## 01. TypeScript 和 JavaScript 区别?

TypeScript JavaScript

面向对象的语言 脚本语言

静态类型 没有静态类型

支持模块 不支持模块

支持可选参数 不支持可选参数

## 02. 什么是 TypeScript?

TypeScript 是 JavaScript 的类型化超集，可以编译成普通的 JavaScript。它是纯面向对象的，有类、接口和像 C# 或 Java 这样的静态类型编程语言。您将需要一个编译器来编译和生成 JavaScript 文件中的代码。基本上，TypeScript 是带有一些附加特性的 JavaScript 的 ES6 版本。

**1 var** message:string = "Welcome to Edureka!"

2 console.**log**(message)

将 TypeScript 代码写入扩展名为.ts的文件中，然后使用编译器编译为 JavaScript。 您可以在任何代码编辑器中写入文件，并且需要在平台上安装编译器。 安装后，命令tsc .ts 将 TypeScript 代码编译为纯 JavaScript 文件。

## 03. 为什么需要TypeScript?

JavaScript 开发人员应考虑使用 TypeScript 的原因有多种。

其中一些包括：

· 使用 ECMAScript 的新功能：TypeScript支持新的ECMAScript标准，并将其转换为您选择的ECMAScript目标。因此，您可以使用ES2015及更高版本的功能。

· 静态键入：JavaScript是动态类型的，在运行时实例化变量之前，不知道变量是什么类型。 TypeScript将类型支持添加到JavaScript。

· 类型推断：通过使用类型推断，TypeScript使键入变得更容易，并且显式性更低。即使您没有明确键入这些类型，它们仍然存在，可以避免您执行某些操作，否则会导致运行时错误。

· 更好的IDE支持：TypeScript的开发经验是对JavaScript的极大改进。有各种各样的IDE对TypeScript具有出色的支持，例如Visual Studio和VS Code，Atom，Sublime和IntelliJ / WebStorm。

· 严格的Null检查：错误，例如无法读取未定义的属性“ x”，在JavaScript编程中很常见。您可以避免大多数此类错误，因为不能使用TypeScript编译器不知道的变量。

· 互操作性：TypeScript与JavaScript紧密相关，因此具有强大的互操作性功能，但是要使用TypeScript中的JavaScript库，还需要做一些额外的工作。

## 04. TypeScript的特性。

· 跨平台：TypeScript编译器可以安装在任何操作系统上，例如Windows，MacOS和Linux。

· 面向对象的语言：TypeScript提供了诸如类，接口和模块之类的功能。 因此，它可以为客户端和服务器端开发编写面向对象的代码。

· 静态类型检查：TypeScript使用静态类型，并在编译时帮助进行类型检查。 因此，您无需运行脚本即可在编写代码时发现错误。

· 可选的静态键入：如果您使用的是JavaScript动态键入，TypeScript还允许可选的静态键入。

· DOM操作：可以使用TypeScript来操作DOM以添加或删除元素。

· ES6功能：TypeScript包括计划的ECMAScript 2015（ES6，7）的大多数功能，例如类，接口，箭头功能等。

## 05. 使用TypeScript有哪些好处？

使用TypeScript的好处是：

· TypeScript快速，简单，易学并且可以在任何浏览器或JavaScript引擎上运行。

· 它类似于JavaScript，并使用相同的语法和语义。

· 这有助于后端开发人员更快地编写前端代码。

· 您可以从现有的JavaScript代码中调用TypeScript代码。而且，它可以与现有的JavaScript框架和库一起正常工作。

· 扩展名为.d.ts的Definition文件提供对现有JavaScript库（如Jquery，D3.js等）的支持。

· 它包含来自ES6和ES7的功能，这些功能可以在ES5级JavaScript引擎（如Node.js）中运行。

## 06. 使用TypeScript有哪些缺点？

TypeScript具有以下缺点：

· TypeScript需要很长时间来编译代码。

· 它不支持抽象类。

· 如果我们在浏览器中运行TypeScript应用程序，则需要执行编译步骤才能将TypeScript转换为JavaScript。

· 网络开发人员使用JavaScript已有数十年了，TypeScript并没有带来任何新的东西。

· 要使用任何第三方库，定义文件是必须的。

· 类型定义文件的质量是一个问题。

## 07. TypeScript的组成部分是什么？

TypeScript中有三种不同类型的组件，包括：

· 语言，它由语法，关键字和类型注释组成。

· TypeScript编译器，编译器（tsc）将用TypeScript编写的指令转换为等效的JavaScript。

· TypeScript语言服务，语言服务在核心编译器管道（类似于编辑器的应用程序）周围暴露了一个附加层。 语言服务支持一组常见的典型编辑器操作。

## 08. 谁开发了 TypeScript ? 当前稳定版本是什么？

Anders Hejlsberg 开发了 TypeScript。 此外，他还是C＃语言开发团队的核心成员之一。 TypeScript 于2012年10月1日首次发布，并被标记为版本0.8。 但是，它是由 Microsoft 根据 Apache 2 许可开发和维护的。 它是为开发大型应用程序而设计的。

TypeScript 的当前稳定版本是 4.0。Typescript 编译为简单的 JavaScript 代码，可在支持 ECMAScript 2015 框架的任何浏览器上运行。 此外，它还支持最新的 JavaScript 功能。

## 09. 如何安装 TypeScript?

通过npm安装

1 npm install -g typescript

2

3 or

4

5 npm tsc

## 10. 如何编译 TypeScript 文件？

任何 TypeScript 文件的扩展名都是.ts。而且任何JavaScript文件都是 TypeScript 文件，因为它是 JavaScript 的超集。 因此，将“ .js”的扩展名更改为“ .ts”后，您的 TypeScript 文件就准备好了。

要将任何.ts文件编译为.js，请使用以下命令：

1 $ tsc <TypeScript File Name>

例如：编译"edureka.ts"

1 tsc edureka.ts

编译结果为edureka.js

## 11. 可以把多个ts文件合并成一个js文件吗？

是的，我们可以合并多个文件。编译时，我们需要添加–outFILE [OutputJSFileName]选项。

1 tsc --outFile common.js file1.ts file2.ts file3.ts

这将编译所有3个“ .ts”文件，并输出到单个“ common.js”文件中。

1 tsc --outFile file1.ts file2.ts file3.ts

如果您不提供输出文件名，则将编译 file2.ts 和 file3.ts 并将输出放置在 file1.ts 中。因此，现在您的 file1.ts 包含 JavaScript 代码。

## 12. TypeScript中的类型。

类型系统表示语言支持的不同类型的值。它在程序存储或操作所提供的值之前检查其有效性。

它可以分为两种类型，

* 内置：包括数字(number)，字符串(string)，布尔值(boolean)，无效(void)，空值(null)和未定义(undefined)。
* 用户定义的：它包括枚举(enums)，类(classes)，接口(interfaces)，数组(arrays)和元组(tuple)。

## 13. 列出TypeScript中的内置数据类型。

在TypeScript中，内置数据类型也称为原始数据类型，包括：

* Number：代表数字类型的值。 这些数字在TypeScript中存储为浮点值。
* String：字符串表示存储为Unicode UTF-16代码的一系列字符。
* Boolean：代表逻辑值。 当我们使用布尔类型时，我们只能获得true或false的输出。
* Null：Null表示变量，其值未定义。 无法直接引用空类型值本身。
* Undefined：未定义类型表示所有未初始化的变量。
* Void：无效是不返回任何类型值的函数的返回类型。

## 14. TypeScript中的变量以及如何声明？

变量是内存中用于存储值的命名空间。

在 TypeScript 中声明变量的类型语法在变量名称后包括一个冒号（:)，后跟其类型。 与 JavaScript 相似，我们使用var关键字声明变量。

在Typescript中声明变量时，必须遵循某些规则：

* 变量名称必须是字母或数字。
* 不能以数字开头名称。
* 除下划线（ \_ ）和美元（ $ ）符号外，它不能包含空格和特殊字符。

## 15. 声明变量的不同方式？

声明变量有四种方法：

1 // 声明类型和值，Declaring type and value in a single statement

2 var [identifier] : [type-annotation] = value;

3

4 // 只声明类型，Declaring type without value

5 var [identifier] : [type-annotation];

6

7 // 只声明值，Declaring its value without type

8 var [identifier] = value;

9

10 // 声明变量无类型和值，Declaring without value and type

11 var [identifier];

## 16. 可以实时编译ts文件吗？

是的，我们可以通过.ts文件中的实时更改自动编译“ .ts”。这可以通过使用–watch编译器选项来完成：

1 tsc --watch file1.ts

上面的命令首先在file1.js中编译file1.ts，并观察文件更改。 如果检测到任何更改，它将再次编译文件。 在这里，我们需要确保在使用–watch选项运行时一定不要关闭命令提示符。

## 17. TypeScript支持哪些面向对象术语？

TypeScript支持以下面向对象的术语：

* 模块（Modules）
* 类（Classes）
* 接口（Interfaces）
* 继承（Inheritance）
* 数据类型（Data Types）
* 成员函数（Member functions）

## 18. TypeScript中的接口。

接口是定义应用程序中合同的结构。 它定义了要遵循的类的语法。 它仅包含成员的声明，派生类负责定义成员。 TypeScript编译器使用接口进行类型检查，并检查对象是否具有特定的结构。

**1 interface** **interface\_name** {

*2 // variables' declaration*

*3 // methods' declaration*

4 }

## 19. TypeScript中的类及其特性。

TypeScript 引入了类，以便它们可以利用诸如封装和抽象之类的面向对象技术的好处。

TypeScript 编译器将 TypeScript 中的类编译为普通的 JavaScript 函数，以跨平台和浏览器工作。

一个类包括以下内容：

* 构造器（Constructor）
* 属性（Properties）
* 方法（Methods）

1 class Employee {

2 empID: number;

3 empName: string;

4

5 constructor(ID: number, name: string) {

6 this.empName = name;

7 this.empID = ID;

8 }

9

10 getSalary(): number {

11 return 40000;

12 }

13 }

类的其他特性有：

* 继承（Inheritance）
* 封装（Encapsulation）
* 多态（Polymorphism）
* 抽象（Abstraction）

## 20. TypeScript支持的访问修饰符。

TypeScript支持访问修饰符 public，private 和 protected，它们决定了类成员的可访问性。

如下所示：

* 公共（Public），类的所有成员，其子类以及该类的实例都可以访问。
* 受保护（Projected），该类及其子类的所有成员都可以访问它们。 但是该类的实例无法访问。
* 私有（Private），只有类的成员可以访问它们。

如果未指定访问修饰符，则它是隐式公共的，因为它符合 JavaScript 的便利性。

## 21. TypeScript是一种可选的静态类型语言吗?

TypeScript是可选的静态类型，这意味着您可以要求编译器忽略变量的类型。 使用任何数据类型，我们都可以为变量分配任何类型的值。 TypeScript在编译期间不会进行任何错误检查。

1 var unknownType: any = 4*;*

2 unknownType = "Welcome to Edureka"*; //string*

3 unknownType = false*; // A boolean.*

## 22. TypeScript中的模块。

模块是创建一组相关变量，函数，类和接口等的强大方法。它可以在自己的范围内执行，但不能在全局范围内执行。 基本上，您不能直接在模块外部访问模块中声明的变量，函数，类和接口。

可以使用export关键字创建一个模块，并可以使用import关键字在其他模块中使用该模块。

1 module module\_name{

2 **class** **xyz** {

3 **export** **sum**(x, y){

**4 return** x+y;

5 }

6 }

7 }

## 23. TypeScript中内部模块和外部模块区别?

| **内部模块** |
| --- |

内部模块将类，接口，函数，变量分组为一个单元，并可以导出到另一个模块中。

内部模块是Typescript早期版本的一部分。

这些是其他模块的本地或导出成员。

内部模块使用指定其名称和主体的ModuleDeclarations进行声明。

**外部模块**

外部模块可用于隐藏模块定义的内部语句，并且仅显示与声明的变量关联的方法和参数。

|  |
| --- |
| 外部模块在最新版本中称为模块。 |

这些是使用外部模块名称引用的单独加载的代码体。

外部模块被编写为一个单独的源文件，其中至少包含一个导入或导出声明。

## 24. TypeScript中的命名空间以及如何声明？

命名空间对功能进行逻辑分组。 它们在内部维护typescript的遗留代码。 它封装了共享某些关系的要素和对象。 命名空间也称为内部模块。 命名空间还可以包括接口，类，函数和变量，以支持一组相关功能。

**1 namespace** <namespace\_name> {

**2 export** **interface** I1 { }

**3 export** **class** **c1**{ }

4 }

## 25. TypeScript支持方法重载吗？

是的，TypeScript支持函数重载。 但是实现很奇怪。 因此，当您在TypeScript中重载时，只有一个实现带有多个签名。

第一个签名具有类型编号的一个参数，而第二个签名具有字符串类型的参数。 第三个函数包含实际的实现，并且具有类型为any的参数。 然后，该实现会检查所提供参数的类型，并根据供应商参数类型执行另一段代码。

## 26. 解释一下TypeScript中的装饰器。

装饰器是一种特殊的声明，可以应用于类，方法，访问器，属性或参数。 装饰器是@expression符号前缀的函数，其中expression必须计算为一个函数，该函数将在运行时使用有关修饰声明的信息进行调用。

TypeScript装饰器的目的是以声明的方式将注释和元数据添加到现有代码中。 要为装饰器启用实验性支持，您需要在命令行或我们的tsconfig.json中启用experimentalDecorators编译器选项：

Command Line

1 $tsc --target ES5 --experimentalDecorators

tsconfig.json

1 {

2 "compilerOptions": {

3 "target": "ES5",

4 "experimentalDecorators": **true**

5 }

6 }

## 27. 什么是混入？

在Javascript中，Mixins是一种从可重用组件构建类，然后通过组合更简单的部分类来构建类的方法。

这个想法很简单，不需要 类A 扩展 类B 来获得它的功能，函数B 接受 类A 并返回具有此新增功能的新类。 在这里，函数B 是混合。

## 28. TypeScript支持可选参数吗？

与JavaScript不同，如果您尝试调用一个函数而未提供其函数签名中声明的确切数量和参数类型，则TypeScript编译器将引发错误。

为解决此问题，您可以通过使用问号（'？'）来使用可选参数。 它表示可以或不可以接收值的参数可以附加一个“？”以将其标记为可选。

**1 function** **Demo**(arg1: number, arg2? :number) {

2

3 }

*4 //So, arg1 is always required, and arg2 is an optional parameter.*

在上面的示例中，arg1是必须的，而arg2是可选参数。

## 29. 什么是作用域变量？

作用域是一组对象，变量和函数，JavaScript有全局作用域变量和局部作用域变量。

您可以在两个不同的作用域内声明变量，例如：

* 局部作用域变量，它是在函数内使用的函数对象；
* 全局作用域变量，您可以在函数外和函数内使用此窗口对象；

## 30. 如何调试TypeScript文件?

要调试任何TypeScript文件，您需要一个.js源映射文件。 因此，您必须使用–sourcemap标志编译.ts文件以生成源映射文件。

1 $ tsc -sourcemap file1.ts

这将创建file1.js和file1.js.map。 file1.js的最后一行将是源映射文件的引用。

*1 //# sourceMappingURL=file1.js.map*

## 31. 什么是TypeScript Definition Manager？为什么需要它？

TypeScript定义管理器（TSD）是一个程序包管理器，用于直接从社区驱动的DefinitelyTyped存储库中搜索和安装TypeScript定义文件。

现在，如果您想在.ts文件中使用一些jQuery代码：

1 $(document).**ready**(**function**() { *//Your jQuery code });*

在这里，当您尝试使用tsc进行编译时，将出现编译时错误：找不到名称“ ”。因此，您需要通知TypeScript编译器“”。 因此，您需要通知TypeScript编译器“ ”。因此，您需要通知TypeScript编译器“”属于jQuery。 为此，TSD发挥了作用。 您可以下载jQuery Type Definition文件并将其包含在我们的.ts文件中。

## 32. 包含类型定义文件的步骤是什么？

包括类型定义文件的过程涉及的步骤为：

首先，安装TSD。

1 $ npm install tsd -g

接下来，在TypeScript目录中，通过运行以下命令创建一个新的TypeScript项目：

1 $ tsd **init**

然后安装jQuery的定义文件。

1 tsd query jquery --action install

上面的命令将下载并创建一个新目录，其中包含以“ .d.ts”结尾的jQuery定义文件。现在，通过更新TypeScript文件以指向jQuery定义来包含定义文件。

*1 /// <reference path="typings/jquery/jquery.d.ts" />*

2 $(document).**ready**(**function**() { *//To Do*

3 });

最后，再次编译。这次生成的js文件没有任何错误。因此，对TSD的需求有助于我们获得所需框架的类型定义文件。

## 33. TypeScript中的 Declare 关键字。

JavaScript库或框架没有TypeScript声明文件。 但是，如果要在TypeScript文件中使用它们而没有任何编译错误，则必须使用declare关键字。 declare关键字用于环境声明和您要定义可能在其他位置存在的变量的方法。

如果要在我们的TypeScript代码中使用该库，则可以使用以下代码：

**1 declare** **var** myLibrary;

TypeScript运行时会将myLibrary变量分配为任何类型。

## 34. 什么是默认参数？

默认情况下，可以为功能参数分配值。参数不能同时声明为可选参数和默认值。

1 let discount = function (price: number, rate: number = 0.40) {

2 return price \* rate;

3 }

4 discount(500); // Result - 200

5 discount(500, 0.45); // Result - 225

在上面的示例中，rate是折扣函数中默认的参数（如数字）。如果我们在折扣的费率参数中传递该值，它将使用此值，否则将使用默认值0.40。

## 35. 解释一下tsconfig.json文件。

tsconfig.json文件是JSON格式的文件。 在tsconfig.json文件中，您可以指定不同的选项来告诉编译器如何编译当前项目。 目录中包含tsconfig.json文件，表明该目录是TypeScript项目的根目录。

## 36. 什么是泛型？

TypeScript Generics是提供创建可重用组件的方法的工具。 它能够创建可以使用多种数据类型而不是单一数据类型的组件。 而且，它在不影响性能或生产率的情况下提供了类型安全性。 泛型允许我们创建泛型类，泛型函数，泛型方法和泛型接口。

在泛型中，类型参数写在左括号（<）和右括号（>）之间，这使它成为强类型集合。 它使用一种特殊的类型变量来表示类型。

1 function identity<T>(arg: T): T {

2 return arg;

3 }

4 let output1 = identity<string>("edureka");

5 let output2 = identity<number>( 117 );

6 console.log(output1);

7 console.log(output2);

## 37. 接口和类型的差别是什么？

| **接口** |
| --- |

接口声明引入了命名对象类型

可以在extends或implements子句中命名

接口创建一个新名称，该名称随处可见

它可以有多个合并的声明

**类型**

类型别名声明为任何类型的类型引入名称，包括基本类型，联合类型和交集类型

对象类型文字的类型别名不能在扩展或实现子句中命名

他们没有创建任何新名称

它不能有多个合并的声明

## 38. 什么是JSX？

JSX是一种类似于XML的可嵌入语法，可以转换为有效的JavaScript。 JSX在React框架中变得很流行。 TypeScript支持嵌入，类型检查以及直接将JSX编译到JavaScript中。

如果要在文件中使用JSX，则需要使用.tsx扩展名命名文件并启用jsx选项。

## 39. TypeScript支持哪些JSX模式？

TypeScript包含三种JSX模式：

* 保留模式将JSX保留为输出的一部分，以供另一个转换步骤进一步使用。 另外，输出将具有.jsx文件扩展名。
* react模式发出React.createElement，在使用之前不需要经过JSX转换，并且输出将具有.js文件扩展名。
* react-native模式等效于保留模式，它保留所有JSX，但是输出具有.js文件扩展名。

## 40. TypeScript中的环境是什么？何时使用它们？

环境声明告诉编译器其他地方存在的实际源代码。 如果这些源代码在运行时不存在，而我们尝试使用它们，则它将中断而不会发出警告。

环境声明文件类似于docs文件。 如果源更改，则需要保持文档更新，如果环境声明文件未更新，则将出现编译器错误。 而且，它使我们能够安全，轻松地使用现有流行的JavaScript库，例如jquery，angularjs，nodejs等。

## 41. 什么是TypeScript映射文件？

TypeScript映射文件是一个源映射文件，其中包含有关我们原始文件的信息。 .map文件是源映射文件，可让工具在发出的JavaScript代码和创建它的TypeScript源文件之间进行映射。 另外，调试器可以使用这些文件，因此我们可以调试TypeScript文件而不是JavaScript文件。

## 42. 什么是类型断言？

类型断言的工作方式类似于其他语言中的类型转换，但不会执行其他类型的检查或数据重组，例如C＃和Java。 类型转换附带运行时支持，而类型断言对运行时没有影响。 但是，类型断言仅由编译器使用，并向编译器提供有关我们希望如何分析代码的提示。

1 let empCode: any = 007*;*

2 let employeeCode = <number> code*;*

3 console.log(typeof(employeeCode))*; //Output: number*

## 43. 什么是剩余参数？

rest参数用于将零个或多个值传递给函数。 通过在参数前加三点字符（“ ...”）来声明它。 它允许函数具有可变数量的参数，而无需使用arguments对象。 如果参数数量不确定，这将非常有用。

## 44. 声明剩余参数的规则是什么？举例说明。

剩余参数遵循的规则：

* 函数中仅允许一个rest参数。
* 它必须是数组类型。
* 它必须是参数列表中的最后一个参数。

## 45. 解释一下TypeScript中的"as"句法。

“as”是TypeScript中类型断言的附加语法。引入as语法的原因是原始语法与JSX冲突。

1 let empCode: any = 007*;*

2 let employeeCode = code as number*;*

将TypeScript与JSX一起使用时，仅允许使用as样式的断言。

## 46. 解释一下TypeScript中的枚举。

枚举是TypeScipt数据类型，它允许我们定义一组命名常量。 使用枚举使记录意图或创建一组不同的案例变得更加容易。 它是相关值的集合，可以是数字值或字符串值。

## 47. 解释一下相对/非相对模块导入。

相对导入： 相对导入可用于我们自己的模块，这些模块可以保证在运行时保持其相对位置。 相对导入以/、./或../开始。

**1 import** **Entry** **from** "./components/Entry";

2 **import** { **DefaultHeaders** } **from** "../constants/http";

非相对导入： 可以相对于baseUrl或通过路径映射解决非相对导入。 换句话说，导入任何外部依赖项时，我们都使用非相对路径。

**1 import** \* **as** $ **from** "jquery";

**2 import** { **Component** } **from** "@angular/core";

## 48. 什么是匿名函数？

匿名函数是没有任何命名标识符声明的函数。 这些函数在运行时动态声明。 同样，匿名函数可以接受输入和返回输出，就像标准函数一样。 最初创建后通常无法访问。

**1 let** myAdd = **function**(x: number, y: number): number {

2 **return** a + b;

3 };

4 console.**log**(**myAdd**())

## 49. 什么是方法重载？

如果子类或子类具有与父类中声明的方法相同的方法，则称为方法重写。 基本上，它在派生类或子类中重新定义了基类方法。

方法覆盖规则：

* 该方法必须与父类中的名称相同。
* 它必须具有与父类相同的参数。
* 必须存在IS-A关系或继承。

## 50. 什么是Lambda函数（箭头函数）？

ES6版本的TypeScript提供了用于定义匿名函数（即函数表达式）的简写语法。 这些箭头函数也称为lambda函数。 Lambda函数是没有名称的函数。 而arrow函数忽略了function关键字。

**1 let** sum = (x: number, y: number): number => {

2 **return** x + y;

3 }

4 console.**log**(**sum**(10, 20)); *//returns 30*