JavaScript

编程语言：

机器语言、汇编语言、高级语言

机器语言和汇编语言本质相同，都属于低级语言，汇编是机器语言加好记的英文标识符。高级语言在运行前需要翻译器翻译成二进制代码才能运行

高级语言分为编译性语言和解释性语言，编译性语言：C、C++、C#、Java；解释性语言：python、JS

标记性语言：HTML

编程语言和标记性语言的区别

编程语言有很强的逻辑和行为能力，并向计算机主动发出指令

标记语言不向计算机发出指令，常用语格式化和链接，被动被读取的

JS初识

js是一种运行在客户端的脚本语言

脚本语言：不需要编译，在运行过程中由JS解释器逐行进行解释并执行

现在也可以基于Node.js技术进行服务器端编程

JS的作用

1. 表单动态校验（密码强度检测）（JS产生最初的目的）
2. 网页特效
3. 服务端开发（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（Cocosd-js）

浏览器执行JS简介

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，如chrome浏览器的blink，老版本的webkit

JS引擎：也称为JS解释器，用来读取网页中的JavaScript代码，对其处理后 运行，比如chrome浏览器的V8

浏览器本身不会执行JS代码，而是通过内置JS引擎来执行JS代码。JS引擎执行代码时逐行解释每一句源码，然后由计算机执行。

JS组成

ECMAScript：JS语法

DOM：页面文档对象模型

BOM：浏览器对象模型

ECMAScript：是由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，被称为JavaScript或JScript，实际上是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

JavaScript由网景公司发布

JScript由微软公司发布

DOM：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）

BOM：提供了独立于内容的、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

JS书写位置：

1. 行内：直接写到元素内部
2. 可以将单行或少量JS代码写在标签元素的事件属性中（以on开头的属性），如onclick
3. 注意单双引号的使用：HTML中推荐使用双引号，JS中推荐使用单引号
4. 行内编写可读性差，引号易错
5. 特殊情况下使用
6. 内嵌：在<head>标签内添加<script></script>标签，并在内部编写JS代码
7. 外部
8. 新建后缀为.js的文件
9. 在其中编写JS代码
10. 在html页面中引入：<script scr=’url’></script>
11. 引用外部JS文件的<script>标签内部不可以写代码

JS注释

单行注释：// 快捷键：ctrl+/

多行注释：/\* \*/ 快捷键：shift+alt+a

注释快捷键再按取消注释

JS输入输出语句

alter(‘xxx’)：

浏览器弹出警示框，归属浏览器

console.log(‘xxx’)：

浏览器控制台打印输出信息，归属浏览器，类似python的print

prompt(info)：

浏览器弹出输入框，用户可以输入，归属浏览器，返回值为字符串

变量

本质是程序在内存中申请的一块用来存放数据的空间

声明变量：

var xxx; //var关键字声明变量

赋值变量：

xxx=aaa; //与其他编程语言一样

变量初始化：声明并赋值

var xxx=aaa; //声明赋值放在一行内，简洁

多变量声明赋值：

var xxx=a, nnn=b, mmm=c;

注：

1.JS变量可直接赋值使用不用必须声明

2.区分大小写

3.不能以数字开头，不能包含空格

4.不能是关键字

5.类似python

6.尽量不用name作为变量名

7.特殊字符开头只能是$和\_

8.var a=b=c=10 此声明方法等效于var a=10；b=10；c=10，若在函数中a为局部变量，b、c为全局变量

数据类型

数据类型本质是为不同的数据分配大小合适的内存空间，这样可以有效节省内存空间。JS和python一样自动定义数据类型

简单数据类型

Number：数字型，包含整型和浮点型

Boolean：布尔型，True，False，等价于1,，0

String：字符串型，”xxxxxx”

Undefined：var a; 声明了变量没赋值，则a为Undefined

Null：var a=null; 声明变量a为空值

Number：

八进制表示：值前加0，如8 ：010

十六进制表示：值前加0x，如21 ：0x15

最大值：Number.MAX\_VALUE

最小值：Number.MIN\_VALUE

无穷大：Infinity，代表无穷大，大于任何数值

无穷小：-Infinity，代表无穷小，小于任何数值

NaN：not a number，代表任何一个非数值

isNaN(xxx)：用来判断一个变量是否为非数字类型，是非数字返回true，不是返回false

String：

使用单引号或双引号定义，JS推荐优先使用单引号

可用引号嵌套，外双内单或者外单内双都行

转义符：

都是\开头

\n：换行符

\\：斜杠\

\’：单引号

\”：双引号

\t：tab缩进

\b：空格

字符串长度

使用length属性可以获取整个字符串的长度，str.length

字符串拼接

用+号进行拼接，字符串和其他任何类型的值拼接都按字符串处理，操作类似python

Boolean：

true和false，在运算中true为1，false为0

Undefined和Null

Undefined和数字相加等于NaN

获取变量数据类型

typeof(xxx)，返回变量的数据类型，不用括号加空格也可

数据类型转换

1. 转换为string类型

toString()：转成字符串，xxx.toString()

String(xxx)：转成字符串

加号拼接字符串：和字符串拼接

1. 转换为Number型

parseInt(string)：将string型转换成整数型，取整，必须开头为数字，并且会自动去掉后面的字母

parseFloat(string)：将string型转换为浮点型，保留小数，必须开头为数字，并且会自动去掉后面的字母

Number(string)：将string型强制转换成数值型

隐式转换：利用算术运算符隐式转换为数值型

1. 转换为布尔型

Boolean(xxx)，空字符串，0，NaN，null，undefined会转换成false，其他的都会转换成true

标识符

开发人员为变量、属性、参数、函数起的名字

关键字

js所使用的字，有特殊含义了，不能作为标识符

保留字

未来可能会成为关键字的字，不能作为标识符

运算符

+、-、\*、/、%（取余）

浮点数存在精度问题不能直接判断相等与否

递增

++m、m++，类似c，递增1

前置递增++m，先递增再返回值

后置递增m++，先返回值在递增

递减

--m、m--，类似c，递减1

前置递减--m，先递减再返回值

后置递减m--，先返回值再递减

比较运算符

和python一样

==：等于，会给变量转换数据类型，如38==’38’true

===：全等，数值和数据类型必须都一样才行，如38===’38’false

逻辑运算符

&&：与

||：或

！：非，取反

赋值运算符

=：直接赋值

+=：先加一个数再赋值（类似python，该角度解释更确切）

-=：先减一个数再赋值

\*=、/=、%=：乘、除、取模后再赋值

流程控制

分三种结构：顺序结构、分支结构、循环结构

if语句

挨条判断，速度相对慢

if (条件表达式) {

执行语句2;

} else if {

执行语句2;

} else {

执行语句3;

}

三元表达式

条件表达式？表达式1：表达式2

若条件表达式为true，返回表达式1的值，若为false返回表达式2的值

switch语句

直接跳转符合的执行语句，分支多的时候速度比if快

switch (表达式) {

case 值1:

执行语句1;

break;

case 值2:

执行语句2;

break;

...

default:

执行最后的语句;

}

注：如果不写break将不判断直接执行下一个case

for循环

语法类似c++

for (i=1; i<10; i++) {

执行语句

}

数组array

arr=[1,2,3,4,5]，可以存放任意类型的数据

创建数组

1. 利用new关键字

var 数组名=new Array( );

1. 利用数组字面量创建数组

var arr=[ ];

访问数组

arr[索引]

数组长度

arr.length

数组扩容修改

arr=[1,2,3]

1. 修改数组索引

arr[3] // 相当于把数组扩容了一个，arr[3]=undefined

1. 修改索引并赋值

arr[3]=xxx // 直接扩容并赋值为xxx

1. 修改数组内数据

arr[0]=4 // 修改索引为0的数据为4

注：不要给数组名直接赋值，会把数组抹掉并重新赋值，如arr=5，数组arr会重置为一个普通变量并赋值为5

函数

封装了一段可以被重复调用的代码块，为了让代码更好的复用

声明函数

常用动名短语作为函数名

1. 关键字声明

function 函数名(参数名) {

函数体

}

1. 匿名函数

var 变量名=function(形参) {

函数体

}

调用函数

1. 函数名(参数)
2. 匿名函数：变量名(实参)

参数传递

只有位置传参，多个参数之间用英文逗号分隔

1. 实参比形参多，无影响，多的实参不会被接收
2. 实参比形参少，缺少的形参会被定义为undefined

函数返回值

return 变量名

1. return只能返回一个值，若想返回多个值可以使用数组

return [1,2,3,4,5]

1. return后的代码不会被执行，可以退出循环，并且直接结束当前函数
2. 函数没有return返回undefined

arguments的使用

一个存储了所有传入的参数值的伪数组，具有length属性，没有pop、push方法

作用域

变量在某个范围内起作用和效果，目的是为了提高程序的可靠性，减少命名冲突

全局作用域

整个<script>标签内，或者单独的一个js文件

局部作用域

函数内部，只在函数内部起效果和作用

块级作用域

带花括号的{ }就是块级作用域

全局变量

1. 在全局作用域下声明的变量为全局变量
2. 在函数内不声明直接赋值的变量也是全局变量

全局变量只有在浏览器关闭的时候才会销毁，比较占内存资源

局部变量

函数内部声明的变量为局部变量

局部变量在程序执行完毕就会销毁，比较节约内存资源

案例：var a=b=c=10 此声明方法等效于var a=10；b=10；c=10，若在函数中a为局部变量，b、c为全局变量

作用域链

内部函数访问外部函数的变量，采取链式查找的方式来决定取哪个值，这种结构我们称为作用域链（就近原则）

预解析

js引擎运行js代码先预解析，再执行

js预解析会把所有var和function提升到当前作用域的最前面

代码执行会按照从上到下的顺序

预解析分为变量预解析（变量提升）和函数预解析（函数提升）

变量提升就是把素有的变量声明提升到当前的作用域最前面，不提升赋值操作

函数提升就是把函数声明提升到当前作用域的最前面，不调用函数

对象

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

属性：事物的特征，在对象中用属性来表示（常用名词）

方法：事物的行为，在对象中用方法来表示（常用动词）

创建对象（object）

1. 字面量创建对象

花括号（大括号）创建对象{ }

var obj={

uname: ‘张三丰’,

age: ‘18’,

sex: ‘男’,

sayHi: function() {

函数体

}

}

1. 里面的属性或者方法采取键值对形式：属性名: 属性值 方法名: 匿名函数
2. 多个属性或者方法中间用逗号隔开
3. 方法冒号后面跟的是一个匿名函数
4. 利用new Object创建对象

var obj=new Object();

obj.uname=’xxx’ //追加属性

obj.age=18 //追加属性

obj.sayHi=function() {

函数体;

} //追加方法

1. 构造函数创建对象（类）

构造函数就是把对象里面一些相同的属性和方法抽象出来封装到函数里，方便重复使用大量创建对象

构造函数语法

function 构造函数名(形参名) {

this.属性名=值;

this.方法名=function() {}

}

调用构造函数创建对象（对象的实例化）

new 构造函数名(实参);

使用对象

1. 调用对象的属性方法1，对象名.属性名
2. 调用对象的属性方法2，对象名[‘属性名’]
3. 调用对象的方法，对象名.方法名()

遍历对象

for (var k in obj) {

console.log(k); //输出属性名

console.log(obj[k]); //输出属性值

console.log

}