### 一．Es6-basic

Day1 ················································1

Day2 ················································1

Day3 ················································1

Day4 ················································1

Day5 ················································1

Day6 ················································1

Day1

1、表单控件

1、input

属性：type,name,value,disabled

1、文本框 与 密码框

type="text"

type="password"

属性：maxlength，readonly

2、单选按钮 和 复选框

type="radio"

type="checkbox"

属性：

1、name ：一组的radio或checkbox的name属性必须一致

2、checked

3、按钮

type="submit"

type="reset"

type="button"

<button type=""></button>

4、隐藏域 和 文件选择框

type="hidden"

type="file"

1、method 必须为 post

2、enctype 必须为 multipart/form-data

2、textarea

<textarea></textarea>

name，rows，cols，readonly

3、select 和 option

<select>

<option></option>

<option></option>

</select>

select:

name,size,multiple

option:

value,selected

4、其它元素

1、label

for ：要关联的表单控件的ID

2、fieldset 和 legend

<fieldset>

<legend></legend>

... ...

</fieldset>

3、浮动框架

<iframe></iframe>

src,width,height,frameborder

5、新input类型

1、type="email"

2、type="url"

3、type="search"

4、type="color"

5、type="number"

min,max,step

6、type="tel"

7、type="range"

min,max,step,value

8、type="date"

9、type="month"

10、type="week"

2、JavaScript

1、基础语法

1、浏览器内核

内核：由内容排版引擎 和 脚本解释引擎 来组成

2、搭建以及运行JS程序

1、使用独立安装的JS解释器 -- NodeJS

console.log("");

console.log('');

2、使用浏览器内核中嵌入的JS解释器

1、直接在Console中输入JS脚本并执行

2、将JS脚本嵌入在HTML中执行

1、html元素的事件

<button onclick="JS脚本"></button>

=====================================

1、基础语法

1、浏览器内核

2、搭建以及运行JS程序

1、使用独立安装的JS解释器 -- NodeJS

2、使用浏览器内核中嵌入的JS解释器

1、直接在Console中输入JS脚本并执行

2、将JS脚本嵌入在HTML中执行

1、html元素事件

2、<script>元素中嵌入JS脚本

<script>

JS脚本代码1;

JS脚本代码2;

... ...

</script>

console.log("打印在控制台上的信息");

document.write("显示在网页上的一句话");

window.alert("以弹框的方式显示的一句话");

3、在外部脚本文件中嵌入 JS 代码

1、创建 js 脚本文件 (\*\*\*.js)

2、在脚本文件中编写JS脚本代码

3、在页面中对脚本文件进行引用

<script src="\*\*\*.js"></script>

注意：

在引入js文件的<script>中不允许再编写JS脚本代码

练习：

1、创建01-exercise-1.html，创建一个script标记并编写脚本，在页面上打印输出自己的姓名

2、再添加一个按钮，为按钮添加单击事件，在控制台上打印输出自己的姓名

3、使用外部脚本文件，编写JS脚本，能够在控制台 以及 页面中分别打印自己的姓名

3、JS调试

1、JS的代码错误

JS是解释型脚本语言，碰到错误，解释器则终止执行。但不会影响后续"块"的代码执行

<script></script> 第一块

<script></script> 第二块

练习：

1、创建 02-exercise.html，编写<script>元素，使用行语句在页面/控制台打印输出 "Hello1","Hello2","Hello3" ，故意将第二行语句编写错误，查看结果

4、语法规范

1、语句 - 能被JS引擎所解释的一句代码

1、由表达式，关键字，运算符组成

2、严格区分大小写

name,Name 是两个东西

console.log()

Console.log()

3、每条语句都是由 ; 作为结尾

4、语句中所有的符号都是 英文

英文 中文

" ”

' ’

. 。

; ；

: ：

5、通常情况下，每条语句独占一行

2、注释

单行注释： //

多行注释： /\* \*/

2、变量 与 常量

1、什么是变量

变量(Variable)

一个允许被反复使用并且可以变化的数据

2、变量的声明

1、声明变量

var 变量名;

2、为变量赋值

变量名=值;

注意：

1、变量最好被赋值后再使用，否则的话，值为 undefined

2、声明变量时最好加上var关键字

3、变量声明的简便写法

var 变量名=值;

ex:

1、声明变量保存用户姓名

var uname="张无忌";

var sname='张三丰';

2、声明变量保存用户年龄

var uage=18;

4、一条语句中声明多个变量

var 变量名1,变量名2,变量名3;

ex

var uname,uage,ugender,uhobby;

uname="张三丰";

uage=18;

ugender="男";

uhobby="太极";

var 变量名1=值,变量名2=值;

ex

var uname="张三丰",uage=28,ugender;

5、变量的使用

console.log(变量名);

document.write(变量名);

window.alert(变量名);

以上代码是将变量值放在不同的作用中

注意：千万不能加 "" 或 ''

var uname="张三丰";

console.log(uname); // 张三丰

console.log("uname"); //uname

字符串的拼接：使用 "+"

6、变量名的命名规范

1、可以包含字母、数字、下划线以及 $

2、不能以数字开头

3、尽量见名知意

4、尽量采用驼峰命名法

var userName="";

7、变量的复杂使用

1、未经初始化的变量

1、定义变量但未赋值

var userName;

console.log(userName);

结果为 ：undefined

2、直接使用未被声明过的变量

console.log(userAge);

结果为：语法错误

userAge = 25;

console.log(userAge);

结果为：25

2、对变量进行赋值的操作 - SET操作

只要变量出现在 = 的左边，一律都是赋值操作

var age = 18;

var newAge = age; //将 age 的值赋值给 newAge变量，针对newAge是赋值操作，针对age是取值操作

3、对变量进行取值的操作 - GET操作

只要变量没出现在=的左边，一律都是取值操作

var age = 18;

console.log(age);

document.write(age);

var newAge = age;//newAge是赋值操作，age是取值操作

age = age +1;

=右边的age，是取值操作(GET)

=左边的age，是赋值操作(SET)

3、常量

1、什么是常量

一经声明，就不允许被修改的数据叫常量

2、声明常量

const 常量名=值;

注意：

1、常量名称一般采用全大写形式

2、常量声明时必须赋值

练习：

1、声明一个变量 r ，用来表示一个圆的半径并赋值

2、声明一个常量 PI ，用来保存圆周率3.14

3、计算圆的周长并保存在变量 l 中，并输出

周长 = 2 \* π \* r

4、计算圆的面积并保存在变量 s 中，并输出

面积 = π \* r \* r

3、数据类型

1、什么是数据类型

数据在内存中的类型是什么样，解决了数据在内存中占据空间大小的问题

JS的数据类型分为：

1、基本数据类型 的值

又称为 "原始数据类型"

1、number 数字类型

变量中保存的值是数字形式的，就是数字类型，并且在使用过程中，不能用 "" 或 ''括起来的数字，是数字类型

var age = 32; //数字类型

var myAge = "32";//字符串

可以存储32位的整数以及64位的浮点数

8bit = 1byte (8位 = 1字节)

1024byte = 1KB

1024KB = 1M

1024M = 1G

1024G = 1T

整数占4个字节，浮点数(小数)占8个字节

有浮点数参与到运算的话，就容易造成精度丢失的问题

var result=数字/变量.toFixed(保留的位数);

表示整数：

1、十进制整数

由0-9十个数字组成

逢十进一

2、八进制整数

由0-7八个数字组成

逢八进一

在JS中，八进制数字是以0作为开始的

var n1 = 010;

console.log(n1);//统一按10进制打印

3、十六进制

由0-9，A-F 来组成

逢十六进一

在JS中，十六进制数字是以0x作为开始的

var n1=0xA;

表示浮点数：

1、由小数点计数法表示

var n1 = 36.85;

2、由指数方式表示

var n2 = 36.85E23;

2、string 字符串类型

1、什么是字符串

表示的是一段信息，在使用时用"" 或 ''标示出来

由Unicode字符，数字，标点来组成。

JS中的每个字符也都有一个Unicode码来进行表示

ex:

"张"的二进制表示法

"张".charCodeAt() : 10进制表示

"张".charCodeAt().toString(2) : 2进制表示

"张".charCodeAt().toString(16) : 16进制表示

将 Unicode码转换为 JS字符

var str = "\u5f20";

ex:

中文汉字的范围

从 4e00 开始 到 9fa5 结束

在JS中的特殊字符需要通过转义字符来表示

如 "

在控制台中打印输出 Hello "World"

\" : 表现为 "

\' : 表现为 '

\\ : 表现为 \

\n : 表现为 换行

\t : 表现为 缩进距离

exer:

使用一个输出 (Console.log) ,完成以下内容输出

=====欢迎来到王者荣耀=====

1、匹配赛

2、排位赛

3、人机练习

4、训练营

3、boolean 布尔类型

1、作用

通常用于表示条件的结果

2、取值

true 或 false

在做+-\*/%运算的时候，true可以当做1运算，false可以当做0运算

ex:

通过boolean变量来表示

1、现在是否下雨

var isRain=false;

2、现在饿不饿

var isHungry=true;

4、null 空类型

表示的是对象为 空 (释放对象的空间)

5、undefined 类型

语义："数据不存在"

在声明变量但未赋值的时候，或访问的对象的属性不存在时也是 undefined

2、引用数据类型 的对象

2、数据类型的转换

1、如何查看变量或数据的数据类型

在 JS 中允许使用 typeof() 查看变量或数据的数据类型

语法：

var result = typeof(变量);

2、隐式转换

不同类型的数据之间会进行自动转换来保证类型统一

数字 + 字符串 ：将数字转换成字符串

var num = 123;

var str = "abc";

var result = num + str; //123abc

数字 + 布尔值 ：将布尔值转换成数字

true 转换为 1

false准换为 0

字符串 + 布尔值 ：将布尔值转换为字符串

true 转换为字符串 "true"

false转换为字符串 "false"

布尔值 + 布尔值 ：转换为数字类型

true 转换为 1

false准换为 0

true + false 结果为 1

问题：

var num1 = "123";

var num2 = 456;

var result = num1 + num2;

结果：

result : 123456

类型 : string

3、显示转换 - 由转换函数完成的转换操作

1、toString()

将任意类型的数据转换为string

语法：

var result=变量.toString();

2、parseInt()

将任意类型的数据转换为 整数

语法：

var result=parseInt(待转换的数据);

result 是最终转换完成的整数结果

注意：

1、碰见第一个非数字字符，则停止转换

var result=parseInt("35abc");

结果：35

var result=parseInt("35.34");

结果：35

2、如果第一个字符就是非数字字符，结果为NaN(Not a Number)

var result = parseInt("abc");

结果：NaN

var result = parseInt("A35");

结果：NaN

3、parseFloat()

将任意类型的数据转换为 浮点类型

var result = parseFloat("35.563");

结果：35.563

var result = parseFloat("A35.563");

结果：NaN

var result = parseFloat(".563");

结果： ？？？

4、Number()

将一个任意类型数据转换为number类型

注意：

如果转换的数据中包含非法字符，则为Na

语法：

var result = Number(待转换的数据);

ex:

var result = Number("35.3");

结果：35.3

var result = Number("35.A");

结果：NaN

作业：

1、弹出一个输入提示框，提示用户输入金额

var ret = window.prompt("请输入金额:");

最终 ret中保存的就是用户输入的数据

2、将用户输入的金额 + 10 ， 再打印输出

3、将上一句代码注销，使用typeof函数检查输入金额的数据类型,再分析步骤2的结果的原因

4、尝试着将用户输入的数据转换为一个数字，再重复步骤2的操作

var uAge = 18;

var newAge = 1800;

var uName = "张三丰";

var newName="我是张三丰";

Day2

1、浏览器内核中执行JS

1、元素事件

2、内嵌的 <script>

3、外部的 脚本文件 \*\*\*.js

2、语法规范

1、语句

2、注释

3、变量 和 常量

1、变量

语法：var 变量名=值;

使用：

1、SET

变量出现在 = 的左边

2、GET

变量出没出现在 = 的左边

2、常量

语法：const 常量名=值;

4、数据类型

1、基本数据类型

1、number

表示32位整数

8进制 : 010,07

10进制 : 123,12,156

16进制 : 0x1A

表示64位浮点数

3.5e8

2、string

使用的时候用 "" 或 '' 引起来

张：5f20

将Unicode码转换为字符

var str = "\u5f20";

转义字符:

\\ : \

\" : "

\' : '

\n : 换行

\t : 缩进

var str = "Hello" + \n + "World"; //错误

var str = "Hello \n World";

3、boolean 类型

true 和 false

4、null

空

5、undefined

访问的数据不存在

2、引用数据类型

Object : 对象

Number : 数字

String : 字符串

Boolean : 布尔

Date : 日期

5、数据类型转换

1、隐式转换

1、字符串 + 数字 ：字符串

2、数字 + 布尔 ：数字

3、字符串 + 布尔 ：字符串

"Hello" + true : "Hellotrue"

4、布尔 + 布尔 ：数字

2、显示转换 - 转换函数

1、toString()

转换为字符串

toString(2) : 转换为2进制

toString(8) :

toString(16) :

2、parseInt()

转换为整数

parseInt("35") : 35

parseInt("35.3") : 35

parseInt("35.a") : 35

parseInt("a35") : NaN

3、parseFloat()

转换为浮点型数据

4、Number()

如果转换的内容中包含非法字符，结果为 NaN

Number("35.3") : 35.3

Number("35abc") : NaN

=============================================

注意：

只要是从页面中得到的数据，全部都是string类型

1、运算符和表达式

1、表达式

a = 5; //a的值为5，表达式的值也为5

2、运算符

1、算术运算符

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

ex:

1、 5 / 2 结果 2.5

2、 5 % 2 结果 1

3、

var a = 5;

++a; //6

console.log(a); //6

console.log(a++);//6

console.log(a);//7

注意

1、自增 和 自减，如果单独使用的话，无论在前缀 还是 在后缀，结果都是相同的

var a = 5;

a ++;

console.log(a); 结果 ：6

============================

var a = 5;

++ a;

console.log(a); 结果 ：6

2、配合其他语句一起使用，++在后，先使用变量的值再自增

var a = 5;

console.log(a++); 先打印 5 ，再变成6

3、配合其他语句一起使用，++在前，先自增再使用变量的值

var a = 5;

console.log(++a); 先变成6 ，再打印 6

var a=5;

5 [6]6 6[7] 7[8] [9]9 9

var result = a + ++a + a++ + a++ + ++a + a;

结果：

1、42

2、40

3、45

4、43

5、41

练习：

从弹框中随意输入4位整数，通过运算符，将每位的数字单独保存在一个变量中，并打印输出

输入：1234

输出：

千位：1

百位：2

十位：3

个位：4

注意：

1、-，\*，/，%，两边的操作数如果不是数字类型，解释器会隐式的转换为数字类型，转换成功则正常运算，否则结果为 NaN

ex:

"5" \* "3" : 结果为 15

2、关系运算符

1、什么是关系运算符

判断数据之间大小 和 相等关系的，一般是组成条件判断的 重要运算符之一

2、运算符

>,<,>=,<=,==,!=,===,!==

Q : "张三丰">"张无忌" ? false

Q : "10" > "5" ? false

Q : "10" > 5 ? true

Q : 10 > 5 ? true

注意：

1、如果运算符两边都是数字类型，则比较数字大小

2、如果运算符两边有一个数数字的话，则试图将另外一个数据隐式转换为数字，再进行比较数字的大小

"10" > 5 ?

试图将 "10" 转换为 数字 10

再与 5 进行大小的比较

3、如果元素安抚两边都是字符串的话，则比较每位字符的Unicode码，Unicode码大的为大

"张三丰" > "张无忌"

张三丰

张无忌

"10" > "5"

10

5

5的Unicode码 大于 1的Unicode 码

最后的结果是 "5" 大于 "10"

Q : 如何判断一个数据是否为数字???

var num = 123;

var str = "123";

var str1 = "123a";

通过 isNaN() 判断一个数据是否为 非数字

var result=isNaN(数据);

如果 数据 是数字的话，result 为 false

如果 数据 不是数字的话result 为 true

3、逻辑运算符

1、运算符

!,&&,||

2、作用

关联多个条件的

true && false : false

false && false : false

true || false : true

!true : false

3、短路逻辑

1、短路与

条件 && 操作

2、短路或

操作1 || 操作2

var input = prompt("输入一段数据:");

var result = input || "这个家伙很懒...";

注意：

null,"",false,undefined,0,0.0

以上内容，在条件中，全部当假处理

3、练习

1、输入一个年份，判断是否为闰年，如果是闰年打印true，不是打印false

年份 能被 4 整除 但不能被 100整除

或

年份 能被400整除

2、在弹框中输入一个字符

在控制台中输出

是一个英文吗 ? ...

是一个数字吗 ? ...

是一个汉字吗 ? ...

输入字符：a

是一个英文吗 ? true

是一个数字吗 ? false

是一个汉字吗 ? false

输入字符：你

是一个英文吗 ? false

是一个数字吗 ? false

是一个汉字吗 ? true

4、位运算

1、什么是位运算符

将数字转换为二进制后，再进行运算

位运算会以整数的方式进行运算

2、位运算符

1、按位与 &

语法：a & b

语义：

1、将两个数字都转换为2进制数，然后逐位比较

ex: 5 & 3

5: 101

3: 011

===========

2、每位上对应的两个数字都为1的时候，那么该位的结果为1，否则，该位结果为0

ex: 5 & 3

5: 101

3: 011

===========

001

最终结果为：1

查看：

1 & 1 ? 1

2 & 1 ? 0

3 & 1 ? 1

4 & 1 ? 0

5 & 1 ? 1

2、按位或 |

语法：a | b

语义：

1、将两个数字都转换为2进制数，然后逐位比较

2、每位上对应的两个数字，至少有一个为1的话，那么该位的结果为1，否为，该位结果为0

ex

5 | 3

5 : 101

3 : 011

========

111

结果为 ：7

使用场合 ：将小数取整时使用

num | 0

3、按位异或 ^

语法：a ^ b

语义：

1、将两个数字都转换为2进制数，然后逐位比较

2、每位上对应的两个数字，相同时该位的结果为0，不同时，该位的结果为1

ex 5 ^ 3

5 : 101

3 : 011

=========

110

练习：

声明两个变量(a 和 b)分别保存两个数字

a = a ^ b;

b = b ^ a;

a = a ^ b;

结果 : 交换位置

5、赋值运算符

1、赋值运算符 =

将右边的表达式计算出结果赋值给左边的变量

var a = b = c = 50;

2、扩展的赋值运算符

+=,-=,\*=,/=,%=,&=,|=,^=

a += b; ==> a = a + b;

使用 扩展的赋值运算符 整理 异或 运算

6、条件(三目)运算符

单目运算符：只有一个操作数的运算符

++，--，!

双目运算符：有两个操作数的运算符

+,-,\*,/,%

&&,||

>,<,>=,<=,==,!=,===,!==

&,|,^,<<,>>

三目运算符：有三个操作数的运算符

? :

条件 ? 结果1 : 结果2;

练习：

根据身高体重计算克来托指数

指数=体重/(身高\*身高);

20-25 : 正常

20以下:偏瘦

25以上:偏胖

7、使用 "+" 进行字符串的拼接

+ 用于两个数字/布尔,表示算术运算

+ 用于字符串，表示的是字符串的拼接

8、运算符优先级

var num = 7;

var result = num & 1 == 1;

console.log(result); // 结果为 ：1

result = (num & 1) == 1;

以下优先级，由高到低

1、(),[],.

2、++，--，!

3、\*，/，%

4、+，-

5、<<,>>

6、<,<=,>,>=

7、==,!=,===,!==

8、&

9、^

10、|

11、&&

12、||

13、? :

14、=，+=，-=，\*=，.......

2、函数

1、什么是函数

函数(function),也称为方法

是一段预定义好，并可以被反复使用的代码块

1、预定义好

提前声明，但不会马上被执行的内容

2、反复使用

可以被多个元素，多次调用

3、代码块

函数中允许包含多条可执行的语句

函数的本质上也是一个功能完整的对象

2、函数的定义 和 调用

1、普通函数的声明 和 调用

function 函数名(){

若干条可执行语句;

}

ex:

function sayHello(){

console.log("欢迎来到王者荣耀");

console.log("敌人还有5秒钟到达战场");

}

函数的调用

在任何JS的合法位置处，都允许通过

函数名();

来实现函数的调用

练习：

创建一个函数，函数内部要实现以下功能

1、通过三个弹框输入三个数字

2、将三个数字进行升序排列

3、打印输出三个变量的值

a = 15

b = 3

c = 26

输出结果：

a=3

b=15

c=26

2、带参数的函数声明 和调用

1、什么是参数

函数在运行中，需要外部传递进来的辅助运算的数据

参数，可以让函数变得更加灵活

2、语法

function 函数名(参数列表){

}

ex:

function zhaZhiJi(fruit){

console.log("准备榨汁");

console.log("正在榨"+fruit+"汁");

}

调用：必须传参

zhaZhiJi("西瓜");

3、带返回值函数的声明

1、什么是返回值

在函数体内计算出一个数据带给函数的外部

2、语法

function 函数名(){

语句1;

语句2;

return 数值;

}

调用

var result = 函数名();

//result中保存的就是 函数的返回值

用过的带返回值的函数：

var result = parseInt(35.3);

练习：

声明一个函数，定义两个参数，在函数体内，将两个参数的值进行相加，再返回给函数调用出

再声明一个函数，调用上面的函数，将返回值打印在控制台上

3、变量/函数的作用域

1、什么是作用域

作用域指的是变量或函数的可访问范围

在JS中，作用域分为两种：

1、函数作用域

只能在当前函数被访问的内容

2、全局作用域

一经定义，在代码的任何地方都能访问

2、函数作用域中的变量

又称为 "局部变量",只能在定义的函数中被访问到，离开定义的函数就无法使用

注意：所有的参数，都是局部变量

3、全局作用域中的变量

又称为 "全局变量",一经定义好在代码的任何位置处都能访问

如何声明全局变量：

1、直接在<script>标记中声明的变量

2、声明变量不使用 var

注意：

该方式尽量少用，容易出问题

4、声明提前

Javascript程序在正式执行前，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端，但对变量的赋值还保留到原位置

注意：

所有的变量声明，尽量放在所在作用域的最顶端

作业1：

1、定义一个可以接收两个参数的函数，函数体内，计算两个数的和，并返回

2、定义一个可以接收三个参数的函数，函数体内，返回三个参数中最大的数

问题1：

输入一个数字：123

123是奇数吗：true

输入一个数字：456

456是奇数吗：false

问题2：

输入一个价格：7888

涨价了吗 : true

输入一个价格：5888

涨价了吗 : false

a = 5 , b = 3

a = a ^ b;

5 : 101

3 : 011

=======

110 : 结果 6，即a的值为6

b = b ^ a;

3 : 011

6 : 110

========

101 : 结果 5，即b的值为5

a = a ^ b;

6 : 110

5 : 101

========

011 : 结果 3，即a的值为3

((year % 4 == 0) && (year % 100 !=0)) || (year % 400 == 0)

Day3

1、运算符和表达式

1、算术运算符

2、关系运算符

3、逻辑运算符

允许通过短路逻辑完成简易的条件判断和执行

4、位运算

按位与 ：每位数字都为1，整体结果为1

按位或 ：每位数字至少有一位为1，整体结果为1

按位异或 ：^,每位数字相同为0，不同为1

5、赋值运算符

&=,|=,^=

6、条件运算符

? :

2、函数

1、什么是函数

预定义好，可以被反复执行的代码块

2、函数的声明 和 调用

1、普通的函数

function 函数名(){ ... }

函数名();

2、带参数函数

function 函数名(参数列表){ ... }

函数名(参数列表);

3、带返回值函数

function 函数名(参数列表){

... ...

return 值;

}

var result = 函数名();

ex:

function test(){

console.log("...");

}

var result = test();

console.log(result);//undefined

3、作用域

1、作用域

1、函数作用域

2、全局作用域

1、声明再 <script> 中

2、声明变量不适用 var

2、声明提前

function test(){

console.log(num); // undefined

var num = 15;

}

推荐：任何的变量的声明最好都放在所在作用域最顶端

==================================================

1、函数

1、值传递

基本类型的数据在做参数的时候，都是按值传递的方式进行传递的

引用类型的数据在做参数的时候，都是按引用的方式进行传递的

1、什么是值传递

基本数据类型在做参数传递时，传递的并不是值本身，而是值的一个"副本"

注意：以后做基本数据传递参数时，尽量不要在函数体中修改值，因为改完后对实际数值是没有影响的

2、函数的作用域

也分为全局作用域的函数 - 全局函数，以及函数作用域的函数 - 局部函数

1、全局函数

在最外层<script>中定义的函数就是全局函数

全局函数一经定义，任何位置都能使用

2、局部函数

在一个函数体内定义的函数就是局部函数

局部函数只能在定义的函数体内被访问，出了范围后就无法使用

测试：

1、声明一个函数 fun1 -- 内容自定

2、在 fun1 的内部 再声明一个函数 inner

在inner中向控制台输出一句话 ：内部函数

3、尝试着在 最外层 调用 inner 函数 - 看结果

4、尝试着在 fun1 中调用 inner 函数 - 看结果

3、ECMAScript 提供的全局函数

ES提供的全局函数不用编写，在JS引擎中直接就自带的函数，直接调用即可

1、parseInt()

2、parseFloat()

3、isNaN()

4、isFinite()

判定指定数据是否有有限值

var result = 20 / 0;

结果 ：Infinity

5、encodeURI()

对统一资源标识符(URI)进行编码，并返回编码后的结果

6、decodeURI()

对已编码的统一资源标识符进行解码，返回解码后的形式

练习：

1、创建一个函数，testEncode

2、通过弹框接收一个用户要查询的关键词(允许是中文的)

3、拼到百度的请求地址后，返回拼好的并且已编码的地址

http://www.baidu.com/s?wd=

7、encodeURIComponent()

对统一资源标识符的有效组件进行编码，并返回编码后的结果

实际上 encodeURIComponent() 允许将 :// 也一同进行编码

8、decodeURIComponent()

... ...

9、eval()

执行 以字符串方式传递进来的js代码

var msg = "console.log('Hello World')";

eval(msg);

练习：

1、创建一个函数，testEval

2、要求用户通过弹框输入一句可执行的JS代码

3、执行用户输入的JS代码(使用eval)

4、递归 - 算法

1、问题

1、用一个函数实现5!

5! = 5 \* 4 \* 3 \* 2 \* 1 ;

5! = 5 \* 4!

4! = 4 \* 3!

3! = 3 \* 2!

2! = 2 \* 1

2、求指定数字(n)的阶乘

n! = n \* (n-1)!

(n-1)! = (n-1) \* (n-2)!

/\*\*

\* 计算 n的阶乘

\*/

function f(n){

计算 n 的阶乘的代码 ... ...

}

f(5) = 5 \* f(4)

f(4) = 4 \* f(3)

f(3) = 3 \* f(2)

f(2) = 2 \* f(1)

f(1) = 1

2、什么是递归

在一个函数中，直接或间接的再调用自己的过程就是递归

3、递归的实现

1、边界条件

决定的递归何时向上返回

2、递归前进

只要没满足递归的边界条件，就一直向下调用自己

3、递归返回

当满足边界条件时，向上返回(将所有计算出来的值，依次返回给调用者)

4、练习 ：计算菲波那切数列的第n个位置的数字

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

该数列中，第一个数和第二个数永远都是1

/\*\*

\* 获取第n个位置的数字

\*/

function f(n){

}

f(5) = f(4) + f(3)

f(4) = f(3) + f(2)

f(3) = f(2) + f(1)

f(2) = 1

f(1) = 1

2、分支结构

1、流程控制结构

1、顺序结构

2、分支结构(选择结构)

根据条件选择一段代码去执行

3、循环结构

根据条件重复的执行一段代码

2、顺序结构

练习：模拟收款程序

编写收银柜台收款程序，根据商品单价，购买数量，以及收款金额计算并输出应收金额和找零

1、定义输入数据

由外部传递的程序中的数据称为输入数据

商品单价(price)

购买数量(count)

收款金额(money)

2、定义输出数据

由程序计算后要显示外部的信息

应收金额(total)

找零(change)

3、解决步骤

1、为商品单价，购买数量 赋值(JS弹框)

2、计算应收金额=单价\*数量

3、为收款金额(money)赋值

4、找零=收款金额(money)-应收金额(total)

4、如果消费满500，则打八折

5、异常情况

如果收款金额大于等于应收金额的话，正常找零

如果收款金额小于应收金额的话，提示异常

3、分支结构

1、if

根据条件选择某块代码去执行

1、语法

if(条件){

满足条件要执行的语句

}

练习：

从弹框中输入一个数字，如果是偶数的话，则打印是偶数

注意：

1、if后的条件尽量是boolean类型，如果不是布尔类型的话，会自动转换

以下数值，会自动转换为false

if(0){}

if(0.0){}

if(""){}

if(null){}

if(undefined){}

if(NaN){}

ex:

if(false) : 假

if("false") : 真

if(35.58) : 真

if("李文华帅吗") : 真

2、if后的{}是可以省略的，如果省略的话，只控制if后的第一条语句

if(条件)

语句1;

语句2;

Q:

条件为真，执行 语句1，语句2

条件为假，执行 语句2

2、if ... else ... 结构

语法：

if(条件){

//满足条件时执行的内容

... ...

}else{

//不满足条件时执行的内容

... ...

}

1、输入一个年份，判断是否为闰年

如果是闰年：打印输出 \*\*\*\*年是闰年

否则：打印输出 \*\*\*\*年不是闰年

3、else ... if ... 结构 (多重if结构)

语法：

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}... ...else{

语句块n;

}

练习：

1、从弹框中录入考试成绩

2、如果成绩小于0分或大于100分，提示成绩错误

3、根据成绩判断，给出不同的奖励提示

如果成绩为100分 ，奖励一台兰博基尼

如果是90分以上(100以下)，奖励一台保时捷

如果成绩在80分以上，奖励一台奥迪

如果成绩在70分以上，奖励一台奥拓

如果成绩在60分以上，奖励一台永久

否则：奖励俩大嘴巴子

4、switch ... case

1、使用场合

等值判断

2、语法

switch(变量){

case 常量1:

语句块1;

break;

case 常量2:

语句块2;

break;

... ...

default:

语句块n;

break;

}

switch(变量){

case 1:

case 2:

case 3:

console.log("... ...");

break;

case 4:

case 5:

case 6:

... ...

break;

}

注意：switch后的变量与各个case后的常量是采用 "===" 判断是否相等，所以数值以及数据类型完全相等时，才能匹配成功

练习：

1、用户从弹框中输入星期 ?

2、根据用户输入的数据进行匹配

1 ：晚上吃红烧肉

2 ：晚上吃红烧鸡块

3 ：晚上吃红烧鱼

4 ：晚上吃红烧翅根

5 ：晚上吃红烧狮子头

6 ：今天休息

7 ：今天休息

否则：提示输入有误

练习2：

1、用户分别输入年，月

2、提示出 该月有多少天

小提示：

1、1.3.5.7.8.10.12 ：都有31天

2、4.6.9.11 ：都有30天

3、2

闰年：29天

平年：28天

ex:

输入年：2017

输入月：7

2017年7月有31天

作业：

1、分别定义4个函数模拟执行银行操作功能

1、模拟查询余额

余额查询中，请稍候...

2、模拟取款

取款进行中，请稍候...

3、模拟转账

转账进行中，请稍候...

0、模拟退出

拜拜... ...

2、定义一个电话银行函数，允许用户输入一个数字，根据输入的数字执行上述函数的调用

输入1，执行 查询余额 函数

输入2，执行 取款 函数

输入3，执行 转账 函数

输入0，执行 退出

其它：提示输入有误

3、假如，临时系统维护，取款和转账功能不可用，统一输出 "系统维护中"

https://www.baidu.com/s?ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=%E4%BD%B3%E8%83%BD&rsv\_pq=8b508d2900006957&rsv\_t=709eiWXlBCyq01qE3AdUCGN4n7ExL5QNgSs699oD0yGSdupqUdQSepE0G6c&rqlang=cn&rsv\_enter=1&rsv\_sug3=9&rsv\_sug1=6&rsv\_sug7=100&rsv\_sug2=0&inputT=4117&rsv\_sug4=5947

Day4

1、变量的作用域

1、值传递

值传递，传递的是值的副本

2、函数的作用域

3、ECMAScript提供的全局函数

parseInt()

parseFloat()

isNaN()

isFinite()

encodeURI()

decodeURI()

encodeURIComponent()

decodeURIComponent()

eval()

4、递归的调用

自己调自己

基本实现：

1、边界条件

2、递归前进 - 继续调用自己

3、递归返回 - 不再调用自己

2、分支结构

1、if(条件){}

注意：

1、条件不是boolean的话，则会自动转换为boolean

if(0)

if(0.0)

if("")

if(null)

if(undefined)

if(NaN)

以上内容，当做假处理

2、if后的{}是可以省略的，如果省略则只控制if下的第一条语句

2、if(条件){} else{}

3、else if

if(条件){

}else if(条件){

} ... else{}

4、switch ... case

switch(变量){

case 值1:

语句块1;

break;

case 值2:

语句块2;

break;

case 值3:

语句块3;

break;

default:

语句块n;

break;

}

switch(变量){

case 1:

case 2:

case 3:

语句块;

break;

}

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

1、循环结构

1、循环的特点(要素)

1、循环条件 - 循环的次数

2、循环操作 - 一遍又一遍要执行的相同的操作是什么

2、while

语法：

while(条件){

循环操作

}

注意：

1、在循环操作中，要更新循环条件，否则就是死循环

练习：

1、1 ~ 100 之间所有数字之和

2、1 ~ 100 之间所有偶数的和

3、改版 银行程序，用户可以反复输入，直到输入 0 则退出程序

4、使用while循环完成猜数字游戏

1、程序随机生成一个 1-100之间的数字

Math.random() : 生成并返回一个 0-1之间的小数

2、让用户输入一个整数

输入的数 如果 比随机数大，输出 猜大了

输入的数 如果 比随机数小，输出 猜小了

否则 ：输出 恭喜您，猜对了

3、如果用户没猜正确，那么可以循环猜，直到猜对为止，或者输入 "exit" 也可以退出

3、循环的流程控制语句

1、break

作用：提前结束整个循环

2、continue

作用：终止本次循环，继续执行下次循环

练习：

1、打印 1-100之间非3的倍数的数字

4、do ... while()

1、语法

do{

循环操作

}while(条件);

2、do while 与 while 的区别

1、do while

先执行循环体，再判断条件

至少执行一次循环体

2、while

先判断条件，再执行循环体

最少一次都不执行

练习：改版 猜数字游戏

5、for

1、语法

for(表达式1;表达式2;表达式3){

循环操作;

}

表达式1：循环条件的声明

表达式2：循环条件的判断

表达式3：更新循环条件

ex:

1、用while循环打印1-100之间所有的数字

var i = 1;

while(i <= 100){

console.log(i);

i ++;

}

2、用 for 循环改版 打印 1-100之间所有的数字

for(var i=1;i<=100;i++){

console.log(i);

}

注意:

在确定循环次数的情况下，优先使用 for

不确定循环次数的情况下，优先使用 while / do ... while

练习：

1、定义一个函数，接收一个参数，打印乘法口诀表中的任意一行

1\*1=1

...

1\*3=3 2\*3=6 3\*3=9

2、打印 1+1/3+1/5+1/7+...+1/999 的和

3、随意输入一个数字，判断是否为素数(质数)

素数：从2开始，只能被1和它本身整除的数字才是素数

4、判断天数

1、要求用户输入年，月，日(分3次输入)

2、判断该日是该年的第多少天

ex:

输入年：2017

输入月：3

输入日：1

2017年3月1日是2017年的第60天

输入年：2017

输入月：7

输入日：24

2017年7月24日是2017年的第...天

5、猴子吃桃：猴子第一天摘了若干个桃子，当即，吃了一半，还不过瘾，又多吃了一个，第二天，又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃前一天剩下的一半零一个。到第10天，发现只剩下一个桃子。求：第一天摘了多少桃子

day left total

天数 当天剩余 前一天剩余

10 1 4

9 4 10

8 10 22

7 22 46

6 46 94

5 94 190

4 190 382

3 382 766

2 766 1534

(total / 2) - 1 = left

(total/2)=left+1

total=(left+1)\*2

1、共计算 9次(循环次数)

条件：从10开始，到2结束

通过第10天剩余的算第9天

通过第09天剩余的算第8天

通过第08天剩余的算第7天

...

通过第02天剩余的算第1天

2、在循环中

1、通过 left 计算 total

total = (left + 1) \* 2

2、将total的值赋给left再进行运算

left = total;

6、兔子问题：有一对兔子，从出生后的第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问某个月的兔子总数为多少？

月数 1 , 2 , 3 , 4 , 5 , 6

兔子 1 , 1 , 2 , 3 , 5 , 8

用程序打印出 前10个月，每个月的兔子数量是多少(打印菲波那切数列的前10个数)

分析：

1、已知条件

第一个数字(n1) ：1

第二个数字(n2) ：1

2、求第三个数字

第三个数字 = 第一个数字 + 第二个数字

n1 = n1 + n2;

此时的n1 相当于是第3个数

3、求第四个数字

第四个数字 = 第二个数字 + 第三个数字

n2 = n2 + n1;

此时的n2 相当于是第4个数

4、求第五个数字

第五个数字 = 第四个数字 + 第三个数字

n1 = n2 + n1

此时的n1 相当于是第5个数

5、求第六个数字

第六个数字 = 第五个数字 + 第四个数字

n2 = n1 + n2;

此时的n2 相当于是第6个数

2、for循环三个表达式特殊用法

1、三个表达式都可以被省略

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

console.log(i);

}

1、省略第一个表达式

缺失变量，可以省略，但尽量在循环外部将声明补齐

2、省略第二个表达式

死循环

3、省略第三个表达式

死循环

2、表达式1 和 表达式3 的多样化

表达式1 和 表达式3 的位置处，分别可放置多个表达式，中间用 , 隔开即可

练习：

1 + 6 = 7

2 + 5 = 7

3 + 4 = 7

4 + 3 = 7

5 + 2 = 7

6 + 1 = 7

3、作业

要求用户输入年，月，日，判断这一天是星期?

前提：1900年1月1日 是星期1

ex:

输入年：2017

输入月：7

输入日：24

2017年7月24日是星期1

整体思路：

计算从1900.1.1 到 2017.7.24 共有多少天

用总天数 % 7 ,得出来的数据就是星期几

得1 ：星期一

得2 ：星期二

...

得0 ：星期日

推荐步骤：

1、通过循环计算1900 到 2016(输入年的前一年)，共有多少天

平年 ， 累加365

闰年 ， 累加366

2、通过循环计算2017.1.1 到 2017.6.30(从输入年份第一天开始，到输入月份的前一个月为止)进行累加

3、再输入的天数 (本例中是24) 再进行累加

4、通过上述计算出来的数字(天数)，再跟7取余，并打印结果

ex:

2017年7月24日是星期一

最后星期用汉字体现

星期一，星期二，星期三 ..

而不是

星期1，星期2，星期3..

Day4

1、while

while(条件){

循环操作

}

循环的流程控制：

1、break

退出整个循环

2、continue

终止本次循环，继续执行下次循环

2、do ... while()

do{

... ...

}while(条件);

3、for

for(表达式1;表达式2;表达式3)

表达式1:循环条件的声明

表达式2:循环条件的判断

表达式3:更新循环变量

使用场合：

确定循环次数时，优先使用for循环

三个表达式的特殊用法：

1、三个表达式可以任意的省略

2、表达式1 和 表达式3 可以多样化

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

1、循环

1、循环的嵌套

在一个循环体内再出现一个循环

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

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

}

}

练习1：页面打印

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

练习2：页面打印

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

练习3：在控制台中打印 九九乘法表

1\*1=1

1\*2=2 2\*2=4

... ...

练习4：打印在控制台上

\*

\*\*\*

\*\*\*\*\*

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

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

第1行 ：4个空格 1个星 (2\*1-1)

第2行 ：3个空格 3个星 (2\*2-1)

第3行 ：2个空格 5个星

第4行 ：1个空格 7个星

第5行 ：0个空格 9个星 (2\*5-1)

分析：

1、空格数量与行数的关系

空格从1开始，到5-行为止

2、星星数量与行数的关系

星星从1开始，到2\*行-1为止

注意

外层循环走一次，内层循环走一轮

程序 = 数据结构 + 算法

1、数组

1、什么是数组

数组，就是数据的组合，即一个变量中允许存放多个值

数组中的元素是按照线性顺序来排列的，也是典型的"栈"式结构。

2、声明数组(一维)

1、声明一个空数组变量

var 数组名 = [];

2、声明数组并赋值(初始化)

var 数组名 = [数据1,数据2,数据3];

3、声明一个空数组变量

var 数组名 = new Array();

4、声明数组并赋值(初始化)

var 数组名 = new Array(数据1,数据2,数据3);

练习：

用2种不同的方式创建数组(arr1,arr2)，分别保存数据 "张无忌","赵敏","周芷若","金花婆婆",并打印输出

3、数组是引用类型的对象

数据是存放在 "堆" 中的

在"栈"中会对 "堆" 中数据的地址进行引用

1、将数组赋值给变量的时候，实际上赋值的是地址

练习：

1、创建一个数组 arr1，包含"张无忌","赵敏","周芷若"

2、将 arr1 赋值给变量 arr2

3、将 arr2 的第一个元素 更改为 "金毛狮王"，打印输出 arr1 和 arr2中所有的元素

2、数组作为参数的时候，也是按值传递，但是传递的是数组的地址的副本

练习：

1、创建一个数组array，保存 "张无忌","赵敏","周芷若" 三个元素

2、创建一个函数change，接收数组作为参数，在函数体内，将第一个元素更改为 "金花婆婆"

3、调用change函数，将 array 作为参数传递进去，调用完成后，在change函数外打印array的值，观察结果

4、修改 change 函数中的内容，将 传递进来的参数 重新new 一个 Array，并初始化数据("孙悟空","猪八戒","沙悟净")

5、重复步骤3

3、null

表示 不再指向任何地址 即 释放对象的空间

null 和 undefined

1、相同点

都是原始类型，数据都保存在 "栈" 中

2、不同点

1、undefined

表示变量声明过但未赋值，一般为未赋值变量的默认值。从不主动使用。

2、null

表示变量不再指向任何地址。一般用于主动释放指向对象的引用(手动释放空间)

4、数组的使用

1、访问数组元素

使用的是 下标即索引

var arr = ["张无忌","赵敏"];

arr[5] = "金毛狮王";

console.log(arr[3]);

2、length 属性

作用：获取数组的长度(元素的个数)

1、允许将 数组的length属性设置为0，来完成数组元素的清空操作

2、配合循环做 循环遍历 数组中的每一个元素

var arr = [3,5,79,8];

for(var i=0;i<arr.length;i++){

console.log(arr[i]);

}

练习：

声明一个数组，保存数据 "张无忌","赵敏","周芷若"

通过一个循环遍历打印输出以下格式的内容：

第1名：张无忌

第2名：赵敏

第3名：周芷若

再倒着打印一次(倒叙输出)

第3名：周芷若

第2名：赵敏

第1名：张无忌

3、配合着数组名称，找到最新元素要插入的位置

var array = ["张无忌","赵敏","周芷若"];

array[array.length] = "金花婆婆";

练习：

完成"信息录入系统"，可以不停的接收录入的员工的姓名，直到输入 "exit"为止，将这些姓名保存在一个数组中。在控制台中打印所有的元素

3、使用 for ... in 遍历数组元素

允许使用 for ... in 遍历数组中每个元素的下标

语法：

for(var 变量 in 数组){

//变量：表示的是数组中的每个元素的索引

}

4、关联数组

用字符串做下标

var array = ["张三丰","张无忌"];

array["xiyouji"]="孙悟空";

array["shuihu"]="西门庆";

array["honglou"]="刘姥姥";

console.log(array.length);

关联数组的内容，只能通过 字符串 做下标来取值

允许通过 for...in 来循环遍历关联数组

练习：

1、定义函数，找出一个整数数组中的最大值

var nums = [78,65,12,5,98,3];

该数组中最大值为:98

1、声明函数 findMax，在函数体内声明数组

2、声明一个变量 max ，用于保存该数组中的最大值。设置默认值为 该nums中的第一个数

3、从第二个元素开始循环遍历，依次与 max 进行比较，谁大，谁赋值给max

2、定义函数，查找元素的位置(下标)

var nums = [78,65,12,5,98,3];

输入数字：12

该元素的下标为：2

输入数字：999

您输入的元素不存在!

提示：

声明一个变量 index ，用于保存查找到的元素的下标，默认值为 -1

3、作业

在上面练习基础上，输入一个数字，打印出该数字数组中所有的位置

var nums = [78,65,12,5,12,98,3,12];

输入一个数字：12

出现的位置是：2,4,7

提示：将出现的所有的下标，保存在一个数组中

基本数据类型:

number,string,boolean,null,undefined,NaN

值传递

function change(a,b){

a = 50;

b = 50;

}

var a = 500;

var b = 500;

change(a,b);

console.log(a,b); //500,500

Day6

1、循环

1、嵌套循环

在一个循环的内部又嵌入了一个循环

外层循环执行一次，内层循环执行一轮

2、数组

1、声明数组

var 数组名 = [];

var 数组名 = [数据1,数据2];

var 数组名 = new Array();

var 数组名 = new Array(数据1,数据2);

使用数组元素靠下标(从0开始 到 length-1 为止)

2、数组是引用类型的数据

堆中存数据，栈中存地址

1、引用类型在赋值的时候实际上赋的地址

var array = ["张无忌"];

var a1 = array; //将 array的地址赋值给了a1

a1[0] = "赵敏";

console.log(array[0]); // 赵敏

2、引用类型在做参数传递的时候，实际传的是地址的副本

1、

var array = ["张无忌"];

function change(arr){

arr[0] = "赵敏";

}

change(array);

console.log(array[0]); //赵敏

2、

var array = ["张无忌"];

function change(arr){

//将副本的地址重新指向了一个引用(地址)

arr = new Array("赵敏");

}

change(array);

console.log(array[0]); //张无忌

3、null

空，表示元素不再指向任何地址(释放空间)

var arr = ["张无忌"];

arr = null;

3、数组的使用

1、length 属性

获取数组的长度

2、for ... in

3、关联数组

由字符串充当数组的下标

=========================================

1、数组

1、练习

"冒泡排序":对一组数字进行排序，原理：两两数字相比较，小的往前排

ex ：[23,9,78,6,45] -> [6,9,23,45,78]

特点：

1、外层循环 ：控制比较的轮数,总共比较了 length-1 轮

2、内层循环 ：控制每轮中比较的次数，随着轮数的增加，次数反而减少

比较两个数字的大小，小的靠前，大的靠后

2、数组的常用API方法

API : Application Programming Interface

应用程序 编程 接口

1、字符串拼接

1、toString()

把数组转换为字符串，并返回结果

var array = [2,15,8];

var str = array.toString();

2、join(分隔符)

将数组的元素，用指定的 分隔符 连接到一起

练习：

1、使用数组实现三角形(\*)的输出

2、使用数组实现乘法表的输出

2、连接数组

函数：concat()

作用：拼接两个或更多的数组,并返回拼接后的结果

语法：arr1.concat(arr2,arr3);

注意：

1、该函数不改变现有数组，仅仅是返回拼接后的

2、每个参数可以是具体数值，也可以是数组

练习：

1、声明两个全局数组，每个数组分别保存一些国家的名称

数组1 ：入选国家

sel = ["中国","美国","俄罗斯"];

数组2 ：备选国家

unsel = ["泰国","菲律宾","老挝"];

2、将 备选国家 全部移入到 入选国家中,打印两个数组中的值

3、获取子数组

从某数组中，截取几个连续的元素组成新的数组

函数：slice()

语法：arr.slice(start,[end]);

start : 从哪个下标处开始选取，取值为正，从前向后算，取值为负，则从后向前算

0 1 2 3

var arr=["中国","美国","日本","英国"]

-4 -3 -2 -1

end : 指定结束处的下标(不包含);可以省略，如果省略的话，相当于是从 start 一直截取到结束

注意：

1、该函数不会改变现有数组，会返回全新的子数组

4、修改数组

允许从数组中删除一部分元素，同时再添加另外一部分元素

函数：splice()

语法：

arr.splice(start,count,e1,e2,...);

start : 指定添加，或删除元素的起始位置

count : 要删除的元素个数，取值为0则表示不删除

e1,e2,e3,... : 在start位置处要添加的新内容

返回值：被删除元素的数组

注意：splice 会改变现有的数组

练习：

既上午练习后的内容继续

1、从 备选国家 中，选择一个国家移入到 选中国家

var sel = ["中国","美国","俄罗斯","日本","朝鲜"];

var unsel=["泰国","老挝","缅甸","柬埔寨"];

//推荐步骤 ：以老挝为例

1、通过老挝，在unsel数组中找到老挝的下标(查找算法)

2、通过老挝的下标，将老挝从unsel数组中移除出去(使用 splice 函数),splice 的返回值就是从数组中移除的元素

3、将 splice 的返回值在添加到 sel 数组后

备选国家如下：泰国，老挝，缅甸，柬埔寨

输入：老挝

选中的国家：中国，美国，俄罗斯，日本，朝鲜，老挝

备选国家：泰国，缅甸，柬埔寨

2、从 选中国家 中，选择一个国家日入到 备选国家

5、数组反转

颠倒数组中元素的顺序

语法：array.reverse();

注意：该函数会改变当前数组的内容

练习：

1、声明一个整数数组

2、倒序打印输出

var array = [1,2,3,4,5];

array.reverse();

console.log(array);

6、数组排序

函数：sort

语法：array.sort([sortBy]);

作用：

默认情况下，按照元素的Unicode码的大小按升序排列

注意：直接改变现有数组的内容

练习：

var array = [12,6,89,4,72,115];

使用 sort 函数对以上数组进行默认排序，打印输出结果

115,12,4,6,72,89

特殊：允许自己指定排序函数，从而实现对数字的升序 或 降序的排列

array.sort(排序函数);

//排序函数

function sortAsc(a,b){

return a - b;

}

var array = [12,6,89,4,72,115];

array.sort(sortAsc);

原理：

1、指定 排序函数 sortAsc，定义 a 和 b两个参数，数组会自动传递进去，如果返回值是>0的数，则交换两个数字的位置，否则不变

2、将排序函数指定给sort，数组会自动的两两比较

匿名函数：

是一段没有名称只有功能的函数，只为某一个操作而存在，不能被其它函数调用

练习：

1、声明一个全局数组，数据随意

2、页面上提供2个按钮，一个"升序"按钮，一个"降序"按钮

3、点击 "升序" 按钮时，将全局数组升序排序，并打印输出结果

4、点击 "降序" 按钮时，将全局数组降序排序，并打印输出结果

array.sort(function(a,b){return a-b});

7、进出栈操作

JS是按照标准的"栈式操作"访问数组的

讲究的是 "入栈" 和 "出栈"

所有的"栈式操作"的特点是"后进先出"

var array = [];

arra[12] = "张三丰";

1、出入栈函数

1、push()

入栈，在栈顶(数组的尾部)添加指定的元素，并返回新的长度

var arr = [10,20];

//向栈顶增加新数据30(将30保存在数组的尾部)

var length = arr.push(30);

//length 保存的是 30 入栈后 数组的长度

2、pop()

出栈，删除并返回栈顶(数组尾部)的元素

var arr = [10,20,30];

var r1 = arr.pop();

var r2 = arr.pop();

==============================

arr : [10]

r1 : 30

r2 : 20

3、shift()

作用：删除数组头部的元素并返回

var arr = [10,20,30];

var r1 = arr.shift();

=========================

arr : 20,30

r1 : 10

4、unshift()

作用：在数组的头部(第一个元素)增加新元素

var arr = [10,20,30];

arr.unshift(40);

====================

arr 的结果为：40,10,20,30

2、练习

创建一个函数，接收一个十进制整数，返回其对应的二进制数表示

function decode2bin(num){

... ...

}

ex:decode2bin(5) : 返回 101

decode2bin(3) : 返回 11

十进制转二进制 ：除2取余法

5(num) % 2 ： 结果为1

num = parseInt(num / 2); -- 2

2(num) % 2 ： 结果为0

num = parseInt(num / 2); -- 1

1(num) % 2 ： 结果为1

num = parseInt(num / 2); -- 0

8、二维数组

1、什么是二维数组

一个数组中的元素又是一个数组,即数组中的数组

ex:

var arr=[[10,20,30],[40,50,60],[70,80,90]];

2、访问二维数组

1、访问数组中的单个元素

arr[0][1] : 结果为 20

想获取 80 : arr[2][1]

2、循环遍历二维数组 - 嵌套的循环结构

3、作业

根据省份，查询城市列表

1、创建一个全局数组，保存所有的省份名称

2、创建一个全局二维数组，保存不同省份中的地级信息(数据要与保存省份名称的数组对应上)

3、创建一个函数，接收一个省份名称做参数，打印该省中的所有地级市的信息