**Typescript**

Typescript is a statically-typed superset of JavaScript. Typescript builds on top of javascript.

Typescript is a compiled language. It undergoes a compilation process to transform typescript code into javascript code, which is then executed by a javascript runtime. The typescript compiler is responsible for this process.

**Typescript improves our productivity while helping avoid bugs**

Types increase productivity by helping us avoid many mistakes. By using types, we can catch bugs at the compile tim instead of having them occur at runtime

**Typescript brings the future javascript today**

Typescript supports the upcoming features planned in the ES next for the current javascript engines.

**Start and Run typescript**

**Install typescript globally**

>npm install –g typescript

**Create typescript file**

Write typescript code in a file with .ts extension.

example.ts

function greet(name: string): string {

  return `Hello, ${name}!`;

}

const result = greet("TypeScript");

console.log(result)

**compile typescript code**

run the typescript compiler (‘tsc’) to transpile the typescript code into javascript.

>tsc example.ts

**Run javascript code**

Now that we have the javascript file.

>node example.js

**Optional watch mode**

To recompile our typescript code whenever changes are made, we can use ‘--watch’ option with ‘tsc’

>tsc example.ts –watch

Typescript and javascript compare

|  |  |  |
| --- | --- | --- |
| Criteria | Javascript | Typescript |
| First release | 1995 | 2012 |
| Created by | Netscape | Microsoft |
| Trademark owner | Oracle | Microsoft |
| Standard | ECMAScript 5 | ECMAScript 2015 |
| Typing | Dynamic | Optional strong typing |
| Community and adoption | Larger than typescript | Smaller than javascript |
| Compilation | Not required | Compile into javascript |
| File extension | .js and .jsx | .ts and .tsx |
| IDE support | Limited validation and refactoring capabilities | Extensive validation and refactoring capabilities |
| Complexity | Easy scripting language for new developers to learn | Additional features require a investment of tim to learn |
| Readability | Large code bases are harder to read and decipher | Type system and object oriented approach improve readability |
| compatibility | Javascript code is also valid typescript code | Typescript code gets compiled into fully compliant javascript |

|  |  |  |
| --- | --- | --- |
| Criteria | Javascript | Typescript |
| Static typing | It is dynamically typed language where variable types are determined at runtime. | It is statically typed language allowing developers to specify types for variables, function parameters, and return types |
| Type annotations | It doesn’t have built in syntax for type annotations. | It supports type annotation to provide explicit type information |
| Compilation | It is interpreted language with code executed directly by the javascript engine. | It is compiled language. Code is transpiled to javascript before execution. |
| Tooling and IDE support | Limited tooling support compared to typescript | Enhanced tooling support, including better autocompletion, refactoring tools and error checking in modern IDEs |
| ECMAScript features | Adheres to ECMAScript standard features supported by the runtime environment. | Can include features from future ECMAScript versions before they are officially supported |
| Code readability | As type information is not explicitly declared, which makes code less readable | Type annotation can improve code readability by providing clear information about data types |
| Compatibility | It is compatible with all major browsers and javascript runtime environments | It transpiled to javascript, ensuring compatibility while allowing the use of typescript specific features |
| Error checking | Errors related to type mismatches may only be discovered at runtime | Type checking is performed at compile tim catching potential errors before the code is executed |
| Community and ecosystem | a mature and extensive ecosystem with a vast number of libraries and frameworks |  |
| Gradual adoption | No need for explicit type annotation. Suitable for small projects and quick prototyping | Supports gradual adoption allowing developers to introduce static typing gradually in existing projects |

Types and type aliases

“type” is a keyword in typescript that we can use to define the shape of data. It can describe not only object but also unions, intersection.

Basic types in typescript

String

Boolean

Number

Array

Tuple

Enum

Type aliases in typescript mean “a name for any type”. They provide a way of creating new names for existing type. Type aliases don’t define a new types; instead, they provide alternative name for existing type.

type MyNumber = number;

type User = {

  id: number;

  name: string;

  email: string;

};

// MyNumber is type aliases for number type

// User is a type aliases to represent the type of a user

Interfaces

Interface in typescript is like a contract that defines the shape of object. it specifies what properties and methods object should have.

interface Client {

  name: string;

  address: string;

}