ES6的Unicode扩展

1、字符的unicode表示法

js允许采用\uxxxx形式表示一个字符，其中xxxx表示字符的unicode码点。只表示\u0000~\uFFFF之间。

若超出这个范围，则必须使用双字节的形式表示，ES6对其进行改进，允许使用大括号{码点}解读。

2、codePointAt()

js内部，字符以utf-16的格式存储，每个字符固定为2个字节，对于那些需要4个字节存储的字符（unicode码点大于0xFFFF），js认为是两个字符。

charCodeAt() //只能返回两个字节的值；对于读取4个字节的字符，如汉字等，ES6提供codePointAt()方法，能正确处理4个字节存储的字符。

codePointAt方法是测试一个字符由两个字节还是四个字节组成：

function is32Bit(c){

return c.codePointAt(0)>0xFFFF;

}

for...of循环可以正确识别32位的utf-16字符：

var str= “hello”;

for(let ch of str){

ch.codePointAt(0).toString(16); // 字符在字符串中是从零开始，若想要16进制可使用toString

}

3、String.fromCodePoint()

ES5提供String.fromCodeAt()方法，用于从码点返回对应的字符串，但是该方法不能识别32位的utf-16。

ES6提供String.fromCodePoint()方法，可以识别0xFFFF字符。

注意：fromCodePoint方法定义在String对象上，而codePointAt方法定义在字符串的实例对象上。

4、字符串的遍历器接口

ES6为字符串添加了遍历器接口，使得字符串可以被for...of 循环遍历

for(let c of str){

console.log(c);

}

除了遍历字符串，这个遍历器还有最大的优点就是可以识别0xFFFF的码点，传统的for循环不能。

eg. var text = String.fromCodePoint(0x20BB7);

for(let i of text){ console.log(i); }// 若for循环，则text[i]无法得到输出值。

5、at()

ES5对字符串对象提供charAt()方法，返回字符串给定位置的字符，该方法只能识别utf-16编码的字符。

ES6提案：at() 识别unicode>0xFFFF的字符，该方法可以通过垫片库实现。

6、includes()、startsWith()、endsWith()

ES5提供indexOf，可以用来确定一个字符串是否包含在另一个字符中。

ES6提供三种新方法：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头部；

endWith() 返回布尔值，表示参数字符串是否在原字符串尾部；

注意：使用第二个参数n时，endsWith的行为与其他两个方法不同。它针对前n个字符，而其他两个方法针对从第n个位置直到字符串结束。

7、padStart()、padEnd()

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。均接受两个参数：

第一个参数用来指定最小长度，第二个参数是用来补全的字符串，默认为空格。

‘x’.padStart(5, ‘ab’); //ababx ‘x’.padEnd(5, ‘ab’); //xabab ‘x’.padEnd(4, ‘ab’); //xaba

8、模板字符串

模板字符串是用反引号(｀ )标识。可当作普通字符串使用，也可以用来定义多行字符串，或在字符串中嵌入变量。

对比传统js语言：

$(‘#res’).append(‘These are <b>’+basket.count + ‘</b>’+’items in your basket.’);

$(‘#res’).append(`These are <b> ${basket.count}</b>items in your basket.`);