Methods

The ability to modify data is performed by special functions that are a part of the class: ***the methods***.

The JavaScript classes support both **`instance`** and **`static`** methods.

Instance Methods

Instance methods can access and modify instance data. Instance methods can call other instance methods, as well as any static method.

For example, let's define a method `getName()` that returns the name in the `User` class:

```
class User {
  name = 'Unknown';

  constructor(name) {
    this.name = name;
  }

  getName() {
    return this.name;
  }
}

const user = new User("Jon Snow");
console.log(user.getName());
```

```
// Output
Jon Snow
```

`getName() { ... }` is a method inside the `User` class. `user.getName()` is a method invocation: it executes the method and returns the computed value if any.

In a class method, as well as in the constructor, `this` value equals to the class instance. Use `this` to access instance data: `this.field`, or even call other methods: `this.method()`.

Let's add a new method `nameContains(str)` that has one parameter and calls another method:

```
class User {
  name;

  constructor(name) {
    this.name = name;
  }

  getName() {
    return this.name;
  }

  nameContains(str) {
    return this.getName().includes(str);
  }
}

const user = new User("Jon Snow");
console.log(user.nameContains("Jon"));
console.log(user.nameContains("Stark"));
```

```
// Output
true
false
```

`nameContains(str) { ... }` is a method of `User` class that accepts one parameter `str`. More than that, it executes another method of the instance `this.getName()` to get the user's name.