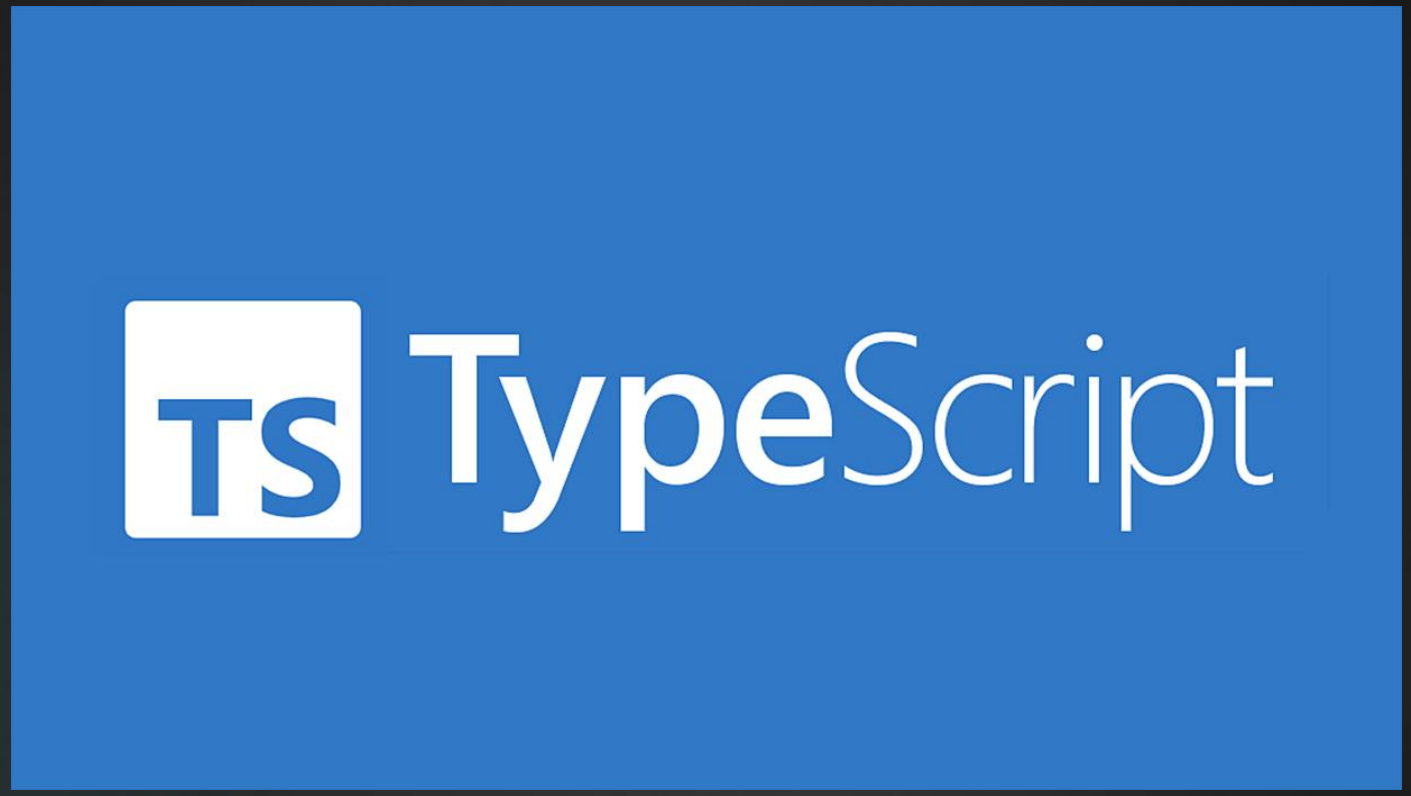
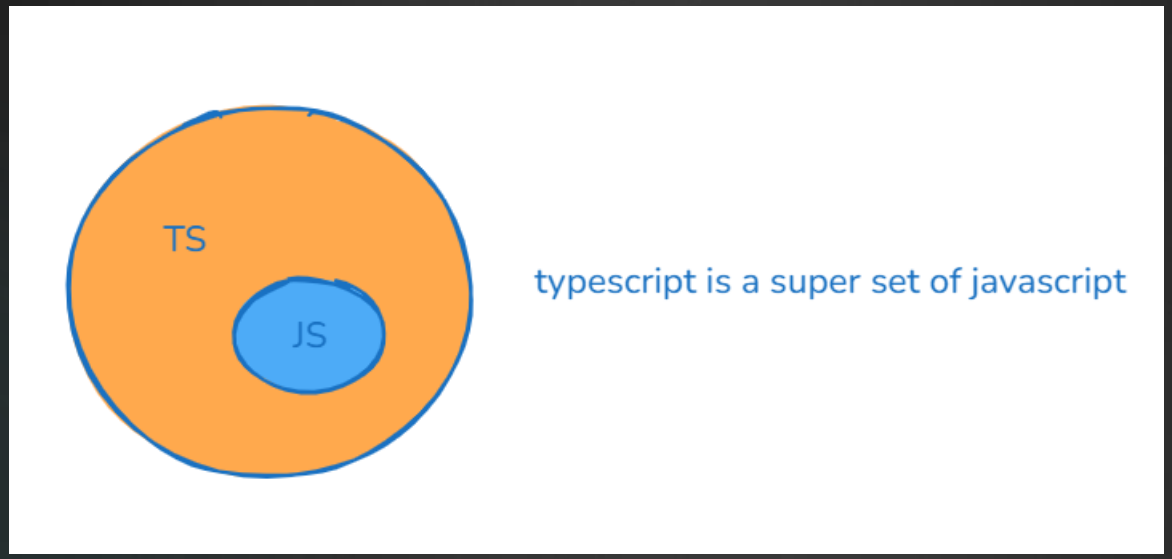
TYPESCRIPT



PRAJWAL C

Typescript

- Typescript is the superset of java script



JavaScript → Dynamic
Typescript → Static

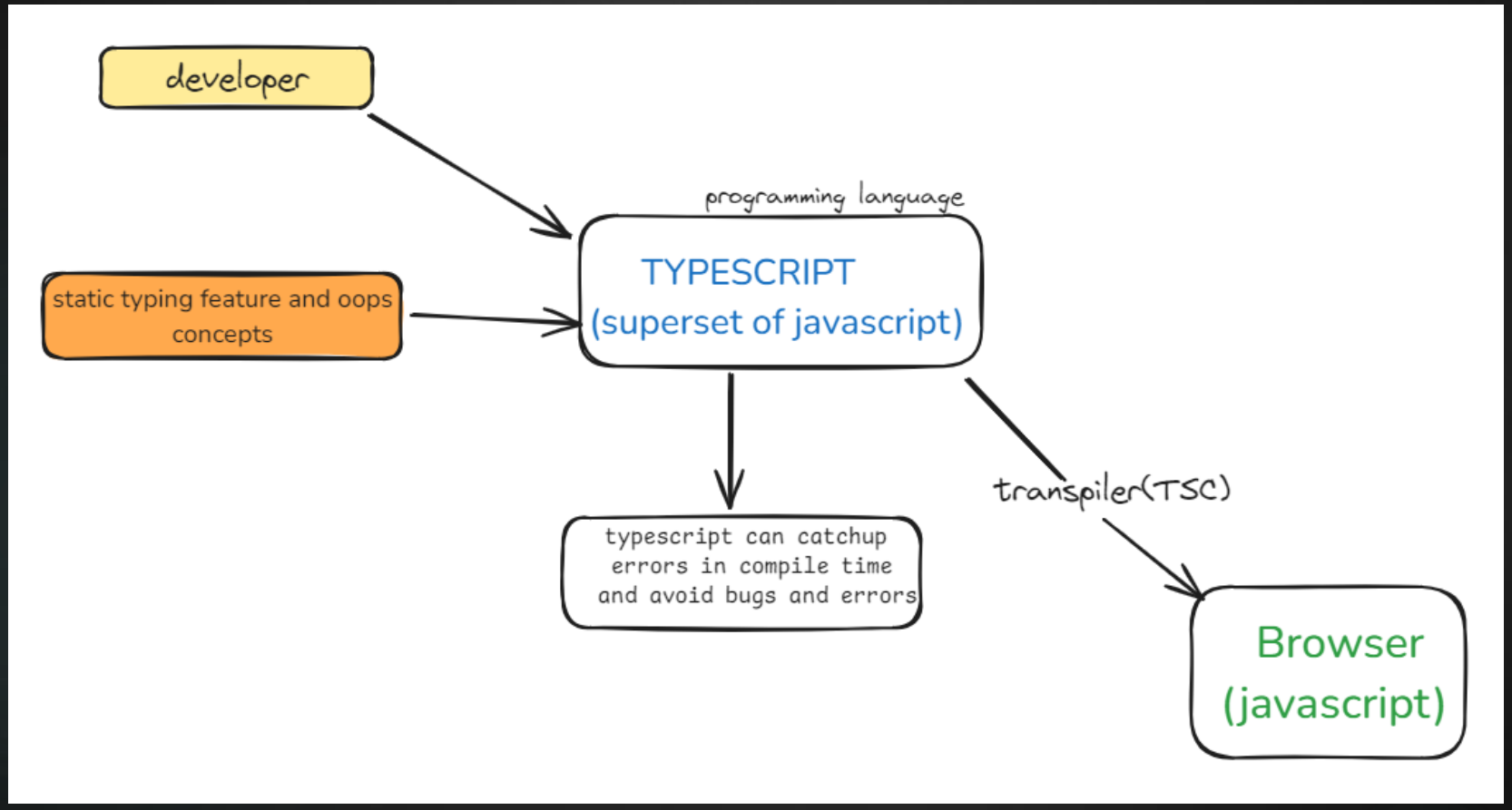
What is Typescript?

- Typescript is a strongly typed programming language that builds on top of JavaScript.
- Typescript is the superset of JavaScript.

Who developed Typescript?

- Typescript was developed by Microsoft Company by the developer Andres Hejlsberg in 2012.

How Typescript works?



What is Typescript Compiler [TSC]

- Typescript compiler takes the typescript code and it compiles into java script code later javascript can be executed by the java script engine or NodeJS platform.

Why we need Typescript?

- Typescript is a strongly typed programming language that helps to prevent common mistakes before executing programs.

Benefits of Typescript

- Optionally static typing feature.
- Type Inference.
- Access ES6 and ES7 features.
- Cross Platform and Cross Browser Compatibility.

NOTE:

- We can use java script feature in typescript.
- We cannot use typescript feature in java script.

Extensions

- Java script = [.js]
- typescript = [.ts]

Installations

- node installation
- **https://nodejs.org/en/download/prebuilt-installer**

- To install typescript compiler
- **npm install -g typescript**

Commands for installations

- to check the version of typescript
  - tsc –-version
- to check the version node
  - node –-version
- to uninstall typescript
  - npm uninstall –g typescript
  - npm cache clean –force
- to compile the typescript file
  - tsc filename.ts
- to run the typescript file in node environment
  - node filename

Typescript types

- Typescript language supports different types of value. It provides data types for the java script to transform it into a strongly typed language.
- Java script doesn't support static typing but with the help of typescript we can use static typing feature in java script.

Note:

type is a convenient way to refer to different properties as a value.

Type Annotations

- Typescript is a typed language where we can specify the type of variables, function and objects properties. We can specify the type using [:type] after the name of the variable, parameter or property.
- In simple words Explicitly we are specifying the type of variables.

Type Inference

- In TypeScript, it is not necessary to annotate type always. The TypeScript compiler infers the type information when there is no explicit information available in the form of type annotations.

Datatypes in Typescript

- Primitive Datatypes.
  - String
  - Number
  - Boolean
  - Null
  - Undefined
- Object Reference Datatypes.
  - Object
  - Functions
  - Arrays

Type Alias

- In Typescript, Type Aliases give a type a new name. They are similar to interfaces in that they can be used to name primitives and any other kinds that you'd have to define by hand otherwise. Aliasing doesn't truly create a new type; instead, it gives that type a new name

Tuple

- Typescript introduced a new datatype called tuple.
- Tuple contain 2 values of different data types.
- Tuple are like advanced arrays with extra feature that ensure type safety, particular where we need to account for a list containing a fixed number of elements with multiple known types.

Difference between arrays and tuple

- The major difference between array and tuple is that when we assign values to a tuple, these values must match the types defined in the tuple declaration in the same order.

- On the other hand array can support multiple types with the any type or the bitwise OR [|] operator, but the order or structure of the element doesn’t came into play.

Type Union

- Typescript union has the ability to combine one or more different types of data
- It is the most powerful way to express a variable with the multiple types
- Use pipe(“|”) to combine one or more data types.

Types of union

- Tagged union
- Union narrowing

Tagged Union

- a tagged union is a union type where each member type has a unique "tag" property that allows typescript to determine which member type belongs to
- this tag is usually a string literal or a numeric literal type.

Union Narrowing

- type narrowing means removal of types from a union, to narrow a variable into specific type
- to remove the type we use typegards i.e "typeof" operator

Intersection

- it combines multiple types into one.
- this allows you to add together existing types to get a single type that has all the feature you need.
- for intersection we use "&"

ENUMS

- it is used to define a set of named constraints i.e. a collection a related values
- typescript supports both numeric and string Enums.
- to declare Enums we need use keyword “enum”.

Types of Enums

- Numeric Enums
- String Enums
- Heterogenous Enums

Special Types

- Any:
  - Typescript has a special type “any” that you can use whenever you don’t want to particular value to cause type checking errors.
  - When value is of any type you can access any properties of it.

Type Assertions

- Type assertion is a technique that informs the compiler about the type of a variable. Type assertion is similar to typecasting but it doesn’t reconstruct code.
- We can either use “ < >”angular brackets or “as” keywords to do type assertion.

Note:

- Type assertions do not perform any runtime type checks conversions.
- Use angular brackets (< >) or (as) keyword.
- Especially we use type assertions in DOM elements.

Literal Types

- Literal types in typescript allow you to specify exact values for variables or properties.
- Instead of regular types we can represent a range of possible values.
- Literal types restrict the value to a specific literal value.

Types of Literals

- String Literal
- Numeric Literal
- Function Literal
- Object Literal

Interface

- Interfaces are similar to type aliases, except they only apply to object types.
- Interface contain only declaration of the members. It is the responsibility of the deriving class to define the member.

Note:

- Interface defines the structure of objects and specifying the types of the properties and methods

OOP's in Typescript

- Object Oriented Programming is a programming language based on classes and objects rather than functions and logic
- The software programs in OOP are structured into reusable pieces of code known as classes.

Benefits of OOP

- Easier Debugging.
- Reuse of code through inheritance.
- Flexibility through polymorphism.
- Effective problem solving.
- Project decoupling [Separate project into groups]

Typescript Classes

- In TypeScript, classes are blueprints for creating objects. They encapsulate data (properties) and behavior (methods) into a single unit. By defining classes, you can organize your code, promote reusability, and enhance readability.
- Key Components of TypeScript Classes

Methods: Functions defined within a class that perform specific actions.

Constructors: Special methods called when an object is created from a class. Constructors initialize class properties.

Properties: Variables associated with a class instance.

Constructor

- Constructor is a special method within a class that is to initialized object properties when an instance of the class is created.

NOTE:

A class is a template or a blueprint to create Objects

Inheritance

- It allows one class to inherit properties and methods from another class.
- The class that inherits is called the child class, and the class whose properties and methods are inherited is called the parent class.
- Inheritance enables code reusability, allowing a class to leverage the functionality of an existing class without rewriting it.

Access Modifiers

- Access modifiers change the visibility of the properties and methods of a class.
- TypeScript provides three access modifiers:
  - Private
  - protected
  - Public
  - Read-only property

Private

- The private modifier limits the visibility to the same class only. When you add the private modifier to a property or method, you can access that property or method within the same class. Any attempt to access private properties or methods outside the class will result in an error at compiled time.

Public

- The public modifier allows class properties and methods to be accessible from all locations. If you don’t specify any access modifier for properties and methods, they will take the public modifier by default.

Protected

- The protected modifier allows properties and methods of a class to be accessible within the same class and subclasses.

Read-only property

- Not allowed to change the content means constant data member.

Getters & Setters

- In TypeScript, there are two supported methods getter and setter to access and set the class members.
- Methods of the Typescript accessor property:
  - **getter**: This method comes when you want to access any property of an object. A getter is also called an accessor.
  - **setter**: This method comes when you want to change any property of an object. A setter is also known as a mutator.

Abstract Class

- Define an abstract class in Typescript using the abstract keyword. Abstract classes are mainly for inheritance where other classes may derive from them. We cannot create an instance of an abstract class.
- An abstract class typically includes one or more abstract methods or property declarations. The class which extends the abstract class must define all the abstract methods.

Generics

- Generic in typescript allow you to create reusable components or functions that can work with multiple data types.
- In generics, we need to write a type parameter between the open (<) and close (>) brackets, which makes it strongly typed collections. Generics use a special kind of type variable <T> that denotes types. The generics collections contain only similar types of objects.

Decorators

- Decorators are a powerful syntactic feature in typescript that allow you to add functionality or modify the behavior of classes, methods, properties & parameters at runtime.
- Decorators are essentially functions that are applied using “@” symbol before the declaration you want to decorate.

THANK YOU

Prajwal C