Basic For Qt® Description

Contents

Basic For Qt® is an open source project backed by years of continual development. 1
Feature Overview.....2
Lots of Documentation!.....2
Thousands of examples!.....2
Feature Detail.....3
Basic For Qt® is object oriented with objects and classes, single inheritance and polymorphism and private, public and global scope of objects' elements 3
Basic For Qt® is compatible to common BASICs like VB.....3
Basic For Qt® contains a managed runtime.....3
Of course like other programming languages, Basic For Qt® comes with commands for control flow, conversion/casts, error handling, events and library functions like for GUI, input or output, maths and so on. See the language reference for more. But besides this you should be familiar with the following main parts. 3
Annotations.....3
(C)Copyright KBasic Software 2012.....4

Basic For Qt® is an open source project backed by years of continual development.

It comes with VB like object orientation and backward support for VB6 and QBasic®, but it is not a VB6 or VB.NET™ clone! Though it comes with support for VB.NET™ syntax, functions and similar objects and classes. Basic For Qt® combines the expressive power of Object-Oriented languages like C++ with the familiarity and ease of use of VB6. It allows developers with an installed base of VB applications to start developing for a mixed Windows®, Mac® OS X and Linux® environment without having to face a steep learning curve: Basic For Qt® uses the familiar visual design paradigm and has a full implementation of the BASIC language.

Basic For Qt® is made up of the following programs:

- a development environment with visual form designer (IDE)
- a compiler
- an interpreter
- a graphical user interface component

It is really easy to develop multi-platform GUI applications with well known BASIC syntax in a modern fashion, because Basic For Qt® uses the familiar visual design paradigm and has a full implementation of the BASIC language. Basic For Qt® uses Qt® to provide cross-platform functionality (Qt® is the leading cross-platform technology available worldwide).

Feature Overview

On the following sections you will get know Basic For Qt®. Though I cannot describe everything in detail like in a programming book, I will describe some of Basic For Qt®'s nature and features at an expert level. Nevertheless if you know C++, VB or similar BASICs you will not have difficulties to get along with Basic For Qt®. It shares many concepts of those programming languages.

The General Manual: this manual contains detailed information about Basic For Qt®. You can find it in the menu 'Help'. So if you are interested in a line-by-line coverage please refer to The General Manual.

- Write once and deploy native applications for Windows®, Mac® OS X and Linux®
- OOP RAD features deliver high productivity, be more productive
- Much cheaper than other BASIC's
- "backward" support for VB and "forward" support for inheritance and other OOP features
- do it on multiple platforms
- Porting existing VB projects is easy, because Basic For Qt® has nearly same syntax as VB
- Familiar development process and environment
- Easy to learn: built-in Tips and language reference
- Built-in memory management via reference counting
- True cross-platform deployment
- Familiar language features: OOP, single inheritance, etc.
- Drag & drop GUI development
- Rich UI widgets set
- Familiar editing features: easy and fast browsing of your source code
- Familiar editing features II: Auto-Completion of built-in-functions and data types, even user defined functions and types
- Familiar debugging features: single step, showing variables' values, local and global scope

Lots of Documentation!

Basic For Qt® comes with extensive documentation, with hypertext cross-references throughout, so you can easily click your way to whatever interests you. The part of the documentation that you will probably use the most is the Basic For Qt® Language Reference. Each link provides a different way of navigating the Basic For Qt® Language Reference; try them all to see which work best for you. You might also like to try the other manuals containing detailed information about Basic For Qt®, and it provides a full text search facility. There are also a growing number of Basic For Qt® books.

Thousands of examples!

Basic For Qt® ships with lots of small and some medium-sized example programs that teach you how to implement various tasks with Basic For Qt®. Most of them will show how to use a certain class or module, others aim at programming techniques and Basic For Qt® basics, and some of them simply want to show you what is possible.

Note that most of the examples assume that you have some experience with Basic For Qt® and Object-Oriented programming and therefore are not commented extensively. If you are interested in a line-by-line coverage please refer to the "Learning Coding" and "The General Manual".

Feature Detail

Basic For Qt® is object oriented with objects and classes, single inheritance and polymorphism and private, public and global scope of objects' elements

Basic For Qt® is compatible to common BASICs like VB

- named arguments [mySub(param1 := 23, param2 := 100)]
- label and goto
- property handling
- arrays and user defined types can be passed to functions by reference

Basic For Qt® contains a managed runtime

- automatic garbage collection
- protected data and arrays
- modern error management through exception handling

Of course like other programming languages, Basic For Qt® comes with commands for control flow, conversion/casts, error handling, events and library functions like for GUI, input or output, maths and so on. See the language reference for more. But besides this you should be familiar with the following main parts.

- Class / Module
- Sub / Function / Method
- Variable / Constant / Property
- Array
- Type
- Enum

If you start working on your first Basic For Qt® program keep in mind that is very similar to VB. You have modules or classes and forms, which work together. Events in your forms are triggered by the user and you can react to them within your program inside its subs. That's it.

Annotations

- Type
- Variable / Constant / Property
- Variable Scope
    - global
    - module
    - class
    - local
        - module function or module sub
        - class instance method
- Array
    - static
    - dynamic
- Sub / Method
    - recursive calls are possible
    - local variables (objects) will not be automatically destroyed if there is a reference to

it

- Function / Method
  - arguments by reference or by value depending on primitive type are possible
  - function overloading possible
  - recursive calls are possible
  - user defined types and arrays can be returned as well
  - local variables (objects) will not be automatically destroyed if there is a reference to it

- Scope
  global
  - module
  - class instance
  - local
    module
    - class instance
- Class
  - Init Constructor (method overloading possible)
  - Default Constructor (automatically called if there is no constructor defined)
  - Finalize Destructor (automatically called by garbage collector)
  - instance methods (works on variables of an object of a class)
  - singleton pattern provided for each class automatically
  - instance variables (private, public), variables of an object of a class
  - global variables (works without any relation to an object)
  - global constants
  - property support
  - variables (private, public)
  - constants (private, public)
  - types / enum can be private or public

- Module
  - subs or functions
  - variables (private, public)
  - constants (private, public)
  - types / enum can be private or public
- Global
  - subs or functions
  - variables
  - constants
  - types / enum