--------------整合与练习-------------

for(var i=0;i<5;i++){

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

document.write("\*&nbsp")

}

document.write("<br/>")

--------------九九成法表（终极）包含css-------------

<script type="text/javascript">

for (var a = 1; a <= 9; a++) {

for (var e = 1; e <= a; e++) {

document.write("<span>" + e + "\*" + a + "=" + e \* a + "</span>")

}

document.write("<br/>")

}

</script>

body{

width:2000px

}

<style type="text/css">

span {

display: inline-block;

width: 80px;

}}

-----------------------------------------------------------------------------

第二十七课 （break和continue）

区别：

一. Break

1.退出循环语句包括（ switch 语句）

2.不能在 if 语句中使用 break 语句

3.会终止离其最近的循环语句

例子：在外层循环和内层循环中，由于break的就近原则，只会中止内层循环，外层则继续运行

例如两个For循环只会终止第二个

解决方法：

1.给循环起个名 语句：

Wdnmd :

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

document.write("666")

for(var e=1;e<20;e++){

break wdnmd

document.write("666")

}

document.write("<br/>")

}

注意：该方法差不多等于终止全部

---------------------------

二. Continue

1.continue 可以跳过当次循环

例子：

for(var a=0;a<10;a++){

if（a ==1){

continue

}

document.write(a)

}

输出：023456789

-----------------------------------------------------------------------------

第二十八课 （质数练习的改进）

1.Math.sqrt( ) -- 对一个数字进行开方

-----------------------------------------------------------------------------

第二十九课 （Object.对象）

对象属于一种复合的数据类型，在对象中可以保存多个不同类型的属性

对象的分类：

**1.内建对象**

- 由ES标准中定义的对象，在任何的ES的实现中都可以使用

- 比如，Math String Number Boolean Function Object

**2.宿主对象**

- 由JS的运行环境提供的对象，目前来讲主要指由浏览器提供的对象

- 比如 BOM（浏览器对象模型） DOM（文档对象模型）

**3.自定义对象**

- 由开发人员自己创建的对象

---------------------------

**创建对象：**

var 任意名 = new Object( ) 构造对象，注意大小写

使用**new方法构造的函数叫constructor**

构造函数是专门用来创建对象的函数

**1.向对象中添加属性**

语法：对象.属性名 = 属性值

例子：

var obj = new Object( );

obj.name="7"

obj.nam="6"

obj.na="5"

document.write(obj)

**2.单个读取**

语法：对象.属性名 =属性名**.**属性值

例子：

var obj = new Object( );

obj.name="7"

obj.nam="6"

obj.na="5"

document.write(obj.name)

**替换方法**

var obj = new Object();

1. obj.name="wdnmd"

2. obj.name = "我带你们打"

document.write(obj.name)

输出：

我带你们打

}》玩意差不多跟覆盖一样......

---------------------------

删除单个对象

语法：delete 对象 **.** 属性名

例子：

var obj = new Object( ) ;

obj.name="wdnmd"

delete obj.name;

document.write(obj.name)

**输出：undefined** （未定义）

-----------------------------------------------------------------------------

第三十课 （Object.对象）

**属性名：**

对象的属性名不强制要求标识符的规范

什么**乱七八糟的名字都行**

如果使用特殊的属性名，不能采用 **.** 的方式操作

语法：对象["属性名"]= 属性值

例子：

var obj = new Object();

obj["123"]="6"

document.write(obj["123"])

读取时也得采用这种方式

---------------------------

**传奇方法：**

var obj = new Object();

obj["123"]= 6

var n = "123"

document.write(obj[n])

个人理解：计算机从上往下读 ，首先在obj["123"]中存储了 6,

var n = "123" ，则 n 对应 obj["123"]中的 6，

所以 n = 6

document.write(obj[n]),也就说 n ="123" = obj["123"] = 6

所以 n = 6

假设中间干扰 n ，则会返回 undefined ，例如将n转为number后自增

---------------------------

**属性值：**

JS的属性值，可以是任意的数据类型

}》差不多像文件地址一样

例子：千层饼

**in 运算符**

1.检查一个对象中是否含有指定的属性

有则true , 没则false

例子：

document.write(text2 in obj)

输出：false

var obj = new Object();

obj["123"]=6

var n = "123"

document.write(obj[n])

alert(n in obj)

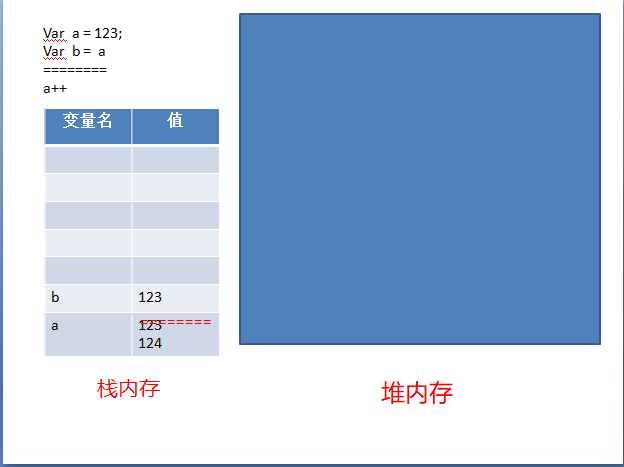
输出：true

-----------------------------------------------------------------------------

第三十一课 （基本数据类型和引用数据类型）

**本章有PPT以完成**

**1.基本数据类型**

String Number Boolean Null Undefined

**2.引用数据类型**

Object

---------------------------

**1.基本数据类型**

JS中的变量都是保存到栈内存中的

基本数据类型的值直接在栈内存中存储

值与值之间独立存在，修改一个值不会影响其他变量

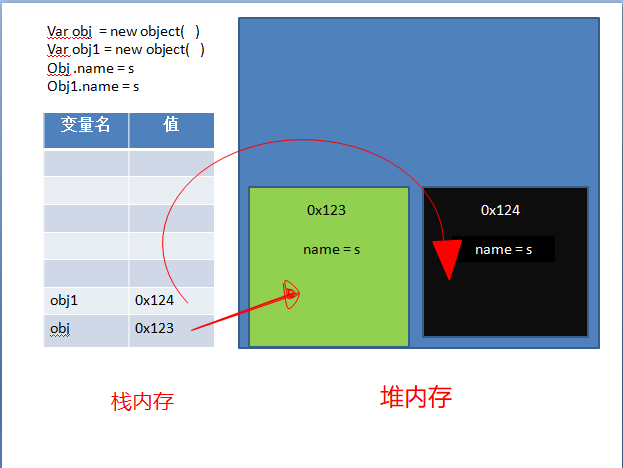
---------------------------

**2.引用数据类型**

对象是保存到堆内存中的，每次创建一个新对象，就会在堆内存中开辟一个新的空间

而变量保存的是对象的内存地址（对象的引用）

所以当两个变量用的是同一个地址，改一个的话，另一个也会受到影响

假设比较引用数据类型时，它比较的是内存地址

终于看到50集了

第三十二课 （对象字面量）

使用对象字面量创建一个对像

原本：var obj = new object( );

简化：var obj = { }

语法:{属性名：属性值，属性名：属性值}

方法: var obj = {

name：你们那，

asd：xdm

}

对象字面量的属性名可以加引号也可以不加

属性名和属性值是一组一组的**明值对结构**

名与值之间用**：**连接，多个名对之间使用**，**隔开

大括号就是创建对象 { }

var obj2 = {

name:"我带你们打"

age：13

gender: "男"

text:{ name:"问那么多" }

-----------------------------------------------------------------------------

第三十三课 （函数的简介）

函数

- 函数也是一个对象

- 函数可以封装一些功能（代码），在需要时可以执行这些功能（代码）

- 函数可以保存一些代码在需要时调用

- 使用typeof检查一个函数对象时，会返回Function

- 封装到函数中的代码不会立即执行

- 函数中的代码会在函数调用时执行

- 调用函数 语法：函数对象（ ）

- 当调用时，函数中的代码会**按照顺序**执行

- 在开发中很少使用构造函数来创建一个函数

- 使用函数声明来创建一个对象

语法：

function 函数名（[形参1，形参2......形参N]）{

语句...

}

运用方法：

function fun2( ){

document.write("wdnmd")

}

fun2( )

-----------------------------------------------------------------------------

第三十四课 （函数的参数）

1.定义一个用来求两个数和的**函数**

2.可以在函数的（ ）中来指定一个或多个**形参**（形式参数）

3.多个形参之间使用，隔开.声明形参就相当于在函数内部声明了对应变量

4.但是并不赋值

5.再调用函数时，可以在（ ）中指定**实参**（实际参数）

**6.实参将会赋值给函数中的形参**

例子：

Function nb( a , b ){

document.write( a + b )

}

nb( 1 , 2 )

**7.调用函数时解析器不会检查实参的类型，所以注意所接受的参数**

例子：nb(123,hello) 输出：123hello

**8.调用函数时，解析器也不会检查实参的数量**

**多余实参不会被赋值**

**9.如果实参少于形参，则返回undefined**

10.函数的实参可以是**任意的**数据类型

-----------------------------------------------------------------------------

第三十五课 （函数的返回值）

使用 return 来设置函数的返回值

语法:

return 值

**》return 后的值将会作为函数的执行结果返回**

》定义一个变量来接受结果

例子：

function nmd(n,m,d){

var s =n+m+d

return s

}

var nb = nmd(6,8,5)

》变量 nb 的值就是函数的执行结果

》函数返回什么 nb 的值就是什么

**》在函数中 return 后的语句都不会执行**

》如果 return 语句后不跟任何值就相当于返回一个undefined

》如果函数中不写 return ，则也会返回 undefined

**》return 可以返回任何类型的值**

例子：function n(a,b,c){

var p = a+b+c

return p

}

var s = n(4,5,1)

alert(s)

》个人理解：

1.首先 n 后（）中三变量，下面赋值4，5，1，

将三数相加的和赋值给 p ，因为return的语法就是 return 单词

所以 p 等于 10 ，而后返回 p 值给 n

在 var s = n(4,5,1),所以当输出 s 时会蹦出10[](#_第一课)

function sum(o){

return o % 2 == 0

}

var t = sum(4)

document.write(t)

设置变量 o ，return目的就是判断对错，下面传参4，4 = o = 4%2 = true

将 true 传给sum ,又 var 了一个变量 t 来接受 sum 的 true ，再调用t

-----------------------------------------------------------------------------

第三十六课 （实参可以是任何值）

课前题

function s(a){

return a%2 ==0

}

alert( s ( prompt( )))

**之所以返回布尔值，是因为 =0 具有了判断意义**

实参可以是任何数据类型，也可以是一个对象

当我们参数过多时，可以将参数装到一个对象中，然后通过对象传递

使用函数作为参数**（尚硅谷的第54集）**

传参，传参，传参.........

参数名（ ）

—— 调用函数

相当于使用函数的返回值

参数名

—— 函数对象

相当于直接使用函数对象

-----------------------------------------------------------------------------

第三十七课 （返回值的类型）

break:终止循环

continue：跳过当次循环

return：退出函数

返回值可以是任意数据类型

俄罗斯套娃......

-----------------------------------------------------------------------------

第三十八课 （立即执行函数）

函数对象（ ）

（function（ ）{

alert（“666”）

}）（ ）

拆分：

首先开头结尾俩括号代表他们是一个整体

尾部一个括号代表他是一个函数

调用简写：zbc( )

注意：这玩意往往只会执行一次

例子：

(function(a,b){

document.write(a)

document.write(b)

})(123,456)

-----------------------------------------------------------------------------

第三十九课 （方法）

var obj = new Object( )

obj.name = 123

obj.age = 18

obj.sayName = function(){

document.write(obj.name)

}

obj.sayName( )

只有函数才有（ ），obj.sayname是在对象中添加了一个东西

所以实际实际上调用的（ )实际上是调用了对象中的函数

函数也可以成为对象的属性

如果一个函数作为一个对象的属性保存

那么我们称这个函数是这个对象的方法

调用函数就说调用对象的方法

obj.fun( )调方法

fun( )调函数

-----------------------------------------------------------------------------

第四十课 （枚举对象中的属性）

使用 for ... in 语句

语法：for( var 变量 in 对象){

}

for（var n in obj){

document.write(obj[n])

}

-----------------------------------------------------------------------------

第四十一课 （全局作用域）

作用域

- 作用域指一个变量的作用的范围

- 在JS中一共有两中作用域

1.全局作用域

- 直接编写在Script标签中的JS代码,都在全局作用域

- 全局作用域在页面打开时创建，在页面关闭时销毁

- 在全局作用域中有一个全局对象window,我们可以直接使用

它代表是一个浏览器窗口，它由浏览器创建，我们可以直接使用

- 在全局作用域中

创建的变量都会作为window对象的属性保存

创建的函数都会作为window对象的方法保存

- 全局作用域中的变量都是全局变量

在页面的任意的部分都可以访问得到

2.函数作用域

-----------------------------------------------------------------------------

第四十二课 （变量的声明提前）

变量的声明提前

- 使用var关键字声明的变量，会在所有代码执行之前被声明（但是不会赋值）

但是如果声明变量不使用var关键字，则变量不会被声明提前

函数的声明提前

- 使用函数声明形式创建函数 function 函数( ){ }

他会在所有的代码执行之前就被创建，所以我们函数声明前来掉用函数

- 使用函数表达式创建的函数，不会被声明提前，所以不能在声明前调用

函数声明：function fun ( ）{ }

函数表达式：var fun2 = function( ){ }

-----------------------------------------------------------------------------

# **第四十三课 （this.）**

以函数形式调用时，this永远是Windo

以方法的形式调用时，this 就是调用方法的那个对象

function fun(){

document.write(this.name)

}

var obj = {

name :"傻逼",

sayName:fun

}

var obj2 = {

name :"混球",

sayName:fun

}

obj2.sayName()

个人理解：调用obj2的sayName首先找到fun，看见obj2调用this，

这也就是为什么浏览器返回 [object Object] ，当更明确的只向this.name时

返回 “混球”

*越来越他妈难了*

-----------------------------------------------------------------------------

第四十四课 （使用工厂方法创建对象）

使用工厂方法创建的对象，使用的构造函数都是Object

所以创建的对象都是Object这个类型

就导致我们无法区分出多种不同类型的对象

所以使用不多

function person (name,age){

obj = new Object()

obj.name = name

obj.age = age

obj.sayName = function(){

this.name

}

return obj

}

var love = person("monika",18)

var love1 = person("sayori",18)

console.log(love)

console.log(love1)

疑问：这个sayName就起个连线效果吗？

-----------------------------------------------------------------------------

第四十五课 （构造函数）

创建一个构造函数，专门用来创建Person对象的

构造函数就是一个普通的函数，创建方式和普通函数没有区别

首字母大写

普通函数与调用函数的区别是调用方式不同

构造函数执行流程

1.立即创建一个新的对象

2.将新建的对象设置为函数中的This，在构造函数中可以使用this来引用新建对像

3.逐行运行函数中的代码

4.将新建的对像作为返回值返回

5.使用同一个构造函数创建的对象，我们称为一类对象

6.我们将通过一个构造函数创建的对象，称为是该类的实例

7.使用 instanceof 检查类与实例

8.this的情况

当以函数的形式调用时， this 是 window

当以方法的形式调用时，谁调用就是谁

当以构造函数的形式调用时， this 就是新创建的那个对象

例子：

function Person( ) {

this.name = "孙悟空"

this.age = 18

this.gander = "男"

this.sayName = fun

}

function fun( ) {

alert(this.name)

}

var per = new Person( )

console.log(per)

per.sayName( )

个人理解：如二所说，在碰见new时会立即创建一个新的对象，也就是this.

而per = new = this. 所以this.name这些会自动赋值给per

function Wdnmd (name,age){

this.name = name

this.age = age

}

var wdnmd = new Wdnmd("wdnmd",99)

console.log(wdnmd)

-----------------------------------------------------------------------------

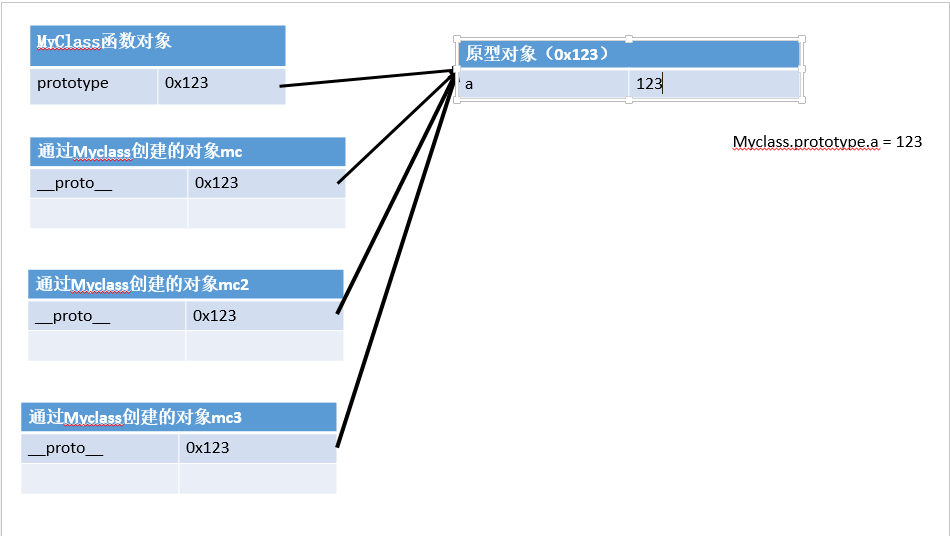
# 第四十六课 **（原型对象）**

1.将函数定义在全局作用域，污染了全局作用域的命名空间

2.原型 Prototype

3.我们所创建的每一个函数，解析器都会向解析器添加一个属性 Prototype

4.这个属性对应着一个对象，这个对象就是我们所谓的原型对象

5.如果函数作为普通函数调用Prototype没有任何作用

6.如果函数作为构造函数调用时，它所创建的对象中都会有一个隐含的属性指向该构造函数的原型对象，我们可以

通过 \_\_proto\_\_ 来访问该属性

7.原型对象就相当于一个公共的区域，所有同一个类的实例都可以访问到这个原型对象

所以我们可以将对象中共有的内容，统一设置到原型对象中

8.当我们访问对象的一个属性或方法时，他会先自检，如果有则使用

没则会在原型对象中找，找到使用

例子（加强）

function Myclass( ){ }

var Mc = new Myclass( )

Myclass.prototype.Sayname = function(name,gender){

alert(name)

alert(gender)

}

Mc.Sayname("莫妮卡","18")

《全是在原型中添加的》

进阶：

function Person( ) {

this.name = "孙悟空"

this.age = 18

this.gander = "男"

}

Person.prototype.sayName = function() {

alert(this.name)

}

var per = new Person( )

per.sayName( )

-----------------------------------------------------------------------------

第四十七课 （原型对象）

1.原型对象也是对象，所以它也有原型，所以当去调用一个对象时，会逐步寻找（最多两层）

2.原型对象和隐式原型对象（只有两层）

3.如果没有则去原型的原型中去找，直到找到Object对象的原型

function Myclass( ){ }

var mc = new Myclass( )

Myclass.prototype.name = "我是原型中的名字"

alert(mc.name)

所以还是先自检，后大检

alert("name" in mc)

返回“true”

证明in检查对象中是否含有某个属性时，如果原型中有，也会返回true

可以使用hasOwnProperty( )来检查对象自身是否含有该属性，含有则返回True

-----------------------------------------------------------------------------

第四十八课 （ \_toString( ) ）

1.当我们直接在页面上打印一个对象时，事实上是输出的对象的toString( )方法的返回值

2.所以我们选择覆盖，也叫截断？

3.草，忘写笔记了

代码：

function Person(name,age,gender){

this.name = name

this.gender = gender

this.age = age

}

var per = new Person("莫妮卡","18","女")

Person.prototype.toString = function( ){

return "Person[name = "+this.name+"age = "+this.age+"gender = "+this.gender

}

alert(per)

-----------------------------------------------------------------------------

第四十九课 （垃圾回收）

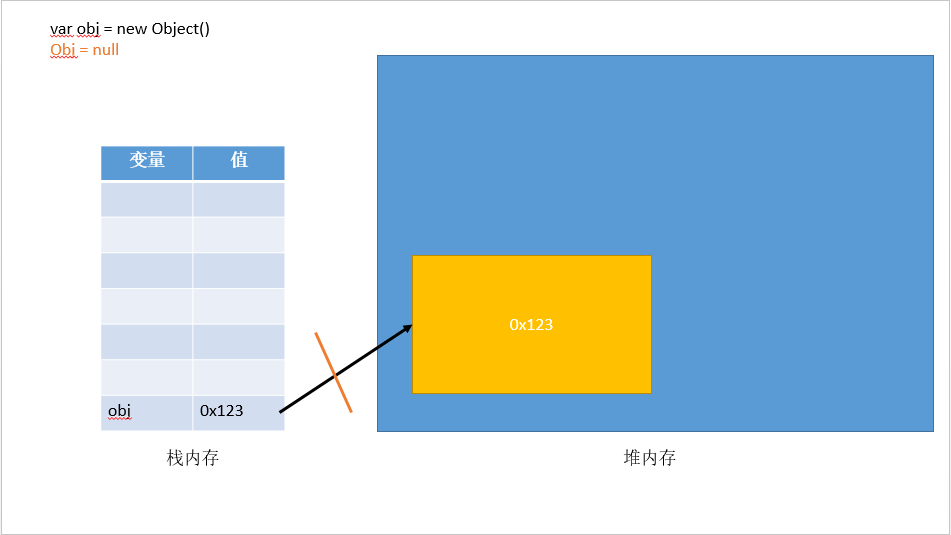
1.程序运行过程中也会产生垃圾，积攒过多后，会导致程序运行速度过慢

2.所以要回收

3.当一个对象没有任何的变量或属性对他进行引用，此时我们将永远无法操作该对象，此时对象就是垃圾

4.在JS中拥有自动的垃圾回收机制，会自动将这些垃圾对象从内存中销毁，我们不需要也不能进行垃圾回收的操作

5.我们需要做的只是要将不在使用的对象设置为Null即可

-----------------------------------------------------------------------------

第五十课 （数组简介）**，**

**记录：2021年6月5日 星期六 21：08**

先前讲的对象分为三个

内建对象，宿主对象，自定义对象

之前一直讲的是自定义对象

1.数组（Array）

- 数组也是一个对象

- 它和普通对象功能相似，也是用来存储一些值的

- 不同的是普通对象是使用字符串作为属性名的

而数组是使用数字来作为索引操作元素

- 索引

从0开始的整数就是索引

- 数组的存储性能比普通对象好，所以经常使用数组来存储一些数据

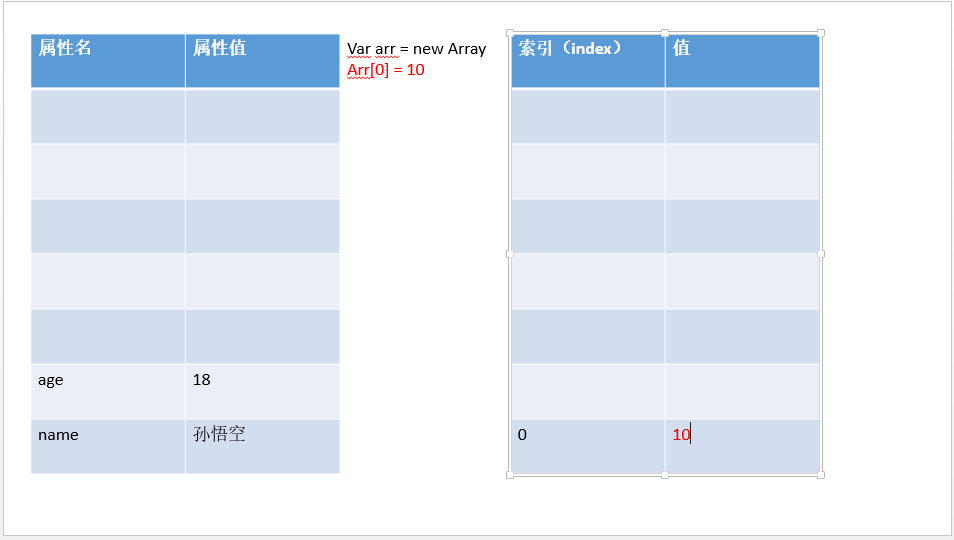
2.语法

语法：数组[ 索引 ] = 值

3.获取数组长度（元素个数）

语法：数组.length

例子：

var arr = new Array

arr[0] = 10

arr[1] = 11

arr[2] = 15

arr[3] = 19

arr[4] = 16

alert(arr.length)

返回：5

4.lenth可以被修改，改大则多，，，，改小则少

例子：arr.length = 10

输出: 10,11,15,19,16,,,,,

5.像数组最后添加元素

语法：数组[数组.length] = 值

arr[1] = 9

arr[arr.length] = 10

输出：9，10

-----------------------------------------------------------------------------

# **第五十一课** **（数组字面量）**

1.数组里面可以放任意的数据类型

2.对象能干的事数组也能干

3.语法 ： var arr = [ ]

4.var obj = {name : "孙悟空"}

arr[arr.length] = obj

arr = [{name : “孙悟空”}]

console.log(arr[0].name)

5.放函数

arr[function( ){alert(1)},function( ){alert(2)}]

调用：arr[0]( )

-----------------------------------------------------------------------------

第五十二课 （数组的常用方法）

1.push( )

- 该方法可以向数组的末尾一个或多个数组，并返回长度

例子：

var arr = [孙悟空]

arr.push(“唐山”，“666”)

alert（arr）

输出：孙悟空，唐山，666

所以： arr.push(?) = arr[arr.length] = ?

---------------------------

2.pop( )

- 该方法删除数组最后一个元素，并将删除的元素返回

---------------------------

3.unshift( )

- 向数组开头添加一个或多个元素，并返回新的数组长度

- 像前面插入元素，后面的索引会依次调整

---------------------------

4.shift

- 该方法删除数组前面的一个元素，并将删除的元素返回

- 像前面删除元素，后面的索引会依次调整

---------------------------

5.forEach

1.数组的遍历

常用方法：

var arr = ["per","per1","per2","per3","per4","per5"]

arr.forEach(function(a){

alert(a)

})

.forEach( )==for( ){ }

---------------------------

6.forEach

- 一般我们都是使用for循环去遍历数组，JS中为我们提供了一个方法来遍历数组

- forEach( )

- 数组中有几个元素函数就会执行几次，每次执行时，浏览器会将遍历到的元素

以实参的形式传递进来，我们可以定义形参来读取这些内容

- 浏览器会在回调函数中传递三个参数

- 第一个参数，就是当前正在遍历的元素

- 第二个参数，就是当前正在遍历的元素的索引

- 第三个参数，就是正在遍历的数组

---------------------------

7.slice( )

- 可以用来从数组中提取指定元素

- 该方法不会改变原数组，而是将截取到的元素封装到一个新的数组中返回

- 参数：

1.截取开始的位置索引，包括开始索引

2.截取结束的位置索引，不包括结束索引

- 第二个参数可以省略不写，此时会截取从开始索引往后的所有元素

- 索引可以传递一个负值

-1 倒数第一个

-2 倒数第二个

---------------------------

8.splice( )

- 可以用于删除数组中的指定元素

- 使用splice（ ）会影响到原数组，会将指定元素从原数组中删除

并将被删除的元素作为返回值返回

- 参数

第一个，表示开始位置的索引

第二个，表示删除的位置

第三个及以后......

可以传递一些新的元素，这些元素将会自动插入到开始位置索引前边

例子：

var arr = ["per","per1","per2","per3","per4","per5"]

var Monika = arr.splice(0,3,"Just Monika")

alert(arr)

输出：Just Monika,per3,per4,per5

---------------------------

例子：

var arr = ["per","per1","per2","per3","per4","per5"]

var Monika = arr.splice(0,3,"Just Monika")

alert(Monika)

输出：per,per1,per2

---------------------------

9.concat( )可以连接两个或多个数组，并将新的数组返回

-该方法不会对原数组产生影响

例子：

var arr = [1,2,3]

var arr2 = [4,5,6]

var arr3 = [7,8,9]

var result = arr.concat(arr2,arr3,"牛魔王")

alert(result)

输出：1,2,3,4,5,6,7,8,9,牛魔王

---------------------------

10.join( )

-该方法可以将数组转换为一个字符串

-该方法不会对原数组产生影响

-在join( )中可以指定一个字符串作为参数，这个字符串将会成为数组中元素的连接符

例子：

arr = ["孙悟空","沙和尚","猪八戒"]

var result = arr.join("@-@")

alert(result)

输出：孙悟空@-@沙和尚@-@猪八戒

---------------------------

11.reverse( )

- 该方法用来反转数组（颠倒世界）

- 该方法会对原数组产生影响

例子:

arr = ["孙悟空", "沙和尚", "猪八戒"]

arr. reverse()

alert(arr)

输出:

猪八戒，沙和尚，孙悟空

---------------------------

12.sort( )

- 可以对数组中的数字进行排序,默认会按照Unicode

- 该方法会对原数组产生影响

- 对数字进行排序时，可能会得到错误的结果

- 我们可以在sort( )添加一个回调函数，来指定排序规则

- 浏览器会分别使用数组中的元素作为实参去调用回调函数

- 使用哪个元素不确定，但a一定在b前边

- 浏览器会根据回调函数的返回值来决定元素排序

- 规则：

- 大于0，则元素交换位置

- 小于0，则元素位置不变

- 等于0，则认为两个元素相等，不交换位置

- 升序则 a-b

- 降序则 b-a

例子:

arr = ["b", "c", "a"]

arr. sort( )

alert(arr)

输出:

a,b,c

---------------------------

arr = [11,8,55,66,7,4,3,2,5]

arr.sort(function(a,b){

return a-b

})

alert(arr)

输出:

2,3,4,5,7,8,11,55,66

-----------------------------------------------------------------------------

第五十三课 （数组的遍历）

代码：

任务：将perArr中满18的提取

然后封装到新的数组中返回

function Person(name, age) {

this.name = name

this.age = age

}

Person.prototype.toString = function() {

return "Person[name = " + this.name + "age = " + this.age + "}"

}

var per = new Person("莫妮卡", "18", "女")

var per2= new Person("孙悟空","25","男")

var per3= new Person("沙和尚", "58", "男")

var per4= new Person("小孩子1","16","女")

var per5= new Person("电灯泡", "98", "男")

var per6= new Person("鸟飒","23","男")

var per4= new Person("小孩子2","16","女")

var per4= new Person("小孩子3","16","女")

var per4= new Person("小孩子4","16","女")

var per4= new Person("小孩子5","16","女")

var per4= new Person("小孩子6","16","女")

var perArr = [per,per2,per3,per4,per5,per6]

var monika = []

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

if(perArr[i].age>=18){

var a = perArr[i]

monika.push(a)

}

}

for(v=0;v<monika.length;v++){

alert(monika[v])

}

自己写的，圆满成功

记录：2021年6月8日22时58分

---------------------------

老师思路

function getAdult(arr){

//创建一个新的数组

var newArr = [ ]

//遍历Arr,获取arr中Person对象

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

var p = arr[ i ]

//判断Person对象age是否大于18

if( p.age>=18 ){

//如果大于18，则将这个对象添加到newArr中

//将对象放入到新数组中

newArr.push( p )

}

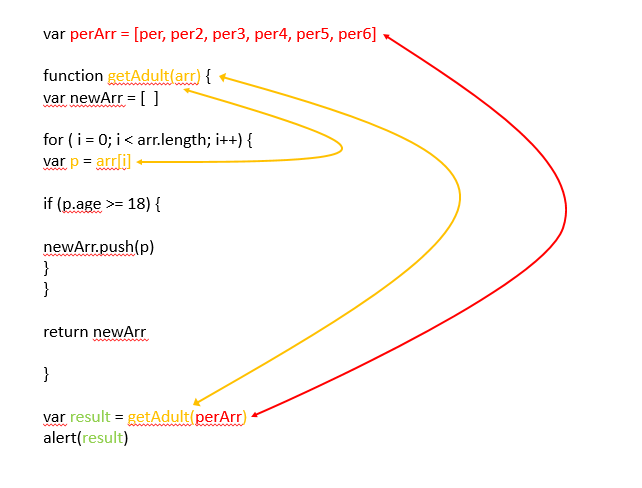
}

//将新的数组返回

return newArr

}

var result = getAdult( perArr )

alert( result ) 

......从未设想的道路，引入进去...，比我的好了不知多少倍啊......

-----------------------------------------------------------------------------

第五十四课 （数组去重练习）

创建一个数组

var arr = [1,2,3,2,2,2,2,1,3,4,4,2,5]

去重，获取元素

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

获取当前元素后的元素

for(var j=i+1;j<arr.length;j++){

判断两个值是否相等

if(arr[i]==arr[j]){

相等则消除J对应元素

arr.splice(j,1)

消除元素后，后面元素自动补位

需要再一次比较J

j--

}

}

}

alert(arr)

-----------------------------------------------------------------------------

第五十五课（函数的方法）

1.call( )

2.apply( )

- 这两个方法都是函数对象的方法，需要通过函数对象来调用

- 两者与fun（）用法一致

3.两者的作用可以改变指针方向

- call( )方法可以将实参在对象之后传递

- apply( )方法需要将实参封装到一个数组中统一传递

例子：

function fun(){

alert(this.name)

}

var obj = {name:"孙悟空"}

var obj1 = {name:"白骨精"}

var obj2 = {name:"沙和尚"}

fun.apply(obj)

输出：孙悟空

例子：

function fun(a,b){

console.log("a = " +a)

console.log("b = " +b)

}

var obj = {

name:"obj",

sayName:function(){

console.log(this.name)

}

}

fun.call(obj.sayName( ),6,7)

输出：

obj

a = 6

b = 7

---------第79课

-----------------------------------------------------------------------------

第五十六课（argumets）

1.在调用函数时，浏览器每次都会传递进两个隐含的参数

- 函数的上下文对象 this

- 封装实参的对象 arguments

- arguments是一个类数组对象，它也可以通过索引来操作数据，也可以获取长度

- 在调用函数时，我们所传递的实参都会封装到arguments中

- 它里面有一个属性叫做callee

这个属性对应一个函数对象，就是当前正在指向的函数的对象

-----------------------------------------------------------------------------

第五十七课（Date 对象）

1.在JS中用Date来表示一个时间

例子：

time ：var d = new Date()

document.write(d)

输出：Fri Jul 02 2021 12:08:46 GMT+0800 (中国标准时间)

2.如果直接使用构造函数创建一个Date对象，则会封装为当前代码执行的时间

例子：所以我们可以自定义时间  **按照月/日/年 时：分：秒排序**

var d = new Date("9/22/2017 11:10:50")

document.write(d)

Fri Sep 22 2017 11:10:50 GMT+0800 (中国标准时间)

---------------------------

3.getDate（）（抽出几日/月）

var d = new Date("9/22/2017 11:10:50")

var date = d.getDate()

document.write(date)

返回：22

---------------------------

4.getDay（）（抽出几日/星期）

var d = new Date("9/22/2017 11:10:50")

var date = d.getDay( )

document.write(date)

返回：5

0表示周日

---------------------------

5.getMonth（）（抽出几日/月）

var d = new Date("9/22/2017 11:10:50")

var date = d.getMonth()

document.write(date)

输出：8

0表示一月

---------------------------

6.getFullyear（）（抽出年）

var d = new Date("9/22/2017 11:10:50")

var date = d.getFullyear( )

document.write(date)

输出：2017

---------------------------

7.getTime（）（抽出当前时间戳）

- 时间戳，指的是从格林威治标准时间的1970年1月1日，0时0分0秒

到当前日期花费的毫秒数（1秒 = 1000毫秒）

- 计算机底层在保存时间时使用的都是时间戳

var d = new Date("9/22/2017 11:10:50")

var date = d.getTime()

document.write(date)

输出：1506049850000 毫秒

-----------------------------------------------------------------------------

第五十八课（Math）

1.Math.PI = 圆周率

例子:

alert(Math.PI)

输出：3.141592653589793

---------------------------

2.Math.abs(内容) = 数字的绝对值

例子：

alert(Math.abs(-1))

输出：1

---------------------------

3.Math.sail( )

- 向上取整

例子：

alert(Math.sail(5.1))

输出：6

---------------------------

4.Math.floor( )

- 向下取整

例子：

alert(Math.sail(5.9))

输出：5

---------------------------

5.Math.round( )

- 四舍五入

例子：

alert(Math.round(5.5)

输出：6

---------------------------

6.Math.random( )

- 可以用来生成 0 - 1 之间的随机数

- 生成0 - 100 的随机数

Math.random()\*100

生成 x - y 之间的随机数

公式:Math.round(Math.random( )\*(y - x)+x

---------------------------

7.Math.Max( )

- 最大值

例子:

alert（Math.Max(10,60,90)）

输出：90

---------------------------

8.Math.min( )

- 最小值

例子：

alert(Math.min(90,20,50))

输出：20

---------------------------

8.Math.pow( )

- 返会x的y次幂

例子：

alert(Math.pow(2,2))

输出:4

---------------------------

9.Math.sqrt( )

- 对一个数字进行开方

例子：

alert（Math.sqrt(9))

输出：3

# 第五十九课（包装类）

基本数据类型

String Number Boolean Null Undefined

引用数据类型

Object

在JS中为我们提供了三个包装类，通过这三个包装类

可以将基本数据类型转换为对象

1.String( )

- 可以将基本数据类型字符串转换为String对象

2.Number( )

- 可以将基本数据类型字符串转换为Number对象

3.Boolean( )

- 可以将基本数据类型字符串转换为Boolean对象

用法：

var num = new Number( )

var str = new String( )

var Bool = new Boolean( )

然鹅这种方法不用，因为对象比较的是地址

**但是方法和属性能添加给对象，不能添加给基本数据类型**

当我们对一些基本数据类型去调用属性和方法是浏览器会临时使用包装类将其转化为对象，然后调用对象的属性和方法,调用完之后，再将其转换为基本数据类型

例子：

**var s =123**

**(基本数据类型)**

**s = s.toString**

**(在这里浏览器会临时将s转换为Number对象,在调用Number对象的to.string，调用完之后，再将其转换为基本数据类型)**

**s.hello = "你好”（）**

**(在这里浏览器会临时将s转换为对象，将“你好”塞进去，然后销毁这也就是为什么不报错）**

**console.log(s.hello)**

**(在这里浏览器会临时将s转换为对象,但是一个新对象，所以找不到）**

**输出： Undefined**

-----------------------------------------------------------------------------

第六十课（字符串的方法）

1.在底层字符串是以字符数组的形式保存的

例子:

var arr = "Monika"

document.write(arr.length)

输出：6

---------------------------

2.charAt( )

- 不会影响原字符串

- 可以返回字符串中指定位置的字符

- 根据索引获取指定的字符

和一效果一样，了解下

---------------------------

3.charCodeAt( )

- 不会影响原字符串

- 可以返回字符的 Unicode 编码

---------------------------

4.String.fromCharCode( )

- 不会影响原字符串

- 可以根据字符的Unicode编码去获取字符

- 注意调用方式不太一样

- Unicode编码都是16进制（0x）

例子:

result = String.fromCharCode(70)

alert(result)

输出：F

---------------------------

5.concat( )

- 不会影响原字符串

- 可以连接两个或多个字符串

- 作用和加号一样

了解一下~~~

---------------------------

6.indexOf( )

- 不会影响原字符串

- 该方法检查一个字符串中是否含有指定内容

- 如果字符串中含有该内容，则会返回其第一次出现的索引

- 如果没找到，则返回 -1

- 可以指定一个第二个参数，指定开始查找的位置

例子：

var str = "hello world"

var result = str.indexOf("l",1)

alert( result )

输出: 2

---------------------------

7.lastIndexOf( )

- 不会影响原字符串

- 用法和上文一致，不同的是它是从后往前的

---------------------------

8.slice( )

- 不会影响原字符串

- 可以从字符串/数组中截取指定内容

- 前开后闭

- 也可以传递一个负数

---------------------------

9.substring( )

- 不会影响原字符串

- 和slice类似

- 和slice最大的区别就是没有区别（狗头）

- 无法传递负数

- 它会自动调整参数的位置，如果第二个参数小于第一个，则自动交换

---------------------------

10.substr( )

- 不会影响原字符串

- 可以从字符串截取指定内容

- 开始，长度

---------------------------

11.**split( )**

- 不会影响原字符串

- 可以将一个字符串拆分为一个数组

- 方法中可以传递一个正则表达式作为参数

用法：

var monika = "123,84,9,58,48,41,5,2,875,48,4614,984,548"

var result = monika.split(",")

alert(result[1])

输出：84

---------------------------

12.toUpperCase( )

- 不会影响原字符串

- 将字符串都变成大写

例子：

var s = "sayori"

var result = s.toUpperCase( )

alert(result)

输出：SAYORI

---------------------------

13.toLowerCase( )

- 不会影响原字符串

- 将字符串都便成小写

例子：

var s = "JUST MONIKA"

var result = s.toLowerCase( )

alert(result)

输出：just monika

-----------------------------------------------------------------------------

第六十一课（正则表达式的简介）

1.邮件的规则 （论如何告诉计算机规则）

- 前边可以是 x x x x 乱七八糟

- 跟着一个@

- 后面可以是 x x x x 乱七八糟

- .com 或其他的乱七八糟

2.正则表达式用于定义一些字符串的规则

- 计算机可以根据正则表达式，来检查一个字符串是否符合规则

- 获取字符串符合规则的内容提取出来

- 在构造函数中可以传递一个匹配模式作为第二个参数

可以是：

i 忽略大小写

g 全局匹配模式

3.语法：

var 变量 = new RegExp("正则表达式","匹配模式")

- typeof 检查 reg 会返回 object

4.正则表达式的方法

test( )

- 使用这个方法可以用来检查一个字符串是否符合正则表达式的规则

- 如果符合则 true ，不符合则false

例子：

var reg = new RegExp("a")

alert(reg.test("abcde"))

返回：true

var reg = new RegExp("a")

alert(reg.test("bcde"))

返回：false

---------------------------

var reg = new RegExp("a")

var result = "a"

alert(reg.test( result ))

返回：true

---------------------------

- 使用这个方法可以用来检查一个字符串是否符合 a 的

- 注意严格区分大小写

---------------------------

var reg = new RegExp("a","i")

var result = "a"

alert(reg.test("Abcde"))

- 在构造函数中可以传递一个匹配模式作为第二个参数

可以是：

i 忽略大小写

g 全局匹配模式

用法：

Var sayaaa = "gahsvsygAaasdsbgfxgddfhyswaaeatrdgtrwA"

var reg = new RegExp( "aaa","ig" )

alert( reg.tes ( sayaaa ) )

-----------------------------------------------------------------------------

第六十二课（正则语法）

1.可以使用字面量来创建正则表达式

- 字面量简单，构造函数创建更灵活

语法：var 变量 = /正则表达式/匹配模式

var reg = /ab/i

2.使用 | 来表示或者的意思

var reg = /a|b/i

3.使用 [ ] 也是或的关系，但有个优点

var reg = /[a-z]/

var reg = /[A-z]/ 任意字母

4.使用[ ^ ] 表示除了

var reg = /[^ab]/i

-----------------------------------------------------------------------------

第六十三课（字符串和正则相关的方法）

1.根据字母来拆分数组

var str = "1e2b3c4d5e6f"

var reg = /[a-z]/i

alert(str.split(reg) )

---------------------------

2.search( )

- 可以搜索字符串中是否含有指定内容

- 如果搜索到指定内容，则会返回第一次出现的索引，如果没有搜索到返回 -1

- 它可以接受一个正则表达式作为参数，然后根据正则表达式去检索字符串

- 无法全局

例子：

var str = "hello abc hello aec afc"

var result = str.search(/a[bef]c/)

alert(result)

输出：6

---------------------------

2.match( )

- 可以根据正则表达式，从一个字符串中将符合条件的内容提取出来

- 默认情况下match只会找到一个符合要求的内容，在正则表达式后面填g

- match会将匹配到的内容封装到一个数组中返回，即使只查询到一个结果

例子：

var str = "hello Abc hello Aec afc"

var result = str.match(/a[bef]c/gi)

alert(result)

输出：abc,aec,afc

---------------------------

3.replace( )

- 不会影响到原字符串

- 可以将字符串中指定内容替换为新的内容

- 默认只会替换第一个

- 参数：

1.被替换的内容

2.新的内容

result = str.replace("1","2")

例子：

var str = "11s5saaatrh8trgrf5ewe"

var result = str.replace(/[a]/ig,"666")

alert(result)

输出;11s5s666666666trh8trgrf5ewe

-----------------------------------------------------------------------------

第六十四课（正则表达式语法）

1.量词

- 通过量词可以设置一个内容出现的次数（一例）

- 量词是个就近原则（二例）

- { n } 正好出现 N 次（二例）

- { m,n } 出现 m - n 次（三例）

- { m, } 出现 m 次以上（三例）

- + 至少一个，相当于{ 1, }

- \* 零个或多个，相当于{ 0, }

- ？零个或一个，相当于{ 0,1 }

- ^ 表示开头/^a/i , 第一个必须是 a

- $ 表示末尾/a$/i , 最后一个必须是 a

- 如果在正则表达式中同时使用 ^和$ 则要求字符串完全符合正则表达式

例子：

var reg = /a{6}/i

alert(reg.test("aaaaaa"))

例子：

var reg = /(ab){3}/i

alert(reg.test("ababab"))

例子：

var reg = /ab{1,3}c/

alert(reg.test("abbbc"))

手机号的规范：

var reg = /^1[3-9][0-9]{9}$/

var number = +prompt()

var t = reg.test(number)

if( t == true){

alert("注册成功")

}

-----------------------------------------------------------------------------

第六十五课（正则表达式语法）

1.“ . " 表示任意字符

2.在正则表达式中使用\来表示转义字符

3.字面量使用方法:/\w/

4.

\w

- 任意字母，数字，\_ 为true

\W

- 除了任意字母，数字，\_ 为true

\d

- 任意数字 为true

\D

- 除了任意数字 为true

\s

- 空格 为true

\S

- 除了空格 为true

\b

- 单词边界 为true

例子：/\b child \b/

- 划定独立单词，单词前后无任何多余

\B

- 除了单词边界 为true

例子：

var name = " wdn md "

alert(name.replace(/^\s\*|\s\*$/g,"" ))

输出：

wdn md

-----------------------------------------------------------------------------

第六十五课（邮件的正则）

//创造规则，秩序永恒

var reg = /^\w{3,}(\.\w+)\*@[A-z0-9]+(\.[A-z]{2,5}){1,2}$/

var email = "wangxubo780@gmail.com"

alert(reg.test(email))

-----------------------------------------------------------------------------

第六十四课（正则表达式语法）

1.量词

**- 通过量词可以设置一个内容出现的次数（一例）**

**- 量词是个就近原则（二例）**

**- { n } 正好出现 N 次（二例）**

**- { m,n } 出现 m - n 次（三例）**

**- { m, } 出现 m 次以上（三例）**

- + 至少一个，相当于{ 1, }

- \* 零个或多个，相当于{ 0, }

- ？零个或一个，相当于{ 0,1 }

- ^ 表示开头/^a/i , 第一个必须是 a

- $ 表示末尾/a$/i , 最后一个必须是 a

**- 如果在正则表达式中同时使用 ^和$ 则要求字符串完全符合正则表达式**

例子：

**var reg = /a{6}/i**

**alert(reg.test("aaaaaa"))**

例子：

**var reg = /(ab){3}/i**

**alert(reg.test("ababab"))**

例子：

**var reg = /ab{1,3}c/**

**alert(reg.test("abbbc"))**

**手机号的规范：**

var reg = /^1[3-9][0-9]{9}$/

var number = +prompt()

var t = reg.test(number)

if( t == true){

alert("注册成功")

}

-----------------------------------------------------------------------------

第六十五课（正则表达式语法）

1.“ . " 表示任意字符

2.在正则表达式中使用 \ 来表示转义字符

3.字面量使用方法:/\w/

4.

\w

- 任意字母，数字，\_ 为true

\W

- 除了任意字母，数字，\_ 为true

\d

- 任意数字 为true

\D

- 除了任意数字 为true

\s

- 空格 为true

\S

- 除了空格 为true

\b

- 单词边界 为true

例子：/\b child \b/

- 划定独立单词，单词前后无任何多余

\B

- 除了单词边界 为true

例子：

var name = " wdn md “

alert(name.replace(/^\s\*|\s\*$/g,"" ))

输出：

wdn md

-----------------------------------------------------------------------------

## （邮件的正则）

//创造规则，秩序永恒

var reg = /^\w{3,}(\.\w+)\*@[A-z0-9]+(\.[A-z]{2,5}){1,2}$/

var email = "wangxubo780@gmail.com"

alert(reg.test(email))

-----------------------------------------------------------------------------

第六六课（DOM简介）

六六大顺，希望开个好头

这也是JS的正式开始了八......

1.Dom全称分为三个单词

Document Object Model 文档，对象，模型

JS 通过 DOM 来对 HTML 文档进行操作，只要理解了DOM就可以随心所欲的

操作 WEB 页面

- 文档示的是整个 HTML 网页文档

- 对象表示将网页中的每个部分都转换为了一个对象

- 使用模型来表示对象之间的关系，这样方便我们获取对象

2.节点（Node）

- 文档节点：整个 HTML 文档

- 元素节点：HTML 文档中的 HTML 标签

- 属性节点：元素的属性

- 文本节点：HTML 标签中的文本内容

3.浏览器为我们提供文档节点对象这个对象是 window 属性

可以在页面中直接使用，文档节点代表的是整个网页

更改按钮例子：

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Just Monika</title>

<button id = "btn">我不是一个按钮</button>

<script type="text/javascript">

//正文

var btn = document.getElementById("btn")

btn.innerHTML = "My name is Monika"

</script>

</head>

<body>

</body>

</html>

输出：

[object HTMLButtonElement]

-----------------------------------------------------------------------------

第六十七课（事件的简介）

1.事件就是用户和浏览器之间的交互行为

比如，点击按钮，关闭窗口，鼠标移动

2.我们可以在事件对应属性中设置一些JS代码

这样当事件被触发时，这些代码会执行

但这种写法我们成为结构和行为耦合，不方便维护

例子：

<button id = "btn" onmousemove = "alert('JUST MONIKA')">我是一个按钮</button>

3.可以为按钮的对应事件绑定处理函数的形式来响应事件

例子：

var Sayori = document.getElementById("btn")

Sayori.onclick = function( ){

alert("HELP ME")

}

属于回调函数

像这种为单机事件绑定的函数，我们成为单击函数（依次类推）

整合：

var btn = document.getElementById("btn")

btn.innerHTML = "wdnmd"

btn.onclick = function(){

alert("nmbs")

}

-----------------------------------------------------------------------------

第六十八课（文档的加载）

1.onload事件会在整个页面加载完成后触发

- 为window绑定一个onload事件

该事件的响应会在页面加载完之后执行

这样才可以确保我们的代码执行时所有的DOM执行完毕

例子：

<title>Just Monika</title>

<script type="text/javascript">

window.onload = function(){

var btn = document.getElementById("btn")

btn.onclick = function(){

alert("nmbs")

}

}

</script>

<button id = "btn" >我是一个按钮</button>

-----------------------------------------------------------------------------

第六十九课（dom查询）

通过document对象调用的方法

**1.getElementById( )**

- 通过id属性获取一个元素节点对象

**2.getElementsTagName( )**

- 通过标签名获取一组元素节点对象

**3.getElementsByName( )**

- 通过Name属性获取一组元素节点对象

---------------------------

2.这个方法会给我们返回一个类数组对象，所有查询到的元素都会封装到对象中

即使查询到的元素只有一个，也会封装到数组中返回

3.innerHTML用于获取元素内部的HTML代码的

对于自结束标签，这个属性没有意义

- 如果需要读取元素节点属性：

直接使用元素.属性名

例子：元素.id 元素.name 元素.value

注意：class属性不能采用这种方式

读取class属性时需要使用 元素.className

(由于没学 css 和 HTML ，所以无法完成笔记)

-----------------------------------------------------------------------------

第七十课（图片切换练习）

1.代码直接奉上，我还没学css，这节课听的迷迷糊糊的

-----------------------------------------------------------------------------

第七十一课（Dom查询）

1. **getElementsByTagName( )**

- 获取元素节点的子节点

- 方法：返回当前节点的所有子节点.获取元素节点的子节点

**2.childNodes**

- 属性：表示 当前节点的所有子节点

- 属性会获取包括文本节点在内的所有节点

- 就是把换行也加进去了

- 推荐使用第三个

**3.children**

- 属性可以获取当前元素的所有子元素

- 推荐使用这个

**4.firstChild**

- 属性：表示当前节点的第一个子节点

- 属性会获取包括文本节点在内的所有节点

- 就是把换行也加进去了

**5.firstElementChild**

- 获取当前元素的第一个元素

**6.lastChild**

- 属性：表示当前节点的最后一个子节点

-----------------------------------------------------------------------------

第七十二课（Dom查询）

**1.parentNode**

- 属性，表示当前节点的父节点

**2.previousSibling**

- 属性，表示当前节点的前一个兄弟节点

- 属性会获取包括文本节点在内的所有节点

- 就是把换行也加进去了

- 推荐使用第三个

**3.previousElementSibling**

- 属性，表示当前节点的前一个兄弟节点

**4.nextSicling**

- 属性，表示当前节点的后一个兄弟节点

**5.innerText**

- 该属性可以获取到元素内部的文本内容

- 它和innerHTML类似，不同的是他会将HTML去除

**6.nodeValue**

- 根据节点的类型设置，或返回节点的值

-----------------------------------------------------------------------------

第七十三课（dom查询的剩余方法）

1.<body>

- 在 document 中 有个属性就叫 Body ，它保存的是 Body 的引用

- var body = document.body

2.<html>

- documentElement 保存的是 html 跟标签

- var html = document.documentElement

3.all

- document.all 代表页面中的所有元素（就是标签）

- 现在听说 弃用 了 ， 所以不必纠结

**4.getElementsByClassName( )**

- 根据元素的 class 属性值查询一组元素节点对象

**5.querySelector( )**

- 需要一个选择器的字符串作为参数，可以根据一个css选择器来查询一个

元素节点对象

- 该方法总会返回唯一的一个元素，如果满足条件的元素有多个，则会返回

第一个

**6.querySelectorAll( )**

- 该方法和 querySelector( ) 用法一致，不同的是它会将符合条件的

封装到数组中返回

- 数组返回

-----------------------------------------------------------------------------

第七十四课（dom增删改）

**1.createElement( )**

- 创建元素节点

- 需要一个标签名作为参数，根据该标签名创建元素节点对象

- 将创建好的对象作为返回值反回

**2.createTextNode( )**

- 创建文本节点对象

- 需要以文本内容为参数，根据该内容创建文本节点，并将新的节点返回

**3.appendChild( )**

- 把新的子节点添加到指定节点

- 向一个父节点中添加一个新的子节点

- 用法：父节点.appendChild( 子节点 )

var city = document.getElementById("city")

city.innerHTML += "<li>广州</li>"

**4.insertBefore( )**

- 将指定字节点前面插入新的字节点

- 语法：

父节点 . insertBefore ( 新节点，旧节点 )

**5.replaceChild( )**

- 替换子节点

- 语法：

父节点.replaceChild( 新节点，旧节点 )

**6.removeChild( )**

- 删除子节点

- 语法：

父节点.removeChild(子节点)

子节点.parentNode.removeChild(子节点)

**7.innerHTML**

- 使用innerHTML也能实现DOM增删改的相关操作

- 但有缺点，所以两种方式结合使用

推荐使用方式：

var city = document.getElementById("city")

var li = document.createElement("li")

li.innerHTML = "广州"

city.appendChild(li)

-----------------------------------------------------------------------------

第七十五课（添加删除记录 - 删除）

**1.value**

- 获取HTML中的输入框内容

**2.confirm( )**

- 用于弹出一个带确认和取消的提示框

- 需要一个字符串作为参数

-----------------------------------------------------------------------------

第七十六课（操作内联样式）

1.通过Js修改元素样式

- 语法：元素 .style. 样式名 = “样式值”

注意：如果 CSS 的样式名中含有 —

这种名称在JS中是不合法的，比如 background-color

需要将这种样式名改为 **驼峰命名法**

去掉 - 然后将 - 后的字母大写

- 我们通过style属性设置的样式都是内联样式

而内联样式有较高的优先级，通过JS修改的样式往往会立即显示

但是如果在样式中写了 ！ important ，则此时样式会有最高的优先级

通过JS无法覆盖该样式，会导致JS修改样式失效

---------------------------

2.通过Js修改元素样式

- 语法：元素.style.样式名

通过style属性设置和读取的都是内联样式

无法读取样式表中的样式

-----------------------------------------------------------------------------

第七十七课（操作内联样式）

1.getComputedStyle( )

- 获取元素当前的样式

需要两个参数

- 要获取样式的元素

- 可以传递一个伪元素，**一般都传null**

该方法会返回一个对象，对象中封装了当前元素对应的样式

var 变量名 = getComputedStyle(元素，null)

alert( 变量名 . 样式名 )

如果获取的样式没有设置，则会获取到真实的值，而不是默认值

( 三原色的值 )

-----------------------------------------------------------------------------

第七十八课（其他样式相关的属性）

-- client( 客户端 ) --

1.clientHeight

- 返回元素可见高度

2.clientWidth

- 返回元素可见宽度

- 这两个属性可以获取元素的可见高度和宽度

- 这些属性都是不带 px 的，返回都是一个数字，可以直接进行计算

- 会获取元素宽度和高度，包括内容区和内边距

- 这些属性都是只读的，不能修改

（实时数据）

---------------------------

3.offsetHeight

- 返回元素高度

4.offsetWidth

- 返回元素宽度

- 这两个属性可以获取元素整个的可见高度和宽度

- 会获取元素宽度和高度，包括内容区和内边距和边框

- 这些属性都是只读的，不能修改

---------------------------

5.offsetParent

- 返回元素的偏移容器

- 会获取到离当前元素最近的开启了定位的祖先元素

如果所有的祖先元素都没有开启定位，则返回body

---------------------------

6.offsetLeft

- 当前元素相对于其定位父元素的水平偏移量

7.offsetTop

- 当前元素相对于其定位父元素的垂直偏移量

---------------------------

8.scrollHeight

- 返回元素整体高度

9.scrollWidth

- 返回元素整体宽度

---------------------------

10.scrollLeft

- 可以获取水平滚动条滚动的距离

11.scrollTop

- 可以获取垂直滚动条滚动的距离

例子：

当满足scrollHeight - scrollTop == clientHeight

说明垂直滚动条滚动到底了

当满足scrollWidth - scrollLeft == clientWidth

说明水平滚动条滚动到底

alert(box4.scrollHeight - box4.scrollTop); // 600

---------------------------

12.onscroll

- 该事件会在元素的滚动条滚动时触发

13.disabled

- 设置或返回 checkbox 是否应被禁用

- true 则元素禁用

- false 则元素不禁用

-----------------------------------------------------------------------------

第七十九课( 获取鼠标位置 )

1.onmousemove

- 鼠标移动事件

事件对象

- 当事件的响应函数被触发时，浏览器每次都会将一个事件对象作为实参传递进响应函数

在事件对象中封装了当前事件一切的相关信息

比如：

鼠标的坐标

键盘哪个按键按下

鼠标滚轮滚动方向

---------------------------

2.clientX

- 返当事件被触发时，鼠标指针的水平坐标

- 可见位置 x与y

3.clientY

- 返当事件被触发时，鼠标指针的垂直坐标

- 可见位置 x与y

---------------------------

4.pageX

- - 返当事件被触发时，鼠标指针的水平坐标

- 整个页面

5.pageY

- 返当事件被触发时，鼠标指针的水平坐标

- 整个页面

-----------------------------------------------------------------------------

第八十课( 事件的冒泡 )

1.所谓的冒泡指的就是事件的向上传导，当后代元素上的事件被触发时，

- 其祖先元素的相同事件也会被触发

2.在开发中大部分情况冒泡都是有用的，但可以通过事件对象来取消冒泡

3.关键对象：

- cancelBubble

- 取消冒泡

例子：

var box1 = document.getElementById("box1")

box1.onclick = function( event ){

alert("我是div的单击响应函数")

//取消冒泡

event.cancelBubble = true

}

-----------------------------------------------------------------------------

第八十一课( 事件的委派 )

1.指事件统一绑定给元素的祖先元素，这样当后代元素上的事件被触发时

会一直冒泡到祖先元素，从而通过祖先元素的响应函数来处理

事件。

- 事件委派是利用了冒泡，通过委派可以减少事件绑定的次数，提高程序性能

- target

- 返会触发此事件的元素

-----------------------------------------------------------------------------

第八十二课( 事件的绑定 )

1.使用 对象.事件 = 函数 的形式绑定响应函数.

它只能同时为一个元素的一个事件绑定一个响应函数

不能绑定多个，如果绑定多个，则后面覆盖前面

2. addEventListener( )

- 通过这个方法也可以为元素绑定响应函数

- 参数：

1.事件的字符串，不要 on

2.回调函数，当事件被触发时该函数会被调用

3.是在捕获阶段触发事件，需要一个布尔值，一般都传 false

4.使用 addEventListener（ ）可以同时为一个元素绑定多个响应函数

5.这样当事件被触发时，响应函数将会按照函数的绑定顺序执行

例子：btn01.addEventListener( "click" , function( ){ } , false )

-----------------------------------------------------------------------------

第八十三课( 事件的传播 )

事件的传播

- 关于事件的传播 网景公司 和 微软公司 有不同的理解

- 微软公司认为事件应该是由内向外传播，也就是当事件触发时，应该先触发当前元素上的事件，

然后再向当前元素的祖先元素上传播，也就说事件应该在冒泡阶段执行。

- 网景公司认为事件应该是由外向内传播的，也就是当前事件触发时，应该先触发当前元素的最外层的祖先元素的事件，

然后在向内传播给后代元素

W3C综合了两个公司的方案，将事件传播分成了三个阶段

1.捕获阶段

- 在捕获阶段时从最外层的祖先元素，向目标元素进行事件的捕获，但是默认此时不会触发事件

2.目标阶段

- 事件捕获到目标元素，捕获结束开始在目标元素上触发事件

3.冒泡阶段

- 事件从目标元素向他的祖先元素传递，依次触发祖先元素上的事件

- 如果希望在捕获阶段就触发事件，可以将addEventListener()的第三个参数设置为true

一般情况下我们不会希望在捕获阶段触发事件，所以这个参数一般都是false

- IE8及以下的浏览器中没有捕获阶段

例子：btn01.addEventListener( "click" , function( ){ } , true )

-----------------------------------------------------------------------------

第八十四课( 拖拽 )

1.onmousedown

- 当鼠标按钮被按下

2.onmouseup

- 当鼠标按钮松开时

-----------------------------------------------------------------------------

第八十五课( 滚轮的事件 )

1.onwheel

- 滚轮事件

- 全支持，无兼容问题

2.deltaY

- 可以获取鼠标滚轮滚动方向

- 只看正负

- 全支持，无兼容问题

-----------------------------------------------------------------------------

第八十六课( 键盘事件 )

1.onkeydown

- 某个键盘按键被按下

- 不松手则一直触发

2.onkeyup

- 某个键盘按键被松开

- 键盘事件一般都会绑定给一些可以获取到焦点的对象或 document

3.keyCode

- 通过 keyCoad 来知道按键的编码

4.altKey 按下 true 否则 false

5.ctrlKey 按下 true 否则 false

6.shiftKey 按下 true 否则 false

-----------------------------------------------------------------------------

第八十七课 ( BOM 简介 )

1.BOM

- 浏览器对象模型

- Bom 可以使我们通过Js来操作浏览器

- 在 Bom 中为我们提供了一组对象，用来对浏览器进行操作

- BOM对象

- window

- 代表整个浏览器窗口,同时 window 也是网页中的全局对象

- Navigator

- 代表当前浏览器的信息,通过该对象来识别不同的浏览器

- Location

- 代表当前浏览器的地址信息,通过Location可以获取地址栏信息,

或者操作浏览器跳转页面

- History

- 代表浏览器的历史纪录,可以通过该对象来操作浏览器的历史记录,

由于隐私原因,该对象不能获取到具体的历史记录,

只能操作浏览器向前或向后翻页,而且该操作只在当次访问时有效

- Screen

- 代表用户的屏幕信息，通过该对象可以获取到用户的显示器的相关信息

- 这些 Bom 对象在浏览器中都是作为 Window 对象的属性保存的，

可以通过 window 对象来使用

-----------------------------------------------------------------------------

第八十八课 ( BOM - Navigator )

2.Navigator

- 代表的当前浏览器的信息，通过该对象可以来识别不同的浏览器

- 由于历史原因，Navigator对象中的大部分属性都已经不能帮助我们识别浏览器了

- 一般我们只会使用userAgent来判断浏览器的信息，

- userAgent是一个字符串，这个字符串中包含有用来描述浏览器信息的内容，

不同的浏览器会有不同的userAgent

- 用法查文档

本节课重点：

if("ActiveXobject" in window){

alert("用你妈IE")

}

-----------------------------------------------------------------------------

第八十九课 ( BOM - History)

1.查文档去，不难

-----------------------------------------------------------------------------

第九十课 ( BOM - Location）

assign( )

- 用来跳转到其他的页面，作用和直接修改location一样

reload( )

- 用于重新加载当前页面，作用和刷新按钮一样

- 如果在方法中传递一个true，作为参数，则会强制清空缓存刷新页面

replace( )

- 可以使用一个新的页面替换当前页面，调用完毕也会跳转页面

不会生成历史记录，不能使用回退按钮回退

-----------------------------------------------------------------------------

第九十一课 ( 定时器简介）

1.Js的程序执行速度是非常非常快的

如果希望一段程序，可以间隔一段时间执行一次，可以使用定时调用

2.setInterval( )

- 定时调用

- 可以将一个函数，每隔一段时间执行一次

- 参数:

1.回调函数，该函数会每隔一段时间被调用一次

2.每次调用间隔的时间，单位是毫秒

- 返回值：

1.返回一个 Number 类型的数据

2.这个数字来作为定时器的唯一标识

3.clearInterval( )

- 用来关闭一个定时器

- 如果参数是一个有效的定时器的标识，则停止对应的定时器

如果参数不是一个有效的标识，则什么也不做

-----------------------------------------------------------------------------

第九十二课 ( 延时调用 ）

1.setTimeout( )

- 延时调用

·延时调用一个函数不马上执行，而是隔一段时间以后在执行，而且只会执行一次

·延时调用和定时调用的区别，定时调用会执行多次，而延时调用只会执行一次

·延时调用和定时调用实际上是可以互相代替的，在开发中可以根据自己需要去选择

2.clearTimeout( )

·使用clearTimeout( )来关闭一个延时调用

-----------------------------------------------------------------------------

第九十三课 ( 类的操作 ）

1.通过style属性来修改元素的样式，每修改一个样式，浏览器就需要重新渲染一次页面

- 这样的执行的性能是比较差的，而且这种形式当我们要修改多个样式时，也不太方便

2.我们可以通过修改元素的class属性来间接的修改样式

- 这样一来，我们只需要修改一次，即可同时修改多个样式，

- 浏览器只需要重新渲染页面一次，性能比较好，

- 并且这种方式，可以使表现和行为进一步的分离

-----------------------------------------------------------------------------

第九十四课 ( 二级菜单 ）

1.我的天...这后面真难

-----------------------------------------------------------------------------

第？&·课 ( ？级菜单 ）

1.???

-----------------------------------------------------------------------------

第$#^&#$课 ( ？@菜}{ ）

1.???

-----------------------------------------------------------------------------

第\_^&\*&^课 ( ^%&？？ ）

1.???

-----------------------------------------------------------------------------

第~\*/·课 **( /)!^#？ ）**

1.???

-----------------------------------------------------------------------------

第NaN课 **( NaN ）**

1.???

-----------------------------------------------------------------------------

# **第一百课 ( 结束了？** ) **- JSON**

- JS 中的对象只有JS自己认识，其他的语言都不认识

- JSON 就是一个特殊格式的字符串，这个字符串可以被任意的语言所识别，

并且可以转换为任意语言中的对象，JSON 在开发中主要用来数据的交互

- JSON

- JavaScript Object Notation JS 对象表示法

- JSON 和 JS 对象的格式一样，只不过 JSON 字符串中的属性名必须加双引号

其他的和JS语法一致

- JSON 分类：

1.对象 { }

2.数组 [ ]

- JSON 中允许的值：

1.字符串

2.数值

3.布尔值

4.null

5.对象

6.数组

---------------------------

1.在JS中，为我们提供了一个工具类，就叫 JSON

- 这个对象可以帮助我们将一个 JSON 转换为 JS对象

也可以将一个 JS 对象转换为 JSON

- JSON.parse

- JSON ——>> JavaScript

- 可以将JSON字符串转换为js对象

- 它需要一个JSON字符串作为参数，会将该字符串转换为JS对象并返回

- JSON.stringify( )

- JavaScript——>> JSON

- 可以将JS对象转换为 JSON 字符串

- 需要将一个 JS 对象作为参数，会返回一个 JSON 字符串

————————————————————————————————

Hello World ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Figure 1

Just Monika