# 一，啥是JavaScript

## 1，JS介绍

JavaScript，简称为JS。

html是超文本标记语言，css样式语言，JS编程语言。

**和html，css的共同点：**

都是在浏览器中运行的。

**和html，css的不同点：**

Html是标签语言，css是样式语言，js它是编程语言。

历史：

<http://www.ruanyifeng.com/blog/2011/06/birth_of_javascript.html>

在历史中，JAVA和JS是没有一点关系。 类似于北大与北大青鸟。

**JS能做什么？**

1，写交互

2，写游戏

3，node.js

## 2，如何学习JS

**内容：**

有三部分组成：

1，ECMAScript: 是JS的一个语法标准。这几天学习的基础语法都是属性ES的内容。

2，DOM（Document Object Model）文档对象模型。

3，BOM（Brower Object Mode）浏览器对象模型。

**版本：**

95年JS诞生了，97年，ECMAScript成立，99，ES3出现，09，ES5出现，15，ES2015出现，简称为ES6。

**心态：**  
 学习HTML+CSS，它是立即可以看到效果，但是JS是一门编程语言，刚开始学的时 间，比较枯燥，需要坚持，

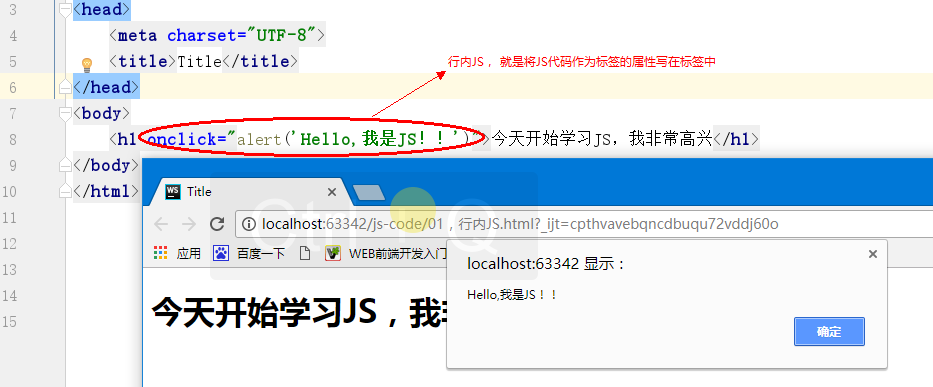
**参考书**：JS高级程序设计，JS权威指南

## 3，页面引入JS

JS类似于CSS，写法也有三种：

* 行内的
* 内部的
* 外部的

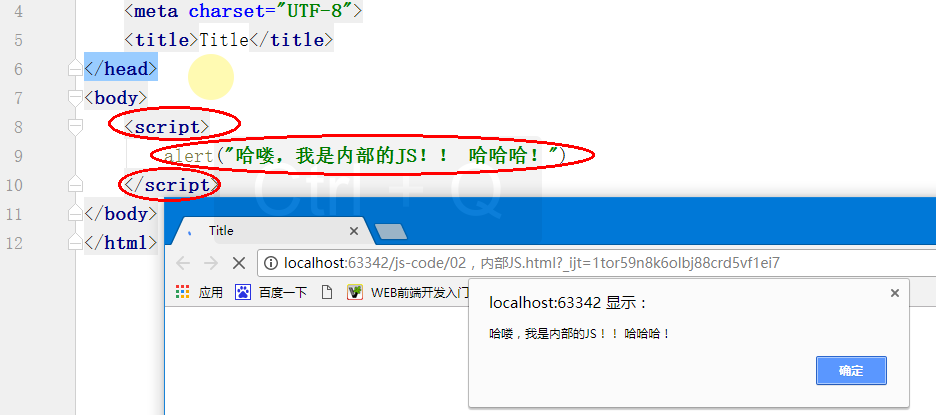
**行内：**



注意细节：JS代码是作为开始标签的属性。

**内部：**

需要通过script标签，将JS代码写在标签中，如下：



注意细节：

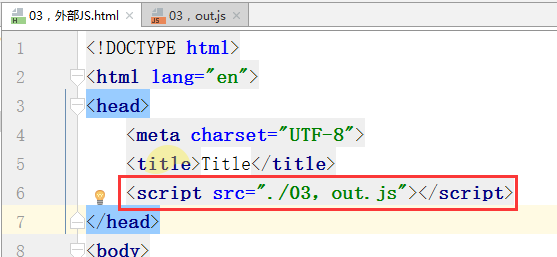
1，script可以放在任何地方

2，script标签中有一个type属性，属性值为text/javascript，是默认的，可以不写

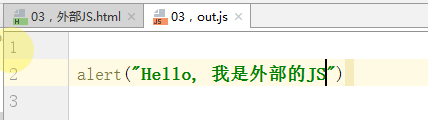
3，JS代码需要放在sciprt标签中， 如果不放，则运行不了

**外部：**

通过script标签中的src属性来引入外部的。引入外部的css是通过link标签。

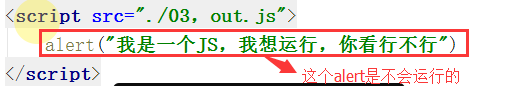


外部的JS， 如下：

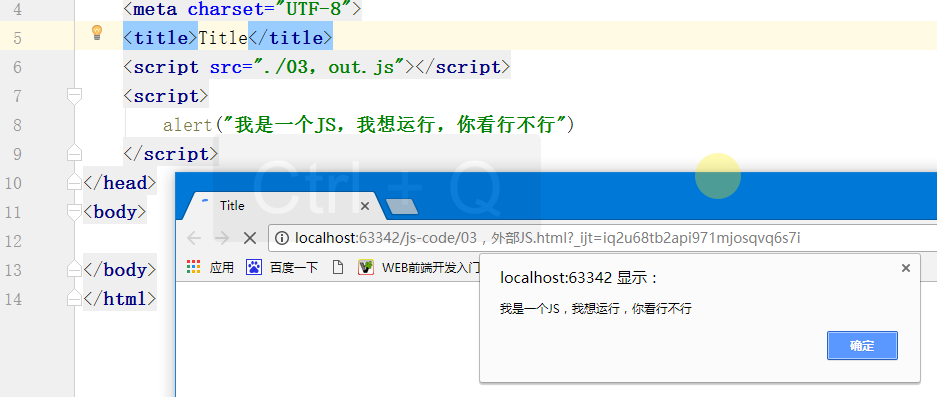


注意细节：

1，外部的，也是通过scirpt标签中的src来引入的。在这两个script标签中间是不能再写其它JS代码的。



2，对于scirpt标签，可以有多个。



**最好习惯：**学习的时候，用内部JS，做项目时，用外部JS。

当用内部JS时，将scirpt标签放在最后面。

# 二，JS的基本语法

## 1，变量

### 啥是变量

电脑（计算机）产生的一个目的是？ 刚开始就是纯粹的计算。

现在我要计算一个数学问题：1+2=？

当我们去计算时，一定要找两个容器来放数字，一个是放1， 一个放2。当计算是地，先去这两个容器中取出值，然后进行运算。

在编程中，这个容器（菜篮子），我们叫它变量。

**一个变量有两部组成：**

变量名：一般来说，在赋值运算符的左边

变量值：一般来说，在赋值运算符的右边

如：

var a = 110; = 在数学中，叫等于符号 ，在编程中叫赋值符号。

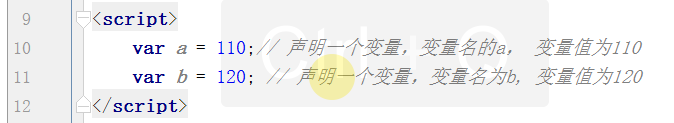
a叫变量名， 110叫变量值。

### 如何声明变量

暂时把声明理解成创造。

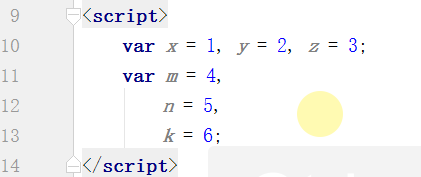
通过一个关键字来声明，var。 var是variable（变化）的缩写

**第一种：单独声明**



**第二种：一次性声明多个**

**只使用一个var， 声明多个变量**



**第三种：重复声明**

**多个变量名是一样的，在js中， 是允许重复声明的，其它有些编程语言中是不允许的。**



注意细节：

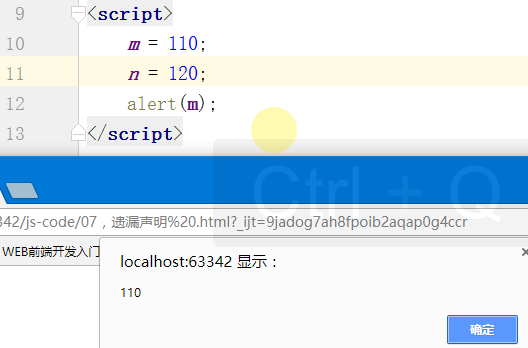
1，重复声明虽然在JS中是允许的，但在开发中，不要这样用。

2，在JS，可以通过alert();来弹出一个变量的值。



**第四种：遗漏声明**

**在JS中，也是允许，但在开发中， 也是尽量避免的。**



**复习：**

**1，什么是JS？**

JS是一门运行在浏览器端的脚本语言。

常见的编程语言：C，C++，JAVA，PHP.....

编程语言都有几个特性：变量，数据类型，运算符，表达式，函数，数组....

JS（JavaScript）和JAVA是没有关系的。

**2，引用JS的三种方式**

行内写法

将JS代码写在标签中 如: <h1 onclick = “alert()”></h1>

内部写法

将JS代码写在<script></script>标签中，这个标签可以放在页面中任何位置。

外部写法

将JS代码单独写一个JS文件，然后通过script标签中的src属性来引入。

**3，什么是变量？如何声明一个变量？**

所谓的变量它是一个专业术语，是编程中的。目前就可以理解成一个容器，它里面是用来放东西的。如var a = 110; 把110这个东西放到了a的个容器的。

声明变量就是使用var这个关键字，如var b = 120;

关键字是就是发明JS这个人，事先起好的名字。不能这样写：var var = 130;

### JS中的词法结构

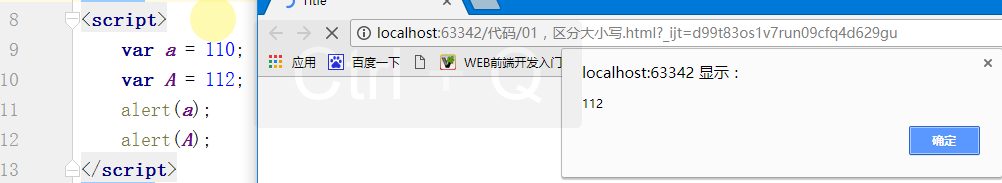
理解与JS的一些基础规则，它用来表示JS代码如何写，有如下几个规则：

* 1，区分大小写
* 2，忽略空白符
* 3，标识符和关键字
* 4，注释

**1，区分大小写**

在写JS代码时，大小和小写是不一样的。如var a = 110; 和var A = 110;

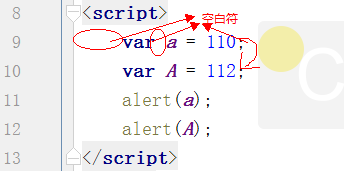
代码如下：



**2，忽略空白符**

如空格，换行符，制表符（tab）等，在JS中是被忽略的。

如：



**3，标识符，关键字，保留字**

所谓的标识符，就是一些名字。如变量名，函数名等。

所谓的关键字，就是发明JS时，已经被占用的名字。

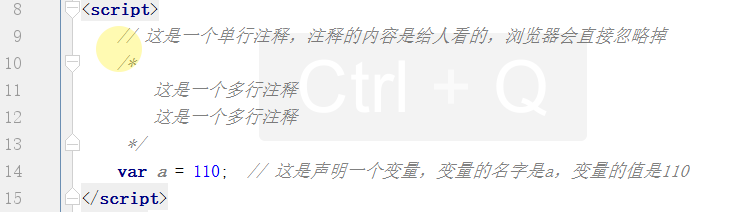
所谓的保留字，就是发明JS时，为以后保留的一些名字。

**4，注释**

单行注释： //

双行注释： /\* \*/

如：

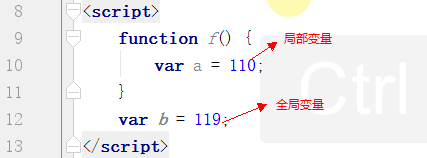


### 全局变量与局部变量

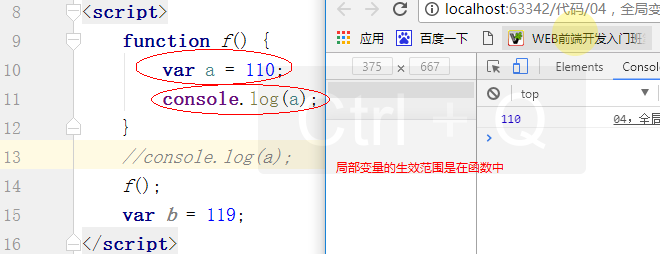
全局和局部是指变量的**生效范围**。类比于普通话与方言。

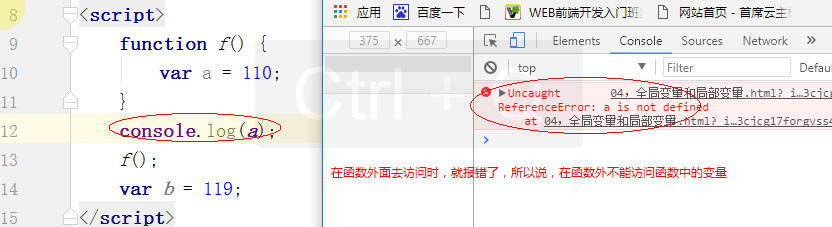
函数：是数学中的概念。在计算中，借鉴了数学中的函数的概念。目前，函数就是一个功能块。函数是也需要声明的。声明一个函数写法：funciton f(){}

局部变量是声明在函数中的变量，全局变量是声明在函数外面的变量，如下：

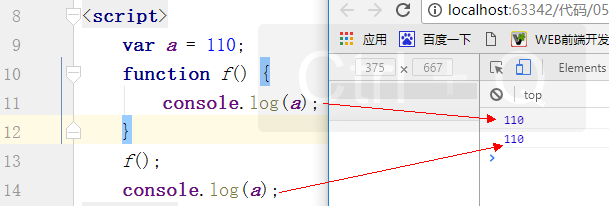


局部变量的生效范围是函数内，如下：





全局变量的生效范围是函数内，如下：



**总结：**

全局变量是指在函数外面的变量，它的生效范围是函数内外。类似普通话。

局部变量是指在函数内的变量，它的生效范围是函数内。类似方言。

## 2，数据类型

### 啥是数据类型

变量就是一个容器，现实中有各种各样的容器。变量也是放东西，如放一个110，是一个数字，放“HTML5”它是一个字符串，放一个真，它就是一个布尔值 ...

数据也是有类型。或者说，变量也是有不同的类型的，就叫数据类型。不同的变量可能有不同的数据类型。它是用来描述变量的。

**在JS中，有五大基本数据类型，还有引用数据类型。**

**五大基本数据类型：**

* 数值型（number）
* 字符串（string）
* 布尔型（boolean）
* undefined
* Nulll

**引用类型：**

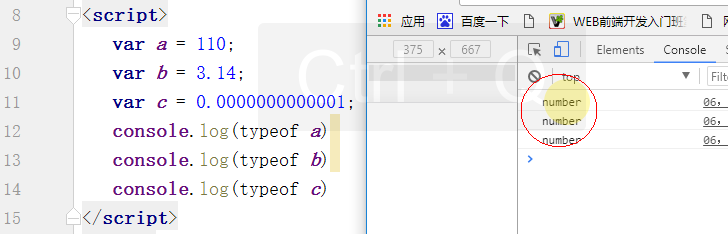
* 对象（Object）

### number

数值类型，作用是用来表示数字，如商品价格，年份，年龄....

可以通过typeof来查看一个变量的数据类型。

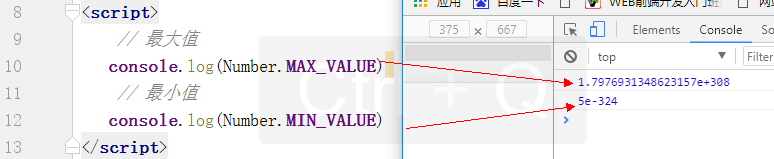
代码如下：



**注意细节：**

1，在有的编程语言中，数字分整数和浮点数。在JS中不分，统一叫number。

2，JS中的number有最大值和最小值，如下：

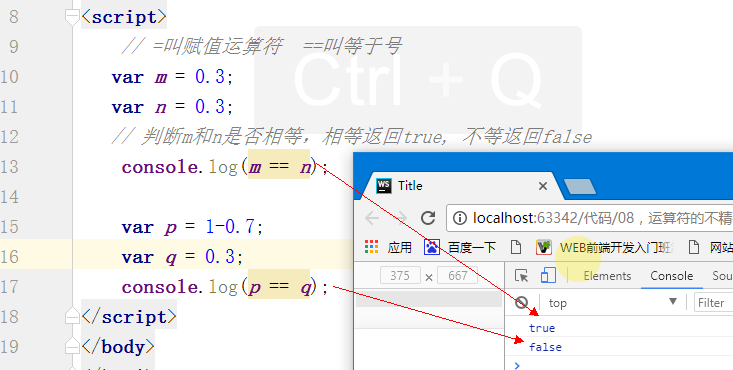


3，number分多种形式表示，有十进制，八进制，十六进制。如：



4，小数也有多种表示，如 3.14 .333(原本是0.333，如果是0可以省略) 6.02e3

5，运算不精确，如下： 在JS中不要尝试去比较两个小数。真要比较的话，先把们转化成整数。

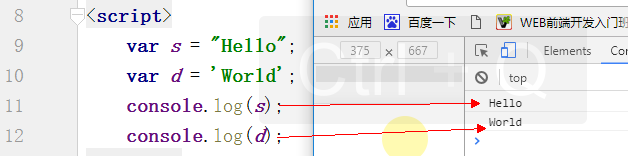


### string

作用：用来描述字符串相关的数据，如姓名，密码，用户名，地址...

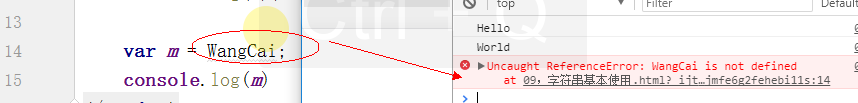
格式：”wangCai” ‘xiaoQiang’ 对于字符串，要使用引号包起来。

基本使用：

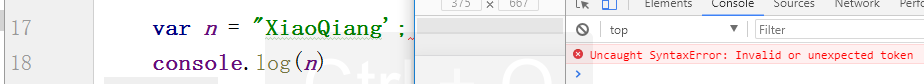


注意细节：

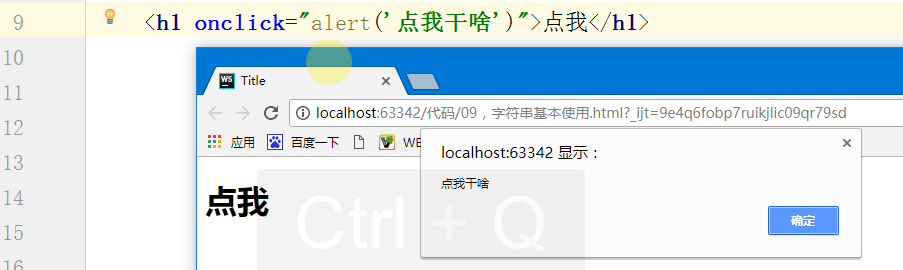
1，字符串，一定要加引号，不加的话，它就会它看作变量，如下：



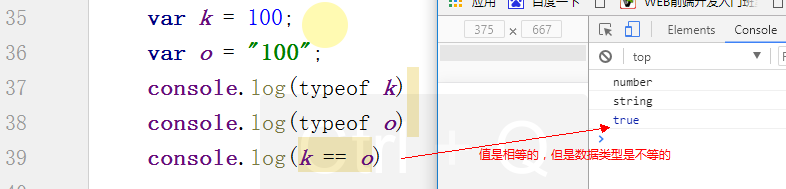
2，单引号和双引号不能混用，如下：



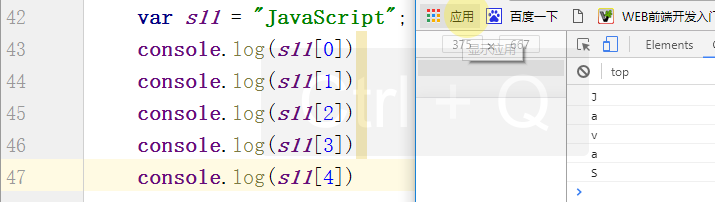
3，当在一个字符串，还要用一个引号，怎么办，如下: 外单内双或外双内单



4，数字可以作数值，也可以作字符串，如下：



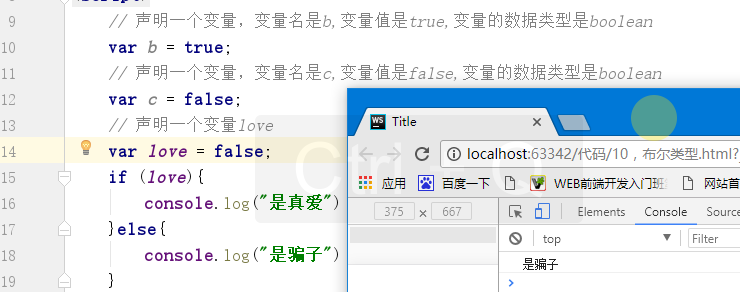
5，字符串，可以以数组形式来使用，如下：



### boolean

前面，数值类型，它对应的值有无数。字符串，它对应的值也有无数个。但是拉着要学习的boolean类型，它只有两个值，一个是true一个是false。

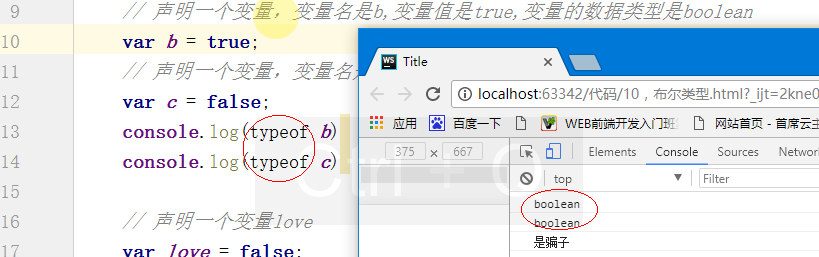
基本使用：



注意细节：

1，区分大小写，true和True和TRUE是不一样。

2，可以使用typeof来获取数据类型，如下：



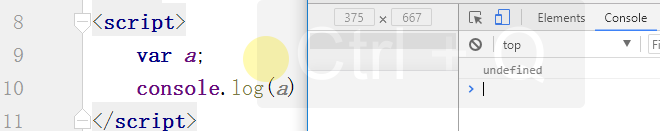
3，下面几个值会转化成false，其它都会转化成ture.

* undedined
* null
* 0 / -0
* NaN
* “”

### undefined和null

un是前缀 defined是定义的意思 undefined就是没有定义的意思

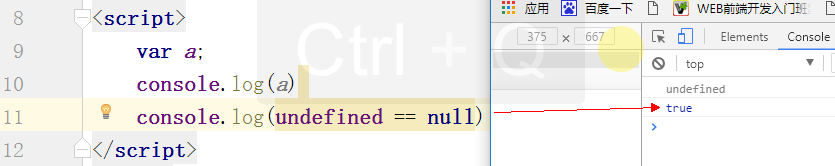
当一个变量没有赋值初值时，会出现undefined，它是not defined是不一样的，如下：



Undefined是一个数据类型，也是一个值。对比，var a = 110; a的数据类型是number它的值是110。Var b; b的数据类型是undefined，它的值也是undefined。所以说，undefined这种数据类型只有一个值，这个值也是undefined。

Null

本意就是空的意思。和undefintd还是有点区别，undefind是用来表示基本数据类型的，null是用来表示引用数据类型，它两个都是表示不存在。当比较它俩时，它俩是相等的。如下：



### 引用类型

引用数据类型说白了就是对象， 对象叫Object。

在JS中有两个特殊的对象：

* 数组 array
* 函数 funciton

除了上面的两个，可以把对象分成三类：

* 内置对象 ，ECMAScript中提供对象
* 宿主对象 ，DOM和BOM提供的对象
* 自定义对象

代码如下：



关于引用数据类型，暂时了解就OK， 就业班会当作重点难点来讲解。

## 3，运算符

### 啥是运算符

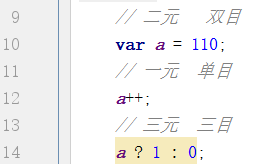
用来表示某种功能的符号，运算符在执行代码运算时，会针对一个以上的操作数进行运算。如: 1 + 2 +就是运算符，计算机在运算时，会将左操作数和右操作数进行相加的操作。

在JS中，有很多运算符，分类如下：

* 算术运算符：+ - \* / % **++ --**
* 赋值运算符：= += -= \*= /= %=
* 位运算符 略
* 关系运算符：== === != !== > < <= >=
* 逻辑运算符：&& || !
* 其它运算符：typeof ?: delete in +

根据操作娄分类：根据操作数可以分为一元运算符，二元运算符，三元运算符

如下：

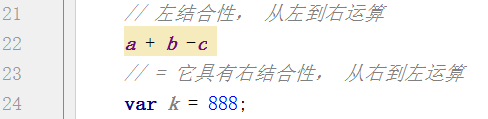


优先级：当多个运算符在一起时，优先算哪一个，如下：



如果不知道，哪一个运算符的优先高，可以另（），改变优先级。

结合性：左结合 ，右结合， 如下：



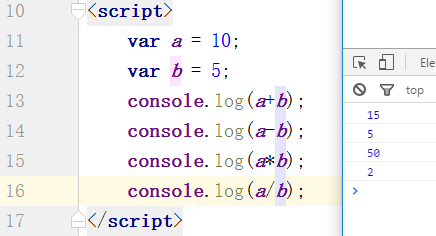
左值：了解 在赋值运算符左边的，就是左值。

### 算术运算符

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

**+ - \* /**

代码如下：

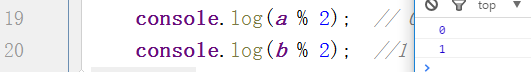


\* / 的优先级是高于+ -

如果优先级相同，具有左结合性。

**%**

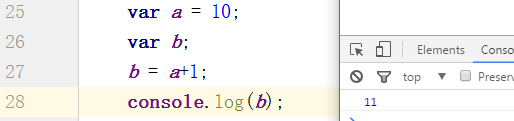
取模，取余



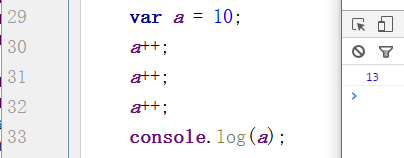
**++ 分为前置++ 和 后置++**

表示自增

首先，看一个需求，需要将一个数加1，然后赋给其它地方使用：

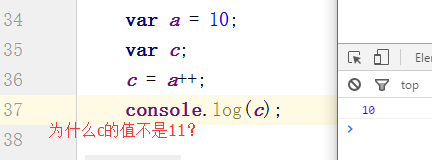


在js中有很多类似这个的情况，所以js提供了一个++的运算符，专门用来实现+1的操作，如下：

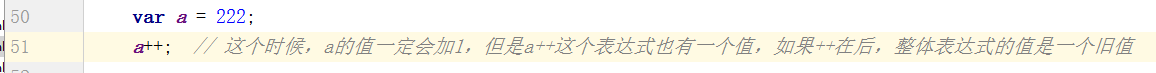


对于++，它只需要一个操作数，所以它是一元运算符，也叫单目运算符。

但是++并没有那么简单：

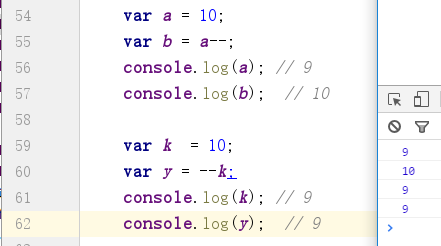


++在前，整体表达式的值是一个新值，++在后，整体表达式的值是一个旧值



**--**

表示自减, 和++类似， 道理不多说了。



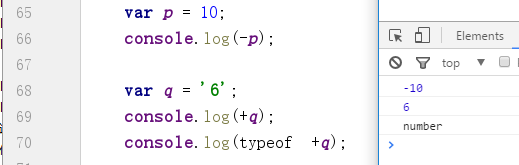
在实际开发中，++和--用的非常多，尤其在循环中用的最多。

最后补充：

+ 和 - 除了可以作为二元运算外，还可以作为一元运算符。

-作为一元运算符时，表示取反操作

+作为一元运算符时，转换成数字



对于一元的+和-，用的不多，但是在某些场合下，也可以使用。

### 赋值运算符

在js中，赋值运算符，用=。

大家在读代码时，尽量读作赋值操作，不要读成等于操作

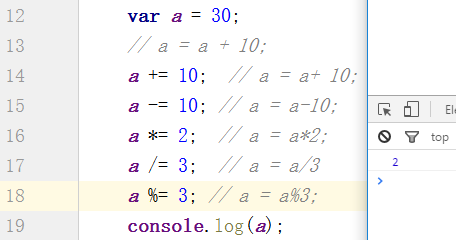


注意：等于用==表示，=表示赋值操作

赋值运算符是一个二元运算符，需要有两个操作数，赋值运算符的优先级非常低，基本上是最低的。

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

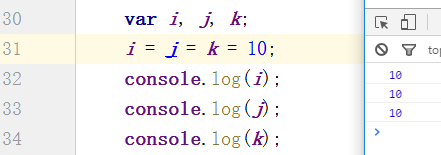
解释一下，+=，-=等是什么意思 ？



重点：

* =具备右结合性
* =的优先级比较低
* =和==的区别
* +=可以用于数字或字符串，视情况而定

**解释一下，i=j=k=10?**



有两层含义：

其一，赋值运算符具有右结合性，首先就是将10赋给了k,这个时候k就等于了10;

其二，赋值运算符有副作用，副作用是这个赋值的表达式本身也具备一个值，也就是意味着，k = 10， 这个赋值表达式，本身也有值，这个值就是10

代码的理解相当于：

第一步：i=j=k=10,此时k的值为10， 将k=10整个替换成10

第二步：i=j=10,此时j的值是10，将j=10整个替换成10

第三步：i=10

### 关系运算符

作用：用来进行比较的。比较的结果通常是布尔值。

== ===

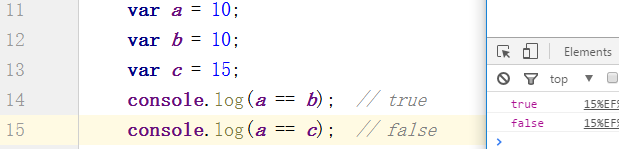
!= !==

> >=

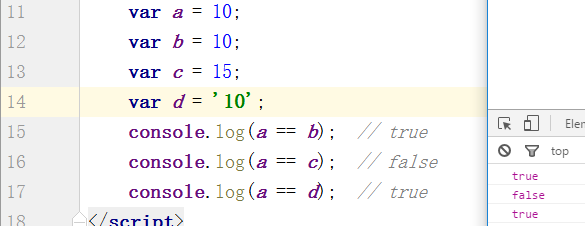
< <=

== 相等运算符，使用两个==表示

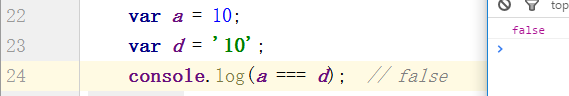
要判断两个变量的值，是否相等，如果相等，返回真，否则返回假



在使用==运算符时，是不需要考虑类型的，只要两个变量的值是相等的，就返回true:



如果我们想判断两个变量的值相等，数据类型也相等，这个时候，我们就可以使用全等：===

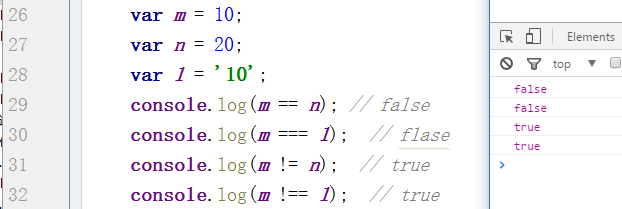


小结：

* =表示赋值运算符，右结合性
* ==表示相等运算符，只判断两个变量的值是否相等
* ===表示全等运算符，不只判断两个变量的值是否相等，还判断两个变量的数据类型是否相等

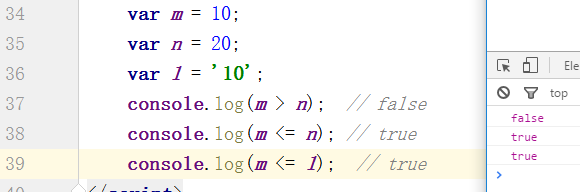
!= 和 !==

!=是==反操作 !== 是===反操作



在实际开发中，== === !=使用比较多， !==使用比较少。

> >= < <=和数学比较是类似的。



注意细节：

1，使用==和!=，可以对所有的类型进行比较

2，但是>, >=,< <=的时候，只是针对数值进行比较。如果要比较字串要使用函数

### 逻辑运算符

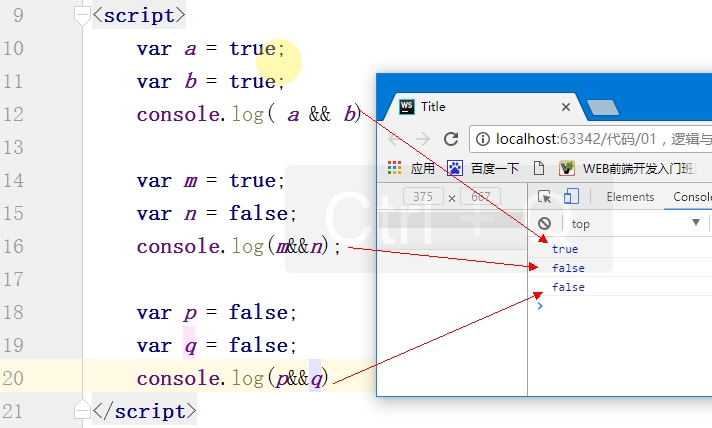
在JS中有三个逻辑运算符：

* && 逻辑与
* || 逻辑或
* ! 逻辑非

**逻辑与 （&&）**

表示进行与运算，是并且有意思，或AND，当两个条件都为真，整体结果才为真。

以两个变量为例，如下：



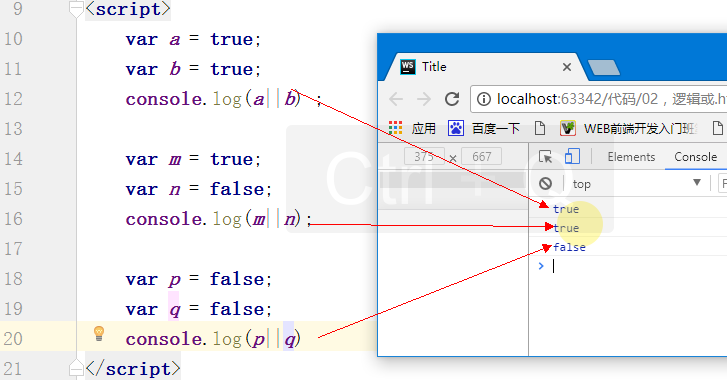
结论：

|  |  |  |
| --- | --- | --- |
| 表达式1 | 表达式2 | 结果 |
| true | true | true |
| true | false | false |
| false | ture | false |
| false | false | false |

**逻辑或（||）**

表示进行一个或运算，或OR，只要有一个条件为真，那么结果就为真。

代码如下：

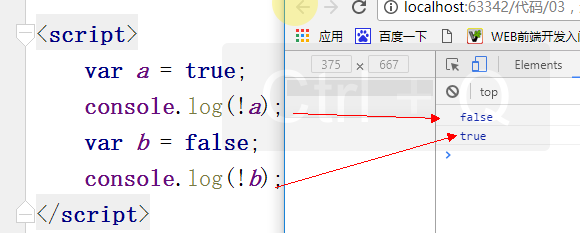


结论：

|  |  |  |
| --- | --- | --- |
| 表达式1 | 表达式2 | 结果 |
| true | true | true |
| true | false | true |
| false | ture | true |
| false | false | false |

**逻辑非（！）**

表示取反操作 NOT 单目运算符，是指只有一个操作数， 代码如下：



结论：

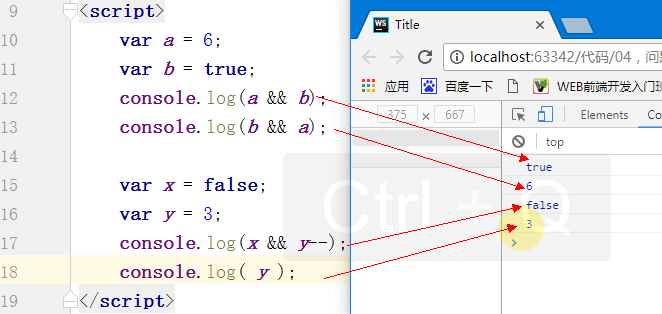
|  |  |
| --- | --- |
| 表达式 | 结果 |
| true | false |
| false | true |

**注意细节：**

1，一元的运算是比二元运算符的优先级要高。

**深入逻辑运算符（经常出现在笔试中）**

问题如下：



总结知识点如下，以**&&**为例：

**第一个方面：**如果操作都是布尔类型，只有两个都为true时，结果为才true。

**第二个方面：**&&可以对真值和假值进行运算，如果两个都是真值，则返回一个真值，否则返回一个假值，真值并不仅仅限于true，假值也是仅仅限出false，下面这些值会转换成false，其它值转化成true:

* Undefined
* Null
* 0/-0
* NaN
* “”

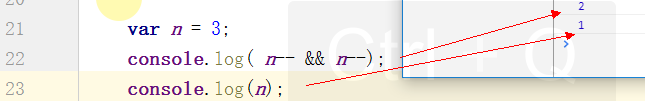
在js中，进行&&操作时，结果并不总是true和false,而是当前值。当前的值可以是number,也可以是string。

如果第一个表达式的值为真，那么就以第二个表达式的值作为结果，这个结果不是转化后的值，而是这个值本身。

如果，这个结果也不是转化后的值，而这这个值的本身第一个表达式的值为假，那么就以第一个表达式的值作为结果。

**第三个方面：**运算过程中，首先计算左操作数的值，如果计算结果就是假值，则整个表达式的值就为假数。也就是左操作数的值。同时停止对右操作数的运算。如果说左操作数结果为真值，则整个结果依赖于右操作数的值。

测试：



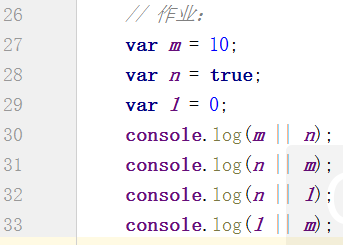
n = 3 n-- n的值变为2 2是真 此时就第二个表达式的值作为结果 第二个表达式又是一个n-- n-- 整体表达式的值是一个旧值 说明 n--这个整体的值是2 n的值还是要-1的，所以说，n的值最后是1

对于**||**来说，和&&也是相通的：

如果第一表达式的值是真值，结果就是第一个表达式的值，这个值不是转化之后的值，而这个值本身。

如果第一个表达式的值是假值，结果就是第二个表达式的值，这个也不是转化后的值，而是这个值本身。

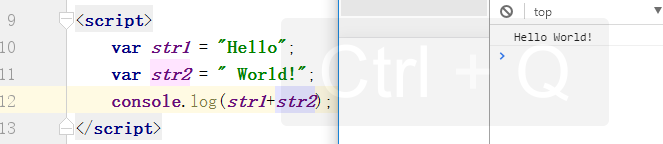
作业：



### 其它运算符

* 字符串连接运算符 +
* 三元运算符 ？：
* 获取类型运算符 typeof

**字符串连接运算符 +** 代码如下：



说明 ：

1，字符串连接运算符是一个二元运算符，它还有一个功能，进行算术运算。

2，如果在+两边都是字符串，它的意思就是将这两个字符串连接起来，如果说是两个数字，它的意思是进行一个求和运算，那么如果一个是字符串，一个是数字，它的意思也是进行字符串连接操作，如下：

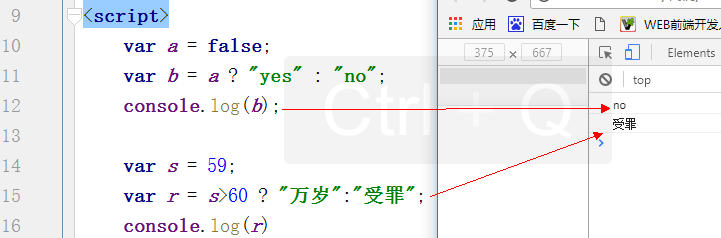


**三元运算符 ？：**

格式：表达式1 ？ 表达式2 ： 表达式3

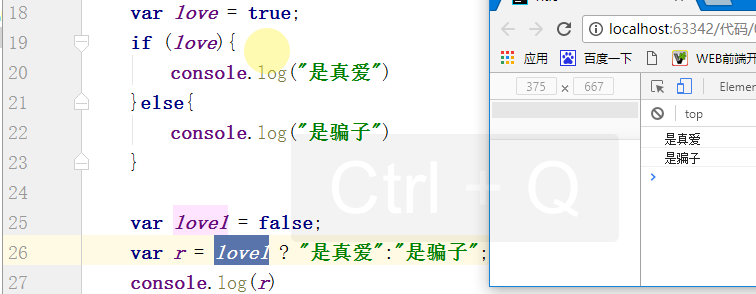
规则：如果表达式1的值为真，那么这个整体的结果就是表达式2，否则就是表达式3

代码如下：

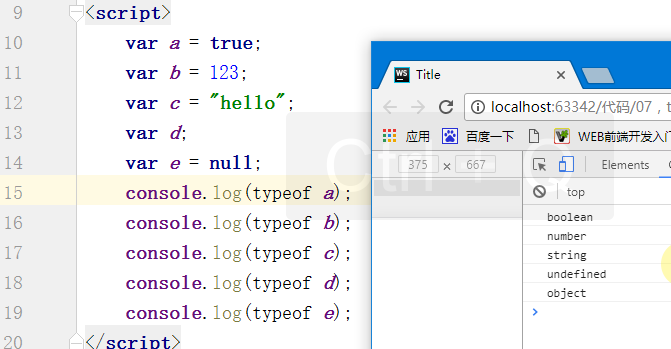


说明 ：

1，有点类似于if .. else .. 如：

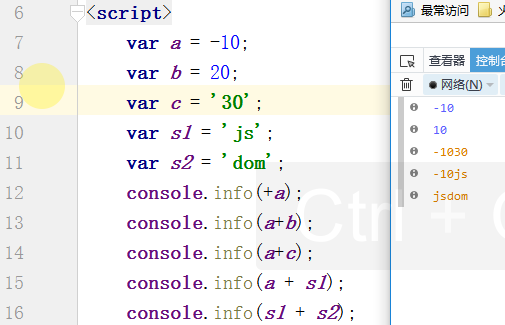


**获取类型运算符 typeof 代码如下：**



### 类型转化

思考一个问题：



在JS中，有一个严格的规则：运算符在运算时，一定要保证运算符两则的操作数的数据类型是完全一样的。

在上面的案例中，console.info(a+c), a是number，c是string，+它两边的操作数的数据类型是不一样，这个时候，JS就会进行一个数据类型的转化。

关于类型转换，有两种方式：

* 强制转换
* 隐式转换

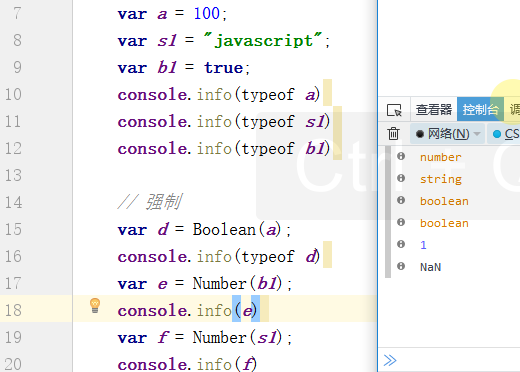
这里的转换一般是指基本数据类型的数据。

**强制类型转换：**

使用Boolean(), Number(),String(),Object()函数。

在转化数据时，还可以使用parseInt()函数，parseFloat()函数。

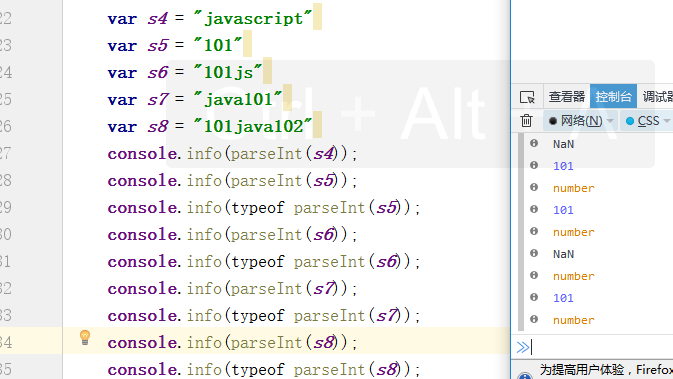
代码如下：

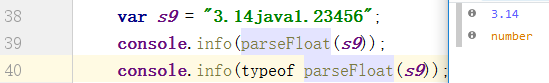


如果我们只对数字进行转换，可以使用parseInt()函数，parseFloat()函数:

parseInt()函数 转化成整数

parseFloat()函数 转化成小数





Number和parseInt之间的区别？

parseInt可以将数字开头的字符串转成数值。转换规则就是将数字开头连续的数字提取出来，直到碰到字符为止，后面的都会被截掉。

**隐式类型转换**

让变量参与运算，一般都会有隐式的类型转换。只不过一般我们察觉不到。

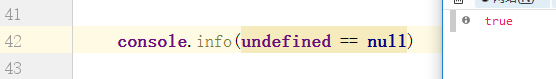
**哪些地方，会有隐式的类型转换**：

1，如果+运算符一个操作数是字符串，另一个是数值，它会把数值转换成字符，目的就是保证运算符两边的操作数的数据类型一样。

2，！运算符将操作数转换成布尔类型然后再取反。

3，对于数值，如果转换没有意义，则返回NaN Not a Number

**有一个特例：**==当比较undefined和null时，它不是把undefined转换成null,也不是把null转换成undefined。而是它两个都转换成了false。



对于隐式转换，下面的五个值会被转换成false，其它的值都会转换成true:

* undefined
* null
* 0/-0
* naN
* “”

## 4，表达式

### 啥是表达式

**任何有值的内容都是表达式。**

**常见的表达式有如下几个：**

* 原始表达式
* 运算符表达式
* 对象和数组初始化表达式
* 函数定义表达式
* 调用表达式
* 对象创建表达式

### 原始表达式

原始表达式是表达式的最小单位，它不能再包含其它表达式。

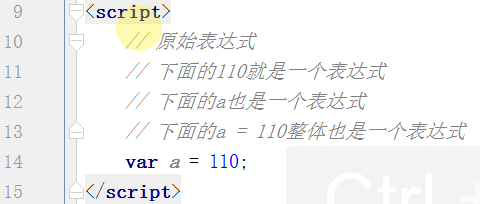
常见的原始表达式有如下几个：

直接量 如：110 “hello”

关键字 如：true false null this

变量 如：a， b， .......

代码：

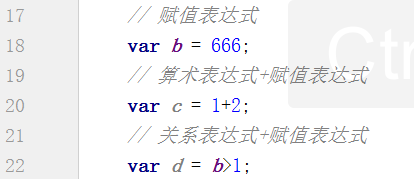


### 运算符表达式

运算符有很多。常见由运算符构成的表达式有：

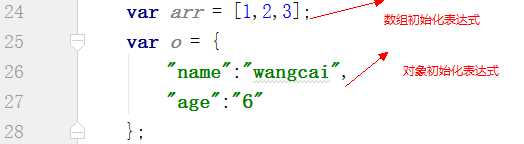
* 算术表达式
* 关系表达式
* 逻辑表达式
* 赋值表达式

代码：

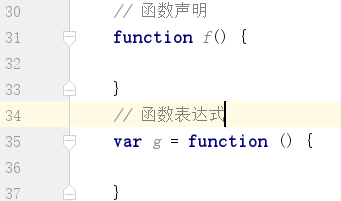


### 对象和数组初始化表达式

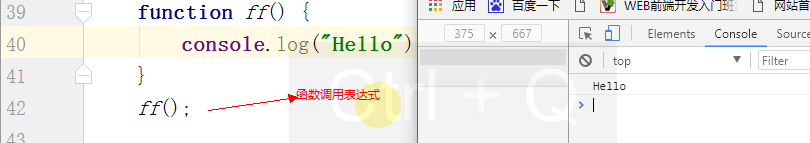
对象和数组目前还没有讲， 先了解就OK：



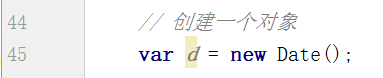
### 函数定义表达式



### 调用表达式



### 对象创建表达式



# 三，语句

## 1，啥是语句

语句：要js中，表达式只是短语，语句就是整句或命令。

作用：用来执行某种指令。

特征：以分号结尾。

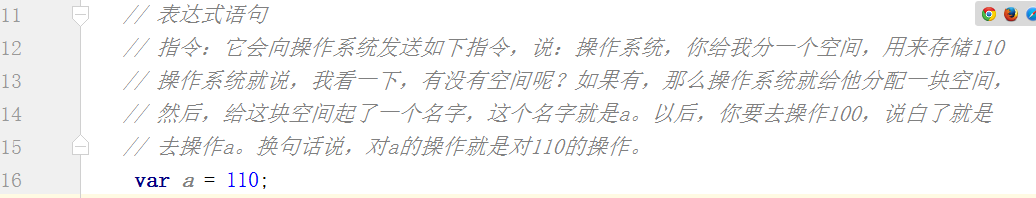
**分类：**

* 表达式语句
* 声明语句
* **条件语句**
* **循环语句**
* **跳转语句**

## 2，表达式语句

在表达式后面加上一个分号，那么整体就是一个表达式。

代码如下：

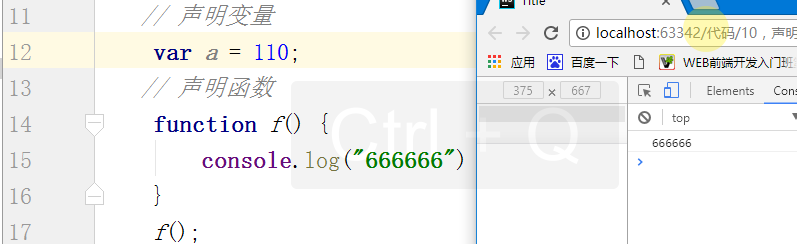


## 3，声明语句

JS中有两个声明：

* 声明变量 使用var来声明
* 声明函数 使用function来声明

代码如下：

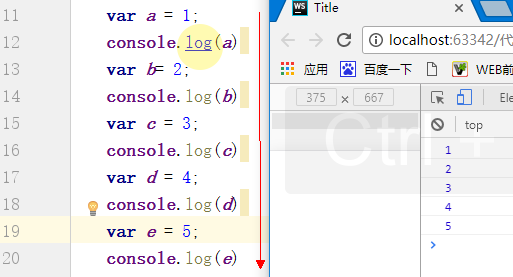


## 4，条件语句

世间万物都脱离不了三种结构：

* 顺序结构
* 选择结构
* 循环结构

计算机产出的目的也是为了解决现实中的问题，所以计算机编程语言中也有对应的语法来完成这三种结构。先看一个顺序结构。是指代码从上到下依次执行。如下：



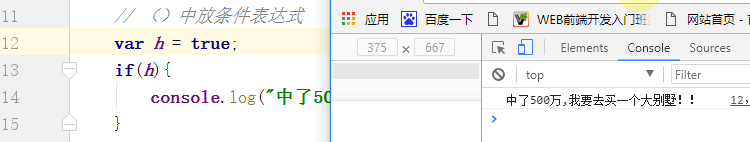
如果只有顺序结构，解决不了太多问题，所以说选择和循环结构就产生了。

对于选择结构，我们是使用选择语句来实现的。在JS中选择语句有四4：

* If
* If...else
* If...else if ...else
* Switch

### if语句

代码如下：

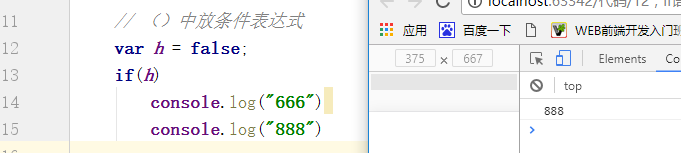


注意细节：

1，关键字是if，是如果的意思， 在if后面有一个()，()与if之间可以有空格也可以没有。

2，在()中，可以放各种表达式，可以是原始表达式，也可以其它表达式。

3，{}可有可无，如果没有的话，它只会执行if后面的一条语句。如果有的话，它会执行{}中的所有语句。代码：



在写代码时，尽量加上{}, 不然有歧义。

### if...else...语句

双分支，二选一。

代码：



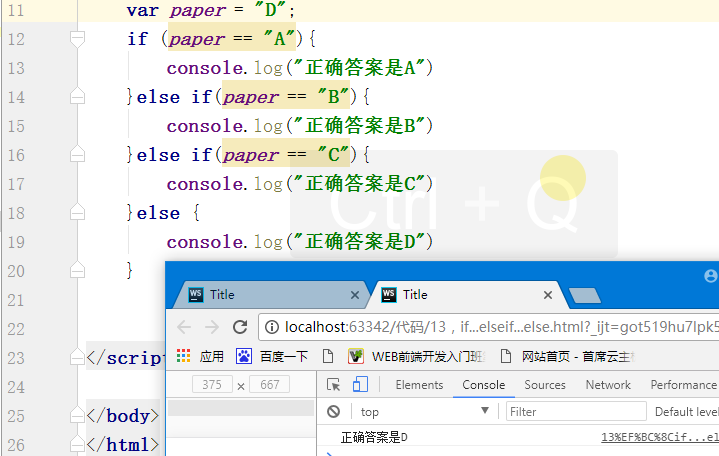
注意细节：

1，对于if...else...一般可以使用三元运算符进行简化。

### if...else if...else...语句

多分支，多选一。

代码如下：

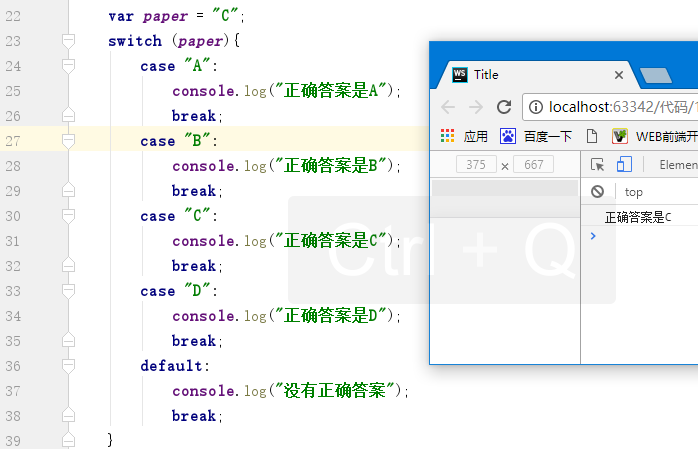


**注意细节：**  
 1，else if中间要有空格，不能挨着。

### （4）switch语句

和if...else if ... else一样， 也是多分支。

代码如下：

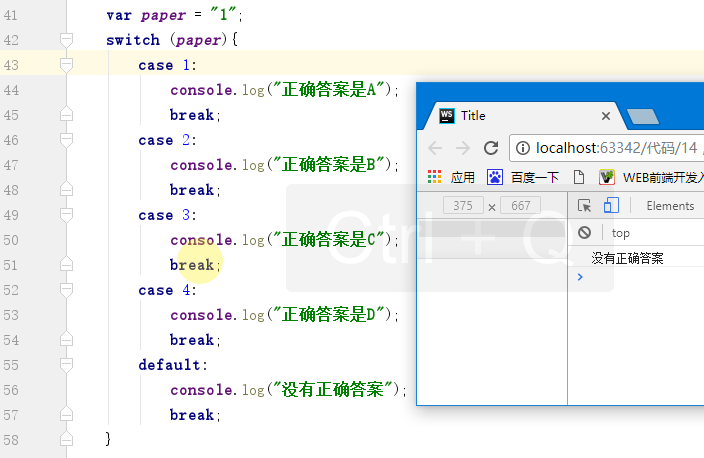


注意细节：

1，对于每个case，后面都要加break，如果不加，当一个条件满足时，后面的所有程序都会执行。

2，在switch后面，一般都加上一个default，表示默认情况。

3，对于switch中的判断，它使用的=== 不是== ， 所以说，在写sitch，一定要注意数据类型，如下：



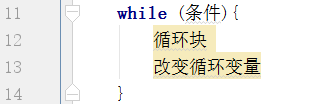
## 5，循环语句

循环语句有下面几种：

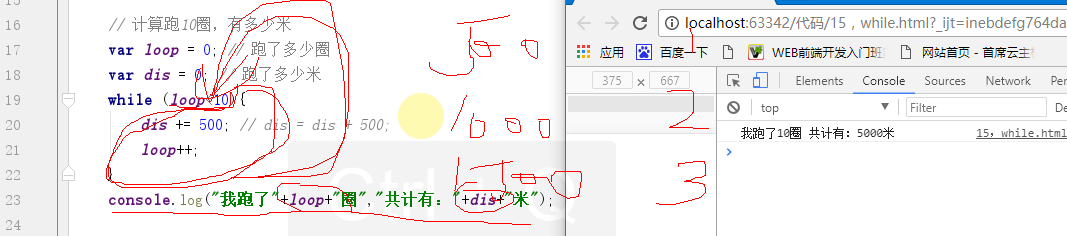
* While
* Do...while
* For

### while循环

格式：



代码：



注意细节：

1，在循环体中，一定要改变循环变量，否则陷入死循环。

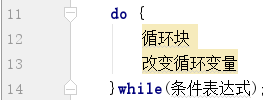
2，循环变量的初始化一定要放在while循环前面。

3，在while这个关键字后面有一个()，()中放一个条件表达式。

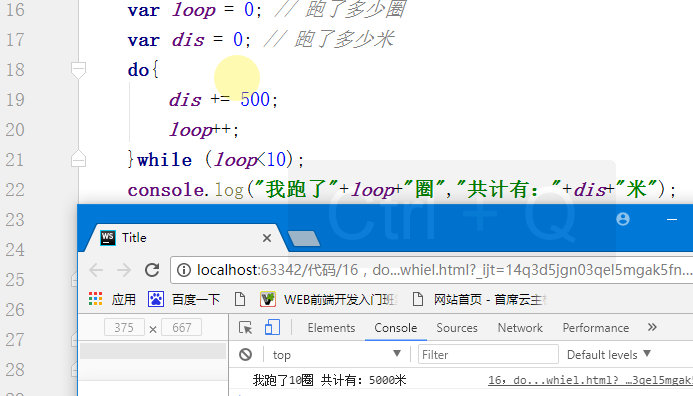
4，在()后面有一个{}，这个{}可写，可不写，建议都写，如果不写，就只会执行while后面的一条语句。

### do...while循环

格式：



代码如下：



注意细节：

1，循环变量的初始化要放在do的前面。

2，在循环体中，要改变循环变量的值，否则会陷入死循环。

3，循环体要使用{}包起来。

**问题：while和do...while有什么区别？**

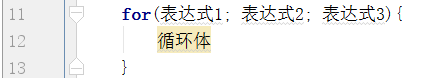
do...while的循环体一定会执行一次，不管条件如何，但是while可能一次也不执行。

在实际开发中，while的使用频率远远高于do...while。

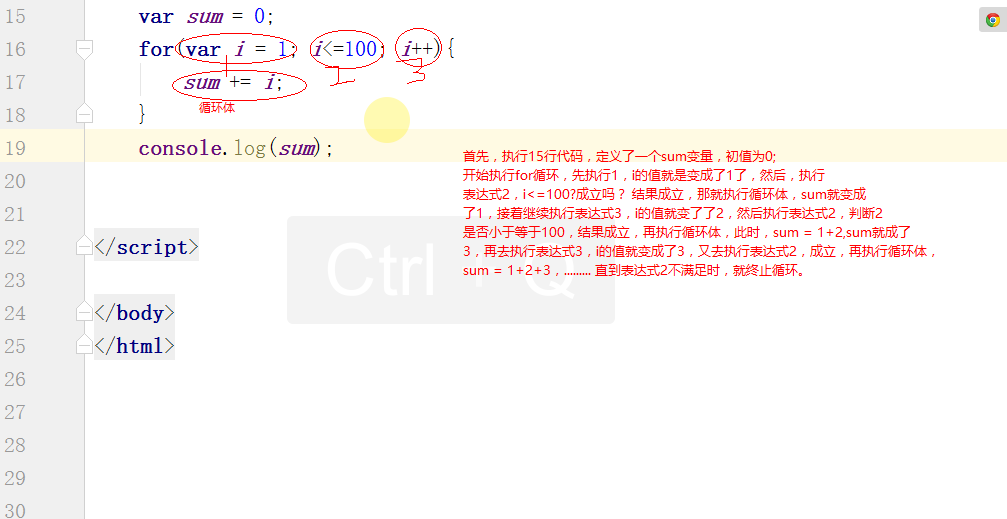
### for循环

这个非常重要，使用频率也是最高的。

格式：



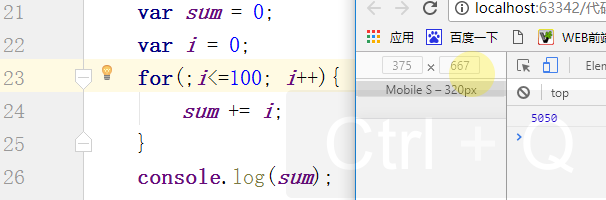
求1到100的种，代码如下：



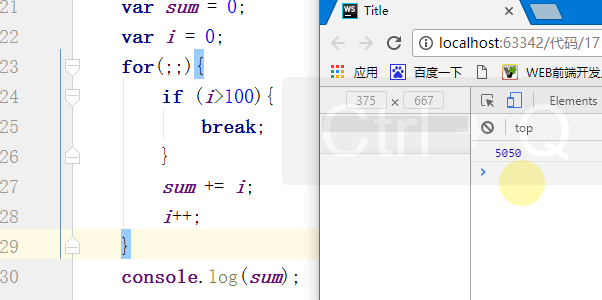
注意细节：

1，for循环后面的三个表达式都是可以省略的。尽量不要省。

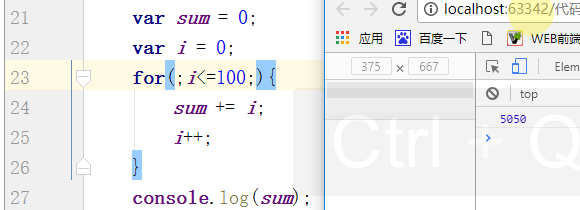
省略表达式1：



省略表达式2：



省略表达式3：

、

**问题：什么情况下用for，什么情况下用while？**

大部情况下，两者是一样的。

for循环通常用于循环次数确定的情况下，while循环通常用于循环次不确定的情况下。

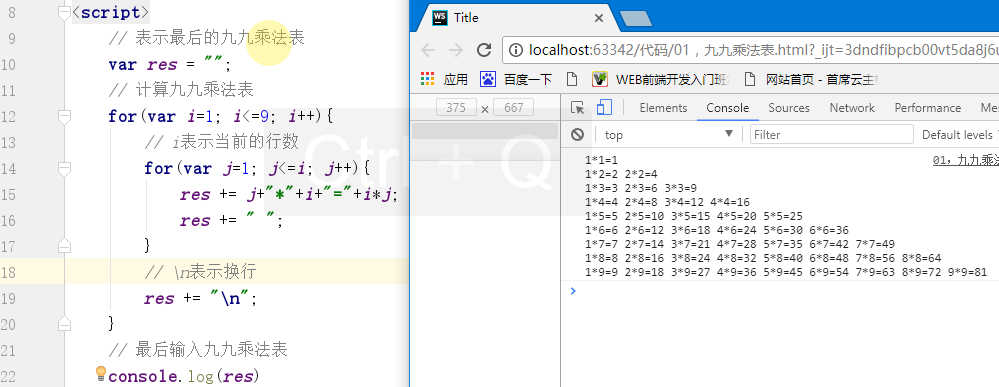
**作业：**

1，分析，在for循环，表达式1和表达式2，表达式3各执行了多少次？

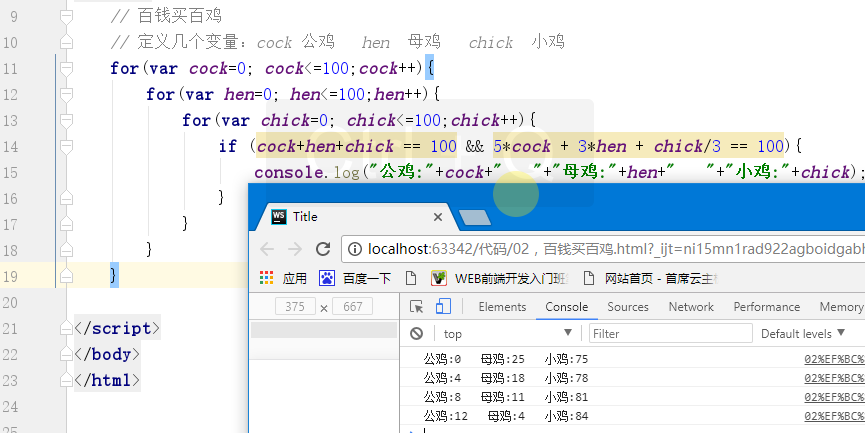
### （4）多重循环

前面说的循环这几种：while, do...while...,for。这里所说的多重循环是指在循环中嵌套一个循环，如果嵌套一个，那么叫二重循环，如果嵌套两个，那么三重循环，目前只需要了解二重和三重的就OK，至于四重，五重，暂时不用管。

**使用多重循环，打印九九乘法表：**



**百钱买百鸡：有个人有100块钱，买鸡，公鸡5块钱，母鸡3块钱，小鸡1块可以买3个，100块可以买各种鸡各多少个？ 代码如下：**



## 6，跳转语句

在JS中，跳转语句有如下几个：

* break
* continue
* 标签语句
* return

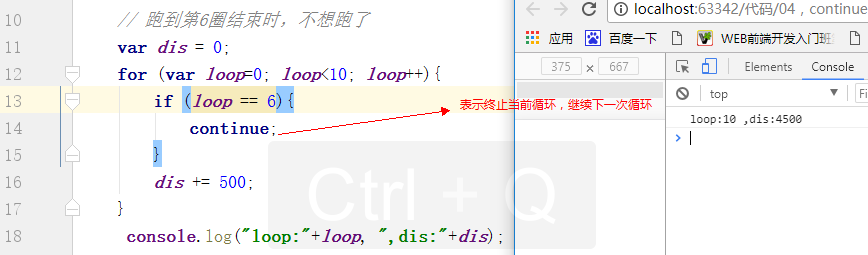
**break 表示跳出循环或swich**

模拟跑圈：一圈500米，跑10圈，代码如下：



**continue**

**当跑完第6圈，休息一了会，又继续跑，代码：**



**break和continue的区别？**

breake表示终止整个循环，continue表示终止当前循环，继续下一次循环。

一般情况下，在循环语句中，通常会使用break和continue，通常是配合if语句来使用。

还有一个特殊情况，就是在switch语句中，也会使用到break语句，表示跳出当前switch语句。

**标签语句**

目前已经淘汰了，现去学习就没有意义了。

**return**

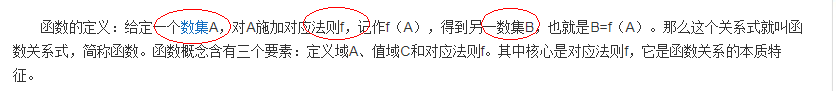
在函数中，使用最多，讲到函数时，重点讲return语句。

# 四，认识函数（重点）

## 1，啥是函数

在js中，函数是一等公民，是非常重要的。

也就是说，编程中的函数，也是模拟数学中的函数的，在数学中有很多函数，如y=x, y=x^2...... 数学中的函数，如下：



函数就是一个功能块，给一个输入，通过一个关系，会得到一个输出。如豆浆机，需要准备大豆，然后把豆放到机器中，最后得到一个杯豆浆。

回顾一下，之前见的函数有哪些？

alert("HHHHHH")

**函数有什么样的作用？**

封装---->代码重用。

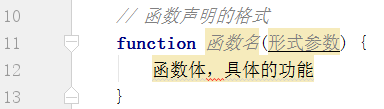
作为构造器，用于生成对象。

## 2，如果定义函数

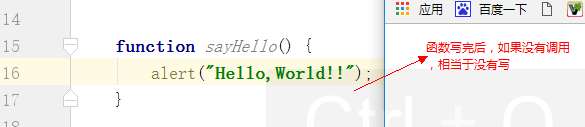
在js中，定义函数有两种方式：

* 函数声明
* 函数表达式

**第一种，函数声明，使用function关键字，基本格式 如下：**



基本使用：



上面的函数写完后，没有执行，原因是没有调用，如何调用呢？

通过函数名+()就可以调用 了，如下：



**注意细节：**

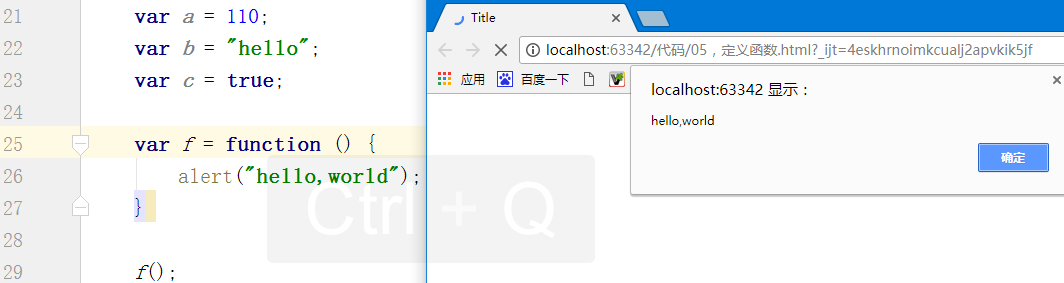
1，声明函数时，使用关键字function, 这个关键字和函数名之间有有空格的。

2，函数名的命令规则和变量的命令规则是一样的，在JS，命令规则如下：

* A, 大小写区分，在定义函数时，首字母小写，如果有多个单词，除了第一个单词，后面的每个单词的首字母大写，也就是小驼峰命令法，如：sayHello
* B, 函数名不能以数字打头，一般是以字母打头。

3，函数名后面跟上一个小括号，不能省略。小括号的作用是用来装载形式参数的。

**第二种，函数表达式，如下：**

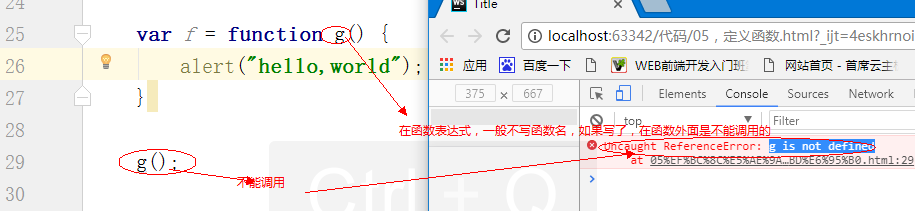


**注意细节：**

1，在函数表达式中，我们是将function整体，放在赋值运算符右边，是作为表达式来使用的。

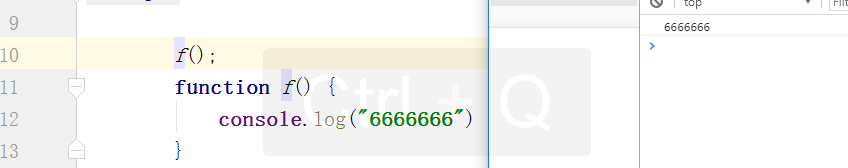
2，通过一个变量来引用当前funcitno这个表达式的，目的是为了方便后面使用。

3，函数名的地可以加上函数名，也可以不加，如果加函数名的话，只在函数内部可以使用。如下：

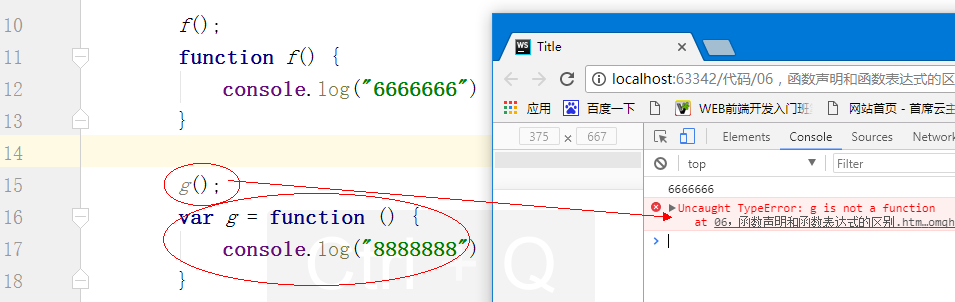


## 3，函数声明和函数表达式的区别？（如果难以理解，先放一下，但是需要知道它是重点）

函数声明，我们可以在声明函数的前面调用这个函数，如下：



对于函数表达式，能不能像上面一样，也把调用写在前面呢？

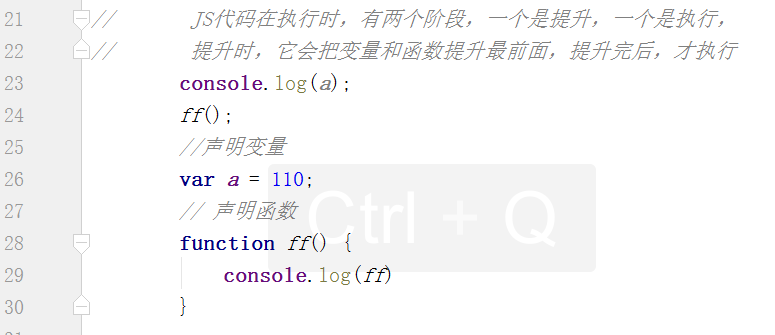


发现，函数表达式，不能像函数声明一样，把调用写在声明前面。原因是为什么呢？如下：

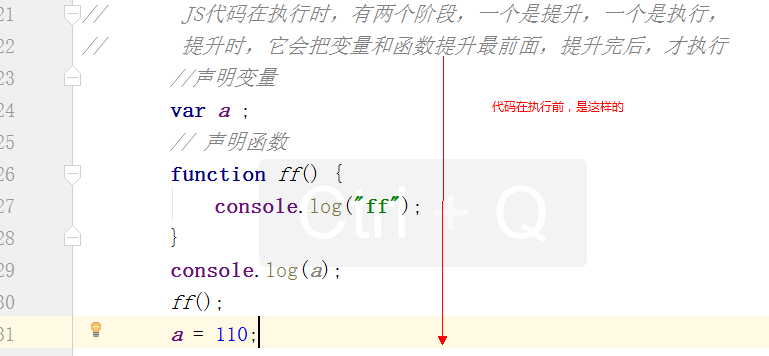
在JS中，代码执行时，是分两个阶段的，第一个阶段叫提升，第二个阶段叫执行。

提升阶段时，它会把声明的部分提到最前面。如下：

没有提升前，代码如下：



当提升后，代码如下：

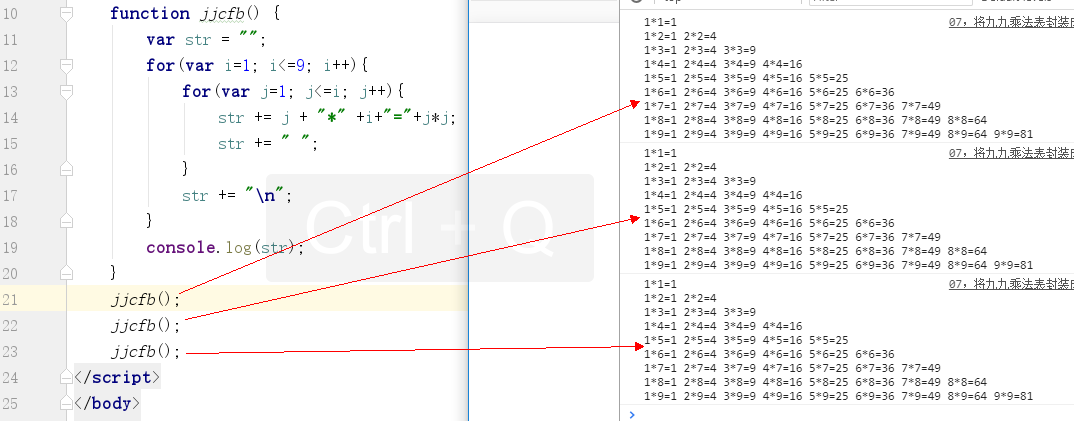


这个原因解释完后，就知道 ，为什么函数声明时，在声明前调用是OK的，函数表达式，在之前调用 是不OK的。

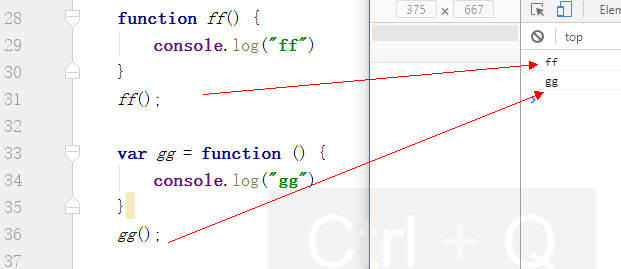
重点理解一下，JS代码的执行过程，分两个阶段，一个是提升，一个是执行。

**函数强化：**

对九九乘法表，进行一个封装，使用函数，如下：

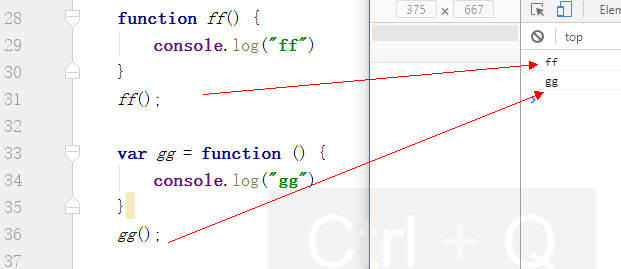


在封装一个函数，或定义一个函数时，有两种方式，一种是函数声明，另一种是函数表达式，如下：



## 3，函数调用及返回值

函数调用是非常简单，就是通过函数+()就OK了。代码如下：



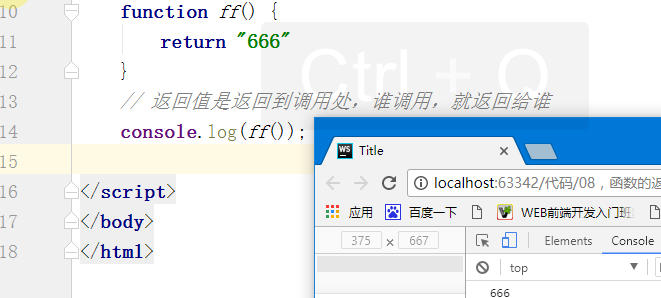
**在JS中， 函数调用有四种形式：**

* 作为函数
* 作为方法（是对象中的内容，对象在就业班是重难点）
* 作为构造函数（是对象中的内容，对象在就业班是重难点）
* 使用call(apply)间接调用（JS高级中的内容）

**关于函数的返回值：**

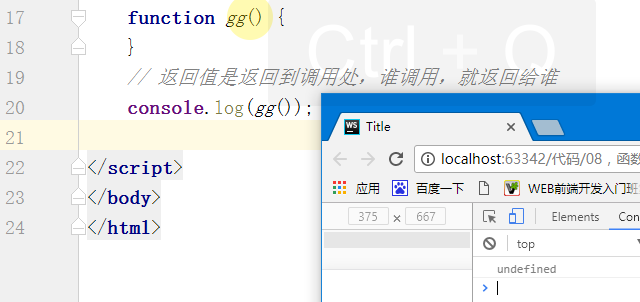
我们前面说过，函数就是一个功能块，对于这个功能块，我可以给这个功能块提供原材料，有时候，我们也需要功能块，给我返回一个结果，这个时候就涉及到了函数的返回值。类似于数学中，y = x^2 给一个x, 就有一个y与之对应。在编程中，想得有这个返回值，就要用到return。

代码如下：



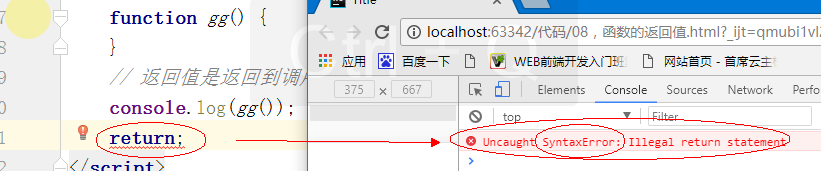
**如果没有return语句，函数给我们返回值东西呢？**

当不写return语句时，函数会给我返回一个undefined。如下：

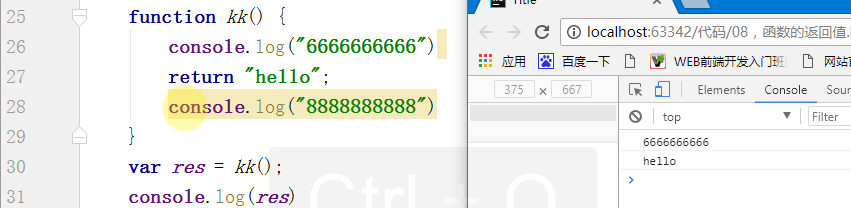


或者说，可以这样理解：当不写return语句时，可以理解成在函数最后有这个一句：reutrn undefined。

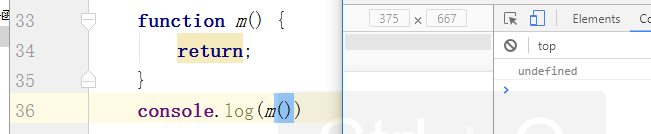
**return只能出现在函数中，不能出现在函数外：**



**在一个函数中，执行到retrun语句时，后面的代码就不再执行了，如下：**



**如果只写了reuturn语句，在reurn后面什么也没有，也是返回undefined**



## 4，函数的参数

豆浆机为例：

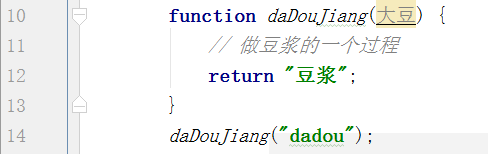
过程：

首先向机器中放大豆。

打开开关

得到豆浆

代码模拟如下：



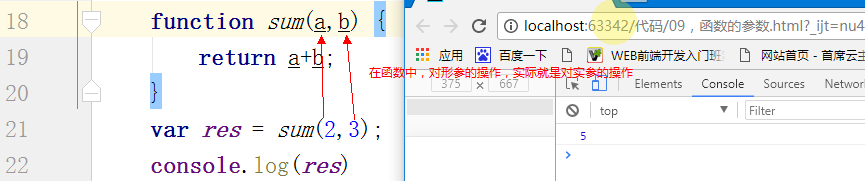
**在定义函数和调用函数时，有两个参数：**

* 形式参数，简称为形参，表示在定义函数时，小括号中的参数。
* 实际参数，简称为实参，表示在调用函数时，小括号中的参数。

如下：



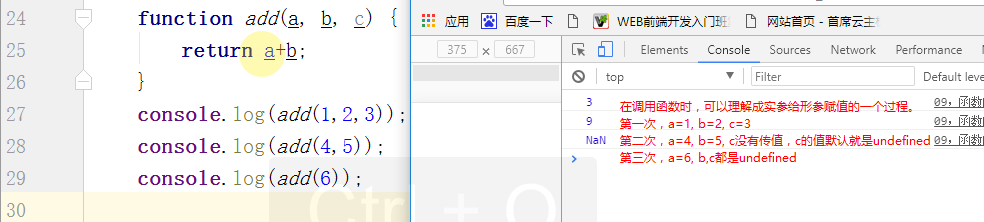
案例：写一个函数，这个函数的功能是求两个数的和：



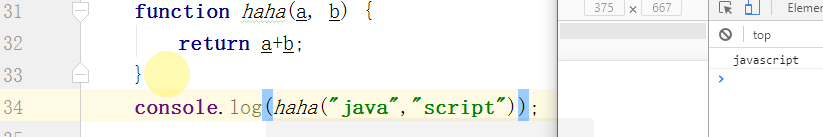
正是由于函数的参数，才使得JS中的函数变得很强大。

注意细节：

1，函数的实参与形参不需要严格对应。如下：



2，在JS中，对于形参，是没有数据类型要求的：



**小小面试题：**

JS是一门弱数据类型语言，具体有哪些体现？

* 变量在声明时，不需要指定数据类型。
* 函数中的形参在声明时，也不需要指定它的数据类型

## 5，作用域

**什么是作用域？**

作用域针对变量来说的，要定义一个变量，那么这个变量就有相应的作用域。

作用域是指变量的一个生效范围，在程序中，什么地方可以访问这个变量，什么地方不能访问这个变量。

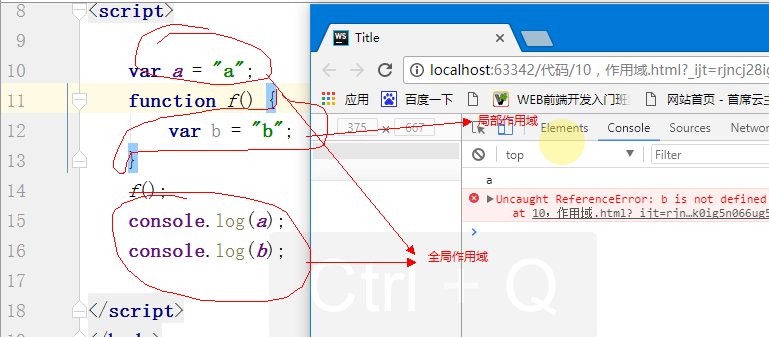
**在JS中，作用域有两个：**

* 全局作用域 ---> 全局变量
* 局部作用域 ---> 局部变量

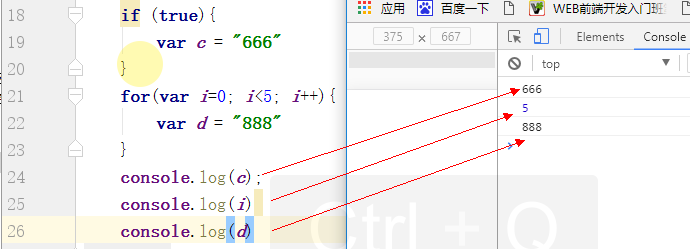
在讲变量时，变量分两种，一种是全局变量，一种是局部变量。是以函数作为分界点。

在函数中的变量，叫局部变量，在函数外的变量，叫全局变量。

代码如下：



在JS中，是没有块级作用域，块级作用域是使用{}来表示的。



**小结：**

局部作用域（函数中）：可以访问全局变量，也可以访问**当前作用域**中的局部变量

全局作用域（函数外）：是不能访问局部作用域中的变量的

**复习：**

**1，什么是JS？**

它是一门运行在浏览器端的脚本语言。是门真正的编程语言（变量，数据类型，运算符与表达式，函数，数组，字符串....）。

目前JS，不只可以运行在浏览器端，也可以运行在手机端和服务器端。

JS也可以开发游戏。

**2，什么是变量？如何去声明一个变量？**

可以理解成一个容器，用来放各种数值的。

变量有两部分组成：变量名（出现在赋值运算符的左边）

变量值（出现在赋值运算符的右边）

可以使用var关键字来声明一个变量。如var a = 110; 这就是声明了一个变量，变量的名为a，变量的值为110，变量的数据类型为number。

声明变量四种常见的形式：

**1，单独声明**

**2，一次声明多个**

3，重复声明（避免）

4，遗漏声明（避免）

**3，全局变量和局部变量？**

它们的分界点是函数。在函数中的变量叫局部变量，在函数外的变量叫全局变量。

**4，在JS中，有哪些数据类型？**

JS中的数据类型分两点：

基本数据类型：

数值(number)

字符串(string)

布尔(boolean)

undefined

null

引用数据类型

Object

**5，在JS，有哪些运算符呢？**

运算符是用来表示某种功能的符号。

把运算符分类如下：

算术运算符（+，-，\*，/，**++，--，**%）

赋值运算符（**=**，+=，-=，\*=，/=，%=）

位运算符（就业班说）

关系运算符（**==**，===，！=，！==，>，<，>=， <=）

逻辑运算符（**&&，||**，！）

其它运算符（字符串连接运算符+，三元运算符？：，判断数据类型运算符typeof）

**6，什么是表达式，JS中都有哪些表达式呢？**

* 任何有值的内容都是表达式。
* 原始表达式
* 运算符构成的表达式
* 对象和数组初始化表达式
* 函数定义表达式
* 调用表达式
* 对象创建表达式

**理解：=的副作用。**

如果给一个变量赋值了，如：i=k=j=110; =运算符具有右结合性，首先，把110赋给j，但是赋值运算符有副作用，它的副作用就是整个表达式也有一个值，也就是说j=100这个整体也有一个值，这个值就是110. 接着，就把就110又赋给了k ....

**7，什么是语句？在JS中都有哪些语句？**

语句就是一个命令。一般以分号结束。作用是用来发送某条指令。

JS中的语句有：

表达式语句

声明语句 （声明变量和声明函数）

**条件语句（if if..else... if...else if...else... switch）**

**循环语句（do...while... while for）**

**跳转语句 （break, continue,return）**

**8，什么是函数？在JS，如果定义函数？ 函数调用 ？ 函数参数？ 函数返回值？**

函数就是一个功能块，它的作用就是封装代码，代码重用。还有其它作用，如组织代码，作为构造器，用于生成对象。

在JS中，定义函数有两种方式：一种是函数声明，另一种是函数表达式。

函数声明：funciton ff(){函数体}

函数表达式：var gg = function (){}

函数调用是通过 **函数名（）**

函数参数分：形式参数（形参）和实际参数（实参）。函数调用的过程就是实参向形参赋值的一个过程。在函数体中，对形参的操作，说白了就是对实参的操作。

函数的返回值是通过return语句返回的。这个值是返回到函数的调用处。

# 五，认识对象（了解）

对象在JS，很重要，非常重要，分两个阶段讲：

* **对象的初识**
* 面向对象（就业班）

## 1，啥是对象

在现实中，不说男女朋友间的这个对象。**现实中，一切都是对象。**如桌子，笔记本，饮水机，黑板，垃圾箱.....，也就是客观存在的一切物体。

在现实中的对象，我们如何去描述它？

**描述一台电脑：**

静态的特征：颜色，尺寸，CPU，内存，硬盘，显卡，系统....(名词)

动态的特征：玩游戏，发信息，上网，拍照.... (动词)

**描述一个人：**

静态的特征：姓名，身高，性别，体重，住址...(名词)

动态的特征：打篮球，画画，开车... (动词)

所以说，在描述一个对象时，一般我们是通过静态的特征和动态的特征进行描述的。

在JS中，静态特征和动态的特征，我们都叫它**属性**。

**静态的特征，在JS，一般是使用变量来描述。**

**动态的特征，在JS，一般是使用函数来描述。**

代码如下：



**重新理解一下：console.log()?**

Console是一个对象，在console这个对象中，有一个方法，这个方法叫log()。调用时，是使用点语法进行调用的。

**在JS中，对象是分三类？**

内置对象（Number，String, Boolean, Date...）

宿主对象（BOM， DOM）

自定义对象

## 2，如何创建对象

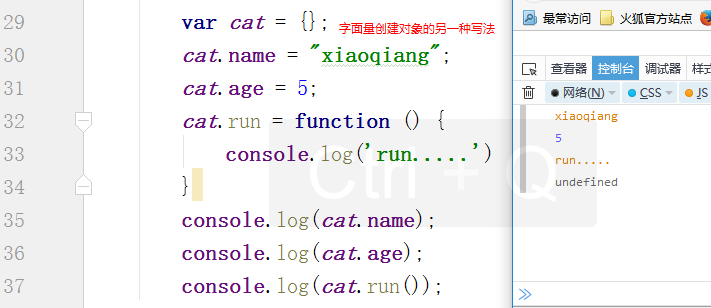
在JS中，创建对象两种方式：

* 字面量
* new

通过字面量创建对象，代码如下：



字面量创建对象方式，还可以这样写，如下：



**使用new的方式来创建对象，如下：**



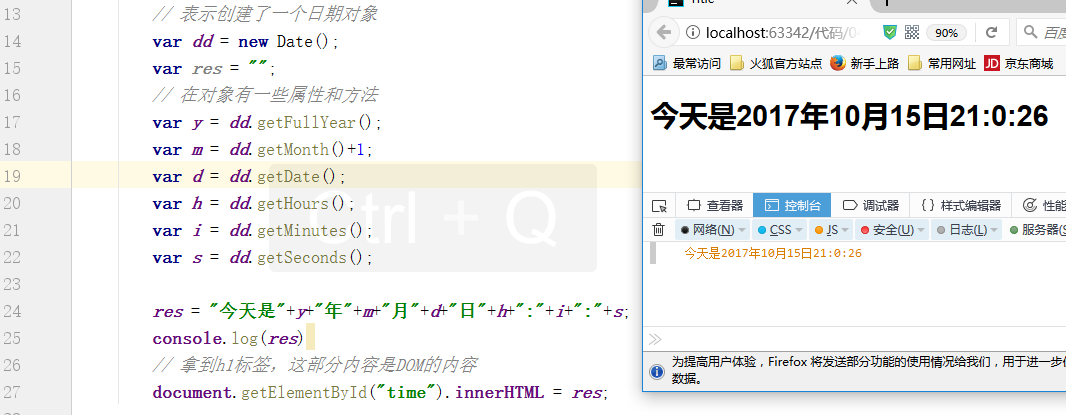
## 3，date对象

Date对象是JS内置的一个对象，作用是提供关于日期相关的操作。

代码如下：



案例：

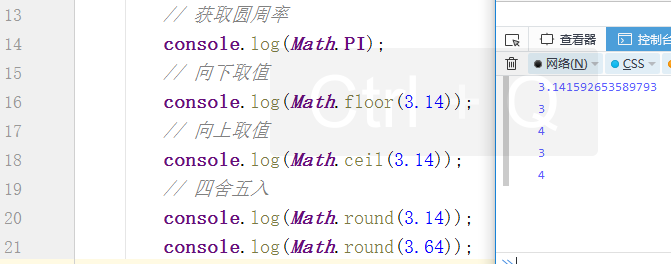


## 4，啥是单体内置对象

不需要使用new的对象，就可以创建的对象。叫单体内置对象。

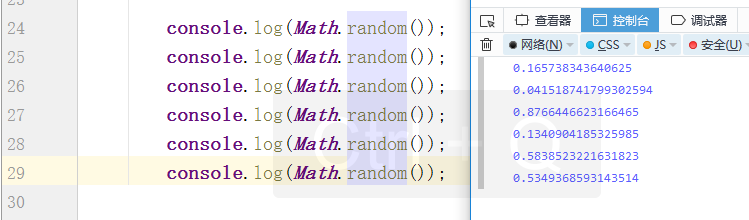
学一个：Math 数学的意思

代码： floor是地板 ceil 是天花板 round周围

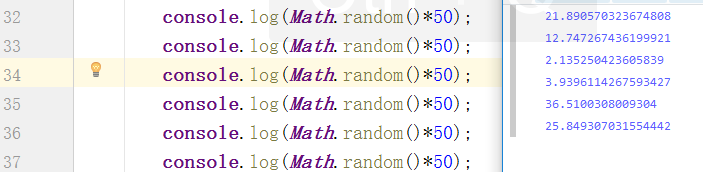


有一个非常重要的方法，随机数：random();

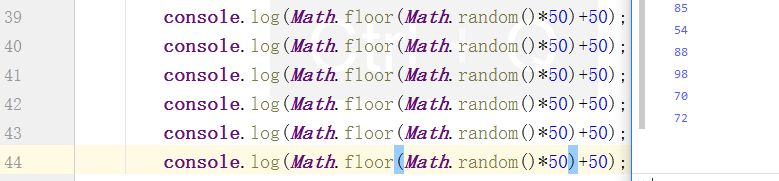
默认是随机一个小数，范围是[0,1), 包括0，不包括1



随机一个[0, 50)之间的数，如下：



随机一个[50, 100)之间的数，如下：



**为什么叫单体内置对象？**

由于，在JS运算时，Math这个对象就常驻在内存了，我们可以直接使用了，不需要再去创建它，也就new它了

## 5，自定义对象

略

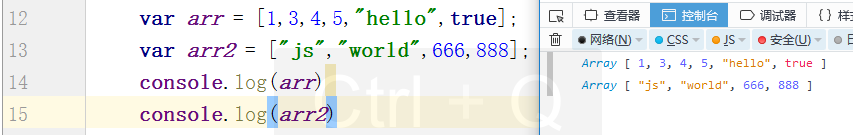
# 六，数组

## 1，啥是数组

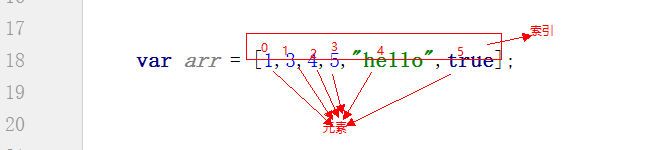
数组：数，理解成数值。组，多个，一组。数组就是多个数值的集合。

数值值就是前面我们所说的各种数据类型的值。

定义一个数组: var arr = [1,2,3,”hello”,true]; 代码如下：



在数组中，每一个值，叫元素，每一元素在数组中都有一个位置，这个位置，叫索引，索引是用数字来表示的，默认是0开始。如下：



**JS看的数组是比较特殊，如下：**

* 1，JS数组是无类型，意思就是在[]中，你想放什么数值就放什么数值。
* 2，数组中的元素可以是任意值，可以是对象，也可以数组。



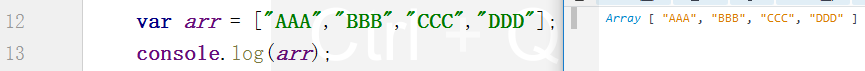
* 3，数组是动态的，可以根据我们的需要增长或缩减
* 4，数组元素的索引，默认是连续，可以改变，让它们之间有空缺。
* 5，JS中的数组是对象的特殊形式，

## 2，如何创建数组

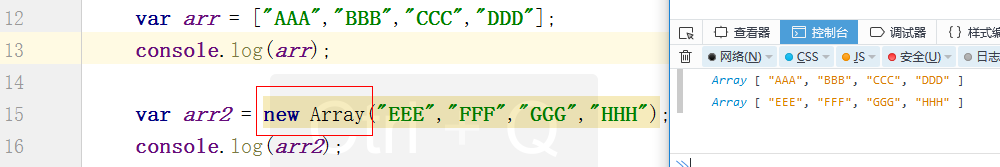
在JS中，创建数组两种方式：

* 字面量
* New

**字面量：**



**new的方式：**

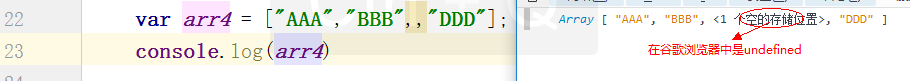


**注意细节：**

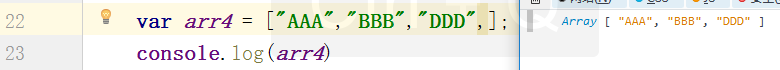
1，使用字面量创建数组时，它里面的元素不只可以是常量，也可以是其它表达式。



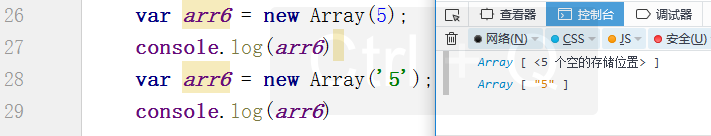
2，如果省略了某个元素，省略的元素默认被赋于undefined



3，数组字面量创建时，最后的逗号可以有，可以没有



4，new Array()等价于[],表示创建了一个空数组。但是new Array(5)，它不表示创建了一个数组，数组中放了一个数值5，而是表示创建一个有5个长度的数组。

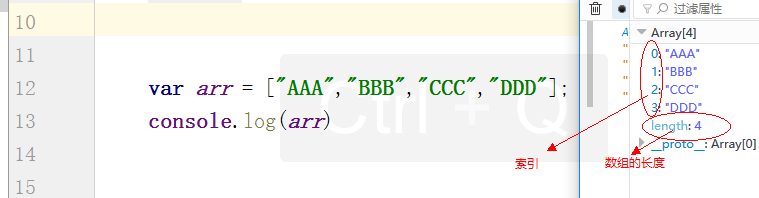


说明：当new的时候，只有一个参数，需要注意的它数据类型。如果是number，那么就表示创建一个空数组，数组的长度为number。

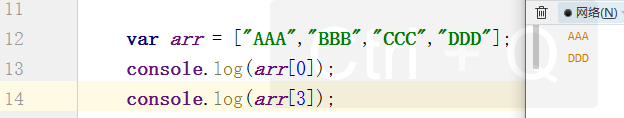
## 3，数组的一些操作

在计算机中，说到操作，一般要想到：增删改查。需要用到索引。

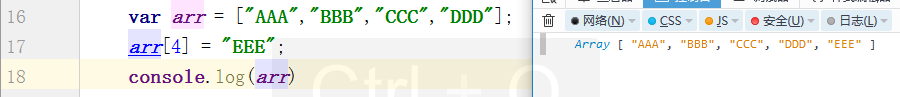
代码：



如何获取一个指定索引的数组元素？ 格式：数组名[索引]

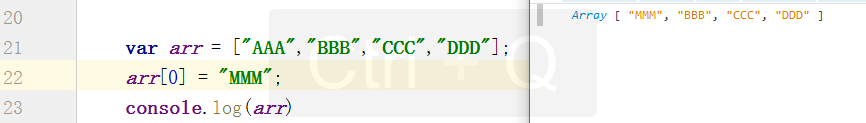


增加：



增加后，数组的长度是也变的。

修改：

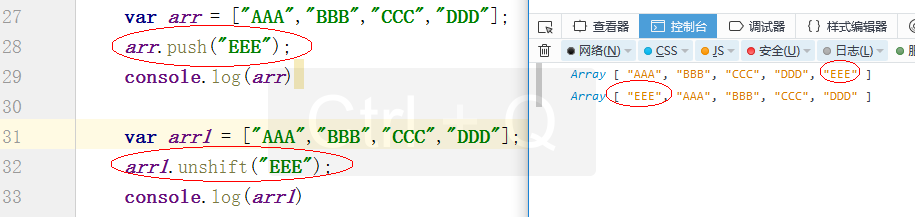


对于添加和删除，我们有其它的方式，重点掌握。

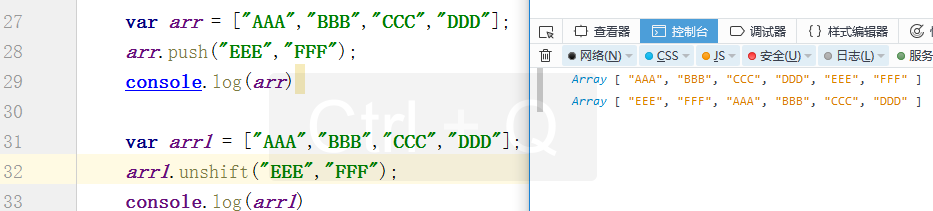
**添加：push(推)和unshift()**

**删除：pop和shift**

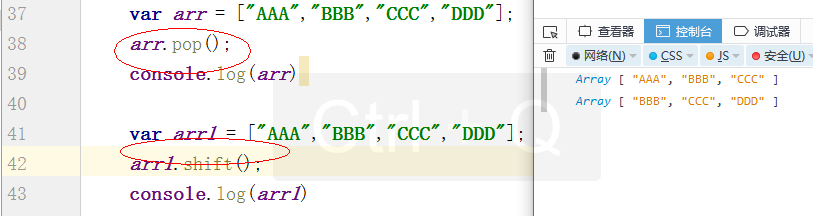
**添加：**



添加，一次性可以添加多个。如下：



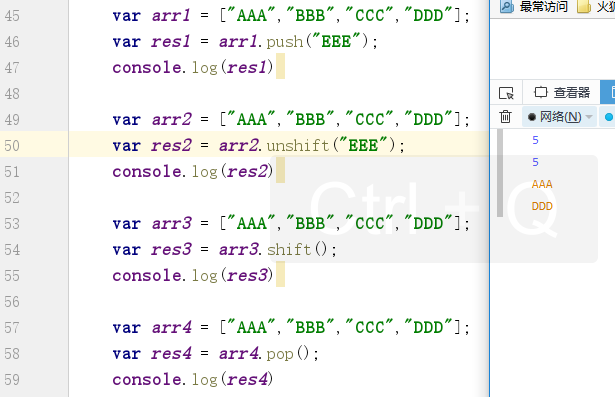
**删除：**



**注意细节：**

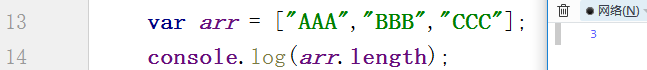
1，pop和shift是没有参数的，每一次只能删除一个。

2，pop和shift方法返回的是被删除的元素，push和unshift返回的是一个新数组的长度。如下：



## 4，数组的长度

每一个数组都有一个length属性，用来表示数组的长度。如下：



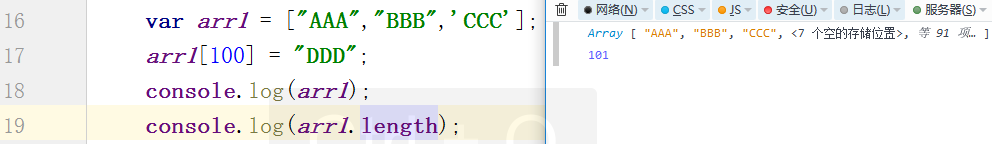
在数组中肯定找不到一个元素的索引值大于或等于数组的长度。

**如何理解上面的话？**

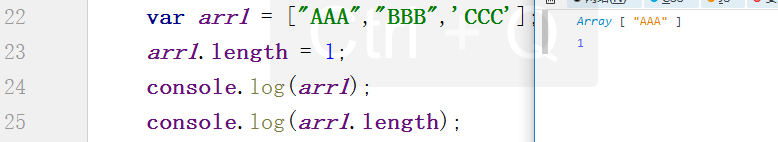
在数组中肯定找不到一个元素的索引值大于或等于数组的长度。为了维持这个规则不变，数组有两个特殊的行为：

* 1，如果为一个数组元素赋值，它的索引i大于或等于现有的长度，length的值将设置为i+1。
* 2，当设置了length属性为一个小于当前长度的非负整数n时，当前数组中的那么索引值大于或等于n的元素将被删除。

第一句话的理解：



第二名话的理解：



## 5，如何遍历数组

遍历：说白了，就是数组中的元素，一个一个取出来。取出的过程，就是遍历的过程。

在JS中，遍历数组有两个方法：

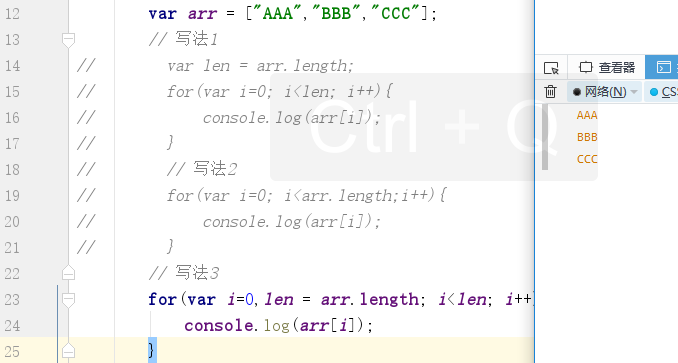
* for循环
* forEach方法

**for循环遍历：**

需要结合下面几个条件：

* For循环
* 数组的长度
* 使用数组名[索引值]获取每一个元素

如下：



**注意细节：**

1，后面我们会学到for...in... 不要使用for...in...去遍历数组。For...in...是用来遍历对象的。

**练习几处题：**

|  |
| --- |
| 马克思的手稿中有这样一道有趣的数学题：有30个人，其中有男人，女人，小孩。他们在一家饭馆中吃饭，共花费50先令。 如果每个男人吃饭要花3先令，每个女人要花2先令， 每个小孩要花1先令，问男人，女人，小孩各多少人？ |

|  |
| --- |
| 计算e=1+1/1！+1/2！+1/3！……+1/n！的前50项 |

|  |
| --- |
| 输入一个数（不限位数），输出一个数字的位数。 |

|  |
| --- |
| 打印九九乘法表 |

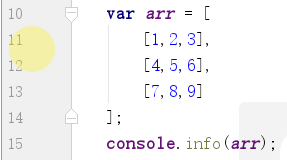
|  |
| --- |
|  |

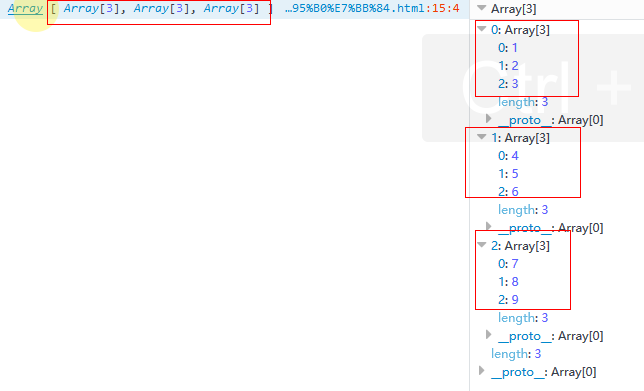
## 6，多维数组

前面所学的数组都是一维的。只有一个维度。[1,2,3]

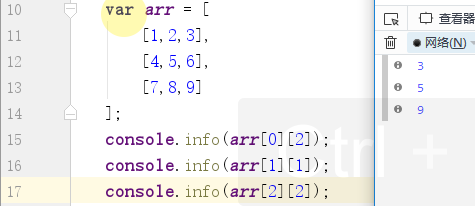
在JS中，经常会用到多维数组，数组中嵌套一个数组，叫二维，在嵌套的数组中再嵌套一个，叫三维，目前，只需要了解二维和三维就OK。

二维数组：在一个数组中，一个元素也是一个数组。如下：

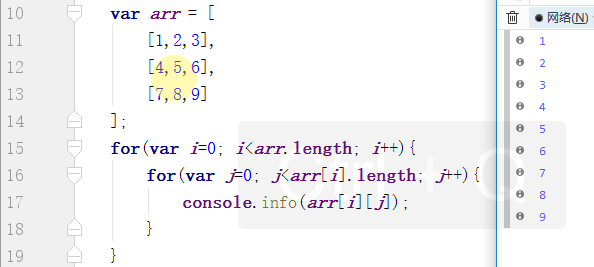




**如何访问二维数组中的元素？**



**如何遍历二维数组？**



## 7，数组中常见的一些方法

说明：在编程中，有很多手册，程序员需要多看手册，在网上就可以下载。

手册中的内容都很全。

如JS手册中的关于数组方法的内容，如下：



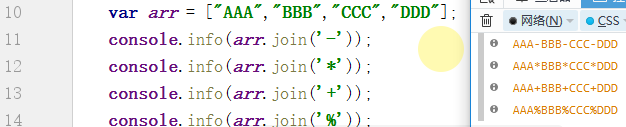
学习时，把常见的方法学一下，就OK了，其它的，用到的时候再去查手册。

常见的如下：

* join
* reverse
* concat
* sort
* slice
* push
* pop
* unshift
* shift
* toString

**join**

**将数组中的元素，使用某种符号连接起来，形成一个字符串。**

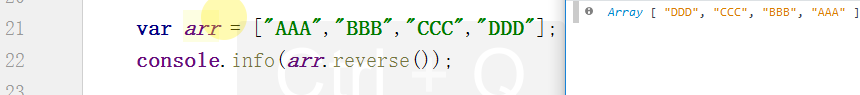


如果join中不带参数，那么它默认是使用逗号连接的：

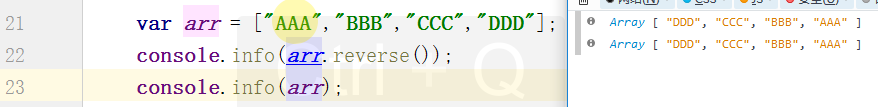


**reverse**

意思：反转，倒置 将数组中的元素，顺序倒置：

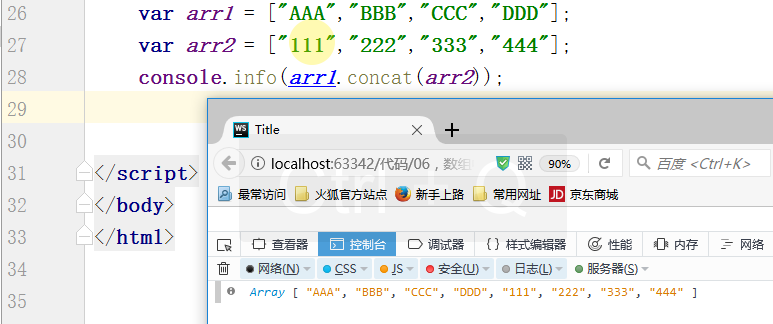


使用reverse时，数组是立即发生变化。



**concat**

**将多个数组连接成一个数组，可以理解成合并**

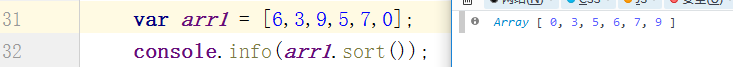


注意：如果数组是多维的，是不会展开的。



**sort**

**排序，用于对数组中的元素进行排序**

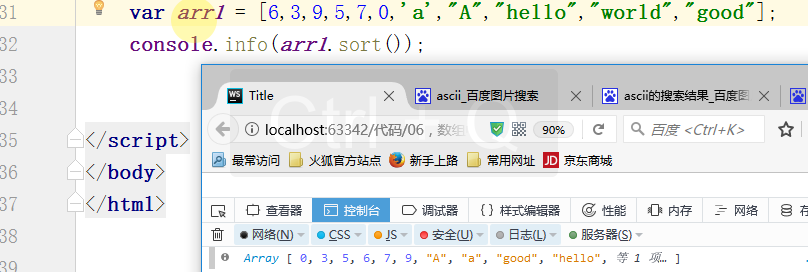


sort方法，默认情况下，是按照ascii来排序的，从小到到进行排序。

啥是ascii码：



如果是字母时，也会根据ascii码来进行一个排序：



假如：要对hello和hi进行排序，它很先从第一个字母进行比较，如果第一个字母的ascii码值一样，然后，比下一个ascii值，直到比较出结果。

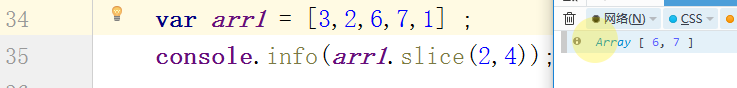
**slice**

**切割的意思，用来截取数组中的片段。**

slice(start, end) 索引 不包括end这个索引对应的元素。

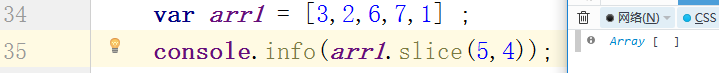


练习：数组如 [3,2,6,7,1] 截取 6，7

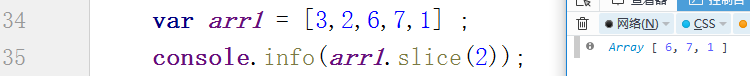


注意细节：

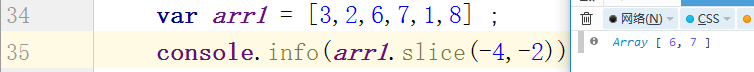
1，在使用slice时，必须要保证start<end。否则会得到一个空数组。



2，如果省略了end，表示截取到最后一个元素



3，start和end可以是负值，负数表示从后向前截取。如果是负数，只要使用负数+length转成正数就OK



**push**

**pop**

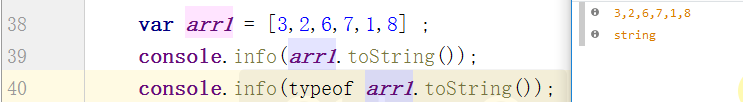
**unshift**

**shift**

**略**

**toString**

**是从Object继承过来。表示将数组转化成字符串。**



**使用数组中的内容，做一个小案例：**



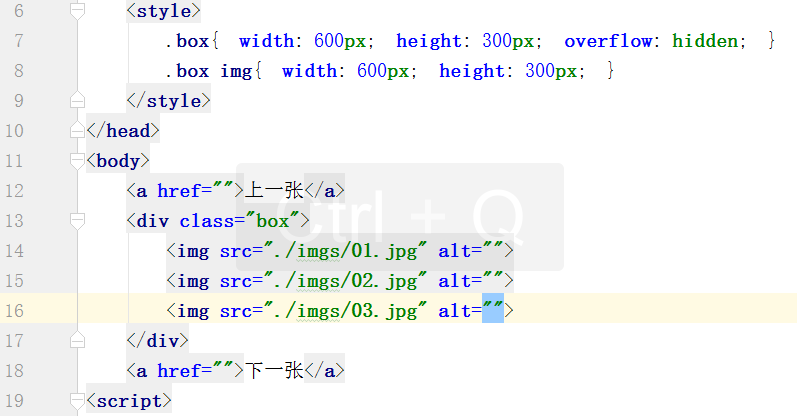
分析：

1，用到DOM中的内容，需要得到a标签。然后给a标签注册一个点击事件。

2，当点击时，需要切换图片，说白了就是去改变img标签中的src这个属性值。

3，可以将img的所有src放到一个数组中，当到改变数组中的src的一个顺序就OK了。

HTML和CSS代码如下：



JS代码如下：



至此，这个小案例就完成了。



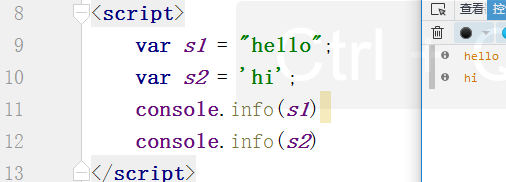
只学会JS，不学DOM，那么几乎什么效果都写不了，只有学完DOM后，就可以完成各种各样的特效了。DOM的内容也是在就业班中讲。

# 七，字符串

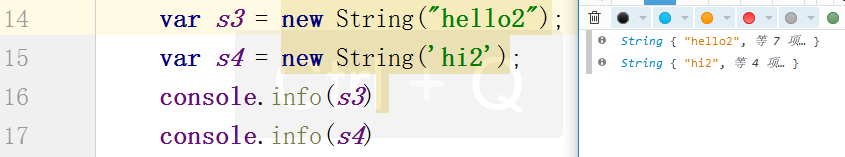
## 1，创建字符串

* 字面量
* New

字面量创建字符串。在JS中字符串，使用单引用和双引号都可以。



new的方式创建字符串：



## 2，字符串的常见操作

* 获取字符串长度 -- length属性
* 字符串连接 -- concat
* 查找 -- charAt、charCodeAt、indexOf、lastIndexOf、search
* 截取 -- substr、substring、slice
* 大小写转换 -- toLowerCase、toUpperCase、
* 替换 -- replace
* 转数组 -- split

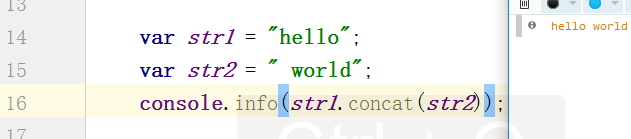
**length属性**

表示字符中的长度，也就是字符的个数。



**concat**

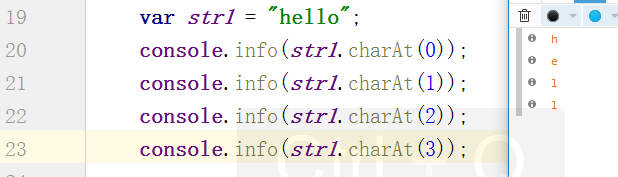
表示字符串的连接，类似于+



**charAt**

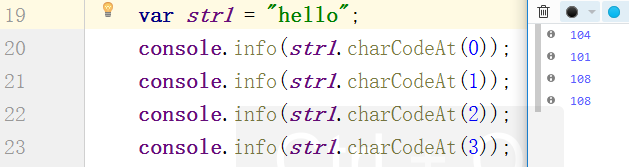
char是字符的意思，at是位置的意思

可以将字符串，看作是一个字符数组，是有索引，可以通过charAt(索引)-->对应的字母。



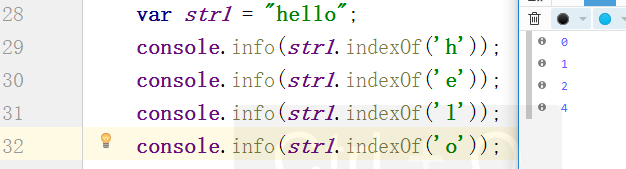
**charCodeAt**

可以将字符串，看作是一个字符数组，是有索引，可以通过charAt(索引)-->对应的字母的ascii码。



**indexOf**

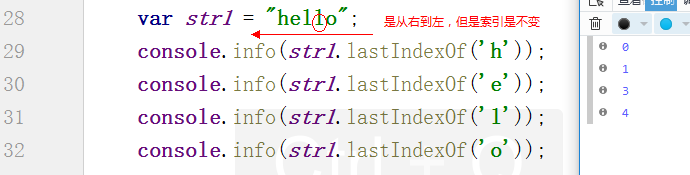
根据字母来获取索引，和charAt是相反的。



注意：indexOf，自左到右，找到的是相对应的第一个字符的位置。

**lastIndexOf**

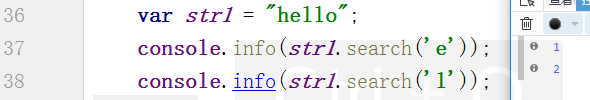
和indexOf一样，只不过是从右向左进行查找



注意：是从右向左，但是索引的值还是自左向右的。

**search**

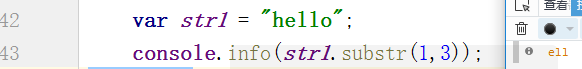




注意：search在正则表达式中用的最多。

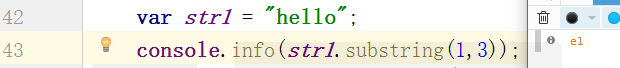
**substr**

substr(start,len)，第一个参数表示开始位置，第二个参数表示截取长度



**substring**

substring(start,stop)，第一个参数表示开始位置，第二个参数结束的位置

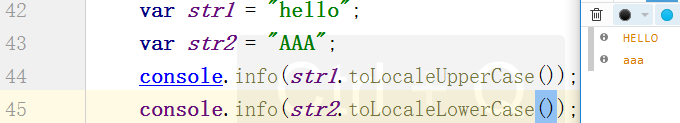


**slice**

**查手册！**

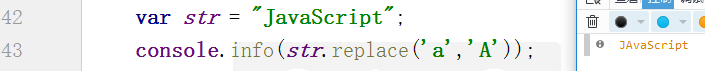
**toLowerCase toUpperCase**

**表示大小写转化**



**replace**

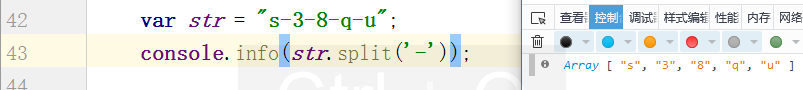
可以实现字符串替换操作



注意：只会替换一次。也是在正则表达式中用的最多。

**split**

**转数组。**



注意点：字符串需要有一定的规则才能转化成数组。

# 八，利用学过的内容做一些编程题

|  |
| --- |
| 题目：某班的成绩出来了，现在老师要把班级的成绩打印出来。  效果如下：xxx年xx月x日  星期x--班级平均分为：81  格式要求：  1.显示打印的日期。格式为类似“xxxx年xx月xx日 星期x”的当前的时间。  2.计算出该班级的平均分（保留整数）。  同学成绩数据如下： "小明:87;小花:81;小红:97;小天:76;小张:74;小小:94;小西:90;小伍:76;小迪:64;小曼:76" |
| 水仙花数是指一个 n 位数 ( n≥3 )，它的每个位上的数字的 n 次幂之和等于它本身。（例如：1^3 + 5^3+ 3^3 = 153），js实现1000以内水仙花数 234 n=3 2^3+3^3+4^3 = 234 |
| 青年歌手参加歌曲大奖赛，有10个评委打分，试编程求选手的平均得分（去掉一个最高分和一个最低分） 用到数组中的内容。 |
| 方式一： |
| 方式二： |
| 方式三(作业) |
|  |
|  |
|  |
|  |
|  |
| **作业** |
| 题1：计算给定数组arr中所有的元素总合。 |
| 题2：找出元素item在数组arr中的位置。 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |