

Typescript

UI-Angular



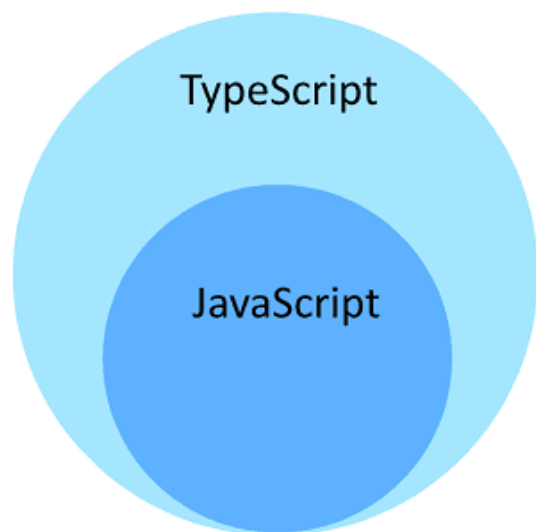
מה נלמד היום?

- Introduction to Typescript
- Typescript vs. Javascript
- Variable Definition
- Data Types
 - Primitives
 - Objects



Typescript

- Superset של JavaScript, מעטפת
- שיפור של JavaScript מבחינה תחבירית, קל יותר להבנה ותכנות
- כל קוד JavaScript הוא גם קוד Typescript
- קוד פתוח (open source) המפותח ע"י Microsoft
- קבצי Typescript יישמרו בסיומת ts או tsx




Typescript

דוקומנטציה רשמית:

- <https://www.typescriptlang.org/docs>

Typescript

 TypeScript

Download Docs Handbook Community Playground Tools

Search Docs

Get Started >

Handbook >

The TypeScript Handbook

Basic Types

Interfaces

Functions

Literal Types

Unions and Intersection Types

Classes

Enums

Generics

Handbook Reference >

Tutorials >

What's New >

Basic Types

For programs to be useful, we need to be able to work with some of the simplest units of data: numbers, strings, structures, boolean values, and the like. In TypeScript, we support the same types as you would expect in JavaScript, with an extra enumeration type thrown in to help things along.

Boolean

The most basic datatype is the simple true/false value, which JavaScript and TypeScript call a **boolean** value.

```
let isDone: boolean = false;
```

Number

On this page

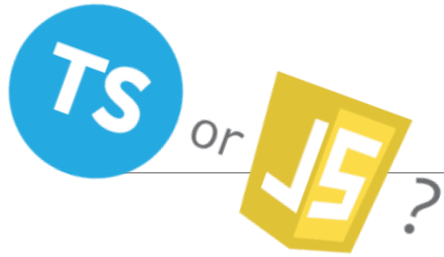
Boolean
Number
String
Array
Tuple
Enum
Unknown
Any
Void
Null and Undefined
Never
Object
Type assertions
A note about let
About Number, String, Boolea...

Is this page helpful?

Yes No

המגמה בפיתוח web

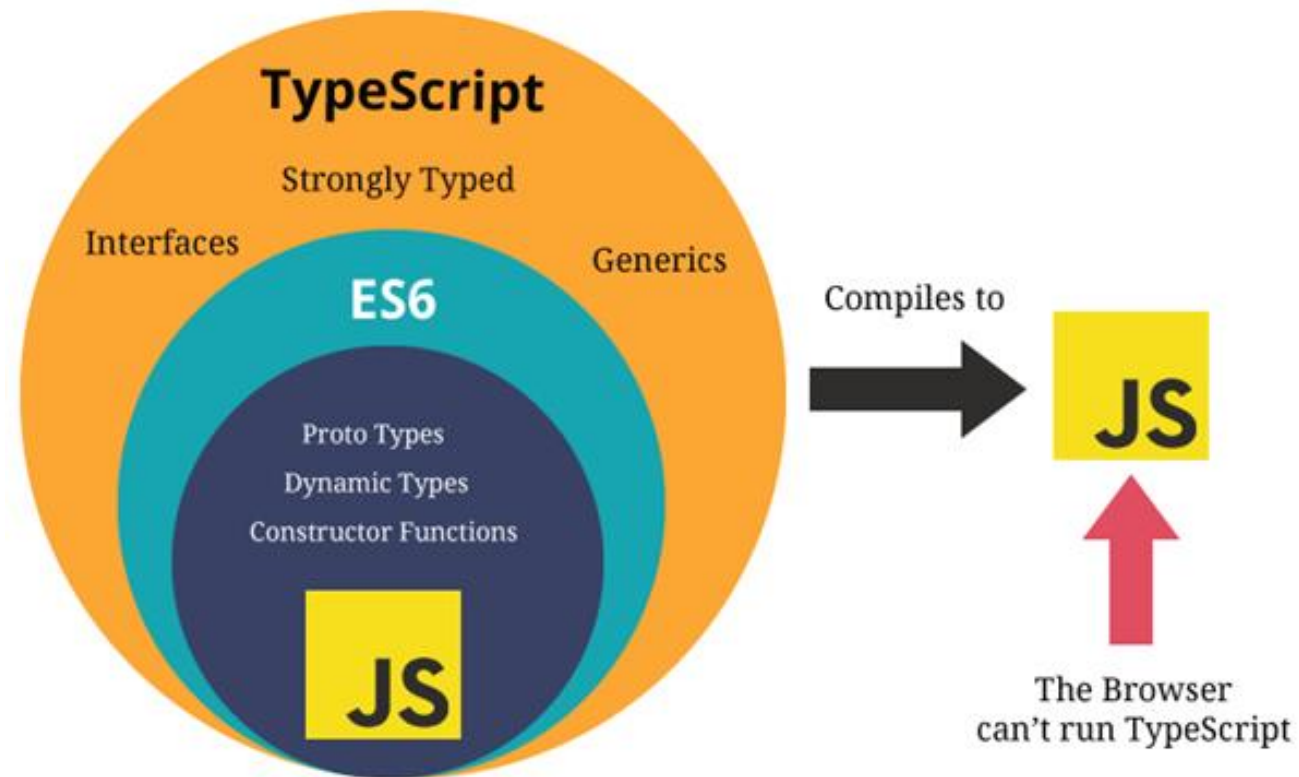




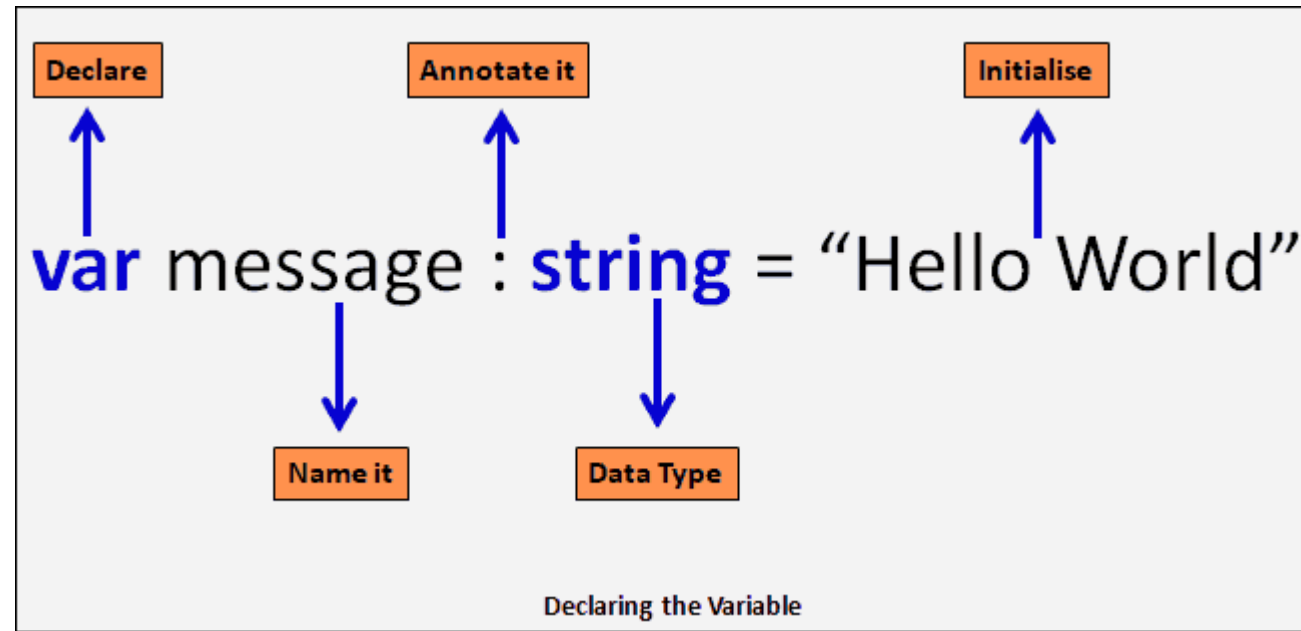
TypeScript vs. Javascript

	TS	JS
Compilation Needed	Yes	No
Project Size Compatibility	large	small
Static Data Types	number, string, array,...	-
Fully Object-Oriented	Yes	No
Server Side/ Client Side	Client	Both

Typescript vs. Javascript



Variable Definition



Variable Definition

- **let / var** - define any variable, with or without type or initial value
- **const** - initialize a constant whose value does not change

```
var width:number = 100;  
let height:number = 200;  
const key:string = 'abc123'
```

```
var width  
let height  
const key = 'abc123'
```

Variable Definition

- מבחינת תחביר, הכרזה על משתנה באמצעות `let` זהה ל- `var`
- ההבדל טמון בסמנטיקה: משתנה שהוגדר עם `let` המילה `let` יתקיים אך ורק בתוך המרחב (scope) של מבנה הנתונים בו הוא נמצא.

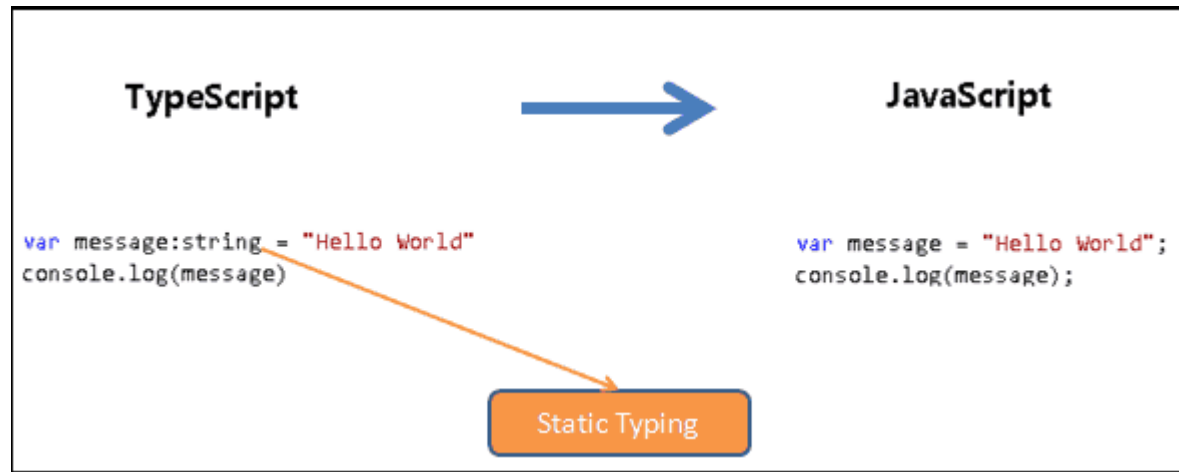
```
for(let x = 1; x<10; x++) {  
  console.log("number " + x);  
}  
console.log(x); // undefined
```

```
for(var x = 1; x<10; x++) {  
  console.log("number " + x);  
}  
console.log(x); // 10
```

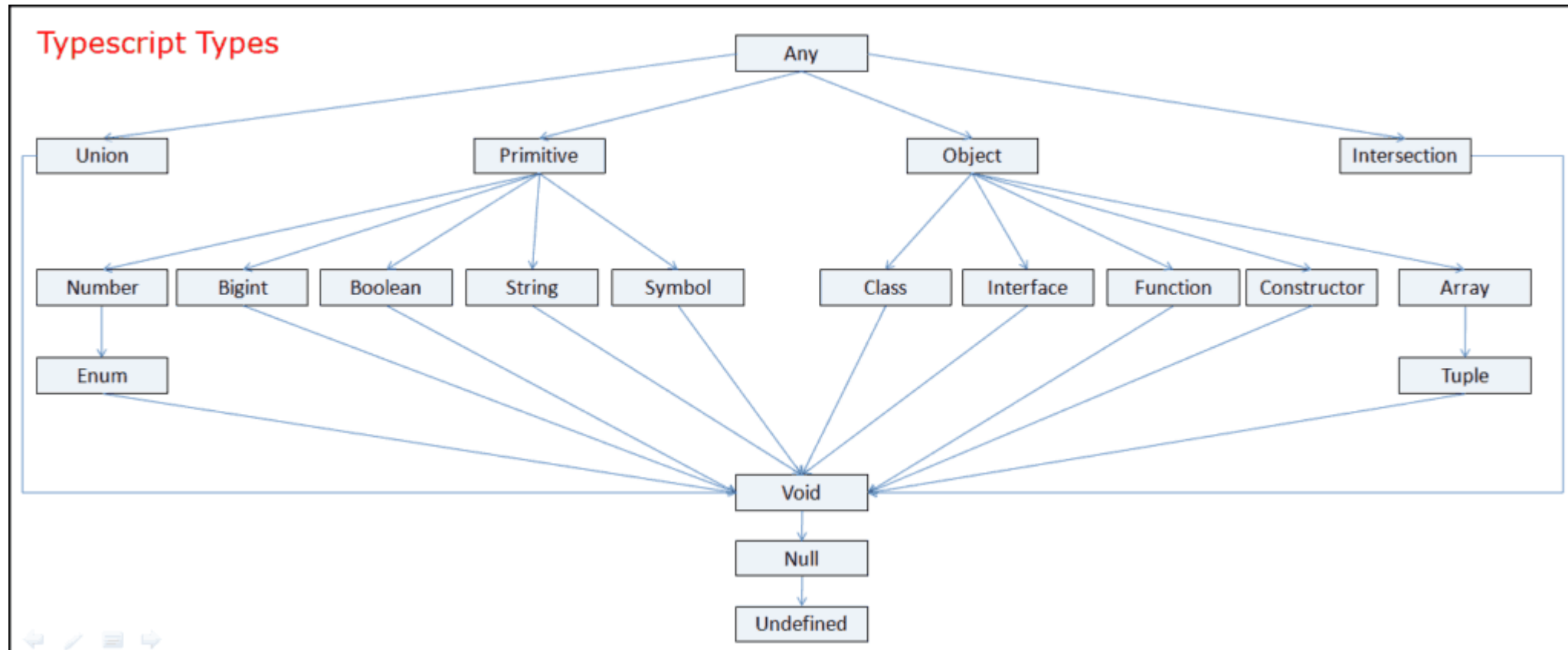
קמפול ל-js

```
var message : string = "Hello World"  
console.log(message)
```

מה קורה כאשר מקמפלים (ng serve)?



Data Types



Any

- יכול להכיל את כל טיפוסי הנתונים, נשתמש כאשר אנחנו לא יכולים לדעת מראש מהו טיפוס הנתונים
- ייחודי ל-TypeScript
- סוג נתונים שמאפשר "לבטל" את קביעת סוג הנתונים הסטטית של המהדר.

Any

```
let notSure: any = 4    //defined as any  
notSure = "maybe a string instead"  
notSure = false
```

```
let stillNotSure        //No Type specified. inferred as any  
stillNotSure = 5  
stillNotSure = "may be a string instead"  
stillNotSure = false
```

string - מחרוזת

1

```
let message : string = "Hello World"  
let color = "red"
```

```
let message : string = 'Hello World'  
let color = 'red'
```

2

```
let sentence: string = `Hello, welcome to the world of typescript,  
the typed super of javascript`
```

בניגוד לשפות כדוגמת Java ובדומה ל – PHP, Python לא קיים הבדל בין גרשיים כפולים (") או גרש אחד (').
כלומר, אין סוג נתונים character (תו)

boolean – משתנה בוליאני

```
let isDone: boolean = true           // primitive boolean type
let isPending:boolean = false        // primitive boolean type

console.log(isDone)                   //output true
console.log(isPending)                //output false

console.log(typeof(isDone))           //output boolean

**** output ****
true
false
boolean
```

סוג נתונים שיכול לקבל שני ערכים: true או false
בניגוד לשפות מסויימות (לדוגמה: Python, C) לא ניתן להשתמש
ב- 1 או 0 כתחליפים.

number - מספר

1

```
let numVal = 100
```

2

```
let numVal : number = 100
```

3

```
let numVal = Number(100)
```

בניגוד לשפות אחרות, אין הפרדה בין מספרים שלמים (integers) למספרים ממשיים (floats) בסוג הנתונים הזה ניתן ליישם ערך ממשי, הקסדצימלי, בינארי או אוקטלי.

enum

סוג נתונים המאפשר לתת לקבוצת ערכים מספריים שמות "ידידותיים".
המהדר נותן לכל איבר ערך מספרי, כשהאיבר הראשון מקבל ערך 0

```
enum Color {  
  Red,  
  Green,  
  Blue,  
}
```

```
let c: Color = Color.Green
```

```
let r: Color = Color[0]      // red
```

Array

כאשר נגדיר מערך, תמיד נצמיד אליו סוג נתונים נוסף: מערך בוליאנים, מערך מחרוזות, מערך מספרים וכו'.

```
var cities: string[] = ['Delhi', 'New York', 'London']
```

//OR

```
var cities: Array<string> = ['Delhi', 'New York', 'London']
```

```
let list: number[] = [1, 2, 3];
```

//OR

```
let list: Array<number> = [1, 2, 3];
```

```
coursesList : any = [  
  {number:292100, name:"Java 1"},  
  {number:292101, name:"Java 2"},  
  {number:292102, name:"Python"},  
  {number:292103, name:"Angular 1"},  
  {number:292104, name:"Angular 2"},  
  {number:292105, name:"React"},  
  {number:292106, name:"MongoDB"},  
  {number:292107, name:"Sql"}  
]
```

Tuple

Tuple זוהי דרך להגדיר רשומה כלשהי. רצף של ערכים עם משמעות למיקום וסדר. ניתן להכניס טיפוסים נתונים שונים ברשומה אחת.

```
let x: [string, number]
```

```
// Initialize it
```

```
x = ["hello", 10] // OK
```

```
// Initialize it incorrectly
```

```
x = [10, "hello"] // Error
```

String

```
var strPrim : string = "I am primitive"    //strPrim is a string primitive type
var strObj= new String("I am a object")    //strObj is a String object

strPrim=strObj;    //Compiler error here.
```

**** Error ***

Type 'String' is not assignable to type 'string'.

'string' is a primitive, but 'String' is a wrapper object. Prefer using 'string' when possible.ts(2322)

String

```
var strPrim = "I am primitive"           //strPrim is a string primitive type
var strObj= new String("I am a object")   //strObj is a String object

strObj=strPrim;    //ok
```


Number

```
let num = new Number(100)  
console.log(num)
```

```
//[Number: 100]
```

```
/**output***/  
//[Number: 100]
```

Number vs. number

```
let numValue=100           //primitive number  
let numObject = new Number(1500) //Number object
```

```
numValue=numObject        //Compile Error  
numObject=numValue        //ok
```

Function

TS

```
function add(x: number, y: number): number {  
  return x + y  
}
```

TS / JS

```
function add(x, y) {  
  return x + y  
}
```

בתוך מחלקה (מחלקת קומפוננטה למשל) אין צורך להשתמש במילה `function`

```
class Greeter {  
  
  greeting: string;  
  
  constructor(message: string) {  
    this.greeting = message;  
  }  
  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}  
  
let greeter = new Greeter("world");
```

בתוך מחלקה (מחלקת קומפוננטה למשל) אין צורך להשתמש ב-let/var

משפט תנאי

```
let x: number = 10, y = 20
```

```
if (x > y)
{
    console.log('x is greater than y.')
}
else if (x < y)
{
    console.log('x is less than y.') //This will be executed
}
else if (x == y)
{
    console.log('x is equal to y')
}
```

לולאת For

מעבר על אינדקסים

```
let myArray: Array<number> = [100, 200, 300]
for(let item in myArray){
    console.log(item) // 0, 1, 2
}
```

לולאת For

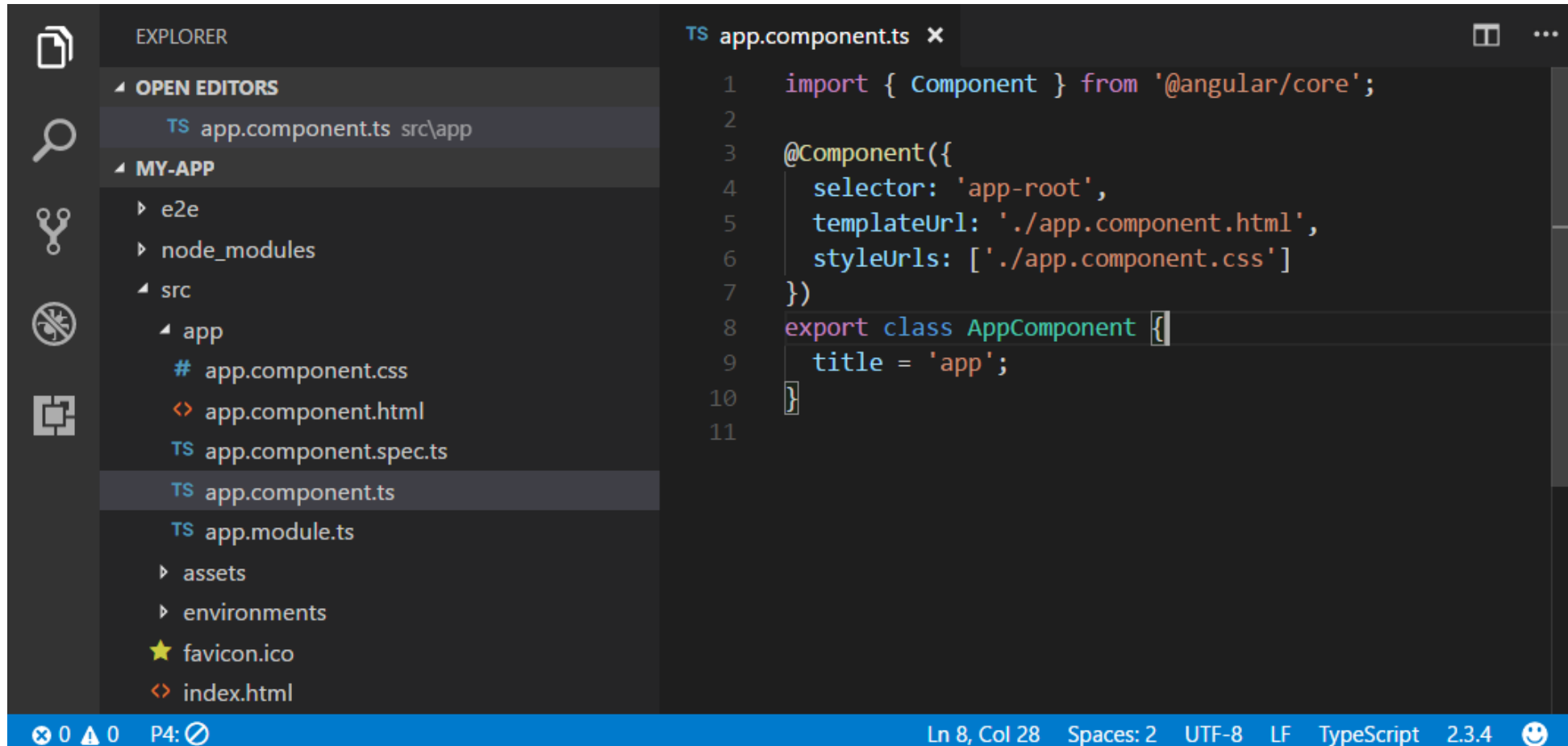
מעבר על ערכים (איברים)

```
let myArray: Array<number> = [100, 200, 300]
for(let item of myArray){
    console.log(item) // 100, 200, 300
}
```

Angular with TS



Angular with TS



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'MY-APP' with a 'src' directory containing an 'app' subdirectory. The 'app' directory lists several files: 'app.component.css' (marked with a '#'), 'app.component.html' (marked with '<>'), 'app.component.spec.ts' (marked with 'TS'), 'app.component.ts' (marked with 'TS' and selected), and 'app.module.ts' (marked with 'TS'). Below these are 'assets', 'environments', 'favicon.ico' (marked with a star), and 'index.html' (marked with '<>'). The main editor area shows the 'app.component.ts' file with the following TypeScript code:

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'app';
10 }
11
```

The status bar at the bottom indicates 'Ln 8, Col 28', 'Spaces: 2', 'UTF-8', 'LF', 'TypeScript', and '2.3.4'.