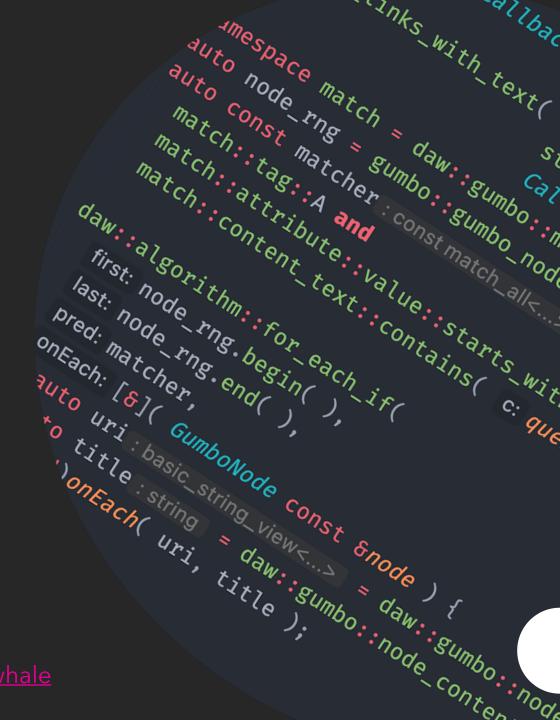
# Contemporary C++ Web Scraping

It's not as low level as one might thing



Darrell Wright
<a href="mailto:University"> @beached whale</a>

## 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';
}</pre>
```

#### Parse Documents

- Wrapping Gumbo to get document tree
- https://github.com/beached/gumbo\_pp
- <a href="https://github.com/google/gumbo-parser">https://github.com/google/gumbo-parser</a>
- 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) operators 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

#### Show me the Code

Find all links that contain a keyword

#### 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 )</pre>
                                     << '\n';
                                 } );
```

# Examples

- Code from slides and slides
   https://github.com/beached/denver\_cug\_web\_scraping
- Full example web service <a href="https://github.com/beached/climate\_change\_api\_example">https://github.com/beached/climate\_change\_api\_example</a>

### What's Next

- Add a full Node type with accessors/matcher methods
- Use/write a library like Puppeteer/Selenium that uses headless Chrome/Firefox

# Questions