数据类型、运算符、流程控制的理解

目录：

1. javascript的组成

2. 数据类型

2.1 数据类型

2.2 typeof类型

3.运算符:

3.1 功能

3.2 处理字符个数

4.运算符优先级

5. 流程控制

**1. javascript的组成**

    Javascript不是一门单体的语言，他是由很多不同的部分组合而成，其中 ECMAScript 这门语言，构成js基础

    ECMAScript：

        DOM(\*\*对文档操作\*\*)

        BOm(\*\*对浏览器的操作\*\*)

        音频、视频、文件等

    ECMAScript 定义了基本语言的特性:

         数据类型

         语法结构

        流程控制

         内置的常用数据的操作

2.程序定义：数据有类型，不同的类型操作会得到不同的结果

**2数据类型**

2.1 按存储方式分值类型和引用类型

值类型：字符串，数组，布尔值

引用类型：对象，函数，数组

**数字：**

        由0-9的数字、NaN、Infinity

        数字有取值范围，超出由+Infinity/-Infinity来表示

        把两个不可能做数学运算的数据做了数学运算，js用 NaN 去表示

**字符串：**

       由0个或多个字符（\*\*文字符号\*\*）组成的串

        由一对 单引号 或 双引号 包含起来，单引号和双引号在一个字符串中必须是对应的,同时也具备\*\*长度\*\*

        特殊字符 - 转义字符:在字符串有一些字符串不方便使用符号去表示，那么我们可以使用转义字符('\')去描述他

**布尔：**

        用来表示逻辑是是与非

        该类型只有两个值：true false

**Null：**

        Null 类型有且只有一个值，称为 null

        如果我们申明了一个变量，但是暂时没有初始值，如果该变量我们准备存储一个对象，那么默认的时候，我们通常会赋值给这个变量 null。

**Undefined:**

        Undefined 类型有且只有一个值，称为 undefined. 任何没有被赋值的变量都有 undefined 值

        在js中，如果一个变量被创建了，但是没有赋值，那么该变量默认值就是undefined

**Symbol**

        表示唯一值

**Object**

        复合型数据，组合型数据；

        通过一对 {} 来表示一个对象，一个对象中可以包含N个数据， 每一个数据都有一个名称与其对应，我们称这样的结构为： 键值对，每一组键值对使用 冒号 来表示，多个键值对使用 逗号 来分隔. \*\*注意：最后一条数据不要加逗号\*\*，我们通常又把对象中的数据称为对象的属性。

            非函数：属性

            函数：方法

            对象下的属性/方法，通过.或[]来访问。

**7 大类型:**

-  Number

-  String

-  Boolean

-  Null

-  Undefined

-  Symbol

-  Object

**typeof 7 大类型:**

-  number

-  strin

-  boolean

-  object

-  Undefined

-  Symbol

-  function

>\*\*function并不属于7大类型中的，在类型定义中function属于object，但是function在js中地位特别的高（称为：一等公民），所以typeof function的结果是function\*\*

**3.运算符:**

**3.1 按功能分**

**一元运算符**

    +,-,typeof

                +,- : 得到Number类型的值

               typeof : 得到字符串类型的值

new：对象创建运算符

()：函数执行运算符

**二元运算符**

**三元运算符/三目运算符**：判断执行

                条件？条件满足时候执行：条件不满足时候执行

**3.2 按运算字符个数分**

**算术运算符**

       +,-,\*,/,%

        除了 + 以外的都返回的是Number类型；

        当 + 的左侧或者右侧，只要有一侧的数据为字符串，那么 + 就是字符串拼接；

        当 + 为字符串连接符号，如果有一侧的数据为非字符串，那么就会在内部转成该数据的字符串形式。

        undefined => 'undefined'

        null => 'null'

        true => 'true'

        false => 'false'

        1 => '1'

        object => '根据对象的不同结果有所不一样'

        console.log( undefined + undefined );//NAN

        console.log( null + null );//0

**关系运算符**

         \>,< \==,=\==,>=,<=,!=,!==

                 所有关系运算符得到的结果：布尔值

                == 比较值;

        === 比较类型，然后比较值;

***console***.log( **true** == 10 ); *//false*

***console***.log(**'5'** == 5); *//true*

***console***.log(**null** == **undefined**); *//true*

***console***.log( **true** === 1 ); *//false*

true==[] //true

先把true转成1，然后和数字比较；number去转[]，得到结果也是0。0==0

**逻辑运算符**

        与或非

        && ||  !

              && || : 得到的结果依据比较的值

                    && ：当左侧的值为真，返回右侧的值，当左侧为假，返回左侧的值

                     || ：当左侧的值为真，返回左侧的值，当左侧为假，返回右侧的值

              ! : 布尔值 ，对值进行布尔类型转换，取反值

                        注意：任意数据转布尔值的技巧

                        Undefined : false

            Null : false

            Number : 0为false，非0为true

            String : 空字符串为false，非空为true

            Object : 对象永远为真，即使空的

**赋值运算符**：=

        把右侧的值赋值给左侧，优先级最低

        包括：+=、-=、\*=、/=、%=

        - +：在JS中不只是数学运算，他还有字符串拼接的作用，所以 a+b 不能完全等于 b+a

**逗号运算符**

**位运算符**

**4.运算符优先级**

        <https://baike.baidu.com/item/%E8%BF%90%E7%AE%97%E7%AC%A6%E4%BC%98%E5%85%88%E7%BA%A7>

        !,(),+,\*,/,>,||,&&,<

        false || false && true

        false || false

        false

**5.流程控制：**

        if（条 件）{

                满足条件执行的代码块

        }else{

                不满足条件执行的代码块}

        while：条件初始值 while（条件判断）{

                当条件为真执行的代码块；条件变化}

for（条件初始值；条件判断；条件变化）{

                当条件为真的时候执行的代码块}

        for（var 变量 in 对象）{

                自动遍历对象，把对象的属性方法名称依次赋值给in前面的变量，然后执行括号的代码}

        switch：分支判断

                switch(变量){

                        case 值1：

                                如果变量的值与case 值相等，则执行；

                        break；

                        case 值2：

                                如果变量的值与case 值相等，则执行；

                        break；

                        default：

                                以上都不满足，则执行；

                        break；

                        }

                  判断条件变量只有一个且判断的是等值```

                  如果没有break，那么case 将从满足最开始的那个条件的位置一直向下执行，直到switch结束或者碰到break为止，这种现象叫做穿透

**break**也可以在循环中使用

        如：for (var i=0; i<10; i+=1) {

                console.log(i);

                if (i == 5) {

                break;  //终止循环

                }

            }

        zmouse:for (var i=0; i<10; i+=1) {

                console.log(i);

                for (var j=0; j<10; j+=1) {

                    console.log(j);

                    if (i==5 && j==5) {

                        //跳出当````前循环

                        // break;

                        break zmouse;

                    }

                }

            }

        这样就可以跳出里面循环，终止外面的循环

**continue**

    - 终止当前循环，进入下一次