字符串中常用的方法

> 所有用 单引号、双引号、反引号 包起来的都是字符串

```
let str = 'zhufengpeixunyangfanqihang';
// 每一个字符串都是由零到多个字符组成的
str.length //=>字符串长度
str[0] //=>获取索引为零（第一个）字符
str[str.length-1] //=>获取最后一个字符
str.length-1最后一项索引
str[10000] //=>undefined 不存在这个索引

//循环输出字符串中的每一个字符
for (let i = 0; i < str.length; i++) {
    let char = str[i];
    console.log(char);
}
```

`charAt / charCodeAt`

```
/*
 * charAt：根据索引获取指定位置的字符
 * charCodeAt：获取指定字符的ASCII码值（Unicode编码值）
 *  @params
 *      n [number] 获取字符指定的索引
 *  @return
 *      返回查找到的字符
```

```
 *      找不到返回的是空字符串不是undefined，或者对应的编码值
 */
let str = 'zhufengpeixunyangfanqihang';
console.log(str.charAt(0)); //=>'z'
console.log(str[0]); //=>'z'
console.log(str.charAt(10000)); //=>''
console.log(str[10000]); //=>undefined
console.log(str.charCodeAt(0)); //=>122
console.log(String.fromCharCode(122)); //=>'z'
```

`substr / substring / slice`

```
/*
 * 都是为了实现字符串的截取（在原来字符串中查找到自己想要的）
 *    substr(n,m)：从索引n开始截取m个字符，m不写截取到末尾（后面方法也是）
 *    substring(n,m)：从索引n开始找到索引为m处(不含m)
 *    slice(n,m)：和substring一样，都是找到索引为m处，但是slice可以支持负数作为索引，其余两个方法是不可以的
 */
let str = 'zhufengpeixunyangfanqihang';
console.log(str.substr(3, 7)); //=>'fengpei'
console.log(str.substring(3, 7)); //=>'feng'
```

```
console.log(str.substr(3));
//=>'fengpeixunyangfanqihang' 截取到末尾
console.log(str.substring(3, 10000));
//=>'fengpeixunyangfanqihang' 截取到末尾（超过索引的也只截取到末尾）

console.log(str.substring(3, 7));
//=>'feng'
console.log(str.slice(3, 7)); //=>'feng'
console.log(str.substring(-7, -3));
//=>'' substring不支持负数索引
console.log(str.slice(-7, -3));
//=>'nqih' slice支持负数索引 =>快捷查找：负数索引，我们可以按照 STR.LENGTH+负索引 的方式找   =>slice(26-7,26-3)  =>slice(19,23)
```

indexOf / lastIndexof / includes

```
/*
 * 验证字符是否存在
 *   indexOf(x,y)：获取x第一次出现位置的索引，y是控制查找的起始位置索引
 *   lastIndexOf(x)：最后一次出现位置的索引
 *   =>没有这个字符，返回的结果是-1
 */
let str = 'zhufengpeixunyangfanqihang';
console.log(str.indexOf('n')); //=>5
console.log(str.lastIndexOf('n'));
//=>24

```

```
console.log(str.indexOf('@')); //=>-1 不存在返回-1
if (str.indexOf('@') === -1) {
    // 字符串中不包含@这个字符
}

console.log(str.indexOf('feng')); //=>3 验证整体第一次出现的位置，返回的索引是第一个字符所在位置的索引值
console.log(str.indexOf('peiy')); //=>-1

console.log(str.indexOf('n', 7)); //=>12 查找字符串索引7及之后的字符串中，n第一次出现的位置索引

if (!str.includes('@')) {
    console.log('当前字符串不包含@');
}
```

`toUpperCase / toLowerCase`

```
/*
 * 字符串中字母的大小写转换
 *   toUpperCase(): 转大写
 *   toLowerCase(): 转小写
 */
let str = 'ZhuFengPeiXunYangFanQiHang';
str = str.toUpperCase();
console.log(str);
//=>'ZHUFENGPEIXUNYANGFANQIHANG'

```

```
str = str.toLowerCase();
console.log(str);
//=>'zhufengpeixunyangfanqihang'

// 实现首字母大写
str = str.substr(0, 1).toUpperCase() + str.substr(1);
console.log(str);
//=>'Zhufengpeixunyangfanqihang'
```

split

```
/*
 * split([分隔符]):把字符串按照指定的分隔符拆分成数组（和数组中join对应）
 *
 * split支持传递正则表达式
 */
// 需求：把|分隔符变为,分隔符
let str = 'music|movie|eat|sport';
let ary = str.split('|'); //=>["music", "movie", "eat", "sport"]
str = ary.join(',');
console.log(str);
//=>"music,movie,eat,sport"
```

replace

```
/*
 * replace(老字符,新字符)：实现字符串的替换（经常伴随着正则而用）
 */
let str = '珠峰@培训@扬帆@起航';
// str = str.replace('@', '-');
// console.log(str); //=>"珠峰-培训@扬帆@起航" 在不使用正则表达式的情况下，执行一次REPLACE只能替换一次字符

str = str.replace(/@/g, '-');
console.log(str); //=>珠峰-培训-扬帆-起航
```

`match`

`localCompare`

`trim / trimLeft / trimRight`

...

控制台输出 String.prototype 查看所有字符串中提供的方法

实现一些常用的需求

> 时间字符串的处理

```
let time = '2019-7-24 12:6:23';
//=> 变为自己需要呈现的格式，例如：
```

```
// "2019年07月24日 12时06分23秒"
// "2019年07月24日"
// "07/24 12:06"
// ...

//不足十位补零的方法
let addZero = val => val.length < 2 ? '0' + val : val;

//处理方式
let ary = time.split(/(?: |-|:)/g);
//=>["2019", "7", "24", "12", "6", "23"]
time = ary[0] + '年' + addZero(ary[1]) + '月' + addZero(ary[2]) + '日';
```

> 实现一个方法 queryURLParameter 获取一个URL地址问号后面传递的参数信息

```
/*
 * queryURLParams：获取URL地址中问号传参的信息和哈希值
 *    @params
 *       url [string] 要解析的URL字符串
 *    @return
 *       [object] 包含参数和哈希值信息的对象
 * by zhouxiaotian on 2019/07/24 16:29:00
 */
function queryURLParams(url) {
    //1.获取?和#后面的信息

```

```
    let askIn = url.indexOf('?'),
        wellIn = url.indexOf('#'),
        askText = '',
        wellText = '';
    // #不存在
    wellIn === -1 ? wellIn = url.length : null;
    // ?存在
    askIn >= 0 ? askText = url.substring(askIn + 1, wellIn) : null;
    wellText = url.substring(wellIn + 1);

    //2.获取每一部分信息
    let result = {};
    wellText !== '' ? result['HASH'] = wellText : null;
    if (askText !== '') {
        let ary = askText.split('&');
        ary.forEach(item => {
            let itemAry = item.split('=');
            result[itemAry[0]] = itemAry[1];
        });
    }
    return result;
}

/*
```

```
//基于正则封装的才是最完美的
function queryURLParams(url) {
    let result = {},
        reg1 = /([^?=&#]+)=([^?=&#]+)/g,
        reg2 = /#([^?=&#]+)/g;
    url.replace(reg1, (n, x, y) => result[x] = y);
    url.replace(reg2, (n, x) => result['HASH'] = x);
    return result;
}
*/

let aa = 'http://www.zhufengpeixun.cn/index.html?lx=1&name=zhufeng&teacher=aaa#box';
let paramsObj = queryURLParams(aa);
console.log(paramsObj);
```

> 实现一个最LOW的验证码：数字+字母共四位
>
> 验证码目的：防止外挂程序恶意批量注入的

```
<body>
    <input type="text" id="codeInp">
    <br>
    <span id="codeBox">AAAA</span>
    <button id="changeCode">看不清换一张</button>
```

```
6
7       <!-- IMPORT JS -->
8       <script>
9           let codeInp =
document.getElementById('codeInp'),
10              codeBox =
document.getElementById('codeBox'),
11              changeCode =
document.getElementById('changeCode');
12
13          /*
14           * queryCode：获取到四位随机验证码，
然后放到指定的盒子中
15           *   @params
16           *   @return
17           * by Team on 2019/07/24
18           */
19          function queryCode() {
20              // 准备获取范围的字符串 0~61
21              let area =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm
nopqrstuvwxyz0123456789";
22              let result = "";
23              for (let i = 0; i < 4; i++)
{
24                  // 每一次循环都获取一个随机
的数字索引
25                  let ran =
Math.round(Math.random() * 61);
```

```
                // 再根据获取的索引从范围字符串中找到对应的字符，把找到的字符拼接到最后的结果中
                result += area.charAt(ran);
            }
            // 放到盒子里面
            codeBox.innerHTML = result;
        }

        // 第一次加载页面需要执行方法，让其显示在页面中
        queryCode();

        // 点击看不清按钮，需要重新执行方法生成新的验证码
        changeCode.onclick = queryCode;

        // 文本框失去焦点的时候：验证用户输入的内容和验证码是否相同，给予相关的提示，如果不一样需要重新生成验证码
        // onblur：文本框失去焦点事件
        codeInp.onblur = function () {
            // 获取用户和验证码内容（表单元素.value / 非标单元素.innerHTML 获取内容）
            let val = codeInp.value,
                code = codeBox.innerHTML;
            // 不区分大小写的验证（都转成小写）
```

```
            if (val.toLowerCase() === code.toLowerCase()) {
                alert('温馨提示：验证码输入成功！');
            } else {
                alert('温馨提示：验证码输入有误，请重试！');
                codeInp.value = '';
                // 重新生成验证码
                queryCode();
            }
        }
    </script>
</body>
```