4、字符串的扩展

- 模板字符串
- 模板字符串实例
- 标签模板

① 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。
- 它可以当作普通字符串使用，也可以用来定义多行字符串，或者在字符串中嵌入变量
- 如果在模板字符串中需要使用反引号，则前面要用反斜杠转义。
- 如果使用模板字符串表示多行字符串，所有的空格和缩进都会被保留在输出之中。
- 模板字符串之中可以放入js表达式、对象属性、还能调用函数。
- 注意：如果模板字符串中的变量没有声明，将报错。

```
//模板字符串的使用
let url = "http://www.baidu.com";
let title = "百度";

//传统的子串拼装
console.log('<a href="'+url+'">'+title+'</a>');
//输出：<a href="http://www.baidu.com">百度</a>

//使用模板子串
console.log(`<a href="${url}">${title}</a>`);
//输出：<a href="http://www.baidu.com">百度</a>

//在模板子串中输出反引号，可使用反斜线转义
console.log(`Hello \`ZhangSan\`!~`);
//输出：

//传统定义多行子串格式
let tpl1 = "<ul>" +
    "<li>北京</li>" +
    "<li>上海</li>" +
    "<li>广州</li>" +
    "</ul>";
console.log(tpl1);
//输出：<ul><li>北京</li><li>上海</li><li>广州</li></ul>

//使用模板子串定义多行子串格式，而且所有的空格和缩进都会被保留
let tpl2 = `
    <ul>
        <li>北京</li>
        <li>上海</li>
```

```
        <li>广州</li>
    </ul>`;
console.log(tpl2);
/* 输出:
    <ul>
        <li>北京</li>
        <li>上海</li>
        <li>广州</li>
    </ul>
*/

//模板子串中支持使用对象属性、表达式运算及函数调用等操作
let stu = {name:"张三",age:18};
console.log(`我叫${stu.name},今年${stu.age}岁，就读于清华大一，等毕业我就${stu.age+4}岁了！`);
//输出：我叫张三,今年18岁，就读于清华大一，等毕业我就22岁了!
```

② 模板字符串实例：

```
//将下面的数据输出到模板ul标签中
let data = [
    {name:"张三",sex:1},
    {name:"李四",sex:0},
    {name:"王五",sex:1},
    {name:"赵六",sex:0},
];

let tpl = `<ul>${data.map(stu => `<li>${stu.name}</li>`).join("")}</ul>`;
console.log(tpl);
//输出结果：<ul><li>张三</li><li>李四</li><li>王五</li><li>赵六</li></ul>

//也可以写出如下格式
let tpl2 = `<ul>
    ${data.map(stu => `
        <li>
            ${stu.name}
            ${stu.sex==1?"男":"女"}
        </li>
    `).join("")}
</ul>`;
document.body.innerHTML = tpl2; //输出页面body中
```

③ 模板标签

- 模板字符串的功能，不仅仅是上面这些。
- 它可以紧跟在一个函数名后面，该函数将被调用来处理这个模板字符串。这被称为“标签模板”功能（tagged template）。

```
alert("hello1");
//等同于
alert`hello2`;
```

- 标签模板其实不是模板，而是函数调用的一种特殊形式。
- “标签”指的就是函数，紧跟在后面的模板字符串就是它的参数。
- 如果模板字符中有变量，就不是简单的调用了，而是会将模板字符串先处理成多个参数，再调用函数。

```
let a = 5;
let b = 10;

tag`Hello ${ a + b } world ${ a * b }`;
// 等同于
//tag(['Hello ', ' world ', ''], 15, 50);

//function tag(tpldata, value1, value2){
    // ...
//}
// 等同于
function tag(tpldata, ...values){
    console.log(tpldata);//["Hello ", " world ", ""]
    console.log(values); //[15, 50]
    console.log(arguments);
    //[["Hello ", " world ", ""],15,50]
}
```

- “标签模板”的一个重要应用，就是过滤 HTML 字符串，防止用户输入恶意内容。

```
//定义一个安全处理html标签函数
function SaferHTML(tpldata, ...values){
    let s = tpldata[0];
    for (let i = 0; i < values.length; i++){
        let arg = String(values[i]);
        //在替换中转义特殊字符
        s += arg.replace(/&/g, "&amp;")
                .replace(/</g, "&lt;")
                .replace(/>/g, "&gt;");

        //不要转义模板中的特殊字符。
        s += tpldata[i+1];
    }
    return s;
}
//测试
let content = '<script>alert("Hello")<\/script>'; // 恶意代码
let message = SaferHTML`<p>${content} 已向您发送消息.</p>`; //使用标签模板
console.log(message);
```

```
//输出：<p>&lt;script&gt;alert("Hello")&lt;/script&gt; 已向您发送消息.</p>
```