Document	P1339R0
Date	2019-01-04
Author	CJ Johnson <johnsoncj@google.com>
Audience	Library Evolution Working Group (LEWG)

Disallowing the friending of names in namespace std

From Hyrum's Law [1]:

> With a sufficient number of users of an API,
> it does not matter what you promise in the contract:
> all observable behaviors of your system
> will be depended on by somebody.

Background

`absl::FixedArray<T, N, A>` is a `std::array<T, N>`-like type that takes advantage of the Small Buffer Optimization (SBO). In an effort to make the type exception safe while still allowing it to be compiled with exceptions turned off, work was done to update the type to switch between two implementations (using macros) conditional on the presence of exceptions. As a result, the code that called the constructor of `T` was relocated from inside the class to a non-member function.

While this change did not intentionally affect the API of FixedArray, as a result of Hyrum's Law, it was not only observed but it caused a breakage in at least one case. The breakage came from a class depending on FixedArray's internals being entirely contained within its own member functions. The class in question had a private default constructor and friended FixedArray (such as `friend absl::FixedArray<T>;`). The private default constructor used to be called by FixedArray directly and so friending the type compiled properly. Now that `T`'s constructor was being called from a free function, the fact that the constructor was private caused a build break.

The important takeaway from this experience is to never friend a type you don't own unless the author of that type tells you to. After bringing this up at CppCon 2018 [2], the below wording change to SD-8 was proposed.

Wording change to SD-8

The standard doesn't generally deal in implementation details for library types; however, SD-8 is the current vehicle for clearly telling users what not to rely upon. Issues such as the one expressed above will hopefully remain infrequent, but given that Abseil has already experienced it, we'd like to apply an ounce of prevention for the standard library. Thus, the below green-highlighted lines should be added to SD-8.

```
[...]

Primarily, the standard reserves the right to:

* Add new names to namespace std
* Add new member functions to types in namespace std
* Add new overloads to existing functions
* Add new default arguments to functions and templates
* Change return-types of functions in compatible ways (void to anything, numeric
  types in a widening fashion, etc).
+ * Assume that users have not given types in namespace std access to private
+   members of their types unless otherwise permitted by the standard
* Make changes to existing interfaces in a fashion that will be backward
  compatible, if those interfaces are solely used to instantiate types and
  invoke functions. Implementation details (the primary name of a type, the
```

```
   implementation details for a function callable) may not be depended upon.

[...]
```

References

[1] Hyrum's Law

[2] CppCon 2018: Titus Winters "Standard Library Compatibility Guidelines (SD-8)" @ 54m56s