STRING METODLARI

1) **Concat :** İki ya da daha fazla diziyi birleştirmek için kullanılır.

Kullanımı:

aylar = [“Ocak”,“Subat”,“Mart”,“Nisan”,“Mayis”,“Haziran”]
mevsimler = [“Ilkbahar”,“Yaz”,“Sonbahar”,“Kıs”]
yil = aylar.concat(mevsimler);
yil = [“Ocak”, “Subat” , “Mart” , “Nisan” , “Mayis” , “Haziran” , “Ilkbahar” , “Yaz” , “Sonbahar” , “Kıs”]

2) **EndsWith :** Yöntem dönen, belirli bir karakter dizesinin bir dizge uçları belirleyen “true” veya “false” olarak değerlendirir.

Kullanımı:

```
const str1 = 'Cats are the best!';

console.log(str1.endsWith('best', 17));
// expected output: true
```

3) **StartsWith :** Yöntem, bir dizi geri belirtilen bir dize karakterleriyle başlayan belirler “true” ya da “false” olarak değerlendirir.

Kullanımı:

```
const str1 = 'Saturday night plans';

console.log(str1.startsWith('Sat'));
// expected output: true
```

4) **Includes :** Yöntem, bir dizi geri dönen bir dizge içinde bulunabilir olup olmadığı tespit edip “true” ya da “false” olarak değerlendirir.

Kullanımı:

```
const str = 'To be, or not to be, that is the question.'


console.log(str.includes('To be'))       // true
console.log(str.includes('question'))    // true
console.log(str.includes('nonexistent')) // false
console.log(str.includes('To be', 1))    // false
console.log(str.includes('TO BE'))       // false
```

```
console.log(str.includes(''))            // true
```

5) IndexOf : Yöntem, çağrı içinde dizini döndürür String, aramanın başlangıç belirtilen değerin ilk geçtiği, nesne fromIndex. -1Değer bulunmazsa döndürür .

Kullanımı:

```
const str = 'Brave new world'


console.log('Index of first w from start is ' + str.indexOf('w'))   // logs 8
console.log('Index of "new" from start is ' + str.indexOf('new'))   // logs 6
```

6) LastIndexOf : Yöntem döner arama içinde göstergesi Stringgelen geriye doğru arama belirtilen değerin son bir oluşum nesne, fromIndex. Değer bulunmazsa -1 değerini döndürür.

Kullanımı:

```
'Blue Whale, Killer Whale'.lastIndexOf('blue'); // returns -1
```

7) PadEnd : Yöntem, pedler, belirli bir dizeye mevcut dizisi (gerekirse tekrar), böylece elde edilen dizi, belirli bir uzunluğa ulaşmaktadır. Dolgu geçerli dizenin sonundan uygulanır.

Kullanımı:

```
'abc'.padEnd(10);          // "abc       "
'abc'.padEnd(10, "foo");   // "abcfoofoof"
'abc'.padEnd(6, "123456"); // "abc123"
'abc'.padEnd(1);           // "abc"
```

8) PadStart : Elde edilen dizi verilen uzunluğuna gelene kadar yöntemi (gerekirse birden çok kez,) başka bir dize mevcut dize doldurur. Doldurma, geçerli dizenin başından itibaren uygulanır.

Kullanımı:

```
'abc'.padStart(10);         // "       abc"
```

```
'abc'.padStart(10, "foo");  // "foofoofabc"
'abc'.padStart(6,"123465"); // "123abc"
'abc'.padStart(8, "0");     // "00000abc"
'abc'.padStart(1);          // "abc"
```

9) Slice : Yöntem bir dize bir bölümünü ayıklar ve özgün dize değiştirmeden, yeni bir dizge olarak döndürür.

Kullanımı:

```
let str = 'The morning is upon us.'
```

```
str.slice(-3)      // returns 'us.'
str.slice(-3, -1)  // returns 'us'
str.slice(0, -1)   // returns 'The morning is upon us'
```

10) Split : Yöntem, bir döner String, belirli bir ayırıcı dizisinin her bir derecede dizi ayırarak şeritlerinin bir diziye.

Kullanımı:

```
const myString = ''
```

```
const splits = myString.split()


console.log(splits)


// ↪ [""]
```

11) SubString: Yöntem bir kısmını geri döndürür stringbaşlangıç ve bitiş indeksleri arasında ya da karakter dizisinin sonuna kadar.

Kullanımı:

```
const str = 'Mozilla';

console.log(str.substring(1, 3));
// expected output: "oz"
```

12) Trim : Yöntem kaldırır bir dize her iki ucundan whitespace. Bu bağlamdaki boşluk, tüm boşluk karakterleri (boşluk, sekme, aralıksız boşluk vb.) Ve tüm satır sonlandırıcı karakterleridir (LF, CR vb.).

Kullanımı:

```
const greeting = '   Hello world!   ';

console.log(greeting);
// expected output: "   Hello world!   ";

console.log(greeting.trim());
// expected output: "Hello world!";
```

13) TrimEnd : Yöntem kaldırır bir dize sonundan whitespace. trimRight(), bu yöntemin takma adıdır.

Kullanımı:

```
const greeting = '   Hello world!   ';

console.log(greeting);
// expected output: "   Hello world!   ";

console.log(greeting.trimEnd());
// expected output: "   Hello world!";
```

14)TrimStart : Yöntem kaldırır bir dizinin başlangıcından itibaren whitespace. trimLeft(), bu yöntemin takma adıdır.

Kullanımı:

```
const greeting = '   Hello world!   ';

console.log(greeting);
// expected output: "   Hello world!   ";

console.log(greeting.trimStart());
// expected output: "Hello world!   ";
```

15)Repeat : Yöntem yapılar ve iadeler birlikte birleştirilmiş o çağrıldığı dizesinin kopyalarının belirtilen sayıda içeren yeni dize.

Kullanımı:

```
const chorus = 'Because I\'m happy. ';

console.log('Chorus lyrics for "Happy": ' + chorus.repeat(5));
```

// expected output: "Chorus lyrics for "Happy": Because I'm happy. Because I'm happy. Because I'm happy. Because I'm happy. Because I'm happy. "

16)Replace : Yöntem bazı veya tüm maçları ile yeni bir dize döndürür patternbir yerine replacement. . pattern, Bir dize veya a olabilir RegExpve replacementher eşleşme için çağrılacak bir dize veya işlev olabilir. Eğer pattern bir dizidir sadece ilk oluşum değiştirilecektir. Orijinal dize değişmeden kalır.

Kullanımı:

```
var str = 'Twas the night before Xmas...';

var newstr = str.replace(/xmas/i, 'Christmas');

console.log(newstr);  // Twas the night before Christmas...
```