Devcontainer für dieses Quarto-Projekt

Diese DevContainer-Konfiguration stellt eine Entwicklungsumgebung bereit, in der Quarto und eine LaTeX-Umgebung (XeLaTeX) installiert sind. Damit kannst du die Website bauen und PDF-Ausgaben erzeugen.

Was ist enthalten - `.devcontainer/Dockerfile` — baut ein Ubuntu-basiertes Image mit Quarto CLI und einer minimalen TeX-Installation (`xelatex`). - `.devcontainer/devcontainer.json` — VS Code DevContainer-Konfiguration.

Schnellstart 1. Öffne das Projekt in VS Code. 2. Drücke F1 → `Remote-Containers: Reopen in Container` (oder verwende das grüne Remote-Icon). VS Code baut dann das Image (das erste Mal kann es einige Minuten dauern).

Testen innerhalb des Containers - Um die Website lokal zu rendern und eine Vorschau zu starten, öffne ein Terminal in VS Code (wird im Container ausgeführt) und führe aus:

```
\ExtensionTok{quarto}\NormalTok{ preview}
```

- Um die komplette Seite zu rendern (statische Ausgaben in `docs/`):

```
\ExtensionTok{quarto}\NormalTok{ render}
```

- Für PDF-Ausgabe einer einzelnen Datei:

```
\ExtensionTok{quarto}\NormalTok{ render path/to/file.qmd
}\AttributeTok{{-}{-}to}\NormalTok{ pdf}
```

Extra: Notebooks/Assets in `docs/` kopieren

Wenn du sicherstellen willst, dass zusätzliche Dateien (z. B. `.ipynb`, `.txt`, `.pdf`) unter `docs/` landen und über GitHub Pages erreichbar sind, führe nach dem Rendern das mitgelieferte Kopier□Script aus. Beispiel (trockenlauf):

```
\ExtensionTok{python3}\NormalTok{ scripts/copy\_notebooks\_to\_docs.py
}\AttributeTok{{-}{-}dry{-}run} \AttributeTok{{-}{-}ext}\NormalTok{
ipynb,txt,pdf}
```

Wenn der Dry□Run passt, kopiere tatsächlich:

```
\ExtensionTok{python3}\NormalTok{ scripts/copy\_notebooks\_to\_docs.py
}\AttributeTok{{-}{-}ext}\NormalTok{ ipynb,txt,pdf}
```

Du kannst beide Schritte auch in einer Zeile koppeln (die Kopie läuft nur, wenn das Rendern erfolgreich ist):

```
\ExtensionTok{quarto}\NormalTok{ render }\AttributeTok{{-}{-}execute}
\AttributeTok{{-}{-}no{-}cache} \KeywordTok{\&\&}
\ExtensionTok{python3}\NormalTok{ scripts/copy\_notebooks\_to\_docs.py
}\AttributeTok{{-}{-}ext}\NormalTok{ ipynb,txt,pdf}
```

Hinweis: Das einfache Kopieren in `docs/` reicht aus, damit der Link in der gerenderten HTML auf die Dateien funktioniert. Wenn du hingegen möchtest, dass Quarto die `.ipynb` selbst als zusätzliches Output□Format auflistet (erscheint unter “Other formats”), dann musst du Quarto anweisen, beim Rendern auch `ipynb` als Ausgabe zu erzeugen (z. B. `--to html,ipynb,pdf`).

Hinweis - Wenn du zusätzliche LaTeX-Pakete benötigst, kann es nötig sein, das Dockerfile zu erweitern (z. B. `texlive-pictures`, `texlive-lang-german` usw.). - Die DevContainer-Konfiguration legt das Arbeitsverzeichnis unter `/workspace` und verwendet den Benutzer `vscode`.

Manueller Docker-Build (optional)

Wenn du VS Code nicht verwenden möchtest, kannst du das Image manuell bauen und einen Container starten:

```
\CommentTok{\# aus dem Ordner .devcontainer (ein Verzeichnis höher als das
Projekt{-}Root)}
\ExtensionTok{docker}\NormalTok{ build }\AttributeTok{{-}t}\NormalTok{
quarto{-}devcontainer }\AttributeTok{{-}f}\NormalTok{
.devcontainer/Dockerfile ..}

\ExtensionTok{docker}\NormalTok{ run }\AttributeTok{{-}{-}rm}
\AttributeTok{{-}it} \DataTypeTok{\textbackslash{}}
    \AttributeTok{{-}v} \StringTok{"}\VariableTok{$(}\BuiltInTok{pwd}\Variable
Tok{)}\StringTok{:/workspace"}
\DataTypeTok{\textbackslash{}}
    \AttributeTok{{-}w}\NormalTok{ /workspace
}\DataTypeTok{\textbackslash{}}
\NormalTok{    quarto{-}devcontainer bash}
```

Im Container kannst du dann `quarto render` oder `quarto preview` ausführen.