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

Interfaceیا رابط قراردادی سینتکسی است که هویت ها باید از آن پیروی کنند. به بیان دیگر، یک رابط سینتکسی را تعریف می کند که هویت ها باید از آن پیروی کنند.

رابط ها مشخصه ها، متدها و رویدادها را تعریف می کنند. که این ها اعضای رابط محسوب می شوند. رابط ها تنها می توانند شامل اعلان این اعضا باشند. تعریف کردن این اعضا بر عهده ی کلاس حاصل شده (deriving class) می باشد. رابط معمولا سازه ی استانداردی را فراهم می کند، به گونه ای که کلاس های حاصل شده از آن پیروی می کنند.

بیایید شیئی را در نظر بگیریم.

```
var person = {
  FirstName:"Tom",
  LastName:"Hanks",
  sayHi: ()=>{ return "Hi"}
};
```

امضای این شیء می تواند به صورت زیر باشد.

```
{
  FirstName:string,
  LastName:string,
  sayHi()=>string
}
```

برای اینکه بتوانیم این امضا را در سراسر اشیاء مجددا استفاده کنیم، می توانیم آن را به عنوان یک رابط تلقی کنیم.

آموزش اعلان رابط ها در TypeScript

برای اعلان یک رابط از عبارت کلیدی interface استفاده می شود. سینتکس مورد نیاز برای انجام این کار در زیر آمده است.

سینتکس

```
interface interface_name {
}
```

مثال : رابط و اشیاء

```
interface IPerson {

  firstName:string,

  lastName:string,

  sayHi: ()=>string

}


var customer:IPerson = {

  firstName:"Tom",

  lastName:"Hanks",

  sayHi: ():string =>{return "Hi there"}

}


console.log("Customer Object ")

console.log(customer.firstName)

console.log(customer.lastName)

console.log(customer.sayHi())


var employee:IPerson = {

  firstName:"Jim",

  lastName:"Blakes",

  sayHi: ():string =>{return "Hello!!!"}

}
```

```
console.log("Employee  Object ")

console.log(employee.firstName) console.log(employee.lastName)
```

در مثال بالا، یک رابط تعریف شده است. نوع شیء customer، IPerson است، که از این به بعد برای تعریف تمامی مشخصه ها براساس چیزی که رابط مشخص کرده است، به object مقید خواهد شد.

شیء دیگری که امضای آن در زیر آمده است نیز همچنان IPerson در نظر گرفته می شود. زیرا این شیء براساس اندازه یا امضا مدیریت می شود.

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

```
//Generated by typescript 1.8.10

var customer = { firstName: "Tom", lastName: "Hanks",

  sayHi: function () { return "Hi there"; }

};

console.log("Customer Object ");

console.log(customer.firstName);

console.log(customer.lastName);

console.log(customer.sayHi());

var employee = { firstName: "Jim", lastName: "Blakes",

  sayHi: function () { return "Hello!!!"; } };


console.log("Employee  Object ");

console.log(employee.firstName);

console.log(employee.lastName);
```

و خروجی به صورت زیر نمایش داده می شود.

```
Customer object
Tom
Hanks
Hi there
Employee  object
Jim
```

```
Blakes
Hello!!!
```

رابط ها به جاوا اسکریپت تبدیل نمی شوند. بلکه بخشی از تایپ اسکریپت هستند. اگر شما به اسکرین شات TS Playground tool نگاه کنید، برخلاف کلاس زمانی که شما رابطی را اعلان می کنید، خبری از جاوا اسکریپت نیست. به همین دلیل رابط ها هیچ تاثیری در زمان اجرای (runtime) جاوا اسکریپت ندارند.

آموزش انواع یونیون و رابط در TypeScript

در مثال زیر نوع یونیون و رابط در کنار هم استفاده شده اند.

```
interface RunOptions {

  program:string;

  commandline:string[]|string|(()=>string);

}


//commandline as string
```

```
var options:RunOptions = {program:"test1",commandline:"Hello"};

console.log(options.commandline)


//commandline as a string array

options = {program:"test1",commandline:["Hello","World"]};

console.log(options.commandline[0]);

console.log(options.commandline[1]);


//commandline as a function expression

options = {program:"test1",commandline:()=>{return "**Hello World**";}};


var fn:any = options.commandline;

console.log(fn());
```

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

```
//Generated by typescript 1.8.10

//commandline as string

var options = { program: "test1", commandline: "Hello" };

console.log(options.commandline);


//commandline as a string array

options = { program: "test1", commandline: ["Hello", "World"] };

console.log(options.commandline[0]);

console.log(options.commandline[1]);


//commandline as a function expression
```

```
options = { program: "test1", commandline: function () { return "**Hello World**"; } };

var fn = options.commandline;

console.log(fn());
```

و خروجی به صورت زیر نمایش داده می شود.

```
Hello
Hello
World
**Hello World**
```

آموزش رابط ها و آرایه ها در TypeScript

رابط ها می توانند هم نوع کلیدی که آرایه ها استفاده می کنند و هم نوع ورودی ای که آرایه ها در برمی گیرند را تعریف کنند. نوع ایندکس می تواند رشته یا عدد باشد.

مثال

```
interface namelist {

  [index:number]:string

}



var list2:namelist = ["John",1,"Bran"] //Error. 1 is not type string

interface ages {

  [index:string]:number

}



var agelist:ages;

agelist["John"] = 15   // Ok

agelist[2] = "nine"   // Error
```

آموزش رابط ها و وراثت در TypeScript

یک رابط را می توان با استفاده از رابط های دیگر توسعه داد. به بیان دیگر، رابط ها می توانند وارث رابط های دیگر باشند. تایپ اسکریپت این امکان را فراهم می کند تا یک رابط وارث چندین رابط دیگر باشد.

برای پیاده کردن وراثت بین رابط ها از عبارت کلیدی extends استفاده می کنیم.

سینتکس: وراثت در یک رابط

```
Child_interface_name extends super_interface_name
```

سینتکس : وراثت بین چند رابط

```
Child_interface_name extends super_interface1_name,
super_interface2_name,...,super_interfaceN_name
```

مثال : وراثت رابط ساده

```
interface Person {

  age:number

}



interface Musician extends Person {

  instrument:string

}



var drummer = <Musician>{};

drummer.age = 27

drummer.instrument = "Drums"

console.log("Age: "+drummer.age) console.log("Instrument: "+drummer.instrument)
```

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

```
//Generated by typescript 1.8.10
```

```
var drummer = {};

drummer.age = 27;

drummer.instrument = "Drums";

console.log("Age: " + drummer.age);

console.log("Instrument: " + drummer.instrument);
```

و خروجی به صورت زیر نمایش داده می شود.

```
Age: 27
Instrument: Drums
```

مثال : وراثت بین چند رابط

```
interface IParent1 {

  v1:number

}


interface IParent2 {

  v2:number

}


interface Child extends IParent1, IParent2 { }

var lobj:Child = { v1:12, v2:23}

console.log("value 1: "+this.v1+" value 2: "+this.v2)
```

نوع شیء lobj، interface leaf است. به دلیل عمل وراثت اکنون interface leaf به ترتیب دارای دو attribute، v1 و v2 است. به همین دلیل شیء lobj باید دارای این attribute ها باشد.

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

```
//Generated by typescript 1.8.10

var lobj = { v1: 12, v2: 23 };
```

```
console.log("value 1: " + this.v1 + " value 2: " + this.v2);
```

و خروجی به صورت زیر نمایش داده می شود.

```
        value 1: 12  value 2: 23
```