آموزش TypeScript – آموزش متغیرها در تایپ اسکریپت (TypeScript)

متغیر فضایی دارای نام در حافظه است که مقادیر را ذخیره می کند. به بیان دیگر، متغیر در یک برنامه به عنوان نگهدارنده ی مقادیر عمل می کند. متغیرهای تایپ اسکریپت باید از مقررات اسم گذاری جاوا اسکریپت پیروی کنند.

- اسم متغیرها می تواند شامل حروف الفبا و اعداد باشد.
- متغیرها به غیر از (_) و ($) نمی توانند شامل جای خالی و کاراکترهای ویژه ی دیگر باشند.
- اسم متغیرها نمی تواند با یک عدد شروع شود.

قبل از اینکه بتوان از متغیری استفاده کرد، باید آن را اعلان کرد. برای اعلان متغیرها از عبارت کلیدی var استفاده کنید.

آموزش اعلان متغیر در تایپ اسکریپت(TypeScript)

سینتکس اعلان یک متغیر در تایپ اسکریپت به این صورت است که باید یک (:) را بعد از اسم متغیر قرار داد. بعد از آن نوع داده ی این متغیر را مشخص کرد. درست مانند جاوا اسکریپت برای اعلان یک متغیر از عبارت کلیدی var استفاده می کنیم.

زمانی که متغیری را اعلان می کنید، 4 گزینه پیش رو دارید:

- نوع متغیر و مقدار آن را در یک دستور اعلان کنید.

- نوع آن را اعلان کنید، اما مقدار آن را اعلان نکنید. در این حالت متغیر بر روی مقدار تعریف نشده قرار می گیرد.

- مقدار آن را اعلان کنید، اما نوع آن را اعلان نکنید. در این صورت نوع متغیر بر روی any قرار میگیرد.

- هیچ کدام را اعلان نکنید. در این حالت نوع داده ی متغیر بر روی any قرار گرفته و مقدار آن برابر با تعریف نشده می شود.

در جدول زیر همان طور که در بالا بحث شد، سینتکس معتبر برای اعلان متغیر نشان داده شده است.

ردیف	سینتکس اعلان متغیر و توضیحات آن
1	var name:string = ”mary” متغیر مقداری از نوع نوشته را ذخیره می کند.
2	var name:string; متغیر، یک متغیر رشته ای است. به صورت پیش فرض مقدار متغیر بر روی تعریف نشده تنظیم می شود.
3	var name = ”mary” نوع متغیر از نوع داده ی متغیر استنتاج می شود. در اینجا متغیر از نوع رشته ای است.
4	var name; نوع داده ی متغیر any است. مقدار آن به صورت پیش فرض بر روی تعریف نشده تنظیم می شود.

مثال : متغیرها در تایپ اسکریپت

```
var name:string = "John";
var score1:number = 50;
var score2:number = 42.50
var sum = score1 + score2
console.log("name"+name)
console.log("first score: "+score1)
console.log("second score: "+score2)
console.log("sum of the scores: "+sum)
```

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

```
//Generated by typescript 1.8.10
var name = "John";
var score1 = 50;
```

```
var score2 = 42.50;
var sum = score1 + score2;
console.log("name" + name);
console.log("first score: " + score1);
console.log("second score : " + score2);
console.log("sum of the scores: " + sum);
```

خروجی برنامه بالا را می توانید در زیر مشاهده کنید.

```
name:John
first score:50
second score:42.50
sum of the scores:92.50
```

اگر مقداری را به متغیری بدهید که نوع آن ها یکسان نباشد، در این صورت کامپایلر تایپ اسکریپت خطا می دهد. به همین دلیل است که تایپ اسکریپت از Strong Typing پیروی می کند. سینتکس Strong Typing مطمئن می شود که نوع های مشخص شده در یکی از طرفین عملگر تخصیص (=) یکسان هستند. به همین دلیل است که از کد زیر خطای کامپایلی حاصل می شود.

```
var num:number = "hello"    // will result in a compilation error
```

آموزش Type Assertion در تایپ اسکریپت(TypeScript)

تایپ اسکریپت این امکان را فراهم می کند تا بتوان نوع متغیر را از حالتی به حالت دیگر تغییر داد. تایپ اسکریپت این فرآیند را *Type Assertion* می نامد. سینتکس این فرآیند به این صورت است که باید نوع مطلوب را بین علامت های <> گذاشته و آن را در مقابل متغیر یا عبارت قرار داد. در مثال زیر چگونگی انجام این کار نشان داده شده است:

مثال :

```
var str = '1'
var str2:number = <number> <any> str   //str is now of type number
console.log(str2)
```

اگر در Visual Studio Code نشانگر موس را بر روی عبارت type assertion ببرید، در این صورت این عبارت تغییر در نوع داده ی متغیر را نشان می دهد. اساسا در صورتی عمل assertion را می توان از نوع S به T انجام داد، که یا S زیرنوعی (subtype) از T باشد یا برعکس T زیرنوعی از S باشد.

دلیل اینکه به این فرآیند "type casting" گفته نمی شود، این است که casting معمولا به نوعی از runtime support اطلاق می شود. این در حالی است که "type assertions" صرفا یک ساختار compile time بوده و روشی است که با کمک آن می توان به کامپایلر تفهیم کرد که می خواهید کدتان چگونه توسط کامپایلر آنالیز شود.

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

```
//Generated by typescript 1.8.10
var str = '1';
var str2 = str;    //str is now of type number
console.log(str2);
```

و خروجی زیر تولید می شود.

```
1
```

آموزش تایپینگ استنتاجی در تایپ اسکریپت(TypeScript)

درست است که تایپ اسکریپت اصطلاحا strongly typed محسوب می شود. اما این ویژگی اختیاری است. تایپ اسکریپت از تایپینگ پویای متغیرها نیز پشتیبانی می کند. این یعنی در تایپ اسکریپت می توان متغیر را بدون هیچ نوعی اعلان کرد. در چنین مواردی کامپایلر بر اساس مقدار اختصاص داده شده به متغیر، نوع متغیر را مشخص می کند. تایپ اسکریپت اولین باری که متغیر داخل کد استفاده شده است را پیدا می کند، نوع مقدار اولیه ی این متغیر را تعیین می کند و سپس نوع همین متغیر را برای دیگر بخش های کدتان در نظر می گیرد.

این موضوع در کد زیر نشان داده شده است.

مثال : تایپینگ استنتاجی

```
var num = 2;   // data type inferred as  number
console.log("value of num "+num);
num = "12";
console.log(num);
```

در کد بالا :

- متغیری اعلان می شود و مقدار آن بر روی 2 تنظیم می شود. توجه داشته باشید که اعلان متغیر باعث مشخص شدن نوع داده نمی شود. از همین روی برنامه برای تعیین نوع دادهی متغیر از تایپینگ استنتاجی استفاده می کند. به بیان بهتر برنامه نوع اولین مقداری که متغیر بر روی آن تنظیم شده است را برای بخش های دیگر تخصیص می دهد. در این حالت نوع num بر روی عدد تنظیم شده است.
- زمانی که کد سعی می کند مقدار متغیر را بر روی رشته قرار دهد، در این صورت با توجه به اینکه نوع متغیر پیش از این بر روی عدد تنظیم شده است، کامپایلر خطا می دهد.

خروجی زیر نمایش داده می شود:

```
error TS2011: Cannot convert 'string' to 'number'.
```

آموزش حیطه ی متغیر(Scope Variable) در تایپ اسکریپت(TypeScript)

حیطه ی یک متغیر، مکان تعریف شدن متغیر را مشخص می کند. در دسترس بودن یک متغیر داخل برنامه توسط حیطه ی آن مشخص می شود. حیطه ی متغیرهای تایپ اسکریپت عبارتند از:

- حیطه ی جهانی (Global Scope) : متغیرهای جهانی خارج از ساختارهای برنامه نویسی اعلان می شوند. به این متغیرها می توان از هرجایی از کدتان دسترسی داشت.
- حیطه ی کلاس (Class Scope) : به این متغیرها field هم گفته می شود. این متغیرها داخل کلاس اما خارج از متدها اعلان می شوند. به این متغیرها می توان با استفاده از شیء کلاس دسترسی داشت. Field ها می توانند استاتیک هم باشند. به Field های استاتیک می توان با استفاده از اسم کلاس دسترسی داشت.
- حیطه ی محلی (Local Scope) : متغیرهای محلی همان طور که از اسم آن ها پیداست، داخل ساختارهایی مانند متدها، حلقه ها و .. اعلان می شوند. به این متغیرهای تنها می توان از داخل ساختاری که اعلان شده اند، دسترسی پیدا کرد.

در مثال زیر حیطه های متغیر در تایپ اسکریپت نشان داد شده است.

مثال : حیطه ی متغیر

```
var global_num = 12       //global variable
class Numbers {
  num_val = 13;         //class variable
  static sval = 10;      //static field

  storeNum():void {
    var local_num = 14;   //local variable
  }
}
console.log("Global num: "+global_num)
console.log(Numbers.sval)  //static variable
var obj = new Numbers();
console.log("Global num: "+obj.num_val)
```

بعد از انجام transpiling کردن، کد جاوا اسکریپت زیر ایجاد می شود.

```
var global_num = 12;          //global variable
var Numbers = (function () {
  function Numbers() {
```

```
    this.num_val = 13;        //class variable
  }
  Numbers.prototype.storeNum = function () {
    var local_num = 14;       //local variable
  };
  Numbers.sval = 10;          //static field
  return Numbers;
}());

console.log("Global num: " + global_num);
console.log(Numbers.sval);      //static variable

var obj = new Numbers();
console.log("Global num: " + obj.num_val);
```

و خروجی زیر نمایش داده می شود.

```
Global num: 12
10
Global num: 13
```

اگر سعی کنید که از خارج از متد به متغیر محلی دسترسی پیدا کنید، خطای کامپایل زیر نمایش داده می شود.

```
error TS2095: Could not find symbol 'local_num'.
```