

+ 25

How to Affect the Future of C++ Standard in 60 Minutes

RIVER WU



20
25



About river :)

- Recent graduate of University of Michigan Ann Arbor.
- Core contributor of the beman project
- Building accelerators at Tenstorrent
- Looking for cool new friends
 - Email: river.wu@wusatosi.com
 - LinkedIn: Xueqing Wu
 - LinkedIn QR:



Looking for new colleges!

If you have experience in embedded engineering, automotive, HPC, performance critical applications, or just simply cool, come talk to me!

*I do not represent my employer



Who I am not (yet)

- An ohio state graduate
- C++ developer with decades of experience
- An ISO member who knows the ins and outs of the standardization process.



Urban Planning

NEW YORK

BUSINESS PORT OF NEW YORK CITY
ON ENLARGED SCALE.

(RAND-McNALLY & CO. SECTION) NYC

A detailed historical map of New York City, showing the business portion on an enlarged scale. The map features a grid of streets, water bodies like the Hudson River and New York Bay, and various landmarks. Overlaid on the map is the text: 'Affects everyone', 'Term visionary & practical investment', 'Hard to revert bad choices', and 'Make or break a place'. The map includes a coordinate grid with letters A through Y along the bottom and numbers 19 through 36 along the left side. The title 'BUSINESS PORTION OF NEW YORK CITY ON ENLARGED SCALE' is visible at the bottom center. The publisher's name 'RAND-McNALLY & SONS' is at the bottom right, along with 'NYC' and a small logo.

Affects everyone

Term visionary & practical investment

Hard to revert bad choices

Make or break a place

BUSINESS PORTION OF
NEW YORK CITY
ON ENLARGED SCALE.

(RAND-McNALLY & SONS)
NYC

Affects everyone

Long Term visionary & practical investment

Hard to revert bad choices

Make or break a language



Influencing Urban Planning

Public Engagement

Problem with accessibility

1. Meetings take a long time
2. You don't know how a plan pans out in 10 years
3. The plan is not easy to understand.

Results in engagement from highly opinionated individuals, not a mixture of ideas

Language Evolution

Engaging in standardization should be a lot easier:

1. The impacted are professionals of the field.
2. Should be doable online, no need to walk to a library/ city council.
3. There are modern tools chains to provide feedback

How do one usually
explore a new library?

Language Evolution

Access difficulty still persists

Access difficulty as outsider:

1. Standardese is not code
2. Standardese is cryptic
3. The standardization process is opaque.
4. There is not a central place to look up implementation and engage in discussion.
5. Feels off...

We need a more
accessible way to
evaluate library
proposals as libraries
using modern toolchains.

What is
the beman project.

A collection of library
proposal
implementations ready
for production.

An accessible platform
of feedback cycle and
discussion.

A welcoming
community of folks
interested in
advancing the C++
standard.

To support the efficient design and adoption of the highest quality C++ standard libraries through *implementation experience, user feedback,* and *technical expertise.*

For the greater C++ community

With Beman Project

Accessible, central, standard,
welcoming.

1. Library forward, easy to experiment
2. Accessible navigation, feedback and discussion
3. A visible, open and inclusive community with expertises and breadcrumbs

For Library Implementers

With Beman Project

1. Greater Exposure
2. Community of expertise
3. Able to focus more on implementation, less on GitHub/ infra

Library Evolution Chair (Inbal Levi)

github.com/cplusplus/LEWG

Paper Forwarding Checklist

Inbal Levi edited this page on Aug 14 · [4 revisions](#)

The following should be reviewed before taking a forwarding poll:

- Why this facility should go into the standard library?
- Discussion of prior art?
- Examples?
- Field experience?
 - Implementation experience?
 - Usage experience?
 - Deployment experience?
- Performance considerations?
- Support for "std::format", "hash", and allocators (if required)?
- Feature test macro?
- `constexpr` ?
- Freestanding?
- Changes Library Evolution previously requested?
- Wording?

What have the beman project done this past year?

Library Highlight

beman.optional

`std::optional<T&>` — Standardizing Optionals over References

A Case Study

Steve Downey

beman.optional: overview

- One of the first beman project.
- Currently the most matured beman project.
- Fills an intentionally left open hole in `std::optional`
- Implements `std::optional<T&>` and Give `std::optional` Range Support
- `std::optional<T&>` and Give `std::optional` Range Support are already in C++26 draft.

beman.optional: example

```
std::optional<Cat&> cat = find_cat("Ember");  
cat.and_then([](Cat& thecat){  
    /* mutate the cat. */});
```

Beman.optional: example #2 (P3091)

```
template<class Key, class T, class Compare = less<Key>,  
        class Allocator = allocator<pair<const Key, T>>>  
class map {  
    // ...  
    optional< mapped_type&> get(const key_type& k);  
    optional<const mapped_type&> get(const key_type& k) const;  
    //...  
};
```

Before	After
<pre>auto iter = m.find(k); T x = iter == m.end() ? T{} : iter->second;</pre>	<pre>T x = m.get(k).value_or(T{});</pre>

beman.optional:

1. It exposed library author's optional implementation to more attention.
2. This lead to external contributor discovering a major bug in the implementation that also exist in other optional implementations. (Optional can be stolen!)
3. Library Author: "I don't have to understand more GitHub Actions than I need to".
4. There are still left over infrastructure cleaning work needs to be done.

Use of badge / icon to indicate maturity

beman.optional: C++26 Extensions for std::optional

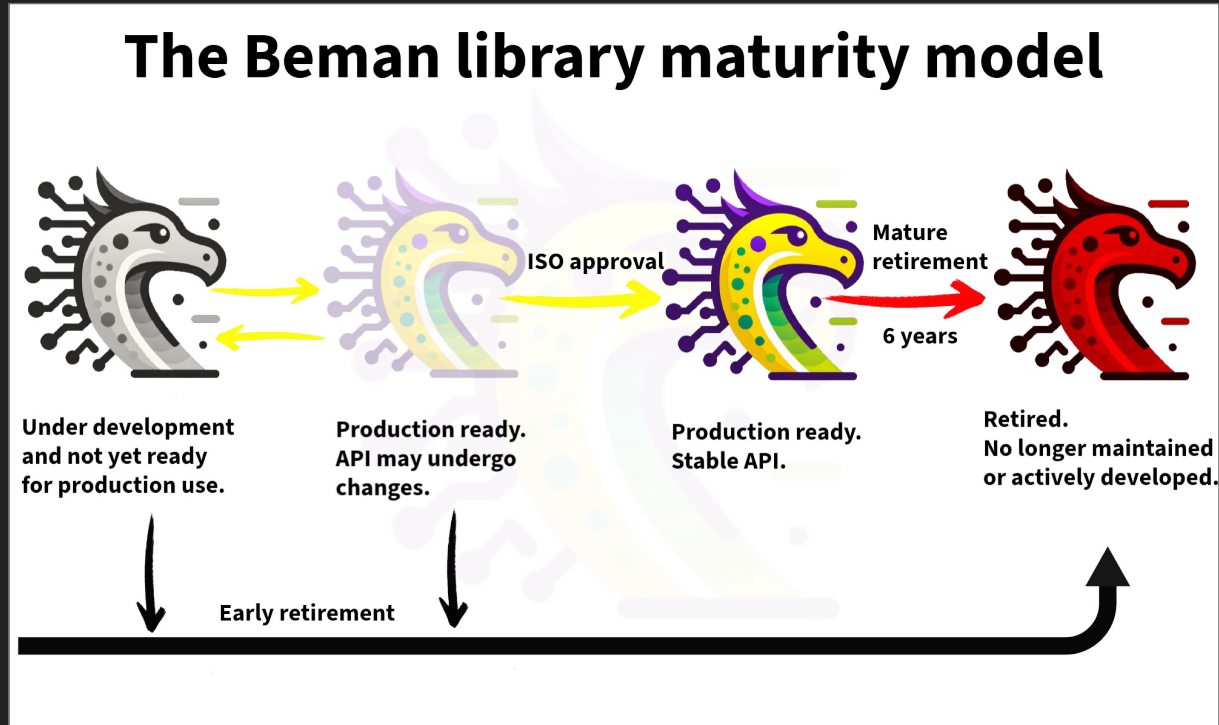
Library status **production ready (api may undergo changes)** Standard Target **C++26**  Try it on Compiler Explorer

This repository implements `std::optional` extensions targeting C++26. The `beman.optional` library aims to evaluate the stability, the usability, and the performance of these proposed changes before they are officially adopted by WG21 into the C++ Working Draft. Additionally, it allows developers to use these new features before they are implemented in major standard library compilers.

Implements: `std::optional<T>` (P2988R12) and [Give std::optional Range Support \(P3168R2\)](#).

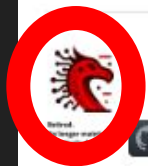
Status: [Production ready. API may undergo changes.](#)

The lifetime of a beman project



std::dump P2879

beman.dump: A tool for dumping and to standard output



Continuous Integration Tests **passing**

`beman::dump::dump()` prints its arguments space-separated with a new-line. utility. A call to `beman::dump::dump(arg1, arg2, ..., argn)` is equivalent to `std::cout << arg1, arg2, ..., argn`

Implements: [Proposal of std::dump \(P2879R0\)](#)

Status: [Retired. No longer maintained or actively developed.](#)

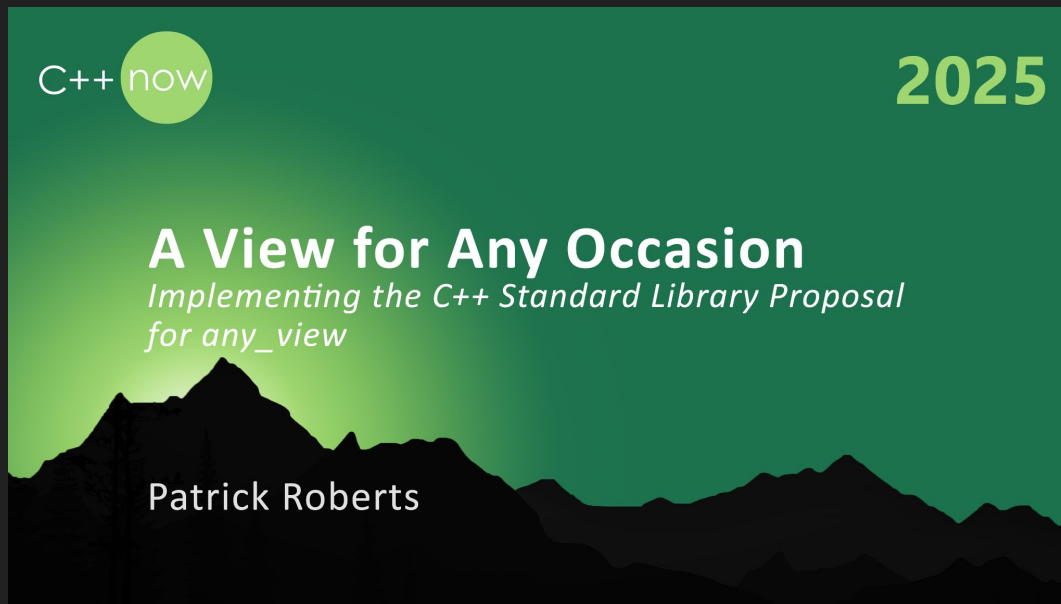


Retired.
No longer maintained
or actively developed.

beman.any_view

beman.any_view: overview

- Provides a type-erased range wrapper that expresses assumptions for the underlying range.
- Targets C++29.



beman.any_view: example

For a function that consumes a range of element type A and produces a range of element type B:

```
// With current facility  
auto map(R &&) { /* */ };
```

```
// With any_view  
any_view<B> map(any_view<A>) { /* */ };
```

beman.any_view:

- Library author: “Now I have a community that is obligated to talk to me”.
- Library author is looking for more use case examples of any_view.
- Library author is trying to understand the performance impact of any_view on real world applications.

There's a lot more!

- `beman.lazy`
- `beman.exemplar`
- `beman.scope`
- `beman.any_view`
- `beman.execution`
- `beman.optional`
- `beman.inplace_vector`
- `beman.elide`
- `beman.net`
- `beman.utf_view`
- `beman.iterator_interface`
- `beman.dump`
- `beman.task`
- 16 libraries
- 2379 commits
- 272 issues
- 572 pull requests
- 44 distinct contributors
- 111 Discourse threads
- 137 Discourse members

A lot of the projects are on Compiler Explorer!

These projects are all on compiler explorer:

- `beman.exemplar`
- `beman.optional`
- `beman.iterator_interface`
- `beman.any_view`
- `beman.inplace_vector`
- `beman.net`
- `beman.execution`



Try it on Compiler Explorer

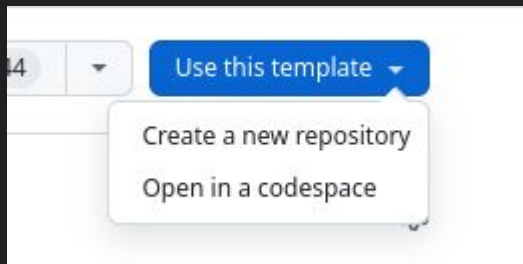
If you're interested in helping any individual library.

All library authors are looking for real world use examples of their libraries.

Post your feedback on GitHub issue help a lot!

This is a good time to contribute a new library!

beman.exemplar



Beman's answer to minimize setup fractions for library authors.

Bootstrapping beman.exemplar

1. Click “use this template” on GitHub and create a GitHub Repo
2. Clone the newly created repo (or open it in codespace)
3. Invoke cookie cutter and input necessary parameters
4. Start hacking!

```
riverwu@River-Wu's-Mac cookiecutter % ln -s ../inplace_vector
riverwu@River-Wu's-Mac cookiecutter % cookiecutter .
[1/6] project_name (my_project_name): optional
[2/6] cpp_build_version (26): 20
[3/6] paper (PnnnnRr): P2988
[4/6] owner (github-user-name): fake-steve
[5/6] description (Short project description.): Optional with a ref
[6/6] godbolt_link (https://www.example.com):
```

Comes with the template: CMake



Pragmatic CMake

BRET BROWN



2025 | 
September 13 - 19

Comes with the template: CMake Preset

- Provides a one line approach to run cmake config, build and test.
- Provides a combination between compilers (gnu, llvm, apple-clang, msvc) and two flavors of build configuration (debug, release).
- Example use case:
- `cmake --workflow --preset gcc-debug`

Comes with the template: CMake Preset

- debug presets (e.g. gcc-debug) compiles the project with address sanitizers and all sanitizers that does not conflict with address sanitizer.
- This usually means everything besides the thread sanitizer.
- debug presets are portable over compilers (even MSVC!)
- These sanitizer configurations are configured in toolchain files under the cmake folder.
- e.g. for gcc: we enable address, leak, pointer-compare, pointer-subtract, undefined sanitizers.

Comes with the template: codespace support

- VSCode on the web
- Underlying dev-container tailored for beman project development
- *Will* be able to allow easy development on customized compilers (e.g. when a library feature is dependent on a language feature).
- Workflow to hack around with a library should be as easy as:
 - Read README
 - Open in codespace
 - Build project
 - Start hacking.

Everything needs to
be automatically
tested.

Main Unit Testing:

- Configurable Main Unit Testing matrix:
 - Testing Across platforms: windows, macos, linux
 - Testing Across compilers: gcc, clang, apple-clang, msvc
 - Testing Across versions: 17 to 23
 - Testing Across cmake arguments: default, with thread sanitizer, with address sanitizer.
 - We have testing environment support for reflection capable compilers.

Declarative testing config

```
matrix_config: >
{
  "gcc": [
    { "versions": ["15"],
      "tests": [
        { "cxxversions": ["c++26"],
          "tests": [
            { "stdlibs": ["libstdc++"],
              "tests": [
                "Debug.Default", "Release.Default", "Release.TSan",
                "Release.MaxSan", "Debug.Werror", "Debug.Dynamic",
                "Debug.Coverage"
              ]
            }
          ]
        }
      ]
    },
    { "cxxversions": ["c++23", "c++20", "c++17"],
      "tests": [{ "stdlibs": ["libstdc++"], "tests": ["Release.Default"]} ]
    }
  ],
  { "versions": ["14", "13"],
    "tests": [
      { "cxxversions": ["c++26", "c++23", "c++20", "c++17"],
        "tests": [{ "stdlibs": ["libstdc++"], "tests": ["Release.Default"]} ]
      }
    ]
  },
}
```

Comes with the template: Code quality enforcement

- pre-commit is a widely used pre-commit hook manager
- It manages a configured set of tools and invokes it pre-commit
- Can be manually triggered by running pre-commit

```
riverwu@River-Wu's-Mac exemplar % pre-commit
trim trailing whitespace.....Passed
fix end of files.....Passed
check yaml.....(no files to check)Skipped
check for added large files.....Passed
clang-format.....Failed
- hook id: clang-format
- files were modified by this hook
CMake linting.....(no files to check)Skipped
codespell.....Passed
riverwu@River-Wu's-Mac exemplar %
```



Comes with the template: Linting

- repo: <https://github.com/pre-commit/mirrors-clang-format>
rev: v18.1.8
hooks:
 - id: clang-format
types_or: [c++, c]
- repo: <https://github.com/BlankSpruce/gersemi>
rev: 0.15.1
hooks:
 - id: gersemi
name: CMake linting

PSA: Double check if your lint tool is maintained



Looking for maintainers

#27 · tkruse opened on Feb 19, 2024

9222cdf · 4 years ago



12 Commits

Linters's secret weapon

- Review dog!





github-actions bot reviewed on Nov 14, 2024

[View reviewed changes](#)

examples/identity_direct_usage.cpp

Comment on lines +10 to +11

```
10 + std::cout <<
11 +     exe::identity()(2024) << '\n';
```



github-actions bot on Nov 14, 2024

Contributor ...

[pre-commit] reported by [reviewdog](#) 🐕

Suggested change

```
10 - std::cout <<
11 -     exe::identity()(2024) << '\n';
10 + std::cout << exe::identity()(2024) << '\n';
```

[Commit suggestion](#) ▼



1



Reply...

[Resolve conversation](#)

Let's build a better feature for C++

- Beman Project meets up every Monday at 12:00 PM EST
- Help us grow by contributing implementation, help out with infrastructure work, and/or provide expertise.
- Our website:
- Chat and hangout with river
- If you're interested in joining tenstorrent, or want to know more about us, please reach out to me on linkedin.

