String 字符串

语法

字符串型可以是引号中的任意文本，其语法为：双引号 `""` 或者单引号 `''`。

来看个示例。下面的这些，都是字符串：

```
  var a = "time.geekbang.org";
  var b = "极客时间";
  var c = "网络安全";
  var d = 'JavaScript';
  var e = "";     //空字符串

  var f = haha; // 没使用引号，到这里会直接报错。因为会被认为是js代码，但是之前并没有定义haha。

  console.log(typeof a);
  console.log(typeof b);
  console.log(typeof c);
  console.log(typeof d);
  console.log(typeof e);
```

控制台输出如下：

```
  string
  string
  string
  string
  string
```

引号的注意事项

1、单引号和双引号不能混用。比如下面这样写是不可以的：

```
var str = 'hello";  // 报错：Uncaught SyntaxError: Invalid or unexpected token
```

2、同类引号不能嵌套：双引号里不能再放双引号，单引号里不能再放单引号，但是可以用\实现转义。

3、单引号里可以嵌套双引号；双引号里可以嵌套单引号。

转义字符

在字符串中我们可以使用`\`作为转义字符，当表示一些特殊符号时可以使用`\`进行转义。

- `\"` 表示 `"` 双引号
- `\'` 表示 `'` 单引号
- `\\` 表示`\`
- `\r` 表示回车
- `\n` 表示换行。n 的意思是 newline。

- `\t` 表示缩进。t 的意思是 tab。
- `\b` 表示空格。b 的意思是 blank。

举例：

```
    var str1 = "我说:\"今天\t天气真不错! \"";
    var str2 = "\\\\\\\";

    console.log(str1);
    console.log(str2);
```

上方代码的打印结果：

```
我说:"今天  天气真不错! "
\\\
```

获取字符串的长度

字符串是由若干个字符组成的，这些字符的数量就是字符串的长度。我们可以通过字符串的 length 属性可以获取整个字符串的长度。

代码举例：

```
var str1 = '极客时间';
var str2 = 'time.geekbang.org';
var str3 = 'geektime';
var str4 = 'geektime, keep moving!';

console.log(str1.length); //
console.log(str2.length); //
console.log(str3.length); //
console.log(str4.length); //
```

由此可见，字符串的 length 属性，在判断字符串的长度时，会认为：

- 一个中文算一个字符，一个英文算一个字符
- 一个标点符号（包括中文标点、英文标点）算一个字符
- 一个空格算一个字符

字符串拼接

多个字符串之间可以使用加号 `+` 进行拼接。

拼接语法：

```
字符串 + 任意数据类型 = 拼接之后的新字符串;
```

拼接规则：拼接前，会把与字符串相加的这个数据类型转成字符串，然后再拼接成一个新的字符串。

代码举例：（字符串与六大数据类型相加）

```
var str1 = '极客时间';
var str2 = '极客时间' + 666;
var str3 = '极客时间' + true;
```

```
var str4 = '极客时间' + null;
var str5 = '极客时间' + undefined;

var obj = { name: '极客时间', age: 28 };
var str6 = '极客时间' + obj;

console.log(str1);
console.log(str2);
console.log(str3);
console.log(str4);
console.log(str5);
console.log(str6);
```

打印结果：

```
极客时间

极客时间666

极客时间true

极客时间null

极客时间undefined

极客时间[object Object]
```

字符串的不可变性

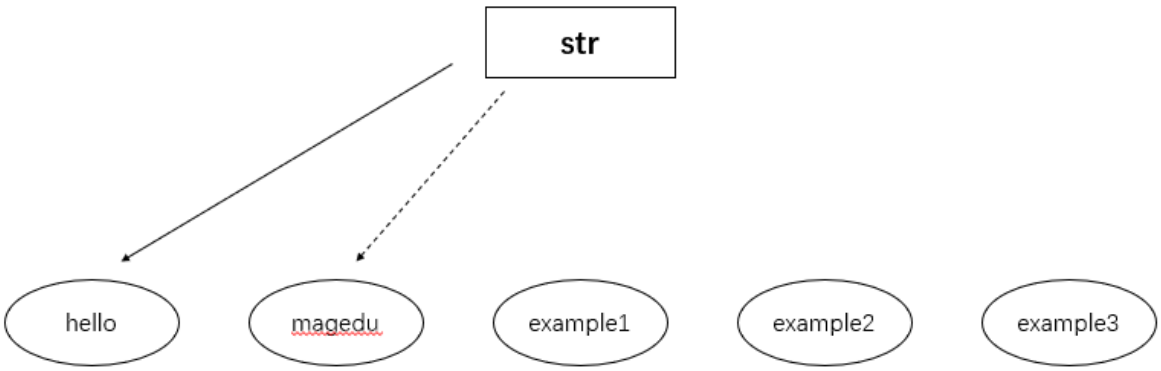
字符串里面的值不可被改变。虽然看上去可以改变内容，但其实是地址变了，内存中新开辟了一个内存空间。

代码举例：

```
var str = 'hello';

str = 'geektime';
```

比如上面的代码，当重新给变量 str 赋值时，常量`hello`不会被修改，依然保存在内存中；str 会改为指向`geektime`。



模板字符串（模板字面量）

ES6中引入了**模板字符串**，让我们省去了字符串拼接的烦恼。下面一起来看看它的特性。

模板字符串中插入变量

以前，让字符串进行拼接的时候，是这样做的：（传统写法的字符串拼接）

```
var name = 'geektime';
var age = '26';
console.log('name:' + name + ',age:' + age); //传统写法
```

这种写法，比较繁琐，而且容易出错。

现在，有了 ES6 语法，字符串拼接可以这样写：

```
var name = 'geektime';
var age = '26';

console.log('我是' + name + ',age:' + age); //传统写法

console.log(`我是${name},age:${age}`); //ES6 写法。注意语法格式
```

注意，上方代码中，倒数第二行用的符号是单引号，最后一行用的符号是反引号（在 tab 键的上方）。

参考链接：

- ES6 的 rest 参数和扩展运算符

模板字符串中插入表达式

以前，在字符串中插入表达式的写法必须是这样的：

```
const a = 5;
const b = 10;
console.log('this is ' + (a + b) + ' and\nnot ' + (2 * a + b) + '.');
```

现在，通过模板字符串，我们可以使用一种更优雅的方式来表示：

```
const a = 5;
const b = 10;

// 下面这行代码，故意做了换行。
console.log(`this is ${a + b} and
not ${2 * a + b}.`);
```

打印结果：

```
this is 15 and
not 20.
```

模板字符串中可以换行

因为模板字符串支持换行，所以可以让代码写得非常美观。

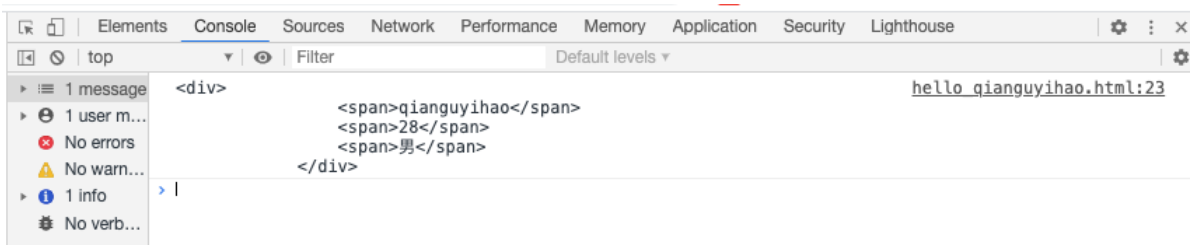
代码举例：

```
const result = {
  name: 'geektime',
  age: 28,
  sex: '男',
};

// 模板字符串支持换行
const html = `<div>
  <span>${result.name}</span>
  <span>${result.age}</span>
  <span>${result.sex}</span>
</div>`;

console.log(html); // 打印结果也会换行
```

打印结果：



模板字符串中可以调用函数

模板字符串中可以调用函数。字符串中调用函数的位置，将会显示函数执行后的返回值。

举例：

```
function getName() {
  return 'geektime';
}

console.log(`www.${getName()}.com`); // 打印结果：www.geektime.com
```

模板字符串支持嵌套使用

```
const nameList = ['极客时间', '渗透测试', '网络安全'];

function myTemplate() {
  // join('') 的意思是，把数组里的内容合并成一个字符串
  return `<ul>
  ${nameList
    .map((item) => `<li>${item}</li>`)
    .join('')}
  </ul>`;
}
document.body.innerHTML = myTemplate();
```

布尔值：Boolean

布尔型有两个值：true 和 false。主要用来做逻辑判断： true 表示真，false 表示假。

布尔值直接使用就可以了，千万不要加上引号。

代码：

```
var a = true;
console.log(typeof a);
```

控制台输出结果：

```
boolean
```

布尔型和数字型相加时， true 按 1 来算 ， false 按 0 来算。

```
var str1 = 1;
var str2 = str1 + true;

console.log(str1);
console.log(str2);
```