

Modelit  
Elisabethdreef 5  
4101 KN Culemborg  
+31(345)531717



info@modelit.nl  
www.modelit.nl

# **Modelit User Interface Components Toolbox for Matlab**

Date June 10, 2008

# ModeliT

Release: 1.00  
Manual: Modelit User Interface Components Toolbox for Matlab.  
Manual version: 1.00 (June 10, 2008)  
Author: Nanne van der Zijpp, Kees-Jan Hoogland  
Copyright: 2008, Modelit  
Contact: [info@modelit.nl](mailto:info@modelit.nl)  
[www.modelit.nl](http://www.modelit.nl)

<a href="#">1 Introduction.....</a>	<a href="#">1</a>
<a href="#">1.1 Introducing the Modelit User Interface ComponentsToolbox for Matlab.....</a>	<a href="#">1</a>
<a href="#">1.2 Background.....</a>	<a href="#">1</a>
<a href="#">1.3 The jacontrol object.....</a>	<a href="#">1</a>
<a href="#">1.4 Getting started with jacontrol objects.....</a>	<a href="#">2</a>
<a href="#">1.5 On-line help.....</a>	<a href="#">3</a>
<a href="#">1.5.1 Online help: Documents.....</a>	<a href="#">3</a>
<a href="#">1.5.2 Online help: command-line help.....</a>	<a href="#">4</a>
<a href="#">1.6 System requirements.....</a>	<a href="#">6</a>
<a href="#">1.7 How to proceed from here.....</a>	<a href="#">6</a>
<a href="#">2 Installation.....</a>	<a href="#">8</a>
<a href="#">3 JaControl Reference Manual.....</a>	<a href="#">9</a>
<a href="#">3.1 General JaControl properties.....</a>	<a href="#">9</a>
<a href="#">3.2 DatePicker.....</a>	<a href="#">11</a>
<a href="#">3.3 HyperLink.....</a>	<a href="#">14</a>
<a href="#">3.4 Label.....</a>	<a href="#">16</a>
<a href="#">3.5 DomEcho.....</a>	<a href="#">18</a>
<a href="#">3.6 Browser.....</a>	<a href="#">19</a>
<a href="#">3.7 Button.....</a>	<a href="#">20</a>
<a href="#">3.8 CheckBox.....</a>	<a href="#">23</a>
<a href="#">3.9 ToggleButton.....</a>	<a href="#">26</a>
<a href="#">3.10 RadioButton.....</a>	<a href="#">29</a>
<a href="#">3.11 ProgressBar.....</a>	<a href="#">32</a>
<a href="#">3.12 Slider.....</a>	<a href="#">34</a>
<a href="#">3.13 TextArea.....</a>	<a href="#">37</a>
<a href="#">3.14 EditorPane.....</a>	<a href="#">39</a>
<a href="#">3.15 PasswordField.....</a>	<a href="#">42</a>
<a href="#">3.16 TextField.....</a>	<a href="#">44</a>
<a href="#">3.17 Countdown.....</a>	<a href="#">46</a>
<a href="#">3.18 PieceBar.....</a>	<a href="#">49</a>
<a href="#">3.19 AutoComboBox.....</a>	<a href="#">51</a>
<a href="#">3.20 Spinner.....</a>	<a href="#">54</a>
<a href="#">3.21 Separator.....</a>	<a href="#">56</a>
<a href="#">3.22 ToolBar.....</a>	<a href="#">58</a>
<a href="#">3.23 JTabbedPane.....</a>	<a href="#">60</a>
<a href="#">3.24 JXTable.....</a>	<a href="#">62</a>
<a href="#">4 Utility functions reference manual.....</a>	<a href="#">67</a>
<a href="#">5 Technical Background of the toolbox.....</a>	<a href="#">68</a>
<a href="#">6 Compilation and deployment .....</a>	<a href="#">69</a>
<a href="#">6.1 Deployment.....</a>	<a href="#">69</a>
<a href="#">6.2 Compilation.....</a>	<a href="#">69</a>



# 1 Introduction

## 1.1 Introducing the Modelit User Interface Components Toolbox for Matlab

The Modelit User Interface Components Toolbox for Matlab (in short: User Interface Components Toolbox) has been designed for Matlab developers who want to provide their Matlab applications with intuitive and powerful user-interfaces. Building such interfaces is greatly helped by the availability of User Interface Components such as tabbed panes, tables, trees and so forth which are not part of standard Matlab. The Modelit User Interface Components Toolbox for Matlab offers easy access to these extra User Interface Components for Matlab programmers, without the need to switch to a different programming language or complex code.

## 1.2 Background

Matlab offers a number of components such as buttons, sliders and edit boxes to build graphical user interfaces. However, more complex components such as tables or trees are not directly available. Fortunately Matlab has built-in Java support so Java objects can be created and accessed by using Matlab commands. Graphical User Interfaces can be greatly enhanced by using for example components from the Java swing library. Using Java components however requires of course knowledge of Java.

The Modelit User Interface Components Toolbox for Matlab extends the number of available components for GUI building by using java. This is achieved by defining a new object, the `jacontrol`, which is similar to the Matlab `uicontrol` and which acts as an interface between Matlab and Java, effectively hiding the Java implementation from the user. As a result common Matlab users can use these new components without having any knowledge of Java.

The Modelit User Interface Components Toolbox for Matlab extends the standard Matlab components with complex components such as sortable, editable and filterable tables, treetables and trees, editable comboboxes with autocompletion and webbrowsers. The Modelit User Interface Components Toolbox for Matlab makes use of the swingx project (<http://swinglabs.org>) and the glazed lists project (<http://publicobject.com/glazedlists>).

## 1.3 The `jacontrol` object

Central to the User Interface Components Toolbox is the `jacontrol` object. A `jacontrol` (Java-control) object is initialized and modified in a similar way as a Matlab `uicontrol` (user-interface-control) object. For example the position of a `jacontrol` objects is controlled by the "position" and "units" properties. Like the `uicontrol` object the `jacontrol` object has a "style" property that determines the appearance and capabilities see the table below for a summary or chapter 3 for a full reference.

<b>Style</b>	<b>Description</b>
<code>JXDatePicker</code>	Calendar
<code>HyperLink</code>	Hyperlink
<code>JLabel</code>	Label or iconified label
<code>DomEcho</code>	Interactive tree representation of XML data
<code>Browser</code>	Internet browser supporting dynamic content
<code>JButton</code>	Pushbutton
<code>JCheckBox</code>	Box that can be checked or unchecked
<code>JToggleButton</code>	Pushbutton that shows its state
<code>JRadioButton</code>	Rounded box that can be checked or unchecked
<code>JProgressBar</code>	Bar indicating some progress
<code>JSlider</code>	Slider
<code>JTextArea</code>	Multiline area to display and edit text.
<code>JEditorPane</code>	Multiline area to display and edit formatted text.

<i>JPasswordField</i>	Editable textfield which hides input
<i>JTextField</i>	Editable textfield
<i>CountDown</i>	A visual timer that triggers a callback
<i>PieceBar</i>	An advanced progressbar that can display a series of colored rectangular patches
<i>AutoComboBox</i>	An editable combobox with autocompletion support
<i>Spinner</i>	A single line input field that lets the user select a value from an ordered sequence.
<i>JSeparator</i>	A selectable visual divider
<i>JToolBar</i>	Toolbar for jacontrol objects
<i>JTabbedPane</i>	A container for jacontrol objects which can be selected through corresponding tabs
<i>JXtable</i>	A sortable and filterable table

### 1.4 Getting started with jacontrol objects

Creating a jacontrol is similar to creating a uicontrol.

#### General calling syntax

handle = jacontrol(hParent, 'PropertyName',PropertyValue,...)

handle = jacontrol('PropertyName',PropertyValue,...)

handle = jacontrol(PropertyStruct,...)

#### Example

```
hfig = figure;
handle = jacontrol(hfig,...
    'style', 'JXDatePicker',...
    'value', now,...
    'dateform', 'dd-MM-yyyy',...
    'position', [10 220 200 20]);
```

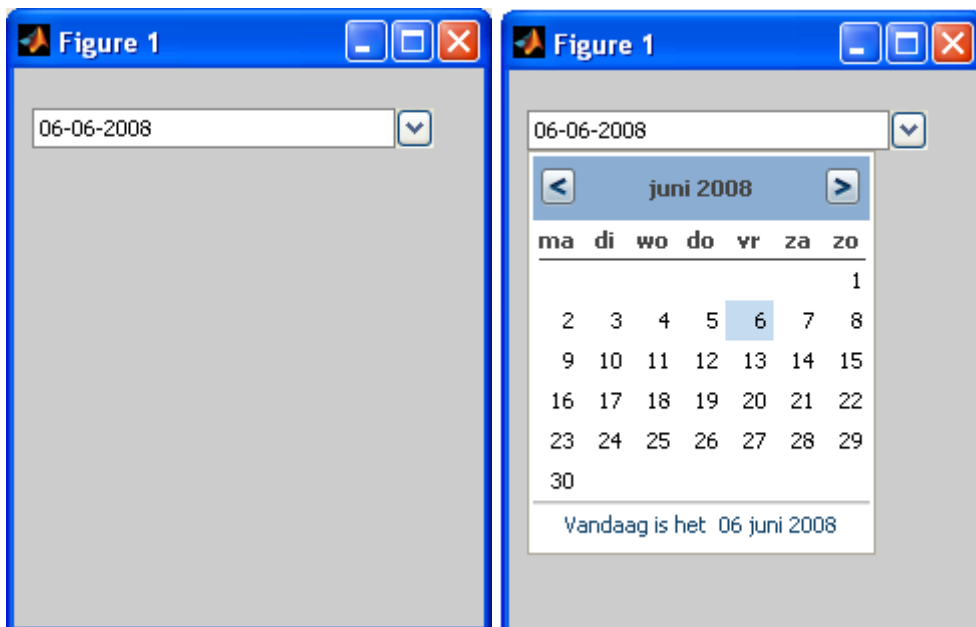


Figure 1: jacontrol object of style "JXDatePicker"

Jacontrol properties can be queried and altered after the object has been created by using the "get" and "set" commands. Property names and values are case insensitive and do not need to be fully specified as long as the name can be unambiguously determined.

*Example*

```
set(handle,'callback','disp(datestr(get(handle,"value"))')  
get(handle,'callback')  
get(handle,'call')
```

## 1.5 On-line help

The following on-line help is available:

- Documents
  - HTML documentation
    - Examples
    - Reference manual
  - PDF documentation
    - User guide and reference manual (this document)
- Command-line help
  - Present feasible style-property pairs in tree
  - Property inspector
  - Retrieve help information

### 1.5.1 Online help: Documents

*HTML documentation*

A set of HTML documentation is provided. This documentation contains additional examples for each jacontrol style.

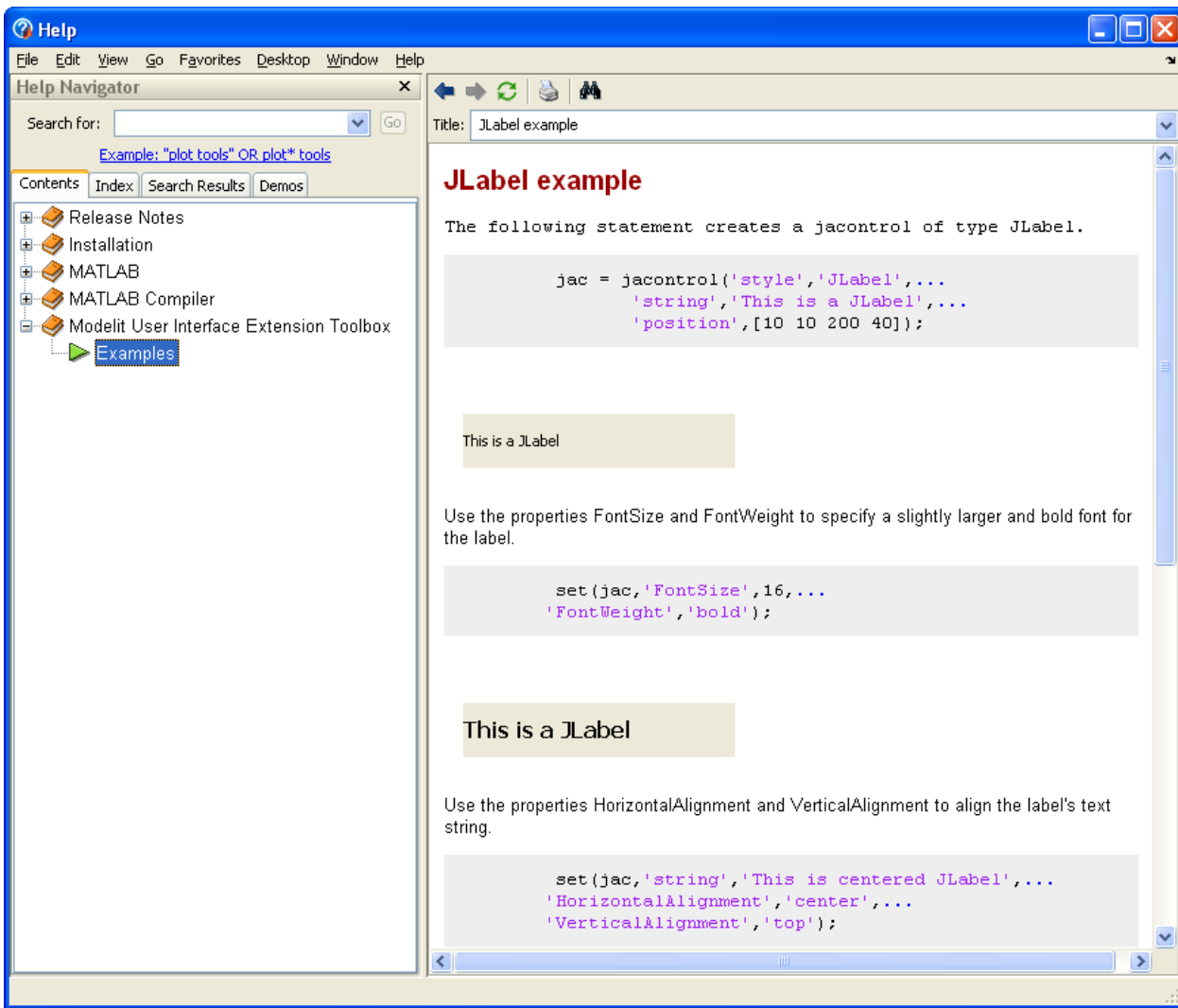


Figure 2: A set of examples is provided to illustrate how each jacontrol object style can be used.

**PDF documentation**

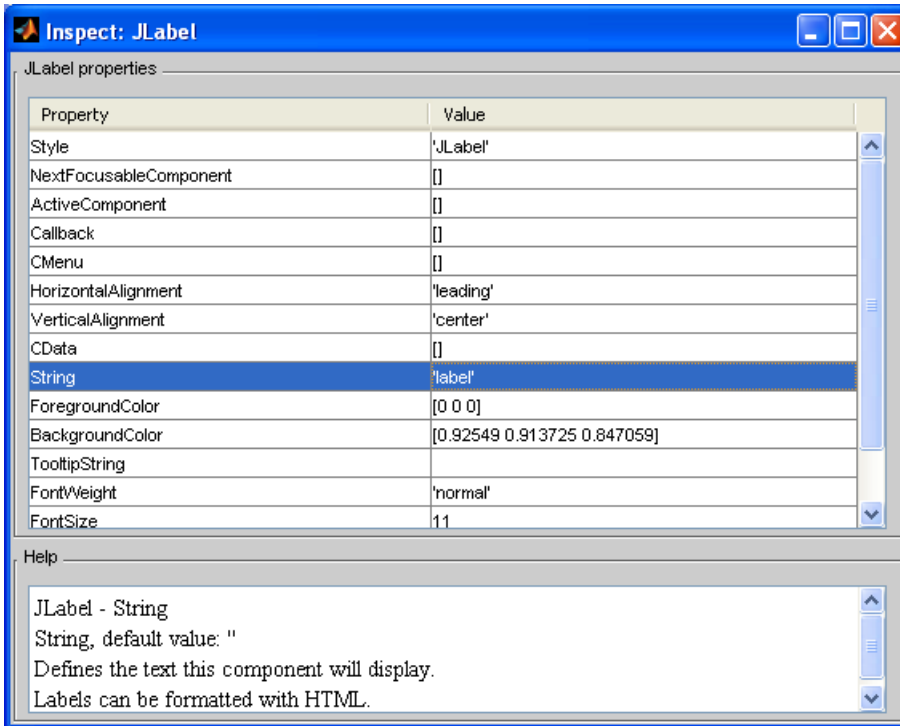
The current document provides the user guide and reference manual and is provided as a PDF document

**1.5.2 Online help: command-line help**

The on-line tools can be used to quickly find out the available styles, the available properties for each style, or documentation on a specific property.

Command line:	Result:
<code>jacontrol</code>	Shows all possible styles and properties in an interactive tree (see Figure 3). When a property is selected the "Help" panel shows information for that property.
<code>jac = jacontrol('style','jlabel',... 'str','label' ); inspect(jac)</code>	Shows all properties and values of a specific jacontrol object (see Figure 4). The "Help" panel shows help information for selected properties.
<code>jac = jacontrol('style','jlabel',... 'string','label' ); help(jac)</code>	Presents the selection of the reference manual that corresponds to the style of the current jacontrol object in a table. One may copy and paste rows of this table in a word document.





**Figure 4:** Type "inspect(h)" (where h is a jacontrol handle) on the Matlab command prompt to display a tree that display the properties and values for a specific jacontrol object. Click on any property to display available help information.

## 1.6 System requirements

The User Interface Components Toolbox has been tested with Matlab R2006b, R2007a and R2007b. To create standalone applications the toolbox can be used together with Matlab builder products such as the Matlab compiler and the Matlab Builder for Java. Compatibility with Matlab version 2008a is currently under review.

The Modelit User Interface Components Toolbox works seamlessly with other toolboxes and subroutine libraries provided by Modelit. Two toolboxes are particularly useful when building user interfaces:

- *Modelit Layout Manager*. This toolbox supports the design of modular and resizable GUI's. The toolbox allows specifying normalized and absolute sizes of GUI elements, positioning them relative to frames. It also automates computation of frame sizes recursively;
- *Modelit Application Framework for Matlab*. This toolbox implements among others an automated link between datamodel and visualization and undo/redo functionality.

Users who combine the UIC-toolbox with other Modelit products benefit from extra synergies. For example : a separator is part of the UIC-toolbox and is a visual divisor between two areas in a figure. When in addition to the UIC-toolbox the layout manager is available, the position of the divisor can be made "draggable" just by adding 1 line of code. This allows users to change the layout of their interface interactively.

## 1.7 How to proceed from here

Depending on your needs you may read the following chapters:

- Chapter 2 contains all information for installing the toolbox;
- Chapter 3 contains the jacontrol reference manual. For each style, the reference manual describes the available properties and their use;
- Chapter 4 describes jacontrol utility functions. This section describes functions that are needed, but for some reason can not be implemented as a method of the jacontrol object;

- Chapter 5 provides technical backgrounds on the toolbox. It should not be a problem to apply the toolbox without having read this chapter. However, a basic understanding of the technical background of the toolbox will allow more effective troubleshooting and will help to exploiting the toolbox to the full.
- Chapter 6 describes which steps one should take to compile applications that include the User Interface Components Toolbox to a standalone application and how to deploy these applications.

## 2 Installation

Please follow the next steps to automatically install the Modelit Java toolbox for Matlab:

1. Unzip the files from the JavaToolbox.zip file.  
This creates a folder 'java\_toolbox'.
2. Run install.m from the 'java\_toolbox' folder. This adds the path with the Modelit Java toolbox for Matlab and the online help to the matlab path. It also adds the java archive modelit.jar to the static javapath.
3. Restart Matlab.

The directory structure of the Modelit Java toolbox for Matlab is shown in Figure 5 and contains the following directories:

- **java**  
This directory contains the following files:
  - **modelit.jar**  
Contains the extra java classes which are used with the toolbox. It also contains the jdic library (see <http://swinglabs.org>), the swingx library (see <http://swinglabs.org>) and the glazed lists library (see <http://publicobject.com/glazedlists>).
  - **MozEmbed.exe**  
Native Mozilla Firefox browser embedding binary for the jdic library.
  - **leEmbed.exe**  
Native Internet Explorer embedding binary for the jdic library.
  - **jdic.dll**  
*Native library file for the jdic library.*
  - **tray.dll**  
*Native library file for the jdic library.*
- **jacontrol**  
Helper files for the jacontrol-object.
- **@jacontrol (the jacontrol-object)**  
The files in this directory are the methods of the jacontrol-object.
- **@jacontrol /private**  
The files in this directory are the private methods of the jacontrol-object
- **html**  
This directory contains the online help and examples for the Modelit User Interface Components Toolbox for Matlab. This help can be viewed from the Matlab help browser.



Figure 5: Directory structure of the Modelit User Interface Components Toolbox for Matlab.

## 3 JaControl Reference Manual

In this section the properties of the jacontrol objects are listed and described. A jacontrol object can be created by specifying the values for one or more properties. After creating the jacontrol object these properties can be set and queried by using the jacontrol's set and get commands. Property names and values are case insensitive and do not need to be fully specified as long as the name can be unambiguously determined.

The jacontrol properties can be divided into two classes, properties which are common to all the jacontrol objects (see section 3.1) and properties that depend on the style (e.g. Spinner or HyperLink) of the jacontrol (see section 3.2 and further). Properties are not case sensitive and partial names, if not ambiguous, are allowed.

### 3.1 General JaControl properties

#### MatlabHandle

Handle

Handle of the Matlab container in which the java object resides.

#### JavaHandle

Java object

The actual java object.

#### HandleVisibility

{on} | callback | off

Control access to object's handle. This property determines when an object's handle is visible in its parent's list of children. When a handle is not visible in its parent's list of children, it is not returned by functions that obtain handles by searching the object hierarchy or querying handle properties. Handles that are hidden are still valid. If you know an object's handle, you can set and get its properties, and pass it to any function that operates on handles.

Handles are always visible when HandleVisibility is on.

Setting HandleVisibility to callback causes handles to be visible from within callback routines or functions invoked by callback routines, but not from within functions invoked from the command line. This provides a means to protect GUIs from command-line users, while allowing callback routines to have complete access to object handles.

Setting HandleVisibility to off makes handles invisible at all times. This may be necessary when a callback routine invokes a function that might potentially damage the GUI (such as evaluating a user-typed string), and so temporarily hides its own handles during the execution of that function.

You can set the root ShowHiddenHandles property to on to make all handles visible, regardless of their HandleVisibility settings. This does not affect the values of the HandleVisibility properties.

#### Parent

Handle, default value: gcf

Jacontrol parent. The handle of the jacontrol's parent object. You can move a jacontrol object to another figure, uipanel, or uibuttongroup by setting this property to the handle of the new parent.

#### Position

1x4 vector, default value is system dependent

Size and location of the jacontrol. The rectangle defined by this property specifies the size and location of the control within the parent. Specify Position as [left bottom width height]

left and bottom are the distance from the lower-left corner of the parent object to the lower-left corner of the jacontrol object. width and height are the dimensions of the jacontrol rectangle. All measurements are in units specified by the Units property.

### ResizeFcn

String, cell array or function handle, default value: []

Resize callback routine. Matlab executes this callback routine whenever a user resizes the jacontrol. You can query the jacontrol Position property to determine its new size and position.

### Tag

String, default value: ""

User-specified object label. The Tag property provides a means to identify graphics objects with a user-specified label. This is particularly useful when constructing interactive graphics programs that would otherwise need to define object handles as global variables or pass them as arguments between callback routines. You can define Tag as any string. Use the function gcjh to find a jacontrol with a specified tag.

### Units

{pixels} | normalized | inches | centimeters | points | characters

Units of measurement. Matlab uses these units to interpret the Extent and Position properties. All units are measured from the lower-left corner of the parent object.

Normalized units map the lower-left corner of the parent object to (0,0) and the upper-right corner to (1.0,1.0). pixels, inches, centimeters, and points are absolute units (1 point = 1/72 inch). Character units are characters using the default system font; the width of one character is the width of the letter x, the height of one character is the distance between the baselines of two lines of text.

If you change the value of Units, it is good practice to return it to its default value after completing your computation so as not to affect other functions that assume Units is set to the default value.

### Visible

{on} | off

Jacontrol visibility. By default, all jacontrols are visible. When set to off, the jacontrol is not visible, but still exists and you can query and set its properties.

### Userdata

Matrix, default value: []

User-specified data. Any data you want to associate with the uicontrol object. Matlab does not use this data, but you can access it using set and get.

## 3.2 DatePicker

### Summary

The DatePicker is a component that combines a button, an editable field and a component that displays a monthly calendar. The user can select a date from the calendar component, which appears when the button is pressed. The selection from the calendar component will be displayed in an editable field in customizable date format. Values may also be modified manually by entering a date into the editable field using one of the supported date formats.

### Example

```
jac = jacontrol('style','JXDatepicker',...
    'value','now',...
    'dateformat','dd-MM-yyyy',...
    'position',[10 10 200 20]);
```



### Properties

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user changes the date in the datepicker. The date can be changed by entering the date explicitly in the text field, or by selecting a date in the calendar component.

#### MouseEnteredCallback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the datepicker.

#### MouseExitedCallback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the datepicker.

#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JFormattedTextField

The component that can receive focus.

**CMenu**

Handle, default value: []

The handle of the context menu associated with the datepicker.  
 The context menu is displayed whenever you right-click over the datepicker.  
 Use the Matlab `uicontextmenu` function to create the context menu.

**Value**

Datenum, default value: now

The current date in the datepicker as a Matlab serial datenum.  
 For more information see Matlab's `datenum` function.

**TooltipString**

String, default value: "

The `TooltipString` property specifies the text of the tooltip associated with the datepicker.  
 When the user moves the mouse pointer over the datepicker and leaves it there, the tooltip is displayed.  
 HTML can be used to format the tooltipstring's text.

**DateFormat**

Cellstring, default value: {}

Specifies the allowed input formats.  
 The first item of the cellstring specifies the display format.  
 Available format characters:

Character	Description	Format	Example
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
W	Week in month	Number	2
Z	Time zone	RFC 822 time zone	-0800
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the datepicker's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the datepicker's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the text in the datepicker.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the datepicker's font, when it is available on your system.

### 3.3 HyperLink

#### Summary

A component that displays a hyperlink. A callback is triggered when the user clicks on the hyperlink.

#### Example

```
jac = jacontrol('style', 'HyperLink',...
    'string', 'www.modelit.nl',...
    'FontSize', 16,...
    'FontWeight', 'bold',...
    'position', [10 10 200 20]);
```



#### Properties

##### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

##### ActiveComponent (Read only)

Java object of type org.jdesktop.swingx.JXHyperLink

The component that can receive focus.

##### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user clicks on the hyperlink.

##### String

String, default value: ""

The text displayed as an hyperlink. HTML can be used to format the hyperlink's text.

##### HorizontalAlignment

{leading} | trailing | left | right | center

The alignment of the hyperlink contents along the X axis.

##### VerticalAlignment

{center} | bottom | top

The alignment of the hyperlink contents along the Y axis.

##### TooltipString

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the hyperlink. When the user moves the mouse pointer over the hyperlink and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

##### BackgroundColor

RGB triple, default value is system dependent

The background color of the hyperlink.

**ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the hyperlink.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the hyperlink's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the hyperlink's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the hyperlink.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the hyperlink's font, when it is available on your system.


### 3.4 Label

#### Summary

A Label is a display area for a text string or an image, or both. The label's contents can be aligned by setting the vertical and horizontal alignment. HTML can be used to format the label's text.

#### Example

```
jac = jacontrol('style', 'JLabel',...
    'string', 'This is a JLabel',...
    'FontSize', 16,...
    'FontWeight', 'bold',...
    'position', [10 10 200 20]);
```



#### NextFocusableComponent (Not used)

default value: []

A label cannot receive focus.

#### ActiveComponent (Read only)

default value: []

A label cannot receive focus.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user clicks on the label.  
A label can have either a callback or a contextmenu attached to it, not both.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the label.  
The context menu is displayed whenever you right-click or left-click over the label.  
Use the Matlab `uicontextmenu` function to create the context menu.  
A label can have either a callback or a contextmenu attached to it, not both.

#### HorizontalAlignment

{leading} | trailing | left | right | center

The alignment of the label's contents along the X axis.

#### VerticalAlignment

{center} | bottom | top

The alignment of the label's contents along the Y axis.

#### CData

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the label.  
Each value must be an RGB value between 0.0 and 1.0.  
CData(:, :, 1) defines the red components of the pixels  
CData(:, :, 2) defines the green components of the pixels

CData(:, :, 3) defines the blue components of the pixels  
Transparent pixels are marked with NaN's

### **String**

String, default value: ""

Defines the text this component will display.  
Labels can be formatted with HTML.

### **ForegroundColor**

RGB triple, default value is system dependent

The color of the label's text.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the label.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the label.  
When the user moves the mouse pointer over the label and leaves it there, the tooltip is displayed. HTML  
can be used to format the tooltipstring's text.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the label's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the label's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the label.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

### **FontAngle**

{normal} | italic

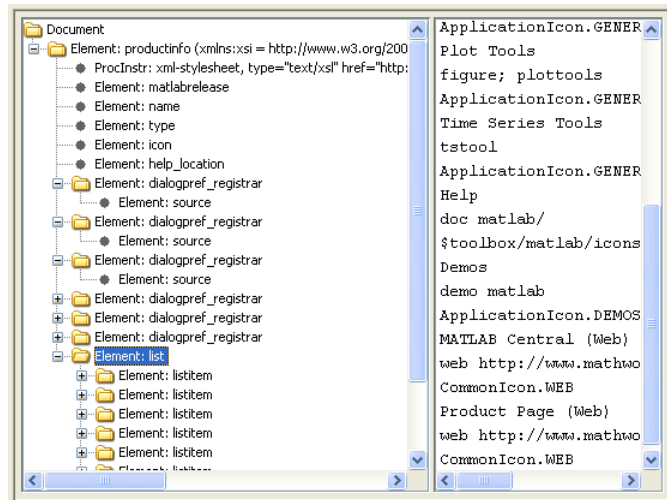
Setting this property to italic selects a slanted version of the label's font, when it is available on your system.

### 3.5 DomEcho

DomEcho is a component that displays the (D)ocument (O)bject (M)odel of an xml-object in a tree. The user can inspect the xml structure and contents by browsing through its tree representation.

#### Example

```
xDoc = xmlread(fullfile(matlabroot,...
'toolbox/matlab/general/info.xml'));
jac = jacontrol('style','DomEcho',...
'document',xDoc,...
'position',[0 0 560 420]);
```



#### NextFocusableComponent (Not used)

default value: []

The DomEcho component cannot receive focus.

#### ActiveComponent (Read only)

default value: []

The DomEcho component cannot receive focus.

#### Document

org.apache.xerces.dom.DeferredDocumentImpl object  
Modelit xml-object, default value: []

The xml which is displayed as a D(ocument) O(bject) M(odel) in the tree.

An org.apache.xerces.dom.DeferredDocumentImpl object can be obtained with xmlread.

### 3.6 Browser

The Browser is a component that enables Matlab applications to access web browsing capabilities, provided by full-featured system browsers (Internet Explorer or Mozilla on Windows, and Mozilla on Linux/Solaris). Besides static HTML content the browser also supports dynamic content such as JavaScript, Netscape plug-ins, ActiveX, Java Plug-ins.

#### Example

```
jac = jacontrol('style','Browser',...
    'position',[0 0 560 420],...
    'URL','http://www.modelit.nl');
```



#### NextFocusableComponent (Not used)

default value: []

Browsers cannot receive focus.

#### ActiveComponent (Read only)

default value: []

Browsers cannot receive focus.

#### Url

String, default value: ""

Address of the current page displayed in the browser.

#### Content

String, default value: ""

The current contents of the browser, can be set directly or indirectly by setting the browser's url.

### 3.7 Button

An ordinary push button with a text string or an image, or both. To activate a push button, click the mouse button on the push button and a callback will be generated.

#### Example

```
jac = jacontrol('style','JButton',...
               'string','JButton',...
               'position',[10 10 200 20]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JButton

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the button.

The context menu is displayed whenever you right-click over the button.

Use the Matlab `uicontextmenu` function to create the context menu.

#### Mnemonic

Character, default value: ""

The mnemonic is the key which when combined with the look and feel's mouseless modifier (usually Alt) will activate this button. Mnemonics are case-insensitive. If the character defined by the mnemonic is found within the button's label string, the first occurrence of it will be underlined to indicate the mnemonic to the user.

#### CData

Matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the button.

Each value must be an RGB value between 0.0 and 1.0.

`CData(:, :, 1)` defines the red components of the pixels

`CData(:, :, 2)` defines the green components of the pixels

`CData(:, :, 3)` defines the blue components of the pixels

Transparent pixels are marked with NaN's

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user clicks on the button.

#### MouseEnteredCallback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the button.

### **MouseExitedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the button.

### **String**

String, default value: ""

The text displayed on the button.  
HTML can be used to format the text.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the button. When the user moves the mouse pointer over the button and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the button.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the button.

### **HorizontalAlignment**

{leading} | trailing | left | right | center

The alignment of the button's contents along the X axis.

### **VerticalAlignment**

{center} | bottom | top

The alignment of the button's contents along the Y axis.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the button's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the button's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the button.

## **FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the button's font, when it is available on your system.

### 3.8 CheckBox

An item with a text string or an image, or both, that can be selected or deselected. Checkboxes are intended for providing the user with a number of independent choices. Check boxes trigger a callback when selected and indicate their state on the display. To activate a checkbox, click the mouse button on the object.

#### Example

```
jac = jacontrol('style','JCheckBox',...
               'string','JCheckBox',...
               'position',[10 10 200 20]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JCheckBox

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the checkbox. The context menu is displayed whenever you right-click over the checkbox. Use the Matlab `uicontextmenu` function to create the context menu.

#### Mnemonic

Character, default value: ""

The mnemonic is the key which when combined with the look and feel's mouseless modifier (usually Alt) will activate this checkbox.

Mnemonics are case-insensitive. If the character defined by the mnemonic is found within the checkbox's label string, the first occurrence of it will be underlined to indicate the mnemonic to the user.

#### UnselectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the checkbox if its value is false.

Each value must be an RGB value between 0.0 and 1.0.

`CData(:,1)` defines the red components of the pixels

`CData(:,2)` defines the green components of the pixels

`CData(:,3)` defines the blue components of the pixels

Transparent pixels are marked with NaN's

#### SelectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the checkbox if its value is true.

Each value must be an RGB value between 0.0 and 1.0.

CData(:, :, 1) defines the red components of the pixels  
CData(:, :, 2) defines the green components of the pixels  
CData(:, :, 3) defines the blue components of the pixels  
Transparent pixels are marked with NaN's

### **Callback**

Function handle, default value: []

Specifies the function that should be executed when the user selects or deselects the checkbox.

### **MouseEnteredCallback**

Function handle, default value: []

Specifies the function that should be executed when the mouse enters the checkbox.

### **MouseExitedCallback**

Function handle, default value: []

Specifies the function that should be executed when the mouse exits the checkbox.

### **String**

String, default value: ""

The text displayed with the checkbox. If the component is too small to accommodate the specified text the string is truncated with an ellipsis. HTML can be used to format the text.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the checkbox. When the user moves the mouse pointer over the button and leaves it there, the tooltip is displayed. HTML can be used to format the text.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the checkbox.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the checkbox.

### **Value**

Boolean, default value: False

False if the checkbox is not selected, True otherwise.  
If the checkbox is placed in a toolbar the checkbox's state is given as 'On' if the checkbox is selected, 'Off' otherwise.

### **HorizontalAlignment**

{leading} | trailing | left | right | center

The alignment of the checkbox's contents along the X axis.

### **VerticalAlignment**

{center} | bottom | top

The alignment of the checkbox's contents along the Y axis.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the checkbox's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the checkbox's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the checkbox's text string.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

### **FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the checkbox's font, when it is available on your system.

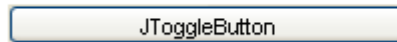
### 3.9 ToggleButton

A button with a text string or an image, or both, that can be selected or deselected.

Toggle buttons are controls that execute callbacks when clicked on and indicate their state, either on or off. To activate or deactivate a push button, click the mouse button on the toggle button.

#### Example

```
jac = jacontrol('style','JToggleButton',...
               'string','JToggleButton',...
               'position',[10 10 200 20]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JToggleButton.

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the togglebutton.

The context menu is displayed whenever you right-click over the togglebutton.

Use the Matlab uicontextmenu function to create the context menu.

#### Mnemonic

Character, default value: "

The mnemonic is the key which when combined with the look and feel's mouseless modifier (usually Alt) will activate this togglebutton. Mnemonics are case-insensitive. If the character defined by the mnemonic is found within the togglebutton's label string, the first occurrence of it will be underlined to indicate the mnemonic to the user.

#### UnselectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the togglebutton if its value is false. Each value must be an RGB value between 0.0 and 1.0.

CData(:, :, 1) defines the red components of the pixels

CData(:, :, 2) defines the green components of the pixels

CData(:, :, 3) defines the blue components of the pixels

Transparent pixels are marked with NaN's

#### SelectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the togglebutton if its value is true.

Each value must be an RGB value between 0.0 and 1.0.

CData(:, :, 1) defines the red components of the pixels

CData(:,2) defines the green components of the pixels  
CData(:,3) defines the blue components of the pixels  
Transparent pixels are marked with NaN's

### **Callback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user selects or deselects the togglebutton.

### **MouseEnteredCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the togglebutton.

### **MouseExitedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the togglebutton.

### **String**

String, default value: ""

The text displayed on the togglebutton.  
HTML can be used to format the text.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the togglebutton.  
When the user moves the mouse pointer over the togglebutton and leaves it there, the tooltip is displayed.  
HTML can be used to format the tooltipstring's text.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the togglebutton.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the togglebutton.

### **Value**

Boolean, default value: False

False if the togglebutton is not selected, True otherwise.  
If the togglebutton is placed in a toolbar the togglebutton's state is given as 'On' if the togglebutton is selected, 'Off' otherwise.

### **HorizontalAlignment**

{leading} | trailing | left | right | center

The alignment of the togglebutton's contents along the X axis.

### **VerticalAlignment**

{center} | bottom | top

The alignment of the togglebutton's contents along the Y axis.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the togglebutton's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the togglebutton's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the string in the togglebutton.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the togglebutton's font, when it is available on your system.

### 3.10 RadioButton

A RadioButton is an item with a text string or an image, or both, that can be selected or deselected. Radio buttons are similar to check boxes, but are intended to be mutually exclusive within a group of related radio buttons (i.e., only one is in a pressed state at any given time). Radiobuttons generate an action when selected and indicate their state on the display.

To activate a radiobutton, click the mouse button on the object.

#### Example

```
jac = jacontrol('style','JRadioButton',...
               'string','JRadioButton',...
               'position',[10 10 200 20]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JRadioButton.

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the radiobutton.  
The context menu is displayed whenever you right-click over the radiobutton.  
Use the Matlab `uicontextmenu` function to create the context menu.

#### Mnemonic

Character, default value: "

The mnemonic is the key which when combined with the look and feel's mouseless modifier (usually Alt) will activate this radiobutton.

Mnemonics are case-insensitive. If the character defined by the mnemonic is found within the radiobutton's label string, the first occurrence of it will be underlined to indicate the mnemonic to the user.

#### UnselectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the radiobutton if its value is false.

Each value must be an RGB value between 0.0 and 1.0.

`CData(:,1)` defines the red components of the pixels

`CData(:,2)` defines the green components of the pixels

`CData(:,3)` defines the blue components of the pixels

Transparent pixels are marked with NaN's

#### SelectedIcon

matrix, default value is []

M x N x 3 truecolor matrix of an M x N pixels image to be displayed on the radiobutton if its value is true.  
Each value must be an RGB value between 0.0 and 1.0.  
CData(:, :, 1) defines the red components of the pixels  
CData(:, :, 2) defines the green components of the pixels  
CData(:, :, 3) defines the blue components of the pixels  
Transparent pixels are marked with NaN's

### **Callback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user selects or deselects the radiobutton.

### **MouseEnteredCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the radiobutton.

### **MouseExitedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the radiobutton.

### **String**

String, default value: ""

The text displayed with the radiobutton.  
HTML can be used to format the text.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the radiobutton.  
When the user moves the mouse pointer over the button and leaves it there, the tooltip is displayed. HTML can be used to format the text.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the radiobutton.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the radiobutton.

### **Value**

Boolean, default value: False

False if the radiobutton is not selected, True otherwise.  
If the radiobutton is placed in a toolbar the radiobutton's state is given as 'On' if the radiobutton is selected, 'Off' otherwise.

### **HorizontalAlignment**

{leading} | trailing | left | right | center

The alignment of the radiobutton's contents along the X axis.

**VerticalAlignment**

{center} | bottom | top

The alignment of the radiobutton's contents along the Y axis.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the radiobutton's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the radiobutton's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the string in the radiobutton.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the radiobutton's font, when it is available on your system.

### 3.11 ProgressBar

A progress bar typically communicates the progress of some work by displaying its percentage of completion and possibly a textual display of this percentage. To indicate that a task of unknown length is executing, the progress bar can be put into indeterminate mode. While the bar is in indeterminate mode, it animates constantly to show that work is occurring.

*Example*

```
jac = jacontrol('style','JProgressBar',...
               'value',40,...
               'position',[10 10 200 20]);
```



#### NextFocusableComponent

default value: []

Progressbars cannot receive focus.

#### ActiveComponent (Read only)

default value: []

Progressbars cannot receive focus.

#### Min

Scalar, default value: 0

The progress bar's minimum value.

#### Max

Scalar, default value: 100

The progress bar's maximum value.

#### Value

Scalar, default value: 0

The progress bar's current value, NaN for indeterminate mode.

#### ForegroundColor

RGB triple, default value is system dependent

The color of the progress bar.

#### BackgroundColor

RGB triple, default value is system dependent

The color of the progress bar's bounding box.

#### TooltipString

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the progressbar. When the user moves the mouse pointer over the progressbar and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### **Orientation**

{horizontal} | vertical

The progress bar's orientation.

### **String**

String, default value: "

Determines which string the progress bar should render if the paintstring property is true.

### **Paintstring**

Boolean, default value: False

Determines whether the progress bar should render a progress string.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the progressbar's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the progressbar's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the progressbar.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

### **FontAngle**

{normal} | italic

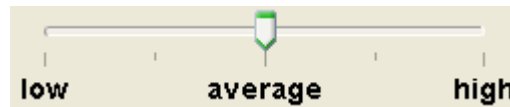
Setting this property to italic selects a slanted version of the progressbar's font, when it is available on your system.

### 3.12 Slider

A slider is a component that lets the user graphically select a value by sliding a knob within a bounded interval. The slider can show major tick marks, minor tick marks and labels between them. Users move the bar by pressing the mouse button and dragging the pointer over the bar, or by clicking in the trough. By releasing the mouse button a callback is triggered. The location of the bar indicates a numeric value.

#### Example

```
jac = jacontrol('style','JSlider',...
    'paintticks',true,...
    'minimum',0,'maximum',100,...
    'minortick',25,...
    'majortick',50,...
    'foregroundcolor',[0 1 1],...
    'label',{0 'low';50 ...
    'average'; 100 'high' },...
    'paintlabels',true,...
    'fontsize',14,...
    'fontweight','bold',...
    'position',[10 10 400 60]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JSlider.

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the slider.

The context menu is displayed whenever you right-click over the slider.

Use the Matlab `uicontextmenu` function to create the context menu.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user moves the slider.

#### Value

Scalar, default value: 0

The slider's current value.

#### ForegroundColor

RGB triple, default value is system dependent

The color of the slider.

#### BackgroundColor

RGB triple, default value is system dependent

The background color of the slider.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the slider. When the user moves the mouse pointer over the slider and leaves it there, the tooltip is displayed. HTML can be used to format the text.

### **Minimum**

Scalar, default value: 0

The slider's minimum allowed value.

### **Maximum**

Scalar, default value: 0

The slider's maximum allowed value.

### **MajorTickSpacing**

Scalar, default value: 0

Represents the distance, measured in values, between each major tick mark.

### **MinorTickSpacing**

Scalar, default value: 0

Represents the distance, measured in values, between each minor tick mark.

### **Orientation**

{horizontal} | vertical

The slider's orientation.

### **SnapToTicks**

Boolean, default value: False

Determines whether the slider knob (and the data value it represents) resolves to the closest tick mark next to where the user positioned it.

### **PaintLabels**

Boolean, default value: False

Determines whether labels are painted on the slider.

### **PaintTicks**

Boolean, default value: False

Determines whether tick marks are painted on the slider.

### **Labels**

Cellarray Nx2, default value: []

The First column is the value.

The second column is the corresponding string to be displayed in the slider if paintlabels is true.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the slider's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the slider's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the slider.

To display and print properly, this must be a font that your system supports.

Use listfonts to list all available system fontnames.

### **FontAngle**

{normal} | italic

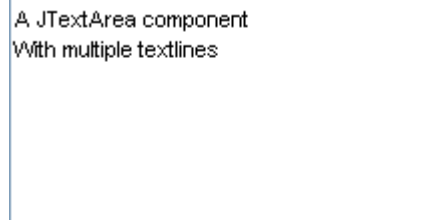
Setting this property to italic selects a slanted version of the slider's font, when it is available on your system.

### 3.13 TextArea

A TextArea can display multiple lines of editable text. Although a TextArea can display text in any font, all of the text is in the same font. A TextArea can be used to allow the user to enter unformatted text of any length or to display unformatted help information. A callback is triggered after the text entry is complete.

#### Example

```
jac = jacontrol('style','JTextArea',...
    'position',[10 10 540 400],...
    'string',strvcat(...
    'A JTextArea component',...
    'with multiple textlines'));
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the ctrl-tab key or ctrl-shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JTextArea

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the textarea.

The context menu is displayed whenever you right-click over the textarea.

Use the Matlab uicontextmenu function to create the context menu.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the textarea loses focus, i.e. when the user finishes editing.

#### String

String, default value: ""

The text displayed in the textarea.

#### TooltipString

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the textarea.

When the user moves the mouse pointer over the textarea and leaves it there, the tooltip is displayed. HTML can be used to format the text.

#### Editable

Boolean, default value: True

Indicates whether or not the textarea component is editable.

**MouseEnteredCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the textarea.

**MouseExitedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the textarea.

**BackgroundColor**

RGB triple, default value is system dependent.

The background color of the textarea.

**ForegroundColor**

RGB triple, default value is system dependent.

The foreground (text) color of the textarea.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the textarea's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the textarea's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the string in the textarea.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

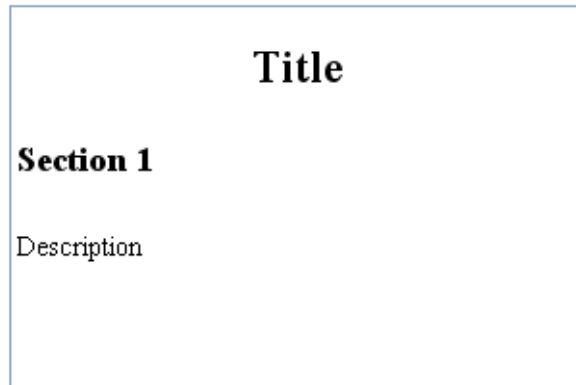
Setting this property to italic selects a slanted version of the textarea's font, when it is available on your system.

### 3.14 EditorPane

An EditorPane is a styled text component that can display editable text using more than one font. JEditorPanes can be easily loaded with HTML formatted text from a URL, which makes them useful for displaying uneditable help information.

#### Example

```
jac = jacontrol('style','JEditorPane',...
    'position',[10 10 300 300]);
s = '<html> <h1 align="center"> Title</h1>
<h2>Section 1</h2> <p>Description</p>';
set(jac,'string',s);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the ctrl-tab key or ctrl-shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JEditorPane

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the editorpane.  
The context menu is displayed whenever you right-click over the editorpane.  
Use the Matlab uicontextmenu function to create the context menu.

#### Page

The current URL being displayed.

The 'EditorContentType' is determined from the URL.

#### String

String, default value: ""

The text displayed in the editorpane. HTML can be used to format the text if 'EditorContentType' is set to 'text/html'.

#### TooltipString

String, default value: ""

The `TooltipString` property specifies the text of the tooltip associated with the editorpane. When the user moves the mouse pointer over the editorpane and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### **Editable**

Boolean, default value: True

Indicates whether or not the editorpane component is editable.

### **Callback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the editorpane loses focus, i.e. when the user finishes editing.

### **MouseEnteredCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the editorpane.

### **MouseExitedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the editorpane.

### **EditorContentType**

{text/plain} | text/html | text/rtf

text/plain - produces a wrapped plain text view.

text/html - HTML text.

text/rtf - RTF text.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground color of the editorpane.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the editorpane's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the editorpane's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the editorpane.

To display and print properly, this must be a font that your system supports.

Use `listfonts` to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the editorpane's font, when it is available on your system.

### 3.15 PasswordField

A PasswordField is a specialized text fields for password entry. For security reasons, a password field does not show the characters that the user types. Instead, the field displays a character different from the one typed, such as an asterisk '\*'. Like an ordinary text field, a password field can trigger a callback when the user indicates that text entry is complete, for example by pressing the Enter button.

#### Example

```
jac = jacontrol('style','JPasswordField',...
               'position',[10 10 200 20]);
```

```
set(jac,'string','password');
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JPasswordField

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the passwordfield.

The context menu is displayed whenever you right-click over the passwordfield.

Use the Matlab uicontextmenu function to create the context menu.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user changes the password.

#### String

String, default value: ""

The current password.

#### EchoChar

Character, default value: '\*'

The echo character which is displayed instead of the actual characters typed by the user.

#### TooltipString

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the passwordfield.

When the user moves the mouse pointer over the passwordfield and leaves it there, the tooltip is displayed.

HTML can be used to format the tooltipstring's text.

**ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the passwordfield.

**BackgroundColor**

RGB triple, default value is system dependent

The background color of the passwordfield.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the passwordfield's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the passwordfield's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the string in the passwordfield.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic


Setting this property to italic selects a slanted version of the passwordfield's font, when it is available on your system.

### 3.16 TextField

A text field which can display only one line of editable text. The textfield's contents can be aligned by setting the horizontal alignment. A callback is triggered after the text entry is complete.

#### Example

```
jac = jacontrol('style','JTextField',...
              'string','A JTextField',...
              'position',[10 10 200 20]);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JTextField

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the textfield.

The context menu is displayed whenever you right-click over the textfield.

Use the Matlab uicontextmenu function to create the context menu.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user changes the textfield's contents.

#### String

String, default value: ""

The current displayed text in the textfield.

#### HorizontalAlignment

{leading} | trailing | left | right | center

The alignment of the textfield's contents along the X axis.

#### ForegroundColor

RGB triple, default value is system dependent

The foreground (text) color of the textfield.

#### BackgroundColor

RGB triple, default value is system dependent

The background color of the textfield.

### **TooltipString**

String, default value: "

The TooltipString property specifies the text of the tooltip associated with the textfield. When the user moves the mouse pointer over the textfield and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the textfield's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the textfield's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the textfield. To display and print properly, this must be a font that your system supports. Use listfonts to list all available system fontnames.

### **FontAngle**

{normal} | italic

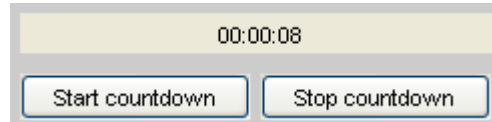
Setting this property to italic selects a slanted version of the textfield's font, when it is available on your system.

### 3.17 Countdown

A Countdown is a timer that counts down and which executes a callback upon reaching zero.

#### Example

```
jac = jacontrol('style','CountDown',...
               'Seconds',10,...
               'position',[10 10 200 20]);
startCountDown(jac);
```



#### NextFocusableComponent (Not used)

Jacontrol, default value: []

A countdown object cannot receive focus.

#### ActiveComponent (Read only)

Jacontrol, default value: []

A countdown object cannot receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the countdown object.

The context menu is displayed whenever you right-click over the countdown component.

Use the Matlab `uicontextmenu` function to create the context menu.

#### HorizontalAlignment

{leading} | trailing | left | right | center

The alignment of the countdown's contents along the X axis.

#### VerticalAlignment

{center} | bottom | top

The alignment of the countdown's contents along the Y axis.

#### String

String, default value: ""

The text displayed on the countdown object together with the remaining time.

#### TooltipString

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the countdown component. When the user moves the mouse pointer over the countdown component and leaves it there, the tooltip is displayed. HTML can be used to format the text.

#### Seconds

Integer, default value: 0

The time (in number of seconds) displayed on the countdown object.  
 If the component is counting down a callback is executed after this number of seconds has passed.

**BackgroundColor**

RGB triple, default value is system dependent

The background color of the countdown component.

**ForegroundColor**

RGB triple, default value is system dependent

The color of the text in the countdown component.

**Callback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the countdown has reached zero.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the countdown's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent

A number specifying the size in pixels of the countdown's font.

**FontName**

String, default is system dependent

The name of the font in which to display the string in the countdown.  
 To display and print properly, this must be a font that your system supports.  
 Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the countdown's font, when it is available on your system.

A number of utility functions are available for the JaControl with style Countdown:

- **startCountDown**

```
startCountDown - start the counting down process for a jacontrol of type
                  Countdown

CALL:
  startCountDown(obj)

INPUT:
  obj: <jacontrol object> of type Countdown

OUTPUT:
  No direct output, the countdown will start counting down

See also: jacontrol
```

- **stopCountdown**

**stopCountdown** - stop the counting down process for a jacontrol of type Countdown

**CALL:**  
stopCountdown(obj)

**INPUT:**  
obj: <jacontrol object> of type Countdown

**OUTPUT:**  
No direct output, the countdown will stop counting down

**See also:** jacontrol

### 3.18 PieceBar

A PieceBar is an object that can display a series of colored rectangular patches. Can be used for example to visualize the status or values of timeseries. Or to simultaneously visualize the progress of various jobs.

#### Example

```
colorset = [[1:64]' hsv(64)];
content = struct('widths',ones(4*2*64,1),...
    'value',repmat([[1:64]';[64:-1:1]'],4,1));
jac = jacontrol('style','PieceBar',...
    'content',content,...
    'colorSet',colorset,...
    'position',[10 10 200 20]);
```



#### NextFocusableComponent (Not used)

Default value: []

A piecebar cannot receive focus.

#### ActiveComponent (Read only)

Default value: []

A piecebar cannot receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the piecebar.

The context menu is displayed whenever you left-click or right-click over the piecebar.

Use the Matlab `uicontextmenu` function to create the context menu.

#### Content

Structure with fields

- widths: n x 1 vector with cell widths
- value: n x 1 vector with colorindices

For every cell its color is looked up in the colorset.

If the colorindex is not present in the colorset the corresponding cell is painted black.

#### ColorSet

Matrix Nx4, default value: []

First column is an index

Second, third and fourth column specify the RGB values for the corresponding index.

The colorset is used to determine the color of every cell in the piecebar by using the color index specified for every cell.

#### BackgroundColor

RGB triple, default value is system dependent

The color of the countdown's border.

#### TooltipString

String, default value: "

The TooltipString property specifies the text of the tooltip associated with the piecebar. When the user moves the mouse pointer over the piecebar and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### 3.19 AutoComboBox

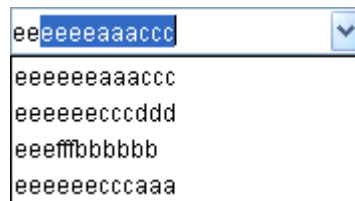
A component that combines a button and an editable field. A combobox can open up by pressing the button to display a list of choices which can be selected with the mouse. A new choice may also be modified by typing in the textfield, as the user types the set of possible choices will narrow down. When not open, a combobox indicates the current choice. Comboboxes are useful when you want to provide users with a number of mutually exclusive choices, but do not want to take up the amount of space that a series of radio buttons requires. A callback is triggered after a user has changed the selected item.

#### Example

```
jac = jacontrol('style','AutoComboBox',...
    'position',[10 10 200 20]);

content = {'eeeeeeaaacc',...
    'eeeeeeccddd',...
    'eeeffbbbbb',...
    'eeeeeeccaaa'};

set(jac, 'content', content);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.ComboBoxEditor.

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the combobox. The context menu is displayed whenever you right-click over the combobox. Use the Matlab `uicontextmenu` function to create the context menu.

#### MouseEnteredCallback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse enters the combobox.

#### MouseExitedCallback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the mouse exits the combobox.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user has finished editing the combobox or when the user has selected a new item.

#### Editable

Boolean, default value: True

If true new items can be added by the user to the choicelist  
If false the selection is restricted to the choices

### **Content**

Cellarray, default value: {}

Specifies the choicelist for the combobox. For example {'Choice 1', 'Choice 2', 'Choice 3'}.

### **Value**

Integer, default value: 0

The index in the choicelist of the current chosen option.

### **String**

String, default value: ""

The name of the current chosen option.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the combobox. When the user moves the mouse pointer over the combobox and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

### **ForegroundColor**

RGB triple, default value is system dependent

The foreground (text) color of the combobox.

### **BackgroundColor**

RGB triple, default value is system dependent

The background color of the combobox.

### **FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the combobox's font, when it is available on your system.

### **FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the combobox's font.

### **FontName**

String, default is system dependent.

The name of the font in which to display the string in the combobox.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

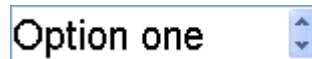
Setting this property to italic selects a slanted version of the combobox's font, when it is available on your system.

### 3.20 Spinner

A spinner is a single line input field with two arrow buttons that lets the user select a value from an ordered sequence. The user may type a (legal) value directly into the spinner or use the arrow buttons or the keyboard up/down arrow keys to select a value from the sequence. Although combo boxes provide similar functionality, spinners are sometimes preferred because they don't require a drop down list that can obscure important data. A callback is triggered when the user changes the selected value in the spinner.

#### Example

```
Content = {'Option one',...
          'Option two',...
          'Option three',...
          'Option four' };
jac = jacontrol('style','Spinner',...
              'position',[10 10 200 20],...
              'Content',Content);
```



#### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object of type javax.swing.JFormattedTextField

The component that can receive focus.

#### CMenu

Handle, default value: []

The handle of the context menu associated with the spinner.

The context menu is displayed whenever you right-click over the spinner.

Use the Matlab `uicontextmenu` function to create the context menu.

#### Callback

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user changes the selected value.

#### Value

Object, default value: first item from contents list

The current selected choice, can be a string or a number.

#### Content

Cellarray or 1x4 vector, default value: [0 -Inf Inf 1]

Cellarray: contains the elements of the choice list, elements can be numbers or strings and can be mixed.

1x4 vector: [value begin end step]

value: current selected value

begin: lowest possible value

end: highest possible value

step: stepsize

**ForegroundColor**

RGB triple, default value is system dependent

The color of the text in the spinner.

**BackgroundColor**

RGB triple, default value is system dependent

The background color of the spinner.

**TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the spinner. When the user moves the mouse pointer over the spinner and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

**FontWeight**

{normal} | bold

Setting this property to bold selects a bold version of the spinner's font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the spinner's font.

**FontName**

String, default is system dependent.

The name of the font in which to display the string in the spinner. To display and print properly, this must be a font that your system supports. Use listfonts to list all available system fontnames.

**FontAngle**

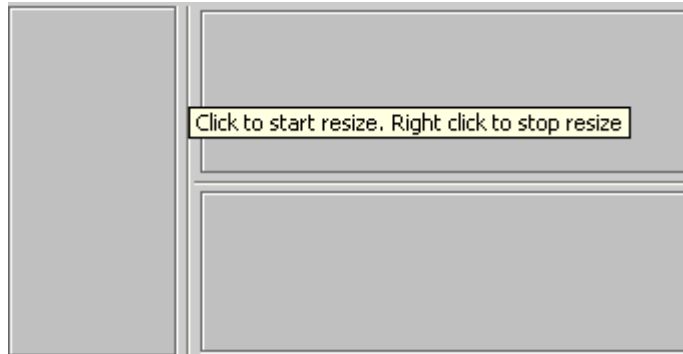
{normal} | italic

Setting this property to italic selects a slanted version of the spinner's font, when it is available on your system.

### 3.21 Separator

A Separator is a visual divider line that can be used in a GUI. Separators can detect mouse dragging and are therefore useful for making resizable panes.

*Example*



#### **NextFocusableComponent**

Jacontrol, default value: []

Separators cannot receive focus.

#### **ActiveComponent (Read only)**

Default value: []

Separators cannot receive focus.

#### **Orientation**

{horizontal} | vertical

The slider's orientation.

#### **MousePressedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user presses the mouse over the separator.

#### **MouseDraggedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user drags the separator.

#### **MouseReleasedCallback**

String, cell array or function handle, default value: []

Specifies the function that should be executed when the user releases the mouse over the separator.

#### **BackgroundColor**

RGB triple, default value is system dependent  
The background color of the separator.

#### **CursorAware**

on | {off}

If this property is set to 'on' then the mousepointer will change into a vertical or horizontal arrow, according to the separator's orientation, when the user hovers over the separator.

### **TooltipString**

String, default value: ""

The TooltipString property specifies the text of the tooltip associated with the separator. When the user moves the mouse pointer over the separator and leaves it there, the tooltip is displayed. HTML can be used to format the tooltipstring's text.

## 3.22 ToolBar

A ToolBar is a container that groups several jacontrols — usually buttons with icons — into a row or column.

```
jac = jacontrol('style','JToolBar',...
               'position',[10 10 400 20]);
jacontrol('style','JButton',...
          'string','jbutton',...
          'parent',jtoolbar);
jacontrol('style','JToggleButton',...
          'string','togglebutton',...
          'parent',jtoolbar);
jacontrol('style','JRadioButton',...
          'string','radiobutton',...
          'parent',jtoolbar);
jacontrol('style','JCheckBox',...
          'string','checkbox',...
          'parent',jtoolbar);
jacontrol('style','JLabel',...
          'string','label',...
          'parent',jtoolbar);
```



### NextFocusableComponent

Jacontrol, default value: []

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

### ActiveComponent (Read only)

Java object, not used for toolbars.

The component that can receive focus.

### Orientation

{horizontal} | vertical

The toolbar's orientation.

### BackgroundColor

RGB triple, default value is system dependent

The color of the toolbar.

### Floatable

Boolean, default value: True

Indicates if the toolbar can be dragged into a separate window. If this property is true, it has bumps painted at its left edge.

### Rollover

Boolean, default value: False

Indicates if the toolbar buttons are visually indicated when the user passes over them with the cursor.

**Border**

Boolean, default value: True

Indicates whether a border is painted around the toolbar.

A number of utility functions are available for the JaControl with style JToolBar:

- **addGlue**

`addGlue` - add stretchable space to toolbar

**CALL:**

`addGap(toolbar)`

**INPUT:**

`addGap`: <jacontrol object> of type JToolBar

**OUTPUT:**

no output, stretchable space is added to the end of the toolbar

See also: `addGap`, `addSeparator`

- **addGap**

`addGap` - add space to toolbar

**CALL:**

`addGap(toolbar, space)`

**INPUT:**

`toolbar`: <jacontrol object> of type JToolBar

`space`: <integer> size of space in pixels

**OUTPUT:**

no output, space is added to the end of the toolbar

See also: `addGlue`, `addSeparator`

- **addSeparator**

`addSeparator` - add a vertical separator to toolbar

**CALL:**

`addSeparator(toolbar)`

**INPUT:**

`toolbar`: <jacontrol object> of type JToolBar

**OUTPUT:**

no output, a vertical separator is added to the end of the toolbar

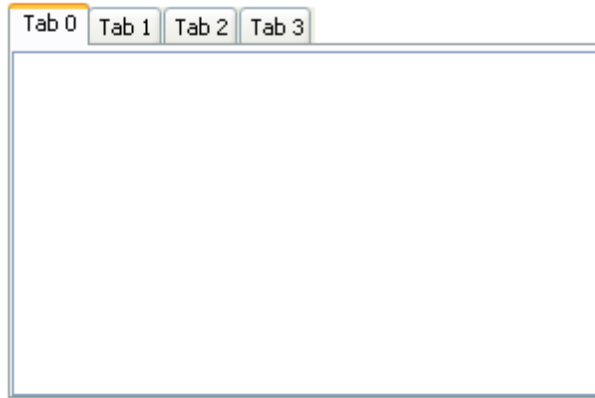
See also: `addGap`, `addGlue`

### 3.23 JTabbedPane

With a JTabbedPane several jacontrols can share the same space. The user chooses which component to view by selecting the tab corresponding to the desired component.

#### Example

```
jac = jacontrol('style','JTabbedPane',...
               'backg',[1 1 1],...
               'position',[10 10 400 200]);
jacontrol('style','jlabel',...
          'parent',jac,...
          'backgroundcolor',[1 1 1]);
jacontrol('style','jlabel',...
          'parent',jac,...
          'backgroundcolor',[0 1 0]);
jacontrol('style','jlabel',...
          'parent',jac,...
          'backgroundcolor',[0 0 1]);
jacontrol('style','jlabel',...
          'parent',jac,...
          'backgroundcolor',[0 0 1]);
```



#### NextFocusableComponent

Jacontrol, default value: [], not used for tabbedpanes.

Specifies the next component to get the focus after this one, for example, when the tab key or shift-tab key is used.

#### ActiveComponent (Read only)

Java object, not used for tabbedpanes.

The component that can receive focus.

#### TabLayoutPolicy

{wrap} | scroll

The policy which the tabbedpane uses in laying out the tabs when all the tabs will not fit within a single run.

#### TabPlacement

{top} | bottom | left | right

The location of the tabs.

#### BackgroundColor

RGB triple, default value is system dependent

The background color of the tabs.

#### ForegroundColor

RGB triple, default value is system dependent

The foreground (text) color of the tabs.

#### FontWeight

{normal} | bold

Setting this property to bold selects a bold version of the tabs' font, when it is available on your system.

**FontSize**

Integer, default value is system dependent.

A number specifying the size in pixels of the tabs' font.

**FontName**

String, default is system dependent.

The name of the font in which to display the text in the tabs.  
To display and print properly, this must be a font that your system supports.  
Use listfonts to list all available system fontnames.

**FontAngle**

{normal} | italic

Setting this property to italic selects a slanted version of the tabs' font, when it is available on your system.

### 3.24 JXTable

A JXTable is a table with sorting and filtering support. Hierarchical data can also be organized as a treetable. Furthermore the table has some build in features to render icons, dates, comboboxes, rearrange and hide and show columns, export to Excel format and to color the foreground and background of the table cells.

*Example*

```
jac = jacontrol('style','JXTable',...
'dateformat',{5 'dd-MM-yyy'},...
'position',[10 10 400 300]);
```

```
Content = dir;
set(jac, 'Content', Content);
```

name	date	bytes	isdir	datenum
.	10-jun-2008 09:43:31	0	<input checked="" type="checkbox"/>	10-06-2008
..	09-jun-2008 16:04:54	0	<input checked="" type="checkbox"/>	09-06-2008
AutoComboBox.asv	09-jun-2008 12:23:12	722	<input type="checkbox"/>	09-06-2008
AutoComboBox.m	09-jun-2008 12:30:25	825	<input type="checkbox"/>	09-06-2008
Browser.asv	06-jun-2008 15:02:33	333	<input type="checkbox"/>	06-06-2008
Browser.html	06-jun-2008 15:21:07	2574	<input type="checkbox"/>	06-06-2008
Browser.m	09-jun-2008 14:18:57	368	<input type="checkbox"/>	09-06-2008
Browser.png	06-jun-2008 15:21:06	174	<input type="checkbox"/>	06-06-2008
Browser.xml	06-jun-2008 15:21:06	2685	<input type="checkbox"/>	06-06-2008
Browser_01.png	06-jun-2008 15:21:06	1795	<input type="checkbox"/>	06-06-2008
Browser_02.png	06-jun-2008 15:21:06	1795	<input type="checkbox"/>	06-06-2008
CountDown.asv	10-jun-2008 09:30:58	691	<input type="checkbox"/>	10-06-2008
CountDown.m	10-jun-2008 09:35:08	691	<input type="checkbox"/>	10-06-2008
DomEcho.asv	05-jun-2008 13:53:44	516	<input type="checkbox"/>	05-06-2008
DomEcho.m	05-jun-2008 13:56:12	516	<input type="checkbox"/>	05-06-2008
UsefulLib.asv	07-jun-2008 07:00:01	1005	<input type="checkbox"/>	07-06-2008

**NextFocusableComponent**

Default value: []

Tables cannot receive focus.

**ActiveComponent (Read only)**

Default value: []

Tables cannot receive focus.

**CMenu**

Handle, default value: []

The handle of the context menu associated with the table.  
 The context menu is displayed whenever you right-click over the table.  
 Use the Matlab uicontextmenu function to create the context menu.

**Busy**

Boolean, default value: False

If true a indeterminate progress indicator appears on the faded table.  
 While this indicator is visible the table is disabled.

**DateFormat**

Cellarray, default value: {}

Specifies the dateformat per table column, format:  
 {columnnumber1 dateformatstring ... columnnumbern dateformatstring}

For example {1 'dd-MM yyyy' 2 'HH:mm'}

Character	Description	Format	Example
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
W	Week in month	Number	2
Z	Time zone	RFC 822 time zone	-0800
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00

### ComboboxFormat

Cellarray, default value: {}

Specifies the combobox editor per table column, format:

{columnnumber1 choicelist ... columnnumbern choicelist}

For example {1 {'first', 'second', 'third'} 2 {'first', 'second', 'third'}}

See the help of the Matlab function sprintf for more info on possible number format strings.

### NumberFormat

Cellarray, default value: {}

String, default value: '%g'

Specifies the numberformat per table column, format:

{columnnumber1 numberformatstring ... columnnumbern numberformatstring}

For example {1 '%g' 2 '%6.2f'}

Or

Specifies the numberformat for all columns which contain floating numbers.

Specification of a numberformat in a column overrules the general format set with the string argument.

Warning: it is not possible to use integer formatting for doubles.

### ColorSet

Matrix Nx4, default value: []

First column is an index

Second, third and fourth column specify the RGB values for the corresponding index.

The colorset is used to determine the color of every cell in the table by using the color index specified for every cell.

### CellBackground

Matrix, default value: []

specifies the colorindex per table cell, the matrix with colorindices has the same size as the table itself. Colorindices are used by the property Colorset to determine every cell's background color.

### **CellForeground**

Matrix, default value: []

specifies the colorindex per table cell, the matrix with colorindices has the same size as the table itself. Colorindices are used by the property Colorset to determine every cell's foreground (text) color.

### **Scrollbar**

Boolean, default value: False

Normally a JXTable in a JScrollPane will fit all columns within the scrollpane's viewport, possibly causing some of them to resize beneath an optimal width and making them illegible.

By setting the Scrollbar property to true all columns will be resized to their preferred width, possibly extending the table outside the scrollpane's viewport.

### **Highlighter**

{none} | | notePadBackground | beige | classicLinePrinter | floralWhite | linePrinter | quickSilver

Rows can be highlighted to make it easier to distinguish between different rows by using a subtle background color scheme for every other row.

### **ColumnControlVisible**

Boolean, default value: False

The Column Control is an icon that renders to the right of the column headers, just above the vertical scrollbar. When a user clicks the icon, a popup menu appears, allowing the user to show or hide columns from the view, and to set a couple of other properties.

### **ShowGrid**

{grid} | horizontal | vertical | none

Specifies whether the table draws horizontal lines, vertical lines, both or none between the table cells.

### **Callback**

String, cell array or function handle, default value: []

Function that should be executed when the user changes the current selection.

### **DataCallback**

String, cell array or function handle, default value: []

Function that should be executed when the user changes the value of a cell in the table.

### **keyPressedCallback**

String, cell array or function handle, default value: []

Function that should be executed when the user presses a key or combination of keys.

### **Content**

Structure, default value: []

+----header

+----data

+----hierarchy (Optional)

+----icon (Optional, only used when hierarchy specified)

data for table:

header: cellstring with the table's columnnames.

Data: cell array with cell values, every column should have the same format e.g. all strings, number or java objects.

Hierarchy: vector of integer, describes for every row which row the row number of its predecessor.

Icon: cellstring with names of icons to be displayed in every row.

### **Editable**

Vector of integers, default value: []

Specifies the columns which are editable.

### **Value**

Integer: default value: []

A vector of indices corresponding to the currently selected table rows.

1 corresponds to the first row in the table.

### **SelectionMode**

{1} | 2 | 3

The selection mode. The following selectionMode values are allowed:

1: SINGLE\_SELECTION

Only one list index can be selected at a time.

2: SINGLE\_INTERVAL\_SELECTION

One contiguous index interval can be selected at a time.

3: MULTIPLE\_INTERVAL\_SELECTION

In this mode, there's no restriction on what can be selected.

### **GridBackground**

RGB triple, default value is system dependent

The background color of the table and the color of the visible part of the viewport.

### **FilteredRows**

Vector of integers or vector of boolean, default value: []

Specifies the visible rows. If a vector of booleans is used its length must be equal to the number of rows in the table.

### **ExpandedRows**

Vector of integers or vector of boolean, default value: []

Specifies the visible rows if a hierarchy is specified by expanding or collapsing nodes in the tree. If a vector of booleans is used its length must be equal to the number of rows in the table.

### **Iconset**

String, default value: "

The name of the jar file with icons or the full path to the jar file.

### FontWeight

{normal} | bold

Setting this property to bold selects a bold version of the table's font, when it is available on your system.

### FontSize

Integer, default value is system dependent.

A number specifying the size in pixels of the table's font.

### FontName

String, default is system dependent.

The name of the font in which to display the string in the table.

To display and print properly, this must be a font that your system supports.

Use listfonts to list all available system fontnames.

### FontAngle

{normal} | italic

Setting this property to italic selects a slanted version of the table's font, when it is available on your system.

A number of utility functions are available for the JaControl with style JXTable:

- **getTableValue**

**getTableValue** - this function can be used to retrieve the column and row in the original datamodel for which a edit action took place. this function is typically used in a callback.

**CALL:**  
[value, row, col, colname] = getTableValue(obj, event)

**INPUT:**  
obj: <n1.modelit.mdltttable.mdlttTable object> the table on which the event took place. (argument of the datacallback of a table)  
event: <n1.modelit.mdltttable.event.TableChangedEvent> description of the event giving information about the row and column of the table in which the cell was edited (starting from row == 1 and column == 1) (argument of the datacallback of a table)

**OUTPUT:**  
value: the value of the tablecell which was edited.  
row: <integer> row of changed cell in the original datamodel counting from 1.  
col: <integer> column of changed cell in the original datamodel counting from 1.  
colname: <string> with columnname

- **tableAction**

**tableAction** - programmatically select a row in a jxtable. with this routine the first, last, previous or next row can be selected in the current table (with its current sorting and filtering status).

**CALL:**  
tableAction(sorttable, actionKey)

**INPUT:**  
sorttable: <jacontrol-object> of type jxtable  
actionKey: <string> action, allowed values:  
- 'selectPreviousRow'  
- 'selectNextRow'  
- 'selectFirstRow'  
- 'selectLastRow'

**OUTPUT:**  
No direct output, the selection of the table will be changed.

See also: jacontrol

## 4 Utility functions reference manual

- **gcjh**

**gcjh** - find jacontrol object handles with specified tags

**CALL:**

`h = gcjh(tag, hwin, h)`

**INPUT:**

`tag`: string or cellstring with tags  
`hwin`: (optional) handle of window to search in  
          default value: `gcf`  
`h`: (optional) the default value

**OUTPUT:**

`h`: array with handles of jacontrol object with the specified tag

See also: `gch`, `findjac`

- **findjac**

**findjac** - find all jacontrol object handles under a specified handle

**CALL:**

`h = findjac(HWIN)`

**INPUT:**

`HWIN`: handle of window to search in

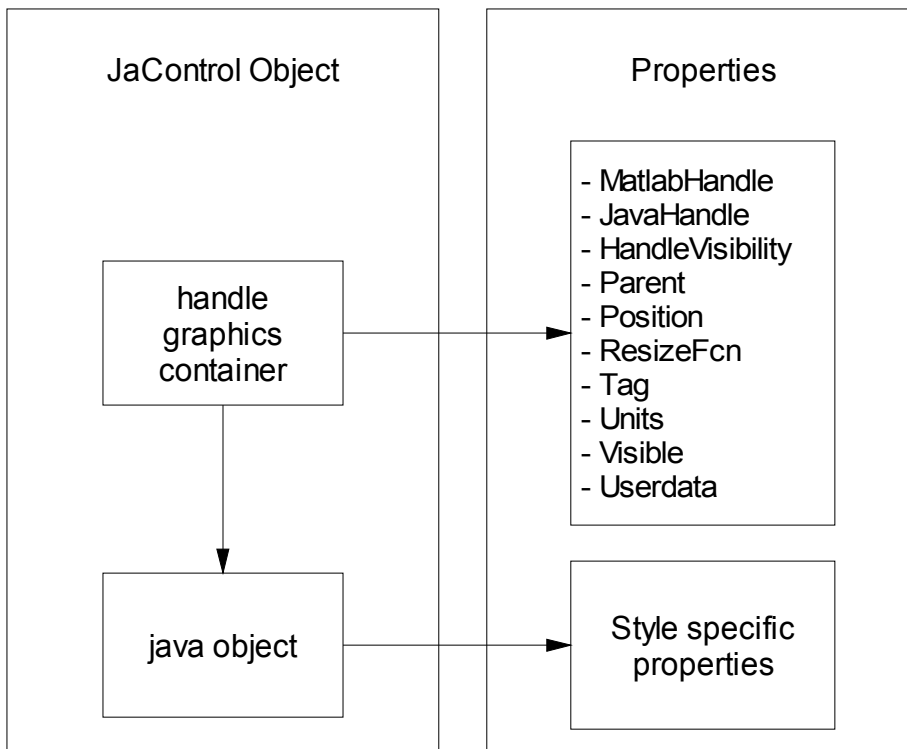
**OUTPUT:**

`h`: array with handles of jacontrol object with the specified handle as parent

See also: `gcjh`

## 5 Technical Background of the toolbox

The JaControl object is based on the Matlab function `javacomponent`. This function makes it possible to embed java objects in a Matlab figure. The JaControl object aggregates the Matlab handle graphics container and the Java component (as created by the `javacomponent` function) into a single object. Matlab uses the handle graphics container to embed the java object in the Matlab figure. Both components can be retrieved from a JaControl Object by using the JaControl get method, the handle graphics container is stored in the `MatlabHandle` field and the java object is stored in the `JavaHandle` field of the JaControl object. See Figure 6.



**Figure 6:** Schematic The Jacontrol Object.

The JaControl object acts as an interface between the Java object and Matlab, effectively hiding the java object from the user. All relevant properties can be queried and set by using the JaControl's set and get methods.

## 6 Compilation and deployment

This chapter describes which steps should be taken to compile applications that include the User Interface Components Toolbox to a standalone application and how to deploy these applications.

### 6.1 Deployment

To deploy a compiled application the procedure to install the toolbox for the development environment must be repeated for the deployed environment. This means that the following is needed:

- The required toolbox files;
- The Java classpath must be set.

A number of utilities are provided with the toolbox that copy the required files.

#### Toolbox files

The required toolbox files are installed in the development environment. The utility "installpackage" is provided with the toolbox and can be used to copy these files to the deployed environment.

#### Java classpath

In the the development environment the classpath is set once by running "installjar". The same procedure should be followed in the deployed environment. To make this possible the installjar function must be compiled and run prior to running the compiled application.

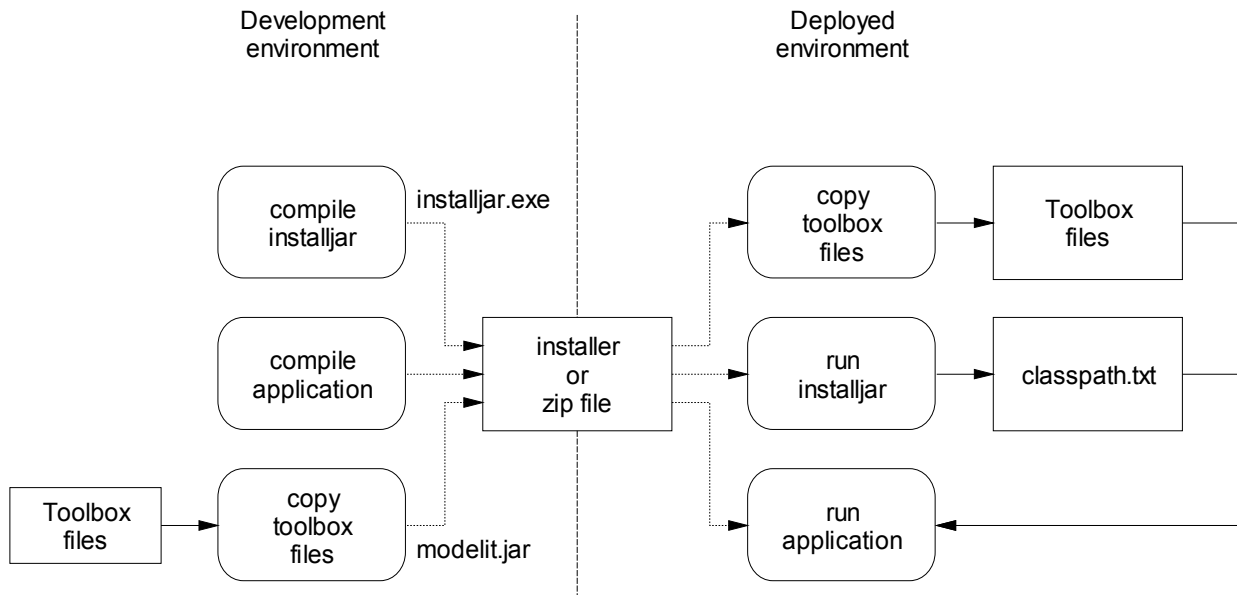


Figure 7: Overview of compile and deployment process

### 6.2 Compilation

If an application runs from a matlab development environment, no specific steps are needed to compile this application. In view of what is need for the deployed environment (see section 6.1), the following build script can

be used as a template for creating a install file. In the current example all required files are copied to the directory pwd/exe.

example build script:

```
%copy toolbox files to directory pwd/exe:  
installPackage({'modelit'}, 'exe');  
  
%compile application to directory pwd/exe  
mcc -m -d exe myApplication  
  
%compile installjar utility to directory pwd/exe  
mcc -m -d exe installjar
```