Idioms and Patterns: Tag Dispatching

In this lesson, we'll learn about tag dispatching in idioms and patterns.

WE'LL COVER THE FOLLOWING

- Tag Dispatching

Tag Dispatching

Tag Dispatching enables us to choose a function based on type characteristics.

- The decision takes place at compile-time.
- Traits make the decision based on specific properties of an argument.
- The main benefit is performance.

The iterator categories from the Standard Template Library are a typical use-case:

```
struct input_iterator_tag{};
struct output_iterator_tag{};
struct forward_iterator_tag: public input_iterator_tag{};
struct bidirectional_iterator_tag: public forward_iterator_tag{};
struct random_access_iterator_tag: public bidirectional_iterator_tag{};
```

In the concrete case, `std::random_access_iterator` is a refinement of `std::bidirectional_iterator`, `std::birectional_iterator` is a refinement of `std::forward_iterator`, and `std::forward_iterator` is a refinement of `std::input_iterator`.

In the next lesson, we'll look at an example of tag dispatching in idioms and patterns.