

# francy

## Framework for Interactive Discrete Mathematics

2.0.3

16 April 2023

**Manuel Martins**

**Manuel Martins**

Email: [manuelmachadomartins@gmail.com](mailto:manuelmachadomartins@gmail.com)

Homepage: <http://github.com/mcmartins>

Address: Departamento de Ciências e Tecnologia da Universidade  
Aberta  
Lisboa, Portugal

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Francy . . . . .	4
1.2	Applications . . . . .	4
1.3	Functionality . . . . .	4
1.4	Installation . . . . .	4
1.5	How it works . . . . .	5
1.6	Publications . . . . .	5
<b>2</b>	<b>Francy Callbacks</b>	<b>6</b>
2.1	Categories . . . . .	6
2.2	Families . . . . .	7
2.3	Representations . . . . .	7
2.4	Operations . . . . .	7
2.5	Globals . . . . .	9
2.6	Attributes . . . . .	9
<b>3</b>	<b>Francy Canvas</b>	<b>11</b>
3.1	Categories . . . . .	11
3.2	Families . . . . .	11
3.3	Representations . . . . .	11
3.4	Operations . . . . .	12
3.5	Global . . . . .	14
3.6	Attributes . . . . .	14
<b>4</b>	<b>Francy Charts</b>	<b>17</b>
4.1	Categories . . . . .	17
4.2	Families . . . . .	18
4.3	Representations . . . . .	18
4.4	Operations . . . . .	19
4.5	Global . . . . .	21
4.6	Attributes . . . . .	21
<b>5</b>	<b>Francy Core</b>	<b>23</b>
5.1	Categories . . . . .	23
5.2	Families . . . . .	24
5.3	Global . . . . .	24

5.4	Attributes . . . . .	24
<b>6</b>	<b>Francy Graphs</b>	<b>25</b>
6.1	Categories . . . . .	25
6.2	Families . . . . .	26
6.3	Representations . . . . .	26
6.4	Operations . . . . .	27
6.5	Global . . . . .	30
6.6	Attributes . . . . .	30
<b>7</b>	<b>Francy Menus</b>	<b>37</b>
7.1	Categories . . . . .	37
7.2	Families . . . . .	37
7.3	Representations . . . . .	37
7.4	Operations . . . . .	37
7.5	Attributes . . . . .	39
<b>8</b>	<b>Francy Messages</b>	<b>40</b>
8.1	Categories . . . . .	40
8.2	Families . . . . .	40
8.3	Representations . . . . .	40
8.4	Operations . . . . .	41
8.5	Global . . . . .	41
8.6	Attributes . . . . .	41
<b>9</b>	<b>Francy Renderers</b>	<b>43</b>
9.1	Categories . . . . .	43
9.2	Families . . . . .	43
9.3	Representations . . . . .	43
9.4	Operations . . . . .	44
9.5	Global . . . . .	44
9.6	Attributes . . . . .	44
<b>10</b>	<b>Francy Util</b>	<b>46</b>
10.1	Operations . . . . .	46
	<b>Index</b>	<b>47</b>

# Chapter 1

## Introduction

### 1.1 Francy

Francy arose from the necessity of having a lightweight framework for building interactive graphics, generated from GAP, running primarily on the web, specially in [Jupyter](#) environments. An initial attempt to re-use XGAP and port it was made, but the lack of a standardized data exchange format between GAP and the graphics renderer, and the simplistic initial requirements of the project were the basis for the creation of a new GAP package. Take a look at Francy functionality on these [Jupyter Notebooks](#).

### 1.2 Applications

Francy has potentially many applications and can be used to provide a graphical representation of data structures, allowing one to navigate through and explore properties or relations of these structures. In this way, Francy can be used to enrich a learning environment where GAP provides a library of thousands of functions implementing algebraic algorithms as well as large data libraries of algebraic objects. [FrancyMonoids](#) and [SubgroupLattice](#) are some example packages using Francy.

### 1.3 Functionality

Francy provides an interface to draw graphics using objects. This interface is based on simple concepts of drawing and graph theory, allowing the creation of directed and undirected graphs, trees, line charts, bar charts and scatter charts. These graphical objects are drawn inside a canvas that includes a space for menus and to display informative messages. Within the canvas it is possible to interact with the graphical objects by clicking, selecting, dragging and zooming.

### 1.4 Installation

This package requires the GAP packages `JupyterKernel`, `json` and `uuid`, all of which are distributed with GAP by default. Francy requires [Jupyter](#) to be installed on your system. Please note that you need to run the installation commands from the same python version [Jupyter](#) is installed on. Currently, Francy is supported only on [Jupyter Lab](#), if you want to use it on the old [Jupyter Notebooks](#),

please install the latest supported version: v1.2.4. In order to install or update Francy, please run the following command to download the latest version available from <https://pypi.org/>:

Example

```
mcmartins@local:~$ pip install -U jupyterlab-francy
```

## 1.5 How it works

Francy requires a rendering package to display graphics. Francy provides by default 3 Renderers based on D3.js, Vis.js and Graphviz, for rendering the semantic model representations produced by the GAP package. The renderers can be switched at any time using the user interface, by selecting 'Settings > Renderers' in the main menu. This library is distributed both as a browser module and as a [Jupyter](#) Lab extension. This library can be used in [Jupyter](#) Lab, using the JupyterKernel and the MIME type 'application/vnd.francy+json' to render the document. Please check the [JavaScript Documentation](#) for more information.

## 1.6 Publications

[ICMS 2018](#)

## Chapter 2

# Francy Callbacks

Callbacks are objects holding a Function, a list of arguments and a trigger event. Callbacks are used to execute GAP code from a remote client using the Trigger Operation.

Callbacks can be added directly to Menus and Shapes.

Please see Francy-JS for client implementation.

## 2.1 Categories

In this section we show all Francy Callback Categories.

### 2.1.1 IsCallback (for IsFrancyObject)

▷ IsCallback(*arg*) (filter)  
**Returns:** true or false  
Identifies Callback objects.

### 2.1.2 IsRequiredArg (for IsFrancyObject)

▷ IsRequiredArg(*arg*) (filter)  
**Returns:** true or false  
Identifies RequiredArg objects.

### 2.1.3 IsArgType (for IsFrancyTypeObject)

▷ IsArgType(*arg*) (filter)  
**Returns:** true or false  
Identifies ArgType objects.

### 2.1.4 IsTriggerType (for IsFrancyTypeObject)

▷ IsTriggerType(*arg*) (filter)  
**Returns:** true or false  
Identifies TriggerType objects.

## 2.2 Families

In this section we show all Francy Callback Families.

## 2.3 Representations

In this section we show all Francy Callback Representations.

### 2.3.1 IsCallbackRep (for IsComponentObjectRep)

- ▷ IsCallbackRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a Callback internal representation.

### 2.3.2 IsRequiredArgRep (for IsComponentObjectRep)

- ▷ IsRequiredArgRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a RequiredArg internal representation.

### 2.3.3 IsArgTypeRep (for IsComponentObjectRep)

- ▷ IsArgTypeRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a ArgType internal representation.

### 2.3.4 IsTriggerTypeRep (for IsComponentObjectRep)

- ▷ IsTriggerTypeRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a TriggerType internal representation.

## 2.4 Operations

In this section we show all Francy Callback Operations.

### 2.4.1 Callback (for IsTriggerType, IsFunction, IsList)

- ▷ Callback(*IsTriggerType*, *IsFunction*, *IsList(object)*) (operation)  
**Returns:** Callback  
 Creates a Callback object that holds a Function and the RequiredArgs to be executed by a TriggerType.  
*Please note, the Function must always Return*  
 Examples:  
 Create a simple Callback with no arguments:

## Example

```
gap> MyFunction := function() return "Hello World!"; end;
gap> callback := Callback(MyFunction);
gap> Id(callback);
```

Create a Callback with one required argument of type string:

## Example

```
gap> MyFunction := function(str) return Concatenation("Hello", " ", str); end;
gap> callback := Callback(MyFunction);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

Create a Callback with one known argument of type string:

## Example

```
gap> MyFunction := function(args) return args; end;
gap> callback := Callback(MyFunction, ["Hello World"]);
```

Create a Callback with one known argument and one required argument of type string:

## Example

```
gap> MyFunction := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(MyFunction, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

Create a Callback with one known argument and one required argument of type string and double click trigger Type:

## Example

```
gap> MyFunction := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(TriggerType.DOUBLE_CLICK, MyFunction, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> Add(callback, arg);
```

In order to see the output of the previous examples, we have to simulate the external call to Trigger operation:

## Example

```
gap> MyFunction := function(a,b) return Concatenation(a, b); end;
gap> callback := Callback(TriggerType.DOUBLE_CLICK, MyFunction, ["Hello "]);
gap> arg := RequiredArg(ArgType.STRING, "Your Name");
gap> SetTitle(arg, "Enter your name");
gap> Title(arg);
gap> Add(callback, arg);
gap> SetValue(arg, "Manuel"); # simulate the user input
gap> Value(arg);
gap> Trigger(GapToJsonString(Sanitize(callback))); # simulate the external trigger
```

Create a Noop Callback, useful for Menu holders, where no Function is required:

## Example

```
gap> callback := NoopCallback();
```



### 2.4.2 NoopCallback

- ▷ `NoopCallback()` (operation)  
**Returns:** Callback  
 Creates an empty Callback object that does nothing. Useful to create menu holders.

### 2.4.3 RequiredArg (for IsArgType, IsString)

- ▷ `RequiredArg(IsArgType, IsString(title))` (operation)  
**Returns:** RequiredArg  
 Creates a Callback with a RequiredArg. RequiredArg is user input driven and required for the execution of a Callback Function. The value for this argument will be provided by the user.

### 2.4.4 Trigger (for IsString)

- ▷ `Trigger(IsString(JSON))` (operation)  
**Returns:** the result of the execution of the Callback defined Function  
 Triggers a Callback Function in GAP. Gets a JSON String object representation of the callback to execute.

### 2.4.5 Add (for IsCallback, IsRequiredArg)

- ▷ `Add(IsCallback[, IsRequiredArg, List(IsRequiredArg)])` (operation)  
**Returns:** Callback  
 Adds a RequiredArg to a specific Callback.

### 2.4.6 Remove (for IsCallback, IsRequiredArg)

- ▷ `Remove(IsCallback[, IsRequiredArg, List(IsRequiredArg)])` (operation)  
**Returns:** Callback  
 Removes a RequiredArg from a specific Callback.

## 2.5 Globals

In this section we show the Global Callback Francy Records for multi purpose.

## 2.6 Attributes

In this section we show the Francy Callback Attributes

### 2.6.1 Title (for IsRequiredArg)

- ▷ `Title(arg)` (attribute)  
**Returns:** IsString with the title of the object  
 A title on a RequiredArg is used to retrieve input from a user.

### 2.6.2 Title (for IsRequiredArg)

▷ Title(*arg1*) (operation)

### 2.6.3 SetTitle (for IsRequiredArg, IsString)

▷ SetTitle(*IsRequiredArg*, *IsString*) (operation)

Sets the title of the RequiredArg.

### 2.6.4 Value (for IsRequiredArg)

▷ Value(*arg*) (attribute)

**Returns:** IsString with the value of the object

A value on a RequiredArg is the actual input value to be passed to back gap from the client GUI. These values are stored as String for convenience, even if the ArgType specified for the RequiredArg is of another type. Hence, explicit conversion is required within the Callback Function.

### 2.6.5 Value (for IsRequiredArg)

▷ Value(*arg1*) (operation)

### 2.6.6 SetValue (for IsRequiredArg, IsString)

▷ SetValue(*IsRequiredArg*, *IsString*) (operation)

Sets the value of the RequiredArg.

### 2.6.7 ConfirmMessage (for IsCallback)

▷ ConfirmMessage(*arg*) (attribute)

**Returns:** a IsString with the message to be shown to the user before the Callback execution.

This will display a confirmation message before any Callback is executed.

### 2.6.8 ConfirmMessage (for IsCallback)

▷ ConfirmMessage(*arg1*) (operation)

### 2.6.9 SetConfirmMessage (for IsCallback, IsString)

▷ SetConfirmMessage(*IsRequiredArg*, *IsString*) (operation)

Sets the value of the confirmation message to display to the user.

## Chapter 3

# Francy Canvas

A Canvas is an area where the graphics representation of Francy live.  
Please see Francy-JS for client implementation.

### 3.1 Categories

In this section we show all Francy Canvas Categories.

#### 3.1.1 IsCanvas (for IsFrancyObject)

▷ `IsCanvas(arg)` (filter)  
**Returns:** true or false  
Identifies Canvas objects.

#### 3.1.2 IsCanvasDefaults (for IsFrancyDefaultObject)

▷ `IsCanvasDefaults(arg)` (filter)  
**Returns:** true or false  
Identifies CanvasDefaults objects.

### 3.2 Families

In this section we show all Francy Canvas Families.

### 3.3 Representations

In this section we show all Francy Canvas Representations.

#### 3.3.1 IsCanvasRep (for IsComponentObjectRep)

▷ `IsCanvasRep(arg)` (filter)  
**Returns:** true or false  
Checks whether an Object has a Canvas internal representation.

### 3.3.2 IsCanvasDefaultsRep (for IsComponentObjectRep)

- ▷ `IsCanvasDefaultsRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a CanvasDefaults internal representation.

## 3.4 Operations

In this section we show all Francy Canvas Operations.

### 3.4.1 Canvas (for IsString, IsCanvasDefaults)

- ▷ `Canvas(IsString(title)[, IsCanvasDefaults])` (operation)  
**Returns:** Canvas  
 A Canvas represents the base element where rendering happens. This object is inspired by the HTML canvas tag element, which is used to draw graphics in runtime via JavaScript. Examples:  
 Create a simple Canvas:

Example

```
gap> canvas := Canvas("");
gap> Id(canvas);
gap> SetTitle(canvas, "Quaternion Group Subgroup Lattice");
gap> Title(canvas);
gap> SetHeight(canvas, 400); # default 600
gap> Height(canvas);
gap> SetWidth(canvas, 400); # default 800
gap> Width(canvas);
gap> SetZoomToFit(canvas, false); # default true
gap> ZoomToFit(canvas);
gap> Draw(canvas);
```

### 3.4.2 Add (for IsCanvas, IsFrancyGraph)

- ▷ `Add(IsCanvas[, IsFrancyGraph, List(IsFrancyGraph)])` (operation)  
**Returns:** Canvas  
 Adds a FrancyGraph to a specific Canvas. Well, the api is abstract enough to allow Adding a list of IsFrancyGraph to a canvas, but this results in setting the graph property only to the last IsFrancyGraph in the list.

### 3.4.3 Remove (for IsCanvas, IsFrancyGraph)

- ▷ `Remove(IsCanvas[, IsFrancyGraph, List(IsFrancyGraph)])` (operation)  
**Returns:** Canvas  
 Removes a FrancyGraph from a Canvas.

### 3.4.4 Add (for IsCanvas, IsChart)

- ▷ `Add(IsCanvas[, IsChart, List(IsChart)])` (operation)  
**Returns:** Canvas

Adds a Chart to a specific Canvas. Well, the api is abstract enough to allow Adding a list of IsChart to a canvas, but this results in setting the graph property only to the last IsChart in the list.

### 3.4.5 Remove (for IsCanvas, IsChart)

- ▷ `Remove(IsCanvas[, IsChart, List(IsChart)])` (operation)  
**Returns:** Canvas  
 Removes a Chart from a Canvas.

### 3.4.6 Add (for IsCanvas, IsMenu)

- ▷ `Add(IsCanvas[, IsMenu, List(IsMenu)])` (operation)  
**Returns:** Canvas  
 Adds a Menu to a specific Canvas.

### 3.4.7 Remove (for IsCanvas, IsMenu)

- ▷ `Remove(IsCanvas[, IsMenu, List(IsMenu)])` (operation)  
**Returns:** Canvas  
 Removes a Menu from a Canvas.

### 3.4.8 Add (for IsCanvas, IsFrancyMessage)

- ▷ `Add(IsCanvas[, IsFrancyMessage, List(IsFrancyMessage)])` (operation)  
**Returns:** IsCanvas  
 Adds a FrancyMessage to a specific IsCanvas.

### 3.4.9 Remove (for IsCanvas, IsFrancyMessage)

- ▷ `Remove(IsCanvas[, IsFrancyMessage, List(IsFrancyMessage)])` (operation)  
**Returns:** IsCanvas  
 Removes a FrancyMessage from a specific IsCanvas.

### 3.4.10 Add (for IsCanvas, IsFrancyRenderer)

- ▷ `Add(IsCanvas[, IsFrancyRenderer, List(IsFrancyRenderer)])` (operation)  
**Returns:** IsCanvas  
 Adds a FrancyRenderer to a specific IsCanvas.

### 3.4.11 Remove (for IsCanvas, IsFrancyRenderer)

- ▷ `Remove(IsCanvas[, IsFrancyRenderer, List(IsFrancyRenderer)])` (operation)  
**Returns:** IsCanvas  
 Removes a FrancyRenderer from a specific IsCanvas.

### 3.4.12 Draw (for IsCanvas)

▷ `Draw(IsCanvas)` (operation)

**Returns:** `rec` with the JSON metadata model representation of the `Canvas`

Generates the JSON metadata model representation of the `Canvas` object and all children objects.

### 3.4.13 DrawSplash (for IsCanvas)

▷ `DrawSplash(IsCanvas)` (operation)

**Returns:** `IsString` with HTML generated

Generates an HTML page with an offline version of Francy JS and opens it within the default browser of the system.

## 3.5 Global

In this section we show all Global Francy Canvas Records for multi purpose.

## 3.6 Attributes

In this section we show the Francy Attributes.

### 3.6.1 Width (for IsCanvas)

▷ `Width(arg)` (attribute)

**Returns:** `IsPosInt`

The Width of the `Canvas` in pixels `IsPosInt`.

### 3.6.2 Width (for IsCanvas)

▷ `Width(arg1)` (operation)

### 3.6.3 SetWidth (for IsCanvas, IsPosInt)

▷ `SetWidth(IsCanvas, IsPosInt)` (operation)

Sets the Width of the `Canvas` in pixels `IsPosInt`.

### 3.6.4 Height (for IsCanvas)

▷ `Height(arg)` (attribute)

**Returns:** `IsPosInt`

The Height of the `Canvas` in pixels `IsPosInt`.

### 3.6.5 Height (for IsCanvas)

▷ `Height(arg1)` (operation)

### 3.6.6 SetHeight (for IsCanvas, IsPosInt)

▷ `SetHeight(IsCanvas, IsPosInt)` (operation)

Sets the Height of the Canvas in pixels IsPosInt.

### 3.6.7 ZoomToFit (for IsCanvas)

▷ `ZoomToFit(arg)` (attribute)

**Returns:** `IsBool` True if enabled otherwise False

`ZoomToFit` is a property that sets the objects within the Canvas to fit within the GUI visible area, after rendering in the client implementation.

### 3.6.8 ZoomToFit (for IsCanvas)

▷ `ZoomToFit(arg1)` (operation)

### 3.6.9 SetZoomToFit (for IsCanvas, IsBool)

▷ `SetZoomToFit(IsCanvas, IsBool)` (operation)

`ZoomToFit` is a property that sets the objects within the Canvas to fit within the GUI visible area.

### 3.6.10 Title (for IsCanvas)

▷ `Title(arg)` (attribute)

**Returns:** `IsString` with the title of the object  
The Canvas title to show on the GUI.

### 3.6.11 Title (for IsCanvas)

▷ `Title(arg1)` (operation)

### 3.6.12 SetTitle (for IsCanvas, IsString)

▷ `SetTitle(IsCanvas, IsString)` (operation)

Sets the title of the Canvas.

### 3.6.13 TexTypesetting (for IsCanvas)

▷ `TexTypesetting(arg)` (attribute)

**Returns:** `IsBool` with the title of the object

Enables usage of TeX Typesetting on the client implementation, if supported.

### 3.6.14 **TexTypesetting (for IsCanvas)**

▷ `TexTypesetting(arg1)` (operation)

### 3.6.15 **SetTexTypesetting (for IsCanvas, IsBool)**

▷ `SetTexTypesetting(IsCanvas, IsBool)` (operation)

Sets TeX Typesetting on the canvas objects.



## Chapter 4

# Francy Charts

It is possible to build Charts with simple Datasets.  
Currently, Francy, supports Bar, Line and Scatter charts.  
Please see Francy-JS for client implementation.

### 4.1 Categories

In this section we show all Francy Chart Categories.

#### 4.1.1 IsChart (for IsFrancyObject)

▷ `IsChart(arg)` (filter)  
**Returns:** true or false  
Identifies Chart objects.

#### 4.1.2 IsChartType (for IsFrancyTypeObject)

▷ `IsChartType(arg)` (filter)  
**Returns:** true or false  
Identifies ChartType objects.

#### 4.1.3 IsChartDefaults (for IsFrancyDefaultObject)

▷ `IsChartDefaults(arg)` (filter)  
**Returns:** true or false  
Identifies ChartDefaults objects.

#### 4.1.4 IsAxisScaleType (for IsFrancyTypeObject)

▷ `IsAxisScaleType(arg)` (filter)  
**Returns:** true or false  
Identifies AxisScaleType objects.

### 4.1.5 IsXAxis (for IsFrancyObject)

- ▷ `IsXAxis(arg)` (filter)  
**Returns:** true or false  
 Identifies XAxis objects.

### 4.1.6 IsYAxis (for IsFrancyObject)

- ▷ `IsYAxis(arg)` (filter)  
**Returns:** true or false  
 Identifies YAxis objects.

### 4.1.7 IsDataset (for IsFrancyObject)

- ▷ `IsDataset(arg)` (filter)  
**Returns:** true or false  
 Identifies Dataset objects.

## 4.2 Families

In this section we show all Francy Chart Families.

## 4.3 Representations

In this section we show the Francy Chart Representations.

### 4.3.1 IsChartRep (for IsComponentObjectRep)

- ▷ `IsChartRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a Chart internal representation.

### 4.3.2 IsChartDefaultsRep (for IsComponentObjectRep)

- ▷ `IsChartDefaultsRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a ChartDefaults internal representation.

### 4.3.3 IsChartTypeRep (for IsComponentObjectRep)

- ▷ `IsChartTypeRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a ChartType internal representation.

#### 4.3.4 IsAxisScaleTypeRep (for IsComponentObjectRep)

- ▷ IsAxisScaleTypeRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a AxisScaleType internal representation.

#### 4.3.5 IsAxisRep (for IsComponentObjectRep)

- ▷ IsAxisRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a AxisRep internal representation.

#### 4.3.6 IsDatasetRep (for IsComponentObjectRep)

- ▷ IsDatasetRep(*arg*) (filter)  
**Returns:** true or false  
 Checks whether an Object has a DatasetRep internal representation.

### 4.4 Operations

In this section we show all Francy Chart Operations.

#### 4.4.1 Chart (for IsChartType, IsChartDefaults)

- ▷ Chart(*IsChartType*[, *IsChartDefaults*]) (operation)  
**Returns:** Chart  
 Every object will be a subclass of Chart object. All objects contain the same base information.  
 Examples:  
 Create a simple Chart of type ChartType.BAR:

Example

```
gap> chart:=Chart(ChartType.BAR);
gap> SetAxisXTitle(chart, "X Axis");
gap> AxisXTitle(chart);
gap> SetAxisXDomain(chart, ["domain1", "domain2", "domain3", "domain4", "domain5"]); # default []
gap> AxisXDomain(chart);
gap> SetAxisYTitle(chart, "Y Axis");
gap> AxisYTitle(chart);
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> data2 := Dataset("data2", [51,60,72,38,97]);
gap> data3 := Dataset("data3", [50,60,70,80,90]);
gap> Add(chart, [data1, data2, data3]);
gap> Remove(chart, data1);
gap> Add(chart, data1);
gap> Remove(chart, [data2, data3]);
gap> Length(RecNames(chart!.data)) = 1;
```

Create a simple Chart of type ChartType.LINE:

## Example

```
gap> chart:=Chart(ChartType.LINE);
gap> SetAxisXTitle(chart, "X Axis");
gap> SetAxisYTitle(chart, "Y Axis");
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> Add(chart, data1);
```

Create a simple Chart of type `ChartType.SCATTER`:

## Example

```
gap> chart:=Chart(ChartType.SCATTER);
gap> SetAxisXTitle(chart, "X Axis");
gap> SetAxisYTitle(chart, "Y Axis");
gap> data1 := Dataset("data1", [100,20,30,47,90]);
gap> Add(chart, data1);
```

#### 4.4.2 Add (for IsChart, IsDataset)

- ▷ `Add(IsChart[, IsDataset, List(IsDataset)])` (operation)  
**Returns:** Chart  
 Adds a Dataset to a specific Chart.

#### 4.4.3 Remove (for IsChart, IsDataset)

- ▷ `Remove(IsChart[, IsDataset, List(IsDataset)])` (operation)  
**Returns:** Chart  
 Removes a Dataset from a specific Chart.

#### 4.4.4 Dataset (for IsString, IsList)

- ▷ `Dataset(IsString(title), IsList(data))` (operation)  
**Returns:** Dataset  
 Creates a dataset.

#### 4.4.5 DefaultAxis (for IsChartType)

- ▷ `DefaultAxis(IsChartType)` (operation)  
**Returns:** rec  
 Returns the default settings for a ChartType

#### 4.4.6 XAxis (for IsAxisScaleType, IsString, IsList)

- ▷ `XAxis(IsAxisScaleType, IsString(title), IsList(domain))` (operation)  
**Returns:** XAxis  
 Creates a XAxis

#### 4.4.7 YAxis (for IsAxisScaleType, IsString, IsList)

- ▷ `YAxis(IsAxisScaleType, IsString(title), IsList(domain))` (operation)  
**Returns:** YAxis  
 Creates a YAxis

## 4.5 Global

In this section we show all Global Chart Francy Records for multi purpose.

## 4.6 Attributes

In this section we show all Francy Attributes

### 4.6.1 ShowLegend (for IsChart)

▷ `ShowLegend(arg)` (attribute)

**Returns:** `IsBool` True if enabled otherwise False

`ShowLegend` is a property that enables or disables displaying the Chart legend in the client implementation.

### 4.6.2 ShowLegend (for IsChart)

▷ `ShowLegend(arg1)` (operation)

### 4.6.3 SetShowLegend (for IsChart, IsBool)

▷ `SetShowLegend(IsChart, IsBool)` (operation)

`ShowLegend` is a property that enables or disables displaying the Chart legend in the client implementation.

### 4.6.4 AxisXTitle (for IsChart)

▷ `AxisXTitle(arg)` (attribute)

**Returns:** `IsString` with the title of the object

This is used to display the X Axis Title in the client implementation.

### 4.6.5 AxisXTitle (for IsChart)

▷ `AxisXTitle(arg1)` (operation)

### 4.6.6 SetAxisXTitle (for IsChart, IsString)

▷ `SetAxisXTitle(IsChart, IsString)` (operation)

This is used to display the X Axis Title in the client implementation.

#### 4.6.7 AxisYTitle (for IsChart)

▷ AxisYTitle(*arg*) (attribute)

**Returns:** IsString with the title of the object

This is used to display the Y Axis Title in the client implementation.

#### 4.6.8 AxisYTitle (for IsChart)

▷ AxisYTitle(*arg1*) (operation)

#### 4.6.9 SetAxisYTitle (for IsChart, IsString)

▷ SetAxisYTitle(*IsChart*, *IsString*) (operation)

This is used to display the Y Axis Title in the client implementation.

#### 4.6.10 AxisXDomain (for IsChart)

▷ AxisXDomain(*arg*) (attribute)

**Returns:** IList

This is the domain of the X Axis values in the client implementation.

#### 4.6.11 AxisXDomain (for IsChart)

▷ AxisXDomain(*arg1*) (operation)

#### 4.6.12 SetAxisXDomain (for IsChart, IList)

▷ SetAxisXDomain(*IList*, *IList*) (operation)

This is the domain of the X Axis values in the client implementation.

## Chapter 5

# Francy Core

Francy is responsible for generating JSON metadata models allowing external tools / libraries / frameworks to add visual representations to math structures. This JSON model representation defines the contract between this package (server) and a GUI framework (client), enabling complete SoC (Separation of Concerns). Francy can be used to provide graphical interactive environments on existing GAP packages.

A JSON schema is available and can be used to produce compliant clients for this package. *See schema/francy.json*

To map required and optional attributes from the schema into GAP code, the implementation follows the following criteria:

- Object creation requests mandatory attributes, i.e. required with no default value, e.g. `canvas:=Canvas("Title")`
- Object creation accepts an object of defaults, i.e. default values for mandatory fields but that might repeat throughout the creation of multiple similar objects, e.g. `defaults:=DefaultCanvas; defaults!.zoomToFit:=false; canvas:=Canvas("Title",defaults);` Where `DefaultCanvas` contains defaults for width (800) and height (600)
- Attributes associated with the object that can be set, i.e. optional with no defaults, e.g. `canvas:=Canvas("Title"); SetTexTypesetting(canvas,true);`

The API follows common GAP naming conventions and adding objects is done using `Add(objectHolder,object)` and removing object is done using `Remove(objectHolder,object)`.

Although Francy has the concept of a Graph, it does not implement any Mathematics Graphs Theory.

Please see Francy-JS for client implementation.

### 5.1 Categories

In this section we show all Francy Core Categories.

#### 5.1.1 IsFrancyObject (for IsObject)

- ▷ `IsFrancyObject(arg)` (filter)  
**Returns:** true or false  
Identifies all Objects in Francy.

### 5.1.2 IsFrancyDefaultObject (for IsObject)

- ▷ `IsFrancyDefaultObject(arg)` (filter)  
**Returns:** true or false  
 Identifies all Default record objects in Francy.

### 5.1.3 IsFrancyTypeObject (for IsObject)

- ▷ `IsFrancyTypeObject(arg)` (filter)  
**Returns:** true or false  
 Identifies all Type record objects in Francy.

## 5.2 Families

In this section we show all Francy Core Families.

## 5.3 Global

In this section we show all Francy Core Types

## 5.4 Attributes

In this section we show all Francy Core Attributes

### 5.4.1 FrancyId (for IsFrancyObject)

- ▷ `FrancyId(arg)` (attribute)  
**Returns:** `IsString` with the unique identifier of the object  
 All Objects created in Francy have a generated unique identifier.

### 5.4.2 FrancyId (for IsFrancyObject)

- ▷ `FrancyId(arg1)` (operation)  
**Returns:** a `IsString` with the unique id of the object  
 Prints the object unique identifier.

### 5.4.3 SetFrancyId (for IsFrancyObject, IsString)

- ▷ `SetFrancyId(o, s)` (operation)

Use with care! Changing the unique ID might be useful in certain cases, but bear in mind it might cause unexpected behavior. Use at your own risk.



## Chapter 6

# Francy Graphs

A graph is visual representation or a diagram that represents data or values in an organized mode. With Francy, it is possible to build direct and indirect Graphs.

Please see examples section.

Please see Francy-JS for client implementation.

### 6.1 Categories

In this section we show all Francy Graph Categories.

#### 6.1.1 IsFrancyGraph (for IsFrancyObject)

▷ IsFrancyGraph(*arg*) (filter)  
**Returns:** true or false  
Identifies Graph objects.

#### 6.1.2 IsFrancyGraphType (for IsFrancyObject)

▷ IsFrancyGraphType(*arg*) (filter)  
**Returns:** true or false  
Identifies GraphType objects.

#### 6.1.3 IsFrancyGraphDefaults (for IsFrancyDefaultObject)

▷ IsFrancyGraphDefaults(*arg*) (filter)  
**Returns:** true or false  
Identifies GraphDefaults objects.

#### 6.1.4 IsShape (for IsFrancyObject)

▷ IsShape(*arg*) (filter)  
**Returns:** true or false  
Identifies Shape objects.

### 6.1.5 IsShapeType (for IsFrancyObject)

- ▷ `IsShapeType(arg)` (filter)  
**Returns:** true or false  
 Identifies ShapeType objects.

### 6.1.6 IsShapeDefaults (for IsFrancyDefaultObject)

- ▷ `IsShapeDefaults(arg)` (filter)  
**Returns:** true or false  
 Identifies ShapeDefaults objects.

### 6.1.7 IsLink (for IsFrancyObject)

- ▷ `IsLink(arg)` (filter)  
**Returns:** true or false  
 Identifies Link objects.

### 6.1.8 IsLinkDefaults (for IsFrancyDefaultObject)

- ▷ `IsLinkDefaults(arg)` (filter)  
**Returns:** true or false  
 Identifies LinkDefaults objects.

## 6.2 Families

In this section we show all Francy Graph Families.

## 6.3 Representations

In this section we show all Francy Graph Representations.

### 6.3.1 IsFrancyGraphRep (for IsComponentObjectRep)

- ▷ `IsFrancyGraphRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a Graph internal representation.

### 6.3.2 IsFrancyGraphDefaultsRep (for IsComponentObjectRep)

- ▷ `IsFrancyGraphDefaultsRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a GraphDefaults internal representation.

### 6.3.3 IsFrancyGraphTypeRep (for IsComponentObjectRep)

- ▷ `IsFrancyGraphTypeRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a GraphType internal representation.

### 6.3.4 IsShapeRep (for IsComponentObjectRep)

- ▷ `IsShapeRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a Shape internal representation.

### 6.3.5 IsShapeDefaultsRep (for IsComponentObjectRep)

- ▷ `IsShapeDefaultsRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a ShapeDefaults internal representation.

### 6.3.6 IsShapeTypeRep (for IsComponentObjectRep)

- ▷ `IsShapeTypeRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a ShapeType internal representation.

### 6.3.7 IsLinkRep (for IsComponentObjectRep)

- ▷ `IsLinkRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a Link internal representation.

### 6.3.8 IsLinkDefaultsRep (for IsComponentObjectRep)

- ▷ `IsLinkDefaultsRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a LinkDefaults internal representation.

## 6.4 Operations

In this section we show all Francy Graph Operations.

### 6.4.1 Graph (for IsFrancyGraphType, IsFrancyGraphDefaults)

- ▷ `Graph(IsFrancyGraphType[, IsFrancyGraphDefaults])` (operation)  
**Returns:** Graph  
 Every object will be a subclass of this Graph. All the objects contain the same base information.  
 Examples:  
 Create a simple Graph of type `GraphType.DIRECTED` with simple Shape and connected with Links:

## Example

```
gap> graph := Graph(GraphType.DIRECTED);
gap> SetSimulation(graph, false);
gap> shape1 := Shape(ShapeType.SQUARE);
gap> shape1!.layer := 1;
gap> shape2 := Shape(ShapeType.TRIANGLE);
gap> shape2!.layer := 3;
gap> link := Link(shape1, shape2);
gap> Add(graph, link);
gap> Add(graph, [shape1, shape2]);
```

Create a simple Graph of type GraphType.UNDIRECTED with simple Shape and with a TriggerEvent.RIGHT\_CLICK Callback:

## Example

```
gap> graph := Graph(GraphType.UNDIRECTED);
gap> shape := Shape(ShapeType.SQUARE);
gap> MyFunction := function() Add(graph, Shape(ShapeType.Circle)); return graph; end;
gap> callback := Callback(TriggerType.RIGHT_CLICK, MyFunction);
gap> Add(shape, callback);
gap> Add(graph, shape);
```

### 6.4.2 UnsetNodes (for IsFrancyGraph)

▷ UnsetNodes(*arg*) (operation)

Removes all nodes from a graph.

### 6.4.3 UnsetLinks (for IsFrancyGraph)

▷ UnsetLinks(*arg*) (operation)

Removes all nodes from a graph.

### 6.4.4 Add (for IsFrancyGraph, IsLink)

▷ Add(*IsFrancyGraph*[, *IsLink*, *List(IsLink)*]) (operation)

**Returns:** Graph

Add *IsLink* to a specific Graph.

### 6.4.5 Remove (for IsFrancyGraph, IsLink)

▷ Remove(*IsFrancyGraph*[, *IsLink*, *List(IsLink)*]) (operation)

**Returns:** Graph

Remove *IsLink* from a specific Graph.

### 6.4.6 Add (for IsFrancyGraph, IsShape)

▷ Add(*IsFrancyGraph*[, *IsShape*, *List(IsShape)*]) (operation)

**Returns:** Graph

Add *IsShape* to a specific Graph.

#### 6.4.7 Remove (for IsFrancyGraph, IsShape)

- ▷ `Remove(IsFrancyGraph[, IsShape, List(IsShape)])` (operation)  
**Returns:** Graph  
 Remove IsShape from a specific Graph.

#### 6.4.8 Shape (for IsShapeType, IsString, IsShapeDefaults)

- ▷ `Shape(IsShapeType[, IsString(title), IsShapeDefaults])` (operation)  
**Returns:** Shape  
 Every object will be a subclass of Shape object. All the objects contain the same base information.

#### 6.4.9 GetShape (for IsFrancyGraph, IsString)

- ▷ `GetShape(IsFrancyGraph, IsString)` (operation)  
**Returns:** Shape  
 Gets a Shape node from a Graph by ID.

#### 6.4.10 GetShapes (for IsFrancyGraph)

- ▷ `GetShapes(IsFrancyGraph, IsString)` (operation)  
**Returns:** List(Shape)  
 Gets a Shape node from a Graph by ID.

#### 6.4.11 Add (for IsShape, IsMenu)

- ▷ `Add(IsShape[, IsMenu, List(IsMenu)])` (operation)  
**Returns:** Shape  
 Add a Menu to a specific Shape.

#### 6.4.12 Remove (for IsShape, IsMenu)

- ▷ `Remove(IsShape[, IsMenu, List(IsMenu)])` (operation)  
**Returns:** Shape  
 Remove a Menu from a specific Shape.

#### 6.4.13 Add (for IsShape, IsCallback)

- ▷ `Add(IsShape[, IsCallback, List(IsCallback)])` (operation)  
**Returns:** Shape  
 Add a Callback to a specific Shape.

#### 6.4.14 Remove (for IsShape, IsCallback)

- ▷ `Remove(IsShape[, IsCallback, List(IsCallback)])` (operation)  
**Returns:** Shape  
 Remove a Callback from a specific Shape.

#### 6.4.15 Add (for IsShape, IsFrancyMessage)

- ▷ `Add(IsShape[, IsFrancyMessage, List(IsFrancyMessage)])` (operation)  
**Returns:** Shape  
 Add a Callback to a specific Shape.

#### 6.4.16 Remove (for IsShape, IsFrancyMessage)

- ▷ `Remove(IsShape[, IsFrancyMessage, List(IsFrancyMessage)])` (operation)  
**Returns:** Shape  
 Remove a Callback from a specific Shape.

#### 6.4.17 Link (for IsShape, IsShape, IsLinkDefaults)

- ▷ `Link(IsShape, IsShape)` (operation)  
**Returns:** Link  
 Creates a Link or edge between the two Shape.

#### 6.4.18 Links (for IsList, IsList, IsLinkDefaults)

- ▷ `Links(List(IsShape), List(IsShape))` (operation)  
**Returns:** List(Link)  
 Creates a Link or edge between the Shape of the first list and the Shape of the second list.

#### 6.4.19 GetLink (for IsFrancyGraph, IsString)

- ▷ `GetLink(IsFrancyGraph, IsString)` (operation)  
**Returns:** Link  
 Gets a Link or edge from a graph by ID.

#### 6.4.20 GetLinks (for IsFrancyGraph)

- ▷ `GetLinks(IsFrancyGraph, IsString)` (operation)  
**Returns:** List(Link)  
 Gets a Link or edge from a graph.

### 6.5 Global

In this section we show all Global Callback Francy Records for multi purpose.

### 6.6 Attributes

In this section we show all Francy Core Attributes

### 6.6.1 Title (for IsShape)

▷ Title(*arg*) (attribute)

**Returns:** IsString with the title of the object

Sets the Shape label title. Supports TeX syntax and will be Typeset, if supported by the client implementation.

### 6.6.2 Title (for IsShape)

▷ Title(*arg1*) (operation)

### 6.6.3 SetTitle (for IsShape, IsString)

▷ SetTitle(*IsRequiredArg*, *IsString*) (operation)

Sets the Shape label title.

### 6.6.4 Color (for IsShape)

▷ Color(*arg*) (attribute)

**Returns:** IsInt

The Color of the current Shape. This should be an hexadecimal colour value, e.g.: #ff0000

### 6.6.5 Color (for IsShape)

▷ Color(*arg1*) (operation)

### 6.6.6 SetColor (for IsShape, IsString)

▷ SetColor(*IsShape*, *IsString*) (operation)

Sets the Color value. This should be an hexadecimal colour value, e.g.: #ff0000

### 6.6.7 PosX (for IsShape)

▷ PosX(*arg*) (attribute)

**Returns:** IsInt

The Position in the X Axis of the Shape in the Canvas in pixels.

### 6.6.8 PosX (for IsShape)

▷ PosX(*arg1*) (operation)

### 6.6.9 SetPosX (for IsShape, IsInt)

▷ SetPosX(*IsShape*, *IsInt*) (operation)

Sets the Position in the X Axis of the Shape in the Canvas in pixels.

### 6.6.10 PosY (for IsShape)

▷ PosY(*arg*) (attribute)

**Returns:** IsInt

The Position in the Y Axis of the Shape in the Canvas in pixels.

### 6.6.11 PosY (for IsShape)

▷ PosY(*arg1*) (operation)

### 6.6.12 SetPosY (for IsShape, IsInt)

▷ SetPosY(*IsShape*, *IsInt*) (operation)

Sets the Position in the Y Axis of the Shape in the Canvas in pixels.

### 6.6.13 Size (for IsShape)

▷ Size(*arg*) (attribute)

**Returns:** IsPosInt

The Size of the Shape in pixels.

### 6.6.14 Size (for IsShape)

▷ Size(*arg1*) (operation)

### 6.6.15 SetSize (for IsShape, IsPosInt)

▷ SetSize(*IsShape*, *IsPosInt*) (operation)

Sets the Size of the Shape in pixels.

### 6.6.16 Layer (for IsShape)

▷ Layer(*arg*) (attribute)

**Returns:** IsInt

The Layer level in which the node will be placed. This property might also be used to apply a different color depending on the layer level. Depends on the client implementation.



### 6.6.17 Layer (for IsShape)

▷ `Layer(arg1)` (operation)

### 6.6.18 SetLayer (for IsShape, IsInt)

▷ `SetLayer(IsShape, IsInt)` (operation)

Sets the Layer number on a Shape.

### 6.6.19 ParentShape (for IsShape)

▷ `ParentShape(arg)` (attribute)

**Returns:** `IsShape`

The `ParentShape` in which the node will be placed. This property might also be used to apply a different color depending on the parent level. Depends on the client implementation.

### 6.6.20 ParentShape (for IsShape)

▷ `ParentShape(arg1)` (operation)

### 6.6.21 SetParentShape (for IsShape, IsShape)

▷ `SetParentShape(IsShape, IsShape)` (operation)

Sets the `ParentShape` on a Shape.

### 6.6.22 Simulation (for IsFrancyGraph)

▷ `Simulation(arg)` (attribute)

**Returns:** `IsBool` True if enabled otherwise False

`Simulation` is a property that sets the simulation behavior by applying forces to organize the graphics, without the need to provide custom positions on the rendered GUI. Depends on the client implementation.

### 6.6.23 Simulation (for IsFrancyGraph)

▷ `Simulation(arg1)` (operation)

### 6.6.24 SetSimulation (for IsFrancyGraph, IsBool)

▷ `SetSimulation(IsCanvas, IsBool)` (operation)

Sets the `Simulation` behavior, as per de description above.

### 6.6.25 Collapsed (for IsFrancyGraph)

▷ Collapsed(*arg*) (attribute)

**Returns:** IsBool True if enabled otherwise False

Collapsed is a property that sets Graph Tree structures to fold by default on the rendered GUI.

### 6.6.26 Collapsed (for IsFrancyGraph)

▷ Collapsed(*arg1*) (operation)

### 6.6.27 SetCollapsed (for IsFrancyGraph, IsBool)

▷ SetCollapsed(*IsCanvas*, *IsBool*) (operation)

Sets the Collapsed behavior, as per the description above.

### 6.6.28 Selected (for IsShape)

▷ Selected(*arg*) (attribute)

**Returns:** IsBool True if enabled otherwise False

Selected is a property that sets Shape objects as selected by default on the rendered GUI.

### 6.6.29 Selected (for IsShape)

▷ Selected(*arg1*) (operation)

### 6.6.30 SetSelected (for IsShape, IsBool)

▷ SetSelected(*IsCanvas*, *IsBool*) (operation)

Sets the Selected behavior, as per the description above.

### 6.6.31 ConjugateId (for IsShape)

▷ ConjugateId(*arg*) (attribute)

**Returns:** IsBool True if enabled otherwise False

ConjugateId is a property that is used to group Shape objects by default on the rendered GUI.

### 6.6.32 ConjugateId (for IsShape)

▷ ConjugateId(*arg1*) (operation)

### 6.6.33 SetConjugateId (for IsShape, IsInt)

▷ SetConjugateId(*IsCanvas*, *IsBool*) (operation)

Sets the Conjugate behavior, as per the description above.

### 6.6.34 Weight (for IsLink)

▷ Weight(*arg*) (attribute)

**Returns:** IsInt

The Weight of the current Link or edge.

### 6.6.35 Weight (for IsLink)

▷ Weight(*arg1*) (operation)

### 6.6.36 SetWeight (for IsLink, IsInt)

▷ SetWeight(*IsLink*, *IsInt*) (operation)

Sets the Weight value on a Link or edge.

### 6.6.37 Length (for IsLink)

▷ Length(*arg*) (attribute)

**Returns:** IsInt

The Length of the current Link or edge.

### 6.6.38 Length (for IsLink)

▷ Length(*arg1*) (operation)

### 6.6.39 SetLength (for IsLink, IsInt)

▷ SetLength(*IsLink*, *IsInt*) (operation)

Sets the Length value on a Link or edge.

### 6.6.40 Invisible (for IsLink)

▷ Invisible(*arg*) (attribute)

**Returns:** IsBoolean

The Invisible property of the current Link or edge.

**6.6.41 Invisible (for IsLink)**

▷ `Invisible(arg1)` (operation)

**6.6.42 SetInvisible (for IsLink, IsBool)**

▷ `SetInvisible(IsLink, IsBool)` (operation)

Sets the `Invisible` property value on a `Link` or `edge`.

**6.6.43 Color (for IsLink)**

▷ `Color(arg)` (attribute)

**Returns:** `IsInt`

The `Color` of the current `Link` or `edge`.

**6.6.44 Color (for IsLink)**

▷ `Color(arg1)` (operation)

**6.6.45 SetColor (for IsLink, IsString)**

▷ `SetColor(IsShape, IsString)` (operation)

Sets the `Color` value on a `Link` or `edge`.

**6.6.46 Title (for IsLink)**

▷ `Title(arg)` (attribute)

**Returns:** `IsInt`

The `Title` of the current `Link` or `edge`.

**6.6.47 Title (for IsLink)**

▷ `Title(arg1)` (operation)

**6.6.48 SetTitle (for IsLink, IsString)**

▷ `SetTitle(IsShape, IsString)` (operation)

Sets the `Title` value on a `Link` or `edge`.

## Chapter 7

# Francy Menus

Menus are lists of actions that are rendered on the GUI. Menus can have SubMenus, and are constituted by a label Title and an action defined as a Callback.

Please see Francy-JS for client implementation.

### 7.1 Categories

In this section we show all Francy Menu Categories.

#### 7.1.1 IsMenu (for IsFrancyObject)

▷ `IsMenu(arg)` (filter)  
**Returns:** true or false  
Identifies Menu objects.

### 7.2 Families

In this section we show all Francy Menu Families.

### 7.3 Representations

In this section we show all Francy Menu Representations.

#### 7.3.1 IsMenuRep (for IsComponentObjectRep)

▷ `IsMenuRep(arg)` (filter)  
**Returns:** true or false  
Checks whether an Object has a Menu internal representation.

### 7.4 Operations

In this section we show all Francy Menu Operations.

### 7.4.1 Menu (for IsString, IsCallback)

▷ `Menu(IsString(title)[, IsCallback])` (operation)

**Returns:** Menu

Creates a Menu with a label title and an action Callback. Is up to the client implementation to sort out the Menu and invoke the Callback. Examples:

Create a FrancyMenu:

Example

```
gap> canvas := Canvas("Callbacks in action");
gap> SetHeight(canvas, 100);
gap> graph := Graph(GraphType.HASSE);
gap> shape := Shape(ShapeType.CIRCLE);
gap> Add(graph, shape);
gap> Add(canvas, graph);
gap> HelloWorld := function(name)
>   Add(canvas, FrancyMessage(Concatenation("Hello, ", name)));
gap>   return Draw(canvas);
gap> end;
gap> callback1 := Callback(HelloWorld);
gap> arg1 := RequiredArg(ArgType.STRING, "Your Name?");
gap> Add(callback1, arg1);
gap> menu := Menu("Example Menu Holder");
gap> menu1 := Menu("Hello Menu Action", callback1 );
gap> menu2 := Menu("Hello Menu Action", callback1 );
gap> Add(menu, menu1);
gap> Remove(menu, menu1);
gap> Add(menu, [menu1, menu2]);
gap> Remove(menu, [menu1, menu2]);
gap> Add(canvas, [menu, menu1]);
gap> Remove(canvas, menu1);
gap> Add(canvas, menu1);
gap> Add(shape, menu1);
gap> Remove(shape, menu1);
gap> Add(shape, [menu1, menu2]);
gap> Remove(shape, [menu1, menu2]);
```

### 7.4.2 Add (for IsMenu, IsMenu)

▷ `Add(IsMenu[, IsMenu, List(IsMenu)])` (operation)

**Returns:** Menu

Add one Menu to a specific Menu creating a Submenu. Is up to the client implementation to handle this.

### 7.4.3 Remove (for IsMenu, IsMenu)

▷ `Remove(IsMenu[, IsMenu, List(IsMenu)])` (operation)

**Returns:** Menu

Remove a Menu from a specific Menu. Is up to the client implementation to handle this.

## 7.5 Attributes

In this section we show all Francy Core Attributes

### 7.5.1 Title (for IsMenu)

▷ `Title(arg)` (attribute)

**Returns:** `IsString` with the title of the object

A label title on a Menu is used to identify what action it is about on a menu entry.

### 7.5.2 Title (for IsMenu)

▷ `Title(arg1)` (operation)

### 7.5.3 SetTitle (for IsMenu, IsString)

▷ `SetTitle(IsMenu, IsString)` (operation)

Sets the title of the Menu.

## Chapter 8

# Francy Messages

FrancyMessage is an object that holds a message.

These messages can be used to provide information to users in the form of SUCCESS, INFO, WARNING, ERROR. It is up to the client implementation to handle these messages and their types in a fashionable manner. Please see Francy-JS for client implementation.

### 8.1 Categories

In this section we show all Francy FrancyMessage Categories.

#### 8.1.1 IsFrancyMessage (for IsFrancyObject)

▷ IsFrancyMessage(*arg*) (filter)  
**Returns:** true or false  
Identifies FrancyMessage objects.

#### 8.1.2 IsFrancyMessageType (for IsFrancyObject)

▷ IsFrancyMessageType(*arg*) (filter)  
**Returns:** true or false  
Identifies MessageType objects.

### 8.2 Families

In this section we show all Francy FrancyMessage Families.

### 8.3 Representations

In this section we show all Francy FrancyMessage Representations.

#### 8.3.1 IsFrancyMessageRep (for IsComponentObjectRep)

▷ IsFrancyMessageRep(*arg*) (filter)  
**Returns:** true or false



Checks whether an Object has a FrancyMessage internal representation.

### 8.3.2 IsFrancyMessageTypeRep (for IsComponentObjectRep)

- ▷ `IsFrancyMessageTypeRep(arg)` (filter)
- Returns:** true or false
- Checks whether an Object has a FrancyMessage internal representation.

## 8.4 Operations

In this section we show all Francy FrancyMessage Operations.

### 8.4.1 FrancyMessage (for IsFrancyMessageType, IsString, IsString)

- ▷ `FrancyMessage(IsString, IsString)` (operation)
- Returns:** FrancyMessage
- Adds an info label with the format "label: value"
- Examples:
- Create FrancyMessage of all types within a canvas:

Example

```
gap> canvas := Canvas("Example Canvas / Shape with Messages");
gap> graph := Graph(GraphType.HASSE); # will go throughout graphs later
gap> shape := Shape(ShapeType.CIRCLE); # will go throughout shapes later
gap> Add(graph, shape);
gap> Add(canvas, graph);
gap> Add(canvas, FrancyMessage(FrancyMessageType.INFO, "Hello"));
gap> Add(shape, FrancyMessage(FrancyMessageType.INFO, "Hello"));
gap> Add(canvas, FrancyMessage(FrancyMessageType.ERROR, "Oops", "Hello"));
gap> Add(shape, FrancyMessage(FrancyMessageType.ERROR, "Oops", "Hello"));
gap> Add(canvas, FrancyMessage(FrancyMessageType.WARNING, "Hello"));
gap> Add(shape, FrancyMessage(FrancyMessageType.WARNING, "Hello"));
gap> Add(canvas, FrancyMessage(FrancyMessageType.SUCCESS, "Hello"));
gap> Add(shape, FrancyMessage(FrancyMessageType.SUCCESS, "Hello"));
gap> Add(canvas, FrancyMessage("Hello", "World"));
gap> Add(shape, FrancyMessage("Hello", "World"));
```

## 8.5 Global

In this section we show all Global FrancyMessage Records for multi purpose.

## 8.6 Attributes

In this section we show all FrancyMessage Core Attributes

### 8.6.1 Title (for IsFrancyMessage)

▷ Title(*arg*) (attribute)

**Returns:** IsString with the title of the object

A title on a FrancyMessage is used to display the title information to the user.

### 8.6.2 Title (for IsFrancyMessage)

▷ Title(*arg1*) (operation)

### 8.6.3 SetTitle (for IsFrancyMessage, IsString)

▷ SetTitle(*IsFrancyMessage*, *IsString*) (operation)

Sets the title of the FrancyMessage.

### 8.6.4 Value (for IsFrancyMessage)

▷ Value(*arg*) (attribute)

**Returns:** IsString with the title of the object

A value on a FrancyMessage is used to display the information to the user.

### 8.6.5 Value (for IsFrancyMessage)

▷ Value(*arg1*) (operation)

### 8.6.6 SetValue (for IsFrancyMessage, IsString)

▷ SetValue(*IsFrancyMessage*, *IsString*) (operation)

Sets the actual message of the FrancyMessage.

### 8.6.7 Add (for IsFrancyMessage, IsCallback)

▷ Add(*IsFrancyMessage*[, *IsCallback*, *List(IsCallback)*]) (operation)

**Returns:** FrancyMessage

Add a Callback to a specific FrancyMessage.

### 8.6.8 Remove (for IsFrancyMessage, IsCallback)

▷ Remove(*IsFrancyMessage*[, *IsCallback*, *List(IsCallback)*]) (operation)

**Returns:** FrancyMessage

Remove a Callback from a specific FrancyMessage.

## Chapter 9

# Francy Renderers

`FrancyRenderer` is an object that holds the renderer name to be used by the client to display the graphics.

Francy JS allows users to switch between renderers in runtime, but if the renderer is specified in the GAP code, then the user won't be able to switch between renderers on the client side.

This implementation knows only about the official supported renderers: D3, Vis and Graphviz. Please see Francy-JS for client implementation.

### 9.1 Categories

In this section we show all Francy `FrancyRenderer` Categories.

#### 9.1.1 `IsFrancyRenderer` (for `IsFrancyObject`)

▷ `IsFrancyRenderer(arg)` (filter)  
**Returns:** true or false  
Identifies `FrancyRenderer` objects.

#### 9.1.2 `IsFrancyRendererType` (for `IsFrancyObject`)

▷ `IsFrancyRendererType(arg)` (filter)  
**Returns:** true or false  
Identifies `FrancyRendererType` objects.

### 9.2 Families

In this section we show all Francy `FrancyRenderer` Families.

### 9.3 Representations

In this section we show all Francy `FrancyRenderer` Representations.

### 9.3.1 IsFrancyRendererRep (for IsComponentObjectRep)

- ▷ `IsFrancyRendererRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a FrancyRenderer internal representation.

### 9.3.2 IsFrancyRendererTypeRep (for IsComponentObjectRep)

- ▷ `IsFrancyRendererTypeRep(arg)` (filter)  
**Returns:** true or false  
 Checks whether an Object has a FrancyRendererType internal representation.

## 9.4 Operations

In this section we show all Francy FrancyRenderer Operations.

### 9.4.1 FrancyRenderer (for IsFrancyRendererType)

- ▷ `FrancyRenderer(IsFrancyRendererType)` (operation)  
**Returns:** FrancyRenderer  
 Adds an info label with the format "label: value"  
 Examples:  
 Configure FrancyRendererType.VIS as the renderer for a specific canvas:

Example

```
gap> canvas := Canvas("Example Canvas / Shape with Messages");
gap> graph := Graph(GraphType.UNDIRECTED);
gap> Add(canvas, graph);
gap> vis := FrancyRenderer(FrancyRendererType.VIS);
gap> Add(canvas, vis);
```

## 9.5 Global

In this section we show all Global FrancyRendererType records for multi purpose.

## 9.6 Attributes

In this section we show all FrancyRenderer Core Attributes

### 9.6.1 Value (for IsFrancyRenderer)

- ▷ `Value(arg)` (attribute)  
**Returns:** IsString with the renderer of the object  
 A value on a FrancyRenderer is used to specify which renderer will be used to draw the graphics on the client.

### 9.6.2 Value (for IsFrancyRenderer)

▷ `Value(arg1)` (operation)

### 9.6.3 SetValue (for IsFrancyRenderer, IsFrancyRendererType)

▷ `SetValue(IsFrancyRenderer, IsString)` (operation)

Sets the actual renderer FrancyRenderer.

# Chapter 10

## Francy Util

Utility functions used across the package.

### 10.1 Operations

In this section we show all Francy Util Operations. Contains utility methods to handle Object printing/viewing, Sanitizing, etc.

#### 10.1.1 JUPYTER\_ViewString (for IsObject)

▷ `JUPYTER_ViewString(arg)` (operation)  
**Returns:** String  
This method will pretty print in jupyter environment.

#### 10.1.2 Sanitize (for IsObject)

▷ `Sanitize(IsObject)` (operation)  
**Returns:** `rec`  
This method will clone an `Object` and return a sanitized record. It traverses all the components sanitizing when appropriate. Sanitizing in this context means: replace everything that can't be converted into JSON, with its string representation!

#### 10.1.3 MergeObjects (for IsFrancyObject, IsFrancyObject)

▷ `MergeObjects(IsFrancyObject, IsFrancyObject)` (operation)  
**Returns:** `rec`  
This method will merge the properties of 2 `IsFrancyObjects` into one `rec`.

#### 10.1.4 GenerateID

▷ `GenerateID()` (operation)  
**Returns:** `IsString`  
This method will generate a sequential ID to be used as object identifier. These IDs are used to identify the objects between the client and the server, and are crucial for the communication between both.

# Index

- Add
  - for IsCallback, IsRequiredArg, 9
  - for IsCanvas, IsChart, 12
  - for IsCanvas, IsFrancyGraph, 12
  - for IsCanvas, IsFrancyMessage, 13
  - for IsCanvas, IsFrancyRenderer, 13
  - for IsCanvas, IsMenu, 13
  - for IsChart, IsDataset, 20
  - for IsFrancyGraph, IsLink, 28
  - for IsFrancyGraph, IsShape, 28
  - for IsFrancyMessage, IsCallback, 42
  - for IsMenu, IsMenu, 38
  - for IsShape, IsCallback, 29
  - for IsShape, IsFrancyMessage, 30
  - for IsShape, IsMenu, 29
- AxisXDomain
  - for IsChart, 22
- AxisXTitle
  - for IsChart, 21
- AxisYTitle
  - for IsChart, 22
- Callback
  - for IsTriggerType, IsFunction, IsList, 7
- Canvas
  - for IsString, IsCanvasDefaults, 12
- Chart
  - for IsChartType, IsChartDefaults, 19
- Collapsed
  - for IsFrancyGraph, 34
- Color
  - for IsLink, 36
  - for IsShape, 31
- ConfirmMessage
  - for IsCallback, 10
- ConjugateId
  - for IsShape, 34
- Dataset
  - for IsString, IsList, 20
- DefaultAxis
  - for IsChartType, 20
- Draw
  - for IsCanvas, 14
- DrawSplash
  - for IsCanvas, 14
- FrancyId
  - for IsFrancyObject, 24
- FrancyMessage
  - for IsFrancyMessageType, IsString, IsString, 41
- FrancyRenderer
  - for IsFrancyRendererType, 44
- GenerateID, 46
- GetLink
  - for IsFrancyGraph, IsString, 30
- GetLinks
  - for IsFrancyGraph, 30
- GetShape
  - for IsFrancyGraph, IsString, 29
- GetShapes
  - for IsFrancyGraph, 29
- Graph
  - for IsFrancyGraphType, IsFrancyGraphDe-  
faults, 27
- Height
  - for IsCanvas, 14
- Invisible
  - for IsLink, 35, 36
- IsArgType
  - for IsFrancyTypeObject, 6
- IsArgTypeRep
  - for IsComponentObjectRep, 7
- IsAxisRep
  - for IsComponentObjectRep, 19

- IsAxisScaleType
  - for IsFrancyTypeObject, [17](#)
- IsAxisScaleTypeRep
  - for IsComponentObjectRep, [19](#)
- IsCallback
  - for IsFrancyObject, [6](#)
- IsCallbackRep
  - for IsComponentObjectRep, [7](#)
- IsCanvas
  - for IsFrancyObject, [11](#)
- IsCanvasDefaults
  - for IsFrancyDefaultObject, [11](#)
- IsCanvasDefaultsRep
  - for IsComponentObjectRep, [12](#)
- IsCanvasRep
  - for IsComponentObjectRep, [11](#)
- IsChart
  - for IsFrancyObject, [17](#)
- IsChartDefaults
  - for IsFrancyDefaultObject, [17](#)
- IsChartDefaultsRep
  - for IsComponentObjectRep, [18](#)
- IsChartRep
  - for IsComponentObjectRep, [18](#)
- IsChartType
  - for IsFrancyTypeObject, [17](#)
- IsChartTypeRep
  - for IsComponentObjectRep, [18](#)
- IsDataset
  - for IsFrancyObject, [18](#)
- IsDatasetRep
  - for IsComponentObjectRep, [19](#)
- IsFrancyDefaultObject
  - for IsObject, [24](#)
- IsFrancyGraph
  - for IsFrancyObject, [25](#)
- IsFrancyGraphDefaults
  - for IsFrancyDefaultObject, [25](#)
- IsFrancyGraphDefaultsRep
  - for IsComponentObjectRep, [26](#)
- IsFrancyGraphRep
  - for IsComponentObjectRep, [26](#)
- IsFrancyGraphType
  - for IsFrancyObject, [25](#)
- IsFrancyGraphTypeRep
  - for IsComponentObjectRep, [27](#)
- IsFrancyMessage
  - for IsFrancyObject, [40](#)
- IsFrancyMessageRep
  - for IsComponentObjectRep, [40](#)
- IsFrancyMessageType
  - for IsFrancyObject, [40](#)
- IsFrancyMessageTypeRep
  - for IsComponentObjectRep, [41](#)
- IsFrancyObject
  - for IsObject, [23](#)
- IsFrancyRenderer
  - for IsFrancyObject, [43](#)
- IsFrancyRendererRep
  - for IsComponentObjectRep, [44](#)
- IsFrancyRendererType
  - for IsFrancyObject, [43](#)
- IsFrancyRendererTypeRep
  - for IsComponentObjectRep, [44](#)
- IsFrancyTypeObject
  - for IsObject, [24](#)
- IsLink
  - for IsFrancyObject, [26](#)
- IsLinkDefaults
  - for IsFrancyDefaultObject, [26](#)
- IsLinkDefaultsRep
  - for IsComponentObjectRep, [27](#)
- IsLinkRep
  - for IsComponentObjectRep, [27](#)
- IsMenu
  - for IsFrancyObject, [37](#)
- IsMenuRep
  - for IsComponentObjectRep, [37](#)
- IsRequiredArg
  - for IsFrancyObject, [6](#)
- IsRequiredArgRep
  - for IsComponentObjectRep, [7](#)
- IsShape
  - for IsFrancyObject, [25](#)
- IsShapeDefaults
  - for IsFrancyDefaultObject, [26](#)
- IsShapeDefaultsRep
  - for IsComponentObjectRep, [27](#)
- IsShapeRep
  - for IsComponentObjectRep, [27](#)
- IsShapeType
  - for IsFrancyObject, [26](#)



- IsShapeTypeRep
  - for IsComponentObjectRep, 27
- IsTriggerType
  - for IsFrancyTypeObject, 6
- IsTriggerTypeRep
  - for IsComponentObjectRep, 7
- IsXAxis
  - for IsFrancyObject, 18
- IsYAxis
  - for IsFrancyObject, 18
- JUPYTER\_ViewString
  - for IsObject, 46
- Layer
  - for IsShape, 32, 33
- Length
  - for IsLink, 35
- Link
  - for IsShape, IsShape, IsLinkDefaults, 30
- Links
  - for IsList, IsList, IsLinkDefaults, 30
- Menu
  - for IsString, IsCallback, 38
- MergeObjects
  - for IsFrancyObject, IsFrancyObject, 46
- NoopCallback, 9
- ParentShape
  - for IsShape, 33
- PosX
  - for IsShape, 31
- PosY
  - for IsShape, 32
- Remove
  - for IsCallback, IsRequiredArg, 9
  - for IsCanvas, IsChart, 13
  - for IsCanvas, IsFrancyGraph, 12
  - for IsCanvas, IsFrancyMessage, 13
  - for IsCanvas, IsFrancyRenderer, 13
  - for IsCanvas, IsMenu, 13
  - for IsChart, IsDataset, 20
  - for IsFrancyGraph, IsLink, 28
  - for IsFrancyGraph, IsShape, 29
  - for IsFrancyMessage, IsCallback, 42
  - for IsMenu, IsMenu, 38
  - for IsShape, IsCallback, 29
  - for IsShape, IsFrancyMessage, 30
  - for IsShape, IsMenu, 29
- RequiredArg
  - for IsArgType, IsString, 9
- Sanitize
  - for IsObject, 46
- Selected
  - for IsShape, 34
- SetAxisXDomain
  - for IsChart, IsList, 22
- SetAxisXTitle
  - for IsChart, IsString, 21
- SetAxisYTitle
  - for IsChart, IsString, 22
- SetCollapsed
  - for IsFrancyGraph, IsBool, 34
- SetColor
  - for IsLink, IsString, 36
  - for IsShape, IsString, 31
- SetConfirmMessage
  - for IsCallback, IsString, 10
- SetConjugateId
  - for IsShape, IsInt, 35
- SetFrancyId
  - for IsFrancyObject, IsString, 24
- SetHeight
  - for IsCanvas, IsPosInt, 15
- SetInvisible
  - for IsLink, IsBool, 36
- SetLayer
  - for IsShape, IsInt, 33
- SetLength
  - for IsLink, IsInt, 35
- SetParentShape
  - for IsShape, IsShape, 33
- SetPosX
  - for IsShape, IsInt, 32
- SetPosY
  - for IsShape, IsInt, 32
- SetSelected
  - for IsShape, IsBool, 34
- SetShowLegend
  - for IsChart, IsBool, 21
- SetSimulation

- for IsFrancyGraph, IsBool, [33](#)
- setSize
  - for IsShape, IsPosInt, [32](#)
- SetTextTypesetting
  - for IsCanvas, IsBool, [16](#)
- SetTitle
  - for IsCanvas, IsString, [15](#)
  - for IsFrancyMessage, IsString, [42](#)
  - for IsLink, IsString, [36](#)
  - for IsMenu, IsString, [39](#)
  - for IsRequiredArg, IsString, [10](#)
  - for IsShape, IsString, [31](#)
- SetValue
  - for IsFrancyMessage, IsString, [42](#)
  - for IsFrancyRenderer, IsFrancyRender-  
erType, [45](#)
  - for IsRequiredArg, IsString, [10](#)
- SetWeight
  - for IsLink, IsInt, [35](#)
- SetWidth
  - for IsCanvas, IsPosInt, [14](#)
- SetZoomToFit
  - for IsCanvas, IsBool, [15](#)
- Shape
  - for IsShapeType, IsString, IsShapeDefaults,  
[29](#)
- ShowLegend
  - for IsChart, [21](#)
- Simulation
  - for IsFrancyGraph, [33](#)
- Size
  - for IsShape, [32](#)
- TexTypesetting
  - for IsCanvas, [15](#), [16](#)
- Title
  - for IsCanvas, [15](#)
  - for IsFrancyMessage, [42](#)
  - for IsLink, [36](#)
  - for IsMenu, [39](#)
  - for IsRequiredArg, [9](#), [10](#)
  - for IsShape, [31](#)
- Trigger
  - for IsString, [9](#)
- UnsetLinks
  - for IsFrancyGraph, [28](#)
- UnsetNodes
  - for IsFrancyGraph, [28](#)
- Value
  - for IsFrancyMessage, [42](#)
  - for IsFrancyRenderer, [44](#), [45](#)
  - for IsRequiredArg, [10](#)
- Weight
  - for IsLink, [35](#)
- Width
  - for IsCanvas, [14](#)
- XAxis
  - for IsAxisScaleType, IsString, IsList, [20](#)
- YAxis
  - for IsAxisScaleType, IsString, IsList, [20](#)
- ZoomToFit
  - for IsCanvas, [15](#)