1. QML objects vs Qobjects:

QML nesne türleri somutlaştırılabilen nesne türüdür. Qobjects ile arasındaki en belirgin fark budur.

Kod:

```
import QtQuick 2.12
import QtQuick.Window 2.12

Window {
    width: 640
    height: 480
    visible: true
    title: qsTr("Hello World")

    Image {
        id: name

        source:"http://upload.wikimedia.org/wikipedia/commons/0/0b/Qt_logo_
        2016.svg"
        width: 150
        height: 100
        opacity: 0.25
        anchors.centerIn: parent
    }

    //Writes the written text on the image(qt symbol). Real time changing
on the texts.
    TextInput{
        id:myInput
        text: "Hello World!!"
        anchors.centerIn: parent
        font.pixelSize: 25

    }
    //Types the binding text as same as with the myInput corner of the
screen! Real time chanhcing on the texts.
    Text {
        id: myText
        text: myInput.text //Property Binding
        font.pixelSize: 25
    }

}
```

Output:

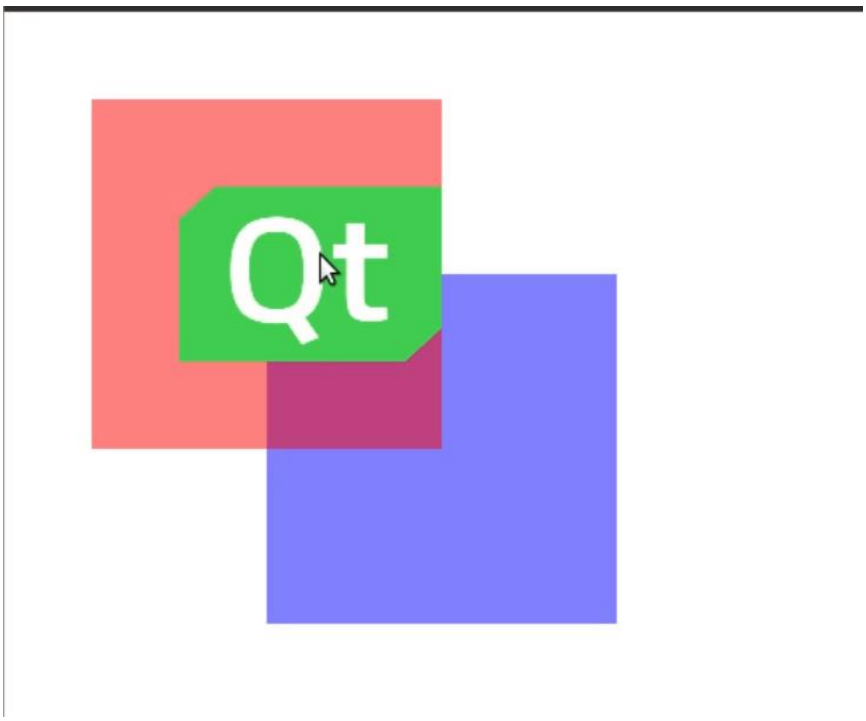
2. X, Y, Z positioning:

QML' deki X, Y, Z koordinatlarını tanımak üzere yapılan uygulama parents yapısını da rahatlıkla kodda görebilirsiniz.

Kod:

```
import QtQuick 2.12
import QtQuick.Window 2.12

// X Y Z
Window {
    width: 640
    height: 480
    visible: true
    title: qsTr("Hello World")

    Image { // If it becomes last thing aded it figures out on the screen!!
z-axis is not needed to be added.
        id: image

        source:"http://upload.wikimedia.org/wikipedia/commons/0/0b/Qt_logo_
        2016.svg"
        width: 150
        height: 100
        x: 100
        y: 100
        z: 4 //top the logo
    }
    Rectangle{
        color: "red"
        width: 200
        height: 200
        x: 50
        y: 50
        opacity: 0.5
        z: 2
    }

    Rectangle{
        color: "blue"
        width: 200
        height: 200
        x: 150
        y: 150
        opacity: 0.5
        z: 1
    }

}
```

Output:



3. Parents and Child Transformations:

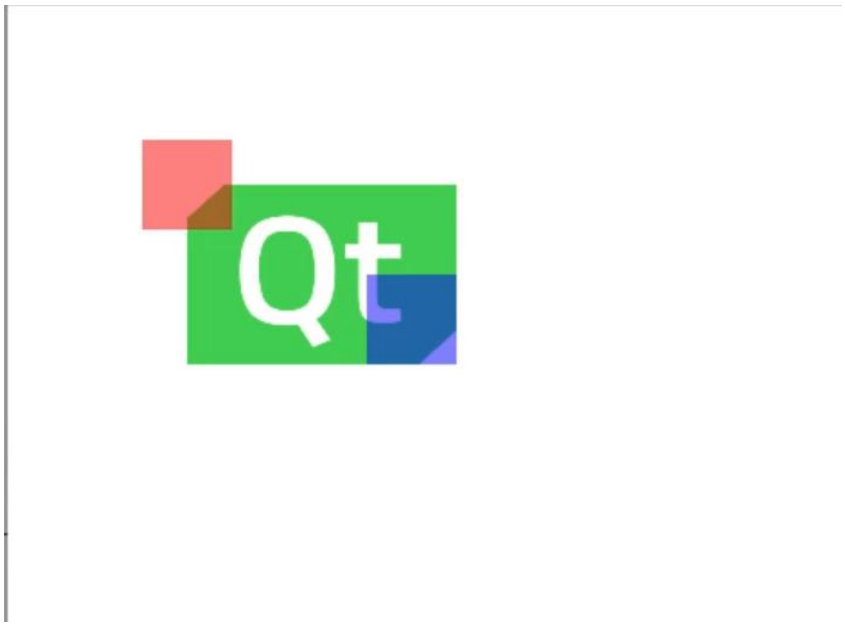
Parents ve child yapılarını tanımak için yapılmış bir uygulamadır.

Parents and Child -> Dependent structures.

Kod:

```
import QtQuick 2.12
import QtQuick.Window 2.12

// Parent and child trasformations.
Window {
    width: 640
    height: 480
    visible: true
    title: qsTr("Hello World")

    Image {
        id: image

        source:"http://upload.wikimedia.org/wikipedia/commons/0/0b/Qt_logo_
        2016.svg"
        width: 150
        height: 100
        x: 100
        y: 100
        z: 0

        Rectangle{
            color: "red"
            x: 0
            y: 0
            width: 50
            height: 50
            opacity: 0.5
            z: 3

        Rectangle{
            color: "blue"
            x: parent.width - width
            y: parent.height - height
            width: 50
            height: 50
            opacity: 0.5
            z: 4
        }
    }
}
}
```

Output:



4. Object Interaction – TopHandler:

Mouse inputunu aldığında renk değiştiren 2 bağımsız kare uygulaması yapılmıştır. 2. ve 3. bölümlerdeki QML konuları da kullanılmıştır.

Kod:

```
import QtQuick 2.12
import QtQuick.Window 2.12

Window {
    width: 640
    height: 480
    visible: true
    title: qsTr("Hello World")

    Rectangle{

        width: 100
        height: 100

        x: 100
        y: 100

        color: inputHandle.pressed ? "red" : "blue"
        // if mouse is pressed, it is red color otherwise it is blue.

    Rectangle{

        width: 100
        height: 100

        x: 150
        y: 25

        color: inputHandler2.pressed ? "red" : "blue"

        TapHandler{
            id: inputHandler2
        }

        }


        TapHandler{
            id: inputHandle
        }




    }

}
```

Output: