1. 对象

属于 引用类型数据

对象：是一组属性（property）和方法（method）的集合

例：一台电脑：属性有颜色尺寸品牌等...方法有看视频上网玩游戏等

回忆：JS共六种数据类型：

原始类型数据：null undefined string number boolean

引用类型数据 object

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划分（nodeJS或浏览器）

自定义对象：自己创建的对象

1. 创建自定义对象

对象字面量 var a = {...};

内置构造函数 var a = new object ();

自定义构造函数

1. 对象字面量创建对象

1）使用大括号创建空对象

2）属性名和属性值间用冒号隔开

3）多组属性间用逗号隔开

4）属性名中引号可加可不加，若含有特殊字符则必须加

{'come-from':'China'} //属性名有特殊字符--，要加引号

例：

//使用 对象字面量 创建对象

// 添加属性 用':'赋值

var employee={id:2333,

name:'lily',

salary:3000,

birthday:'1997-10-4',

'come-from':'China'} //属性名有特殊字符要加引号

//访问对象中的属性2种方式

console.log(employee.birthday);

console.log(employee['come-from']);

employee.num=15000; //添加不存在的属性

console.log(employee);

1. **访问**对象中的属性（注意等于）

对象.属性名 用" . "连接 添加/修改属性也可以这样写，后接等号赋值

对象['属性名'] 添加属性，属性名必须加引号，不加会认为是变量

例：var book = {

id:113,

title:'海猫鸣泣之时',

author:'龙骑士07',

price:38

}

book.price = 88; //修改

book.publish='非洲农民出版社'; //添加不存在属性

console.log(book);

访问对象中的属性的2种方式：

console.log(book.id);

console.log(book['lue']); //若访问属性不存在，会返回undefined

1. 内置构造函数创建对象（目前了解）

new Object( ) 创建一个空对象

需要**单独**为对象添加每一个属性

例

var Notebook = new Object() （括号中不能添加数据）

Notebook.Id = 1024;

Notebook.price = 3000;

console.log(Notebook);

1. 遍历对象中的属性

访问对象中的每一个属性

|  |
| --- |
| for (var key in 对象){  key //代表对象中的每个属性名  对象[key] //通过属性名或者对应属性值（不要加引号）遍历  } |

例：for(var dt in book){ //dt代替上面所有属性名

console.log(dt,book[dt]);

}

例2：创建对象，输入成绩，遍历对象属性，计算总成绩

var score={

chi:94,

math:98,

eng:88,

mus:96

}

var sum = 0; //要写外面

for(var test in score){ //遍历，类似循环...

sum += score[test];

console.log(test,score[test],sum);

}

console.log(sum);

1. 判断对象中是否有包含某个属性

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

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

3)'属性名' in 对象 存在->true 不存在->false

var person={

name:'tom',

age:20

}

//检测对象中是否含有某个属性

console.log( person.sex===undefined );

console.log( person.hasOwnProperty('score') );

console.log( 'name' in person );

1. 对象中的方法

|  |
| --- |
| var person={  name:'tom',  say:function(){  this.name //访问当前对象的属性名，this指代当前所在的对象  }  }  person.say() //调用对象中的方法 |

例1：

var person={

name:'张三', //成员属性

sex:'女',

say:function(){ //成员方法

console.log('hai~，我是'+this.name);

}} person.say(); //调用对象中的方法

打印结果：hai~，我是张三

例2：求圆的周长和面积

var yuan={

r:5,

pai:3.14,

c: function(){

return this.r \*this.r\*this.pai //结果要return出来

},

l: function(){

return this.r \*2\*this.pai

}

}

yuan.c(); //注意括号！

yuan.l();

console.log(yuan.c());

console.log( yuan.l());

1. **引用类型存储**

例：

var person1={ //

name:'小明',

age:19};

var person2=person1; //person1 赋予了person2

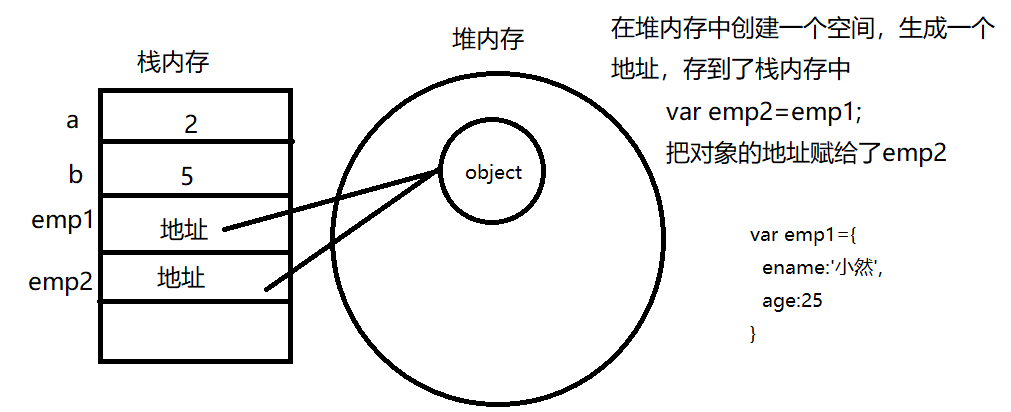
person1.name = '小可';

person2.age = 26;

console.log(person1);

console.log(person2);

输出结果都为{ name: '小可', age: 26 }



引用类型数据存储过程：

数据存储到堆内存后会自动生成一个地址（也成指针、引用）放到变量中存储，实际复制的是地址而不是堆内存内容。（堆内存生成的16进制地址，无法看到)

对象的销毁：当对象没有被任何地址引用时，会自动被销毁

手动释放(销毁)：person1 = null; person2=null；

还记得null的属性是object吗

1. **数组（array）**

是由多个元素组成的集合，每个元素就是一个数据（可以让一个变量包含多个元素）

1. 创建数组：
2. 数组字面量

var 数组=[ 元素1 , 元素2, .... ] 中括号括起来，中间用逗号分隔

1. 内置构造函数

var 数组 = new Array (元素1, 元素2, ... );

... new Array(3) 初始化元素个数3，可添加更多

1. 访问数组中的元素

数组名[ 下标 ] 例： ！！不要加'.'

cities[0] （下标是从0开始编号的）

求数组的长度： （比较像对象）

console.log( cities.length );

使用长度在数组末尾添加元素：

数组 [ 数组.length ]= ' 元素 '；

1. 数组的分类

分为索引数组和关联数组

索引数组 以0以上的整数作为下标

关联数组 以字符串作为下标，只能单独的添加元素（不建议使用）

1. 遍历数组

for--in

|  |
| --- |
| for（var key 数组）{  key 下标  数组[key]  } 既可遍历关联数组，也可遍历索引数组 |

例：

var sum=0 ; //要写外面

var grade = [96,77,82,54,79,86,98];

for (var ky in grade )

{ sum += grade[ ky ];

};

console.log(sum);

循环

|  |
| --- |
| for ( var i=0,i<数组.length,i++){  i 下标  数组[i ] 下标对应的元素  } 只能遍历索引数组 |

使用循环获取数组中的元素

var sum=0;

var grade = [96,77,82,54,79,86,98];

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

{ sum+= grade[i];

}

console.log(sum);

例2：替换数据中为'中'的值为'china'

var country = ['中','日','美','韩','英'];

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

{ if ( country[i] ==='中' )

{ country[i] = 'china';

}

}

console.log(country);

例3：创建数组，包含多个数字，获取数组中最大值

var hh=[84,75,35,99,45];

//准备遍历，用于保存最大值，默认把第一个元素放进去

var max = hh[0]; //从第二个开始遍历

for (var i=1; i<hh.length ; i++)

{ if (hh[i]>max)

{ max = hh[i];

}

}

console.log(max);

例4：创建函数getAvg，传递一个参数（数组），返回平均值

function getAvg(a){

for (var i=0,sum=0; i<a.length;i++ ) //sum如果在里面声明就要return

{ sum += a[i];

}

return sum/a.length;

}

console.log(getAvg([27,39,15,54]));

1. **数组API（方法）**

API 应用程序编程接口，预定义好的一些方法或者函数

都放在arr . \_\_\_\_\_\_部分

1. toString() 将数组中元素按指定符号组合成字符串（默认为'，'）

（tostring（2）可以将数字转字符串同时按2进制显示）

1. join('-') 将数组转换为字符串，同时可改变分隔符 ，（括号内留空默认为逗号分隔，引号空字符则无缝拼接）
2. arr1.concat(arr2,arr3...) 多个数组的拼接（注意不要拼接重复了）
3. slice(start,end) 截取数组中的元素，start开始的下标，end结束的下标，但不包含end本身。如果是负数，表示倒数。end为空表示截取到最后。
4. splice(strat,count,value1,value2..) 三个参数，删除数组中的元素，start开始的下标（为负表倒数），count删除的长度（不能为负，为空删除到最后）。value删除后补充的元素，返回值为删除的元素，原数组会发生改变（要重新打印查看）。
5. reverse() 翻转数组中的元素
6. sort() 对数组元素排序，默认是按照首字母Unicode码从小到大排序

插入一个回调 （前面还是要加arr . ）

|  |
| --- |
| sort(function(a,b){  return a-b; //按照数字从小到大  //return b-a; 按照数字从大到小  }); |

例：

var arr=[23,9,78,6,45];

//数组元素排序 默认按照Unicode码从小到大

console.log(arr.sort());

console.log(arr.sort(function(a,b){

return a-b;

}) );

打印结果 [ 23, 45, 6, 78, 9 ] 按照首字母Unicode排序结果

[ 6, 9, 23, 45, 78 ]

1. push(...) 在数组的末尾添加元素（类似arr[arr.length]=...）返回

当前数组元素总数。 ↑优先使用push

1. unshift() 往数组开头添加元素，返回数组的长度
2. pop() 删除数组最后一个元素，返回删除成功的元素
3. shift() 删除数组开头的元素，返回删除元素

拓展：冒泡函数

var arr=[23,9,78,6,45];

for (var i=1;i<arr.length ;i++ )

{ for (var j=0;j<arr.length-i; j++) //内存循环，控制每轮比较次数

{ if (arr[j]>arr[j+1] ) //如果当前元素大于下一个元素，则位置交换

{ var tem =arr[j];

arr[j]=arr[j+1];

arr[j+1]=tem;

}

}

} console.log(arr);

打印结果 ： [ 6, 9, 23, 45, 78 ]