Using access modifiers

In this lesson, we'll learn the different access modifiers that are available on class members and how to implement them.

WE'LL COVER THE FOLLOWING

We are going to continue the implementation of the `Counter` component we worked on in the last lesson.

Click the link below to open the exercise in CodeSandbox:

CodeSandbox project

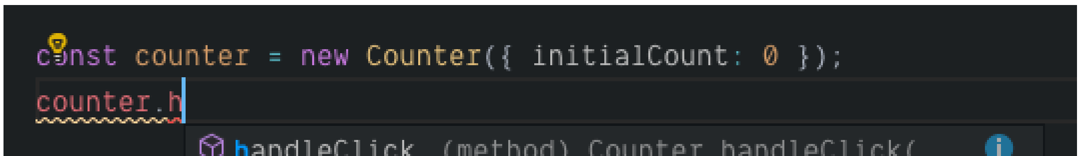
This is the component we implemented in the last lesson with the button click handler extracted into a method within the class component. There is also a class property called `clicked`, which tracks how many times the button has been clicked.

Public members

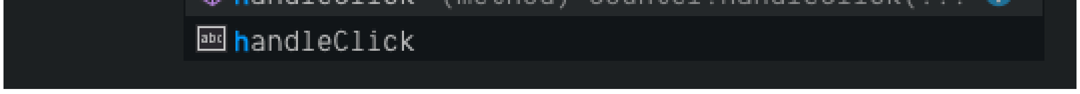
At the bottom of `index.tsx` add the following statement:

```
const counter = new Counter({ initialCount: 0 });
```

On the next line, inspect the members that are available in `counter`:



```
const counter = new Counter({ initialCount: 0 });
counter.h
```



```
handleClick
```

We can see that both `handleClick` and `clicked` are accessible when we don’t want them to be. We’ll resolve this later in the lesson.

By default, a method or property in a class is accessible by the consumer of the class if no access modifier is specified. So, the default access modifier is `public`.

To specify a `public` access modifier on a member, we put the keyword `public` in front of the member.

In the `Counter` component, explicitly specify the access modifier on the public members.

Show Answer

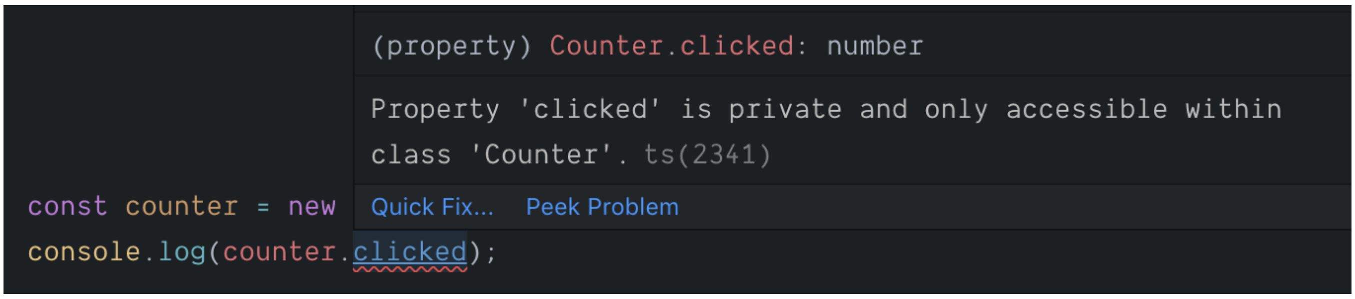
Private members

We must use the `private` keyword in front of a class member to make it private.

In the `Counter` component, make the `clicked` property and `handleClick` method private. Also make the static property, `defaultProps`, private as well.

Show Answer

Verify that these members are private by trying to reference them by a consumer of the class.



```
(property) Counter.clicked: number
Property 'clicked' is private and only accessible within
class 'Counter'. ts(2341)
```

```
const counter = new
console.log(counter.clicked);
```

At runtime, will these private members be accessible?

Show Answer

Protected members

The final access modifier is `protected`. What do you think this does?

Show Answer

Wrap up

Good stuff! We now know how to enforce class member access using the `public`, `private`, and `protected` accessors. We are also aware that it is TypeScript enforcing this access at development time and not at runtime.

Next, let’s double-check what we have learned from the last few lessons with a quiz.