Event Mechanism

[Create a JavaFX Canvas]

↳ [Write event handler to respond to event]

↳ [Register the event handler as a listener]

[User interacts with the Canvas]

↳ [The Canvas generates an event]

↳ [Event handler called to deal with event]

Event Handler Code

Write a new handler class to deal with the event:

```
class MouseHandler implements
                        EventHandler<MouseEvent> {
  void handle(MouseEvent event) {
    System.out.println("Mouse Pressed !");
  }
}
```

In "start" method create & register a new handler:

```
MouseHandler handler = new MouseHandler();
canvas.setOnMousePressed(handler);
```

This is the Observer pattern in action !

Method Reference Operator

You may see (perhaps in some online tutorials !)
The use of the Method Reference Operator ::
for event handling in JavaFX:

```
canvas.setOnMousePressed(this::handleMousePressed);
```

This is fine and "some" people like this way

It's a bit too much like Javascript for my liking
(And it doesn't work with processing !)

Anonymous Inline Classes

You might also see things like the following:

```
canvas.setOnKeyPressed(new EventHandler<KeyEvent>() {
  public void handle(KeyEvent event) {
    System.out.println("Key Pressed");
  }
});
```

Although this works just fine, it's pretty ugly (IMHO)
This is just a simple example...
Imagine something much more complex !

My advice: Use named event handlers !

More Events !

JavaFX Canvas supports a whole variety of events:

- Keys: pressed, released, typed
- Mouse buttons: pressed, released, clicked
- Mouse movement: moved, dragged, enter, exit
- Mouse scroll wheel: start, scrolling, finish

There are drag & drop events, but I'd avoid these !
(They are a bit fiddly, with a complex API)

Also some events specific to mobile devices
(touch, swipe, rotate etc)