TypeScript Access Modifiers

Summary: in this tutorial, you will learn about the access modifiers in TypeScript including private, protected, and public.

Access modifiers change the visibility of the properties and methods of a class. TypeScript provides three access modifiers:

- private
- protected
- public

Note that TypeScript controls the access logically during compilation time, not at runtime.

The private modifier

The `private` modifier limits the visibility to the same class only. When you add the `private` modifier to a property or method, you can access that property or method within the same class. Any attempt to access private properties or methods outside the class will result in an error at compiled time.

The following example shows how to use the `private` modifier to the `snn`, `firstName`, and `lastName` properties of the `person` class:

```
class Person {
  private ssn: string;
  private firstName: string;
  private lastName: string;
```

```
  // ...
}
```

Once the `private` property is in place, you can access the `ssn` property in the constructor or methods of the `Person` class. For example:

```
class Person {
  private ssn: string;
  private firstName: string;
  private lastName: string;

  constructor(ssn: string, firstName: string, lastName: string) {
    this.ssn = ssn;
    this.firstName = firstName;
    this.lastName = lastName;
  }

  getFullName(): string {
    return `${this.firstName} ${this.lastName}`;
  }
}
```

The following attempts to access the `ssn` property outside the class:

```
let person = new Person('153-07-3130', 'John', 'Doe');
console.log(person.ssn); // compile error
```

The public modifier

The `public` modifier allows class properties and methods to be accessible from all locations. If you don't specify any access modifier for properties and methods, they will take the `public` modifier by default.

For example, the `getFullName()` method of the `Person` class has the `public` modifier. The following explicitly adds the public modifier to the `getFullName()` method:

```
class Person {
    // ...
    public getFullName(): string {
        return `${this.firstName} ${this.lastName}`;
```

```
    }
    // ...
}
```

It has the same effect as if the `public` keyword were omitted.

The protected modifier

The `protected` modifier allows properties and methods of a class to be accessible within the same class and subclasses.

When a class (child class) inherits from another class (parent class), it is a subclass of the parent class.

The TypeScript compiler will issue an error if you attempt to access the protected properties or methods from anywhere else.

To add the protected modifier to a property or a method, you use the `protected` keyword. For example:

```
class Person {
    protected ssn: string;

    // other code
}
```

The `ssn` property now is protected. It will be accessible within the `Person` class and in any class that inherits from the `Person` class. You'll learn more about inheritance here.

The `Person` class declares the two private properties and one protected property. Its constructor initializes these properties to three arguments:

```
class Person {
  protected ssn: string;
  private firstName: string;
  private lastName: string;

  constructor(ssn: string, firstName: string, lastName: string) {
    this.ssn = ssn;
    this.firstName = firstName;
```

```
    this.lastName = lastName;
  }

  getFullName(): string {
    return `${this.firstName} ${this.lastName}`;
  }
}
```

To make the code shorter, TypeScript allows you to both declare properties and initialize them in the constructor like this:

```
class Person {
  constructor(
    protected ssn: string,
    private firstName: string,
    private lastName: string
  ) {}

  getFullName(): string {
    return `${this.firstName} ${this.lastName}`;
  }
}
```

When you consider the visibility of properties and methods, it is a good practice to start with the least visible access modifier, which is private.

Summary

- TypeScript provides three access modifiers to class properties and methods: `private`, `protected`, and `public`.
- The `private` modifier allows access within the same class.
- The `protected` modifier allows access within the same class and subclasses.
- The `public` modifier allows access from any location.
- Properties and methods have `public` access if you omit the access modifiers.