Ts本质上是JS的超集，包含了JS的所有元素，可以使用 JS中的所有代码和编码概念，并新增了静态类型、类、模块、接口和类型注解，可以载入JS代码运行，并扩展了JS的语法。支持JS的所有网络协议。

JS特点：

* 脚本编写语言，无需编译，只要嵌入 HTML 代码中，就能由浏览器逐行加载解释执行
* JS 是一种基于对象的语言，可以创建对象同时使用现有对象。但是JS并不支持其它面向对象语言所具有的继承和重载功能
* 语法简单，使用的变量为弱类型，类型可以任意转换（隐式/显式转换）
* 语言安全，仅在浏览器端执行，不会访问本地硬盘数据
* 具有动态性，JS是事件驱动的，只根据用户的操作做出相应的反应处理
* JS 只依赖于浏览器，与操作系统的因素无关。因此JS是一种跨平台的语言
* 兼容性较好，能够与其他技术一起使用（如XML，REST API等）

TS特点：

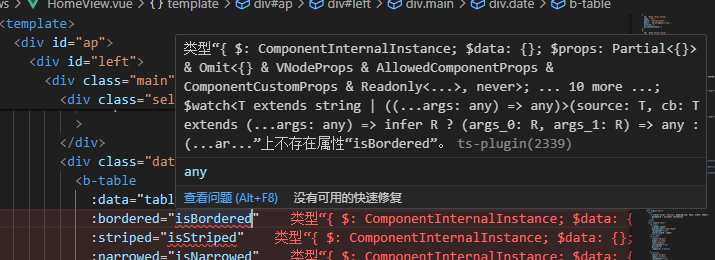
* JS 代码可以在无需任何修改的情况下与 TS 一同工作，同时可以使用编译器将 TS 代码转换为 JS
* TS 通过类型注解，提供编译时的静态类型检查（定义类型与赋的值类型不匹配会编译报错）
* TS 中的数据要求带有明确的类型，JS不要求
* TS 为函数提供了缺省参数值（void）
* TS 引入了JS中没有的“类”概念（但后续JS在ES6中也引入了class概念）
* TS 中引入了模块的概念，可以把声明、数据、函数和类封装在模块中

TS VS JS ：

|  |  |  |
| --- | --- | --- |
| **范围** | **Typescript** | **JavaScript** |
| 开发者 | Microsoft in 2012 | Brendan Eich (Netscape)，1995 年 |
| 定义 | TypeScript 是一种强大的面向对象语言，作为 JavaScript 的超集，具有泛型和 JS 特性来解决 JS 的复杂问题。 | JavaScript 是一种脚本语言，具有创建动态网页的一流功能。 |
| 类型 | 强类型。TypeScript 支持静态类型和动态类型。 | 弱类型 。JavaScript 仅支持动态类型。 |
| 生态 | TypeScript 更像是一种支持静态类型的强大而直观的语言。 | JavaScript 是一种简单的语言，它优化了代码的兼容性，易于阅读和编写。 |
| 编译 | TypeScript 需要编译。 | JavaScript 不需要编译。 |
| 数据绑定 | TypeScript 利用接口和类型等概念来定义所使用的数据。 | JavaScript 中没有这样的概念。 |
| 学习曲线 | TypeScript 有一条不友好的学习曲线。此外，它需要JavaScript知识。 | 它易于学习并且是一种用于编写 Web 脚本的灵活语言。 |
| Npm 包 | 使用Typescript，许多npm包要么有静态类型定义，要么有一个更容易安装的外部类型定义。 | JavaScript提供了不需要任何构建步骤就可以搜索和生成代码的替代方案。 |
| 客户端或服务器端 | TypeScript 专门用于客户端。 | JavaScript 用于服务器端和客户端。 |
| 文件扩展名 | .tsx 和 .ts | .js |
| 社区 | TypeScript 有一个较小的软件开发者社区。 | JavaScript 拥有庞大的软件开发人员社区。 |
| 原型设计 | 原型设计特性在 TypeScript 中可用。 | JS 不支持原型设计。 |
| 代码示例 | function multiply (a, b){ return a\*b;}var result = multiply(a, b);console.log('答案是 - ' + result); | <script>function multiply (a, b){ return a\*b;}var result = multiply(a, b);document.write (‘The answer is – ’ + result);</script> |

Buefy in TS:

Buefy 目前官方尚未提供完整的TS类型支持，有的属性值可以使用已有的类型定义，但有很多属性需要自行创建类型声明，考量一下我觉得还是使用JS更加便捷于开发



TS编译器：

TS使用的编译器是基于Node.js的TypeScript compiler（TSC）

TSC会自带transpiler工具库，它把新的JavaScript语言特性的代码，转换为目标环境老版本JavaScript语言的代码，可以在tsconfig.json中进行配置，如：

{

"compilerOptions": {

"target": "es5"

}

}

当编译TS项目时，编译器会输出如下内容：

* JavaScript 文件 .js
* Source maps文件 .js.map，用来将生成的js文件映射到原ts文件。这对代码debug非常有用。很多开发工具比如Chrome DevtTools等，在出错时，会显示原ts文件，而不是编译后js文件。
* 类型声明文件 .d.ts，让其他TS项目可以引用您的项目的类型。
* 类型声明映射（maps）文件 .d.ts.map，生成TS的自定义类型与JS的对应变量的映射关系，对于大型的TS项目可以加快项目的编译速度

常见的编译器：

* GCC（GNU Compiler Collection）：

GCC 是一套由 GNU 开发的编程语言编译器。最初只能处理C语言，但后来演进成支持C++、Fortran、Pascal、Objective-C、Java等多种语言

优点：对Windows和Linux等多个平台支持良好，编译速度较快。

* LLVM（Low Level Virtual Machine）/ Clang：

LLVM 提供了与编译器相关的支持，能够进行程序语言的编译期优化、链接优化、在线编译优化、代码生成。可作为多种编译器的后台来使用。

Clang是LLVM的前端，用于编译C、C++、Objective-C等语言。它以LLVM为后端，高效易用，并与IDE集成良好。

优点：速度快，内存占用小，诊断信息可读性强，错误提示更友好。

* MSVC（Microsoft C++）：

微软开发，只能用于Windows系统，支持C++和C#等语言，与Visual Studio集成

浏览器对于ES6支持问题：

绝大部分的浏览器都支持es6，如果有需要也可以在webpack打包时使用babel进行语法降级为es5

* Chrome：51版起，可支持97%的ES6新特性
* Firefox：53版起，可支持97%的ES6新特性
* Safari：10版起，可支持99%的ES6新特性
* Opera：版本16及以上
* Microsoft Edge：所有版本
* IE：Edge 15可支持96%的ES6新特性。Edge14可支持93%。（IE7~11基本不支持ES6）