**TYPESCRIPT**

1. **Defination :** TypeScript is an open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript, which means that any valid JavaScript code is also valid TypeScript code. TypeScript adds static typing to JavaScript, allowing developers to catch errors at compile time rather than at runtime.

**Static Typing:** TypeScript allows developers to specify types for variables, function parameters, return types, and more. This helps catch type-related errors during development and improves code quality.

1. **Installation : npm install -g typescript**
2. **Basic Types :**

|  |
| --- |
| **a. boolean**  let isWorking: boolean = true;  **b. number : Represents both integer and floating-point numbers.**  let num: number = 1000;  **c. string**  let myname: string = "Piyush";  **d. Array : Represents an array of elements of a specific type.**  let myarray: number[] = [1, 2, 3, 4, 5];  **e. Tuple : Represents an array with a fixed number of elements, where each element may be of a different type.**  let mytuple: [string, number, string, number, number] = ["Sam", 2, "Rajput", 4, 2];  **f. Any : Represents a dynamic type where the variable can hold values of any type.**  let mydata: any = 1000;  mydata = "Sam";  mydata = false;  **g. Null & Undifined : Used to represent null or undefined values, respectively**  let nullValue: null = null;  let undefinedValue: undefined = undefined; |

1. **Type Annotations :**

|  |
| --- |
| **a. Variable Declarations:**  let num1: number = 1000;  let name1: string = "Piyush"  let isActive: boolean = false  console.log(num1, name1, isActive);  **b. Function Parameters and Return Types:**  **Function with explicit return type void (no return value)**  function addNumber(x: number, y: number) {      console.log(x + y);  }  addNumber(10, 20);  **Function with type annotations for parameters and return type**  function mulNumber(a: number, b: number) {      return a \* b;  }  let mynum = mulNumber(10, 5);  console.log(mynum);  **c. Object Type Annotations:**  let person: { id: number, fname: string, lname: string, email: string } = {      id: 10,      fname: "Shreyash",      lname: "Thaware",      email: "shreyash@gmail.com"  }  console.log(person);  **d. Array Type Annotations:**  let numbers: number[] = [1, 2, 3, 4, 5];  let names: string[] = ["Alice", "Bob", "Charlie"];  **e. Union Types :** Union types in TypeScript allow you to declare a type that can hold values of multiple specified types. This means that a variable with a union type can store values of different types at different times, providing flexibility while maintaining type safety. Union types are denoted using the pipe (|) symbol between the types you want to combine.  let input:string | number ;  input = 10  console.log(input);  input = "my name is piyush"  console.log(input); |

1. **Type Aliases :**

|  |
| --- |
| **a. Basic Type Alias**  type mynumber1 = number;  let mynum1: mynumber1 = 10;  console.log(mynum1);  **b. Union Type Alias**  type mynumber2 = number | string | null;  let mynum2: mynumber2 = 10;  let mynum3: mynumber2 = "Sam";  let mynum4: mynumber2 = null;  console.log(mynum2, mynum3, mynum4);  **c. Object Type Alias**  type myObjType = { id: number, title: string, description: string };  let mynum5: myObjType = {      id: 100,      title: "something",      description: "somewhere in northpole"  }  console.log(mynum5);  **d. Function Type Alias**  type MathOperation = (x: number, y: number) => number;  let add: MathOperation = (a, b) => a + b;  let subtract: MathOperation = (a, b) => a - b; |