

[Beshoy Morad] [[Link](#)]
Individually

TypeScript is a superset of JavaScript. It means that TypeScript provides all the features and functionalities of JavaScript with some added features.

The benefits of TypeScript include:

1) **Compilation:** JavaScript is an interpreted language. Hence, it needs to be run to test that it is valid, in case there is an error. Hence, you have to spend hours trying to find bugs in the code.

The TypeScript transpiler provides the error-checking feature. TypeScript will compile the code and generate compilation errors, if it finds some sort of syntax errors. This helps to highlight errors before the script is run and will speed up the process.

2) **Strong Static Typing:** JavaScript is not strongly typed. TypeScript comes with an optional static typing and type inference system through the TLS (TypeScript Language Service). The type of a variable, declared with no type, may be inferred by the TLS based on its value.

So, to wrap up:

Advantage of TypeScript over JavaScript

- 1) TypeScript always highlights errors at compilation time during the time of development, whereas JavaScript points out errors at the runtime.
- 2) TypeScript supports strongly typed or static typing, whereas this is not in JavaScript.
- 3) It has a namespace concept by defining a module.

Disadvantage of TypeScript over JavaScript

- 1) TypeScript takes a long time to compile the code.
- 2) TypeScript does not support abstract classes.

3) If we run the TypeScript application in the browser, a compilation step is required to transform TypeScript into JavaScript.

[Khaled Abdelrahman] [[Link](#)]
Team

Typescript makes some tests unnecessary.How?

Sometimes users might pass parameters incorrectly so instead of expecting what parameters users pass and test inputs & outputs typescript actually solve this problem. some components may use API incorrectly typescript solve this also.

Typescript can run in any browser or any js engine and it supports namespace concepts(used for logical grouping including interfaces, classes, functions and variables).

[Ahmed M.Osman] [[Link](#)]
Individually

The Question: **What are the features of Typescript beyond static typing?**

1- Type Inference, which gives some of the benefits of types, without actually using them.

If we did not assign a type to a variable, typescript will assign the type by itself. This assignment will prevent the error coming from reassigning the variable to a different type.

Example:

```

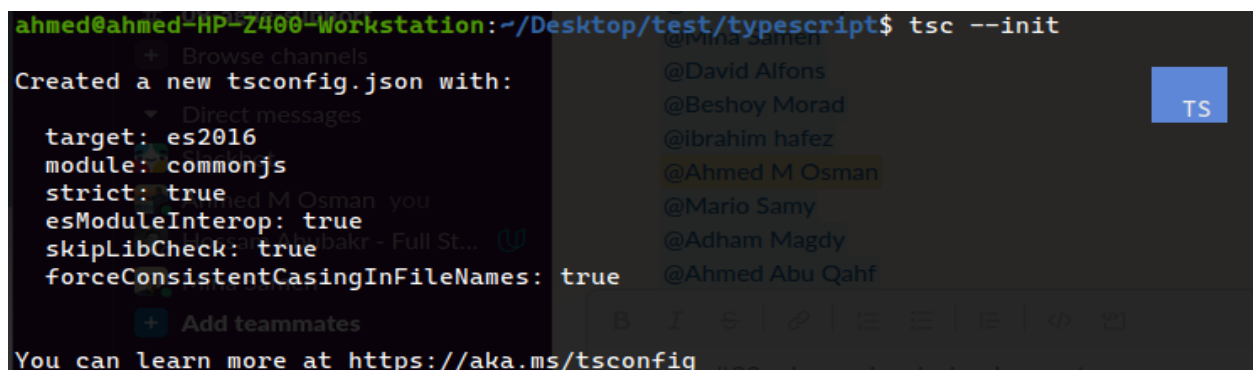
1
2 let Name = "Ahmed M.Osman"
3 let NumberOne = 1
4 let NumberTwo = 2
5
6
7 console.log(typeof Name) // ==> output : string.
8 console.log(typeof NumberOne) // ==> output : number.
9 console.log(typeof NumberTwo) // ==> output : number.
10
11 function sumTwoNum(a,b) { // ==> gives warning that a and b is assing to any which mean can be anytype
12     console.log(a + b)
13 }
14
15 sumTwoNum(NumberOne, NumberTwo) // ==> output : 3
16
17 NumberOne = "Ahmed" // ==> Error: string is not assingabel to type number.
18

```

2- Access to ES6 and ES7 features, before they become supported by major browsers and The ability to compile down to a version of JavaScript that runs on all browsers.

If we look at the tsconfig.json file we can see that there is a target Javascript. Which means we could choose any version that we want (ES6 or ES7). Also we can choose the older version of TS (ES5). When initiating the tsconfig file, the target and module is represented to the terminal.

Example:



```

ahmed@ahmed-HP-Z400-Workstation:~/Desktop/test/typescript$ tsc --init
Created a new tsconfig.json with:
  target: es2016
  module: commonjs
  strict: true
  esModuleInterop: true
  skipLibCheck: true
  forceConsistentCasingInFileNames: true
You can learn more at https://aka.ms/tsconfig

```

[Mario Samy] [[Link](#)]
Individually

What are the features of Typescript beyond static typing?

Aside from the performance benefits you get by letting the compiler know the types of values beforehand and preventing certain glitches, Other features also include:

1- TypeScript is just JavaScript.

Typescript adopts the basic building blocks of your program from JavaScript. Hence, you only need to know JavaScript to use TypeScript.

2- TypeScript supports other JS libraries.

We can easily use all of the JavaScript frameworks, tools, and other libraries.

3- TypeScript is portable.

TypeScript is portable across browsers, devices, and operating systems.

TypeScript doesn't need a dedicated VM or a specific runtime environment to execute.

4- TypeScript supports Object Oriented Programming concepts like classes, interfaces, inheritance, etc.

5- DOM Manipulation: TypeScript can be used to manipulate the DOM similar to JS.

[Mina Sameh][[Link](#)]

Individually

[Ibrahim Hafez Abdallah][[Link](#)]

Team

What are the features of Typescript beyond static typing?

- Better performance than JavaScript as it prevents certain glitches from happening thanks to the compiler knowing what is the type value.
- It will give you a compiler error. That means you'll know the error is there before you even run the program so you don't need to try to hunt it down and waste a couple of hours debugging the issue.
- You can use Enums, Interfaces and Literal types in TypeScript.
- TypeScript doesn't need a dedicated VM or a specific runtime environment to execute
- TypeScript is aligned with the ECMAScript6 specification
- TypeScript can use all frameworks and libraries from JavaScript

What are the features of Typescript beyond static typing?

Ahmed Hamdy [[Link](#)]

[Solo]

1- What are the features of Typescript beyond static typing?

- Variables can be assigned a type at **compile time** (before the program run) and the compiler can therefore deduce the types of expressions using those variables
- If **a** and **b** are declared as number, then the compiler concludes that **a + b** is also **number**
- You can find many **errors** While you're still typing
- **Static-checking**, which mean checking for debugging at compile time bugs are the bane of programming, static checking is the first idea that we've seen for these bugs caused by applying an operation to the wrong types of arguments. If we write a broken line of code like : **"5" * "6"** that tries to multiply two strings, then **static typing** will catch this **error** while we're still programming, rather than waiting until the line is

reached during **execution** this also example:

```
// name of author
const Author : string = "\\udacity\\", NameOfCourse:string="Advance_Web_Developer";
//number of courses
const NumberOfCourses:number=1;
//vaild permission
const applied:boolean=true;
// Student anonymous function
const StudentOfUdacity = ( courses_taken : Array<string> ,StudetName:string,id:number,add:string)=>{
    //user information
    const userDetails = { identify:id <= 0 && "error zero at least be one",BunchOfCourses:courses_taken,name :StudetName}
    //adding courses
    userDetails.BunchOfCourses.push(add);
    console.log(` id |${userDetails.identify} |\\n| Courses:${userDetails.BunchOfCourses} |\\n| name:${userDetails.name}|\\n|`);
}
// let C out This fool!
StudentOfUdacity(["nodejs","javascript"],"Ahmed Hamdy",-1,"PHP");

// -----error-----;

// the bug is found automatically before the program even runs is this why Static type it's useful
// let's make error Argument of type 'number' is not assignable to parameter of type 'string'
StudentOfUdacity(["nodejs","javascript"],"Ahmed Hamdy",-1,1);
```

- **Typescript** is a **gradually-typed superset** of JavaScript language, this mean that every JavaScript program is a typescript program and that types can be optionally a dynamic **map** type, **function** (closure) type. **Class** type, **method** access, **array** (collection) type, there is also **typeannotation** in **swift** and it's similar to typescript as well as **Static-check**

