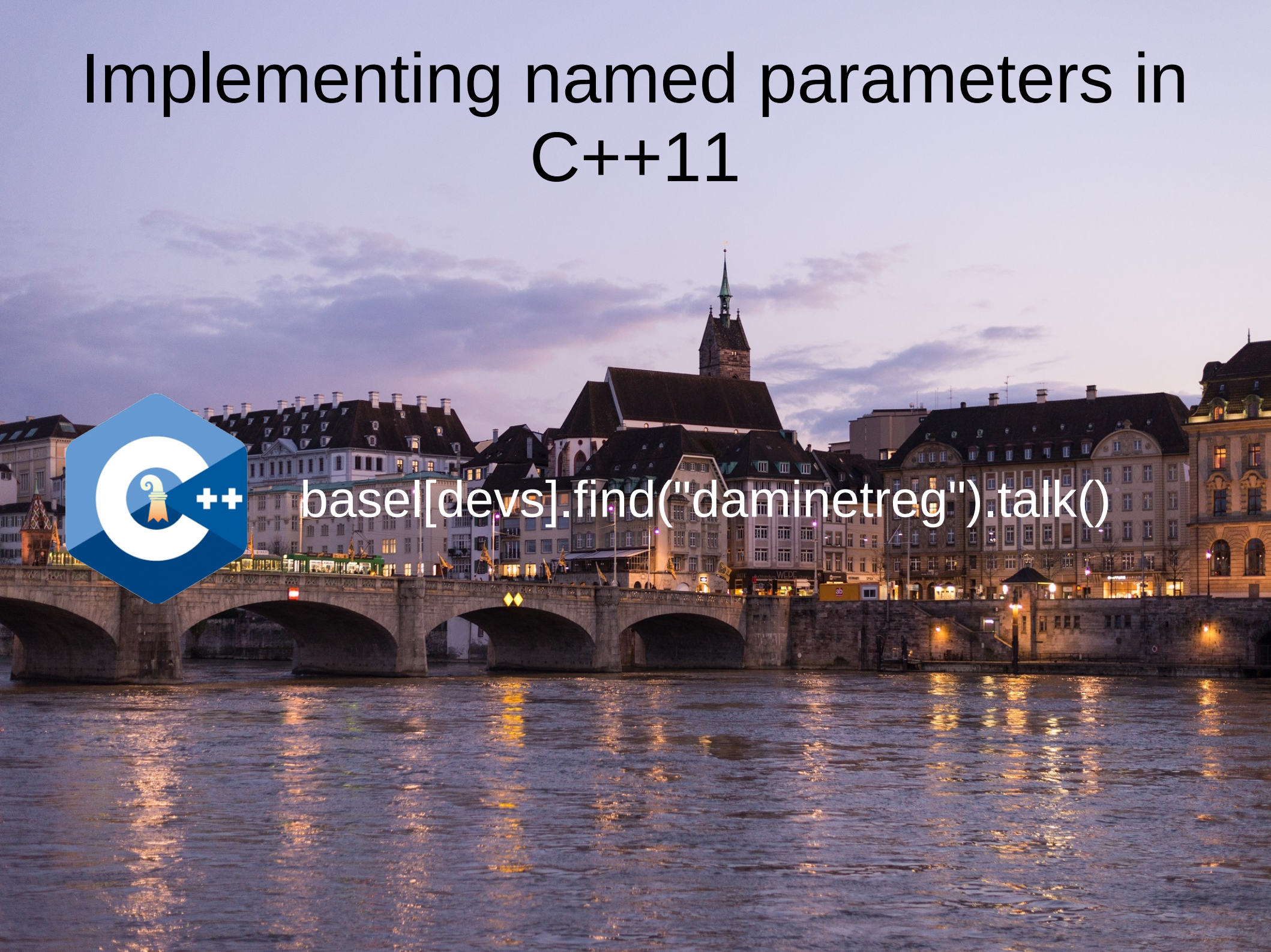
Implementing named parameters in C++11



basel[devs].find("daminetreg").talk()

Unambiguous

```
using namespace xxhr;
GET( Url{"https://basel-devs.github.io/"},
  on_response = [](auto&& resp) {
    //… do something
  }
);
```

Two kinds of named parameters

```
using namespace xxhr;
GET( Url{"https://basel-devs.github.io/"},
  on_response = [](auto&& resp) {
    //… do something
  }
);
```

How can the first be detected ?

```
struct Url { std::string url; };

namespace priv {

  template <typename T>
  void set_option(Session& session, T&& t) { session.SetOption(XXHR_FWD(t)); }

  template <typename T, typename... Ts>
  void set_option(Session& session, T&& t, Ts&&... ts) {
    set_option(session, XXHR_FWD(t));
    set_option(session, XXHR_FWD(ts)...);
  }

} // namespace priv

template <typename... Ts>
void GET(Ts&&... ts) { Session session; priv::set_option(session, XXHR_FWD(ts)...); session.GET(); }
```

So now we know

- Passing parameter out-of-order to the function
- Selectively reacting based on their “name”.

Could we get the {} away ?

```
template<template <class> class Template>
struct make_handler_t {
    template<typename Handler>
    constexpr Template<Handler> operator= (Handler handler) const {
      return Template<Handler>(handler);
    }
};
```

Could we get the {} away ?

```
template <class Handler>
struct on_response_ {
  explicit on_response_(Handler handler) : handler_(handler) {}
  Handler handler_;
};

constexpr make_handler_t<on_response_> on_response;
```

Unambiguous

```
using namespace xxhr;
GET( on_response = [](auto&& resp) {
    //… do something
  },
  Url{"https://basel-devs.github.io/"}
);
```