آموزش Type Script ـ آموزش سینتکس های اصلی Type Script

سینتکس مجموعه ای از مقررات را برای برنامه نویسی تعریف می کند. هر مشخصه ی زبانی سینتکس مخصوص به خود را دارد. یک برنامه ی تایپ اسکریپت از بخش های زیر تشکیل شده است:

- ماژول ها
- توابع
- متغیرها
- عبارات و جملات
- کامنت ها

اولین کد تایپ اسکریپت(Type Script) شما

بیایید کارمان را با مثال همیشگی "Hello World" شروع کنیم.

```
var message:string = "Hello World"

console.log(message)
```

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

```
//Generated by typescript 1.8.10

var message = "Hello World";

console.log(message);
```

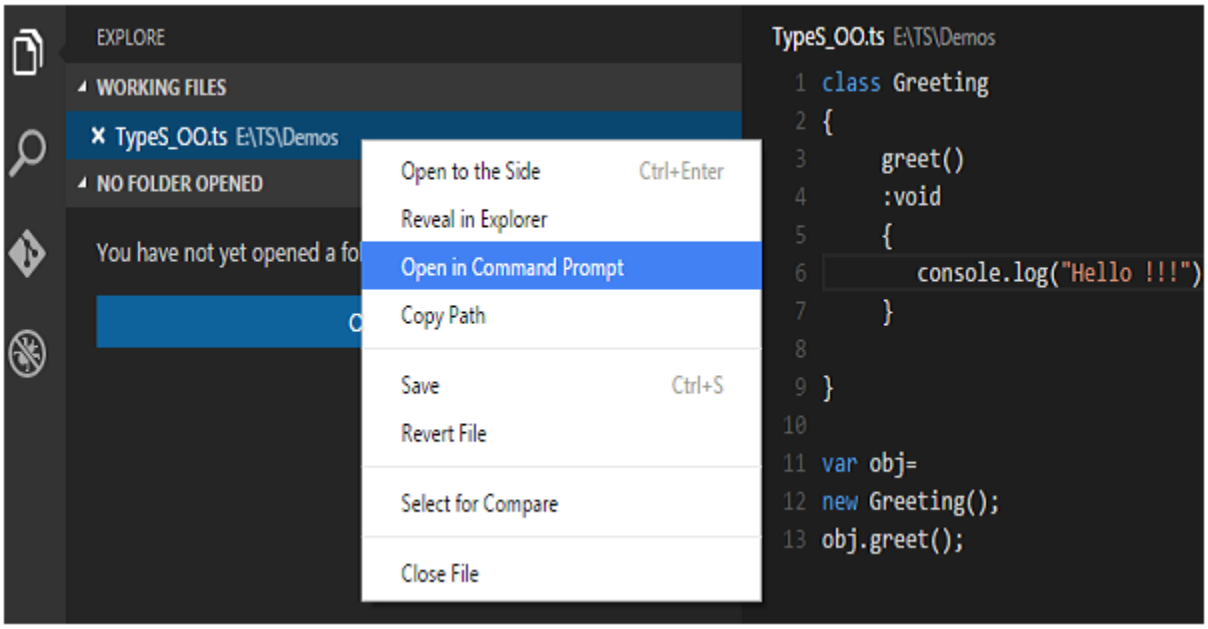
- خط اول متغیری را با نام message اعلان می کند. متغیرها مکانیزمی برای ذخیره سازی مقادیر در برنامه هستند.
- خط دوم مقدار متغیر را در prompt پرینت می کند. در اینجا console به terminal window اشاره دارد. تابع *log ()* برای نمایش متن بر روی صفحه استفاده می شود.

آموزش کامپایل و اجرا کردن یک برنامه ی تایپ اسکریپت(Type Script)

بیایید ببینیم چگونه می توانیم با استفاده از Visual Studio Code یک برنامه ی تایپ اسکریپت را کامپایل و اجرا کنیم. مراحل زیر را دنبال کنید:

مرحله اول : فایل خود را با فرمت ts. ذخیره کنید. می توانید اسم فایل را Test.ts بگذارید. بعد از اینکه فایل خود را ذخیره می کنید، code editor در صورتی که خطایی در کد وجود داشته باشد، آن را نمایش می دهد.

مرحله دوم : در ستون اکسپلور مربوط به VS code زیر گزینه ی Working Files بر روی فایل TypeScript راست کلیک کنید و گزینه ی Open in Command Prompt را انتخاب کنید.



مرحله سوم : برای اینکه بتوانید فایل خود را کامپایل کنید، دستور زیر را در terminal window وارد کنید.

```
tsc Test.ts
```

مرحله چهارم : این فایل به Test.js کامپایل می شود. برای اینکه بتوانید برنامه ی نوشته شده را اجرا کنید در terminal window وارد کنید.

```
node Test.js
```

پرچم های کامپایلر

پرچم های کامپایلر این امکان را به شما می دهند تا در حین فرآیند کامپایل کردن، رفتار کامپایلر را تغییر دهید. هر یک از این پرچم ها تنظیمات مخصوص به خود را دارد، که همین تنظیمات است که امکان تغییر رفتار کامپایلر را برای شما فراهم می کند.

در جدول زیر برخی از پرچم های معمول مربوط به کامپایلر TSC نشان داده شده است. در یک کد دستوری می توان برخی از این پرچم ها یا تمام آن ها را مورد استفاده قرار داد.

ردیف	پرچم کامپایلر و توضیحات
.1	—Help راهنما را نمایش می دهد.
.2	—Module ماژول های خارجی را load می کند.
.3	—Target نسخه ی هدف ECMA را مشخص می کند.
.4	—Declaration یک فایل d.ts. دیگر را اضافه می کند.
.5	—removeComments تمام کامنت ها را از فایل خروجی پاک می کند.
.6	—Out فایل های متعددی را در یک فایل خروجی کامپایل می کند.
.7	—Sourcemap یک فایل sourcemap با فرمت (.map) را ایجاد می کند.
.8	—module noImplicitAny کامپایلر را از استنتاج کردن any type منع می کند.
.9	—Watch تغییرات اعمال شده بر روی فایل را زیر نظر می گیرد و آن ها را بر روی fly مجددا کامپایل می کند.

نکته : فایل های متعددی را می توانیدبه صورت یکجا کامپایل کنید.

```
tsc file1.ts, file2.ts, file3.ts
```

آموزش شناسه ها در تایپ اسکریپت(Type Script)

شناسه ها نام هایی هستند که به المان های موجود در یک برنامه مانند متغیرها، توابع و ... داده می شوند.
قوانین مربوط به شناسه ها عبارتند از:

شناسه ها

- می توانند هم کاراکترها و هم اعداد را شامل شوند. با این حال شناسه ها نمی توانند با عدد شروع شوند.
- نمی توانند شامل نمادهای خاص به غیر از (_) یا ($) باشند.
- نمی توانند عبارات کلیدی باشند.

- باید منحصر به فرد باشند.
- به بزرگی و کوچکی حروف حساس هستند.
- نباید شامل space باشند.

در جدول زیر نمونه هایی از شناسه های معتبر و نامعتبر آورده شده است:

شناسه های معتبر	شناسه های نامعتبر
firstName	Var
first_name	first name
num1	first-name
$result	1number

آموزش عبارت های کلیدی در تایپ اسکریپت(Type Script)

عبارت های کلیدی در متن یک زبان معنی خاصی دارند. در جدول زیر برخی از عبارت های کلیدی موجود در تایپ اسکریپت را می توانید مشاهده کنید.

Break	as	any	switch
Case	if	throw	else
Var	number	string	get
module	type	instanceof	typeof
Public	private	enum	export
Finally	for	while	void
Null	super	this	new
In	return	true	false
Any	extends	static	let
package	implements	interface	function
New	try	yield	const
continue	do	catch	

آموزش Whitespace و Line Break ها در Type Script

تایپ اسکریپت جاهای خالی، tab ها و newline هایی که در برنامه ظاهر می شوند را نادیده می گیرد. در نتیجه بدون هیچ نگرانی می توانید در برنامه خود از این موارد استفاده کنید، تا خواندن و فهم کدتان را آسان تر کنید.

تایپ اسکریپت به بزرگی و کوچکی حروف حساس است

این یعنی تایپ اسکریپت بین حروف بزرگ و کوچک تمایز قائل می شود.

استفاده از Semicolon ها اختیاری است

هر خط موجود در instruction یک statement یا عبارت نامیده می شود. در صورتی که مایلید می توانید در تایپ اسکریپت از Semicolon ها استفاده کنید.

مثال

```
console.log("hello world")

console.log("We are learning TypeScript")
```

یک خط می تواند شامل چندین statement باشد. با این حال این statement ها باید توسط یک semicolon تفکیک شوند.

آموزش کامنت ها(Comment) در تایپ اسکریپت(Type Script)

کامنت ها روشی برای خواناتر کردن برنامه می باشند. از کامنت ها می توان برای وارد کردن اطلاعات اضافی در رابطه با برنامه استفاده کرد. مانند نویسنده ی کد، راهنمایی های مربوط به تابع یا یک construct و . . . . کامپایلر کامنت ها را نادیده می گیرد.

تایپ اسکریپت از این نوع کامنت ها پشتیبانی می کند:

- کامنت های یک خطی (//) – هر متنی که بین / / و در انتهای یک خط قرار داشته باشد، کامنت در نظر گرفته می شود.
- کامنت های چند خطی (/* */) – این کامنت ها می توانند چندین خط را اشغال کنند.

مثال:

```
//this is single line comment


/* This is a
  Multi-line comment
*/
```

آموزش تایپ اسکریپت(Type Script) و شیء گرایی(Object Oriented)

تایپ اسکریپت جاوا اسکریپت شیء گرا است. شیء گرایی نمونه ای از برنامه نویسی نرم افزار است که از مدل سازی جهان واقعی پیروی می کند. شیء گرایی یک برنامه را به عنوان مجموعه ای از اشیاء در نظر می گیرد به گونه ای که این اشیاء از طریق مکانیزمی به نام methods با یکدیگر ارتباط برقرار می کنند. تایپ اسکریپت از کامپوننت های شیء گرا نیز پشتیبانی می کند.

- شیء (Object) : هر object ارائه ای real time از هر هویتی (entity) است. براساس Grady Brooch هر شیء باید سه ویژگی داشته باشد:
  - حالت (State) : با استفاده از attribute های یک شیء توصیف می شود.
  - رفتار (Behavior) : چگونگی عملکرد شیء را توصیف می کند.
  - هویت (Identity) : مقدار منحصر به فردی است که یک شیء را از مجموعه ای از اشیاء مشابه به خود متمایز می کند.
- کلاس (Class) : کلاس برحسب OOP نقشه ی اولیه ای برای ایجاد اشیاء محسوب می شود. کلاس داده های مورد نیاز شیء را در خود نگه می دارد.
- متد (Method) : متدها رابطه ی بین اشیاء را تسهیل می کنند.

مثال : تایپ اسکریپت و شیء گرایی

```
class Greeting {

  greet():void {

    console.log("Hello World!!!")

  }

}

var obj = new Greeting();

obj.greet();
```

در مثال بالا کلاس *Greeting* تعریف شده است. این کلاس متدی به نام *greet ()* دارد. این متد رشته ی "Hello World" را در terminal پرینت می کند. عبارت کلیدی new ، شیء کلاس obj را ایجاد می کند. این شیء متد *greet ()* را احضار می کند.

بعد از کامپایل کردن کد جاوااسکریپت زیر ایجاد می شود.

```
//Generated by typescript 1.8.10

var Greeting = (function () {
```

```
  function Greeting() {

  }

  Greeting.prototype.greet = function () {

    console.log("Hello World!!!");

  };

                return Greeting;

}());


var obj = new Greeting();

obj.greet()
```

خروجی برنامه ی بالا در زیر آمده است:

```
Hello World!!!
```