JSDAY05

学习目标

函数相关（重点）

变量作用域（重点）

函数作用域（重点）

递归调用（掌握）

1. 函数相关

   parseInt()/parseFloat()/typeof()....

   分类：系统函数和自定义函数

   自定义函数

   function：功能体，函数可以接收若干个数据，返回处理的结果。

   用于封装反复执行的代码——饺子机

   （1） 创建普通函数

   ```
   function 函数名称(){
   函数体——要封装的要反复执行的代码
   }
   ```

   调用

   函数名称();

   练习：使用函数来封装 10+20 结果，并打印出来。调用三次

   练习：计算 1-100 间所有数字的和，打印结果，调用三次

   （2） 创建带有参数的函数

   ```
   function  函数名称(参数列表){   //形参->形式上参数
   函数体
   }
   ```

   调用

   函数名称（参数列表）； //实参->实际参数

   参数列表：可以是 0 个或多个，之间用逗号隔开。创建函数是的参数成为形参，调用函数时的参数为实参，调用时，实参赋值给形参

   形参本质上就是一个声明的变量，但未赋值

   练习：创建函数 getSum，传递 1 个参数，计算 1~任意数字之间的和

   练习：创建 getRun，传递两个参数，打印连个年份间的所有闰年个数

   （3） 带有返回值的函数

   ```
   function 函数名称(参数列表){
   函数体
   return  返回值
   };
   ```

   调用

   函数名称（参数列表）

   return 表示函数执行后，所返回的结果。

   ① 如果没有 return 或其后没有返回值，结果都是 undefined

   ② return 可以返回任意类型数据

   ③ return 后的所有代码都不会被执行

   练习：创建函数 add，传递 3 各参数，返回 3 个参数和

   练习：创建函数 getMax，传递两个数，返回最大值

   练习：创建函数 getMax，传递 3 个数，返回最大值

   练习：创建函数 isRun,传递一个参数（年份），是闰年返回 true，否则返回 false

   练习：创建函数 getstatus，传递一个参数，返回对应中文；1-待付款，2-待发货，3-运输中，4-已签收，5-已取消，其他-无法查询

   对比 return、break、continue

   return 返回结果，用于终止函数的执行，常用于函数中；

   break 用于结束循环，结束 swith 语句；

   continue 跳过当前语句，继续下一个循环；

2. 变量的作用域

   （1） 作用域

   变量或者函数的可访问范围，分为两种：

   全局作用域：在全局作用域下声

的变量可以在任意位置访问
函数（局部）作用域：函数作用域下证明的变量只能在函数内部访问
在函数内部使用 var 关键字声明的变量是局部变量，不使用 var 关键字声明的是全局变量。

（2） 变量声明提升
JS 程序执行前，使用 var 声明的变量会提升到所在作用域最前端，但赋值还是在原来位置

```
console.log（a）
var a=1；//var a 提升到最前端，但赋值还是在原来位置
```

（3） 函数作用域
函数和变量类似，也分为全局作用域和函数作用域
全局作用域：在全局作用域下创建的函数可以在任意位置调用
函数作用域：在函数（局部）作用域下创建的函数只能在函数内部调用

函数声明提升：
和变量一样，JS 在程序执行前，把使用 function 声明的函数提升到最前边。
注意：如果函数名称和 var 声明的变量名称一致，则变量值会覆盖函数
函数调用 fun()获取函数的返回值（return 后的值）
函数名称： fun 保存的是函数在堆内存中的地址

（4） 递归调用
在函数内部调用自身
递归要有递归条件，需要结合 return 使用。
斐波那契数列
1 1 2 3 5 8 13 21 34 55 89 144
练习：创建函数，传递一个参数，计算菲薄导图第 n 项

JSDAY06

1.匿名函数
没有名字的函数 function(){ }

```
创建函数——函数声明
function 函数名称(){}
```

(1)创建函数——函数表达式

```
var 函数名称=function(形参列表){
  函数体
  return 返回值;
}
调用：函数名称(实参列表)
```

对比函数声明和函数表达式的区别
函数声明存在函数提升，在任何位置都可以调用
函数表达式不存在函数提升，必须先创建再调用
练习：使用函数表达式计算 1~100 之间的和。
(2)匿名函数自调用
创建函数(局部)作用域，防止造成全局污染。

```
(function(){
  函数体//就是局部作用域,创建的变量和函数都是不能被外部访问
})();
```

(3)作为回调函数
把匿名函数作为实参传递给形参，此时的形参就是函数名称

```
function fn(num){
  //num 就是函数的名称
  num(); //调用传递的匿名函数
}
fn(function(){ ... });
```

练习：创建函数 add，传递 2 个参数，2 个参数都是以匿名函数的形式传递；在匿名函数中分别返回 1 个数字。
最后在 add 中计算两个数字相加。
2.系统函数(全局函数)
encodeURI 对一个 url 进行编码
decodeURI 对一个已经编码的 url 进行解码
parseInt 将数据转为整型
parseFloat 将数据转为浮点型
isNaN 检测一个数据是否为 NaN 是->true 不是->false
isFinite 检测一个数据是否为有限值，是->true 不是->false 1/0 -> Infinity 无限值
eval 执行字符中表达式 eval('1+2') -> 3

练习：使用弹出提示框输入一组运算，使用 eval 执行这组运算。 06_eval.html 06_eval.js

3.对象

是一种引用类型的数据，存储在堆内存中。

对象：是一组**属性**(property)和**方法/功能**(method)的集合

哪些是对象？

一个手机: 属性有颜色，品牌，尺寸，厚度... 功能有打电话、发短信、照相、聊天、游戏....

一辆汽车：属性有品牌、外观、车型... 功能有代步、撞人、拉货、取暖、乘凉...

(1)JS 中的对象

内置对象：JS 提供的

宿主对象：根据不同的执行环境来划分

自定义对象：自己创建的对象

(2)创建自定义对象

①对象字面量(直接量)

②内置构造函数

③自定义构造函数

4.使用对象字面量创建对象

使用大括号{ }创建空对象

属性名和属性值之间用冒号隔开

多组属性之间用逗号隔开

属性名中引号可加可不加，如果出现特殊字符，必须添加引号

```
var phone={ color:'red', 'made-in':'china' };
```

练习：创建一个部门对象，属性有部门编号，名称，员工数量

练习：创建一个员工对象，属性有编号，姓名，性别，生日，工资，所在部门

5.使用内置构造函数创建对象

```
var book=new Object();  //创建一个空对象
book.id=103;  //添加属性 ,属性名不能添加引号
book['titlte']='三国演义'; //添加属性，属性名必须加引号；如果不加引号，会被认为是变量.
```

练习：创建一个汽车对象，属性有编号，品牌，颜色，价格

练习：创建一个电脑对象，属性有型号，品牌，尺寸，产地

6.访问对象中的属性

(1)获取属性值

```
emp.eid
emp['ename']
如果要获取的属性名不存在，返回 undefined
```

(2)遍历对象中的属性(for-in)

获取对象中每一个属性名，进而获取属性值

```
for(var key in emp){
  //emp  要遍历的对象
  //key  要遍历的每一个属性名
  //emp[key]  通过属性名获取对应的属性值
}
```

注意：for-in 不能遍历预定义(JS 默认为每一个对象添加的)属性。

练习 :创建一个商品对象 ,包含编号 ,标题 ,价格，是否在售(isOnsale)，商品库存量(stockCount)，使用 for-in 遍历所有的属性。

7.检测对象中是否含有某个属性

'属性名' in 对象 // true->存在 false->不存在

对象.hasOwnProperty('属性名') //true->存在 false->不存在

对象.属性名=== undefined //true->不存在 false->存在

8.对象中的方法

```
var person={
  name: 'tom',
  say: function(){
    this.name  //this 指代当前的对象
  }
}
person.say(); //调用对象中的方法
```

练习：

创建一个圆对象，添加属性半径，圆周率；添加方法计算周长(getLength)和计算面积(getArea)；分别返回圆的周长和面积

JSDAY07

1.创建数组

(1)数组字面量

[元素 1,元素 2...]

数组中可以存放任意类型的数据

(2)内置构造函数

```
new Array(5)  //初始化数组的长度为 5 ,可以继续添加第 6 个元素...
new Array(元素 1,元素 2....);
```

访问数组中的元素

数组名称[下标]，下标是从 0 开始

练习：创建数组，初始化长度为 5，添加 5 个课程。

练习：创建数组，直接添加 5 个员工的姓名。

2.获取数组元素的个数

数组名称.length

使用数组长度添加一个新的元素

数组[数组.length]=值

练习：创建一个空数组，使用数组长度添加 3 个国家名称。

3.数组的分类

数组分为**索引数组**和**关联数组**

索引数组：以整数作为下标

关联数组：以字符串作为下标

```
var arr=[]
arr['eid']=1;
arr['ename']='tom';
```

4.遍历数组中的元素

for 循环，循环遍历数组元素的下标

```
var scrore=[85,79,93];
for(var i=0;i<score.length;i++){
    i   代表数组元素的下标
    score[i]   下标对应的元素
}
注意：只能遍历索引数组。
```

for-in

```
for(var key in arr){
   key  数组的下标
   arr[key]   下标对应的元素
}
注意 :既可以遍历索引数组,也可以遍历关联数组。
```

练习：创建数组，保存所有员工的工资，使用 for 循环遍历计算平均工资。

练习：创建数组，保存 4 个汽车的品牌名称，把品牌名称为'宝马'改为'BMW';

使用循环来找到宝马的下标，使用下标改变元素的值为'BMW'

练习：创建数组，包含元素推荐、热点、娱乐、财经 ,分别使用 for 循环和 for-in 遍历该数组中元素。

练习：创建一个函数 getIndex，传递 2 个参数(数组，字符串)，返回字符串的下标，如果找不到该字符串的下标，则返回-1；

5.数组中的方法(API)

API-应用程序编程接口，预先定义好的函数/方法

toString() 将数组中的元素按逗号分隔成字符串

join('|') 将数组中的元素按照指定的字符分隔成字符串

concat(arr1,arr2...) 拼接两个或者更多的数组

slice(start, end) 截取数组中的元素，start 开始的下标，end 结尾的下标，不包含 end 本身；负数表示倒数。

splice(start, count,value1,value2....) 删除数组中的元素；start 开始的下标，count 删除的个数，value1,value...删除后补充的元素。

reverse() 翻转数组中的元素

sort() 对数组中的元素进行排序，默认按照 Unicode 由小到大

```
对数字排序
sort(function(a,b){
    return a-b;   //由小到大
    //return b-a;   //由大到小
})
```

练习：

将数组中的元素进行翻转 ,不能使用 reverse

['a','b','c','d'] -> ['d','c','b','a']

了解冒泡排序

JSDAY08

1.数组 API(方法)

push() 往数组的末尾添加元素，返回数组的长度

pop() 删除数组末尾的元素，返回删除的元素

unshift() 往数组的开头添加元素，返回数组的长度

shift() 删除数组开头的元素，返回删除的元素

2.二维数组

数组中的每一个元素也是数组

```
var arr=[ [], [], []... ]
```

访问二维数组中的元素 arr[下标][下标]

3.字符串操作

包装对象：目的是让原始类型的数据也可以像引用类型一样，具有属性和方法。

JS 提供了 3 种包装类型：String、Number、Boolean

将任意数据包装成字符串对象

new String(true) 返回 object 类型

String(true) 返回字符串 'true'

(1)转义字符 —— \

转换字符的意义

\n 将字符 n 转义成换行

\t 将字符 t 转义成制表符(tab 键)

\' 将引号转成普通的字符

...

练习：打印出现 welcome to chi\na

(2)字符串中常用方法

toUpperCase() 将英文字母转为大写

toLowerCase() 将英文字母转为小写

练习：初始化 4 个英文字母(有大小写)保存变量中，循环弹出提示框，输入 4 个字符(不区分大小写)，如果输入正确结束循环。 06_code.html 06_code.js

length 获取字符串的长度

charAt() 获取下标对应的字符

charCodeAt() 获取某个字符对应的 Unicode 码

练习：使用变量保存字符串“javascript”，获取字符“a”出现的个数。

indexOf(value, start) 查找某个字符串的下标，value 是要查找的字符串，start 开始查找的下标，默认是 0，如果找不到返回-1

lastIndexOf(value) 查找某个字符串最后一次出现的下标，找不到返回-1

练习：声明变量保存字符串，检测该字符串是否为邮箱格式 ;如果是打印 true ,不是打印 false；(查看是否含有@)

slice(start, end) 截取字符串，start 开始的下标，end 结束的下标，不包含 end 本身；如果 end 为空，截取到最后

substr(start, count) 截取字符串，start 开始的下标，count 截取的长度；如果 count 为空，截取到最后

substring() ?

练习：使用变量存储身份证号，获取其中年月日和性别。打印 1997 年 05 月 20 日 男

110230199705203310 倒数第 2 位 偶数->女 奇数->男

练习：变量保存邮箱地址，分别截取邮箱的用户名和域名

jerry1995@sina.com.cn

练习：将一个英文单词首字母转大写，其余转小写

'hELlo' -> 'Hello'

split(sep) 按照指定的字符分隔为数组 ,sep 是分隔符

练习：使用 split 获取邮箱的用户名和域名(jerry@qq.com)

练习：将一句英文所有单词的首字母转大写，其余字母转小写； 'hOW arE yOu' -> 'how are you'

['hOW', 'arE', 'yOu'] -> ['How', 'Are', 'You']

4.匹配模式(掌握)

作用：用于查找、替换字符串。

replace(value1, values) 查找并替换。value1 要查找的字符串 ,value2 要替换的字符串 ;value1 可以使用字符串形式，也可以使用正则表达式形式 /china/ig

i -> ignore 忽略大小写

g -> global 全局查找

match(value) 用于查找匹配的字符串 ,返回一个数组

search(value) 用于查找满足条件的第一个字符的下标，如果找不到返回-1

5. Math 对象

Math 对象不需要使用 new 创建 ,可以直接使用。

PI 获取圆周率

abs() 取绝对值

floor() 向下取整

ceil() 向上取整

round() 四舍五入取整

max() 取一组数字的最大值

min() 取一组数字的最小值

pow(x,y) 取 x 的 y 次幂

random() 取随机 >=0 <1

练习：

创建数组，存储若干个员工的姓名，每次运行随机取出 1 个员工的姓名打印出来

创建数组，包含 a-z, A-Z, 0-9 元素；在数组中随机取 4 个字符，放到一个新数组中。 push