ChatGPT

WiFiWebManager – Entwickler-Übersicht & Schnellreferenz

Diese Datei fasst **alle Nutzungsmöglichkeiten, API-Optionen und Besonderheiten** der WiFiWebManager-Library in kompakter Form zusammen.\ Nutze sie als „Cheat-Sheet“ für eigene Projekte und bei der Weiterentwicklung.

---

1. Grundstruktur

Einbinden & Instanziieren:

```
#include <WiFiWebManager.h>

WiFiWebManager wifiWebManager;
```

---

2. Lebenszyklus im Sketch

setup():

```
void setup() {
    Serial.begin(115200);
    wifiWebManager.begin();
    // eigene Seiten hinzufügen (siehe unten)
}
```

loop():

```
void loop() {
    wifiWebManager.loop();
    // optional: z.B. Hardware-Reset auswerten
}
```

---

3. Eigene Seiten und die Startseite anpassen

Die Startseite „/“ (Home) selbst belegen

Standardmäßig liefert `/` eine leere Seite (nur Menü).\ **Du kannst im Hauptprogramm eine eigene Startseite so einrichten:**

```
wifiWebManager.addPage("Home", "/", [](AsyncWebServerRequest *request) {
    return "<h1>Willkommen!</h1><p>Dies ist die zentrale Startseite deines
ESP32!</p>";
});
```

Der Menüpunkt „Home“ (ganz vorne) verweist immer auf `/` und erscheint nur oben. Der eigene Handler für `/` überschreibt den Standard-Handler – die Seite erscheint **nicht** ein zweites Mal im Custom-Menü.

WLAN-Konfiguration unter /wlan

Der Menüpunkt „WLAN“ verweist immer auf `/wlan` und öffnet die WLAN-Konfigurationsseite.\ Du musst dafür nichts weiter tun – das Routing ist in der Library fest eingebaut.

Weitere Seiten hinzufügen

Wie gewohnt:

```
wifiWebManager.addPage(
    "MeinMenü", "/meinpfad",
    [](AsyncWebServerRequest *request) {
        return "<h1>Meine Seite</h1><p>Eigener Inhalt!</p>";
    },
    // Optional: POST-Handler
    [](AsyncWebServerRequest *request) {
        return "<p>Gespeichert!</p>";
    }
);
```

- **GET-Handler:** Muss immer angegeben werden; gibt den HTML-Body als String zurück.
- **POST-Handler:** Optional; gleiche Signatur, ebenfalls HTML-String.

---

4. Persistente Datenspeicherung (Key/Value)

```
wifiWebManager.saveCustomData("mykey", "meinwert");
String wert = wifiWebManager.loadCustomData("mykey", "default");
```

- Die Daten bleiben auch nach Neustart erhalten.

---

5. Menü-System

- **Oben:** Standardseiten:
- Home (`/`)
- WLAN (`/wlan`)
- NTP (`/ntp`)
- Firmware (`/update`)
- Reset (`/reset`)
- **Darunter:** Alle eigenen Seiten in zweiter Zeile, mit automatischem Umbruch bei vielen Links
- Markierung des aktiven Links automatisch (CSS-Klasse `selected`)
- Die Startseite `/` erscheint **niemals** als Custom-Link

---

6. WLAN-Konfiguration & WebInterface

- Bei Erststart: ESP32 als Access Point „ESP32_SETUP“ (192.168.4.1)
- Nach erfolgreicher WLAN-Konfiguration: Modul im Client-Modus, WebInterface weiter verfügbar
- Statische IP möglich
- Hostname konfigurierbar

---

7. Firmware-Update (OTA)

- Im Menüpunkt „Firmware“ kann per Web-Upload ein neues Firmware-Binary geladen werden.

---

8. Werksreset

- Über Web („Reset“) oder per Code:

```
wifiWebManager.reset();
```

Alle Einstellungen werden gelöscht, Modul startet im AP-Modus neu.

---

9. Typische Sketch-Struktur

```
#include <WiFiWebManager.h>
WiFiWebManager wifiWebManager;

void setup() {
    Serial.begin(115200);
    wifiWebManager.begin();

    // Eigene Startseite
    wifiWebManager.addPage("Home", "/", [](AsyncWebServerRequest *request) {
        return "<h1>Willkommen!</h1><p>Deine ESP32-Projektzentrale.</p>";
    });

    // Weitere Seiten
    wifiWebManager.addPage("Info", "/info", [](AsyncWebServerRequest
*request){
        return "<h1>Info</h1><p>Projekt XYZ</p>";
    });
}

void loop() {
    wifiWebManager.loop();
}
```

---

10. Alle Methoden im Überblick

Methode	Funktion
`begin()`	Initialisiert das Modul
`loop()`	Muss im Hauptloop laufen
`addPage(menutitel, pfad, get, post)`	Fügt eine neue Seite/Menüpunkt hinzu
`removePage(pfad)`	Entfernt eine eigene Seite/Menüpunkt
`saveCustomData(key, wert)`	Speichert einen Wert persistent
`loadCustomData(key, default)`	Liest einen Wert (oder Default)
`reset()`	Setzt alle Einstellungen zurück, startet AP-Modus

---

11. Hinweise und Besonderheiten

- **UTF-8:**\ Content-Type ist überall korrekt gesetzt (`text/html; charset=utf-8`) → Umlaute/Sonderzeichen funktionieren.

- **Menü/Design:**\ Responsive, mobilfähig, Standardseiten oben, eigene Seiten immer in der zweiten Zeile.

- **AsyncWebServer:**\ Nutzt ESPAsyncWebServer (und AsyncTCP).\ → Richtiges Package für ESP32 verwenden!

- **Kompatibilität:**\ Library, Ordner, Klasse und Includes **immer exakt „WiFiWebManager“** (großes F!).

---

12. Beispiel: Hardware-Reset per Taster

```
void loop() {
    wifiWebManager.loop();
    if (digitalRead(0) == LOW) {
        wifiWebManager.reset();
        delay(1000); // Entprellen
    }
}
```

---

13. Typische Fehlerquellen

- **Fehler „.h not found“:**\ → Library-Ordner/Dateinamen prüfen (`WiFiWebManager`!), IDE neu starten.

- **Fehler im Menü/CSS:**\ → Custom- und Standardseiten immer über die bereitgestellten Methoden einbinden.

---

14. Erweiterbarkeit

- Du kannst beliebig viele eigene Seiten (GET/POST) und Menüpunkte anlegen!
- API bleibt stabil, auch bei zukünftigen Updates der Library.

Letzte Änderung:\ Juni 2024

Viel Spaß mit der Library und happy hacking!