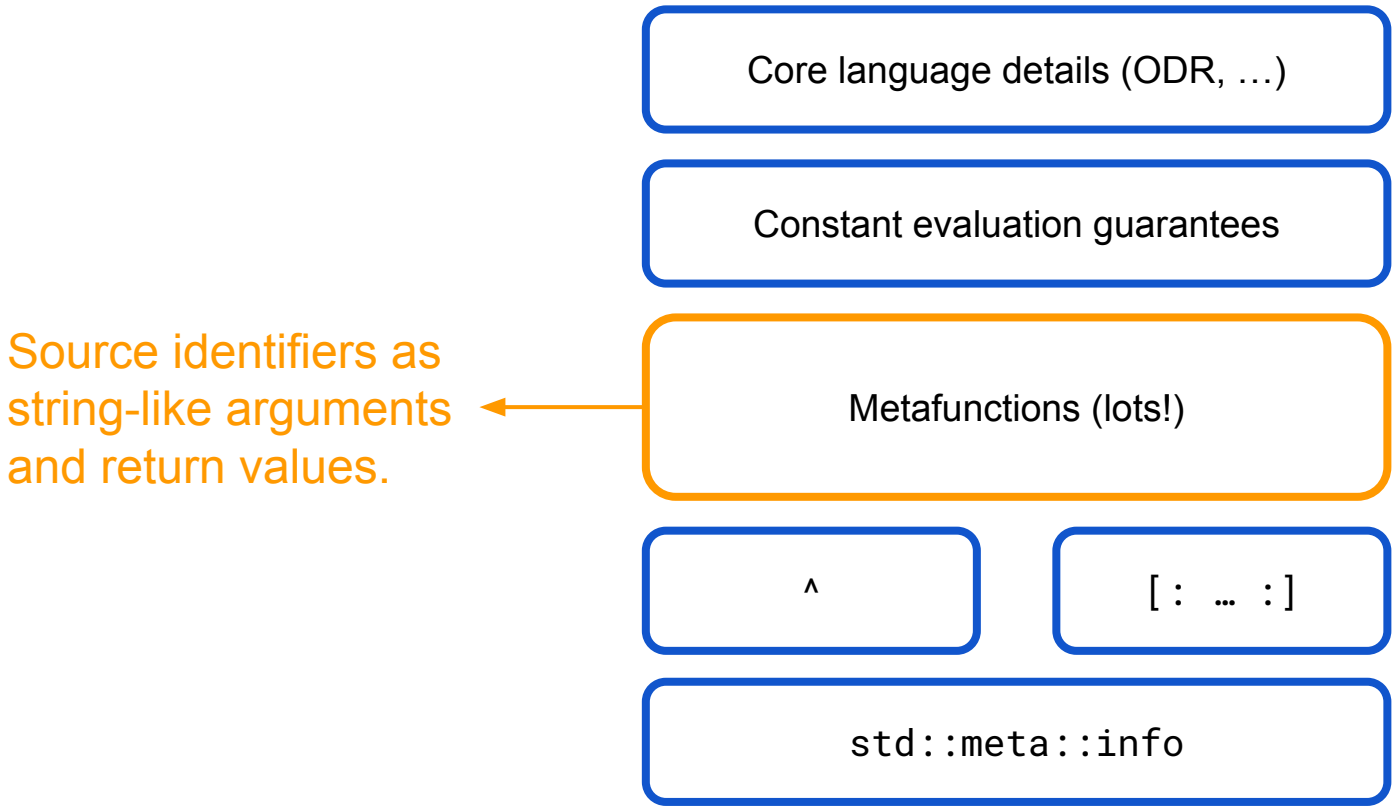
Reflection for C++26

SG16 Telecon — April 24, 2024

```cpp
#include <experimental/meta>
#include <iostream>

enum MyEnum { x, y, e = -1, f, z = 99 };
struct S {
  int key:24;
  int flags:8;
  double value;
} s = { 100, 0x0d, 42.0 };

int main() {
  constexpr std::meta::info  refl = ^MyEnum;
  std::cout << name_of(refl) << '\n';
  std::cout << name_of(enumerators_of(refl)[1]) << '\n';
  std::cout << [:enumerators_of(refl)[4]:] << '\n';
  std::cout << name_of(nonstatic_data_members_of(^S)[1]) << '\n';
  std::cout << s.[:nonstatic_data_members_of(^S)[1]:] << '\n';
}
```


Core language details (ODR, …)
Constant evaluation guarantees
Source identifiers as string-like arguments and return values.
Metafunctions (lots!)
^
[: … :]
std::meta::info

API elements that produce text

```
namespace std::meta {
    consteval str_type name_of(info);
    consteval str_type qualified_name_of(info);
    consteval str_type display_name_of(info);
}
```

API elements that consume text (as of P2996R2)

```
namespace std::meta {
  struct data_member_options_t {
    optional<string_view> name;
    bool is_static = false;
    optional<int> alignment;
    optional<int> width;
  };
  consteval info data_member_spec(info type,
                                  data_member_options_t options = {});
  consteval info define_class(info class_type, span<info const>);
}
```

Constraints

- Round-tripping must work
- Output to `std::cout` must work reasonably
- Some text may not be source-like text (std::meta::display_name_of)
  - Currently only expected to be “output text” with no need for round-tripping

Tension

- After phase 1 of translation, source is effectively Unicode
  - That’s good
- There are solid Unicode types in standard C++
  - UTF-8 types in particular
- Support in the standard library is inadequate
  - In particular, no `std::cout << u8”Hello, World\n”;`

Proposal sketch #1 (Daveed's preferred)

- Traffic in both `std::string_view` and `std::u8string_view` (and maybe `std::string` and `std::u8string`)
- Require round-tripping
  - Either using an implementation-defined identifier-exchange representation, or by standardizing such a representation (e.g., using UCNs)

```
namespace std::meta {
  template<char_or_char8_t CharT = char> consteval
    std::basic_string_view<CharT> name_of(info);
}
```

Proposal sketch #2

- Traffic only in `std::u8string_view` (and maybe `std::u8string`)
- (Naturally round-trips)
- Ensure UTF-8 output works for `std::cout` and `std::format` in C++26
  - Via a separate paper
  - Danger of standardization race condition

Proposal sketch #3 (Victor’s idea)

- Introduce a new type (`std::meta::identifier`?) that encapsulates the internal encoding
  - Could have a specific formatter in the `<format>` library
- Can be consumed directly by the `define_class` interface