Ctrl+r 运行js

Document 一个页面就是一个文档document

三个输出语句

Alert（‘1’）

Document.write(‘1’)

Console.log(‘1’)

Js代码编写位置

虽然可以写在属性中，但是他们属于结构与行为耦合，不方便维护，不推荐使用

Js代码编写在标签onclick属性中

<button onclick=”alert(‘你点我干嘛’)”>按钮</button>

Js代码写在a标签的href属性中

<a href=”javascript: alert(‘让你点一下’)”></a>

Js代码写在script标签中

可以将js代码编写在外部js文件，然后通过script标签引入，写到外部文件中在不同的页面中同时使用，也可以利用到浏览器的缓存机制，推荐使用的方式

Script标签一旦引入外部文件了，就不能再标签中编写代码了，即使写了也会被忽略，如果需要内部写js代码，需要新建一个script

Js代码是按照从上到下的顺序一行一行执行的

在js中，如果需要表示16进制的数字，则需要以0x开头

如果需要表示八进制的数字，则需要以0开头

如果要表示二进制数字，则需要以0b开头，但是不是所有的浏览器都有支持。

像“070”这种字符串，有些浏览器会当成8进制解析，有些会当成10进制解析，parseInt(“070”, 10) 这里的10是进制。

**基本语法**

多行注释：/\*\*/

当行注释： //

Js中严格区分大小写

Js中每一条语句以；（英文）结尾，如果不写分号，浏览器会自动添加，但是会消耗一些系统资源（影响性能），有些时候，浏览器会加错分号，所以在开发中分号必须写。

Js中会忽略多个空格和换行，所以我们可以利用空格和换行对代码进行格式化。

**字面量**，都是一些不可改变的量，字面量都是可以直接使用的，但是我们一般都不会直接使用字面量，eg：1 、2、 3、 4、 5

**变量，**可以用来保存字面量，而且变量的值是可以任意改变的，变量更加方便我们使用

声明变量，可以通过变量对字面量进行描述，在js中使用var关键字来声明一个变量，eg：var a;

**标识符**，在js中所有的可以由我们自主命名的都可以称为标识符，eg：变量名、函数名、属性名都属于标识符，命名一个标识符时需要遵守如下规则：

1. 标识符中可以含有字母、数字、下划线、$；
2. 标识符不能以数字开头；
3. 标识符不能是ES中关键字或者保留字；
4. 标识符一般都采用驼峰命名法
5. Js底层保存标识符时实际上是采用的Unicode编码，所以理论上讲，所有的utf-8中含有的内容都可以作为标识符

**数据类型**

数据类型指的是字面量类型，在js中一共有六种数据类型：

可以使用一个运算符typeof来检查一个变量的类型，语法：typeof 变量console.log(typeof a)

1. String: 字符串，在js中需要使用引号引起来（单双引号都行，但是不能混用，引号不能嵌套）；

如果必须使用单/双引号，在字符串中我们可以使用\作为转义字符，当表示一些特殊字符时可以使用\进行转义

\” 表示“

\’ 表示‘

\n 表示换行

\t 表示制表符

\\ 表示\

1. Number：数值，在js中所有的数值都是Number类型，包括整数和浮点数（小数），js中可以表示数字的最大值 Number.MAX\_VALUE // 1.7976931348623157e+308

Js中可以表示数字的最小值Number.MIN\_VALUE //5e-324

如果使用Number表示的数字超过了最大值，则会返回一个infinity

infinity，表示正无穷，

-infinity，表示负无穷，

infinity是一个字面量，使用typeof检查infinity也会返回Number

NaN是一个特殊的数字，表示Not a Number，也是一个字面量，使用typeof检查NaN时返回number。

在js中整数的运算基本可以保证准确，如果使用js进行浮点数运算，可能得到一个不精确的结果，所以千万不要使用js进行对精确度比较高的运算。

1. object：对象
2. boolean：布尔值，布尔值只有两个true和false，true表示逻辑上的真，false表示逻辑上的假，布尔值主要做逻辑判断的，使用typeof检测布尔值时，会返回boolean
3. unbefined：未定义，类型的值只有一个undefined，当声明一个变量但是并不给变量赋值时，它的值就是undefined，使用typeof检测undefined时，返回一个undefined
4. null：空值，类型的值只有一个null，专门用来表示为空的对象，使用typeof检查null时，返回一个object。

其中string、number、boolean、undefined、null属于基本数据类型

而object属于引用数据类型

**强制的类型转换**，只将一个数据类型强制转换成其他的数据类型。类型转换主要指将其他的数据类型转换成String Number Boolean

1. 将其他类型转换为String类型

方法一：调用被转换数据类型的toString()方法，该方法不会影响到原变量，它会将转换的结果返回的。但是要注意，null和undefined这两个值没有toString()方法，如果调用他们的方法会报错。

方式二：调用String()函数，并将被转换的数据作为参数传递给函数，使用String()函数做强制类型转换时，对于Number和Boolean实际上就是调用的toString()方法，但是对于null 和undefined，就不会调用toString()方法，它会将null和undefined直接转换为“null”和“undefined”

1. 将其他的数据类型转换为Number类型

方法一：直接使用Number()函数

字符串转数字：

* 如果是纯数字字符串，则直接将其转换为数字。
* 如果字符转中有非数字内容，则转换为NaN。
* 如果字符中是一个空串或者是一个全是空格的字符串，则转换为0

布尔值转数字：true转成1，false转成0

null转换成数字结果为0。

undefined转换成数字结果为NaN

方法二：这种方式专门用来对付字符串。

parseInt() 把一个字符串转换为一个整数，可以将一个字符串中的整数内容取出来，然后转换为number

parseFloat() 把一个字符串转换为浮点数，作用和parseInt()类似，不同的是它可以获得有效的小数。

如果对非String使用parseInt()或者parseFloat()它会先将其转换为string然后在操作。

1. 将其他的数据类型转换为boolean。

使用boolean()函数，数字转布尔，除了0和NaN，其余都是true

字符串转布尔，除了空串，其余的都是true

null和undefined的都会转换为false

对象也会转换为true

为任意的数据类型做两次非运算，即可将其转换为布尔值。

**运算符也叫操作符**，通过运算符可以对一个或多个值进行运算，并获取运算结果。比如typeof就是一个运算符，可以用来获取一个值的类型，它会将该值的类型以字符串的形式返回。

算数运算符：

当对非number类型的值进行运算时，它会将这些值转化为number然后在运算，任何值和NaN做运算结果都为NaN

* +：可以对两个值进行加法运算，并将结果返回，如果对两个字符串进行加法运算，则会做拼串操作，会将两个字符串拼接成一个字符串，并返回。任何的值和字符串做加法运算，都会先转换为字符串，然后在拼串。可以利用这一特点，将任意的数据类型转换为string，只需要将任意的数据类型+””即可将其转换为字符串，这是一种隐式的类型转换，由浏览器自行完成，实际上他也是执行的String()
* –：可以对两个值进行减法运算，并将结果返回。
* \*：可以对两个值进行乘法运算
* /：可以对两个值进行除法运算
* %：取模运算（取余）

任何值做- \* /运算时都会自动转换为Number，可以利用这一特点，做隐式的类型转换，可以通过一个值-0、\*1、/1来将其转换为Number，原理和Number()一样，使用起来更加简单

**一元运算符**，只需要一个操作数

1. + 正号，不会对数值产生任何影响。
2. – 负号，负号可以对数字进行负号的取反。
3. 于非number值，会先转换为number在进行取反，可以对一个其他的数据类型使用+，来将其转换为number，它的原理和Number()函数一样，

**自增（++）**

1. 通过自增可以使变量在自身的基础上增加1
2. 对于一个变量自增以后，原变量的值会立即自增1
3. 自增分成两种，++a和a++，无论是a++还是++a都会立即使原变量的值自增1

不同的是a++和++a的值不同，

a++的值等于原变量的值（自增前的值）

++a的值等于原变量的新值（自增后的值）

**自减（--）**

1. 通过自减可以使变量在自身的基础上减1；
2. 自减分成两种，a-- 和 --a，无论是a-- 还是 --a都会使原变量的值自减1，不同的是a-- 和 --a的值不同。

a-- 是变量的原值（自减前的值）

--a 是变量的新值（自减后的值）

**逻辑运算符**，JS中提供了三种逻辑运算符

1. **！** 非，可以用来对一个值进行非运算，所谓非运算就是对一个布尔值进行取反操作就是true变false，false变true，如果对一个值进行两次取反，它不会变化，如果对非布尔值进行运算，则会将其转换为布尔值，在进行运算。所以可以利用该特点，来将一个其他的数据类型转换成布尔值。可以为任意一个数据类型取两次反，来将其转换为布尔值，原理和boolean()函数一样。
2. **&&** 与，可以对符号两侧的值进行与运算，并返回结果。

运算规则：两个值中，只要有一个值为false就返回false，只有两个值都为true时，才会返回true。如果两端两个值都是true，则返回true，只要有一个false就返回false。Js中的‘与’属于短路的与，如果第一个值为false，则不会看第二个值。第一个值为true时，会检查第二个值，第一个值为false时，则不会检查第二个值。

1. **||** 或，可以对符号两侧的值进行或运算，并返回结果。

运算规则：两个值中只要有一个true，就返回true，如果两个值都为false，就返回false。Js中的“或”属于短路的或，如果第一个值为true时，则不会检查第二个值，第一个值为false时，则会检查第二个值。

相等运算符，比较两个值是否相等。如果相等会返回true，否则会返回false。使用==来做相等运算。当使用==来比较两个值时，如果值的类型不同，则会自动进行类型转化，将其转换为相同的类型，然后在进行比较。console.log(‘1’ == 1)

undefined衍生自null，所以这两个值做相等判断是会返回true。

NaN不和任何值相等包括它本身。

可以通过isNaN()函数来判断一个值是否是NaN。如果该值是NaN，则返回true，否则返回false。

不相等 != 用来判断两个值是否不相等，如果不相等则返回true，否则返回false。

!= 来判断，不相等也会对变量自动进行类型转换，如果转换后相等也会返回false。

=== 全等于，用来判断两个值是否全等，它和相等类似，不同的是它不会做自动类型转换，如果两个值的类型不同，直接返回false

!== 不全等，用来判断两个值是否不全等，和不等类似，不同的是它不会做自动的类型转换，如果两个值的类型不同直接返回true。

**条件运算符**，也叫三元运算符。

语法：

1. 条件表达式? 语句一 : 语句二;

执行流程：

1. 条件运算符在执行时，首先对条件表达式求值，如果该值为true，则执行语句一，并返回结果，如果该值为false，则执行语句二，并返回结果。
2. 如果条件的表达式的求值结果是一个非布尔值，会将其转换为布尔值，然后在运算。

**逗号运算符**“ , 运算符”使用，可以分割多个语句，一般可以声明多个变量时使用，使用,运算符，可以同时声明多个变量并赋值。

**运算符的优先级**，和数学中一样，在js中运算符也有优先级，比如：先乘除后加减。

在js中有一个运算符优先级的表，在表中越靠上的优先级越高，优先级越高越优先计算，如果优先级一样，则从左往右计算。但是这个表不需要记，如果遇到优先级不清楚，可以使用括号来改变优先级。

var result = 1 || 2 && 3;

如果&&的有限极高，则应该返回1

**语句**：我的程序是由一条一条语句构成的。语句是按照自上向下的顺序一条一条的执行，在js中可以使用{ }来为语句进行分组。同一个{}中的语句，我们称为是一组语句，他们要么都执行，要么都不执行，{}中的语句，也称为一个代码块。在代码块后不用再写分号。JS中的代码块只具有分组的作用，没有其他的用途。代码块内部的内容，在外部是完全可见的。

**流程控制语句**，JS中的程序是从上到下一行一行的执行的，通过流程控制语句可以控制程序执行流程，使程序可以根据一定的条件来选择执行。

语句分类：

1. 条件判断语句：使用条件判断语句可以在执行某个语句之前进行判断，如果条件成立才会执行语句，条件不成立则语句不执行。
2. if语句：

语法一：

if（条件表达式）{

语句…（表示可以写多条语句）

}，if语句在执行时，会先对条件表达式进行求值判断，如果条件表达式的值为true，则执行if后的语句，如果条件表达式的值为false，则不会执行if后的语句。If语句只能控制紧随其后的那个语句。如果希望if语句可以控制多条语句，可以将这些语句统一放到代码块中。If语句后的代码块不是必须的，但是在开发中尽量写上代码块，即使if后只有一条语句。

语法二：

if(条件表达式) {

语句….

} else {

语句…

}

if…else…语句：当该语句执行时，会先对if后的条件表达式进行求值判断，如果该值为true，则执行if后的语句，如果该值为false，则执行else的语句。

语法三：

if (条件表达式) {

语句…

} else if (条件表达式) {

语句…

} else if (条件表达式) {

语句…

} else {

语句…

}

if…else if … else…语句：当该语句执行时，会从上到下依次对条件表达式进行求值判断，如果值为true则执行当前语句，如果值为false则继续向下判断，如果所有的条件都不满足，则执行最后一个else语句。该语句中，只有会一个代码块被执行，一旦代码块执行，则直接结束语句。

prompt（）可以弹出一个提示框，该提示框中会带有一个文本框，用户可以在文本框中输入一段内容，该函数需要一个字符串作为参数，该字符串将会作为提示框的提示文字。用户输入的内容将会作为函数的返回值返回，可以定义一个变量来接收该内容。返回值是string类型。

1. 条件分支语句：也叫switch语句

语法：

1. switch(条件表达式) {

case 表达式:

语句…

break;

case 表达式:

语句…

break;

case 表达式:

语句…

break;

case 表达式:

语句…

break;

case 表达式:

语句…

break;

default:

语句…

break；

}

1. 执行流程：

switch…case…语句

在执行时会依次将case后的表达式的值和switch后边条件表达式进行全等比较。

如果比较结果为true，则从当前case处开始执行代码，当前case的所有的代码都会执行，可以在case后边跟着一个break关键字，这样可以确保只会执行当前case后的语句，而不会执行其他的case。

如果比较为false，则继续向下比较。

如果所有的比较结果都为false，则只执行default后的语句

1. switch语句和if语句的功能实际上是有重复的，使用switch可以实现if的功能，同样使用if也可以实现switch的功能，所以我们在使用时，可以根据自己的习惯来使用。
2. 循环语句：通过循环语句可以反复的执行一段代码多次
   1. while循环

语法：

while(条件表达式){

语句… (循环体)

}

While语句在执行时：

先对条件表达式进行求值判断，如果值为true，则继续执行循环体，以此类推如果值为false，则终止循环。

创建一个循环，往往需要三个步骤：

1. 创建初始化一个变量 var I = 0;
2. 在循环中设置一个条件表达式 while(i<10){}
3. 定义一个更新表达式，每次更新初始化变量
   1. do…while…循环

语法：

do{

语句…

}while (条件表达式)

执行流程：

do…while语句在执行时，会先执行循环体，循环体执行完毕后，在对while后的表达式进行判断，如果结果为true，则继续执行循环体，执行完毕继续判断以此类推，如果结果为false，则终止循环。

实际上这个两个语句功能类似，不同的是while是先判断在执行，而do…while…是先执行后判断。do…while可以保证循环体至少执行一次，而while不能。

* 1. for语句，也是一个循环语句，也称为for循环，在for循环中，为我们提供了专门的位置用来放三个表达式
     1. 初始化表达式
     2. 条件表达式
     3. 更新表达式

语法：

for(初始化表达式; 条件表达式; 更新表达式){

语句….

}

执行流程：

1. 执行初始化表达式，初始化变量(初始化表达式只执行一次)
2. 执行条件表达式来判断是否执行循坏

如果为true，则执行循坏体3)

如果为false，则终止循环

1. 第三步是花括号里的循环体
2. 执行更新表达式，更新表达式执行完毕继续重复2)

For循环中的三个部分都可以省略，也可以直接写在外部

如果在for循环中不写任何的表达式，只写两个“;”此时这个循环是一个死循环，会一直执行下去，慎用。

* 1. 嵌套的for循环

**break关键字**，可以用来退出switch或循环语句，不能在if语句中使用break。

break关键字会立即终止离他最近的那个循环语句。

如果是嵌套for循环，想在内部for循环中终止外部for循环，可以为for循环创建一个label标签，来标识当前的循坏（label: 循环语句）。使用break语句时，可以在break后跟着一个label，这样break将会终止指定的循环，而不是最近的。

**continue关键字**，可以用来跳过当次循环。同样continue也是默认只会对离他最近的循环起作用。

Math.sqrt(4) 可以通过Math.sqrt()对一个数进行开方。

回顾：数据类型有

String、number、boolean、null、undefined这五种类型属于基本数据类型，以后我们看到的值只要不是以上的五种，全都是对象。基本数据类型的不足：都是单一的值，值与值之间没有任何联系。如果使用基本数据类型的数据，我们所创建的变量都是独立的，不能成为一个整体。

Object对象

**Object对象**

**对象**，属于一种复合的数据类型，在对象中可以保存多个不同数据类型的属性。

对象分类：

1. 内建对象：由ES标准中定义的对象，在任何的ES的实现中都可以使用。

比如：Math、String、Number、Boolean、Function、Object…

1. 宿主对象：由JS的运行环境提供的对象，目前来讲主要指由浏览器提供的对象

比如：BOM（浏览器对象模型）、DOM（文档对象模型），console是对象，document是对象，这两个对象都是由浏览器提供的叫做宿主对象

1. 自定义对象：由开发人员自己创建的对象。

使用new关键字调用的函数，是构造函数constructor

构造函数是专门用来创建对象的函数

使用typeof检查一个对象时，会返回object

var obj = new Object();

在对象中保存的值称为属性

向对象中添加属性

语法：对象.属性名 = 属性值

obj.name = ‘孙悟空’;

obj.age = 18;

obj.gender = ‘男’;

读取对象中的属性

语法：对象.属性名

如果读取对象中没有的属性，不会报错的，而是会返回undefined。

console.log(obj.name); console.log(obj.gender); console.log(obj.age);

修改对象属性值

语法：对象.属性名 = 新值

obj.name = ‘大圣’;

删除对象属性

语法：delete 对象.属性

delete obj.age;

属性值：

js的属性值可以是任意的数据类型，甚至他也可以是一个对象。

In运算符，通过该运算符可以检查一个对象中是否含有指定的属性，如果有则返回true，如果没有则返回false。

语法： “属性名” in 对象。

**基本和引用数据类型**

基本数据类型

String 、number、boolean、null、undefined

引用数据类型

Object

Js中的变量都是保存到栈内存的（变量名和变量值），

基本数据类型的值直接在栈内存中存储，值与值之间是独立存在的，修改一个变量不会影响其他的变量。

对象是保存在堆内存中，每创建一个新的对象，就会在堆内存中开辟出一个新的空间，而变量保存的是对象的内存地址（内存引用），如果两个变量保存的是同一个对象的引用，当通过一个变量修改属性时，另一个也会受到影响。

当比较两个基本数据类型的值时，就是比较的值；

当比较两个引用数据类型时，它是比较的对象的内存地址，如果两个对象是一模一样，它也会返回false。

**对象字面量**，使用字面量来创建一个对象， var obj = {};

使用对象字面量，可以在创建对象时，直接指定对象中的属性；

语法： {属性名：属性值，属性名：属性值…}

对象字面量的属性名可以加引号也可以不加，建议不加。

如果要使用一些特殊的名字，必须要加引号。

属性名和属性值是一组一组的名值对结构，名和值之间使用冒号（:）连接，多个名值对之间使用逗号（,）隔开，如果一个属性之后没有其他的属性就不要写逗号（,）了。

**函数**

**函数**也是一个对象

函数中可以封装一些功能（代码），在需要时可以执行这些功能（代码）。