آموزش TypeScript – آموزش رشته ها در تایپ اسکریپت (TypeScript)

شیء String این امکان را به شما می دهد تا با مجموعه ای از کاراکترها کار کنید. این شیء نوع داده ای اولیه ی رشته ای را به همراه تعدادی از متدهای کمکی wrap می کند.

سینتکس

```
var var_name = new String(string);
```

لیستی از مشخصه های موجود در شیء String به همراه توضیحات آن ها را در زیر مشاهده می کنید:

ردیف	مشخصه ها و توضیحات
.1	Constructor مرجعی را به تابع String ای که این شیء را ایجاد کرده است، برمی گرداند.
.2	Length طول رشته را برمی گرداند.
.3	Prototype این مشخصه این امکان را به شما می دهد تا بتوانید مشخصات و متدهایی را به شیئی اضافه کنید.

نمونه ها در TypeScript

.1 Constructor

مرجعی را به تابع String ای که این شیء را ایجاد کرده است، برمی گرداند.

مثال

```
var str = new String( "This is string" );

console.log("str.constructor is:" + str.constructor)
```

بعد از کامپایل کردن کد بالا، همین کد در جاوااسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
str.constructor is:function String() { [native code] }
```

2. Length

طول رشته را برمی گرداند.

مثال

```
var uname = new String("Hello World")

console.log(uname)

console.log("Length "+uname.length)  // returns the total number of characters

                    // including whitespace
```

بعد از کامپایل کردن کد بالا، همین کد در جاوااسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
Hello World
Length 11
```

3. Prototype

این مشخصه این امکان را به شما می دهد تا بتوانید مشخصات و متدهایی را به شیئی اضافه کنید.

مثال

```
function employee(id:number,name:string) {

  this.id = id

  this.name = name

}

var emp = new employee(123,"Smith")

employee.prototype.email="smith@abc.com"

console.log("Employee 's Id: "+emp.id)
```

```
console.log("Employee's name: "+emp.name)

console.log("Employee's Email ID: "+emp.email)
```

بعد از کامپایل کردن کد بالا، کد جاوا اسکریپت زیر ایجاد می شود.

```
//Generated by typescript 1.8.10

function employee(id, name) {

  this.id = id;

  this.name = name;

}

var emp = new employee(123, "Smith");

employee.prototype.email = "smith@abc.com";

console.log("Employee 's Id: " + emp.id);

console.log("Employee's name: " + emp.name);

console.log("Employee's Email ID: " + emp.email);
```

و خروجی به صورت زیر نمایش داده می شود.

```
Employee’s Id: 123
Emaployee’s name: Smith
Employee’s Email ID: smith@abc.com
```

آموزش متدهای String در TypeScript

لیستی از متدهای موجود در شیء String به همراه توضیحات آن ها را در زیر مشاهده می کنید:

ردیف	متدها و توضیحات
.1	charAt() کاراکتری با ایندکس مشخص را برگشت می دهد.
.2	charCodeAt()

	عددی را برگشت می دهد که حاکی از مقدار Unicode کاراکتر موجود در ایندکس داده شده است.
.3	concat() متن دو رشته را ترکیب کرده و رشته ی جدیدی را برگشت می دهد.
.4	indexOf() ایندکس را داخل فراخوانی شیء String مربوط به اولین رخداد مقدار مشخص شده برگشت می دهد. در صورتی که مقداری پیدا نکند، 1ـ را برگشت می دهد.
.5	lastIndexOf() ایندکس را داخل فراخوانی شیء String مربوط به آخرین رخداد مقدار مشخص شده برگشت می دهد. در صورتی که مقداری پیدا نکند، 1ـ را برگشت می دهد.
.6	localeCompare() عددی را برگشت می دهدکه نشان می دهد یک رشته ی مرجع در فرآیند مرتب سازی، قبل یا بعد از رشته ی داده شده می آید، یا اینکه با رشته ی داده شده یکی است.
.7	match() برای منطبق کردن یک عبارت معمولی بر یک رشته استفاده می شود.
.8	replace() بین عبارتی معمولی و یک رشته به دنبال زیررشته ای می گردد و جای زیررشته ی پیدا شده را با زیر رشته ی جدیدی عوض می کند.
.9	search() بین یک عبارت معمولی و یک رشته ی مشخص به جستجو می پردازد.
.10	slice() بخشی از رشته را خارج کرده و رشته ی جدیدی را برگشت می دهد.
.11	split() با تفکیک رشته به زیر رشته ها، شیء String را به آرایه ای از رشته ها تقسیم می کند.
.12	substr() از موقعیت مشخص شده تا تعداد مشخصی از کاراکترها، کاراکترها را در رشته ای برگشت می دهد.
.13	substring() کاراکترهای موجود در یک رشته را بین دو ایندکس موجود در این رشته برگشت می دهد.
.14	toLocaleLowerCase() در عین رعایت کردن تنظیمات متنی و محلی فعلی، حروف کاراکترهای موجود در یک رشته را کوچک می کند.
.15	toLocaleUpperCase()

	در عین رعایت کردن تنظیمات متنی و محلی فعلی، حروف کاراکترهای موجود در یک رشته را بزرگ می کند.
.16	toLowerCase() مقدار رشته ای را به صورت حروف کوچک برگشت می دهد.
.17	toString() رشته ای را برگشت می دهد که بیانگر شیء مشخصی است.
.18	toUpperCase() مقدار رشته ای را به صورت حروف بزرگ برگشت می دهد.
.19	valueOf() مقدار اولیه ی شیء مشخصی را برگشت می دهد.

1. charAt()

کاراکتری با ایندکس مشخص را برگشت می دهد. کاراکترهای موجود در یک رشته از چپ به راست ایندکس بندی می شوند. ایندکس اولین کاراکتر برابر با صفر و ایندکس آخرین کاراکتر در رشته ای به نام stringName قرار داشته و برابر با stringName.length – 1 است.

سینتکس

```
string.charAt(index);
```

جزییات آرگومان

ایندکس : عدد صحیحی بین صفر و یک واحد کمتر از طول رشته می باشد.

مقدار برگشتی

```
var str = new String("This is string");

console.log("str.charAt(0) is:" + str.charAt(0));

console.log("str.charAt(1) is:" + str.charAt(1));

console.log("str.charAt(2) is:" + str.charAt(2));

console.log("str.charAt(3) is:" + str.charAt(3));

console.log("str.charAt(4) is:" + str.charAt(4));
```

```
console.log("str.charAt(5) is:" + str.charAt(5));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
str.charAt(0) is:T
str.charAt(1) is:h
str.charAt(2) is:i
str.charAt(3) is:s
str.charAt(4) is:
str.charAt(5) is:i
```

2. charCodeAt()

این متد مقدار عددی ای را برگشت می دهد که بیانگر مقدار Unicode کاراکتر در ایندکس داده شده است.

کد Unicode مقادیری از ۰ تا ۱۱۱۴۱۱۱ را شامل می شود. ۱۲۸ کد Unicode اول، مواردی هستند که مستقیما در رمزگذاری کاراکتر ASCII وجود دارند. مقدار برگشتی توسط charCodeAt() همیشه کمتر از ۶۵۵۳۶ است.

سینتکس

```
string.charCodeAt(index);
```

جزییات آرگومان

ایندکس: عدد صحیحی بین صفر و یک واحد کمتر از طول رشته می باشد. در صورتی که ایندکس رشته ای مشخص نباشد، مقدار پیش فرض آن برابر با صفر خواهد بود.

مقدار برگشتی

مقدار عددی ای را برگشت می دهد که بیانگر مقدار Unicode کاراکتر در ایندکس داده شده است. در صورتی که مقدار ایندکس داده شده بین صفر و یک واحد کمتر از طول رشته نباشد، مقدار NaN برگشت داده می شود.

مثال

```
var str = new String("This is string");
```

```
console.log("str.charAt(0) is:" + str.charCodeAt(0));

console.log("str.charAt(1) is:" + str.charCodeAt(1));

console.log("str.charAt(2) is:" + str.charCodeAt(2));

console.log("str.charAt(3) is:" + str.charCodeAt(3));

console.log("str.charAt(4) is:" + str.charCodeAt(4));

console.log("str.charAt(5) is:" + str.charCodeAt(5));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
str.charAt(0) is:84
str.charAt(1) is:104
str.charAt(2) is:105
str.charAt(3) is:115
str.charAt(4) is:32
str.charAt(5) is:105
```

3. concat()

این دو یا چند رشته را اضافه کرده و رشته جدیدی را برگشت میدهد.

سینتکس

```
string.concat(string2, string3[, ..., stringN]);
```

جزئیات آرگومان

string2...stringN – این ها رشته هایی هستند که قرار است با یکدیگر ترکیب شوند.

مقدار برگشتی

یک رشته ترکیب شده را برگشت می دهد.

مثال

```
var str1 = new String( "This is string one" );

var str2 = new String( "This is string two" );

var str3 = str1.concat( str2 );
```

```
console.log("str1 + str2 : "+str3)
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
str1 + str2 : This is string oneThis is string two
```

4. indexOf()

این متد ایندکسی را از داخل شی calling String اولین رخداد مربوط به یک مقدار مشخص شده برگشت می دهد. و کار جستجوی خود را از fromIndex آغاز می کند. در صورتی که مقدار مشخصی پیدا نشود، کار جستجو از ۱ـ آغاز می شود.

سینتکس

```
string.indexOf(searchValue[, fromIndex])
```

جزئیات آرگومان

- searchValue: رشته ای است که بیانگر مقداری است که باید به جست و جوی آن پرداخت.
- fromIndex : مکانی داخل رشته calling است که کار جست و جو باید از آن آغاز شود. این مکان می تواند هر عدد صحیحی بین صفر و طول رشته را شامل شود. مقدار پیشفرض برابر با صفر است.

مقدار برگشتی

درصورتی که رخدادی پیدا شود، ایندکس مربوط به آن را برگشت می دهد، در غیر این صورت مقدار ۱ـ برگشت داده می شود.

مثال

```
var str1 = new String( "This is string one" );

var index = str1.indexOf( "string" );

console.log("indexOf found String :" + index );

var index = str1.indexOf( "one" );
```

```
console.log("indexOf found String :" + index );
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
indexOf found String :8

indexOf found String :15
```

5. lastIndexOf()

این متد ایندکسی را از داخل شی calling String آخرین رخداد مربوط به یک مقدار مشخص شده برگشت می دهد. و کار جستجوی خود را از fromIndex آغاز می کند. در صورتی که مقدار مشخصی پیدا نشود، کار جستجو از ۱ـ آغاز می شود.

سینتکس

```
string.lastIndexOf(searchValue[, fromIndex])
```

جزئیات آرگومان

- searchValue : رشته ای است که بیانگر مقداری است که باید به جست و جوی آن پرداخت.
- fromIndex : مکانی داخل رشته calling است که کار جست و جو باید از آن آغاز شود. این مکان می تواند هر عدد صحیحی بین صفر و طول رشته را شامل شود. مقدار پیشفرض برابر با صفر است.

مقدار برگشتی

درصورتی که آخرین رخداد پیدا شود، ایندکس مربوط به آن را برگشت می دهد، در غیر این صورت مقدار ۱ـ برگشت داده می شود.

مثال

```
var str1 = new String( "This is string one and again string" );

var index = str1.lastIndexOf( "string" );

console.log("lastIndexOf found String :" + index );
```

```
index = str1.lastIndexOf( "one" );

console.log("lastIndexOf found String :" + index );
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
lastIndexOf found String :29

lastIndexOf found String :15
```

6. localeCompare()

عددی را برگشت می دهدکه نشان می دهد یک رشته ی مرجع در فرآیند مرتب سازی، قبل یا بعد از رشته ی داده شده می آید، یا اینکه با رشته ی داده شده یکی است.

سینتکس

```
string.localeCompare( param )
```

جزئیات آرگومان

param : رشته ای است که باید با شی رشته ای مقایسه شود.

مقدار برگشتی

- 0 : در صورتی که رشته مقایسه شده ۱۰۰ درصد مطابقت داشته باشد.
- 1 : در صورتی که مطابقتی وجود نداشته باشد و مقدار پارامتر قبل از مقدار شی رشته ای در مرتب سازی locale آمده باشد.
- مقدار منفی : در صورتی که مطابقتی وجود نداشته باشد و مقدار پارامتر بعد از مقدار شی رشته ای در مرتب سازی locale آمده باشد.

مثال

```
var str1 = new String( "This is beautiful string" );
```

```
var index = str1.localeCompare( "This is beautiful string");

console.log("localeCompare first :" + index );
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
localeCompare first :0
```

8. replace()

بین عبارتی معمولی و یک رشته به دنبال زیررشته ای می گردد و جای زیررشته ی پیدا شده را با زیر رشته ی جدیدی عوض می کند.

این رشته می تواند شامل الگوهای جا به جایی ویژه ی زیر باشد.

الگو	ورودی ها
$$	"$" را وارد می کند.
$&	زیر رشته ی مطابق را وارد می کند.
$`	بخشی از زیر رشته را وارد می کند که این بخش بر زیر رشته ی مطابق مقدم است.
$'	بخشی از زیر رشته را وارد می کند که این بخش بر زیر رشته ی مطابق موخر است.
$n یا $nn	در حالی که n و nn اعداد بر مبنای 10 هستند، با این فرض که اولین آرگومان، شیء RegExp باشد، n امین زیر رشته ی مطابق پرانتزدار را وارد می کند.

سینتکس

```
string.replace(regexp/substr, newSubStr/function[, flags]);
```

جزئیات آرگومان

- regexp : این آرگومان شیء RegExp نیز گفته می شود. مورد مطابق توسط مقدار برگشتی پارامتر شماره 2 جایگزین می شود.
- substr : رشته ای که باید توسط newSubStr جایگزین شود.

- newSubStr : رشته ای که جای خود را به زیر رشته ی دریافتی از پارامتر اول می دهد.
- function : تابعی است که برای ایجاد زیر رشته ی جدید احضار می شود.
- flags : رشته ای است که شامل یکی از ترکیب های پرچم های RegExp باشد.

مقدار برگشتی

به سادگی رشته ی جدیدی که دستخوش تغییر شده است، برگشت داده می شود.

مثال

```
var re = /apples/gi;

var str = "Apples are round, and apples are juicy.";

var newstr = str.replace(re, "oranges");

console.log(newstr)
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
oranges are round, and oranges are juicy.
```

مثال

```
var re = /(\w+)\s(\w+)/;

var str = "zara ali";

var newstr = str.replace(re, "$2, $1");

console.log(newstr);
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
ali, zara
```

9. search()

بین یک عبارت معمولی و این شیء رشته ای به جستجو می پردازد.

سینتکس

```
string.search(regexp);
```

جزئیات آرگومان

Regexp : مخفف شیء عبارت معمولی (regular expression) است. در صورتی که شیء obj ای که regexp نیست، به تابعی تحویل داده شود، این شیء به صورت غیرمستقیم با استفاده از RegExp(obj) جدید به regexp تبدیل می شود.

مقدار برگشتی

در صورتی که کار جستجو موفقیت آمیز باشد، شاخص عبارت معمولی موجود در رشته را برگشت می دهد. در غیر این صورت عدد 1- برگشت داده می شود.

مثال

```
var re = /apples/gi;

var str = "Apples are round, and apples are juicy.";

if (str.search(re) == -1 ) {

  console.log("Does not contain Apples" );

} else {

  console.log("Contains Apples" );

}
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
Contains Apples
```

slice() 10

بخشی از رشته را خارج کرده و رشته ی جدیدی را برگشت می دهد.

سینتکس

```
string.slice( beginslice [, endSlice] );
```

جزئیات آرگومان

- beginSlice : ایندکس مبتنی بر صفری است که قرار است کار استخراج از آن آغاز شود.
- endSlice : ایندکس مبتنی بر صفری است که قرار است کار استخراج در آن پایان یابد. اگر بخشی حذف شده باشد، slice کار استخراج را تا انتهای رشته ادامه می دهد.

مقدار برگشتی

در صورتی که کار استخراج موفقیت آمیز باشد، شاخص عبارت معمولی داخل رشته برگشت داده می شود، در غیر اینصورت عدد -1 برگشت داده می شود.

مثال

```
var str = "Apples are round, and apples are juicy.";

var sliced = str.slice(3, -2);

console.log(sliced);
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
les are round, and apples are juic
```

11. split()

با تفکیک رشته به زیر رشته ها، شیء String را به آرایه ای از رشته ها تقسیم می کند.

سینتکس

```
string.split([separator][, limit]);
```

جزئیات آرگومان

- separator : کاراکتری را مشخص می کند که قرار است در تفکیک رشته استفاده شود. اگر separator وجود نداشته باشد، آرایه ی برگشتی شامل المانی متشکل از کل رشته می باشد.
- limit : عدد صحیحی است که حد تعداد split هایی که قرار است پیدا شود را مشخص می کند.

مقدار برگشتی

متد split آرایه ی جدید را برگشت می دهد. همچنین در صورتی که رشته خالی باشد، این متد به جای برگشت دادن آرایه ی خالی، آرایه ای را برگشت می دهد که شامل یک رشته ی خالی است.

مثال

```
var str = "Apples are round, and apples are juicy.";

var splitted = str.split(" ", 3);

console.log(splitted)
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
[ 'Apples', 'are', 'round,' ]
```

substr() .12

از موقعیت مشخص شده تا تعداد مشخصی از کاراکترها، کاراکترها را در رشته ای برگشت می دهد.

سینتکس

```
string.substr(start[, length]);
```

جزئیات آرگومان

Start : مکانی که قرار است کار استخراج کاراکترها در آن آغاز شود (عدد صحیحی بین صفر و یک واحد کمتر از طول رشته)

Length : تعداد کاراکترهایی که قرار است استخراج شوند.

نکته : در صورتی که start منفی باشد، substr از آن به عنوان ایندکس کاراکتر انتهای رشته استفاده می کند.

مقدار برگشتی

متد substr() بر اساس پارامترهای ارائه شده زیر رشته جدیدی را برگشت می دهد.

مثال

```
var str = "Apples are round, and apples are juicy.";

console.log("(1,2): "  + str.substr(1,2));

console.log("(-2,2): "  + str.substr(-2,2));

console.log("(1): "  + str.substr(1));

console.log("(-20, 2): " + str.substr(-20,2));

console.log("(20, 2): " + str.substr(20,2));
```

بعد از کامپایل کردن کد بالا، کد جاوا اسکریپت یکسانی ایجاد می شود.

و نتیجه زیر نمایش داده می شود :

```
(1,2): pp
(-2,2): y.
(1): pples are round, and apples are juicy.
(-20, 2): nd
(20, 2): d
```

13. substring()

این متد زیرمجموعه ای از یک شیء رشته را برگشت می دهد.

سینتکس

```
string.substring(indexA, [indexB])
```

جزئیات آرگومان

- indexA: عدد صحیحی بین صفر و یک واحد کمتر از طول رشته است.
- indexB (اختیاری) : عدد صحیحی بین صفر و طول رشته است.

مقدار برگشتی

متد substring براساس پارامترهای داده شده زیر رشته ی جدیدی را برگشت می دهد.

مثال

```
var str = "Apples are round, and apples are juicy.";

console.log("(1,2): "   + str.substring(1,2));

console.log("(0,10): "  + str.substring(0, 10));

console.log("(5): "    + str.substring(5));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
(1,2): p
(0,10): Apples are
(5): s are round, and apples are juicy.
```

14. toLocaleLowerCase()

در عین رعایت کردن تنظیمات متنی و محلی فعلی، حروف کاراکترهای موجود در یک رشته را کوچک می کند. در اغلب زبان ها، خروجی برگشتی این متد همان خروجی toLowerCase است.

مثال

```
string.toLocaleLowerCase( )
```

مقدار برگشتی

رشته ای را با حروف کوچک و همراه با locale فعلی برگشت می دهد.

مثال

```
var str = "Apples are round, and Apples are Juicy.";

console.log(str.toLocaleLowerCase( ));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
apples are round, and apples are juicy.
```

15. toLocaleUpperCase()

در عین رعایت کردن تنظیمات متنی و محلی فعلی، حروف کاراکترهای موجود در یک رشته را بزرگ می کند. در اغلب زبان ها، خروجی برگشتی این متد همان خروجی toUpperCase است.

سینتکس

```
string.toLocaleUpperCase( )
```

مقدار برگشتی

رشته ای را با حروف بزرگ و همراه با locale فعلی برگشت می دهد.

مثال

```
var str = "Apples are round, and Apples are Juicy.";

console.log(str.toLocaleUpperCase( ));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
APPLES ARE ROUND, AND APPLES ARE JUICY.
```

16. toLowerCase()

مقدار رشته ای را به صورت حروف کوچک برگشت می دهد.

سینتکس

```
string.toLowerCase( )
```

مقدار برگشتی

مقدار رشته ای را به صورت حروف کوچک برگشت می دهد.

مثال

```
var str = "Apples are round, and Apples are Juicy.";

console.log(str.toLowerCase( ))
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
apples are round, and apples are juicy.
```

toString() .17

رشته ای را برگشت می دهد که بیانگر شیء مشخصی است.

سینتکس

```
string.toString( )
```

مقدار برگشتی

رشته ای را برگشت می دهد که بیانگر شیء مشخصی است.

مثال

```
var str = "Apples are round, and Apples are Juicy.";

console.log(str.toString( ));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
Apples are round, and Apples are Juicy.
```

toUpperCase() .18

مقدار رشته ای را به صورت حروف بزرگ برگشت می دهد.

سینتکس

```
string.toUpperCase( )
```

مقدار برگشتی

مقدار رشته ای را به صورت حروف کوچک برگشت می دهد.

مثال

```
var str = "Apples are round, and Apples are Juicy.";

console.log(str.toUpperCase( ));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
APPLES ARE ROUND, AND APPLES ARE JUICY.
```

valueOf() .19

مقدار اولیه ی شیء رشته ای را برگشت می دهد.

سینتکس

```
string.valueOf( )
```

مقدار برگشتی

مقدار اولیه ی شیء رشته ای را برگشت می دهد.

مثال

```
var str = new String("Hello world");

console.log(str.valueOf( ));
```

بعد از کامپایل کردن کد بالا، همین کد در جاوا اسکریپت ایجاد می شود.

و خروجی به صورت زیر نمایش داده می شود.

```
Hello world
```