



MATLAB User Interfaces

Week 9

Not covered in the book



Graphical User Interface

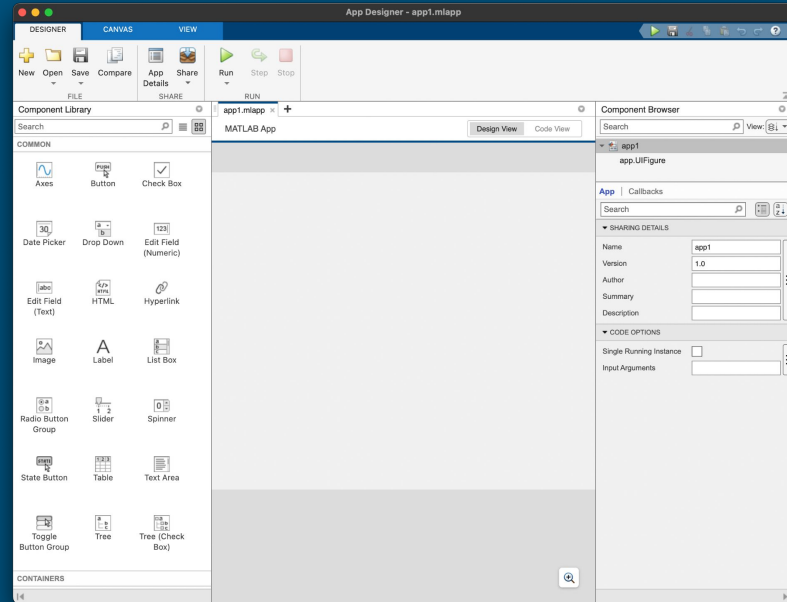
- User interaction thus far has been limited to `input()` function calls
- Graphical User Interfaces (GUIs) are a more sophisticated means of interaction including
 - Buttons
 - Text fields
 - Drop downs
 - etc.

APPDESIGNER

- MATLAB has a tool called APPDESIGNER (Formerly GUIDE - Graphical User Interface Development Environment)
- APPDESIGNER is used to create GUIs
- To open APPDESIGNER, simply type `appdesigner` in the command window

APPDESIGNER

- The **appdesigner** command opens a figure window which contains all of the tools necessary for developing a GUI.
- APPDESIGNER generates a .mlapp file



GUI .mlapp File

- Allows you to run the GUI
- Contains the functions required to launch and control the GUI
- Contains mostly *callback* functions
 - Subfunctions within the file
 - Written mostly by you
 - Determine what action is taken when a user interacts with the GUI

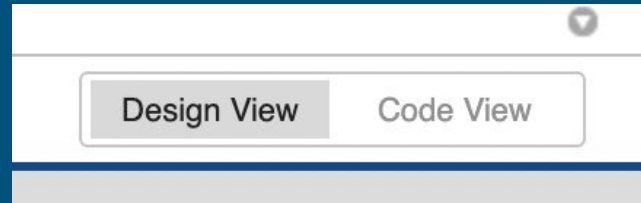
Example

As an example, let's get the current time

- Run **appdesigner** from the command line
- Place a push button in the layout area
 - Click and drag or click and then draw with the crosshairs cursor
- Select the button to view its **Properties**
- Change the button's **Text** property to '**Time**'
 - You should see the text on the button change to **Time**
- Change the button's **Name** to '**btnGetTime**'

Example

- Select the “Callbacks” tab from the **Component Browser** (be sure the button is still selected)
- Click the dropdown for “**ButtonPushedFcn**”
- Select “<**Add ButtonPushedFcn Callback**>”
 - This will automatically open “Code View” and create a function for you.
 - The function created will be called “**btnGetTimePushed**” and accepts two arguments
 - We will not be editing any of the arguments
- At any time, you can switch between Design View and Code View using the toggle buttons in the upper right.



Example

Now let's review some functions we will use in our logic

- `datetime` (formerly `clock`) provides the current date and time in vector form
 - `hour`, `minute`, `second` will be used to extract the specific components we want
- `sprintf` allows us to use a format specification to create a string like `fprintf`

Example

```
% get the current date and time
date = datetime("now");
% Extract the time elements and convert them to a string
time = sprintf("%02.0f:%02.0f:%02.0f", hour(date), minute(date), second(date));
% Set the String property of btnGetTime programmatically
app.btnGetTime.Text = time;
```

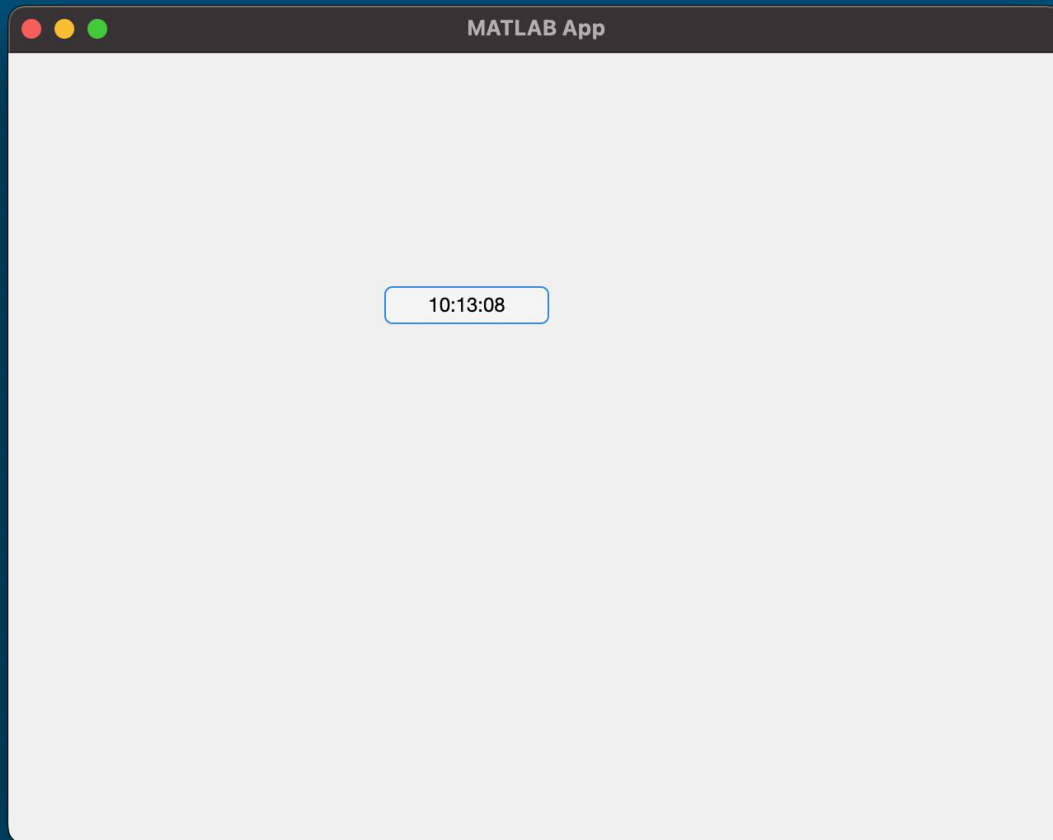
- Store the date
- Extract the hour, minute, and second portions
- Programmatically set the '**Text**' property to the current time

Example

- Save your GUI layout as something like '**TimeGUI**'
- You should see the new **TimeGUI.mlapp** file

Example

Now run the app and
click the button



General Form for GUI Objects

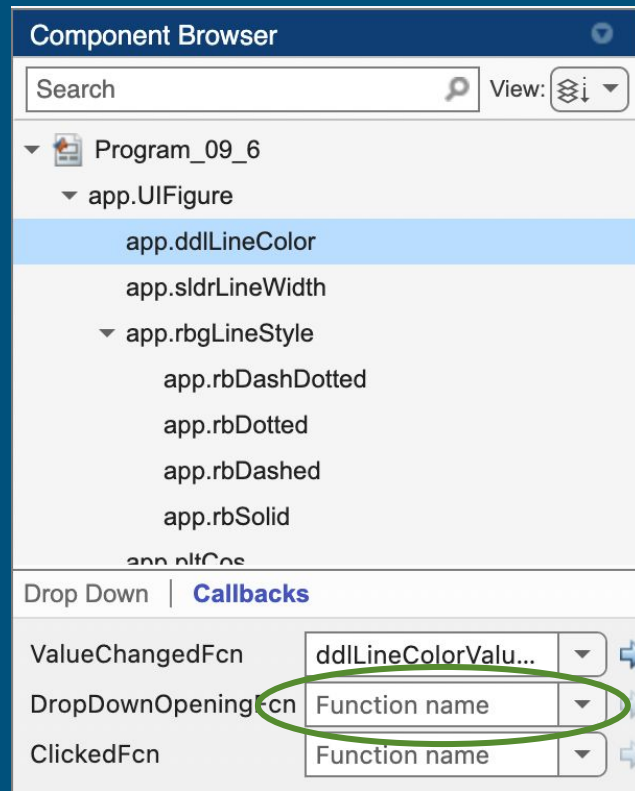
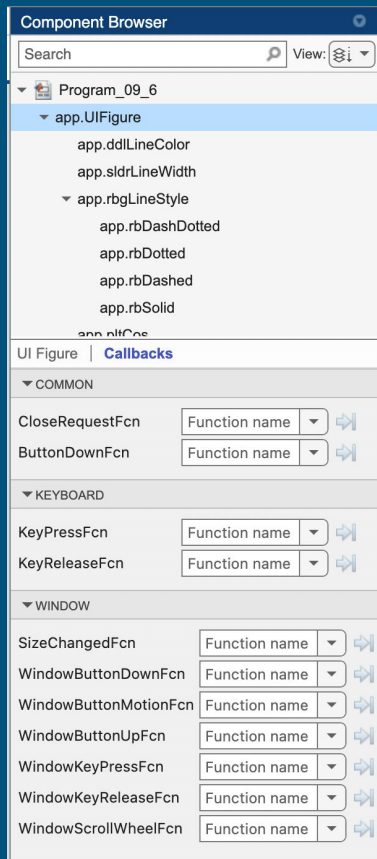
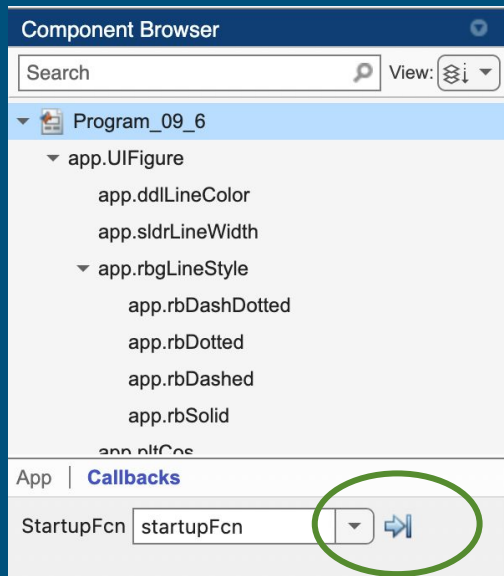
A handle is a reference to an object, a property is simply a property like “Value”. Think of it like a “link” to a website, it’s not the website itself you’re sending, just a pointer to the website

- `app` - The application as a whole (handle)
- `app.element` - Some element of the application (handle)
- `app.element.Value` - The value of some element (text, input, selection, etc)
- NOTE:
 - Properties will generally be in text form (may need to be converted to numerical form)

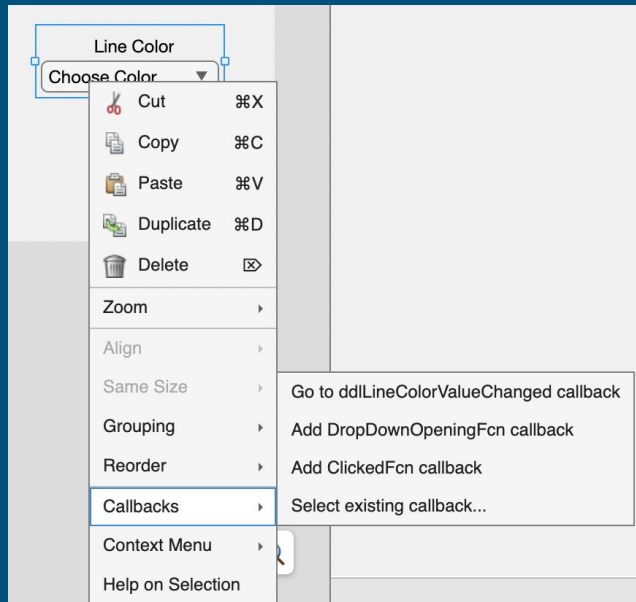
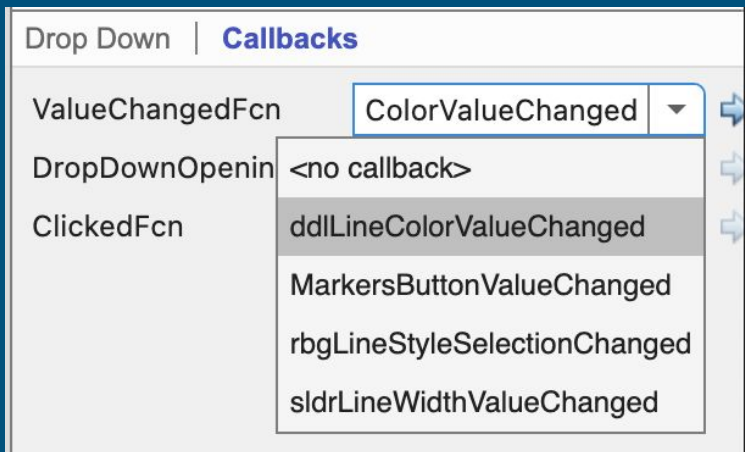
Callbacks

- Essentially functions called after some event
- Most every object has one or more
 - Button Pressed
 - Selection Changed
 - App startup
 - App shutdown

Callbacks



Accessing Callbacks



Local Functions

- Apps have local functions too
- Again, generally helper tools
- These are not callback functions
- Used to eliminate repetition (`updatePlot()`, `clearForm()`, etc)

“Workspace”

- GUIs do not have a “workspace”
- Custom data can be stored in “UserData”
- Most objects have a UserData property
 - UserData is a struct, to add data, just use the . operator
- `app.pltLinear.UserData.x = 0:100`
- `app.pltLinear.UserData.y = 0:100`
- `app.pltLinear.UserData.whatevs = “Whatevs”`

Functions need a “handle”




- Most of the functions we’ve used have no handle
 - `plot(x,y)`
 - `xlabel(“text”)`
 - etc
- With GUIs, we need to provide the handle
 - `plot(app.pltLinear, x, y)`
 - `xlabel(app.pltLinear, “text”)`
 - etc
 - (You’ve done this before with the file handle in `fprintf`)

Search around

Some assignments say

- Only accept digits
- Make a field uneditable
- Start at x
- Provide a default value on startup
- etc

Axes | Callbacks

▼ LABELS

Title.String	<input type="text" value="Cos(x)"/>
XLabel.String	<input type="text" value="X"/>
YLabel.String	<input type="text" value="Cos(x)"/>
ZLabel.String	<input type="text" value="Z"/>
Subtitle.String	<input type="text"/>
TitleHorizontalAlignment	<input type="text" value="center"/> ▼

▼ FONT

FontName	<input type="text" value="Helvetica"/> ▼
FontSize	<input type="text" value="12"/> ▼
FontWeight	<input type="text" value="B"/>

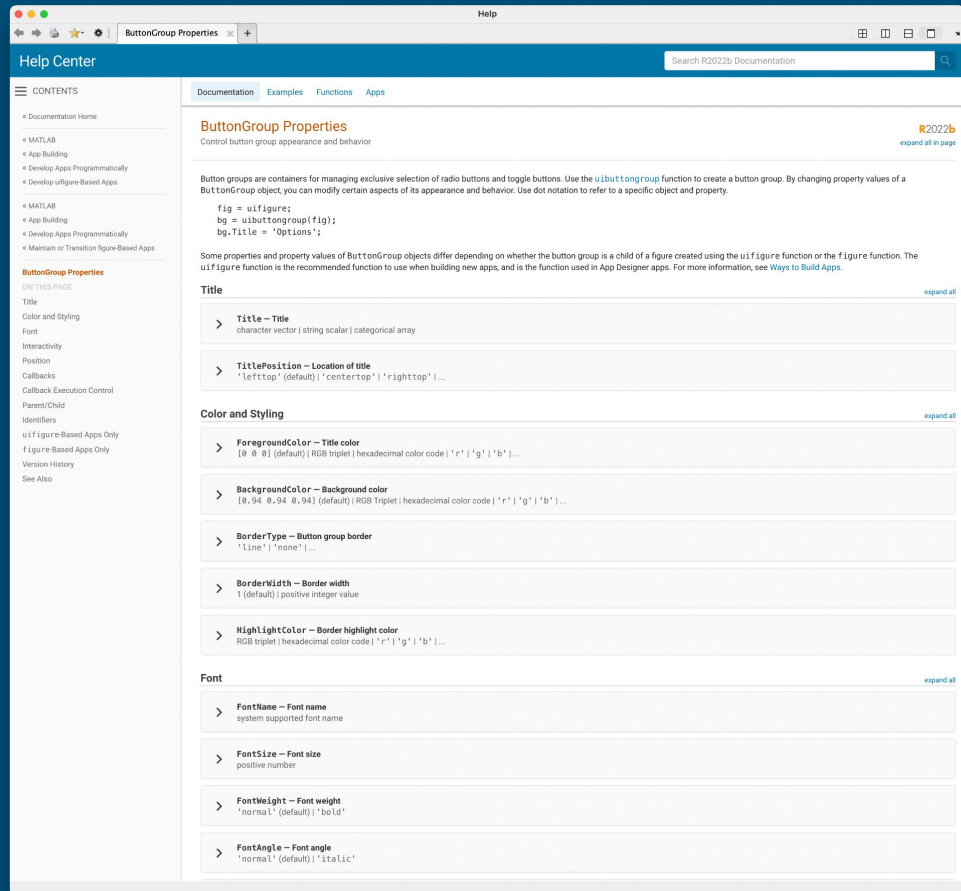
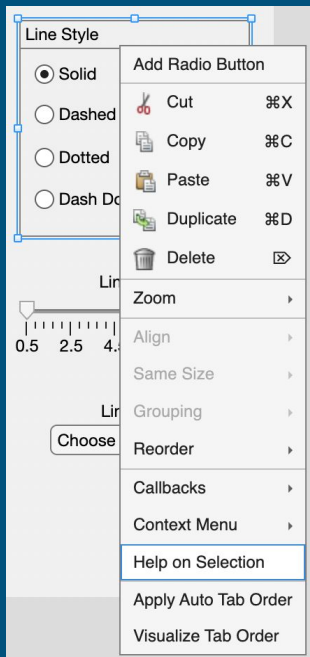
▼ TICKS

XTick	<input type="text" value="0,0.1,0.2,0.3,0.4,0.5,0.6,0"/>	⋮
XTickLabel	<input type="text" value="0,0.1,0.2,0.3,0.4,0.5,0.6,0"/>	
YTick	<input type="text" value="0,0.1,0.2,0.3,0.4,0.5,0.6,0"/>	⋮
YTickLabel	<input type="text" value="0,0.1,0.2,0.3,0.4,0.5,0.6,0"/>	

► RULERS

► GRIDS

USE THE HELP DOC



Have Fun

- Be creative
- Adjust the design to your liking
- Follow the requirements
- Keep it intuitive
- Throw an easter egg in there if you want