数组13种常用方法

第二阶段 扩展的数组的方法
filter,forEach,map,some,every,reduce,reduceRight,copyWithin,find,findIndex,fill,includes

- 1.push 往数组的末尾添加内容
  - 参数：添加的内容
  - 返回值：添加内容后数组的长度
  - var res = arr.push(6,7);
- 2.unshift 往数组的起始位置添加内容
  - 参数：添加的内容
  - 返回值：添加内容后数组的长度
- 3.pop 删除数组的最后一项
  - 参数：无
  - 返回值：删除的内容
  - var res = arr.pop();
- 4.shift 删除数组第一项
  - 参数：无
  - 返回值：删除的内容
- 5.splice(n) 从索引n开始删除到最后
  - 返回值:将删除的内容放在一个新数组中返回
  - var ary = [10,5,3,2,15];
    var res = ary.splice(2);
- splice(n,m) 从索引n开始删除m个
  - 返回值:将删除的内容放在一个新数组中返回
  - var ary = [10,5,3,2,15];
    var res = ary.splice(2,2)
- splice(n,m,x) 从索引n开始删除m个，用x的内容代替删除的内容 从第三个开始的参数用x表示(添加的内容)
  - var ary = [10,5,3,2,15];
    var res = ary.splice(1,2,77,88,99);
    console.log(res);//[5,3]
  - var res = ary.splice(1,0,55,66); n=0表示在索引n之前添加内容
- 6.reverse 反向排列
  - 参数：无
  - 返回值：反向排列的数组
  - var ary = [10,5,3,2,15];
    var res = ary.reverse();
- 7.sort

- 参数：可以不传或传函数的定义
- var ary = [10,5,3,2,15];
  //var ary1 = [5,3,2,4,1];
  //var res = ary.sort(); //只能对10以内的数进行排序 (按照unicode编码（ASCII码）进行排列)
  var res = ary.sort(function(a,b){ //自己设置排序的方式是从大到小，还是从小到大
  return a-b //从小到大
  //return b-a //从大到小
  });

以上方法数组会改变

原有数组没有发生改变

```
var ary = [10,5,3,2,15];
```

- 1.toString 将数组转换成字符串
  - var res = ary.toString();
    //console.log(res);//”10,5,3,2,15”
    //[].toString() ->””
- 2.join() 将数组按照指定的字符拼接成字符串
  - 参数：指定的字符或不传
  - var res = ary.join(“+”)
    console.log(res);//”10+5+3+2+15”
  - var res = ary.join(“”); //将数组的每一项靠在一起返回一个字符串
  - var res = ary.join();//不传参返回结果跟toString返回结果一样

console.log(eval(res));//35 eval是全局下的方法，将字符串转换成JS代码执行

- 3.concat 合并数组的方法
  - 返回值：返回合并后的数组
  - var ary1 = [10,5];
    var ary2 = [20,3];
    var res = ary1.concat(ary2); //把ary2中的内容合并到ary1中
    console.log(res);//[10, 5, 20, 3]
  - var res = [].concat(ary1,ary2);//ary1和ary2合并到空数组中
    console.log(res)//[10, 5, 20, 3]
  - var res = ary1.concat(); //没有传参表示克隆
- 4.slice 截取数组中某些项
  - 参数：无或一个或两个
  - console.log(ary.slice()); //把原数组克隆一份
  - console.log(ary.slice(0)); //把原数组克隆一份
  - var res = ary.slice(2) //从索引2开始截取到最后
    console.log(res);//[3,2,15]

    - slice(n,m) 从索引n截取到索引m(包前不包后) n,m还可以是负数
    - var res = ary.slice(2,4);
      console.log(res)//[3,2]
    - console.log(ary.slice(-3, -1));//[3,2] 负数索引+数组的长度 倒数第三项到倒数第一项
- 5.indexOf和lastIndexOf 查找数组是否有这一项 若有则返回这一项的索引，若没有查找到则返回-1
    - indexOf 从左往右查找
    - lastIndexOf 从右往左查找
    - var ary = [10,5,2,5,13];
      console.log(ary.indexOf(7));//-1 没有7这一项，所以结果为-1
    - 扩展indexOf和lastIndexOf 还可以设置第二个参数(表示设置起始查找的位置)
      console.log(ary.indexOf(5, 2));//3 从左往右查找，从索引2位置开始查找

扩展方法：

- filter()：“过滤”功能，数组中的每一项运行给定函数，返回满足过滤条件组成的数组。
    - ```
      var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
      var arr2 = arr.filter(function(x, index) {
      return index % 3 === 0 || x >= 8;
      });
      console.log(arr2); //[1, 4, 7, 8, 9, 10]
      ```
- every()：判断数组中每一项都是否满足条件，只有所有项都满足条件，才会返回true。
    - ```
      var arr = [1, 2, 3, 4, 5];
      var arr2 = arr.every(function(x) {
      return x < 10;
      });
      console.log(arr2); //true
      var arr3 = arr.every(function(x) {
      return x < 3;
      });
      console.log(arr3); // false
      ```
- some()：判断数组中是否存在满足条件的项，只要有一项满足条件，就会返回true。
    - ```
      var arr = [1, 2, 3, 4, 5];
      var arr2 = arr.some(function(x) {
      return x < 3;
      });
      console.log(arr2); //true
      var arr3 = arr.some(function(x) {
      return x < 1;
      });
      console.log(arr3); // false
      ```
      ```

迭代方法

- forEach() 遍历数组没有返回值
- map（ ）遍历数组并能修改数组内容有返回值
- some() 数组中只要有一个成立，结果为true，都不成立返回false
- every（ ）都成立返回true 只要有一个不成立返回false
- reduceRight 从右往左求数组的累计值

ES6迭代方法

- find() 查找满足条件的这一项，找到了则方法终止运行
- findIndex() 返回满足条件这一项的索引，找到了则方法终止运行
- from() 类数组转成数组
- Array.of() 将参数（一个或多个）转换成数组

```
var ary = ["lily","20","gender"];
    for(let value of ary){
        console.log(value);//数组的每一项
    }
    for(let key of ary.keys()){
        console.log(key);//索引
    }
    for(let ele of ary.entries()){
        console.log(ele);//[索引,成员]
        let [key,value] = ele;
    }
```

- copyWithin(target,start,end) 拷贝数组的部分内容到指定位置（会覆盖原来的位置），数组的长度变，
  - target指定放置的起始位置索引1‥5
  - start拷贝内容的起始位置索引
  - end拷贝内容的结束位置索引，（包前不包后）

字符串常用方法

> 掌握字符串常用的11个方法
> **根据索引查找字符**
> console.dir(String.prototype);//查看字符串中有哪些方法

- 1.charAt 根据索引查找对应的字符，若找不到则返回”“;
- 2.charCodeAt 根据索引返回对应字符的ASCII码值
  - var str = “abcABCDpfg”;

var res = str.charCodeAt(3);

- 2.截取字符串的方法
  - 以下方法若只有一个参数表示从索引n截取到最后
    - substr(n,m) 从索引n开始截取m个
    - substring(n,m) 从索引n截取到索引m (包前不包后)
    - slice(n,m) 从索引n截取到索引m (包前不包后) 可以是负数索引
    - var str = “abcABCDpfg”;
      /*var res = str.substr(2,5);*/
      //var res = str.substring(-1,2); //负数转换成0
      //var res = str.substring(2,5); //cAB
      var res = str.slice(2,4);//cA
      console.log(str.length);
      var res = str.slice(-4,-2);//负数索引 = 负数索引+字符串的长度
  - 3.split 将字符串按照指定的字符拆分成数组中的每一项
  - var str = “2018-08-28”;
    var res = str.split(“-“);
    console.log(res);//[“2018”, “08”, “28”]
  - var res = str.split(“”);将每个字符串拆开放入数组中console.log(res);// [“2”, “0”, “1”, “8”, “-“, “0”, “8”, “-“, “2”, “8”]
  - var res = str.split();//将这个字符串放入数组中
    console.log(res);//[“2018-08-28”]
- 4.转换大小写的方法
  - toUpperCase 全部转换成大写
  - toLowerCase 全部转换成小写
  - var str = “abcDFG”;
    console.log(str.toUpperCase());//”ABCDFG”
    console.log(str.toLowerCase());//”abcdfg”
- 5.查找字符串是否有这个字符（通过字符串->索引）
  - indexOf 从左往右查找，找到则返回该字符索引，若找不到则返回-1
  - lastIndexOf 从右往左查找，找到则返回该字符索引，若找不到则返回-1
  - var str = “abcDFGabc”;
    console.log(str.indexOf(“b”));//1
    console.log(str.lastIndexOf(“b”));//7
    console.log(str.lastIndexOf(“k”));//-1
- 6.replace 替换字符串 返回值是替换后的结果
  - var str = “zhufeng2018zhufeng”;
    var res = str.replace(“zhufeng”,”珠峰”).replace(“zhufeng”,”珠峰”);
    var res = str.replace(/zhufeng/g,”珠峰”);
    console.log(res);
- 7.match 参数：字符串或正则 将匹配的字符串放在一个数组中返回
- 8.search 检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串。参数：字符串或正则 返回：位置索引，找不到返回-1

ES6新增字符串方法

- 9.includes 查看是否包含某个字符或字符串 返回true 或false
- 10.startsWith 查看是否以某个字符或字符串开头，第二个参数设置查找的位置，默认从索引0开始查找 返回true 或false
- 11.endsWith 查看是否以某个字符或字符串结尾 ，默认查找到最后，第二个参数设置最后的索引
- 12.repeat 重复某个字符或字符串，参数值得是重复的次数
- //”2018-08-28 17:54:30” ->”2018年08月28日17时54分30秒”

  var date = “2018-08-28 17:54:30”;
  //先把字符串截取成日期部分和时间部分
  //日期部分通过-拆分成数组的每一项
  //时间部分通过：拆分成数组的每一项

  //1.先获取空格的索引
  var index = date.indexOf(” “);//10
  var str1 = date.slice(0,index);//”2018-08-28”
  var str2 = date.slice(index+1);//”17:54:30”
  var ary1 = str1.split(“-“);//[“2018”,”08”,”28”];
  var ary2 = str2.split(“:”);//[“17”,”54”,”30”]
  var resStr =
  ary1[0]+”年”+ary1[1]+”月”+ary1[2]+”日”+ary2[0]+”时”+ary2[1]+”分”+ary2[2]+”秒”;
  console.log(resStr);//”2018年08月28日17时54分30秒”

DOM节点

浏览器渲染时，页面上的内容会渲染成有层次结构的节点，一个页面只有一个根节点`document`,根节点下根元素只有一个，就是html标签
文档->文档节点
文本->文本节点
注释->注释节点
标签->元素节点

nodeName(节点名称)nodeType(节点类型)|nodeValue(节点内容)

》	nodeName	nodeType	nodeValue
文档节点	`#document`	9	null
文本节点	`#text`	3	文本的内容（包括换行）
注释节点	`#comment`	8	注释的内容
元素节点	`大写的标记名`	1	null

DOM节点之间相互关系的属性

1. `childNodes` 所有的子节点(文本节点，元素节点，注释节点)
2. `children` 所有的子元素节点 （ IE8下把文本节点当成元素节点 ）
3. `firstChild` 第一个子节点
4. `firstElementChild` 第一个子元素节点 （ ie6~ie8不支持 ）
5. `lastChild` 最后第一个子节点
6. `lastElementChild` 最后一个子元素 （ ie6~ie8不支持 ）
7. `nextSibling` 相邻弟弟节点
8. `nextElementSibling` 相邻弟弟元素节点 （ ie6~ie8不支持 ）
9. `previousSibling` 相邻哥哥节点
10. `previousElementSibling` 相邻哥哥元素节点 （ ie6~ie8不支持 ）
11. `parentNode` 父元素节点

获取DOM元素

- id名 `document.getElementById(id名)`
- 标记名 `context.getElementsByTagName("li")`
- 类名 `context.getElementsByClassName("a")` ie6-ie8不支持
- name属性 `document.getElementsByName("")`

  > 在标准浏览器下对所有元素起作用 但在ie下只对表单元素起作用

- 选择器 `document.querySelectorAll()` 一组元素
  `document.querySelector()` 一个元素

  > 在移动端常用的方法

- 设置DOM元素的自定义属性
  - `ele.setAttribute(key,value);`
  - `ele.getAttribute(key);`

递归调用

- 递归调用：指方法自己调用自己 (重复执行方法里的功能)
  - 1.在方法内部调用自己的方法写在return后面
  - 2.设置边界条件，让递归调用停下来
- 将10以内的奇数想乘 1*3*5*7*9

```
function fn(n){
```

```
        if(n===1){
            return 1;
        }
        if(n%2==0){
            return fn(n-1);
        }else{
            return n*fn(n-2);
        }
    }
    console.log(fn(10));
```

- 100以内的数求和

```
function fn(n){
    if(n==1){
       return 1;
        }
       return n+fn(n-1);
    }
fn(100);
```

快速排序

- 思路：先取出数组的中间项，将数组的其他项跟中间项进行比较，若比中间项小则放在左手边，若比中间项大则放在右手边，左手边和右手边再重复上面的步骤，最后把所有的数合并在一起

```
    var ary1 = [33,2,5,12,18,9,10];
    function quickSort(ary){
处理边界
        if(ary.length<=1){
            return ary;
        }
先获取中间项索引
        var pointIndex =Math.floor(ary.length/2);
通过中间索引，把中间项从数组中删除
        var pointValue = ary.splice(pointIndex, 1)[0];
        var left = [];//左手边
        var right = [];//右手边
       for(var i = 0;i<ary.length;i++){
           if(ary[i]<=pointValue){
              left.push(ary[i]);
           }else{
              right.push(ary[i]);
```

```
            }
        }
        return quickSort(left).concat(pointValue,quickSort(right)) ;
    }
quickSort(ary1);
```

动态操作dom元素

- 创建dom节点 document.createElement（“div”）
- 创建文本节点 document.createTextNode（“珠峰”）
- 添加dom元素（添加到父节点内容的末尾位置） 父节点.appendChild（“oDiv”）
- 插入 父节点.insertBefore(newEle,oldEle)
- 替换 父节点.replaceChild(newEle,oleEle);
- 删除 父节点.removeChild(ele);
- 克隆 ele.cloneNode(true); true表示把所有的后代都克隆，不加true表示只克隆元素本身
- mainin。remove（ ）//可以删除自己

Math

- Math.sqrt() 开方
- Math.pow() 幂次方
- Math.abs() 绝对值
- Math.ceil() 向上取整
- Math.floor() 向下取正
- Math.round() 四舍五入
- Math.random() 取0~1的随机数
- Math.max() 求最大值
- Math.min() 求最小值

取0~10之间的随机整数 能取到0，取不到10

Math.floor(Math.random()*10)

取0~10之间的随机整数 不能取到0，取到10

Math.ceil(Math.random()*10)

取0~10之间的随机整数 能取到0，能取到10

Math.round(Math.random()*10)

2~62之间的随机整数 0~60 +2

Math.round(Math.random()*(62-2)+2);

n~m之间的随机整数 （ n<m ） 公式如下：

Math.round(Math.random()*(m-n)+n)

获取6个随机不重复整数

需求：从15~100之间随机的取6个不重复的整数放入数组中返回

- 1.取15~100之间的随机整数 Math.round(Math.random()*(100-15)+15)
- 2.如何取6次？ for循环取六次
- 3.解决不重复 存入数组之前判断下数组中是否有这一项，若没有则存入数组中，若有则重新取随机整数

```
function getRandom(){
        var ary = [];
        for(var i = 0;i<6;i++){
            var random = Math.round(Math.random()*(100-15)+15);
            if(ary.indexOf(random)===-1){//表示数组中没有这一项
         ary.push(random);
         continue;
            }
//问题：把重复项丢掉后，最终的数组中的成员就会少几项，就不是6个
              i--; //若重复的i的值不累加，拿这次取随机整数的机会就没有丢掉
        }
        return ary;
    }
    console.log(getRandom());

//循环的次数不确定，则推荐用while循环，若条件成立则一直执行循环体的内容，只有条件不成立时才退出while循环
function getRandom2(){
//while循环要找到条件判断是什么？
var ary = [];
while(ary.length<6){
     var random = Math.round(Math.random()*(100-15)+15);
     if(ary.indexOf(random)===-1){
        ary.push(random);
           }
       }
       return ary;
   }
console.log(getRandom2());
```

获取4位随机验证码

需求：取四个随机不重复的字符放入div1中

```
var str = "asdsadwersdv32twegrw3t2qevad2t4qegavd";
function getRandom(str){
//存入数组中，最后再把数组转换成字符串作为返回值
var ary = [];
while(ary.length<4){
//1.随机索引 ->随机的字符
var index = Math.floor(Math.random()*str.length)
var s = str.charAt(index);
if(ary.indexOf(s)===-1){
ary[ary.length] = s;//往数组中添加字符
  }
}
return  ary.join("");//相当于这个意思 ["a","b","c","d"]->"abcd" 数组的每一项靠一起返回一个字符串
}
var oDiv = document.getElementById("div1");
oDiv.innerHTML = getRandom(str)
```

动态操作dom元素

```
<div id="div2"></div>
<p>
<span>11111</span>
</p>
<script>
    var oDiv = document.createElement("div");
    oDiv.id = "div1";
   // oDiv.innerHTML = "珠峰培训";
    var oText = document.createTextNode("珠峰");
    oDiv.appendChild(oText);
    //oDiv.innerHTML = oText; //不对的 innerHTML的值必须是字符串
    //document.body.appendChild(oDiv);

   //插入
  /*  var oDiv2 = document.getElementById("div2");
    document.body.insertBefore(oDiv,oDiv2);*/

   //替换
    var oDiv2 = document.getElementById("div2");
    //document.body.replaceChild(oDiv,oDiv2);
```

```
    //删除
    document.body.removeChild(oDiv2);

    //克隆
    var oP = document.querySelector("p");
    var newP = oP.cloneNode(true);
    document.body.appendChild(newP);//克隆完后还需添加到页面才能显示
```

倒计时

- 定时器
  - window.setTimeout() 执行一次
  - window.setInterval() 如果不停止会一直执行

```
<div>距离下课还有<span>0时0分0秒</span></div>

需求：现在距离下课还有多少时多少分多少秒
var oSpan = document.querySelector("div>span");
function getComputed(){
   var now = new Date() ;//以现在时间为基准创建日期对象
   var target = new Date("2018/08/30 18:00:00");//以参数为基准创建日期
对象
//获取这个时间相差的毫秒数
//var time1 = target - now;
   var time = target.getTime() - now.getTime();
//1小时 = 60*60*1000
//1分钟 = 60*1000
//1秒 = 1000
//有多少小时数
   var hour = Math.floor(time/(60*60*1000));
//除去小时毫秒数剩下的毫秒数
   time = time - hour*(60*60*1000);
//有多少分钟数
   var minute = Math.floor(time/(60*1000));
//除去分钟毫秒数剩下的毫秒数
   time  =time - minute*(60*1000);
//有多少秒
   var second = Math.floor(time/1000);
   var str = hour+"时"+minute+"分"+second+"秒";
   oSpan.innerHTML = str;
    }
getComputed();
var timer = window.setInterval(getComputed,1000);
```

```
var oDiv  = document.querySelector("div");
oDiv.onclick = function(){
//若定时器是启动则让其停止
//若定时器是停止的则让其启动
console.log("点击刚开始",timer);
if(timer){//说明现在定时器是启动,要让其停止
     clearInterval(timer);
     timer = null;
}else{
timer = window.setInterval(getComputed,1000);//重新启动定时器
    }
console.log("点击结束后",timer);
}
```

求n到m之间的随机整数公式
Math.round(Math.random()*(m-n)+n)

考试：选项卡 、数组去重 、冒泡排序 、