js 文件的使用（推荐）

- 创建 js 文件
- 将 js 文件引入页面

引用： <script src="js/main.js"></script>

注意：

- script 标签一旦用于引入外部文件了，就不能在编写代码
- 如果需要则可以在创建一个新的 script 标签用于编写内部代码

JS 中会忽略多个空格和换行

- 利用空格和换行对代码进行格式化

字面量

- 不可改变的值，就是字面的意思
  - 示例 ：1 2 3 4 5
- 字面量都是可以直接使用，通常都不会直接使用字面量

- 变量

- 变量可以用来保存字面量的值
- 变量的值是可以任意改变
- 变量更加方便我们使用，所以在开发中都是通过变量去保存一个字面量的值

- 声明变量

使用 var 关键字来声明变量

示例： var a;

显示出来 a ： console.log(a);

-赋值

给变量一个 字面量 的值

示例： a = 7;

- console.log(a);

-

声明并且赋值

var a = 0;

编码规范

变量名必须有语义，不允许写无意义的

标识符 （在 JS 中所有的可以由程序员自主命名）

例如：变量名、函数名、属性名都属于标识符

命名规则：（一般用小驼峰 helloWordHi）

1.标识符中可以含有 ** 字母、数字、_、$ **

2.标识符不能以数字开头

3.标识符不能是 ES 中的 关键字 或 保留字

- 关键字：js 系统内部已经使用过的标识符， 例如： var for while
- 保留字：当前版本的 js 系统内部还没有使用的标识符，但是，为将来做准备

转移符号：

使用双引号或**单引号(一般用)**都可以，但是不要混着用，不能嵌套，要是想打印出来用转移符（’/ “/ t/这是空格）

\" 表示 "

\' 表示 '

\n 表示换行

\t 制表符 空格

\\

数据类型(重点) （数据类型指的就是字面量的类型）

在 JS 中一共有**六种**数据类型（5 大基础数据类型 + 引用类型 Object 对象（引用类型）

console.log(typeof b); 打印 b 的类型
console.log(b); 打印 b 的值
console.log('b'); 打印 b

数据类型	类型含义	可选值	书写	查类型输出值	备注
String	字符串	所有串	Var str = = '123A 啊';	**String**	使用双引号或单引号都可以 单引号双引号不能嵌套，可用转移符
Number	数值	整数和浮点数（小数）	var Num = false;	**Number**	Number.MAX_VALUE（最大值） Number.MIN_VALUE 大于 0 的最小值 Infinity 表示正无穷 -Infinity 表示负无穷
Boolean	布尔值 用来做逻辑判断	true false	var bool = false; 没引号	Boolean	
Null	null 这个值专门用来表示一个为空的对象	Null	var Null = null; 没引号	**Object**	
Undefined	未定义：当声明一个变量，但是并不给变量赋值时	undefined	var b = undefined;	**Undefined**	

数据类型转换

其他转化成	方法	Str	Num	Bool	Null	Undefined
Str (隐式：加"")	**.toString()** Num=number()		原形	原形	没有	没有
	String() Num= String()		用 toString()	用 toString()	Null	Undefined
Number（隐式：前面加一个+，或者-） 如果对非 String 使 parseInt 或 parseFloat() 它会先将其转换为 String 然后在操作	**Number()**	数字=>数字 非数字的内容=>NaN 空串或者是一个全是空格=>0		True 转成 1 - false 转成 0	**0**	**NaN**
	parseInt() **把**字符串转成一个整数 **取**得字符串中的有效的整数到达第一个非数字就停止计算	尽可能多的数字 空串或者是一个全是空格=>0	**NaN**			
Boolean(隐式：用！)	**Boolean()**	**除了空的都转成 True**	**非 0 为真** **0 ---> false** **NaN ---> false**		**false**	**false**
var c = 1234; var num=null 初始化 num=c.toString()	num= String(c) num= Number(c)					

运算符

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 就是运算符， 可以来获得一个值的类型

它会将该值的类型以字符串的形式返回

number string boolean undefined object

算数运算符

- **通常用于两个 Number 类型的运算**
  - 就是字面量的值的运算
  - 加法 + 可以对两个值进行加法运算，并将结果返回
  - 减法 - 可以对两个值进行减法运算，并将结果返回
  - 乘法 * 可以对两个值进行乘法运算，并将结果返回
  - 除法 / 可以对两个值进行除法运算，并将结果返回
  - 求模 % 可以对两个值进行求模取余数，并将结果返回
- **当对非 Number 类型的值进行运算**

  先努力转化为 Number 类型， 然后再运算

  减法 -

  乘法 *

  除法 /

  加法 + 特殊对待

  求模 %

- **注意：**

  任何**数值**和 NaN 做运算都得 NaN

  任何**字符串**和 NaN 做运算都得 串

  一个非数字的字符串，就会转化为 NaN

  示例： 对 NaN 做加减乘除求模运算

  - 如果有一个字符串进行加法运算，则会做拼串
  - 会将两个字符串拼接为一个字符串，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
    - 然后再和字符串做拼串的操作
- **技巧**
  - 将一个任意的数据类型转换为 String
    - 任意类型 + '' --> String
    - 隐式类型转换，由浏览器自动完成
    - 实际上它也是调用 String() 函数
  - 将一个任意的数据类型转换为 Number
    - 任意类型 - 0 --> Number
    - 任意类型 * 1 --> Number
    - 任意类型 / 1 --> Number
    - 实际上它也是调用 Number() 函数

自增与自减(谁先出现就算谁,先赋值或者先计算)

自增 ++

- 变量在自身的基础上增加 1
  - 示例： a++; console.log(a);
- 自增分成两种
  - a++;
  - ++a;
- 对比（难点）

  相同点

  - 无论是 a++ 还是 ++a，都会立即使原变量的值自增 1

  不同点

  - a++ 和 ++a 的值不同
  - a++ 的值等于原变量的值（自增前的值）
  - ++a 的值等于新值 （自增后的值）

自减 --

- 变量在自身的基础上减 1
  - 不同点
    - 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 （自减以后的值

逻辑运算符

! 取反

与：bool1 && bool2 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

或：bool1 || bool2 两个值中只要有一个 true，就返回 true

- 如果两个值都为 false，才返回 false

逻辑运算符作用于非布尔类型

对于非布尔值进行与或运算时，

会先将其转换为布尔值，然后再运算，并且返回原值

与运算：共同满足

做布尔运算

如果第一个值为 true， 则必然返回第二个值

如果第一个值为 false， 则直接返回第一个值

或运算：满足一个就是满足

如果第一个值为 true， 则直接返回第一个值

如果第一个值为 false， 则返回第二个值

赋值

- 可以将符号右侧的值赋值给符号左侧的变量

+= a += 5 等价于 a = a + 5
-= a -= 5 等价于 a = a - 5
*= a *= 5 等价于 a = a * 5
/= a /= 5 等价于 a = a / 5
%= a %= 5 等价于 a = a % 5

关系运算符：

console.log(1>ture).....1>1 false
console.log(1>=ture).....1>=1 ture
console.log(1>'0').....1>0 ture
console.log(10>null).....1>1 ture
console.log(1>" hello").....1>Nav **任何值与 NaN 比较都是 false**
console.log(true>false) 1>0 ture
console.log("11">"5") 1>0 ture

如果两边都是**字符串**，比较字符串的编码，一位一位的比较(先 1 和 5 比，取大来运算)，两位一样就比较下一样，在网页中、使用 Unicode 编码：&#编码 编码是十进制要在搜索中相应的找出来后由 16 进制转化成 10 进制

相等运算符（==）

判断他们是否相等，相等返回 ture 不相等返回 false
当两边值的类型不同时，会自动转换为相同类型在进行比较
console.log("11"=="5") 11>5 ture
console.log(null==0) false **null 和什么都是 flase**
console.log(undefined>null) 因为 undefined 衍生至 null 比较返回 ture
NaN，null 不和任何值相等包括他本身

不相等运算符（！=）

不相等用于判断两个值是否不相等
不相等对变量进行自动转换

全等（===）不全等（！==）

不会进行转换，类型不同直接返回 false

条件运算符（三元运算符）
语法：
条件表达式？语句 1：语句 2
首先对运算符进行求值：如果是 ture 执行语句 1
如果是 ture 执行语句 1
Var max=a>b?(a>c?a:c):(b>c?b:c)这个是取 abc 的最大值

如果是非布尔值，先转换成布尔值再计算，字符串是 ture
空白和空格是 false

运算符的优先级

- .、[]、 new
- ()
- ++、 --
- !、~、+(单目)、-(单目)、typeof、void、delete
- %、*、/
- +(双目)、-(双目)
- << 、 >>、 >>>
- <、<=、>、>=
- ==、!==、===
- &
- ^
- |
- &&
- ||
- ?:
- =、+=、-=、*=、/=、%=、<<=、>>=、>>>=、&=、^=、|=

在表中越靠上优先级越高，优先级越高越优先计算
如果优先级一样，则从左往右计算
括号可以改变优先级