Event Handling

Handling Objects and the Introduction of Functionality in Programs

How Event Functions Work

(1) Declaring an Event Table

1. An event table must be declared in **Frame.h**.

```
#pragma once

#include <wx/wx.h>

class Frame : public wxFrame
{
public:
    Frame();

private:
    DECLARE_EVENT_TABLE()
```

2. This event must then be initialized in **Frame.cpp**.

```
BEGIN_EVENT_TABLE(Frame, wxFrame)
    EVT_BUTTON(10001, Frame::setName)
    EVT_BUTTON(10002, Frame::setEmail)
    EVT_BUTTON(10003, Frame::setPhone)
END_EVENT_TABLE()
```

BEGIN_EVENT_TABLE(Frame, wxFrame) ~ **Frame (our frame class) & wxFrame (derived class)**

(2) Creating a Command Event

When creating an **event function**, said function must include a **wxCommandEvent&** object as an argument. This will categorize the function as an event function.

```
void setName(wxCommandEvent&);
```

(3) Connecting Event Function to an Object & Using Event Functions

1. Within the initialized event table, we will connect an **object's ID** to an event function. Our **wxCommandEvent** will then inherit whichever object caused the event, like a button.
    1. **Example (Line 4):** EVT_BUTTON(10001, Frame::setName)
        1. When ID 10001 is pressed (button to set full name), it will cause setName to execute.
        2. **wxCommandEvent** in this function shows that it has taken the button which caused the event and inherits the button's ID (printed within the console).
2. Event types
    1. For a button, EVT_BUTTON is used, for a menubar item, EVT_MENU is used, etc.
3. What is possible with event functions now?
    1. Within any event function: we may use any member variable in said class we're executing the function within (like our Frame class), use any other command available to us, or use our wxCommandEvent reference argument. Examples of other commands and processes could be closing a program or changing the background colour of a program.