A.Y. 2023-2024

Class: TE-ITA/B, Semester: VI

Subject: **Web Lab**

**Experiment – 2:  To study the basics of TypeScript through small code snippets.**

1. **Aim:** To study simple program in TypeScript compile and run it.

1. **Objectives:** Aim of this experiment is that, the students will be able

* To know Installing TypeScript
* Read and understand commonly-used TypeScript syntax and patterns
* to understand TypeScript error handling, Looping

1. **Outcomes:** After study of this experiment, the students will be able

* To install Typescript.
* Write code, compile and execute the code to get result.
* To know syntaxes for loop constructs, defining variables, printing on screen, taking input from user

1. **Prerequisite:** Basic knowledge of JavaScript is required

1. **Requirements:** Personal Computer, Windows operating system, VSCode editor, browser, Internet Connection, google doc.

**Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

**Laboratory Exercise**

1. **Procedure:**

**a. Answer the following:**

1. JavaScript vs Typescript.

| JAVASCRIPT | TYPESCRIPT |
| --- | --- |
| It doesn't support strongly typed or static typing. | It supports strongly typed or static typing feature. |
| Netscape developed it in 1995. | Anders Hejlsberg developed it in 2012. |
| JavaScript source file is in ".js" extension. | TypeScript source file is in ".ts" extension. |
| It is directly run on the browser. | It is not directly run on the browser. |
| It is just a scripting language. | A superset of JavaScript with type safety. |
| Larger and older community with extensive resources. | Growing community, primarily among professional developers. |
| Directly interpreted and executed, faster without compilation. | Slower initially due to compilation but improves debugging efficiency. |

1. How to use Typescript?

ANS:

1. Install TypeScript: Use npm install -g typescript to install the TypeScript compiler globally on your system.
2. Set Up the Project: Initialize a TypeScript project by running tsc --init to create a tsconfig.json file for configuring compilation settings.
3. Write TypeScript Code: Create .ts files and write code with features like type annotations, interfaces, enums, and classes for better type safety.
4. Transpile TypeScript to JavaScript: Use the tsc command to compile .ts files into .js files, which can be executed in browsers or Node.js.
5. Run the Compiled JavaScript: Execute the generated .js file using node or include it in an HTML file for browser-based applications.
6. Use Modern IDEs: Leverage editors like Visual Studio Code for features like type checking, autocompletion, and inline error detection.
7. Integrate with Build Tools (Optional): For larger projects, integrate TypeScript with tools like Webpack, Babel, or Parcel for automated builds and bundling.
8. TypeScript Features?

ANS:

1. **Type Inference**TypeScript can automatically infer the type of a variable based on its initial value, reducing the need for explicit type annotations.
2. **Object-Oriented Programming (OOP)**TypeScript supports OOP concepts like classes, interfaces, inheritance, and access modifiers (public, private, protected), making it suitable for complex application development.
3. **Interfaces**It provides interface definitions to describe the structure of an object, ensuring consistency across the codebase.
4. **Generics**TypeScript supports generics, enabling the creation of reusable, type-safe components or functions.
5. **Enums**Enums allow developers to define a set of named constants, making code more readable and manageable.
6. **Compile-Time Error Checking**Errors are caught during compilation, reducing the chances of runtime failures.
7. **Support for ES Features**TypeScript supports all modern ECMAScript (ES) features, including modules, arrow functions, async/await, and destructuring.
8. **Cross-Platform Compatibility**TypeScript transpiles to plain JavaScript, making it compatible with any JavaScript runtime, including browsers and Node.js.
9. **Tooling and IDE Support**TypeScript integrates seamlessly with popular IDEs like Visual Studio Code, offering features like autocompletion, inline documentation, and error checking.
10. **Namespaces and Modules**TypeScript supports namespaces and modules for organizing code, making it easier to maintain and scale projects.
11. **Advanced Type Features**It offers advanced types like union, intersection, and mapped types, along with type aliases and utility types for better flexibility.
12. **Decorators (Experimental)**TypeScript supports decorators, a meta-programming feature for adding annotations to classes, methods, and properties.
13. **Compatibility with JavaScript**Being a superset of JavaScript, TypeScript allows existing JavaScript code to run seamlessly, ensuring easy adoption.

**b**. **Attach screenshots:**

* Typescript installation
* Typescript Program code and output with your own comments and

indentation.

1. **Post-Experiments Exercise**
2. **Extended Theory:**

Nil

1. **Questions:**

* Draw and explain Components of TypeScript
* List out the built-in data types in TypeScript.

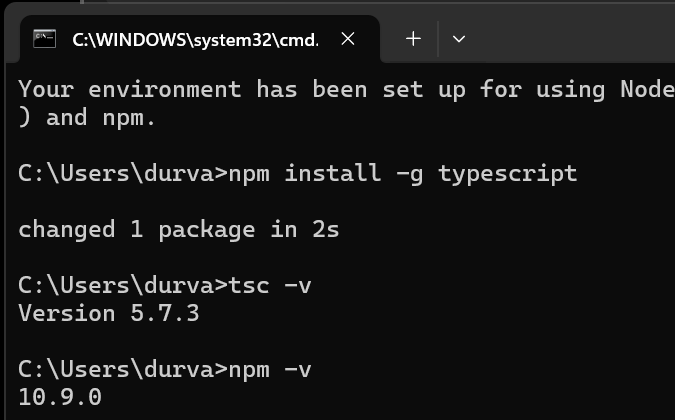
1. **Conclusion:**

* Write what was performed in the experiment.
* Write the significance of the topic studied in the experiment.

1. **References:**
2. <https://www.typescriptlang.org/assets/typescript-handbook.pdf>
3. <http://basarat.gitbooks.io/typescript/>

**B. Attach screenshots:**

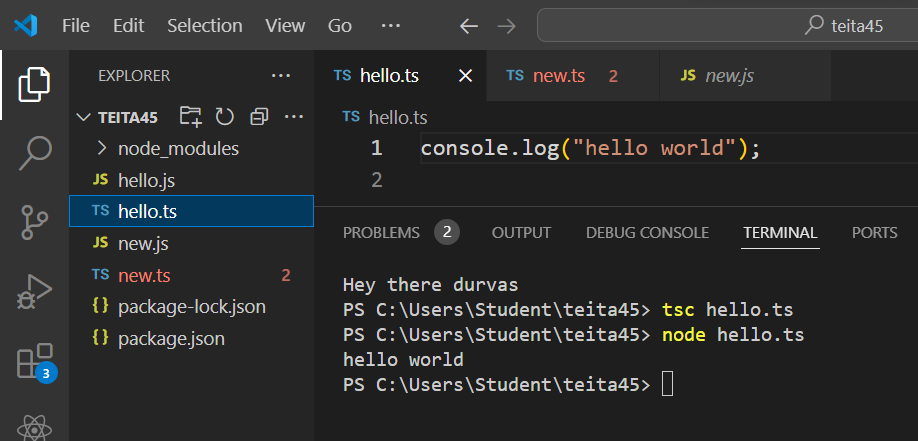
1. Typescript installation



2. Typescript Program code and output with your own comments and

indentation.

**CODE 1:**



**CODE 2:**

| const prompt = require('prompt-sync')();  let b:string = "Hello World we learn here  variable"  console.log(b)  function hello(){  return "Hello World we learn here function"  }  console.log(hello())  const n = prompt('What is your name?');  console.log(`Hey there ${n}` |  |
| --- | --- |

1. **VARIABLE DECLARATION & TYPE CHECKING**

let totalAmount: number = 100;

let discount: number = 15;

function calculateFinalAmount(total: number, discount: number): number {

return total - (total \* (discount / 100));

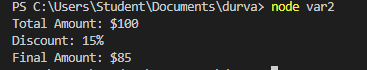
}

let finalAmount = calculateFinalAmount(totalAmount, discount);

console.log(`Total Amount: $${totalAmount}`);

console.log(`Discount: ${discount}%`);

console.log(`Final Amount: $${finalAmount}`);

OUTPUT**: **

1. **CONTROL STRUCTURES**

1)IF-ELSE:

let number = -5;

if (number > 0) {

console.log("The number is positive.");

} else if (number < 0) {

console.log("The number is negative.");

} else {

console.log("The number is zero.");

}



2)SWITCH CASE:

let timeOfDay = 15;

switch (true) {

case (timeOfDay >= 5 && timeOfDay < 12):

console.log("Good morning!");

break;

case (timeOfDay >= 12 && timeOfDay < 17):

console.log("Good afternoon!");

break;

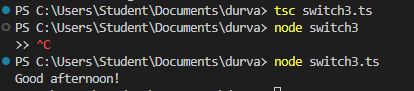
case (timeOfDay >= 17 && timeOfDay < 21):

console.log("Good evening!");

break;

default:

console.log("Good night!");

OUTPUT: 

3) LOOPS

let i = 1;

while (i <= 10) {

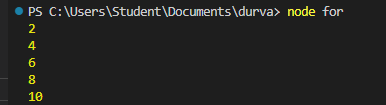
if (i % 2 === 0) {

console.log(i);

}

i++;

}

OUTPUT: 

1. FUNCTIONS

function elevatorDestination(currentFloor, destinationFloor) {

if (destinationFloor > currentFloor) {

return "The lift is moving up to floor " + destinationFloor;

} else if (destinationFloor < currentFloor) {

return "The lift is moving down to floor " + destinationFloor;

} else {

return "You are already on the destination floor.";

} }

let currentFloor = 3;

let destinationFloor = 7;

console.log(elevatorDestination(currentFloor, destinationFloor));

OUTPUT: 

1. **OPERATORS --> TERNARY OPERATORS:**

let age = 25;

let isMember = true;

let ticketPrice = age < 12

? "Free"

: age <= 18

? "$5 (Child Ticket)"

: isMember

? "$10 (Discounted Member Ticket)"

: "$15 (Regular Ticket)";

console.log("The ticket price is: " + ticketPrice);

OUTPUT: 