Chapter 4: Interaction

After completing this module, you’ll be able to:

• identify events and actions.
• create simple actions.
• identify four states of buttons: up, over, down, and hit.
• create buttons with ActionScripts.
• using ready-made and invisible buttons.
• use behaviors for simple activities.
• use Timeline control actions.
• create presentations and exhibits.
• create draggable activities.
• add components.
• create interactive presentations.
• build quizzes.
• apply three types of text boxes: static, dynamic, and input.
• repurpose ActionScripts.

Although Flash creates excellent animations, it’s also possible to make your project interactive. In other words, the project can change its behavior or appearance in response to user actions. For example, pressing a button may trigger music to play or an animation to start.

The following are examples of projects that can be created using the interactive features of Flash.

• tutorial
• simulation
• interactive slide show, timeline, map
• matching game
• movie controller
• quiz
• video game
Developing quality interaction takes careful planning. You’ll be using an authoring language called ActionScript to write simple programs to control interactions in your project. All interactions have two elements: events and actions.

**Events** are triggers. They are things that happen while the movie is playing such as clicking a button or rolling over an image. An event also occurs when things happen on the Timeline such as when the Playhead reaches a particular keyframe. In ActionScript, you’ll be using words such as `on release` and `on mouseOver` to describe these events.

**Actions** are short instructions that tell Flash what things should happen as a result of events. For example, when a mouse is clicked a sound might play, elements might appear or disappear, a series of frames might be skipped, or data might be sent to a database for analysis.

Regardless of the type of interactivity, ActionScript is an authoring language that is used to specify what should happen when an event is triggered. You need to learn enough about ActionScript to be able to build in the specific commands needed to accomplish your goals.

Buttons are a common source of interactivity. Flash buttons are designed to automatically detect events such as mouse roll overs and clicks. You can write simple ActionScripts to control the action.

**ActionScript 2.0 vs ActionScript 3.0**

Two versions of ActionScript are currently being used to create interactivity on Flash projects. Both **ActionScript 2.0** and **ActionScript 3.0** are available in the new versions of Flash. As a result, you can choose which version best meets your needs.

ActionScript 2.0 is continuing to be supported because it’s much easier to use in some ways than the newer ActionScript 3.0. When you create a NEW flash project, you can choose between ActionScript 2 or 3. The options are Flash File (ActionScript 3.0) or Flash File (ActionScript 2.0). When you start your project, you should make a decision about which you will be using. Choose the option for 2.0 or 3.0 from the New Document window as you create your new project.

If you’re not sure which you are using or are given a document to use, choose the Properties panel and it will show you in the PUBLISH area.

If you want more details or want to switch from one option to the other, choose File>Publish Settings>Flash and it will provide the ActionScript Version. Although it’s possible to convert a project from 2.0 to 3.0, it messes up coding and is not recommended.

If you are a beginning user, consider using ActionScript 2.0. If you are an advanced developer, you may choose to use ActionScript 3.0.

Consider your experience with computer programming. Getting into extensive online quizzes and the automatic checking of answers may be too
big a leap for beginners. Getting things to drag is easy, but detection and checking answers is much more complicated. Use your best judgement in ‘stretching’ your skills and selecting your techniques.

**Understanding Coding**

Before jumping into coding, keep in mind that ActionScript is picky about syntax.

- Coding is exact. Check your syntax. Be sure to carefully follow wording, spacing, and punctuation.
- When possible, use the script suggested in the Actions toolbox. Or, find ActionScripts online that you can copy. This is much easier than writing original code.
- Remember that some codes require an event. For example, gotoAndPlay must be preceded by on (release) or another event.
- Lines end with a semicolon such as `stop();`

**ActionScript 2.0**

We’ll start with an exploration of ActionScript 2.0. Then we’ll provide ideas for using ActionScript 3.0.

You’ll need to switch to ActionScript 2.0 if you want to try the following script.

To switch from ActionScript 3.0 to 2.0:

Choose File>Publish Settings>Flash.
In the window, choose the Flash tab and change to ActionScript 2.0.

Let’s add actions to our Timeline that will stop the animation on the first and last frames. Then, we’ll use a button to control the movie.

To add stop actions to the first and last frame of the Timeline:

Open your airplane animation.
Create a new layer called *action*.
Select Frame 1 of the *action* layer.
Pull down the Window menu, choose Actions.
The Action panel will appear on the screen.
Click in the programming window and enter the following code.

```javascript
stop();
```

Notice that the letter `a` appears in Frame 1 on the Timeline. This indicates that there is an action on this Keyframe.

Select Frame 120 of the *action* layer.
Pull down the Insert menu, choose Timeline, select Keyframe.
Notice that there is now another Script tab in the Script pane.
You'll want to put the stop action under the Action: 120 script tab.
Click in the programming window of the Action panel and enter the stop command.
Again, the stop script will appear in the window.
You now have two stop scripts. One for Frame 1 and one for Frame 120.
Creating a Button using ActionScript 2.0

Now that we’ve identifying the stop action for the first and last frames, we need to create a way for the end user to control the action. Let’s create a button that the user will click to play the animation.

To create a button:
Create a new layer. Name it start.
Click on the first frame of the Timeline in the Start layer.
Use the oval tool to create a oval.
Place the oval near the top of the screen in the center.
Pull down the Window menu, choose Color.
Spend some time exploring the color types including solid, linear, and radial.
Click the Color Swatches and explore the options.
Change the fill to a red starburst.
In the Properties panel, experiment with other options related to the lines and fill.
For example, you could create beveled or rounded edges.
Create white text that reads: Start.
Choose the pointer/arrow tool.
Click the oval. Shift-click the text.
The oval and text should now both be selected.
Pull down the Modify menu, choose Group.
Pull down the Modify menu, choose Convert to Symbol under Type, choose Button.
Name this button, start.

To add an ActionScript to the button:
Select the button.
Pull down the Window menu, choose Actions.
The Action panel will appear. Notice that the start button tab (bottom center of Actions Window) is active. This time we’ll enter a script by hand.
Click in line 1 of the Script pane.
Type the following script:

```javascript
on (release) {
    gotoAndPlay(2);
}
```

The release command indicates that when
the mouse has been clicked, the program will go to Frame 2 and begin playing.
Pull down the Control menu, choose Test Movie.
Click the Start button for the movie to run.
Notice that the movie stops on Frame 1 and waits for the user to click the button. At Frame 120, the system stops the movie.

You can also use the tool in the Actions panel to select from a long list of possible commands. This saves time and also helps with syntax.

![Actions Panel](image)

**Using Buttons**

You’re probably familiar with buttons from using them on web pages or in PowerPoint. Buttons are graphic images, but they also react to user input. For example, rolling over a button may change its appearance or clicking a button may take you to a different page.

In Flash, **button** is simply a symbol that responds to user input. A button has four states: up, over, down, and hit. Each state may exhibit a different effect.

- **Up** is what the button looks/acts like when you first see the button but haven’t taken any action.
- **Over** is what the button looks/acts like when the mouse is over the button but hasn’t been clicked.
- **Down** is when the button has been clicked and the mouse button is being released.
- **Hit** is the area of the button that responds to the mouse.

Many developers like to create different versions of their buttons to distinguish these mouse states. For example, you might create a lightened version for “over” and a darkened version for “down.”

Buttons require planning. Carefully consider the placement and function of each button. Let’s modify our “start” button.
To create the behaviors for the buttons:
   Double-click the Start button.
   The Timeline will change. It now shows the Flash symbol editing mode for buttons. You already have the button made for the “up” state.
   Right-click or Control-click the Over frame and choose Insert Key-frame.
   Change the button color to a blue starburst.
   Right-click or Control-click the Down frame and choose Insert Key-frame.
   Change the button color to a red starburst.
To test your buttons:
   Pull down the Control menu and choose Enable Simple Buttons.
   When you’re done playing, be sure to go back and turn off the Enable Simple Buttons option.

**Ready-Made Buttons.** Although you can make your own buttons, you can also use the ready-made buttons in Flash.

To use a ready-made button:
   Pull down the Window menu, choose Common Libraries, select Buttons.
   You’ll see a long list of folders. Double-click on a button folder to see many versions of buttons in different colors. You’ll also see slides, knobs, and other button-like objects.
   Drag a button onto the Stage.
   Double-click the button and you’ll see that it contains many layers.
   Pull down the Control menu, choose Test Movie.
   Notice that many of the buttons contain highlights and animations.

**Invisible Buttons.** Sometimes you may want to create an invisible button. For instance, let’s say you have a map of North American on your Stage and cover the area of Canada with a transparent button. When users click on Canada, the national anthem may play. In this case, you would only want to activate the “hit” state of your button. Or, you might want to create invisible buttons over the areas of a soccer field. During a quiz, you could ask students to click on a particular area of the field such as the goal box.

Invisible buttons are useful when you want a “hot spots” on your screen without obscuring your graphics.

To create an invisible button:
   Click on the main Stage in your project.
   If you are still in the “button” symbol, click on Scene 1.
   Create a new layer and call it invisbutton.
Create a Keyframe.
Select the Rectangle tool in the Tools palette.
Find the Colors section of the Tools panel and click the pencil icon to select the Stroke color control. Select No Color.
This will put a red diagonal line in the fill and disable the line around the rectangle.
Drag the mouse diagonally across the Stage to create a rectangle the size that will cover your object such as the goal box on a soccer field.
Click the selection tool (arrow/pointer) and select the rectangle you just made (both the inside and outside).
A cross-hatch pattern appears over the rectangle when you select it.
Press F8 (or Modify Menu>Convert to Symbol) to change the rectangle into a symbol. Give it a name such as invis1. Select Button, and click OK.
Double-click the new button on the Stage to enter Symbol editing mode.
You only want the rectangle on the Hit frame.
Click the keyframe at the Up frame and drag the keyframe to the Hit frame.
Now the rectangle is clickable, but users can’t see it.
Click Scene 1 to return to the main Timeline.
There is a light blue rectangle on the screen that represents the invisible button’s Hit area. See the invisible button over the goal below.
Using Behaviors in ActionScript 2.0

You can create some interesting actions without learning to do much in ActionScript. You simply add code directly into the button from the Behavior panel. Behaviors are prewritten scripts for common actions.

If you have lots of coding to do, you’ll probably want to use the Action panel and write the script yourself. However, the Behavior panel works great for simple actions such as linking to websites.

Let’s create a button on the last frame of our project. It will take users to a website on the topic of Flight.

To add action directly to a button using the Behaviors panel:

1. Create a graphic and convert it into a symbol called flightsite.
2. Pull down the Windows menu and choose the Behaviors panel.
3. Click the plus sign and choose Web and Go to Web Page.
4. Enter the website address and click OK.

The new event shows on the Behaviors list. By using the Behaviors menu, you avoided having to type in the code yourself. However, if you pull down the Windows menu and choose the Action panel, you can see the code that was created:

```javascript
on (release) {
    //Goto Webpage Behavior
    getURL("http://42explore.com/flight.htm","_self");
    //End Behavior
}
```

Behaviors can be used to control objects such as graphics, sound, video, and movie clips. For example, you can play, pause, rewind, display, stop, and hide video.
Using Timeline Control Actions & ActionScript 2

Now that you have a sense for how ActionScript works, let’s explore the actions in-depth. Some of the most useful actions relate to controlling elements of the Timeline.

**Go To.** There are a number of actions that tell your movie to go to a different frame and do something.

```javascript
gotoAndPlay();
gotoAndStop();
nextFrame();
nextScene();
play();
prevFrame();
prevScene();
stop();
stopAllSounds();
```

With the `gotoAndPlay`, you need to indicate the frame. You can define the frame with a frame number such as (50), frame label such as (zion), expression such as a calculation, next frame, or previous frame. Use this option when you want to go to a frame and begin playing an animation.

With `gotoAndStop`, you also need to indicate the frame. This option is used when you’re going to wait for user input. For example, the user will read and make a choice before continuing.

If the action will take place in response to a mouse state, then you’ll use an On (mouse event) statement before the action such as

```javascript
on (release) {
  gotoAndPlay(2);
}
```

**Using Other Actions for ActionScript 2.0**

If you look at the topics in the Global Functions area, you’ll find many other actions. Below are some of the most common actions. You can always use the Flash help to learn more actions.

**getURL.** We already used the `getURL` action. It’s used for launching a website. Use `_self` to open the new Web page in the current window. Use `_blank` for a new browser window.

```javascript
getURL("http://eduscapes.com","_blank");
```
Creating Presentations and Exhibits

One of the most common applications of Flash is the creation of an informational presentation or exhibit. These projects may be linear and simply move from slide to slide. Or, they may be branched and allow users to choose the order of their viewing.

To create a presentation:
Create a new Flash document and save it as present1.
Name your first layer, information.
Change the background color (Modify>Document).
Create a text box for the title text and a text box for the body text.
Create a new layer called navigation.
Create a button symbol to be used as our navigation button.
Add an instance of your button to the Stage.
Add the following ActionScript to the button:

```
on (release) {
    gotoAndStop (2);
}
```

Add a new layer called actions.
Add a stop action script.

```
stop();
```

Add additional Keyframes to your information layer for the text and graphics you’ll need on each different frame/slide.
Add additional Keyframe to your navigation layer for the navigation button you’ll need on each frame/slide.
Select the Keyframe for the Frame 2 navigation layer.
Change the ActionScript for the instance of the button on Frame 2, so on Frame 2 you’ll change the gotoAndStop to Frame 3.
Go to each frame of the navigation layer and change the button script.

Using Objects in ActionScript 2.0

ActionScript makes it easy to call on the objects you create such as buttons, text, movie clips, and sounds. When you give an object a name, you can call on that object when you need it. You specify the object by name, followed by a dot (.), the method, and the parameters.
Creating Draggable Objects in ActionScript 2.0

It’s fun to drag objects around the screen and Flash makes it easy! Draggable objects are a great way to involve people in building things. For example, you can give users visuals and ask them to create a collage. Or, give users houses, cars, and roads and have them build a town. Ask students to sort objects into categories. How about a fun game of magnet poetry? Provide the words and ask users to write their own poems. You could even add a print button so users could print their work! Here are some other ideas:

- Drag healthy food into a bag, mouth, table.
- Drag words into an order, categories, matching picture
- Drag puzzle pieces
- Drag tools to perform operation, solve problem, etc.
- Drag to reveal answers underneath visual
- Drag letters for a scrabble game, hangman
- Drag words for magnet poetry, fill-in-blank
- Drag objects to create a snowperson, dressed person, Christmas tree, Gingerbread house, decorated cookie, pizza

Let’s create the outline for a snowperson and ask users add the eyes, nose, and hat. Be sure you’ve chosen ActionScript 2.0.

To create a draggable symbol:

Create a new project.
Pull down the Insert menu, choose New Symbol.
Choose movie clip behavior and OK.
Create an object such as an eye, nose, or hat.
Drag the movie clip that you created from the Library to the Stage.
Select the object.
In the Property inspector, type in a name for the object such as snow1, snow2, snow3, etc.
Create as many objects as you wish.
  My snowperson has five black dots, a hat, a carrot, and a red scarf.
Select the first frame in a layer of your movie.
Pull down the Window menu, choose Action panel.
Enter the following code adding a new section for each draggable object. My examples uses snow1 through snow8 for the eight pieces children will drag to the snow person:

```javascript
snow1.onPress=function () {
  this.startDrag () ;
}
snow1.onRelease=stopDrag;
```

```javascript
snow2.onPress=function () {
  this.startDrag () ;
}
snow2.onRelease=stopDrag;
```
Create the body of the snowperson and add some directions.
Pull down the Control menu, choose Test Movie.
  Enjoy dragging objects around on the screen!

**Detecting Drop Position in ActionScript 2.0**

The simple drag-and-drop approach works great for many projects. However if you want to use it as part of a game or activity, you may want to detect the drop position and take some action, so the user knows that they found the correct position.
Let’s ask our users to put a circle inside a square. We’ll create a circle symbol and add an ActionScript so we can drop it on a square target. Be sure you’ve chosen ActionScript 2.0.

To detect a drop:
Create a new file and save it as *dropit*.
Create a new movie clip symbol called *circledrop*.
Add the following ActionScript to this symbol:

```actionscript
on (press) {
    this.startDrag();
}
on (release, releaseOutside) {
    this.stopDrag();
}
```

Rename the layer, *circle*.
Drag an instance of the *circledrop* symbol to the Stage.
Create another layer called *square*.
Create a new movie clip symbol called *squaretarget*.
Drag an instance of the *squaretarget* to the Stage.
Create another layer called *action*.
Add the following ActionScript to Frame 1 of the *action* layer.

```actionscript
stop();
// start drop detective
circledrop.onPress = function() {
    this.startDrag();
};
circledrop.onRelease = function() {
    this.stopDrag();
    if (this._droptarget == "\squaretarget") {
        circledrop._alpha = 50;
    } else {
        this._x = 400;
        this._y = 40;
    }
};
```
Creating an Interactive in ActionScript 2.0

An interactive presentation contains a number of ways to navigate. Let’s create a presentation on the topic of Utah National Parks. The first slide will contain an overview. Five additional slides will provide detailed information about each park. In addition to previous and next buttons, buttons on a state map will take users to their choice of parks.

In addition to the interactive navigation, the slide show will also use ActionScript to enter information into the title and body area text on each page. We’ll need to set the properties of the text boxes to match the action. There are three types of text boxes: static, dynamic, and input.

A **static** box is used to display text that won’t change during the project.

A **dynamic** text box is used when information may be generated or modified using an ActionScript.

An **input** text box is used when users will be typing information into the text box.

If you’ll be using a text box for dynamic or input information, you should give the box a name. Be sure you’ve chosen ActionScript 2.0.

To set up the layers:
- Rename Layer 1, *map*.
- Draw a map of Utah.
- Draw a filled rectangle across the top and bottom of the screen.
- Create a new layer and name it *text*.
- Create text boxes for the title and each of the five national parks.
- Create a new layer and name it *photos*.
- Create a new layer and name it *parkinfo*.
- Create a text box for the park title and body text.
- In the Properties panel, rename the title textbox, *parktitle*, and the body textbox, *parkbody*.
- Change Static Text to Dynamic Text.
- Create a new layer and name it *action*.
- Create a button symbol for each of the national parks.
- Create a next and previous button symbol.

To enter the ActionScripts:
- Click on Frame 1 of the action layer.
- Open the Actions panel and enter the following script:

```javascript
parktitle.text="Welcome";
parkbody.text="Utah is a beautiful state featuring five national parks. This automated tour will highlight each of these natural wonders. ";
stop();
```
Select the Zion button. Open the Action panel and enter the following script:

```javascript
on (release) {
    gotoAndStop (2);
}
```

Create actions for each of the other buttons going to the correct Frame number.
Select the Previous button. Open the Action panel and enter the following script:

```javascript
on (release) {
    prevFrame();
}
```

Select the Next button. Open the Action panel and enter the following script:

```javascript
on (release) {
    nextFrame();
}
```

Click on Frame 2 of the action layer. Open the Action panel and enter the information for the parktitle and parkbody for Zion National Park. Enter the text information on each of the additional frames.
Try it!
Keep in mind that the text won’t appear in the text areas until you test it.
Building a Quiz in ActionScript 2.0

Whether you’re creating a formal quiz or an interactive game, Flash has tools to meet your needs. Many of the objects and ActionScripts can be reused depending on your needs. More involved projects require more detailed planning. It’s a good idea to create a storyboard containing sketches of the frames containing the main action. You’ll also want to keep track of your layers, symbols, text boxes, and named variables.

Let’s go step-by-step through a sample quiz. In this case, invisible buttons cover answers. When the correct answer is clicked, the student goes to the next problem. If the incorrect answer is clicked, students are told to try again.

Although you might want to create a project based on my healthy foods topic, consider following the procedure with your own topic. You can also open the original foodquiz.fla file if you want to copy the scripts or just edit the existing project.

Setting Up the Quiz

To download the foodquiz go to:

http://eduscape.com/flash/quiz/foodquiz.fla

Our project will have eight frames. However you can make your project as many frames as you wish. The first page contains a standard title and graphic. It also contains a button from the Flash Common Libraries collection. An ActionScript is added that takes the user to Frame 2.

An ActionScript declares the variables that will be used in the project. It’s a good idea to do this on your very first frame. Variables are temporary storage places for words or numbers. In this case, we’re going to store the name of the student as a variable called yourName. If you were keeping track of the student’s score, you could also declare that variable here.

Let’s work with Frame 1. Be sure you’ve chosen ActionScript 2.0.

To set up the first page:
Create any introductory text or graphics.
Remember to keep elements in separate layers.
Create a button and set the ActionScript to go to Frame 2.

To set up a name variable:
Create, then select the actions layer of Frame 1.
Open the Action panel and enter the following script:

```
var yourName;
yourName = "";
stop();
```
This script declares the variable called yourName that will be used to store the user’s name. It also empties out any other versions of this variable. Then, the frame stops and waits for user action.

**Using Static, Dynamic, and Input Text Boxes**

On the next couple frames we’re going to use the three different types of text boxes: static, dynamic, and input. Using Static, Dynamic, and Input Text Boxes. If you’ll be using a text box for dynamic or input information, you should give the box a name. You’ll also want to include the variable name under Var:

Let’s work with Frame 2. On this frame, you’ll want any introductory graphics along with a static text box asking for students to enter their name. You’ll also create an empty input text box where students can write their name. I drew a box around the text input box to make it easier for users to see.

To create the text input box:
- Create and select a new name layer.
- Create a text box.
- In the Properties panel,
  - Set the name to *myName*.
  - Set the var to *yourName*.
- Select the actions layer of Frame 2.
Frame 1: Intro text and graphics; declare variables; button to next frame.

Frame 2: Text and graphics; store student name from input text box; button to next frame.

Frame 3: Intro graphics; generated welcome message with user name in dynamic textbox; button to next frame.

Frame 4-6: Generated question in dynamic text box; four graphic symbols; three incorrect buttons with generated feedback in dynamic text box, one correct button with script to go to next frame.

Frame 7: Intro text and graphics; generated goodbye message with user name in dynamic text box.
Open the Action panel and enter the following script:

```actionscript
stop();
Selection.setFocus("_root.yourName");
```

This tells Flash to store the user’s name.
Edit the arrow button to go to Frame 3.

Frame 3 will make use of the student’s name in a welcome message.

To respond to user input:

Create and select a new `textarea` layer.
Select Frame 3 of the `textarea` layer.
Insert a Keyframe.
Create a text box.
Create a text box.
In the Properties panel,
Set the name to `inputName`.
Set the var to `getName`.
Set to Multiline.
Select the `actions` layer of Frame 3.
Open the Action panel and enter the following script:

```actionscript
getName = yourName + "Let’s have some fun!";
stop();
```

This tells Flash to put the user’s name along with the welcome message in the text box called `getName`.
Edit the arrow button to go to Frame 3.

**Creating Interactive Question Frames in ActionScript 2.0**

The next series of Frames (4-6) will present the questions and ask students to respond. We created two invisible button symbols called `correctanswer` and `incorrectanswer` that we can reuse. In this project, we simply cover graphics that represent the student’s multiple choices. If you prefer, you can use these invisible buttons to cover text or elements of a single large graphic.

To create the look for the question frames:
Insert Keyframes and get rid of the introductory graphics.

To create a dynamic text area for the question:
Create a new `question` layer.
Select Frame 4 of the `question` layer.
Insert a Keyframe.
Create a text box.
In the Properties panel,
  Set the name to question.
  Set to Multiline.
Open the Action panel and enter the following script:

  question.text="Click on the food that belongs to the grain group. ";
  stop();

To create the graphics layer for the possible answer graphics:
  Create a new graphics layer.
  Select Frame 4 of the graphics layer.
  Insert a Keyframe.
  Drag instances of the multiple choice options for the question.

To create the answer layer for the invisible answer buttons:
  Create a new answer layer.
  Select Frame 4 of the answer layer.
  Insert a Keyframe.

To create the correct answer button:
  Select the answer layer of Frame 4.
  Create and drag an instance of the button symbol called correctanswer.
  Open the Action panel and enter the following script:

  on (release) {
      gotoAndPlay(5);
  }

  The script will need to be revised to go to the correct frame.
  If you wanted to add a score aspect, you could add to a cumulative score here.

To create the incorrect answer button:
  Select the answer layer of Frame 4.
  Create and drag an instance of the button called incorrectanswer.
  Open the Action panel and enter the following script:

  on (release) {
      feedback.text="The correct answer is bread. Try Again.";
  }

  The script will need to be revised to go to the correct frame.

To create a dynamic text area for the feedback:
  Create a new feedback layer.
  Select Frame 4 of the feedback layer.
  Insert a Keyframe.
Create a text box.
In the Properties panel,
  Set the name to *feedback*.
  Set to Multiline.

To create the other question frames:
Create Keyframes for the graphics layers and rearrange the graphics as needed for each frame.
Create Keyframes for the answer layers on each frame and edit the scripts for the correct and incorrect answer buttons.
Edit the ActionScript on the action layers for the new questions.

The final Frames provide a conclusion. In this case, we’ll use the user’s name again in the goodbye. Be sure to set Keyframes as needed and add or remove elements.

**Repurposing ActionScripts**

We’ve explored the key ActionScript coding that you’ll need to develop most projects. However there are specific times, where you’ll need to find additional ActionScripts to meet your needs. Rather than creating these from scratch, you can often repurpose coding. In other words, take existing code and adapt or modify it to meet your needs.

For example, you don’t have to build a project from scratch. The *Exploring Primary Source Materials* project is based on the Food Quiz. In other words, I simply opened the Food Quiz and modified the background, text, buttons, and ActionScripts. Rather than a quiz, this project is a tutorial that introduces three words: digital reproduction, transcription, and interpretation. Instead of a quiz format, the same transparent buttons are used to promote thinking within the tutorial.

**Exploring ActionScript 3.0**

Although you may wish to use ActionScript 2.0 for many of your project, consider a move to Actionscript 3.0. It’s a more complex authoring language, but it’s also more powerful. Flash is supporting a number of tools that will make using ActionScript 3.0 easier for beginners.

ActionScript is an object-oriented programming language. Rather than one long string of code, smaller chunks of programming are used for specific tasks. These chunks are reusable in a variety of situations. The key is telling the system when and were to use these chunks.

Let’s explore some of the basic terminology:
- **Classes.** A class is the general object such as a button or graphic symbol.
- **Instances.** An instance is a specific object such as the start button that appears on the first frame.
Exploring Primary Source Materials

A primary source is a piece of information created from direct experience and often used for understanding history. These sources include actual records and artifacts that have survived from the past such as diaries, letters, photographs, articles of clothing, or coins.

The photo on the right was taken about 100 years ago. It shows my great, great, aunt, Laura May Wilson with her bike.

Primary sources are created by people who witnessed or participated in an event and recorded it in some way. They also include any informational item created in the past such as a newspaper advertisement from the 1940s, political cartoon from the 1920s, or recipe from the 1800s.

Press the arrow button to continue...

Exploring Primary Source Materials

Primary resources provide interesting insights into people and places. The key to effective use of primary resources is careful observation and interpretation. Many times a primary source only provides part of a larger story. By examining and comparing the information from many of these resources, it's possible to gain a better understanding of the people, places, and technologies of the past.

This is a marriage certificate.

Think of all the things you could learn from this single document.

Write your name in the box. Then, continue. Annette

Exploring Primary Source Materials

Great job! You found the digital reproduction. Transcription is the conversion of one form of language into another such as handwritten letters into typewritten documents. It can also be process of matching the sounds of human speech to a text format. Many historical primary resources are transcribed into a digital form to make them easier to access and search.

Wednesday, January 1

Up to Edna's all day. Gertrude's, Lillian's and Lucille's and we were there. At 11:45 read book and listened to Gracie Allen. "The Music goes Round & Round."

Great job! You found the digital reproduction. Transcription is the conversion of one form of language into another such as handwritten letters into typewritten documents. It can also be process of matching the sounds of human speech to a text format. Many historical primary resources are transcribed into a digital form to make them easier to access and search.

Wednesday, January 1

Up to Edna's all day. Gertrude's, Lillian's and Lucille's and we were there. At 11:45 read book and listened to Gracie Allen. "The Music goes Round & Round."

Press this arrow button to continue...
• **Properties.** Properties are the characteristics of an object such as a button that’s set to the color red or a text frame that’s set to dynamic text.

• **Methods.** Methods are actions that an object can perform such as playing an audio clip.

• **Events.** Events are the triggers that start an event such as playing a video clip.

• **Encapsulation.** Encapsulation are those activities that occur without the end user realizing that things are happening. They are happening “behind the scenes.”

Because you’ll be making reference to specific instances of objects, get accustomed to naming each instance in the instance name box in the property panel. If you completed the activities associated with the Utah and Food projects you’ll remember the importance of naming your objects.

Before we jump into programming, let’s explore some of the aspects of the Actions panel that you might not has noticed earlier. The toolbar provides some useful resources. For example, the Check Syntax button is useful in avoiding coding errors and the Auto Format button will automatically clean up the look of your coding work.

**Changing Commands from ActionScript 2.0 to ActionScript 3.0**

Let’s explore the airplane example using the ActionScript 3.0 option. Open your project created in ActionScript 2.0. Go to Publish Settings>Flash option and change to FlashPlayer 10 and ActionScript 3.0. If you change the settings, you’ll receive a message that says: “WARNING: Actions on buttons or MovieClip instances are not supported in ActionScript 3.0. All scripts on object instances will be ignored.”

We’re going to update the scripts, so don’t let the warning worry you. One of the big changes with the new version is that all scripts are now placed at the Timeline rather than on individual objects. The *stop* action that you placed on the first and last frames should work fine because those actions are on the Timeline. However changes will need to be made to the *start* and *flight* buttons.
You should already have a layer where you’ve been placing scripts called `action`. We’ll use this for our `start` and `flight` buttons. It’s a good idea to put your action layer at the very top of your Timeline for easy access.

Since scripts are no longer placed on the objects, you’ll use event listeners to check for situations such as mouse events that will trigger the action. This is more complicated than the old way, but works better overall.

Although you can use the number of a frame, it’s good practice to begin naming your frames because you may want to add frames within your project. When you add frames, your frame numbers will be revised. For instance if you add a new frame at the very beginning of your project, then frame 1 becomes frame 2 and frame 2 becomes frame 3. Instead, name your first first frame “title” and your second frame “intro”. Name other important frames such as “chapter1,” “credits” and “end”. Remember to keep your names to a single word.

To remove the old scripts:

Open your airplane project.

Open the Action panel.
Click on the start button in the column on the left.
Remove the on three lines of code.
Also, remove the code from your flight button.

To revise the airplane start button action:

Move your action layer to the top of the Timeline.

Go to Frame 2 on the Timeline, insert a Keyframe and name this frame: `start`.
You’ll notice that a red flag appears on the frame indicating that this frame has the name start.
Go to Frame 1 of the action layer on the Timeline, open the Action panel, and add the following script:

```
stop();
btnStart.addEventListener(MouseEvent.CLICK, clickStartListener);

function clickStartListener(evt:MouseEvent) {
gotoAndPlay("start");
}
```

Keep in mind that ActionScript 3.0 is VERY PICKY. It is case sensitive, so be sure to check your use of upper and lowercase.
The syntax can be tricky, so be sure to use the Syntax checker if you think you might have an error. The results will show under the Compiler Error tab. In the example below, a colon was incorrectly placed at the end of the line. Although it doesn’t provide the solution, it does identify the error. The developer must figure out that they need to replace the colon with a semicolon to fix the problem.

The second button in our airplane example will also need to be changed. This button is on Frame 120 and links to a website.

To revise the airplane website link button:

Go to the website layer, click on the Flight button, name it button_flight.
Go to Frame 120 of the action layer and add a keyframe.
Enter the following script:

```
stop();

var link:URLRequest = new URLRequest("http://42explore.com/flight.htm");

button_flight.addEventListener(MouseEvent.CLICK, onClick);

function onClick(event:MouseEvent):void
{
    navigateToURL(link);
}

button_flight.buttonMode = true;
```

Test your project. Your button should now work.

Although most of the scripts that were written for ActionScript 2.0 can be adapted to work with ActionScript 3.0, there are some activities that are much more difficult. For instance, there’s no direct translation for the drag and drop projects.
Adding Components ActionScript 2 & 3

Flash has many cool features that are already built in. Components are pre-coded actions that make interactivity easy. These reduce the amount of time that it takes to code your own projects. You can also add sophisticated features without much coding experience. These features can be found on the Components panel. Components are available for both ActionScript 2 and ActionScript 3.

To explore the components:
Pull down the Windows menu, choose Components.
You’ll find features in five categories.
Drag the component you’d like to use onto the Stage.

Explore Components for ActionScript 2 Projects

In some cases, the component works immediately. In other cases, you’ll need to add an ActionScript for the object to be meaningful.
Let’s try the Button component as an example.

To use the Button component:
Create a new Flash document.
Select Frame 1 on your Timeline.
Pull down the Windows menu, choose Actions.
Add a stop to this frame.

    stop();

Pull down the Windows menu, choose Components.
Go to the User Interface area, drag the Button onto the Stage.

Use the Free Transform Tool to resize the button.
Pull down the Window menu and select the Properties & Parameters.
A panel will appear.
Change the label of the button from “Button” to Go!, Start, Previous, Next, or whatever you wish.
The label placement can also be changed to left, right, top, or bottom.

To add a behavior to the button component:
Pull down the Window menu and select Behaviors.
A panel will appear.
Click the “+” sign and a drop-down menu will appear.
Add any action you wish to this button. You can embed movies, play sounds, go to websites, or go to a particular frame. Choose Web and Go to Web Page. Enter the name of a website such as http://cnn.com. This Behavior will now show up on the list. An ActionScript has automatically been added to this button. You can always go back and edit the Behavior later.

Try It!

**Explore Components for ActionScript 3 Projects**

There are many components that will make Flash project production much easier such as alerts, menus, text boxes, and media objects. This is particularly important when using ActionScript 3. Before trying to create your own scripts, see if a component is already available.

Let’s say you want to create a quiz or survey. Use the built-in checkbox.

To use the Components:

- Pull down the Windows menu, choose Components.
- Drag the Checkbox component onto the Stage.
- Pull down the Windows menu, choose Component Inspector.
- Click next to the word label to change the name to healthy or unhealthy.
- Notice that the default attributes have already been established.

**Create an Interactive Project**

Try an interactive project of your own. Start with basic programming such as movement between frames. Then, add in some advanced activities such as inserting text or checking answers. Rather than starting from scratch, look for components that will help.
Incorporate interactive elements into a Flash project. Incorporate at least three of the following elements:

- Create a unique appearance for each of the four button states (up, over, down, hit)
- Use invisible buttons
- Use draggable symbols
- Use item from Component panel

Try some scripting in ActionScript 2.0. Incorporate at least three of the following ActionScript functions: stop, gotoAndPlay, gotoAndStop, getURL, nextFrame().

Or, create the equivalent project in ActionScript 3.0 using the examples from the chapter.

Be sure to make effective use of layers.

Save and upload both the .fla and .swf versions. In addition, write a short project plan to describes the purpose and audience for your interactive elements. In other words, your project should include interactive elements that serve an entertainment, informational, instructional, or persuasive purpose.

The following criteria will be used in evaluating your activity:
- Interaction elements
- ActionScript elements
- Project Plan
- Effective, efficient, and appealing project