

Übersicht

Ein gut gefütterte Notiz setzt sich aus einem Titel, sog. Properties (Metadaten wie Tags oder selbst definierte Datenfelder) und den eigentlichen Notizen zusammen.

Properties sind Optional, sie helfen jedoch beim wiederfinden von Informationen. Zwar können Tags auch in den Notizen gesetzt werden, jedoch ist es übersichtlicher diese in den Properties als Metadaten einzugeben.

The screenshot shows a Notion workspace with a sidebar on the left containing various databases and notes. The main area displays a note titled 'Go - Cheat Sheet'. The note has a 'Properties' section with 'tags' set to 'Go' and 'CheatSheet'. Below the properties is a table of contents with sections like 'Variablen', 'Funktionen', 'Benannte Rückgabewerte', 'Schleifen', 'Bedingungen', and 'Zeiger'. The 'Funktionen' section is expanded, showing a code snippet for a swap function and a note about multiple return values. The 'Benannte Rückgabewerte' section is also expanded, showing a note about using defer to manage function calls. The right sidebar shows a list of links and a search bar.

Go - Cheat Sheet

Properties

tags: Go, CheatSheet

+ Add property

4 > **Variablen** ...

110 **Funktionen**

111 **Mehrere Rückgabewerte**

112 Es kann nicht nur eine Variable zurückgegeben werden, sondern es können bei der Funktionsdefinition mehrere Rückgabewerte angegeben werden.

113

114 `func swap(a, b string) (string, string) {`

115 `return b, a`

116 `}`

117

118 > **Benannte Rückgabewerte** ...

127 **Ausführung einer Funktion zurückstellen**

128 Mittels `defer` ist es möglich, die Ausführung einer Funktion zurückzustellen. Diese wird dann erst ausgeführt, wenn die übergeordnete Funktionen (welche die zurückgestellte Funktion aufruft) beendet ist, also dessen `return` erreicht ist.

129 **Mehrere defer**

Es können mehrere `defer` angegeben werden. Diese werden dann in umgekehrter Reihenfolge aufgerufen. D. h. das zuerst angegebene `defer` wird als letztes aufgerufen und das zuletzt angegebene `defer` als erstes nach dem `return`. Die `defer` Statements werden auf einen Stack gelegt und nach dem Last-In-First-Out Prinzip verarbeitet.

131

132

133 `func main() {`

134 `defer fmt.Println("world") // wird nach hello ausgegeben`

135 `fmt.Println("hello")`

136 `}`

137

138 > **Schleifen** ...

161 > **Bedingungen** ...

226 > **Zeiger** ...

0 Rückverweise 1.085 Wörter 7.565 Zeichen Sync: +3 ↓0

Version #1

Erstellt: 24 Februar 2024 20:25:25 von Marcel

Zuletzt aktualisiert: 24 Februar 2024 20:30:18 von Marcel