TypeScript Basics for Automation Testers – Day 1

Topic: Why TypeScript? How it differs from JavaScript

What is TypeScript?

TypeScript is JavaScript with added syntax for **types**.

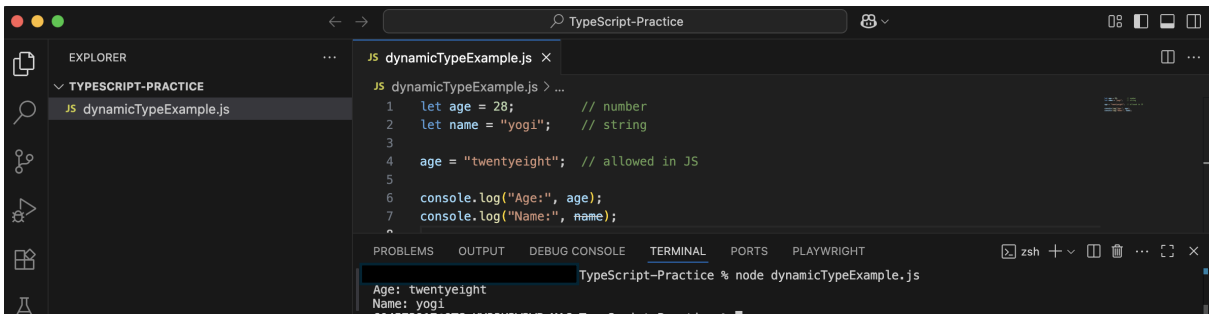
- **JavaScript** is *dynamically typed* → you don't define the type when declaring a variable.
- **TypeScript** is *statically typed* → you define the type during declaration.

JavaScript Example (Dynamically Typed)

In JavaScript, the type is decided at runtime:

```
let age = 28;          // number
let name = "yogi";     // string

age = "twentyeight";   // allowed in JS
```

Here, `age` starts as a number but later becomes a string.
JavaScript allows this flexibility — but such mistakes are caught **only during runtime**.



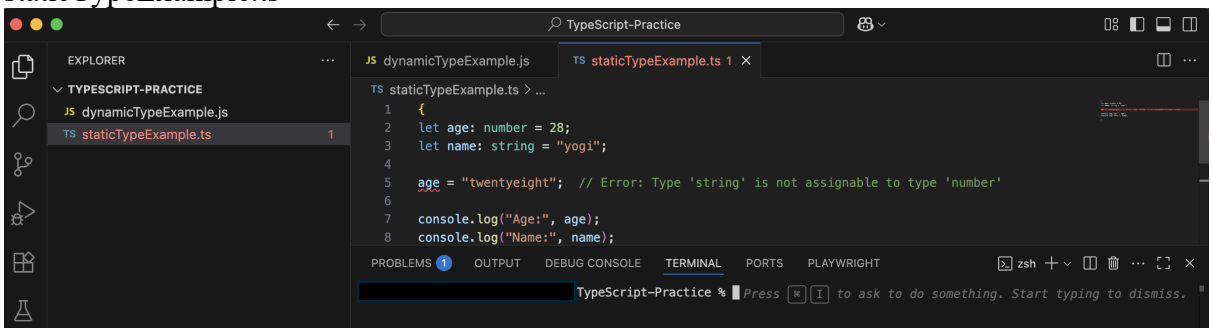
TypeScript Example (Statically Typed)

In TypeScript, you declare types explicitly:

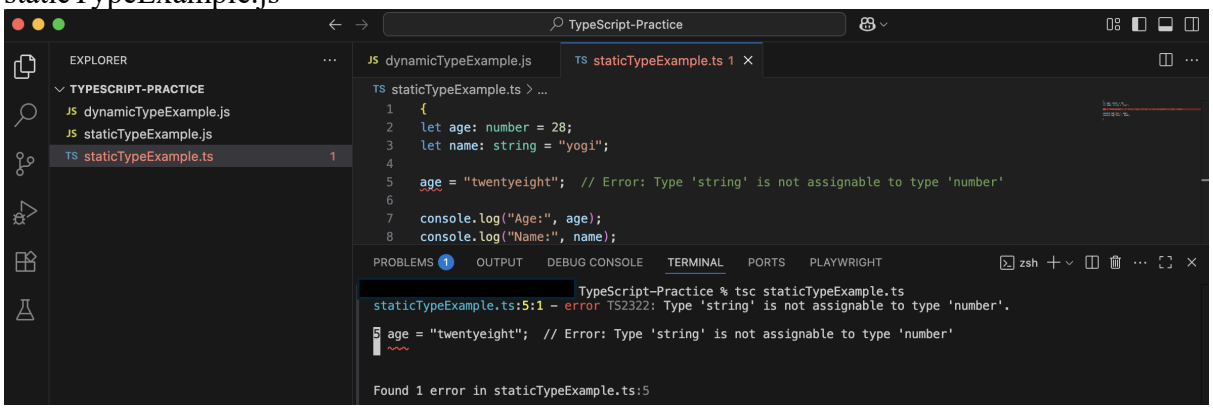
```
let age: number = 28;
let name: string = "yogi";

age = "twentyeight";  // Error: Type 'string' is not assignable to type
'number'
```

TypeScript enforces **type safety** — such errors are caught **at compile time**, before the code runs.

staticTypeExample.ts



staticTypeExample.js



Note: After running the command **tsc staticTypeExample.ts**, TypeScript automatically creates a file named **staticTypeExample.js** because the TypeScript compiler converts **.ts** files into plain JavaScript files.

Why Do We Need TypeScript?

1. Strong Typing & Early Error Detection

- In JavaScript:
  - `var x = 20;`
  - `x = "yogi"; // No error`

  Errors appear only when the code runs.

- In TypeScript:
  - `var x: number = 20;`
  - `x = "yogi"; // Compile-time error`

  This helps prevent unexpected behaviour early in development.

---

2. Superset of JavaScript

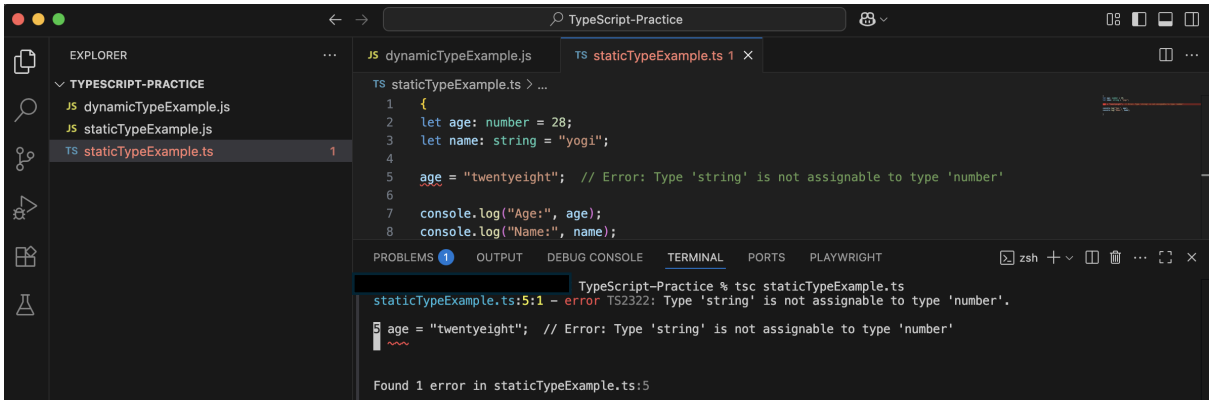
- TypeScript includes everything from JavaScript and adds more features.

- Supports OOP concepts, generics, and advanced collections.
- Whatever works in JS also works in TS — plus, you get better tooling and structure.

---

3. Compilation Process

- Browsers don't run TypeScript directly.
- TypeScript code is **compiled into JavaScript** before execution.
- Once compiled, it can run anywhere JavaScript runs.



About ECMAScript (ES)

- ECMAScript (ES3, ES5, ES6, etc.) is the official standard for JavaScript.
- Microsoft created TypeScript to add modern features without breaking these standards.
- TypeScript is a **superset** of JavaScript — built on top of ECMAScript.

Key Takeaways

- **JavaScript** → Dynamically typed and flexible, but prone to runtime errors.
- **TypeScript** → Statically typed, safer, and better for large automation frameworks.
- **Superset** → Everything in JS works in TS, plus extra features.
- **Compilation** → TypeScript always compiles to JavaScript.

Practice

1. Create a new file `example.ts` and add:

```
let message: string = "Hello TypeScript";
console.log(message);
```

2. Compile it using:

```
tsc example.ts
```

3. Check that it generated `example.js` and run it using:

```
node example.js
```

Additional Notes:

Dynamic vs Static Typing

Many of us still get confused between *static* and *dynamic* typing.
Let's first understand the words in **simple English** before linking them to code.

- **Dynamic** means *changing or flexible*.
  Something that can **change anytime** — like water taking the shape of any container.
- **Static** means *fixed or constant*.
  Something that **does not change easily** — like a rock that stays in one shape.

Now, let's connect this to programming.

1. Dynamic Typing (JavaScript)

- In **dynamic typing**, the data type of a variable is decided **at runtime**.
- You can assign any type of value to the same variable — and JavaScript won't complain.
- But the problem is, **errors appear only when you run the code** (runtime errors).

Example (JavaScript – Dynamically Typed):

```
let value = 100;       // number
value = "hundred";     // now string
value = true;          // now boolean

console.log(value);    // Output: true
```

Real-time Example (Dynamic Typing):
Imagine you're testing a login page and you store a user ID in a variable.
At first, you assign a number (like `12345`), and later by mistake you assign `"admin"` (a string).
The script still runs — but fails later during execution because JavaScript didn't warn you.

---

2. Static Typing (TypeScript)

- In **static typing**, you must declare the type **when you create the variable**.
- The compiler checks for type mismatches **before running** the code (compile-time).
- This means many bugs are caught early, before execution.

Example (TypeScript – Statically Typed):

```
let value: number = 100;
value = "hundred";  //  Error: Type 'string' is not assignable to type
'number'
```

Real-time Example (Static Typing):
In automation testing, let's say you store test result counts in a variable:

```
let totalPassed: number = 50;
```

If you accidentally assign `"fifty"`, TypeScript immediately shows an error.
This helps catch mistakes before running the tests — **saving debugging time.**

In short:

Concept	Typing Style	When Type is Checked	Example Language	Error Detected
JavaScript	Dynamic	At runtime	JavaScript	When the code runs
TypeScript	Static	At compile time	TypeScript	Before running

Questions

1. What is the main difference between JavaScript and TypeScript?
2. Does TypeScript run directly in browsers?
3. Can you use JavaScript code inside a TypeScript project?
4. Why is TypeScript preferred for automation projects?
5. What happens when you compile a `.ts` file using `tsc filename.ts`?
6. What does "superset of JavaScript" mean?
7. Can you mix static and dynamic typing in TypeScript?
8. What is ECMAScript, and how is it related to TypeScript?

Answers

1. JavaScript is dynamically typed, while TypeScript is statically typed and enforces type safety.
2. No. TypeScript must first be compiled into JavaScript.
3. Yes. Since TypeScript is a superset of JavaScript, all JS code works in TS.
4. TypeScript catches errors at compile-time, supports OOP concepts, and makes large automation frameworks easier to maintain.

5. The TypeScript compiler creates a `.js` file with the same name that browsers or Node.js can execute.
6. "Superset" means TypeScript includes all features of JavaScript plus additional ones like types, interfaces, and classes.
7. Yes. You can still declare variables without types in TypeScript (they behave dynamically), but it's not recommended for strict typing.
8. **ECMAScript (ES)** is the official standard that defines how JavaScript should work. Versions like **ES5**, **ES6 (also called ES2015)**, and later ones introduced modern JavaScript features such as `let`, `const`, arrow functions, classes, and modules.

JavaScript is simply an implementation of the ECMAScript standard — meaning, it follows those rules to make the code run in browsers.

TypeScript, on the other hand, is **built on top of ECMAScript**.
It adds new capabilities like **static typing, interfaces, generics, and decorators** — features that ECMAScript doesn't yet include — but it still compiles down to standard JavaScript that follows ECMAScript rules.

In short:

Concept	Description
ECMAScript (ES)	The official standard or rulebook for JavaScript.
JavaScript (JS)	A language that implements ECMAScript.
TypeScript (TS)	A superset of JavaScript that adds extra features and compiles into ECMAScript-compatible JavaScript.

So, **ECMAScript defines the core language**, and **TypeScript builds on top of it** to make development safer and more powerful — especially for automation testing and large projects.
