| **Experiment No. – 2** | | | | |
| --- | --- | --- | --- | --- |
| **Date of Performance:** | **15/1/25** | | | |
| **Date of Submission:** | **22/1/25** | | | |
| Program Execution/  formation/  correction/  ethical practices  (06) | Timely  Submission  (01) | Viva  (03) | Experiment  Total (10) | Sign with Date |
|  |  |  |  |  |

**Experiment No. 2**

**2.1 Aim:** Design a calculator Web Application Using Typescript and HTML

**2.2 Course Outcome:** CO2: Understand how TypeScript and AngularJS framework can build dynamic, responsive single-page web applications

**2.3 Learning Objectives:** Learn to design a web-based calculator by integrating HTML for UI and TypeScript for logic, handling events, performing arithmetic operations, validating input, debugging, and deploying the application.

**2.4 Requirement:**

Visual Studio Code or any code editor  
 - Browser (Chrome, Firefox, etc.)  
 - TypeScript compiler

2.5 Related Theory:

TypeScript is a typed superset of JavaScript developed by Microsoft. This means it builds upon JavaScript by adding optional static typing, interfaces, enums, and object-oriented programming features such as classes, inheritance, and access modifiers. TypeScript code is transpiled to plain JavaScript, ensuring compatibility with all major browsers and platforms. By enforcing type checks at compile-time, TypeScript helps developers catch errors early, leading to more reliable and maintainable code.

HTML (HyperText Markup Language) is the standard language used to structure the content of web pages. It defines elements like headings, paragraphs, buttons, and input fields, providing the skeleton that browsers render visually.

In the context of a calculator application, these technologies work together to demonstrate key web development concepts:

● Event Handling: Responding to user interactions such as button clicks to perform calculations.

● Basic Logic Implementation: Executing arithmetic operations (addition, subtraction, multiplication, division) based on user input.

● DOM Manipulation: Dynamically updating the content of the web page (e.g., displaying results) using TypeScript to interact with HTML elements.

Combining TypeScript with HTML in a calculator app allows developers to build interactive, error-resistant, and scalable web applications.

**1.6 Procedure:**

1. Set up the development environment.  
2. Create HTML structure for calculator buttons and display.  
3. Write TypeScript logic for arithmetic operations.  
4. Compile TypeScript to JavaScript.  
5. Link the script in HTML.  
6. Test functionality in the browser.

**1.7 Program and Output:**

type Operator = '+' | '-' | '\*' | '/';

function calculate(a: number, b: number, operator: Operator): number {

switch (operator) {

case '+':

return a + b;

case '-':

return a - b;

case '\*':

return a \* b;

case '/':

if (b === 0) {

throw new Error("Division by zero is not allowed.");

}

return a / b;

default:

throw new Error(`Unsupported operator: ${operator}`);

}

}

// Example usage:

const num1 = 10;

const num2 = 5;

console.log(`Addition: ${calculate(num1, num2, '+')}`);

console.log(`Subtraction: ${calculate(num1, num2, '-')}`);

console.log(`Multiplication: ${calculate(num1, num2, '\*')}`);

console.log(`Division: ${calculate(num1, num2, '/')}`);

****

**1.8 Conclusion:**

In conclusion, a basic calculator web application was successfully developed using TypeScript and HTML. This project serves as a practical demonstration of how front-end technologies can be integrated to create interactive and functional web tools. Through this application, core programming concepts such as scripting logic, conditional statements, function calls, and DOM manipulation were applied in a real-world scenario.

The use of TypeScript provided a structured and type-safe coding environment, which not only helped in minimizing runtime errors but also enhanced code readability and maintainability. Meanwhile, HTML offered the foundational layout and structure for the calculator interface.

Overall, this project reinforces essential web development skills and provides a solid foundation for building more complex, user-driven applications in the future.

**1.9 Questions:**

**1.What are the advantages of TypeScript over JavaScript?**

● Static Typing: TypeScript allows optional static typing, which helps catch errors at compile time rather than at runtime.

● Improved IDE Support: Better autocompletion, navigation, and refactoring tools.

● Enhanced Code Quality: Type annotations and interfaces improve readability and maintainability.

● Early Error Detection: Helps detect potential bugs during development.

● ES6+ Features: TypeScript supports modern JavaScript features and compiles them down to older versions for compatibility.

● Large-Scale Applications: Better suited for large projects with complex codebases due to stronger tooling and structure.

**2. How does the TypeScript compiler work?**

The TypeScript compiler (tsc) performs the following steps:

1. Parsing: Reads and parses .ts files to build an Abstract Syntax Tree (AST).
2. Type Checking: Validates the code based on defined types and reports errors.
3. Transpilation: Converts TypeScript code into plain JavaScript.
4. Output Generation: Outputs JavaScript files (.js) along with optional source maps (.map) and declaration files (.d.ts), depending on the configuration.

**3. What happens when division by zero is performed?**

● In JavaScript/TypeScript:

○ Positive number / 0 → Infinity

○ Negative number / 0 → -Infinity

○ 0 / 0 → NaN (Not a Number)

○ No runtime error or crash occurs.