What is it?

A `span<T>` is:

- A very lightweight abstraction of a contiguous sequence of values of type `T` somewhere in memory.
- Basically a `struct { T * ptr; std::size_t length; }` with a bunch of convenience methods.
- A non-owning type (i.e. a "reference-type" rather than a "value type"): It never allocates nor deallocates anything and does not keep smart pointers alive.

It was formerly known as an `array_view` and even earlier as `array_ref` .

When should I use it?

First, when *not* to use it:

- Don't use it in code that could just take any pair of start & end iterators, like `std::sort` , `std::find_if` , `std::copy` and all of those super-generic templated functions.
- Don't use it if you have a standard library container (or a Boost container etc.) which you know is the right fit for your code. It's not intended to supplant any of them.

Now for when to actually use it:

> Use `span<T>` (respectively, `span<const T>`) instead of a free-standing `T*` (respectively `const T*`) for which you have the length value. So, replace functions like:
>
> ```
> void read_into(int* buffer, size_t buffer_size);
> ```
>
> with:
>
> ```
> void read_into(span<int> buffer);
> ```

Why should I use it? Why is it a good thing?

Oh, spans are awesome! Using a `span` ...