JavaScript基础

1.初识JavaScript

1.1JavaScript是什么

JavaScript是世界上最流行的语言之一，是一种运行在客户端的脚本语言（Script是脚本的意思）

脚本语言：不需要编译，运行过程中由js解释器逐行来进行解释并执行

现在也可以基于Node.js技术进行服务端编程

JavaScript简称JS

1.2JS的作用

表单动态校验（密码强度测试）

网页特效

服务端开发（Node.js）

桌面程序（Electron）

App（Cordova）

控制硬件-物联网（Ruff）

游戏开发（cocos2d-js）

1.3浏览器执行JS

浏览器分为两部分，渲染引擎和JS引擎

渲染引擎：用来解析HTML和CSS，俗称内核

JS引擎：也成为JS解释器，用来读取网页中的JS代码，对其处理后运行

1.4JS组成

三部分组成：

1、ECMAScript（JavaScript语法）

2、DOM（页面文档对象模型）

3、BOM（浏览器对象模型）

1.5ECMAScript

由ECMA国际进行标准化的一门编程语言，这种语言在万维网上应用广泛，它往往被称为JavaScript或JScript，但实际上后两者是ECMAScript语言的实现和扩展

ECMAScript规定了JS的编程语法和基础核心知识，是所有浏览器厂商共同遵守的一套JS语法工业标准

1.6DOM

文档对象模型，是W3C组织推荐的处理可扩展标记语言的标准编程接口，通过DOM提供的接口可以对页面上的各种元素进行操作（大小、位置、颜色等）

1.7BOM

浏览器对象模型，它提供独立于内容的、可以与浏览器窗口进行互动的对象结构，通过DOM可以操作浏览器窗口，比如弹出框、控制浏览器跳转、获取分辨率等

1.8JS书写位置

JS有3种书写位置，分别为行内、内嵌和外部

1、行内式



注意：

1）可以将单行或少量JS代码写在HTML标签的事件属性中（以on开头的属性）

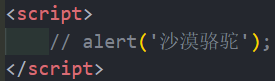
2）注意单双引号的使用：在HTML中推荐使用双引号，JS中推荐单引号

3）可读性差

4）引号易错

5）特殊情况下使用

2、内嵌式



注意：

1）可以将多行JS代码写到<script>标签中

2）内嵌JS是学习时常用的方式

3、外部JS文件



注意：

1）利用HTML页面代码结构化，把大段JS代码独立到HTML页面之外，既美观，也方便文件级别的复用

2）引用外部JS文件的script标签中间不可以写代码

3）适合于JS代码量较大的情况

1.9JS注释

单行注释 ctrl+/

多行注释 shift+alt+A

1.10JS输入输出语句

|  |  |  |
| --- | --- | --- |
| 方法 | 说明 | 归属 |
| alert(msg) | 浏览器弹出警示框 | 浏览器 |
| console.log(msg) | 浏览器控制台打印输出信息 | 浏览器 |
| prompt(info) | 浏览器弹出输入框，用户可以输入 | 浏览器 |

注意：prompt取过来的值是字符串类型

2.变量

2.1什么是变量

变量是用于存放数据的容器，我们通过变量名获取数据，甚至数据可以修改

本质：是程序在内存中申请的一块用来存放数据的空间

2.2变量的使用

变量在使用时分为两步：

1、声明变量

var age;声明一个名称为age的变量

var是一个JS关键字，用来声明变量，使用该关键字声明变量后，计算机会自动为变量分配内存空间

2、赋值

age = 10;给age这个变量赋值为10

变量的初始化：var age = 10;声明一个变量并赋值

2.3变量语法扩展

1、更新变量

一个变量被重新赋值后，它原有的值就会被覆盖，变量值将以最后一次赋的值为准

2、声明多个变量

var age = 18,

address = ‘地址’,

gz = 2000;

3、声明变量的特殊情况

1）只声明，不赋值，结果：undefined

2）不声明，不赋值，结果：报错

3）不声明，只赋值，结果：正常

2.4变量命名规范

1）由字母(A-Za-z)、数字(0-9)、下划线(\_)、美元符号( $ )组成，如：usrAge, num01, \_name

2）严格区分大小写。var app; 和 var App; 是两个变量

3）不能以数字开头

4）不能是关键字、保留字。例如：var、for、while

5）变量名必须有意义。

6）遵守驼峰命名法。首字母小写，后面单词的首字母需要大写。myFirstName

3.数据类型

3.1为什么需要数据类型

在计算机中，不同的数据所需占用的存储空间是不同的，为了便于把数据分成所需内存大小不同的数据，充分利用存储空间，于是定义了不同的数据类型

3.2变量的数据类型

1）变量是用来存储值的所在处，它们有名字和数据类型

2）变量的数据类型决定了如何将代表这些值的位存储到计算机的内存中

3）JavaScript是一种弱类型或者说动态语言。这意味着不用提前声明变量的类型，在程序运行过程中，类型会被自动确定

4）在代码运行时，变量的数据类型是由 JS引擎根据等于符号右边变量值的数据类型来判断 的，运行完毕之后，变量就确定了数据类型

5）JavaScript 拥有动态类型，同时也意味着相同的变量可用作不同的类型

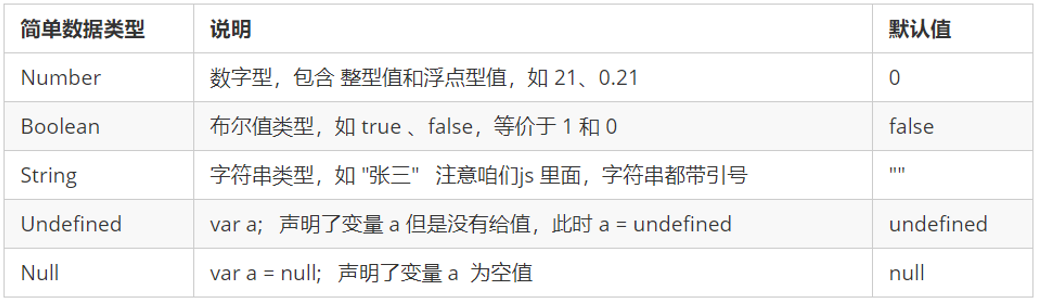
3.3数据类型的分类

JS把数据类型分为两类：

1、简单数据类型（Number,String,Boolean,Undefined,Null）

2、复杂数据类型（object）

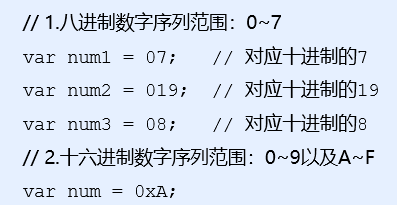
3.4简单数据类型



1、数字型Number

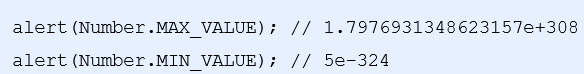
JS数字型既可以用来保存整数值，也可以保存小数

1）数字型进制：

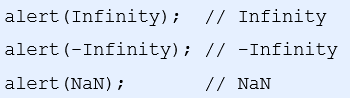


注意：在JS中八进制前面加0，十六进制前面加 0x

2）数字型范围：



3）数字型3个特殊值：



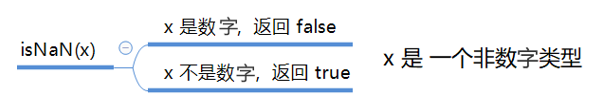
Infinity：代表无穷大，大于任何数值

-Infinity：代表无穷小，小于任何数值

NaN：Not a number，代表一个非数值

4）isNaN()

用来判断一个变量是否为非数字的类型，返回 true 或者 false



2、字符串型String

字符串型可以是引号中的任意文本，其语法为 双引号" "和单引号' '

因为HTML标签里面的属性使用的是双引号，JS这里我们更推荐使用单引号

1）字符串引号嵌套

JS可以用单引号嵌套双引号，或者用双引号嵌套单引号 (外双内单，外单内双)

2）字符串转义符



3）字符串长度

字符串是由若干字符组成的，这些字符的数量就是字符串的长度。通过字符串的length属性可以获取整个字符串的长度

str.length

4）字符串拼接

多个字符串之间可以使用 + 进行拼接，拼接前会把与字符串相加的任何类型转成字符串，再拼接成一个新的字符串

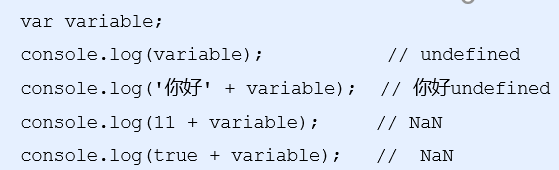
3、布尔型Boolean

布尔类型有两个值：true 和 false ，其中true表示真，而 false 表示假

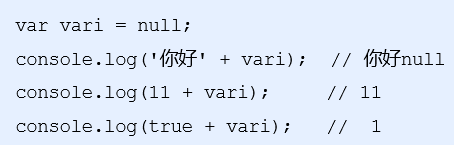
布尔型和数字型相加的时候，true的值为1，false的值为0

4、Undefined和Null

一个声明后没有被赋值的变量会有一个默认值undefined

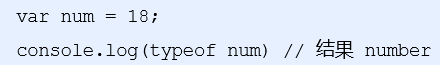


一个声明变量给null值，里面存的值为空

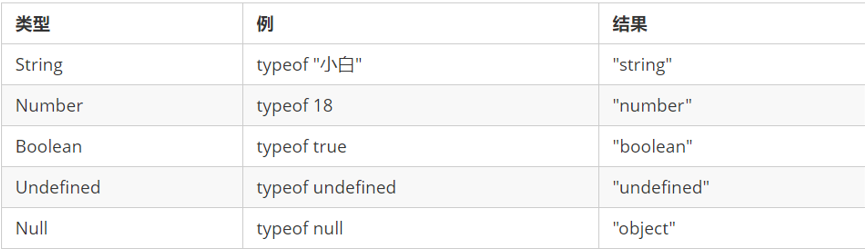


3.5获取变量数据类型

typeof可用来获取检测变量的数据类型



不同类型的返回值：



3.6数据类型转换

使用表单、prompt 获取过来的数据默认是字符串类型的，此时就不能直接简单的进行加法运算，而需要转换变量的数据类型。通俗来说，就是把一种数据类型的变量转换成另外一种数据类型

我们通常会实现3种方式的转换：转换为字符串型、转换为数字型、转换为布尔型

1、转换为字符串型



注意：三种转换方式，我们更喜欢用第三种加号拼接字符串转换方式，这一种方式也称之为隐式转换

2、转换为数字型



3、转换为布尔型



代表空、否定的值会被转换为false，如''、0、NaN、null、undefined，其余值都会被转换为true

4.JS运算符

4.1运算符

运算符也被称为操作符，是用于实现赋值、比较和执行算数运算等功能的符号

JS中常用的运算符有：算术运算符、递增和递减运算符、比较运算符、逻辑运算符、赋值运算符

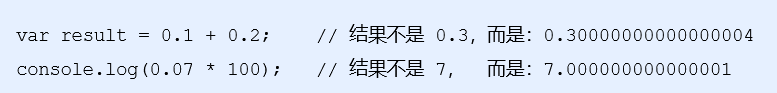
4.2算术运算符

概念：算术运算使用的符号，用于执行两个变量或值的算术运算



4.3浮点数精度问题

浮点数值的最高精度是17位小数，但在进行算术计算时其精确度远远不如整数



注意：不要直接判断两个浮点数是否相等

4.4表达式和返回值

表达式：是由数字、运算符、变量等以能求得数值的有意义排列方法所得的组合

表达式最终都会有一个结果，返回给我们，我们称为返回值

4.5递增和递减运算符

1、概述

在 JavaScript 中，递增（++）和递减（ -- ）既可以放在变量前面，也可以放在变量后面。放在变量前面时，我们可以称为前置递增（递减）运算符，放在变量后面时，我们可以称为后置递增（递减）运算符

注意：递增和递减运算符必须和变量配合使用

2、前置递增运算符

++num 前置递增，就是自加1，类似于 num = num + 1，但是 ++num 写起来更简单

使用口诀：先自加，后返回值

3、后置递增运算符

使用口诀：先返回原值，后自加

4.6比较运算符

概念：比较运算符（关系运算符）是两个数据进行比较时所使用的运算符，比较运算后，会返回一个布尔值（true / false）作为比较运算的结果

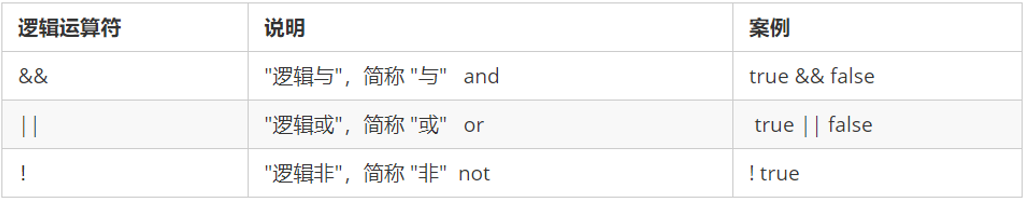


区别：

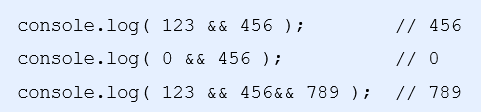


4.7逻辑运算符

概念：逻辑运算符是用来进行布尔值运算的运算符，其返回值也是布尔值。后面开发中经常用于多个条件的判断



短路运算：当有多个表达式（值）时,左边的表达式值可以确定结果时,就不再继续运算右边的表达式的值



4.8赋值运算符

概念：用来把数据赋值给变量的运算符



4.9运算符优先级



5.JS流程控制-分支

5.1流程控制

流程控制就是来控制我们的代码按照什么结构顺序来执行

流程控制主要有三种结构，分别是顺序结构、分支结构和循环结构，这三种结构代表三种代码执行的顺序

5.2顺序流程控制

顺序结构是程序中最简单、最基本的流程控制，它没有特定的语法结构，程序会按照代码的先后顺序，依次执行，程序中大多数的代码都是这样执行的

5.3分支流程控制if语句

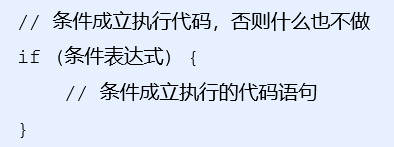
由上到下执行代码的过程中，根据不同的条件，执行不同的路径代码（执行代码多选一的过程），从而得到不同的结果

JS 语言提供了两种分支结构语句：

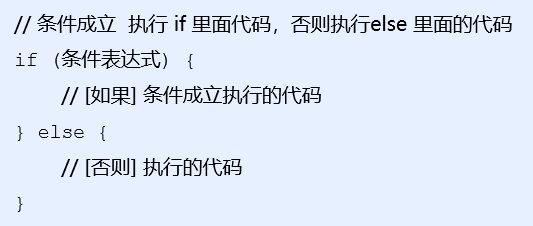
1）if语句

2）switch语句

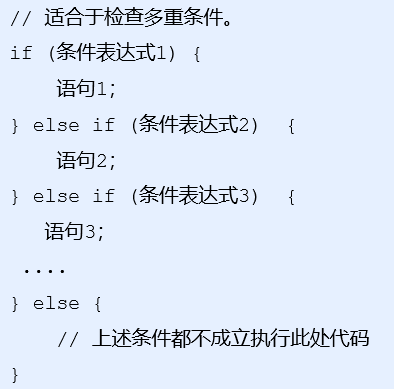
1、if语句



2、 if else语句



3、多分支语句



5.4三元表达式

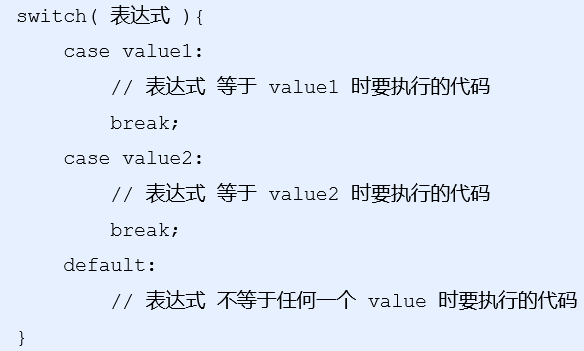
三元表达式也能做一些简单的条件选择。 有三元运算符组成的式子称为三元表达式

语法结构：表达式1 ? 表达式2 : 表达式3;

执行思路：如果表达式1为 true ，则返回表达式2的值，如果表达式1为 false，则返回表达式3的值

5.5分支流程控制switch语句

switch 语句也是多分支语句，它用于基于不同的条件来执行不同的代码。当要针对变量设置一系列的特定值的选项时，就可以使用 switch



语法结构：

1）switch ：开关 转换 ， case ：小例子 选项

2）关键字 switch 后面括号内可以是表达式或值， 通常是一个变量

3）关键字 case , 后跟一个选项的表达式或值，后面跟一个冒号

4）switch 表达式的值会与结构中的 case 的值做比较

5）如果存在匹配全等(===) ，则与该 case 关联的代码块会被执行，并在遇到 break 时停止，整个 switch 语句代码执行结束

6）如果所有的 case 的值都和表达式的值不匹配，则执行 default 里的代码

注意：执行case 里面的语句时，如果没有break，则继续执行下一个case里面的语句

6.JS流程控制-循环

6.1循环

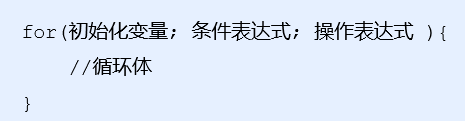
在实际问题中，有许多具有规律性的重复操作，因此在程序中要完成这类操作就需要重复执行某些语句

在Js 中，主要有三种类型的循环语句：for循环、while循环、do while循环

在程序中，一组被重复执行的语句被称之为循环体，能否继续重复执行，取决于循环的终止条件。由循环体及循环的终止条件组成的语句，被称之为循环语句

6.1for循环

语法结构：



1）初始化变量：通常被用于初始化一个计数器，该表达式可以使用 var 关键字声明新的变量，这个变量帮我们来记录次数

2）条件表达式：用于确定每一次循环是否能被执行。如果结果是 true 就继续循环，否则退出循环

3）操作表达式：每次循环的最后都要执行的表达式。通常被用于更新或者递增计数器变量。当然，递减变量也是可以的

6.2断点调试

断点调试是指自己在程序的某一行设置一个断点，调试时，程序运行到这一行就会停住，然后你可以一步一步往下调试，调试过程中可以看各个变量当前的值，出错的话，调试到出错的代码行即显示错误，停下

断点调试可以帮我们观察程序的运行过程

1）浏览器中按 F12--> sources -->找到需要调试的文件-->在程序的某一行设置断点

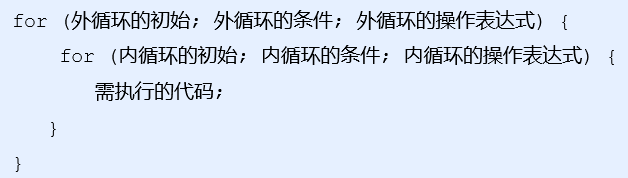
2）Watch: 监视，通过watch可以监视变量的值的变化，非常的常用

3）F11: 程序单步执行，让程序一行一行的执行，这个时候，观察watch中变量的值的变化

6.3双重for循环

循环嵌套是指在一个循环语句中再定义一个循环语句的语法结构，例如在for循环语句中，可以再嵌套一个for 循环，这样的 for 循环语句我们称之为双重for循环

语法结构：

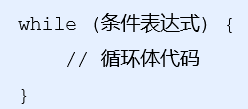


注意：外层循环执行一次，内层循环要执行全部次数

6.4while循环

while 语句可以在条件表达式为真的前提下，循环执行指定的一段代码，直到表达式不为真时结束循环

语法结构：

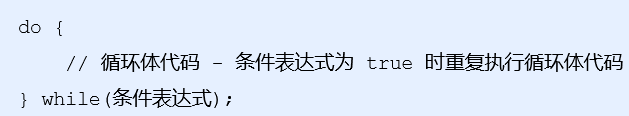


注意：while 循环和 for 循环的不同之处在于 while 循环可以做较为复杂的条件判断，比如判断用户名和密码

6.5do while循环

do... while 语句其实是 while 语句的一个变体。该循环会先执行一次代码块，然后对条件表达式进行判断，如果条件为真，就会重复执行循环体，否则退出循环

语法结构：



注意：do…while 循环语句至少会执行一次循环体代码

6.6continue break关键字

continue 关键字用于立即跳出本次循环，继续下一次循环（本次循环体中 continue 之后的代码就会少执行一次）

break 关键字用于立即跳出整个循环（循环结束）

7.JS命名规范及语法格式

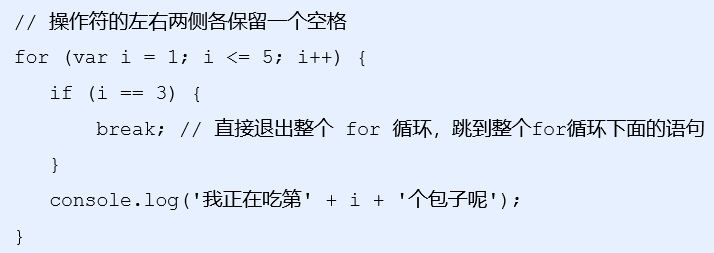
7.1标识符命名规范

1）变量、函数的命名必须要有意义

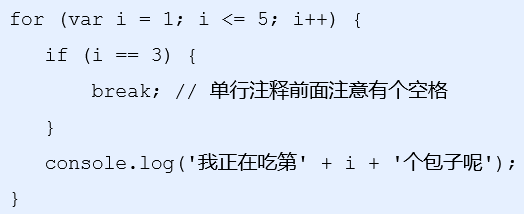
2）变量的名称一般用名词

3）函数的名称一般用动词

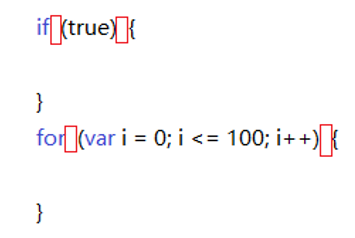
7.2操作符规范



7.3单行注释规范



7.4其他规范



8.JS数组

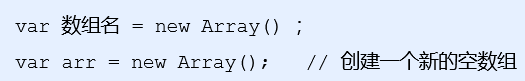
8.1数组的概念

数组是指一组数据的集合，其中的每个数据被称作元素，在数组中可以存放任意类型的元素。数组是一种将一组数据存储在单个变量名下的方式

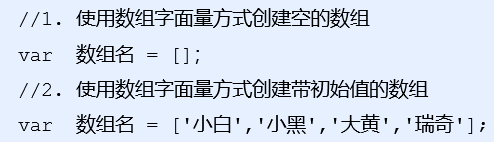
8.2创建数组

JS 中创建数组有两种方式：

1）利用 new 创建数组



2）利用数组字面量创建数组



注意：数组的字面量是方括号[ ]

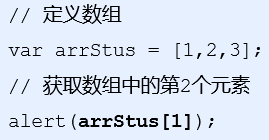
数组中可以存放任意类型的数据，例如字符串，数字，布尔值等



8.3获取数组中的元素

索引 (下标) ：用来访问数组元素的序号（数组下标从 0 开始）

数组可以通过索引来访问、设置、修改对应的数组元素，我们可以通过“数组名[索引]”的形式来获取数组中的元素

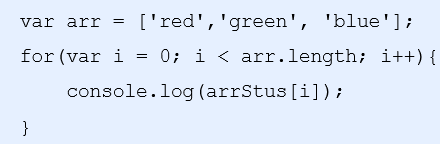


8.4遍历数组

遍历: 就是把数组中的每个元素从头到尾都访问一次

使用“数组名.length”可以访问数组元素的数量（数组长度）

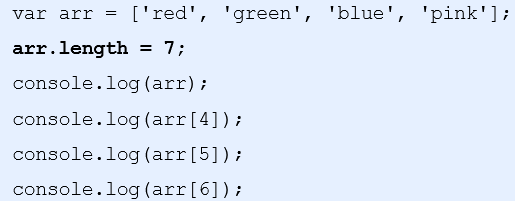
注意：此处数组的长度是数组元素的个数



8.5数组中新增元素

1、通过修改length 长度新增数组元素

可以通过修改 length 长度来实现数组扩容的目的，length 属性是可读写的



其中索引号是 4，5，6 的空间没有给值，就是声明变量未给值，默认值就是 undefined

2、通过修改数组索引新增数组元素

1）可以通过修改数组索引的方式追加数组元素

2）不能直接给数组名赋值，否则会覆盖掉以前的数据

9.JS函数

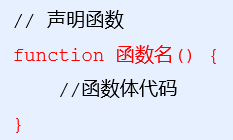
9.1函数的概念

函数：就是封装了一段可被重复调用执行的代码块。通过此代码块可以实现大量代码的重复使用

9.2函数的使用

函数在使用时分为两步：声明函数和调用函数

1、声明函数

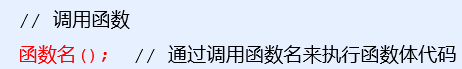


注意：

1）function 是声明函数的关键字,必须小写

2）由于函数一般是为了实现某个功能才定义的，所以通常我们将函数名命名为动词

2、调用函数



注意：声明函数本身并不会执行代码，只有调用函数时才会执行函数体代码

函数的封装是把一个或者多个功能通过函数的方式封装起来，对外只提供一个简单的函数接口

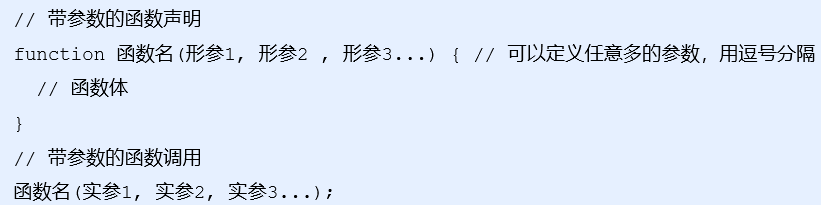
9.3函数的参数

1、形参和实参

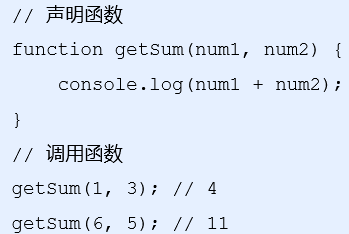
在声明函数时，可以在函数名称后面的小括号中添加一些参数，这些参数被称为形参，而在调用该函数时，同样也需要传递相应的参数，这些参数被称为实参



参数的作用：在函数内部某些值不能固定，我们可以通过参数在调用函数时传递不同的值进去



2、函数参数的传递过程



调用的时候实参值是传递给形参的

3、函数形参和实参个数不匹配问题

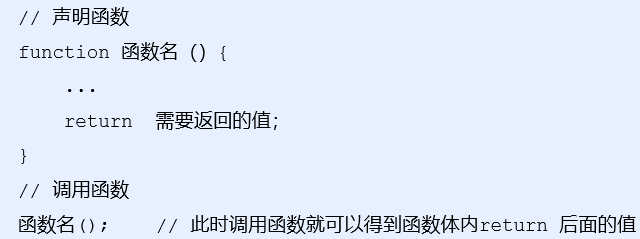


注意：在JavaScript中，形参的默认值是undefined

9.4函数的返回值

1、return语句

语法格式：



注意：

1）在使用return语句时，函数会停止执行，并返回指定的值

2）如果函数没有return，返回的值是undefined

3）return 只能返回一个值。如果用逗号隔开多个值，以最后一个为准

4）函数都是有返回值的

5）函数可以调用另外一个函数

2、break,continue,return的区别

break ：结束当前的循环体（如 for、while）

continue ：跳出本次循环，继续执行下次循环（如 for、while）

return ：不仅可以退出循环，还能够返回 return 语句中的值，同时还可以结束当前的函数体内的代码

9.5arguments的使用

当我们不确定有多少个参数传递的时候，可以用 arguments 来获取。在 JavaScript 中，arguments 实际上它是当前函数的一个内置对象。所有函数都内置了一个 arguments 对象，arguments 对象中存储了传递的所有实参

arguments展示形式是一个伪数组，因此可以进行遍历。伪数组具有以下特点：

1）具有 length 属性

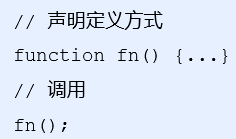
2）按索引方式储存数据

3）不具有数组的 push , pop 等方法

9.6函数的两种声明方式

1. 自定义函数方式(命名函数)

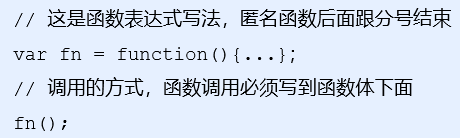
利用函数关键字 function 自定义函数方式



因为有名字，所以也被称为命名函数

注意：调用函数的代码既可以放到声明函数的前面，也可以放在声明函数的后面

2、函数表达式方式(匿名函数）



因为函数没有名字，所以也被称为匿名函数

10.JS作用域

10.1作用域概述

通常来说，一段程序代码中所用到的名字并不总是有效和可用的，而限定这个名字的可用性的代码范围就是这个名字的作用域。作用域的使用提高了程序逻辑的局部性，增强了程序的可靠性，减少了名字冲突

JavaScript（es6前）中的作用域有两种：

1）全局作用域

作用于所有代码执行的环境(整个 script 标签内部)或者一个独立的 js 文件

2）局部作用域（函数作用域）

作用于函数内的代码环境，就是局部作用域。因为跟函数有关系，所以也称为函数作用域

注意：JS 没有块级作用域

10.2变量的作用域

在JavaScript中，根据作用域的不同，变量可以分为两种：全局变量、局部变量

1、全局变量

在全局作用域下声明的变量叫做全局变量（在函数外部定义的变量）

注意：

1）全局变量在代码的任何位置都可以使用

2）在全局作用域下 var 声明的变量 是全局变量

3）特殊情况下，在函数内不使用 var 声明的变量也是全局变量（不建议使用）

2、局部变量

在局部作用域下声明的变量叫做局部变量（在函数内部定义的变量）

注意：

1）局部变量只能在该函数内部使用

2）在函数内部 var 声明的变量是局部变量

3）函数的形参实际上就是局部变量

3、全局变量和局部变量的区别

全局变量：在任何一个地方都可以使用，只有在浏览器关闭时才会被销毁，因此比较占内存

局部变量：只在函数内部使用，当其所在的代码块被执行时，会被初始化；当代码块运行结束后，就会被销毁，因此更节省内存空间

10.3作用域链

根据在内部函数可以访问外部函数变量的这种机制，用链式查找决定哪些数据能被内部函数访问，就称作作用域链

作用域链：采取就近原则的方式来查找变量最终的值

11.JS预解析

11.1预解析

JavaScript 代码是由浏览器中的 JavaScript 解析器来执行的。JavaScript 解析器在运行 JavaScript 代码的时候分为两步：预解析和代码执行

预解析：在当前作用域下, JS 代码执行之前，浏览器会默认把带有 var 和 function 声明的变量在内存中进行提前声明或者定义

代码执行： 从上到下执行JS语句

预解析只会发生在通过 var 定义的变量和 function 上。学习预解析能够让我们知道为什么在变量声明之前访问变量的值是 undefined，为什么在函数声明之前就可以调用函数

11.2变量预解析和函数预解析

预解析也叫做变量、函数提升

1、变量预解析（变量提升）

变量提升：变量的声明会被提升到当前作用域的最上面，变量的赋值不会提升

2、函数预解析（函数提升）

函数提升：函数的声明会被提升到当前作用域的最上面，但是不会调用函数

12.JS对象

12.1对象

在JavaScript中，对象是一组无序的相关属性和方法的集合，所有的事物都是对象，例如字符串、数值、数组、函数等

对象是由属性和方法组成的：

1）属性：事物的特征，在对象中用属性来表示（常用名词）

2）方法：事物的行为，在对象中用方法来表示（常用动词）

12.2创建对象的三种方式

在JavaScript中，现阶段我们可以采用三种方式创建对象（object）：

1）利用字面量创建对象

2）利用new Object创建对象

3）利用构造函数创建对象

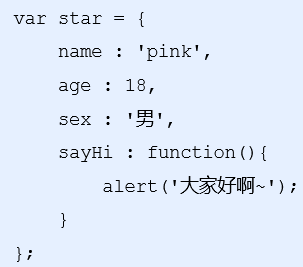
1、利用字面量创建对象

对象字面量：就是花括号 { } 里面包含了表达这个具体事物（对象）的属性和方法

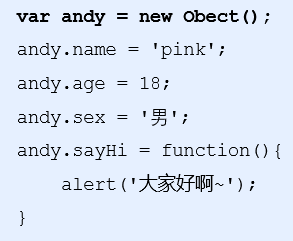
{ } 里面采取键值对的形式表示

键：相当于属性名

值：相当于属性值，可以是任意类型的值(数字类型、字符串类型、布尔类型，函数类型等)



2、利用new Object创建对象



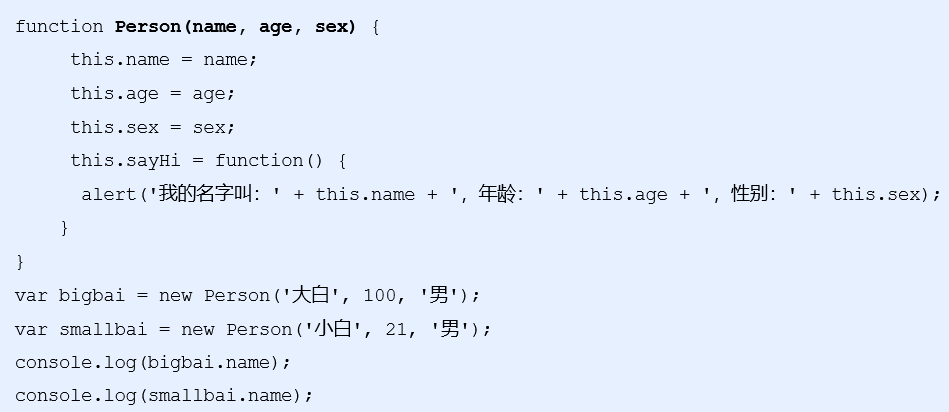
3、利用构造函数创建对象

构造函数 ：是一种特殊的函数，主要用来初始化对象，即为对象成员变量赋初始值，它总与 new 运算符一起使用。我们可以把对象中一些公共的属性和方法抽取出来，然后封装到这个函数里面

在 js 中，使用构造函数要时要注意以下两点：

1）构造函数用于创建某一类对象，其首字母要大写

2）构造函数要和 new 一起使用才有意义



注意：

1.构造函数约定首字母大写。

2.函数内的属性和方法前面需要添加 this ，表示当前对象的属性和方法。

3.构造函数中不需要 return 返回结果。

4.当我们创建对象的时候，必须用 new 来调用构造函数

12.3对象的调用

1）对象里面的属性调用：对象.属性名 ，这个小点 . 就理解为“ 的 ”

2）对象里面属性的另一种调用方式：对象[‘属性名’]，注意方括号里面的属性必须加引号

3）对象里面的方法调用：对象.方法名() ，注意这个方法名字后面一定加括号

12.4new关键字

new 在执行时会做四件事情：

1.在内存中创建一个新的空对象

2.让 this 指向这个新的对象

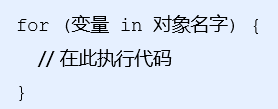
3.执行构造函数里面的代码，给这个新对象添加属性和方法

4.返回这个新对象（所以构造函数里面不需要return）

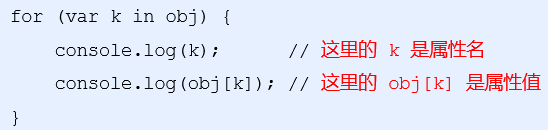
12.5遍历对象属性

for...in 语句用于对数组或者对象的属性进行循环操作

语法结构：



语法中的变量是自定义的，它需要符合命名规范，通常我们会将这个变量写为k或者key



13. JS内置对象

13.1内置对象

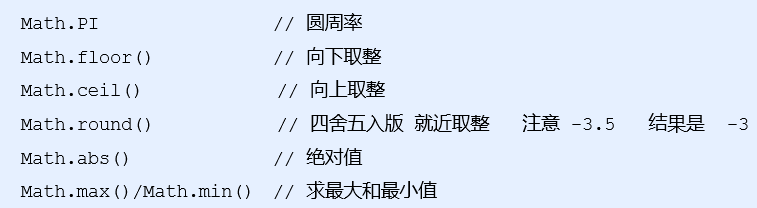
JavaScript中的对象分为3种：自定义对象、内置对象、浏览器对象

内置对象就是指 JS 语言自带的一些对象，这些对象供开发者使用，并提供了一些常用的或是最基本而必要的功能（属性和方法）

JavaScript 提供了多个内置对象：Math、Date、Array、String等

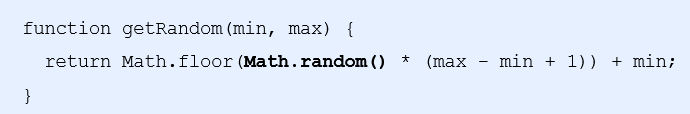
13.2Math对象

Math 对象不是构造函数，它具有数学常数和函数的属性和方法。跟数学相关的运算（求绝对值，取整、最大值等）可以使用 Math 中的成员



随机数方法 random()：可以随机返回一个小数，其取值范围是 [0，1)，左闭右开

得到一个两数之间的随机整数，包括两个数在内：



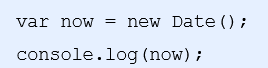
13.3日期对象

Date 对象和 Math 对象不一样，他是一个构造函数，所以我们需要实例化后才能使用

Date 实例用来处理日期和时间

1、Date()方法的使用：

1）获取当前时间必须实例化



2）Date() 构造函数的参数

如果括号里面有时间，就返回参数里面的时间。例如日期格式字符串为‘2019-5-1’，可以写成new Date('2019-5-1') 或者 new Date('2019/5/1')

注意：如果Date()不写参数，就返回当前时间。如果Date()里面写参数，就返回括号里面输入的时间

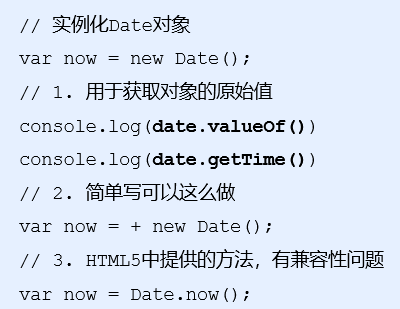
2、日期格式化



3、获取日期的总毫秒形式

Date 对象是基于1970年1月1日（世界标准时间）起的毫秒数

我们经常利用总的毫秒数来计算时间，因为它更精确

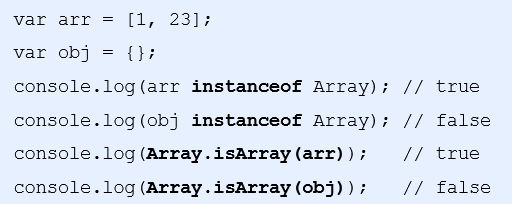


13.4数组对象

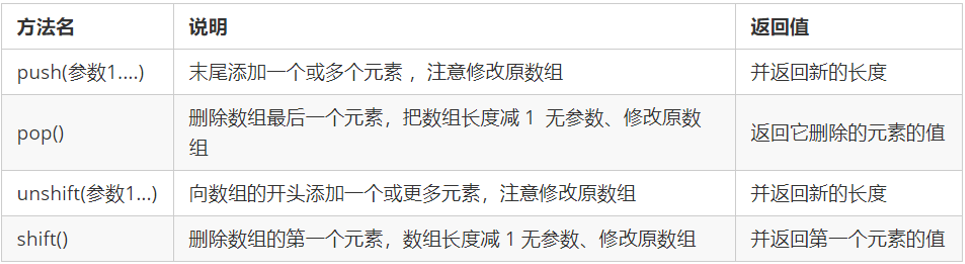
1、检测是否为数组

instanceof 运算符，可以判断一个对象是否属于某种类型

Array.isArray()用于判断一个对象是否为数组，isArray() 是 HTML5 中提供的方法

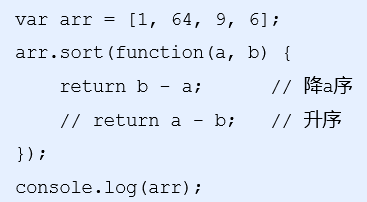


2、添加/删除数组元素



3、数组排序

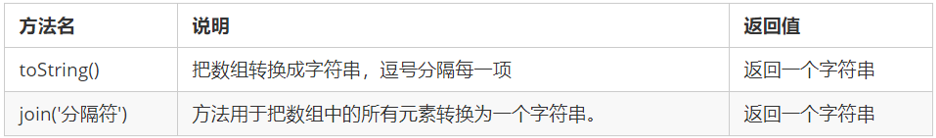




4、数组索引的方法



5、数组转换为字符串



6、其他功能



13.5字符串对象

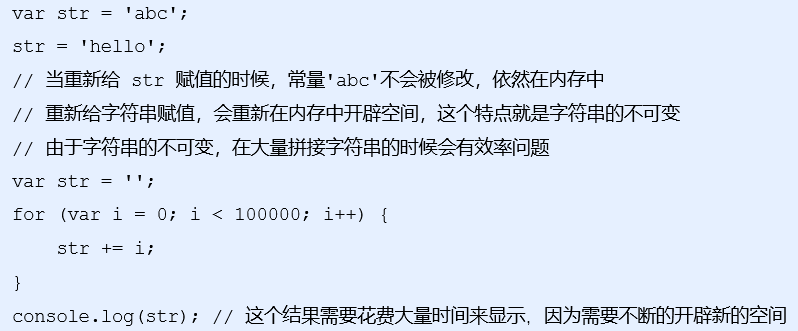
1、基本包装类型

为了方便操作基本数据类型，JavaScript 还提供了三个特殊的引用类型：String、Number和 Boolean

基本包装类型就是把简单数据类型包装成为复杂数据类型，这样基本数据类型就有了属性和方法

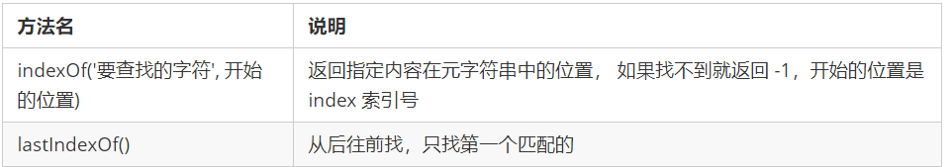
2、字符串的不可变

指的是里面的值不可变，虽然看上去可以改变内容，但其实是地址变了，内存中新开辟了一个内存空间



3、根据字符返回位置

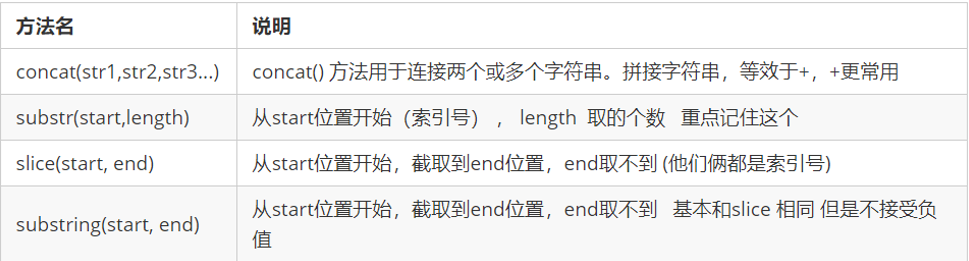
字符串所有的方法，都不会修改字符串本身(字符串是不可变的)，操作完成会返回一个新的字符串



4、根据位置返回字符



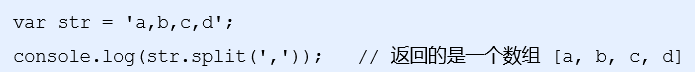
5、字符串操作方法



replace() 方法用于在字符串中用一些字符替换另一些字符，语法结构如下：



split()方法用于切分字符串，它可以将字符串切分为数组。在切分完毕之后，返回的是一个新数组



toUpperCase() //转换大写

toLowerCase() //转换小写

14.JS简单类型和复杂类型

14.1概述

简单类型又叫做基本数据类型或者值类型，复杂类型又叫做引用类型

值类型：简单数据类型/基本数据类型，在存储时变量中存储的是值本身，因此叫做值类型

string ，number，boolean，undefined，null

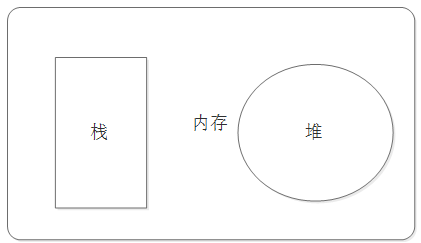
引用类型：复杂数据类型，在存储时变量中存储的仅仅是地址（引用），因此叫做引用数据类型，通过 new 关键字创建的对象（系统对象、自定义对象），如 Object、Array、Date等

14.2栈和堆

堆栈空间分配区别：

1、栈（操作系统）：由操作系统自动分配释放存放函数的参数值、局部变量的值等。其操作方式类似于数据结构中的栈。简单数据类型存放到栈里面

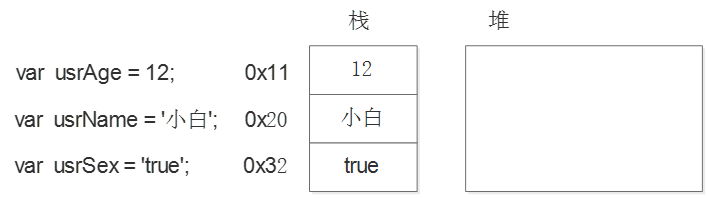
2、堆（操作系统）：存储复杂类型(对象)，一般由程序员分配释放，若程序员不释放，由垃圾回收机制回收。复杂数据类型存放到堆里面



注意：JavaScript中没有堆栈的概念，通过堆栈的方式，可以让大家更容易理解代码的一些执行方式，便于将来学习其他语言

14.3简单类型的内存分配

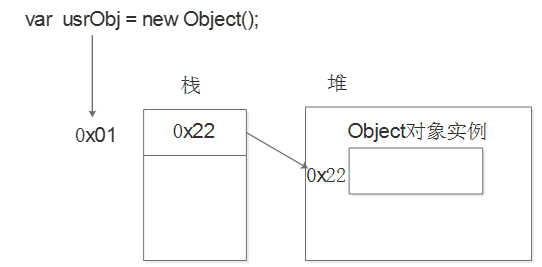
值类型（简单数据类型）： string ，number，boolean，undefined，null



14.4复杂类型的内存分配

引用类型（复杂数据类型）：通过 new 关键字创建的对象（系统对象、自定义对象），如 Object、Array、Date等

引用类型变量（栈空间）里存放的是地址，真正的对象实例存放在堆空间中



14.5简单类型传参

函数的形参也可以看做是一个变量，当我们把一个值类型变量作为参数传给函数的形参时，其实是把变量在栈空间里的值复制了一份给形参，那么在方法内部对形参做任何修改，都不会影响到的外部变量

14.6复杂类型传参

函数的形参也可以看做是一个变量，当我们把引用类型变量传给形参时，其实是把变量在栈空间里保存的堆地址复制给了形参，形参和实参其实保存的是同一个堆地址，所以操作的是同一个对象