正课:

\*\*\*数组:

1. \*\*\*关联数组:

2. 数组API

3. 栈和队列

4. 二维数组

1. \*\*\*关联数组:

什么是: 可自定义下标名称的数组

vs 索引数组: 下标都是数字的数组

为什么: 为了让每个元素都有专门的名称，更便于维护和快速查找

何时: 2种

1. 希望每个元素都有明确的意义时

2. 快速定位想要的元素时

如何:

创建: 2步:

1. 先创建空数组: var ym=[];

2. 向数组中添加新元素，使用自定义的下标名称:

ym["name"]="杨幂";

ym["math"]=81;

ym["chs"]=59;

ym["eng"]=89;

访问: 用法同索引数组中的元素，同变量的用法

.length: 失效，永远为0

遍历: for(var key in ym){//in 自动依次取出ym中每个下标名称保存在变量中key中

ym[key] //进一步获得当前key对应的元素值

强调: key不要加""

}

优点: 查找极快

因为: 直接通过下标定位元素位置

查找速度和元素个数以及元素存储位置无关

vs 索引数组: 只能靠遍历查找:

缺: 受元素个数和元素存储位置的影响极大

2. 数组API:

什么是API: 别人已经写好的，咱们用现成的程序

什么是数组:

存储多个数据，并提供操作数据的API的对象

包括:

将数组转字符串: 2种:

String(arr): 将arr中每个元素都转为字符串，默认用逗号连接。——拍照

何时: 查看数组的中间修改状态，判断是否发生了变化

arr.join("自定义连接符"):将arr中每个元素都转为字符串，可自定义连接符。

何时: 只要不希望用逗号连接时，就必须用join

固定套路:

1. 无缝拼接: 错误: arr.join() => String(arr)

正确: arr.join("")

2. 动态生成页面元素: 2步

1. var html=

"<ANY>"+arr.join("</ANY><ANY>")+"</ANY>"

2. elem.innerHTML=html;

拼接和选取:

拼接: 将多个数组或元素拼接为一个新数组

如何: var newArr=arr1.concat(值1,值2,arr2,...);

强调:1. 无法修改原数组arr1，只能返回新数组

2. 打散数组类型的参数为单个元素再拼接

选取: 选择原数组中指定开始位置到结束位置之间的多个元素组成新数组

如何: var subArr=arr.slice(starti,endi+1);

强调:

1. 不修改原数组，仅复制出想要的元素组成新数组

2. 如果一个API两个参数都是下标时，就含头不含尾

简写:

1. 支持负数参数: 代表倒数第n个位置

选择到倒数第n个位置:

arr.slice(starti,arr.length-n+1)

可简写为: arr.slice(starti,-n+1)

原理: 其实slice自动用length-n

2. 可省略第二个参数，表示一直选取到结尾!

arr.slice(starti) 从starti开始一直选到结尾

3. 两个参数都省略: 从头到尾完整复制一个数组

var arr\_clone=arr.slice();

修改: 删除, 插入, 替换——splice

删除: arr.splice(starti,n) 删除starti位置开始的n个元素

强调: 1. 直接修改原数组

2. 不考虑含头不含尾

简写: 1. 支持负数参数: 代表倒数第n个位置

其实自动用length-n

2. 省略第二个参数: 一直删到结尾

其实: splice有返回值:

有时删除的元素，依然有用

splice会将删除的元素组成新的数组返回

var deletes=arr.splice(starti,n)

其中: deletes保存了删除的元素

插入: arr.splice(starti,0,值1,值2,...)

在starti位置插入新的值1,值2,...

强调: 1. 原starti位置的值及其之后的值被向后顺移

2. splice不支持打散数组类型参数

如果插入一个数组，则变为二维数组

替换: 其实就是先删除，再插入:

arr.splice(starti,n,值1,值2,...)

强调: 删除的元素个数和插入的新元素个数不必一致

翻转: arr.reverse()

\*\*\*排序: 将数组中的元素按从小到大或从大到小的顺序重新排列

何时: 任何数据在给用户展示前，必须先排序

如何: arr.sort()

强调: 直接修改原数组

原理: 默认将每个元素都转为字符串，再按字符串比较大小，并排序

问题: 只能排序字符串类型的数据

解决: 自定义比较器函数

什么是: 专门比较任意两值大小的函数

何时: 如果sort默认的排序规则不是想要的，可自定义比较器代替sort中默认的排序规则

如何: 2步:

1. 定义比较器函数:

要求: 2个参数

返回值: 如果a>b，就返回正数

如果a<b, 就返回负数

否则,就返回0

var cmp=function(a,b){return a-b;}

2. 将比较器函数作为参数传入sort函数中

arr.sort(cmp);

强调: cmp后不要带()

为什么: 回调函数callback

什么是回调: 函数定义后，传递给其它函数，由其它函数负责调用。——给

简写: arr.sort(function(a,b){return a-b;});

问题: 如何降序:

解决: 只要颠倒比较器结果的正负

function(a,b){return b-a;}

栈和队列:

js中没有专门的栈和队列类型, 都是用普通数组模拟的.

栈stack:

什么是: 一端封闭，只能从另一端进出的数组

何时: 今后只要希望始终使用最后进入数组的新元素

特点: FILO

如何:

1. 结尾出入栈:

结尾入栈: arr.push(值) =>arr[arr.length]=值

在末尾追加一个新值

结尾出栈: var last=arr.pop()

移出数组末尾的最后一个元素

2. 开头出入栈:

开头入栈: arr.unshift(值)

开头入栈后的结果和结尾入栈后的结果刚好颠倒

开头出栈: var first=arr.shift()

队列queue:

什么是: 只能从一端进入，从另一端出的数组

何时: 只要希望按照先来后到的顺序使用数组元素时

特点: FIFO

如何:

结尾入队列: arr.push(值)

开头出队列: var first=arr.shift()

二维数组:

什么是: 数组中的元素又引用了另一个子数组

何时: 2种:

1. 保存横行竖列的二位数据

2. 如果大的数组中的元素需要进一步细致分类时

如何:

创建: 2种:

1. 先创建空数组，再添加子数组  
 var data=[];

data[0]=[0,0,0,0];

data[1]=[0,0,0,0];

data[2]=[0,0,0,0];

data[3]=[0,0,0,0];

2. 创建数组同时，初始化数组内容:

var data=[

[0,0,0,0],

[0,0,0,0],

[0,0,0,0],

[0,0,0,0]

]

访问: data[r][c] 用法和普通数组的元素以及变量完全一样

特殊: 二维数组行下标r不能越界！越界报错！

遍历: 外层循环控制行，内层循环控制列

for(var r=0;r<data.length;r++){

for(var c=0;c<data[r].length;c++){

data[r][c] //获得当前元素

}

}