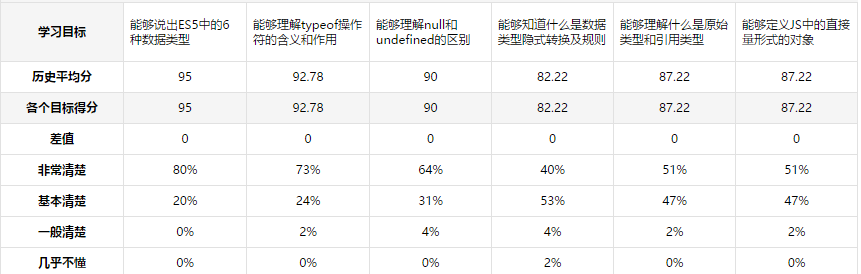
一、前天反馈

**1、ES5中6种数据类型**：

* 数值型：number（int、float）
* 字符串：string
* 布尔：Boolean
* 未定义：undefined
* 空：null
* 对象：Object（Array、Object）

**2、typeof作用**：

判断变量的数据类型：

typeof 123 返回 number

typeof "hello" 返回string

typeof true 返回boolean

typeof undefined 返回undefined

typeof null 返回 object

typeof ['apple', 'banana'] 返回 object

typeof {name:'李四', age:23} 返回 object

区分null、数组、对象使用instanceof。

**3、null和undefined的区别**

一个变量被声明，但是没有给值，则它的值就是undefined；手动设置一个变量的值为null，则这个变量的值才为null。

转换为布尔之后，他们都是false。

**4、直接量形式的对象**

var obj = {};

var obj1 = {name:'lisi', age:20};

var obj2 = {

name:'lisi',

age:20,

fn:function(){

console.log(this.age); //this表示当前的对象obj2

}

}

**5、原始类型和引用类型**

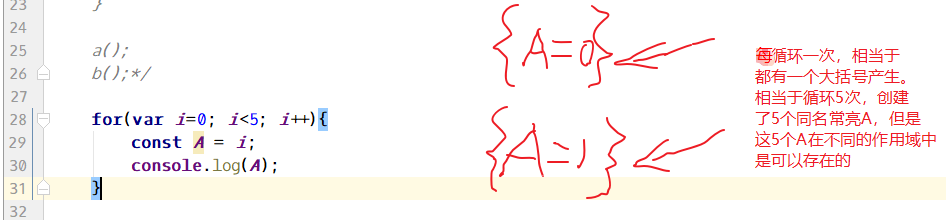
PHP中数组是值传递的，对象是引用传递。

JS中，数组和对象都是引用传递的。

**6、常量**

定义常量用const

常量不能在同一个作用域中从新定义。



二、今日目标

1. 能够知道什么是数据类型隐式转换及规则
2. 能够用代码写出JS函数的定义形式和调用形式（多种）
3. 能够说出函数的形参和实参的区别
4. 能够理解匿名函数立即调用的定义形式和调用形式
5. 能够理解变量作用域的概念
6. 能够说出局部变量和全局变量的区别
7. 能够说出作用域的访问规则及原理
8. 能够说出js中常见的系统对象（Math、Date、String、Array）
9. 能够查阅系统对象的常用方法

三、数据类型转换

3.1、转换成字符串

* 使用字符串方法toString([进制])转换，但是null和undefined无法转换。
* 使用顶层函数String()可以将任何数据类型转换成字符串，包括null和undefined。
* 在其他数据类型和字符串类型进行连接（+操作）操作时，会自动对其他数据类型使用String()强制转换成字符串，然后在相加（连）

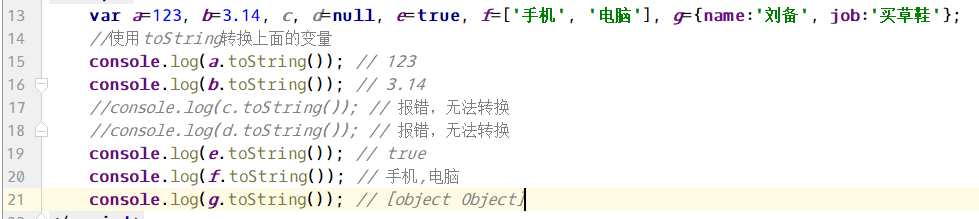
将其他数据类型的值转换成字符串，可以使用toString()，也可以使用String()。区别在于String可以转换任何类型的值为字符串，toString()不能转换undefined和null。区别二是语法不同。

**使用语法**：

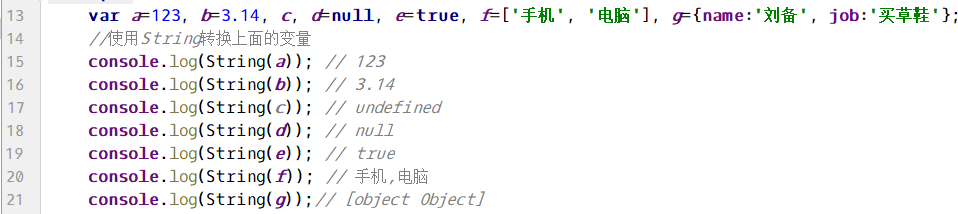
使用toString() ： 待转换的变量.toString();

使用String() ： String(待转换的变量);

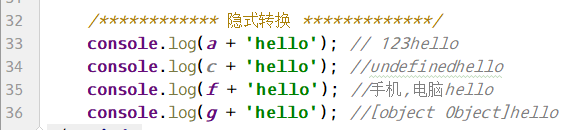
下面演示使用toString转换：



下面演示使用String转换：



**隐式转换**：在其他数据类型和字符串类型进行连接（+操作）操作时，会自动对其他数据类型使用String()强制转换成字符串，然后在相加（连）



注释：[object Object] 第一个object表示是何种数据类型，第二个Object表示是那种类型的对象

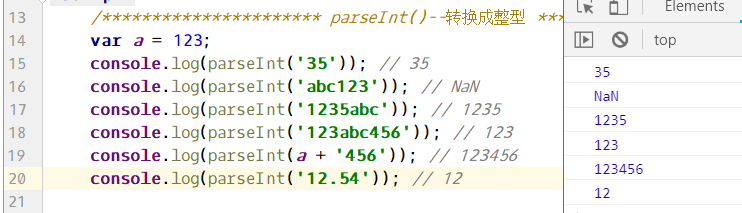
3.2、转换成数值型

3.2.1、parseInt() -- 转换成整型

parseInt() 方法首先查看位置 0 处的字符，判断它是否是个有效数字；如果不是，该方法将返回 NaN(not a number)，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置 1 处的字符，进行同样的测试。这一过程将持续到发现非有效数字的字符为止，此时 parseInt() 将把该字符之前的字符串转换成数字。

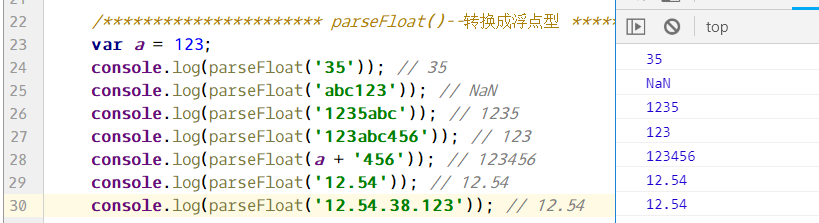
例如，如果要把字符串 "a12345red" 转换成整数，那么 parseInt() 将返回 12345，因为当它检查到字符 r 时，就会停止检测过程。

字符串中包含的数字字面量会被正确转换为数字，比如 "0xA" 会被正确转换为数字10。不过，字符串 "22.5" 将被转换成22，因为对于整数来说，小数点是无效字符。



3.2.2、parseFloat() -- 转换成浮点型

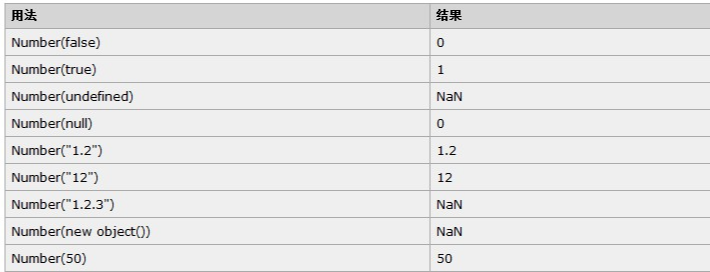
道理和转换成整型道理一样，只不过浮点型允许有一个小数点出现。



3.2.3、Number() -- 强制转换

Number() 函数的强制类型转换与 parseInt() 和 parseFloat() 方法的处理方式相似，只是它转换的是整个值，而不是部分值。

用 Number() 进行强制类型转换，"1.2.3" 将返回 NaN，因为整个字符串值不能转换成数字。如果字符串值能被完整地转换，Number() 将判断是调用 parseInt() 方法还是 parseFloat() 方法。



3.3、转换成布尔型

显示的转换是使用Boolean()函数，对需要转换的内容进行转换。

以下内容在转换成布尔值时会被转换成false：

* 数值型的 0
* 数值型的 0.0
* 布尔型的 false
* 空字符串 ""
* 非数字 NaN
* undefined
* null

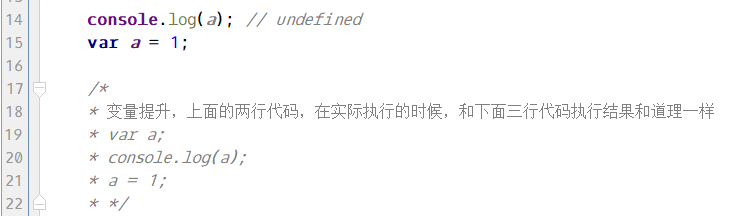
除此以外，其他所有值都会转换成true，包括 "0"、空数组 [] 和空对象 {} 。

四、流程控制

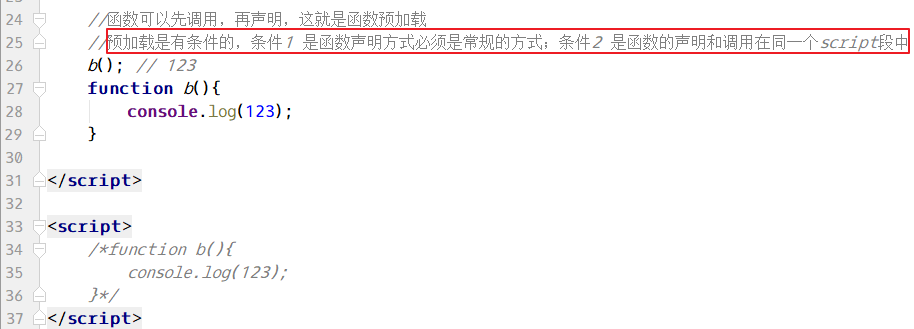
4.1、顺序结构

顺序结构，体现了代码从上到下依次执行。但是在JS中，要注意两点能够打破顺序结构的知识点，一点是变量提升。另一点是函数预加载。

变量提升的例子：



函数预加载的例子：



4.2、分支结构

if()…else….

if()…else if()…else…

switch (待比较的变量) {

case xxx:

xxx;

break;

}

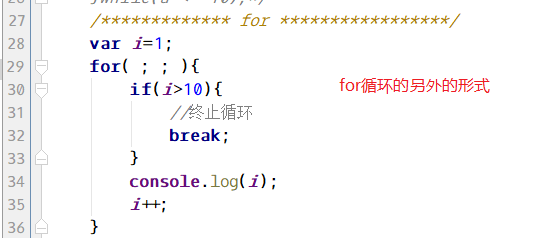
*//new Date() 得到了时间日期对象  
//对象调用里面的成员使用点语法。 new Date().getDay();***var *week*** = **new** Date().getDay(); *//获取一个数字格式的星期  
//console.log(week);***var *week\_zh\_cn***; *//中文格式的星期***switch** (***week***) {  
 **case** 2:  
 ***week\_zh\_cn*** = **'二'**;  
 **break**;  
 **case** 3:  
 ***week\_zh\_cn*** = **'三'**;  
 **break**;  
 **case** 1:  
 ***week\_zh\_cn*** = **'一'**;  
 **break**;  
 **case** 4:  
 ***week\_zh\_cn*** = **'四'**;  
 **break**;  
 **case** 5:  
 ***week\_zh\_cn*** = **'五'**;  
 **break**;  
 **case** 6:  
 **case** 7:  
 ***week\_zh\_cn*** = **'末'**;  
 **break**;  
 **default**:  
 **console**.log(**'格式错误'**);  
 **break**;  
}  
  
**console**.log(**'今天是星期'** + ***week\_zh\_cn***);

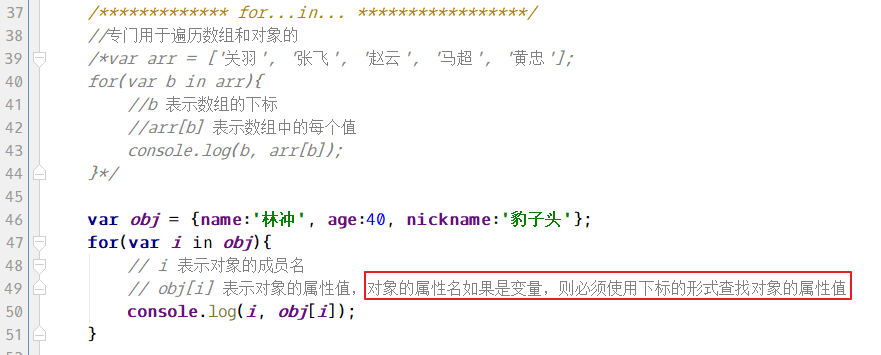
4.3、循环结构

while…

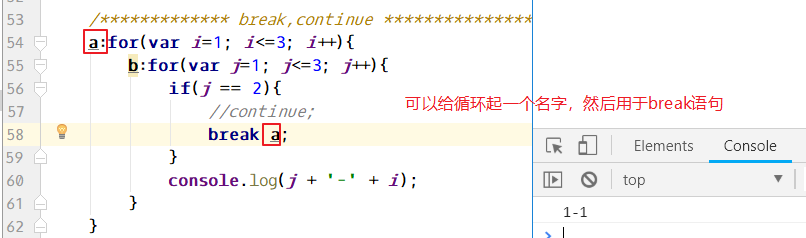
do…while…

for…

for…in…



4.4、continue和break



五、函数基础

5.1、定义与调用

定义语法：

function 函数名(参数列表) {

//函数体

//return xxx;

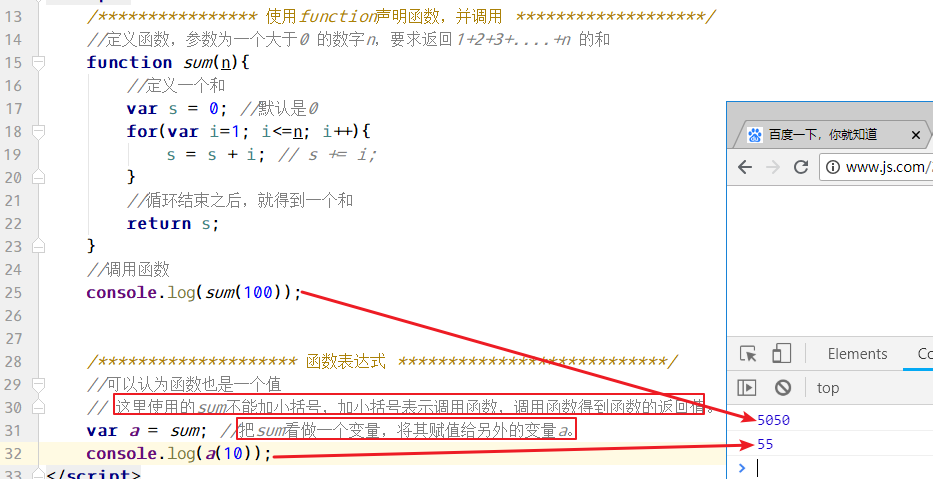
}

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 使用function声明函数，并调用 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
//定义函数，参数为一个大于0 的数字n，要求返回1+2+3+....+n 的和***function** *sum*(n){  
 *//定义一个和* **var** s = 0; *//默认是0* **for**(**var** i=1; i<=n; i++){  
 s = s + i; *// s += i;* }  
 *//循环结束之后，就得到一个和* **return** s;  
}  
*//调用函数***console**.log(*sum*(100));

这种方式定义的函数可以先调用，后定义，也就是函数预加载。

5.2、函数表达式

**函数也是值**。

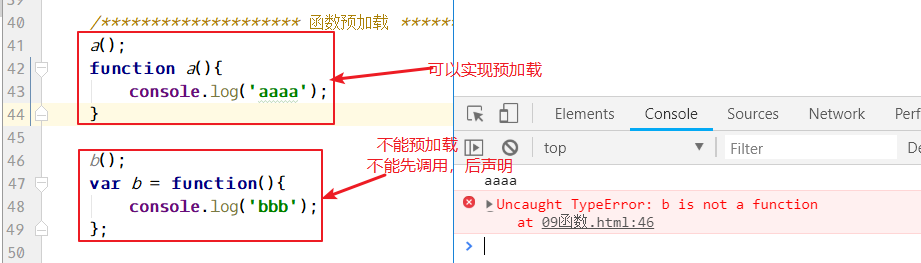


既然函数可以看做是变量，那么就可以像定义变量一样来定义函数，这就是函数表达式的形式：

*//定义一个变量，值是一个函数类型。这种方式定义函数也是可以的***var** *b* = **function** (x){  
 **console**.log(**'今天是周'** + x);  
};  
*b*(**'三'**); *//调用函数*

5.3、函数预加载

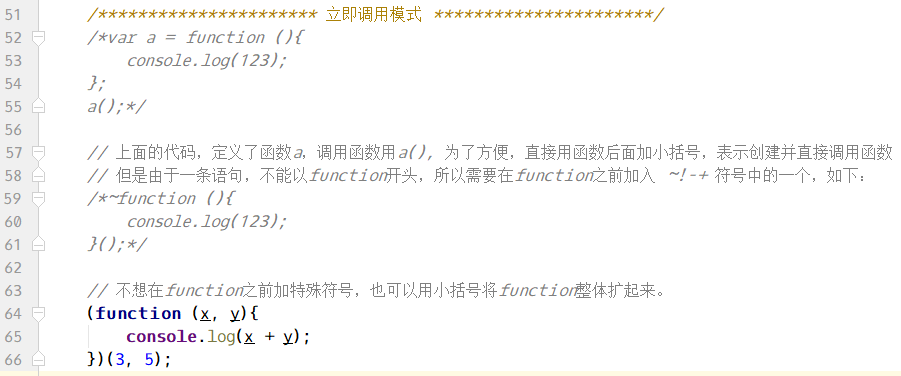
函数预加载指的是哪种方式定义的函数呢？



**函数预加载指定是在同一个script代码段中，由“function xxx(){}”这种方式定义的函数，可以先调用函数，再声明函数**。

注意不要在非函数的代码块中声明函数。

5.4、立即调用模式

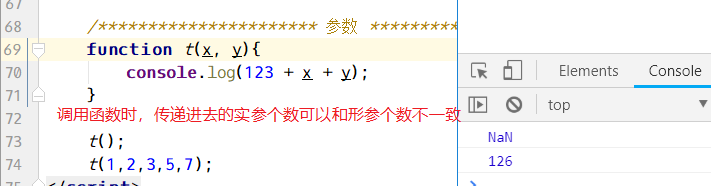


5.5、参数

形参：定义函数时，约定的参数

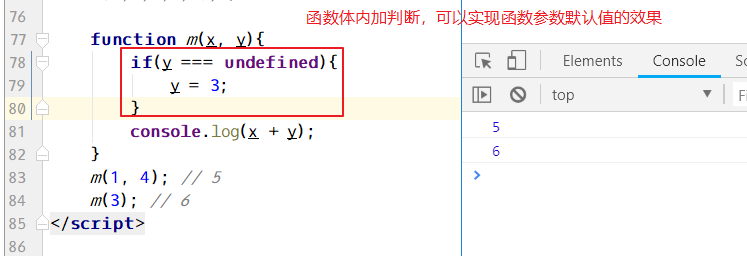
实参：调用函数时，传递给函数的实际值。

JS函数，参数传递非常灵活，定义的形参和实际传入的实参个数可以不一样。



**ES5中，函数的参数不可以用默认值。ES6中，函数的参数可以有默认值的。目前IE11只支持部分ES6的内容**。

那么在ES5中，如何实现形参有默认值的写法呢？



六、作用域

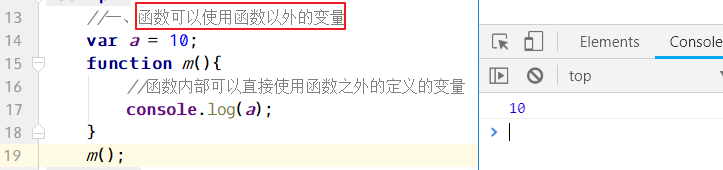
6.1、作用域分类

作用域指的是变量起作用的范围。

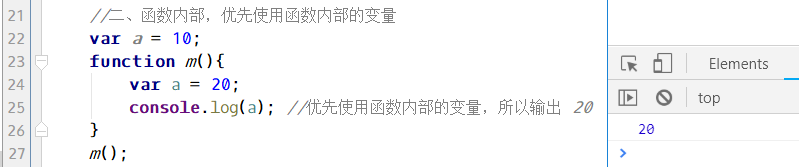
分为**全局作用域**和**局部作用域**。其中局部作用域也叫做函数作用域。

6.2、作用域规则

规则一：函数可以使用函数以外的变量



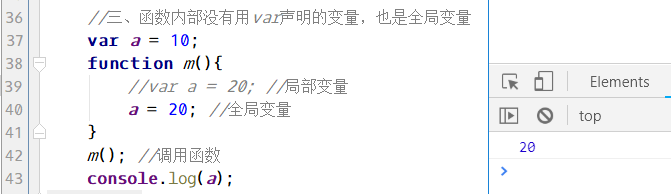
规则二：函数内部，优先使用函数内部的变量



函数内部也会发生变量提升：



规则三：函数内部没有用var声明的变量，也是全局变量

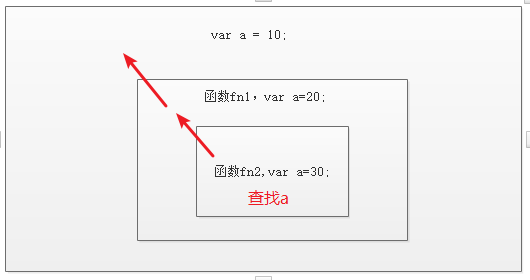


6.3、作用域链

代码：

**var *a*** = 10;  
**function** *fn1*(){  
 **var** a = 20;  
 **function** *fn2*(){  
 **var** a = 30;  
 **console**.log(a);  
 }  
 *fn2*();  
}  
  
*fn1*(); *//*

下面是对应的图示



在内部函数中查找变量的时候，优先从函数内部自身查找，如果没有查到，则向外层查找，如果外层还没有，则继续向上一层查找，一直查询到全局作用域。这种链式的查找方式就是作用域链。

七、标准库（内置对象）

参考资料：

火狐开发社区：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global\_Objects

微软开发社区：https://msdn.microsoft.com/zh-cn/library/aa155110.aspx

w3c官网：http://www.w3school.com.cn/jsref/index.asp

7.1、Math对象--数学对象

Math对象成员调用方式： Math.成员

属性：

Math.PI --- 返回圆周率

方法：

Math.abs(); --- 返回绝对值(正数的绝对值是它本身， 负数的绝对值是它的相反数)

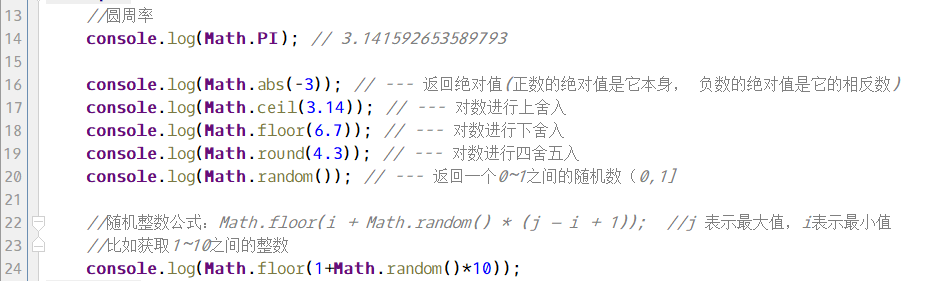
Math.ceil() --- 对数进行上舍入

Math.floor() --- 对数进行下舍入

Math.round() --- 对数进行四舍五入

Math.random() --- 返回一个0~1之间的随机数（0,1]

随机整数公式：Math.floor(i + Math.random() \* (j – i + 1)); //j 表示最大值，i表示最小值



7.2、Date对象--时间日期对象

获取 年、月、日、时、分、秒、（星期）

语法：

var d = new Date(); //实例化，得到对象

//获取年

var nian = d.getFullYear();

*//实例化Date对象***var *d*** = **new** Date();  
*//获取四位数的年***var *nian*** = ***d***.getFullYear();  
*//获取月（0~11）***var *yue*** = ***d***.getMonth() + 1; *//加 1 之后，才是正常的月  
//获取日***var *ri*** = ***d***.getDate();  
*//获取时***var *shi*** = ***d***.getHours();  
*//获取分***var *fen*** = ***d***.getMinutes();  
*//获取秒***var *miao*** = ***d***.getSeconds();  
  
**console**.log(***nian*** + **'年'**+***yue***+**'月'**+***ri***+**'日 '**+***shi***+**':'**+***fen***+**':'**+***miao***);

输出结果：2018年9月1日 17:12:11

7.3、String对象--字符串对象

String对象中提供的方法都是处理字符串。

用法一：

var s = new String(要处理的字符串); //实例化对象

console.log(s.substr(1)); //对象通过点语法调用String对象的成员方法

用法二：把字符串直接当做对象来使用

console.log(要处理的字符串.substr(1));

*//用法一***var *s*** = **new** String(**'hello world'**);  
  
**console**.log(***s***.**length**); *//表示字符串长度***console**.log(***s***.indexOf(**'a'**)); *// 没有查到，返回-1***console**.log(***s***.indexOf(**'l'**)); *//查到了，返回位置 2  
  
//用法二:直接把字符串当做字符串对象来使用***console**.log(**'hello world'**.indexOf(**'e'**)); *// 1***console**.log(**'abcdefg'**.substr(2, 3)); *// cde*

7.4、Array对象--数组对象

语法：

//首先创建一个数组

var arr = ['apple', 'pear'];

var arr = new Array('apple', 'pear');

//因为数组也是对象，那么就可以直接使用 数组.成员方法

arr.push('xigua');

console.log(arr); // ['apple', 'pear','xigua']

*//首先先得到一个数组***var *arr*** = [**'apple'**, **'pear'**];  
*//调用push方法，向数组中添加单元***var *length*** = ***arr***.push(**'xigua'**); *//添加完返回新的长度 3 array\_push()***console**.log(***length***, ***arr***); *//已经是新的数组了，里面有三个水果*

八、总结

1. 能够知道什么是数据类型隐式转换及规则
   1. 隐式转换发送在变量和字符串相连接（+）的时候，会自动的（隐式的）将其他数据类型先根据String()转换成字符串，然后在相连。
   2. 在比较的时候，会将其他数据类型的值转换成布尔型，然后在比较。
2. 能够用代码写出JS函数的定义形式和调用形式（多种）
   1. function a (){} //调用函数 a();
   2. var a = function(){}; //调用函数 a();
3. 能够说出函数的形参和实参的区别
   1. 略
4. 能够理解匿名函数立即调用的定义形式和调用形式
   1. (function(){})(); 立即调用
5. 能够理解变量作用域的概念
   1. 作用域表示变量能够起作用的范围。理解就可以。
6. 能够说出局部变量和全局变量的区别
   1. 局部变量只能局部作用域中使用
   2. 全局变量在所有地方都可以使用
7. 能够说出作用域的访问规则及原理
   1. 函数可以使用函数以外的变量
   2. 函数优先使用函数内部的变量，注意变量提升
   3. 函数内部没有用var声明的变量也是全局变量（需要调用函数）
8. 能够说出js中常见的系统对象（Math、Date、String、Array）
   1. 注意，这里一定要会查手册
9. 能够查阅系统对象的常用方法