**Type Aliases Vs Interface**

In TypeScript, both interfaces and type aliases are used to define custom types, but they have some differences in their behavior and usage.

1. **Syntax**:

Interface: Interfaces are declared using the interface keyword. The syntax for defining an interface is:

interface MyInterface {

// Properties and methods

}

Type Alias: Type aliases are created using the type keyword. The syntax for defining a type alias is:

type MyType = {

// Properties and methods

};

1. **Extensibility**:

Interface: Interfaces can be extended or implemented by other interfaces using the extends keyword. This allows for creating hierarchies of interfaces.

Type Alias: Type aliases cannot be extended or implemented by other types. They represent a direct name for a particular type and cannot be used to create type hierarchies.

1. **Object Literal Type**:

Interface: When defining an interface, you can directly define the structure of an object literal type. This means you can define properties, methods, and their types directly within the interface.

Type Alias: Type aliases are more flexible and can represent various types, including object literals. However, they don't provide a dedicated syntax for defining methods directly within the alias. You would typically use intersection (&) or union (|) types to combine different types.

1. **Declaration Merging**:

Interface: Interfaces in TypeScript support declaration merging. This means you can define multiple interfaces with the same name, and they will be merged together into a single interface.

Type Alias: Type aliases do not support declaration merging. If you define multiple type aliases with the same name, they will result in a type redefinition error.

1. **Usability**:

Interface: Interfaces are commonly used when you want to define the shape of an object or describe the contract that a class should adhere to. They are often used for object-oriented programming concepts like inheritance, polymorphism, and implementing interfaces.

Type Alias: Type aliases are useful when you want to create a name for a complex type or create unions, intersections, or mapped types. They are more flexible and can represent a wider range of type definitions.

In general, if you need to define the structure of an object or describe a contract, interfaces are a good choice. If you need to create a name for a complex type or combine multiple types, type aliases can be more suitable. However, the choice between them ultimately depends on the specific use case and personal preference.