Private Methods

In this lesson, we'll learn how to implement private functions in an object.

WE'LL COVER THE FOLLOWING

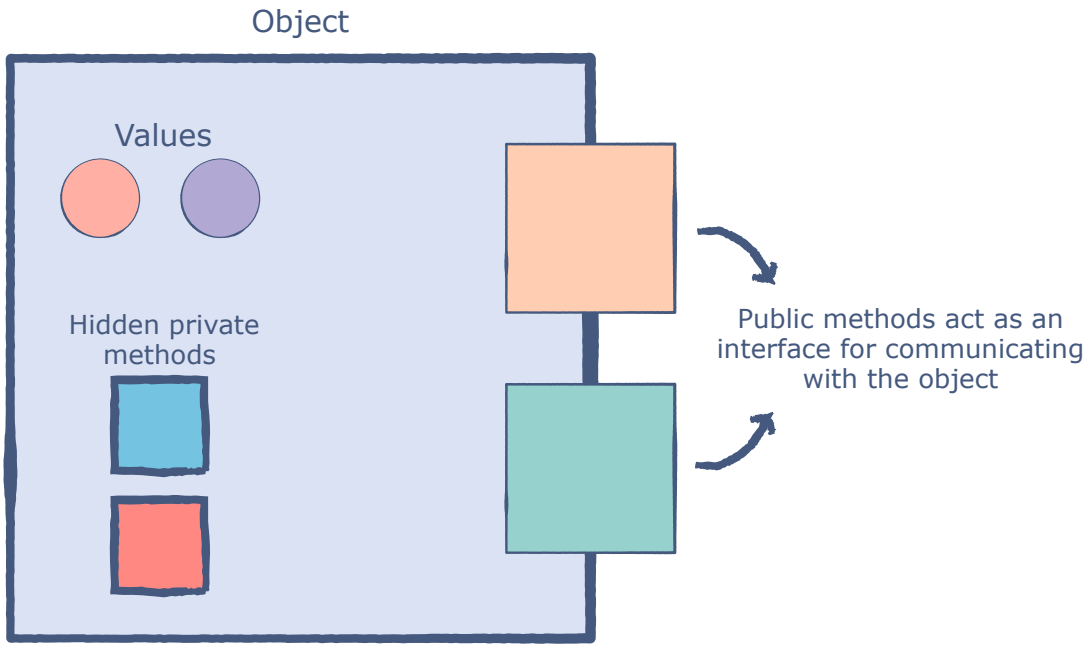
What are Private Methods?

In the last lesson, we implemented the `rectangle` type and created an object out of it. All of its functions were public.

```
type rectangle = {
  .
  getColor: string,
  getLength: float,
  getWidth: float,
  getArea: float
};

let rect: rectangle = {
  /* Values */
  val l = 10.5;
  val w = 5.5;
  val c = "Blue";

  /* Function definitions */
  pub getColor = c;
  pub getWidth = w;
  pub getLength = l;
  pub getArea = this#getWidth *. this#getLength
};

Js.log(rect#getColor);
Js.log(rect#getArea);
```

Sometimes, we need to create functions for our own ease. These functions

should not be exposed like their public counterparts since they are not a part of the object's interface. Such functions are said to be **private**.



A private method can only be used as part of another function. They do not need to be declared in the type definition.

Creating a Private Method

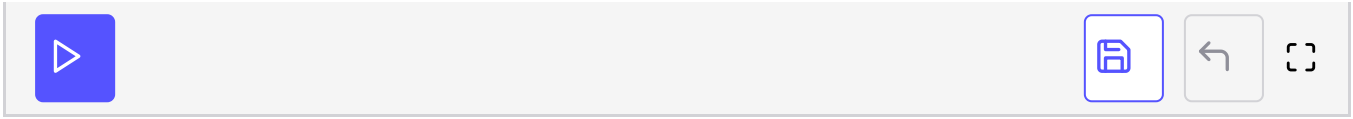
Let's add the public `getDetails()` method to our `rectangle` class. This will return all the values of the object in the form of a tuple.

To help us in this process, we'll create the private `createTuple` method.

Looking at the code will help us understand things better:

```
type rectangle = {
  .
  getColor: string,
  getLength: float,
  getWidth: float,
  getArea: float,
  getDetails: (float, float, float, string)
};

let rect: rectangle = {
  /* Values */
  val l = 10.5;
  val w = 5.5;
  val c = "Blue";

  /* Function definitions */
  pub getColor = c;
  pub getWidth = w;
  pub getLength = l;
  pub getArea = this#getWidth *. this#getLength;
  pub getDetails = this#createTuple;
```

```
  pri createTuple = (this#getWidth, this#getLength, this#getArea, this#getColor);

};

Js.log(rect#getDetails);
```



And it's as simple as that! The `createTuple` method only assists us in getting an internal job done. Hence, it can be kept private.

> **Note**: Private methods cannot be accessed from the outside using the `#` operator.

The next lesson will highlight the difference between **open** objects and **closed** objects.