Fixing `noImplicitAny` Errors

This lesson discusses techniques of addressing compilation errors caused by enabling `noImplicitAny`.

WE'LL COVER THE FOLLOWING

Explicit `any` annotation

The simplest way to fix errors caused by enabling `noImplicitAny` is simply explicitly marking the type as `any`. How is this better than skipping the type annotation altogether? It's still unsafe, right?

Correct; but with explicit annotations, it's much easier to spot these occurrences (for example, during code reviews). Because of the propagating aspect of `any`, it's easy to not realize that some portion of code is not type-checked. Making `any` explicit addresses this issue.

```
function sayHello(person: any) {
  console.log(`Hello, ${person.name}`);
}
```

Run the code to see that there are no compile errors thanks to explicit `any` annotation. `noImplicitAny` enabled.

`TODO` aliasing technique

There is a useful trick that can help you make `any` types stand out even more. It may happen that you need to introduce the `any` annotation because you don't have enough time or context to create a proper type for some variable

don't have enough time or context to create a proper type for some variable

or function parameter. In this case you can create an alias for `any` that starts with the `TODO` prefix.

```
type TODOPerson = any;

function sayHello(person: TODOPerson) {
  console.log(`Hello, ${person.name}`);
}
```

One of the advantages of creating this alias is that it's even easier to spot. Furthermore, by reusing the same alias in several places, you clearly indicate that the same type should be used in these places. Using `any` wouldn't convey this information.

Exercise

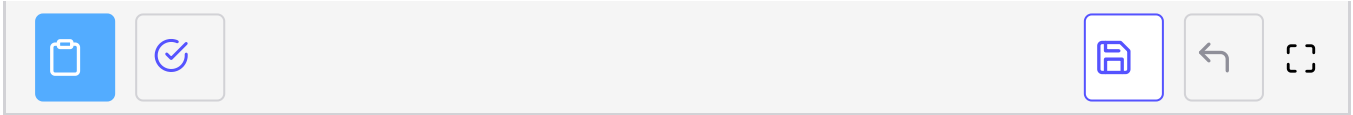
Identify places that would trigger an error with `noImplicitAny` flag enabled. Fix the code to avoid errors.

*Note that the system can only verify that your code compiles without error. To check your solution, click on **Show solution** and compare the code.*

```
class CounterComponent {
  count;

  constructor(private loggingService) {}

  increment(by) {
    this.loggingService.log(`Incrementing by ${by}`);
    this.count = this.count + by;
  }
}
```



In the next lesson, we'll talk about a type-safe alternative to `any`, the `unknown` type.