Functions in Classes

In this lesson, you will cover the concept of functions in classes.

Functions and `class` in a nutshell

Functions written inside a class are a little bit different. In the next chapter, you will talk about the object and all the rich possibilities that TypeScript brings. In short, functions are similar but don't use the keyword `function`. Instead, the explicit visibility of the function is used or sometimes, nothing at all.

The following code defines 3 functions. **Line 2** is a private function. A private function can only be invoked inside the class. **Line 3** is a public function. It can be called inside and outside the function. Finally, **line 4** is a protected function. It can be called from the class but also from any class that extends the class.

```
class MyClass{
    private myFunction1(){}
    public myFunction2(){}
    protected myFunction3(){}
}
```

Private functions

A private function role is to have logic that belongs to the class but should not be exposed outside. The function can access any private members, public members or protected members.

The class defined from **line 1** to **line 8** has a constructor that calls the `private` function. The function is not accessible from outside as **line 11** shows. Uncommenting the line will not provide anything to access.

The instantiation of the class, at **line 10**, can access the private function, hence output the message.

```
class MyPrivateFunctionClass {
  constructor() {
    this.privateFunction();
  }
  private privateFunction(): void {
    console.log("From the private function");
  }
}

const c = new MyPrivateFunctionClass();
// c. // There is nothing to invoke because the function is private
```

Private vs `#`

With **TypeScript version 3.8** and upward, it is possible to use `#` for a private function. The addition comes from a new specification of ECMAScript which TypeScript is based upon. There is a difference between `#` and `private` which is that the private function does not translate to anything private when transpile into JavaScript. However, with `#` the code, if invoked outside the class in JavaScript, will throw an exception. The version is not yet released, thus stay tuned!

Protected function

A protected function is between a `public` and a `private` function. It can be called from within the class as well as by a subclass. **Line 11** calls the protected function defined at **line 3**.

```
class ClassA {
  private a1: number = 1;
  protected a2fct(): void {
    console.log(this.a1);
  }
}

class ClassB extends ClassA {
  private b1: number = 2;
  protected b2(): void {
    super.a2fct(); // Call a protected function from a base class
  }
}

const cab = new ClassB();
// cab. // No access to b2 or a2fct
```

Public function

A public function is a function that can be called from everywhere: inside, outside or from a base class. The public function at **line 3** and at **line 10** can be called from outside the class as seen in **line 16-17**.

```
class ClassC {
  private a1: number = 1;
  public a2fct(): void {
    console.log(this.a1);
  }
}

class ClassD extends ClassC {
  private b1: number = 2;
  public b2(): void {
    super.a2fct(); // Call a protected function from a base class
  }
}

const cd = new ClassD();
cd.a2fct(); // Can call base public function
cd.b2(); // Can call the class public function
```

A function can be of several types of visibility when used inside a class. It is up to the right usage and goal to choose which one. In all cases, there is no use to specify the keyword `function`.