**Generic Class**

Methods and properties described in this section are specific to ESP8266. They are not covered in [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi)documentation. Before they are fully documented please refer to information below.

**onEvent**

void onEvent (WiFiEventCb cb, WiFiEvent\_t event**=**WIFI\_EVENT\_ANY) \_\_attribute\_\_((deprecated))

To see how to use onEvent please check example sketch [WiFiClientEvents.ino](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiClientEvents/WiFiClientEvents.ino) available inside examples folder of the ESP8266WiFi library.

**WiFiEventHandler**

WiFiEventHandler onStationModeConnected (std::function**<** void(const WiFiEventStationModeConnected **&**)**>**)

WiFiEventHandler onStationModeDisconnected (std::function**<** void(const WiFiEventStationModeDisconnected **&**)**>**)

WiFiEventHandler onStationModeAuthModeChanged (std::function**<** void(const WiFiEventStationModeAuthModeChanged **&**)**>**)

WiFiEventHandler onStationModeGotIP (std::function**<** void(const WiFiEventStationModeGotIP **&**)**>**)

WiFiEventHandler onStationModeDHCPTimeout (std::function**<** void(void)**>**)

WiFiEventHandler onSoftAPModeStationConnected (std::function**<** void(const WiFiEventSoftAPModeStationConnected **&**)**>**)

WiFiEventHandler onSoftAPModeStationDisconnected (std::function**<** void(const WiFiEventSoftAPModeStationDisconnected **&**)**>**)

To see a sample application with WiFiEventHandler, please check separate section with [examples :arrow\_right:](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-examples.md) dedicated specifically to the Generic Class..

**persistent**

WiFi**.**persistent (persistent)

Module is able to reconnect to last used Wi-Fi network on power up or reset basing on settings stored in specific sectors of flash memory. By default these settings are written to flash each time they are used in functions like WiFi.begin(ssid, password). This happens no matter if SSID or password has been actually changed.

This might result in some wear of flash memory depending on how often such functions are called.

Setting persistent to false will get SSID / password written to flash only if currently used values do not match what is already stored in flash.

Please note that functions WiFi.disconnect or WiFi.softAPdisconnect reset currently used SSID / password. If persistent is set to false, then using these functions will not affect SSID / password stored in flash.

To learn more about this functionality, and why it has been introduced, check issue report [#1054](https://github.com/esp8266/Arduino/issues/1054).

**mode**

WiFi**.**mode(m)

WiFi**.**getMode()

* WiFi.mode(m): set mode to WIFI\_AP, WIFI\_STA, WIFI\_AP\_STAor WIFI\_OFF
* WiFi.getMode(): return current Wi-Fi mode (one out of four modes above)

**Other Function Calls**

int32\_t channel (void)

bool setSleepMode (WiFiSleepType\_t type)

WiFiSleepType\_t getSleepMode ()

bool setPhyMode (WiFiPhyMode\_t mode)

WiFiPhyMode\_t getPhyMode ()

void setOutputPower (float dBm)

WiFiMode\_t getMode ()

bool enableSTA (bool enable)

bool enableAP (bool enable)

bool forceSleepBegin (uint32 sleepUs**=**0)

bool forceSleepWake ()

int hostByName (const char **\***aHostname, IPAddress **&**aResult)

Documentation for the above functions is not yet prepared.

For code samples please refer to separate section with [examples](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-examples.rst) dedicated specifically to the Generic Class.

**EXAMPLES**

# Generic

In the first [example](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/readme.md#quick-start) of the ESP8266WiFi library documentation we have discussed how to check when module connects to the Wi-Fi network. We were waiting until connection is established. If network is not available, the module could wait like that for ever doing nothing else. Another [example](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/scan-examples.md#async-scan) on the Wi-Fi asynchronous scan mode demonstrated how to wait for scan result and do in parallel something else - blink a LED not disturbing the blink pattern. Let’s apply similar functionality when connecting the module to an access point.

## Introduction

In example below we will show another cool example of getting ESP perform couple of tasks at the same time and with very little programming.

## What are the Tasks?

We would like to write a code that will inform us that connection to Wi-Fi network has been established or lost. At the same time we want to perform some time critical task. We will simulate it with a blinking LED. Generic class provides specific, event driven methods, that will be executed asynchronously, depending on e.g. connection status, while we are already doing other tasks.

## Event Driven Methods

The list of all such methods is provided in [Generic Class](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-class.md)documentation.

We would like to use two of them: \* onStationModeGotIP called when station is assigned IP address. This assignment may be done by DHCP client or by executing WiFi.config(...). \* onStationModeDisconnected called when station is disconnected from Wi-Fi network. The reason of disconnection does not matter. Event will be triggered both if disconnection is done from the code by executing WiFi.disconnect(), because the Wi-Fi signal is weak, or because the access point is switched off.

## Register the Events

To get events to work we need to complete just two steps:

1. Declare the event handler:

cpp   WiFiEventHandler disconnectedEventHandler;

1. Select particular event (in this case onStationModeDisconnected) and add the code to be executed when event is fired.

cpp   disconnectedEventHandler = WiFi.onStationModeDisconnected([](const WiFiEventStationModeDisconnected& event)   {     Serial.println("Station disconnected");   });If this event is fired the code will print out information that station has been disconnected.

That’s it. It is all we need to do.

## The Code

The complete code, including both methods discussed at the beginning, is provided below.

#include <ESP8266WiFi.h>

const char\* ssid = "\*\*\*\*\*\*\*\*";

const char\* password = "\*\*\*\*\*\*\*\*";

WiFiEventHandler gotIpEventHandler, disconnectedEventHandler;

bool ledState;

void setup()

{

Serial.begin(115200);

Serial.println();

pinMode(LED\_BUILTIN, OUTPUT);

gotIpEventHandler = WiFi.onStationModeGotIP([](const WiFiEventStationModeGotIP& event)

{

Serial.print("Station connected, IP: ");

Serial.println(WiFi.localIP());

});

disconnectedEventHandler = WiFi.onStationModeDisconnected([](const WiFiEventStationModeDisconnected& event)

{

Serial.println("Station disconnected");

});

Serial.printf("Connecting to %s ...\n", ssid);

WiFi.begin(ssid, password);

}

void loop()

{

digitalWrite(LED\_BUILTIN, ledState);

ledState = !ledState;

delay(250);

}

## Check the Code

After uploading above sketch and opening a serial monitor we should see a similar log:

Connecting to sensor**-**net **...**

Station connected, IP: 192.168**.**1.10

If you switch off the access point, and put it back on, you will see the following:

Station disconnected

Station disconnected

Station disconnected

Station connected, IP: 192.168**.**1.10

The process of connection, disconnection and printing messages is done in background of the loop() that is responsible for blinking the LED. Therefore the blink pattern all the time remains undisturbed.

## Conclusion

Check out events from generic class. They will help you to write more compact code. Use them to practice splitting your code into separate tasks that are executed asynchronously.

For review of functions included in generic class, please refer to the [Generic Class](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-class.md) documentation.