MATLAB GUI Programming

Event-Driven Programming

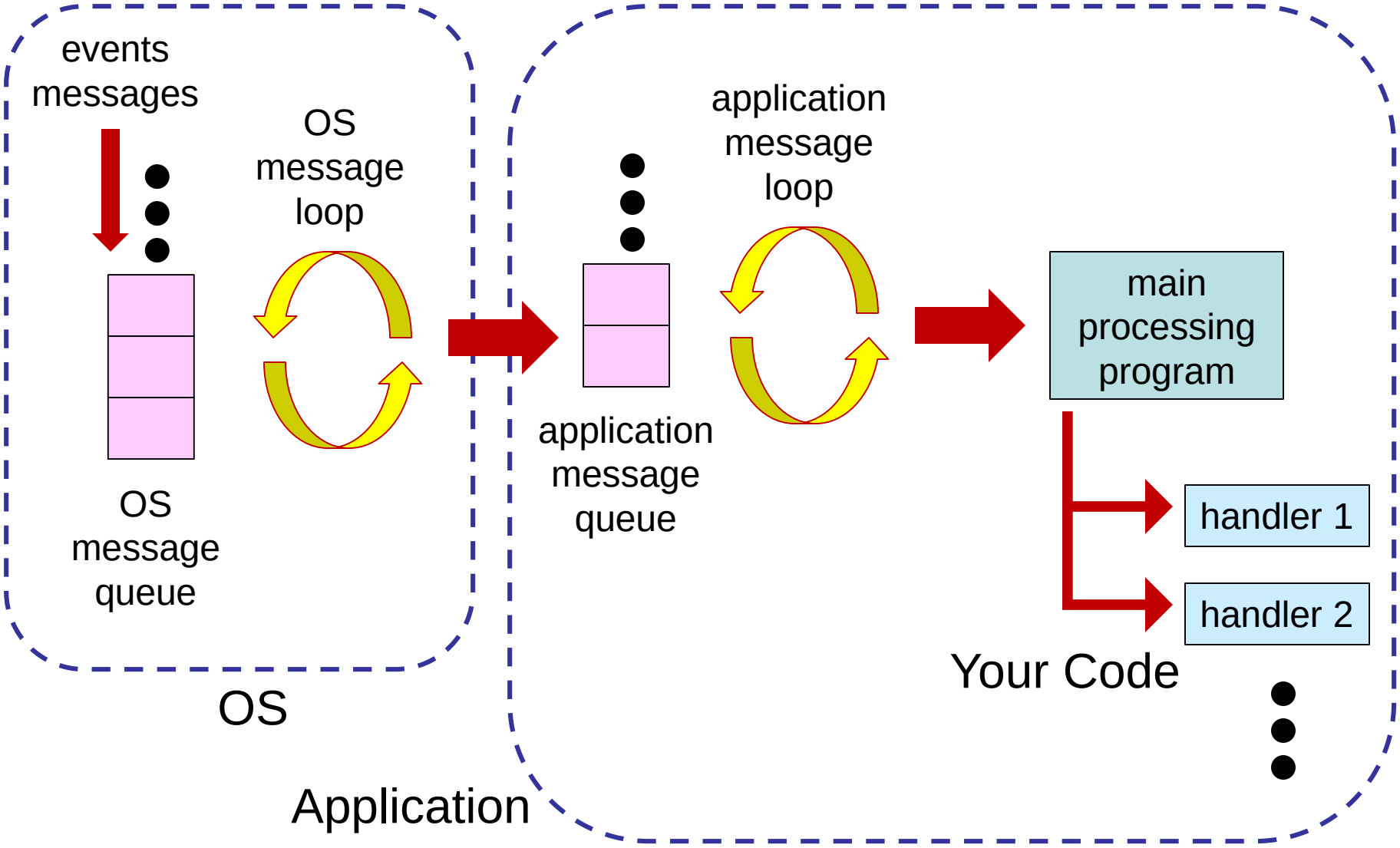
- In an interactive computing environment, a user interacts with a program mostly using the **mouse** or the **keyboard**.
- When a user does anything with the mouse or the keyboard, an **event** is triggered and trapped by the OS. A message about the event is generated and includes its associated information (e.g., the "character typed" in a keyboard-press event, or the screen coordinates in a mouse-left-click event).
- There are many other types of events. In addition to the users, events can also come from other programs, the network, hardware devices, the file system, etc. The exact event types available depend on the OS and the system/devices used.

Event-Driven Programming

- If the OS determines that an event is for a window, the event is passed to the window for processing.
  - The window is said to have the "focus".
  - In MATLAB, this is the current figure window; you can get its handle with the function **`gcf`**.
- For a figure window managed by your code, you can determine how to process these events in the code. You can also ignore events that you don't want to process.

Event-Driven Programming

A simplified illustration of event-based programming:



Interacting with MATLAB Figures

Two approaches:

- Let a figure respond directly to keyboard/mouse events:
  - No GUI objects (buttons, etc.).
  - The command window blocks to wait for user events.
  - Simple to program for very basic user interactions.
- Windows-like GUI programming (← our focus):
  - A number of GUI objects available.
  - MATLAB Apps
  - You can continue to use the command window and do other things while the app is still running.

The Simple Approach for Interaction

- With no GUI components, mouse-click or keyboard-press events are sent directly to the current figure.
- Put the processing within a **`while`** loop (our own message loop).
- The function **`waitforbuttonpress`** blocks and waits for the user event. The return value indicates the event type (**`0`** for mouse clicks and **`1`** for keypoard presses):
- Pressing **Ctrl-C** or closing the figure window breaks the wait and generates an error message.
- Query the **`CurrentCharacter`** and **`CurrentPoint`** properties of the current figure/axes to retrieve the associated information.

The Simple Approach for Interaction

- A typical code example that simulates a message loop:

```
initialization here
while 1
  some processing here
  figure(1);  % need a figure to receive the events
  show something in the figure
  tt = waitforbuttonpress;
  if tt == 1
    c = get(gcf, 'CurrentCharacter');
    process keyboard inputs
  else
    r = get(gca, 'CurrentPoint');
    pt = r(1, 1:2); % coordinates in 2-D
    process mouse clicks
  end
  if ..... % conditions to terminate the loop
    break;
  end
end
```

MATLAB GUI Programming

- A lot more functionality and flexibility.
- A **MATLAB App** is a class derived from the **AppBase** class. It includes a figure as a **property**.
- You can add:
  - GUI components as properties.
  - Properties for your internal data.
  - Methods that are event handlers (callbacks).
  - Custom functions as methods.
- You don't need to implement the message loop itself; it is handled in the base class. It will determine the GUI object to receive the event and call the applicable event handler.

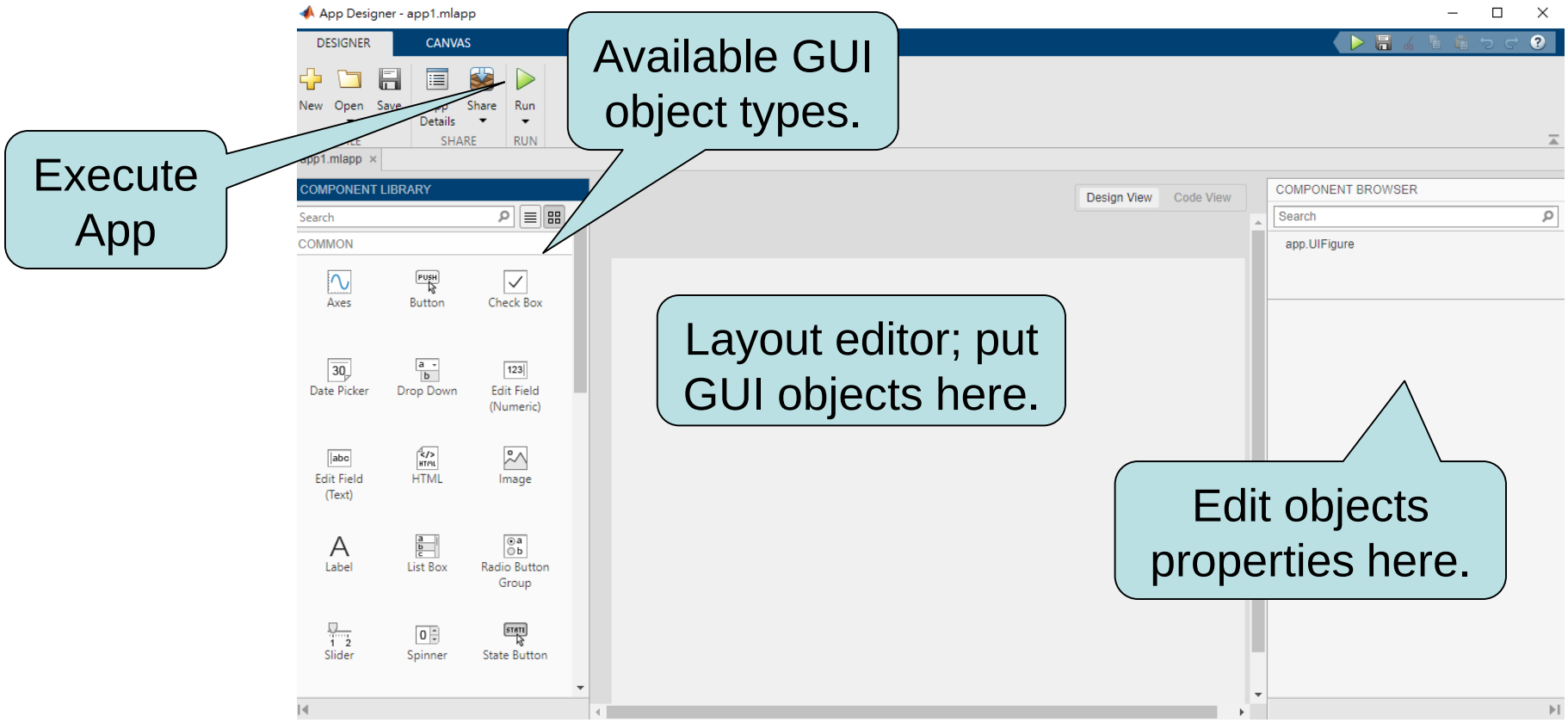
Building MATLAB Apps

- While it is possible to code a GUI program from scratch, today's GUI programming environments often provide tools for the integrated development of the GUI layout and implementation.
- We use MATLAB **App Designer** for this purpose. The app file has the extention ***.mlapp**. (Older versions of MATLAB uses **GUIDE**, which produces a pair of ***.fig** and ***.m** files.)

MATLAB App Designer

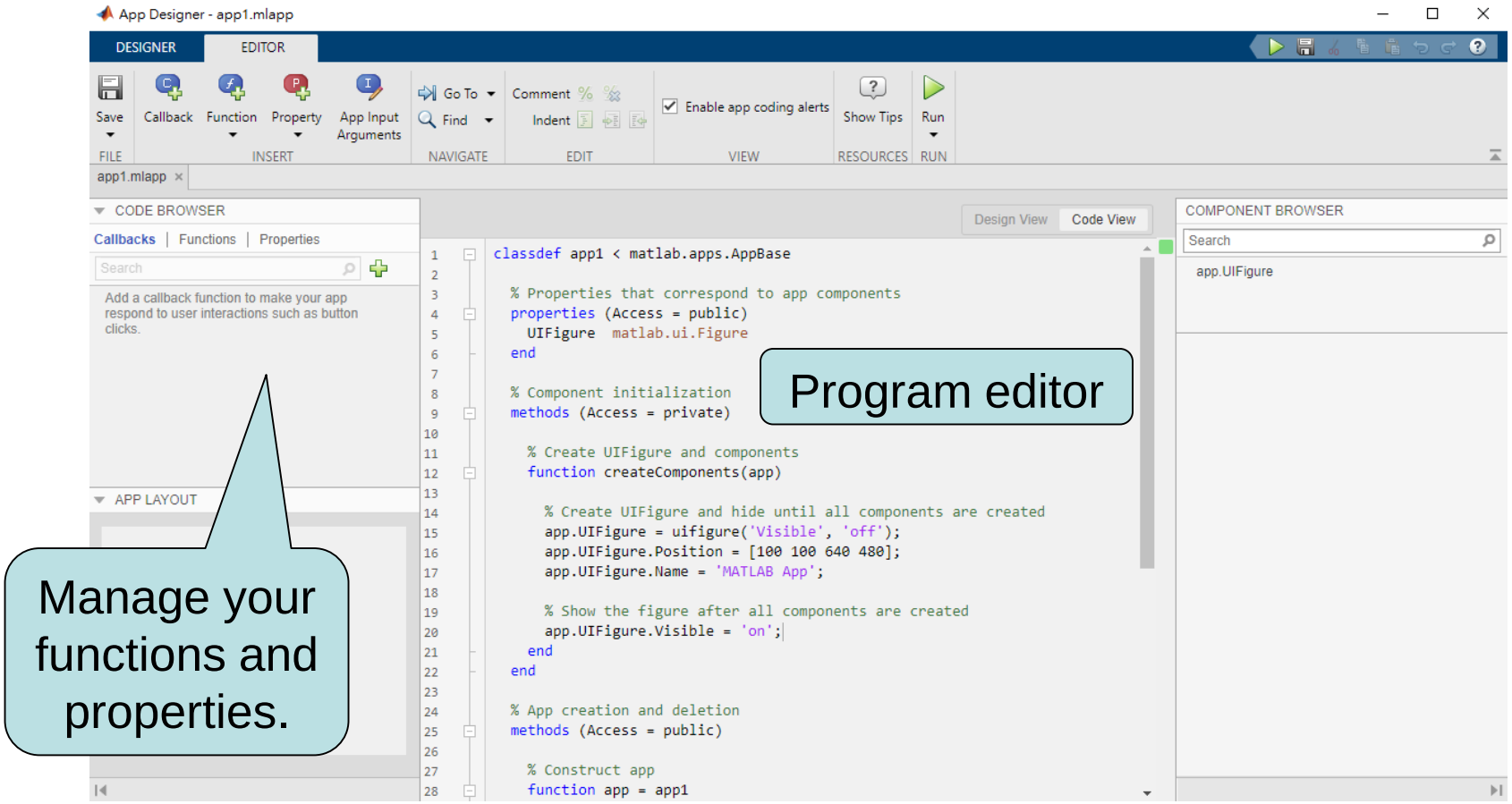
From the main menu, select **APPS**, then **Design App**, then **Blank App**. (There are other app templates or tutorials that you can play with on your own.)

A blank App is created for you and shown in a separate window. You'll see the "**Design View**" first:



MATLAB App Designer

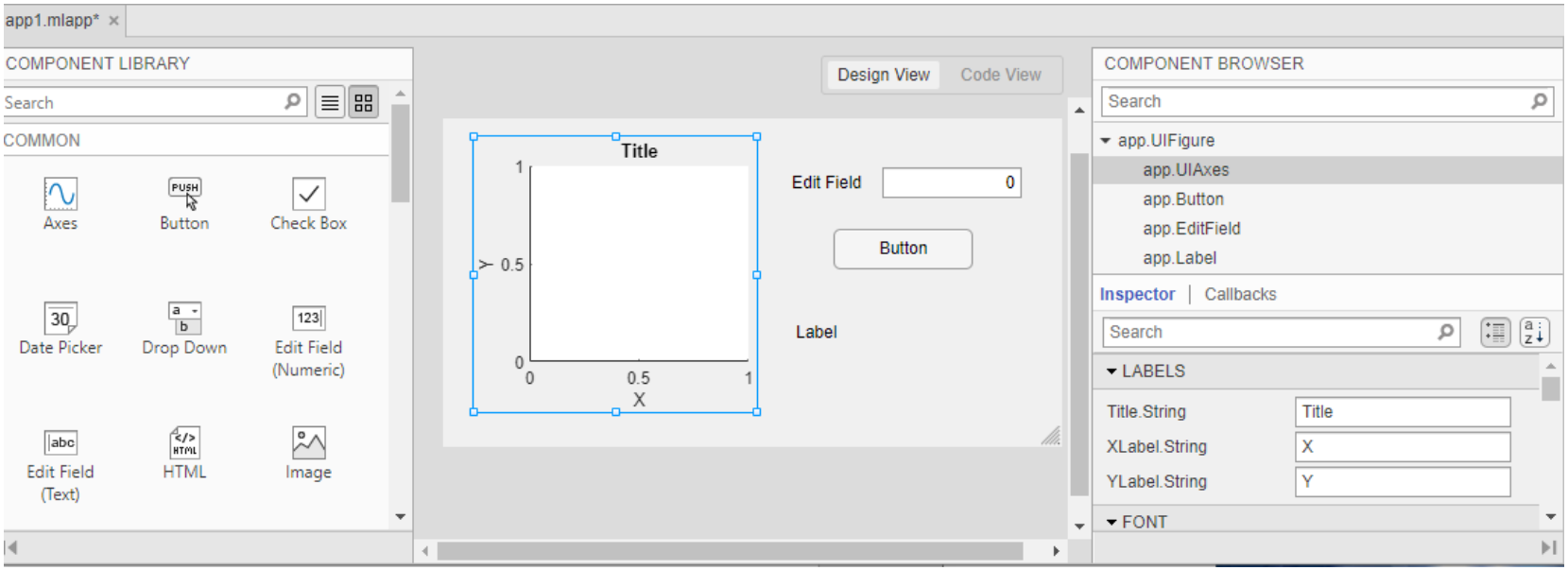
The "Code View":



Shaded portions are the "backbone" that are not use editable.

Using App Designer: Example

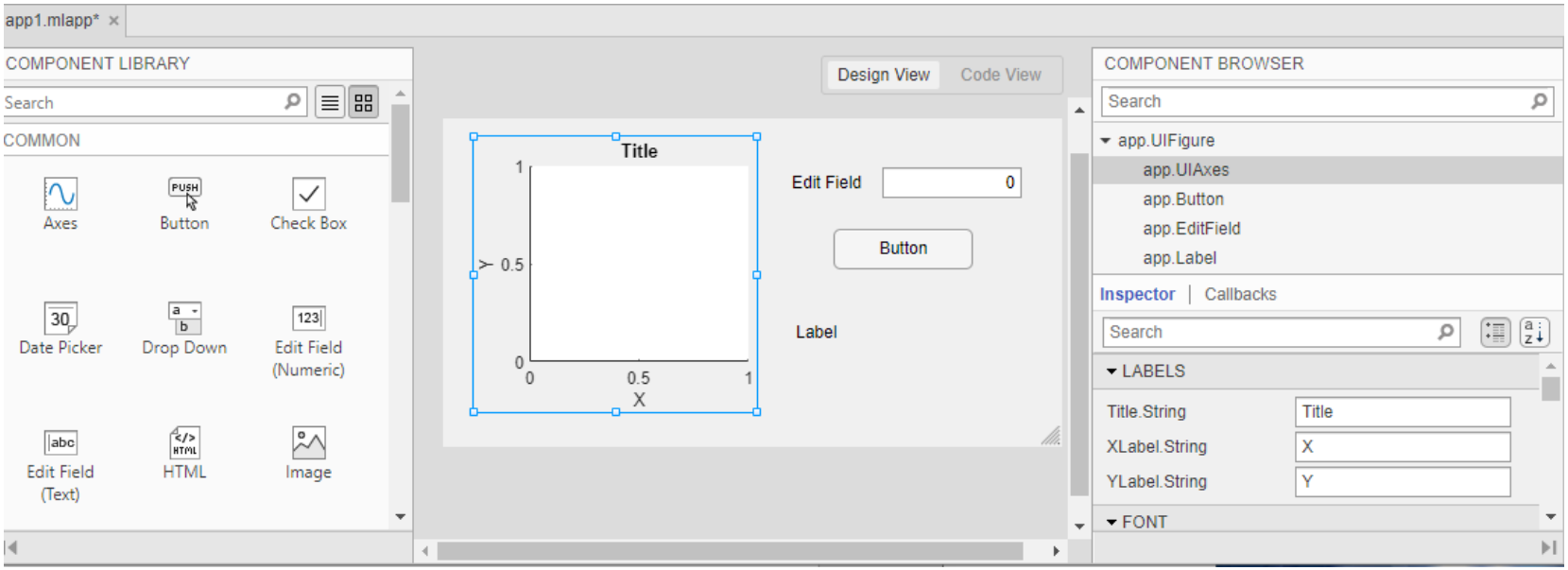
A sample app with four components:



- Drag-and-drop the desired components into the layout editor and adjust their sizes and positions.
- Rename the GUI components (these will become their names within the app class) and edit their properties in the component browser.
- Save your app; you can specify the name of the app here.

Using App Designer: Example

Go to the code view to find the components added.



- Use the "+" in the code browser to add functions/callbacks and properties for internal data.
- **Please follow along with the lecture video …**