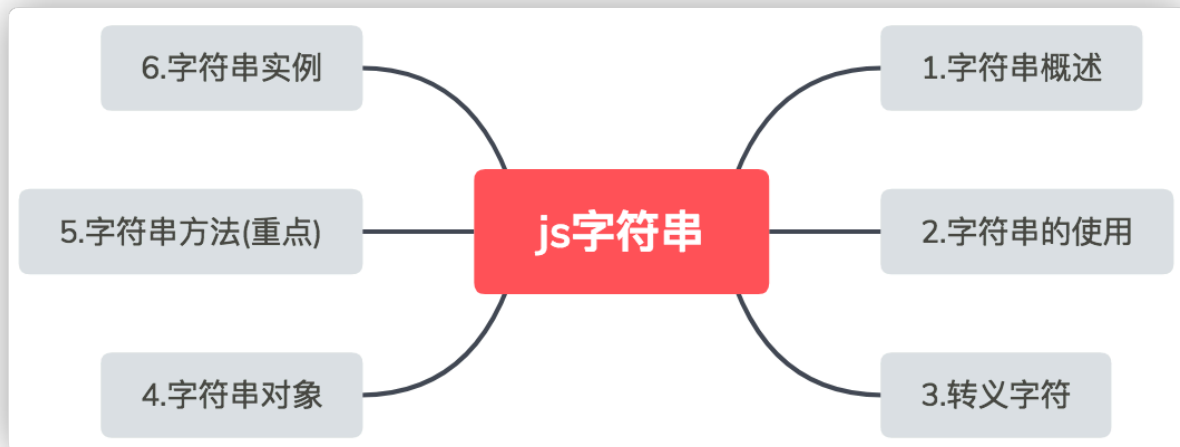


# JS字符串

---

## 0.1. 主要内容：



## 0.1. 学习目标：

| 节数             | 知识点       | 要求 |
|----------------|-----------|----|
| 第一节（字符串概述）     | 字符串介绍     | 了解 |
| 第二节（字符串的使用）    | length属性  | 了解 |
|                | 字符索引      | 了解 |
|                | 获取指定位置的字符 | 了解 |
|                | 字符串连接     | 了解 |
| 第三节（转义字符）      | 常见转义字符    | 掌握 |
| 第四节（字符串对象）     | 创建字符串对象   | 掌握 |
| 第五节（字符串方法(重点)) | 字符串方法     | 掌握 |
| 第六节（字符串实例）     | 字符串的使用    | 掌握 |

## 1. 字符串概述

定义：字符串就是用单引号或者双引号包裹起来的，零个或多个排列在一起的字符。

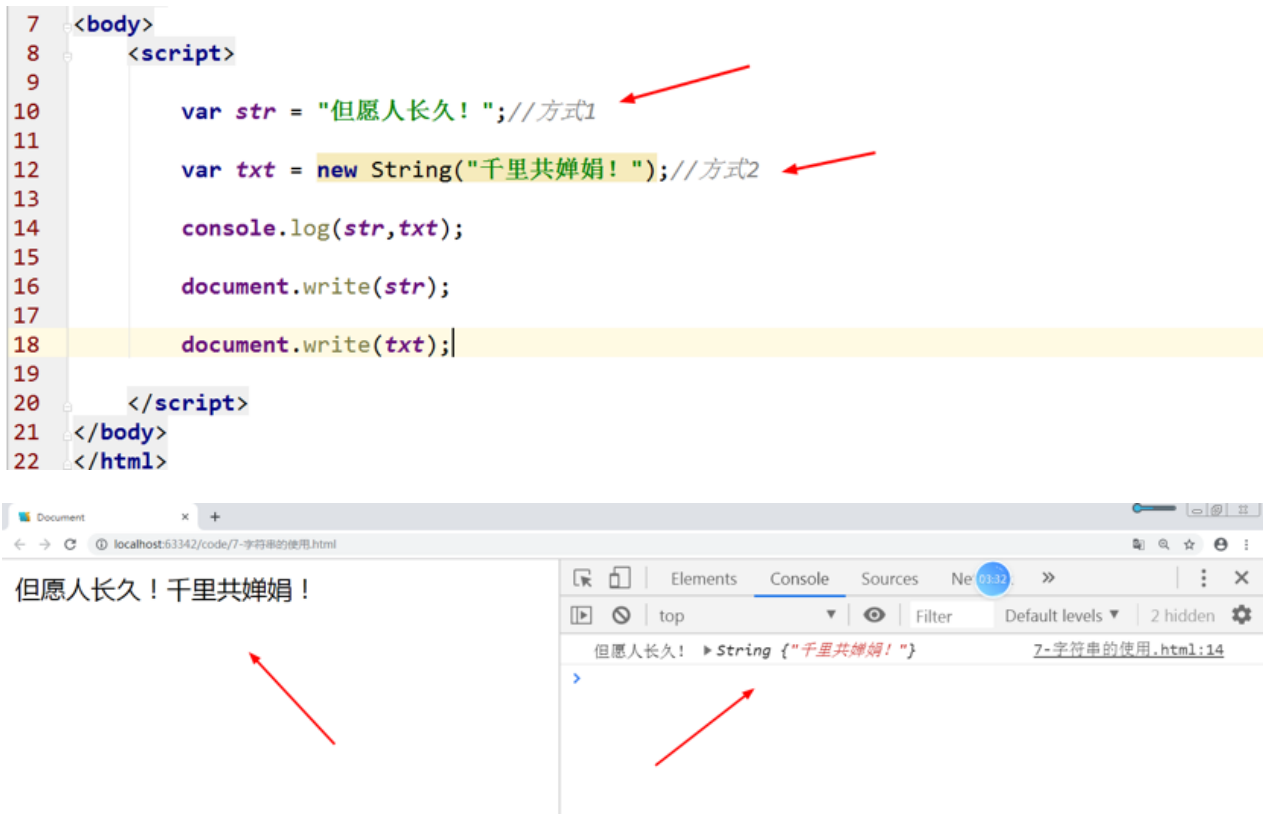
例如：'javascript', "", "345", '9-11a\$', "xiao\_yuanLian"

**嵌套：字符串可以嵌套。**

在单引号包裹的字符串内部，应该使用双引号进行嵌套。

在双引号包裹的字符串内部，应该使用单引号进行嵌套。

例如："I am 'coolMan'", 'are u "kidding" me'



## 2. 字符串的使用

### 2.1. 字符串换行

```
var x = "Hello World!";
```

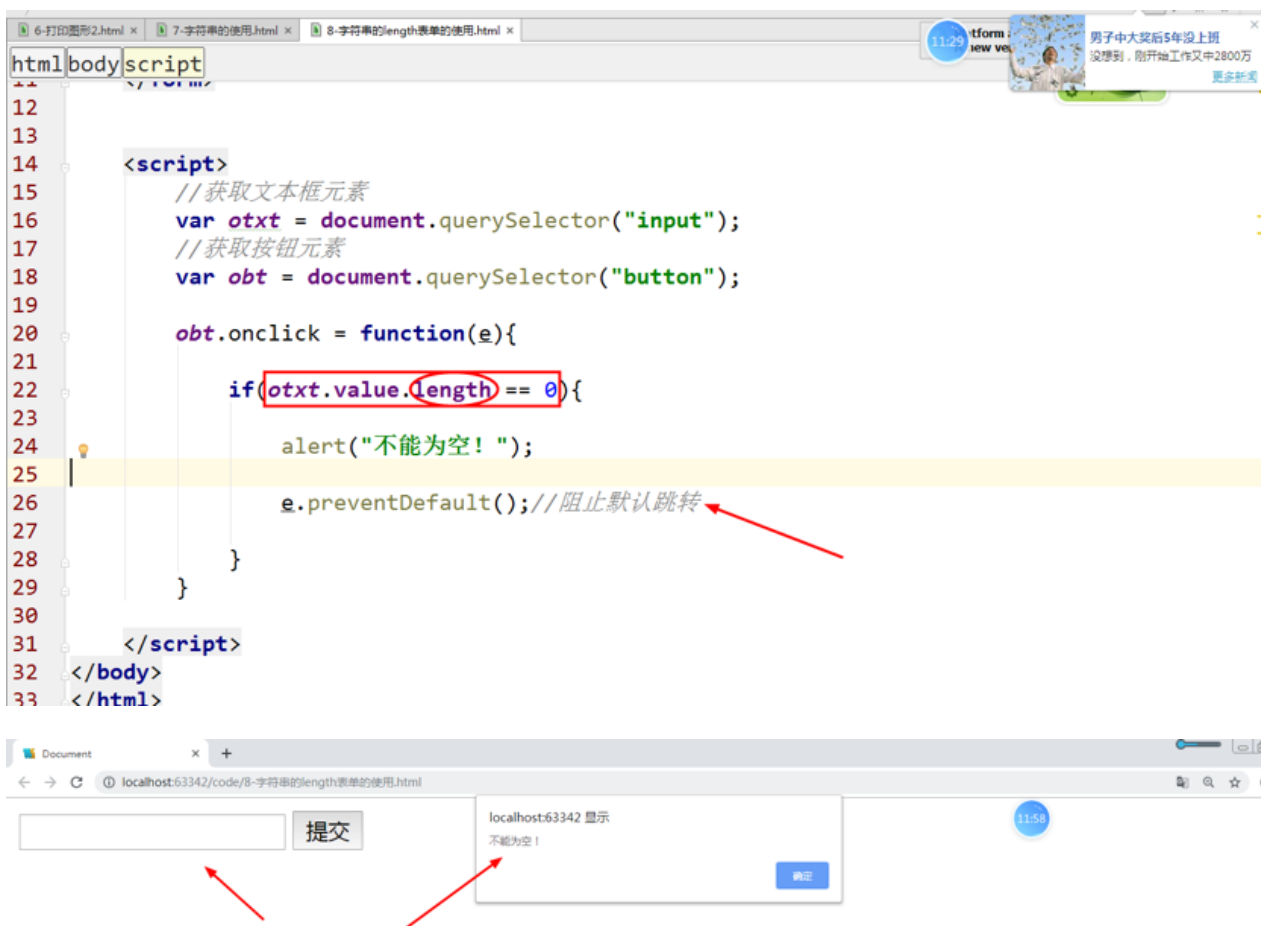
```
var x = "Hello World!";
```

字符串断行需要使用反斜杠()

```
var x = "Hello \ World!";
```

### 2.2. length属性

length:返回的是字符串的长度

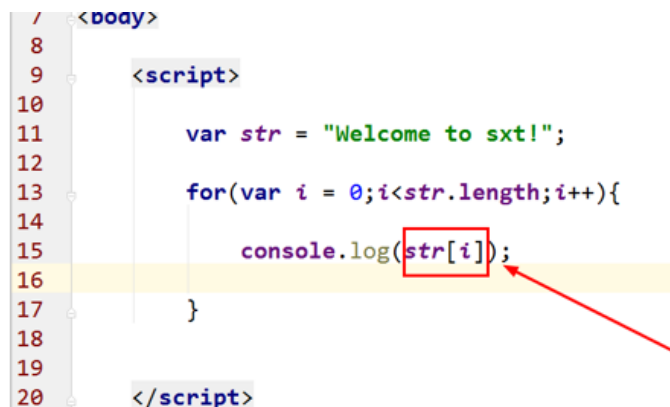


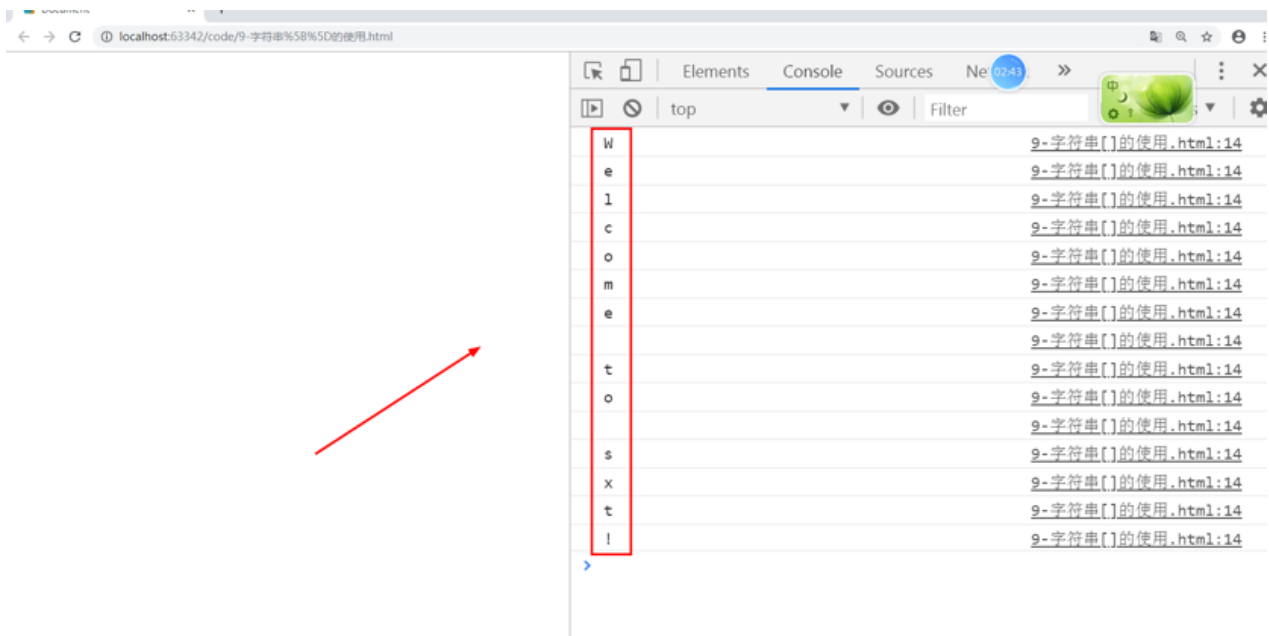
## 2.3. 字符索引

[ ]方法：在字符串后面接中括号，中括号中写数字。能够访问到字符串中的每个字符。

ps:索引一次只能索引一个字符，如果需要多个则需要用+连接符。

pss:索引从0开始，0表示第一个字符。





## 2.4. 获取指定位置字符

1.charAt(index):返回的是具体的字符

Index:就是字符串的位置（它是一个数字）

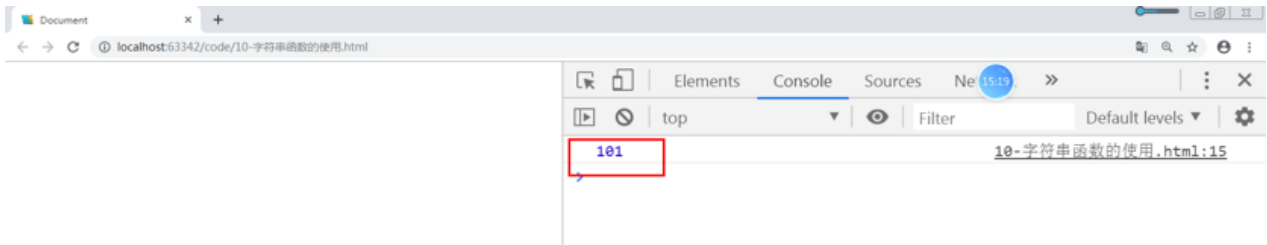
```
8
9   <script>
10
11       var str = "Welcome to sxt!";
12
13       console.log(str.charAt(2)); // 返回指定位置的字符
14
15   </script>
16 </body>
17 </html>
```

2.charCodeAt (index) 返回的是字符对应的Unicode编码  
(ascii编码值)

A:65 a:97 0:48

Index:就是字符串的位置（它是一个数字）

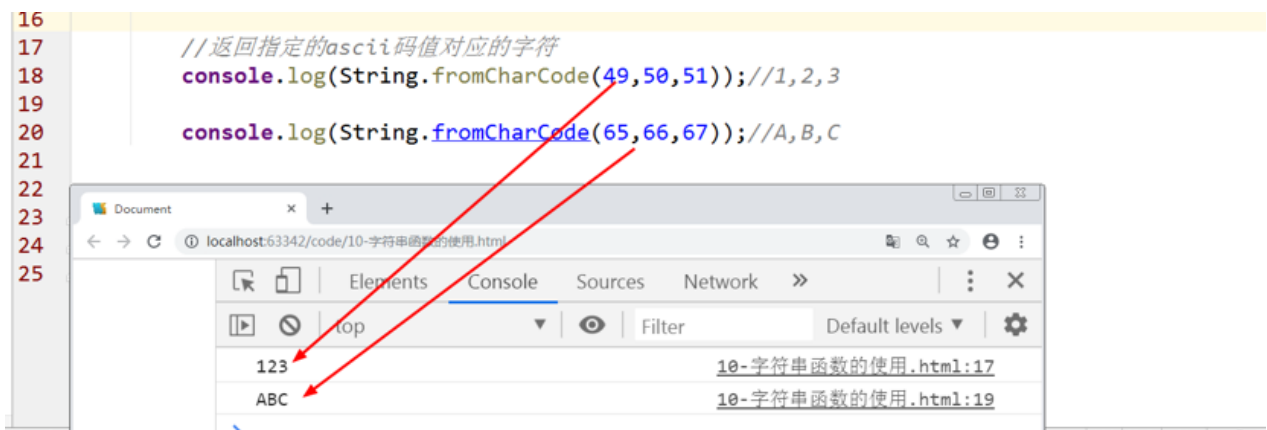
```
var str = "Welcome to sxt!";  
  
// console.log(str.charAt(2)); // 返回指定位置的字符  
  
console.log(str.charCodeAt(1)); // 返回指定位置的字符的ascii码值
```



索引从0开始

字符编码需要记得的两个。A—65，a—97。其他符号累加即可。

\3. fromCharCode将指定的数字(ascii码值)转为对应的字符



## 2.5. 字符串连接

**concat()**方法能够将两个字符串拼接起来，合成一个新的字符串。

可以认为concat和+作用相同。至少在现阶段我们可以认为他们是没有区别的

## 2.6. 模板字符串

模板字符串（template string）是增强版的字符串，用反引号（```）标识。

模板字符串中嵌入变量，需要将**变量名**写在`${}`之中。

## 3. 转义字符

常见转义字符

' 单引号

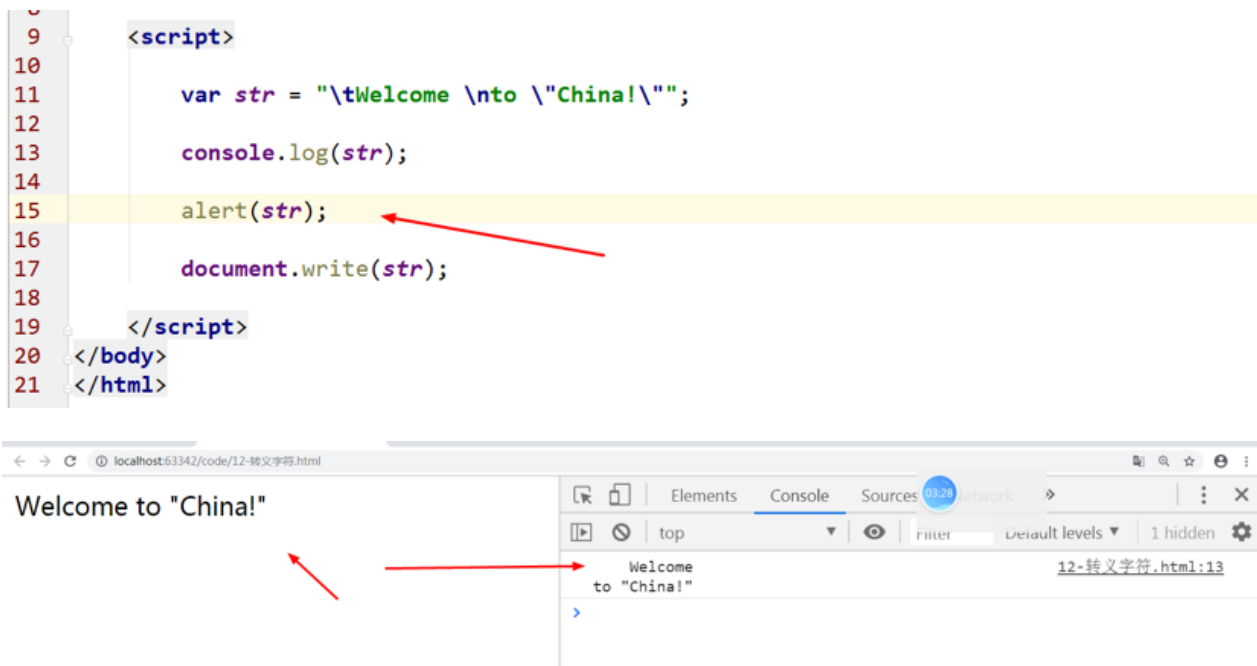
" 双引号

\ 反斜杠

\n 换行

\r 光标到首行

\t tab(制表符)



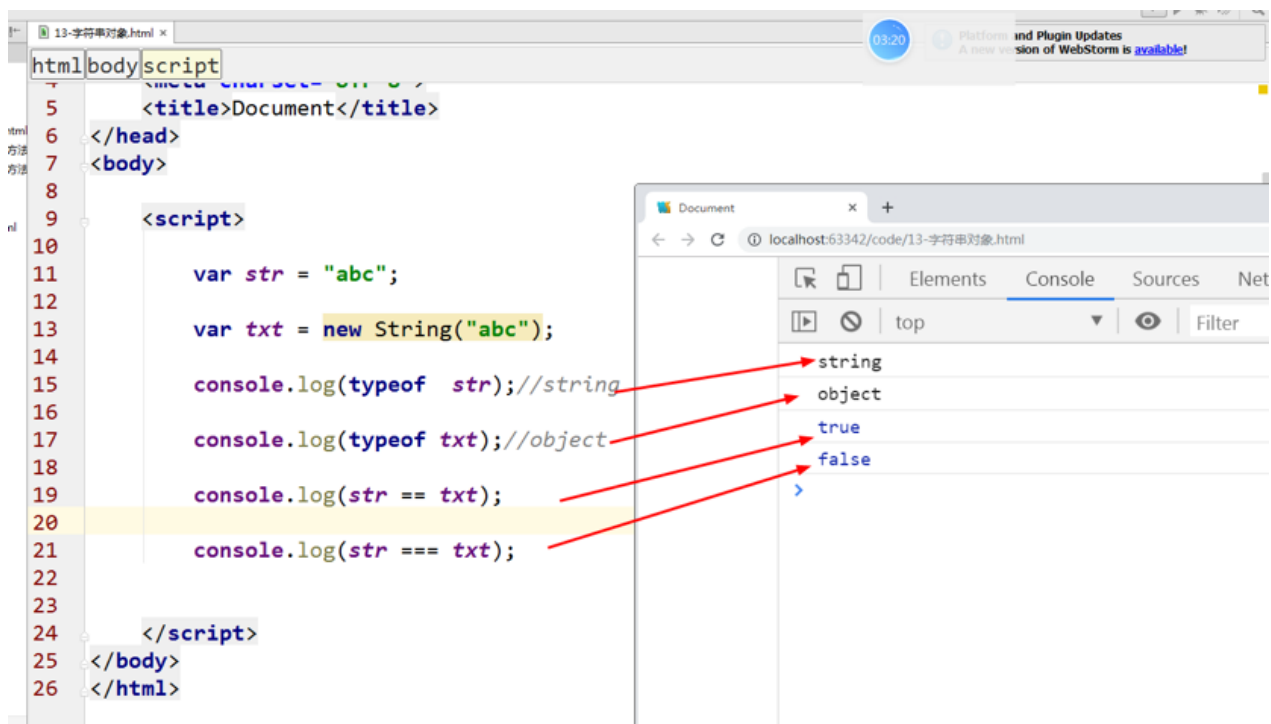
## 4. 字符串对象

javascript中有字符串类型string类型，我们也知道这种基本类型的变量的创建方式。

但javascript中还提供了另外一种字符串的声明方式，这种方式叫字符串对象。使用 `new` 关键字将字符串定义为一个对象

`New String();`





## 5. 字符串方法(重点)

### 5.1. 字符串方法

(2).charAt (number) : 返回当前指定位置的字符

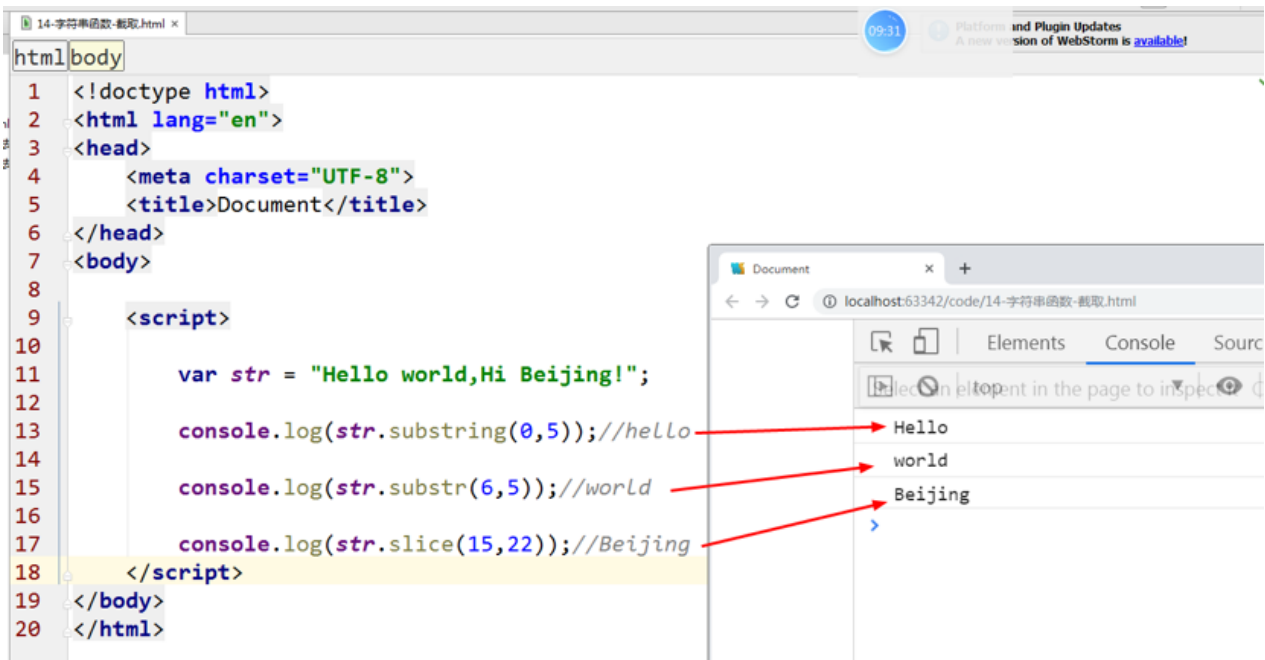
(3).charCodeAt(number): 返回当前指定位置的字符ascii码值

(4).concat:连接字符串

(5).substring(start,end):截取字符串(从哪里开始到哪里结束,end: 不包含end))

(6).substr(start,length):截取字符串 (从哪里开始取多长的字符)

(7).slice(start,end):截取字符串(end: 不包含end)

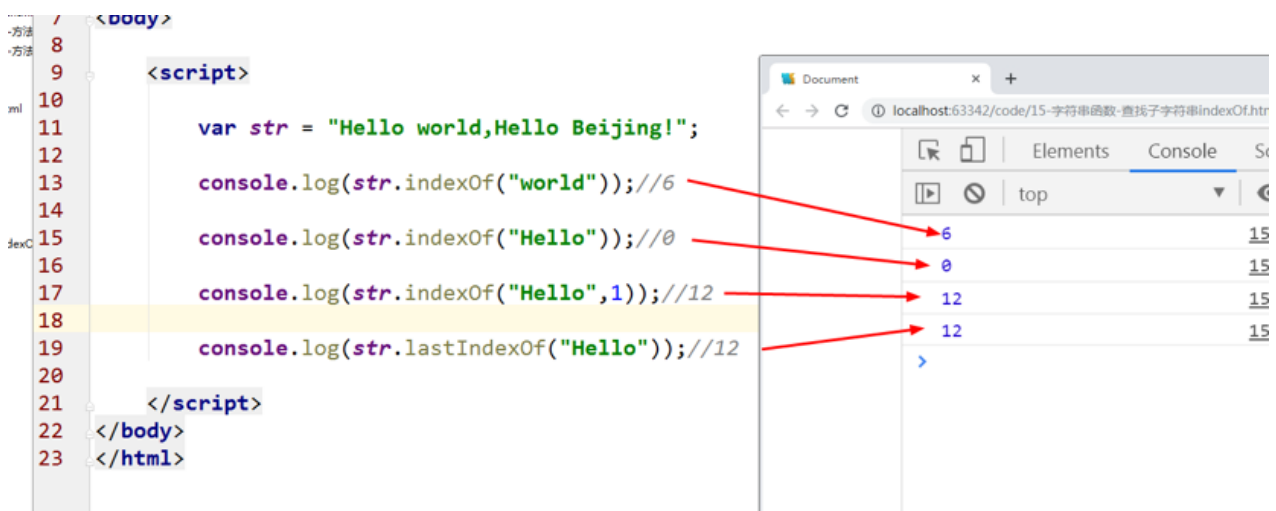


(8) .indexOf(str,offst): 返回当前查找字符串在整个字符串中的首次位置，如果没有返回-1

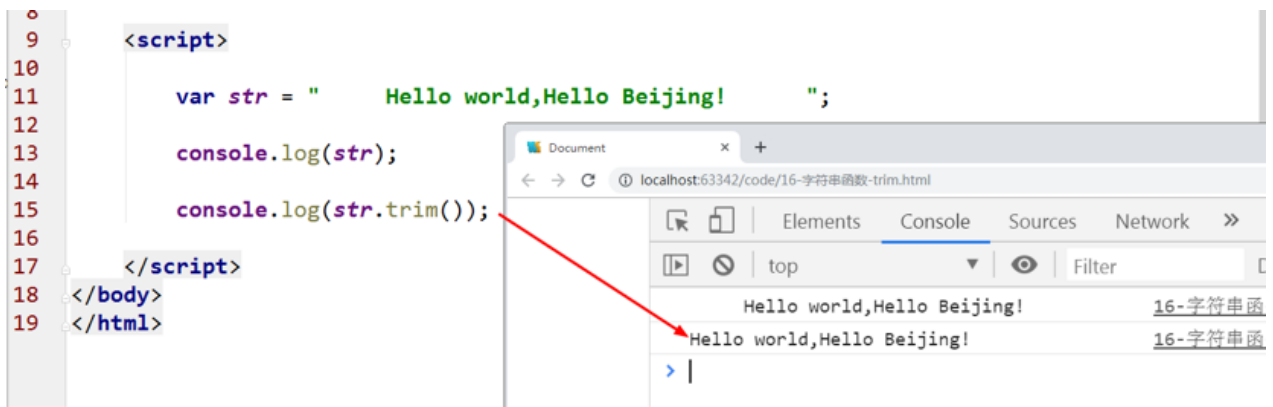
Str:字符串

Offset:从哪里开始查找

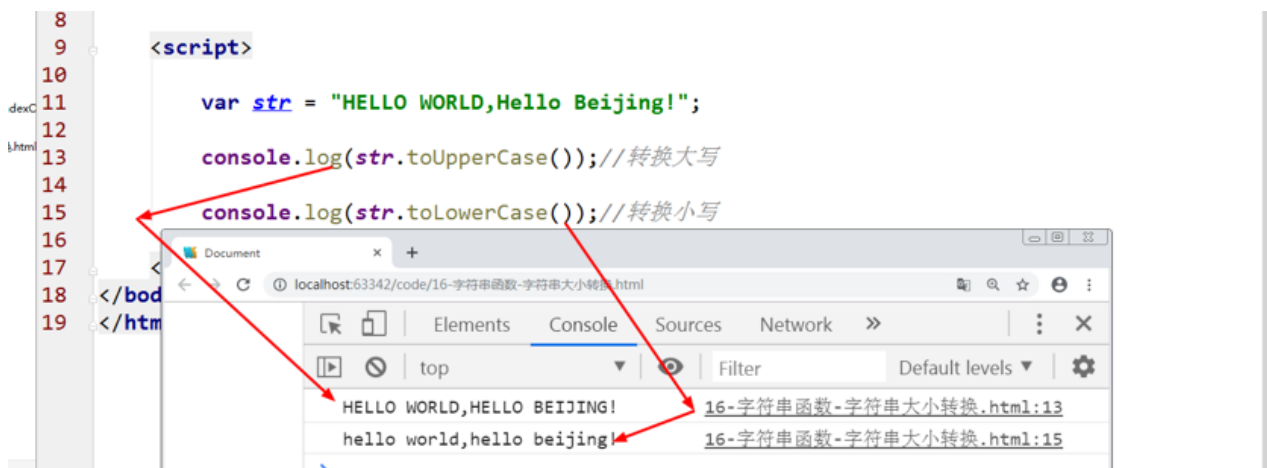
(9) .lastIndexOf:倒过来查找



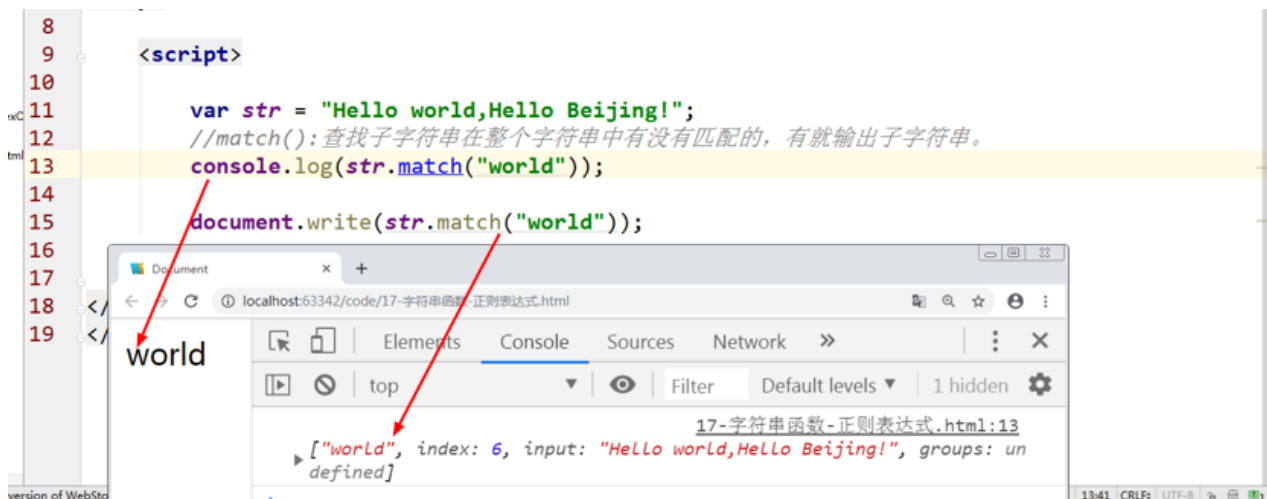
(10) .trim():去掉字符串两端的空格



## (11) .toUpperCase和toLowerCase:大小写转换



## (13).match:返回一个指定字符串的数组



## (14) .search:返回位置

## (15).replace:替换字符串

```

10  var str = "Hello world,Hello Beijing!";
11  //match(): 查找子字符串在整个字符串中有没有匹配的, 有就输出子字符串。
12  console.log(str.match("world"));
13  //
14  //
15  document.write(str.match("world"));
16  //search(): 查找字符串在整个字符串中出现的位置
17  console.log(str.search("world")); //6
18  //
19  document.write(str.search("world")); //6
20  //replace: 替换字符串
21  console.log(str.replace("Beijing", "shanghai"));
22
23
24

```

## (16) .split:字符串切割, 返回数组

```

8
9  <script>
10
11  var str = "Hello|world|Hello|Beijing!";
12
13  console.log(str.split(" "));
14
15  console.log(str.split(""));
16
17  console.log(str.split("|"));
18
19
20
21
22
23  </script>
24  </body>
25  </html>

```

Document

localhost:63342/code/18-字符串函数-split.html

Elements Console Sources Network

top

Filter

▶ ["Hello|world|Hello|Beijing!"] 18-字符!

▶ (26) ["H", "e", "l", "l", "o", "|", "w", "o", "r", "l", "d", "|", "H", "e", "l", "l", "o", "|", "B", "e", "i", "j", "i", "n", "g", "!"] 18-字符!

▶ (4) ["Hello", "world", "Hello", "Beijing!"] 18-字符!

## Es6新增的方法

- **includes()**: 返回布尔值, 表示是否找到了参数字符串。
- **startsWith()**: 返回布尔值, 表示参数字符串是否在原字符串的头部。
- **endsWith()**: 返回布尔值, 表示参数字符串是否在原字符串的尾部。
- 这三个方法都支持第二个参数, 表示开始搜索的位置。
- **repeat** 方法返回一个新字符串, 表示将原字符串重复 **n** 次。

ES2017 引入了字符串补全长度的功能。如果某个字符串不够指定长度，会在头部或尾部补全。`padStart()` 用于头部补全，`padEnd()` 用于尾部补全。

`padStart()` 和 `padEnd()` 一共接受两个参数，第一个参数是字符串补全生效的最大长度，第二个参数是用来补全的字符串。

## 5.2. 字符串Base64编码

**Base64**本身是一种加密方式，可以将任意字符转成可打印字符。

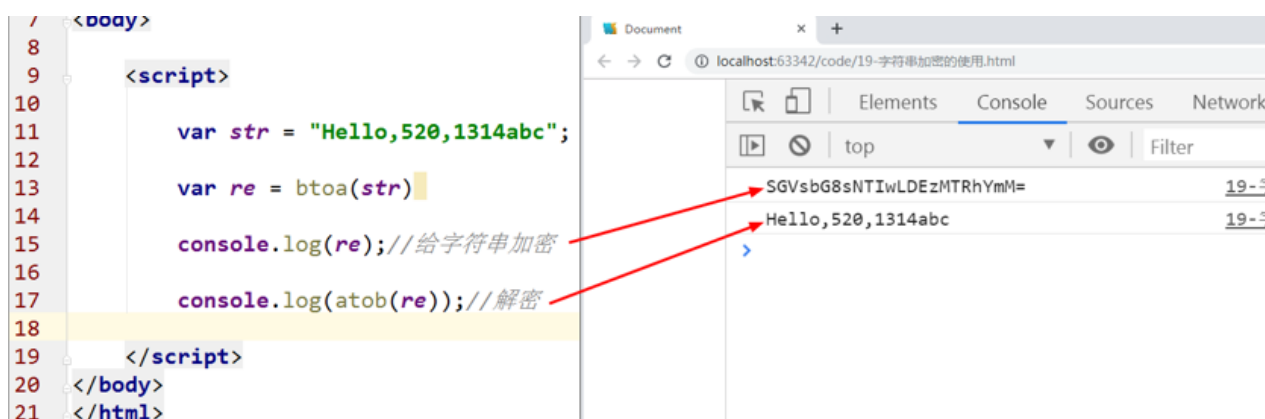
有时需要以文本格式传递二进制数据，那么也可以使用 Base64 编码。

而我们使用这种编码方法，主要不是为了加密，而是为了不出现特殊字符，简化程序的处理。

javascript中字符串提供了两个有关Base64编码的方法：

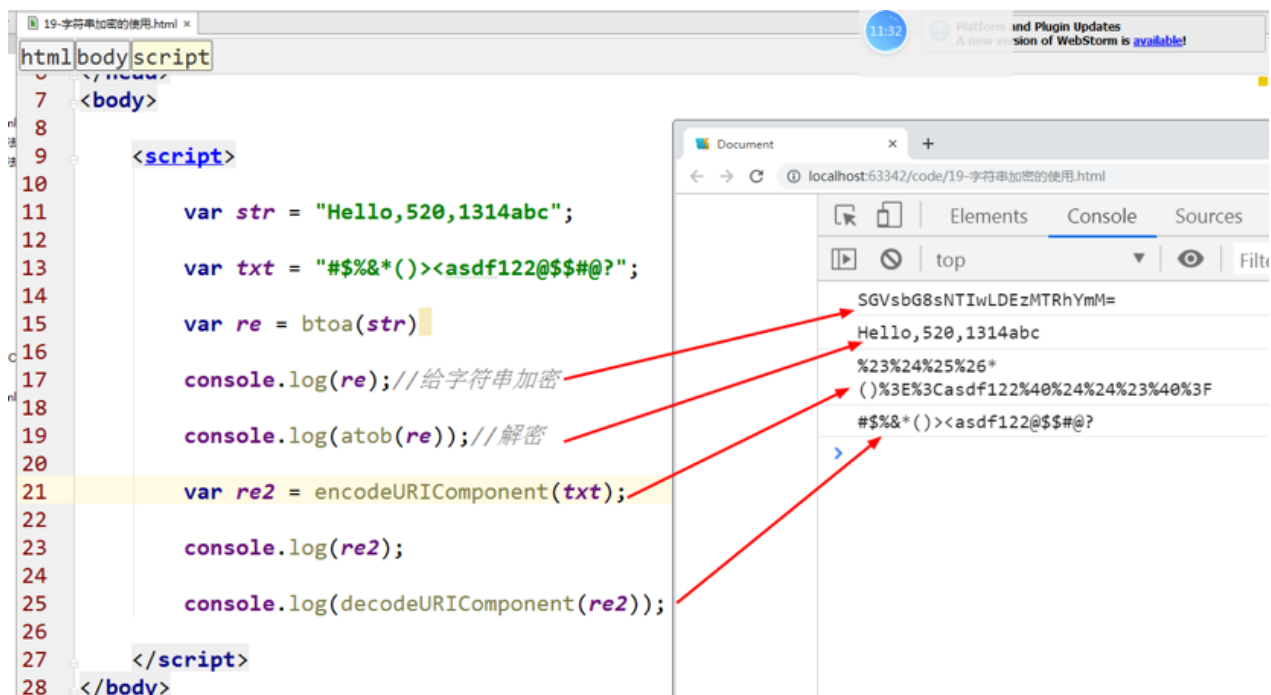
**btoa():** 字符串或二进制值转为Base64编码

**atob():** Base64编码转为原来的编码



**encodeURIComponent():** 要将非 ASCII 码字符转为 Base64 编码

**decodeURIComponent():** 将转码后的内容转为非ASCII内容



## 5.3. 作业1：统计字符串中字母出现的次数

“ababcdeeffhhljlkomp”,统计这个字符串中每个字母出现的次数。

利用字符串的函数：

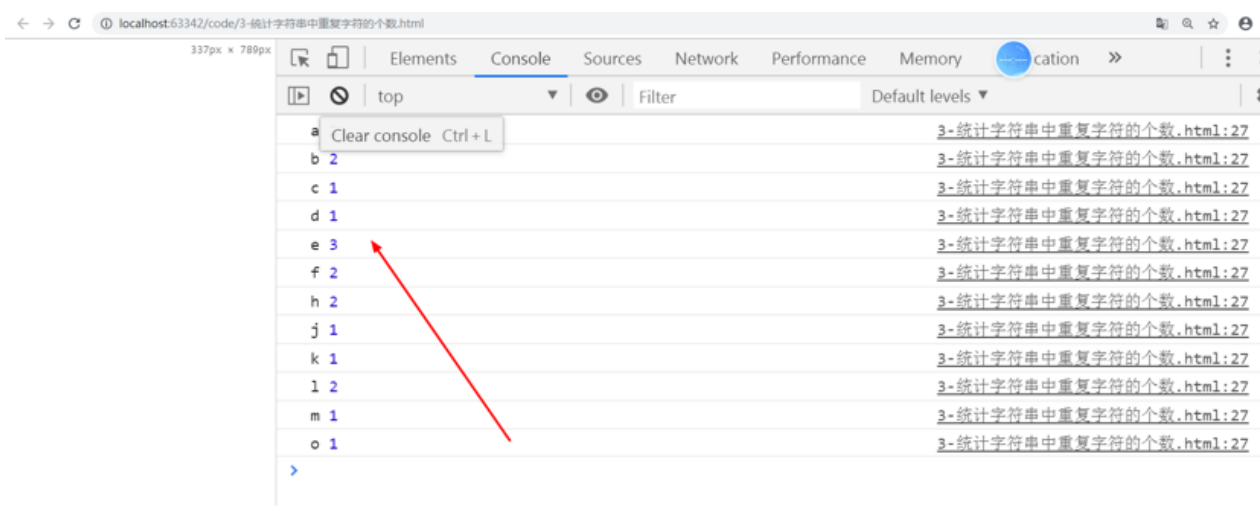
charCodeAt(i):返回指定位置的字符的ascii码值 a:97,A: 65

fromCharCode(97,98,99):返回指定ascii码值的所指定的字符

charAt():返回指定位置的字符

第一种方法：

```
htmlbodyscript
10     var str = "ababcdeeeffhhljlkom";
11
12     // console.log(String.fromCharCode(97,98,99));
13
14     // console.log(str.charCodeAt(1));
15     // ascii值的范围
16     for(var i = 65;i<122;i++){
17         // 字符累加的次数
18         var count = 0;
19         // 找到字符串中指定的ascii码值
20         for(var j = 0;j<str.length;j++){
21             // 如果外层循环的ascii码值等于字符串对于字符的ascii值就累加
22             if(i == str.charCodeAt(j)){
23                 count++; // 累加重复的字符
24             }
25         }
26     }
27     if(count){
28         console.log(String.fromCharCode(i),count);
29     }
30 }
31
32 </script>
```



方法2：创建一个新的字符串，这个新的字符串是不重复的每个字符，然后和老的字符串中有重复的字符进行比较，如果相等就累加。

indexOf():查找子字符串在整个字符串中首次出现的位置。（从前往后）

lastIndexOf():查找子字符串在整个字符串中最后一次出现的位置。（从后往前）

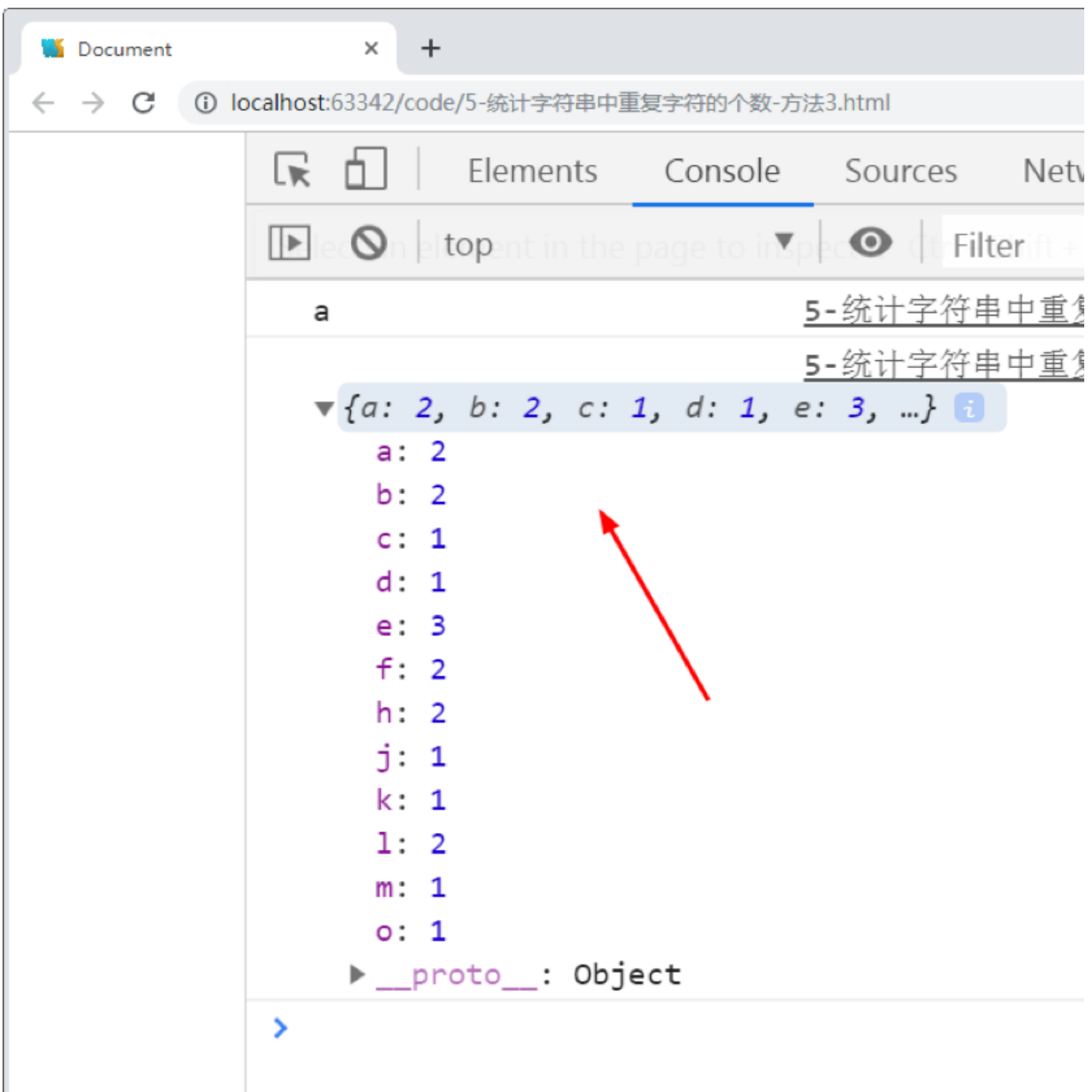
方法3：利用对象的方式进行字符串的累加

```
// 第二步: 新老字符串进行比较
//
console.log(newstr);
for(var j = 0; j < newstr.length; j++){

    var count = 0; // 字符累加的次数

    for(var k = 0; k < str.length; k++){
        // 如果新的字符串中字符等于老的字符串中字符就累加
        if(newstr[j] == str[k]){
            count++;
        }
    }

    console.log(newstr[j], count);
}
```



## 6. 字符串实例



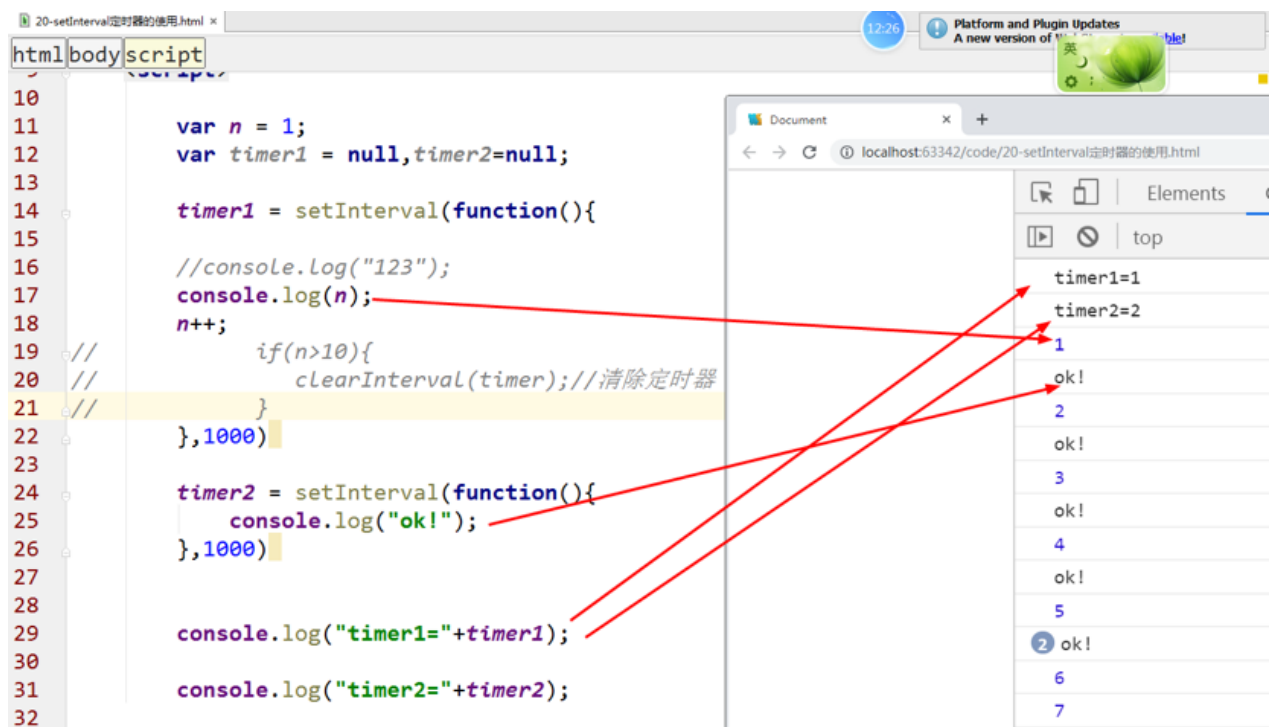
## 6.1. 定时器的简单使用

setInterval():定时器,

按照指定的时间, 重复的执行函数表达式

setInterval(function(){},毫秒数)

## 6.2. 实例：打字机效果（动画）



setTimeout():延时器

方法1：通过substring()或者substr()或者是slice()这三个函数截取字符串

```
21-打字机效果.html x
html body script
7 <body>
8 <div></div>
9 <script>
10
11     var str = "下班了，爸爸陆陆续续的回来了！";
12
13     var odiv = document.querySelector("div");
14
15     var n = 0;
16
17     setInterval(function(){
18         // 截取字符串
19         odiv.innerHTML = str.substr(0,n)
20
21         n++;
22
23         if(n>str.length){
24             n = 0;
25         }
26
27     },1000)
28
29 </script>
```

Document x +

localhost:63342/code/21-打字机效果.html

下班了，爸爸陆陆续续的回来了

方法2:通过数组的方式来访问字符串中字符

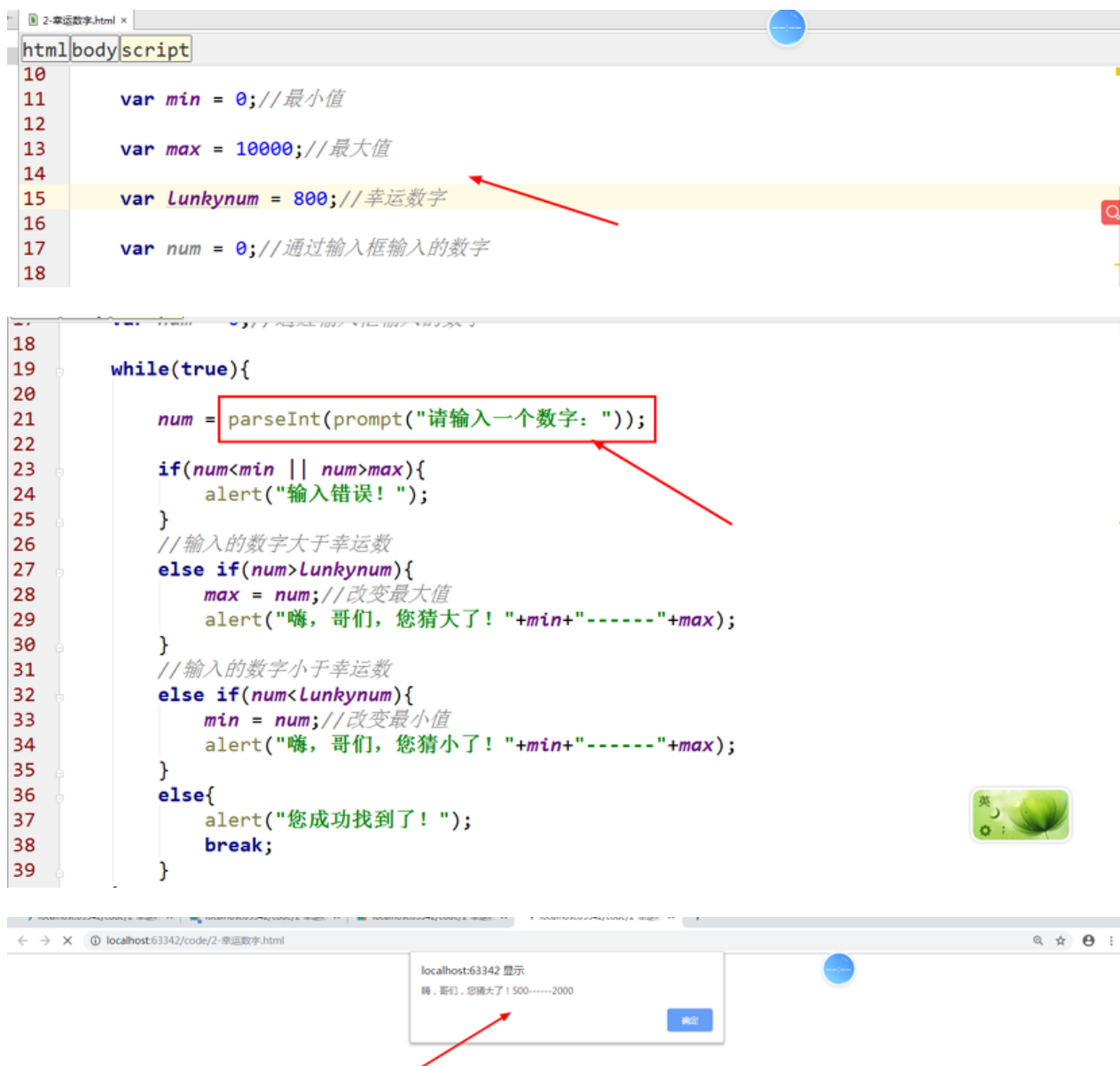
```
1-打字机的效果.html x
html body script
12 //
13 // },1000)
14 //方式2
15 // function change(){
16 //     document.write("abc");
17 // }
18 // setInterval("change()",100)
19 var odiv = document.querySelector("div");//获取div元素
20
21 var str = "生当作人杰，死亦为鬼雄";
22
23 var i = 0;
24 setInterval(function(){
25     odiv.innerHTML += str[i]
26     i++;
27     if(i>str.length){
28         i = 0;
29         odiv.innerHTML = "";
30     }
31 },300)
32
```

localhost:63342/code/1-打字机 x +

localhost:63342/code/1-打字机的效果.html

生当作人杰，死亦为鬼

## 6.3. 实例：幸运数字（0-10000）,800



The screenshot displays a web browser window with a code editor for a file named '2-幸运数字.html'. The code defines a variable `lunkynum` as 800 and a `while` loop that prompts the user to enter a number. If the input is outside the range of 0 to 10000, an alert is shown. If the input is greater than 800, the maximum value is updated and an alert is shown. If the input is less than 800, the minimum value is updated and an alert is shown. If the input is exactly 800, a success alert is shown and the loop breaks. A red arrow points to the line `var lunkynum = 800;` in the code editor. Another red arrow points to the alert message '嗨，哥们，您猜大了！500-----2000' in the browser's confirmation dialog.

```
10
11     var min = 0; // 最小值
12
13     var max = 10000; // 最大值
14
15     var lunkynum = 800; // 幸运数字
16
17     var num = 0; // 通过输入框输入的数字
18
19     while(true){
20
21         num = parseInt(prompt("请输入一个数字: "));
22
23         if(num < min || num > max){
24             alert("输入错误!");
25         }
26         // 输入的数字大于幸运数
27         else if(num > lunkynum){
28             max = num; // 改变最大值
29             alert("嗨，哥们，您猜大了！"+min+"-----"+max);
30         }
31         // 输入的数字小于幸运数
32         else if(num < lunkynum){
33             min = num; // 改变最小值
34             alert("嗨，哥们，您猜小了！"+min+"-----"+max);
35         }
36         else{
37             alert("您成功找到了!");
38             break;
39         }
40     }
```

localhost:63342 显示  
嗨，哥们，您猜大了！500-----2000

## 6.4. 实例：金字塔

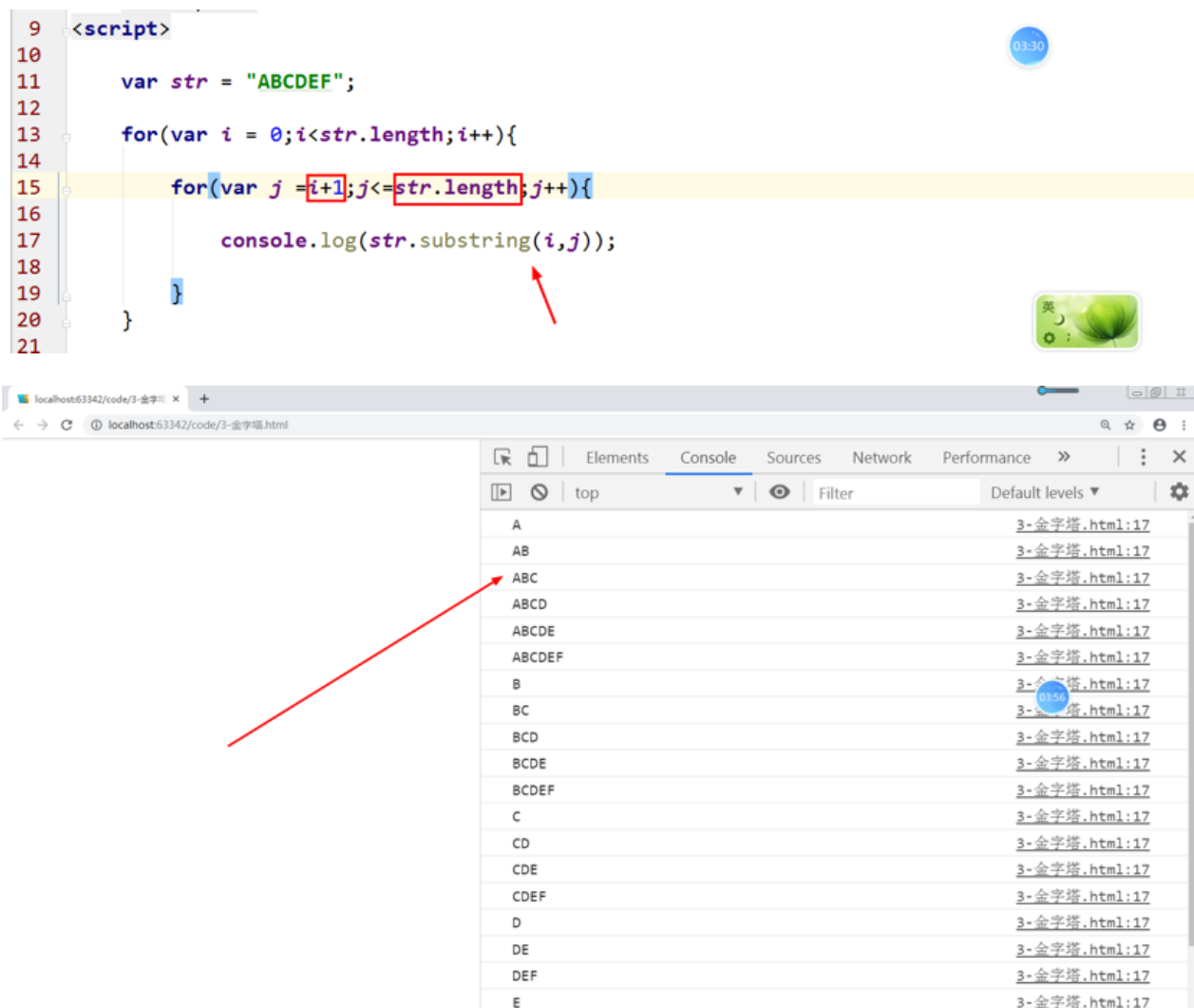
Str = "abcdefg";

A

Ab

Abc

Abcd



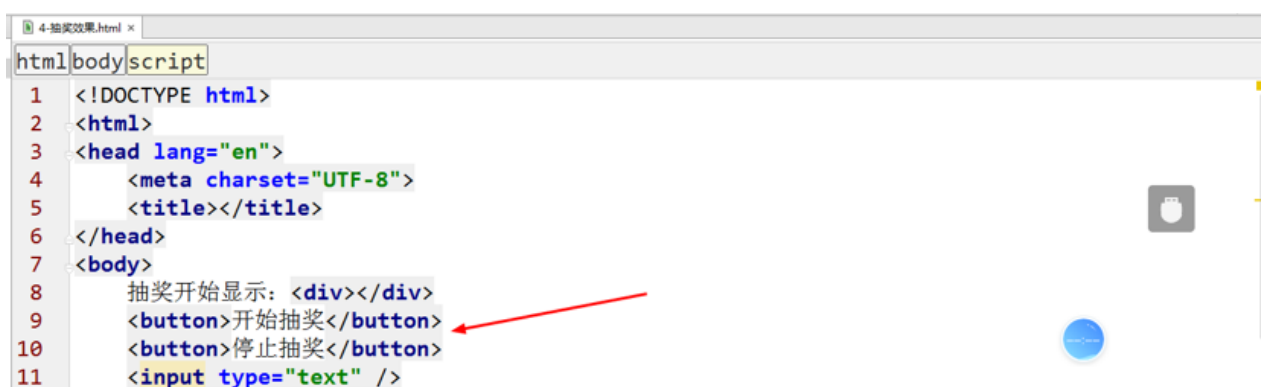
## 6.5. 实例：抽奖效果(动画)

分析一下：How to create your code?

setInterval():

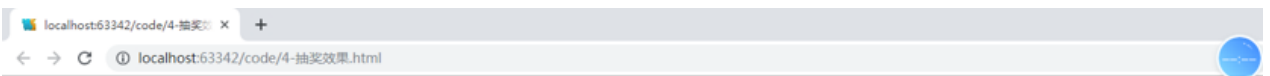
奖品存放在数组中，随机抽奖 (Math.random())

将结果存放到文本框中



```
4-抽奖效果.html x
html body script
12 <script>
13 // 获取div元素
14 var odiv = document.querySelector("div");
15 // 获取按钮元素
16 var obt = document.querySelectorAll("button");
17 // 获取文本框元素
18 var oinput = document.querySelector("input");
19 // 定义一个数组
20 var prize = ["一等奖-p30pro", "二等奖-p20", "三等奖-oppo-r17", "四等奖-小米", "五等奖-水杯一个"];
21 // 定时器对象
22 var timer = null;
23
```

```
4-抽奖效果.html x
html body script
24 // 开始抽奖按钮单击事件
25 obt[0].onclick = function(){
26     clearInterval(timer); // 先停止动画
27     timer = setInterval(function(){
28         odiv.innerHTML = prize[rnd()]; // 随机显示抽奖结果在div元素上
29     }, 90);
30 }
31 // 随机显示数组的下标
32 function rnd(){
33     return Math.floor(Math.random()*prize.length);
34 }
35 // 停止抽奖按钮单击事件
36 obt[1].onclick = function(){
37     clearInterval(timer);
38     oinput.value = odiv.innerHTML; // 将抽奖结果 (odiv.innerHTML) 存储到文本框中
39 }
```



抽奖开始显示：

二等奖-p20

开始抽奖

停止抽奖

二等奖-p20