**一·TypeScript类型整理**

1. **基础数据类型:** number，boolean，string，any，void，null，undefind，never，object，数组，元组，枚举(enum)。
2. **数组:** let list: number[] = [1, 2, 3]; let list: Array<number> = [1, 2, 3];
3. **元组:** 元组类型允许表示一个已知元素数量和类型的数组，各元素的类型不必相同。
4. **enum:** 数字枚举和字符串枚举，数字枚举可以自增长，默认从0开始。
5. **类型推论:** 当赋值时会根据后面的值推断出类型
6. **类型断言:** value: number=<string>value ; val: number=val as string;
7. **泛型:** function identity<T>(arg: T) : T { return arg; }将类型作为变量T传入<>中。

**二·class类**

1. **包含:** 成员属性，成员方法，构造函数。
2. **extends:** 继承，只能单继承，但可以连续继承。

|  |  |  |  |
| --- | --- | --- | --- |
| **基类成员** | **public** | **protected** | **private** |
| **派生类** | **可访问父类成员** | **可访问父类成员** | **不能访问父类成员** |

1. **static静态属性，可以直接用类名.访问属性。**
2. **abstract抽象类:** 抽象类是不允许被实例化的，即不能通过new去实例化，只能被继承。抽象类中的抽象方法不包含具体实现并且必须在派生类中实现。
3. **constructor构造函数:** 在派生类的构造函数中必须调用 super()函数。构造函数可以给实例赋值。
4. **继承类的方法重写:** 类继承后，子类可以对父类的方法重新定义，这个过程称之为方法的重写。其中 super 关键字是对父类的直接引用，该关键字可以引用父类的属性和方法。好像没有@override提示。

**三·interface接口**

1. TypeScript里，接口的作用就是为这些类型命名和为你的代码或第三方代码定义契约。
2. **跳过属性检查:** 在属性后面加“?”,或者将对象赋值给变量。否则，接口中的成员都应该被实现。类型检查器不会去检查属性的顺序，只要相应的属性存在并且类型也是对的就可以
3. **属性接口：**函数实现属性接口用“:”。
4. **类类型接口：**类实现类类型接口用implements，可实现多个接口。
5. **extends:** 一个接口可以继承多个接口，但是只能继承单个类。
6. **接口声明合并:** 可声明相同名字的接口来合并接口成员。

**四·Vue3.0+typescript+Vue-CLI**

1. Vue3.0中的template里面可以分割代码了，不用再使用div包裹。
2. **ref:** ref 对象具有指向内部值的单个属性.value。只要想要在Vue中直接操作DOM元素，就必须用ref属性进行注册。
3. Setup可以接受props和context。
4. Vue-Vite比Vue-Cli编译更快。

**五·遇到的问题？**

1. setup是用来合成api的吗?不知道他的具体意义。
2. Vue中怎么使用namespace?
3. 在Vue3.0中使用Typescript可以更方便在组件之间通信，组件之间可以不再使用bus传值，但是还不清楚具体怎么使用全局参数。
4. 打包了vue3.0项目之后，tsconfig.json应该上传服务器吗。对于项目配置还不是很熟练，应该是还不了解整个项目编译过程。