day01

Javascript Basic

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、Javascript 概述(了解)

2、Javascript 基础语法(重点)

3、变量和常量(重难点)

4、数据类型

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一、Javascript 概述

1、什么是Javascript

Javascript，简称为 JS,是一种运行与JS解释器/引擎中的解释型脚本语言

运行环境：

1、独立安装的JS解释器(NodeJS)

2、嵌入在[浏览器]内核中JS解释器

解释型：运行之前是不需要编译的，运行之前不会检查错误，直到碰到错误为止。

编译型：对源码进行编译，还能检查语法错误

JS使用场合：PC机，手机，平板，机顶盒

2、JS发展史

1、1992年，Nombas 为自己的软件 开发了一款脚本语言 ScriptEase，可以嵌入在网页中的。

2、1995年，Netscape(网景) 开发了 LiveScript,更名为 Javascript，与Java 没有任何关系的。

3、1996年，Microsoft ， IE3.0 发布了 JS克隆版 --> JScript

4、1997年，JS1.1版本 被作为 草案 提交给了 ECMA(欧洲计算机制造商协会)，ECMAScript(ES)标准

3、JS组成

完整的JS是由三部分组成：

1、核心(ECMAScript)

2、文档对象模型(DOM,Document Object Model)

让JS有能力与网页进行对话

3、浏览器对象模型(BOM,Browser Object Model)

让JS有能力与浏览器进行对话

4、JS特点

1、开发工具简单，记事本即可

2、无需编译，直接由JS引擎负责执行

3、弱类型语言

由数据 来决定 数据类型

4、面向对象

5、js用途：

1、客户端(浏览器)

1、客户端数据计算 2、表单控件输入验证

3、浏览器的事件触发和处理 4、网页特效

5、服务器异步数据提交

2、服务器端

1、分布式运算 2、实时服务器 3、窗口应用 4、网络应用

2、JS基础语法

1、浏览器内核

1、作用

一、负责页面内容的渲染。

二、内核主要由两部分组成：

1、内容排版引擎 解析HTML和CSS

2、脚本解释引擎 解析Javascript

1、Microsoft IE

内核：Trident

内容排版引擎：-

脚本解释引擎：Chakra

2、Mozilla Firefox

内核：Gecko

内容排版引擎：-

脚本解释引擎：Monkey(猴子)系列

3、Apple Safari

内核：Webkit

内容排版引擎：Webcore

脚本解释引擎：Nitro

4、Google Chrome

内核：Webkit

内容排版引擎：Webcore

脚本解释引擎：V8

5、Opera

内核：Presto

内容排版引擎：-

脚本解释引擎：Carakan

2013年之后：

内核：Webkit

内容排版引擎：Webcore

脚本解释引擎：V8

2、JS运行环境

一、独立安装JS引擎

NodeJS

console.log("Hello World");

向控制台 打印一句话

二、嵌入在浏览器内核中的引擎

1、直接在浏览器Console中输入并执行JS

2、将JS脚本嵌入在HTML页面中执行

1、将JS代码嵌入在元素"事件"中

1、事件

onclick : 当单击元素时所做的操作

<div id="" onclick="JS代码">xxx</div>

2、将JS代码嵌入在<script>标记中

1、<script></script>

允许出现网页的任意位置处

3、将JS代码写在外部脚本文件中(\*\*.js)

1、创建 js 文件，并编写JS代码

\*\*\*.js

2、在页面中 引入 js文件

<script src="js文件路径"></script>

注意：<script src=""></script>该对标记中，是不允许出现任何内容的

<script src="a.js">console.log();</script>这段代码是错误的。

3、JS调试

1 、JS的代码错误

解释型语言：如果某行代码出现错误，则终止解释器的此次执行，但是不会影响后续块的执行，以及后续的HTML元素的解析。

4、语法规范

一、语句：允许被JS引擎所解释的代码

1.使用 分号 来表示结束

console.log();

document.write();

2.大小写敏感

console.log(); 正确

Console.log(); 错误

3.英文标点符号

console.log(""); 正确

console.log（“”）；错误

由表达式、关键字、运算符 组成

2、注释

单行注释: //

多行注释: /\* \*/

3、变量 和 常量

1、变量

1、什么是变量

内存：保存程序在运行过程中，所需要用到的数据。

变量：就是内存中的一段存储空间

变量名：内存空间的别名，可以自定义

值：保存在 内存空间中的数据

2、变量的声明

1、语法

声明：

var 变量名;

赋值：

变量名=值;

注意：声明过程中，尽量不要省略 var 关键字，否则声明的是"全局变量

1. 问题 ：向银行存100块钱，每年的利息是10%,求：10年之后本息是多少??

8bit = 1byte

1024byte = 1KB

1024KB = 1MB

1024MB = 1GB

1024GB = 1TB

ex:

1、声明一个变量，保存 学员的年龄,值为 25

var age;

age=25;

声明并直接赋值：

var 变量名=值;

ex:var age=25;

使用：

在 console.log() 或 document.write() 使用变量名 取代 ""

ex:打印 变量 stuName 的值到控制台上

var stuName="SF.ZH";

console.log(stuName); -- SF.ZH

console.log("stuName"); -- stuName

2、一次性声明多个变量并赋值

语法：

var 变量名1,变量名2,...变量名n;

var stuName="SF.ZH",stuAge=25,stuGender,stuHeight;

等同于

var stuName="SF.ZH";

var stuAge=25;

var stuGender;

var stuHeight;

3、变量名 命名规范

1、不允许使用JS的关键字和保留关键字

2、由字母、数字、下划线以及$组成

var stuName;

var stuName1;

var stu\_name;

var stu$name;

3、不能以数字开头

var stuName1;

var 1stuName;错误

var $stuName;正确

4、尽量见名知意

var a,b,c,d,ab,ac,ad;

var stuName,stuAge; 更推荐

5、可以采用"匈牙利命名法","驼峰命名法","下划线命名法"

1、匈牙利

控件简写+作用

txtName

2、驼峰命名法

变量名为合成词时，第一个单词全小写，从第二个单词开始，每个单词首字符变大写

var stuName;

如果只有一个单词作为变量名，全小写

var age;

3、下划线命名法

var \_age;

4、变量的使用

1、未经初始化的变量

ex:

var stuName="张三丰";

console.log(stuName);

var stuAge;

console.log(stuAge);

变量声明后，从未赋值，称之为 未经初始化变量，值为 undefined

2、使用 未声明的变量

变量未被定义过，直接打印或使用

console.log(stuGender);

结果为错误

3、对变量进行的存取操作

1、获取变量的值-GET操作

只要变量名不出现在=的左边，都是取值

var stuName="SF.ZH";//赋值

console.log(stuName);//取值

var oldName=stuName;

针对 stuName 来讲：取值

针对 oldName 来讲：赋值

2、保存(设置)变量的值-SET操作

赋值运算符：=

变量出现在 = 的左边，一定是赋值

var stuAge=25;

var stuName="sf.zh";

var oldName=stuName;

注意：=出现的话，永远都是将=右边的东西保存给=左边的变量。

=左边必须是变量

var stuAge=30;//将 30 保存在 变量 stuAge中

35=30;//将30保存在35中,错误!!!!

3、常量

1、什么是常量

量：数据

常量：一旦声明好，就不允许在程序运行过程修改的数据

表示 常理性数据的时候可以用 常量表示

ex:

π

一周七天

一年365天或366天

2、声明常量

const 常量名=值;

注意：

1、要求在声明常量时必须赋值

2、通常情况，常量名尽量全大写

3、练习

声明一个变量(r)，用来表示 一个圆的半径

计算该圆的周长，并且保存在变量l中，打印输出 l=2πr l=2\*π\*r;

计算该圆的面积，并且保存在变量s中，打印输出 s=πrr s=π\*r\*r;

π 要用 常量来表示

4、数据类型

1、问题

要保存一个人的年龄，要在内存中开辟多大的空间

var age=89;

要保存一个人的姓名，要在内存中开辟多大的空间

2、数据类型分类

1、原始类型(基本类型)、

分为五种类型：

数字类型 字符串类型 布尔类型 空 未定义

1、数字类型

number类型

可以表示32位的整数以及64位的浮点数

整数：32位即4字节

浮点数：即小数，64位，8字节

整数：

十进制：生活中常用数字

八进制：逢八进一 var n1=0123;

十六进制：逢16进1 0-9 A-F 组成 var n2 = 0x123;

小数：

var n1 = 34.56;

var n2 = 4.5e10;

2、字符串类型

string 类型

表示一系列的文本字符数据

由Unicode字符，数字，标点组成

Unicode 下所有的 字符，数字，标点 在内存中 都占2字节

"张".charCodeAt().toString(16)

查看 字符 "张" 的 十六进制 表现方式

结果为：5f20

var s="\u5f20";将 16进制数字 转换哪位字符串

\u4e00 : 汉字的起始字符

\u9fa5 : 汉字的结束字符

转义字符：

1、\n换行 2、\r回车 3、\t一个制表符4、\" "

5、\' ' 6、\\

\

3、布尔类型

boolean类型

作用：用于表示 条件的结果

取值：true : 真，肯定的结果

false : 假，否定的结果

ex

var r1 = true;

var r2 = false;

除条件判断外，做运算时，true可以当做1运算，false当做0运算

4、空

声明对象未赋值

null

5、未定义

undefined

1、声明变量未赋值

2、访问对象不存在的属性

2、引用类型

...

Day02

1、数据类型转换

2、运算符和表达式

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、数据类型转换

1、什么数据类型转换

弱类型：由数据来决定变量的数据类型是什么

var stu;//undefined

stu=25;//number

stu="25"//string

不同数据类型的数据在运算的过程中，如何处理？？

var num1=15;//number

var num2=13;//number

var result=num1+num2;//28 -> number

var str1="15";//string

var result1=num1+str1;

2、隐式转换

自动转换，由JS在运算过程中，自己进行转换的操作，不需人为参与

1、函数 或 运算符

typeof() 或 typeof

ex:

var num1 = 15;

var s = typeof(num1); : 获取 num1 的数据类型

var s1 = typeof num1; : 获取 num1 的数据类型

2、NaN

Not a Number

不是一个数字

isNaN(数据) : 判断 数据是否为 非数字

是不是一个数字

结果为 boolean 类型

结果为 true ：不是一个数字

结果为 false：是一个数字

3、所有的数据类型 与 string 做 + 运算时，最后的结果都为 string

3、转换函数 - 显示转换(强制转换)

1、toString()

将任意类型的数据转换为 string 类型

变量.toString();

会得到一个全新的结果，类型为 string

var num=15;

var str=num+"";

2、parseInt()

整型：Integer

作用：获取 指定数据的 整数 部分

语法：var result = parseInt(数据);

注意：parseInt ， 从左向右 依次转换，碰到第一个非整数字符，则停止转换。如果第一个字符就是非整数字符的话，结果为 NaN

3、parseFloat()

Float ：浮点类型->小数

作用：将 指定数据转换成 小数

语法：var result = parseFloat(数据);

ex :

var result=parseFloat("35.25"); //35.25

var result=parseFloat("35.2你好!");//35.2

var result=parseFloat("你好35.2");//NaN

var result=parseFloat("35.");//35

var result=parseFloat(".35");//0.35

var result=parseFloat("35.0");?

4、Number()

作用：将一个字符串解析为number

语法：var result=Number(数据);

注意：如果包含非法字符，则返回NaN

2、运算符&表达式

1、运算符和表达式

运算符：能够完成数据计算的一组符号

比如：+,-,\*,/,... ...

表达式：由运算符和操作数所组成的式子叫作表达式，每个表达式都有自己的值

15+18;

var a=15;//a 变量的值为 15，整个表达式的值 也为15

2、运算符

1 、算术运算符

+,-,\*,/,%,++,--

一、+ : 可以表示加法，也可以用于字符串连接

+ 两边的操作数如果都是 number类型或boolean的话，那么所执行的就是加法操作

如果两边的操作数，有一个是string类型的话，那么执行的操作就是连接

ex:

var num1 = 15;

var num2 = 18;

var str1 = "15";

console.log(num1+num2);//33

console.log(num1+str1);//1515

二、 - : 可以表示减法操作，也表示负数的声明

var num1 = -15;

三、\* :

ex: 15\*2 结果：30

四、/ :

ex: 15/2 结果：7.5

五、% : 取余操作，俗称 模

作用：取两个数字的余数

ex: 5 % 2 结果：1

使用场合：

1、判断数字的奇偶性

var num = 33;

判断 num % 2 的结果 是否为 0，如果为0则为 偶数，否则则为 奇数

2、获取 数字的最后几位

var num = 1234;

问题：想获取 num 的最后一位

num % 10 结果为 ：4

问题：想获取 num 的最后两位

num % 100 结果为：34

注意：如果 -\*/% 的两端都是 数字 形式的字符串，那么所做的操作依然是算术操作

ex:

var num1 = "15";

var num2 = "18";

var r1 = num1+num2;//1518

var r2 = num1-num2;//-3

自增 & 自减

一、++ : 自增，在数值的基础上，进行+1操作

var num = 10;

num++;

解释：++在操作数的后面，先使用num的值，然后再对num进行自增

ex:

var num=10;

console.log(num++);//先输出num的值，然后再对num进行自增,所以该行的结果为 10

console.log(num);//该行的结果为 11

++num;

解释：先对num进行自增，然后再使用num的值

二、-- : 自减，在数值的基础上，进行-1操作

3、关系运算符

1、作用：比较数据之间的大小关系

2、运算符

> : 大于

< : 小于

>= : 大于等于

<= : 小于等于

== : 判断等于 注意：不比较类型，只比较数值

!= : 不等于

=== : 全等 注意：除数值之外，连同类型也会一起比较

!== : 不全等

由关系运算符所组成的表达式称之为"关系表达式"

关系表达式的运算结果为 boolean类型(true 或 false)

3、判断一个 数据 是否为 数字

var input=prompt("请输入一个数据：");

判断 input 是否为 数字??

isNaN(数据)

isNaN()会抛开数据类型来判断数据是否为数字

如果 数据 是数字类型，则返回 false

如果 数据 不是数字类型，则返回 true

4、逻辑运算符

1、作用：关联条件

ex:

判断 考试分数 是否在 60 ~ 80 之间

var score = 52;

条件1：score >= 60

条件2：score <= 80

2、逻辑运算符

逻辑非 : !

逻辑与 : &&

逻辑或 : ||

2.1 逻辑与(&&)

关联两个条件，两个条件都为真的时候，那么整个表达式的结果才为真

语法：条件1 && 条件2

问题：判断 考试分数 是否在 60 ~ 80 之间

var score = 52;

条件1：score>=60

条件2：score<=80

var result = score>=60 && score<=80;

2.2 逻辑或(||)

关联两个条件，两个条件中，只要有一个为真，那么整个表达式的结果就为真

问题：笔试题和面试题,笔试题或面试题的分时，有一个超过60就认为是合格

var bishi = 52;

var mianshi = 82;

条件1：bishi >= 60

条件2：mianshi >= 60

var result = bishi >= 60 || mianshi >= 60;

2.3 逻辑非(!)

作用：对条件取反

注意：逻辑非，只有一个操作数

语法：!条件

非真即假 非假即真

3、短路逻辑

1、短路&&

如果第一个条件的结果已经为 false 的话，那么就不会再判断第二个条件,那么整个表达式的值，就是 false

如果第一个条件的结果是true的话，则继续判断(执行)第二个条件(表达式)，并且以第二个表达式的值，作为整个 表达式的值。

2、短路 ||

如果第一个条件的结果已经为 true的话，那么就不会再判断第二个条件，那么整个表达式的值，就是true

如果第一个条件的结果是false的话，则继续判断(执行)第二个条件(表达式),并且以第二个表达式的值，作为整个表达式的值。

Day03

1、运算符

1、位运算符

1、二进制

2、运算符

十进制换成二进制（可以使用计算器）

0 : 0 1 : 1 2 : 10 3 : 11 4 : 100

5 : 101

**1、&(按位与)**

定义：将两边的操作数转成二进制，对应数字的每位上的数字，都为1的时候，该位结果才为1，否则结果为0

ex:

3 & 5

3 : 011

5 : 101

---------------

001

结果 为 1

使用场合：判断一个数字的奇偶性，n & 1,如果结果为 1，则为奇数，否则为偶数

**2、|(按位或)**

定义：将两边的操作数转换成二进制，按位比较，对应位上的两个数字，只要有一个为1，结果就是1，否则就是0

ex

1 | 3

1 : 01

3 : 11

---------

11

使用场合：针对小数，向下取整时使用

n | 0

3、**^(按位异或)**

两个数的二进制中，对应位的数字只有一个为1时，该位结果才为1，否则 则为0

ex: 5^3

5:101

3:011

-------

110

使用场合：在不借助任何变量的情况下，交换两个数字

Ex:

var num1 = 8;

var num2 = 5;

//借助第三方变量

var temp=num1;

num1 = num2;

num2 = temp;

//使用异或运算符

num1 = num1 ^ num2;

num2 = num2 ^ num1;

num1 = num1 ^ num2;

8 : 1000

5 : 0101

num1 = num1 ^ num2;

8 : 1000

5 : 0101

---------------

1101 ==> 13

num2 = num2 ^ num1;

8 : 1000

13: 1101

---------

0101 ==> 5

num1 = num1 ^ num2;

13 : 1101

5 : 0101

--------------

1000

**4、<<(按位左移)**

将 二进制数 向左移动几位，后边以0补位

0000 0000 0000 0000

0000 0000 0000 0010

ex:2<<1

0000 0000 0000 0000

0000 0000 0000 0100

结果为：4

ex:2<<2

0000 0000 0000 0000

0000 0000 0000 1000

结果为：8

ex:2<<3

结果为：16

场合：对数字做乘2的操作

**5、>>(按位右移)**

将 二进制数 向右移动几位，前边以0补位

ex:16>>1

0000 0000 0000 0000

0000 0000 0000 1000

结果 ：8

ex:2>>1

0000 0000 0000 0000

0000 0000 0000 0001

场合：做对数字做除2的操作

2、赋值运算

**1、=**

var x=y=z=123;

正确，但不推荐

**2、扩展的赋值运算符**

**+=，-=，\*=，/=，%=**

**^=,**

var a = 10;

a += 5;==> a = a + 5;

a -= 5;==> a = a - 5;

注意：推荐 n+=m 的方式 取代 n=n+m,因为效率要高

**3、条件运算符**

1、三目运算符

单目运算符：运算符只需要一个操作数

!,++,--,typeof

双目运算符：运算符需要两个操作数

+,-,\*,/,%,&&,||,^,&,|,<<,>> ...

三目运算符：运算符需要三个操作

2、语法

运算符：?:

**表达式1?表达式2:表达式3;**

解析：表达式1是一个条件，值为boolean类型

若表达式1的值为 true，则执行表达式1的结果，否则则执行表达式2的操作，并且以表达式2的结果作为整个表达式的结果

若表达式1的值为false，则执行表达式3的操作，并且以表达式3的结果作为整个表达式的结果

3、条件运算符的嵌套

输入一个考试成绩，判断

成绩 == 100 ，则输出 恭喜，得了满分

如果成绩 在85以上，则输出 优秀

如果成绩 在75以上，则输出 良好

否则 ：成绩一般

举例：/\*条件运算符的嵌套\*/

var score = prompt("请输入考试成绩:");

var result = score==100?"恭喜":(score>=85?"优秀":(score>=75?"良好":"成绩一般"));

console.log(result);

4、优先级

从高 -> 低

1、()

2、单目运算符:++,--,!；

3、\* / %模

4、+ -

5、<<,>>

5+1>>2 : 先算5+1=6，将6右移2位

6、关系运算符:<,>,<=,>=

7、关系运算符:==,!=,===,!==

8、&

9、^

10、|

a|b&c^d

1、先算 b&c 的结果

2、b&c的结果 和 d 进行 异或操作

3、b&c和d异或的结果 在和a 做 |

11、&&

12、||

13、?:

14、赋值运算符

2、函数

1、什么是函数

函数(function)，也可以被称之为方法,或者 过程

定义：是一段预定义好，并可以被反复执行的代码块。其中可以包含多条可执行语句。

预定义好:事先声明好，但不被执行

反复使用：允许被多个地方(元素，函数中)所应用

代码块：允许包含多条可执行的代码。

函数本质上是功能完整的 对象

2、函数的声明

语法：

function 函数名(){

//可执行的代码块

}

ex ：创建一个函数 名称:sayHello,主体功能为，向控制台上 输出一句 Hello World

function sayHello(){

console.log(Hello World)

}

3、函数的调用

语法：函数名称()

执行函数中的内容

调用场合：

事件中调用<Button></Button>

<script>

任何 JS 的合法位置处，都允许调用函数

4、定义函数的参数和返回值

转换函数

parseInt(数据);//将指定数据转换为 整数

parseFloat(数据);//将指定的数据转换为 小数

console.log("要打印的话");

1、定义带参数函数

function 函数名(参数列表声明){

//代码块(函数体，功能体，方法体)

}

参数列表：由一或多个 变量名称来组成

声明函数时定义的参数，可以称之为叫作 "形参"(形式参数)

ex:

1、创建一个 函数，要求传递过来一个参数，进行打印

function print(msg){

console.log(msg);

}

调用带参函数：

语法：函数名称(参数值列表);

print("Hello World");

在调用函数时所传递的参数值，被称之为"实参"(实际参数)

举例

2、创建一个函数，接受两个参数，函数体中，完成两个参数值相加的运算，并打印输出结果

第一步：声明：

function add(num1,num2){

console.log(num1+num2);

}

第二步：调用：add(15,18);

2、定义带返回值的函数

var r = parseInt("35.5");

第一步、声明

function 函数名(0或多个参数){

//代码块;

return 值;

}

第二步、调用

var 变量 = 函数名(参数);

3、ex

//编写一个函数，名称为 calResult，接收2个数字作为参数，

并且在函数中计算两个数字的和，并返回。

function calResult(n1,n2){

var result = n1 + n2;

return result;

}

<button onclick="console.log(calResult(15,18));">调用函数</button>

function testCalResult(){

/\*主体功能：调用 calResult 函数

并且传递参数，得到返回值，打印输出\*/

var r = calResult(10,20);

console.log("结果为:"+r);

}

<button onclick="testCalResult();">调用函数</button>

5、变量的作用域

1、什么是作用域

作用域就是变量或函数的可访问范围。它控制着变量或函数的可见性和生命周期。

在JS中，变量或函数的作用域可分为：

1、函数作用域，只在当前函数内可访问

2、全局作用域，一经定义，代码的任何位置都可以方式

2、函数作用域

函数作用域中的变量，可以称之为 "局部变量",只能在当前函数中可访问，离开该范围，则失效

3、全局作用域

全局作用域中的变量，称之为 "全局变量",在代码的任何位置处都能访问

声明位置或方式：

1、独立于任何function的位置处声明 变量

2、声明变量时不使用var关键字

4、声明提前

JS在正式执行之前，会将所有var声明的变量和function声明的函数，预读到所在作用域的顶部，但是，对变量的赋值，还保留在原来的位置处

5、按值传递

传参时，实际上是将 实参 复制了一份副本传给了函数。

在 函数体内 对变量进行修改，实际上是不会影响到外部的实参变量的。

Ex:

<script>

function change(num1,num2){

var temp = num1;

num1 = num2;

num2 = temp;

console.log("交换后：");

console.log("num1="+num1);//18

console.log("num2="+num2);//15

}

function testChange(){

var a = 15;

var b = 18;

change(a,b);

console.log("调用change函数后的值：");

console.log("a="+a);//15

console.log("b="+b);//18

}

</script>

<button onclick="testChange()">交换数字</button>

1、函数

1、函数的作用域

函数与变量类似，函数的作用域也分为：

1、全局作用域(全局函数)

2、函数作用域(局部函数)

2、全局函数

ECMAScript预定义的全局函数，可供用户直接使用

1、parseInt()

parseInt("35.5");

2、parseFloat()

3、Number();

4、isFinite()

作用：判断指定的数据是否有有限制

isFinite(5);判断 数字类型 的数据 是否有有限制，结果为 true

isFinite("a");判断 字符串类型 的数据 是否有有限制，结果为 false

5、encodeURI()

对 URI 格式的字符串进行编码，返回编码后的字符串

URI : Uniform Resource Identifier

统一资源标识符，负责定义 URI格式

URL : Uniform Resource Locator

统一资源定位器，符合URI规范的一个具体的地址。

6、decodeURI()

对 已编码的字符串进行解码

7、eval()

作用：用于执行以string形式出现的JS代码

语法：eval(js代码);

eval()函数也可以用于将JSON字符串解析为JS对象。

2、分支结构

1、程序的流程控制

程序模拟人脑的想法 如果怎么样 那就怎么样

程序=数据 + 算法

程序的三种结构：

1、顺序结构

2、分支结构

3、循环结构

2、分支结构

1、什么是分支结构

程序在运行过程中，根据不同的条件，选择执行某些语句

1、if-结构

当条件满足时，运行某些语句。

当条件不满足时，则不运行这些语句。

2、if-else结构

当条件满足时，运行某些语句

当条件不满足时，运行另外一些语句。

2、if-结构

1 、语法

if(条件表达式){

语句块;

}

流程：

1、判断 条件表达式的结果

2、如果结果为 true， 则执行 语句块内容

如果结果为false， 则不执行语句块的内容

2、注意

1、if语句，条件位置处，必须为 boolean的值/表达式/变量

如果 条件 不是boolean 类型的话，JS会自动进行转换

以下情况，if 都会认为是 false

if(0){}

if(0.0){}

if(""){}

if(null){}

if(undefined){}

if(NaN){}

除以上情况外，一律为真

if(1){

console.log("真!");

}

if("我帅吗"){

console.log("真!!!!");

}

2、if语句块的{} ， 可以被省略的

如果省略{},那么if只控制它下面的第一条语句

3、if-else 结构

语法：

if(条件){

语句块1;

}else{

语句块2;

}

如果 条件 为真的话，则执行语句块1中的操作

否则 则执行语句块2中的操作

if(收款金额>=应收金额){

正常计算找零并输出

}else{

给出错误提示

}

4、else if 结构

复杂if结构 或 多重if结构

1、问题

分数判断：

如果 成绩大于 90分 则输出A

如果 成绩大于 80分 则输出B

如果 成绩大于 60分 则输出C

否则 输出 D

2、语法

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}else if(条件3){

语句块3;

}...else{

语句块n;

}

注意：最后的 else 模块，可以选择性添加

5、使用 else if 结构 实现 根据克莱托指数输出胖瘦正常

克莱托指数=体重(kg)/(身高(m)\*身高(m))

指数：

20 以下 则偏瘦

20~25 正常

25 以上 则偏胖

张东老师：

体重：100kg

身高：1.86

指数：28.90...

3、分支结构-2

1、switch-case

1、场合

优先用于 等值判断 的条件中

2、

输入 今天是 星期?

星期一：土豆炖牛肉

星期二：牛肉炖土豆

星期三：牛肉炖柿子

星期四：柿子炖牛肉

星期五：牛肉炖萝卜

星期六：休息

星期日：休息

3、语法

switch(变量/表达式){

case 值1:

语句1;

语句2;

...

break; //可选的，跳出switch-case结构

case 值2:

语句1;

语句2;

break; //可选

... ...

default://以上任何一个case都匹配不上时执行

语句1;

语句2;

break; //可选

}

流程：

1、计算 变量 或 表达式的值

2、将计算出来的值与case 后的数值 做等值判断，如果判断相等的话，则执行对应的case 后的执行语句。

3、如果 碰到 break，则结束执行，否则，继续向下执行。

switch-case 的结束机制：

1、碰到 break 结束

2、整个结构都执行完毕 结束

直落:

两个case 或 多个 case之间，没有任何的可执行代码，那么就以最后一块的case为主

Ex:

<script>

function showFood(){

var date = Number(prompt("今天星期?"));

//通过 switch-case 结构 等值判断

switch(date){

case 1:

console.log("牛肉炖土豆!");

break;

case 2:

console.log("土豆炖牛肉!");

case 3:

console.log("牛肉炖柿子!");

case 4:

console.log("柿子炖牛肉!");

break;

case 5:

console.log("牛肉炖萝卜!");

break;

case 6:

case 7:

console.log("今日休息!");

break;

default:

console.log("您的输入有误!!!");

break;

}

}

</script>

<button onclick="showFood();">显示菜谱</button>

4、循环结构

1、什么是循环结构

问题：

1、控制台打印输出一句Hello World

console.log("Hello World");

2、控制台打印输出十句Hello World

...

循环条件：

从第1句开始

到第10句结束

循环操作：

打印 Hello World

3、控制台打印输出100句Hello World

循环条件：

从第1句开始

到第100句结束

循环操作：

打印 Hello World

4、将100句Hello World 替换成 你好世界!

循环条件：

从第1句开始

到第100句结束

循环操作：

打印 你好世界

5、增加 第几遍 你好世界! 的输出

第1遍 你好世界

第2遍 你好世界

什么是循环：

反复一遍又一遍的做着相同(相似)的事情

生活中的循环：

1、上课

开始：8月31日

结束：12月xx日

2、5000米长跑

循环中的两大要素：

1、循环条件

什么时候开始，什么时候结束

2、循环操作

循环体，循环过程中 干了什么

2、循环结构-while循环

1、语法

while(条件){

循环操作

}

流程：

1、判断 条件

2、如果条件为真，则执行循环操作

3、再判断条件，如果为真，继续执行循环操作

...

n、判断循环条件，如果条件为假，则退出循环结构

注意：如果 循环条件一直为真的话，那么循环就是个死循环，尽量避免死循环的产生。一定要记得更新循环变量(条件)

2、练习

1、打印输出 1~100之间的 每一个数字

2、累加 1~100之间 每个数字的和，最后打印输出

3、循环的流程控制

1、continue 关键字

作用：用在循环中，用于终止本次循环，继续执行下次循环

2、break 关键字

作用：用在循环中的话，终止整个循环结构!!!

Ex1:

function print10Hello(){

//1、循环条件：从第1遍开始，到第10遍结束

//2、循环操作：打印输出 Hello World

var i = 1;//从第1遍开始

while(i<=10){

//循环操作

console.log("第"+i+"遍:你好 世界!");

i++;//更新循环条件

}

}

Ex2:

要求用户 在网页上循环录入数据，并打印输出，如果 录入的 exit ，结束操作

循环条件：录入 exit 为止

循环操作：

1、让用户录入数据

2、如果不是 exit 打印输出

3、如果是exit，则退出循环

//要求用户循环录入数据，直到exit为止

function printMsg(){

while(true){

//1、录入数据

var msg=prompt("输入一段数据：");

//2、判断 msg 是否为 exit (条件)

if(msg == "exit"){

alert("拜拜!!!");

break;

}

//打印输出 msg

console.log(msg);

}

}

<button onclick="printMsg();">循环打印数据</button>

Ex3:

/\*1、编写一个函数，用来计算购物车中的产品数量，价格，以及总价和找零\*/

function calCart(){

//1、定义输入变量:单价(price),数量(count),收款金额(money)

var price,count,money;

//2、定义输出变量:产品总价(total),找零(change)

var total,change;

//3、为输入变量赋值(prompt方式)

//3.1 为 单价(price) 赋值

price=prompt("请输入单价");

//3.2 为 数量(count) 赋值

count=prompt("请输入数量");

//4、根据 单价 和 数量 ，计算 产品总价(total) total = price \* count ,输出产品总价

total=price\*count;

//增加条件判断 大于500打八折

if(total>=500){

alert("打八折")

total\*=0.8;

}

alert("您此次消费"+total+"元");

//5、为收款金额(money)赋值 (prompt)

money=prompt("请输入收款金额");

if(money>=total){

//6、计算找零(change)=收款金额(money)-产品总价(total),输出找零

change=money-total;

alert("共找您"+change);

}else{

alert("对不起，您给的金额不够");

}

}

calCart();

<button onclick=""calCart;>购物车结算</button>

Day04

1、函数

1、函数的作用域

函数与变量类似，函数的作用域也分为：

1、全局作用域(全局函数)

2、函数作用域(局部函数)

2、全局函数

ECMAScript预定义的全局函数，可供用户直接使用

1、parseInt()

parseInt("35.5");

2、parseFloat()

3、Number();

4、isFinite()

作用：判断指定的数据是否有有限制

isFinite(5);判断 数字类型 的数据 是否有有限值，结果为 true

isFinite("a");判断 字符串类型 的数据 是否有有限值，结果为 false

5、encodeURI()

对 URI 格式的字符串进行编码，返回编码后的字符串

URI : Uniform Resource Identifier

统一资源标识符，负责定义 URI格式

URL : Uniform Resource Locator

统一资源定位器，符合URI规范的一个具体的地址。

6、decodeURI()

对 已编码的字符串进行解码

7、eval()

作用：用于执行以string形式出现的JS代码

语法：eval(js代码);

eval()函数也可以用于将JSON字符串解析为JS对象。

Ex:

function testFun(){

//1、调用 isFinite()

var result = isFinite("hello");

console.log(result);

//2、调用 encodeURI 对地址进行编码

console.log("http://www.baidu.com/s?wd=瘦身");

var newURI = encodeURI("http://www.baidu.com/s?wd=瘦身");

console.log(newURI);

console.log("对newURI进行解码:");

var result = decodeURI(newURI);

console.log(result);

}

function exeJS(){

//让用户从 弹框中输入一段JS代码，然后再网页中执行

var js = prompt("请输入一段JS:");

console.log(js);

//动态执行 js

eval(js);

}

</script>

<button onclick="exeJS()">执行用户JS</button>

<button onclick="testFun()">testFun</button>

<a href="http://www.baidu.com/s?wd=瘦身">瘦身</a>

Day05

1、do-while

1、问题：从网页录入数据，直到输入exit为止，如果不是exit就打印输出

while(true){

var input = prompt("输入：");

if(input == "exit"){

break;

}

console.log(input);

}

2、语法

do{

循环操作

}while(条件);

流程：

1、先执行循环操作

2、再判断循环条件

如果条件为真，则继续执行循环操作

如果条件为假，则终止循环即退出循环

特点：

无论条件满足与否，至少要执行一次循环操作

3、while 与 do...while 区别

1、while

先判断，后执行

最少一次循环都不执行(第一次条件就不满足)

2、do...while

先执行，后判断

最少要执行一次循环操作(第一次条件就不满足)

4、练习

将 猜数字的游戏 由 while 结构 更改为 do...while 结构

2、for循环

1、问题

打印 1 - 100 之间所有的数字

var i=1;//1、声明循环条件

while(i<=100){//2、循环条件判断

console.log(i);//3、循环体-打印输出

i++;//4、更新循环条件

}

2、for循环

1、特点:整合while循环

2、语法

for(声明循环条件; 循环条件判断; 更新循环条件){

循环操作

}

表达式1:声明循环条件

表达式2:循环条件判断

表达式3:更新循环条件

3、执行过程

1、计算表达式1的值

2、计算表达式2的值，如果为true，则执行循环体，否则退出循环

3、执行循环体

4、计算表达式3的值

5、计算表达式2的值，... ...

一直循环 2~5的步骤，直到条件为false为止

4、练习

打印1-100之间所有的偶数

function printEven(){

for(var i=2;i<=100;i+=2){

console.log(i);

}

}

5、for，while，do-while 之间的区别

1、for，while

最少执行 0 次

while:多数用于不确定循环次数时使用

for:多数用于确定循环次数时使用

2、do-while

最少执行 1 次

适合用于不确定循环次数时使用

3、for的三个表达式的特殊用法

1、表达式1可以为空

要在循环之外补齐第一个表达式内容

2、表达式2可以为空

死循环

在循环体内 补足 表达式2

3、表达式3可以为空

死循环

在循环体内 补足 表达式3

4、表达式1和表达式3的多样化

表达式1 和 表达式3 允许 编写多个表达式，中间用,隔开即可

4、循环的嵌套

在一个循环体内再出现一个循环

1、控制台打印 \*

console.log("\*");

ex:

function printStar(){

for(var j=1;j<=5;j++){

var star="";

for(var i=1;i<=j;i++){

star+="\*";

}

console.log(star);

}

}

Ex:

//九九乘法表

function printMuti(){

//外层循环

for(var j=1;j<=9;j++){

var msg="";

//内层循环

for(var i=1;i<=j;i++){

msg+=(i+"\*"+j+"="+(i\*j))+"\t";

}

//打印输出

console.log(msg);

}

}

<button onclick="printMuti()">jiujiu</button>

Ex:

<script>

function guess(){

//1、生成一个随机数(0-99)

//var r = parseInt(Math.random()\*100);

/\*1、循环外\*/

//1、生成随机数

var r = parseInt(Math.random()\*100);

/\*2、循环内\*/

//2.1 循环条件：exit while(true){ break; }

//2.2 循环操作

//1、输入数字

//2、判断数据是否为 exit，是否推出

//3、比较大小，给出提示

//4、如果猜对了，提示正确并退出循环

do{

var input=prompt("请输入一个数字，输入exit退出:");

//判断是否为 退出

if(input == "exit"){

alert("");

}else if(input>r){

alert("您输入大了，请重新输入");

}else if(input<r){

alert("您输入小了，请重新输入");

}else{

alert("恭喜您，猜对了!");

break;

}

}while(input!="exit")

alert("程序结束!!!");

}

//猴子吃桃

function peach(){

var total,left=1;

var day = 9;

while(day>=1){

total = (left + 1) \* 2;

left = total;

day--;

}

console.log("第一天，猴子共摘了"+total+"个桃子!");

}

</script>

<button onclick="guess()">猜数字</button>

<button onclick="peach()">猴子吃桃</button>

5、练习

1、打印输出所有的水仙花数

水仙花数：三位数 100~999

水仙花数条件：各个位上数字的立方和等于这个数本身

153=1\*1\*1+5\*5\*5+3\*3\*3

=1+125+27

=153

153 % 10 = 3

parseInt(153 / 10) % 10 = 5

parseInt(153/100) = 1

2、打印 100以内所有的素数(嵌套循环)

问题：随意输入一个数字，判断该数是否为素数

//素数：从2开始，只能被1 和 它本身整除的数字

function isPrime(){

var num = Number(prompt("请输入一个数字："));

var isP = true;//默认 是素数

//条件：从2开始，到num-1为止 -->i

//操作：num%i==0 不是素数

for(var i=2;i<num/2;i++){

if(num%i == 0){

isP = false; //说明不是素数

break;

}

}

if(isP){

console.log("是素数!");

}else{

console.log("不是素数!");

}

}

Ex:直落

function printMonth(){

//用户输入一个月份，如果是1.3.5.7.8.10.12的话，输出有31天，如果是4.6.9.11的话，输出有30天，如果是2的话，输出，闰年29天，平年28天

var month = Number(prompt("Month:"));

switch(month){

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

console.log("有31天");

break;

case 4:

case 6:

case 9:

case 11:

console.log("有30天");

break;

case 2:

console.log("闰年29天，平年28天");

break;

default:

console.log("输入有误!");

}

}

</script>

<button onclick="printMonth();">Month</button>

兔子问题：

有1对兔子，从出生的第3个月起都生1对兔子，小兔子长到第3个月后每个月又生1对兔子，假如兔子都不死，问 某个月 的兔子有多少???

假设 f 函数 用来计算某个月的兔子总量

f(n) : n 哪个月的兔子总量

f(5) : 第5个月有多少对兔子

f(4) : 第4个月有多少对兔子

f(3) : 第3个月有多少对兔子

f(2) : 1对兔子

f(1) : 1对兔子

function f1(n){

if(n<3){

return 1;

}

var f1=1,f2=1,fn;

for(var i=1;i<=n-2;i++){

fn=f1+f2;

f1=f2;

f2=fn;

}

return fn;

}

菲波那切数列

1,1,2,3,5,8,13,21,34,55,89,... ...

问题：打印 菲波那切数列数列中的前20个数字

function printFib(){

var n1=1;

var n2=1;

console.log(n1,n2);

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

n1=n1+n2;

n2=n2+n1;

console.log(n1,n2);

}

}

递归：

在函数内，又调用了自己

递归调用的内层函数，是在外层函数还未结束时就已经开始了，那么外层函数就会被阻塞。

缺点：算法复杂度太高，且浪费内存

解决：大部分递归，可以被循环所代替