JS代码原则：高内聚，弱耦合

1、ecmascript是一种语言标准，第一版标准发布于1997年，javascript是网景公司对ecmascript标准的一种实现。

2、js不区分整数和浮点数，统一用number表示，nan表示 not a number,无法计算结果的时候用nan表示。

3、字符串string,是以单引号或者双引号括起来的任意文本，比如'abc',"abc".

4、布尔值boolean,布尔值只有两种值，true或者false。

5、&&与，||或，！非。

6、编程形式的区别：面向对象、面向过程

7、typeof:六种数据类型：number string boolean undefined object function

typeof返回的是对象的两个object：null和{}

3.8

1、浏览器组成

* shell部分
* 内核部分
* 渲染引擎（html和css基础语法识别、语法规则和渲染）
* js引擎
* 其他模块

2008年谷歌发布最新浏览器chrome，能把js代码直接转化成机械码来执行。

2、js特点

js是解释性语言（浏览器读一行翻译一行，php/python也是解释性语言）

优点：跨平台、单线程（异步传输数据：）

不足：速度较慢

3、js标准

标准是由ecma制定的 所以有时候也叫ecmascript

延申两个部分dom bom

js三大部分：ecmascript、dom、bom

js执行队列：单线程，解释性语言，异步传输

4、主流浏览器的内核及私有前缀

ie 内核：trident 私有前缀：-ms-

chrome webkit/blink -webkit-

firefox gecko -moz-

opera presto -o-

safari webkit -webkit-

5、前端开发要求：结构（html）、行为(js)、样式(css)相分离

6、js变量（声明变量的只有var，浮点型）

变量命名规则：必须以英文字母、下划线和$开头；变量名可以包含英文字母、下划线、$和数字；不可以用特殊语法（关键字）和保留字作为变量名；





7、js基本语法

数据类型：

常见的数据类型：number string Boolean undefined null

引用值和原始值唯一的不同是赋值形式不同

* 原始值：number boolean（true是1，false是0） string(字符串) undefined（变量没有赋值） null（表示占位，值为空）属于object类型

存放地方不一样

原始值是存放在栈（stack）里面的

“先进后出”

引用值和原始值唯一的根本区别是赋值形式不同。

原始值是不可改变的，一个房间已经写了一个值，不可以改变。

删除数据的时候并不会删除数据在内存中的位置，只是让数据包的编号名字与该数据不对应，无从查找，但是依旧存在在内存中。只能二次覆盖。

栈内存之间的赋值是拷贝，互相不影响，定义的进去的值不可改变

var z = 10;

var x = z;

z = 20;

document.write(x);

//输出的x值是10；

* 引用值：array object function data regexp

引用值是存放在堆（heap）里面的

引用值拷贝引用值拷贝过去的是地址

var arr = [1,2];

var arr1 = arr;

arr.push(5);

document.write(arr1);

//输出值为 1 2 5

基本语法

* 一行结束必须添加分号；程序读取代码一行是识别分号，（函数function for循环 if之后不需要加 ；（分号） function for循环 if判断语句 ）
* 任何符号的两侧必须要有一个空格
* 错误分为两种

低级错误：

不可以写中文字符，语法解析错误

控制台会显示如下：

uncaught syntaxerror：

invalid or unexpected token：

逻辑错误：标准错误

8、js运算符

0/0 = NaN，0%0 = NaN

1/0 = infinity infinity是数字类型的 number

-1/0 = -infinity

* +：数字运算、字符串连接（隐式转换两边都是字符串）；

任何数据类型加字符串都等于字符串（隐式转换 + ）

* %：取余

var rr = 3;

rr %= 4;

document.write(rr);

//输出的值是3

// 调换aa bb的值

var aa = 123;

var bb = 234;

var temp = aa;

aa = bb;

bb = temp;

document.write(aa);

document.write("<br>");

document.write(bb);

Boolean：false和true

字符串比较的是asc码的顺序

* 逻辑运算符

&&（与）：(与 是碰到假就返回当前的值)

先看第一个表达式转换成布尔值结果，如果结果为真，那么它会看第二个表达式转换为布尔值的结果，然后如果只有两个表达式的话，那么看到第二个表达式，就可以返回该（第二个）表达式的值了。

var a = 1 && 2 + 2;返回的值为4，因为第一个先看var a = 1的值是否为真，在该例子中为真，则返回&&后面的（最后一个值）计算结果4；

返回结果为false的：undfined，null，NaN，“”，0，false。

var a = 0 && 2 + 2;返回的值为0，因为第一个是假，直接返回第一个值。

var a = undefined && 2 + 2；返回值为undefined。

&&升级：中断作用，短路语句：检查该语句是否存在错误或者能否执行。

data && function（data）；如果data返回false则表示data数据不存在或者错误。

||（或）：（碰到真就返回当前的值）

||或运算符经常用于实现解决浏览器的兼容问题

！（非）：

for循环的执行顺序：

*for* (*var* i = 0; i < 10; i ++)

1. var i = 0;
2. if(i < 10){} 条件判断
3. i ++
4. if (i < 10){}
5. i ++
6. if(i < 10){}

* 条件语句

if和else直接的语句必须是互斥的，满足条件1 一定不满足其余条件。

i++先使用i，再进行计算i = i + 1;

小练习：

**计算2的n次幂，n可输入，n为自然数：**

var n = parseint(window.prompt(“input number”));

//先定义一个mul的初始值

var mul = 1;

for(i = 1;i <= n; i++);{

//2的n次幂

//2的一次幂代表1乘以一个2

//2的二次幂代表1乘以2的值再乘以一个2

mul \*= 2;

}

document.write(mul);

计算n的阶乘，n可输入

var n = parseint(window.prompt(“input a number”));

var jiemul = 1;

for(var i = 1;i <= n; i ++)

{

jiemul \*= i;

}

document.write(jiemul);

输入a,b,c,三个数字，打印出最大的

//因为要输入三个数，所以要写三个parseint

var a = parseint(window.prompt(“input a”);

var b = parseint(window.prompt(“input b”);

var c = parseint(window.prompt(“input c”);

if(a > b){

if(a > c){

document.write(a);

}

else{

document.write(c);

}

}

else{

if(b > c){

document.write(b);

}

else{

document.write(c);

}}

条件语句

输入星期时，显示工作或者休息

var date = window.prompt('input');

switch (date) {

case "monday":

case "tuesday":

case "wednesday":

case "thursday":

case "friday":

console.log('working');

break;

case "saturday":

case "sunday":

console.log('relaxing');

break;

}

编程形式的区别：

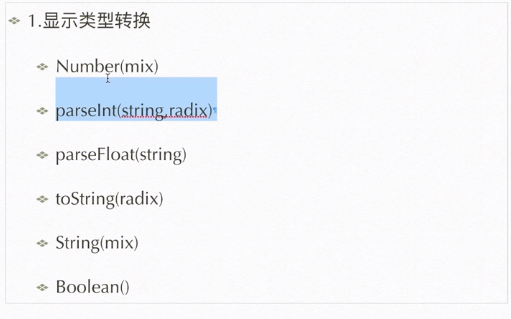
2、面向过程：

2、面向对象

typeof操作符：

可以返回6个值：number string Boolean object undefined function

显示类型转换：



Number：将输入的内容转化成数字

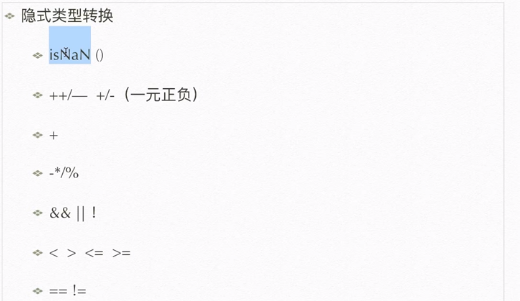
String：将输入的内容转化成字符串：

var demo ="123.3";

var num = String(demo);

console.log(typeof(num) + ":" + num);

隐式类型转换：



console.log(isNaN("abc"));

//输出true：隐式转化

console.log(isNaN(null));

//输出false

//false是0，true是1

类型转换

8、显示类型转化和隐式类型转换

number：将变量值转化成数字，null可以转换成0，true是1，false是0，但是一个是特殊的，undefined被转换成数字类型，会输出：nan.看起来不是数字的也是输出nan,如果是“123abc”也是被转换成NaN,不能转换成具体的数字。

parseint：把内容转换成整型的数，就是整数，输入123.98，会直接舍去，输出123.输入“123abc”输出“123”，p**arseint有个独特的功能，从数字位开始往后看，一直看到非数字位截止，把之前的数字返回**。

parsefloat：更parseint很相似，不同的是，float是浮点数就是小数，如果输入内容是：“123.45abc“，会输出123.45，它也是从数字位开始看，一直看到除了.之外的非数字位截止，后面的做截断，返回前面的值。

string：把内容转换成字符串，

boolean：把内容转换成布尔值，输入0 -0 null undefined nan false以及空字符串””会输出false,其余的均输出true。

tostring：想把谁转换成字符串就用谁.tostring,有两个特殊的，undefined和null不能使用tostring这个方法。

隐式类型转换：

isNaN()-->（调用的是Number）:把变量拿出来，先放到number里面去，转换的结果再与nan 比对，如果是NaN则打印出true，如果不是则输出false.

++/--、+（正） -（负）(调用的是number)：

-（减号）\*/% --》number（调用的也是number）

**+(加号)（调用的是string）**：有一个特殊的语法，当加号两侧有一个是string（字符串），就会调用string将两个全部转换成字符串。例如：var num = (1 + '2');输出的是12，因为+加号已经将1转换成字符串“1”,所以结果是12.

< > <= >=（转换成boolean） ：var num = 1 > "2";打印出false。

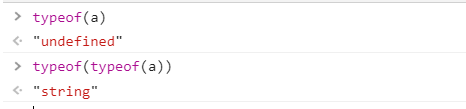
== ！=（调用的也是boolean）： undefined == null;打印true，undefine和null一样，> = < 0都会输出false。

不发生类型转换（左右两侧长得一模一样（类型和数字）才会绝对等于，除了NaN）

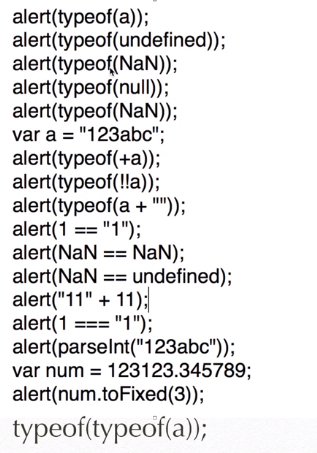
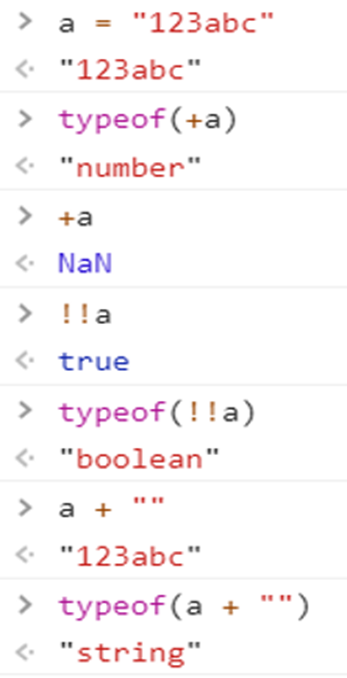
===：绝对等于

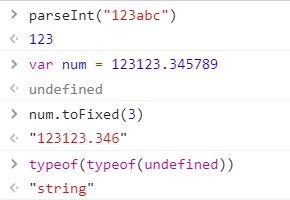
！==：绝对不等于

注意：typeof（a）返回的是字符串”undefined”,所以typeof打印出的结果类型是string。

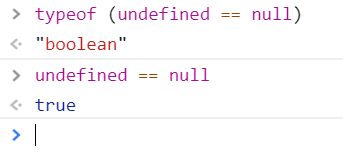


小练习：







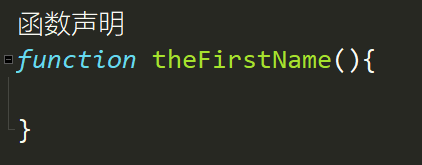
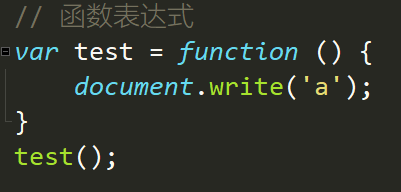


此处注意：NaN不等于NaN,但是undefined == null；

函数：

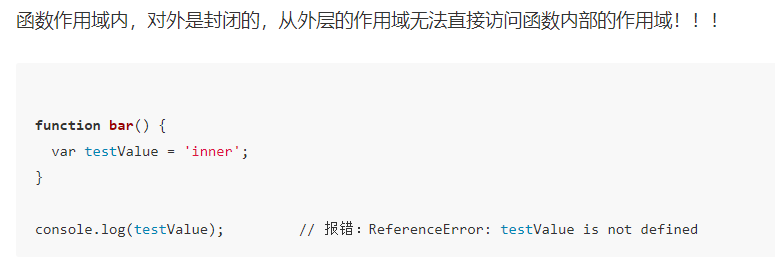
1. 定义：function 函数名（）{}函数名和变量名格式差不多，

但是命名必须符合小驼峰原则：单词首字母小写，后面单词的首字母必须大写。

1. 
2. 
3. 函数组成形式：

函数名称、参数（形参、实参arguments）、返回值。不定参数、

1. 函数的作用域：（函数里面的参数，就相当于函数内部已经定义好了的变量）



Js（单线程、解释性语言）运行三部曲:

语法分析、预编译、解释执行

arguments.length 表示实参的个数，函数名.lenth表示形参的个数。

js预编译：

函数声明整体提升

变量 声明提升

imply global：暗示全局变量，

全局对象：window

window就是全局的域：

var a = 123；

等于先声明var a;再a = 123；

console.log(a);

window相当于仓库，是一个对象和GO（global object）一样的功能。

任何全局变量都是window的属性

预编译：一切声明的全局变量（注意不包括局部变量），全是window的属性。

window 就是全局，

预编译发生在函数执行的前一刻：

预编译的步骤：

第一步：创建AO （activation object）对象。执行期上下文，就是理解的作用域

第二步：找函数的形参和变量声明，将变量声明的名作为AO对象数属性名，值为undefined

第三步：将实参值和形参值统一

第四步：在函数体里面找函数声明（定义一个变量等于函数不算函数声明(var a = function(){})，函数声明必须有函数名才可以），值赋予函数体

预编译声明的变量或者函数输出的时候，如果有自己的AO值，就打印输出自己拥有的AO的值，而不必要打印输出GO就是window的值。

作用域:[[scope]];存储了运行期上下文的集合

作用域链：scope所存储的执行期上下文对象的集合

查找变量：从作用域链的顶端依次向下查找。

补充一点:在哪个函数里面查找变量，就取函数的作用域链的顶端依次向下查找。

闭包：

如果内部的函数被保存到外部，它一定生成闭包。

缺点：闭包会导致原有作用域链不释放，造成内存泄漏

内存泄漏：内存占用多了，可利用的内存就少了。

闭包的作用

实现公有变量：函数累加器

可以做缓存：存储结构

可以实现封装，属性私有化

模块化开发，防止污染全局变量

闭包：内部函数在外部被调用的时候产生闭包，只有闭包才可以是实现在外部调用和改变内部函数的属性