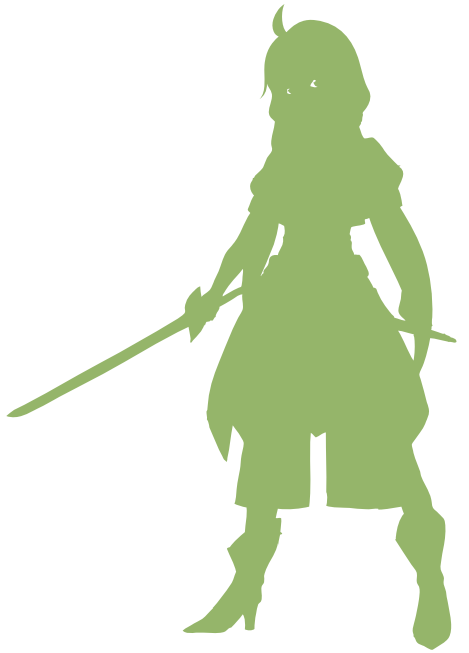
FireMonkey 再入門
分かりづらい原因はスタイルにあった！
スタイル徹底入門

第35回 エンバカデロ・デベロッパーキャンプ



株式会社シリアルゲームズ
取締役　細川 淳

embarcadero®
DEVELOPER CAMP

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

はじめに

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

アジェンダ

- FireMonkey おさらい
- スタイルの仕組み
  - そもそもスタイルとは何なのか
  - スタイルが適用されるときに起こること
  - TextSettings が適用されないとき
- **Delphi 10.2 Tokyo 前提です**
  - Delphi XE8 以前の Style とは大分変わっているので注意してください。

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

FireMonkey おさらい

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

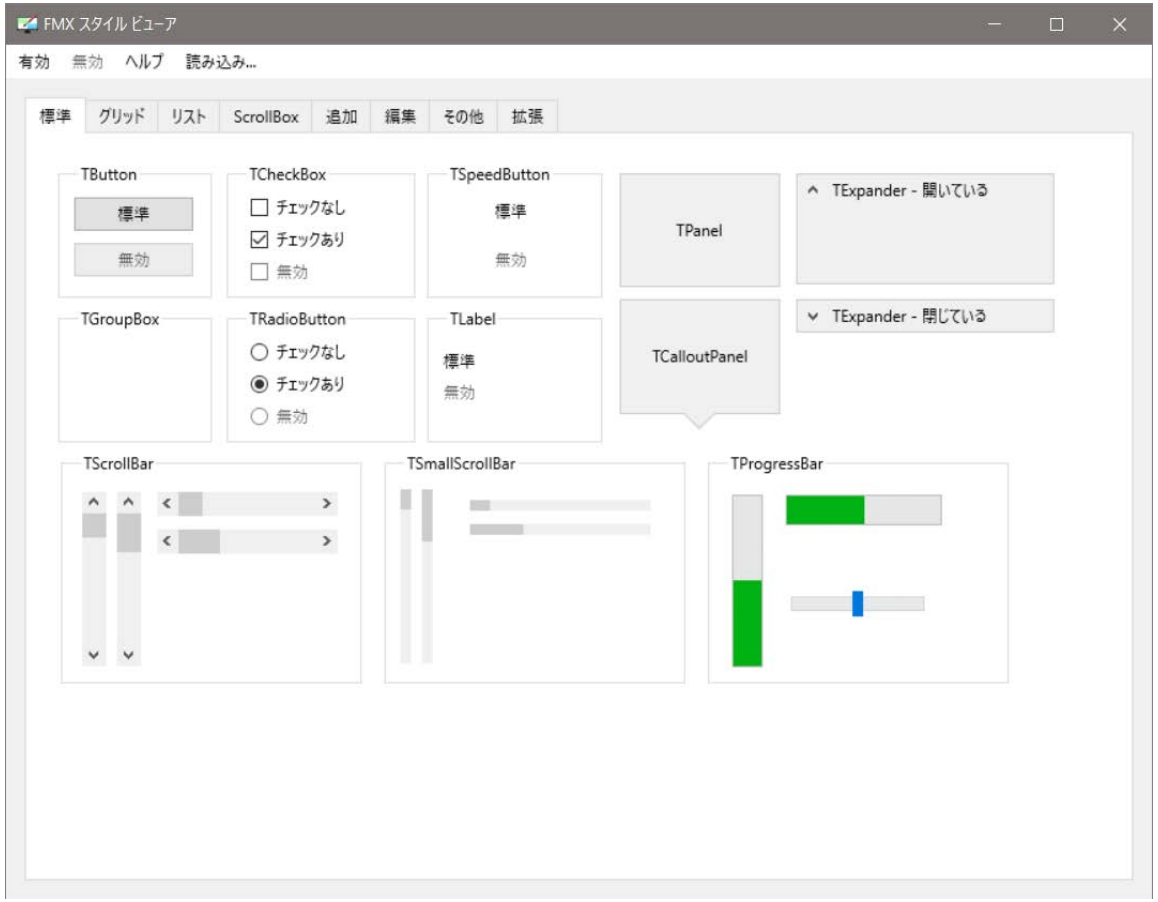
FireMonkey おさらい

- FireMonkey
  - XE2 から導入されたクロスプラットフォームライブラリ
  - コントロールを自分で描画する事で
    クロスプラットフォーム化を果たしています
    - ControlType プロパティを Platform にするとネイティブ化することもできます

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

FireMonkey おさらい

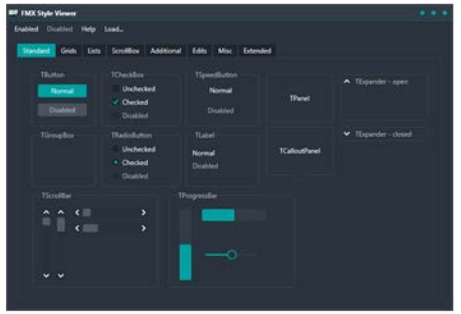
- スタイル
  - FireMonkey が自分で描画する際に利用する「見た目のセット」
    - HTML における CSS のような役割
  - FMXStyleViewer で確認できます
    - インストールパス¥bin¥FMXStyleViewer.exe



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

FireMonkey おさらい

- スタイル
  - 自分で作る事もできます
    - が、非常に大変な作業になります。
    - delphistyles.com で
      スタイルを買っても良いかもしれません。
      https://www.delphistyles.com/



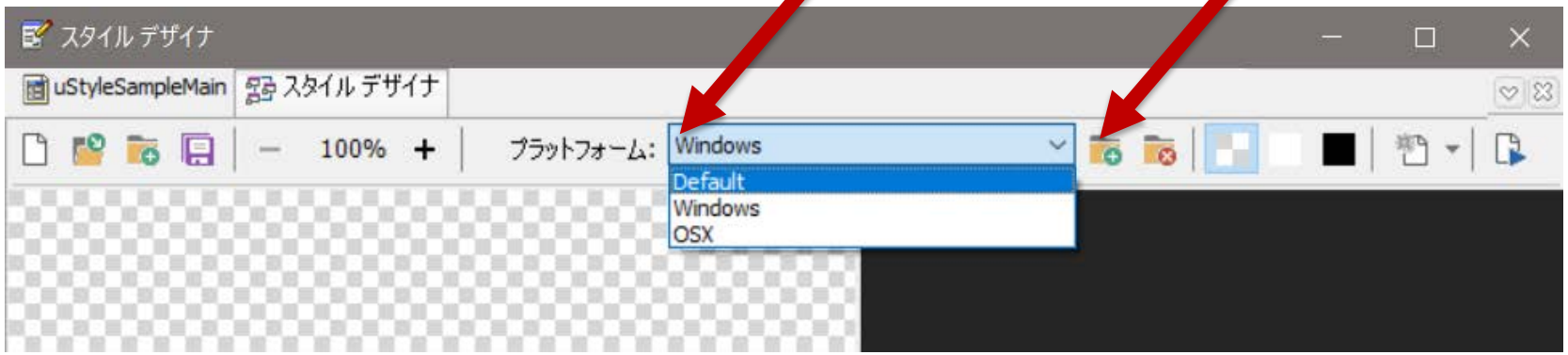
- 第31回 エンバカデロ・デベロッパーキャンプ
  「VCL ユーザーのためのFireMonkey入門」
  も合わせてご覧下さい
  http://www.embarcadero.com/images/jp/dm/dcamp/31/t1.pdf

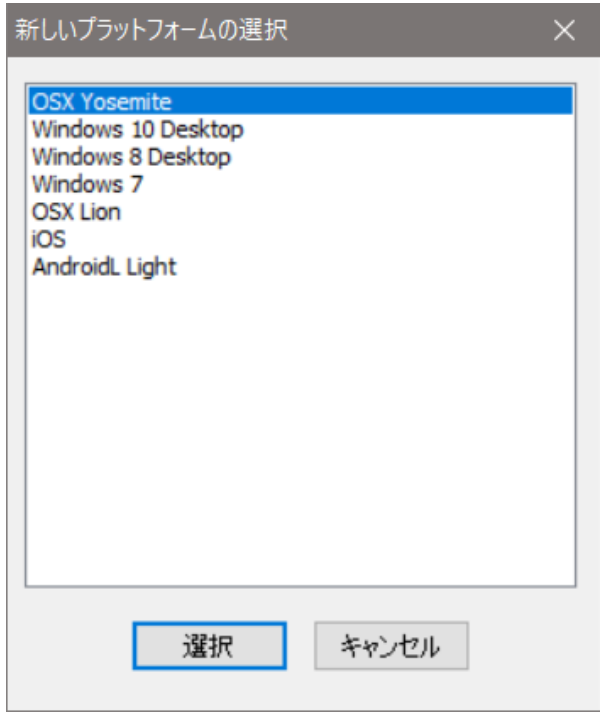
本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 1

- 10 Seattle から StyleBook は
  1 つの Book で複数の OS 用スタイルを保持するようになりました。

プラットフォームの選択リスト　　プラットフォームの追加ボタン

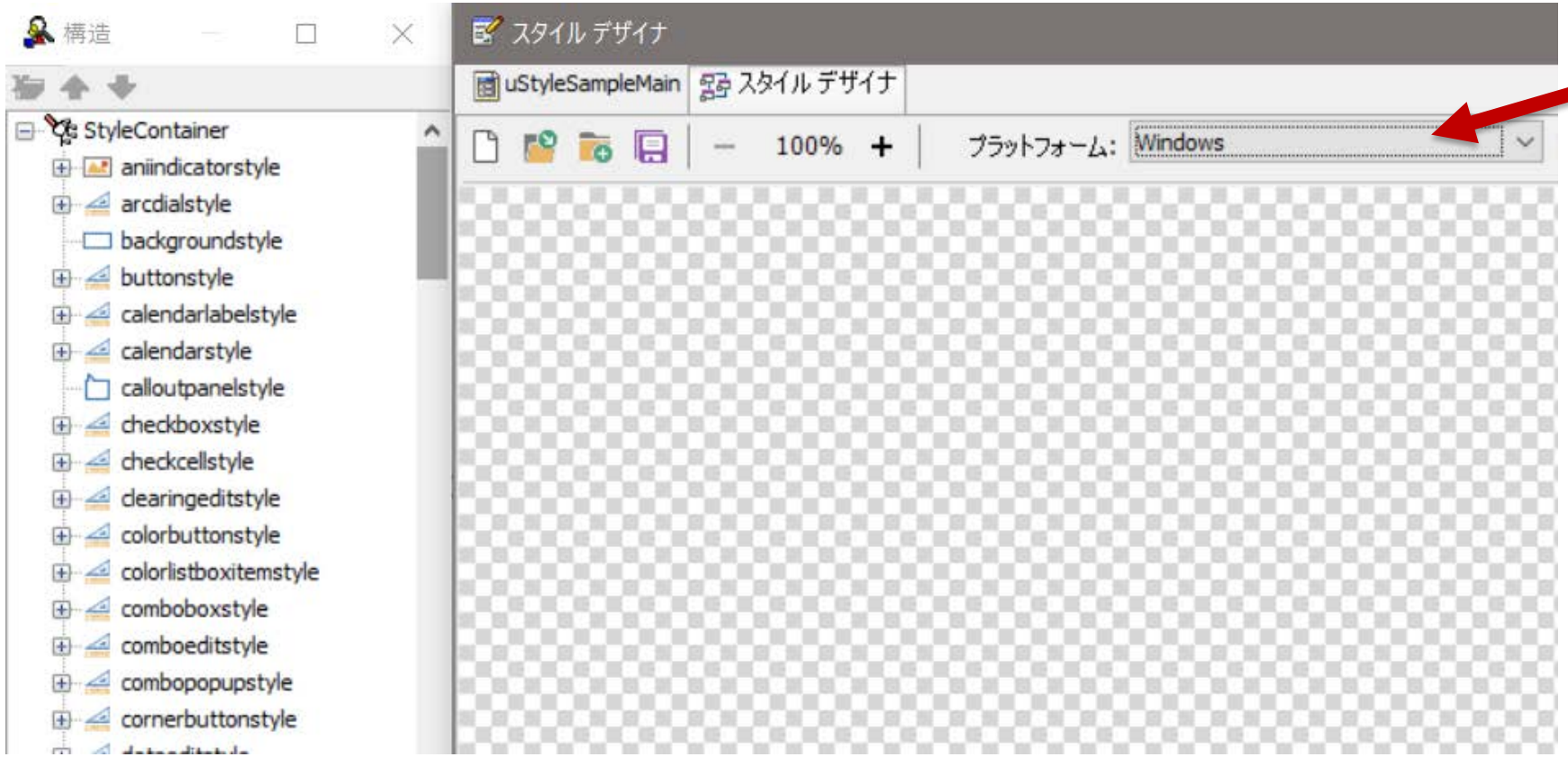




本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 1

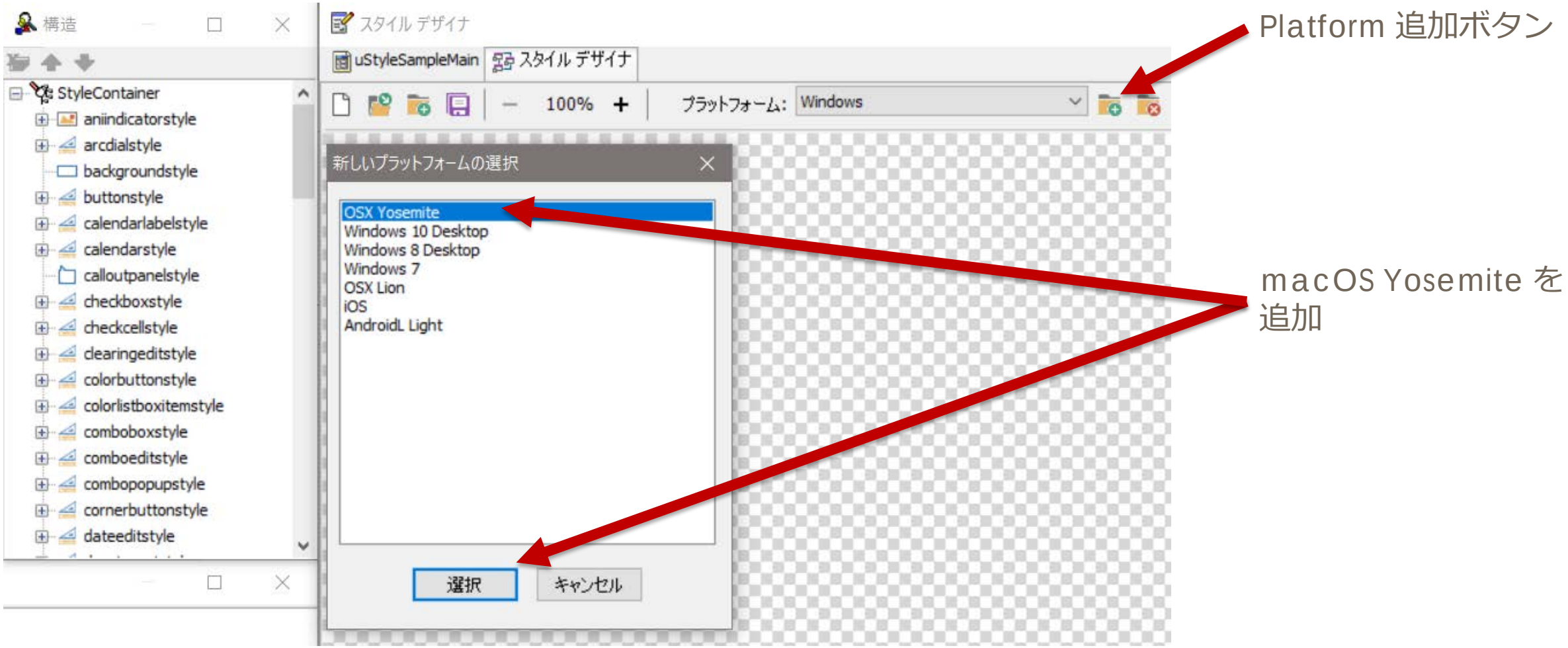
- StyleEditor でプラットフォームの追加をし、
  そのプラットフォームを選択すると、別のスタイルを構築できます。



Windows を選択中

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

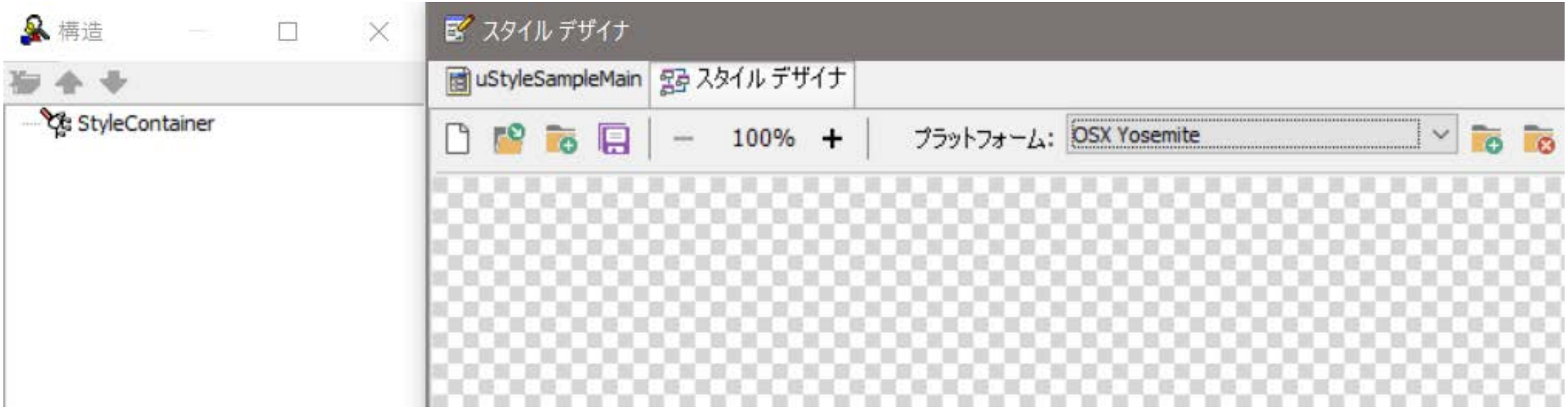
Styleが複雑になっている件 Part 1



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 1

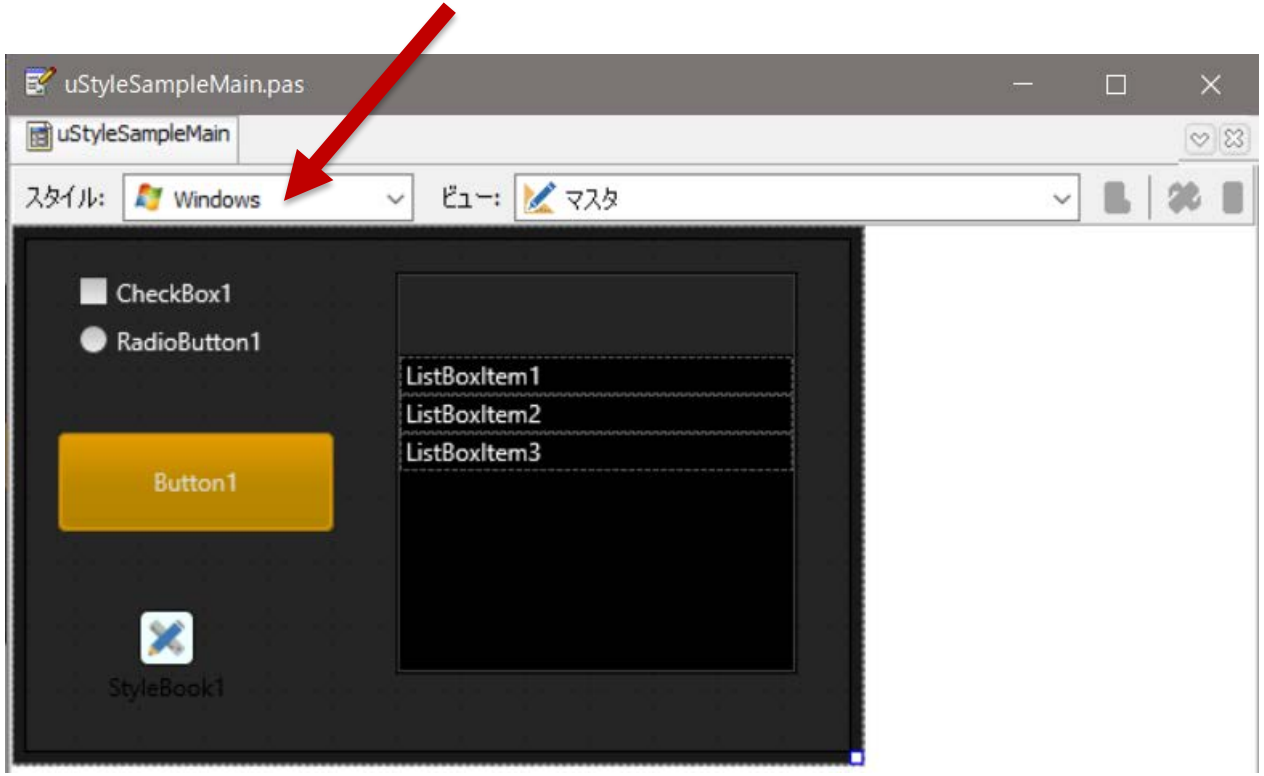
- プラットフォームが変わったので、スタイルがクリアされています。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

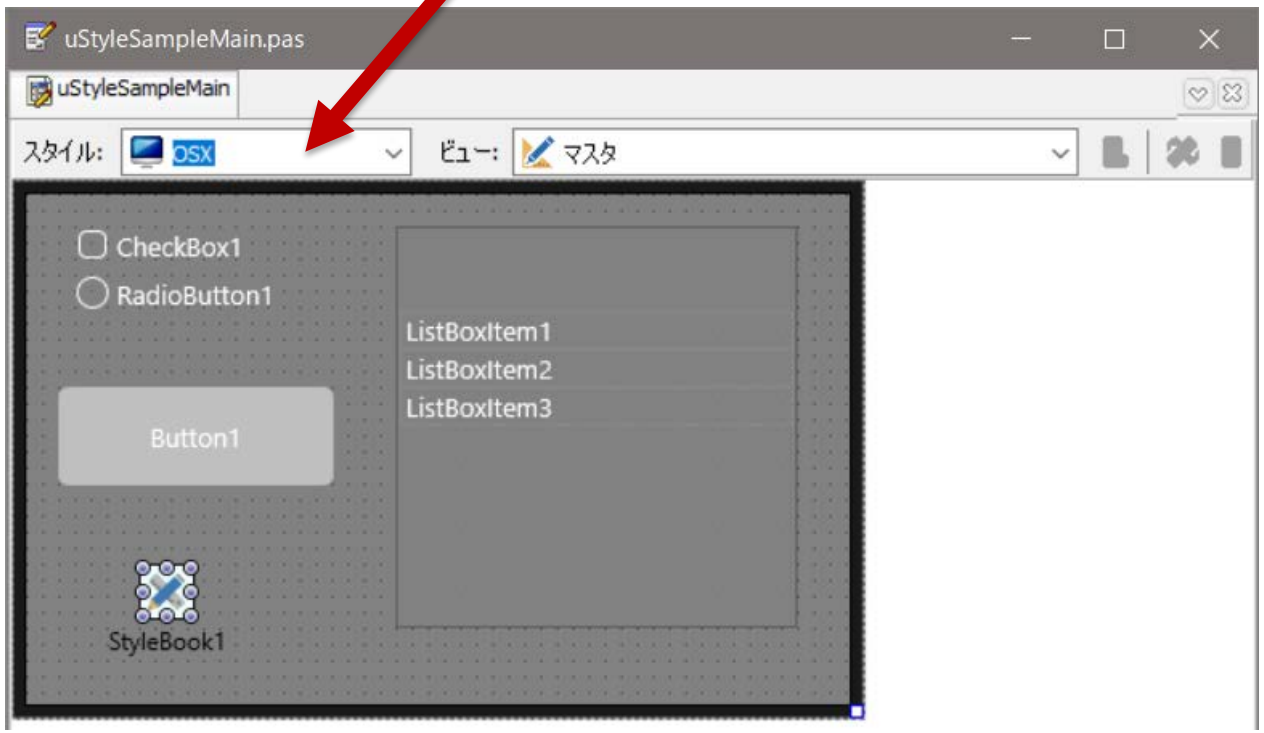
Styleが複雑になっている件 Part 1

- フォームデザイナにもスタイル選択部があります。



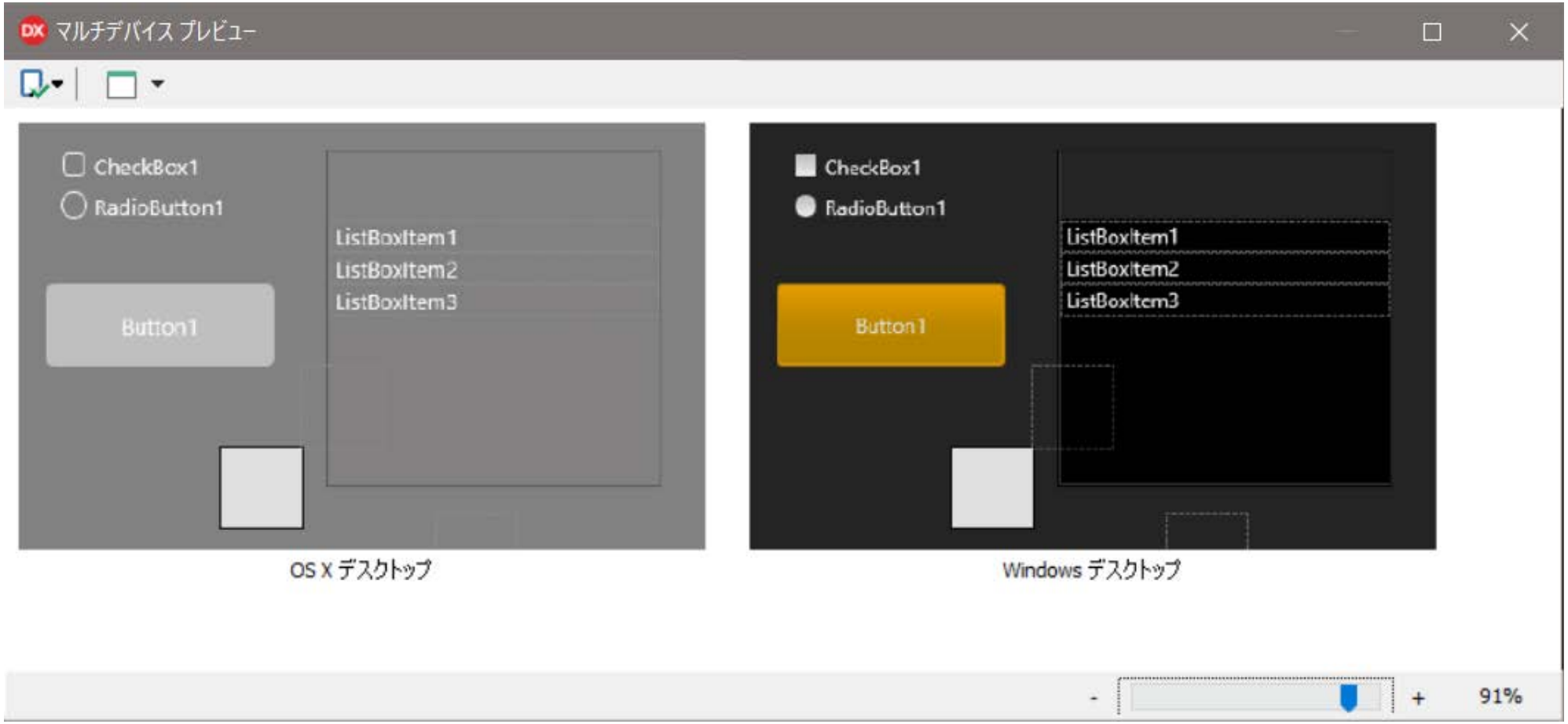
Windows: GoldenGraphite

macOS: YosemiteDark



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

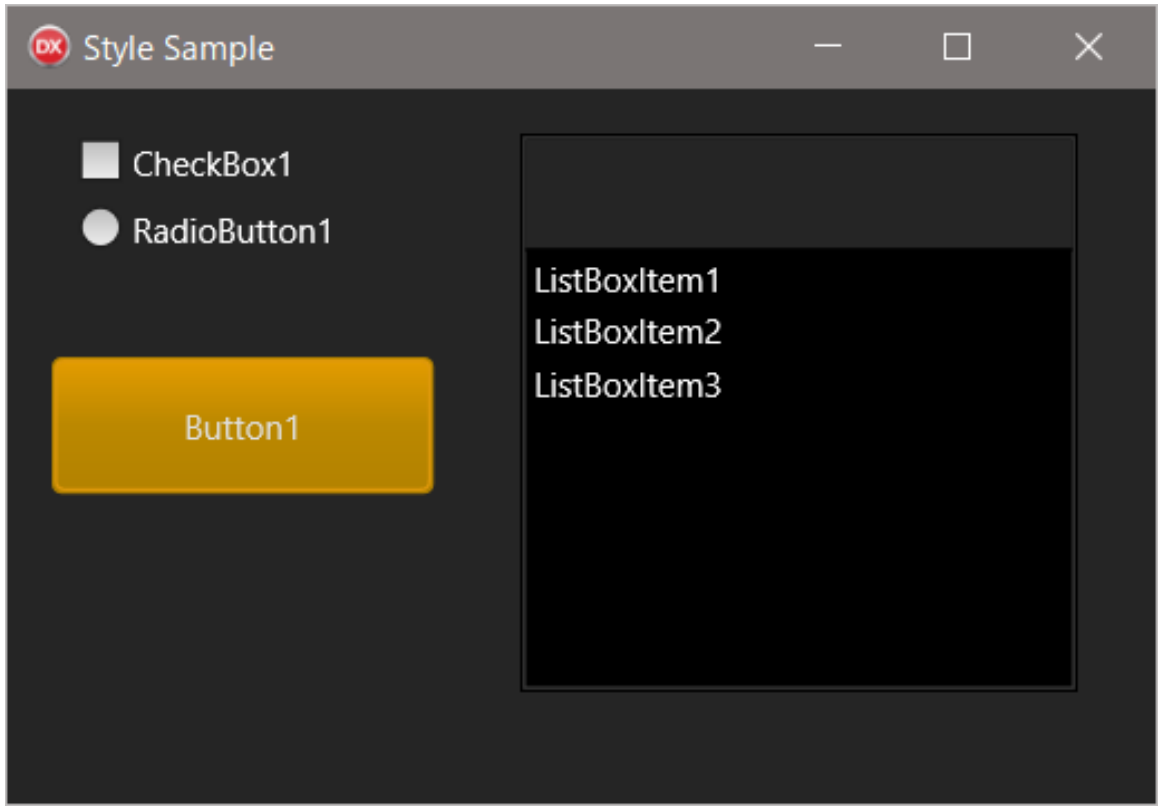
- 当然マルチデバイスプレビューでも表示されます。

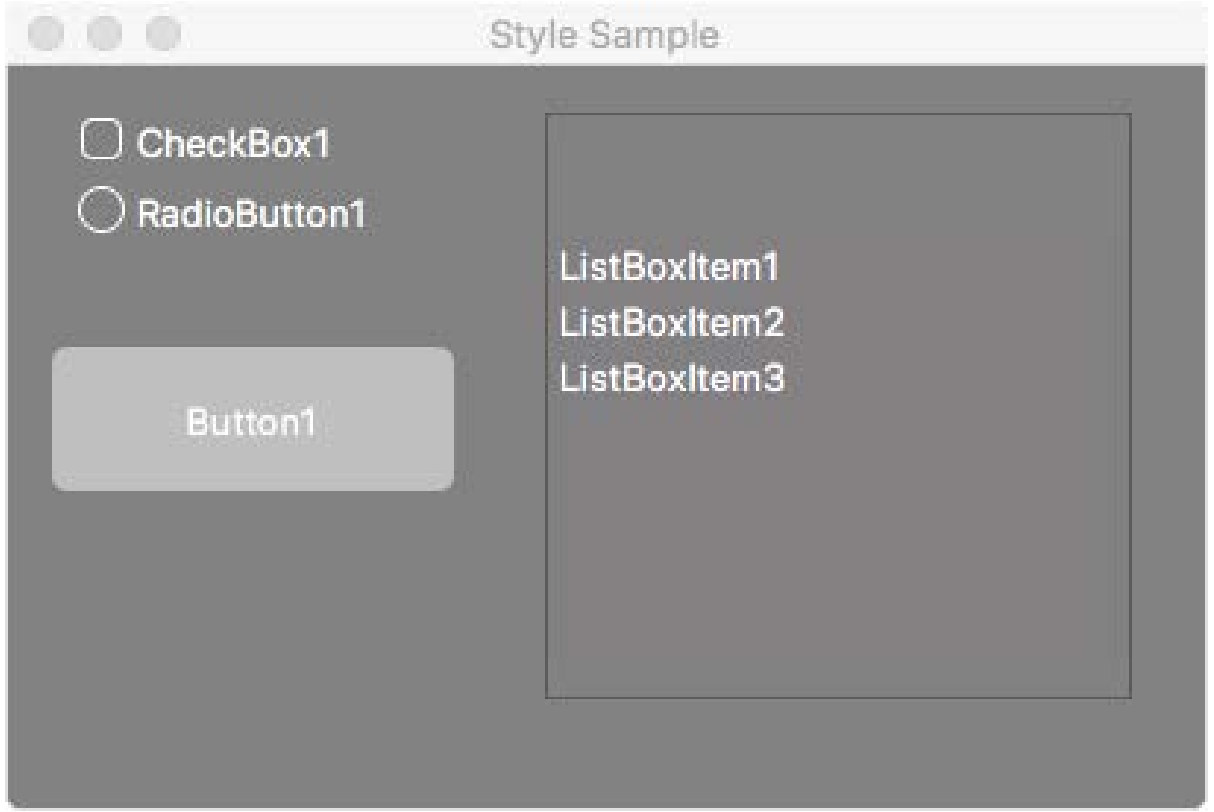


本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 1

- 実際に動作するときは自動的に OS に合わせたスタイルが選択されます。

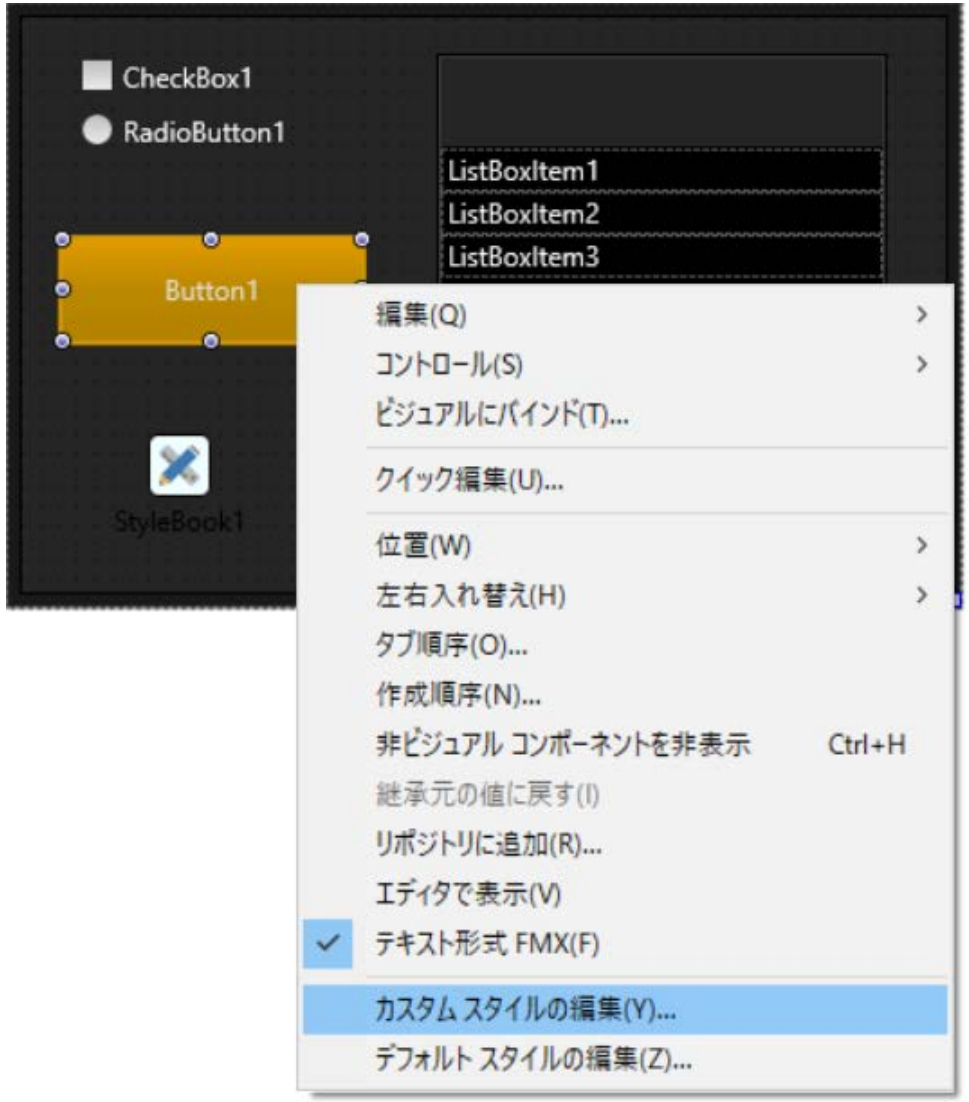




本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 1

- カスタム スタイル
- デフォルト スタイル
  - どちらも現在 IDE で選択されている スタイルに生成されます。
  - まだスタイルを読み込んでいない プラットフォームを選ぶと 自動的にそのプラットフォーム用の スタイルが生成されます。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style の仕組み

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

そもそも Style とは何なのか

- Style の構成要素
  - TControl や TBrush, TStyledControl など
  - TComponent から派生したものは TReader.ReadComponent でストリームからコンポーネントの実体として生成できます。

→つまり、**Style とは**（TForm などと同様に）**ストリームに保存されたコントロール達**

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

そもそも Style とは何なのか

- Style のストリームデータ
  - 前述の通り、コンポーネントストリーム
  - コンポーネント以外のデータ
    - Auther, Title, Version, Platform といった Style の情報を表すデータも保存されています。
    - FMXStyle を表すシグネチャも含まれています
      - FireMonkey Style は現在 Version 2.5

```
// Sign is "FMX_STYLE 2.0"
FireMonkeyStyleSign: TStyleSignature =
  (Byte('F'), Byte('M'), Byte('X'), Byte('_'),
   Byte('S'), Byte('T'), Byte('Y'), Byte('L'), Byte('E'), Byte(' '),
   Byte('2'), Byte('.'), Byte('0'));

// Sign is "FMX_STYLE 2.5"
FireMonkey25StyleSign: TStyleSignature =
  (Byte('F'), Byte('M'), Byte('X'), Byte('_'),
   Byte('S'), Byte('T'), Byte('Y'), Byte('L'), Byte('E'), Byte(' '),
   Byte('2'), Byte('.'), Byte('5'));
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

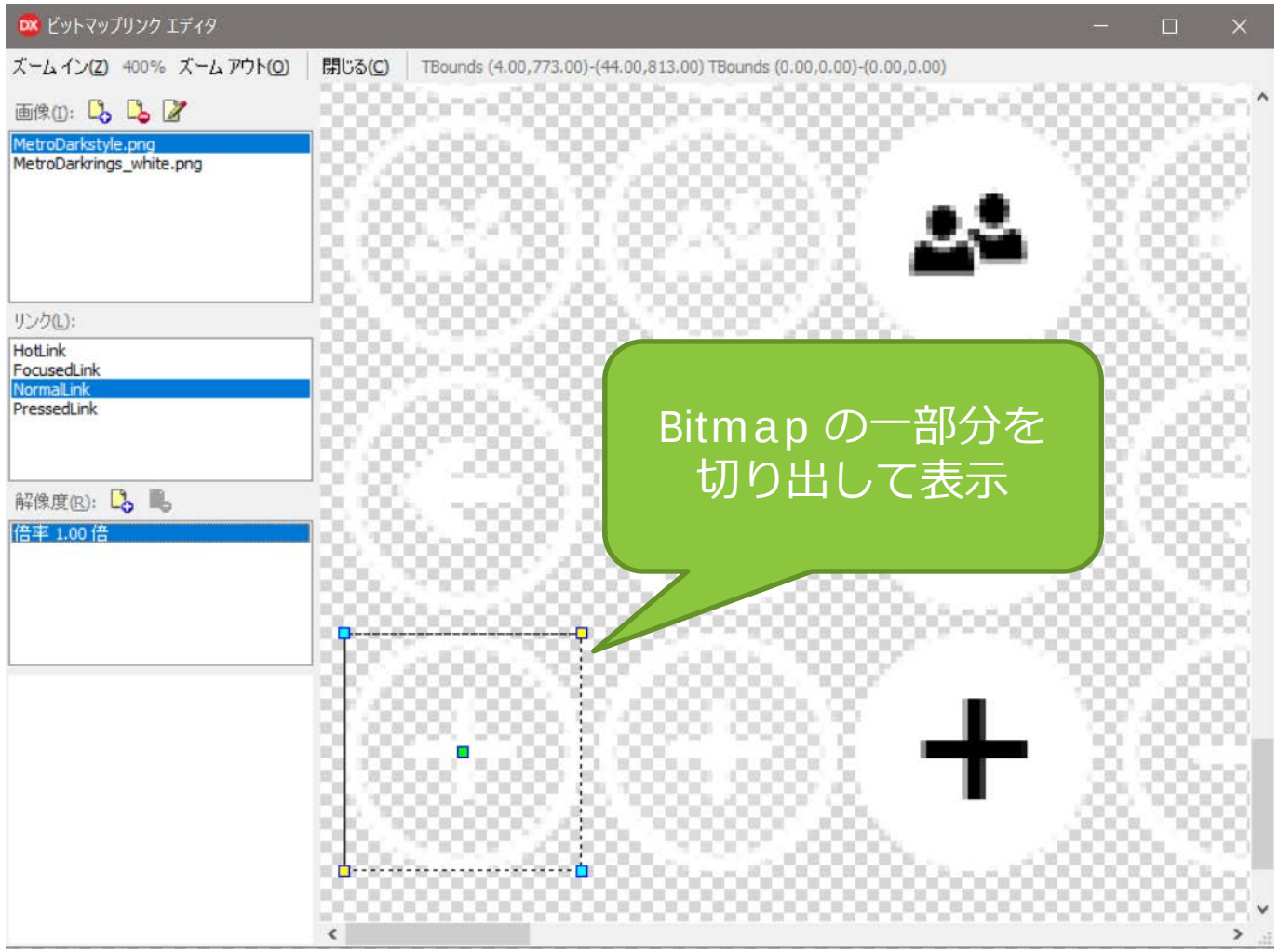
そもそも Style とは何なのか

- Style を形作るコントロール
  - TShape から派生した TRectangle といったシェイプ
  - TText, TImage といった単純にテキストや画像を表示するだけのコントロール
  - TStyledControl から派生しているそれ自体もスタイルを持つコントロール
  - TLayer などのコントロールの整列に関わるコントロール
  - TAnimation や TEffect から派生したアニメーションや効果など
    他のコントロールの見た目に影響するコンポーネント
  - 主に Style でしか使わない特殊なコントロール

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

そもそも Style とは何なのか

- FMX.Styles.Objects
  - ここには TStyleObject など Style に特化したコントロールも登録されています。
  - 例えば TStyleObject は SourceLink という TBitmapLink 型のプロパティで「ラスター画像」をスタイルに埋め込みます。

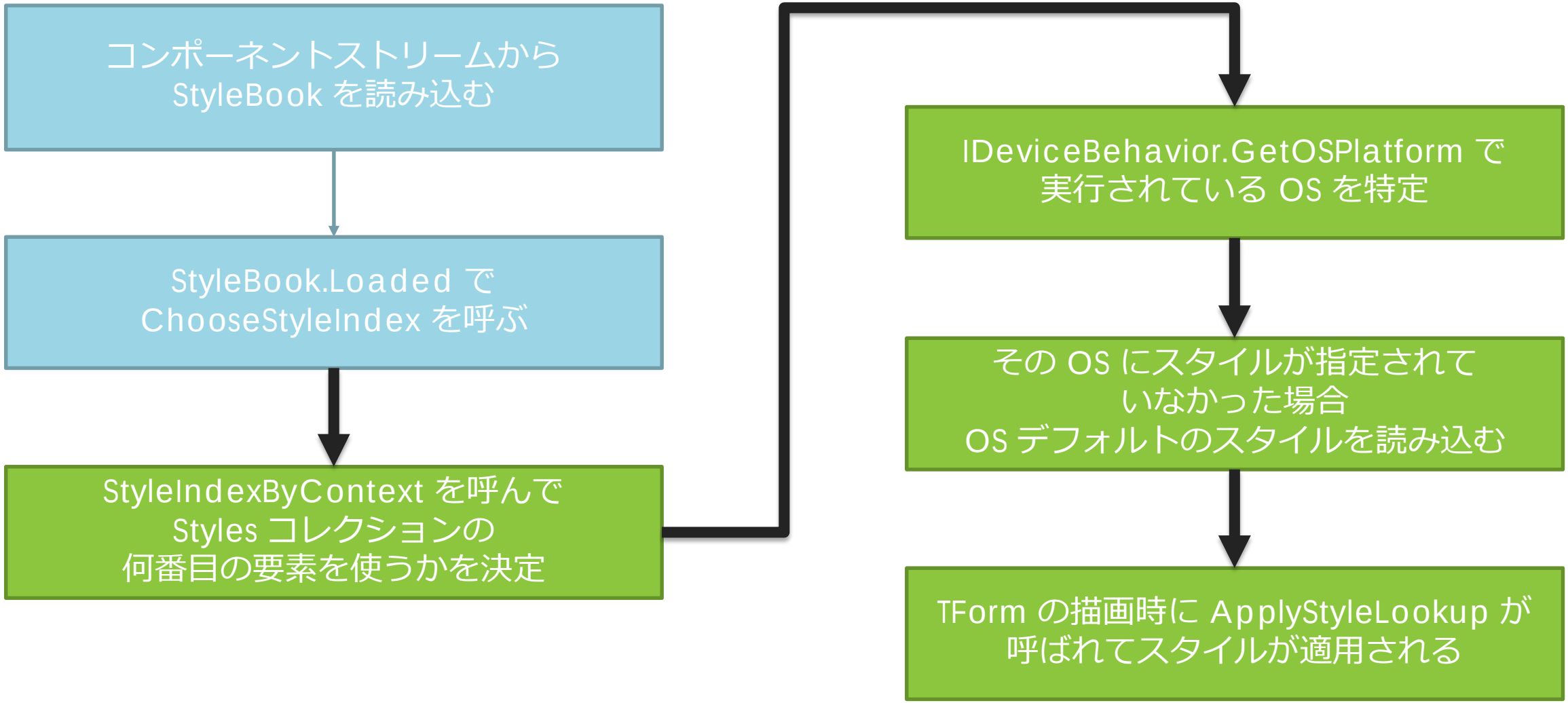


本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

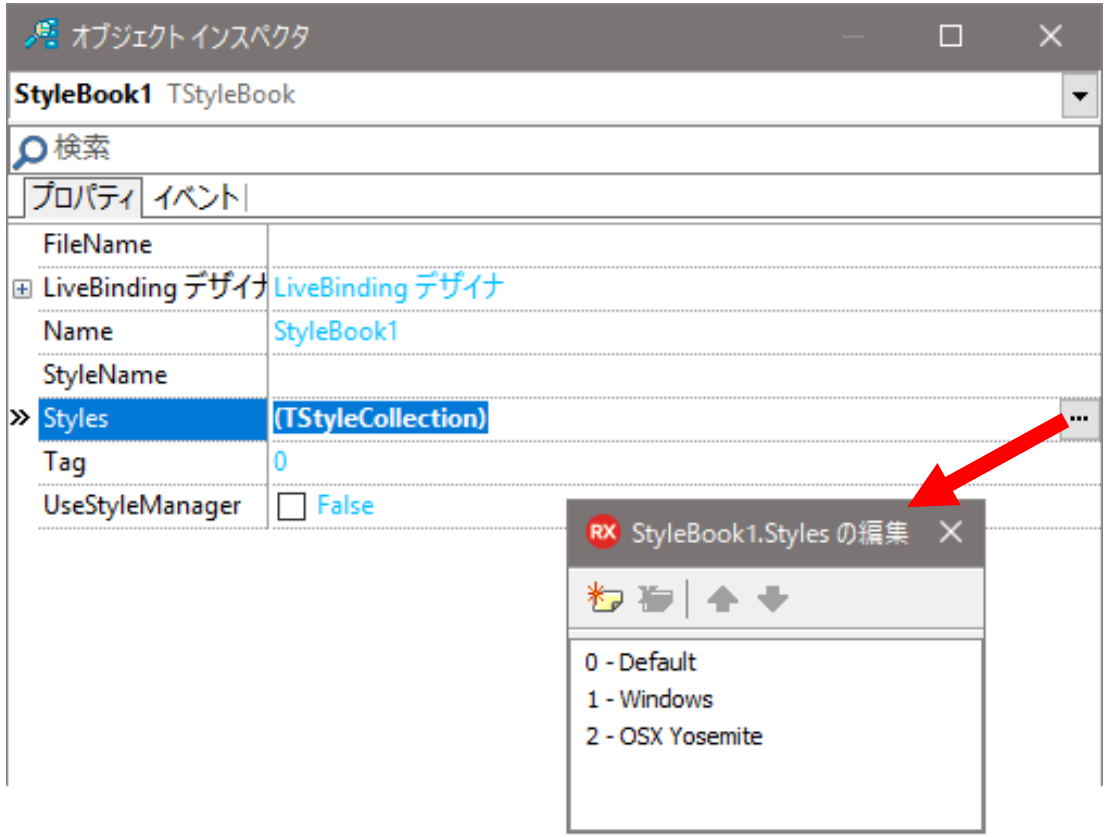
Style 適用時に起こること



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- TStyleBook がスタイルを複数持てるようになったので
  スタイルはコレクションになっています。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- コレクション化された Style の
  どのスタイルを使うか選択する処理が最初に走ります。

- ビヘイビアサービスを使って、実行している OS を特定し
  適したスタイルを決定します。
  - ビヘイビアサービスについては David I が詳しく解説しています
  - https://www.slideshare.net/davidintersimone/rad-in-action-fireui

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- Behavior サービス
  - FireUI 導入時に導入されたサービス
  - 実行環境の情報を返すサービス
    - 例えば、プラットフォームの情報や、ディスプレイのメトリクスなど。

```
function TCustomButton.GetDefaultSize: TSizeF; // TCustomButton のデフォルトサイズを決定する例
var
  DeviceInfo: IDeviceBehavior;
begin
  if TBehaviorServices.Current.SupportsBehaviorService(IDeviceBehavior, DeviceInfo, Self) then
    case DeviceInfo.GetOSPlatform(Self) of
      TOSPlatform.Windows:
        Result := TSizeF.Create(80, 22);
      TOSPlatform.OSX:
        Result := TSizeF.Create(80, 22);
      TOSPlatform.iOS:
        Result := TSizeF.Create(73, 44);
      TOSPlatform.Android:
        Result := TSizeF.Create(73, 44);
```

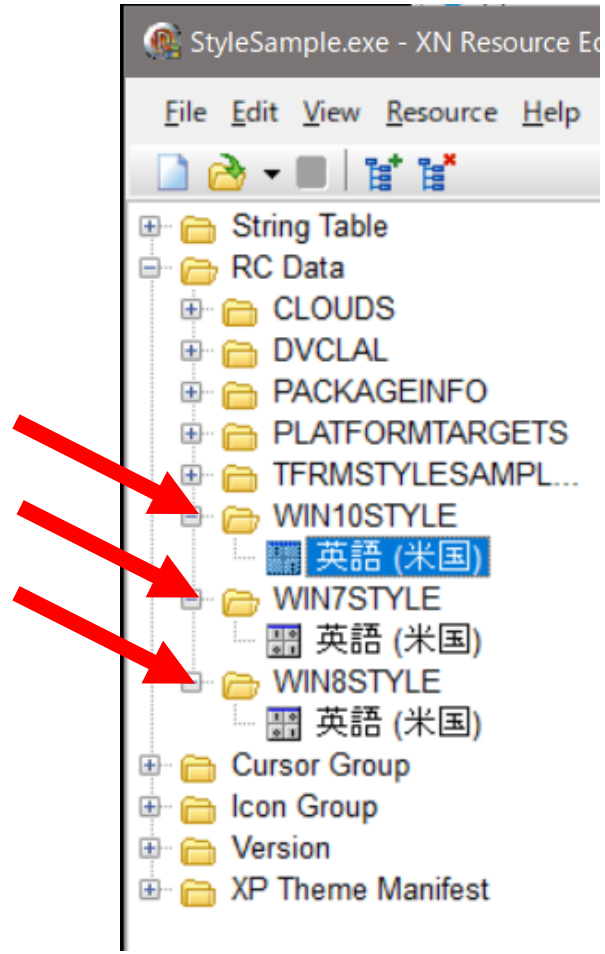
本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- スタイルが含まれない場合はデフォルトのスタイルが選択されます
- RCDATA にデフォルトスタイルが含まれています
  - Win の場合は FMX.Controls.Win.rc
  - macOS の場合は FMX.Conrols.Mac.rc

```
win7style       RCDATA "styles¥platform¥win¥Win7.fsf"
win8style       RCDATA "styles¥platform¥win¥Win8.fsf"
win10style      RCDATA "styles¥platform¥win¥Win10.fsf"
```

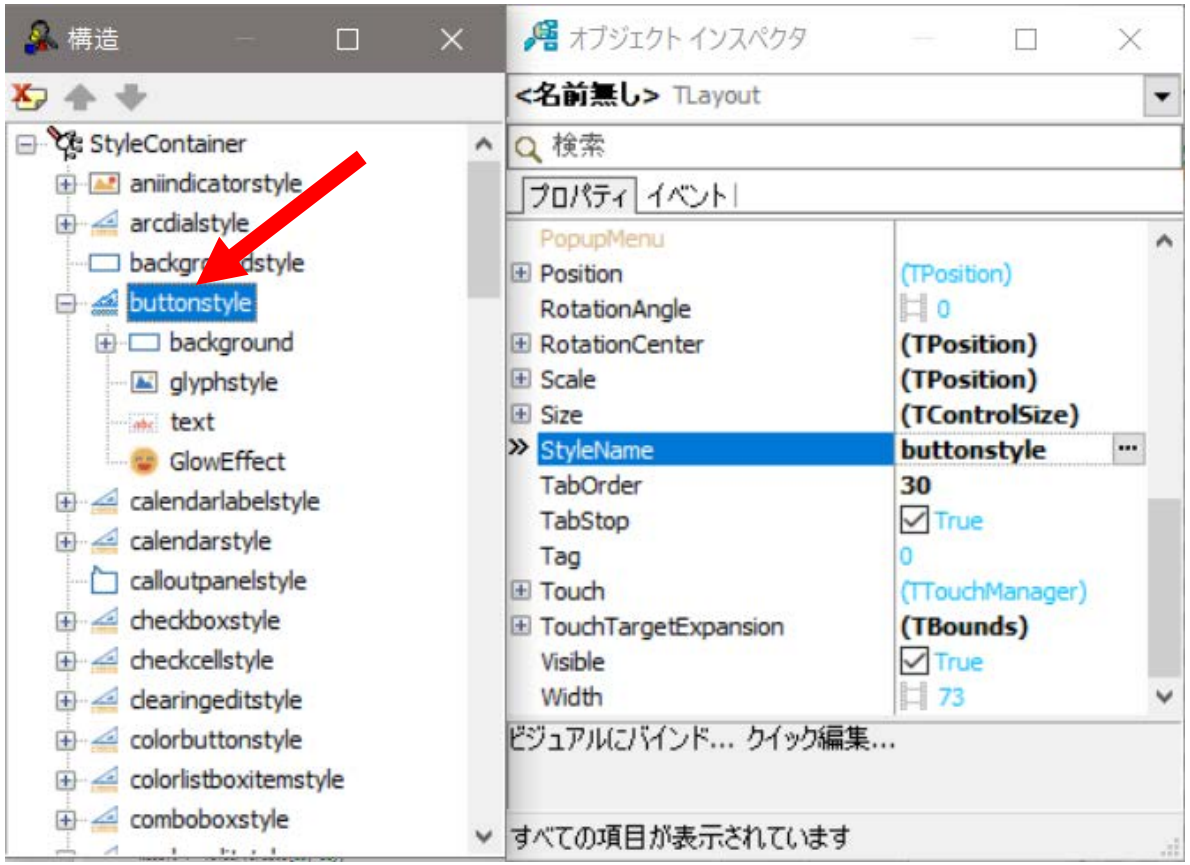
```
osxstyle        RCDATA "styles¥platform¥mac¥Yosemite.fsf"
lionstyle       RCDATA "styles¥platform¥mac¥Lion.fsf"
```



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- ApplyStyleLookup
  - 内部で GetDefaultStyleLookupName が呼ばれて
    どのスタイルを利用するかを決定します
  - 例えば TButton の場合「buttonStyle」が選ばれますが、
    GetDefaultStyleLookupName は override されていません。
  - それでも選ばれる理由は
    TControl.GetDefaultStyleLookupName は「クラス名 - T + style」を
    返すようになっているためです。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- Style は初めての描画時に呼び出されるため
  FormCreate でスタイルを検証しようとしても取得できません

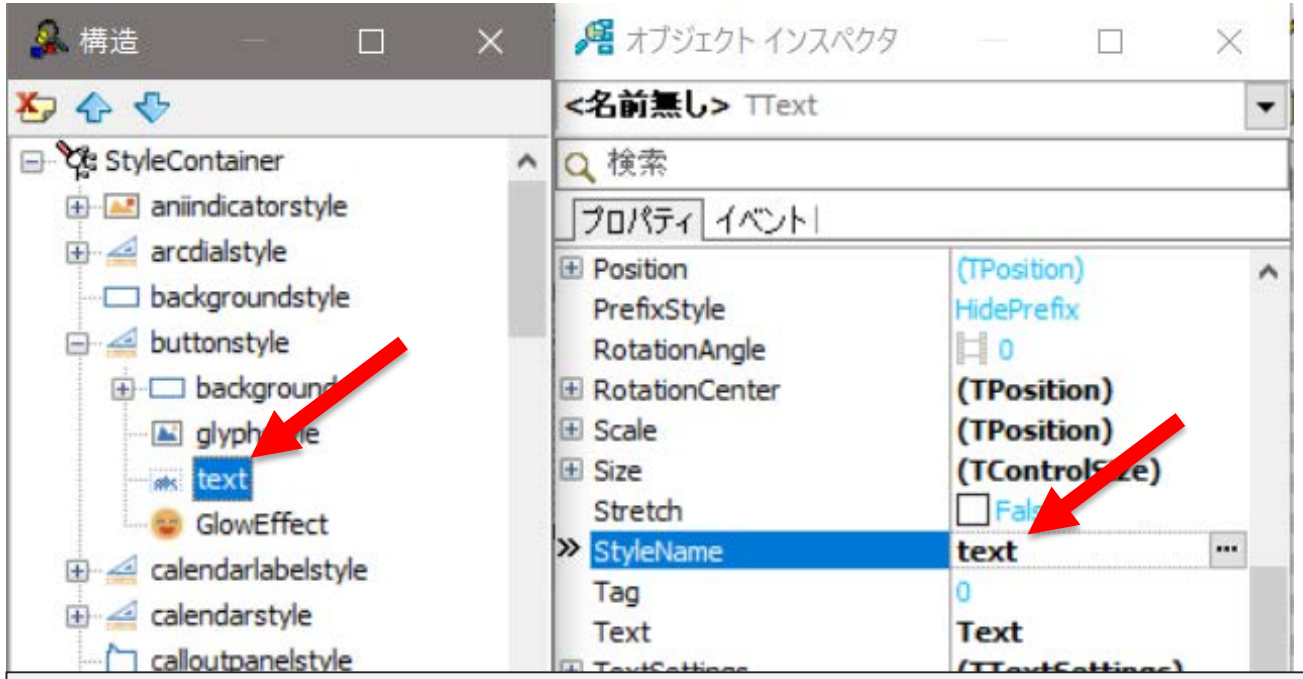
- ApplyStyleLookup でスタイルを適用すると
  スタイルで指定したコントロールが実体化され
  それらが配置されます。

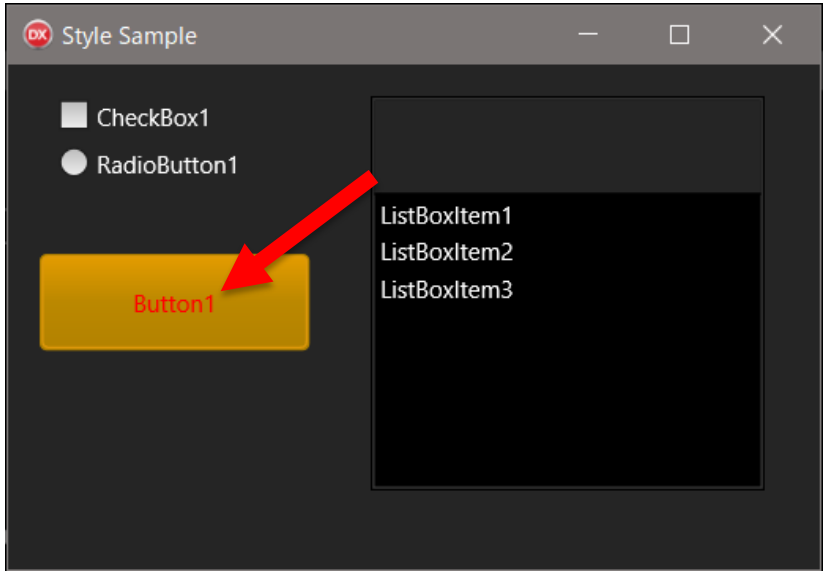
- OnApplyStyleLookup 以降であれば
  FindStyleResrouce で、それらを取り出して変更可能です
  - OnApplyStyleLookup イベントハンドラ は TStyledControl から
    派生しているコントロールが持っています。

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

FindStyleResource で指定するのは
StyleName で指定した名前です



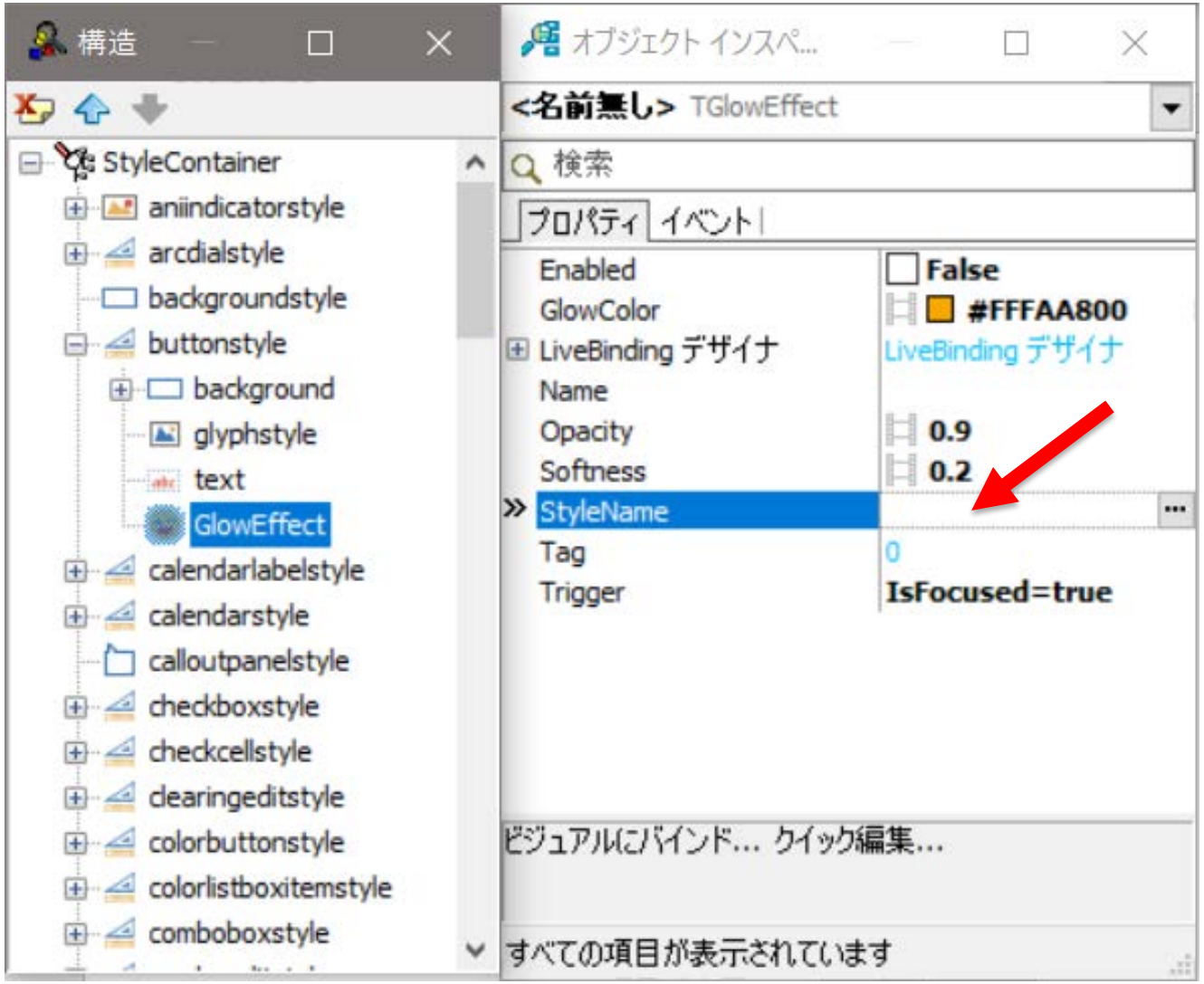



```
procedure TfrmStyleSampleMain.Button1ApplyStyleLookup(Sender: TObject);
var
  Text: TText;
begin
  Text := Button1.FindStyleResource('text') as TText;
  Text.Color := $ffff0000;
end;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- StyleName が
  設定されていない場合は？
- 一番簡単な方法は
  名前を付ける



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Style 適用時に起こること

- StyleName が設定されていない場合は？
  - List から取り出す

```
procedure TfrmStyleSampleMain.Button1Click(Sender: TObject);
var
  Style: TFmxObject;
  Child: TFmxObject;
  Effect: TGlowEffect absolute Child;
begin
  Style := StyleBook1.Style.FindStyleResource(Button1.DefaultStyleLookupName);

  if (Style <> nil) then
    for Child in Style.Children do
      if (Child is TGlowEffect) then
      begin
        Effect.GlowColor := $ff0000ff;
        Effect.Softness := 1;
        Break;
      end;
end;
```

Button1.StyleLookup が
設定されている場合は
そちらを利用するようにします

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 2

- ControlType による変更
  - Presentation と Model の分離
- Presentation
  - コントロールの見た目部分
  - 例えば TEdit であれば、エディットボックスの外見
- Model
  - コントロールのデータ部分
  - 例えば TEdit であれば、保持しているテキスト

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

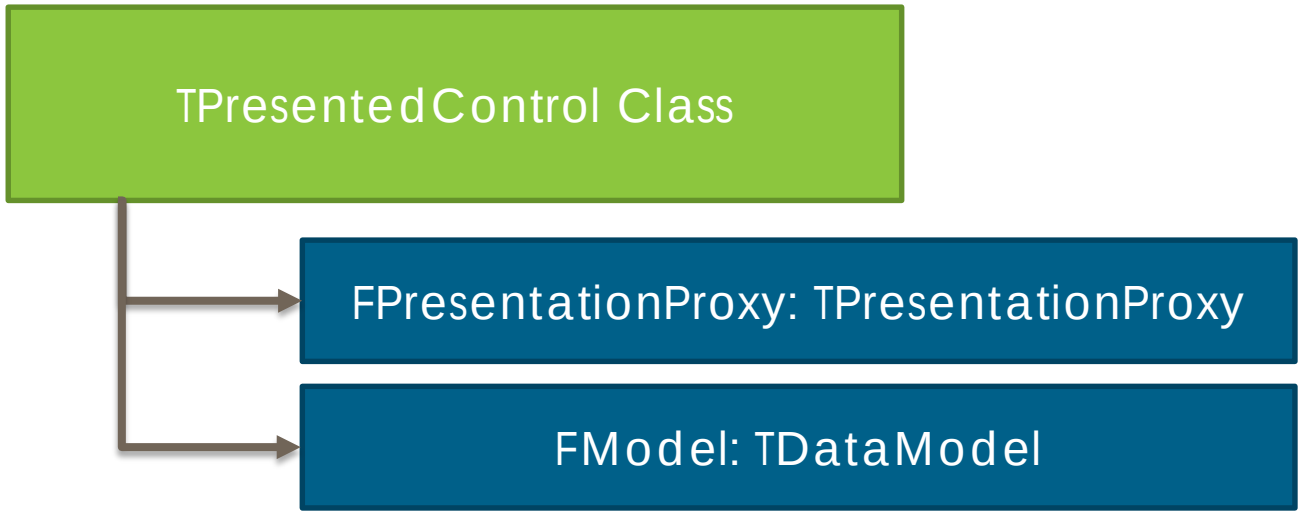
Styleが複雑になっている件 Part 2

- ControlType を持つクラスは TStyledControl を継承した TPresentedControl を継承しています。

- ControlType が platform (= NativeControl) でもデータにアクセスできるようにするため TDataModel というクラスができました。

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 2



FPresentationProxy は TPresentationProxyFactory から得られる TPresentationProxyClass から生成されます

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 2

- TPresentationProxy は見た目の状態変化に応じて
  メッセージを飛ばします

- Presentation の実体では
  そのメッセージに応じて見た目を変化させます。

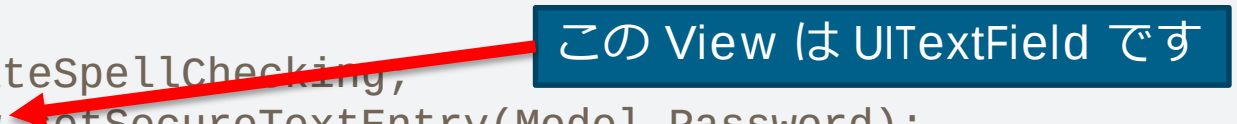
本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

Styleが複雑になっている件 Part 2

- iOS の TEdit の ControlType が platform の時に PresentationProxy が PM_INIT を飛ばすと 下記のコードが呼ばれます。

```
procedure TiOSNativeEdit.PMInit(var AMessage: TDispatchMessage);
begin
  UpdateSpellChecking;
  View.setSecureTextEntry(Model.Password);
  View.setPlaceholder(StrToNSStr(Model.TextPrompt));
  View.setReturnKeyType(ReturnKeyTypeToUIReturnKeyType(Model.ReturnKeyType));
  View.setKeyboardType(VirtualKeyboardTypeToUIKeyboardType(Model.KeyboardType));
  UpdateTextSettings;
  UpdateSelection;
  UpdateVisibleOfClearButton;

  // Hides buttons content, because native TEdit has own buttons
  FSavedButtomsContentVisible := Edit.ButtonsContent.Visible;
  Edit.ButtonsContent.Visible := False;
end;
```



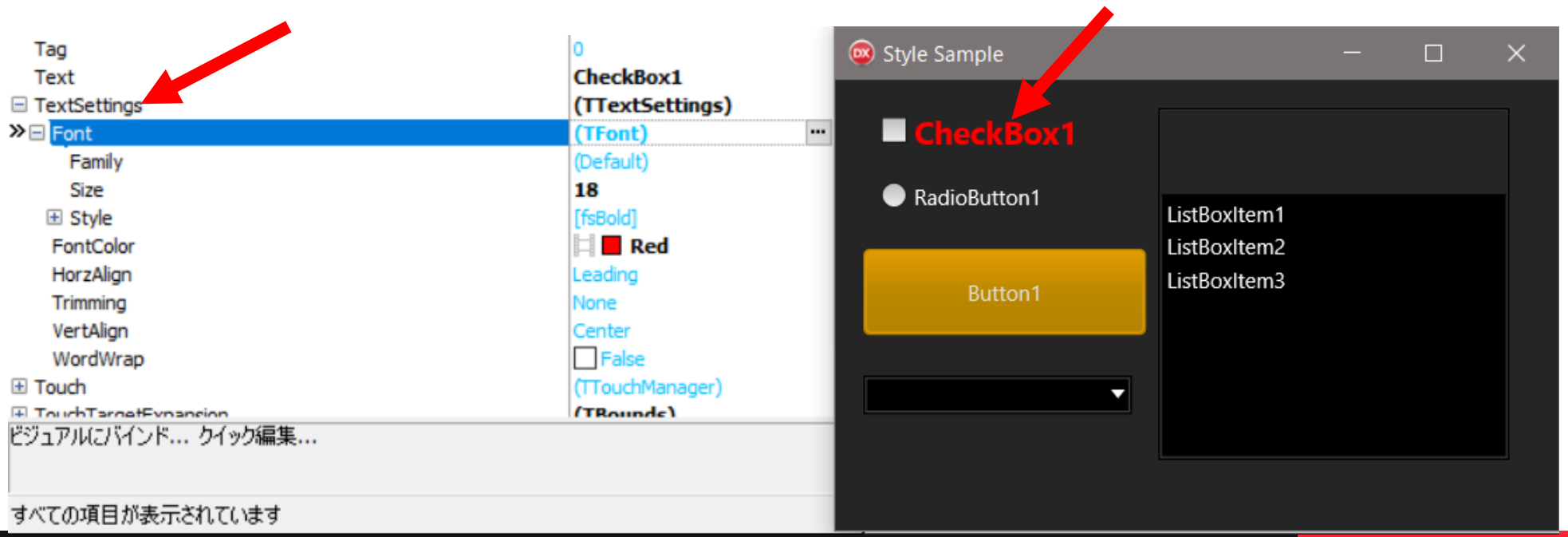
本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- TextSettings は Font や Text Alignment を
  手軽に設定できるプロパティです。
  - 元々は Font も全てスタイルに任されていましたが
    さすがにそれは面倒！...という意見があったのかわかりませんが
    Control のプロパティとして TextSettings が作成されました。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

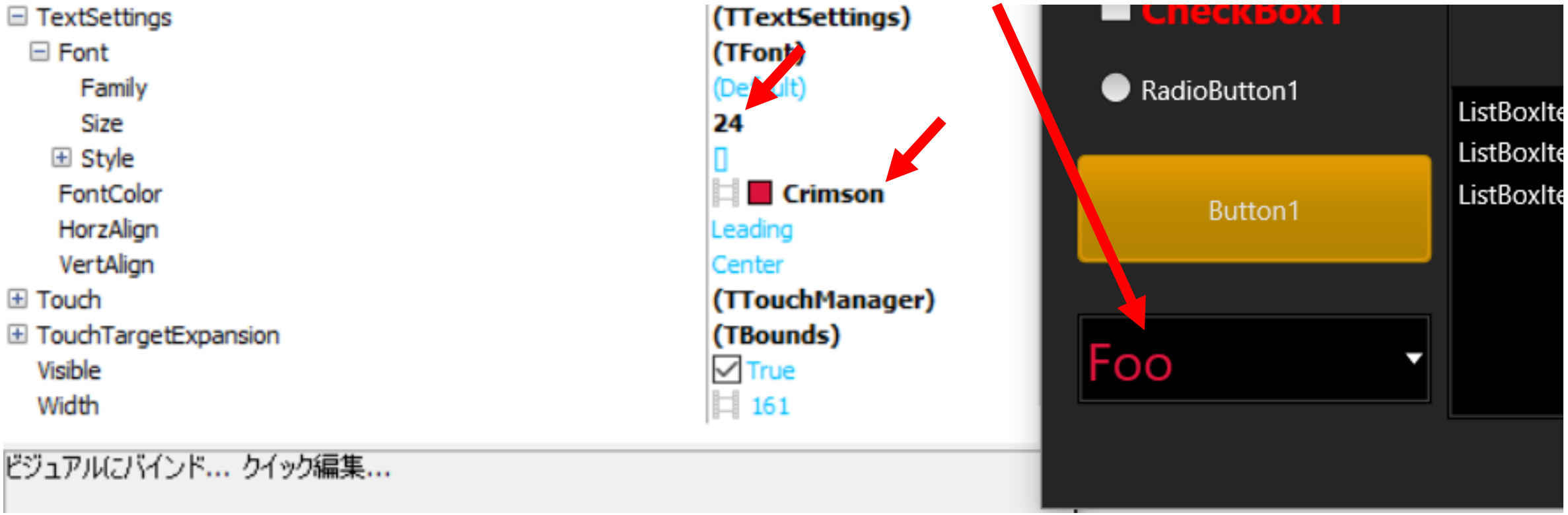
- でも TextSettings が適用されないコントロールがあります。
- これらの TextSettings が適用されていないとき
  何が起きているのでしょうか？

→なにも起きていない！！
**　から設定されない！！**

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

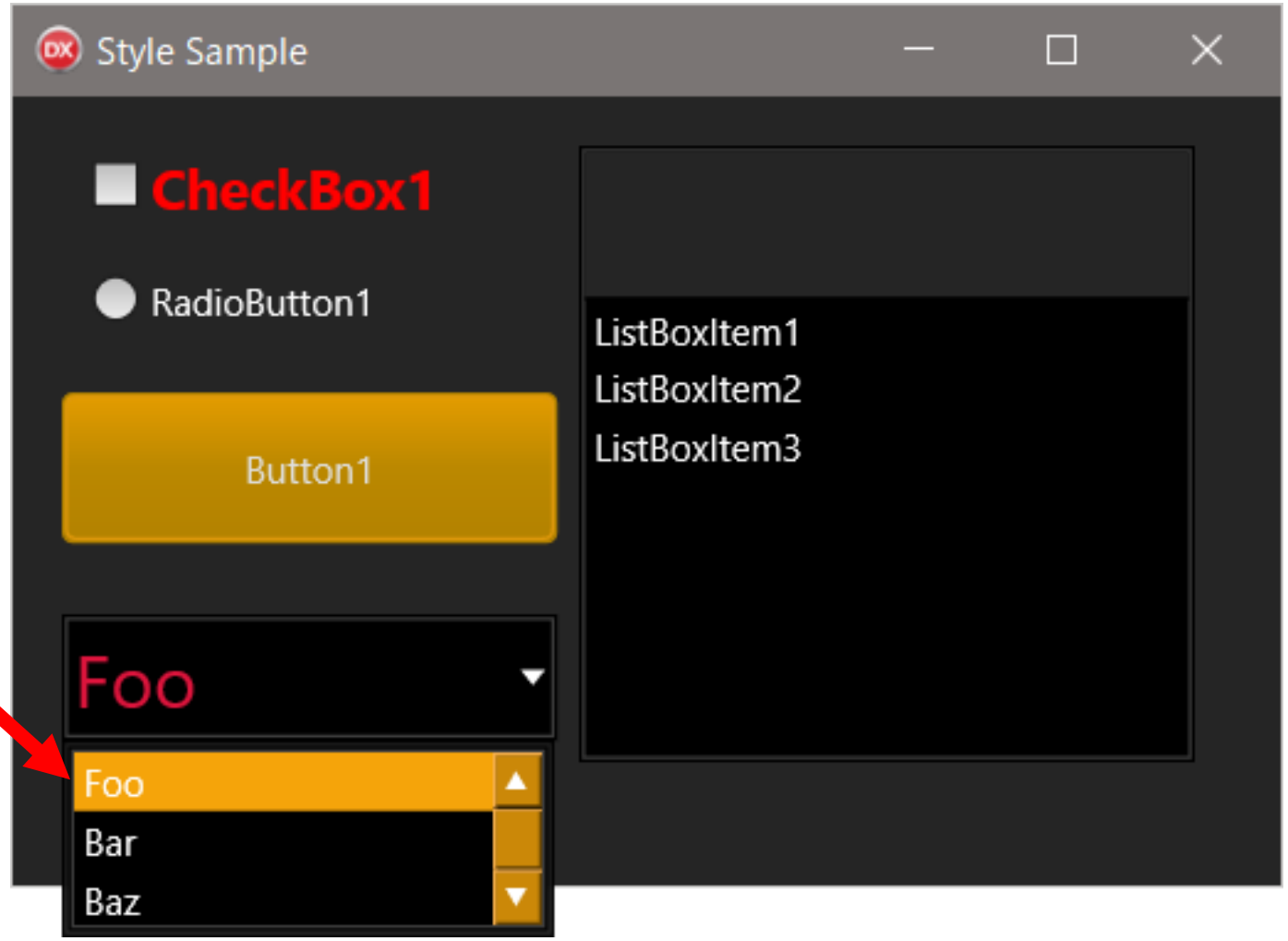
- 例えば TComboEdit について考えてみます
- TextSettings を設定すると…上手く動いているように見えます



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

ドロップしてみると…
出てくる ListBox の見た目は
変更されていません！



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

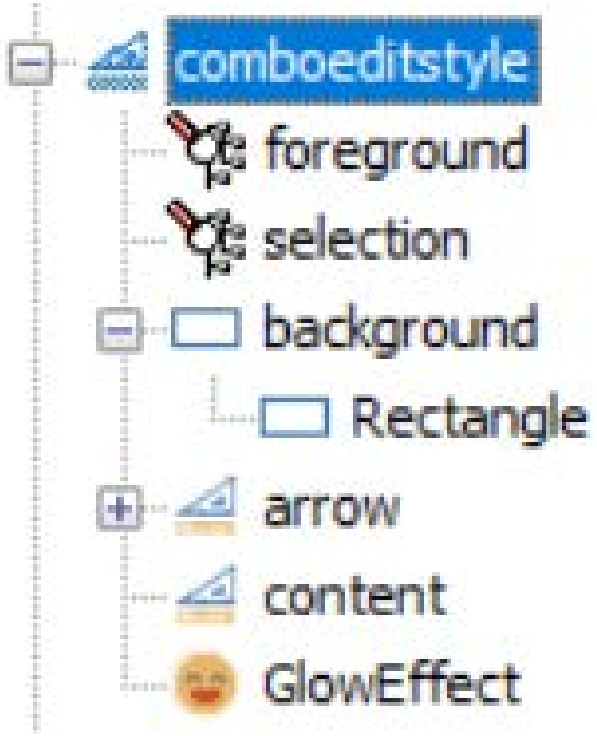
Style が設定されていないんだな！

初めてこの問題を見た僕

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- ComboEdit のスタイルには ListBox 部分がない！



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

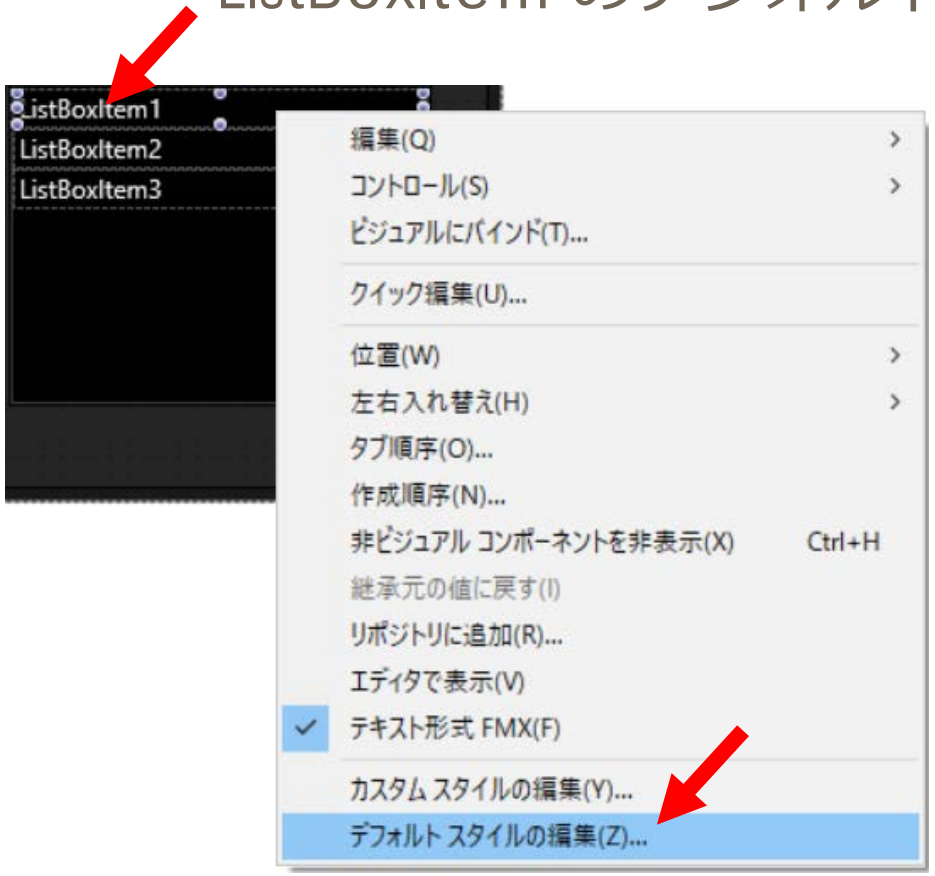
TextSettings が適用されないとき

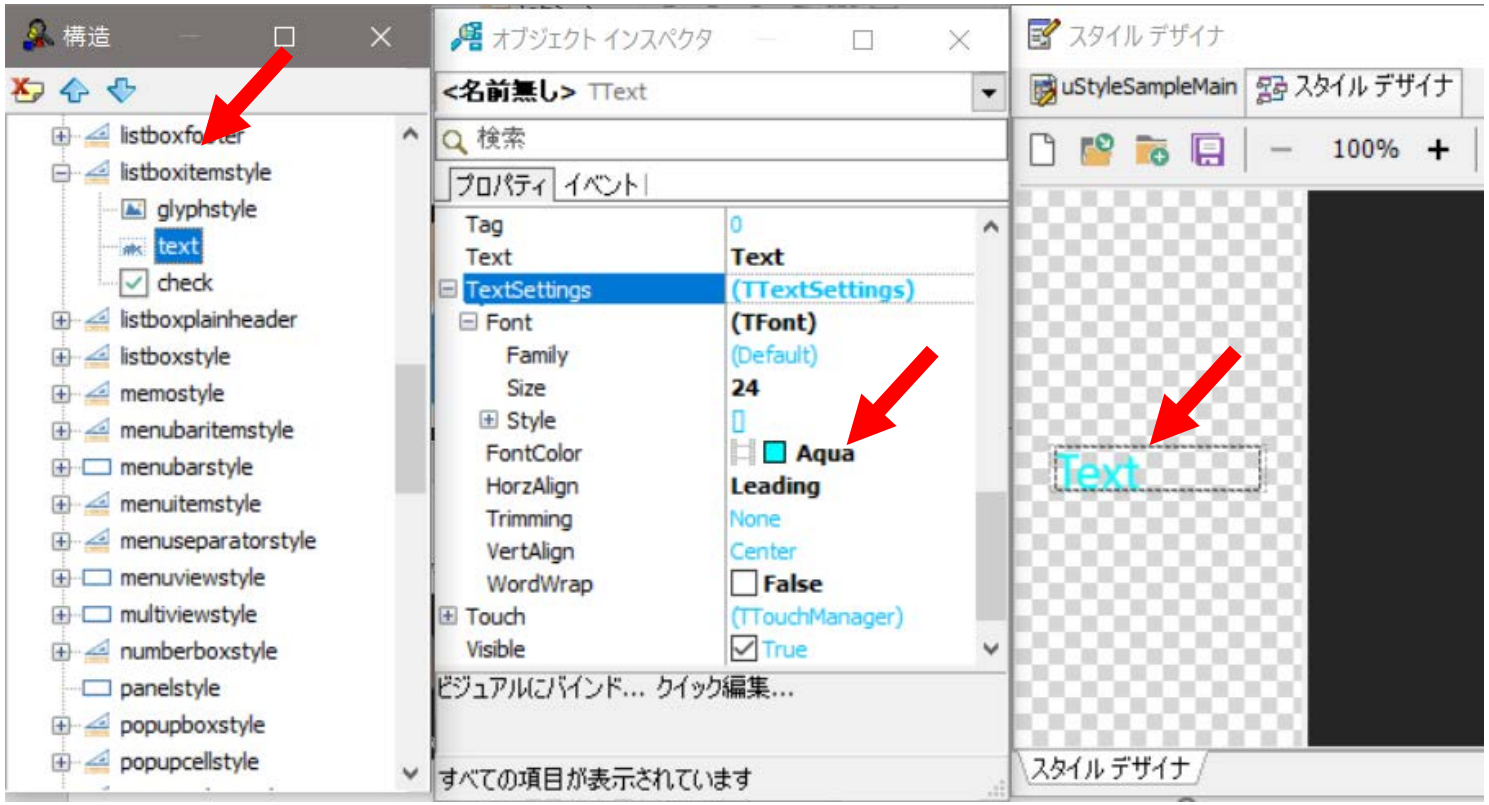
- このような時の解決法
  1. 「デフォルトスタイル」を編集してしまう
  2. 自分でなんとかする

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- デフォルトスタイルの編集
  - Form に ListBox を置いて
    ListBoxItem のデフォルトスタイルを編集します。

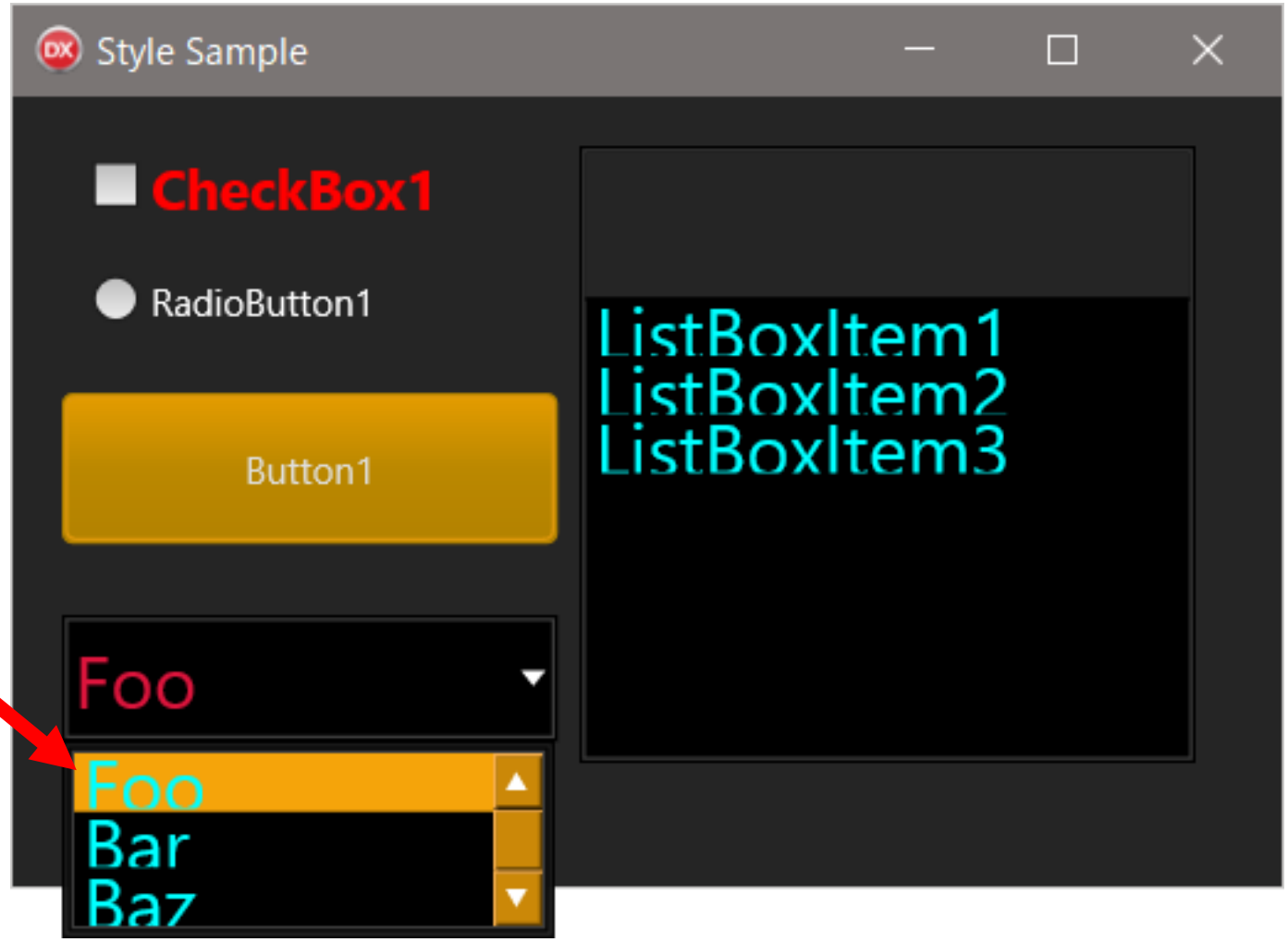




本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

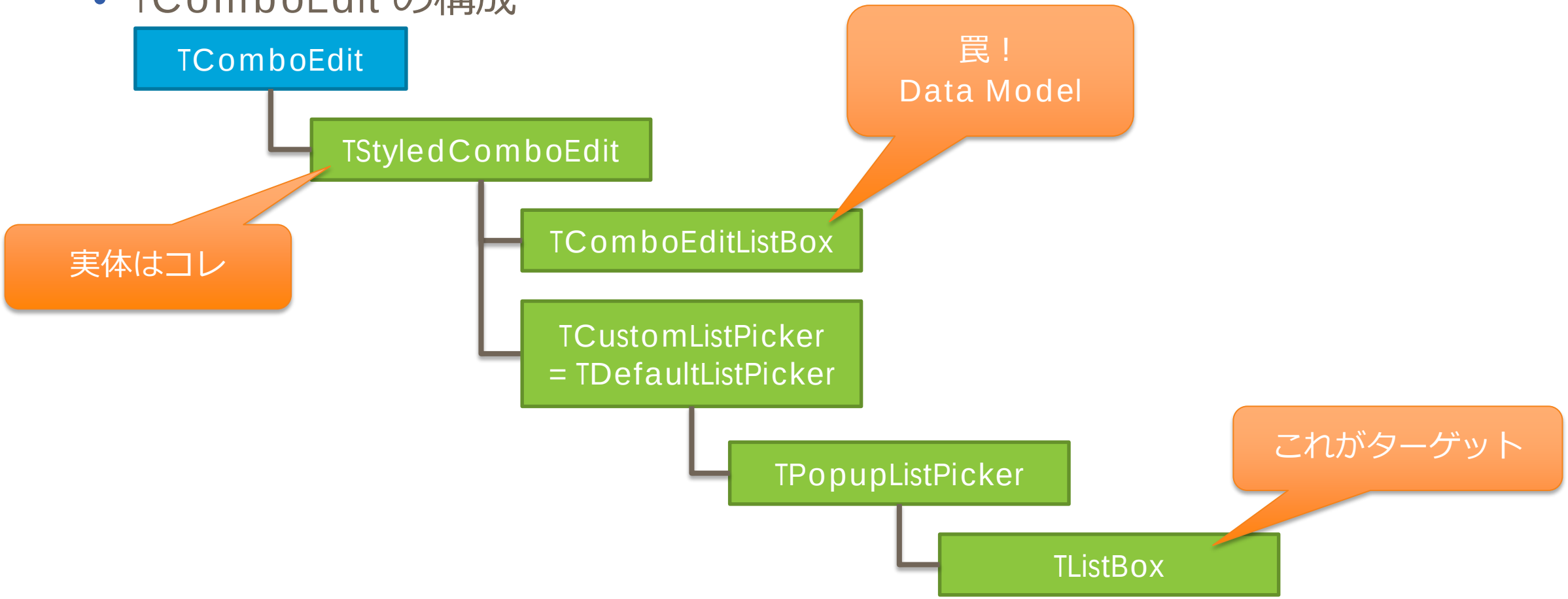
- 全てのリストボックスの見た目が変わってしまいますが、ComboEdit の見た目も変更できました。



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- 自分で何とかする
  - TComboEdit の構成



本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- TComboEdit でドロップされる
  ListBox のフォントを変更するヘルパーを作ります。
  - https://gist.github.com/freeonterminate/bbb7379f13c20c1c0625931bb26a1709

```
type
  TComboEditHelper = class helper for TComboEdit
  private
    function GetEdit: TStyledComboEdit;
    function RemoveFontSettings(
      const Settings: TStyledSettings): TStyledSettings;
    procedure ListBoxApplyStyleLookup(Sender: TObject);
  public
    procedure SetFont(
      const FontFamily: TFontName;
      const FontStyles: TFontStyles;
      const FontSize: Integer;
      const FontColor: TAlphaColor);
  end;
```

SetFont を呼んで
フォントを設定します

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
// TStyledComboEdit を見つけて返します
function TComboEditHelper.GetEdit: TStyledComboEdit;
var
  i: Integer;
begin
  Result := nil;

  for i := 0 to ChildrenCount - 1 do
    if Children[i] is TStyledComboEdit then
    begin
      Result := TStyledComboEdit(Children[i]);
      Break;
    end;
end;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
// Font を設定します。
// ここで RTTI を使って、PopupListPicker の ListBox を取り出します
procedure TComboEditHelper.SetFont(
  const FontFamily: TFontName;
  const FontStyles: TFontStyles;
  const FontSize: Integer;
  const FontColor: TAlphaColor);
var
  Edit: TStyledComboEdit;
  ListBox: TCustomListBox;
  ListBoxPicker: TObject;
  Settings: TTextSettings;
  RttiType: TRttiType;
  RttiField: TRttiField;
begin
  ListBox := nil;

  Edit := GetEdit;
  if Edit = nil then
    Exit;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
RttiType := SharedContext.GetType(Edit.ListPicker.ClassType);
if (RttiType <> nil) then
begin
  RttiField := RttiType.GetField('FPopupListPicker');
  if (RttiField <> nil) then
  begin
    ListBoxPicker := RttiField.GetValue(Edit.ListPicker).AsObject;

    if (ListBoxPicker <> nil) then
    begin
      RttiType := SharedContext.GetType(ListBoxPicker.ClassType);
      if (RttiType <> nil) then
      begin
        RttiField := RttiType.GetField('FListBox');
        ListBox := RttiField.GetValue(ListBoxPicker).AsType<TCustomListBox>;
      end;
    end;
  end;
end;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
  if ListBox = nil then
    Exit;

  GListBoxes.Add(Self, ListBox);
  ListBox.OnApplyStyleLookup := ListBoxApplyStyleLookup;

  Settings := TComboTextSettings.Create(Self); // Owner が破棄
  GFonts.Add(Self, Settings);
  Settings.Font.Family := FontFamily;
  Settings.Font.Style := FontStyles;
  Settings.Font.Size := FontSize;
  Settings.FontColor := FontColor;

  Self.StyledSettings := RemoveFontSettings(Self.StyledSettings);
  Self.TextSettings.Assign(Settings);
end;
```

OnApplyLookup で
ListItem にフォントを設定します

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
procedure TComboEditHelper.ListBoxApplyStyleLookup(Sender: TObject);
var
  ListBox: TCustomListBox;
  Edit: TStyledComboEdit;
  Picker: TCustomListPicker;
  Settings: TTextSettings;
  Item: TListBoxItem;
  i: Integer;
begin
  if not GListBoxes.TryGetValue(Self, ListBox) then
    Exit;

  if not GFonts.TryGetValue(Self, Settings) then
    Exit;

  Edit := GetEdit;
  if Edit = nil then
    Exit;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

```
  Picker := Edit.ListPicker;
  if Picker = nil then
    Exit;

  // Font 設定
  for i := 0 to ListBox.Items.Count - 1 do
  begin
    Item := ListBox.ListItems[i];

    Item.StyledSettings := RemoveFontSettings(Item.StyledSettings);

    if Settings.Font.Family <> '' then
      Item.TextSettings.Font.Family := Settings.Font.Family;

    Item.TextSettings.Font.Style := Settings.Font.Style;
    Item.TextSettings.Font.Size := Settings.Font.Size;
    Item.TextSettings.FontColor := Settings.FontColor;
  end;

  // Font Size に応じた Item の高さ調整
  Picker.ItemHeight := Settings.Font.Size + 2;
end;
```

ListItem.TextSettings に設定します

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

TextSettings が適用されないとき

- 実際に使ってみた結果がこちら

```
procedure TfrmStyleSample.Button1Click(Sender: TObject);
begin
  ComboEdit1.SetFont('', [TFontStyle.fsBold], 24, TAlphaColorRec.Aqua);
end;
```

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

まとめ

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

まとめ

- Style は FireUI や ControlType の追加で進化していた！
- FindStyleResource などの Style 関連処理を知っておくと
  いざという時、なんとかできる！（かも）
- Style の動きを知っておくと
  いざという時、なんとかできる！（かも）
- ライブラリソースを眺めてみると良いかも

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。

THANKS!

www.embarcadero.com/jp

第35回 エンバカデロ・デベロッパーキャンプ

本文書の一部または全部の転載を禁止します。本文書の著作権は、著作者に帰属します。