Dr Ivan Čukić KDAB

THE EXPECTED OUTCOME

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Meeting C++ 2022

Ivan Čukić KDAB

INTRODUCTION

VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Meeting C++ 2022

Ivan Čukić KDAB

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

ABOUT ME

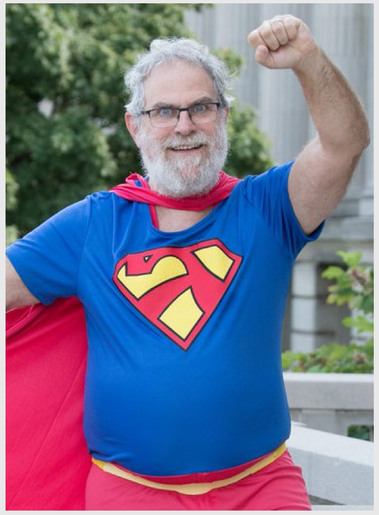
- KDAB

  I assure you we're open^Whiring!
- Author of "Functional Programming in C++"

  available in English, Chinese, Korean, Russian, Polish
- Trainer / Consultant
- KDE developer
- University professor

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

DISCLAIMER



Make your code readable. Pretend the next person who looks at your code is a psychopath and they know where you live. – Philip Wadler

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

DISCLAIMER

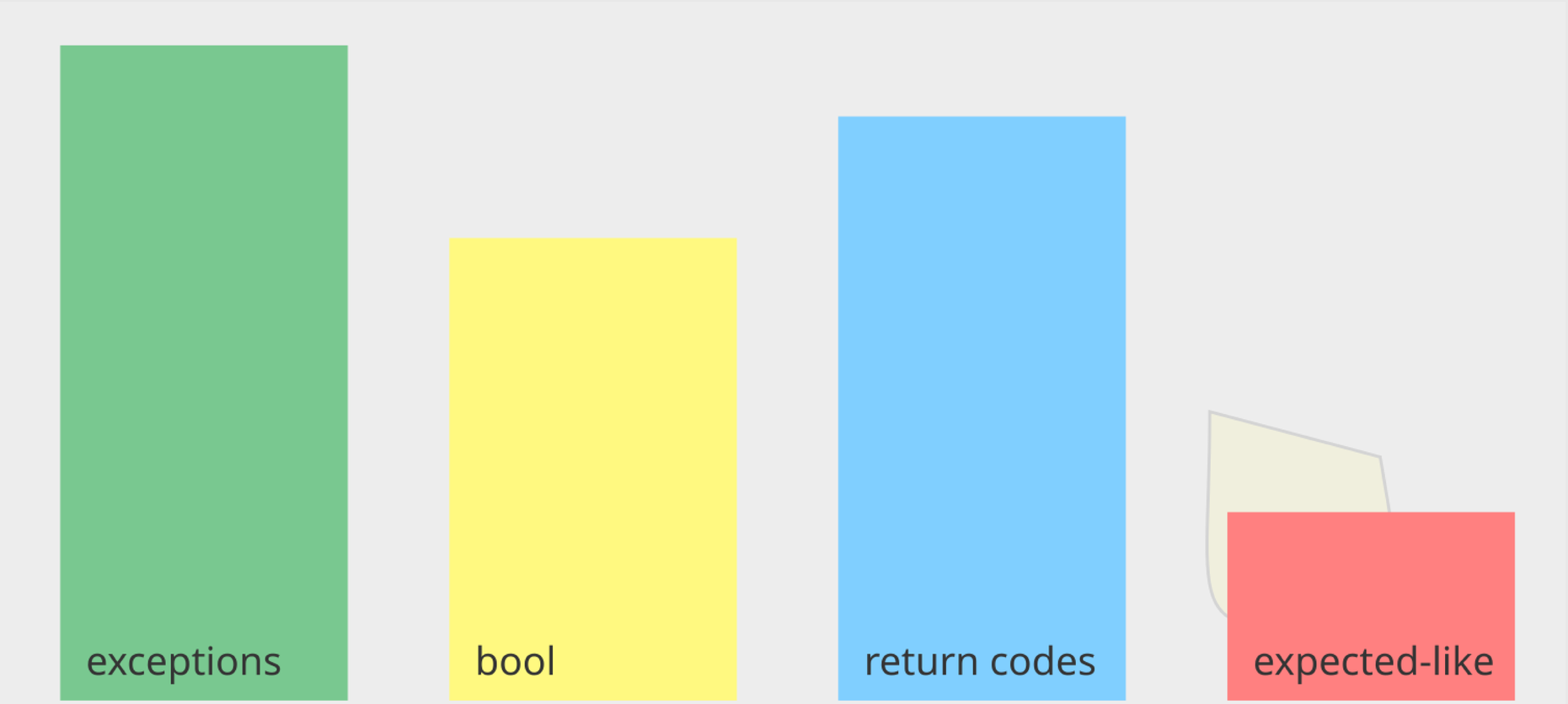
The code snippets are optimized for presentation,
it is not production-ready code.

std namespace is omitted, value arguments used instead

of references to const or forwarding references, etc.

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

WHY AGAIN?



INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

EXCEPTIONS

```
email_t parse_email(string email) {
    if (!email_regex.matches(email)) {
        throw email_parsing_error
            {"not a valid e-mail address"};
    }

    :::
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

EXCEPTIONS

- Forbidden in many projects
- Slow sad path
- Increase the binary size
- Execution paths are not obvious
- Easy to ignore

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

EXCEPTIONS

- Not really a part of the API
- noexcept as the final nail

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

OUT PARAMETERS

```
bool parse_email(string email, email_t& result) {
    if (!email_regex.matches(email)) {
        return false;
    }

    :::
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

OUT PARAMETERS

- Obvious, but awful API design
- Easy to ignore (nodiscard to the rescue)

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

SPECIAL RETURN VALUES

```
email_t parse_email(string email) {
    if (!email_regex.matches(email)) {
        return email_t
            { "invalid char in username", "ERROR" };
    }

    :::
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

SPECIAL RETURN VALUES

- Which values are special?
- Easy to ignore, nodiscard doesn’t help

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

INTRODUCTION

VALUE OR ERROR

BASIC IDIOMS

CONVERSIONS

BEYOND

Meeting C++ 2022

Ivan Čukić KDAB

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

VALUE OR ERROR TYPES

- return a valid value or throw an exception
- return a valid value or a special one denoting the error
- return a valid value via out parameter or a failure

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

VALUE OR ERROR TYPES

```
variant<
    Value,
    Error>
```

A long time ago in a galaxy far away ...

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

SIMPLE WRAPPERS

```
template<typename T, typename ValidationPolicy>
class Result {
private:
    T _value;

public:
    bool has_value() const;
    T value() const;
    T error() const;
};
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

SIMPLE WRAPPERS

Policies:

- value ≥ 0, errors negative
- value = 0, errors positive
- value ≠ 0, error zero
- value → true, error otherwise

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

SIMPLE WRAPPERS

```
if (result.has_value()) {
    process(result.value());
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

SIMPLE WRAPPERS

```
if (result) {
    process(*result);
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

VALUE OR ERROR TYPES

```
variant<
    Value,
    Error> result = :::;

if (result) {
    process(*result);
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

- P0323: Vicente Botet, JF Bastien, Jonathan Wakely
- P2505: Jeff Garland

It’s a bug, not a feature.

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

STD::EXPECTED

```
expected<email_t, err_t> parse_email(string email) {
    if (!email_regex.matches(email)) {
        return unexpected
            ("not a valid e-mail address");
    }

    :::
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

STD::EXPECTED

```
// -> expected<string, err_t>

expected<string, err_t> input = get_input();
if (!input) return input;

auto email = parse_email(*input);
if (!email) return email;

auto html = format_link(email->user, *email);

return html;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

STD::EXPECTED

```
// -> expected<string, err_t>

expected<string, err_t> input = get_input();
if (!input) return unexpected(input.error());

auto email = parse_email(*input);
if (!email) return unexpected(email.error());

auto html = format_link(email->user, *email);

return html;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

STD::EXPECTED

- Predictable execution path
- Unlimited number of errors in the air
- No magic
- Storage – files, databases…
- Transfer over thread and process boundaries, network…

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

TWO NOTATIONS

As we don’t have UCS:

```
voe.some_function();

some_function(voe);

some_other_function(
    some_function(voe), arg1, arg2);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

TWO NOTATIONS

As we don’t have UCS:

```
voe | lf::some_function()
    | lf::some_other_function(arg1,
                              arg2);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

TWO NOTATIONS

As we don’t have UCS:

```
voe | some_function()
    | some_other_function(arg1,
                          arg2);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

TWO NOTATIONS

```
struct some_function {};

auto operator| (auto object, some_function)
{
    return object.some_function();
    // or: some_function(object);
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

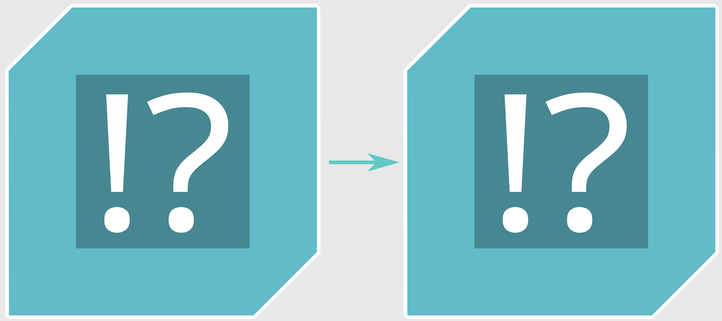
INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES

```
// email: expected<email_t, err_t>

if (!email) return unexpected(email.error());
auto html = format_link(*email);

return html;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES



INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES

```
// email: expected<email_t, err_t>

auto html = email | transform(format_link);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES

```
// email: expected<email_t, err_t>

auto html = email | transform(
    [] (email_t email) {
        return format_link(email.user,
                           email);
    });
```

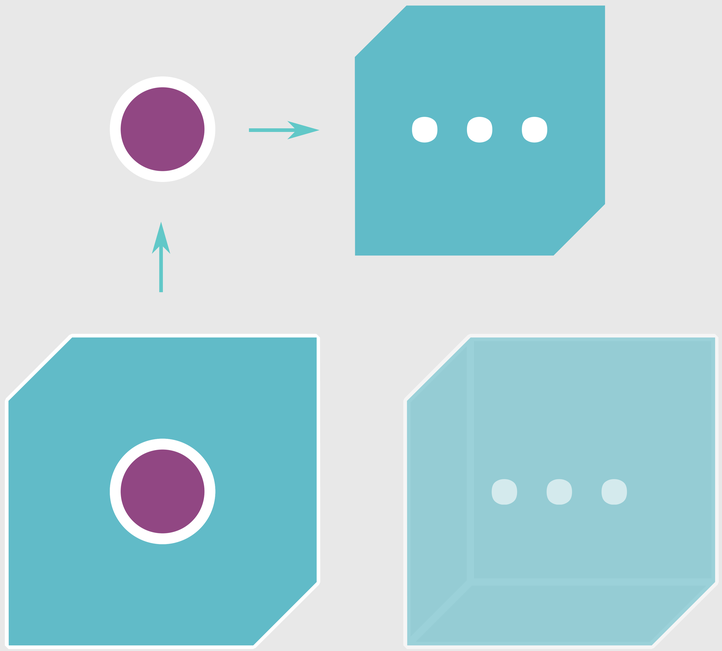
INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

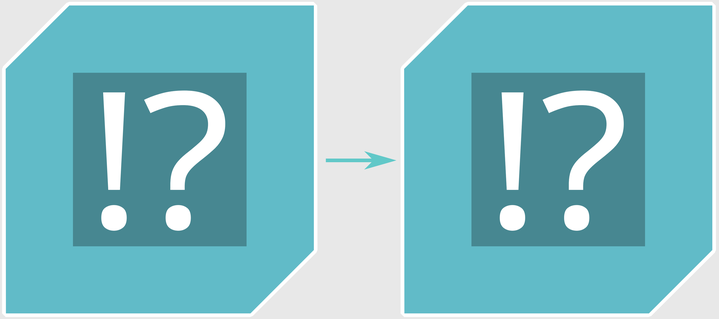
CHANGING VALUES

```
// input: expected<string, err_t>

if (!input) return unexpected(input.error());
auto email = parse_email(*input);

:::
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES





INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES

```
// -> expected<string, err_t>

auto email = input | and_then(parse_email);

:::
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CHANGING VALUES

```
return get_input()
       | and_then(parse_email)
       | transform(
           [] (email_t email) {
               return format_link(email.user,
                                  email);
       });
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FALLBACK

```
return get_input()
       | and_then(parse_email)
       | transform(:::)
       | value_or(string{});
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FALLBACK

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FALLBACK

```
return get_input()
       | and_then(parse_email)
       | transform(:::)
       | or_else([] (err_t err) -> ::: {
           if (err == err_t::empty)
               return {};
           else
               return err;
       });
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Meeting C++ 2022

Ivan Čukić KDAB

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM SPECIAL VALUES

- Just use the simple wrappers
- Write conversion functions between wrappers and expected

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM EXCEPTIONS

```
class err_t {
    err_t(exception_ptr ep);
};
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM EXCEPTIONS

```
template <typename Err, typename Func,
          typename Result = invoke_result_t<Func>>
expected<Result, Err> ex_try(Func function)
    try { return function(); }
    catch (...) {
        return unexpected(
            Err(current_exception()));
    }
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

TO EXCEPTIONS

```
result.value();

// unlike *value;
```

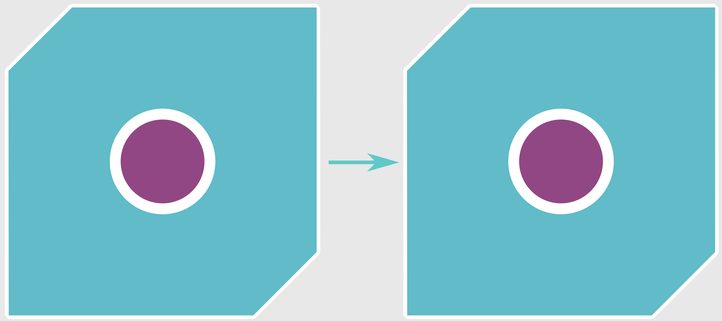
INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

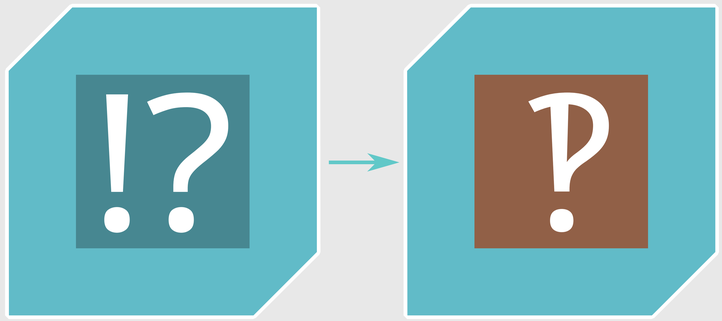
FROM OUT PARAMETERS

```
// int parse_email(string email, email_t& result)
template <typename Value, typename Err, typename Func>
expected<Value, Err> ex_from_out(Func function) {
    Value result;
    if (auto error = function(result)) {
        return unexpected(error);
    }
    return result;
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM OTHER EXPECTEDS





INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM OTHER EXPECTEDS

```
// get_email_link() -> expected<string, err_t>

// to_error_code(err_t error) -> error_code

get_email_link() | transform_error(to_error_code);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM OTHER EXPECTEDS

Or rely on conversion:

```
expected<Value, OldErrorType> ->
        expected<Value, NewErrorType>
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM OTHER EXPECTEDS

```
struct my_error_t {
    my_error_t(err_t e);
};

// get_email_link() -> expected<string, err_t>
expected<string, my_error_t> link =
    get_email_link();
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

FROM OTHER EXPECTEDS

```
using my_error_t =
    variant<err_t, some_other_err_t, ...>;


// get_email_link() -> expected<string, err_t>
expected<string, my_error_t> link =
    get_email_link();
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Meeting C++ 2022

Ivan Čukić KDAB

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

What about functions with multiple arguments?

```
value –[transform]–> result
value –[and_then]–> result
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
// email: expected<email_t, err_t>

auto html = email | transform(
    [] (email_t email) {
        return format_link(email.user,
                           email);
    });
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
f: (arg_1, arg2, ..., arg_n) -> result

ex_invoke(f, ex_1, ex_2, ..., ex_n)
                  -> expected<result, ???>;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

DEDUCING ERRORS

How to deduce the error type

- Explicitly specify error type
- Use the first error type found
- Use variant of all found error types

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

DEDUCING ERRORS

```
template <typename... Expecteds
          >
using find_first_expected =
    typename tag_t::list<Expecteds...>
    ::template transform<remove_cvref_t>
    ::template filter<is_expected_instantiation>
    ::first;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

DEDUCING ERRORS

```
template <typename... Expecteds
          >
using errors_variant =
    typename tag_t::list<Expecteds...>
    ::template transform<remove_cvref_t>
    ::template filter<is_expected_instantiation>
    ::uniq
    ::template pass_to<variant>;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

DEDUCING ERRORS

```
template <template <typename...> Custom,
          typename... Expecteds>
using errors_as_custom_type =
    typename tag_t::list<Expecteds...>
    ::template transform<remove_cvref_t>
    ::template filter<is_expected_instantiation>
    ::uniq
    ::template pass_to<Custom>;
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

CALLING FUNCTIONS

```
template<typename Function, typename... Expecteds>
auto ex_invoke(Function function, Expecteds ...expecteds) -> ...
{
    auto error = find_error(expecteds...);
    if (error) return unexpected(*error);

    auto value_for = []<typename Value>(Value value) {
        return value.value();
    };
    return current_expected(
        function(value_for(expecteds)...));
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
ex_invoke(format_link,
          email | transform(&email_t::user),
          email);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
auto value_for = []<typename Value>(Value value) {
    if constexpr (is_expected_instantiation_v<Value>) {
        return value.value();
    } else {
        return value;
    }
};
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
ex_invoke(format_link,
          email | transform(&email_t::user),
          email,
          "email-link"s);
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
template<typename Function, typename... Expecteds>
auto ex_coalesce(Function function, Expecteds ...expecteds) -> ...
{
    auto value = find_valid(expecteds...);
    if (!value) return unexpected(value.error());

    return current_expected(
        function(*value));
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

- ex_invoke – calling n-ary functions
- ex_coalesce – calling unary functions with first valid value

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

- ex_invoke – calling n-ary functions
- ex_coalesce – calling unary functions with first valid value
- ex_lazy_invoke – calling n-ary functions with early exit
- ex_lazy_coalesce – calling unary functions with first valid result

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
auto value_for = []<typename Value>(this auto self, Value value) {
    if constexpr (is_invocable_v<Value>) {
        return self(value());
    } if constexpr (is_expected_instantiation_v<Value>) {
        return value.value();
    } else {
        return value;
    }
};
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

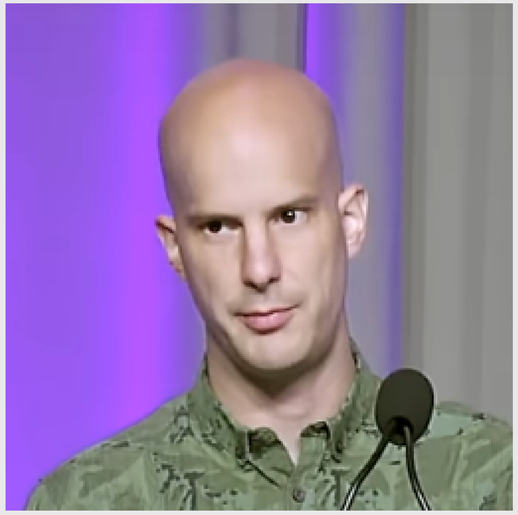
```
auto value_for = []<typename Value>(this auto self, Value value) {
    if constexpr (is_lazy_instantiation_v<Value>) {
        return self(value());
    } if constexpr (is_expected_instantiation_v<Value>) {
        return value.value();
    } else {
        return value;
    }
};
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

CALLING FUNCTIONS

```
// cool is not good, precise is better

ex_invoke(format_link,
          email | transform(&email_t::user),
          email,
          lazy(:::));
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

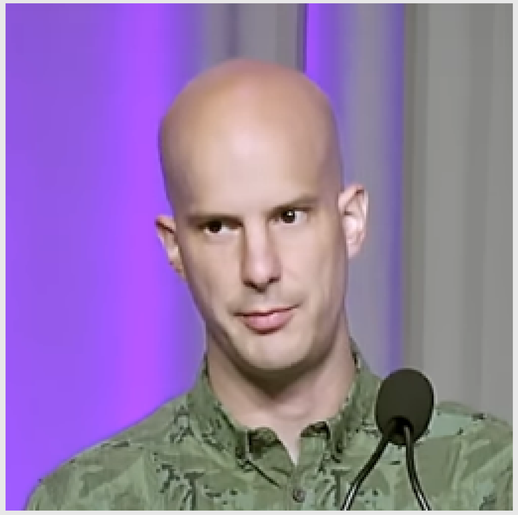


Andreas Weis

Coroutine Basics - Asynchronous computation

```
auto [ec, bytes_read] = read(socket, buffer);
// ...

async_read(socket, buffer,
  [](std::error_code ec, std::size_t bytes_read) {
    // ...
  });
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND



Andreas Weis

Coroutine Basics - Asynchronous computation

```
auto [                    ] = read(                    );
// ...

async_read(                    ,
  [](                                                  ) {
    // ...
  });
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

EXPECTED & CO.

```
task<string> current_user_name() {
    task<user_t> user_task = current_user();
    user_t = co_await user_task;

    task<string> name_task = user.name();
    string name = co_await name_task;

    co_return name;
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

EXPECTED & CO.

```
task<string> current_user_name() {
    return current_user()
           | and_then([] (user_t user) {
                 return user.name();
             })
           | and_then([] (string name) {
                 return mke(name);
             });
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

EXPECTED & CO.

```
task<string> current_user_name() {
    return current_user()
           | and_then(&user_t::name);
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

EXPECTED & CO.

```
expected<string, err_t> get_email_link() {
    string input = co_await get_input();
    auto email = co_await parse_email(*input);
    auto html = format_link(email->user, *email);

    co_return html;
}
```

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND

Ivan Čukić KDAB

- Outcome…
- Value x Error

SUMMARY

- Errors in values
- Outside of domain types
- Interoperability with other mechanisms
- Easy conversion
- Easy composition
- Can simulate exception-like syntax when needed

INTRODUCTION
VALUE OR ERROR
BASIC IDIOMS
CONVERSIONS
BEYOND