学习目标

- 能够理解javascript引入
- 熟练掌握变量的含义和作用
- 能够掌握数据类型
- 能够掌握算术运算符
- 能够理解赋值运算符的含义
- 掌握理解字符串运算符
- 掌握和理解逻辑运算符
- 能够理解三元运算符

JavaScript介绍

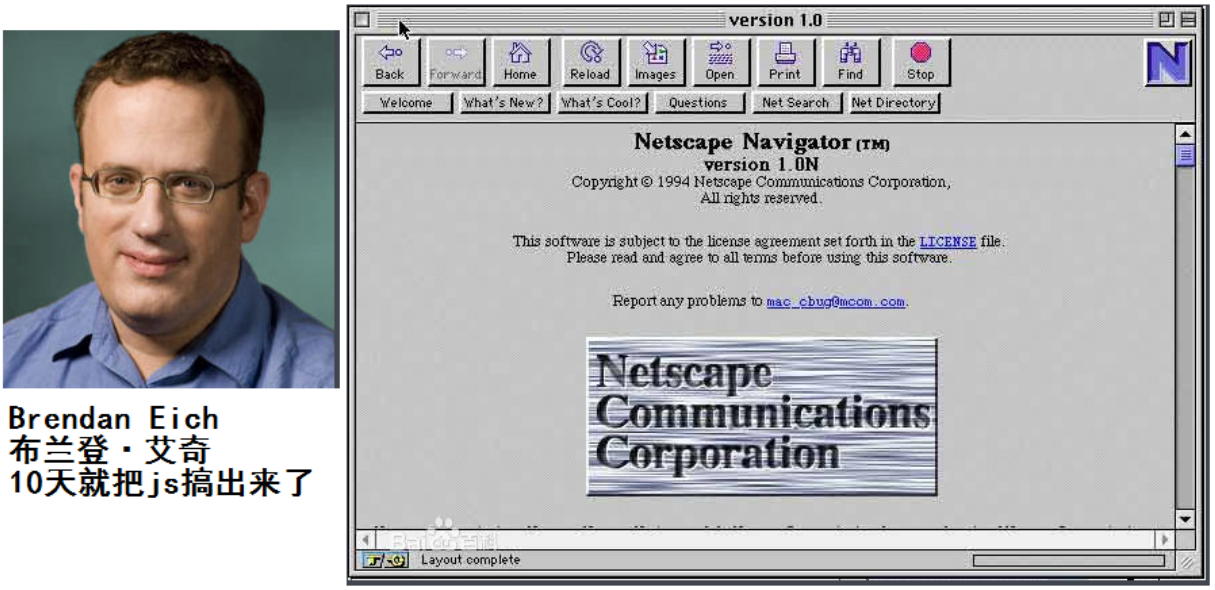
JavaScript是什么

Netscape在最初将其脚本语言命名为LiveScript，后来Netscape在与Sun合作之后将其改名为JavaScript。JavaScript最初受Java启发而开始设计的，目的之一就是“看上去像Java”，因此语法上有类似之处，一些名称和命名规范也借自Java。JavaScript与Java名称上的近似，是当时Netscape为了营销考虑与Sun微系统达成协议的结果。Java和JavaScript的关系就像张雨和张雨生的关系，只是名字很像。

Java 服务器端的编程语言

JavaScript 运行在客户端(浏览器)的编程语言

> JavaScript是一种运行在**客户端** 的**脚本语言** JavaScript的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。



JavaScript最初的目的

最初的目的是为了处理表单的验证操作。

JavaScript现在的意义(应用场景)

JavaScript 发展到现在几乎无所不能。

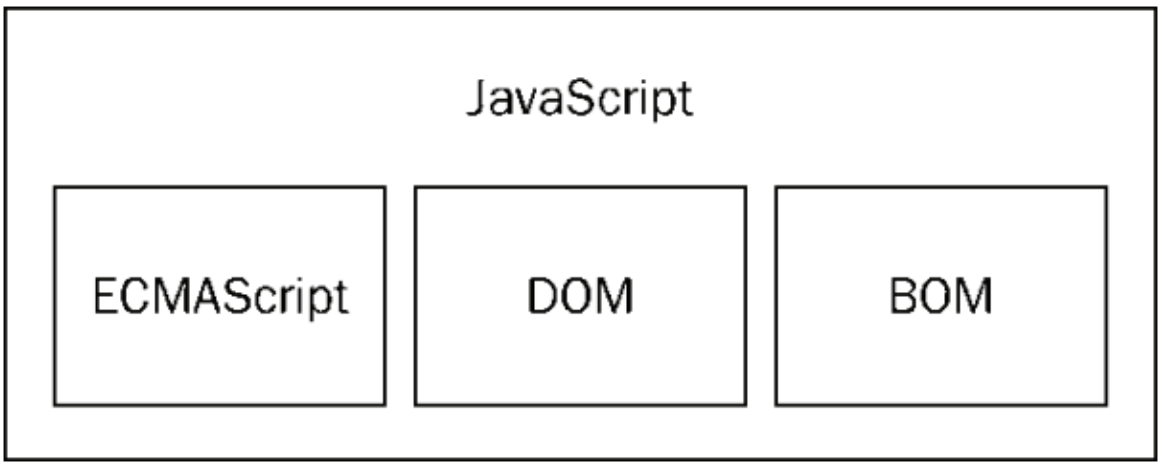
1. 网页特效　比如图片轮播,选项卡....
2. 服务端开发(Node.js)
3. 命令行工具(Node.js)

4. 桌面程序(Electron)
5. App(Cordova)
6. 控制硬件-物联网(Ruff)
7. 游戏开发(cocos2d-js ， 贪吃蛇 ， 微信打飞机....)
8. 表单验证
9. 与服务器进行交互(ajax->谷歌于2005年推出的)

JavaScript和HTML、CSS的区别

1. HTML：提供网页的结构，提供网页中的内容
2. CSS: 用来美化网页
3. JavaScript: 可以用来控制网页内容，给网页增加动态的效果

JavaScript的组成



ECMAScript - JavaScript的核心

ECMA 欧洲计算机制造联合会

网景：JavaScript

微软：JScript

定义了JavaScript的语法规范

JavaScript的核心，描述了语言的基本语法和数据类型，ECMAScript是一套标准，定义了一种语言的标准与具体实现无关

BOM - 浏览器对象模型

一套操作浏览器功能的API

通过BOM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

DOM - 文档对象模型

一套操作页面元素的API

DOM可以把HTML看做是文档树，通过DOM提供的API可以对树上的节点进行操作

JavaScript初体验

> 使用js在页面中输出hello world

JavaScript的书写位置

嵌入式

将JavaScript代码嵌入到HTML文件的script标签中 它是通过一对`<script></script>`标签来嵌入的!

注意：如果H5的文档 type属性可以省略不写!

在HTML文档中script标签可以出现多次

外链式

JS代码它可以单独的保存为一个以**.js**为扩展名的文件 然后通过HTML中的`<script></script>`的**src**属性将其引入到当前的HTML文件中!

语法:

```
<script src="js文件的路径"></script>
```

注意:**引用外部js文件的script标签中不可以写JavaScript代码**

行内式

HTML标签中 有一些**事件属性** 事件属性都是带有on前缀 比如鼠标经过、鼠标离开、鼠标单击等等

如:

onclick 鼠标单击元素时

onmouseover 鼠标移上元素时

onmouseout 鼠标移出元素时

将JS代码书写在HTML标签的事件属性中

格式:

```
<标签名  事件属性 = "js代码" >
```

JS注释

单行注释

用来注释单条代码 , 也可以用来描述下面一行或多行代码的作用

```
// 这是一个变量
var age = 18;
```

```
说明: subline编辑器下单行注释快键键  ctrl+/
```

多行注释

用来注释多条代码

```
/*
var age = 18;
var username = 'zhangsan';
console.log(username, age);
*/
```

```
说明: subline编辑器下多行注释快键键  ctrl+shift+/
```

JS代码的语法规则

- JavaScript严格**区分大小写**
- JavaScript脚本程序须嵌入在HTML文件中
- JavaScript脚本程序可以独立保存为一个外部文件 这个文件是不能自己运行的它必须要依赖于HTML文件
- JavaScript脚本程序中不能包含HTML标记代码
- 每条语句末尾如果加分号一定是英文下的分号(;)，**最好加分号,不要省略**
- 一行写了多条JS语句 这个时候每一条语句就必须要加分号

三个常用的输出语句

document.write("要输出的内容")

作用:

- 它主要是用来向body标签中**输出write()小括号里面的内容**
- document它表示是当前的HTML文档对象
- write在英文中是“写”的意思
- 对象是由属性与方法组成的 **对象.属性**与**对象.方法** 从视觉上面来区分属性与方法 **属性不带小括号 方法带有小括号**
- write它是一个输出的方法 **如果输出的内容是HTML标签,浏览器会帮我们解析**

window.alert("要输出的内容")

作用:

- alert在英文是“警告”的意思
- window它表示的是当前的浏览器窗口对象 **window对象是js中最顶级的对象 可以省略不写**
- 它主要是用来向当前的浏览器窗口中**弹出一个警告提示框**
- 实际开发不太常用 用户体验不好 经常用来检测结果

window对象与document对象之间的区别:

```
1 window对象它代表着当前的浏览器窗口对象
```

```
document对象它代表着当前的HTML文档对象
window对象包含document对象
站在window的角度来说document对象是window对象的一个属性
```

console.log("要输出的内容")

作用:

- **向浏览器的调试工具中的“console” 选项卡里面输出内容**
- console的英文意思是“控制台”
- log是日志的意思
- 经常内部测试用的 程序猿看的

另外：console.log还可以同时输出多个值

```
console.log(值1,值2,值3,值4...)
或者
console.log(变量名1,变量名2,变量名3,变量名4...)
```

变量

什么是变量

变量是一种可以变化的量，变量主要是用于存储数据的。我们命令JavaScript去干活的时候，往往需要产生一些数据，需要临时性存放起来，方便取用。我们也可以理解为，变量就像一个购物袋，我们可以用来装苹果、榴莲（当然也可以用来装玫瑰），变量是存放在内存中，内存是临时存储数据的。

硬盘：持久性存储

内存：临时性存储

总结:

- 变量是计算机内存中存储数据的标识符，根据变量名称可以获取到内存中存储的数据

- 为什么要用变量： 有些数据我们需要保存起来， 方便后面来使用 我们就可以用变量来做
- 变量是用来保存数据的 变量的本质就是一个 盒子 类似我们前面学的 html 标签
- 变量 就是用来保存数据的容器 变量是保存到内存里面的。

如何使用变量

- var声明变量

```
定义变量需要有一个关键字var    英文单词variable变量的意思
```

声明变量,不赋值

```
var 变量名
```

声明变量,并且赋值

```
var 变量名 = 值
var age;
age = 18;

var username = "zhangsan";
```

变量的赋值

将**等于号右边的值**赋值给**等于号左边的变量名**!

第一种方式:**先声明变量然后再来赋值**

```
var age;
age = 18;
```

第二种方式:**声明变量的同时直接给变量赋值**

```
var age = 18;
```

可以同时声明多个变量

方式一:

```
var age, username, sex;
```

```
age = 10;
username = 'zhangsan';
sex = '男'
```

方式二:

```
var age = 10,username = "zhangsan",sex="男";
```

修改变量的值,变量总是喜新厌旧的,变量里面只能存放最后给的值 类似css层叠性

声明一个变量已经给其赋了值 然后再来修改这个变量的值!

格式:

```
变量名 = "新值"  //给变量名重新赋一个值
```

```
var age = 16;
age = 18;
//这个时候，age的值为18，而不是16，18.
```

为什么要使用变量

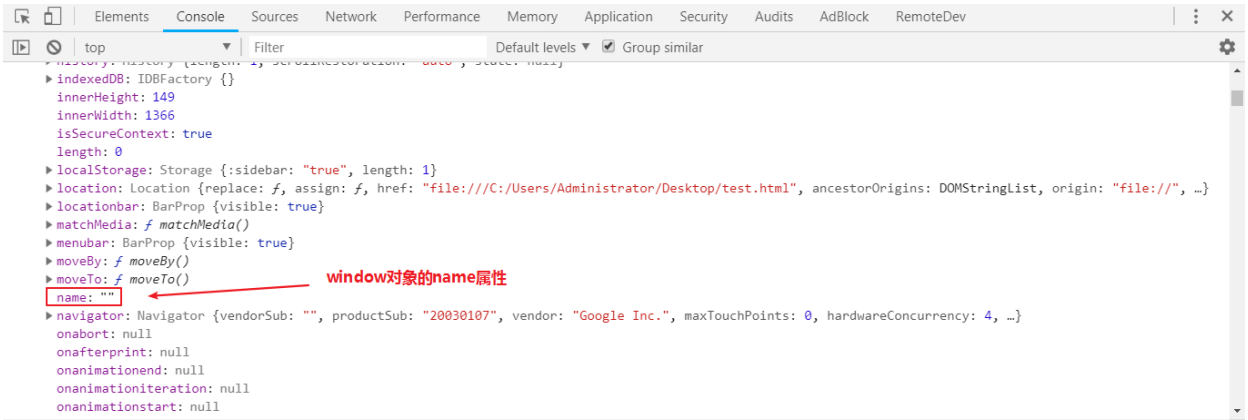
使用变量可以方便的获取或者修改内存中的数据

变量的命名规则和规范

- **规则 - 必须遵守的，不遵守会报错**
  - 由字母(A-Za-z)、数字(0-9)、下划线(_)、$符号组成，**不能以数字开头**
  - **不能是JS中关键字和保留字**，例如：for、while等(**其中变量名为name需要注意,不会错,但是不推荐使用,因为name是window对象下的一个属性,设置的属性值会被加上双引号,成为字符串类型**)

JS关键字				
break	case	catch	continue	default
do	else	finally	for	function
if	in	instanceof	new	return
switch	this	throw	try	typeof
var	void	while	with	

JS保留字				
abstract	boolean	byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	implements
import	int	interface	long	native
package	private	protecter	public	short
static	super	synchronized	throws	transient
volatile	name			



- **区分大小写**

- **规范 - 建议遵守的，不遵守不会报错**
  - 变量名需要**有意义**,希望大家在声明变量的时候变量名要做到"**见名知意**"
  - 变量名可以由多个英文单词组成 建议使用**下划线连接法** 或者是**驼峰法**
    - 下划线连接法 每一个单词之间使用下划线进行连接 比如:get_user_name
    - 驼峰法 第一个英文单词的首字母小写 其它的英文单词首字母大写 比如:getUserName

变量练习1

我叫卡卡西，我住在火影村，我今年30了我的邮箱是kakaxi@qq.com，我的工资2000.

变量练习2

下面哪些变量名不合法

```
a
1
age18
18age
name
$name
_sex
&sex
theworld theWorld
```

变量练习3

交换两个变量的值,比如var a = 10; var b = 20; 交换以后变成 a = 20 , b = 10;

变量的数据类型　重点

为什么变量需要有数据类型

变量主要是用于存储数据的，现实生活中的数据有很多种比如有数值、有字母等等 那么为了将这些数据进行分门别类，所以就引出了变量的数据类型。

JavaScript是一种弱类型的语言。 在声明变量的时候不需要指定变量的数据类型强类型的语言，在声明变量的时候一定要先指明这个变量的数据类型是什么并且值也是这个数据类型

因为JS是一门弱类型的脚本语言，在声明变量的时候不需要先声明变量的类型。但是它也是有类型，JS变量的类型是由变量的值来决定 。

变量的数据类型分为： 两大类、七小种

两大类：基本数据类型(标量数据类型)、复合数据类型！

基本数据类型：只能存储一个值

复合数据类型：至少存储一个值，可以存储多个值

获取变量的类型

在JavaScript中有一个内置的函数可以检测变量的数据类型

语法:

```
typeof(要检测的变量名)
或者
typeof 要检测的变量名
```

举例:

基本(标量)数据类型

只能存储一个值

String(字符串型)、Number(数值型)、Boolean(布尔型)、undefined(未定义型)、null(空型)

String(字符串型) 重点

变量的值加了引号的数据！我们就称之为string字符串数据类型！

引号:单引号和双引号都可以！不管引号里面是什么 只要是加了引号的数据都称之字符串数据。

平时使用的时候，**汉字需要加引号，特殊符号也需要加引号，除了JS中的一些特殊英文单词比如null,undefined,true,false等以外其它的都需要加引号**

举例:

思考:如何打印以下字符串。我是一个"帅气"的人

举例:

一般在情况下：

- **如果是使用的是双引号定义的变量 那么里面就会使用单引号 双包单**

- **如果是使用的是单引号定义的变量 那么里面就会使用双引号　　单包双**

那么如何在双引号定义的变量中是否还能在出现双引号?如何在单引号中定义的变量里面是否还能出现单引号?要解决上述问题 就需要使用转义字符！在JS中有一个叫转义字符 \ 用于将某些字符的含义进行转换

字面量	含义
\n	换行
\t	制表
\b	空格
\r	回车
\f	进纸
\\	斜杠
\'	单引号（'），在用单引号表示的字符串中使用。例如：'He said, \'hey.\''
\"	双引号（"），在用双引号表示的字符串中使用。例如："He said, \"hey.\""
\xnn	以十六进制代码*nn*表示的一个字符（其中*n*为0～F）。例如，\x41表示"A"
\unnnn	以十六进制代码*nnnn*表示的一个Unicode字符（其中*n*为0～F）。例如，\u03a3表示希腊字符Σ

举例:

获取字符串的长度

通过length属性可以获取字符串的长度

```
var str = 'Hello World';
document.write(str.length);
```

字符串拼接

字符串拼接使用 + 连接

```
console.log(11 + 11);
console.log('hello' + ' world');
console.log('100' + '100');
console.log('11' + 11);
```

注意:

1. **两边只要有一个是字符串，那么+就是字符串拼接功能**
2. **两边如果都是数字，那么就是算术功能加法。**

Number(数值型)　　**重点**

包含：整数与小数、NaN (Not a Number) 它不是一个数

整数：正整数、负整数、0

小数：正小数、负小数

举例:

扩展　　Number类型的浮点数

浮点数的精度问题，在计算机的底层只有0或1，所以小数在计算机的底层也需要翻译成二进制数据。但是这个过程，可能会产生精度丢失。

所以要记住：不要随便判断两个浮点数是否相等

举例:

```
var num1 = 0.1;
var num2 = 0.2;
var sum = num1 + num2;
console.log(sum);// 0.30000000000000004
console.log(sum == 0.3);// false
```

Boolean(布尔型)　　**重点**

布尔型它主要是用来表示真与假!

布尔型数据只有两个值:

true(真)和false(假)

如何得到布尔型:

定义一个变量的值为true或者是false 就可以得到布尔型。

注意:

布尔型的值是小写的true和false,区分大小写

举例:

undefined(未定义型)

- undefined表示一个声明了**没有赋值**的变量，变量只声明的时候值默认是undefined
- 定义一个变量给其值赋值为undefined,然后直接打印或者使用这个变量,变量值也是undefined

举例:

null(空型)

null表示一个空，想到得到null类型,只需要在定义变量的时候,赋值为null即可

同学们也许会问，为什么 typeof 运算符对于 null 值会返回 "Object"。这实际上是 因为JavaScript 最初实现中的一个错误，然后被 ECMAScript 沿用了。

举例:

复合数据类型

至少存储一个值，可以存储多个值。

Object(对象)、Array(数组)这两个类型我们后面课程详细讲解!

通过开发者工具判断基本数据类型

不同基础数据类型在浏览器的开发者工具中，颜色会有所不同

- 字符串的颜色是黑色的
- 数值类型是蓝色的
- 布尔类型也是蓝色的
- undefined和null是灰色的。

常量

什么是常量,就是一旦定义,值就不能改变

```
语法:   const 变量名 = 值;
```

注意:

```
1. 声明常量,值不能改变
```

```
2  2. 而且声明的时候必须立即初始化,也就是给常量赋值
```

运算符　重点难点

什么是运算符

运算符就是**可以进行运算**的符号 比如：+、-、*、/

运算符可以为分三种：一目、二目、三目也可以叫一元、二元、三元一目：指的是运算符的操作数只有一个 比如：　i++二目：指的是运算符的操作数有两个 比如：　a+b三目：指的是运算符的操作数有三个　比如：　a>b？ a ： b 运算符分为：

- 算术运算符
- 赋值运算符
- 比较运算符
- 字符串连接运算符
- 逻辑运算符
- 三目运算符

算术运算符

假设： *y=5*，下面的表格解释了这些算术运算符：

运算符	描述	例子	结果
+	加	x = y+2	x = 7
-	减	x = y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	取模,也可以叫求余数	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

举例:

自操作运算符(难点)

++与--这两个运算符的规则是一样的。

++运算符它称之为自加1运算符

++运算符它分为前加加和后加加

前加加： ++符号在变量的前面 比如：++a;

前加加的运算规则：**先自加1 然后再赋值**

后加加： ++符号在变量的后面 比如：a++;

后加加的运算规则：**先赋值然后再自加1**

举例:

--运算符它称之为自减1运算符

--运算符它分为前减减和后减减

前减减：--符号在变量的前面 比如：--a;

前减减的运算规则：**先自减1 然后再赋值**

后减减：--符号在变量的后面 比如：a--;

后减减的运算规则：**先赋值然后再自减1**

举例:

总结:

```
前++:先加1，后参与运算
后++:先参与运算，后加1
前-- :先减1，后参与运算
后-- :先参与运算，后减1
不管是前加加还是后加加自身都会自加1
不管是前减减还是后减减自身都会自减1
```

练习:

同学们猜猜看结果

```
var a = 1; var b = ++a + ++a; console.log(b);
var a = 1; var b = a++ + ++a; console.log(b);
var a = 1; var b = a++ + a++; console.log(b);
```

```
4  var a = 1; var b = ++a + a++; console.log(b);
```

赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 *x=10* 和 *y=5*，下面的表格解释了赋值运算符：

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

= ： 将等号右边的值赋值给等号左边的变量+= ： 将等号左边的变量的值加上等号右边的值然后将其结果赋值给等号左边的变量 -= ： 将等号左边的变量的值减去等号右边的值然后将其结果赋值给等号左边的变量*= ： 将等号左边的变量的值乘以等号右边的值然后将其结果赋值给等号左边的变量/= ： 将等号左边的变量的值除以等号右边的值然后将其结果赋值给等号左边的变量%=： 将等号左边的变量的值与等号右边的值进行求余运算然后将其结果赋值给等号左边的变量

注意:我们在使用复赋值运算符的时候注意事项

1. 如果我们使用赋值运算符,**在变量前面是没有var关键字的**
2. 这些赋值运算符号**必须紧密相连**,比如 += 不要写成了+ =

举例:

比较运算符

比较运算符执行的是比较运算。每个比较运算符都返回一个布尔值。结果为：**true或者false**

假设： y=5 x = 10，下面的表格解释了比较运算符的结果：

运算符	描述	例子	结果
>	大于	x>y	true
<	小于	x<y	false
>=	大于等于	x>=y	true
<=	小于等于	x<=y	false
==	等于 只需要比较值是否相等	x==y	false
!=	不等于	x!=y	true
===	全等于 (要比较数据类型相等和值相等)	x === y	false
!==	不全等于	x !== y	true

注意：比较运算符它最终得到的结果是布尔值：**true和false**

举例:

字符串连接运算符

什么是字符串连接?

就是将多个字符串合并为一个字符串.

符号	功能
+	字符串连接
+=	字符串连接

+运算符:

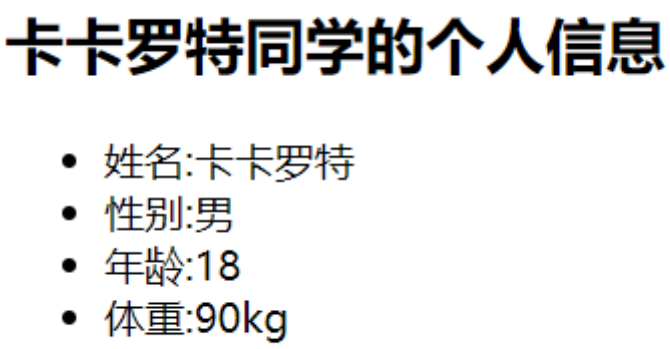
```
这个+号它又是算术运算符又是字符串连接运算符
什么时候它是执行加法运算 什么时候是执行字符串连接运算!
如果+号两边的变量的数据类型都是数值型的时候 就会执行加法运算
如果+号两边的有一个变量的数据类型都是字符串类型的时候 就会执行字符串连接运算
变量与字符串之间拼接要使用:+
```

```
工作中:
"字符串"+变量名+"字符串"    字符串拼接的方法
```

+=运算符：将等号左边的变量的值加上等号右边的值然后将其结果赋值给等号左边的变量

```
比如：var a = 10;    a+=3;    那么a+=3就等价于a=a+3=10+3=13
```

案例：使用字符串拼接的方式打印出某个人的个人简介 要求使用无序列表的格式打印



逻辑运算符

在形式逻辑中，逻辑运算符或逻辑联结词把语句连接成更复杂的复杂语句。例如，假设有两个逻辑命题，分别是“正在下雨”和“我在屋里”，我们可以将它们组成复杂命题“正在下雨，并且我在屋里”或“没有正在下雨”或“如果正在下雨，那么我在屋里”。一个将两个语句组成的新的语句或命题叫做复合语句或复合命题。逻辑运算符用来表示日常交流中的“并且”，“或者”，“除非”等思想。

符号	说明	功能
&&	逻辑与 并且	&&符号两边的表达式的结果同时为真才为真
||	逻辑或 或者	||符号两边的表达式的结果只要有一边为真就为真
!	逻辑非 取反	取反操作 将true变为false 将false变为true

小结:

```
&& 两个操作数同时为true，结果为true，否则都是false
|| 两个操作数有一个为true，结果为true，否则为false
```

```
3 !    取反
```

三目运算符

语法格式：

条件语句 ? 语句1 : 语句2;

说明：

问号前面的位置是判断的条件，判断结果为boolean型，为true时执行语句1，为false时执行语句2。

其逻辑为："如果为真执行第一个，否则执行第二个。"

举例:

运算符的优先级

运算符	描述
. [] ()	字段访问、数组下标、函数调用以及表达式分组
++ -- - ~ ! delete new typeof void	一元运算符、返回数据类型、对象创建、未定义值
* / %	乘法、除法、取模
+ - +	加法、减法、字符串连接
<< >> >>>	移位
< <= > >= instanceof	小于、小于等于、大于、大于等于、instanceof
== != === !==	等于、不等于、严格相等、非严格相等
&	按位与
^	按位异或
|	按位或
&&	逻辑与
||	逻辑或
?:	条件
= +=	赋值、运算赋值
,	多重求值

```
1 优先级从高到底
2    1. () 优先级最高
3    2. 一元运算符 ++    --    !
4    3. 算数运算符 先* / %    后 +    -
5    4. 关系运算符 >    >=    <    <=
6    5. 相等运算符    ==    !=    ===    !==
7    6. 逻辑运算符 先&&    后||
8    7. 赋值运算符
```

```
1 为什么会有运算符的优先级这个问题？
2 在一个式子里面出现了多种运算符 需要考虑哪一个运算符优先运行。
```

小学数学学过：有小括号先算小括号里面的 然后再乘除后加减 如果需要提升某一个运算符的优先级 请加小括号

```
1 // 练习1：
```

```
console.log (   4 >= 6 || '人' != '阿凡达' && !(12 * 2 == 144) && true
);
// 练习2:
var num = 10;
console.log(    5 == num / 2 && (2 + 2 * num) === '22'    );
```

表达式和语句

表达式

一个表达式可以产生一个值，有可能是运算、函数调用、有可能是字面量。表达式可以放在任何需要值的地方。

```
常识:
 只能操作一个值的操作符叫一元操作符,组成的表达式叫一元表达式.(一个值一个运算符)
 由两个值和一个运算符连接起来的表达式叫二元表达式。(两个值一个运算符)
 由三个值和两个运算符连接起来的表达式叫三元表达式.(三个值两个运算符)
```

```
例如:
 12
 aa++
 aa + 56
 5 > 6 ? '5' : '6'
```

语句

语句可以理解为一个行为，循环语句和判断语句就是典型的语句。一个程序有很多个语句组成，一般情况下;分割一个一个的语句

```
JavaScript所提供的语句分为以下几大类:
 1. 变量声明，赋值语句：var。
 语法如下： var 变量名称 [=初始值]
 例如：var num = 25;
 2. 条件和分支语句：if...else，switch。 （后面具体学习）
 3. 循环语句：for， for...in，while,break,continue。（后面具体学习）
 4. 注释语句：//，/*...*/。
 // 这是单行注释
 /* 这可以多行注释.... */
```

今日总结

今日作业