复习:

console.log("向控制台打印一句话");

console.warn("打印警告信息");

console.error("打印错误信息");

向页面打印

document.write("打印页面一句话");

document.write("<h1>我是标题</h1>");

可以识别页面标签,把字符串里面的标签解析出来放到页面上

alert(“普通弹窗”);

输入框弹窗

prompt("请输入:");

选择弹窗

confirm("请选择:");

JS 简单易懂 弱类型

弱类型: 类型是忽略的

2.数据类型使用注意

Number:

浮点:小数是非精确数

浮点型不要进行相等比较

整数:

*//整数最大值*alert(Number.MAX\_VALUE);

在最大值的基础上 进行加法 是不会增加 除法会生效

乘法变成无穷大

console.log("abc"/15);*//NaN*

NaN not a number 不是一个数字

* isNaN() :任何不能被转换为数值的值都会导致这个函数返回 true
* *//判断括号里面不是一个数字*console.log(isNaN(50));//false  
  console.log(isNaN("xx"));//true  
  console.log(isNaN(NaN));//true  
  console.log(isNaN(undefined));//true

字符串的单双引号之间可以互相嵌套

console.log("呵呵:'哈哈'");

document.write("<h1 class='box'>我是box标题</h1>");

*//字符串长度*console.log(*str3*.length);

转义字符:

*转义字符: 1.避免混淆 2.赋予某些字符特殊意义*

console.log("小明说:\"我今天走运了!\"");

JS中 字符串值不可以被更改

字符串拼接操作

*//字符串和任意基本数据类型 都可以进行+运算 运算结果都是拼接*console.log("aa"+"bb");  
console.log("aa"+50);  
console.log("aa"+false);  
console.log("aa"+6.32);  
console.log("aa"+NaN);

*//字符串和基本数据类型拼接时 结果都是字符串*console.log("aa"+50+60);//aa5060  
console.log(typeof ("aa"+50));//String  
console.log("aa"+50+60+"bb");//aa5060bb

类型转换

Boolean()转换

console.log(Boolean(0));*//false*console.log(Boolean(163));*//true*console.log(Boolean("hehe"));*//true*console.log(Boolean(1.36));*//true*console.log(Boolean(undefined));*//false*console.log(Boolean(NaN));*//false*

字符串转换

任意类型都可以调用toString()

console.log(typeof *num*.toString())

String(63)也可以转换成字符串

*//字符串拼接 转换成字符串*var *num1*=63;  
console.log(typeof (*num1*+""));

var *str1*="60x";  
*//如果字符串不是一个纯数字序列那么 结果是NaN*console.log(Number(*str1*));

var *str*="60x";  
*//parseInt()函数 如果字符串是以数字开头的 那么解析到不是数字为止*console.log(window.parseInt(*str*)+5);

var *str*="x60";  
  
console.log(parseFloat(*str*)+5);

用法同上 只不过 parseFloat 没有第二个参数作为进制 解析

算术运算符

a++;*//a=a+1; 自增一*++*b*;*//b=b+1; 自增一*console.log("a:"+a);  
console.log("b:"+*b*);

var a=5;  
var b=3;  
console.log("a:"+a);  
console.log("b:"+b);  
// 运算符 操作数  
//当++ -- 参与其他运算时 从左往右看  
//++放在操作数后边时 先看到操作数 那么先把操作数当前的值拿出来给前边使用 然后再+1  
//++放在操作数前边时 先看到运算符 那么就先运算++ 然后把++后的结果给前边使用  
  
//--同理  
var num1=a++;  
console.log("当前a的值:"+a);  
var num2=++a;  
  
console.log("a:"+a);  
console.log("num1:"+num1);  
console.log("num2:"+num2);

a+*b*;*//双目运算符*a++;*//单目运算符  
  
//三目运算符 判断(结果是true或者false)?值1:值2*var *str*=a>*b*?"<div class='box'>我是div</div>":"<div class='box1'>我是div</div>";  
document.write(*str*);

比较运算:

*//比较运算符的结果都是布尔值 比大小  
// > < >= <= != == ===*

// console.log(a>b);  
// console.log(a<b);  
// console.log(a!=b);  
// console.log(a<=b);

// console.log(c==d);  
// console.log(c!=d);  
// console.log(c===d);  
// console.log(c!==d);

== != 类型不一致 会尝试转换

=== !== 类型不一致 直接[]判断

赋值运算符

* = += -= \*= /= %=

逻辑运算

与运算: 全真才真 一假即假

//与 && 全真才真 一假即假  
// console.log(true && true);//true  
// console.log(false && true);//false  
// console.log(true && false);//false  
// console.log(false && false);//false

或运算:

// || 或 全假才假 一真即真  
// console.log(true || true);//true  
// console.log(false || true);//true  
// console.log(true || false);//true  
// console.log(false || false)//false

非运算 取反

短路与 短路或

在同为与的运算下 看到false直接结束 结果为false

在同为或的运算下 看到true直接结束 结果为true

在与运算和或运算 和非运算 的里面 操作数如果不是布尔值 规则:

非:

*//在逻辑运算中 0 false undefine NaN ""(空字符串) null 统一当做false*

在与运算和或运算的里面 操作数如果不是布尔值 结果不是布尔值 是具体的其他值

与:

在多个与(&&)情况下: 遇到false停止 取导致false的那个值

如果执行结束没有遇到false 则取最后值

或

在多个或(||)情况下: 遇到true停止 取导致true的那个值

如果执行到结束没有遇到true 则取最后值

3.流程控制语句

顺序结构

//顺序结构 从上到下 一旦先后顺序颠倒对结果有直接影响  
// console.log("灰色背景");  
// console.log("敌军还有5秒到达战场");  
// console.log("我能杀他");  
  
  
 console.log(num || 5);  
  
 var num=10;