

# Let's Build a Simple App! 🎉

Goal: Create a “Get Current Time” App ⏳

## Week 9: GUI Development and Symbolic Math



### Objectives:

- Introduction to GUI development using App Designer 🔧.
- Utilize symbolic math capabilities ▾.
- Perform differentiation and integration ✎.



## Topics Covered:

- GUI Development with App Designer 
- Symbolic Math 

# GUI Development with App Designer



## What are GUIs?

Graphical User Interfaces (GUIs) provide a user-friendly way to interact with programs.

### Components Include:

- Buttons
- Text fields
- Drop-down menus
- Sliders
- Plots

## Why Use GUIs?

- Enhance user experience 
- Make programs accessible to non-programmers 
- Visualize data dynamically 

## Introducing App Designer

MATLAB's App Designer is a drag-and-drop environment for building professional apps.

### To Open App Designer:

```
appdesigner
```

- Generates a `.mapp` file containing your app's layout and code.

 Tip: Save your work frequently to prevent data loss!

## General Form for GUI Objects

- `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)

### Analogy

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

## Important Function Difference

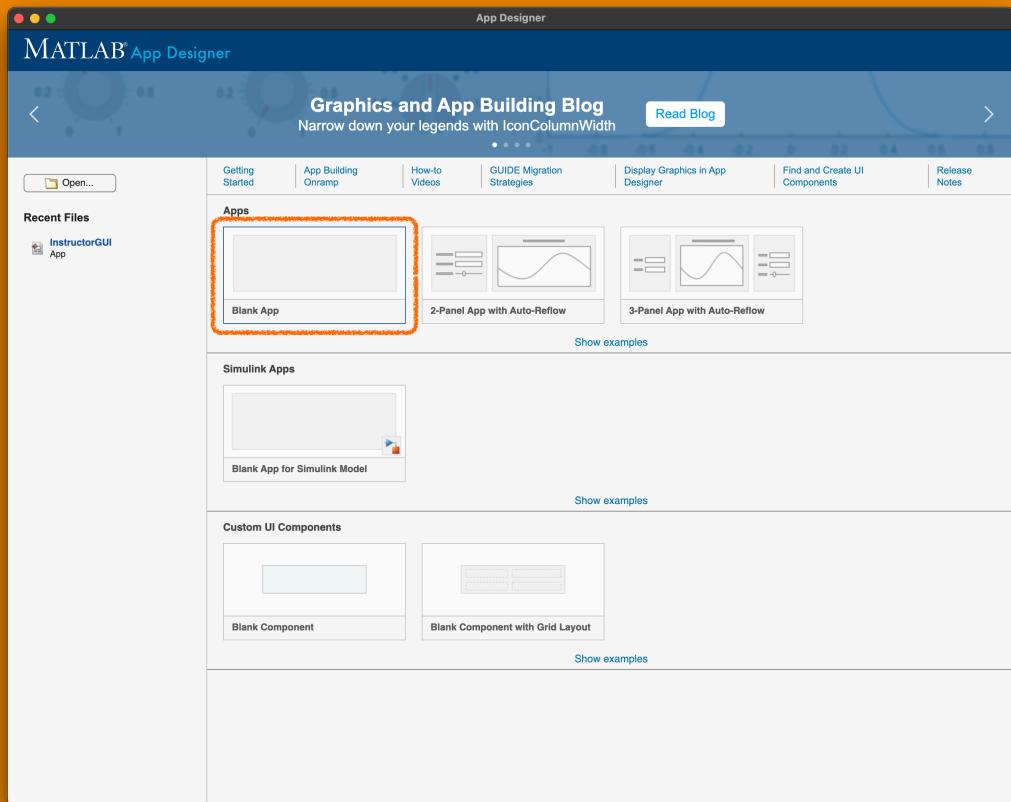
Many functions in an app, require a handle so they know where to apply that function.

Examples:

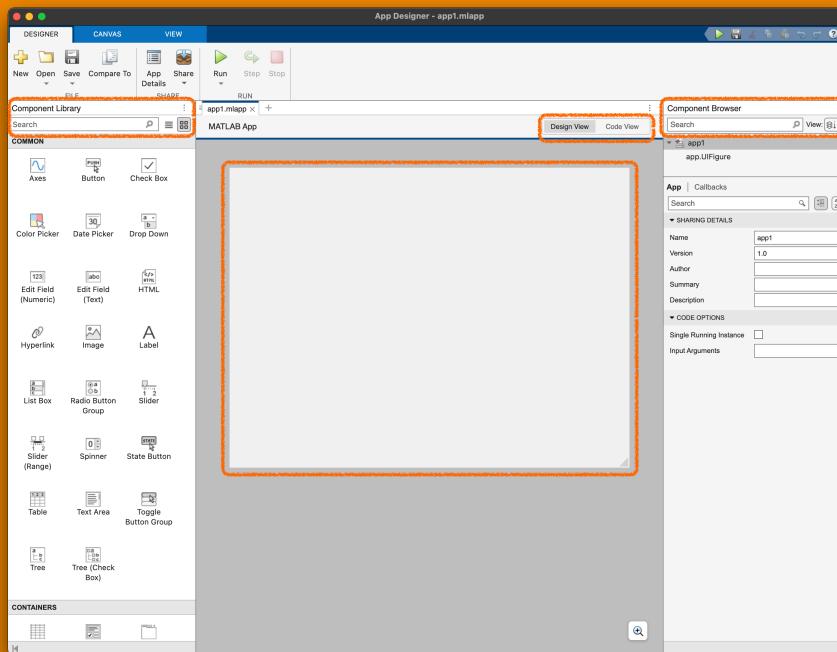
- `plot(app.pltLinear, x, y)`
- `xlabel(app.pltLinear, "text")`

## Step 1: Set Up the App

1. Open App Designer.
2. Select Blank App.

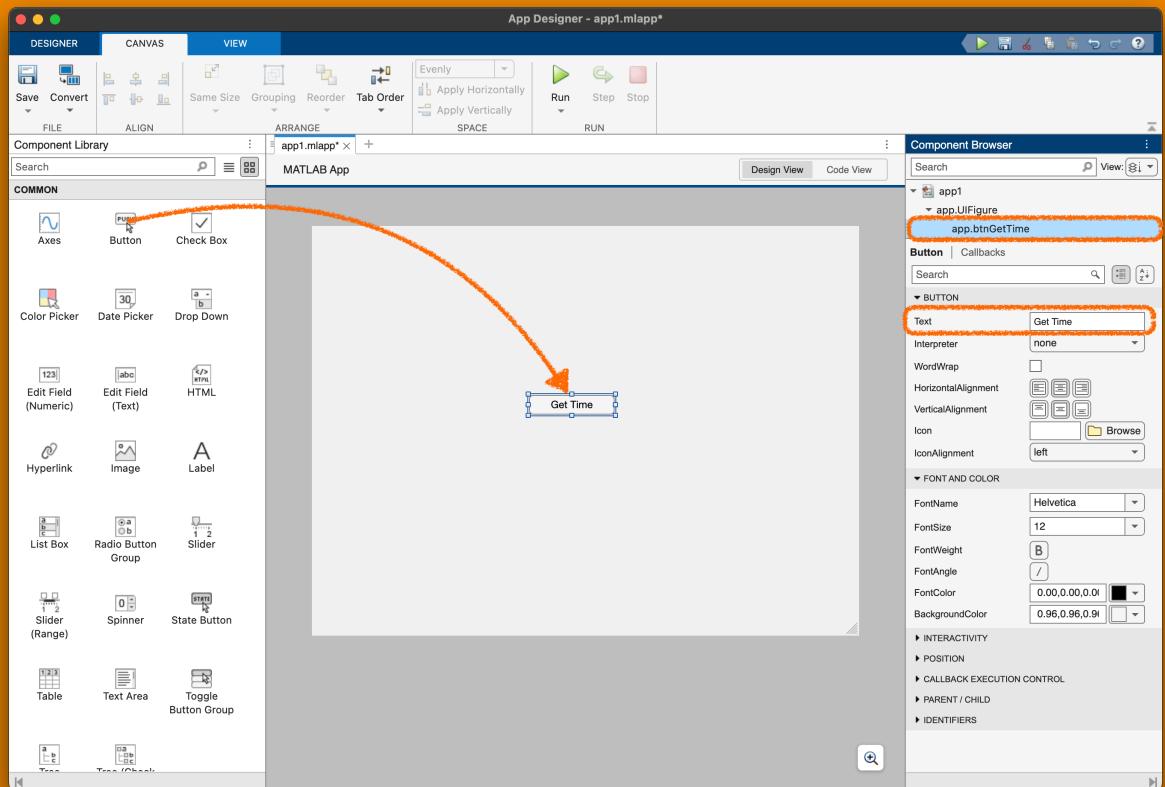


 Have a look around 



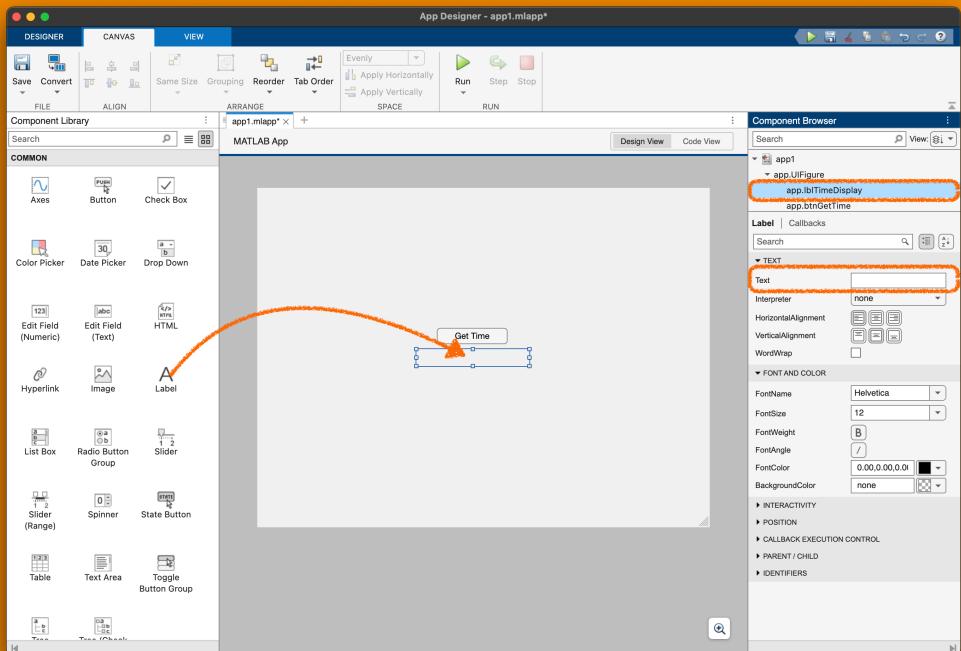
## Step 2: Design the Interface

1. Drag a Button onto the canvas.
2. Customize the button:
  - Text: Get Time
  - Component Name:  
**btnGetTime**



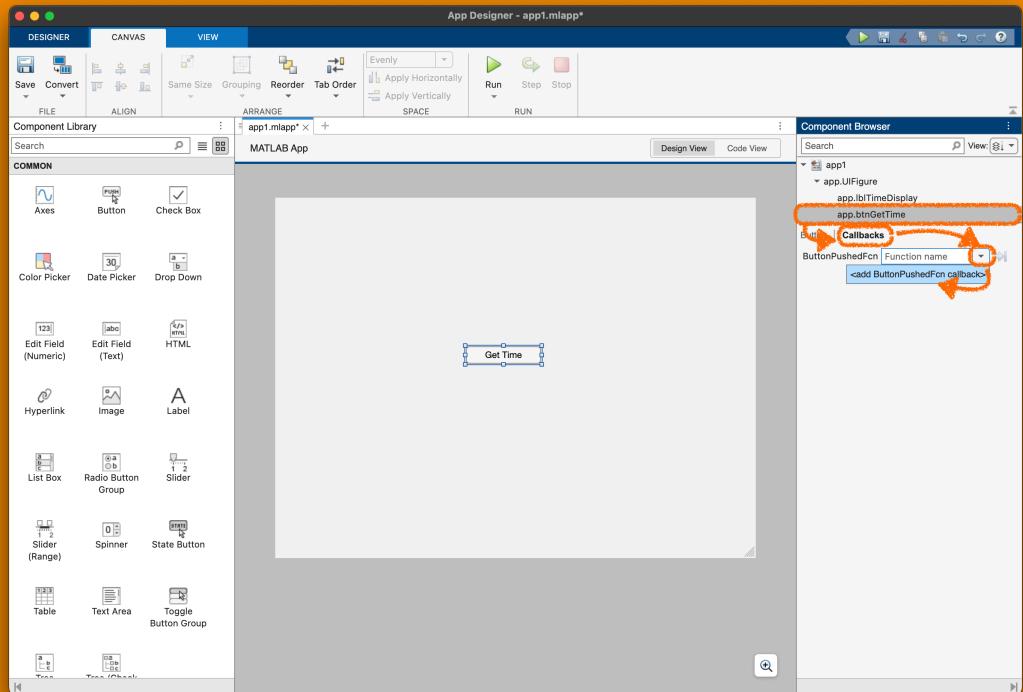
## Step 3: Add a Display Field

1. Drag a Label onto the canvas below the button.
2. Clear the default text (we'll update it programmatically).
3. Component Name:  
**lblTimeDisplay**



## Step 4: Add Callback to Button:

1. Select the button.
2. In the Component Browser, expand Callbacks.
3. Click Add Callback > ButtonPushedFcn.



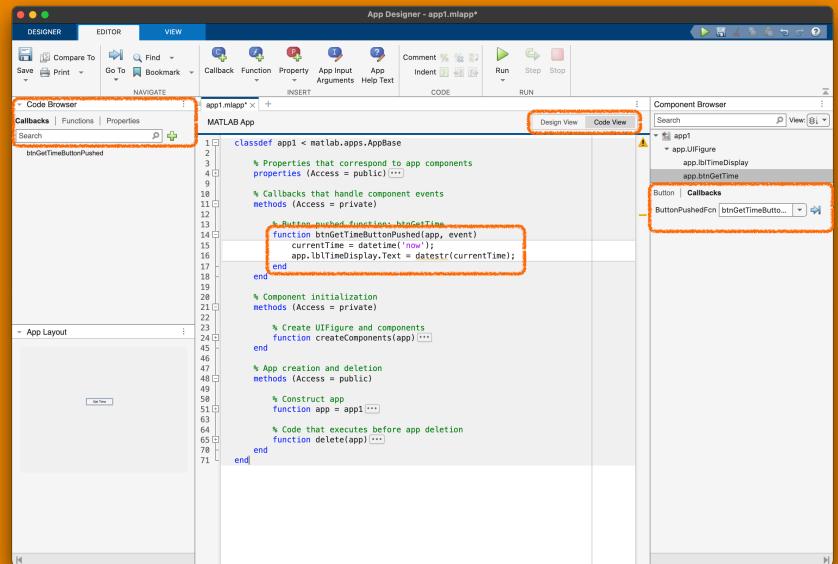
A callback is a function executed in response to a function or in this case; user action.

## Step 5: Implement the Logic

```
% Button pushed function: btnGetTime
function btnGetTimePushed(app, event)
    currentTime = datetime('now');
    app.lblTimeDisplay.Text = datestr(currentTime);
end
```

### Explanation:

- Retrieves the current time.
- Updates the label to display the time.



## Step 6: Run and Test Your App

- Click Run .
- Test the button to see the current time display.



Congratulations! You've built your first app!

# Deep Dive into Callbacks



What Happens When You Click the Button?

- The `btnGetTimePushed` function is called.
- This function defines the app's behavior upon interaction.

## Anatomy of a Callback Function

```
function btnGetTimePushed(app, event)
    % Your code here
end
```

- app: Reference to your app's components.
- event: Data related to the event (e.g., mouse click).

## Best Practices for Callbacks

- Keep them concise.
- Call other functions for complex tasks.
- Use descriptive names.

# Creating Helper Functions

- Use helper functions to organize code.

Example:

```
methods (Access = private)
    function updateTimeDisplay(app)
        currentTime = datetime('now');
        app.lblTimeDisplay.Text = datestr(currentTime);
    end
end
```

Call this function within your callback:

```
function btnGetTimePushed(app, event)
    app.updateTimeDisplay();
end
```

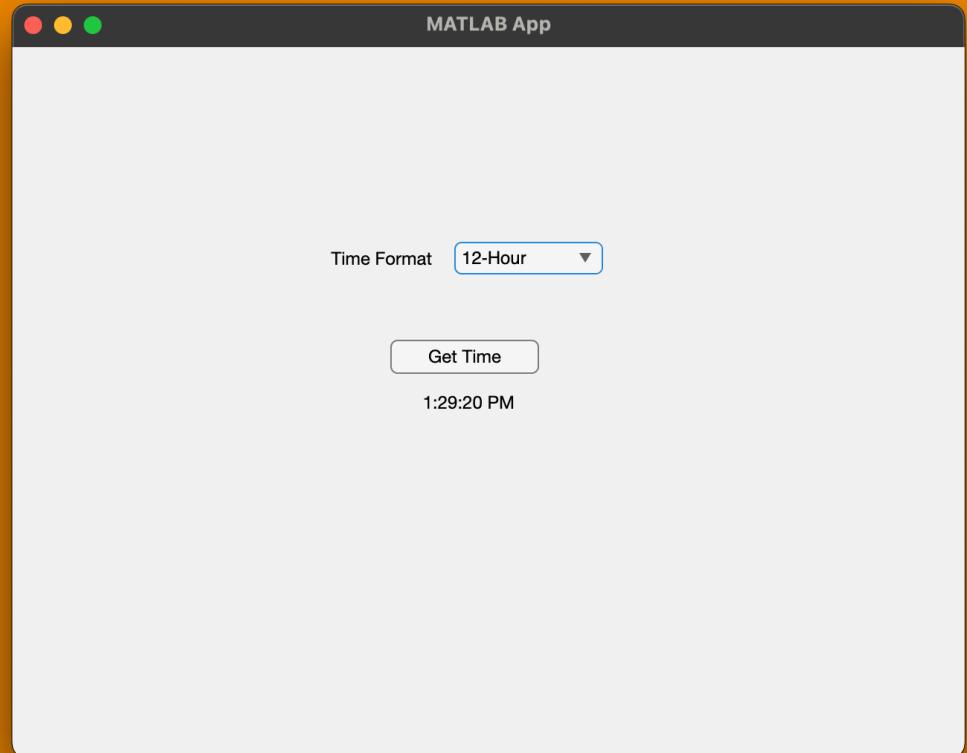
Notes:

- Include **app**
- Can return data like a regular function

# Adding More Features

## Feature 1: Time Format Options

- Add a Drop-Down Menu to select time formats.
  - Options:
    - 12-Hour
    - 24-Hour



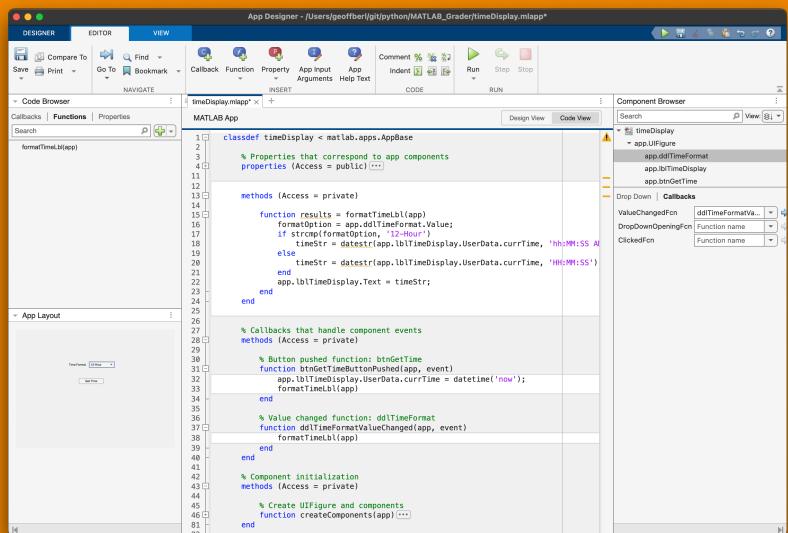
# Update the Callback

```
function btnGetTimePushed(app, event)
    currentTime = datetime('now');
    formatOption = app.ddlTimeFormat.Value;
    if strcmp(formatOption, '12-Hour')
        timeStr = datestr(currentTime, 'hh:MM:SS AM');
    else
        timeStr = datestr(currentTime, 'HH:MM:SS');
    end
    app.lblTimeDisplay.Text = timeStr;
end
```



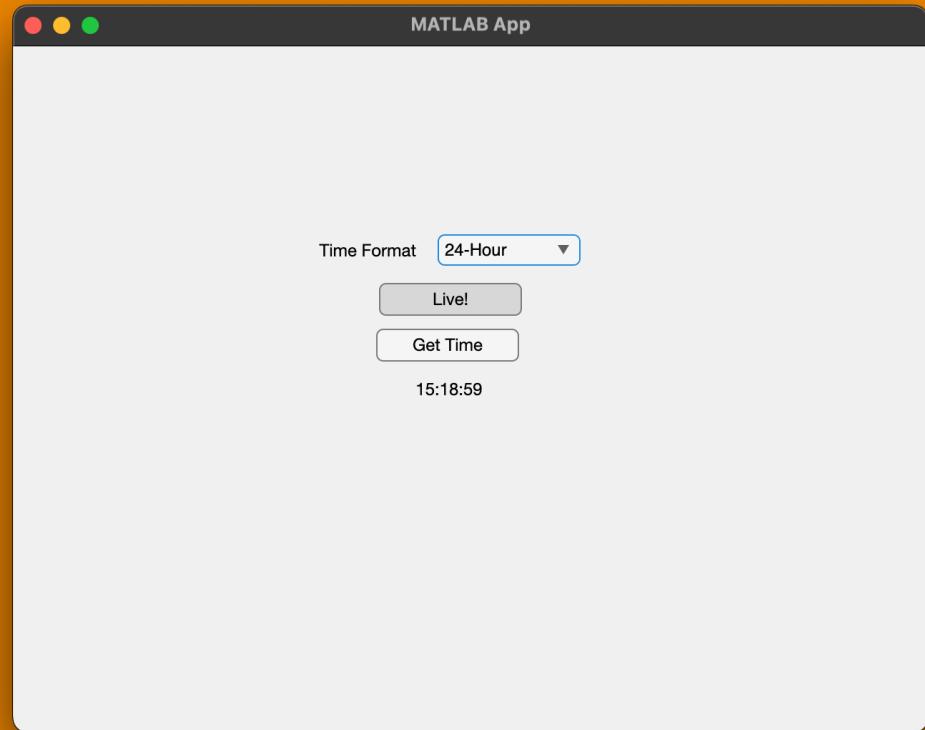
Tip:

- Properties and UserData



## Feature 2: Live Clock

- Add a Switch to toggle a live clock display.
- Use a Timer Object to update the time every second.



# Implementing the Timer

```
properties (Access = private)
    Timer % Timer object for live clock
end

% Value changed function: LiveClockSwitch
function LiveClockSwitchValueChanged(app, event)
    if app.LiveClockSwitch.Value == "On"
        app.Timer = timer('ExecutionMode', 'fixedRate', 'Period', 1, ...
            'TimerFcn', @(~,~) app.updateTimeDisplay());
        start(app.Timer);
    else
        stop(app.Timer);
        delete(app.Timer);
    end
end
```



- Use the debugger to explore and learn
- Clean up the timer (or other resources) when the app closes.

# Final Code

```
1 classdef timeDisplay < matlab.apps.AppBase
2
3     % Properties that correspond to app components
4     properties (Access = public) {}
5
6
7     % Properties (Access = private)
8     properties (Access = private)
9         Timer % Timer object for live clock
10    end
11
12
13
14    methods (Access = private)
15
16        function formatTimeLbl(app)
17            formatOption = app.ddlTimeFormat.Value;
18            if ~isfield(app.lblTimeDisplay.UserData, 'currTime')
19                btnGetTimeButtonPushed(app, []);
20            end
21            if strcmp(formatOption, '12-Hour')
22                timeStr = datestr(app.lblTimeDisplay.UserData.currTime, 'hh:MM:SS AM');
23            elseif strcmp(formatOption, '24-Hour')
24                timeStr = datestr(app.lblTimeDisplay.UserData.currTime, 'HH:MM:SS');
25            else
26                % msgbox("Unhandled selection, please update code")
27                fprintf("Unhandled time format (%s), please update code\n", formatOption)
28                return
29            end
30            app.lblTimeDisplay.Text = timeStr;
31        end
32
33
34        function updateTimeDisplay(app)
35            btnGetTimeButtonPushed(app, [])
36        end
37
38    end
39
40
41
42
43    % Callbacks that handle component events
44    methods (Access = private)
45
46        % Button pushed function: btnGetTime
47        function btnGetTimeButtonPushed(app, event)
48            app.lblTimeDisplay.UserData.currTime = datetime('now');
49            app.formatTimeLbl()
50        end
51
52        % Value changed function: ddlTimeFormat
53        function ddlTimeFormatValueChanged(app, event)
54            app.formatTimeLbl()
55        end
56
57        % Value changed function: btnLiveClock
58        function btnLiveClockValueChanged(app, event)
59            state = app.btnLiveClock.Value;
60            if state
61                app.Timer = timer('ExecutionMode', 'fixedRate', 'Period', 1, ...
62                    'TimerFcn', @(~,~) app.updateTimeDisplay());
63                start(app.Timer);
64            else
65                stop(app.Timer);
66                delete(app.Timer);
67            end
68        end
69
70        % Close request function: UIFigure
71        function UIFigureCloseRequest(app, event)
72            if ~isempty(app.Timer) && isvalid(app.Timer)
73                stop(app.Timer);
74                delete(app.Timer);
75            end
76            delete(app);
77        end
78    end
79
80    % Component initialization
81    methods (Access = private)
82
```

# Packaging and Sharing Your App



## Exporting the App

- Go to Share > Create Standalone Application.
- Follow the prompts to package your app.

## Sharing Options

- MATLAB App Installer (.mlappinstall): Share with other MATLAB users.
- Standalone Desktop App: Runs without MATLAB (requires MATLAB Compiler).

## Tips for Distribution

- Test your app thoroughly before sharing.
- Include user instructions or a help dialog within your app.

# Introduction to Symbolic Math

## Why Use Symbolic Math?

- Perform exact computations.
- Solve algebraic equations analytically.
- Simplify expressions.

# Getting Started

```
syms x y z
```

Declares symbolic variables.

# Differentiation and Integration

Differentiate:

```
f = sin(x) * exp(y);  
diff_f = diff(f, x);
```

Integrate:

```
int_f = int(f, x);
```

Visualize Results:

```
fplot(diff_f, [0, 2*pi])
```

## Solving Equations

```
eqn = x^2 + 5*x + 6 == 0;  
sol = solve(eqn, x);  
disp(sol);
```

Output:

```
sol =  
-2  
-3
```

# Incorporating Symbolic Math into GUIs



Example: Symbolic Differentiation App

## App Features

- Input an expression to differentiate.
- Select the variable to differentiate with respect to.
- Display the symbolic derivative.

## Designing the Interface

- Components:
  - Text Area for input expression.
  - Drop-Down for variable selection.
  - Button to execute differentiation.
  - Text Area to display the result.

## Writing the Callback

```
function btnDerivePushed(app, event)
    expr = app.InputExpressionTextArea.Value;
    var = sym(app.VariableDropDown.Value);
    try
        derivative = diff(sym(expr), var);
        app.ResultTextArea.Value = char(derivative);
    catch
        app.ResultTextArea.Value = 'Invalid expression. Please try again.';
    end
end
```

- Error Handling: Added try-catch block to handle invalid inputs.

## Enhancing User Experience

- Real-Time Updates: Use listeners to update the result as the user types.
- Math Display: Use latex function and display expressions in formatted math.

## Displaying Formatted Math

- Utilize the HTML Component to render LaTeX. Example:

```
latexStr = ['$' latex(derivative) '$'];
app.ResultHTML.Html = ['<html><body>' latexStr '</body></html>'];
```

# Key Takeaways



- App Designer simplifies GUI creation in MATLAB.
- Callbacks are essential for interactive apps.
- Symbolic Math extends MATLAB's capabilities for analytical computations.
- Combining GUIs with symbolic math creates powerful educational tools.

## Next Steps



- Explore More Components: Try sliders, switches, and plots.
- Advanced Features: Look into app packaging and deployment options.
- Join the Community: Share your apps and get feedback.

## Questions?



Feel free to ask any questions or share your thoughts!

## Additional Resources



- MATLAB Documentation: App Designer
- Tutorial Videos: MATLAB App Designer Tutorials
- Community Forums: MATLAB Central

<