An **interface** in programming defines a contract or blueprint for classes. It specifies a set of methods (or properties) that implementing classes must provide, without dictating how these methods are implemented. Interfaces are widely used in object-oriented programming to achieve **abstraction** and **polymorphism**.

Features of an Interface:

1. **No Implementation**: Only method declarations, no method bodies.
2. **Multiple Inheritance**: A class can implement multiple interfaces.
3. **Ensures Consistency**: Forces implementing classes to follow a defined structure.

Example of An Interface in Typescript:

```
// Define an Interface
interface Animal {
    makeSound(): void;  // Method signature
    eat(): void;
}

// Implement the Interface in a Class
class Dog implements Animal {
    makeSound(): void {
        console.log("Dog barks");
    }

    eat(): void {
        console.log("Dog eats bones");
    }
}

// Another Class Implementing the Interface
```

```
class Cat implements Animal {
    makeSound(): void {
        console.log("Cat meows");
    }

    eat(): void {
        console.log("Cat eats fish");
    }
}

// Using the Classes
const myDog: Animal = new Dog();
const myCat: Animal = new Cat();

myDog.makeSound(); // Output: Dog barks
myDog.eat();       // Output: Dog eats bones

myCat.makeSound(); // Output: Cat meows
myCat.eat();       // Output: Cat eats fish
```