# JS基础第一天(8月1日)

**一、JavaScript简介**

1. 什么是JavaScript？

概念：JavaScript是一门编程语言，一种弱类型的解释型的脚本语言。

1. 什么是编程语言？

具有逻辑处理能力的语言，就是编程语言。

1. 什么是弱类型的编程语言？

Java，c#，PHP等都是强类型编程语言，因为他们所操作的数据，在定义前都必须明确的指定它的数据类型。如：int n = 1；double m = 1.1；string c = “a”；等。

JavaScript是弱类型的编程语言，因为他所操作的数据，在定义时，始终是由关键字var来定义的，只不过它的数据类型是由后面的值决定的。

1. 什么是解释型的语言？

编程语言分为两种类型：解释型和编译型。

解释型：简单的说是，边解释边执行。

编译型：简单的说是，先解释（翻译）完在执行。

1. 什么是脚本语言？

所谓脚本语言，指的是它不具备开发操作系统的能力，而是只用来编写控制其他大型应用程序的“脚本”。

但是JavaScript早已超出了其“脚本语言”的范畴，而成为一种集健壮性、高效性和通用性为一体的编程语言。

6.JavaScript和CSS与HTML有什么区别？

JavaScript是一种编程语言，它能使用户和网页互动起来，增强用户的体验性。

HTML（Hyper text markup language）是超文本标记语言，用来组织网页结构，使网页内容语义化。

CSS（cascading style sheet）是层叠样式表，用来搭建网页布局，修饰网页内容。

1. JavaScript的应用领域

PC端： 网站； 移动端：web app、混合app；

服务端： 基于Node.JS平台； 游戏： canvas，水果忍者；

1. JavaScript的发展历史

JavaScript因为互联网而生，紧随着浏览器的出现而问世。回顾它的历史，就要从浏览器的历史讲起。

1990年底，欧洲核能研究组织（CERN）科学家Tim Berners-Lee(蒂姆·伯纳斯-李)，在全世界最大的电脑网络——互联网的基础上，发明了万维网（World Wide Web），从此可以在网上浏览网页文件。最早的网页只能在操作系统的终端里浏览，也就是说只能使用命令行操作，网页都是在字符窗口中显示，这当然非常不方便。

1992年底，美国国家超级电脑应用中心（NCSA）开始开发一个独立的浏览器，叫做Mosaic。这是人类历史上第一个浏览器，从此网页可以在图形界面的窗口浏览。

1994年10月，NCSA的一个主要程序员Marc Andreessen(马克·安德森)联合风险投资家Jim Clark(吉姆·克拉克)，成立了Mosaic通信公司（Mosaic Communications），不久后改名为Netspace(网景)。这家公司的方向，就是在Mosaic的基础上，开发面向普通用户的新一代的浏览器Netscape Navigator。

1994年12月，Navigator发布了1.0版，市场份额一举超过90%。

Netscape公司很快发现，Navigator浏览器需要一种可以嵌入网页的脚本语言，用来控制浏览器行为。当时，网速很慢而且上网费很贵，有些操作不宜在服务器端完成。比如，如果用户忘记填写“用户名”，就点了“发送”按钮，到服务器再发现这一点就有点太晚了，最好能在用户发出数据之前，就告诉用户“请填写用户名”。这就需要在网页中嵌入小程序，让浏览器检查每一栏是否都填写了。

1995年，Netscape公司雇佣了程序员Brendan Eich（布兰登·艾奇）开发这种网页脚本语言，JavaScript1.0。

Netscape公司的这种浏览器脚本语言，最初名字叫做Mocha（摩卡），1995年9月改为LiveScript。12月，当时正逢Java比较火，为了“榜大牌”，所以改名“JavaScript”。所以JavaScript和Java没有关系。

1996年3月，Navigator 2.0浏览器正式内置了JavaScript脚本语言。

**--邪恶的后来者**

因为 JavaScript1.0 如此成功，所以微软也决定进军浏览器，1996年8月，微软模仿JavaScript开发了一种相近的语言，取名为JScript（JavaScript是Netscape的注册商标，微软不能用），首先内置于IE 3.0。Netscape公司面临丧失浏览器脚本语言的主导权的局面。

**--标准的重要**

在微软进入后，有 3 种不同的 JavaScript 版本同时存在：NetscapeNavigator 3.0 中的 JavaScript、IE 中的 JScript 以及 CEnvi 中的 ScriptEase。与 C 和其他编程语言不同的是， JavaScript 并没有一个标准来统一其语法或特性，而这 3 种不同的版本恰恰突出了这个问题。 随着业界担心的增加，这个语言标准化显然已经势在必行。

**--JavaScript标准化**

1996年11月，Netscape公司决定将JavaScript提交给国际标准化组织ECMA（European Computer Manufacturers Association欧洲计算机制造商协会），希望JavaScript能够成为国际标准，以此抵抗微软。1997年7月，ECMA组织发布262号标准文件（ECMA-262）的第一版，规定了浏览器脚本语言的标准，并将这种语言称为ECMAScript。这个版本就是ECMAScript 1.0版。

**--灵敏的微软、迟钝的网景**

虽然网景开发了 JavaScript 并首先提交给 ECMA 标准化，但因计划改写整个浏览器引擎 的缘故，网景晚了整整一年才推出“完全遵循 ECMA 规范”的 JavaScript1.3。而微软早在一 年前就推出了“完全遵循 ECMA 规范”的 IE4.0。这导致一个直接恶果：JScript 成为 JavaScript 语言的事实标准。

**--标准的发展**

在接下来的几年里，国际标准化组织及国际电工委员会（ISO/IEC）也采纳 ECMAScript 作为标准（ISO/IEC-16262）。从此，Web 浏览器就开始努力（虽然有着不同程度的成功和失 败）将 ECMAScript 作为 JavaScript 实现的基础。

**--山寨打败原创**

JScript 成为 JavaScript 语言的事实标准，加上 Windows 绑定着 IE 浏览器，几乎占据全 部市场份额，因此，1999 年之后，所有的网页都是基于 JScript 来开发的。而 JavaScript1.x变成可怜的兼容者。

**--网景的没落与火狐的崛起**

网景在微软强大的攻势下，1998 年全面溃败。但，星星之火可以燎原。同年成立 Mozilla（魔斯拉） 项目中 Firefox(火狐浏览器)在支持 JavaScript 方面无可比拟，在后来的时间里一步步蚕食 IE 的市场，成为全球第二大浏览器。

**--谷歌的野心**

GoogleChrome，又称 Google 浏览器，是一个由 Google（谷歌）公司开发的开放原始码 网页浏览器。他以简洁的页面，极速的浏览，一举成为全球第三大浏览器。随着移动互联网 的普及，嵌有 Android 系统的平板电脑和智能手机，在浏览器这块将大有作为。

**--苹果的战略**

Safari 浏览器是苹果公司各种产品的默认浏览器，在苹果的一体机(iMac)、笔记本(Mac)、 MP4(ipod)、iphone(智能手机)、ipad(平板电脑)，并且在 windows 和 Linux 平台都有相应版 本。目前市场份额全球第四，但随着苹果的产品不断的深入人心，具有称霸之势。

**-幸存者**

Opera 的全球市场份额第五，2%左右。它的背后没有财力雄厚的大公司，但它从“浏览 器大战”存活下来的，有着非常大的潜力。

1. JavaScript的组成部分

ECMAScript(标准语法):就是定义了统一的语法。

BOM：浏览器对象模型，基于标准语法，浏览器提供一些可以操作浏览器的工具（方法与接口）。

DOM：文档对象模型，基于标准语法，浏览器提供一些可以操作浏览器中网页内容的工具。

**二、第一个JavaScript程序**

1. 开发工具

Dreamweaver，editplus，Notepad++等

1. <script>标签

语法：<script type=”text/JavaScript”>

Document.write(“hello world”);

</script>

<script type=”text/JavaScript” src=”路径”></script>

属性：language已废弃，

Src：表示要引入的外部文件

Type：表示脚本语言的类型，告诉浏览器<script>标签里的是什么代码，然后根据type属性所确定的类型进行解析。

1. 代码执行顺序

自上而下进行解析。

1. JavaScript的注释

单行注释：//

多行注释:/\*\*/

1. 转义符

&gt：>

&lt:<

**三、变量**

1. 作用：方便存储数据。

2. 直接变量

如：1；

1. var 声明变量

Var 变量名 = 1；//赋值

1. JS标识符：

标识符是指JS中定义的符号，如：变量名和函数名等；

标识符可以由任意的大小写字母，数字，下划线，和美元符组成，但不能以数字开头，不能是JS中的关键字和保留字。

标识符区分大小写，如： age和Age是不同的变量。

5. 关键字和保留字

--关键字

关键字不可以用作变量名

|  |  |  |  |
| --- | --- | --- | --- |
| Break | Else | New | var |
| Case | Finally | Return | void |
| Catch | For | Switch | while |
| Continue | Function | This | with |
| Default | If | Throw |  |
| Delete | In | Try |  |
| Do | Instanceof | Typeof |  |
|  |  |  |  |

--保留字

保留字不可以用作变量名

|  |  |  |  |
| --- | --- | --- | --- |
| Abstract | Enum | Int | short |
| Boolean | Export | Interface | static |
| Byte | Extends | Long | super |
| Char | Final | Native | synchronized |
| Class | Float | Package | Throws |
| Const | Goto | Private | Transient |
| Debugger | Implements | Protected | Volatile |
| Double | Import | Public |  |
|  |  |  |  |

1. 基本数据类型：

Number类型、string类型、Boolean类型、null类型（object类型）、undefined类型

**\*1.自动转换类型：**

1. 任何数据和字符串类型的数据做加法操作时，其他数据会自动的转化为字符串类型，此时不是数学意义上的相加，而是拼接。

如：var num1 = 1；var num2 =“2”；var result = num1 + num2；document.write(result);//12

B.任何非数字类型的数据之间做非加法操作时，此时数据会自动的转化为数字。

如：var num1 = true；var num2 =“1”；var result = num1- num2；document.write(result);//0

C.任何数据和数字类型的值NaN做算术运算操作时（除了和字符串做加法操作之外），所得的结果始终是NaN，包裹NaN自己与自己做任何运算操作都是NaN。

如：var num1 = 1；var num2 =“2a”；var result = num1 - num2；document.write(result);//NaN

小工具：isNaN(数据)；用来判断数据（自动转化number类型）是（不是一个数字）返回值为Boolean类型

true表示不是一个数字；false表示是一个数字。

**\*2.强制类型转换：**

1. 其他数据转数字

parseInt（数据）；取整//只转化开头的数字整数部分，没有数字为NaN

如：parseInt（“3.14abc”）=3；parseInt（“abc3.14”）=NaN；parseInt（null）=NaN；parseInt（undefined）=NaN；

parseFloat（数据）；浮点数//只转化开头的数字部分，没有数字为NaN

如：parseFloat（“3.14abc”）=3.14；parseFloat（“abc3.14”）=NaN；

Number（数据）；//纯数字的字符串能转化，否则是NaN。

如：Number（“3.14”）=3.14；Number（“3.14abc”）=NaN；Number（null）=0；Number（undefined）=NaN；

1. 其他数据转字符串（拼接）

String（数据）；

1. 其他数据转Boolean类型

Boolean（数据）；

数据-->Boolean 只有0-->false；其他-->true

字符串-->Boolean 只有空字符串（“”）-->false；其他（有值即可）-->true

null-->Boolean false

undefined-->Boolean false

(扩展: alert(undefined==null)-->结果是true，因为undefined值是派生自null值的)

1. 运算符

赋值运算符： =

算术运算符： + - \* / % += -= \*= /= %= ++ --

关系运算符（运算后始终返回Boolean类型的值）： > < == != >= <= === !==

1. 代码规范：

代码缩进；

以“；”结尾；

= ，+ 两边要加空格；

小括号的嵌套要加空格；

双引号和单引号的嵌套; 如： ” ’’ ”，’ ”” ’

# JS基础(8月2日)

**一、逻辑运算符**

1. && 且

如果两个非Boolean类型的值做且运算， 若第一个条件为true，则返回第二个条件的结果，否则会返回第一个条件的结果。

如：var result = 1&&0; //0

var result = 0&&1; //0

var result = 1&&2; //2

Var result = 2&&1; //1

2.|| 或

如果两个非Boolean类型的值做且运算， 若第一个条件为true，则返回第一个条件的结果，否则会返回第二个条件的结果。

如：var result = 1||0;//1

var result = 0||1;//1

var result = 1||2;//1

var result = 2||1;//2

1. ! 非

对任何数据取反，返回结果始终都是Boolean类型的值。

如：var result = !1;//false

var result = !””;//true

\* ! 的优先级要大于关系运算符的优先级

关系运算符大于逻辑运算符（除了!）

如：var age = 18;var gender = “男”;var flag = !age>18||!gender==”男”;

（有括号先算括号内的） False>18->false false==”男”->false

二、选择语句（分支语句）

var num1 = 6;

if(num1%2==0){

alert(num1 + "是偶数");

}

1.单选语句 语法

If(条件表达式){

//条件表达式的值为true-->执行代码

}

var age = 3;

var gender = "男";

if(age>=18&&gender=="男"){

alert("欢迎光临！O(∩\_∩)O哈哈~");

}else{

alert("小屁孩儿，一边玩去！");

}

2.双选语句 语法

If(条件表达式){

//条件表达式的值为true-->执行代码

}else{

//条件表达式的值为false->执行代码

}

三目运算符或三运表达式

条件表达式？结果一：结果二；如：a%2==0？alert(“偶数”)：alert(“奇数”);

3.多选语句 语法

var day = 6；

switch(day){

case 1:

alert("星期一");

case 2:

alert("星期二");

break;

case 3:

alert("星期三");

break;

case 4:

alert("星期四");

break;

case 5:

alert("星期五");

break;

case 6:

alert("星期六");

break;

case 7:

alert("星期日");

break;

default:

alert("给的什么玩意儿");

}

If(条件表达式){

var score = 50;

if(score>=90){

alert("优");

}else if(score>=80){

alert("良");

}else if(score>=70){

alert("还可以");

}else if(score>=60){

alert("还凑合");

}else{

alert("留级！");

}

//执行代码

}else if(条件表达式){

//执行代码

}...

else{

//执行代码

}

4.switch语句

Switch(值){

case 值1:

//执行代码；

break;

case 值2:

//执行代码；

break;

...

default:

//执行代码；

}

//break可以省略，仅仅是语法不报错,但会改变执行逻辑

\*多选和switch的区别

多选不仅可以做等值判断还可以做其他判断

switch仅仅只能做等值判断,在做等值判断多的情况下用switch,因为switch速度快

\*如果对于选择语句只有一行语句的话，则可以省略{}，否则省略会报错

if(true)//不提倡使用这种装逼的格式

document.write("我们六一儿童节快乐");

//document.write("我们六一儿童节快乐");//会报错

else

document.write("happy");

document.write("happy2");//该条语句不会被包裹在else语句当中，而是在else语句之外

# JS基础(8月3日)

**补充：**对于后++直接和本身做运算的话，后面的本身会先完成之前的自增，然后再参与运算。

var a = 1;

**//var result = a++ + a; //3 1 + 2**

**var result = ++a + a; //2 + 2**

**alert(result);**

**一、循环语句**

var a = 1;

while(a<=5){

document.write( a + " ");

a++;//a = a + 1;

}//1 2 3 4 5

1. while循环

语法：while(条件表达式){

//循环体

}

var num = 1;

while(num<=10){

if(num==5){

break;

}

document.write(num + "<br/>");

num++;

}

关键字break;

作用：结束循环体;

例如：打印1到100之间的数字，

若遇到50则结束循环----------------->

var num = 1;

while(num<=100){

if(num>=50&&num<=60){

num++;//不能忘否则会死循环

continue;

}

document.write(num + "<br/>");

num++;

}

关键字：continue;

作用：结束本次循环，

并执行下一次循环。

例如:打印1到100之间的数字,

不打印50-60之间的数字,

包括50和60---------------------------->

2.for循环

语法：for(初始化变量;条件表达式;改变初始化变量的值){

//循环体

}

for(var num = 1;num<=10;num++){

document.write("<h1>" + num + "</h1>");

}

1. do-while循环

var num = 1;

do{

document.write(num + "<br/>");

num++;

}while(num<=100);

语法：do{

//循环体；

}while(条件表达式)

While循环与do-while循环的区别:

除了格式上的区别外，还有do-while一定会先执行一次循环体内的代码，在判断条件表达式是否为真，为真继续循环，为假跳出循环。

而while是先判断条件表达式是否为真，进入循环体执行代码，为假跳出循环。

Var a = 1;//当前行

Var b = 5;//总行数

While(a<=b){

Var c = 1；//星星个数

While(c<=a){

Document.write(“☆”);

C++;//改变要打印的星星

}

Document.write(“<br/>”)//换行

A++;//改变行数

}

1. 循环嵌套

☆

☆☆

☆☆☆ ------------->

☆☆☆☆

☆☆☆☆☆

# JS基础(8月4日)

1. 什么是函数？--------------->工具
2. 函数的作用是什么？

可以反复调用从而减少代码量

1. 函数的特性：封装性。

（函数不调用是不会执行的）

1. 语法

关键字：function

函数声明方式：function 函数名(){

//函数体  
 }

函数表达式： var 函数名 = function (){

//函数体（匿名函数）

}

\*5.程序预编译

程序预编译 -> 程序的执行

程序预编译：

先准备：

变量声明提升：

会把程序当中用var关键字所声明变量名提升到当前执行环境的顶部。

注意：仅仅是提升变量名。 赋值 并没有提升

函数提升:

会把程序中用function关键字 【函数声明的方式】 的函数统一提升 到当前执行环境的顶部(统一：把整体提升到当前执行环境的顶部)

6.事件驱动函数

我们在网页中，因为触发某些行为而调用的函数，就是事件驱动函数;

常见的事件有：

onclick、onmouseover、onmouseout、onkeyup...等

7.函数的参数

语法：无参数function 函数名(){

//函数体  
 }

有参数 function 函数名(参数1,参数2...){

//var 参数1;var 参数2;...  
 }

Function add(a,b,c,d){

Var result = a + b + c;

Document.write(result);//6

Alert(d);//undefined

Alert(Arguments.length);//3个实参

}

Add(1,2,3);

Alert(add.length);//4个形参

在函数定义的时候括号内的参数就是形参；

在函数调用的时候括号内的参数就是实参；

当实参的个数少于形参,后面的形参则为undefined

函数名.length ---------->获取形参的个数；

Arguments.length------>获取实参的个数；

(要写在函数内部)

Console.log();在控制台显示；

1. 关键字：return

function ji(num1,num2,num3,num4,num5){

var result = num1\*num2\*num3\*num4;

return result;

alert("我还会被执行吗？); //不会被执行

}

document.write(ji(1,2,3,4,5) );//24

作用:1、把函数内的结果返回出去

1. 结束当前函数程序的执行
2. 作用域：

全局作用域：

定义变量为全局变量；可以作用在任何地方；

局部作用域：

局部变量；仅仅只能在当前的作用域中使用；

一个函数就是一个独立的作用域，称之为局部作用域

Var a = 123;

Function test(){

//var a;

Document.write(a);//局部变量undefined

Var a = 456;//a=456;

Document.write(a);//局部变量456

}

Test();

Document.write(a);//全局变量123

1. JS没有块级作用域（扩展）

也就是说：选择语句和循环语句内部的变量，外部是可以访问的。

1. with(对象){

//了解就行 对象就是一个工具库

//严格模式下不能使用

}

如：with(document){

Function age(index){  
 if(index==1){

Return 10;

}else{

Return age(index-1) + 2;

}

}

Alert(Age(5) );//18

write("Hello World");

}

1. 递归

程序调用自身的编程技巧称为递归。

例如：10 12 14 16....

# JS基础(8月5日)

**一、编程思想**

1. 面向过程：是一种一过程为中心的编程思想。

2.面向对象opp（Object Oriented Programming）

是一种以事物为中心的编程思想。（用对象！）

核心思想：将真实世界存在事物的复杂关系，抽取成一个个不同的对象，然后由这些对象之间的分工与协作，完成对真实世界的模拟。

类：抽象的；对象：客观存在的；关系：对象是类的一个实例。

**二、对象**

1.对象：可以理解为工具库

2.对象的创建：

new 关键字 用来创建对象

语法：

var 对象名 = new Object();

Var 对象名 = {属性1：值，属性2：值...};//常用的方法

对象组成部分

键/值 或 属性/值

静态属性 特征

动态属性 行为

3. 同类型的对象不相等；因为对于【对象】所比较的是空间地址，总而言之，对象之间是不同的个体。

var obj1 = new Object(); //obj1 类型Object

obj1.uname = "张三"; //会画画

var obj2 = new Object(); //obj2 类型Object

var flag = obj1==obj2; //对于【对象】所比较的是空间地址

alert(flag); //false

alert(obj1.uname); //张三

alert(obj2.uname); //undefined 对于对象，若内部中没有指定属性，则返回undefined

1. 对象的操作

var obj = new Object(); //对象 obj

obj.uName = "陈寅恪";//增加对象属性 obj[“uName”] = “陈寅恪”;

obj.talk = "独立之精神，自由之思想"; obj[“talk”]=“独立之精神，自由之思想”;

alert(obj.uName);//陈寅恪//查找对象属性

obj.uName = "ChenYinKe";//修改对象属性

alert(obj.uName);//ChenYinKe

document.write(obj.talk + "<br/>");//独立之精神，自由之思想

delete obj.talk;//删除对象属性

document.write(obj.talk + "<br/>");//undefined

Var a =” uName”; obj[a] = “张国伟”;

Alert(obj[a]);//张国伟

**三、数组**

1. 数组是一种数据，属于引用类型数据；

2. 语法： var 数组名 = [];//空数组

var 数组名 = [数据1，数据2，数据3...];

Var 数组名 = new Array();//不常用！

1. 数组的作用：方便管理和操作数据。
2. Arr[0]会自动的转化为arr[“0”];但不能写arr.0

数组的长度：数组名.length

遍历数组

Var a = [1,2,3,4,5,6,7,8,9];

Var sum = 0;

For(var i = 0,len = a.length;i<len;i++){

Sum+=a[i];

}

Var avg = sum/len;

Document.write(sum);//总数

Document.write(avg);//平均数

Document.write(len);//个数

5. var i = 0,len = a.length;在for循环中,优化程序,对于数组的个数来说,计算机仅仅只算了一次

6. 对象名 instanceof 类型 //判断一个对象是否属于某个类型；返回值为Boolean值。

Var arr = [];

Var flag = arr instanceof Array;//true

Var flag = arr instanceof Object;//true

alert(typeof arr);//object

对于基本数据类型类型检测，用typeof

对于引用数据类型检测用 instanceof

1. 除for之外的数组遍历

For in

Var arr = [1,2,3,4,5,6,7,8,9];

for(var a in arr){

document.write(a);//0 1 2 3 4 5 6 7 8 索引

Document.write(arr[a]);//1 2 3 4 5 6 7 8 9值

}

forEach

arr.forEach(function (a,b,c){

document.write(a);-->值

document.write(b);-->索引

document.write(c);-->数组本身

});

1. 回调函数：

Function test(a){

a(“我明白了！”);

}

Var say = function (talk){

Alert(talk);

}

test(say);