TypeScript is a **programming language** developed by Microsoft that is a **superset of JavaScript**. This means that any valid JavaScript code is also valid TypeScript code, but TypeScript adds extra features on top of JavaScript.

TypeScript is a **superset of JavaScript** that adds **static types** to catch errors during development. It lets you define variable and function types for **safer, more maintainable code**. TypeScript is **compiled to JavaScript** so it can run in any browser or JavaScript environment.

let's go step by step so you can run a simple "Hello, World!" program in TypeScript on your computer

---

□ Step 1: Install Node.js

You need **Node.js** (which includes `npm`, the Node Package Manager).

□ Check if it's installed:

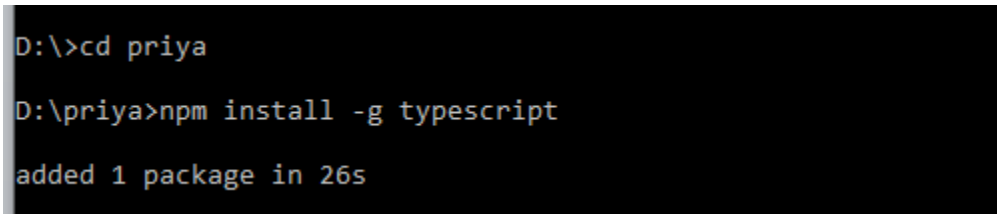
```
node -v
npm -v
```

If you don't have it, download and install it from
□ https://nodejs.org

□ Step 2: Install TypeScript

Use npm to install TypeScript globally:

```
npm install -g typescript
```



Check the installation:

```
tsc -v
```

If it shows a version (e.g., `Version 5.x.x`), you're good.

□ Step 3: Create a TypeScript File

Create a file named **`hello.ts`**.

Add this code:

```
function sayHello(name: string): void {
  console.log(`Hello, ${name}!`);
}

sayHello("World");
```

□ Step 4: Compile TypeScript → JavaScript

Run this in your terminal (from the same folder):

```
tsc hello.ts
```

This creates a new file: **`hello.js`**

You can open it to see the JavaScript version.

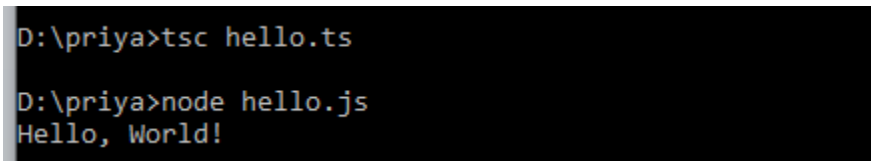
▶□ Step 5: Run the JavaScript File

Now run it using Node:

```
node hello.js
```

□ Output:

```
Hello, World!
```



□ (Optional) Step 6: Run TypeScript Directly

If you want to skip manual compilation, install **ts-node** (runs `.ts` files directly):

```
npm install -g ts-node
```

Then simply run:

```
ts-node hello.ts
```

□ Output:

```
Hello, World!
```

□ Summary

Step	Command	Purpose
1	`npm install -g typescript`	Install TypeScript
2	`tsc hello.ts`	Compile TS to JS
3	`node hello.js`	Run compiled JS
(Optional)	`npm install -g ts-node`	Run TS directly

Note :- To Run typescript file directly first do this :-

Create a `tsconfig.json`

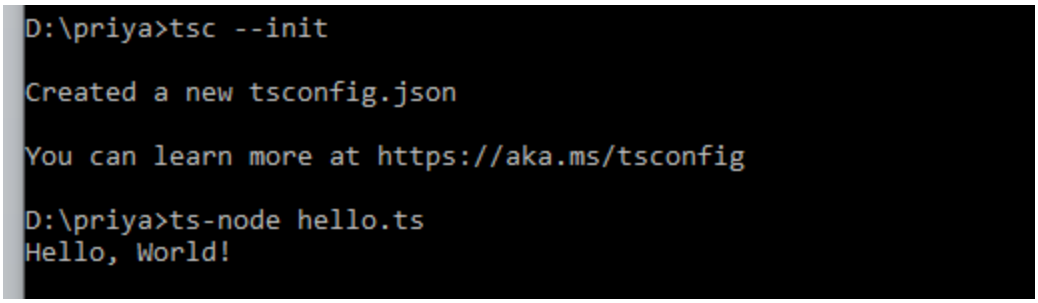
Inside your project folder (`D:\priya`), run:

```
tsc --init
```

This creates a file named `tsconfig.json`.

Then open it and **make sure** these lines exist (or edit them):

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "CommonJS",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true
  }
}
```



run your file :

```
ts-node hello.ts
```

It should print:

```
Hello, World!
```

here's a simple **TypeScript function** that adds two numbers together and prints the result.

□ Example 1 — Basic Addition Function

```
function addNumbers(a: number, b: number): number {
  return a + b;
}

const sum = addNumbers(10, 20);
console.log("Sum:", sum);
```

□ Output:

```
Sum: 30
```

□ Explanation

- `a: number` and `b: number` → Type annotations, ensuring only numbers are passed.
- `: number` after the function → return type is a number.
- If you try to call `addNumbers("10", 20)`, TypeScript will show an error before running.

□ Run It

1. Save as `add.ts`
2. Run:
3. `ts-node add.ts`

□ Output:

```
Sum: 30
```

Here's your TypeScript addition function rewritten using an arrow function with curly braces `{}` and an explicit `return` statement

---

□ TypeScript Code

```
const add = (x: number, y: number): number => {
  return x + y;
};

console.log("Result:", add(5, 15));
```

□ Explanation:

- `const add = (x: number, y: number): number => { ... }`
  - `(x: number, y: number)` → parameters with type annotations
  - `: number` → function returns a number
  - `{ return x + y; }` → curly braces and explicit `return`

▶□ Run it:

Save this as **`add.ts`** and run:

```
ts-node add.ts
```

□ Output:

```
Result: 20
```

here's a **simple TypeScript example** using an `if...else` statement.

□ Example 1 — Check if a number is positive, negative, or zero

```
function checkNumber(num: number): void {
  if (num > 0) {
    console.log(`${num} is positive`);
  } else if (num < 0) {
    console.log(`${num} is negative`);
  } else {
    console.log(`${num} is zero`);
  }
}

checkNumber(5);
checkNumber(-3);
checkNumber(0);
```

□ Output :-

ts-node if.ts

```
5 is positive
-3 is negative
0 is zero
```

□ Example 2 — Check eligibility using `if...else`

```
function checkAge(age: number): void {
```

```
  if (age >= 18) {
    console.log("You are eligible to vote.");
  } else {
    console.log("You are not eligible to vote yet.");
  }
}

checkAge(20);
checkAge(15);
```

□ Output:

ts-node if-else.ts

```
You are eligible to vote.
You are not eligible to vote yet.
```
