2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

JavaFX - Lessons Learned?

Ein Vergleich mit WPF(X)

@phdoerfler

imbus AG

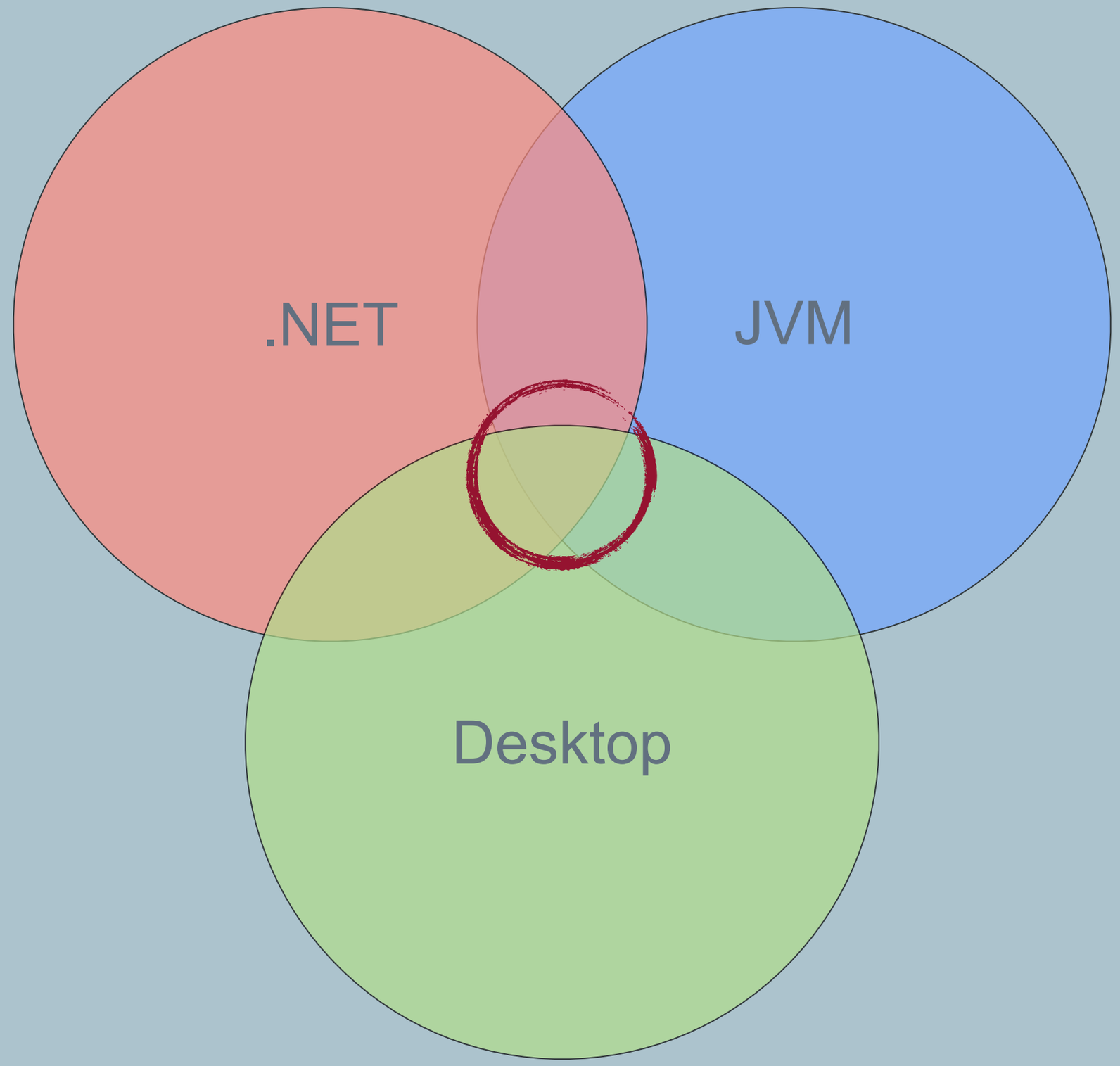
SPEISEKARTE

- Vorspeiße
  - Motivation
- Hauptgang
  - Featurebuffet
  - Scenegraph
  - Properties
  - XML
  - Expression Binding
- Nachtisch
  - Demo
  - Fazit

$: WHOAMI

- Philipp Dörfler
- Medieninformatik an der „Technischen Hochschule Nürnberg - Georg Simon Ohm“
- ~5 Jahre zu viel Enterprise
- Scala
- JavaFX
- sbt-fxml
- Twitter, etc.

WHO CARES?



AND THE

WINNER IS...

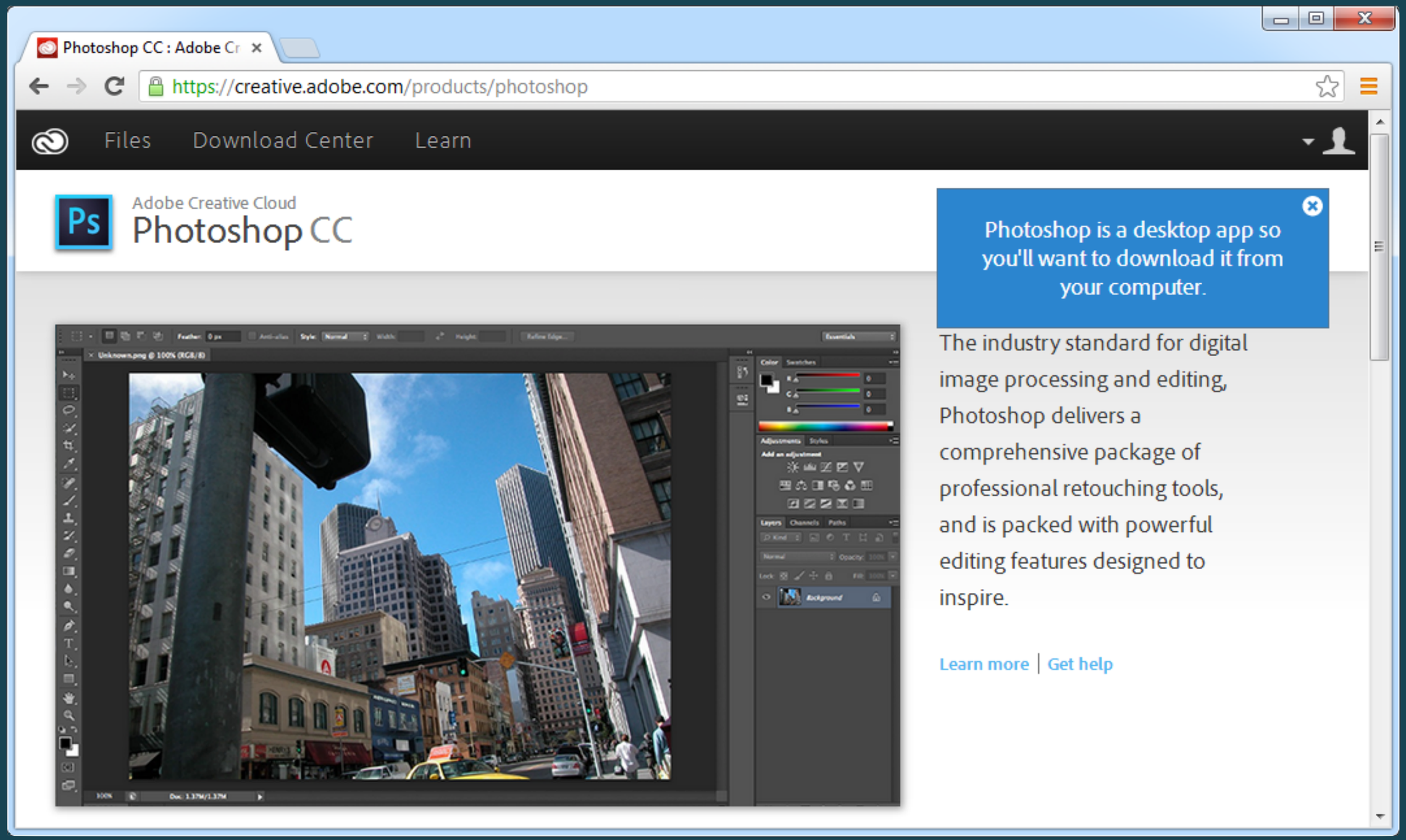
WEB!

- Cross-Platform
- Kaum Deployment-Aufwand
- Trivial
- Wie Desktop-Apps
- Ajax! Echtzeit!
- CSS
- Extensions!

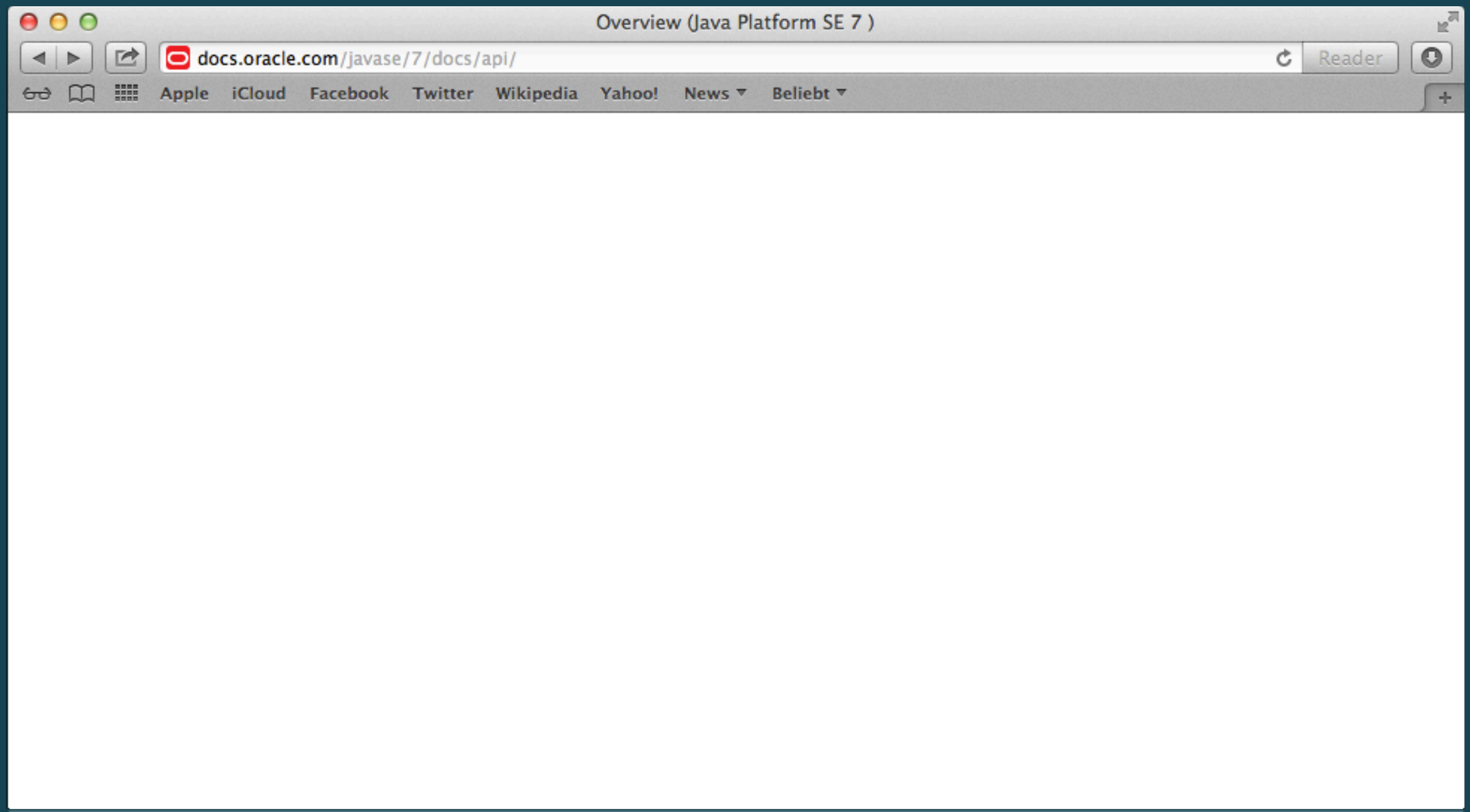
HTML

EXCEPT...

Chrome Desktop



Back vs. Javadoc



´NUFF SA(I)D

FX & WPF

Yay!

EIN KLEINER EXKURS

AWT	1995	
Swing	1997	
SWT	2001	
	2002	Windows Forms
F3	2006	WPF 3.0
	2007	WPF 3.5
JavaFX 1.0	2008	WPF 3.5sp1
	2010	WPF 4.0
JavaFX 2.0	2011	
JavaFX 2.2	2012	WPF 4.5

JAVAFX 8

- Java 8
- Lecker Lambdas
- Printing
- 3D
- Modena
- Native Font Rendering
- High-DPI

JAVAFX 8 *II*

- Default Classpath
- Gradle Build
- SwingNode
- TreeTable
- WebView Performance
- Besseres Multithreading
- ...

GEMEINSAMES

GEMEINSAMES

- Legacy-Ablöser
  - WinForms
  - Swing
- Deklarativ
- Reaktiv
- Properties
- Binding

GEMEINSAMES *II*

- Scene-Graph
  - Hoher Abstraktionsgrad
  - Einheitliche API
- Retained Mode
- Animationen
- Styling
- GUI <|> Daten

GEMEINSAMES *III*

- XML für GUIs
  - XAML
  - FXML
    - Typkonvertierung
- Graphische GUI-Designer
  - Visual Studio
  - Blend
  - Scene Builder

GEMEINSAMES

- Multithreading
  - Application Thread
  - Rendering Thread
  - Media Thread
- Multimedia
- HW-Beschleunigung
  - Direct3D
  - OpenGL

GEMEINSAMES

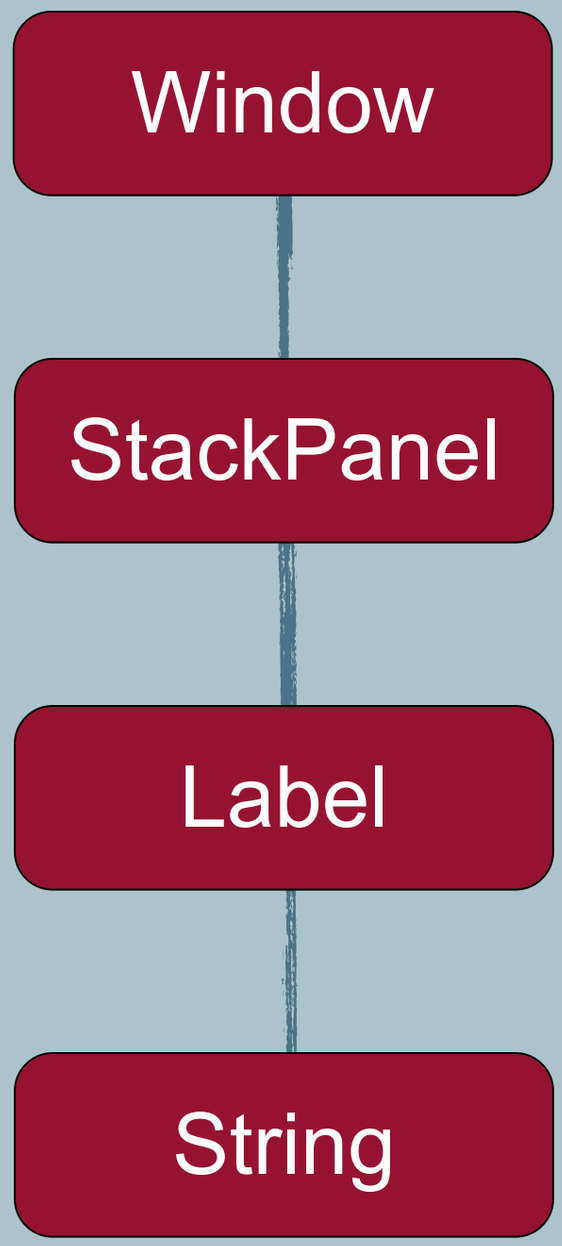
- Accessability
- Effekte
- Multitouch
- Event-Handling
- Browser
  - XAML Browser Applications (XBAP)
  - Silverlight
  - Applets

SCENE GRAPH

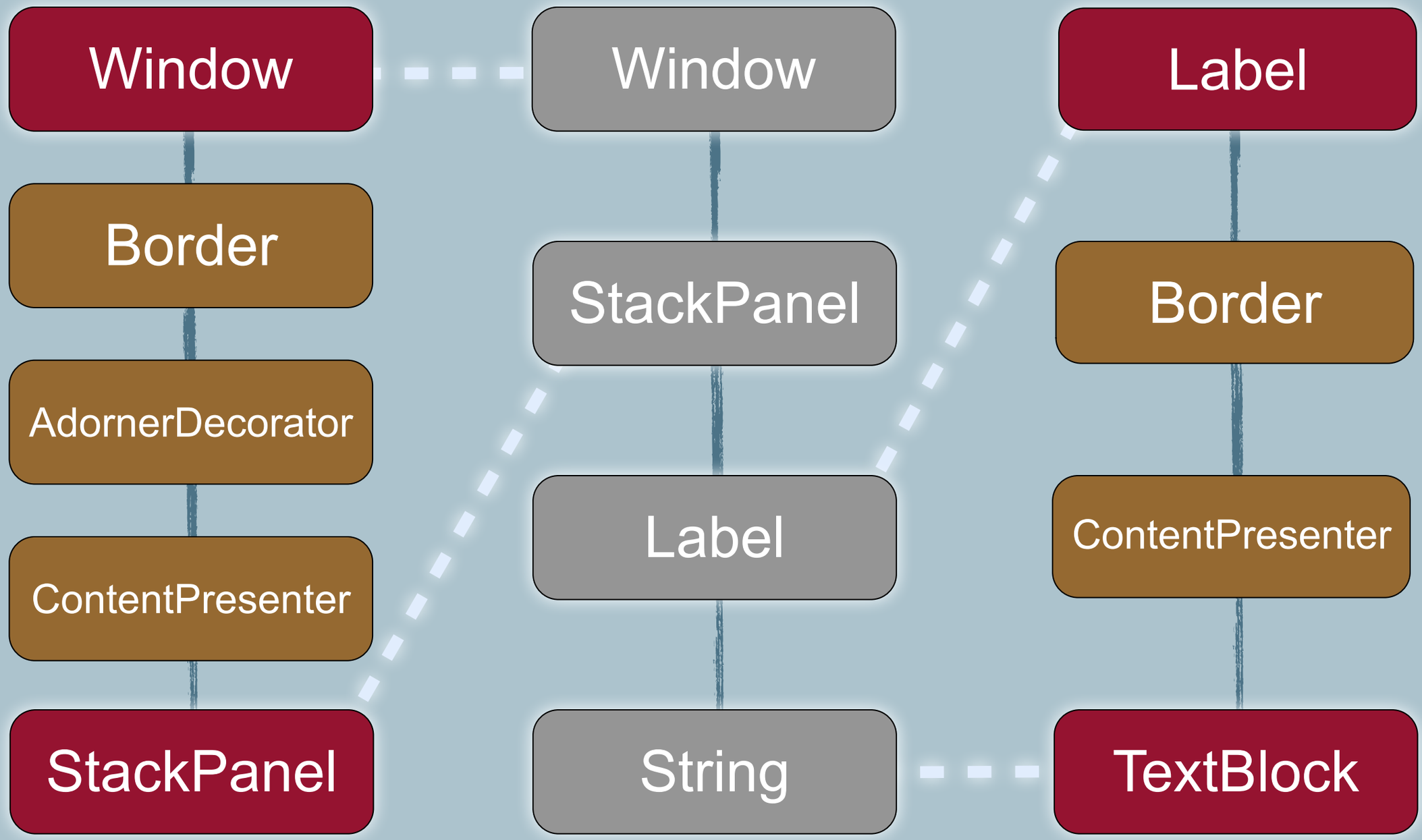
SCENE GRAPH *WPF*

- Logische Bäume *Templates*
  - Deklarierter Scenegraph
  - Vererbungslogik von Properties
- Visueller Baum
  - Rendering
  - Event routing
  - Keine `ContentElement`
  - Nur `Visual` / `DependencyObject`

LOGISCHE BÄUME



UN-LOGISCHE BÄUME



PFADFINDER

- Skin
- SkinBase (FX 8)

FX

- LogicalTreeHelper
- VisualTreeHelper

WPF

PROPERTIES

PROPERTIES

- .NET CLR Properties

```
private int x;
public int X {
   get { return x; }
   set { x = value; }
}
```

- Java Properties *Beans!*

```
private int x;
public int getX() { return x; }
public int setX(int value) { x = value; }
```

zzzZZZ

PROPERTIES

Callbacks

onActionPerformed

Callback Hell

list.addAll(...)

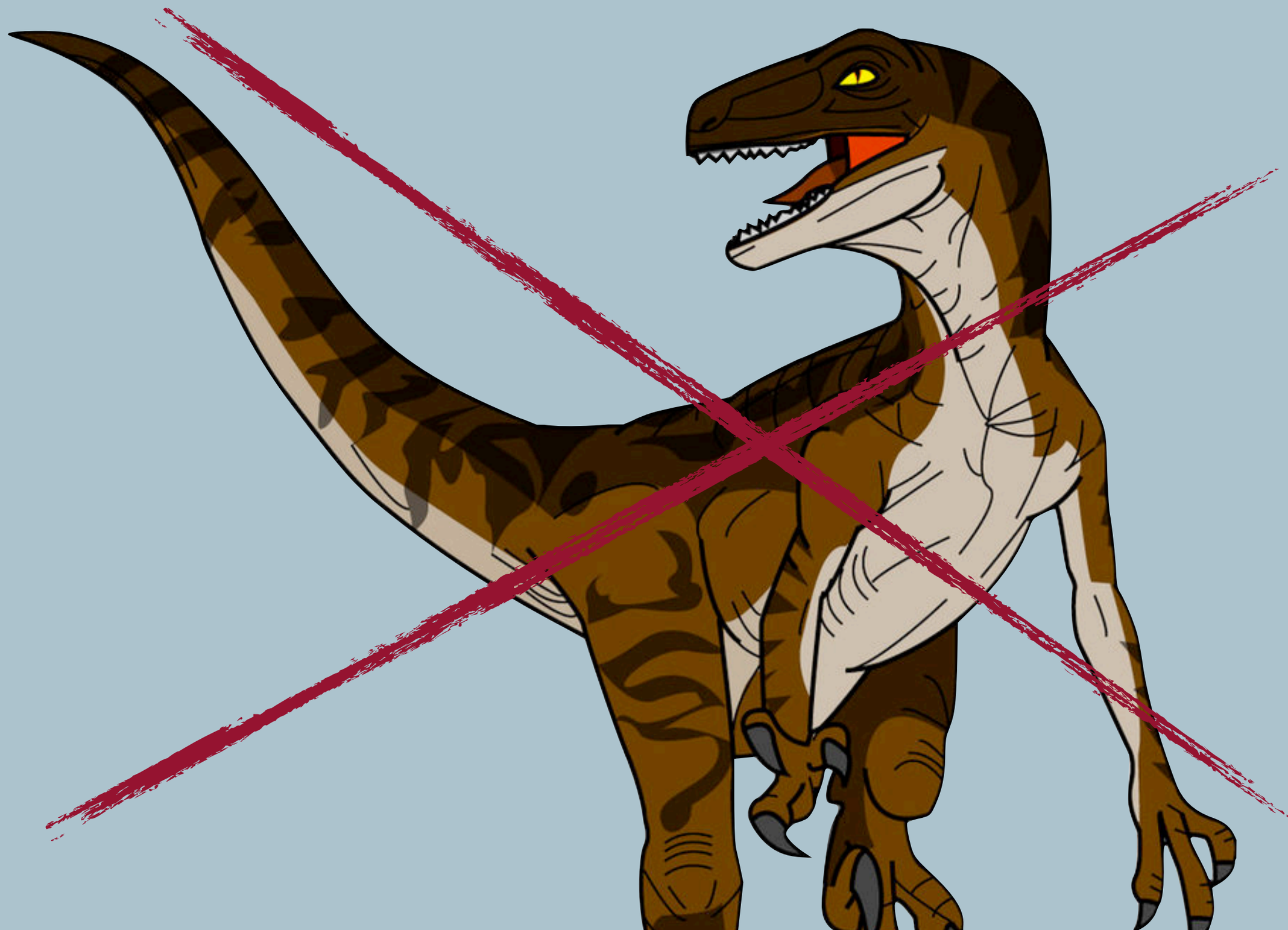
list.remove(model.
getById(42))

PROPERTIES

ScalaFX

```
button.text <== when(ponyAvailable) "Satteln"
                otherwise "Rufen"
```

CALLBACKS?



PROPERTIES *Done Right*

- .NET / WPF
  - Properties
  - **Dependency Properties**
  - **Attached Properties**
- Java / FX
  - Beans
  - **Properties**

FEATURES!

- Change Notification
- Value Validation
- Binding
- Animationen
- Value Inheritance
- Styles

WPF

Dependency-
PROPERTIES IN WPF

```
public static readonly DependencyProperty PoniesAwakeProperty =
DependencyProperty.Register(„PoniesAwake“, typeof(int),
typeof(MainWindow));
```

Owner Type

Property Type

```
public int PoniesAwake
{
    get { return (int)GetValue(PoniesAwakeProperty); }
    set { SetValue(PoniesAwakeProperty, value); }
}

PoniesAwake = 23;
```

```
Button myButton = new Button();
myButton.Width = 200.0;
```

Verwendung

```
<Button Width=“200“ />
```

Dependency-PROPERTIES IN WPF *II*

```
public static readonly DependencyProperty IsPonyProperty =
DependencyProperty.Register(„IsPony“, typeof(bool),
typeof(MainWindow), new FrameworkPropertyMetadata((bool) false,
new PropertyChangedCallback(OnIsPonyChanged)));

public bool IsPony
{
    get { return (bool)GetValue(IsPonyProperty); }
    set { SetValue(IsPonyProperty, value); }
}

private static void OnIsPonyChanged(DependencyObject d,
DependencyPropertyChangedEventArgs e)
{
    MainWindow target = (MainWindow) d;
    bool oldIsPony = (bool)e.OldValue;
    bool newIsPony = target.IsPony;
    target.OnIsPonyChanged(oldIsPony, newIsPony);
}
```

Dependency-PROPERTIES IN WPF *III*

- Metadaten
- Default-Wert
- Callbacks
- Render-Infos
  - AffectsMeasure (sorgt für Größenberechnung)
  - …
- Vererbungs-Infos

DependencyProperty.Register

Attached-
PROPERTIES IN WPF
XAML

```
DependencyObject dep = new DependencyObject();
dep.SetValue(Canvas.TopProperty, 42);

double top;
top = (double) dep.GetValue(Canvas.TopProperty);
```

```
<DockPanel>
    <CheckBox DockPanel.Dock="Top">Hello</CheckBox>
</DockPanel>
```

DependencyProperty.RegisterAttached

PROPERTIES IN WPF

- Globale Property-Tabelle
- Konventionen
  - CLR-Properties
  - Außer bei AttachedProperty
    - GetFoo
    - SetFoo
- Eng mit GUI verzahnt

PROPERTIES IN WPF *II*

- Viel Syntax
- Microsofts Lösung:
  - Snippet in Visual Studio!
- Untypisiert
- Sehr flexibel
- Zu flexibel?
  - Ändern des Scenegraphs über Listener in **`AttachedProperty`**s

PROPERTIES IN FX

```
private DoubleProperty amountDue = new SimpleDoubleProperty();
public final double getAmountDue() { return amountDue.get(); }
public final void setAmountDue(double value) { amountDue.set(value); }
public DoubleProperty amountDueProperty() { return amountDue; }

amountDueProperty().addListener(new ChangeListener() {
    @Override public void changed(ObservableValue o, Object oldVal,
            Object newVal) {
        System.out.println(“yay“);
    }
});

setAmountDue(100.0);
```

PROPERTIES IN FX *II*

- `AttachedProperty`?
- Statischer Setter & Getter!

```
<GridPane>
    <Label text="My Label" GridPane.columnIndex=“0“ />
    ...
</GridPane>
```

```
Label label = new Label("My Label");
GridPane.setColumnIndex(label, 0);
```

```
class GridPane {
  public static void setColumnIndex(Node node, Integer value);
}
```

PROPERTIES IN FX *III*

- Lazy Evaluation
- Change-, InvalidationListener
- Typisiert
  - Generics
- Metadata
  - Annotations

PROPERTIES IN FX

- Unabhängig von der GUI
- Validierung
- Ebenfalls flexibel
- Viel pragmatischer
- Rein berechnete `Propertys` möglich

XML

FXML & XAML

- Serialisierung des logischen Objektgraphen
- Default properties
  - z.B. ListView.children
- Toolsupport
- Skripting-Support

FXML

```
<?xml version="1.0" encoding="UTF-8"?>
```

Obligatorisches XML-Zeug

```
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.paint.*?>
```

Imports als Prozessor-Anweisung

```
<StackPane id="rootPane" prefHeight="400.0" prefWidth="600.0"
xmlns:fx="http://javafx.com/fxml" fx:controller="org.foo.Conny">
    <Button fx:id="heyButton" text="Hey!"
            StackPane.alignment="CENTER_RIGHT" />
</StackPane>
```

Controller-Klasse

`fx:id` ← Variable im Controller

`id` ← CSS ID -> HTML ID

FXML *II*

- Attribute mit „fx:“
  `FXMLLoader`
- Attribute ohne „fx:“
  normale Properties
- Attribute mit „.“
  statische Properties

attached

XAML

Namespaces

```
<Window xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/
presentation”
        xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”
        x:Class="Org.Foo.Conny">
    <StackPanel>
        <Button Name="heyButton" Content="Hey!" />
    </StackPanel>
</Window>
```

Code Behind Klasse

Variable im Code Behind

XAML *II*

- Keine Prozessor-Anweisungen für Imports
- Wird über Namespaces abgebildet
- Mapping von CLR- nach XML-Namespaces nötig

EXPRESSION BINDING

FXML

```
<TextField fx:id="texty" text="some text" />
<Label text="${'This is ' + texty.text}" textFill="BLUE">
```

XAML*

```
<TextBox Name="Texty" Text="SPARTA"/>
<TextBlock Text="{Binding ElementName=Texty, Path=Text,
StringFormat='This is {0}'}"/>
```

** nicht bei WinRT*

„SIMPLE EXPRESSIONS“

„Only **simple expressions** that resolve to property values or page variables are currently supported. Support for more complex expressions involving **boolean** or other operators may be added **in the future**.“ ~ FXML Doc

„SIMPLE EXPRESSIONS“ II

```
text="${'this is some ' + textField.text}"
```

```
selected="${(a.selected &amp;&amp; b.selected) || c.selected}"
```

```
selected=“${! true && false}“
```

= TRUE

„SIMPLE EXPRESSIONS“ *III*

DO

NOT

RELY

ON

IT *yet*

FXML BEISPIEL

```
<?language javascript?>
```
JavaScript, Groovy, Clojure, JRuby, ...

```
...

<fx:script><![CDATA[
   function handleRectEnter(target) {
       target.setText(event.getSource().getText());
       target.getStyleClass().clear();
       target.getStyleClass().add('desc-focused');
   }
   function handleRectLeave(target) {
       target.getStyleClass().clear();
       target.getStyleClass().add('desc');
   }
]]></fx:script>

...

<Button onMouseClicked="java.lang.System.out.println('You clicked me!');"
onMouseEntered="handleRectEnter(quad1Text);" onMouseExited="handleRectLeave(quad1Text);" />

<Text fx:id="quad1Text" text="Action 1" />
```

DEKLARATIONEN

FXML

```
<fx:define>
  <Button fx:id="b" prefHeight="32.0" prefWidth="32.0"
          text="Button" />
</fx:define>
```

```
<Button maxHeight="${b.prefHeight}" />
```

DEKLARATIONEN

XAML

```
<Window.Resources>
  <SolidColorBrush x:Key="backgroundBrush">Yellow</SolidColorBrush>
</Window.Resources>
<Window.Background>
  <StaticResource ResourceKey="backgroundBrush" />
</Window.Background>

...

<Button Background="{StaticResource backgroundBrush}" />
```

PRESENTATION MODEL

PRESENTATION MODEL

„This example is a **proposal**, it won't work today. We are thinking about adding support for this in an upcoming update release." - Richard Bair, 14. Oktober 2011

VERBIEGBARKEIT

VERBIEGBARKEIT

```
<ListBox ItemSource="{Binding Ponies}">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <TextBox Text="{Binding Name}"/>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

DATA TEMPLATES

- In XAML als `Resource` deklarierbar
- DataType-Property: automatisch auf passende Typen angewandt
- `HierarchicalDataTemplate`

Fazit

WPF

- Tooling umfrangreicher
- Windows „only“
- Dafür natives L&F
- Over-engineered
- Property-System komplex
- Gutes Text-Rendering
- Attached- & RoutedEvents

JAVAFX

- Cross-Platform (!)
- Simplere API
- Offene Baustelle
- Tooling schlanker
- Leichter skinbar
- Open-Source
- RoutedEvents

LETZTE WORTE

- Kein Gewinner
- Beide gut
- Windows, .NET => WPF
- Mac, Linux, JVM => JavaFX

2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

@phdoerfler
imbus AG

DISCLAIMER

- Pony Creator
  - http://generalzoi.deviantart.com/art/Pony-Creator-Full-Version-254295904
- Veloziraptor
  - http://www.deviantart.com/art/Velociraptor-164135011
- HTML Logo
  - http://www.w3.org/html/logo/