HEXAWARE

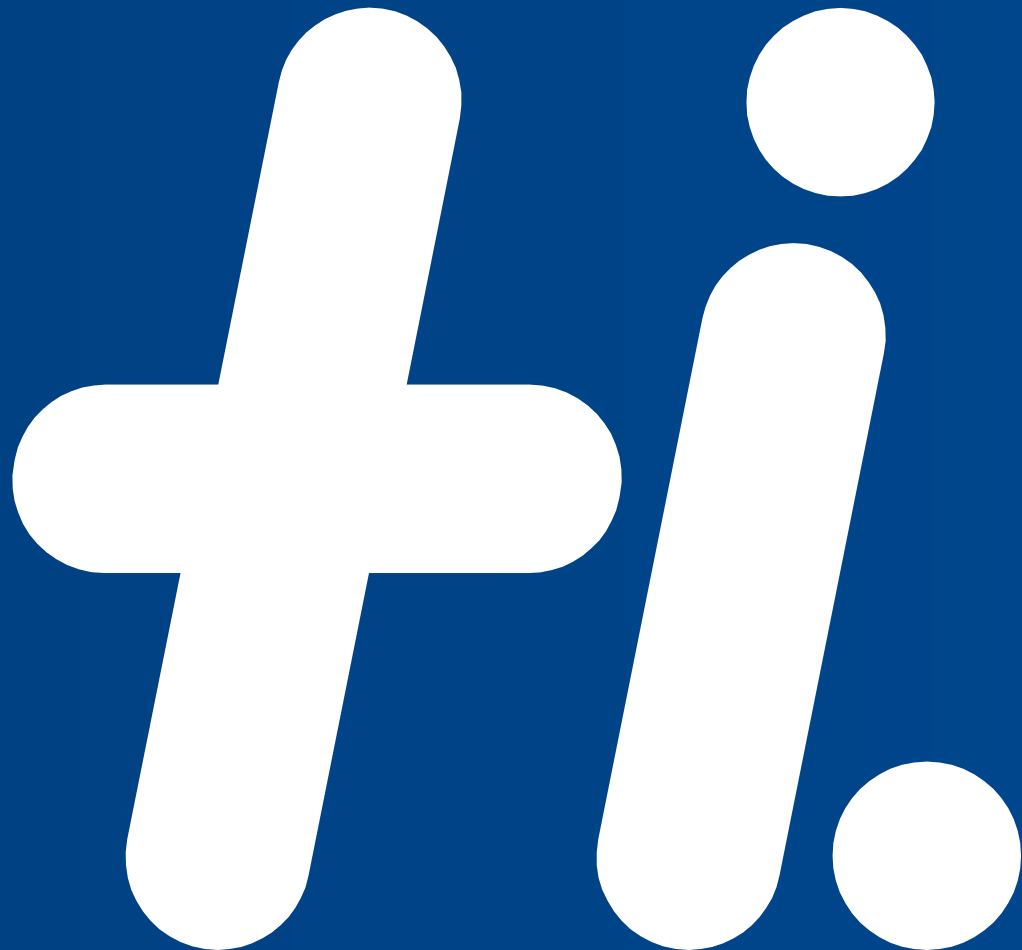
# TypeScript

# Session Objective

- Introduction to TypeScript
- Features of TypeScript
- Identifiers & Variable in TypeScript
- Variables in TypeScript
- Class & Object in TypeScript

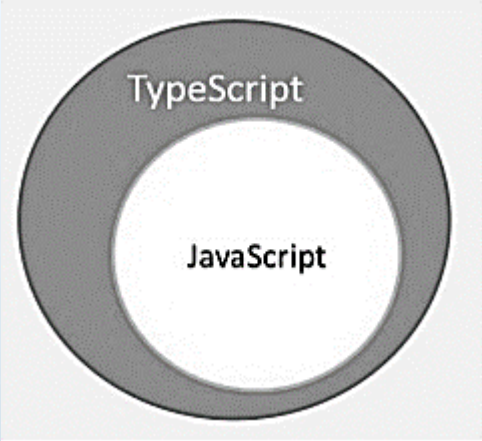
  

# TypeScript Introduction



# TypeScript

- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java.



# What is TypeScript?

- TypeScript is JavaScript for application-scale development.
- TypeScript is a strongly typed, object oriented, compiled language. It was designed by **Anders Hejlsberg** at Microsoft.
- TypeScript is both a language and a set of tools.

# Features of TypeScript Cont…

- **TypeScript is just JavaScript**.
  - TypeScript starts with JavaScript and ends with JavaScript.
  - All TypeScript code is converted into its JavaScript equivalent for the purpose of execution.
- **TypeScript supports other JS libraries**.
  - Compiled TypeScript can be consumed from any JavaScript code.
  - TypeScript-generated JavaScript can reuse all of the existing JavaScript frameworks, tools, and libraries.

# Features of TypeScript

- **JavaScript is TypeScript**.
  - This means that any valid **.js** file can be renamed to **.ts** and compiled with other TypeScript files.
- **TypeScript is portable**.
  - TypeScript is portable across browsers, devices, and operating systems.
  - It can run on any environment that JavaScript runs on.
  - Unlike its counterparts, TypeScript doesn't need a dedicated VM or a specific runtime environment to execute.

# TypeScript Composition Cont…

- A TypeScript program is composed of −
  - Modules
  - Functions
  - Variables
  - Statements and Expressions
  - Comments

# TypeScript Composition

HEXAWARE

```
var message:string = "Hello World"
console.log(message)
```

On compiling, it will generate following JavaScript code.

```
//Generated by typescript 1.8.10
var message = "Hello World";
console.log(message);
```


Compile & Execute

# Compile and Execute a TypeScript Cont..

- **Step 1**
  - Save the file with .ts extension. For example save the file as Test.ts.
  - The code editor marks errors in the code, if any, while you save it.

- **Step 2**
  - Right-click the TypeScript file under the Working Files option in VS Code's Explore Pane.
  - Select Open in Command Prompt option.

# Compile and Execute a TypeScript

- **Step 3**
  - To compile the file use the following command on the terminal window.
  - tsc Test.ts

- **Step 4**
  - The file is compiled to Test.js.
  - To run the program written, type the following in the terminal.
  - node Test.js

# Variable & Identifier

# Identifiers in TypeScript

Identifiers are names given to elements in a program like variables, functions etc.

The rules for identifiers are −

- Identifiers can include both, characters and digits. However, the identifier cannot begin with a digit.
- Identifiers cannot include special symbols except for underscore (_) or a dollar sign ($).

# Identifiers in TypeScript

- Identifiers cannot be keywords.
- They must be unique.
- Identifiers are case-sensitive.
- Identifiers cannot contain spaces.

# Variable

| S.No. | Variable Declaration Syntax & Description |
|---|---|
| 1. | **var name:string = ”mary”**<br>The variable stores a value of type string |
| 2. | **var name:string;**<br>The variable is a string variable. The variable’s value is set to undefined by default |
| 3. | **var name = ”mary”**<br>The variable’s type is inferred from the data type of the value. Here, the variable is of the type string |
| 4. | **var name;**<br>The variable’s data type is any. Its value is set to undefined by default. |

# TypeScript Variable Scope

- The scope of a variable specifies where the variable is defined. The availability of a variable within a program is determined by its scope. TypeScript variables can be of the following scopes −

- Global Scope − Global variables are declared outside the programming constructs. These variables can be accessed from anywhere within your code.

- Class Scope − These variables are also called fields. Fields or class variables are declared within the class but outside the methods. These variables can be accessed using the object of the class. Fields can also be static. Static fields can be accessed using the class name.

- Local Scope − Local variables, as the name suggests, are declared within the constructs like methods, loops etc. Local variables are accessible only within the construct where they are declared.

# Example of Variable scope

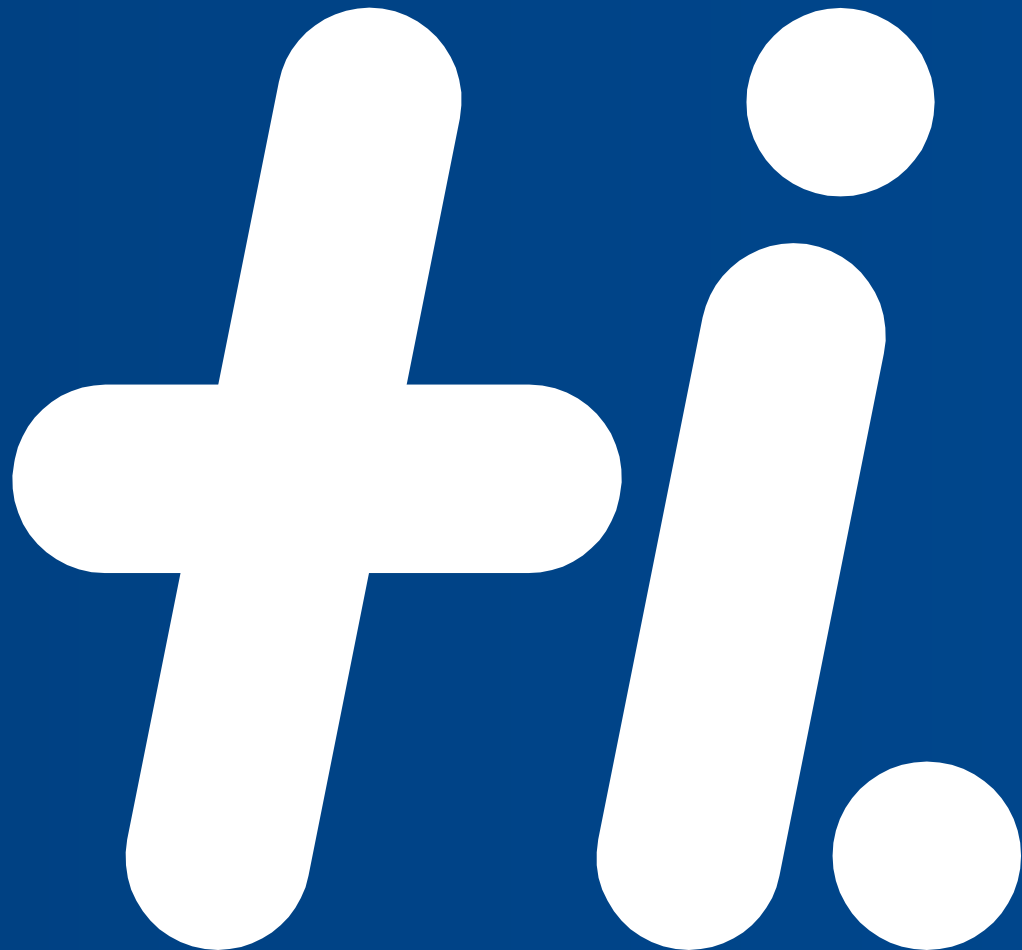
- The following example illustrates variable scopes in TypeScript.

Example: Variable Scope

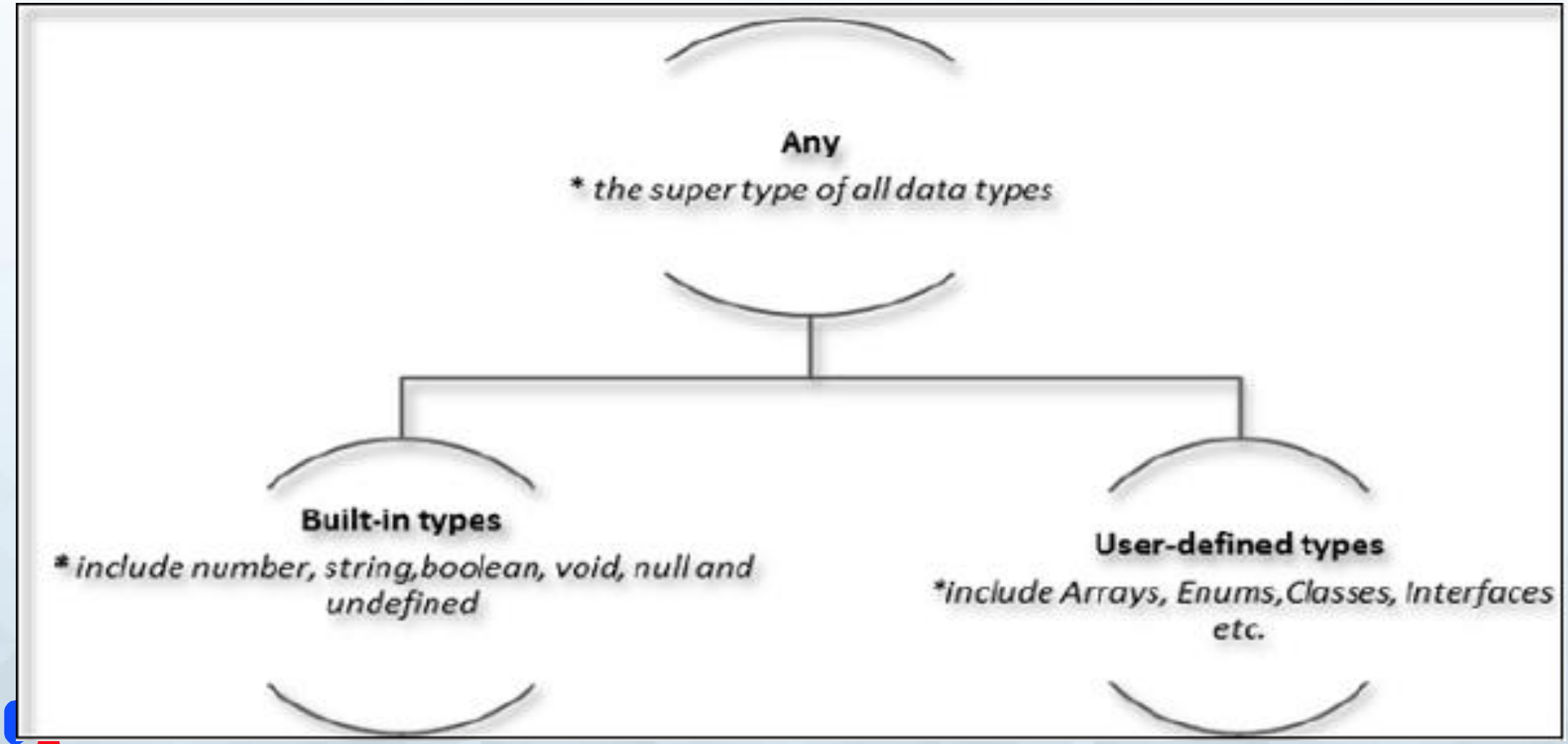
```
var global_num = 12          //global variable
class Numbers {
   num_val = 13;             //class variable
   static sval = 10;         //static field

   storeNum():void {
      var local_num = 14;    //local variable
   }
}
console.log("Global num: "+global_num)
console.log(Numbers.sval)   //static variable
var obj = new Numbers();
console.log("Global num: "+obj.num_val)
```

# Data Type



# Data type classification Cont…



# Data Type Cont…

- The Any type
  - The any data type is the super type of all types in TypeScript.
  - It denotes a dynamic type.
  - Using the any type is equivalent to opting out of type checking for a variable.

# Data Type Cont…

| Data type | Keyword | Description |
|---|---|---|
| Number | number | Double precision 64-bit floating point values. It can be used to represent both, integers and fractions. |
| String | string | Represents a sequence of Unicode characters |
| Boolean | boolean | Represents logical values, true and false |
| Void | void | Used on function return types to represent non-returning functions |
| Null | null | Represents an intentional absence of an object value. |
| Undefined | undefined | Denotes value given to all uninitialized variables |

# Data Type Cont…

- Null and undefined
  - The **null** and the **undefined** datatypes cannot be used to reference the data type of a variable.
  - They can only be assigned as values to a variable.
  - *Null and undefined are not the same*.
  - A variable initialized with undefined means that the variable has no value or object assigned to it while null means that the variable has been set to an object whose value is undefined.

# Data Type

- User-defined Types
  - User-defined types include Enumerations (enums), classes, interfaces, arrays, and tuple.

# Class & Object

# Class Cont…

- Creating classes
- Use the class keyword to declare a class in TypeScript. The syntax for the same is given below −

- Syntax
- class class_name {
- //class scope
- }
- The class keyword is followed by the class name. The rules for identifiers must be considered while naming a class.

# Class

- A class definition can include the following −
  - Fields : A field is any variable declared in a class. Fields represent data pertaining to objects
  - Constructors : Responsible for allocating memory for the objects of the class
  - Functions : Functions represent actions an object can take. They are also at times referred to as methods
- These components put together are termed as the data members of the class.
- Consider a class Person in typescript.

```
class Person {
}
```

# Object

- An object is an instance which contains set of key value pairs. The values can be scalar values or functions or even array of other objects. The syntax is given below −

- Syntax

```
var object_name = {
   key1: “value1”, //scalar value
   key2: “value”,
   key3: function() {
      //functions
   },
   key4:[“content1”, “content2”] //collection
};
```

HEXAWARE

*Innovative Services*

*Passionate Employees*

*Delighted Customers*

# *Thank you*

www.hexaware.com