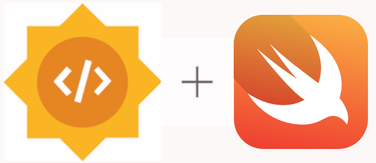


Why Documentation Shouldn’t Feel Like Plain Text

PADMASHREE S S

An **open-source** contributor and a **Google Summer of Code** Mentee at **Swift** organization.

About Me



Padmashree S S

Open Source Contributor | GSoC Mentee with Swift

I am interested in compilers, programming languages, language tooling and low-level systems.



My Open Source Journey:

- Discovered open source an year ago.
- Started contributing to **swift-syntax** and **hyperjump-io** (JSON Schema Tooling).
- Learned about **GSoC** and wrote a proposal for the Swift project after thorough **research** and **contributions**
- Selected as a GSoC mentee with Swift and worked on implementing **DocC Language Features in Sourcekit-LSP**.

Motivation



Source code has rich IDE features like **Diagnostics**, **Go-to Definition**, **Syntax Highlighting**.

But it is **not the same for documentation**.

Then, why should documentation be plain text?

Example:

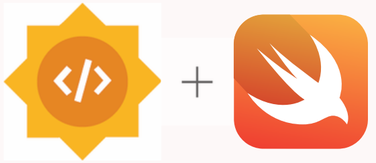
```
/// Creates a new instance of ``MyClass``.
///
/// See ``MyClass/myMethod(x: )`` for more information.
func createInstance() -> MyClass
```

If a developer wants to verify whether a symbol link like ``*MyClass/myMethod(x:)*`` exists

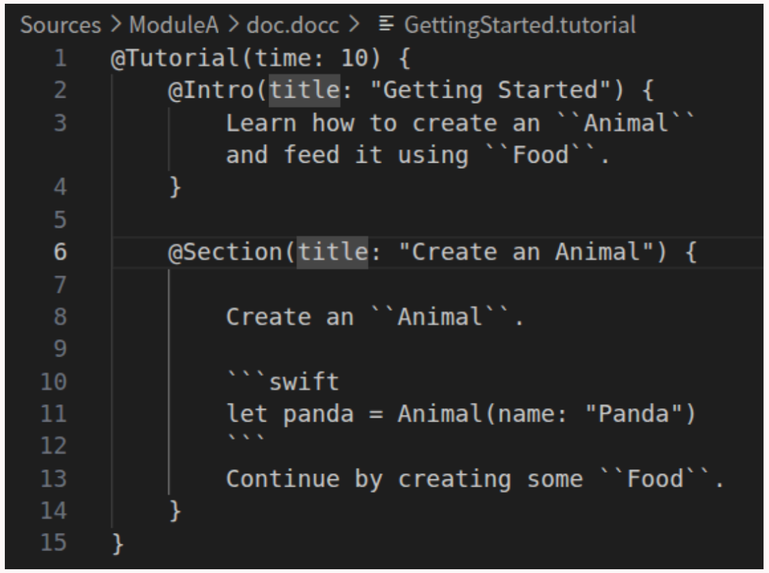
They have to build the entire documentation → check the link → fix it, if necessary.

What if symbol links in documentation had the same IDE features as source code?

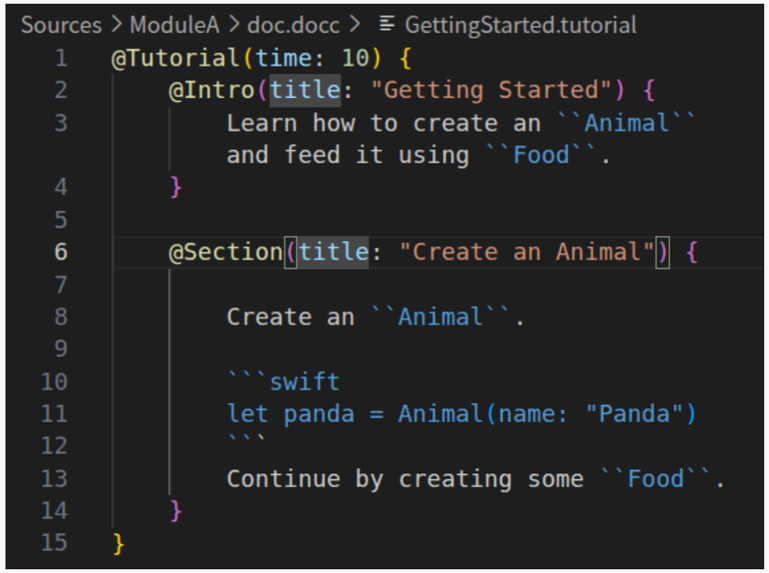
Bringing IDE Features to Documentation



- **Go-To-Definition** for symbol links in DocC Comments, Tutorial and Markdown files.
- **Diagnostics** for missing or invalid symbol links
- **Clickable links** in the Documentation Preview
- **Syntax highlighting** for DocC Tutorial and Markdown files.

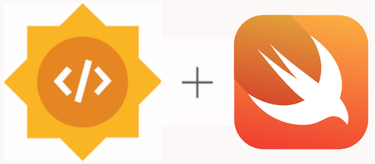


Before



After

What is DocC and SourceKit-LSP?



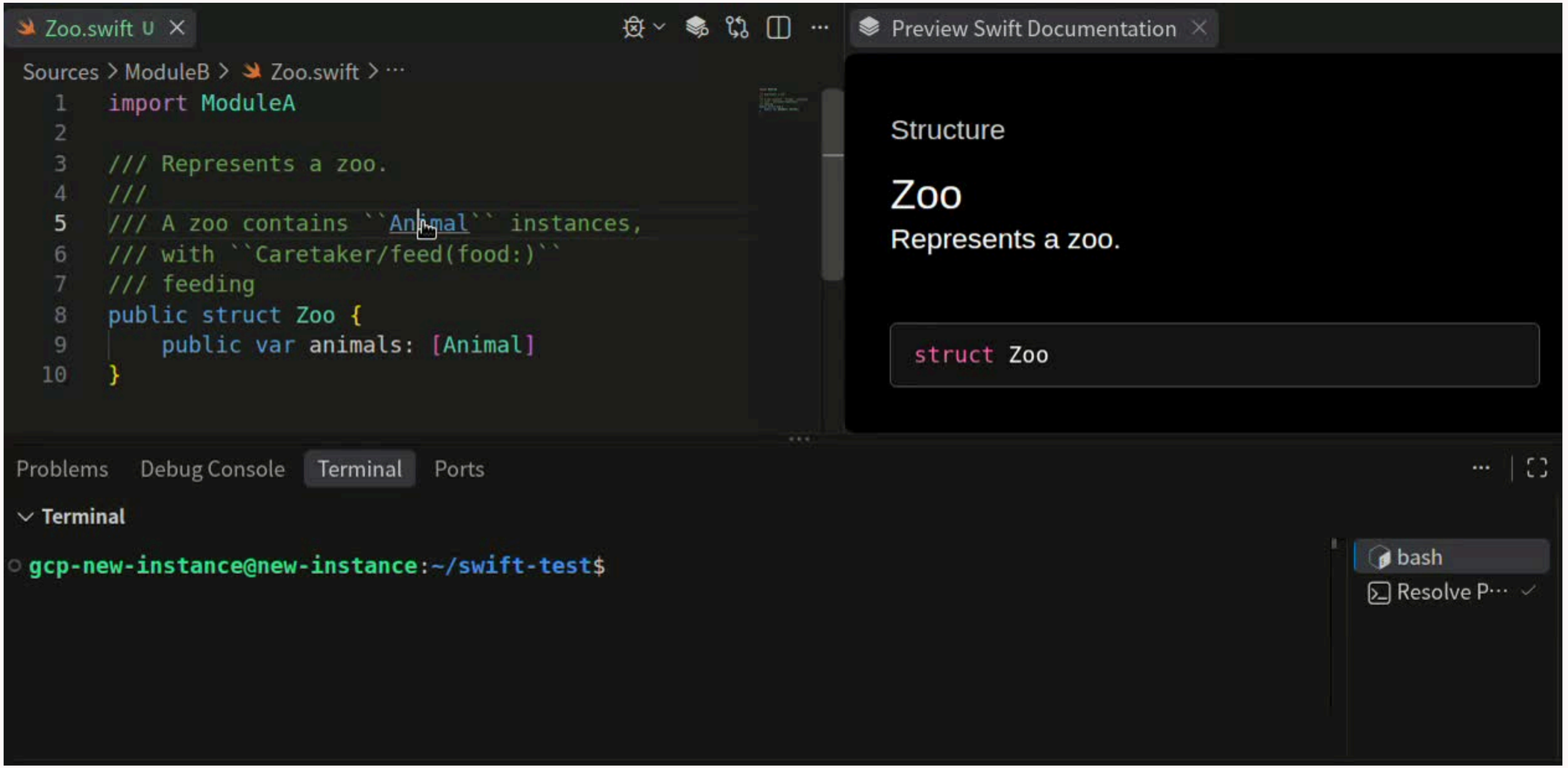
Documentation compiler

- Converts documentation comments and files into a **structured documentation** website.
- **DocC** is Apple's documentation compiler.

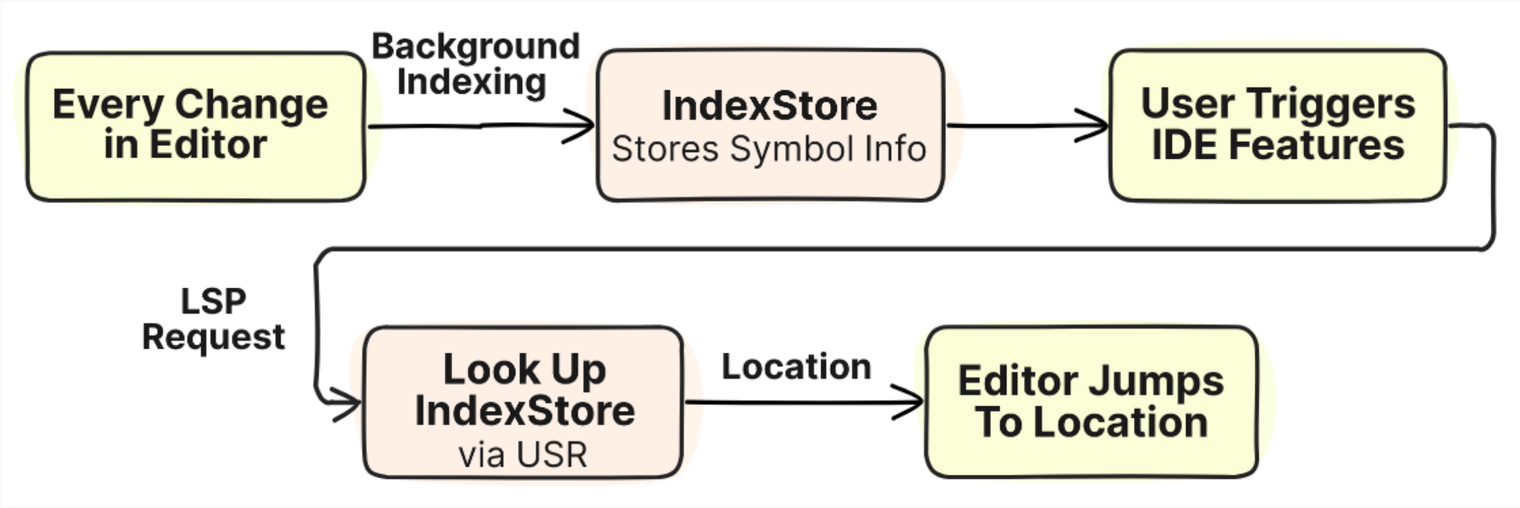
Language Server

- Connects the editor to the language and **provides IDE features** such as diagnostics, auto-complete and navigation.
- **SourceKit-LSP** provides these features for Swift and C-based languages.

Demo Video


Zoo.swift U
Sources > ModuleB > Zoo.swift > ···
1 import ModuleA
2
3 /// Represents a zoo.
4 ///
5 /// A zoo contains ``Animal`` instances,
6 /// with ``Caretaker/feed(food:)``
7 /// feeding
8 public struct Zoo {
9 public var animals: [Animal]
10 }
Preview Swift Documentation
Structure
Zoo
Represents a zoo.
struct Zoo
Problems
Debug Console
Terminal
Ports
Terminal
gcp-new-instance@new-instance:~/swift-test$
bash
Resolve P···

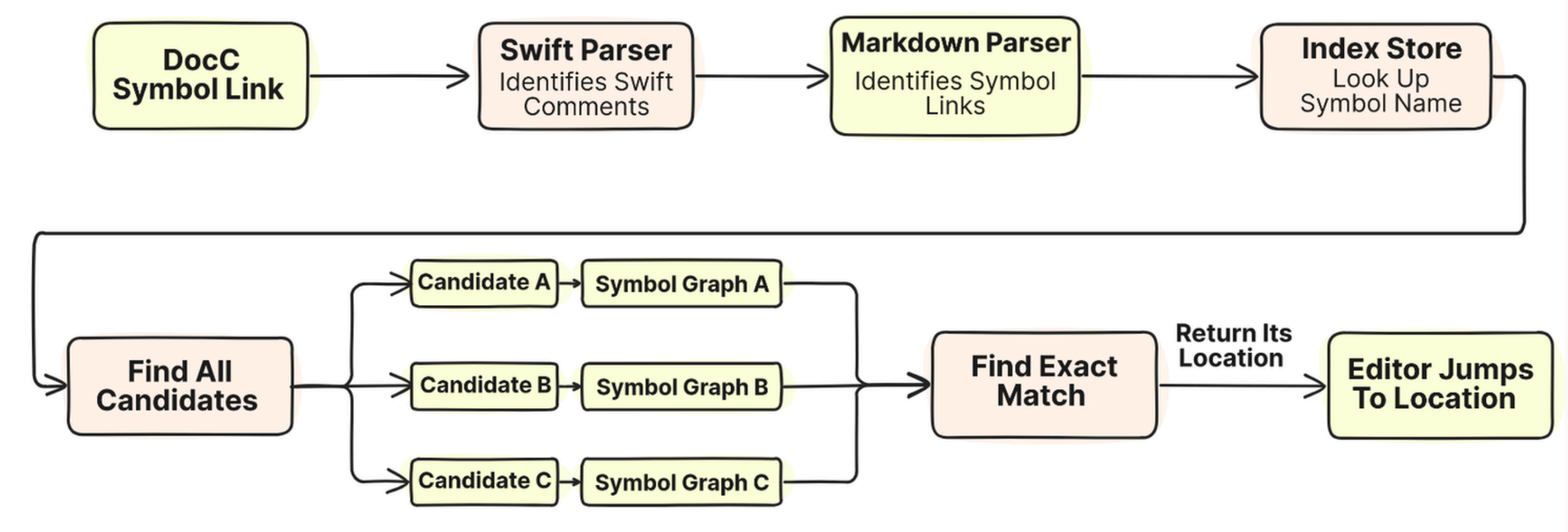
How does it work for Swift symbols?



But what about DocC symbol links?

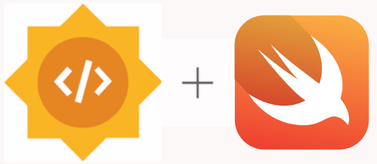
- Documentation is written inside **comments**, which the Swift parser **ignores** during indexing.
- Swift symbols can be looked up using their USR, which **uniquely** identifies each symbol.
- DocC symbol links **don't have a USR**, so the same approach cannot be used.

How was it solved?



For more details:

Current Status : All the features are working locally and are under active review upstream



Thank You!

Let’s Connect!