Getters and Setters

Classes allow using **getters** and **setters**. It is smart to use **getters** and **setters** for the properties, especially if you want to do something special with the value before returning them, or before you set them. To add **getters** and **setters** in the class, use the `get` and `set` keywords.

The `getter` is executed on an attempt to get the field value, while `setter` on an attempt to set a value.

```
class User {
    constructor(name) {
      this.name = name;
    }
    get getName() {
      return this.name;
    }
    set setName(name) {
      if (name === '') {
        throw new Error(`name field of User cannot be empty`);
      }
      this.name = name;
    }
}
  const user = new User('Jon Snow');
  console.log(user.getName); // The getter is invoked, => 'Jon Snow'
  user.setName = 'Jon White'; // The setter is invoked
  console.log(user.getName);  // The getter is invoked = > 'Jon White'
  user.setName = ''; // The setter throws an Error
```

```
// Output
Jon Snow
Jon White
Error: name field of User cannot be empty
```

`get getName() {...}` getter is executed when you access the value of the field: `user.getName`.

While `set setName(name) {...}` is executed when the field is updated `user.setName = 'Jon White'`. The setter throws an error if the new value is an empty string.