1. const readonly 区别

- const 防止变量值被修改
- readonly 防止变量属性被修改

2. 接口 和 类型 别名

- 两者都可以描述对象或函数的类型
- 类型别名还能用于其他类型，如 基本类型、联合类型、元组

3. any、never、unknown、null undefined、void 区别

- any
  - 动态类型，失去类型检查
- never
  - 永不存在的值，那些总是会抛出异常 或 根本没有返回值的 函数表达式、箭头函数表达式 的 返回值类型
- unknown
  - 任何值都可以付给他
  - 但他只能给 unknown 和 any
- null undefined
  - 所有类型的子类型
- void 没有任何类型

4. interface 给 Funtion Array Class 做声明方法

```
// 函数声明
interface Say {
  (name: string): viod;
}
let say: Say = (name: string): viod => {};
// Array 声明
interface NumberArray {
  [index: number]: number;
}
let fibonacci: NumberArray = [1, 1, 2, 3, 5];
// Class 声明
interface PersonalIntl {
  name: string;
  sayHi(name: string): string;
}
```

5. union types 的注意

当 TypeScript 不确定一个联合类型的变量到底是哪个类型的时候，我们只能访问此联合类型的所有类型里共有的属性或方法。

```
function getLength(something: string | number): number {
  return something.length;
}
// index.ts(2,22): error TS2339: Property 'length' does not exist on type
>'string | number'.
//   Property 'length' does not exist on type 'number'.

function getString(something: string | number): string {
  return something.toString();
}
// 公共方法和属性可以访问
```

6. TypeScript 如何设计 Class 的声明?

```
class Greeter {
  greeting: string;
  constructor(message: string) {
    this.greeting = message;
  }
  greet(): string {
    return "Hello, " + this.greeting;
  }
}
let greeter = new Greeter("world");
// 在声明类的时候，一般类中都会包含，构造函数、对构造函数中的属性进行类型声明、类中的方
法。
```

7. 如何联合枚举类型的 key

```
enum str {
  A,
  B,
  C,
}
type strUnion = keyof typeof str; // 'A' | 'B' | 'C'
```

8. type 和 interface 的区别?

相同点:

1. 都可以描述 '对象' 或者 '函数'
2. 都允许拓展(extends)

不同点:

1. type 可以声明基本类型，联合类型，元组

2. type 可以使用 typeof 获取实例的类型进行赋值
3. 多个相同的 interface 声明可以自动合并 使用 interface 描述‘数据结构’，使用 type 描述‘类型关系’

9. 泛型

泛型用来来创建可重用的组件，一个组件可以支持多种类型的数据。这样用户就可以以自己的数据类型来使用组件。 简单的说，“泛型就是把类型当成参数”。

10. ?.、??、!、!.、_、** 等符号的含义?

- ?. 可选链 遇到 null 和 undefined 可以立即停止表达式的运行。
- ?? 空值合并运算符 当左侧操作数为 null 或 undefined 时，其返回右侧的操作数，否则返回左侧的操作数。
- ! 非空断言运算符 x! 将从 x 值域中排除 null 和 undefined
- !. 在变量名后添加，可以断言排除 undefined 和 null 类型
- _ 数字分割符 分隔符不会改变数值字面量的值，使人更容易读懂数字 .e.g 1_101_324。
- ** 求幂

11. TypeScript 中同名的 interface 或者同名的 interface 和 class 可以合并吗?

- 同名的 interface 会自动合并，同名的 interface 和 class 会自动聚合。

12. declare，declare global 是什么?

- declare 是用来定义全局变量、全局函数、全局命名空间、js modules、class 等
- declare global 为全局对象 window 增加新的属性

```
declare global {
   interface Window {
        csrf: string;
   }
}
```

13. 对 TypeScript 类中成员的 public、private、protected、readonly 修饰符的理解?

- public: 成员都默认为 public，被此限定符修饰的成员是可以被外部访问；
- private: 被此限定符修饰的成员是只可以被类的内部访问；
- protected: 被此限定符修饰的成员是只可以被类的内部以及类的子类访问;
- readonly: 关键字将属性设置为只读的。 只读属性必须在声明时或构造函数里被初始化。

14. keyof 和 typeof 关键字的作用?

- keyof 索引类型查询操作符 获取索引类型的属性名，构成联合类型。
- typeof 获取一个变量或对象的类型。

15. Exclude、Omit、Merge、Intersection、Overwrite 的作用

- Exclude<T, U> 从 T 中排除出可分配给 U 的元素。
- Omit<T, K> 的作用是忽略 T 中的某些属性。
- Merge<O1, O2> 是将两个对象的属性合并。
- Compute<A & B> 是将交叉类型合并
- Intersection<T, U>的作用是取 T 的属性,此属性同样也存在与 U。
- Overwrite<T, U> 是用 U 的属性覆盖 T 的相同属性。

16. 数组定义两种方式

```
type Foo = Array<string>;
interface Bar {
  baz: Array<{ name: string; age: number }>;
}

type Foo = string[];
interface Bar {
  baz: { name: string; age: number }[];
}
```