

# Interface

- An Interface is defined using the `interface` keyword
- Interfaces are used only during compilation time to check types
- By convention, interface definitions start with an `I`, e.g. : `IPoint`
- Interfaces are used in classical object oriented programming as a design tool
- Interfaces don't contain implementations
- They provide definitions only
- When an object implements an interface, it must adhere to the contract defined by the interface
- An interface defines what properties and methods an object must implement
- If an object implements an interface, it must adhere to the contract. If it doesn't the compiler will let us know.
- Interfaces also define custom types

## Basic Interface

Below is an example of an Interface that defines two properties and three methods that implementers should provide implementations for:

```
1 interface IMyInterface {  
2     // some properties  
3     id: number;  
4     name: string;  
5  
6     // some methods  
7     method(): void;  
8     methodWithReturnVal(): number;  
9     sum(nums: number[]): number;  
10 }
```

Using the interface above we can create an object that adheres to the interface:

```
1 let myObj: IMyInterface = {  
2   id: 2,  
3   name: 'some name',  
4  
5   method() { console.log('hello'); },  
6   methodWithReturnVal () { return 2; },  
7   sum(numbers) {  
8     return numbers.reduce( (a,b) => { return a + b } );  
9   }  
10 };
```

Notice that we had to provide values to **all** the properties defined by the Interface, and the implementations for **all** the methods defined by the Interface.

And then of course you can use your object methods to perform operations:

```
1 let sum = myObj.sum([1,2,3,4,5]); // -> 15
```