Syntax:

```plaintext
for (var i:int = 0; i < condition; i ++) {
    //normal for loop
}

for (var name:type in array) {
    //loops over the indices of the array
}

for each (var name:type in array) {
    //loops over the values of the array
}
```

The `TextFormat` object can be used to set many other properties as well.

The auto size property must be set so that the TextField knows how it should respond to formatting.

In this case in treats the text as left justified.

FlashVars are a way for us to load a flash movie with parameters. The format is the same as that for HTML query strings. You can also just add a query string to the end of the data attribute.
You can treat the params object as if it were an associative Array
import flash.external.ExternalInterface;
...
ExternalInterface.call("jsFunction", param1, param2,…);
...
var returnValuetype = ExternalInterface.call("jsFunction", param1,…);
Syntax:

**ActionScript**

import flash.external.ExternalInterface;

... 
ExternalInterface.addCallback("name", myFunction);

... 
public function myFunction(param:type, ...):type { /*do stuff*/ }

**Javascript**

$("flashMovie").name(parameters);

- The id flashMovie must be attached to the object tag that you use to embed the flash movie
Now that we have allowed our flash movie to talk to Javascript and vice versa some issues arise.

So now we can call any functions that the page has.

What if we make anonymous functions?

Lucky (or disappointingly) Javascript cannot do things like read files or connect to databases, but we can still cause some damage.
ExternallInterface Security

Syntax:

**HTML**

```html
<object type="application/x-shockwave-flash" data="yourfile.swf"
    width="550" height="400">
    <param name="yourfile" value="yourfile.swf" />
    <param name="allowScriptAccess" value="always" />
</object>
```

**Actionscript**

```javascript
import flash.system.Security;
...
Security.allowDomain("*");
```

- "*" allows all domains, you can specify a specific one instead if you are concerned with security
FlashVars
- Allows HTML to talk to flash
- Allows PHP to talk to flash

ExternalInterface
- Allows Javascript to talk to flash
- Allows flash to talk to Javascript
ActionsScript can load and process XML data.

Syntax:

```plaintext
import flash.net.*;
...
public function loadXML():void {
    var xmlLoader:URLLoader = new URLLoader();
    xmlLoader.addEventListener(Event.COMPLETE, readXML);
    xmlLoader.load(new URLRequest("file.xml"));
}

public function readXML(e:Event):void {
    var xml:XML = new XML(e.target.data);
    //do stuff with the XML
}
```
More Security Issues

- Flash allows you to process files either locally or remotely, but not both
- This guards against stealing data
- The default is to be able to read remote files
- There are several ways to fix this issue in decreasing convenience order
  - Visit this [site](http://labs.mstudio.com/?p=31) and add whatever directories you want to the trusted directories list for your flash environment
  - Load your files onto webster and test there
  - Add the `-use-network=false` option when using mxmlc
When handling XML data you can use the XML object and the XMLList object.

These objects behave similarly and in most cases you think of them as being the same thing.

You can even think of an XMLList as simply a collection of XML objects.

When you create an XML object, that object represents the root node of the XML.
Traversing an XML object is much simpler in Actionscript.

```actionscript
var xml:XML = new XML(e.target.data);

//access each node in the xml with name “nodeName”
for each (var node:XML in xml.nodeName) {
    //access the “attrName” attribute of the node
    var attr:String = node.@attrName;
    //get the text in the node
    var value:String = node.text();
}
```
MXML is a type of XML

MXML describes the layout of user interface components in a flash application

Instead of tedious addChild calls and setting x, y, width and height values, we can describe the layout relationships just like we would in html

You can attach external Actionscript files

You can attach external stylesheets that are very closely in line with CSS

MXML is not a separate language, it is shortage for regular ActionScript and is directly translated to it
MXML Syntax

Syntax:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://www.adobe.com/2006/mxml"
    layout="absolute" height="400" width="400">
    <!-- Your MXML goes here -->
</Application>
```

- MXML looks a lot like HTML
UI Components

Syntax:

```xml
<Text text="blah" />
```


Syntax:

```xml
<TextInput text="blah" />
```


Syntax:

```xml
<Button label="blah" />
```

Syntax:

<Alert text="blah" />


Syntax:

<RadioButton groupName="blah" label="blah" />


Syntax:

<Image source="@Embed('location')" />

Terrible Example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://www.adobe.com/2006/mxml"
    layout="absolute" height="400" width="400">
    <Text fontFamily="Garamond" fontSize="12pt" text="blah" />
</Application>
```
Bad Example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://www.adobe.com/2006/mxml"
    layout="absolute" height="400" width="400">
    <Style>
        @namespace "library://ns.adobe.com/flex/mx”;
        Text {
            fontFamily: “Garamond”,
            fontSize: 12pt;
        }
    </Style>
    <Text text="blah" />
</Application>
```
Good Example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://www.adobe.com/2006/mxml"
    layout="absolute" height="400" width="400">
    <Style source="ex.css"/>
    <Text text="blah"/>
</Application>
```

`ex.css`

@namespace "library://ns.adobe.com/flex/mx";

Text {
    fontFamily: “Garamond”;
    fontSize: 12pt;
}

Flex makes a distinction between style properties and attributes of UI components.
This means that some attributes, most notably, $x$, $y$, width and height cannot be set in your css file.
Instead you can use the verticalCenter and horizontalCenter style properties.
Instead of giving a UI components css class you give it a styleName.
You need that icky line at the top to prevent warning messages.
In the same way that we attach JavaScript to HTML, we attach ActionScript to MXML.

We could write code obtrusively in the MXML file, but this would be bad style.

Many Javascript/HTML paradigms are also present with Actionscript/MXML.
Example:

<?xml version="1.0" encoding="utf-8"?>
<Application xmlns="http://www.adobe.com/2006/mxml"
    layout="absolute" height="400" width="400">

    <Style source="ex.css" />
    <Script source="ex.as" />
    <Text text="blah" />
</Application>
Containers

Syntax:

<VBox>…</VBox>
<HBox>…</HBox>


Syntax:

<Canvas>…</Canvas>


Syntax:

<Panel title="blah">…</Panel>


In Sum

- MXML in junction with ActionScript and CSS is a browser independent alternative to traditional web pages
- Another cool benefit is that these apps can be run on the desktop if you install the Adobe Air Player
- Flash is really awesome!