# Decorators

- Decorators can be used to add additional properties and methods to existing objects.
- Decorators are a declarative way to add metadata to code.
- There are four decorators: ClassDecorator, PropertyDecorator, MethodDecorator, ParameterDecorator
- TypeScript supports decorators and does not know about Angular's specific annotations.
- Angular provides annotations that are made with decorators behind the scenes

# Method Decorators

Goals: - make a method decorator called `log` . - Decorate `someMethod` in a class using `@log`

```
class SomeClass {
  @log
  someMethod(n: number) {
    return n * 2;
  }
}
```

In the usage, `someMethod` has been decorated with `log` using the `@` symbol. `@log` is decorating `someMethod` because it is placed right before the method.

- Decorator Implementation:

```
function log(target: Function, key: string, value: any) {
  return {
    value: function (...args: any[]) {
      var a = args.map(a => JSON.stringify(a)).join();
      var result = value.value.apply(this, args);
      var r = JSON.stringify(result);
      console.log(`Call: ${key}(${a}) => ${r}`);
      return result;
```

```
      }
    };
}
```

A method decorators takes a 3 arguments:

- `target` : the method being decorated.
- `key` : the name of the method being decorated.
- `value` : a property descriptor of the given property if it exists on the object, undefined otherwise. The property descriptor is obtained by invoking the `Object.getOwnPropertyDescriptor` function.

**TODO**

- Add decorator content for each type.