The Qt Company

QT QUICK UI 023-001

Exam Curriculum

The Qt Company provides Qt and QML developers with three kinds of certification exams:

- *Qt and QML Essentials*
- *Widget UI and Application Engine with Qt*
- *Qt Quick UI*

Exam candidates may receive Certified Qt and QML Developer, Certified Qt C++ Specialist, and Certified Qt Quick Specialist certificates by passing the exams. A certificate is a valuable document, endorsing one's Qt and QML knowledge to an employer or a customer.

To achieve the Qt and QML Developer status, an exam candidate is required to pass the Qt and QML Essentials exam. The Qt C++ or Qt Quick Specialist status is granted to candidates, who additionally pass either or both Widget UI and Application Engine and Qt Quick UI exams. The former specialist exam tests candidates' knowledge of Qt C++ APIs, including, e.g., widgets, threads, model/view framework, and QObject. The latter exam tests Qt Quick and QML knowledge.

The exams can be taken in any order, but the candidate cannot receive either of the specialist certificates before the Qt and QML Essentials exam has been passed as well. So the Qt and QML Developer certificate is required for both specialist certificates as well.

Certificate exams can be taken in any authorized PearsonVUE test center. The details of the test center locations and instructions how to make an appointment and attend the exam can be found at http://www.pearsonvue.com/qtcompany/. The exam price varies from test center to test center. The exact price can be inquired directly from test centers.

Qt Quick UI exam will test candidates' knowledge of UI creation using Qt Quick and QML. The exam curriculum is defined in detail in this document. The exam contains 30 multi choice questions and the candidate has 60 minutes to select the correct statement(s). No coding is required in the exam, although the questions and question options may contain short code snippets.

1 QML BASICS

Exam attendees must know essential QML element types. They have to know how to create custom items with custom signals, JavaScript functions, and properties. The use of anchors and positioners is required in the exam. Attendees must be able to add UI interactions to QML programs as well as animations based on either animation element types or states and transitions.

1.1 QML USER INTERFACES

- *Element types*
- *Visual elements*
- *Layouts*
- *User interactions*

1.2 ANIMATIONS

- Animation elements
- Animation groups
- Animation thread
- Animation types
- Easing curves

1.3 STATES AND TRANSITIONS

- State concept
- State machine, managing state transitions
- PropertyChanges
- Transition concept
- Transition-based animations
- Reversible transitions

2 QT QUICK CONTROLS

- Application window
- Views
- Layouts
- Controls
- Styling
- Qt Quick Extra Controls

3 DATA PRESENTATION

3.1 MODELS AND CONTAINERS

- JavaScript string and object arrays
- QML models
- Roles and dynamic roles
- Table and hierarchical models
- Use of C++ models (QAbstractItem-based)
- Use of C++ containers

3.2 VIEWS

- QML views, including Qt Quick Control views
- View Decoration
- Delegates
- Delegate creation
- Delegate caching
- Delegate parenting

4 DYNAMIC QML

- Approaches to dynamically allocate objects in QML
- Object ownership and parentship
- Object deletion
- Asynchronous loading
- Asynchronous object creation

5 QML MULTIMEDIA

- Media player
- Camera
- VideoOutput
- Audio
- Playback and recording

6 MOBILE QML

- Sensors
- Bluetooth clients
- Positioning and location APIs
- In-app purchasing

7 QML AND SCRIPTING

- JavaScript usage
- Global object and its limitations
- Use of *this* keyword
- HTTP requests
- Local storage usage
- Worker scripts

8 CANVAS

- 2D and 3D Canvas usage (no OpenGL or three.js APIs required)

9 GRAPHICAL EFFECTS

- Principles of using ready-made effects (no need to remember properties of *FastBlur*, for example, but need to understand the principle of using *FastBlur*)

9.1 PARTICLE SYSTEM

- Particle system: need to understand the system, but no need to be able to write own particle systems
- Emitter basic properties: dimensions, emit rate
- Particle types
- *Affector* basics

9.2 SHADERS

- Understand how shaders work in QML
- No need to manage GL shader language

10 QML AND C++

10.1 QML ENGINE

- Engine functionality at the high level; loading, compiling, creation, property bindings
- Context and sub-contexts
- Exposing properties and objects
- Accessing QML objects from C++ and vice versa

10.2 CREATING NEW TYPES

- *Using QObject to define new types*
- *QQuickItem and QQuickPaintedItem*
- *Scene graph nodes*
- *Geometry and material (no shaders)*
- *Register functions*

11 SCENE GRAPH RENDERING

- *Rendering principles*
- *Rendering phases*
- *Scene graph rendering optimizations*
- *Rendering customization*
- *2D renderer*