**QT Quick Elements**

QT Quick provides a variety of predefined visual elements that can be used to create user interfaces in Qt applications. These elements are designed to be easily customizable and can be styled using QML. Some of the common Qt Quick elements include:

1. Rectangle: A simple rectangular element used for creating backgrounds or drawing shapes.
2. Text: Used for displaying static or dynamically changing text.
3. Image: Used for displaying images in various formats, including JPEG, PNG, and SVG.
4. Button: A clickable element that can be used to trigger specific actions.
5. TextInput: Allows users to input text into the application.
6. ListView: Displays data in a vertically or horizontally scrollable list.
7. GridView: Displays data in a grid layout with customizable rows and columns.
8. Loader: Dynamically loads QML components or files.
9. Repeater: Instantiates a component multiple times based on a given model.
10. ScrollView: Provides a scrollable view for displaying content that doesn't fit within the available space.
11. Image: A control for displaying images in various formats.
12. TextArea: Allows multi-line text input.
13. CheckBox: A control that allows users to select or deselect an option.
14. ComboBox: A drop-down list that allows users to select one option from a list of predefined options.
15. RadioButton: A selection control that allows users to choose only one option from a set of mutually exclusive options.
16. Slider: A control for selecting a value from a range by sliding a handle.
17. ProgressBar: Visualizes the progression of a task.
18. Menu: Represents a menu that can be displayed on a certain action.
19. Dialog: Represents a pop-up dialog that can be used for displaying information or prompting users for input.

These are some of the commonly used elements in Qt Quick for creating user interfaces. Each element comes with its own set of properties and signals that can be customized and utilized to create rich and interactive applications in Qt.

Here are some common QT Quick elements along with their code examples:

**Rectangle:**

import QtQuick 2.15

Rectangle {

width: 200

height: 200

color: "lightblue"

}

**Text:**

import QtQuick 2.15

Text {

text: "Hello, World!"

font.pointSize: 24

anchors.centerIn: parent

}

**Button:**

import QtQuick 2.15

import QtQuick.Controls 2.15

Button {

text: "Click Me"

onClicked: {

console.log("Button clicked!")

}

}

**TextInput:**

import QtQuick 2.15

import QtQuick.Controls 2.15

TextInput {

width: 200

placeholderText: "Enter text here"

}

ListView:

import QtQuick 2.15

import QtQuick.Controls 2.15

ListView {

width: 200

height: 200

model: ["Item 1", "Item 2", "Item 3"]

delegate: Text {

text: modelData

}

}

**CheckBox:**

import QtQuick 2.15

import QtQuick.Controls 2.15

CheckBox {

text: "Check me"

}

**ComboBox:**

import QtQuick 2.15

import QtQuick.Controls 2.15

ComboBox {

model: ["Option 1", "Option 2", "Option 3"]

}

**Slider:**

import QtQuick 2.15

import QtQuick.Controls 2.15

Slider {

width: 200

from: 0

to: 100

}

**ProgressBar:**

import QtQuick 2.15

import QtQuick.Controls 2.15

ProgressBar {

width: 200

value: 0.6

}

These code examples demonstrate how to use some of the common QT Quick elements in QML to create simple user interfaces. You can further customize these elements by adjusting their properties, styling, and behavior according to the requirements of your application.