How to use callbacks:

What is a callback?

A callback is a function that you have writen and that is used by the library when a message arrives. You put your message handling code in this function, and tell the library to call it back when the corresponding message is received.

For example, you can have a function named DoSomeStuffWithNoteOn, that will look like this:

```
void DoSomeStuffWithNoteOn (byte channel, byte pitch, byte velocity) {

        // Do some stuff with NoteOn here

}
```

The arguments of this function are the data from the message received. When the function is called back from within the library, it will be passed the data in the arguments, so you can access them more easily than calling MIDI.getData1, etc..

How do I use them?

First, you need to set up the correct function arguments. These vary from one type to another (because all types have different data). These function must also return nothing (void).

Here is what your function parameters should look like for each type:

```
void HandleNoteOff (byte channel, byte note, byte velocity)
void HandleNoteOn (byte channel, byte note, byte velocity)
void HandleAfterTouchPoly (byte channel, byte note, byte pressure)
void HandleControlChange (byte channel, byte number, byte value)
void HandleProgramChange (byte channel, byte number)
void HandleAfterTouchChannel (byte channel, byte pressure)
void HandlePitchBend (byte channel, int bend)
void HandleSystemExclusive (byte *array, byte size)
void HandleTimeCodeQuarterFrame (byte data)
void HandleSongPosition (unsigned int beats)
void HandleSongSelect (byte songnumber)
void HandleTuneRequest (void)
void HandleClock (void)
void HandleStart (void)
void HandleContinue (void)
void HandleStop (void)
void HandleActiveSensing (void)
```

```
void HandleSystemReset (void)
```

Once you have your function correctly set, you need to tell the library to use it when the corresponding message is received. Let's take back the example of a NoteOn, handled by the DoSomeStuffWithNoteOn function.

```
void setup() {
        MIDI.begin();
        MIDI.setHandleNoteOn(DoSomeStuffWithNoteOn);
}

void loop() {
        MIDI.read();
}
```

And that's all! Just put the name of the function, and the library will automatically call it when a NoteOn is received.

If for some reason, you want to disconnect the callback, you can do it like this:

```
MIDI.disconnectCallbackFromType(NoteOn);
```

My sketch is very slow, what's going on?

First, you need to call MIDI.read quite fast, to have the lowest latency possible. Raw MIDI data is stored in the Serial buffer, but the library can only extract it when you call MIDI.read.

Make sure that your callbacks are running fast, as they are launched from within MIDI.read (thus in the loop).