Flex is a component-based application development framework. It comes with a set of prebuilt visual and nonvisual components. These components are available through the framework API in both ActionScript 3.0 (which from here on will be referred to as AS3) and MXML form. AS3 is the latest version of ActionScript, the ECMA (third edition: ECMA-262) standard–compliant language that compiles to bytecode for the Flash Virtual Machine (VM). AS3 runs on ActionScript Virtual Machine 2 (AVM2), which is available with Flash Player 9. AVM2 improves the application runtime performance over ten times as compared to ActionScript Virtual Machine 1 (AVM1). AS3 is a statically and strongly typed object-oriented language. MXML is an XML-based user interface markup language that translates to AS3 code before being compiled to run on the Flash VM. This rich set of components, exposed via AS3 and MXML APIs, helps you rapidly build your applications and provides you a starting point to extend the framework to implement advanced controls, containers, and service components. Components can be extended using AS3 or MXML. Either of the approaches has its advantages and disadvantages, and the choice should be driven by the specific use case at hand.

Let’s start by extending MXML components.
Extending components using MXML

Flex exposes its controls and containers in MXML. This makes it possible for you to build rich interactive applications using declarative XML markups. As mentioned earlier, MXML code translates to AS3 classes. Snippets of AS3 code can be embedded in an MXML file within the Script begin and end tags, and this code becomes part of the AS3 class that maps to the MXML component. This technique is useful because often XML markups alone are not sufficient to manipulate data and implement behavior. Another advantage of this technique is that it lets you abstract out reusable pieces as custom components.

Therefore it's possible to extend the available set of framework components and package those components up as reusable pieces without much trouble. This is exactly what we do when creating custom components using MXML. The best way to learn the details of this process is to walk through a few examples.

Walking through a simple example

As a first example, let's create a custom radio button group with five buttons, labeled as follows:

- Strongly Agree
- Agree
- Indifferent
- Disagree
- Strongly Disagree

Such a radio button group can be reused in a questionnaire application, which records responses to various questions on a scale that varies from strong agreement to strong disagreement. Such questions are common when measuring personality tests or invoking responses to political or economic issues. The code for such a button group could be as follows:

```xml
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox
xmlns:mx="http://www.adobe.com/2006/mxml"
width="400" height="300">
    <mx:RadioButton groupName="responseType"
        id="stronglyAgree"
        label="Strongly Agree"
        width="150"/>
    <mx:RadioButton groupName="responseType"
        id="agree"
        label="Agree"
        width="150"/>
    <mx:RadioButton groupName="responseType"
        id="indifferent"
        label="Indifferent"
        width="150"/>
    <mx:RadioButton groupName="responseType"
        id="disagree"
        width="150"/>
</mx:VBox>
```