QML Guidelines:

- Keep binding expressions as simple as possible. Avoid updating bound property when it's not necessary (triggering re-evaluation of binding expressions). See Property bindings for more details.
- Prefer using concrete types instead of var. See Using var vs concrete types for details.
- Avoid using massive JavaScript functions in qml. See JavaScript code fro details.
- Prefer using simple Row, Column types instead if RowLayout, ColumLayout. If usage of Layout type is unavoidable read Layouts and Qt Quick Layouts overview
- Use Loader for lazy instantiation. See Loader for details.
- Almost all qml views should have data models. See Data Models, QAbstractItemModel subclass and qml ListModel
- Don't use Qt.createComponent() and Qt.createQmlObject(). Everything that is described here: Dynamic object creation should be avoided.
- Keep QML object attributes in the same order: https://doc.qt.io/qt-5.11/qml-codingconventions.html#qml-object-declarations
- Qt qml documentation to read:
- Qt Quick Best Practices
- Performance Considerations And Suggestions
- Integration QML and C++
- Overview - QML and C++ Integration
- Exposing Attributes of C++ Types to QML
- Writing QML Extensions with C++
- Data Type Conversion Between QML and C++
- Open question 1: qml Instantiator vs Repeater vs Loader ???
- Instantiator is a combination of Repeater and Loader. It's useful when you have a model that you use to create many objects dynamically. Also supports asynchronous loading.