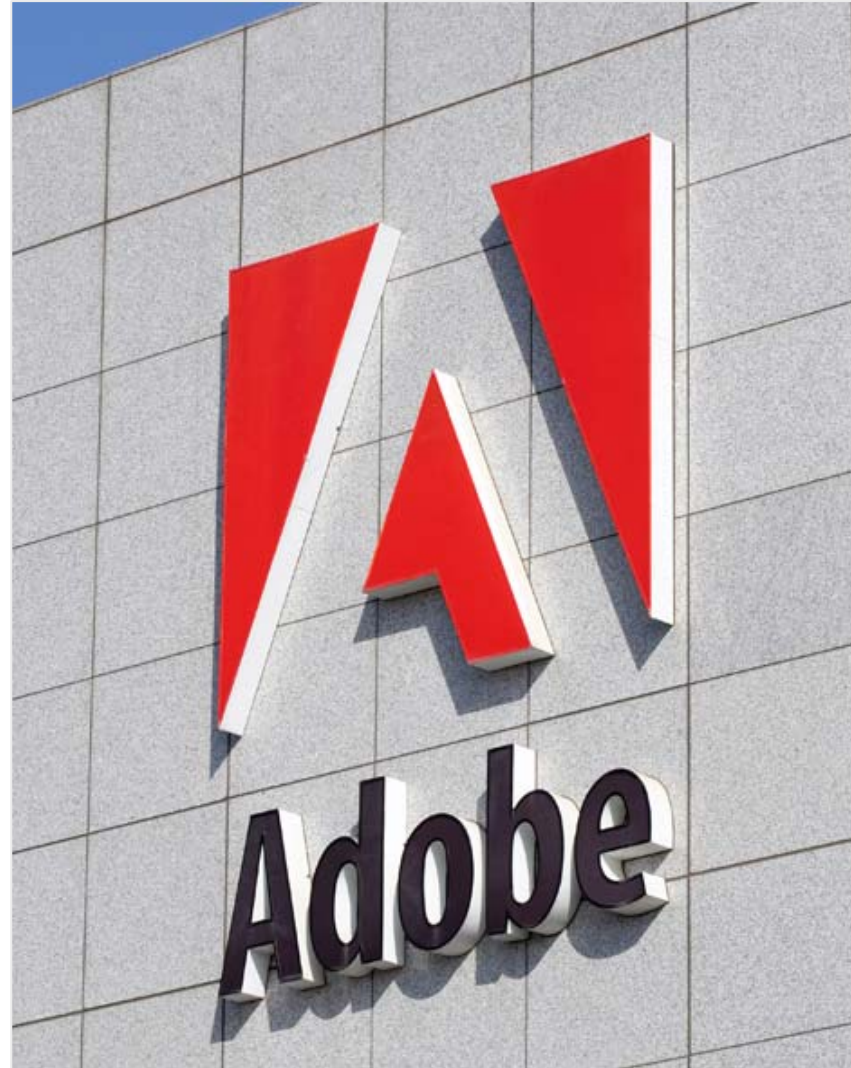


# Flex for Flash Developers

Sho Kuwamoto

Sr. Director, Engineering

Adobe Systems, Inc.



# What is Flex?

- Tag/script-based app development for Flash
- ActionScript class library
  - Layout
  - Effects, skins, etc.
  - Networking and data
- Compiler, debugger, and other tools

The screenshot shows a web application titled "THE GENUINE MOTOR ACCESSORIES CUSTOMIZER". The main content area displays a motorcycle (2006 XL883C Sportster 883 Custom) with the color "Fire Red Pearl/Vivid Black". Below the motorcycle is an "ACCESSORY SELECTOR" section with a "Decorative" category selected, showing various accessories like "Gauge Visor Ring". To the right, there are sections for "ACCESSORIES ON THIS BIKE", "ACCESSORY INFORMATION", "IMPORTANT INFORMATION", and "OTHER PARTS IN COLLECTION".

Below the customization interface is a form titled "1. Shipping Address" with the following fields:

- First Name
- Last Name
- Address (two lines)
- City
- Phone
- State (dropdown menu showing "AK")
- Zip Code

Buttons for "INSTALL ON BIKE" and "ADD TO WORKBOOK" are visible. At the bottom of the form is a "Continue" button. Below the shipping address form are sections for "2. Billing Address", "3. Credit Card Information", and "4. Submit Order".

# HTML vs Flex vs Flash

## HTML

- HTML / CSS files
- Includes
- JavaScript libraries
- Tag-based
- HTML output

## Flex

- format
- reuse
- behavior
- metaphor
- output

## Flash

- FLA files
- Symbols
- ActionScript classes
- Timeline-based
- SWF output

**Unlike HTML, Flex was designed for application development  
(no more creating navbars by slicing up images!)**

**Let's see some code**

# Our first application – Step1.mxml

```
<Application>  
  <Panel>  
    <Label text="Hello, world!" />  
    <Button label="goodbye" />  
  </Panel>  
</Application>
```

# Adding layout – Step2.mxml

```
<Application>  
  <Panel layout="absolute" width="80%" height="80%">  
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>  
    <Button label="goodbye" right="30" bottom="41"/>  
  </Panel>  
</Application>
```

# Adding styles – Step3.mxml

```
<Application>
```

```
  <Style>
```

```
    TextArea {
```

```
      font-size: 36px;
```

```
      font-weight: bold;
```

```
    }
```

```
  </Style>
```

```
  <Panel layout="absolute" width="80%" height="80%">
```

```
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
```

```
    <Button label="goodbye" right="30" bottom="41"/>
```

```
  </Panel>
```

```
</Application>
```

# Adding behavior – Step4a.mxml

```
<Application>
  <Style source="style.css"/>

  <Panel id="myPanel" layout="absolute" width="80%" height="80%">
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
    <Button label="goodbye" right="30" bottom="41"
      click="myPanel.visible = false"/>
  </Panel>
</Application>
```

# Adding behavior – Step4b.mxml

```
<Application>
  <Style source="style.css"/>
  <Script>
    public function goodbye() : void {
      myPanel.visible = false;
    }
  </Script>

  <Panel id="myPanel" layout="absolute" width="80%" height="80%">
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
    <Button label="goodbye" right="30" bottom="41" click="goodbye()"/>
  </Panel>
</Application>
```

# Adding behavior – Step4c.mxml

```
<Application initialize="doInit()">  
  <Style source="style.css"/>  
  <Script source="Step4c_code.as"/>  
  
  <Panel id="myPanel" layout="absolute" width="80%" height="80%">  
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>  
    <Button id="goodbyeButton" label="goodbye" right="30" bottom="41"/>  
  </Panel>  
</Application>
```

# Adding behavior – Step4c\_code.as

```
public function doInit() : void  
{  
    goodbyeButton.addEventListener("click", goodbye);  
}  
  
public function goodbye(event: Event) : void  
{  
    myPanel.visible = false;  
}
```

# Adding states – Step5.mxml

```
<Application initialize="doInit()">
```

```
  <states>
```

```
    <State name="small">
```

```
      <SetProperty target="{myPanel}" property="height" value="200"/>
```

```
      <SetProperty target="{myPanel}" property="width" value="300"/>
```

```
    </State>
```

```
  </states>
```

```
<Style source="style.css"/>
```

```
<Script source="Step5_code.as"/>
```

```
<Panel id="myPanel" layout="absolute" width="80%" height="80%">
```

```
  <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
```

```
  <Button id="goodbyeButton" label="goodbye" right="30" bottom="41"/>
```

```
  <Button id="smallButton" label="small" left="10" bottom="41"/>
```

```
</Panel>
```

```
</Application>
```

# Adding states – Step5\_code.as

```
public function doInit() : void  
{  
    goodbyeButton.addEventListener("click", goodbye);  
    smallButton.addEventListener("click", makeSmall);  
}
```

```
public function goodbye(event: Event) : void  
{  
    myPanel.visible = false;  
}
```

```
public function makeSmall(event: Event) : void  
{  
    currentState = "small";  
}
```

# Adding effects – Step6.mxml

```
<Application>
  <states>
    <State name="small">
      <SetProperty target="{myPanel}" property="height" value="200"/>
      <SetProperty target="{myPanel}" property="width" value="300"/>
    </State>
  </states>

  <Style source="style.css" />
  <Script source="Step6_code.as" />

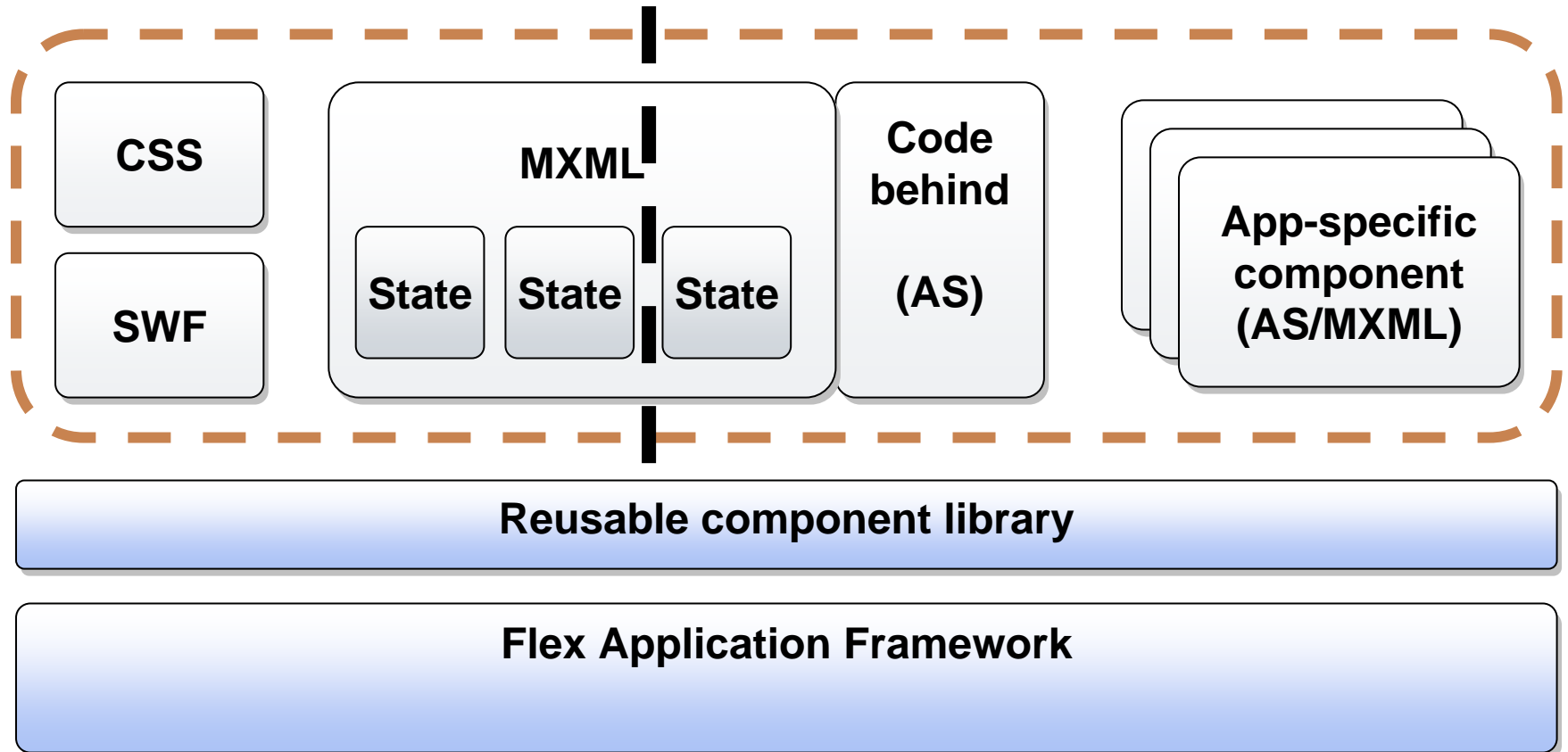
  <Panel id="myPanel" layout="absolute" width="80%" height="80%"
    hideEffect="Fade" resizeEffect="Resize">
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
    <Button id="goodbyeButton" label="goodbye" right="30" bottom="41"/>
    <Button id="smallButton" label="small" left="10" bottom="41"/>
  </Panel>
</Application>
```

**Deeper dives**

# Application structure

## Designer

## Developer



# Designers and developers

Designer

## CSS/SWF

- Responsible for skinning and styling through CSS and SWF

- Does initial layout. May do final layout (via tool)
- Does effect and transition design.

Developer

## AS

- Responsible for program logic and component development

- May be responsible for final layout (via containers)
- May implement effects and transitions
- Connects to data and services

## MXML

# Keys to des/dev workflow

- Separate out code into “code behind” pages
  - Use `<Script source="xxx">` or other methods to separate the ActionScript code
  - Add event handlers from ActionScript, not MXML
  - Try not to write fancy data binding code in MXML
- Do not create visual components programmatically unless absolutely necessary
- Do as much styling in CSS as possible
- Use states when possible to express changes to user interface

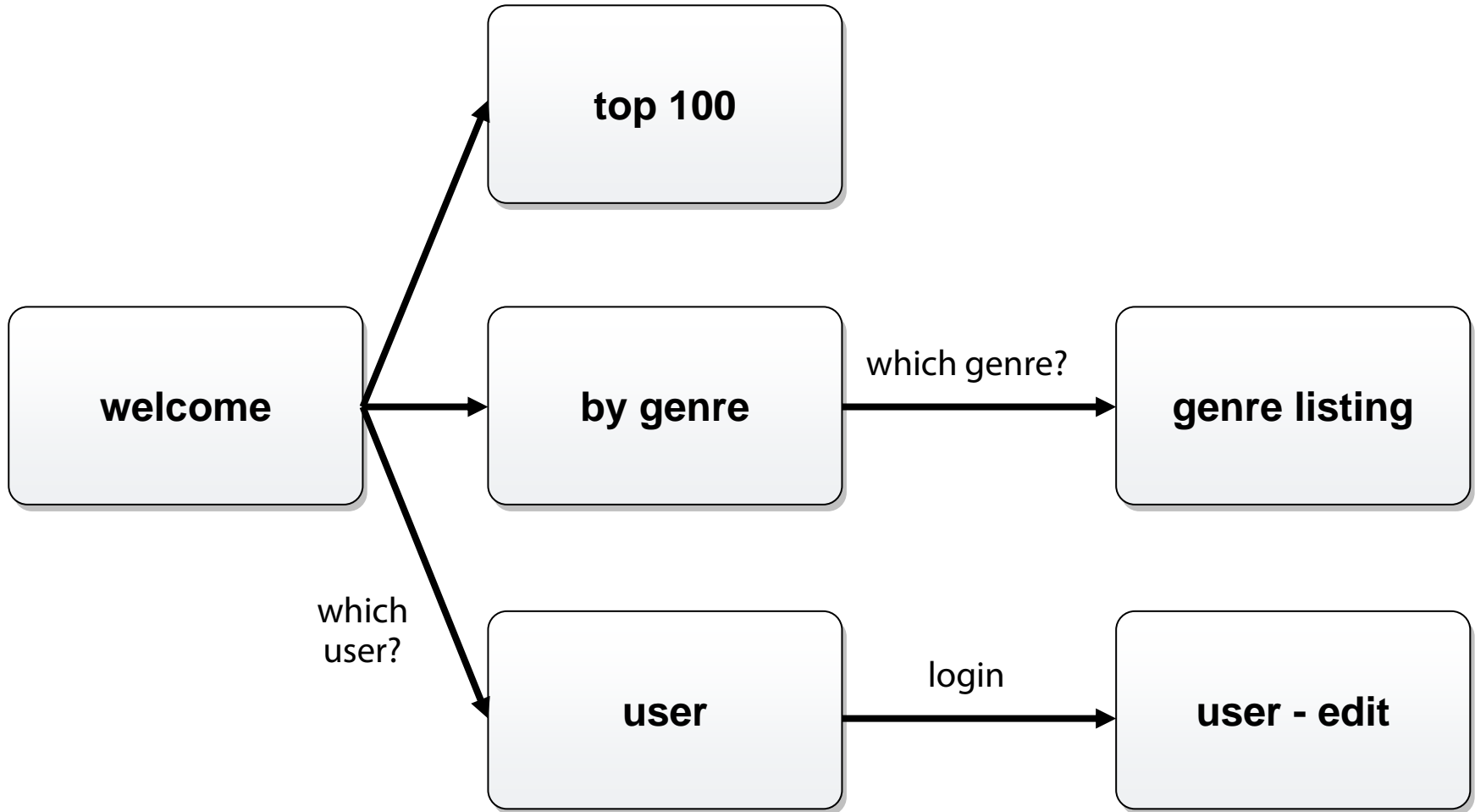
# Skinning and styling

- SWF files or GIF/PNG can be used for skinning
  - SWF attached to controls via CSS
  - Scale 9 can be done in SWF or in CSS
- CSS gives tons of control for the look of Flex controls

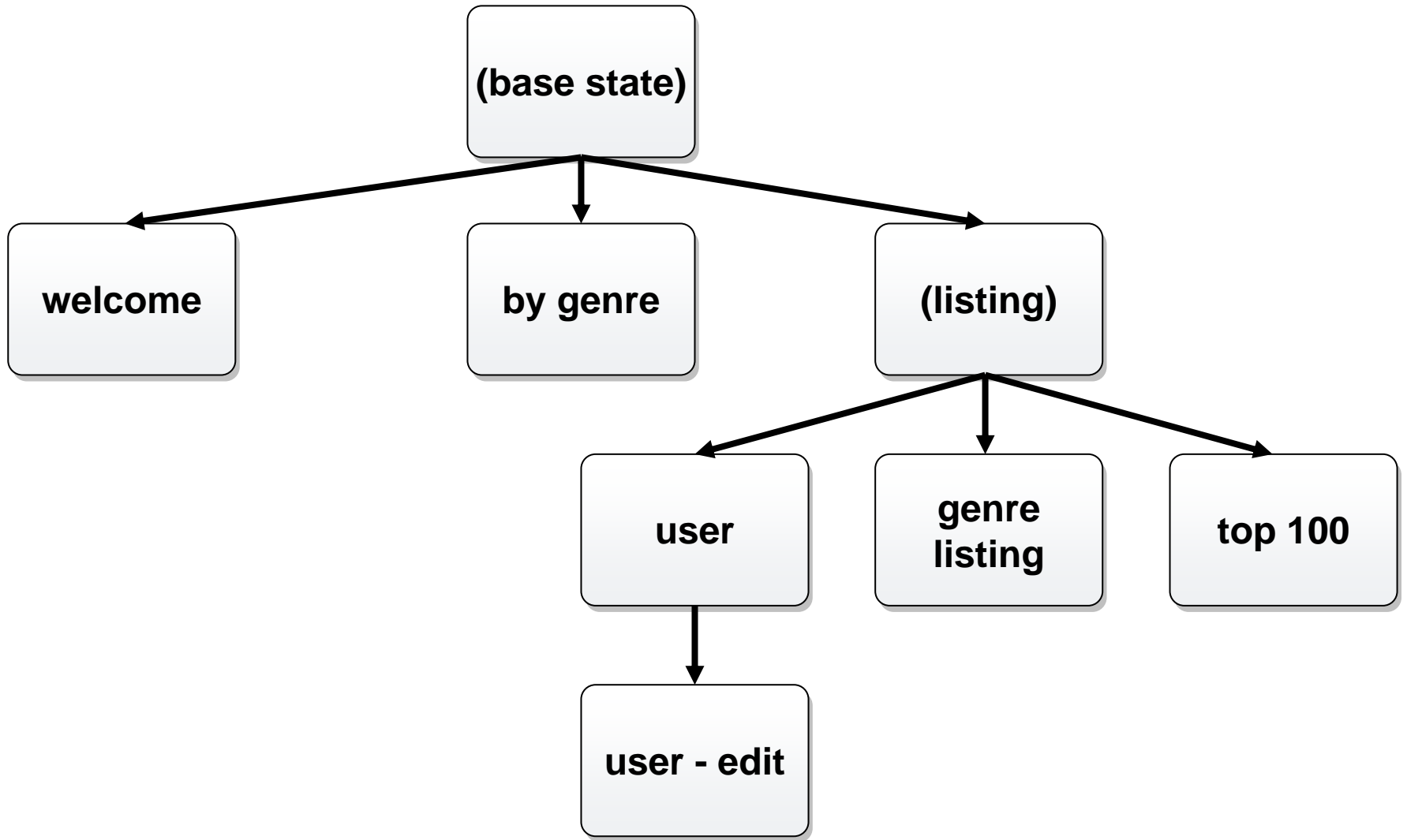
<http://weblogs.macromedia.com/mc/archives/FlexStyleExplorer.html>

- CSS styles can be defined per tag, or per class

# Working with states – music app



# How are the states related?



# When do effects run?

- When a specific trigger fires  
e.g., show, hide, and resize
- When a transition from one state to another occurs  
`<Transition fromState="xxx" toState="yyy">`  
    `<Effect />`  
`</Transition>`
- When called explicitly through ActionScript  
`myEffect.play();`

# Defining your own effects

- Using parallel and sequence

**<Parallel>**

**<Dissolve duration="500" alphaFrom="1" alphaTo="0" />**

**<Dissolve duration="500" startDelay="500" alphaFrom="0" alphaTo="0" />**

**<Resize duration="500" startDelay="500" widthTo="50" heightTo="50" />**

**</Parallel>**

- Effects can also be defined programmatically

# Interstate gotchas

- Control how the state change interacts with effects by using SetPropertyAction, et. al.

```
<Transition fromState="from" toState="to">  
  <Sequence target="{startPanel}">  
    <Dissolve duration="500" alphaFrom="1" alphaTo="0" />  
    <SetPropertyAction property= "visible" value= "false" />  
  </ Sequence >  
</Transition>
```

# Components

- Break your application into pieces using components

## MyApp.mxml

```
<Application>
  <Script>
    function initStartPanel() {...}
    function initDetailPanel() {...}
  </Script>
  <HBox>
    <Panel id="startPanel" creationComplete="initStartPanel()">
      ...
    </Panel>
    <Panel id="detailPanel" creationComplete="initDetailPanel()">
      ...
    </Panel>
  </HBox>
</Application>
```

# Components

- Break your application into pieces using components

## MyApp.mxml

```
<Application>
  <HBox>
    <StartPanel id="startPanel"/>
    <DetailPanel id="detailPanel"/>
  </HBox>
</Application>
```

## StartPanel.mxml

```
<Panel creationComplete="initStartPanel()">
  <Script>
    function initStartPanel() {...}
  </Script>
  ...
</Panel>
```

# Service calls

- Make calls to the server using RPC calls
  - **<HTTPService url="xxx" request="yyy" />**
  - **<WebService wsdl="xxx">**  
    **<operation name="yyy" />**  
**</WebService>**
  - **<RemoteObject destination="xxx">**  
    **<method name="yyy" />**  
**</RemoteObject>**

# Service calls

- Results come asynchronously; listen for the “result” event.
- Alternatively, use data binding.
- Results can be retrieved off of the “result” property of the service itself.
- Always add fault handlers; you never know whether the connection will work.

# Data binding

- Use curly braces to bind ActionScript expressions to MXML attribute values.
- Not all attributes can be bound – use the [Bindable] metadata in order to make a member variable bindable.
- In beta 2, properties with getters and setters can also be made bindable through the same metadata.

**Better by Adobe.™**