

Applying Architecture Patterns in Qt / QML

Leonnardo Verol

Thanks to



Leonnardo Verol

Computer & Software Engineer

System Specialist
Intel Competence Center

Intel

github.com/LeonnardoVerol



linkedin.com/in/leonnardoverol



Today - QML

- ❖ Components (HTML, React & Others)
- ❖ Modules
- ❖ Flux Pattern (Facebook, React, VueJS)
- ❖ Navigation (VueJS, React)
- ❖ Responsive Breakpoints (Web)
- ❖ Demonstration

Case Studies

Case Studies

~/example-authentication-qt-qml

~/example-components-qt-qml



Example Authentication in QT / QML

Log In

Username

Password

Log In

Register

Example Authentication in QT / QML

Register

Name

Username

Password

Register

Cancel

Component Example

<main>

<article>

<section>

<section>

<section>

<aside>

Components

(Introduction & Definition)

Components - Definition

- Code snippets that will be reused
- Encapsulate a specific logic

Essentially, they are “interface functions”

Components - The Cost of Bad Code

“Have you ever been significantly impeded by bad code?”

(...) We wade through bad code. We slog through a morass of tangled brambles and hidden pitfalls. We struggle to find our way, hoping for some hint, some clue, of what is going on; but all we see is more and more senseless code.”

- Clean Code (Page 3)

Components - The Cost of Bad Code

- Maintainability
- Productivity
- Cost

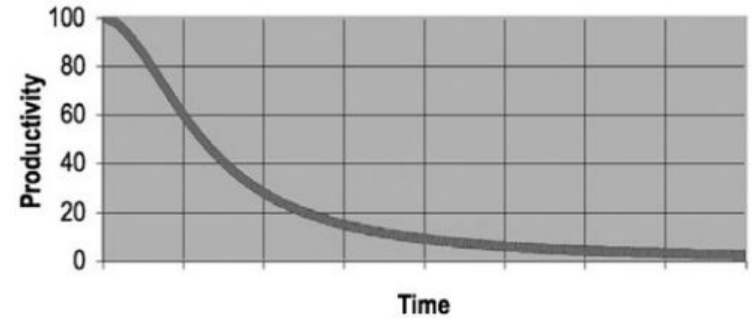


Figure 1-1
Productivity vs. time

Components - The Boy Scout Rule

“Leave the campground cleaner than you found it”

- Boy Scouts of America

Components - The Boy Scout Rule

“If we all checked-in our code a little cleaner than when we checked it out, the code simply could not rot.”

“It’s not enough to write the code well. *The code has to be kept clean over time.*”

Components

("Bad Code")

Components - “Bad Code”

```
5879 }  
5880  
5881 }  
5882  
5883 #endif  
5884
```

```
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451
```

Components - “Bad Code”

```
Rectangle {  
    width: parent.width  
    height: 80  
    Rectangle {  
        width: 50  
        height: 50  
        anchors.leftMargin: 20  
        anchors.rightMargin: 20  
        anchors.left: parent.left  
        anchors.verticalCenter: parent.verticalCenter  
    }  
    Rectangle{  
        anchors.left: logo.right  
        anchors.right: parent.right  
        height: parent.height  
        RowLayout {  
            spacing: 20  
            height: parent.height  
            anchors.right: parent.right  
            Layout.alignment: Qt.AlignVCenter  
            Layout.minimumWidth: 100  
  
            Rectangle {
```

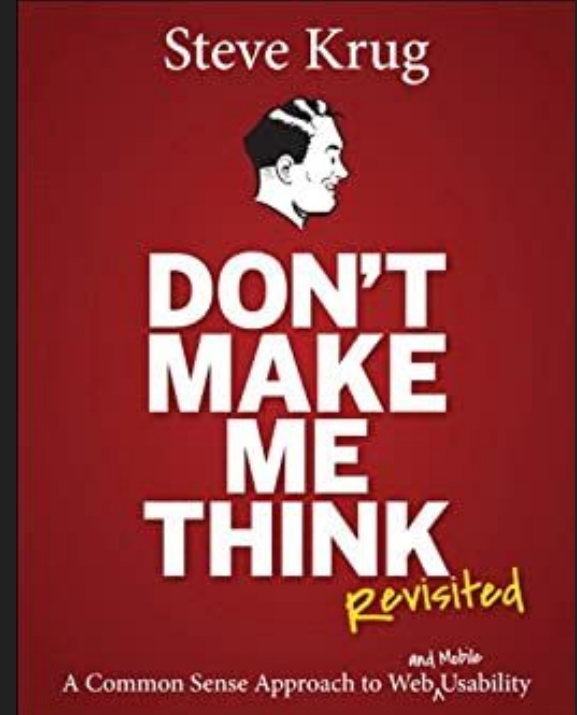
Components

(Local Components)

Components - Don't make me think

"Things should be self-explanatory"

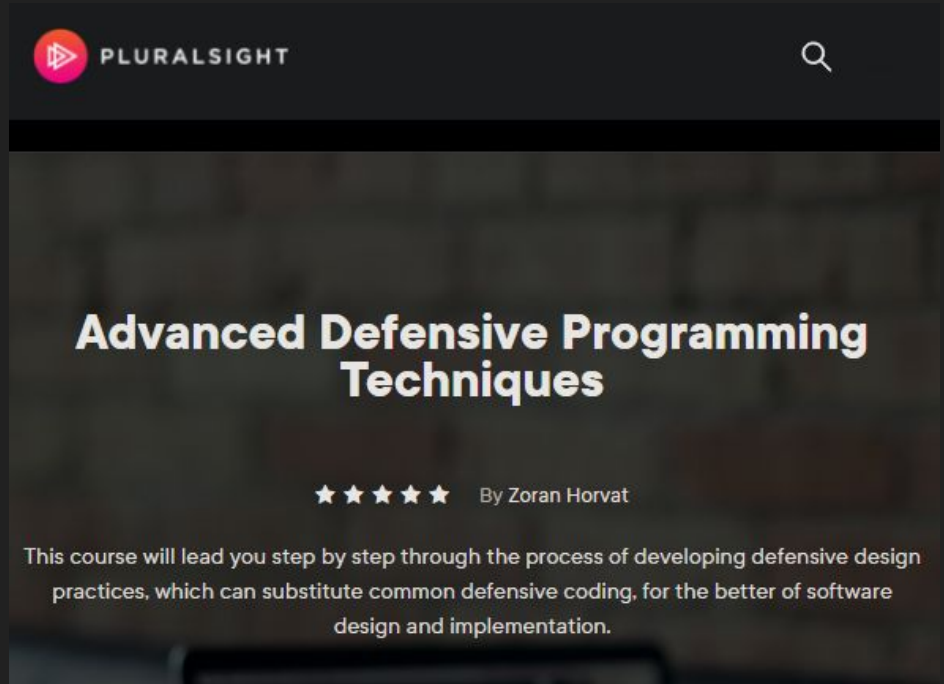
"Don't make me think (and neither do others)"



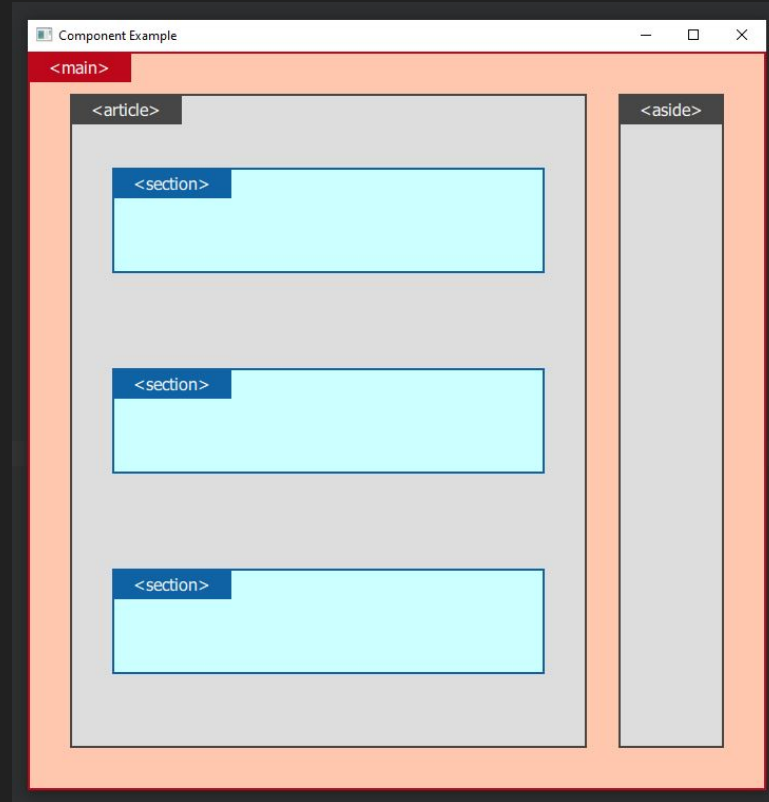
Components - Defensive Programming

“Avoid Primitive Types”

“Primitive types don’t convey any meaningful domain knowledge”



Components - Semantic HTML

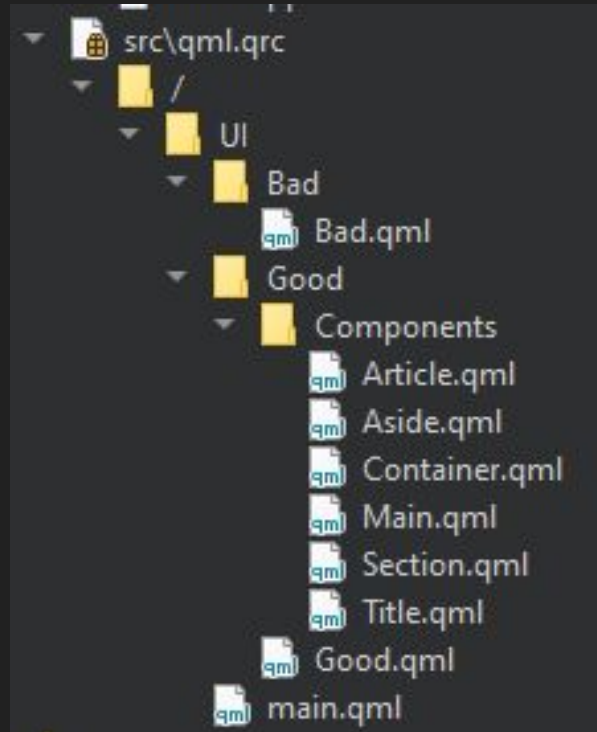


Components - Semantic HTML

```
194         color: "#DDDDDD"
195
196         border.width: 2
197         border.color: "#464646"
198
199         anchors.left: article.right
200         anchors.leftMargin: 30
201
202         y: article.y
203
204         height: 450
205         width: name3.implicitWidth
206
207     Rectangle {
208         color: "#464646"
209         anchors.left: parent.left
210         anchors.top: parent.top
211
212         height: name3.implicitHeight
213         width: name3.implicitWidth
214
215     Text {
216         id: name3
217         text: qsTr("<aside>")
218         color: "white"
219         anchors.fill: parent
220         anchors.centerIn: parent
221         topPadding: 5
222         bottomPadding: 5
223         leftPadding: 20
224         rightPadding: 20
225         font.pointSize: 12
226     }
227 }
228 }
229 }
230 }
231
232
```

```
25
26 Article {
27     id: article
28     Layout.fillHeight: true
29     Layout.fillWidth: true
30     clip: true
31
32     ColumnLayout {
33         anchors.fill: parent
34         anchors.margins: 40
35
36         spacing: 30
37
38         Section {
39             height: 100
40             Layout.fillWidth: true
41         }
42
43         Section {
44             height: 100
45             Layout.fillWidth: true
46         }
47
48         Section {
49             height: 100
50             Layout.fillWidth: true
51         }
52     }
53 }
54
55 Aside {
56     id: aside
57     Layout.preferredWidth: 100
58     Layout.fillHeight: true
59 }
60 }
61 }
62
```

Components - Local Components

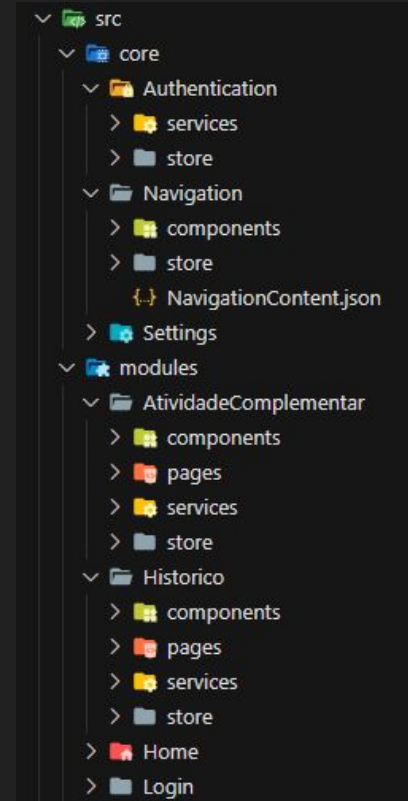
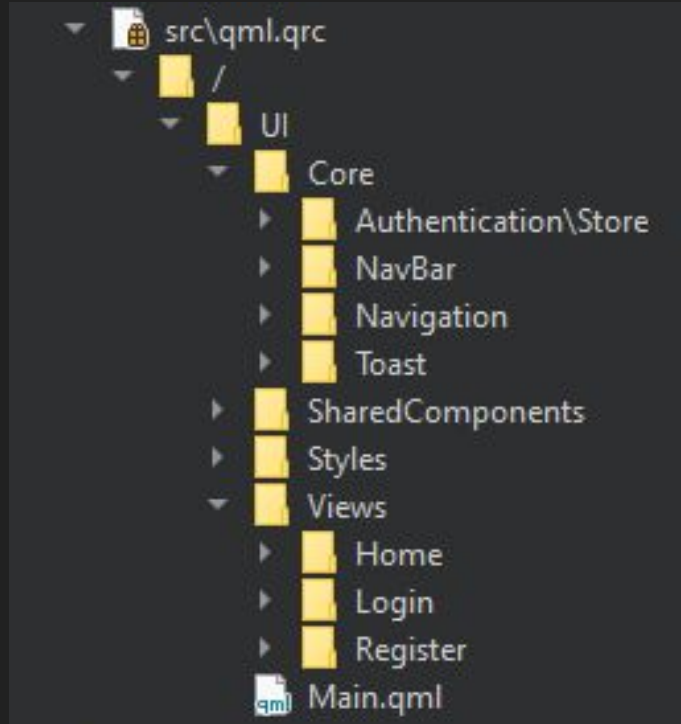


Modules

Modules

Create modules and organize them according to your domain.

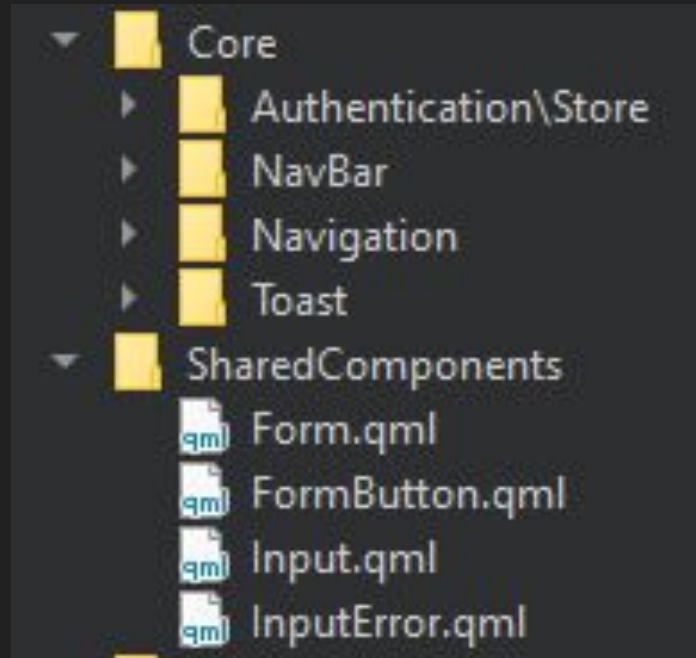
There are several approaches, don't get stuck with one.



Components

(Global Components)

Components - Global Components



Components

(Styled Components - Work in Progress Ideia)

Components - Styled Components (WIP Ideia)

```
return (  
  <Container>  
    <Content>  
      <form onSubmit={handleSubmit}>  
        <h1>Log In</h1>  
  
        0 references  
        <input type="text" placeholder="Usuário" value={username} onChange={(event) => setUsername(event.target.value)} />  
  
        0 references  
        <input type="password" placeholder="Senha" value={password} onChange={(event) => setPassword(event.target.value)} />  
  
        <button type="submit">Entrar</button>  
      </form>  
    </Content>  
  </Container>  
)  
}
```

Components - Styled Components (WIP Ideia)

```
export const Container = styled.div`  
  height: 100vh;  
  
  display: flex;  
  align-items: center;  
  justify-content: center;  
`;  
;
```

```
export const Content = styled.div`  
  display: flex;  
  justify-content: center;  
  flex-direction: column;  
  align-items: center;  
  
  width: 100%;  
  max-width: 700px;  
  
  form {  
    margin: 80px 0;  
    width: 340px;  
    text-align: center;  
  
    h1 {  
      margin-bottom: 24px;  
    }  
  
    input {  
      background: #232129;  
      border-radius: 10px;  
      border: 2px solid #232139;  
      padding: 16px;  
      width: 100%;  
      color: #F4EDEC;  
  
      & + input {  
        margin-top: 8px;  
      }  
    }  
  
    button {  
      background: #FF9000;  
      color: #312E38;  
      border-radius: 10px;  
      border: 0;  
      padding: 16px;  
      width: 100%;  
  
      font-weight: 500;  
      margin-top: 16px;  
    }  
  }  
`;  
;
```

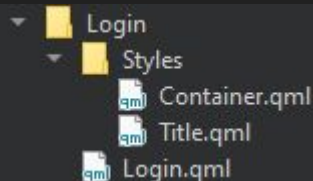
Components - Styled Components (WIP Ideia)

```
Container {  
  Title { text: qsTr("Log In") }  
  
  Form {  
    Input {  
      id: username  
      label.text: qsTr("Username")  
      placeholderText: qsTr("Username")  
    }  
  
    Input {  
      id: password  
      label.text: qsTr("Password")  
      placeholderText: qsTr("Password")  
      echoMode: TextInput.Password  
    }  
  
    FormButton {  
      id: submit  
      text: qsTr("Log In")  
      backgroundColor: Colors.button.primary.background  
      backgroundHoverColor: Colors.button.primary.hovered  
      Layout.fillWidth: true  
  
      onClicked: { ... }  
    }  
  
    FormButton {  
      text: qsTr("Register")  
      textColor: Colors.button.secondary.text  
      backgroundHoverBorderColor: Colors.button.secondary.hoveredBorder  
      Layout.fillWidth: true  
  
      onClicked: Navigation.push(ScreenTypes.REGISTER_SCREEN)  
    }  
  }  
}
```

Components - Styled Components (WIP Ideia)

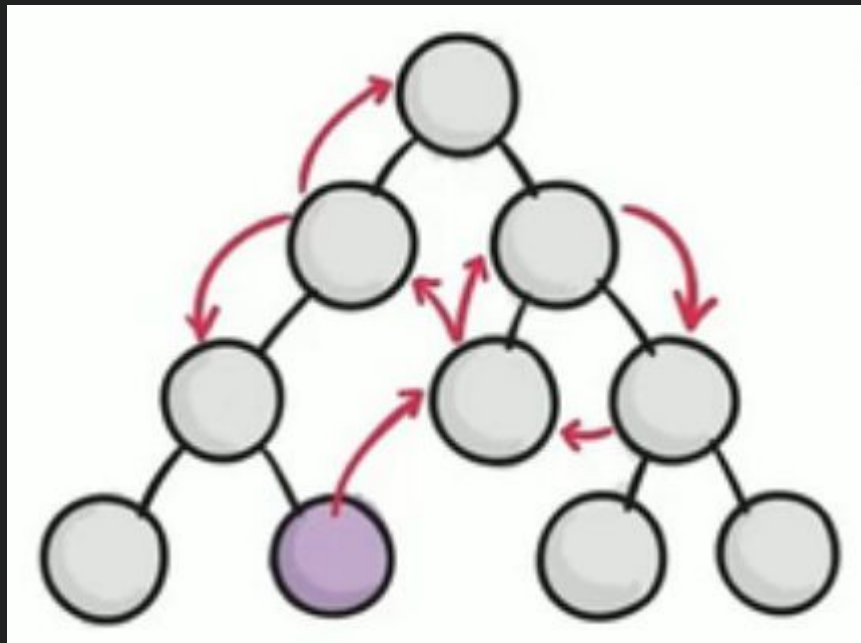
```
Container {  
    Title { text: qsTr("Log In") }  
}
```

```
ColumnLayout {  
    anchors.centerIn: parent  
    width: 300  
    spacing: 50  
}
```



```
Text {  
    Layout.alignment: Qt.AlignHCenter  
    font.pointSize: 36  
    color: "#CBCBCB"  
}
```

Components - Communication between components



source: <https://www.bilibili.com/s/video/BV1rE411A7uk>

Flux Pattern

(Introduction & Definition)

Flux Pattern

“Flux is the application architecture that Facebook uses for building client-side web applications. It complements React's composable view components by utilizing a unidirectional data flow. It's more of a pattern rather than a formal framework, and you can start using Flux immediately without a lot of new code.”

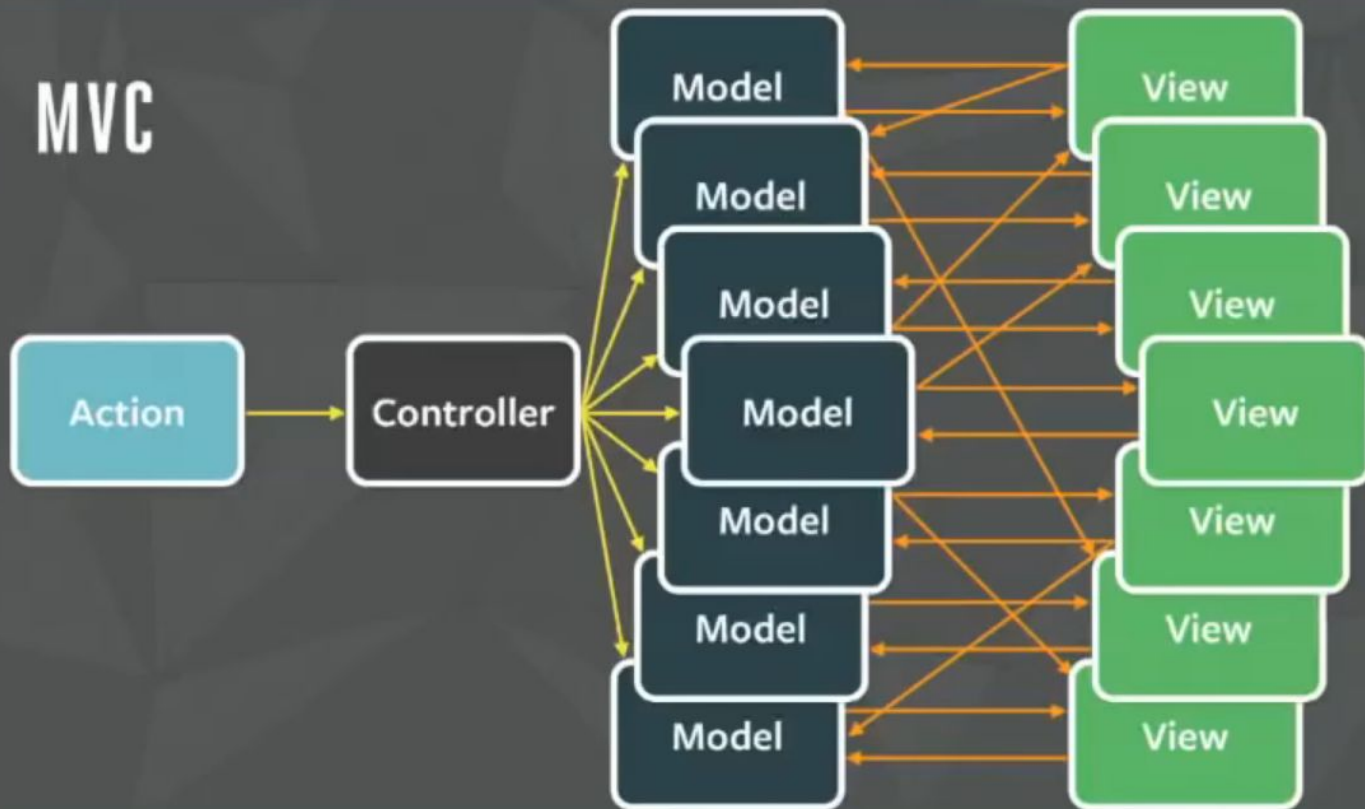
Source

<https://facebook.github.io/flux/docs/in-depth-overview>

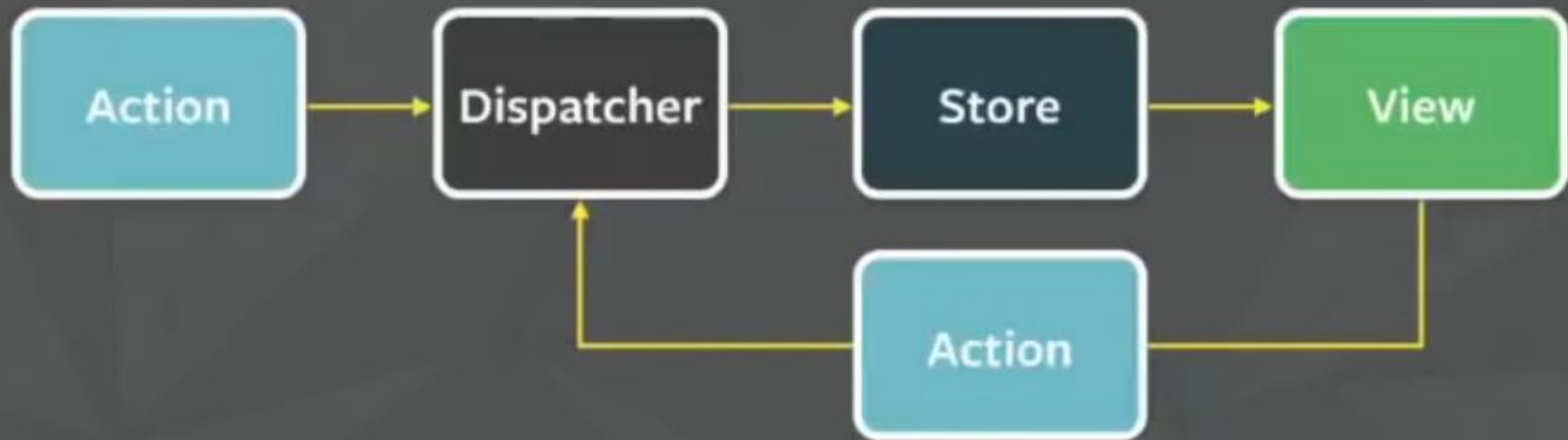
Flux Pattern

MVC DOESN'T SCALE

Flux Pattern



Flux Pattern

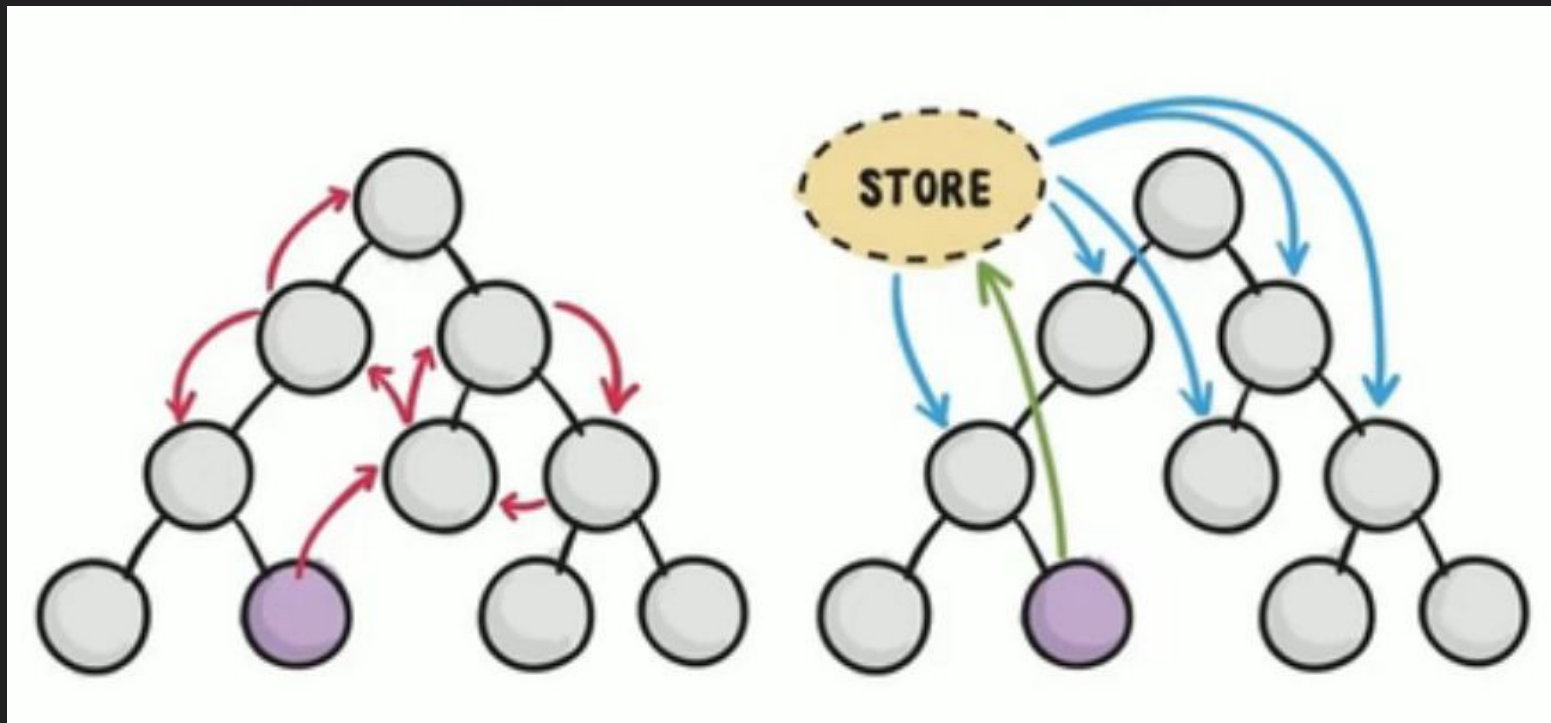


Flux Pattern

INCREASE PREDICTABILITY

- Improved data consistency
- Easier to pinpoint root of a bug
- More meaningful unit tests

Flux Pattern



source: <https://www.bilibili.com/s/video/BV1rE411A7uk>

Flux Pattern

(Authentication Store)

Flux Pattern - Authentication Store

```
1  import QtQuick 2.15
2  import "Types.js" as Types
3
4  pragma Singleton
5  Item {
6      property var fakeDatabase: [
7          {name:"admin",username:"admin",password:"admin"}
8      ]
9      property string status: ""
10     property string errorMessage: ""
11
12     property QObject user: QObject {
13         property string name
14     }
15
16
17     property var commit: (function(state, payload = undefined) { ... })
64
65
66     function register(payload)
67     { ... }
80
81     function login(payload)
82     { ... }
101
102     function logoff()
103     { ... }
106 }
107
```


Flux Pattern - Authentication Store

```
property var commit: (function(state, payload = undefined) {  
    const mutations = { [...]}  
  
    mutations[state](payload);  
})
```

Types.js

```
const REGISTER_REQUEST = "REGISTER_REQUEST"  
const REGISTER_ERROR = "REGISTER_ERROR"  
const REGISTER_SUCCESS = "REGISTER_SUCCESS"  
  
const AUTH_REQUEST = "AUTH_REQUEST"  
const AUTH_ERROR = "AUTH_ERROR"  
const AUTH_SUCCESS = "AUTH_SUCCESS"  
const AUTH_LOGOFF = "AUTH_LOGOFF"
```

```
[Types.REGISTER_REQUEST]: function (payload)  
{  
    status = "Fake API Request";  
    errorMessage = "";  
},  
[Types.REGISTER_ERROR]: function (payload)  
{  
    status = "Register Failed"  
    errorMessage = payload  
},  
[Types.REGISTER_SUCCESS]: function (payload)  
{  
    status = "Register Successful";  
    errorMessage = "";  
  
    fakeDatabase.push(payload)  
    console.log(JSON.stringify(fakeDatabase))  
},  
[Types.AUTH_REQUEST]: function (payload)  
{  
    status = "Fake API Request";  
    errorMessage = "";  
},  
[Types.AUTH_ERROR]: function (payload)  
{  
    status = "Login Failed";  
    errorMessage = payload;  
},  
[Types.AUTH_SUCCESS]: function (payload)  
{  
    status = "Login Successful";  
    errorMessage = "";  
  
    user.name = payload.name  
},  
[Types.AUTH_LOGOFF]: function (payload)  
{  
    status = "Log Off Successful";  
    errorMessage = "";  
  
    user.name = ""  
},
```

Flux Pattern - Authentication Store

```
function register(payload)
{
  commit(Types.REGISTER_REQUEST);

  if(fakeDatabase.find(user => user.username === payload.username))
  {
    commit(Types.REGISTER_ERROR, "Username Already Exists!");
    throw { status, errorMessage };
  }
  else
  {
    commit(Types.REGISTER_SUCCESS, payload);
  }
}
```

```
[Types.REGISTER_REQUEST]: function (payload)
{
  status = "Fake API Request";
  errorMessage = "";
},
[Types.REGISTER_ERROR]: function (payload)
{
  status = "Register Failed"
  errorMessage = payload
},
[Types.REGISTER_SUCCESS]: function (payload)
{
  status = "Register Successful";
  errorMessage = "";

  fakeDatabase.push(payload)
},
```

Flux Pattern - Authentication Store

```
function login(payload)
{
  commit(Types.AUTH_REQUEST)

  const userDB = fakeDatabase.find(user => user.username === payload.username);

  if( userDB === undefined)
  {
    commit(Types.AUTH_ERROR, "User Not Found!");
    throw { status, errorMessage };
  }

  if( userDB.password !== payload.password)
  {
    commit(Types.AUTH_ERROR, "Invalid Password!");
    throw { status, errorMessage };
  }

  commit(Types.AUTH_SUCCESS, userDB);
}

function logout()
{
  commit(Types.AUTH_LOGOFF)
}
```

```
[Types.AUTH_REQUEST]: function (payload)
{
  status = "Fake API Request";
  errorMessage = "";
},
[Types.AUTH_ERROR]: function (payload)
{
  status = "Login Failed";
  errorMessage = payload;
},
[Types.AUTH_SUCCESS]: function (payload)
{
  status = "Login Successful";
  errorMessage = "";

  user.name = payload.name
},
[Types.AUTH_LOGOFF]: function (payload)
{
  status = "Log Off Successful";
  errorMessage = "";

  user.name = ""
},
```

Navigation

Navigation

```
signal pushSignal(string url)
signal popSignal(string url);

property var breadcrumbs: [initialPage]

property string initialPage: ScreenTypes.LOGIN_SCREEN

readonly property var routes: [
    {
        id: ScreenTypes.LOGIN_SCREEN,
        name: "Login",
        path: "qrc:/UI/Views/Login/Login.qml",
    },
    {
        id: ScreenTypes.REGISTER_SCREEN,
        name: "Register",
        path: "qrc:/UI/Views/Register/Register.qml",
    },
    {
        id: ScreenTypes.HOME_SCREEN,
        name: "Home",
        path: "qrc:/UI/Views/Home/Home.qml",
    }
]

function getActiveScreenName()
{ ... }

function router(id)
{ ... }

function push(id)
{ ... }

function goBack()
{ ... }

function pop()
{ ... }
```

```
Navigation.push(ScreenTypes.HOME_SCREEN)
```

```
Navigation.push(ScreenTypes.REGISTER_SCREEN)
```

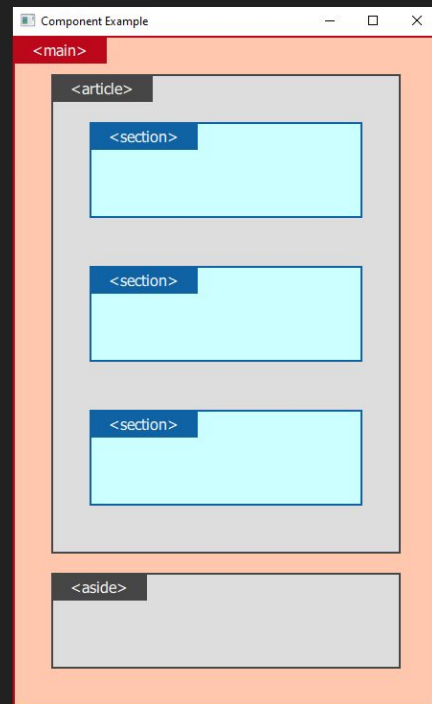
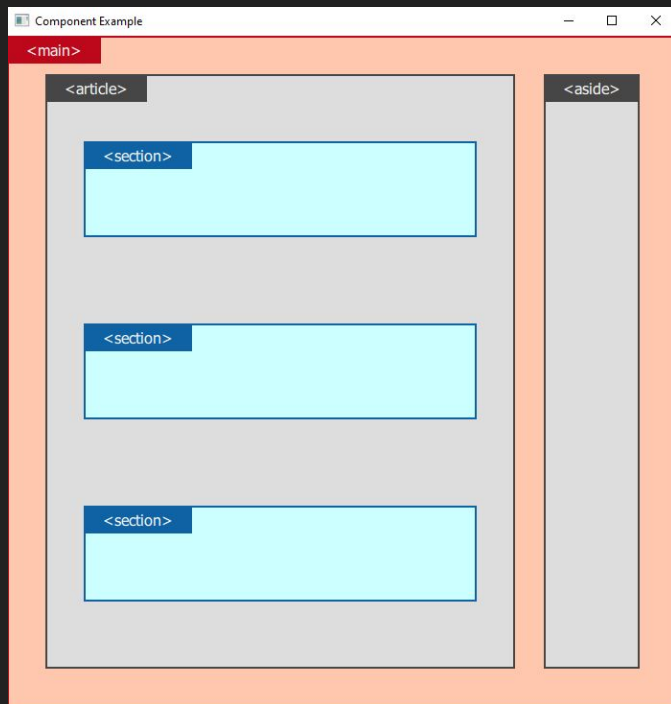
```
Navigation.goBack();
```

Navigation - main.qml

```
< > main.qml Window
1  import QtQuick 2.15
2  import QtQuick.Window 2.15
3
4  import Styles.Colors 1.0
5  import "Core/Navigation"
6  import "Core/Toast"
7
8  Window {
9      minimumWidth: 450
10     width: 450
11     minimumHeight: 768
12     height: 768
13     color: Colors.application.background
14
15     visible: true
16     title: qsTr("Example Authentication in QT / QML")
17
18     Navigation {}
19
20     ToastManager {}
21 }
22
```

Responsive Breakpoints

Responsive Breakpoints



Responsive Breakpoints

```
states: [  
  State {  
    name: "MOBILE_BREAKPOINT"  
    when: width < 450  
    PropertyChanges { target: gridLayout; columns: 1 }  
    PropertyChanges {  
      target: aside;  
      Layout.fillWidth: true;  
      Layout.fillHeight: false;  
      Layout.preferredHeight: 100  
    }  
    PropertyChanges { target: article; Layout.bottomMargin: 15 }  
  }  
]
```

Demonstration

(Code Available at Github)

~/example-authentication-qt-qml

~/example-components-qt-qml



Thanks

