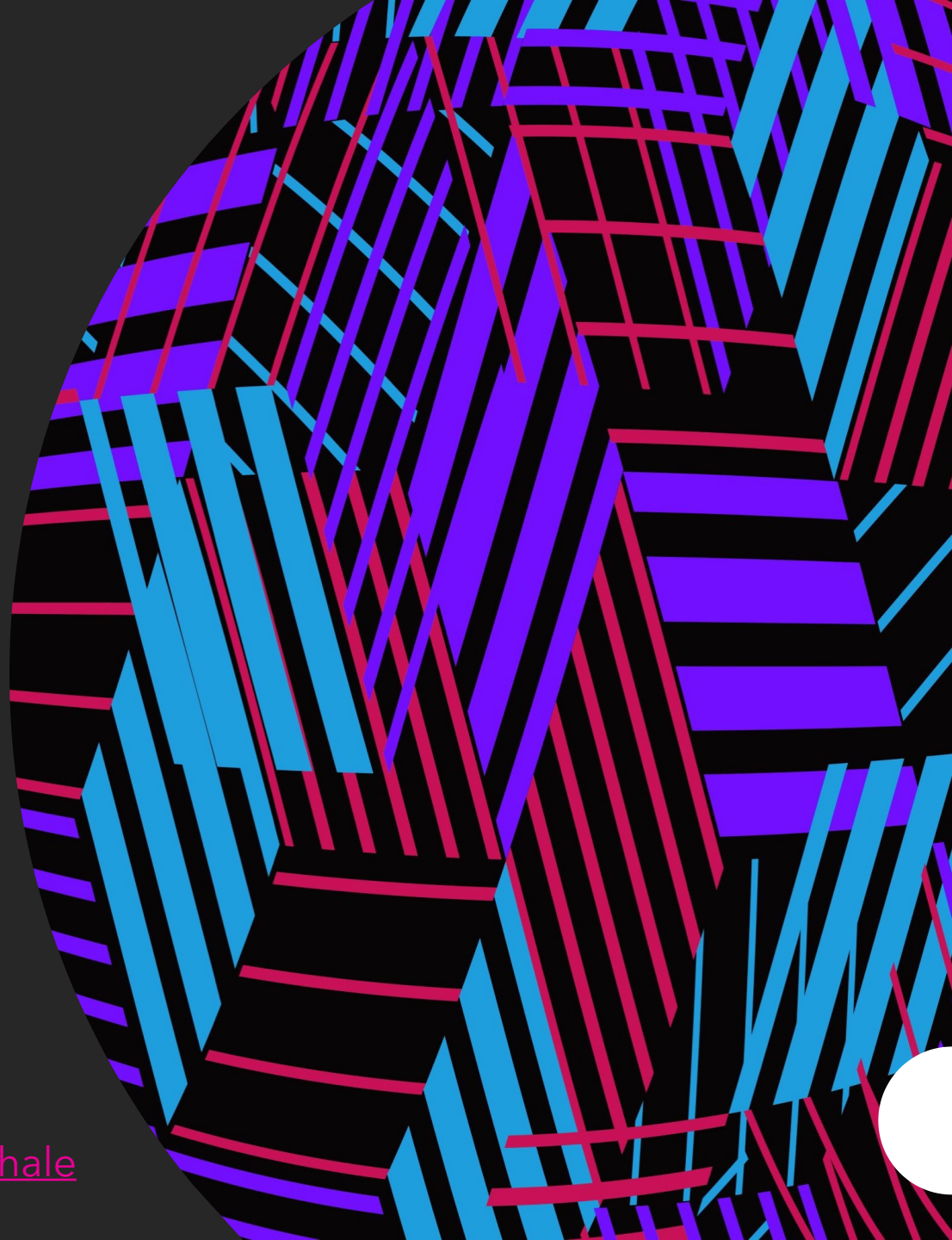
Contemporary C++ Web Scraping

It's not as low level as one might thing



Darrell Wright
@beached_whale

What is needed

- Retrieve documents
- Parse documents
- Query documents

Retrieving Documents

- Surprise, it's Curl.
- Using a simple Curl wrapper
- https://github.com/beached/curl_wrapper

```
int main( ) {
  auto crl = daw::curl_wrapper( );
  crl.retrieve( url: "https://www.google.ca" );
  std::cout << crl.get_body( ) << '\n';
}
```

Parse Documents

- Wrapping Gumbo to get document tree
- https://github.com/beached/gumbo_pp
- https://github.com/google/gumbo-parser
- We have an iterator interface into the document tree that uses DFS ordering

```
int main( ) {
  constexpr std::string_view html =
    R"html(
<html>
  <head>
    <title>Test</title>
  </head>
  <body><div class='hello'><b>Hey folks!</b></div> <a href="https://www.google.com">Google</a></body>
</html>)html";

  auto doc_range = daw::gumbo::gumbo_range( html_document: html );
```

Query Documents

- Gumbo_pp provides a set of combinable predicates based on

attribute	class type	id	inner text
outer text	content text	tag	

  - Each predicate type has associated verbs like attribute::is
  - All have the where clause to allow for using custom matcher predicates
  - They can be combined with and(match_all), or(match_any), not(negate match), xor(match_one) to form complex expressions
  - Usable with std::algorithms, e.g. std::find_if, daw::algorithm::for_each_if
  - Does not allocate unless asked to

Show me the Code

- Enumerate all div tags

```
void enumerate_all_div_tags( daw::gumbo::gumbo_range &doc_range,
                             daw::string_view html_doc ) {
  daw::algorithm::for_each_if(
    first: doc_range.begin( ),
    last: doc_range.end( ),
    pred: match::tag::DIV,
    onEach: [&]( GumboNode const &node ) -> void {
      std::cout << daw::gumbo::node_inner_text( node, html_doc ) << '\n';
    } );
}
```

Show me the Code

- Find all links that contain a keyword

```
template<std::size_t N>
void find_all_links_with_keyword( daw::gumbo::gumbo_range &doc_range,
                                  daw::string_view ( &keyword )[N] ) {
  daw::algorithm::for_each_if(
    first: doc_range.begin( ),
    last: doc_range.end( ),
    pred: match::tag::A and match::attribute::value::starts_with( attribute_name: "href", value_prefix: "http" ) and
     match::content_text::contains( keyword ),
    onEach: [](  GumboNode const &node ) {
      std::cout << "[" << daw::gumbo::node_content_text( node ) << ']';
      std::cout << "(" << daw::gumbo::node_attribute_value( node, attribute: "href" )
                << ")\n";
    } );
}
```

Show me the Code

- Find all Paragraph's with matching matching

```
void find_all_p_tags_with_id( daw::gumbo::gumbo_range &doc_range,
                              daw::string_view id ) {
  daw::algorithm::for_each_if( first: doc_range.begin( ),
                               last: doc_range.end( ),
                               pred: match::tag::P and match::id::is( id_name: id ),
                               onEach: []( GumboNode const &node ) -> void {
                                 std::cout
                                   << daw::gumbo::node_content_text( node )
                                   << '\n';
                               } );
}
```

Examples

- Code from slides and slides

  https://github.com/beached/denver_cug_web_scraping
- Full example web service

  https://github.com/beached/climate_change_api_example

What's Next

- Add a full Node type with accessors/matcher methods
- Have the Gumbo tag enumerations inside daw::gumbo::tags namespace
- Use/write a library like Puppeteer/Selenium that uses headless Chrome/Firefox

Questions