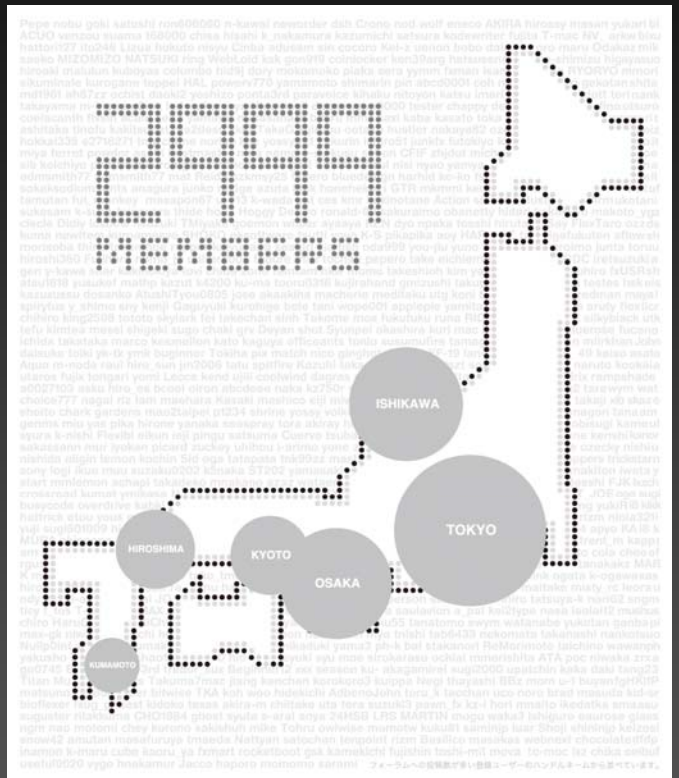
# 詳説 Flex 3 コンポーネント

Flex User Group

廣畑 大雅 (taiga.jp)
有川 榮一 (AKABANA)




## Flex / AIR の悩み相談

Flex User Group では、フォーラム※1 での技術情報交換をはじめ、全国各地での勉強会やハンズオンセミナー等の開催を行っています。最新の RIA 技術を採用した開発は、困難にブチ当たる事も多々あるかと思います。そんな時に Flex User Group をご活用下さい。経験豊富なメンバー※2 がきっと相手してくれますよ！また Adobe 公認の User Group ですから、RIA 最新情報もすぐに入手できるでしょう。一人で悩まず、みんなで楽しむ事が出来ればいいですよね〜。

※1 総トピック数：2438　総投稿数：10543　※2 総メンバー数：2999 人

FxUG
http://fxug.net/

Adobe USER GROUP

# 自己紹介

- 廣畑 大雅
  - HN: taiga
  - フリーランスの Web オーサリングエンジニア
  - taiga.jp
    http://taiga.jp/

- 有川 榮一
  - HN: arkw
  - The Seasar Project コミッタ (yui-frameworks)
  - AKABANA
    http://d.hatena.ne.jp/arkw/

# アジェンダ

- カスタムコンポーネント
- ビジュアルコンポーネント
- コンポーネントライフサイクル
- 使用注意 プロパティ / メソッド
- アイテムレンダラー

# カスタムコンポーネント

- 複合コンポーネント
- 標準コンポーネント拡張
- UIComponent / Container 拡張

# カスタムコンポーネント

- 複合コンポーネント
  - 例 : TextInput と Button を HBox でラップ

ExampleApp.mxml

```
<?xml version=“1.0” encoding=“utf-8”?>
<mx:Application xmlns:mx=…>
    …
    <mx:HBox>
        <mx:TextInput />
        <mx:Button />
    </mx:HBox>
    …
</mx:Application>
```

# カスタムコンポーネント

- 複合コンポーネント
  - 例 : TextInput と Button を HBox でラップ

ExampleApp.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx=…>
    …
    <mx:HBox>
        <mx:TextInput />
        <mx:Button />
    </mx:HBox>
    …
</mx:Hbox>
```

# カスタムコンポーネント

- 複合コンポーネント
  - 例 : TextInput と Button を HBox でラップ

MyComp.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:HBox xmlns:mx=…>
    <mx:TextInput />
    <mx:Button />
</mx:HBox>
```

# カスタムコンポーネント

- 複合コンポーネント
  - 例 : TextInput と Button を HBox でラップ

ExampleApp.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx=…>
    …
    <MyComp />
    …
</mx:Application>
```

# カスタムコンポーネント

- 標準コンポーネント拡張
  - 例 : Button のサブクラス (MXML)

MyButtonA.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Button xmlns:mx=…>
    …
    <mx:Script>…</mx:Script>
    …
</mx:Button>
```

# カスタムコンポーネント

- 標準コンポーネント拡張
  - 例 : Button のサブクラス (AS)

MyButtonB.as

```
package {
    import mx.controls.Button;
    public class MyButtonB extends Button {
        public function MyButtonB() {
            super();
        }
        …
    }
}
```

# カスタムコンポーネント

- 標準コンポーネント拡張
  - 例 : MXML と AS と使い方は同じ

```
<?xml version="1.0" encoding="utf-8"?>
< mx:Application xmlns:mx=…>

    <MyButtonA />

    <MyButtonB />

</mx:Application>
```

# カスタムコンポーネント

- UIComponent / Container 拡張
  - 例 : UIComponent のサブクラス (AS)

```
package {
    import mx.core.UIComponent;
    public class MyComponent extends UIComponent {
        public function MyComponent() {
            super();
        }
        …
    }
}
```

# ビジュアルコンポーネント

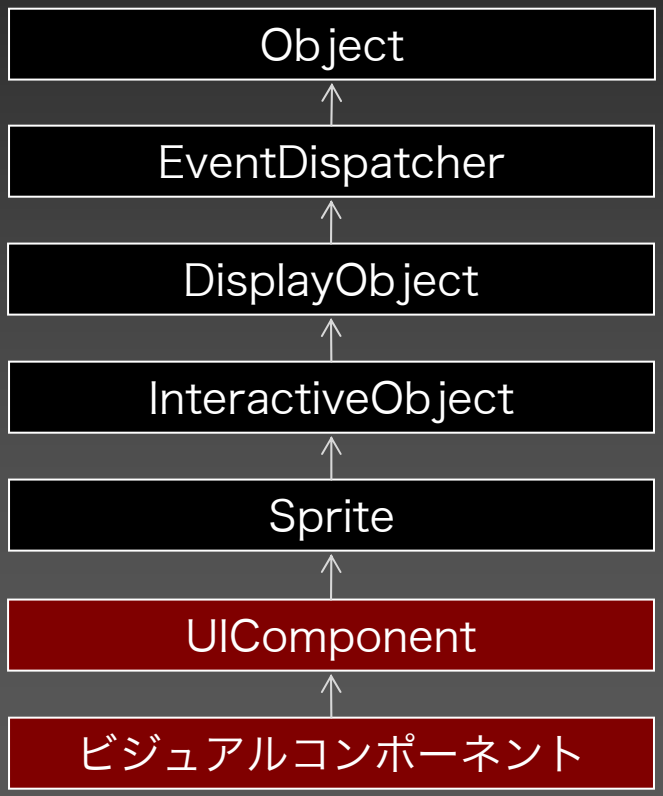
- 特徴
  - サイズ
    - 高さと幅など
  - イベント
    - MouseEvent, FlexEvent など
  - スタイル
    - フォントサイズ、レイアウトなど
  - ビヘイビア
    - マウスのクリックによるコンポーネントの移動やサイズ変更など
  - スキン
    - Button の upSkin, downSkin など

# ビジュアルコンポーネント

- 非ビジュアルコンポーネント
  - Formatter
    - DateFormatter, NumberFormatter, PhoneFormatter など
  - Validator
    - DateValidator, EmailValidator, RegExpValidator など
  - Effect
    - SetPropertyAction, SoundEffect, TweenEffect など

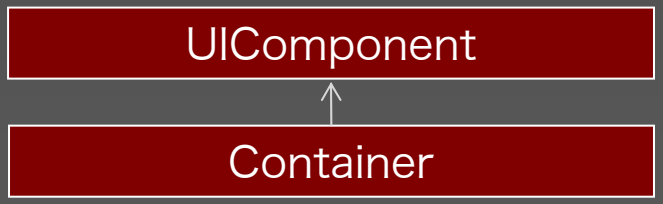
# ビジュアルコンポーネント

- UIComponent
  - すべてのビジュアルコンポーネントの基本クラス



# ビジュアルコンポーネント

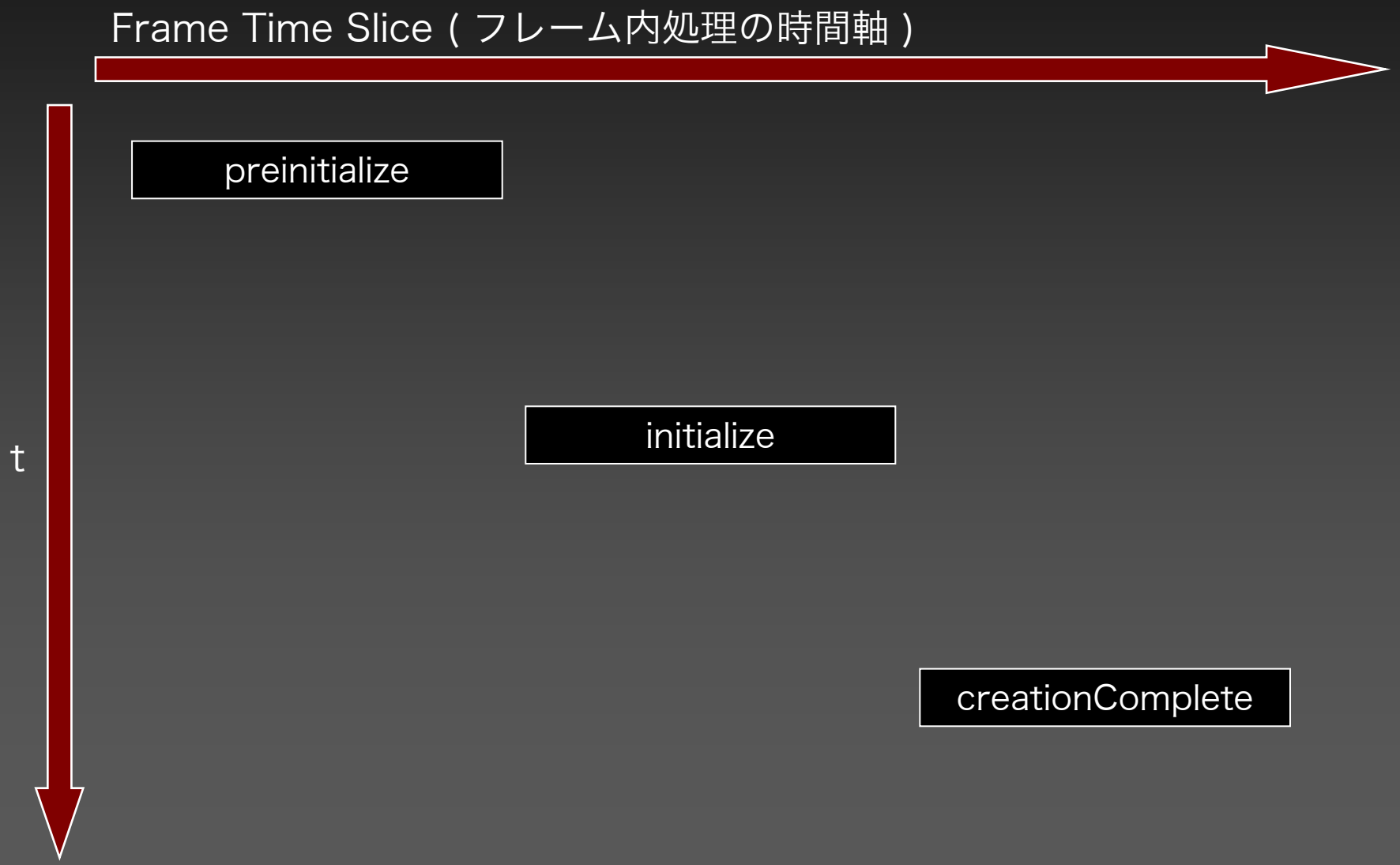
- Container
  - コンポーネントのレイアウト属性を制御できる階層構造を提供
  - スクロールの提供
  - すべての子のサイズと位置の設定、および複数の子コンテナ間のナビゲーションを制御
  - レイアウトとナビゲータの 2 種類



# コンポーネントライフサイクル

- 初期化イベント
  - preinitialize
    - コンポーネントが未加工の状態で作成されたときに送出
  - initialize
    - コンポーネントとそのすべての子が作成された後、コンポーネントのサイズが決定される前に送出
  - creationComplete
    - コンポーネントのレイアウトが完了し、必要に応じてコンポーネントが表示されたときに送出

# コンポーネントライフサイクル

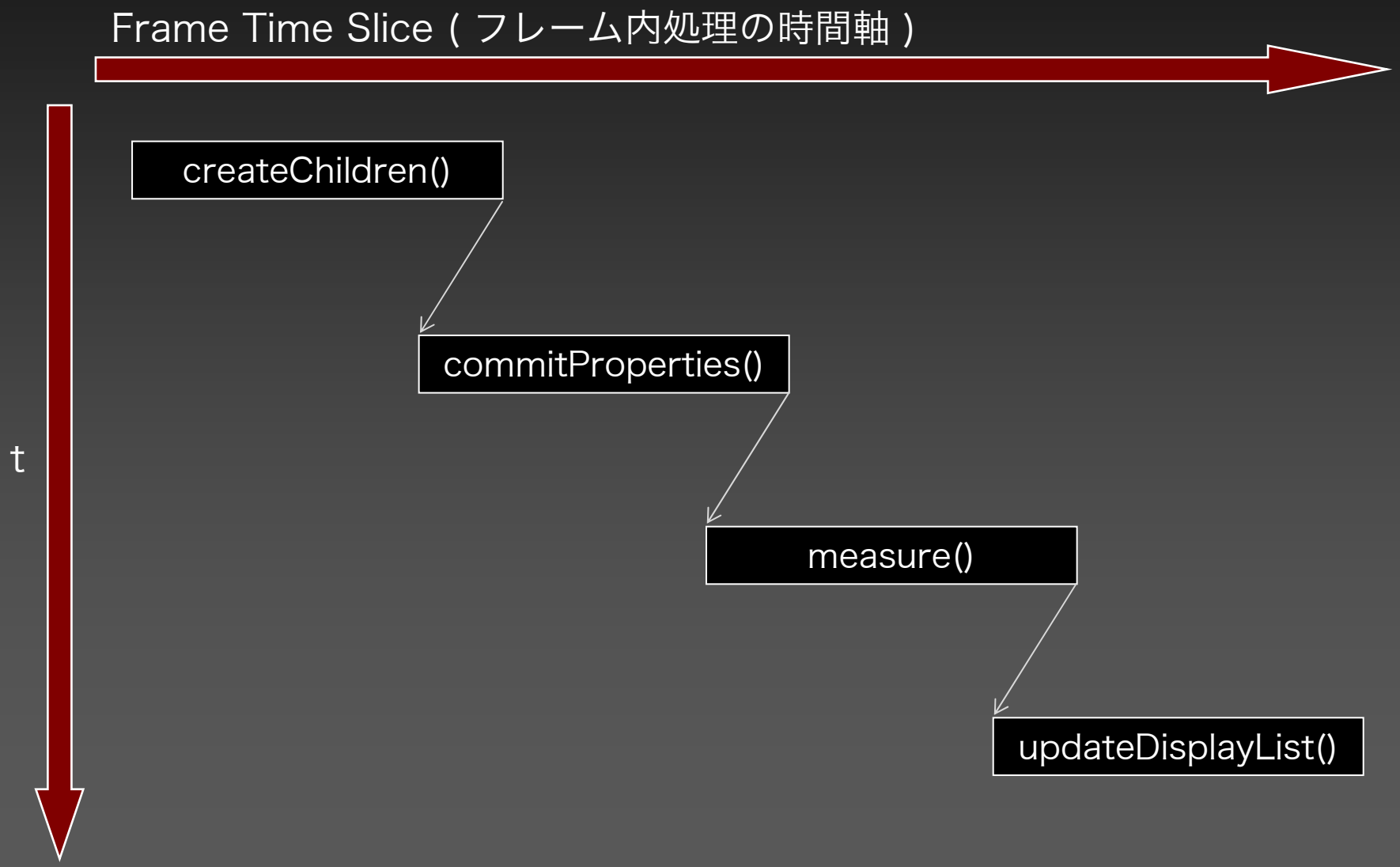


# 【デモ】Demo01_LifeCycleEvent

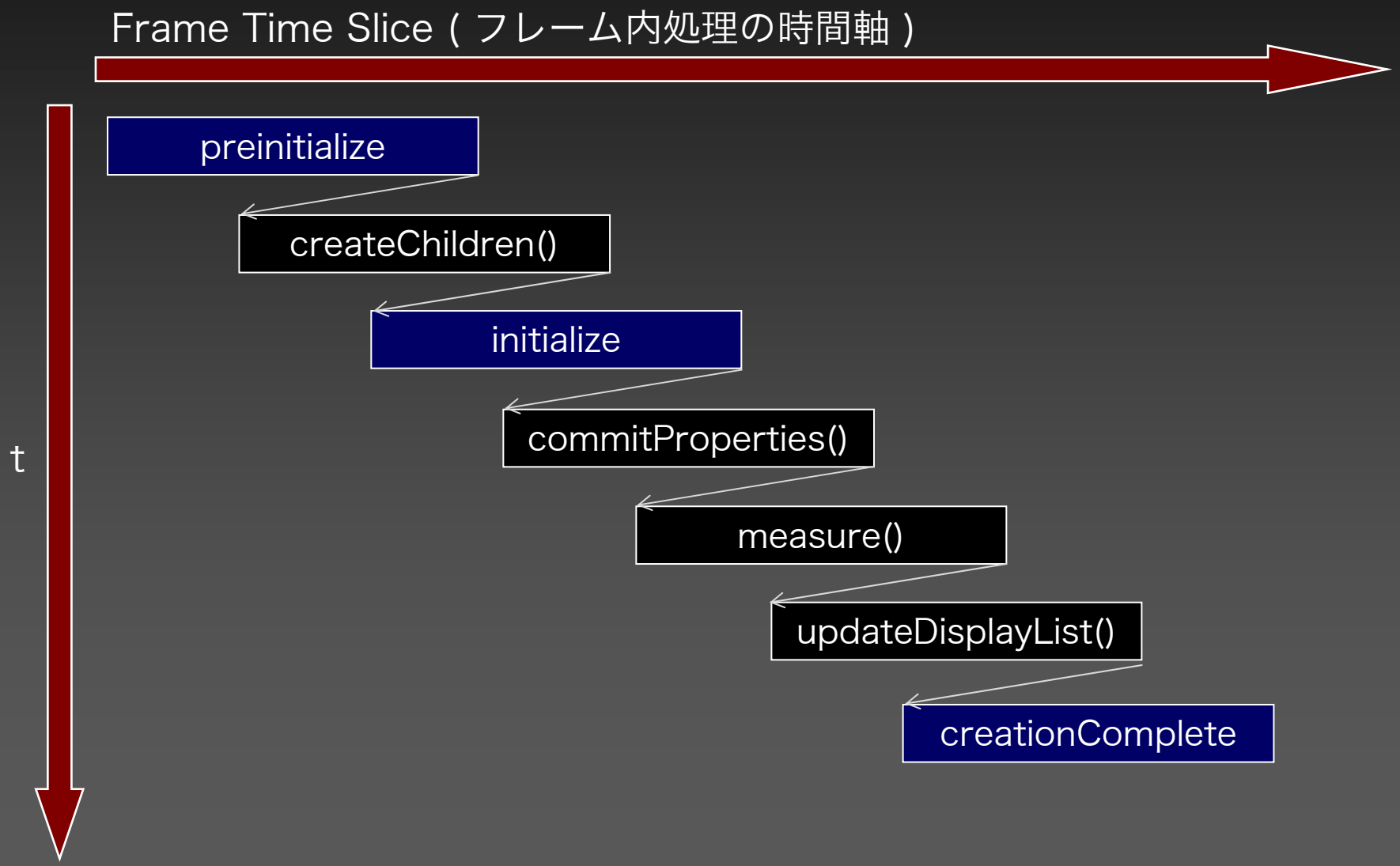
# コンポーネントライフサイクル

- ライフサイクルメソッド
  - createChildren()
    - コンポーネントに属する子コンポーネントを生成
  - commitProperties()
    - コンポーネント自身または、子コンポーネントのプロパティを決定
  - measure()
    - コンポーネントのサイズを決定
  - updateDisplayList()
    - 描画処理
    - 子コンポーネントのレイアウトを決定

# コンポーネントライフサイクル

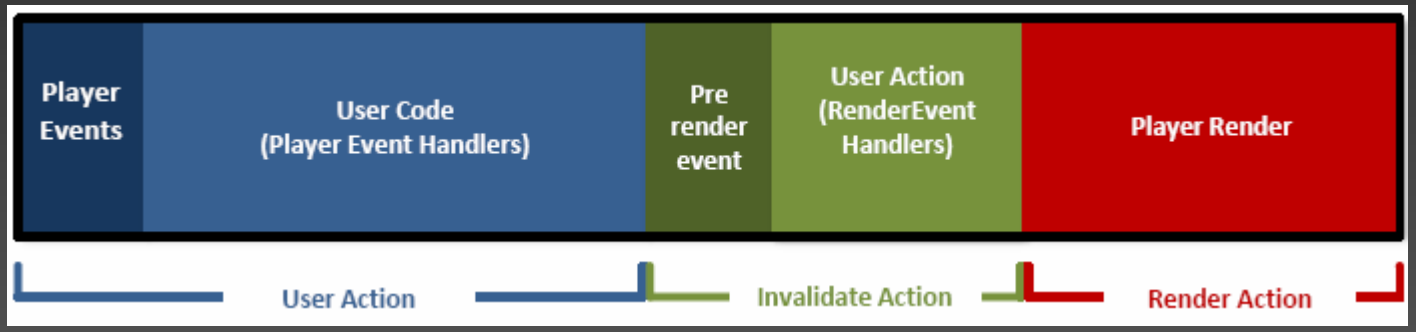


# コンポーネントライフサイクル



# コンポーネントライフサイクル

- AVM2 Marshalled Slice



# コンポーネントライフサイクル

- 無効化メソッド
  - プロパティ更新
    - invalidateProperties() → commitProperties()
  - サイズ更新
    - invalidateSize() → measure()
  - 描画更新
    - invalidateDisplayList() → updateDisplayList()

# コンポーネントライフサイクル

- 検証メソッド
  - プロパティ更新
    - validateProperties() → commitProperties()
  - サイズ更新
    - validateSize() → measure()
  - 描画更新
    - validateDisplayList() → updateDisplayList()
  - プロパティ、レイアウト更新
    - validateNow()

# 【デモ】Demo2_Invalid

# 【デモ】Demo3_Measure

# 使用注意 プロパティ / メソッド

- 動かすプロパティ
  - x, y
    - MoveEvent

# 使用注意 プロパティ / メソッド

- 動かすメソッド
  - move()
    - x, y プロパティとの違い
    - MoveEvent

# 使用注意 プロパティ / メソッド

```
override public function set x(value:Number):void {

    if (super.x == value)
        return;

    super.x = value;

    invalidateProperties();

    dispatchEvent(new Event("xChanged"));
}
```

# 使用注意 プロパティ / メソッド

```
public function move(x:Number, y:Number):void {

    var changed:Boolean = false;

    if (x != super.x) {
        super.x = x;
        dispatchEvent(new Event("xChanged"));
        changed = true;
    }
    …
    if (changed)
        dispatchMoveEvent();
}
```

# 使用注意 プロパティ / メソッド

- サイズを変えるプロパティ
  - width, height
    - ResizeEvent
  - explicitWidth, explicitHeight
    - ResizeEvent
  - measuredWidth, measuredHeight

# 使用注意 プロパティ / メソッド

- サイズを変えるメソッド
  - setActualSize()
    - width, height プロパティとの違い
    - ResizeEvent
  - scaleX, scaleY
    - DisplayObject.scaleX, DisplayObject.scaleY と UIComponent.scaleX, UIComponent.scaleY の違い
    - ResizeEvent

# 使用注意 プロパティ / メソッド

```
override public function set width(value:Number):void {
    if (explicitWidth != value) {
        explicitWidth = value;
        invalidateSize();
    }
    if (_width != value) {
        invalidateProperties();
        invalidateDisplayList();
        var p:IInvalidating = parent as IInvalidating;
        if (p && includeInLayout) {
            p.invalidateSize();
            p.invalidateDisplayList();
        }
        _width = value;
        dispatchEvent(new Event("widthChanged"));
    }
}
```

# 使用注意 プロパティ / メソッド

```
public function setActualSize(w:Number, h:Number):void {
    var changed:Boolean = false;
    if (_width != w) {
        _width = w;
        dispatchEvent(new Event("widthChanged"));
        changed = true;
    }
    if (_height != h) {
        _height = h;
        dispatchEvent(new Event("heightChanged"));
        changed = true;
    }
    if (changed) {
        invalidateDisplayList();
        dispatchResizeEvent();
    }
}
```

# 使用注意 プロパティ / メソッド

```
override public function set scaleX(value:Number):void {
    if (_scaleX == value)
        return;

    _scaleX = value;

    invalidateProperties();
    invalidateSize();

    dispatchEvent(new Event("scaleXChanged"));
}
```

# 使用注意 プロパティ / メソッド

```
protected function commitProperties():void {
    if (_scaleX != oldScaleX) {

        var scalingFactorX:Number = Math.abs(_scaleX / oldScaleX);

        if (!isNaN(explicitMinWidth))
            explicitMinWidth *= scalingFactorX;

        if (!isNaN(explicitWidth))
            explicitWidth *= scalingFactorX;

        if (!isNaN(explicitMaxWidth))
            explicitMaxWidth *= scalingFactorX;

        _width *= scalingFactorX;

        super.scaleX = oldScaleX = _scaleX;
    }
```

# 【デモ】Demo4_ResizeSample

# 【デモ】Demo5_AdjustPanel

# 使用注意 プロパティ / メソッド

- メソッド記述箇所の注意
  - 例 : updateDisplayList() 無限ループ

# 【デモ】Demo6_LoopUpdate

# アイテムレンダラー

- IDataRenderer
  - data プロパティ
- IListItemRenderer
  - listData
- 実装例
  - Canvas

# 【デモ】Demo7_ItemRenderer

# 続きは FxUG 勉強会で

- 2 月 7 日 ( 土 ) 15:00 ～
  Flex3 勉強会 第 62 回@京都 ( 参加受付中 )

- 2 月 19 日 ( 木 )
  Flex3 勉強会 第 64 回@東京 ( 予定 )

- 2 月 21 日 ( 土 )
  Flex3 勉強会 第 60 回@北陸 ( 金沢 )

- 2 月 28 日 ( 土 ) 13:00 ～
  まいど！関西 RIA セミナー
  ( 第 63 回 Flex3/AIR 勉強会@大阪 )

# Q & A