

# Typography & Text Editing in the Text Layout Framework

---

Presented by Huyen Tue Dao

# About Me

---

- Name  $\approx$  “Hwin Tweh Dow”
- Fell in love with programming in a C++ class in high school.
- Flex developer since 2006 (but I do other stuff too, I swear).
- `numTimesSpeaking = 4.`
- Favorite race in SC is the Zerg: Overpool, cracklings + hydras



# Outline

---

- Overview
- Structure of the Text Layout Framework
- Using the Text Layout Framework
  - Typography
  - Layout
  - Graphics
  - Selection + Editing
- Extending the Text Layout Framework
- Related projects
- Wrap up

# Overview

---

- Text Layout Framework: “...delivers advanced, easy-to-integrate typographic and text layout features for rich, sophisticated and innovative typography on the web.”
- More text formatting ability: case and kerning and ligatures, oh my!
- More control over layout: bidirectional text, vertical text, columns, linked containers, inline images.
- Multi-lingual support.
- Selection and editing, including keyboard shortcuts.
- Bottom line: it is better than `htmlText`.

# Overview

---

- Available for Flex and Flash.
- Flash Player 10, Flex 3.2.
- Included in Flex 4 SDK.
- Continuously being updated outside of builds: [blogs.adobe.com/tlf/](http://blogs.adobe.com/tlf/)
- Also on SourceForge: [sourceforge.net/adobe/tlf/](http://sourceforge.net/adobe/tlf/)
- Versions: TLF1.0 included with Flex 4.0, TLF1.1 included in nightly builds, TLF2.0 available but requires file replacement and rebuilding of spark.swc.

# Overview

---

- Blah, blah, blah... Examples!
  - Redefine by Anirudh Sasikumar, Adobe Flex Evangelism team:  
[readefine.anirudhsasikumar.net/](http://readefine.anirudhsasikumar.net/)
  - TimesReader 2.0, New York Times AIR 2.0 application:  
[timesreader.nytimes.com](http://timesreader.nytimes.com)

# Structure of the Text Layout Framework

---

- The Text Layout Framework (TLF) built on top of the Flash Text Engine.
- “The Flash Text Engine (FTE) provides low-level support for sophisticated control of text metrics, formatting, and bidirectional text. It offers improved text flow and enhanced language support.”
- FTE available starting FP10 and AIR 1.5.
- Provides advanced text creation and manipulation but takes work to use.
- Text-rendering is fairly low level and is meant to be used as part of higher-level components.

# Structure of the Text Layout Framework

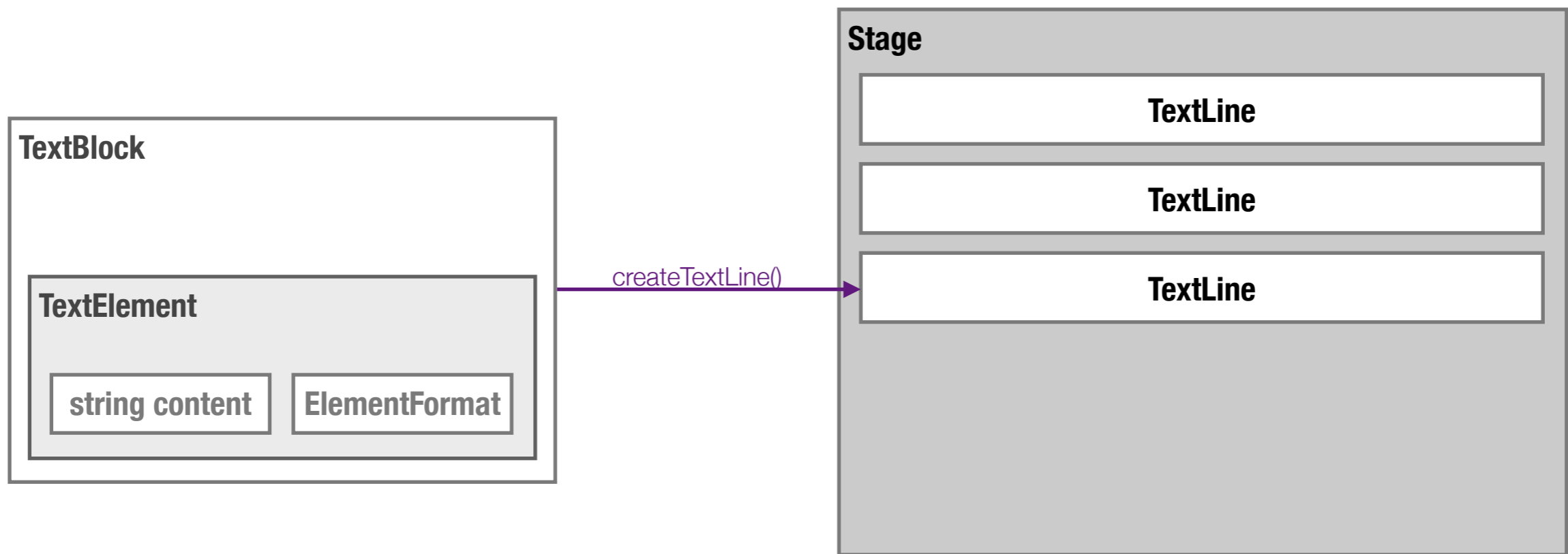
---

- Basic components of FTE:
  - TextBlock: factory for building paragraph of text.
  - TextElement/GraphicElement/GroupElement: containers for paragraph contents.
  - ElementFormat: formatting attributes for paragraph contents.
  - TextLine: line of text as a DisplayObject, what actually appears on stage.

# Structure of the Text Layout Framework

---

- Displaying text with FTE:
  - Create content element (Text/Graphic/GroupElement) from actual content and ElementFormat.
  - Create a TextBlock instance and set the new content element as the TextBlock's content.
  - Call createTextLine() on the TextBlock to generate TextLines.
    - Provide a previous TextLine and width for the TextLine
  - Add TextLines to stage.



# Displaying text with FTE

Artist (not-so-much) rendering

# Structure of the Text Layout Framework

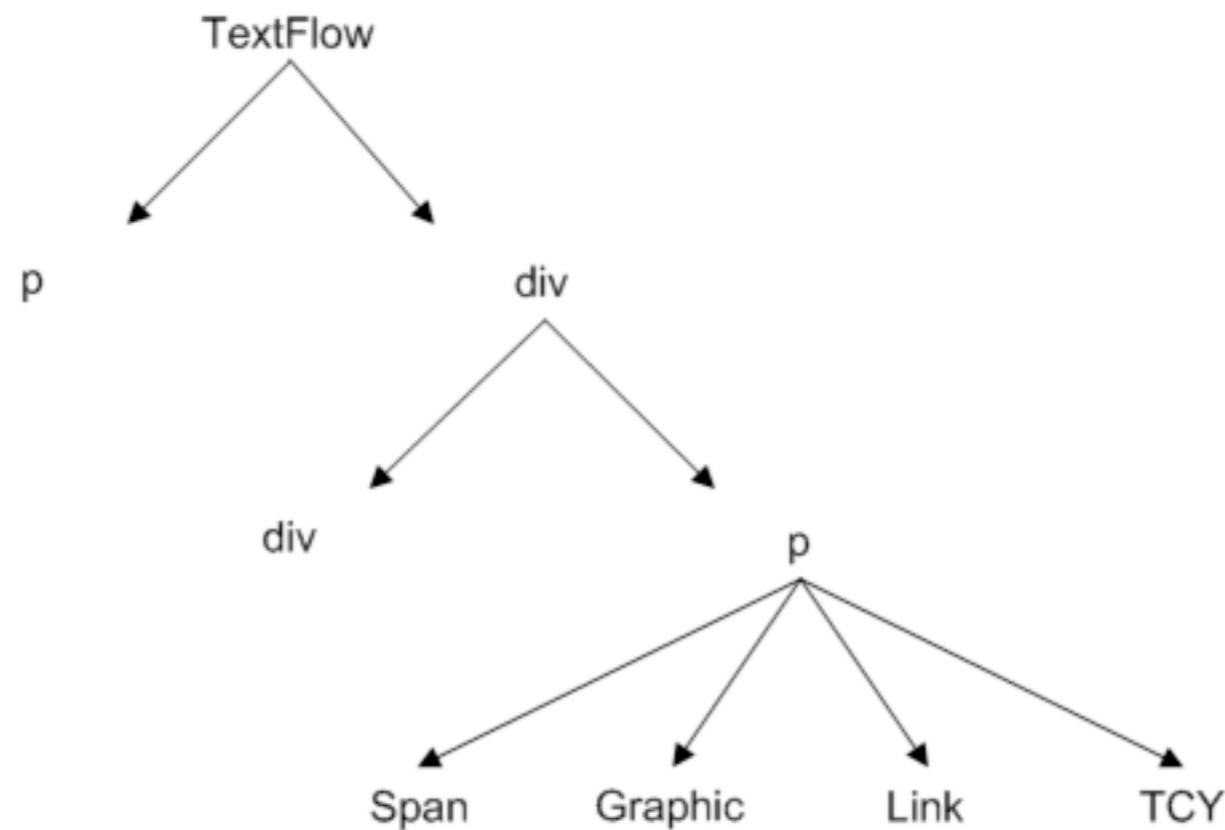
---

- TLF provides a MVC framework on top of the FTE.
- Provides hierarchical representation of a body of text and methods to update and edit that representation.
- TLF translates this representation via FTE to a series of TextLines that are displayed in some containers on the stage.
- Model: `flashx.textLayout.elements*`, `flashx.textLayout.formats.*`, `flashx.textLayout.conversion`
- View: `flashx.textLayout.factory.*`, `flashx.textLayout.container.*`, `flashx.textLayout.compose.*`
- Controller: `flashx.textLayout.edit.*`, `flashx.textLayout.operations.*`

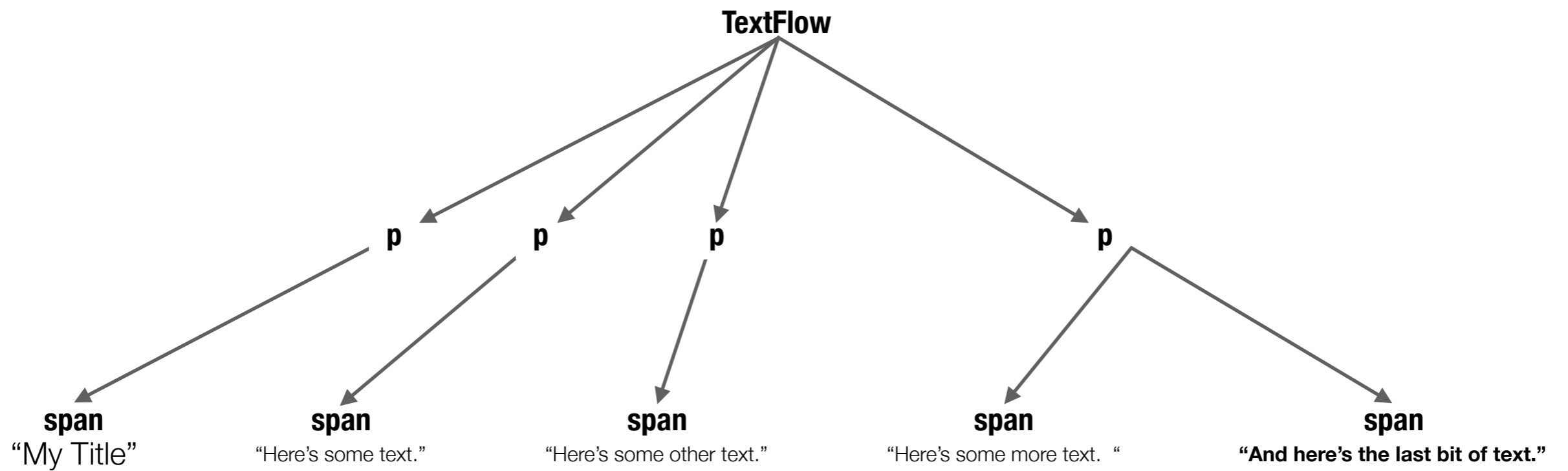
# Structure of the Text Layout Framework

---

- The Model: a “flow” of text represented as a hierarchy of FlowElements.
- Each node is a FlowElement: FlowGroupElement or FlowLeafElement
- Element types named for small subset of HTML tags.



My Title  
Here's some text.  
Here's some other text.  
Here's some more text. **And here's the last bit of text.**



## Text divisions in a TextFlow

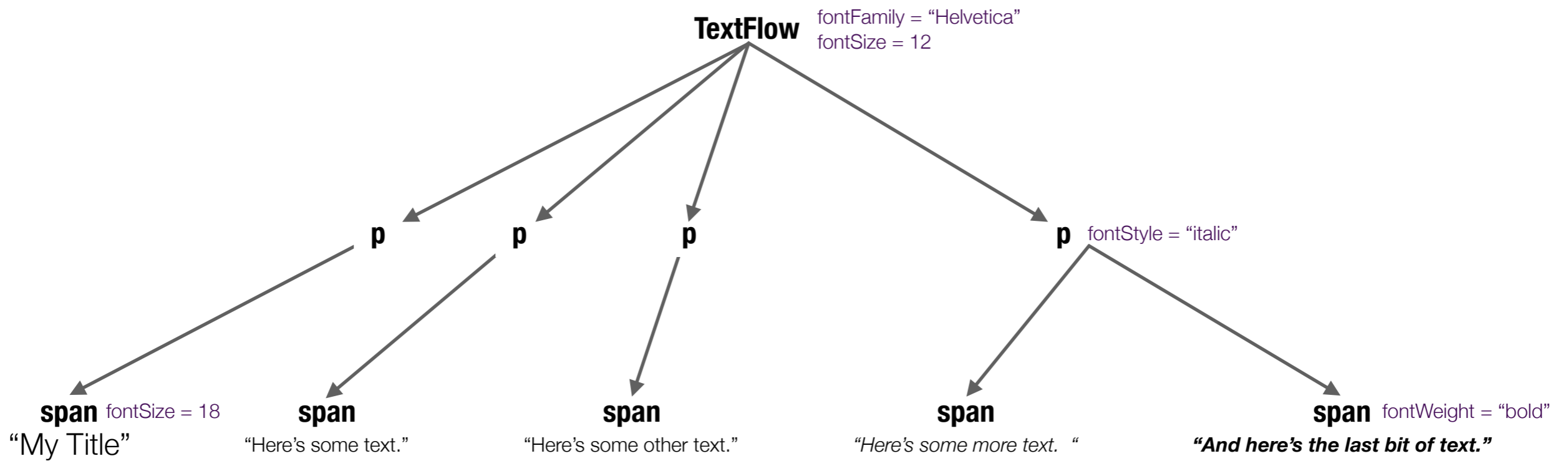
Artist (not-so-much) rendering

# Structure of the Text Layout Framework

---

- Formatting: each FlowElement implements the ITextLayoutFormat interface.
- ITextLayoutFormat defines access to various formatting properties; used in numerous places in TLF-MVC:
  - Font/text properties: color, size, style, weight, case
  - Spacing and line properties: kerning, leading model, line height, paragraph spacing, baseline
  - Layout: direction, alignment, padding
- FlowElements provide read/write access to these properties.
- ITextLayoutFormat properties are hierarchical: children inherit from parents.

My Title  
Here's some text.  
Here's some other text.  
*Here's some more text. **And here's the last bit of text.***

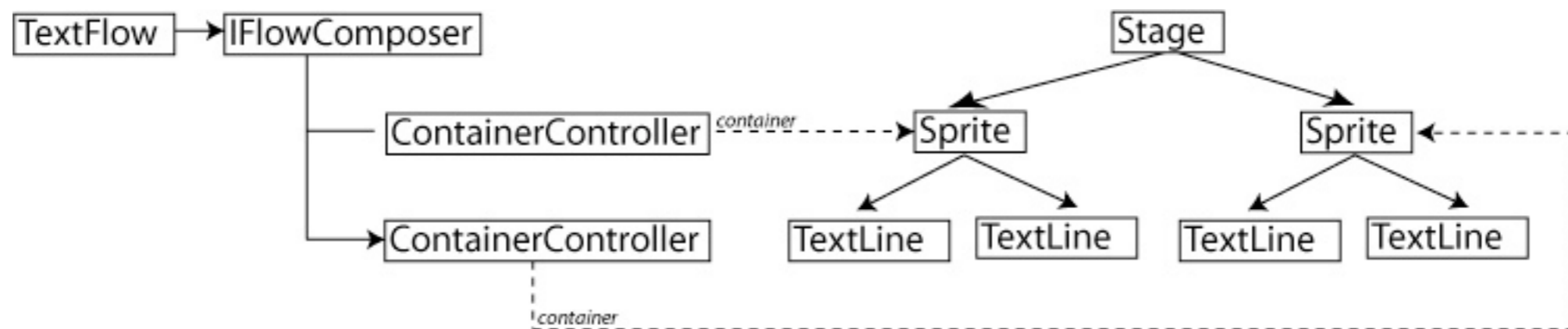


## Format inheritance in a TextFlow

Artist (not-so-much) rendering

# Structure of the Text Layout Framework

- The View: consists of a number of containers for text and TLF managers and composers for text.



- Containers: any number of Sprites on display list.
- IFlowComposer: a concrete implementation of this interface manages the layout and display of the text in the containers.
  - One IFlowComposer implementation in TLF 1.1: StandardFlowComposer.

# Structure of the Text Layout Framework

---

- ContainerController: intermediary between flow composer and individual containers.
  - Define the dimensions of the text area.
  - Populates associated container with TextLines.
  - Also implements ITextLayoutFormat: can specify “container-level” formatting options via ITextLayoutFormat properties on ContainerControllers.
  - Also contains handlers for fielding user input on the container.
  - Yes, it is a “controller” but main function is to facilitate TextLine placement. Handlers really pass events onto another object...

# Structure of the Text Layout Framework

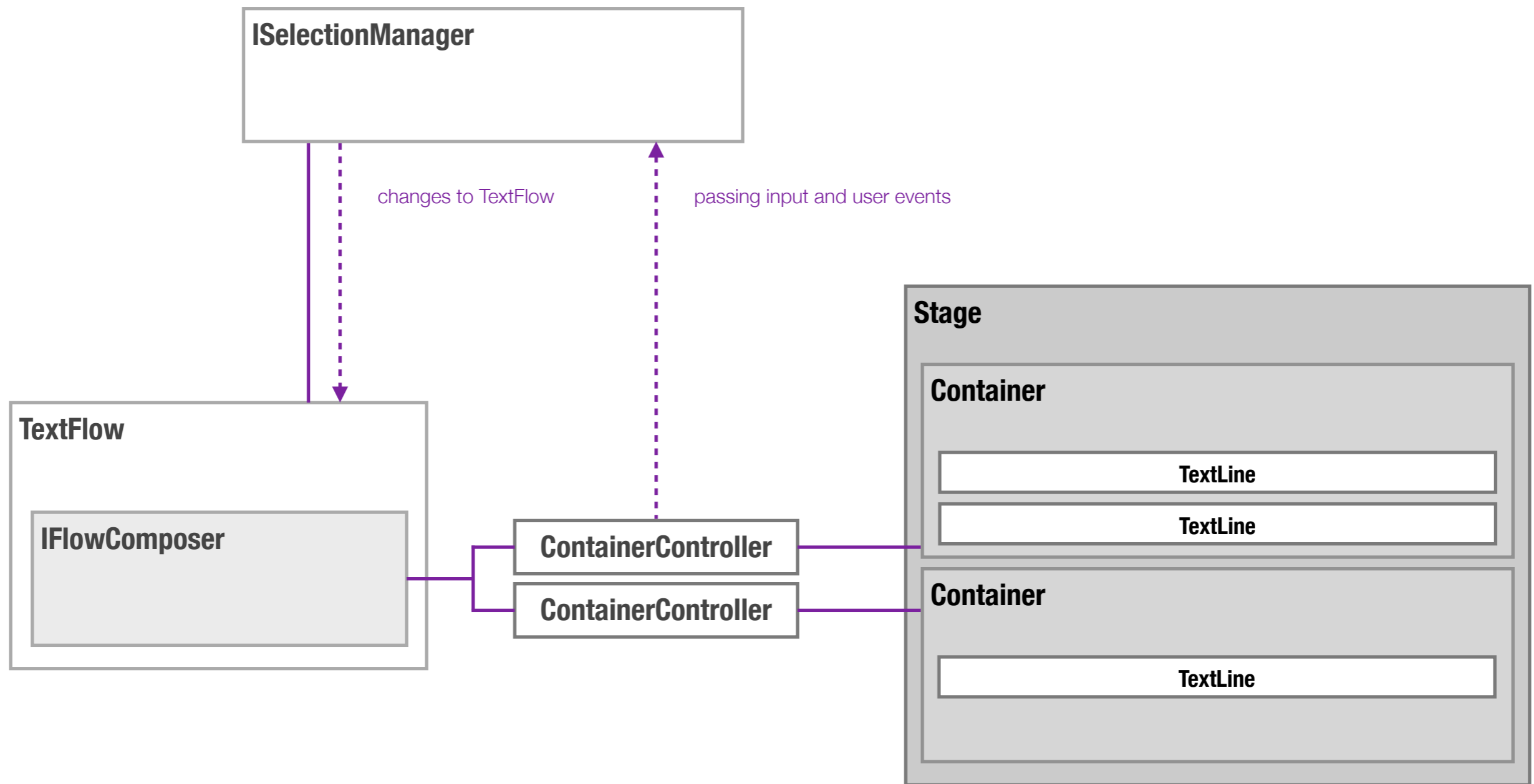
---

- The Controller: Each `TextFlow` can have an `interactionManager`, an implementation of `ISelectionManager`.
- `ISelectionManager`: manages and handles user input including selection and editing. Not required if user interaction with text is unneeded.
  - prescribes handlers that `ContainerControllers` call when fielding keyboard and mouse input.
  - provides interface for accessing information about user text selections including selection range, common formatting properties in selection.
- `SelectionManager`: `ISelectionManager` implementation in TLF1.1.

# Structure of the Text Layout Framework

---

- EditManager: subclass of SelectionManager that adds editing functionality.
  - Implements handler for mouse, keyboard actions involved in text editing.
  - Adds methods for insertion and deletion of text, formatting, changing FlowElement properties, and general structure for performing operations on a Textflow.



# Text Layout Framework MVC Setup

Artist (not-so-much) rendering

# Using the Text Layout Framework

---

- Blah, blah, blah... So how do I use all this?
- Simplest way: Flex components that use TLF.
  - RichText: low-level UIComponent for display one or more lines of rich text via TLF. Does not support scrolling, selection, editing.
  - RichEditableText: low-level UIComponent that does heavy lifting of displaying, scrolling, selecting, and editing (including limited undo/redo) rich text via TLF. Used by TextInput, TextArea.
  - RichText and RichEditableText have both a `text` property and a `textFlow` property: `text` = plain ol' text, `textFlow` = TLF rich text.

# Using the Text Layout Framework

---

- Before we go forward, how to create TextFlows in Flex.
- Can use both MXML and Actionscript.
- MXML: HTML Markup.
  - Can use `<s:content />` at compile time.

```
<s:TextArea id="someTextArea">
  <s:content>
    <s:p fontSize="12">This is some text.</s:p>
    <s:p fontSize="14">This is some bigger text.</s:p>
    <s:p>
      <s:span textDecoration="underline">This is underlined text.</s:span>
      <s:br />
      <s:span fontWeight="bold">This is bold text</s:span>
    </s:p>
  </s:content>
</s:TextArea>
```

# Using the Text Layout Framework

---

- MXML: Text Layout Markup

```
<s:TextArea id="someTextArea">
  <s:textFlow>
    <s:TextFlow>
      <s:p fontSize="12">This is some text.</s:p>
      <s:p fontSize="14">This is some bigger text.</s:p>
      <s:p>
        <s:span textDecoration="underline">This is underlined text.</s:span>
        <s:br/>
        <s:span fontWeight="bold">This is bold text</s:span>
      </s:p>
    </s:TextFlow>
  </s:textFlow>
</s:TextArea>
```

# Using the Text Layout Framework

---

- Actionscript: importing strings/HTML/Text Layout Markup via TextConverter

```
var plainOldText:String = "Hello, <b>World!</b>";  
textArea.textFlow = TextConverter.importToFlow( plainOldText, TextConverter.PLAIN_TEXT_FORMAT );
```

↳ Hello, <b>World!</b>

```
var htmlText:String = "Hello, <b>World!</b>";  
textArea.textFlow = TextConverter.importToFlow( htmlText, TextConverter.TEXT_FIELD_HTML_FORMAT );
```

↳ Hello, **World!**

```
var tlfMarkupXML:XML =  
    <TextFlow xmlns="http://ns.adobe.com/textLayout/2008">  
        <p>Hello <span fontWeight="bold">World!</span></p>  
    </TextFlow>;  
tlfTextArea.textFlow = TextConverter.importToFlow( tlfMarkupXML, TextConverter.TEXT_LAYOUT_FORMAT );
```

↳ Hello, **World!**

# Using the Text Layout Framework

---

- 3 input formats (TextConverter constants): PLAIN\_TEXT\_FORMAT, TEXT\_FIELD\_HTML\_FORMAT, TEXT\_LAYOUT\_FORMAT
- When using TLF Markup, need to follow these guidelines for validity
  - Root tag: `<s:TextFlow xmlns="http://ns.adobe.com/textLayout/2008">`.
  - Must be TextFlow-supported tags: div, p, a, tcy (Tatechuuyoko - ta-tae-chu-yo-ko), span, img, tab, br.
- Need easy mode? TextFlowUtil, easier to use (e.g., does not require `<s:TextFlow>` at root), less flexible.
  - Import methods: `importFromString`, `importFromXML`.

# Using the Text Layout Framework

---

- Can export as well as import.
- `TextConverter.export`:
  - Data type (`ConversionType` constants): `STRING_TYPE`, `XML_TYPE`
  - Format (`TextConverter`): `PLAIN_TEXT_FORMAT`,  
`TEXT_FIELD_HTML_FORMAT`, `TEXT_LAYOUT_FORMAT`
- `TextFlowUtil.export`: exports `TextFlow` as XML in `TEXT_LAYOUT_FORMAT`.

# Using the Text Layout Framework

---

- Actionscript: Manually creating TextFlow and FlowElements.

```
var textFlow:TextFlow = new TextFlow();
var paragraph:ParagraphElement = new ParagraphElement();
var span1:SpanElement = new SpanElement();
var span2:SpanElement = new SpanElement();

span1.text = "Hello, ";
span2.text = "World!";
span2.fontWeight = FontWeight.BOLD;

paragraph.addChild( span1 );
paragraph.addChild( span2 );

textFlow.addChild( paragraph );

textArea.textFlow = textFlow;
```

↳ Hello, **World!**

- Please note that all strings assigned to a `text` property of some `SpanElement`; contrast with markup where text can sit within a `<p>`.

# Using the Text Layout Framework

---

- The much harder, more powerful way of using TLF:
  1. Create a TextFlow
  2. Create some containers
  3. Connect the TextFlow to the containers via flow composer and container controllers.
  4. (Optionally) Create a selection manager or edit manager.
- Have to manually set up scroll bars, selection, editing, container size.
- Blah, blah, blah... So let's look at actual code.

# Using the Text Layout Framework

---

- Typography (that's like don't use Comic Sans, right?) in TLF:
  - 3 levels of typography/formatting properties in TextLayoutFormat:
    - Character: can apply to a single character or a single line of text.
    - Paragraph: apply only to an entire paragraph.
    - Container: apply only to a container holding text.
  - Some TLF methods for apply new formats and styles will be specific to a specific category of styles: important to know which are which.
  - Tip: look at source of TextLayoutFormat

# Using the Text Layout Framework

---

- How to set formatting properties:
  - MXML/Importing from XML or string: formatting properties as attributes in the tags.

```
<TextFlow xmlns="http://ns.adobe.com/textLayout/2008">  
  <p>Hello <span fontWeight="bold">World!</span></p>  
</TextFlow>;
```

- Use `setStyle` on `FlowElements`.

```
var spanElement:SpanElement = new SpanElement();  
spanElement.text = "Some text";  
spanElement.setStyle( "color", 0xFF0000 );
```

- Set properties directly on `FlowElements`.

```
var spanElement:SpanElement = new SpanElement();  
spanElement.text = "Some text";  
spanElement.color = 0xFF0000;
```

# Using the Text Layout Framework

---

- What about CSS?
  - Currently no built-in CSS support (bummer).
  - TextFlow formatResolver property: IFormatResolver.
  - Can develop implementation for IFormatResolver that allows CSS-type styling.
  - Example from the TLF team blog: <http://blogs.adobe.com/tlf/2009/02/iformatresolver-and-using-css-1.html>
  - Blah, blah, blah... Let's look at some more formatting!

# Using the Text Layout Framework

---

- Graphics: InlineGraphicElement class, <img>; to use just specify the source property of the InlineGraphicElement.
  - Relative or full path to an image.

```
<s:TextFlow xmlns="http://ns.adobe.com/textLayout/2008">
  <s:p><s:img source="http://www.flickr.com/photos/24861453@N02/3675093298/" /></s:p>
</s:TextFlow>
```

- Embedded image.

```
<fx:Script>
[Embed( source="http://www.flickr.com/photos/24861453@N02/3674293959/in/photostream/" )]
[Bindable]
public var imageClass:Class;
</fx:Script>

<s:TextFlow xmlns="http://ns.adobe.com/textLayout/2008">
  <s:p><s:img source="{ imageClass }" /></s:p>
</s:TextFlow>
```

# Using the Text Layout Framework

---

- Sprite or FXG.
  - UIComponent == Sprite, right? Yes and no. InlineGraphicElement is not a container => issues with size and validation.
- Other things to note with InlineGraphicElement:
  - When not using embedded graphics, InlineGraphicElement only renders if explicit width and height set.
  - Can retrieve measuredHeight and measuredWidth after image loads via `StatusChangeEvent.INLINE_GRAPHIC_STATUS_CHANGE`.
- Blah, blah, blah... Let's look at some examples!

# Using the Text Layout Framework

---

- Layout: managing and manipulating containers that hold a flow of text.
- Flex 4 Spark components with TLF do not provide many layout options.
- Can use straight-up TLF to manipulate the layout of text and containers.
  - Dynamically sizing containers and rendered text.
  - Multiple, linked containers for a TextFlow.
- TLF 2.0 has floats.

# Using the Text Layout Framework

---

- Multiple, linked containers for a single TextFlow:
  - Instantiate a number of containers.
  - Link to TextFlow.flowComposer with multiple ContainerControllers.
  - Update controllers.
- Dynamically sizing containers and rendered text:
  - Add handler for ResizeEvent.RESIZE on containers.
  - Handlers call updateToController.
- Blah, blah, blah, let's look at some code!

# Using the Text Layout Framework

---

- Selection: to implement, create new `SelectionManager` and assign as `interactionManager` of `TextFlow`.
- `SelectionManager` has properties that describe selection location in text:
  - `anchorPosition`: start position of selection (user mouse down).
  - `activePosition`: end position of selection (user mouse up).
  - `absoluteStart` and `absoluteEnd` of selection in text.
- `SelectionManager.getSelectionState():SelectionState` encapsulates the above.
- When text has cursor, `anchorPosition == activePosition` and `absoluteStart == absoluteEnd`.

# Using the Text Layout Framework

---

- SelectionManager also has properties/methods for format information.
  - SelectionFormat properties: related to selection indication, not ITextLayoutFormat, includes both highlight and cursor attributes.
  - Format information about selected text:
    - getCommonCharacterFormat: common to all characters in selection.
    - getCommonParagraphFormat: common to all paragraphs in selection.
    - getCommonContainerFormat: common to all containers in selection.
    - SelectionManager.pointFormat: formatting applied to inserted text.

# Using the Text Layout Framework

---

- TextFlow dispatches selection change events.
- ElementRange: Handy class for iterating through selected elements in a TextFlow; created from SelectionState.
  - `absoluteStart`, `absoluteEnd`, `firstLeaf`, `lastLeaf`,  
`firstParagraph`, `lastParagraph`
  - `characterFormat`, `paragraphFormat`, `containerFormat`

# Using the Text Layout Framework

---

- Editing: to implement, create new `EditManager` and assign as `interactionManager` of `TextFlow`.
- `EditManager` provides several methods for manipulating text; most take a `SelectionState` as optional parameter: otherwise, use current selection state.
- Applying formats:
  - `applyFormat`: apply a character, paragraph, and container format.
  - `applyCharacterFormat/applyParagraphFormat/applyContainerFormat`.
  - `clearFormat`.

# Using the Text Layout Framework

---

- Inserting/deleting text: `insertText`, `deleteText`, `overwriteText`, `deleteNextCharacter`, `deleteNextWord`, `deletePreviousCharacter`, `deletePreviousWord`.
- Transforming text: `applyLink`, `applyTCY`, `splitParagraph`.
- Cut/copy/paste on `TextFlow`: `cutTextScrap`, `copyTextScrap`, `pasteTextScrap`.
  - `TextScrap`: Fragment of a `TextFlow` created from range of text.
- Graphics: `insertInlineGraphic`, `modifyInlineGraphic`.
- All these methods built on top of `FlowTextOperations`: encapsulates a change to a `TextFlow`, provides interface for undoing and redoing operations.

# Using the Text Layout Framework

---

- Operations-related methods: doOperation, undo, redo.
  - doOperation executes FlowTextOperations and also keeps order and changes from operations consistent.
  - To allow undo/redo, need to construct EditManager with UndoManager instance.
  - UndoManager manages undo and redo queues.

# Using the Text Layout Framework

---

- TLF provides a lot of basic rich text editor functions out-of-the-box.
- Blah, blah, blah... Let's build a rich text editor then.

# Extending the Text Layout Framework

---

- Can extend the TLF in several ways:
  - Extend the SelectionManager, EditManager, ContainerController.
  - Create your own tags/FlowElements.
  - Create your own FlowTextOperations.
- Caveats:
  - Several FlowElement classes are marked final (that you wish weren't).
  - Many methods/properties are private (that you wish weren't).
  - Custom undoable/redoable operations may take a lot of work.

# Related Projects

---

- Text Layout Framework Extended (TLFX): HTML/CSS support for TLF.
  - Replace textLayout.swc in Flex SDK.
  - Helpful to study if making own elements and extensions.
  - <http://code.google.com/p/tlfx/>
- tinytlf by Paul Taylor: Actionscript 3.0 text layout framework
  - Light, modular framework built on FTE.
  - HTML/CSS support.

# Wrap up

---

- Text Layout Framework: library for advanced rich text functionality..
- Can be used existing Flex 4 components or used to create custom ones.
- Rich typographic features, inline graphics, advanced layout, including linked containers and bidirectional/vertical text.
- Provides a wide range of text modes and interaction functionality.
  - Text selection including parsing of format information.
  - Text editing with undo/redo ability.
- With some effort can be extended to add custom operations and elements.

# Wrap up

---

- Things to be mindful of:
  - Custom TLF component can become heavy when using full set of selection and editing features.
  - TLF is still in development so still some hiccups and glitches.
  - Extensibility is there but a challenge.
  - If using the TLF, stay on top of new builds and features: could save you time and open up more functionality.
- Overall, the Text Layout Format provides great rich text functionality out-of-the-box.

# Questions?

---

Thanks for coming!

Huyen Tue Dao  
daotueh@gmail.com  
queencodemonkey@twitter.com  
www.queencodemonkey.com

# References

---