User guide

FastReport Online Designer

Content

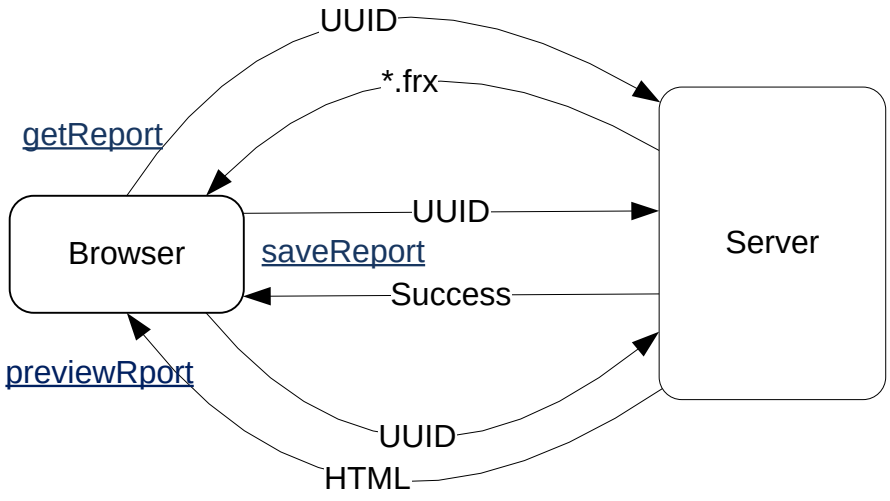
Introduction

FastReport Designer Online is a web version of the designer FastReport.Net. Online report designer is a RIA (Rich Internet application) application that allows you to run it from any device that has a modern Internet browser (online designer will work in 2 recent versions of popular browsers (Chrome, Firefox, Opera, Safari, IE)). But, despite all these cross-platform advantages, the online version is inferior to desktop version in terms of convenience and functionality.

Thus, FastREport Online Designer positioned as an editor of .Net reports that have already been created and placed on any UUID on the server. Online designer communicates with the server through a specified pre-API, which includes 3 requests:

- getReport - is used for initialization. Gets the report template and sends it to the online designer that prepares a report for editing in the browser.
- previewReport (preview mode) - edited report template is sent to the server, which builds the report and returns it in html format. The report runs on the server via FastRport .Net.
- saveReport - saves the report template to the server.

For each query you should pass the UUID of the report to the server using a parameter to identify the report on the server.



The product is developed with the latest capabilities of modern browsers. For example, thanks to HTML5, once downloading the online designer, you can use it when not connected to the network.

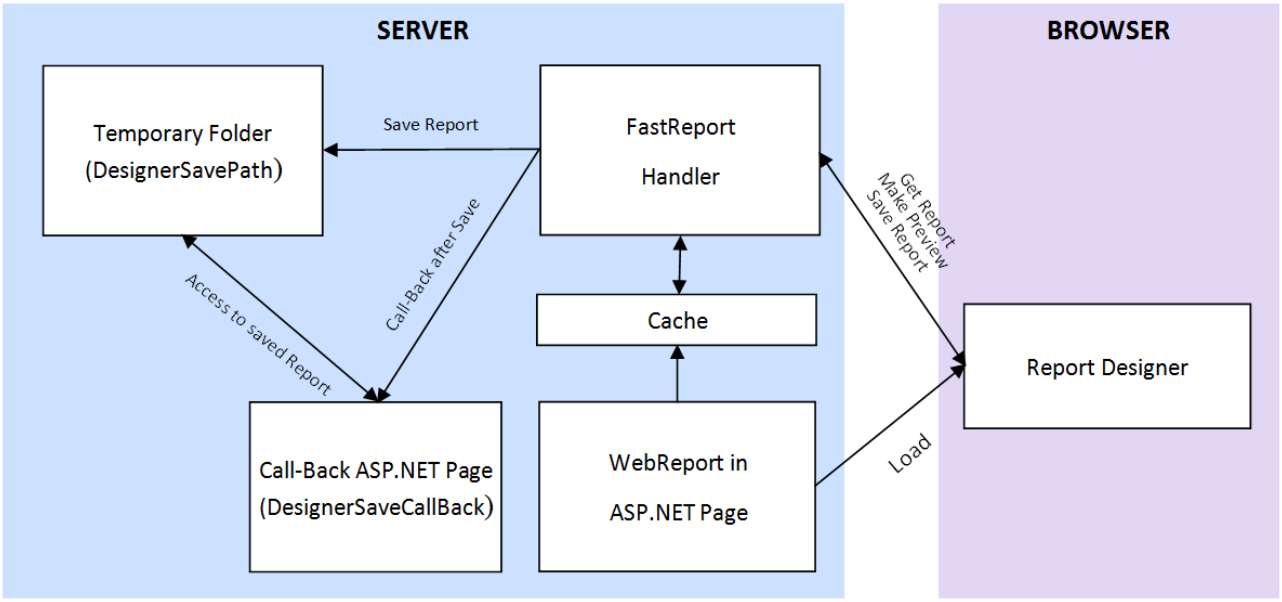
FastReport Designer Online has a monolithic kernel, enabling the connection of modules, which are components of/bands/dialog components and some other parts of the system. To identify these modules and their dependencies the technology RequireJS is used. Such modularity allows you to build a product for the needs of the client with the necessary components, which reduces the size of the project (since this is a web app, the size is very important and the smaller, the better). For individual assemblies the online designer provides the designer constructor.

It is worth mentioning that other technologies are used in FastReport Online Designer. Traditionally used jQuery, and the client's template engine uses jsrender and RequireJS. To edit the code editor uses CodeMirror, that can be embedded in the report.

The current version of FastReport Online Designer is missing a number of key components: MapObject, ChartObject, SparklineObject.

And not all components are available in dialog forms. Only present: Button, CheckBox, CheckedListBox, ComboBox, DateTimePicker, Label, ListBox, MonthCalendar, RadioButton, TextObject.

1. Working principle



The Online Designer can be used together FastReport.Net WebReport objects in editions FastReport .Net Win+Web, Professional, Enterprise.

On-line Designer can change the script of the report and event handlers of the report, but by default for security reasons this option is disabled. This feature can be enabled in the properties of the object WebReport. When this option is disabled the script contents after design will be ignored and replaced with the original text. Also, for security purposes we do not send in Designer built-in connection strings.

1. WebReport object loads in ASP.NET page.
2. WebReport sends an AJAX request to the handler of FastReport to get of on-line Designers container in iframe context (The code of the Report Designer is placed in a separate folder on the application site).
3. When On-line Designer is loaded in browser it sends AJAX query to the handler for get of report template (getReportByUUIDFrom).
4. The server application prepares and sends a report template to the On-line Designer.
5. The Designer can request a preview of the current report. It sends request to the handler in server (makePreviewByUUID). The server application runs a received report and sends back result in html. The Designer shows it in preview window. This preview can be printed or exported in several formats.
6. The Designer can save a report in server through AJAX query (saveReportByUUIDTo) with report contents. The server application prepares received data and sends request to call-back page in application.

Object WebReport exists in the server cache for a limited time, and then deleted from memory. Time of keeping an object in memory is determined in minutes in property WebReport.CacheDelay (by default : 60).

Step-by-step manual for set-up the Online Designer:

1. First let's copy a folder with Online Designer (by default: WebReportDesigner) from installation path to the web applications root.

2. Then check the file web.config for handler settings that needed for WebReport functionality:
for IIS6:

```
<system.web>
…
<httpHandlers>
<add path="FastReport.Export.axd" verb="*" type="FastReport.Web.Handlers.WebExport"/>
</httpHandlers>
</system.web>
```

for IIS7:

```
<system.webServer>
<handlers>
<add name="FastReportHandler" path="FastReport.Export.axd" verb="*"
type="FastReport.Web.Handlers.WebExport"/>
</handlers>
</system.webServer>
```

3. Then check the settings of the Report Designer in the file WebReportDesigner/scripts/config-data.js:
'getReportByUUIDFrom': '/FastReport.Export.axd?getReport=',
'saveReportByUUIDTo': '/FastReport.Export.axd?putReport=',
'makePreviewByUUID': '/FastReport.Export.axd?makePreview=',
These parameters should contain the path to the handler FastReport relative to the root of web site. If your path is different from what is written, it must be corrected, for example:
'getReportByUUIDFrom': '/oursite/FastReport.Export.axd?getReport=',

4. When WebReport is used in ASPX markup, you need drag-n-drop the object on a page and set the properties. For MVC you need write the code in the controller:

4.1. Enables editing of reports:
webReport.DesignReport = true;

4.2. Set the unique object name WebReport, necessary for further identification in the call-back page while maintaining the edited report:
webReport.ID = "MyDesignReport1";

4.3. Prohibits script editing the report in On-line Designer, or if you want to allow editing - assign true:
webReport.DesignScriptCode = false;

4.4. Specify the path to the main file of the report designer, the folder with the designer should be copied to the appropriate place of web-application:
webReport.DesignerPath = "~/WebReportDesigner/index.html";

4.5. Set the path to the call-back page on the site, the call to be executed after the report is saved to a temporary folder. Example for MVC path to View (you have to create new blank View specially for call-back in controller with same name):
webReport.DesignerSaveCallBack = "~/Home/SaveDesignedReport";
or, an example for ASPX:
webReport.DesignerSaveCallBack = "~/DesignerCallBack.aspx";
The following parameters sends in the GET request: reportID="here is webReport.ID"&reportUUID="here is saved report file name". In this context reportID corresponds to the object WebReport.ID, and reportUUID the file name that is stored in a temporary folder. The developer shall perform further actions to save the report to the source file on the disk, database or cloud storage. Temporary file named reportUUID must be removed from the temporary folder after saving. You can use POST query for call-back transfer of report file, read more below in 4.6. Sample code call-back pages will be shown below.

4.6. Set the path to the temporary folder in which to save the edited reports before calling the call-back, in a folder must be set write permissions:
webReport.DesignerSavePath = "~/App_Data/DesignedReports";
You can set the property webReport.DesignerSavePath to blank string for activate POST mode.

4.7. Set the lifetime of WebReport object in servers cache in minutes, the default time is 60:
webReport.CacheDelay = 120;

5. Create a call-back page to save the edited reports.

5.1. If you are using layout ASPX, you need to add the following code in the Page_Load event handler:

```
protected void Page_Load(object sender, EventArgs e)
{
string reportID = Request.QueryString["reportID"];
string reportUUID = Request.QueryString["reportUUID"];
// 1. ReportID value identifies the object that caused the designer. The value
corresponds to the property webReport.ID, which was filled by a call of the designer.
// 2. Combining the path that we have filled in the property webReport.DesignerSavePath,
and the resulting reportUUID, we get the path to the temporary file with edited report.
// 3. This file can be opened and saved in the right place for us to drive or the cloud
or in a database.
// 4. The temporary file must be deleted after saving.
}
```

5.2. In MVC markup you need to create a method in the controller and empty View. The code in the controller:

```
public ActionResult SaveDesignedReport(string reportID, string reportUUID)
{
// 1. ReportID value identifies the object that caused the designer. The value
corresponds to the property webReport.ID, which was filled by a call of the designer.
// 2. Combining the path that we have filled in the property webReport.DesignerSavePath,
and the resulting reportUUID, we get the path to the temporary file with edited report.
// 3. This file can be opened and saved in the right place for us to drive or the cloud
or in a database.
// 4. The temporary file must be deleted after saving.
return View();
}
```

For work with POST transfer you need add next line before controller:

[HttpPost]

public ActionResult SaveDesignedReport(string reportID, string reportUUID)

5.3. You can use any localizations for Online Designer in property webReport.DesignerLocale = "en"; ("en" can be changed to any other supported language, full list of supported languages is similar to files in folder *locales* in Designer distribution package).

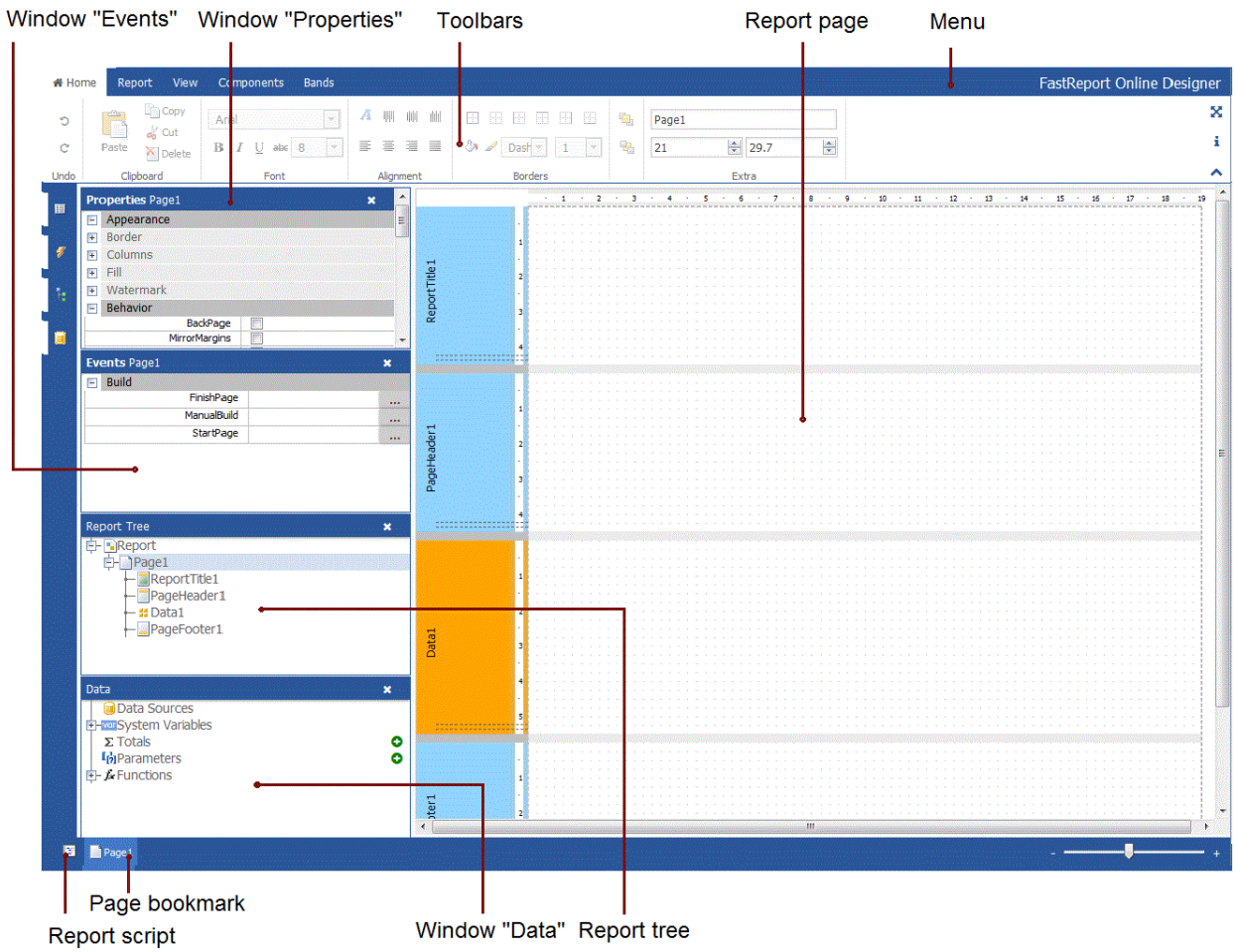
When creating handlers Call-back pages for saving reports special attention should be paid to filtering and checking of received in Get request parameters. Be sure to check them to null.

Best place for object with On-line Designer is at the bottom of the page. The recommended width is 100% or pixels at least 930px. Height of the object is recommended to set at least 600px.

2. Designer

Let’s consider the structure of the interface of FastReport Online Designer. There are the following areas:

- menu;
- report page;
- "Properties" window;
- "Events" window;
- report tree;
- "Data"window;
- toolbar;
- bookmark pages;
- report script.



Let us examine each item more info.

Main menu is placed at the top of the report designer: Main, Report, View, Components, Bands. When we select the menu item, we open the tab with toolbars, similar to Microsoft Office 2007.

The toolbar of the Main tab is used to change the appearance of the report components.

On the "Report" tab you can save the report, add/delete page, add dialog, and run the report in preview mode.

On the tab “View”, you can specify settings for the grid of a report page. The grid helps to position components in accordance with each other.

The“Components” tab contains the FastReport component palette. Components allow you to display different data in the bands. They are an integral part of the report template along with the bands.

The tab "Bands" contains a palette of bands that can be added to the report. The bands represent a container for placing components. The type of band determines its location in the report.

Report page contains the bands and components that make up a report template.

The Properties window is hidden by default, like the other windows. It can be enabled by using the icon on the sidebar. So you can include the "opening Event", the tree report and the "Data" window. For convenience the open windows can be moved anywhere on the screen. To return the window to its original position click on the paperclip icon in the header.

The “Properties” window displays the properties of the selected report object. This object can be band, component, or even the report page.

The "Events" shows the events available for the selected report object. The report tree contains all report objects in a hierarchical list. By the right-click on items in the list you can call the context menu for the selected object.

At the bottom of the report designer you can see the tabs of pages in the report, as well as the icon of the report script. If you use the script, the code editor opens instead of the report page:

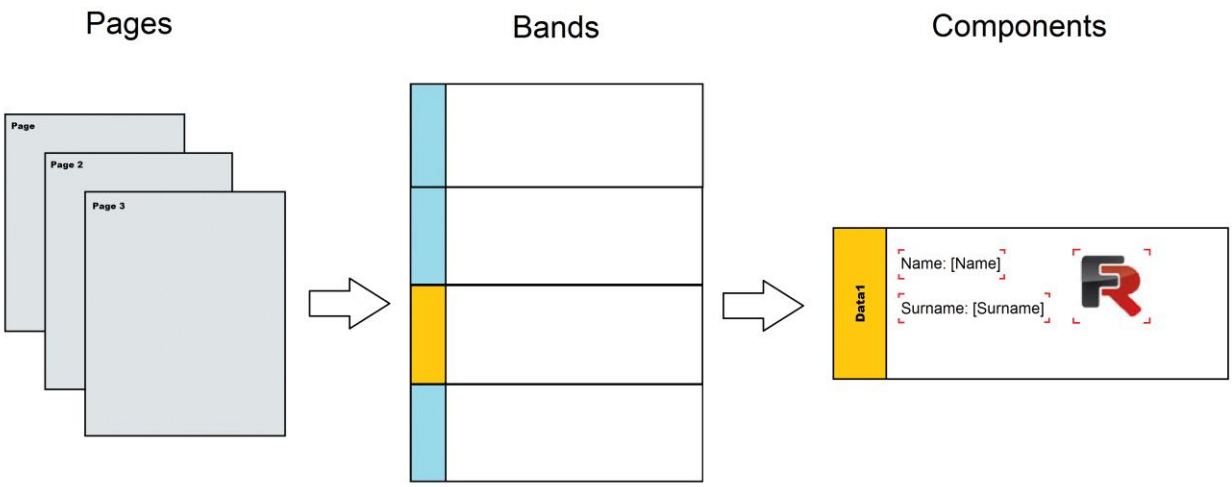
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Drawing;
using System.Data;
using FastReport;
using FastReport.Data;
using FastReport.Dialog;
using FastReport.Barcode;
using FastReport.Table;
using FastReport.Utils;
namespace FastReport
{
    public class ReportScript
    {

    }
};

```

The report script allows the user to define the logic of the report.

2.1. Report pages

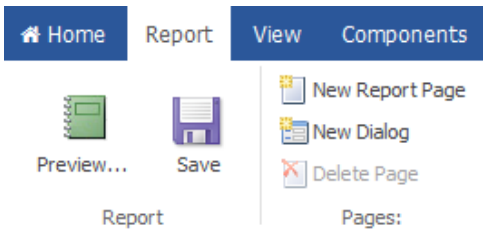
A report can contain many pages. Moreover, for example, the title page printed once. And the number of pages of data will depend on the amount of data in the source.

A report structure can be simplistically represented as follows:



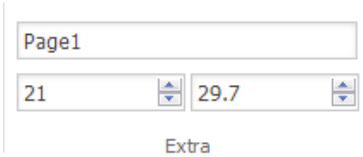
A report contains pages, pages – bands, bands – components, components – data.

A new report already contains one page, but if you want to add another one, go to the "Report" tab and click on new page icon (NewReportPage).

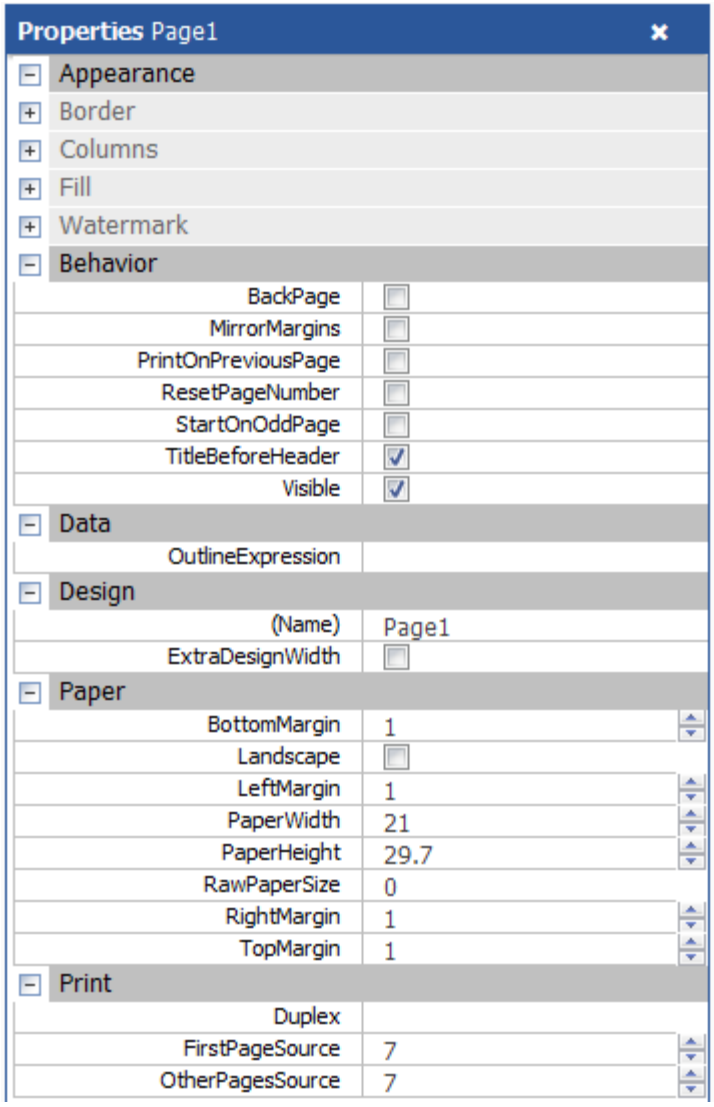


Here you can add dialog form (NewDialog).

To delete a report page, you must navigate to the desired page and click on the delete icon (Delete page). If the report consists of one page the delete icon will be inactive. You can set the page size on the Home tab, in the section "Extra". You need to select the page tab in the bottom panel of the designer.



Other properties of the page can be seen in the window "Properties", if you select a page from the tabs at the bottom of the designer.



You can set the page size and margins in the "Paper" section .

Section "Print" allows you to determine two-sided printing, the source of the first page and the source of other pages.

In addition, you have access to the configuration frames, columns, fills, etc.

2.2. Bands

In FastReport .Net components cannot be simply placed on a blank page of report. They are intended for special containers – bands. The report page should contain at least one band. There are different types of bands, which occupy a certain position on the page of the report, depending on their destination.

Let's look at the types of bands:

Band	How it's printed
Report Title	It is printed once at the very beginning of the report. You can choose the order of printing - before the "Page Header" band or after it - with the help of the "TitleBeforeHeader" page property. Changing this property can be done with the help of "Properties" window. By default, property is equal to true, that is, report title is printed before page header.
Report Summary	It is printed once at the end of the report, after the last data row, but before the "Page Footer" band.
Page Header	It is printed on top of every page of the report.
Page Footer	It is printed at the bottom of every page of the report.
Column Header	This band is used when printing a multi-columned report (when the number of columns indicated in the page setup > 1). It is printed on top of every column after the Page Header band.
Column Footer	Printed at the bottom of every column, before the Page Footer band.
Data	This band is connected to the data source and is printed as many times as there are rows in the source.

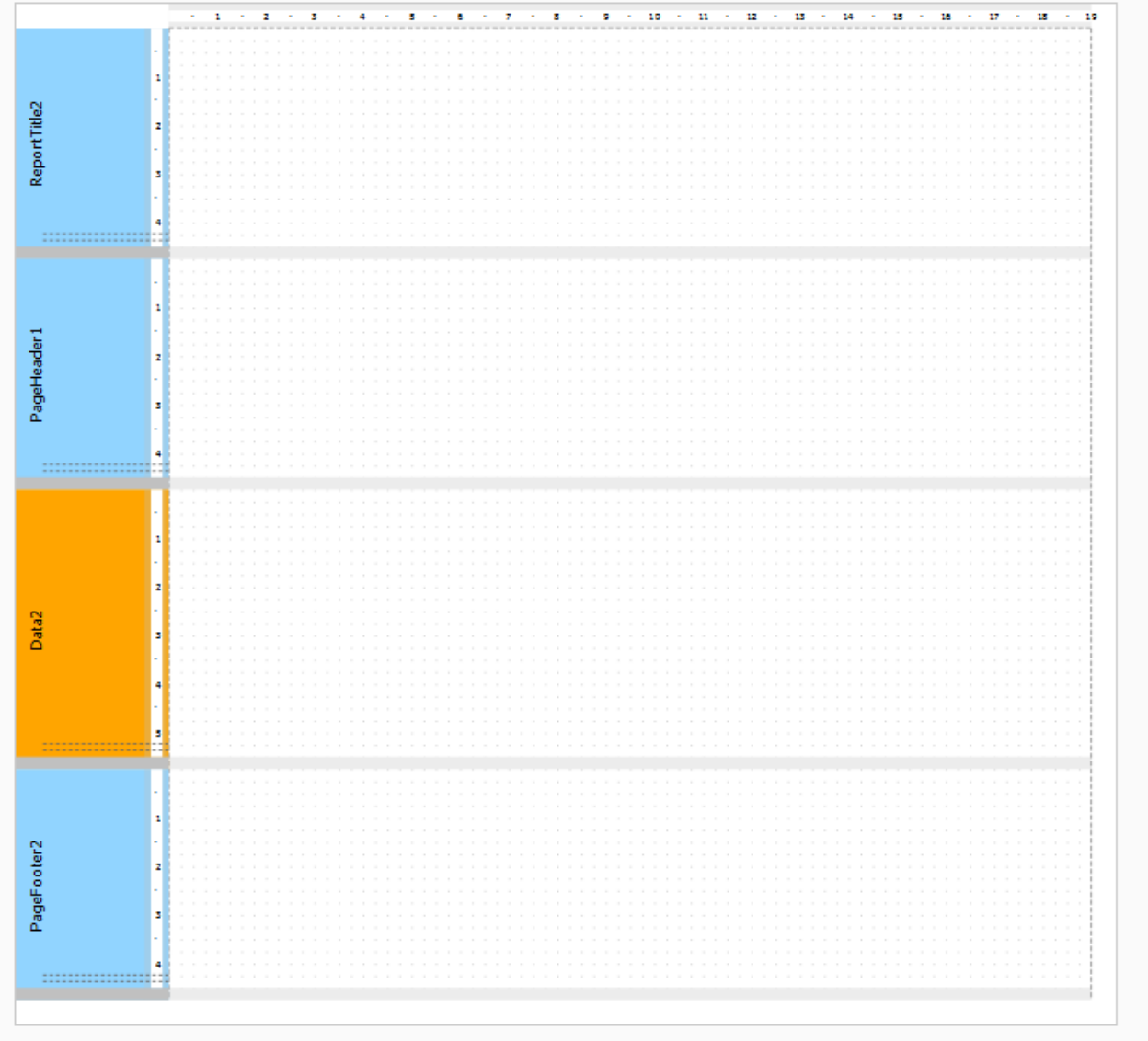
Data Header	This band is connected to the "Data" band and is printed before the first data row.
Data Footer	This band is connected to the "Data" band and is printed after the last data row.
Group Header	It is printed at the beginning of every group, when the value of the group condition changes.
Group Footer	It is printed at the end of every group.
Child	This band can be connected to any band, including another child band. It is printed immediately after its parent.
Overlay	Printed as a background on every report page.

Consider the display of bands in the designer.

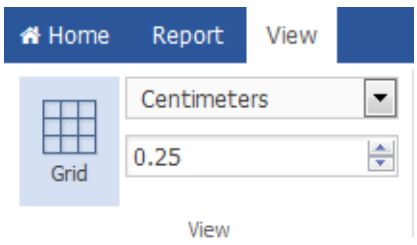
On the left side of the report page are the headers of the bands. By default, a new report contains 4 bands:

- ReportTitle;
- PageHeader;
- Data;
- PageFooter.

In the data area the band has a rectangular shape.



Bands can have a fill and frames that are disabled by default. Also, bands have a grid for easy positioning of components. The grid can be set in "View" menu of the main menu.



To change the size of the band you can use the mouse. Move the cursor to the bottom of the band. The cursor changes. Click the left mouse button and adjust the height of the band up or down.

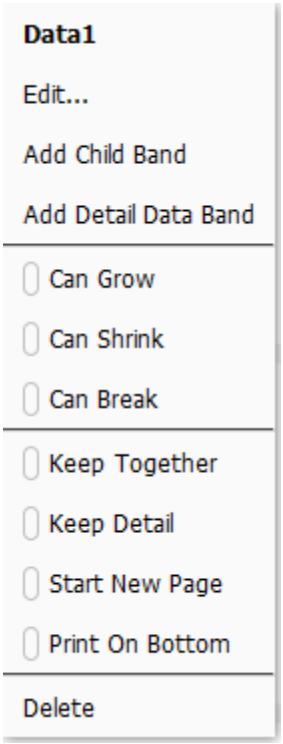
2.2.1. Band settings

To add a band to the report page, click the tab "Bands". Select the required band and click on it.

To add the "Header data" or "data footer" you should pre-select the Data band on the report page.

To add another band "Data" you can choose any of the bands on the report page except the already existing band "Data". Then add the band"Data" by using the icon on the toolbar. If you select bandData on the report page and add another bandData, there will be added detailed band "Data".

There is another way to add a detailed band "Data". Call the context menu for the "Data" band by the right mouse click. And choose from the list “Add Detail Data Band”. In addition, from this menu you can add a child band.



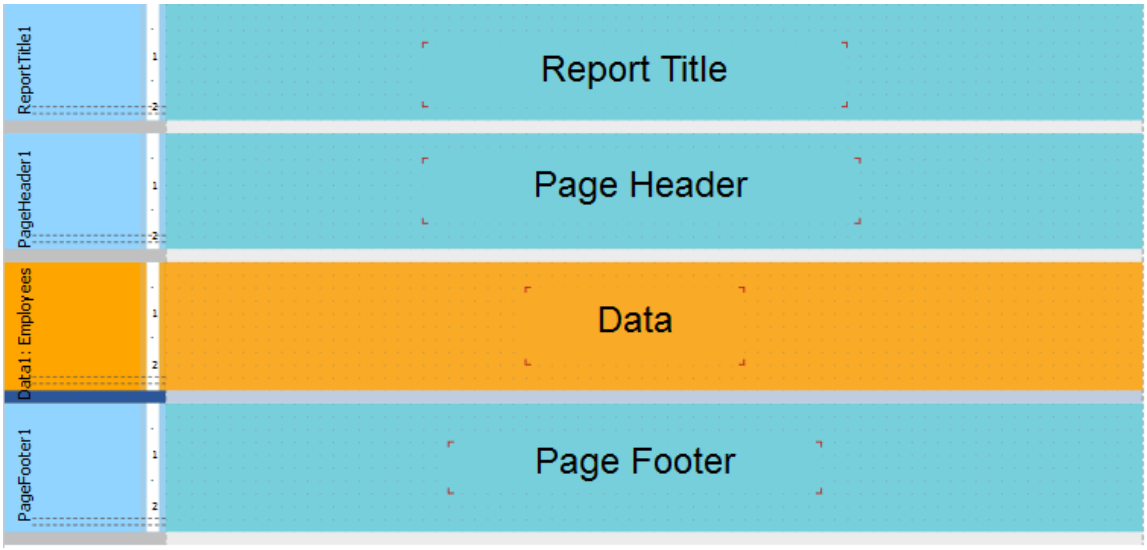
You can delete the selected band by using a context menu or pressing the Delete key.

FastReport will limit your actions which could lead to the creation of an incorrect report template. For example, if you have a band “group Header”, you can't delete the band Data from this group. First you have to remove the bandgroup.

Also, when you delete some of the bands, they will be deleted along with their associated bands. For example, if you delete the band "Data" its header, footer, child band and detail band will be removed too.

2.2.2. Print order

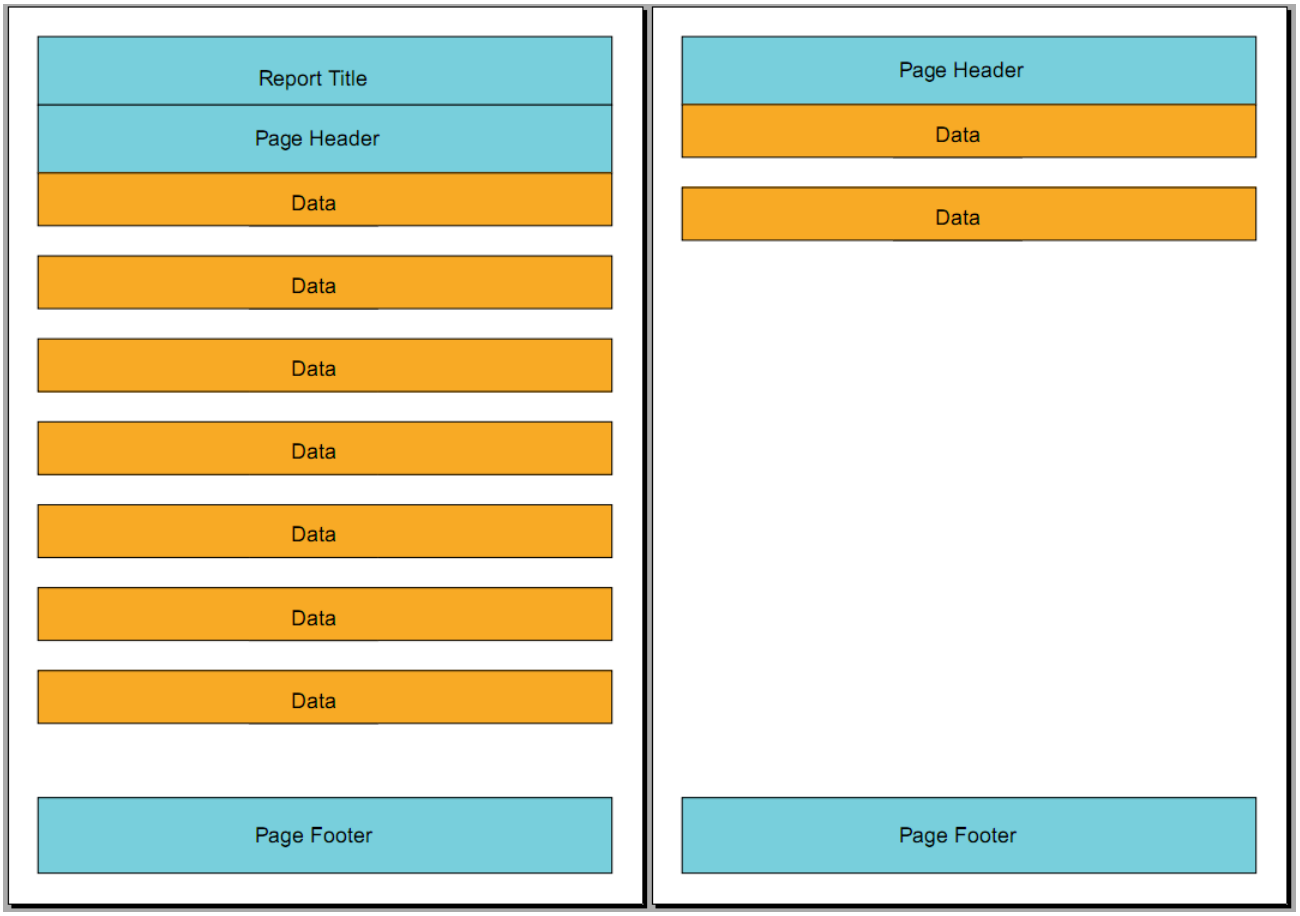
So, there are several bands on the page. Look at how FastReport will create a report:



The order of the bands is as follows:

- report title;
- page header;
- data. Will be printed as many times as there are rows in the data source that is connected to the band;
- report footer.

The printing of the report ends. The finished report will look like this:



In the process of printing, FastReport checks if there is enough space on the current page of a finished report to print the band. If the band cannot fit on the page, the procedure is the following:

- printed band "page footer" on the current page of the report;
- added to the new page;
- printed band "page Header";
- continues to print the band which did not fit on the previous page.

2.2.3. Band properties

All bands have some common properties that affect the printing process. To view the properties of the band, you need to select it with the mouse on the report page, and open the Properties window with the icon on the side bar.

Property	Description
CanGrow, CanShrink	These properties determine whether a band can grow or shrink depending on the size of the objects contained in the band. If both properties are disabled, the band will always have the size specified in the designer. Read more about this in the "Report Creation" chapter.
CanBreak	If the property is enabled, FastReport tries to print a part of the band's contents on the available space, that is, "break" the band. Read more about this in the "Report Creation" chapter.
StartNewPage	Printing a band with such property begins on a new page. This property is usually used when printing groups; that is, every group is printed on a new page.
PrintOnBottom	A band with this property is printed at the bottom of the page, before the "Page Footer" band .This can be useful when printing certain documents, where the total sum is supposed to be printed at the bottom of the page.
RepeatOnEveryPage	The bands - "Data Header", "Data Footer", "Group Header" and "Group Footer" - have got this property. This type of band will be printed on each new page, when data printing is being done. Read more about this in the "Report Creation" chapter.

2.3. Components

2.3.1. Text

The "Text" object is the basis of a data presentation in FastReport. In the components palette, it looks like:

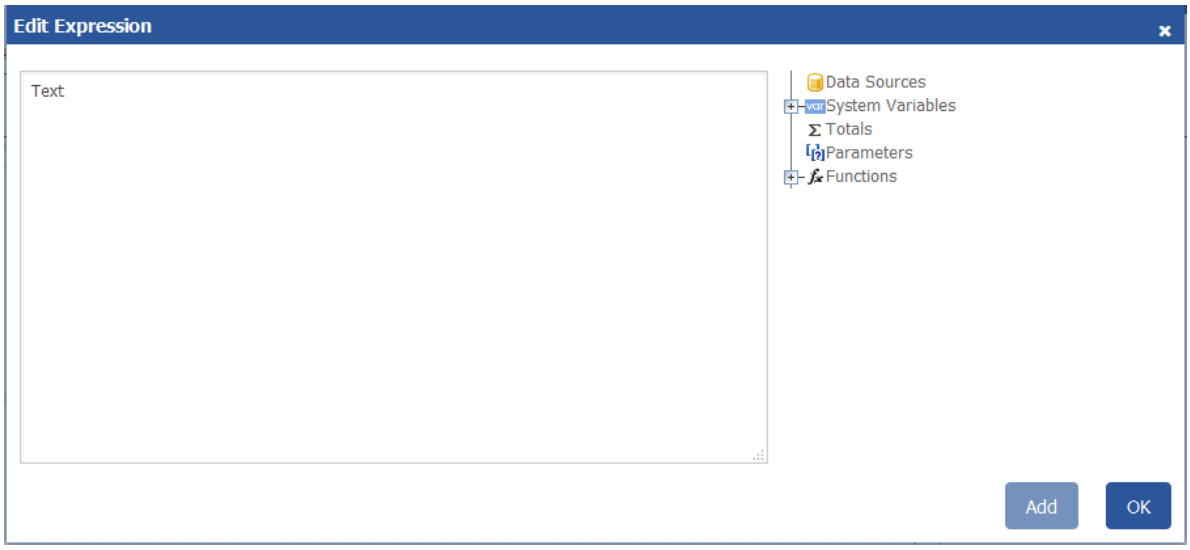
And on the report page:

Object “Text” allows to display the following text information:

- lines of text;
- expressions;
- report parameters;
- totals;
- fields from data sources;
- system variables.

Furthermore, you can combine these data in a text object.

To open the editor of the text object, you need to double-click on the object. This opens the "Expression editor":



The "Text" object can contain a plain text mixed with expressions. For example:

Today is [Date]

When printing such an object, all expressions contained in the text will be calculated. So the result may look like this:

Today is 12.09.2010

As seen, expressions are identified by square brackets. This is set in the "Brackets" property, which by default contains the string "[,]".When needed, you can use a different set of symbols, for example "<,>", or "<!,!>". In the last case, an expression in the text will be like this:

Today is <!Date!>

Apart from this, it is possible to disable all expressions. To do this, set the AllowExpressions property to **false**. In this case the text will be shown "as is".

Inside the square brackets, you can use any valid expression. Read more about expressions in the "Expressions" chapter. For example, an object with the following text:

*2 * 2 = [2 * 2]*

will be printed like this:

*2 * 2 = 4*

A frequent mistake – the attempt to write an expression outside the square brackets. Be reminded, that it is considered as an expression and gets executed only that, which is located inside square brackets. For example:

*2 * 2 = [2] * [2]*

This text will be printed this way:

*2 * 2 = 2 * 2*

There may be elements inside an expression that needs own square brackets. For example, it may be a reference to a system variable. Let's look at the following example:

The next page: [[Page] + 1]

The text contains an expression *[Page] + 1*.*Page* is a system variable which returns the number of the current report page. It is included in own brackets. These brackets must be square brackets, regardless of the "Text" object settings.

Strictly speaking, we were supposed to use two pairs of square brackets when using the "Date" system variable in the examples above:

Today is [[Date]]

However FastReport allows to leave out unnecessary pair of brackets if there is only one member in an expression.

You can print the data column in the following way:

[Datasourcename.Column name]

As you can see, the square brackets are used here. The data source name and data column name are separated by the period. For example:

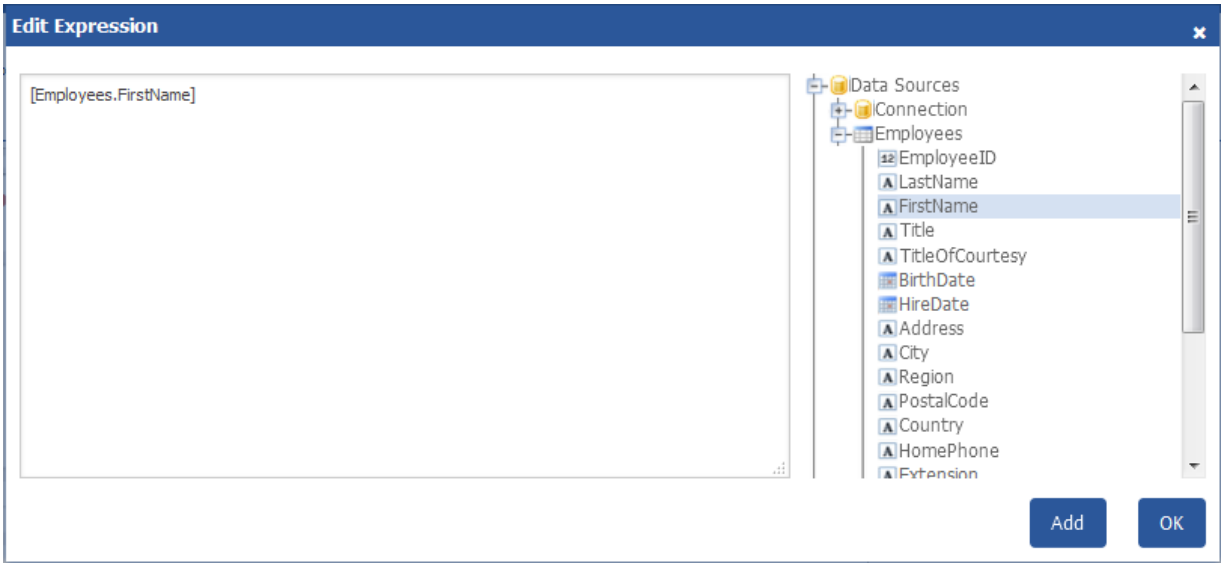
[Employees.FirstName]

Read more about using the data columns in the "Expressions" chapter.

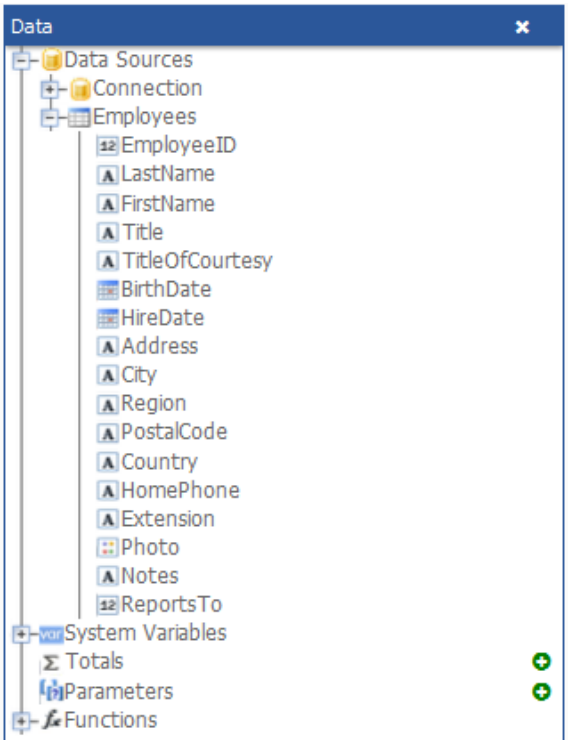
There are several ways to insert a data column into the "Text" object.

1. In the "Text" object's editor we write the name of the data column manually. This method is the most inconvenient as it is easy to make a mistake.

2. In the object's editor we choose the needed data column and drag&drop it into the text:



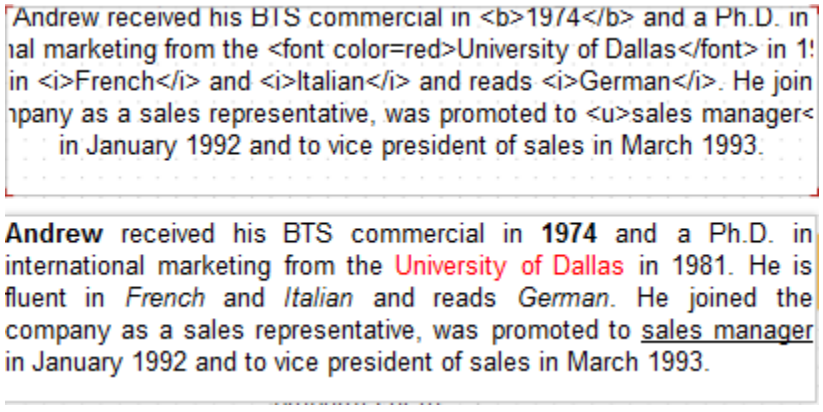
3. Drag&drop a data column from the "Data" window into the report page. In this case the "Text" object is created which contains a link to the column.



You may use some simple HTML tags in the "Text" object. By default, tags are disabled; to enable it, go to the "Properties" window and set the "HtmlTags" property to **true**. Here is a list of supported tags:

Tag	Description
...	Bold text.
...	Italic text.
<u>...</u>	Underlined text.
<strike>...</strike>	Strikeout text.
_{...}	Subscript.
^{...}	Superscript.
...	Font color. The color may be either the named color (such as DarkGray), or a hexadecimal code in the #RGB format, for example *#FF8030*.

Let's see an example of using tags:



Above shows the tagged text. Below is the result of printing the object. Consider the properties of "Text"object:

Property	Description
AllowExpressions	This property allows to turn on or off the expression handling. It is on by default.
Angle	This property indicates the text rotation, in degrees.
AutoShrink	This property allows to shrink the font size or font width automatically to fit the text.
AutoShrinkMinSize	This property determines the minimum size of a font, or the minimum font width ratio, if the AutoShrink property is used.
AutoWidth	This property allows to calculate the width of object automatically.
Brackets	This property contains a pair of symbols that designate an expression.
BreakTo	With this property you can organize the text flow from one text object to another. For example, we have "A" and "B" text objects. The "A" object contains the long text that does not fit in the object's bounds. You can set the A.BreakTo to B, so the "B" object will display the part of text that does not fit in "A".
Clip	This property determines whether it is necessary to clip a text outside of object's bounds. It is on by default.
Duplicates	This property determines how the repeated values will be printed. Read more about this property in the "Formatting" chapter.
FirstTabOffset	This property determines the offset of the first TAB symbol, in pixels.
FontWidthRatio	Use this property to make the font wider or narrower. By

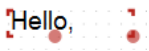
	default the property is set to 1. To make the font wider, set the property to value > 1. To make the font narrower, set the property to value between 0 and 1.
HideValue	This property is of string type. It allows to hide values that are equal to the value of this property. Read more about this property in the "Formatting" chapter.
HideZeros	This property allows to hide zero values. Read more about this property in the "Formatting" chapter.
Highlight	This property allows to setup the conditional highlight. Read more about this in the "Formatting" chapter.
HorzAlign, VertAlign	These properties determine the text alignment.
HtmlTags	Allows simple HTML tags in the object's text. Read more about this property in the "HTML tags" chapter.
LineHeight	This property allows to explicitly set the height of a text line. By default it is set to 0, so the default line spacing is used.
NullValue	The text that will be printed instead of a null value. You also need to uncheck the "Convert null values" option in the "Report/Options..." menu.
Padding	This property allows to setup the padding, in pixels.
RightToLeft	This property indicates whether the text should be displayed in right-to-left order.
TabWidth	This property determines the width of the TAB symbol, in pixels.
Text	This property contains the text of the object.

TextFill	This property determines the text fill. Use this property editor to choose between different fill types.
Trimming	This property determines how to trim the text that does not fit inside the object's bounds. It is used only if the "WordWrap" property is set to false.
Underlines	This property allows to display a graphical line after each text line. This property can be used only if the text is top-aligned.
WordWrap	This property determines whether it is necessary to wrap a text by words.
Wysiwyg	This property changes the display mode of the "Text" object to match the screen and the final printout. This mode is also used if you use the justify-align or non-standard line height.

2.3.2. Rich text

The object "Rich text" allows you to display multiline text in RTF format preserving the layout and styles. On the toolbar it looks like this:

And on the report page, it looks like a simple component "Text":



Try to use the "Text" object to display a text. When you export the report to some document formats, the "Rich Text" object will be exported as a picture.

"Formatted text" can display the data from the source as well as object "Text". To do this, either type the expression manually or connect components to the data field with the properties of DataColumn.

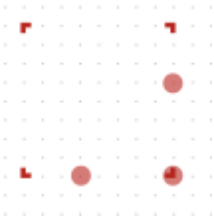
The object has the following properties:

Property	Description
AllowExpressions	This property allows to turn on or off the expression handling. It is on by default.
Brackets	This property contains a pair of symbols that designate an expression.
DataColumn	The data column that this object is bound to.
Text	This property contains the RTF text.
Padding	The padding, in pixels.

2.3.3. Picture

An object can display graphics in the following formats: BMP, PNG, JPG, GIF, TIFF, ICO, EMF, WMF. With the help of the "Picture" object, you can print your company logo, a photo of employee or any graphical information. On the toolbar, it looks like this:

And on the report page, it looks like this:



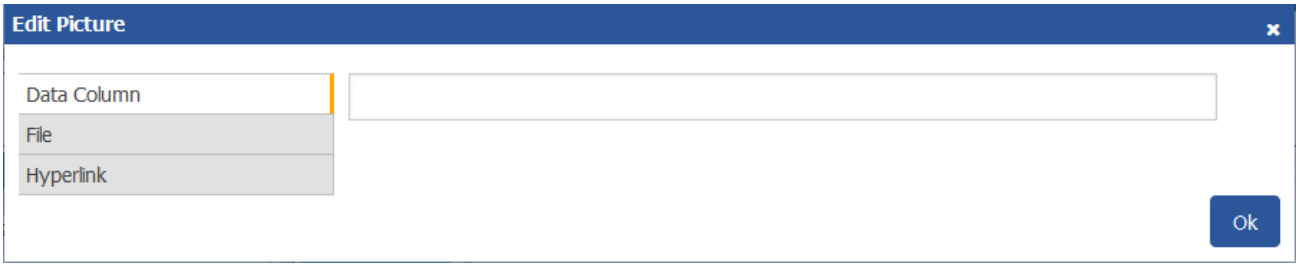
An object can show data from the following sources:

– Data column - Picture from a data column. Name of the column is stored in the "DataColumn" property.

– File with a picture - Picture is loaded from a file and is saved inside the report. Picture is stored in the "Image" property.

– Hyperlink – Picture loaded via the link every time the report is built. Picture is not stored within the report. The address is stored in the property ImageLocation. This may be a URL or link to a local file. In the last case, you have to distribute the image file together with the report.

In order to call a picture editor, double click on the object. In the editor, you can choose the data source for the picture:



You can also set the image source in the object properties: DataColumn, Image, ImageLocation.

You also can drag&drop a data column from the "Data" window into the report page. In this case the "Picture" object is created which contains a link to the column. The column you drag should have the "byte[]" data type.

In the context menu of the "Picture" object you can choose the size mode:

- AutoSize. The object gets the size of the picture.
- CenterImage. The picture is centered inside the object.
- Normal. The picture is displayed in the left corner of the object.
- StretchImage. The picture is stretched to the size of the object.
- Zoom. The picture is stretched to the size of the object in accordance with the aspect ratio.

The difference between modes is shown in the following picture:

It is possible to rotate the image by using the properties of Angle. Results of the pictures rotation by a predetermined angle shown below:

The "Picture" object has the following properties:

Property	Description
Angle	The rotation angle, in degrees. Possible values for this property are 0, 90, 180, 270.
SizeMode	The size mode.
Transparency	The degree of transparency of the pictures. The property can have values between 0 and 1. The value 0 (by default) means that the picture is opaque.
TransparentColor	The color which will be transparent when displaying the picture.
Image	The picture.
DataColumn	The data column that this object is bound to.
ImageLocation	This property can contain name of the file or URL. The picture will be loaded from this location when building the report.
Padding	The padding, in pixels.
ShowErrorImage	Shows the “No picture” picture, in case when the picture is empty. This property makes sense to use if the picture is downloaded from the Internet.

2.3.4. Line

The "Line" object can display horizontal, vertical or diagonal line. The object is as follows:

On the page, the object is as follows:

If possible, use the object's border instead of "Line" object. This will simplify the report and avoid possible problems with the export to different formats.

To add a line in the report, click the icon on the toolbar, "Components". Or drag the icon of the component to the desired location on the report page.

To change the line type from horizontal to vertical or diagonal, turn Diagonal property in the Properties window. After that, the line can be lifted by one edge to enable set the desired angle.

You can set the style of drawing in the properties window, in the section Border. Here you can also set the color and line width.

The "Line" component has the following properties:

Property	Description
Diagonal	The property determines weather the line is diagonal. An ordinary line can be turned into a diagonal one by enabling this property.
StartCap, EndCap	These properties allow to setup the line caps. You can use one of the following cap styles: – ellipse; – rectangle; – diamond;

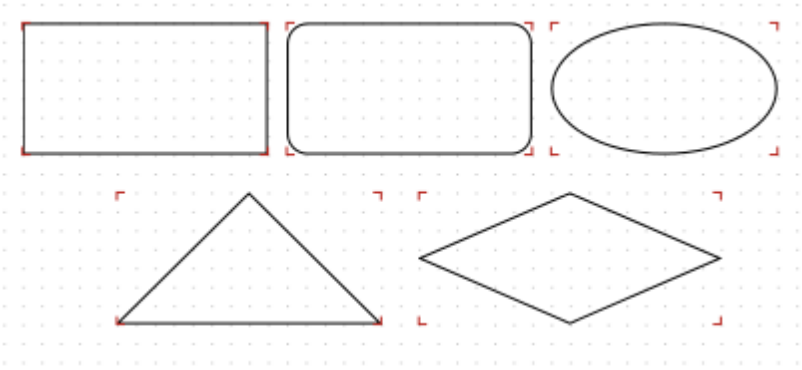
	– arrow. Size of the cap can be set in the Width, Height properties of the cap. You can configure caps for each end of the line.

2.3.5. Shape

Report may require different shapes. The Shape object allows you to add basic shapes, change their color and size. Available in the following shapes:

- rectangle;
- rounded rectangle;
- ellipse;
- triangle;
- diamond.

The objects are as follows:



On the toolbar in looks like this:

To insert a shape into the report, click the picture above shows the icon in the toolbar Components. Then, select the desired shape type from the list in the Shape property of the Shape object.

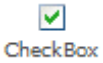
You can make the object's fill (Fill property) and add a border (Border property).

The "Shape" object has the following properties:

Property	Description
Shape	This property determines the type of shape.
Curve	This property is used with the "RoundRectangle" shape. It allows to set the curve.

2.3.6. CheckBox

The object displays the checkbox in the report. On the toolbar, it looks as follows:



And on the report page – as follows:

The object can display two states: "Checked" and "Unchecked". Use the following ways to handle the object's state:

- set the state in the "Checked" property;
- bind the object to a data column using the "DataColumn" property;
- set the expression that returns the **true** or **false** in the "Expression" property.

The "CheckBox" object has the following properties:

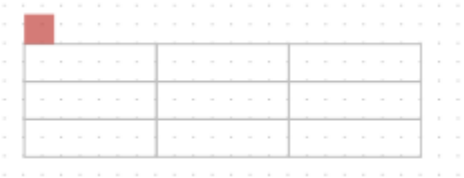
Property	Description
CheckedSymbol, UncheckedSymbol	These properties determine the symbol that is shown in the object, depending on the object's state.
CheckColor	This property determines the color of the check symbol.
CheckWidthRatio	Use this property to set the check width ratio. The width of the check

	symbol depends on the size of the object. You can use values in the 0.2 - 2 range.
HideIfUnchecked	This property allows to hide the object if it is unchecked.
Checked	This property controls the state of the object.
DataColumn	The data column which this object is bound to. The type of column should be either bool or int.
Expression	The expression that returns the **true** or **false**.

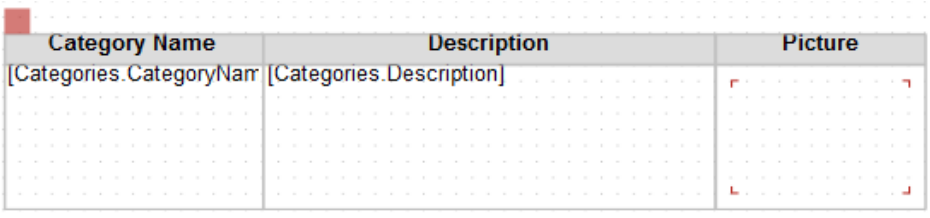
2.3.7. Table

Component "table" is a simplified version of the table Microsoft Excel. On the palette, it looks as follows:

And on the report page – as follows:



You can create a static table, filling cells by hand. And it is possible to create a dynamic table using the fields from the data source. An example of a dynamic table is shown below:



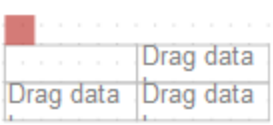
The "Table" object has the following properties:

Property	Description
ColumnCount	Use this property to quickly set the number of columns. If columns in a table are few, they get added, and when they are more, they get deleted.
RowCount	Use this property to quickly set the number of rows. If rows in a table are few, they get added, and when they are more, they get deleted.
FixedColumns	The property determines how many columns in the table are fixed. Fixed columns form the table header. Printing of the header is controlled by the "RepeatHeaders" property.
FixedRows	The property determines how many rows in the table are fixed. Fixed rows form the table header. Printing of the header is controlled by the "RepeatHeaders" property.
RepeatHeaders	The property allows printing the table header on every new page. This property works only for tables which are formed dynamically.

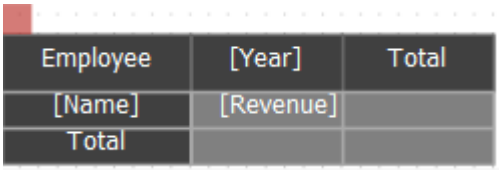
2.3.8. Matrix

The "Matrix" object is, like the "Table" object, made up of rows, columns and cells. At the same time, it is not known beforehand how many rows and columns will be in the matrix - this depends on the data to which it is connected. On the toolbar it looks like this:

And on the page, it is as follows:



Object matrix can be filled with data manually. But you also can create a dynamic matrix with fields from data source:



The "Matrix" object has the following properties:

Property	Description
RepeatHeaders	If matrix is divided on several pages, this property allows printing matrix header on each new page.
CellsSideBySide	This property determines how matrix cells will be located if the matrix has several data cell levels. Possible variants: □ the cells are displayed side by side; □ the cells are displayed under each other.
Style	Using this property you can set a style for the whole matrix. You can choose one from predefined styles.
AutoSize	This property allows to calculate the matrix size automatically. Disable it if you want to control the object size manually.
DataSource	The property allows connecting the matrix to the data source. This property is set up automatically when you drag data column to the matrix. However, if you use expressions in cells, check that this property was set up correctly.
Filter	This property contains expression for data filtering which will be applicable to data source of the matrix (see "DataSource" property).

2.3.9. Barcode

The object displays barcodes in the report. It looks like this:

And on the report page, it is as follows:

The object supports the following types of barcodes:

Code	Length	Allowed symbols
2 of 5 Interleaved		0-9
2 of 5 Industrial		0-9
2 of 5 Matrix		0-9
Codabar		0-9, -$:/.+
Code128		128 ASCII chars
Code39		0-9,A-Z, -. *$/+%
Code39 Extended		128 ASCII chars
Code93		0-9,A-Z, -. *$/+%
Code93 Extended		128 ASCII chars
EAN8	8	0-9
EAN13	13	0-9

MSI		0-9
PostNet		0-9
UPC A	12	0-9
UPC E0	6	0-9
UPC E1	6	0-9
2-Digit Supplement	2	0-9
5-Digit Supplement	5	0-9
PDF417		any
Datamatrix		any
QR code		any
Aztec code		any

Barcode data in an object is of a string type. The string can contain any symbol, allowed for the chosen type of barcode. You can choose the type of barcode in the context menu of the "Barcode" object.

You can connect an object to data by using one of the following methods:

- set the barcode data in the "Text" property;
- bind the object to a data column using the "DataColumn" property;
- set the expression that returns the barcode data in the "Expression" property.

The "Barcode" object has the following properties:

Property	Description
Barcode	This property contains barcode-specific settings. Expand this property to set these settings.
Angle	This property determines the rotation of a barcode, in degrees. You can use one of the following values: 0, 90, 180, 270.

Zoom	This property allows to zoom a barcode. It is used along with the "AutoSize" property.
AutoSize	If this property is on, the object will stretch in order to display a whole barcode. If this property is off, the barcode will stretch to to object's bounds.
ShowText	This property determines whether it is necessary to show the human-readable text.
DataColumn	The data column which this object is bound to.
Expression	The expression that returns the barcode data.
Text	The barcode data.
Padding	The padding, in pixels.

The following properties are specific to the barcode type. To change them, select the barcode, go "Properties" window and expand the "Barcode" property.

Property	Description
WideBarRatio	This property is specific to all linear barcodes. It determines the wide-to-narrow bar ratio. For most of barcode types, the value for this property should be between 2 and 3.
CalcCheckSum	This property is specific to all linear barcodes. It determines whether it is necessary to calculate the check sum automatically. If this property is off, you should provide the check digit in the barcode data.
AutoEncode	This property is specific to the Code128 barcode. This code has three different encodings - A, B, C. You should either set the encoding explicitly in the barcode data, or set this property to true. In this case the encoding will be chosen automatically. Use the following control codes in the barcode data:

<table>
<tr><td></td><td>
<table>
<tr><th>Code</th><th>Meaning</th></tr>
<tr><td>&A;</td><td>START A / CODE A</td></tr>
<tr><td>&B;</td><td>START B / CODE B</td></tr>
<tr><td>&C;</td><td>START C / CODE C</td></tr>
<tr><td>&S;</td><td>SHIFT</td></tr>
<tr><td>&1;</td><td>FNC1</td></tr>
<tr><td>&2;</td><td>FNC2</td></tr>
<tr><td>&3;</td><td>FNC3</td></tr>
<tr><td>&4;</td><td>FNC4</td></tr>
</table>

If you set the "AutoEncode" property to true, all control codes will be ignored.

Example of use the control codes:

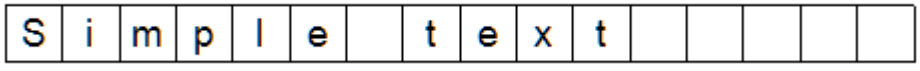
&C;1234&B;ABC
</td></tr>
<tr><td>AspectRatio</td><td>This property is specific to the PDF417 barcode. It determines the height-to-width aspect ratio and is used to calculate the barcode size automatically (in case the "Columns" and "Rows" properties are not set).</td></tr>
<tr><td>CodePage</td><td>This property is specific to the PDF417 and Datamatrix barcode. It determines the code page which is used to convert non-ASCII chars. For example, the default windows codepage is 1251.</td></tr>
<tr><td>Columns, Rows</td><td>These properties are specific to the PDF417 barcode. They determine the number of columns and rows in a barcode. If both properties are set to 0, the size of the barcode will be calculated</td></tr>
</table>

	automatically. In this case the "AspectRatio" property is used as well.
CompactionMode	This property is specific to the PDF417 barcode. It determines the PDF417 data compaction mode.
ErrorCorrection	This property is specific to the PDF417 barcode. It determines the error correction level.
PixelSize	This property is specific to the PDF417 barcode. It determines the size of barcode element, in screen pixels. As a rule, the element's height should be greater than the element width by 3 times or more.
Encoding	This property is specific to the Datamatrix barcode. It determines the Datamatrix data encoding.
PixelSize	This property is specific to the Datamatrix barcode. It determines the size of barcode element, in pixels.
SymbolSize	This property is specific to the Datamatrix barcode. It determines the size of barcode symbol.

2.3.10. Cellular text

This object can display each character of a text in its individual cell. It is often used to print some forms in financial applications. On the toolbar it looks like this:

On the report page, the object is as follows:



In fact, this object is directly inherited from the "Text" object. You may connect it to data in the same manner. For example, you may invoke the object's editor and type the following text:

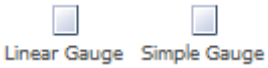
[Employees.FirstName]

The "Cellular Text" object has the following properties:

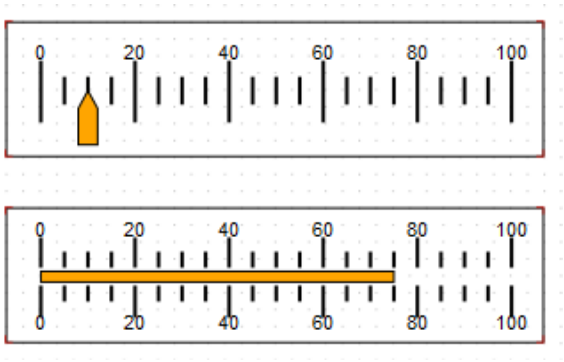
Property	Description
CellWidth, CellHeight	These properties determine the size of a single cell. If both properties are 0 (by default), the cell size will be calculated automatically, depending on the font used.
HorzSpacing, VertSpacing	These properties determine the horizontal and vertical gap between adjacent cells.

2.3.11. Linear scale, simple scale

Some reports may require displaying information, such as gauge, like on the dashboard of industrial facilities. In FastReport there are two components to display information in the form of linear and simple gauge. On the toolbar, they are depicted as follows:



And on the report page – as follows:



Top – line and bottom – a simple gauge.

For the appearance of the components meet the properties of the groups Pointer and Scale in the Properties window. You can set the color of the scales and indicator, the font of the numbers and line width. To calculate the gauge values, enter an expression in the Expression property. This expression can be a field from the data source.

3. Report creation

3.1. Dynamic layout

It is necessary often to print a text whose size is not known when creating a report. For example, this can be a description of goods. In this case, the following tasks will need to be solved:

– calculate the height of the object, such that it encloses the whole text;

– calculate the height of the band, such that it encloses the object with a variable amount of texts;

– move or change the height of other objects, which are contained on the band, such that, they do not disturb the general design of the report.

These tasks can be solved by using some object and band properties:

– "CanGrow" and "CanShrink" properties allow calculating the height of the object automatically;

– "ShiftMode" property allows moving objects that are located under the objects that expand;

– "GrowToBottom" property allows resizing an object to the bottom edge of the band;

– "Anchor" and "Dock" properties allow controlling the size of objects depending on the size of the band.

All these properties will be looked at below.

3.1.1. CanGrow and CanShrink properties

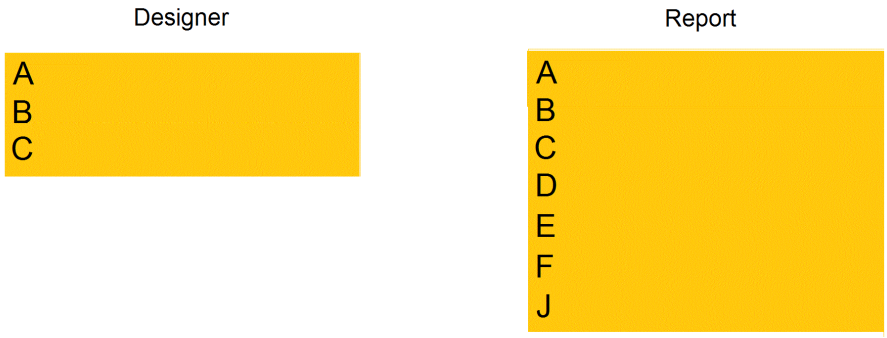
Every band and report object has these properties. They determine whether an object can grow or shrink depending on the size of its contents. If both properties are disabled, the object always has the size specified in the designer.

These properties are very useful if it is needed to print a text whose size is not known when designing. In order for an object to accommodate the entire text, it needs to have the "CanGrow" and "CanShrink" properties enabled.

The text component will look like this if you enable the property CanShrink:

The figure shows that the size of the text component in the designer is too large. When constructing a report the size of the component will be reduced to the optimum.

The text component will look like this if you enable the property CanGrow:

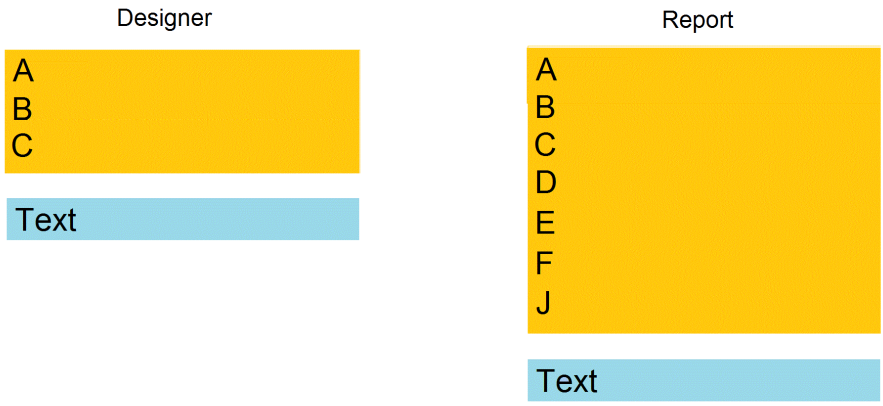


The following objects can affect the height of a band:

- "Text";
- "Rich Text";
- "Picture" (with "AutoSize" property enabled);
- "Table".

3.1.2. ShiftMode property

Every report object has this property. This property is accessible only in the "Properties" window. An object, whose "ShiftMode" property is enabled, will be moving up and down, if the object above on can either grow or shrink.



The "ShiftMode" property can have one of the following values:

– Always (by default). Meaning that the object needs to shift always.

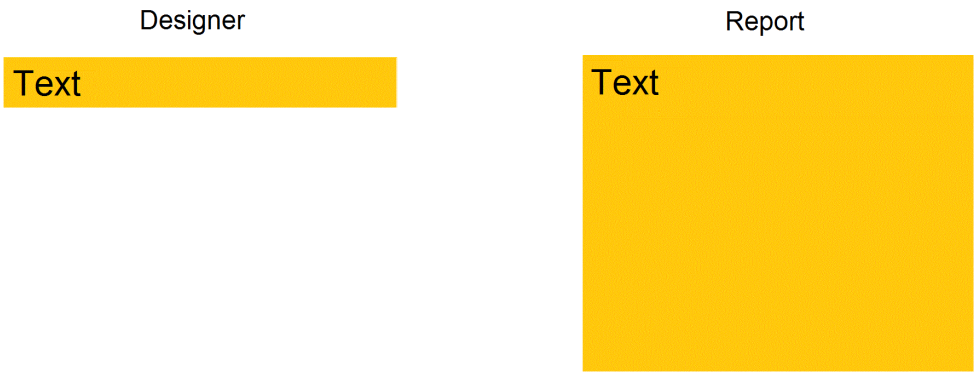
– Never.Implies that the object does not need to shift.

– WhenOverlapped. Implies that the object needs to shift in that case, if the expanding object is located exactly over it (that is, both objects overlap horizontally).

This property is convenient to use when printing information in a table form, when several cells of the table are located on top of each other and can have a variable amount of text.

3.1.3. GrowToBottom property

Every report object has this property. When printing an object with this property, it stretches up to the bottom edge of a band:



This is needed when printing information in a table form. In a table row there can be several objects which can stretch. This property makes it possible to set all objects' height to the maximum height of the band.

3.1.4. Anchor property

Every report object has this property. It determines how the object will be changing its position and/or its size when the container on which it is laying will be changing its size. By using Anchor, it can be done in such a way that, the object expands or moves synchronously with its container.

The container, being referred to, in many cases will be the band. But this is not a must - this can also be the "Table" or "Matrix" objects.

The "Anchor" property can have one of the following values, and also any combination of them:

Value	Description
Left	Anchors the left edge of the object. When the container's size will be changing, the object will not be moving left/right.
Top	Anchors the top edge of the object .When the container's height will be changing, the object will not be moving up/down.
Right	Anchors the right edge of the object .When the container's width will be changing , the distance between the right edge of the object and the container will be constant. If the left edge of the container is anchored as well, then the object will be growing and shrinking synchronously with container.
Bottom	Anchors the bottom edge of the object. When the container's height will be changing, the distance between the bottom edge of the object and the container will be constant. If the top edge of the object is anchored as well, the object will be growing and shrinking synchronously with container.

By default, the value of this property is Left, Top. This means that, when the container's size will be changing, the object will not be changing. In the table below, combinations of some frequent used values are given:

Value	Description
Left, Top	Value by default. The object does not change when the size of the container changes.

Left, Bottom	The object moves up/down when the height of the container changes. The position of the object in relation to the bottom edge of the container does not change.
Left, Top, Bottom	When the height of the container is changing, the height of the object synchronously changes with it.
Left, Top, Right, Bottom	When the width and the height of the container are changing, the object grows or shrinks synchronously with it.

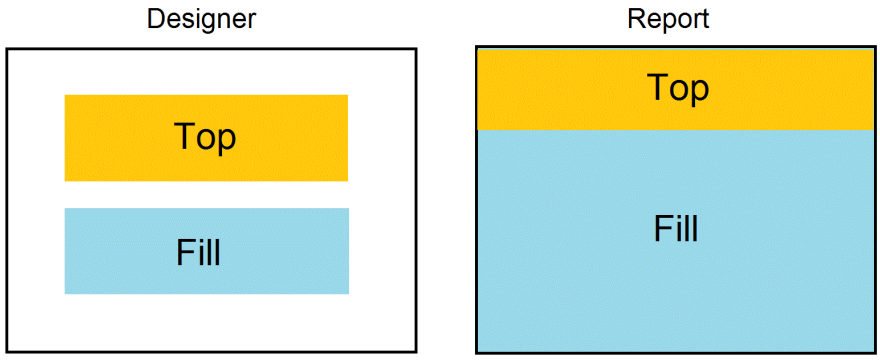
3.1.5. Dock property

Every report object has this property. This property determines on which side of the container the object will be docked.

The "Dock" property can have one of the following values:

Value	Description
None	Value by default. The object is not docked.
Left	The object is docked to the left edge of the container. The height of the object will be equal to the height of the container*.
Top	The object is docked to the top edge of the container. The width of the object will be equal to the width of the container*.
Right	The object is docked to the right edge of the container. The height of the object will be equal to the height of the container*.
Bottom	The object is docked to the lower edge of the container. The width of the object will be equal to the width of the container*.
Fill	The object occupies all the free space of the container.

* This is not quite so, if several objects have been docked at the same time. The figure below shows two objects, the first one has been docked to the top edge of the container and the second - to the left:



As seen, the height of the second object is equal to height of the free space, which remains after docking the first object.

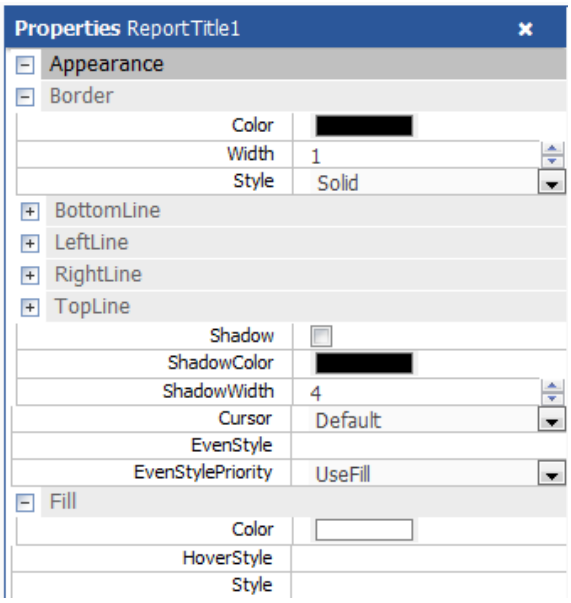
The docking behavior depends on the object's creation order. You can change the order in the context menu of an object. To do this, select either the "Bring to front" or "Send to back" menu items

3.2. Formatting

Consider the tools of changing appearance and format of the data.

3.2.1. Border and fill

Most of the objects in FastReport, can have a frame and fill. For this purpose there are the properties Border and Fill.



To add a border of a component you need to use the toolbar "Borders" in the tab "Home". The color, width and style can be set for each border separately (in the properties window).

As we can see from the figure, you can add the entire frame, or add the separate borders. It is also possible to change the style and width of borders.

The border color can be set in the property "Border".

The fill color can be set on the toolbar "Borders", or in the properties of the component.

3.2.2. Text formatting

The toolbars "Font" and "Alignment" are located on the tab "Home":

Here you can: choose and customize the font, change the position of text within a component (left, right, center) vertically and horizontally, set the text color.

3.2.3. Data formatting

The text component displays data in the format in which they are stored in the data source. It is not always convenient. For example, when the date contains the time. To display only the date, it is necessary to resort to formatting the data. This can be done through using system functions String.Format. Print the current date without time:

Today, [String.Format("{0:d}", [Date])]

3.2.4. Hiding values

The "Text" object has the "HideZeros" property which can be used to hide zero values. Lets look at an object with the following contents:

Total elements: [CountOfElements]

If the value of variable CountOfElements is equal to 0, and the property HideZeros is set to true, the object will be printed as follows:

Total elements:

The "Text" object also has the "HideValue" property which can be used to hide the value of an expression which is equal to the given value. For example, if the property value is "0", then all the zero fields will be hidden. This property can also be used for hiding zero dates. As a rule, it's a date like "1.1.0001" or "1.1.1900". In this case the value of "HideValue" property must be like this:

1.1.1900 0:00:00

As you can see, apart from the date, you need to indicate time as well. This is necessary because the value of the date in .Net contains time also.

Important note: this mechanism depends on the regional settings of your system, which can be set in the control panel. This happens because FastReport compares strings using the "ToString()" method. This method converts an expression value into a string. In relation with this, be careful when building reports which can be launched on a computer with different regional settings.

Finally, the "NullValue" property of the "Text" object allows to print some text instead of a null value. It is often used to print the dash instead of a null value. Lets look at an object with the following contents:

Total elements: [CountOfElements]

If the value of variable CountOfElements is null, and the property NullValue is set to --, the object will be printed as follows:

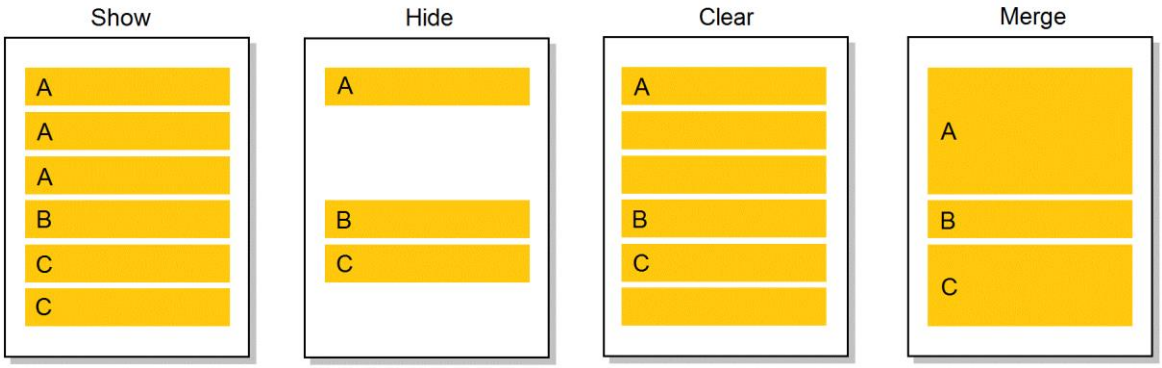
Total elements: --

The "Text" object has "Duplicates" property which allows to control how duplicate values will be printed. This property can be used if the "Text" object is on the "Data" band. The values considered duplicate if they are printed in the near by data rows.

The "Duplicates" property can have one of the following values:

- Show - show the duplicates (by default).
- Hide - hide the object with duplicate value.
- Clear - clear the object's text, but show the object.
- Merge - merge several objects with the same value.

The difference between these modes are shown in the figure below:



3.3. Subreports

Often, in the development of complex reports, the structure becomes very confusing which prevents further development. In such cases, it is appropriate to use the component "Subreport". Put it in band "data". We get a new page of the report, where we can develop a part of the report without being distracted by the structure of the main part of the report.

Component "Subreport" looks like this:

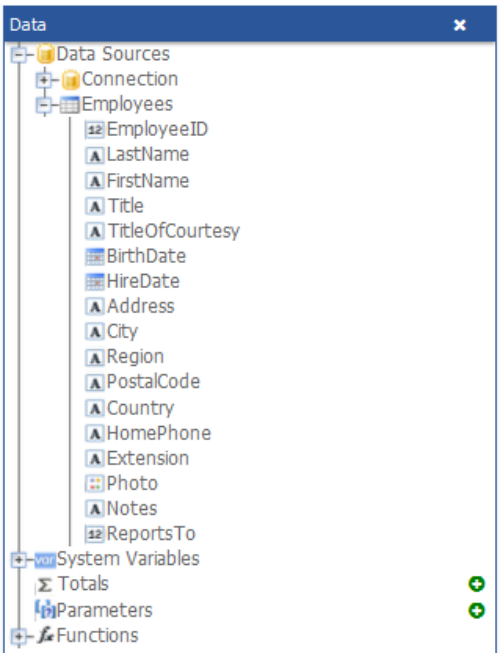
Let’s consider building a report with a subreport:

- printing the bands of the main report, until the component "Subreport";
- printing the bands of the subreport;
- continue to print the main report.

4. Working with data

4.1. Data source

In FastReport Online Designer there is no possibility to add data sources. But if you open the report with the added data source, it will be shown in the "Data" window. You can drag fields from the data source to the report page. This will create a text object with reference to the data field.



The source data can contain related tables. That is, the tables have a relationship on the key field that allows you to create reports of type "Master-detail". One record in the primary table will match one or more records in the detail table.

The figure shows a table Products, which is connected with the table of Categories. FastReport Online Designer can not create the relationship, but allows them to use when working with the already created reports.

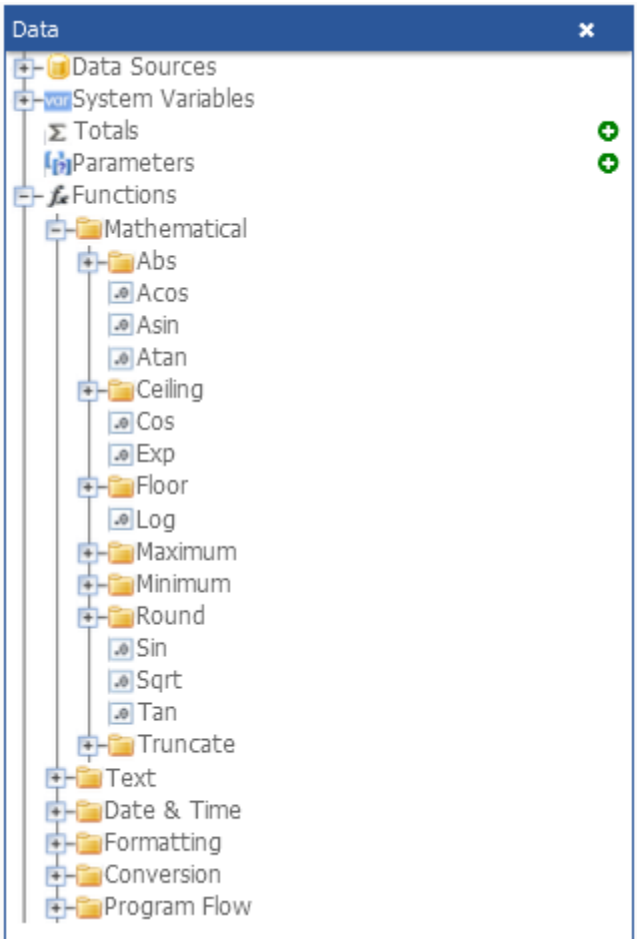
4.2. System variables

There is a list of system variables that can be used in a report:

Variable	Description
Date	Date and time of the report's start.
Page	Current page number.
TotalPages	Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report|Properties..." menu.
PageN	Page number in the form: "Page N".
PageNofM	Page number in the form: "Page N of M".
Row#	Data row number inside the group. This value is reset at the start of a new group.
AbsRow#	Absolute number of data row. This value is never reset at the start of a new group.
Page#	Current page number. If you join several prepared reports into one package, this variable will return current page number in a package. This variable is actually a macro. Its value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression.
TotalPages#	Total number of pages in the report. If you join several prepared reports into one package, this variable will return the number of pages in a package. You don't need to use double pass to get the correct value. This variable is actually a macro. It value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression.
HierarchyLevel	Current level of hierarchy in a hierarchical report (see "Printing hierarchy"). The top level is equal to 1.
HierarchyRow#	Full row number like "1.2.1" in a hierarchical report.

4.3. Functions

FastReport.Net contains a lot of built-in functions (over 60). All functions are splitted into several categories and are accessible through the "Data" window:



You may use a function in any expression, in the script, or print its value in the "Text" object. For example, the following text in the "Text" object:

[Sqrt(4)]

will be printed as "2" (square root of 4).

The following expression will return 4:

Sqrt(4) + 2

Let us look at the ways to insert a function in a report:

– you may drag&drop a function from the "Data" window to a report page. The "Text" object will be created, which contains a function call. You have to edit the text to add parameters to the function call;

– you can drag&drop a function to the script code;

– in the expression editor, you can see a copy of the "Data" window which acts the same way - you may drag items from it and drop them in the expression text.

Below we will describe each function in details.

4.3.1. Mathematical

4.3.1.1. Abs

Function	Parameters	Return value
Abs	sbyte value	sbyte
Abs	short value	short
Abs	int value	int
Abs	long value	long
Abs	float value	float
Abs	double value	double
Abs	decimal value	decimal

Returns the absolute value.

Example:

```
Abs(-2.2) = 2.2
```

4.3.1.2. Acos

Function	Parameters	Return value

Acos	double d	double

Returns the angle (in radians) whose cosine is d. d must be in range between -1 and 1.

Example:

```
Acos(0) * 180 / Math.PI = 90
```

4.3.1.3. Asin

Function	Parameters	Return value
Asin	double d	double

Returns the angle (in radians) whose sine is d. d must be in range between -1 and 1.

Example:

```
Asin(0) = 0
```

4.3.1.4. Atan

Function	Parameters	Return value
Atan	double d	double

Returns the angle (in radians) whose tangent is d.

Example:

```
Atan(1) * 180 / Math.PI = 45
```

4.3.1.5. Ceiling

Function	Parameters	Return value

Ceiling	double d	double
Ceiling	decimal d	decimal

Returns the smallest integer greater than or equal to d.

Example:

```
Ceiling(1.7) = 2
```

4.3.1.6. Cos

Function	Parameters	Return value
Cos	double d	double

Returns the cosine of the specified angle (d). The angle must be in radians.

Example:

```
Cos(90 * Math.PI / 180) = 0
```

4.3.1.7. Exp

Function	Parameters	Return value
Exp	double d	double

Returns *e* (2.71828), raised to the specified power d.

Example:

```
Exp(1) = 2.71828
```

4.3.1.8. Floor

Function	Parameters	Return value
Floor	double d	double
Floor	decimal d	decimal

Returns the largest integer less than or equal to d.

Example:

```
Floor(1.7) = 1
```

4.3.1.9. Log

Function	Parameters	Return value
Log	double d	double

Returns the logarithm of a specified number d.

Example:

```
Log(2.71828) = 1
```

4.3.1.10. Maximum

Function	Parameters	Return value
Maximum	int val1, int val2	int
Maximum	long val1, long val2	long
Maximum	float val1,	float

	float val2	
Maximum	double val1, double val2	double
Maximum	decimal val1, decimal val2	decimal

Returns the larger of val1, val2.

Example:

```
Maximum(1,2) = 2
```

4.3.1.11. Minimum

Function	Parameters	Return value
Minimum	int val1, int val2	int
Minimum	long val1, long val2	long
Minimum	float val1, float val2	float
Minimum	double val1, double val2	double
Minimum	decimal val1, decimal val2	decimal

Returns the smaller of val1, val2.

Example:

```
Minimum(1,2) = 1
```

4.3.1.12. Round

Function	Parameters	Return value
Round	double d	double
Round	decimal d	decimal

Rounds d to the nearest integer.

Example:

```
Round(1.47) = 1
```

Function	Parameters	Return value
Round	double d, int digits	double
Round	decimal d, int digits	decimal

Rounds d to a precision specified in the "digits" parameter.

Example:

```
Round(1.478, 2) = 1.48
```

4.3.1.13. Sin

Function	Parameters	Return value
Sin	double d	double

Returns the sine of the specified angle (d). The angle must be in radians.

Example:

```
Sin(90 * Math.PI / 180) = 1
```

4.3.1.14. Sqrt

Function	Parameters	Return value
Sqrt	double d	double

Returns the square root of d.

Example:

```
Sqrt(4) = 2
```

4.3.1.15. Tan

Function	Parameters	Return value
Tan	double d	double

Returns the tangent of the specified angle (d). The angle must be in radians.

Example:

```
Tan(45 * Math.PI / 180) = 1
```

4.3.1.16. Truncate

Function	Parameters	Return value
Truncate	double d	double
Truncate	decimal d	decimal

Calculates the integral part of d.

Example:

```
Truncate(1.7) = 1
```

4.3.2. Text

Note:

– these functions do not modify the string value passed in. Instead, they return a new modified string;

– the first character in a string has 0 index. Keep it in mind when working with functions that take the character index, such as Insert.

4.3.2.1. Asc

Function	Parameters	Return value
Asc	char c	int

Returns an Integer value representing the character code corresponding to a character.

Example:

```
Asc('A') = 65
```

4.3.2.2. Chr

Function	Parameters	Return value
Chr	int i	char

Returns the character associated with the specified character code.

Example:

```
Chr(65) = 'A'
```

4.3.2.3. Insert

Function	Parameters	Return value
Insert	string s, int startIndex, string value	string

Inserts a "value" substring into the "s" string at a specified index position "startIndex" and returns a new string.

Example:

```
Insert("ABC", 1, "12") = "A12BC"
```

4.3.2.4. Length

Function	Parameters	Return value
Length	string s	int

Returns the length of "s".

Example:

```
Length("ABC") = 3
```

4.3.2.5. LowerCase

Function	Parameters	Return value
LowerCase	string s	string

Converts all characters of "s" to lower case and returns a result.

Example:

```
LowerCase("ABC") = "abc"
```

4.3.2.6. PadLeft

Function	Parameters	Return value
PadLeft	string s, int totalWidth	string

Right-aligns the characters in the "s" string, padding with spaces on the left for a total width specified in the "totalWidth" parameter.

Example:

```
PadLeft("ABC", 5) = "  ABC"
```

Function	Parameters	Return value
PadLeft	string s, int totalWidth, char paddingChar	string

Right-aligns the characters in the "s" string, padding with "paddingChar" characters on the left for a total width specified in the "totalWidth" parameter.

Example:

```
PadLeft("ABC", 5, '0') = "00ABC"
```

4.3.2.7. PadRight

Function	Parameters	Return value
PadRight	string s, int totalWidth	string

Left-aligns the characters in the "s" string, padding with spaces on the right for a total width specified in the "totalWidth" parameter.

Example:

```
PadRight("ABC", 5) = "ABC  "
```

Function	Parameters	Return value
PadRight	string s, int totalWidth, char paddingChar	string

Left-aligns the characters in the "s" string, padding with "paddingChar" characters on the right for a total width specified in the "totalWidth" parameter.

Example:

```
PadRight("ABC", 5, '0') = "ABC00"
```

4.3.2.8. Remove

Function	Parameters	Return value
Remove	string s, int startIndex	string

Deletes all the characters from the "s" string beginning at "startIndex" position and continuing through the last position.

Example:

```
Remove("ABCD", 3) = "ABC"
```

Function	Parameters	Return value
Remove	string s, int startIndex, int count	string

Deletes a number of characters specified in the "count" parameter from the "s" string, beginning at a "startIndex" position.

Example:

```
Remove("A00BC", 1, 2) = "ABC"
```

4.3.2.9. Replace

Function	Parameters	Return value
Replace	string s, string oldValue, string newValue	string

Returns a string "s" in which a specified substring "oldValue" has been replaced with another substring "newValue".

Example:

```
Replace("A00", "00", "BC") = "ABC"
```

4.3.2.10. Substring

Function	Parameters	Return value
Substring	string s, int startIndex	string

Retrieves a substring from the "s" string. The substring starts at a character position specified in the "startIndex" parameter.

Example:

```
Substring("ABCDEF", 4) = "EF"
```

Function	Parameters	Return value
Substring	string s, int startIndex, int length	string

Retrieves a substring from the "s" string. The substring starts at a character position specified in the "startIndex" parameter and has a length specified in the "length" parameter.

Example:

```
Substring("ABCDEF", 1, 3) = "BCD"
```

4.3.2.11.

4.3.2.12. TitleCase

Function	Parameters	Return value
TitleCase	string s	string

Converts the specified string to titlecase.

Example:

```
TitleCase("john smith") = "John Smith"
```

4.3.2.13.

4.3.2.14. Trim

Function	Parameters	Return value

Trim	string s	string

Removes all occurrences of white space characters from the beginning and end of the "s" string.

Example:

```
Trim("  ABC  ") = "ABC"
```

4.3.2.15. UpperCase

Function	Parameters	Return value
UpperCase	string s	string

Converts all characters of "s" to upper case and returns a result.

Example:

```
UpperCase("abc") = "ABC"
```

4.3.3. Date and time

4.3.3.1. AddDays

Function	Parameters	Return value
AddDays	DateTime date, double value	DateTime

Adds the specified number of days ("value") to the "date" date and returns a new date.

Example:

```
AddDays(#7/29/2009#, 1) = #7/30/2009#
```

4.3.3.2. AddHours

Function	Parameters	Return value
AddHours	DateTime date, double value	DateTime

Adds the specified number of hours ("value") to the "date" date and returns a new date.

Example:

```
AddHours(#7/29/2009 1:30#, 1) = #7/29/2009 2:30#
```

4.3.3.3. AddMinutes

Function	Parameters	Return value
AddMinutes	DateTime date, double value	DateTime

Adds the specified number of minutes ("value") to the "date" date and returns a new date.

Example:

```
AddMinutes(#7/29/2009 1:30#, 1) = #7/29/2009 1:31#
```

4.3.3.4. AddMonths

Function	Parameters	Return value
AddMonths	DateTime date, int value	DateTime

Adds the specified number of months ("value") to the "date" date and returns a new date.

Example:

```
AddMonths(#7/29/2009#, 1) = #8/29/2009#
```

4.3.3.5. AddSeconds

Function	Parameters	Return value
AddSeconds	DateTime date, double value	DateTime

Adds the specified number of seconds ("value") to the "date" date and returns a new date.

Example:

```
AddSeconds(#7/29/2009 1:30:01#, 1) = #7/29/2009 1:30:02#
```

4.3.3.6. AddYears

Function	Parameters	Return value
AddYears	DateTime date, int value	DateTime

Adds the specified number of years ("value") to the "date" date and returns a new date.

Example:

```
AddYears(#7/29/2009#, 1) = #7/29/2010#
```

4.3.3.7. DateDiff

Function	Parameters	Return value
DateDiff	DateTime date1, DateTime date2	TimeSpan

Returns the interval (number of days, hours, minutes, seconds) between two dates.

Example:

```
DateDiff(#1/2/2009#, #1/1/2009#) = 1.00:00:00
```

4.3.3.8. DateSerial

Function	Parameters	Return value
DateSerial	int year, int month, int day	DateTime

Creates a new DateTime value from the specified year, month and day.

Example:

```
DateSerial(2009, 7, 29) = #7/29/2009#
```

4.3.3.9. Day

Function	Parameters	Return value
Day	DateTime date	int

Gets the day of the month (1-31) represented by the specified date.

Example:

```
Day(#7/29/2009#) = 29
```

4.3.3.10. DayOfWeek

Function	Parameters	Return value
DayOfWeek	DateTime date	string

Gets the localized name of the day of the week represented by the specified date.

Example:

```
DayOfWeek(#7/29/2009#) = "wednesday"
```

4.3.3.11. DayOfYear

Function	Parameters	Return value
DayOfYear	DateTime date	int

Gets the day of the year (1-365) represented by the specified date.

Example:

```
DayOfYear(#7/29/2009#) = 210
```

4.3.3.12. DaysInMonth

Function	Parameters	Return value
DaysInMonth	int year, int month	int

Returns the number of days in the specified month and year.

Example:

```
DaysInMonth(2009, 7) = 31
```

4.3.3.13. Hour

Function	Parameters	Return value
Hour	DateTime date	int

Gets the hour component (0-23) represented by the specified date.

Example:

```
Hour(#7/29/2009 1:30#) = 1
```

4.3.3.14. Minute

Function	Parameters	Return value
Minute	DateTime date	int

Gets the minute component (0-59) represented by the specified date.

Example:

```
Minute(#7/29/2009 1:30#) = 30
```

4.3.3.15. Month

Function	Parameters	Return value
Month	DateTime date	int

Gets the month component (1-12) represented by the specified date.

Example:

```
Month(#7/29/2009#) = 7
```

4.3.3.16. MonthName

Function	Parameters	Return value
MonthName	int month	string

Gets the localized name of the specified month (1-12).

Example:

```
MonthName(1) = "January"
```

4.3.3.17. Second

Function	Parameters	Return value
Second	DateTime date	int

Gets the seconds component (0-59) represented by the specified date.

Example:

```
Second(#7/29/2009 1:30:05#) = 5
```

4.3.3.18. Year

Function	Parameters	Return value
Year	DateTime date	int

Gets the year component represented by the specified date.

Example:

```
Year(#7/29/2009#) = 2009
```

4.3.4. Formatting

4.3.4.1. Format

Function	Parameters	Return value
Format	string format,	string

	params object[] args	

Replaces the format item in a specified "format" string with the value of a corresponding Object instance in a specified "args" array.

For example, the following function call:

Format("Name = {0}, hours = {1:hh}", myName, DateTime.Now)

contains the following format items: "{0}" and "{1:hh}". They will be replaced with values of myName and DateTime.Now. The result may look as follows:

Name = Alex, hours = 12

Each format item takes the following form:

{index[,alignment][:formatString]}

– index - a zero-based integer that indicates which element in a list of objects to format;

– alignment - an optional integer indicating the minimum width of the region to contain the formatted value. If the length of the formatted value is less than alignment, then the region is padded with spaces. If alignment is negative, the formatted value is left justified in the region; if alignment is positive, the formatted value is right justified;

– formatString -an optional string of format specifiers.

The following table describes the standard numeric format strings.

Format Specifier	**Name**	**Description**
C or c	Currency	The number is converted to a string that represents a currency amount. *Format("{0:C}", 10) = "$10.00"*
D or d	Decimal	This format is supported for integral types only. The number is converted to a string of decimal digits (0-9). *Format("{0:D}", 10) = "10"*
E or e	Scientific	The number is converted to a string of the form

		"-d.ddd…E+ddd" or "-d.ddd…e+ddd", where each 'd' indicates a digit (0-9). *Format("{0:E}", 10) = "1,000000E+001"*
F or f	Fixed-point	The number is converted to a string of the form "-ddd.ddd…" where each 'd' indicates a digit (0-9). *Format("{0:F}", 10) = "10.00"*
G or g	General	The number is converted to the most compact notation. *Format("{0:G}", 10) = "10"*
N or n	Number	The number is converted to a string of the form "-d,ddd,ddd.ddd…", where each 'd' indicates a digit (0-9). *Format("{0:N}", 1234.56) = "1,234.56"*
P or p	Percent	The number is converted to a string that represents a percent. The converted number is multiplied by 100 in order to be presented as a percentage. *Format("{0:P}", 0.15) = "15.00%"*
X or x	Hexadecimal	The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. For example, use 'X' to produce "ABCDEF", and 'x' to produce "abcdef". *Format("{0:X}", 26) = "1A"*

If you format the floating-point values, you may indicate a number of decimal places after the format string:

Format("{0:C1}", 12.23) = "$12.2"

If the standard numeric format specifiers do not provide the type of formatting you require, you can use custom format strings:

Format character	Description
0	Zero placeholder. If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the result string. The position of the leftmost '0' before the decimal point and the rightmost '0' after the decimal point determines the range of digits that are always present in the result string.
#	Digit placeholder. If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string.
.	Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value.
,	Thousand separator. If the format string contains a ',' character, then the output will have thousand separators inserted between each group of three digits to the left of the decimal separator.
%	Percentage placeholder. The presence of a '%' character in a format string causes a number to be multiplied by 100 before it is formatted.
;	Section separator. The ';' character is used to separate sections for positive, negative, and zero numbers in the format string.

Examples of use:

Format("{0:$#,##0.00}", 1024.25) = "$1,024.25"
Format("{0:00%}", 0.25) = "25%"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 1024.25) = "$1,024.25"
Format("{0:$#,##0.00;($#,##0.00);Zero}", -1024.25) = "($1,024.25)"

Format("{0:$#,##0.00;($#,##0.00);Zero}", 0) = "Zero"

The following table describes the standard format specifiers for formatting the DateTime values:

Format Specifier	Name	Example
d	Short date pattern	"8/9/2009"
D	Long date pattern	"Sunday, August 09, 2009"
f	Full date/time pattern (short time)	"Sunday, August 09, 2009 2:44 PM"
F	Full date/time pattern (long time)	"Sunday, August 09, 2009 2:44:01 PM"
g	General date/time pattern (short time)	"8/9/2009 2:44 PM"
G	General date/time pattern (long time)	"8/9/2009 2:44:01 PM"
t	Short time pattern	"2:44 PM"
T	Long time pattern	"2:44:01 PM"

The following table describes the custom date/time format specifiers and the results they produce.

Format Specifier	Description
d	Displays the current day of the month, measured as a number between 1 and 31, inclusive. If the day is a single digit only (1-9), then it is displayed as a single digit.
dd	Displays the current day of the month, measured as a number between 1 and 31,

	inclusive. If the day is a single digit only (1-9), it is formatted with a preceding 0 (01-09).
ddd	Displays the abbreviated name of the day.
dddd	Displays the full name of the day.
f or F	Displays the most significant digit of the seconds fraction.
h	Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is displayed as a single digit.
hh	Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).
H	Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is displayed as a single digit.
HH	Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).
m	Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is displayed as a single digit.
mm	Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is formatted with a preceding 0 (01-09).
M	Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is displayed as a single digit.
MM	Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is formatted with a preceding 0 (01-09).
MMM	Displays the abbreviated name of the month.
MMMM	Displays the full name of the month.
s	Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is displayed as a single digit only.

ss	Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is formatted with a preceding 0 (01-09).
t	Displays the first character of the A.M./P.M. designator.
tt	Displays the A.M./P.M. designator.
y	Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is displayed as a single digit.
yy	Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is formatted with a preceding 0 (01-09).
yyyy	Displays the year, including the century. If the year is less than four digits in length, then preceding zeros are appended as necessary to make the displayed year four digits long.
z	Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading sign (zero is displayed as "+0"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is –12 to +13. If the offset is a single digit (0-9), it is displayed as a single digit with the appropriate leading sign.
zz	Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is –12 to +13. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign.
zzz	Displays the time zone offset for the system's current time zone in hours and minutes. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00:00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is –12:00 to +13:00. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign.
:	Time separator.

/	Date separator.

Examples of use:

Format("{0:d MMM yyyy}", DateTime.Now) = "9 Aug 2009"

Format("{0:MM/dd/yyyy}", DateTime.Now) = "08/09/2009"

Format("{0:MMMM, d}", DateTime.Now) = "August, 9"

Format("{0:HH:mm}", DateTime.Now) = "16:07"

Format("{0:MM/dd/yyyyhh:mmtt}", DateTime.Now) = "08/09/2009 04:07 PM"

4.3.4.2. FormatCurrency

Function	Parameters	Return value
FormatCurrency	object value	string

Formats the specified value as a currency, using the Windows regional settings.

Example:

FormatCurrency(1.25) = "$1.25"

Function	Parameters	Return value
FormatCurrency	object value, int decimalDigits	string

Formats the specified value as a currency. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

Example:

```
FormatCurrency(1.25, 1) = "$1.3"
```

4.3.4.3. FormatDateTime

Function	Parameters	Return value
FormatDateTime	DateTime value	string

Formats the specified value as a date/time, using the Windows regional settings. This function does not include neutral values in the resulting string.

Example:

```
FormatDateTime(#1/1/2009#) = "01/01/2009"
FormatDateTime(#1/1/2009 1:30#) = "01/01/2009 1:30:00 AM"
FormatDateTime(#1:30#) = "1:30:00 AM"
```

Function	Parameters	Return value
FormatDateTime	DateTime value, string format	string

Formats the specified value as a date/time, using the named format specified in the "format" parameter. The valid values for this parameter are:

"Long Date"

"Short Date"

"Long Time"

"Short Time"

Example:

```
FormatDateTime(#1/1/2009 1:30#, "Long Date") = "Thursday, January 01, 2009"
FormatDateTime(#1/1/2009#, "Short Date") = "01/01/2009"
FormatDateTime(#1:30#, "Short Time") = "01:30 AM"
FormatDateTime(#1:30#, "Long Time") = "1:30:00 AM"
```

4.3.4.4. FormatNumber

Function	Parameters	Return value
FormatNumber	object value	string

Formats the specified value as a number, using the Windows regional settings.

Example:

```
FormatNumber(1234.56) = "1,234.56"
```

Function	Parameters	Return value
FormatNumber	object value, int decimalDigits	string

Formats the specified value as a number. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

Example:

FormatNumber(1234.56, 1) = "1,234.6"

4.3.4.5. FormatPercent

Function	Parameters	Return value
FormatPercent	object value	string

Formats the specified value as a percent, using the Windows regional settings.

Example:

```
FormatPercent(0.15) = "15.00%"
```

Function	Parameters	Return value
FormatPercent	object value, int decimalDigits	string

Formats the specified value as a percent. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

Example:

```
FormatPercent(0.15, 0) = "15%"
```

4.3.5. Conversion

4.3.5.1. ToBoolean

Function	Parameters	Return value
ToBoolean	object value	bool

Converts the specified value to boolean.

Example:

```
ToBoolean(1) = true
ToBoolean(0) = false
```

4.3.5.2. ToByte

Function	Parameters	Return value
ToByte	object value	byte

Converts the specified value to byte.

Example:

```
ToByte("55") = 55
```

4.3.5.3. ToChar

Function	Parameters	Return value
ToChar	object value	char

Converts the specified value to char.

Example:

```
ToChar(65) = 'A'
```

4.3.5.4. ToDateTime

Function	Parameters	Return value
ToDateTime	object value	DateTime

Converts the specified value to date/time.

Example:

```
ToDateTime("1/1/2009") = #1/1/2009#
```

4.3.5.5. ToDecimal

Function	Parameters	Return value
ToDecimal	object value	decimal

Converts the specified value to decimal.

Example:

```
ToDecimal(1) = 1m
ToDecimal("1") = 1m
```

4.3.5.6. ToDouble

Function	Parameters	Return value
ToDouble	object value	double

Converts the specified value to double.

Example:

```
ToDouble(1) = 1
ToDouble("1") = 1
```

4.3.5.7. ToInt32

Function	Parameters	Return value
ToInt32	object value	int

Converts the specified value to int.

Example:

```
ToInt32(1f) = 1
ToInt32("1") = 1
```

4.3.5.8. ToRoman

Function	Parameters	Return value

ToRoman	object value	string

Converts the specified numeric value to its roman representation. The value must be in range 1-3998.

Example:

```
ToRoman(9) = "IX"
```

4.3.5.9. ToSingle

Function	Parameters	Return value
ToSingle	object value	float

Converts the specified value to float.

Example:

```
ToSingle(1m) = 1f
ToSingle("1") = 1f
```

4.3.5.10. ToString

Function	Parameters	Return value
ToString	object value	string

Converts the specified value to string.

Example:

```
ToString(false) = "False"
ToString(DateTime.Now) = "08/09/2009 4:45:00 PM"
```

4.3.5.11. ToWords

Function	Parameters	Return value
ToWords	object value	string

Converts the specified currency value to words. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

"USD"

"EUR"

"GBP"

Example:

ToWords(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"

Function	Parameters	Return value
ToWords	object value, string currencyName	string

Converts the specified currency value to words. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

"USD"

"EUR"

"GBP"

Example:

ToWords(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"

Function	Parameters	Return value

ToWords	object value, string one, string many	string

Converts the specified integer value to words. The "one" parameter contains the name in singular form; the "many" parameter contains the name in plural form.

Example:

ToWords(124, "page", "pages") = "One hundred and twenty-four pages"

ToWords(1, "page", "pages") = "One page"

4.3.5.12. ToWordsEnGb

Function	Parameters	Return value
ToWordsEnGb	object value	string

Converts the specified currency value to words in Great Britain english. There are the following differences between this function and ToWords:

- the GBP currency is used by default;
- different words are used when converting billions and trilliards.

Example:

ToWordsEnGb(121) = "One hundred and twenty-one pounds and 00 pence"

Function	Parameters	Return value
ToWordsEnGb	object value, string currencyName	string

Converts the specified currency value to words in Great Britain english. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

"USD"

"EUR"

"GBP"

Example:

ToWordsEnGb(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"

Function	Parameters	Return value
ToWordsEnGb	object value, string one, string many	string

Converts the specified integer value to words in Great Britain english. The "one" parameter contains the name in singular form; the "many" parameter contains the name in plural form.

Example:

ToWordsEnGb(124, "page", "pages") = "One hundred and twenty-four pages"

ToWordsEnGb(1, "page", "pages") = "One page"

4.3.5.13. ToWordsRu

Function	Parameters	Return value
ToWordsRu	object value	string

Converts the specified currency value to words in russian.

Example:

ToWordsRu(1024.25) = "Одна тысяча двадцать четыре рубля 25 копеек"

Function	Parameters	Return value
ToWordsRu	object value, string currencyName	string

Converts the specified currency value to words in russian. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

"RUR"

"UAH"

"USD"

"EUR"

Example:

ToWordsRu(1024.25, "EUR") = "Одна тысяча двадцать четыре евро 25 евроцентов"

Function	Parameters	Return value
ToWordsRu	object value, bool male, string one, string two, string many	string

Converts the specified integer value to words in russian. The "male" parameter indicates the gender of the name. The "one", "two" and "five" parameters contain a form of the name used with "1", "2" and "5" numbers.

Example:

```
// the "страница" word is of female gender, male = false
ToWordsRu(122, false, "страница", "страницы", "страниц") =
"Стодвадцатьдвестраницы"

// the "лист" word is of male gender, male = true
ToWordsRu(122, true, "лист", "листа", "листов") =
"Сто двадцать два листа"
```

4.3.6. Program Flow

4.3.6.1. Choose

Function	Parameters	Return value
Choose	doubleindex, paramsobject[] choice	object

Returns an element of the "choice" array with the index specified in the "index" parameter. The first array element has 1 index.

Example:

```
Choose(2, "one", "two", "three") = "two"
```

4.3.6.2. IIf

Function	Parameters	Return value
IIf	bool expression, object truePart, object falsePart	object

Returns the "truePart" value, if the "expression" is true. Otherwise, returns the "falsePart" value.

Example:

```
IIf(2 > 5, "true", "false") = "false"
```

4.3.6.3. Switch

Function	Parameters	Return value
Switch	params object[] expressions	object

The argument supplied to *expressions* consists of paired expressions and values. The Switch function evaluates the odd-numbered expressions from lowest to highest index, and returns the even-numbered value associated with the first expression that evaluates to True.

Example:

```
// returns one of the following values - "a greater than 0",
// "a less than 0", "a equals to 0", depending on "a" value
Switch(
a> 0, "a greater than 0",
a< 0, "a less than 0",
a == 0, "a equals to 0")
```

4.4. Totals

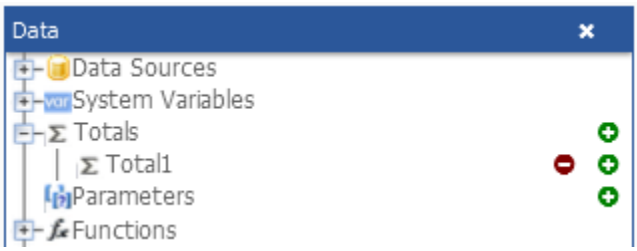
In many reports we may need to show some total information: sum of the group, number of rows in the list, and many others. FastReport uses totals to perform this task. For the total, you need to indicate the following parameters:

- The total function type;
- The expression, which is supposed to be calculated. For the "Count" function, you do not need to indicate the expression;
- The condition. The function will be calculated if the condition is met. It's not obligatory to set up the condition.
- The data band, for which the function will be processed;
- The band, in which the total value will be printed.

The list of total functions is given below:

Function	Description
Sum	Calculates the sum of the expression.
Min	Calculates the minimum value of the expression.
Max	Calculates the maximum value of the expression.
Average	Calculates the average value of the expression.
Count	Returns the number of rows.

To add Total, you should open the window "Data". Click on the green icon with a cross in front node Totals:



After that, set the above parameters in the "Properties". Then, drag the added Total to the report page.

To remove added Total, use the red icon opposite the Total name in the window "Data".

4.5. Report parameters

You can define parameters in a report. Parameter is a variable, the value of which can be defined both in a report itself and outside of it (a program, calling a report can transfer parameters values into it. See details in "Programmer's manual"). A parameter can be used in expressions and be displayed in report objects like the "Text" object.

Most common methods of using parameters:

– data filtering by condition set in a parameter;

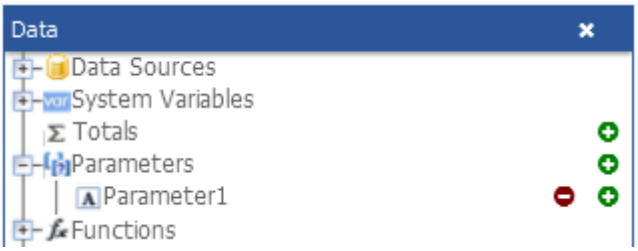
– printing parameter value in a report.

A parameter has the following properties:

Property	Description
Name	Parameter's name can have any symbols except dot ".".
DataType	Parameter data type.
Expression	Expression which returns parameter's value. More details about expressions can be found in the "Expression" chapter. This expression will be processed when calling a parameter.
Value	Parameter value. This property is not available in the designer and can be filled programmatically.

You have to set up "Name" and "DataType" properties. The "Expression" property can be left empty. In this case parameter's value should be passed programmatically.

To add Total, you should open the window "Data". Click on the green icon with a cross in front node Parameters. After that, set the above parameters in the "Properties". Now, you can drag parameter to the desired location on the report page, or use it to filter the data.



To remove added Parameter, use the red icon opposite the Parameter name in the window "Data".

5. Expressions

In many places in FastReport, expressions are used. For example, the "Text" object can contain expressions in square brackets.

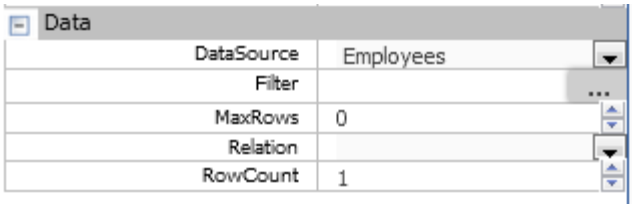
An expression is a code in C# or VB.Net language, which returns any value. For example:

2 + 2

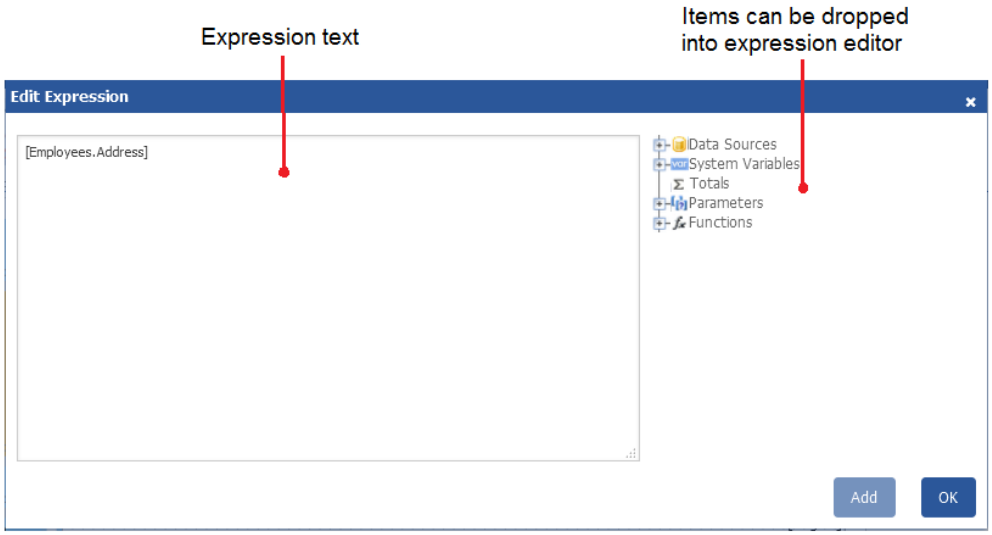
An expression should be written in a language chosen as a script in a report. By default, it is C#. You can change the language in the "Report|Options..." menu by choosing the "Script" element in a window.

5.1. Expression editor

For the convenience of writing expressions use the Expression Editor. It can be opened by double clicking on the text object or by using the button ... in the properties.



The expression editor is a window in which you can enter text expression or insert the items from the window "data":



5.2. Reference to report objects

For referring to report objects, use the object's name. The following example will return the height of the Text1 object:

Text1.Height

For referring to report properties, use Report variable. The following example returns file name from which a report was loaded.

Report.FileName

Besides, you can refer to nested object properties. The following example will return a report name:

Report.ReportInfo.Name

5.3. Using .Net functions

You can use any.Net objects in expressions. The following example demonstrates Max function use:

Math.Max(5, 10)

By default a report uses the following .Net assemblies:

System.dll

System.Drawing.dll

System.Windows.Forms.dll

System.Data.dll

System.Xml.dll

To refer to the function or variable that was defined in a script, just use its name:

myPrivateVariableThatIHaveDeclaredInScript

MyScriptFunction()

In an expression you can use only those functions which return a value

5.4. Reference to data elements

Apart from standard language elements, you can use the following report elements in expressions:

- data source columns;
- system variables;
- total values;
- report parameters.

All these elements are contained in the "Data" window. Any of these elements can be used in an expression, by including it in square brackets. For example:

[Page] + 1

This expression returns next printed page number. A system variable "Page", which returns current report page number, is used in the expression. It is enclosed in square brackets.

5.5. Reference to data source

For reference to data source columns, the following format is used:

[DataSource.Column]

Source name is separated from column name by a period, for example:

[Employees.FirstName]

Source name can be compound in case, if we refer to a data source by using a relation. See more details in the "Data" section. For example, this is how you can refer to a related data source column:

[Products.Categories.CategoryName]

Let us look at the following example of using columns in an expression:

[Employees.FirstName] + " " + [Employees.LastName]

Here it should be noted that: every column has a definite data type, which is set in the "DataType" property (you can see it in the "Properties" window if to choose data column in the "Data" window beforehand). How a column can be used in an expression depends on its type. For instance, in above mentioned example, both columns - first name and last name - have got a string type and that is why they can be used in such way. In the following example, we will try to use "Employees.Age" column of numeric type, which will lead to an error:

[Employees.FirstName] + " " + [Employees.Age]

The error occurs because you never mix strings with numbers. For this you need to convert the number into a string:

[Employees.FirstName] + " " + [Employees.Age].ToString()

In this case we refer to the "Employees.Age" column as if it is an integer variable. And it is so. We know that all expressions are compiled. All non-standard things (like referring to data columns) from a compiler's point of view are converted into another type, which is understandable to a compiler. So, the last expression will be turned into the following form:

(string)(Report.GetColumnValue("Employees.FirstName")) + " " +

(int)(Report.GetColumnValue("Employees.Age")).ToString()

As seen, FastReport changes reference to data columns in the following way:

[Employees.FirstName] -->(string)(Report.GetColumnValue("Employees.FirstName"))

[Employees.Age] -->(int)(Report.GetColumnValue("Employees.Age"))

That is, we can use data column in an expressions as if it is a variable having a definite type. For example, the following expression will return the first symbol of an employee's name:

[Employees.FirstName].Substring(0, 1)

5.6. Reference to system variables

You can use the following system variables in expressions (they are accessible in the "Data" window):

Variable	.Net data type	Description
Date	DateTime	Date and time of the report's start.

Page	int	Current page number.
TotalPages	int	Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report|Properties..." menu.
PageN	string	Page number in the form: "Page N".
PageNofM	string	Page number in the form: "Page N of M".
Row#	int	Data row number inside the group. This value is reset at the start of a new group.
AbsRow#	int	Absolute number of data row. This value is never reset at the start of a new group.

Every variable has a definite data type. And how it will be used in the expression depends thereon. Here is an example of an expression where date is being used:

[Date].Year

This expression returns the current year. As "Date" variable has DateTime type, we can refer to its "Year" property. We can get the current month similarly ([Date].Month).

FastReport converts reference to system variable into the following form (for example, the "Date" variable):

```
((DateTime)Report.GetVariableValue("Date"))
```

5.7. Reference to Total values

In order to refer to a total value, use its name:

[TotalSales]

FastReport converts reference to totals into the following form:

Report.GetTotalValue("TotalSales")

As you can see, the data type is not used here. It is so, because of the total value is of FastReport.Variant type. It can be used directly in any expressions because it is automatically converted to any type. For example:

*[TotalSales] * 0.2f*

5.8. Reference to report parameters

In order to refer to report parameters, you should use its name:

[Parameter1]

Parameters can be nested. In this case, you should use both parent and child parameter names in the following form:

[ParentParameter.ChildParameter]

Parameters have a definite data type. It is set in the "DataType" property of the parameter. The way it can be used in an expression depends on parameter's data type.

FastReport converts reference to a report parameter into the following way:

```
((string)Report.GetParameterValue("Parameter1"))
```

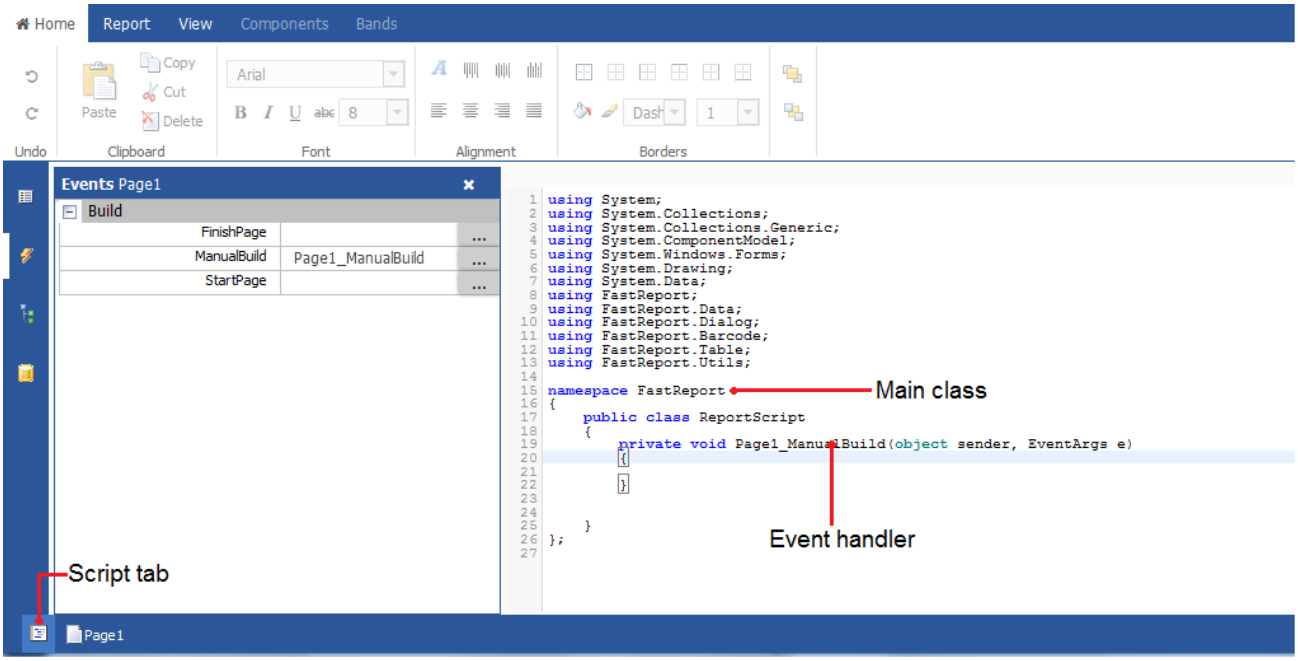
6. Script

A distinctive feature of FastReport is the ability to create scripts in the report which allows you to control the behavior of the report. Script is a higher-level programming language, which is a part of the report. Script can be written by the language #C:

A script is applicable in many places. Using the script, you can do the following:

- perform data handling, which cannot be done via regular means of the FastReport engine;
- control the printing of report pages and bands on the page;
- control the interaction between elements on dialog forms;
- control the formation of dynamic "Table" objects;
- and many more.

In order to see the report's script, switch to the "Code" tab in the designer:



In the figure you can see the window "Event" for the object Page1. This window allows you to create event handlers for components and bands.

Contrary to other report generators, script in FastReport contains only what you have written. In the script, you can:

- add your variables, methods and properties to the main script class;
- create a report object's events handler;

– add new classes to the script, if needed. A class can be added either before the ReportScript main class or after it.

You cannot:

– delete, rename or change the visibility area of the ReportScript main class;

– rename a namespace in which the main class is located.

When the report is running, the following occurs:

– FastReport adds into the script a list of variables, whose names correspond with the names of the report objects. This is done before compiling the script and allows you to refer to the report objects by their names;

– an expression handler is added to the script, which handles all expressions found in the report;

– a script is compiled, if it is not empty;

– the script class is initialized;

– the report is run.

6.1. Events

Event handlers are needed to determine the behavior of a component or band. To add an event, select the report object and open the "Event". Select the handler and click on the icon ... .

To delete an added event handler, delete the name in the "Event" and the body of the handler on tab "Script".

The object "Report" can also have events. To select the object "Report", open the "Report tree" and select Report1. Now the "Event"window displays event handlers available for the report.

You can control the report behavior at each step of its construction by using event handlers. Consider the sequence of events when building a simple report with a band "Data":

1. When constructing a report called the first event StartReport for the object "Report";
2. Called the event StartPage of the page before the formation of the page. This event is called once for each page of the report template (but not the pages of the finished report);
3. Starts printing the lines of band "Data". Strings are printed until there is data in the source:
   a. called event BeforePrint for the band;

b. called event BeforePrint for all the components of the band;
c. all the components are filled with data;
d. called event AfterData for all the components of the band;
e. called event BeforeLayout for the band;
f. components are placed on the band; calculated height of the band;
g. called event AfterLayout for the band;
h. if the band does not fit in the space of the page, a new page is created;
i. the band and all its components are displayed on the page of a finished report;
j. called event AfterPrint for the band;
k. called event AfterPrint for all the components of the band.

4. Called event FinishPage;
5. The final step - an event FinishReport object "Report".

6.2. Reference to report objects

For referring to report objects (for example, "Text" object) use the name of the object. The following example returns the height of Text1 object:

```
float height = Text1.Height;
```

Note that report's native unit of measurement is screen pixels. Keep it in mind when using such object's properties like Left, Top, Width, and Height. To convert pixels into centimeters and back, use the constants defined in the "Units" class:

```
Float heightInPixels = Text1.Height;
Float heightInCM = heightInPixels / Units.Centimeters;
Text1.Height = Units.Centimeters * 5; // 5см
```

6.3. Engine and Report objects

Apart from objects, which are contained in the report, there are two variables defined in the script: Report and Engine.

The Report variable refers to the current report. In the list below, a list of the report object's methods is given:

Method	Description
`object Calc(` `  string expression)`	Calculates an expression and returns the value. When calling this method the first time, an expression gets compiled, which needs some time.
`object GetColumnValue(` `  string complexName)`	Returns the value of the data column. The name must be presented in the "DataSource.Column" form. If the column has got the **null** value, it is converted into a value by default (0, empty string, false).
`object GetColumnValueNullable(` `  string complexName)`	Returns the value of the data column. Contrary to the previous method, it does not get converted into a default value and may be **null**.
`Parameter GetParameter(` `  string complexName)`	Returns the reports parameter with the indicated name. Name can be compounded when referring to the nested parameter: "MainParam.NestedParam".
`object GetParameterValue(` `  string complexName)`	Returns the value of the report parameter with the indicated name.

`void SetParameterValue(`
`  string complexName,`
`  object value)`
	Sets the value of the report parameter with the indicated name.

`object GetVariableValue(` `  string complexName)`	Returns the value of the system variable, for example, "Date".
`object GetTotalValue(` `  string name)`	Returns the value of the total, defined in the "Data" window, by its name.
`DataSourceBaseGetDataSource(` `  string alias)`	Returns the data source, defined in the report, by its name.

The Engine object is an engine that controls the report creation. By using the methods and properties of the engine, you can manage the process of placing bands onto the page. You can use the following properties of the Engine object:

Property	Description
`float CurX`	Current coordinates on the X-axis. A value can be assigned to this property in order to shift the printed object.
`float CurY`	Current printing position on the Y-axis. A value can be assigned to this property to shift the printed object.
`int CurColumn`	Number of the current column in a multicolumn report. The first column has the number 0.
`int CurPage`	Number of the page being printed. This value can be received from the "Page" system variable.
`float PageWidth`	Width of the page minus the size of the left and right

	margins.
`float PageHeight`	Height of the page minus the size of the top and bottom margins.
`float PageFooterHeight`	Height of the page footer (and all its child bands).
`float ColumnFooterHeight`	Height of the column footer (and all of its child bands).
`float FreeSpace`	Size of the free space on the page.
`bool FirstPass`	Returns **true**, if the first (or only) report pass is being executed. Number of passes can be obtained from the Report.DoublePass property.
`bool FinalPass`	Returns **true**, if the last (or only) report pass is being executed.

Engine.PageWidth and Engine.PageHeight properties determine the size of the printing area, which is almost always less than the actual size of the page. Size of the printed area is determined by the page margins, which is given by the LeftMargin, TopMargin, RightMargin and BottomMargin page properties.

Engine.FreeSpace property determines the height of the free space on the page. If there is the "Page footer" band on the page, its height is considered when calculating the FreeSpace. Note that after printing a band, free space is reduced.

How does the formation of a prepared report page take place? FastReport engine displays bands on the page until there is enough space for band output. When there is no free space, the "Report footer" band is printed and a new empty page is formed. Displaying a band starts from the current position, which is determined by the X and Y coordinates. This position is retuned by the Engine.CurX and Engine.CurY properties. After printing a band, CurY automatically increases by the height of the printed band. After forming a new page, the position of the CurY is set to 0. The position of the CurX changes when printing a multicolumn report.

Engine.CurX and Engine.CurY properties are accessible not only for reading, but also for writing. This means that you can shift a band manually by using one of the suitable events. Examples of using these properties can be seen in the "Examples" section.

When working with properties, which return the size or position, remember that these properties are measured in the screen pixels

In the Engine object, the following methods are defined:

Method	Description
`void AddOutline(string text)`	Adds an element into the report outline and sets the current position to the added element.
`void OutlineRoot()`	Sets the current position on the root of the outline.
`void OutlineUp()`	Shifts the current position to a higher-level outline element.
`void AddBookmark(string name)`	Adds a bookmark
`int GetBookmarkPage(string name)`	Returns the page number on which the bookmark with the indicated name is placed.
`void StartNewPage()`	Starts a new page. If the report is multicolumn, a new column is started.

By using the AddOutline, OutlineRoot, OutlineUp methods, you can form the report outline manually. Usually, this is done automatically with the help of the OutlineExpression property, which every band and report page have got.

The AddOutline method adds a child element to the current outline element, and makes it current. The current report page and the current position on the page are associated with the new element. If you call the AddOutline method several times, then you will have the following structure:

Item1
  Item2
    Item3

For controlling the current element, there are OutlineUp and OutlineRoot methods. The first method moves the pointer to the element, located on a higher level. So, the script

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineUp();
Engine.AddOutline("Item4");
```

will create the following outline:

Item1
  Item2
    Item3
    Item4

The OutlineRoot method moves the current element to the root of the outline. For example, the following script:

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineRoot();
Engine.AddOutline("Item4");
```

will create the following outline:

Item1
Item2
Item3
Item4

For working with bookmarks, the AddBookmark and GetBookmarkPage methods of the Engine object are used. Usually bookmarks are added automatically when using the Bookmark property, which all objects of the report have got.

By using the Add Bookmark method, you can add a bookmark programmatically. This method creates a bookmark on the current page at the current printing position.

The GetBookmarkPage method returns the page number on which the bookmark is placed. This method is often used when creating the table of contents, for displaying page numbers. In this case, the report must have a double pass.

6.4. Reference to Data Sources

Contrary to the FastReport expressions, never use square brackets in script for referring to the data sources. Instead of this, use the GetColumnValue method of the Report object, which returns the value of the column:

```
stringproductName = (string)Report.GetColumnValue("Products.Name");
```

As seen, you need to indicate the name of the source and its column. The name of the source can be compound in case, if we are referring to the data source by using a relation. For example, you can refer to a column of the related data source in this way:

```
stringcategoryName =
(string)Report.GetColumnValue("Products.Categories.CategoryName");
```

For making the work easier, use the "Data" window. From it you can drag data elements into the script, during this FastReport automatically creates a code for referring to the element.

For referring to the data source itself, use the GetDataSource method of the Report object:

```
DataSourceBase ds = Report.GetDataSource("Products");
```

Help on properties and methods of the DataSourceBase class can be received from the FastReport.Net Class Reference help system. As a rule, this object is used in the script in the following way:

```
// get a reference to the data source
DataSourceBase ds = Report.GetDataSource("Products");

// initialize it
ds.Init();

// enum all rows
while (ds.HasMoreRows)
{
// get the data column value from the current row
stringproductName = (string)Report.GetColumnValue("Products.Name");
// do something with it...
// ...

// go next data row
ds.Next();
}
```

6.5. Reference to system variables

For reference to system variables, use the GetVariableValue method of the Report object:

```
DateTime date = (DateTime)Report.GetVariableValue("Date");
```

A list of system variables can be seen in the "Data" window. From it, you can drag a variable into a script, during this FastReport automatically creates a code for referring to the variable.

6.6. Reference to Total values

For reference to the total value, use the GetTotalValue method of the Report object:

```
float sales = Report.GetTotalValue("TotalSales");
```

A list of totals can be seen in the "Data" window. From it, you can drag a total into the script, during this FastReport automatically creates a code for referring to the total.

Total value has the FastReport.Variant type. It can be used directly in any expression, because the FastReport.Variant type is automatically converted to any type. For example:

```
float tax = Report.GetTotalValue("TotalSales") * 0.2f;
```

Reference to the total value can be done at that time when, it is being processed. Usually the total is "ready to use" at the moment of printing the band, on which it is located in the report.

6.7. Reference to report parameters

For referring to report parameters, use the GetParameterValue method of the Report object:

```
intmyParam = (int)Report.GetParameterValue("MyParameter");
```

Parameters can be nested. In this case, indicate the name of the parent parameter and after the period, the name of the child parameter:

```
Report.GetParameterValue("ParentParameter.ChildParameter")
```

Parameters have got a definite data type. It is given in the DataType property of the parameter. You must take this into account when referring to parameters. You can see a list of parameters in the "Data" window. From it, you can drag parameters into the script, during this FastReport automatically creates a code for referring to the parameters.

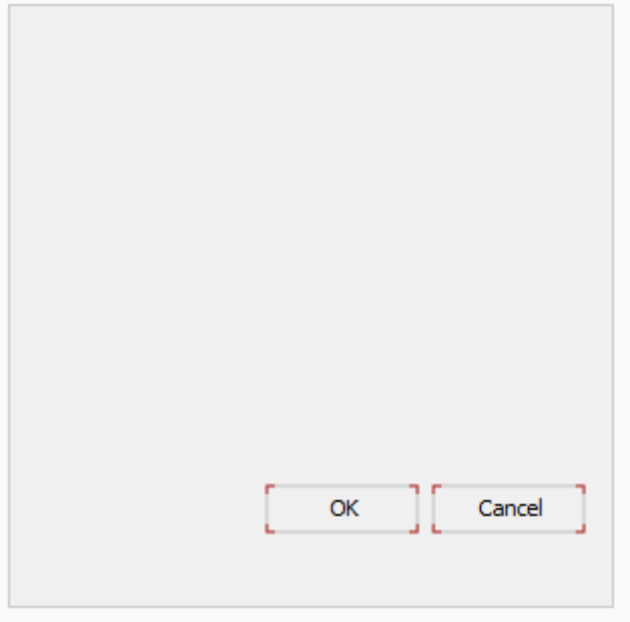
For changing the value of the parameter, use the SetParameterValue method of the report object:

```
Report.SetParameterValue("MyParameter", 10);
```

7. Dialog forms

Apart from an ordinary page, a report can contain one or several dialog forms. Such dialog form will be shown when the report is started. In the dialog, you can enter any type of information needed for creating a report. Also, a dialog may be used to filter data, which is shown in the report.

For adding a dialog into a report, press the icon on the toolbar in the tab “Report”. A new dialog looks as follows:



To delete the dialog, select the bookmark with dialog form and press the Delete page on the tab "Report".

A report, having one or several dialogs, works like this:

– when started, the report shows the first dialog;

– if the dialog is closed with the help of the "OK" button, then the next dialog is shown; otherwise, the report stops working;

– after all the dialogs have been shown, the report is built.

7.1. Controls

On a dialog form, the following controls can be used:

Icon	Name	Description
	ButtonControl	Represents a Windows button control.
	CheckBoxControl	Represents a Windows CheckBox.
	CheckedListBoxControl	Displays a ListBox in which a check box is displayed to the left of each item.
	ComboBoxControl	Represents a Windows combo box control.
	DateTimePickerControl	Represents a Windows control that allows the user to select a date and time and to display the date and time with a specified format.
	LabelControl	Represents a standard Windows label.
	ListBoxControl	Represents a Windows control to display a list of items.
	MonthCalendarControl	Represents a Windows control that enables the user to select a date using a visual monthly calendar display.
	RadioButtonControl	Enables the user to select a single option from a group of choices when paired with other RadioButton controls.
	TextBoxControl	Represents a Windows text box control.

All controls, except the DataSelectorControl, are a full analog of the standard Windows.Forms controls. The name of the element has got a Control suffix, in order to avoid duplicate names. So, the FastReport'sButtonControl corresponds to the standard Button control.

Referencing to a control can be done by using its name:

```
TextBoxControl1.Text = "my text";
```

In fact, the FastReport's control is just a wrapper for the standard .Net control. It wraps many, but not all, properties of the standard control. If you need some property that is not implemented by the FastReport control, you may access a wrapped standard control in the following way:

– using the "Control" property, which is of System.Windows.Forms.Control type:

```
(TextBox1.Control asTextBox).ShortcutsEnabled = false;
```

– using the property which has the same name as the control itself, but without the "Control" suffix. For example, the TextBoxControl has got the "TextBox" property, which is of System.Windows.Forms.TextBox type and returns the wrapped TextBox control:

```
TextBox1.TextBox.ShortcutsEnabled = false;
```

Help on properties and methods of the controls can be accessed from the MSDN.

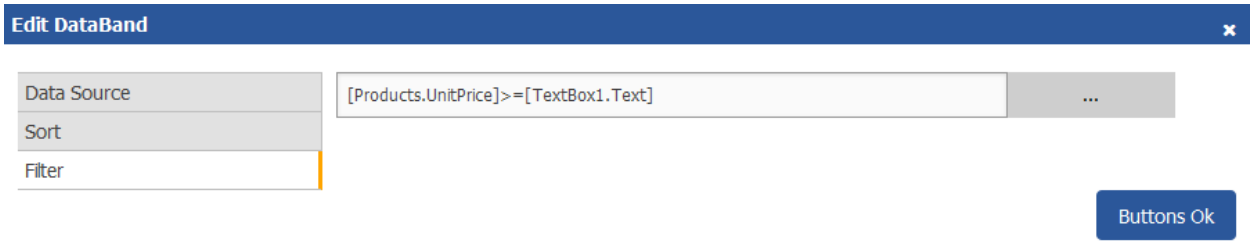
7.2. Data filtering

As previously mentioned, often dialog forms are used to filter the data in the report. For example, you can filter the list of products for the price.

In order to use the filter it is required that the original report will contain all the data from the source. Unnecessary data will be discarded during the report creation.

The easiest way to organize data filtering is to use the option "Filter" of a band "Data" in the band editor. To open the editor of a band "Data", double-click on it.

Using a dialogue a report can request value from the user and use it in a filter expression.



7.3. Automating filtering

Control is connected to the data column using the "DataColumn" property. If the control can display a list of values (for example, ListBoxControl), it fills with values from the indicated data column. This happens automatically when the dialog is shown. Further on the user works with the dialog, selects one or several items in the control and closes the dialog. At this time, the data source which was indicated in the "DataColumn" property is filtered automatically.

The advantage of this method is that you can use it in any report without writing any code.

Automatic filtering is supported by the following controls:

Icon	Name
	CheckBoxControl
	CheckedListBoxControl
	ComboBoxControl
	DateTimePickerControl
	ListBoxControl
	MonthCalendarControl
	RadioButtonControl
	TextBoxControl

7.4. Filter operations

By default FastReport filters the data rows, which contain the value equal to the value of the control. This behavior is defined in the "FilterOperation" property of the control. You can use the following operations:

Operation	Equivalent	Effect
Equal	=	Filter the value if it is equal to the control's value.
NotEqual	<>	Filter the value if it is not equal to the control's value.
LessThan	<	Filter the value if it is less than the control's value.
LessThanOrEqual	<=	Filter the value if it is less than or equal to the control's value.
GreaterThan	>	Filter the value if it is greater than the control's value.
GreaterThanOrEqual	>=	Filter the value if it is greater than or equal to the control's value.

For example, if the "FilterOperation" property of the control is set to "LessThanOrEqual" and you enter the value 5 in the control, then all the data rows will be chosen, for which the corresponding column value is less than or equal to 5.

For the data of "string" type, you can use extra operations:

Operation	Effect
Contains	Filter the value if it contains the control's value.
NotContains	Filter the value if it does not contain the control's value.
StartsWith	Filter the value if it starts with the control's value.
NotStartsWith	Filter the value if it does not start with the control's value.
EndsWith	Filter the value if it ends with the control's value.
NotEndsWith	Filter the value if it does not end with the control's value.

For example, if the "FilterOperation" property of the control is set to "StartsWith" and you enter the "A", then all data rows whose corresponding data column's value starts with "A", will be chosen.

7.5. Filtering on data range

This method of filtering is convenient for using when working with values with a quantitative characteristic, for example, the cost. You can filter goods, having the cost less than or more than the given. In order to indicate how to interpret the value, entered in the element, use the "FilterOperation" property, looked at above.

Using two controls, which are connected to the same data column and have different settings of the "FilterOperation" property, you can indicate the beginning and the ending of the data range. For the first control, you need to indicate the FilterOperation = GreaterThanOrEqual, for the second - LessThanOrEqual.

7.6. Filtering on related data column

As we know, between two data sources a relation can be set. See more details in the "Data" chapter. With the help of relation, it is possible to filter data in the source, by using a data column from a different source.

Assuming, you have placed on the dialog a ListBoxControl and indicated the following data column in the "DataColumn" property:

Products.Categories.CategoryName

How will filtering work?

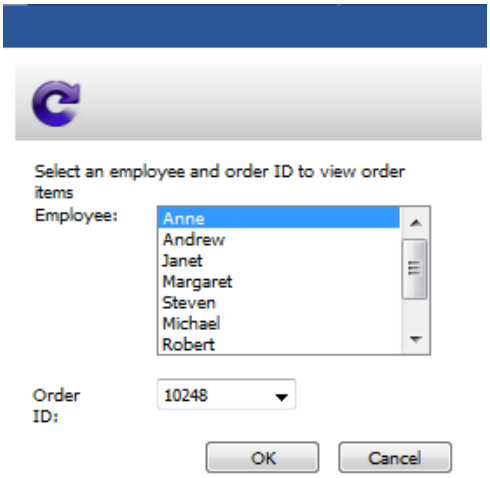
– when filling the control with values, the "CategoryName" column from the "Categories" data source will be used;

– the filter will be applied to the "Products" table. Those data rows will be filtered for which the following condition is correct:

the [Products.Categories.CategoryName] contains one of the values, selected by the user

7.7. Filtering with cascading lists

A cascading list is a list with choices that change based on the value a user selects in another list. For example, you have two lists on a form - one with categories, another one with products. When you choose a category in the first list, you will see in the second list the products in a chosen category.



To create a cascading list, you need to use two data sources with master-detail relation between them (read more about data sources and relations in the "Data" chapter). Attach the master list to a column in the master data source; attach the detail list to a column in the detail data source. Also set the master list's "DetailControl" property to the detail list.

7.8. Controlling the filtering from the code

Even if automatic filtering is enough for many cases, you have the possibility of managing it manually. For this, the following methods and properties are used.

The "AutoFill" property controls the filling of controls with data. It is used by controls, which can show a list of values, for example, the ListBoxControl. Before showing a dialog, FastReport fills such controls with data. By default, the property is set to true. If it is disabled, the control will not be filled, and you must do it yourself, by calling the "FillData" method:

```
ListBox1.FillData();
```

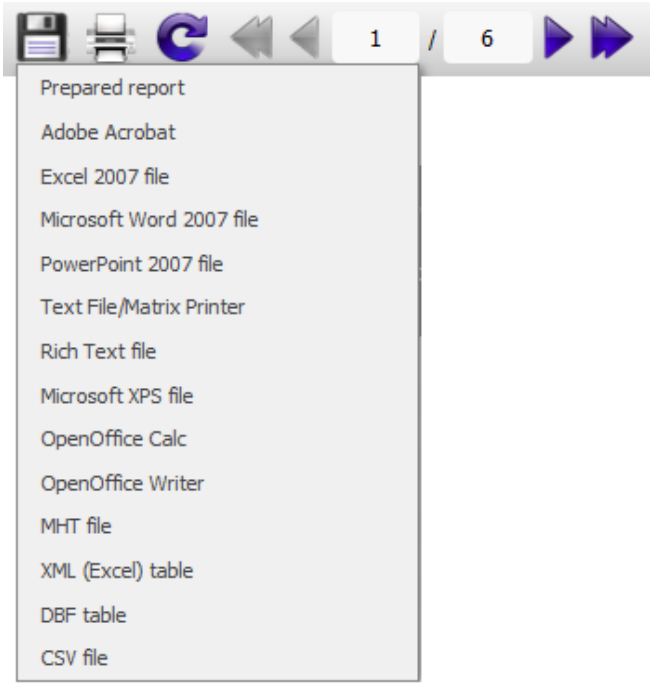
The "AutoFilter" property controls the data filtering. It is used by all controls. After the dialog has been closed by the "OK" button, FastReport applies data filter automatically. By default, the property is set to true. If it is disabled, the filtering will not happen, and you must do it yourself, by calling the "FilterData" method:

```
ListBox1.FilterData();
```

8. Export

Rendered report can be saved in one of formats: a file prepared report (fpx), Adobe Acrobat (PDF), Microsoft Word 2007, Excel 2007, PowerPoint 2007, TXT, RTF, Microsoft XPS, OpenOffice Calc, OpenOffice Writer, MHT, HTML, Excel, XML, DBF table, CSV.

To select the export format, run the report in preview mode, hover the mouse pointer over the floppy icon, and select one of the formats in the drop-down list:



After that, the standard Windows file saving will be shown.

Also, you can print a report in preview mode. To do this, click on the printer icon in the top toolbar. This opens the standard Windows printing.

8.1. Saving in FPX format

FPX format is FastReport's "native" format. The advantages of this format are as follows:

– saving the report without losing the quality. Opening an already saved file, you can do all operations with it, like print, export, editing;

– compact format based on XML, compressed with the help of ZIP;

– when needed, the report file can be unpacked by any archiver that supports the ZIP-format and corrected manually in any text editor.

The only thing lacking with the format is that, to view it, you need to have FastReport.Net.

In order to save in the FPX format, press the "Save" button in the preview window and choose the "Prepared report" file type. For opening already saved files, press the "Open" button.

8.2. Export to AdobeAcrobat (PDF)

PDF (Portable Document Format) is a platform-independent format of electronic documents created by Adobe Systems. The free Acrobat Reader package is used for viewing. This format is rather flexible – it allows the inclusion of necessary fonts, vector and bitmap images; it allows transferring and storage of documents intended for viewing and further printing.

Export method is a layered one.

8.3. Export to Excel 2007

MS Excel 2007, is an application for working with spreadsheets included in MicrosoftOffice 2007.

Export method: tabular.

8.4. Export to Microsoft Word 2007(DOCX)

A Microsoft Word 2007 - the text editor included in the package of the programs Microsoft Office 2007.

Export method: tabular.

8.5. Export to PowerPoint 2007

PowerPoint 2007 is an application for working with electronic presentations. It is included into Microsoft Office 2007.

Export method is a layered one.

8.6. Export to TXT

TXT is a regular text file that can be opened in any text editor, or printed to a dot-matrix printer.

Export method: tabular.

8.7. Export to RTF

RTF (Rich Text Format) was developed by Microsoft as a standard format for exchanging text information. At the moment RTF-documents are compatible with many new text editors and operation systems.

Export method: tabular.

8.8. Export to Microsoft XPS

XPS format is a graphics format fixed layout data based on XML. XPS file is similar to a file in PDF, but to describe the layout and metadata using XML instead of PostScript. The advantage is a smaller file size compared to PDF.

Export method is a layered one.

8.9. Export в OpenOfficeCalc

OpenDocument Format (ODF, OASIS Open Document Format for Office Application) was designed by OASIS and based on XML format used in OpenOffice.

FastReport supports export to table (.ods file). These files can be opened in OpenOffice.

Export method: tabular.

8.10. Export to OpenOfficeWriter

OpenOfficeWriter is a text processor open source from the OpenOffice Suite. Similar to Microsoft Office Word format. Has its own file format .odt.

The export method: tabular.

8.11. Export to MHT (web archive)

MHT, short for MIME HTML, is a web page archive format used to bind resources which are typically represented by external links (such as images, Flash animations, Java applets, audio files) together with HTML code into a single file. The content of an MHT file is encoded as if it were an HTML e-mail message, using the MIME type multipart/related.

Export method: tabular.

8.12. Export to Excel (XML)

Excel is an application for working with electronic worksheets. It is included into Microsoft Office.

Export method: tabular.

8.13. Export to DBF

The DBF format is designed for storing and transferring data. It is a standard way to store data in some DBMS. Also, supports a variety of tabular editors, such as Microsoft Excel. The export method: tabular.

8.14. Export to CSV

The CSV file is used for the digital storage of data structured in a table of lists form. Each line in the CSV file corresponds to a row in the table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files are often used for moving tabular data between two different computer programs, for example between a database program and a spreadsheet program.

Export method: tabular.