IMPERIAL

Early Career Researcher Institute

Further Website Development for Software Projects

Jay DesLauriers

Imperial College London

Intended Learning Outcomes

- Appreciate and produce high quality, accessible, and readable documentation
- Use Material for MkDocs to render Markdown files into a working website
- Automate the MkDocs build process with GitHub Actions continuous integration
- Customise the look, feel and navigation of the generated MkDocs pages
- Understand best practices like tutorial-based and documentation-driven development

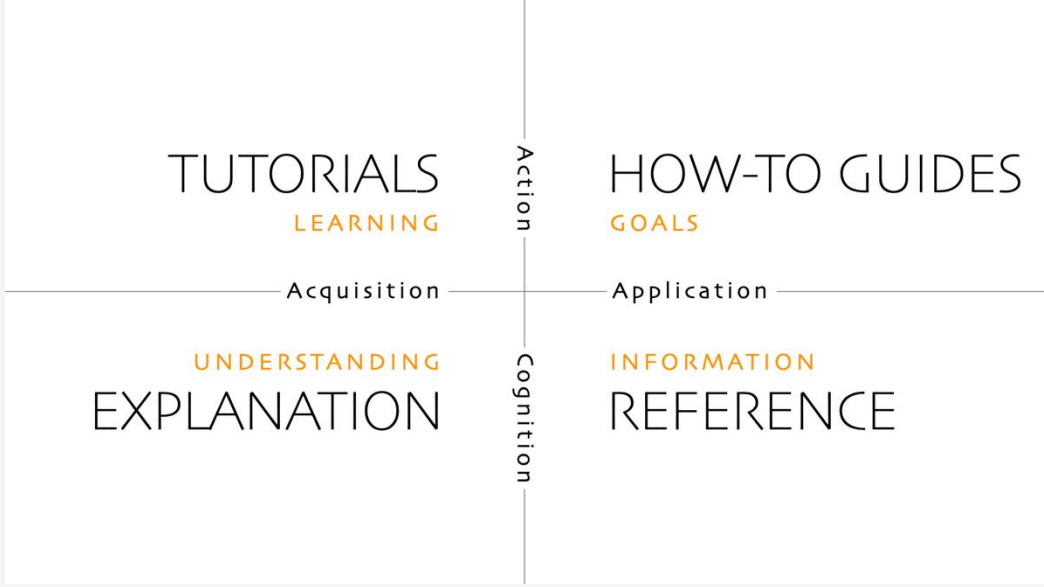
Documentation

...why does it matter?

- Good docs reduce support burden and onboarding time for new users
- Documentation is often the first impression of your software project
- Poor docs lead to misuse, frustration, and abandonment of your tool
- Well-structured docs help contributors understand and extend your code
- Writing docs forces you to think clearly about your own design decisions

Documentation Frameworks

...structuring your docs



- Diataxis[1] organises docs into four forms:
  - **Tutorials**: learning-oriented, guided experiences for newcomers
  - **How-to guides**: task-oriented, solving a specific problem step by step
  - **Explanation**: understanding-oriented, clarifying concepts and design decisions
  - **Reference**: information-oriented, exhaustive technical details of your API

1. diataxis.fr

Documentation Frameworks

...beyond the four types

- Chris Nicholas[2] uses those forms as sections, and extends with three more:
  - **Quickstarts**: minimal, step-by-step setup to get running fast
  - **Examples**: small, single-feature demos with copy-pasteable code
  - **Templates**: full production-ready starter projects for real-world use

2. chrisnicholas.dev/blog/how-to-write-exceptional-documentation

Development Practices

...writing your docs

Documentation-driven Development

- Write the docs before the code
- Clarifies design and API decisions early
- Catches usability issues before they ship

Docs as Code

- Write the docs with the tools you use to write code
- Version-control your docs with git
- Test and review your docs

These ideas and more from an authority on documentation: writethedocs.org

MkDocs (Material)

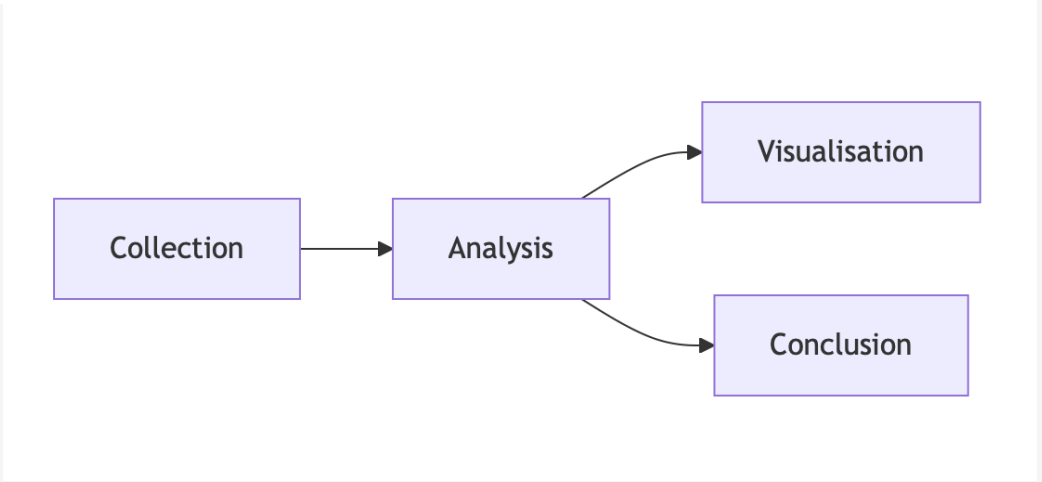
...presenting your docs

- Static Site Generation for documentation pages
- Written in Markdown, rendered to HTML/JS

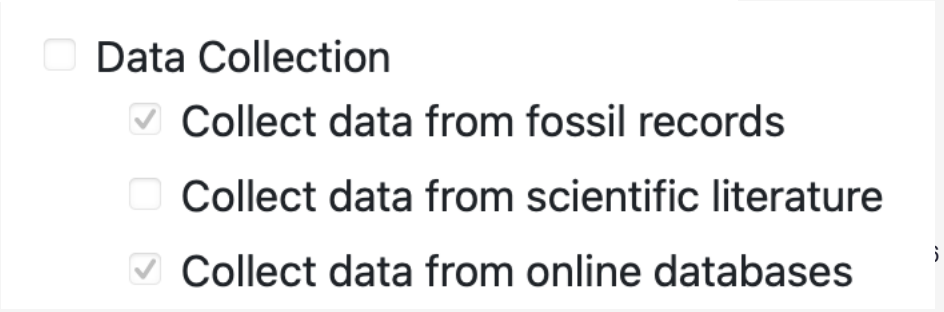
- Extended by plugins...
  - Search, tags, autodocs

- ...and extensions
  - Flow charts, task lists, callouts

````
```mermaid
flowchart LR
  Collection --> Analysis --> Visualisation
                 Analysis --> Conclusion
```
````




```
- [ ] Data Collection
  - [X] Collect data from fossil records
  - [ ] Collect data from scientific literature
  - [X] Collect data from online databases
```



- Also consider:
  - Astro Starlight
  - Docusaurus

GitHub Actions

...automating your docs

- Automatically run code or scripts, e.g.
  - Building and deploying MkDocs
  - Checking for broken links
- Defined by a workflow YAML file, which has
  - Name, triggers, permissions
  - Jobs and steps. In our case:
    - Checkout (clones your repository)
    - Setup Python
    - Install Material for MkDocs
    - Build & deploy

.github > workflows > ! publish.yml

```yaml
name: ci
on:
  push:
    branches:
      - main
permissions:
  contents: write
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v5
      - uses: actions/setup-python@v5
        with:
          python-version: 3.x
      - run: pip install mkdocs-material
      - run: mkdocs gh-deploy --force
```

Thank you!