TypeScript

Why TypeScript?

• Catches errors at compile-time, so that you can fix it before you run
code.
• Supports object-oriented programming features like data types, classes, enums, etc., allowing JavaScript to be used at scale.
• TypeScript implements the future features of JavaScript a.k.a ES Next
so that you can use them today.

Architecture

Design Goals of Typescript

• Statically identify JavaScript constructs that are likely to be errors
• strongly-typed programming language and perform static type checking at compile time.
• Be a cross-platform development tool
• Microsoft released TypeScript under the open source Apache license and it
can be installed and executed in all major operating systems.
•Impose no runtime overhead on emitted programs
• the term design time or compile time to refer to the TypeScript code that is

written while designing an application, whereas the term execution time or
runtime to refer to the JavaScript code executed after compiling some
TypeScript code.
• High compatibility with existing JavaScript code
• any valid JavaScript program is also a valid TypeScript program (with a few small exceptions)
• Provide a structuring mechanism for larger pieces of code
• Features like class-based object-orientation, interfaces, namespaces, and modules, help us to structure our code in a much better way.
• reduce potential integration issues within the development team
• Making code easier to maintain and scale
• Impose no runtime overhead on emitted programs
• Some features of Typescript are only available at design time. For example, we can declare interfaces in TypeScript, but since JavaScript doesn't support interfaces, the TypeScript compiler will not declare or try to emulate this feature at runtime (in the output JavaScript code
• Align with current and future ECMAScript proposals
• TypeScript compiler with some mechanisms, such as code transformations (converting TypeScript features into plain JavaScript implementations) and type erasure (removing static type notation), to generate clean JavaScript code.

Components of TypeScript

Language:

• Features the TypeScript language elements.
• It consists of syntax, keywords, and type annotations.

Compiler:

• Changes the instructions written in TypeScript to its JavaScript equivalent.
• Performs the parsing, type checking, and transformation of your TypeScript
code to JavaScript code.
• The TypeScript compiler configuration is given in tsconfig.json file.

Language Service:

• Generates information that helps editors and other tools provide better
assistance features, such as IntelliSense or automated refactoring.
• Assists the common set of typical editor operations like statement
completion, signature help, code formatting and outlining, colorization, etc.

IDE integration (VS Shim):

• The developers of the IDEs and text editors must perform some integration
work to take advantage of the TypeScript features. TypeScript was designed to
facilitate the development of tools that help to increase the productivity of
JavaScript developers.