Event-Driven Programming

Programming Languages

William Killian
Millersville University
Outline

• Event Types
• Handlers
• Event-Driven Architectures
• Case Studies:
  • Java (with Swing)
  • JavaScript (with HTML)
Events

• What are events?

• What type of events do we care about?

• How should a programming language handle the different types of events?
Event Types

• Key press
  • Key release
• Button click
• Mouse move
• Mouse drag
  • Mouse down
  • Mouse up
• Touch?
What do we do with Events?

• We ultimately want to execute some (arbitrary) code when a particular event happens
  • A mouse move should be a unique event compared to a button click or a key press
  • Different classes of events may have different options
• We need an abstraction to this code that should run when different events occur
  • This abstraction is called a **Handler**
Event Handler

• A piece of code that runs in response to an event

We may wait here for a while...

We want to wait for more events

1. Show *Interface*
2. User triggers *Event*
3. *Handle* Event
Event Handler

• Often modeled as a function or special Interface
• Usually takes in an event record
• An event record may contain information:
  • The source – what element caused the event to trigger
  • The Event Type – click, press, drag, down, up, etc
  • Metadata – any extra information
• Programmers should be able to react to different events in different ways
Event-Driven Architecture
Event-Driven Architectures

• How does a Graphical User Interface work?
• How do I ...
  • ... create contents?
  • ... show a window?
  • ... add new elements?
  • ... update elements?

• ... all while still do other things?!
Step 1: Create another thread

- A *normal* program runs on a *single thread*
- A thread is a single control flow of instructions and data. Basically, we can only do one thing at a time.
- GUI-based programs often need to do more than one thing, so a GUI framework will often run its own thread.
Event-Driven Architectures

Step 2: Design an Event Creation Model

• Define an event Hierarchy
  • Examples to Follow...

• Answer hard questions:
  • What types of components should trigger certain events?
  • What information should be available to the programmer when an event occurs?
  • Can the programmer add additional information? How?
Event-Driven Architectures

Step 3: Hook in Event Creation to your Components

Components should have the ability to trigger events. (For a button, it’s as simple as knowing its pressed)

• Each component will keep track of a List of Handlers
• When a component triggers an Event:
  1. Create a new Event
  2. Send the Event to each handler in the handler list
  3. Each handler will process the event on the same “thread”
Java Swing

- Various Components:
  - JTextField (text box)
  - JRadioButton (radio button)
  - JButton (a normal button)
  - JPanel (a place to put many components)
  - JFrame (a window)

- Programmatically construct a user interface (rather than visually)

- A **LayoutManager** can be applied to a panel for arrangement of components in a particular way
Java Swing – Component Creation

```java
final var field =
    new JTextField("Enter your name");

final var button =
    new JButton("Click Me");

final var label =
    new JLabel("A simple label");
```
Java Swing – Component Creation

```java
final var panel = new JPanel();
final var layout =
    new BoxLayout(panel, BoxLayout.Y_AXIS);
panel.setLayout(layout);
panel.add(field);
panel.add(button);
panel.add(label);
```
Java Swing – Window Creation

```java
final var window = new JFrame("My App");
window.setContentPane(panel);
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.pack();
window.setVisible(true);
```
final Counter count = new Counter();
button.addActionListener((ActionEvent e) -> {
    int counter = count.increment();
    String text = new StringBuilder()
        .append(field.getText())
        .append(" has clicked ")
        .append(counter)
        .append(" times")
        .toString();
    label.setText(text);
    window.pack();
});
Components and HTML

• I’m not covering HTML in this class 😊
• DOM – Document Object Model
  • Provides an interface within JavaScript to create/access/manipulate components

Javascript:
```
var btn = document.getElementById("button");
bbtn.addEventListener("click", () => {
    alert("you clicked the button");
});
```

Components and HTML

```html
<html>
<head>
  <title>My App</title>
</head>
<body>
  <input type="text" id="name" value="Enter your name">
  <input type="button" id="button" value="Click Me"/>
  <span id="output"></span>
  <script type="text/javascript" src="app.js"></script>
</body>
</html>
```
```javascript
var counter = 0;
function domReady() {
    var btn = document.getElementById("button");
    btn.addEventListener("click", () => {
        counter += 1
        var text = document.getElementById("output");
        var field = document.getElementById("name");
        text.textContent = field.value + " clicked the button " + counter + " times";
    });
}
```
if (document.readyState === "complete" ||
    (document.readyState !== "loading" &&
    !document.documentElement.doScroll)) {
    domReady();
} else {
    document.addEventListener("DOMContentLoaded",
        "DOMContentLoaded",
        domReady);
}