Clang-Doc

Where We've Been and Where We're Going

LLVM Dev Meeting 2025

Erick Velez

Agenda

1. What, Why, and How?
2. Recent Improvements and Future Work
3. Getting Involved

What is Clang-Doc?

Clang-Doc

- Lives in clang-tools-extra
- Documentation generator started in 2018
  - Lightning Talk: https://www.youtube.com/watch?v=bTzvPhKN0YI
- Generates HTML, Markdown, YAML documentation

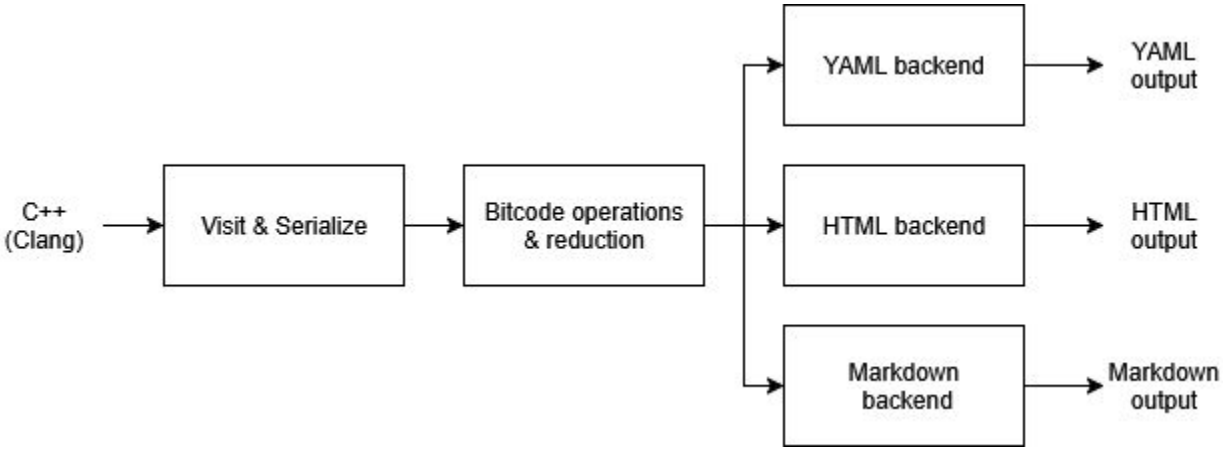
Why another documentation generator?

Other languages supply documentation generators.

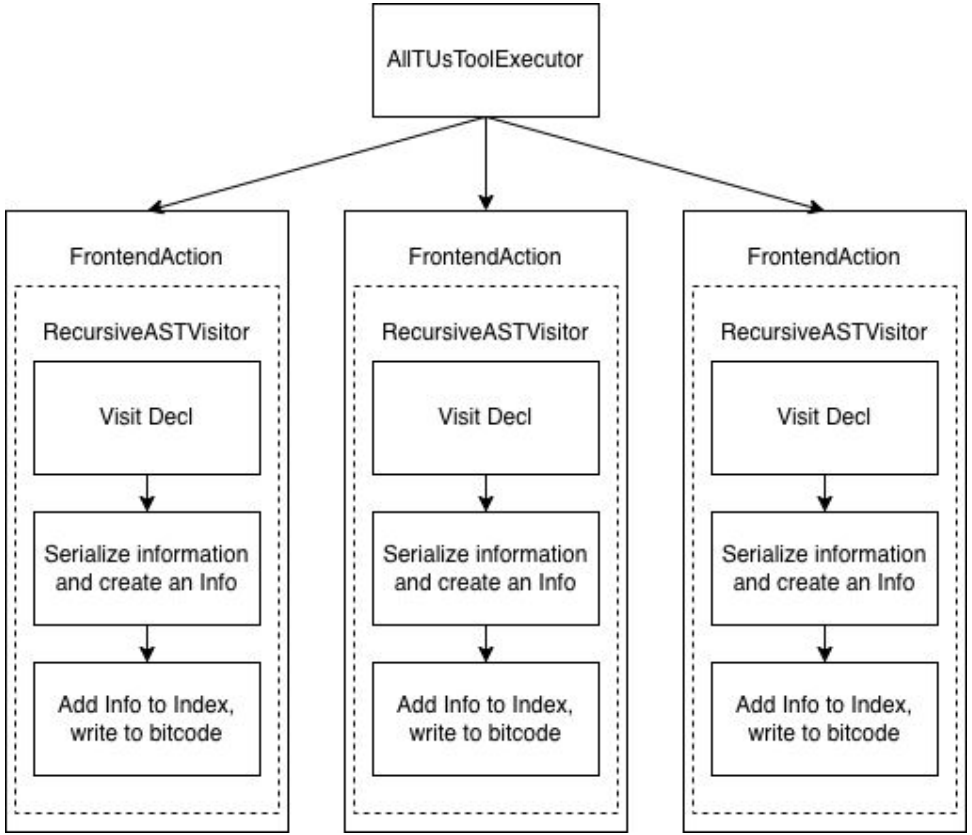
Compiler frontends are the perfect tool to generate documentation.

How Does Clang-Doc Work?

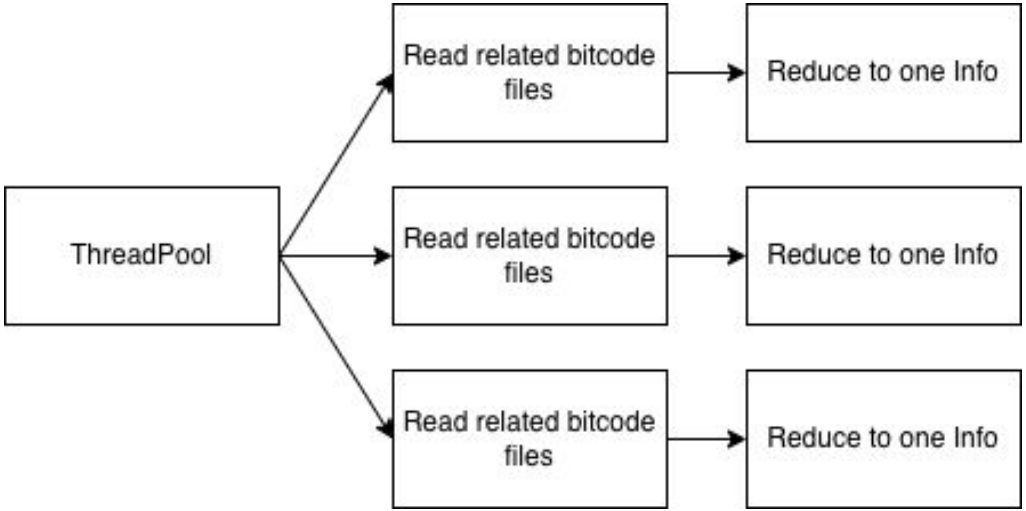
Clang-Doc’s Architecture



LibTooling-based Visitation



Bitcode Operations



Documentation Output

class Composite

Defined at line 16 of file ../../src/devices/tests/multibind-composite-test/drivers/composite.cc

Inherits from DeviceType

Functions

Composite

public void Composite(zx_device_t * parent)

Defined at line 18 of file ../../src/devices/tests/multibind-composite-test/drivers/composite.cc

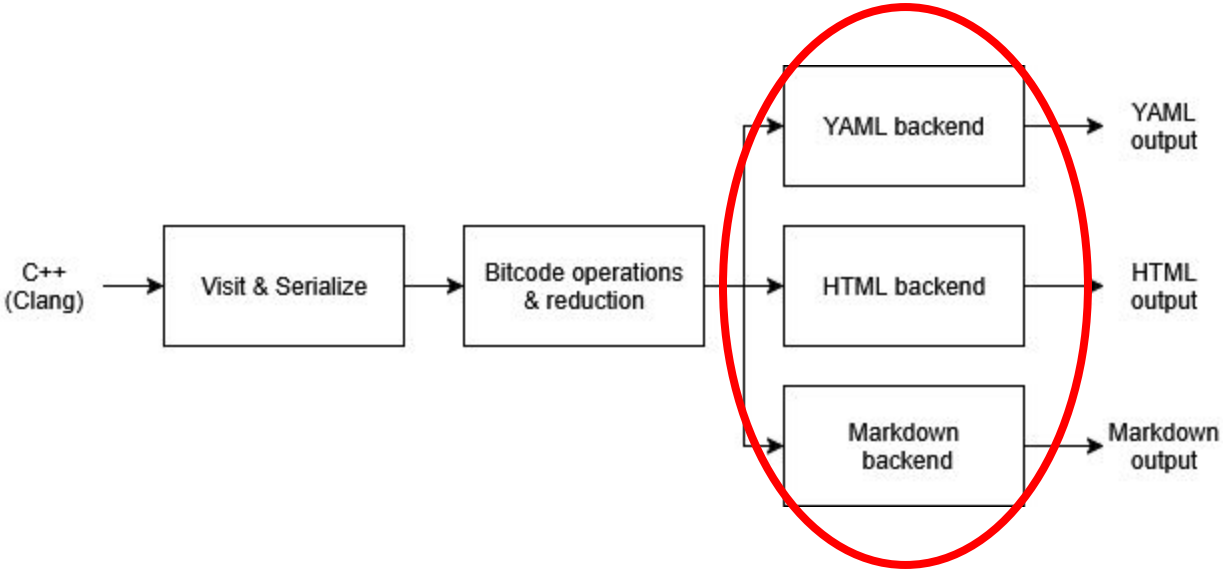
Bind

public static zx_status_t Bind(void * ctx, zx_device_t * device)

Defined at line 20 of file ../../src/devices/tests/multibind-composite-test/drivers/composite.cc

Why not iterate on the output?

Clang-Doc's Weakness



No cohesive framework!

Backend Disparity

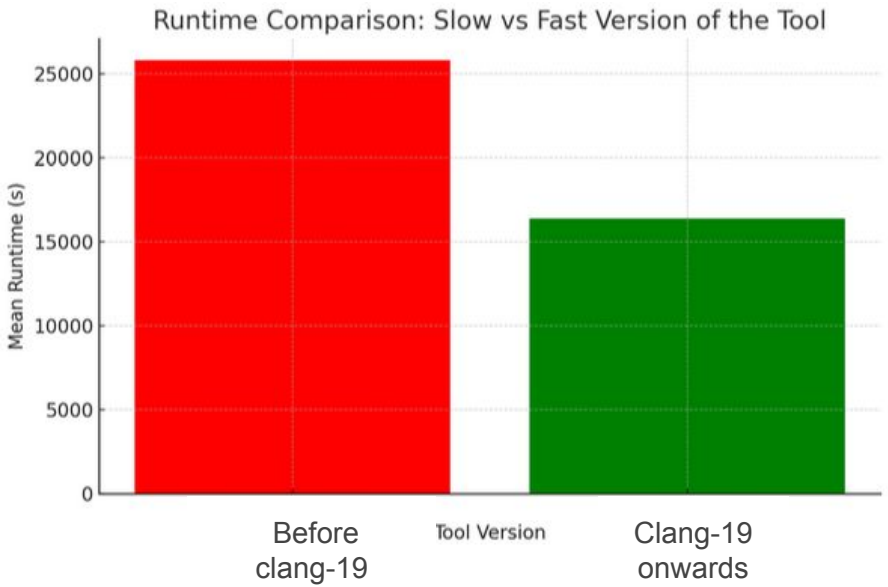
```
// HTMLGenerator.cpp
std::vector<std::unique_ptr<HTMLNode>> Parents = genReferenceList(I.Parents,
I.Path);

// MDGenerator.cpp
std::string Parents = genReferenceList(I.Parents);
```

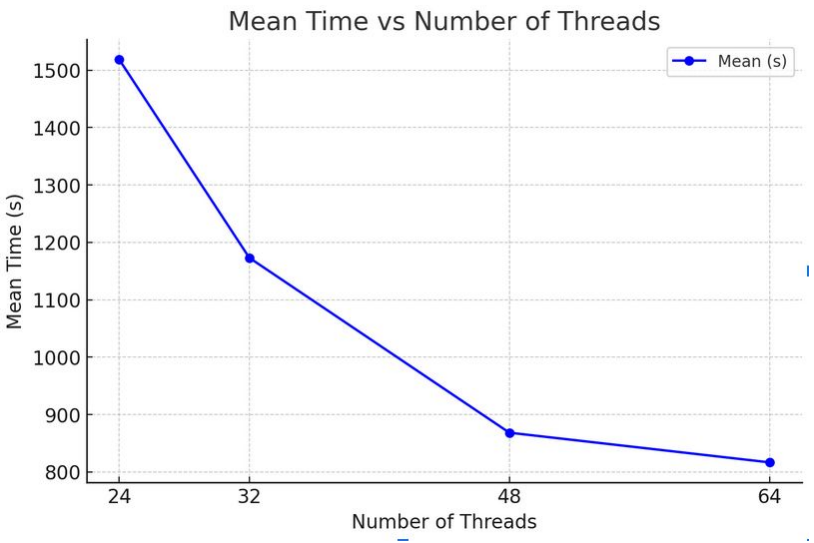
What does this web page look like?

```
void TagNode::render(llvm::raw_ostream &OS, int IndentationLevel) {
  // Children nodes are rendered in the same line if all of them are text nodes
  bool InlineChildren = true;
  for (const auto &C : Children)
    if (C->Type == NodeType::NODE_TAG) {
      InlineChildren = false;
      break;
    }
  OS.indent(IndentationLevel * 2);
  OS << "<" << Tag.toString();
  for (const auto &A : Attributes)
    OS << " " << A.first << "=\"" << A.second << "\"";
  if (Tag.isSelfClosing()) {
    OS << "/>";
    return;
  }
  OS << ">";
  if (!InlineChildren)
    OS << "\n";
  bool NewLineRendered = true;
  for (const auto &C : Children) {
    int ChildrenIndentation =
        InlineChildren || !NewLineRendered ? 0 : IndentationLevel + 1;
    C->render(OS, ChildrenIndentation);
    if (!InlineChildren && (C == Children.back() ||
                            (C->Type != NodeType::NODE_TEXT ||
                             (&C + 1)->get()->Type != NodeType::NODE_TEXT))) {
```

Improvements Over the Last Year

Performance Improvements





Mustache

A simple templating engine. Added to LLVM Support.

```
<div class="sidebar">
    <h2>{{TagType}} {{Name}}</h2>
    <ul>
        {{#HasPublicMembers}}
        <li class="sidebar-section">
            <a class="sidebar-item" href="#PublicMembers">Public Members</a>
        </li>
        <ul>
            {{#PublicMembers}}
            <li class="sidebar-item-container">
                <a class="sidebar-item" href="#{{Name}}">{{Name}}</a>
            </li>
            {{/PublicMembers}}
        </ul>
        {{/HasPublicMembers}}
```

Old HTML vs New HTML

```cpp
void TagNode::render(llvm::raw_ostream &OS, int IndentationLevel) {
  // Children nodes are rendered in the same line if all of them are text nodes
  bool InlineChildren = true;
  for (const auto &C : Children)
    if (C->Type == NodeType::NODE_TAG) {
      InlineChildren = false;
      break;
    }
  OS.indent(IndentationLevel * 2);
  OS << "<" << Tag.toString();
  for (const auto &A : Attributes)
    OS << " " << A.first << "=\"" << A.second << "\"";
  if (Tag.isSelfClosing()) {
    OS << "/>";
    return;
  }
  OS << ">";
  if (!InlineChildren)
    OS << "\n";
  bool NewLineRendered = true;
  for (const auto &C : Children) {
    int ChildrenIndentation =
        InlineChildren || !NewLineRendered ? 0 : IndentationLevel + 1;
    C->render(OS, ChildrenIndentation);
    if (!InlineChildren && (C == Children.back() ||
                            (C->Type != NodeType::NODE_TEXT ||
                             (&C + 1)->get()->Type != NodeType::NODE_TEXT))) {
```

```html
<div class="sidebar">
    <h2>{{TagType}} {{Name}}</h2>
    <ul>
        {{#HasPublicMembers}}
        <li class="sidebar-section">
            <a class="sidebar-item" href="#PublicMembers">Public Members</a>
        </li>
        <ul>
            {{#PublicMembers}}
            <li class="sidebar-item-container">
                <a class="sidebar-item" href="#{{Name}}">{{Name}}</a>
            </li>
            {{/PublicMembers}}
        </ul>
        {{/HasPublicMembers}}
```

New HTML Look

class Calculator

Public Members

public_val

static_val

Public Method

add

subtract

multiply

divide

mod

class Calculator

A simple calculator class.
Provides basic arithmetic operations.

Public Members

```
int public_val
```

```
const int static_val
```

Public Methods

```
int add (int a, int b)
```

Adds two integers.

Parameters

a

First integer.

b

Second integer.

Returns

int The sum of a and b.

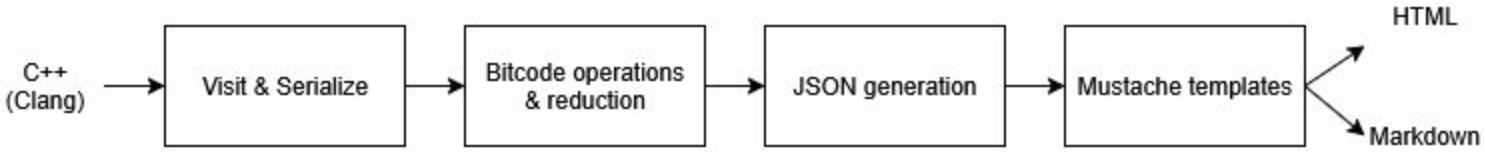
Addressing Backend Disparity

Using JSON as an IR


```
"HasPublicFunctions": true,
"InfoType": "record",
"IsTypedef": false,
"Location": {
  "Filename": "llvm/include/llvm/Transforms/Utils/SimplifyIndVar.h",
  "LineNumber": 36
},
"MangledName": "_ZTVN4llvm9IVVisitorE",
"Name": "IVVisitor",
"Namespace": [
  "llvm"
],
"Path": "llvm",
"ProtectedFunctions": [
  {
    "InfoType": "function",
    "IsStatic": false,
    "Location": {
      "Filename": "llvm/lib/Transforms/Utils/SimplifyIndVar.cpp",
      "LineNumber": 1005
    },
```

JSON improves modularity.

Streamlined Modular Architecture



Other things I did

- Concepts
- Global variables
- Friends
- Comment support revamp
- In-progress Markdown parsing

Future Work

- Cross-referencing
- Improving comment support in the Clang AST
  - Issue: https://github.com/llvm/llvm-project/issues/123582
- Transitioning backends to Mustache templates

Getting Involved

How to Use Clang-Doc

Projects:

```
$ clang-doc --format=json --executor=all-TUs ./build/compile-commands.json
```

Single file:

```
$ clang-doc --format=mustache --executor=standalone ./main.cpp
```

Docs: https://clang.llvm.org/extra/clang-doc.html

Please get involved!

- We need users to tell us what they'd like or what they're missing.
- We have a lot of areas for improvement! We'd love contributions!

Please get involved!

- We need users to tell us what they'd like or what they're missing.
- We have a lot of areas for improvement! We'd love contributions!
- Docs: https://clang.llvm.org/extra/clang-doc.html
- LLVM Blog: https://blog.llvm.org/posts/2025-gsoc-clang-doc/

Discord: erick.velez
E-mail: erick@erickvelez.com
Github: evelez7
Discourse: evelez
erickvelez.com

Thanks!