Adobe

Adobe® Edge Animate

CLASSROOM IN A BOOK®

The official training workbook from Adobe Systems

CD-ROM Included for Windows and Mac OS

Adobe

Adobe® Edge Animate

CLASSROOM IN A BOOK®

The official training workbook from Adobe Systems

CD-ROM Included for Windows and Mac OS

Adobe® Edge Animate Classroom in a Book®

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110-2704, USA

Adobe Press books are published by Peachpit, a division of Pearson Education located in San Francisco, California. For the latest on Adobe Press books, go to www.adobepress.com. To report errors, please send a note to errata@peachpit.com. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Acquisitions Editor: Rebecca Gulick
Writer: Russell Chun
Development and Copy Editor: Stephen Nathans-Kelly
Production Coordinator: Myrna Vladic
Compositor: David Van Ness
Technical Reviewer: Joseph Labrecque
Keystroker: H. Paul Robertson
Proofreader: Patricia Pane
Indexer: Rebecca Plunkett
Cover Designer: Eddie Yuen
Interior Designer: Mimi Heft

Printed and bound in the United States of America

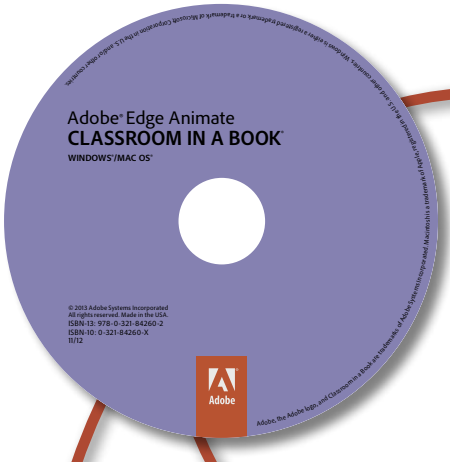
ISBN-13: 978-0-321-84260-2
ISBN-10: 0-321-84260-X

9 8 7 6 5 4 3 2 1

WHAT'S ON THE DISC

Here is an overview of the contents of the Classroom in a Book disc

The *Adobe Edge Animate Classroom in a Book* disc includes the lesson files that you'll need to complete the exercises in this book, as well as other content to help you learn more about Adobe Edge Animate and use it with greater efficiency and ease. The diagram below represents the contents of the disc, which should help you locate the files you need.



Lesson files

Each lesson has its own folder inside the Lessons folder. You will need to copy these lesson folders to your hard drive before you can begin each lesson.

Online resources

Links to Adobe Community Help, product Help and Support pages, Adobe Press, Adobe certification programs, Adobe TV, and other useful online resources can be found in a handy HTML file. Just open it in your web browser and click the links, including a special link to this book's product page where you can access updates and bonus material.

Adobe Press

ADOBE PRESS

Adobe

Find information about other Adobe Press titles, covering the full spectrum of Adobe products, in the Online Resources file.

CONTENTS

INTRODUCTION

Adobe Edge Animate is a new tool that provides a comprehensive authoring environment for creating animated, interactive, and media-rich content for the Web. Based on open, modern browser standards using HTML5, CSS3, and JavaScript, your Edge Animate creations run seamlessly across desktops, smartphones, and tablets. There is no need for the Flash Player, Silverlight, QuickTime, or downloading of any apps.

Use Edge Animate to build animated banner ads, dynamic websites, and even interactive games with the full capabilities of JavaScript.

Veteran Flash Professional users and animators will feel at home with a familiar interface consisting of a Timeline, Stage, and Library panel. They'll add motion to images and HTML elements such as text and simple shapes with property-based keyframing, easing, and nested animations. Coders at all levels of experience can add interactivity with the built-in code snippets or with JavaScript. With sophisticated, yet intuitive controls for development, and platform-independent content, Adobe Edge Animate will be sure to expand your creative reach.

Why Adobe Edge Animate?

Adobe Edge Animate represents the next step in the evolution of interactive and animated Web content development. With the growing adoption of HTML5 standards, modern browsers are now able to display rich media without the need for plug-ins, such as the Flash Player. In conjunction with CSS3 and JavaScript, Edge Animate enables users to integrate animation and complex interactivity for stunning visuals and engaging user experiences.

Unfortunately, the rapid rise in the popularity of HTML5, CSS3, and JavaScript has not coincided with the emergence of tools specifically for creative professionals. Coding motion graphics and interactive content by hand has been the usual course, but that approach takes time and effort, and for designers and animators who are more accustomed to graphical user interfaces, it's more difficult. Adobe Edge Animate opens the door to designers and animators by providing an intuitive interface and a familiar toolbox: tools for creating shapes, options for styling, transformations, precision layout, and typography,

and a timeline with keyframe controls for motion graphics. Rather than spend time on coding, designers and animators can spend their energies on what they do best: designing and animating.

The Adobe family

How does Adobe Edge Animate differ from other Adobe tools for the Web? Although there are overlaps in capabilities, each application has its own strengths and support different technologies. Edge Animate shines particularly when it comes to creating motion graphics and interactive sites with HTML5, CSS3, and JavaScript. Adobe Dreamweaver, another application that creates Web content with HTML5 and CSS3, is intended more for overall site design and navigation. For example, you would create an animated banner ad in Edge Animate, and use Dreamweaver to integrate the banner ad within the larger site.

Adobe Flash Professional and Flash Builder are two other tools for creating animated and interactive content. However, both rely on the ActionScript programming language rather than HTML, CSS3, and JavaScript for interactivity. They both require the Flash Player or Adobe AIR to play Flash content. The Flash Player, although pervasive (it comes pre-installed with Google's Chrome browser), is not supported in all devices, and is not an open standard. However, the benefits of using the Flash Player or AIR are a uniform experience across all browsers, and the delivery of robust control—for example, with controlling your webcam or saving files to your desktop.

About Classroom in a Book

Adobe Edge Animate Classroom in a Book is part of the official training series for Adobe graphics and publishing software developed with the support of Adobe product experts. The lessons are designed so you can learn at your own pace. You'll learn the fundamental concepts and features you'll need to use the program.

Classroom in a Book also teaches many advanced features, including tips and techniques that will help you get the most out of Adobe Edge Animate.

Prerequisites

Before you begin using *Adobe Edge Animate Classroom in a Book*, make sure your system is set up correctly and that you've installed the required software. You should have a working knowledge of your computer and operating system. You should know how to use the mouse and standard menus and commands, and also

how to open, save, and close files. If you need to review these techniques, see the printed or online documentation included with your Microsoft Windows or Apple Mac OS software.

Installing Edge Animate

You must purchase the Adobe Edge Animate software as a download from Adobe Creative Cloud. The following specifications are the minimum required system configurations:

Windows

- Intel® Pentium® 4 or AMD Athlon® 64 processor
- Windows 7 or Windows Vista® (Windows XP is NOT supported)
- 1 GB of RAM
- 200 MB of available hard-disk space for installation
- 1280x800 display with 16-bit video card
- Broadband Internet connection for online services and to validate Subscription Edition (if applicable) on an ongoing basis.

Mac OS

- Multicore Intel processor
- Mac OS X v10.6 and v10.7 (Mac OS X 10.5 is NOT supported)
- 1 GB of RAM
- 200 MB of available hard-disk space for installation
- 1280x800 display with 16-bit video card
- Broadband Internet connection for online services and to validate Subscription Edition (if applicable) on an ongoing basis.

For updates on system requirements and complete instructions on installing the software, visit http://adobe.com/edge.

Copying the lesson files

The lessons in *Adobe Edge Animate Classroom in a Book* use specific source files, such as image files created in Adobe Photoshop and prepared Edge Animate documents. To complete the lessons in this book, you must copy these files from the

Adobe Edge Animate Classroom in a Book CD-ROM to your hard drive. Follow these steps to copy the lesson files:

1 On your hard drive, create a new folder in a convenient location and name it **Edge_Animate_CIB**, following the standard procedure for your operating system:

- **Windows:** In Explorer, select the folder or drive in which you want to create the new folder and choose File > New > Folder. Then type the new name.
- **Mac OS:** In the Finder, choose File > New Folder. Type the new name and drag the folder to the location you want to use.

2 Drag the Lessons folder (which contains folders named Lesson01, Lesson02, and so on) from the *Adobe Edge Animate Classroom in a Book* disc onto your hard drive to your new Edge_Animate_CIB folder.

When you begin each lesson, navigate to the folder with that lesson number to access all the graphics, images, and other project files you need to complete the lesson.

If you have limited storage space on your computer, you can copy each lesson folder as you need it, and then delete it after you've completed the lesson if desired. Some lessons build on preceding lessons. In those cases, a starting project file is provided for you for the second lesson or project. You do not have to save any finished project if you don't want to or if you have limited hard-drive space.

Copying the sample projects

You will create and publish HTML files and related JavaScript files in the lessons in this book. The files in the End folders (01End, 02End, and so on) within the lesson folders are samples of completed projects for each lesson. Use these files for reference if you want to compare your work in progress with the project files used to generate the sample movies. The end project files vary in size, so you can either copy them all now if you have ample storage space or copy just the end project file for each lesson as needed. Then you can delete it when you finish that lesson.

How to use the lessons

Each lesson in this book provides step-by-step instructions for creating one or more specific elements of a real-world project. Some lessons build on projects created in preceding lessons; most stand alone. All the lessons build on each other in terms of concepts and skills, so the best way to learn from this book is to proceed through the lessons in sequential order. In this book, some techniques and processes are explained and described in detail only the first few times you perform them.

The organization of the lessons is also project-oriented rather than feature-oriented. That means, for example, that you'll work with symbols on real-world design projects over several lessons rather than in just one chapter.

Note: Many aspects of the Edge Animate application can be controlled by multiple techniques, such as a menu command, a button, dragging, and a keyboard shortcut. Only one or two of the methods are described in any given procedure so that you can learn different ways of working, even when the task is one you've done before.

Additional resources

Adobe Edge Animate Classroom in a Book is not meant to replace documentation that comes with the program or to be a comprehensive reference for every feature. Only the commands and options used in the lessons are explained in this book. For comprehensive information about program features and tutorials, please refer to these resources:

Adobe Community Help: Community Help brings together active Adobe product users, Adobe product team members, authors, and experts to give you the most useful, relevant, and up-to-date information about Adobe products.

To access Community Help: To invoke Help, press F1 or choose Help > Edge Animate Help.

Adobe content is updated based on community feedback and contributions. You can add comments to both content or forums—including links to web content, publish your own content using Community Publishing, or contribute Cookbook Recipes. Find out how to contribute at www.adobe.com/community/publishing/download.html.

See http://community.adobe.com/help/profile/faq.html for answers to frequently asked questions about Community Help.

Adobe Edge Animate Help and Support: http://helpx.adobe.com/edge-animate/topics.html, where you can find and browse Help and Support content on adobe.com.

Adobe Forums: http://forums.adobe.com lets you tap into peer-to-peer discussions, questions, and answers on Adobe products.

Adobe TV: http://tv.adobe.com is an online video resource for expert instruction and inspiration about Adobe products, including a How To channel to get you started with your product.

Adobe Design Center: http://www.adobe.com/designcenter offers thoughtful articles on design and design issues, a gallery showcasing the work of top-notch designers, tutorials, and more.

Adobe Developer Connection: http://www.adobe.com/devnet is your source for technical articles, code samples, and how-to videos that cover Adobe developer products and technologies.

Resources for educators: http://www.adobe.com/education includes three free curriculums that use an integrated approach to teaching Adobe software and can be used to prepare for the Adobe Certified Associate exams.

Also check out these useful links:

Adobe Marketplace & Exchange: https://www.adobeexchange.com/ is a central resource for finding tools, services, extensions, code samples, and more to supplement and extend your Adobe products.

Adobe Edge Animate product home page: http://html.adobe.com/edge/animate/.

Adobe Labs: http://labs.adobe.com gives you access to early builds of cutting-edge technology, as well as forums where you can interact with both the Adobe development teams building that technology and other like-minded members of the community.

Adobe certification

The Adobe training and certification programs are designed to help Adobe customers improve and promote their product-proficiency skills. There are four levels of certification:

- Adobe Certified Associate (ACA)
- Adobe Certified Expert (ACE)
- Adobe Certified Instructor (ACI)
- Adobe Authorized Training Center (AATC)

The Adobe Certified Associate (ACA) credential certifies that individuals have the entry-level skills to plan, design, build, and maintain effective communications using different forms of digital media.

The Adobe Certified Expert program is a way for expert users to upgrade their credentials. You can use Adobe certification as a catalyst for getting a raise, finding a job, or promoting your expertise.

If you are an ACE-level instructor, the Adobe Certified Instructor program takes your skills to the next level and gives you access to a wide range of Adobe resources.

Adobe Authorized Training Centers offer instructor-led courses and training on Adobe products, employing only Adobe Certified Instructors. A directory of AATCs is available at http://partners.adobe.com.

For information on the Adobe Certified programs, visit http://www.adobe.com/support/certification/main.html.

This page intentionally left blank

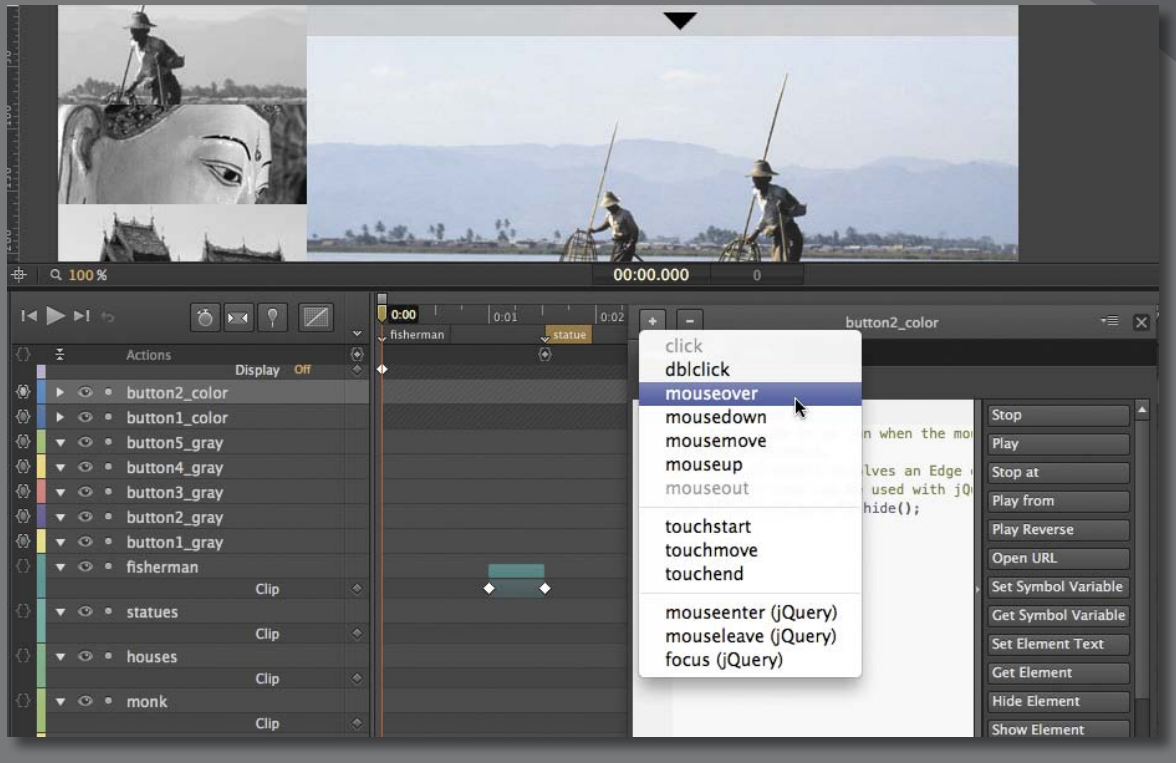
5 ADDING BASIC INTERACTIVITY

Lesson overview

In this lesson, you'll learn how to do the following:

- Understand interactivity
- Work with the syntax of JavaScript
- Recognize the relationship between JavaScript, jQuery, and the Edge Animate API
- Differentiate triggers, events, and actions
- Add triggers to the Timeline
- Insert labels
- Create actions to respond to events
- Control the behavior of the Timeline playhead
- View and edit script with the Code panel
- Use comments to annotate code
- Hide and show elements to incorporate visual feedback for buttons
- Control animated elements
- Customize the mouse cursor

This lesson will take approximately two hours to complete. If needed, remove the previous lesson folder from your hard drive and copy the Lesson05 folder onto it.



Let your viewers explore your composition and become active participants. Use Adobe Edge Animate's built-in code snippets and intuitive panels to add actions to create engaging, user-driven, interactive experiences.

Getting started

To begin, view the travel guide that you'll create as you learn to make interactive projects in Adobe Edge Animate.

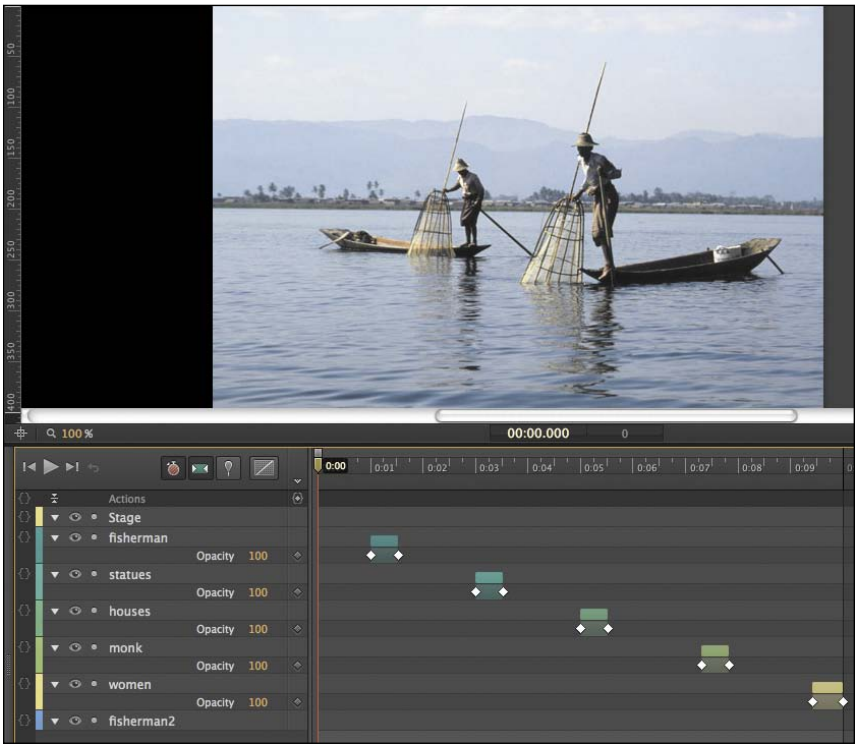
1 Double-click the 05End.html file in the Lesson05/05End folder to play the composition in a browser.

The project is an interactive photo gallery showing images from Myanmar. Viewers can watch the short slideshow automatically play and loop back to the beginning. Or, viewers can click any thumbnail image on the left side of the Stage to go directly to a particular image. Move your mouse over the triangular button at the top to see a caption unravel.

In this lesson, you'll create interactive buttons with rollover highlights and learn to incorporate the proper code that tells Edge Animate where to move the playhead on the Timeline to display the particular animation or image there.

2 Close the 05End.html file and quit your browser.

3 Double-click the 05Start Edge Animate file in the Lesson05/05Start folder to open the initial project file in Edge Animate.



The file includes all the assets already placed on the Stage and the transitions between each image on the Timeline. The Stage has already been sized properly. In this lesson, you'll make this linear animation interactive.

4 Choose File > Save As. Name the file **05_workingcopy** and save it in the 05Start folder. Saving a working copy ensures that the original start file will be available if you want to start over.

About interactive compositions

Interactive compositions change based on the viewer's actions. For example, when the viewer clicks a button, a different graphic with more information could appear. Interactivity can be simple, such as the click of a button, or it can be more complex, involving different kinds of interactions with the same element—for example, moving your mouse cursor over a button, clicking the button, and moving your mouse cursor off the button are three unique events that could each result in different visual changes on the Stage.

In Edge Animate, you use JavaScript to achieve interactivity. JavaScript is a popular and standard script for Web browsers. JavaScript runs on browsers for desktops as well as on devices such as tablets and mobile phones.

If you have no idea what JavaScript is, or how to write code—don't panic! Adobe Edge Animate provides an easy interface to add JavaScript to your compositions and integrate common interactive functions. When you get more comfortable with the syntax of the script, you can begin to delve deeper and customize the interactivity.

In this lesson, you'll learn to create nonlinear navigation, meaning the animation doesn't have to play straight from the beginning to the end, and stop there. You'll add code that gets triggered when the playhead reaches a certain point in time. You'll also add code that moves the playhead to different parts of the Timeline to display particular elements. You'll also learn to make elements on the Stage respond to different interactions with the mouse cursor.

Understanding JavaScript

JavaScript is the scripting language that adds additional functionality to a Web page. Many of the common interface elements on websites, such as pull-down menus, check boxes, or search boxes, are created with JavaScript. Edge Animate also uses JavaScript to power its interactivity, as well as the animations and other effects.

Where the JavaScript lives

Even without adding any interactivity to your composition, your project depends on JavaScript. The JavaScript code is contained in several separate text documents that have the file extension ".js". Look at the files associated with your Edge Animate composition, 05_workingcopy. There are four JavaScript files within the folder called edge_includes:

- edge.1.0.0.min.js
- jquery-1.7.1.min.js
- jquery.easing.1.3.js
- json2_min.js

These files contain the basic code required for any Edge Animate composition. There are also additional JavaScript files, which are unique to your project. Those files are located outside the edge_includes folder, and are automatically named with your Edge Animate filename. Your files are named as follows:

- 05_workingcopy_edge.js
- 05_workingcopy_edgeActions.js
- 05_workingcopy_edgePreload.js

When your Web browser first launches your Edge Animate project, it loads the JavaScript code so all the functionality is available when your project plays. All the code is organized as functions, which group commands together. Since each function has a unique name, you can trigger the commands simply by referencing the name of the function. Programmers say that a function is "called," or that the browser "calls" a function.

jQuery and the Edge Animate API

While JavaScript is useful, it's meant to control all the details of a Web page, which is powerful but often clumsy and complicated. That's where jQuery and the Edge Animate API come in handy. jQuery is an open-source JavaScript library that provides an easy way to select, control, and animate elements in a browser. jQuery is not another language, but simply a set of well-written JavaScript functions. If you look again at the JavaScript files in the edge_includes folder, you'll see that two of those files are, in fact, files for jQuery.

Along with jQuery, Edge Animate provides additional functions it has built for you. The library of JavaScript functions that Edge Animate has built for your use constitute the Edge Animate API (Application Programming Interface).

You can think of JavaScript, jQuery, and the Edge Animate API as different layers of control. The Edge Animate API is the top, most superficial layer of control, jQuery is the middle, and the core JavaScript is the deepest layer. A useful analogy is the control of an automobile. The Edge Animate API would represent the controls you see in the driver's seat—the steering wheel, the parking brake, or the gas pedal. They allow you to drive the car without needing to know much about its inner workings. They're created from a combination of levers, dials, and shafts to make

controlling your vehicle simple and easy. Those levers, dials, and shafts represent the jQuery level of control. At the most granular level, you have JavaScript, represented by the individual nuts and bolts and gears.

Just as it is so much easier to drive a car using the steering wheel and gas pedal, so it is to control your Edge Animate composition with the Edge Animate API. But in both cases, there's no reason why you couldn't tinker with the deeper-level controls for a more customized experience. You can start coding in jQuery and JavaScript to make your own interactivity. You just need to be sure you're a competent mechanic, or know your way around JavaScript!

In this lesson, you'll first learn to add interactivity with the Edge Animate API. Later, as you gain more confidence and comfort, you'll delve a little deeper and insert some jQuery for more sophisticated effects.

Triggers, events, and actions

Edge Animate uses actions, triggers, and events to incorporate JavaScript in your composition.

Actions are the things that Edge Animate can do, which, given the full JavaScript language at its disposal, is quite a lot. Actions can range from loading a hyperlink, to changing a particular visual property of an element on the Stage, to storing a piece of information in a variable for later retrieval.

Triggers are actions that are placed along the Timeline. When the playhead reaches the trigger, the actions are executed. Use triggers when you want code to be synchronized by your animation, and not by user control.

Events are things that happen in a composition that Edge Animate can respond to with an action. Typically, events are user-generated, such as the click of a mouse button, the pressing down of a key, or the tilting of a mobile device. However, events can also happen automatically. For example, the point when the composition is ready (when all the assets and code libraries have been downloaded) is an event. Events are always paired with actions. When an event happens, an action—or set of actions—is executed.

Timeline triggers

Triggers are the simplest way to add code to your Edge Animate composition. Triggers are executed automatically when the playhead reaches them on the Timeline. You can have multiple triggers along the Timeline. The minimum time interval between triggers is 1/1000th of a second, but practically, you'd never need or want actions to be executed so close together.

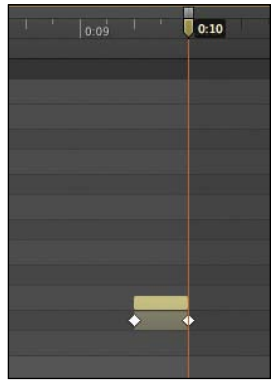
Creating a loop

For this slideshow of Myanmar, you'll insert a trigger at the end of the Timeline to make the playhead automatically return to the beginning, creating a loop.

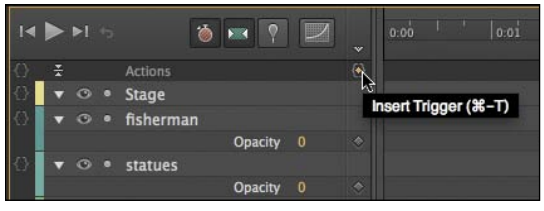
1 Click the Zoom Timeline to Fit button at the bottom of the Timeline.

The entire slideshow animation appears in the available space in the Timeline panel.

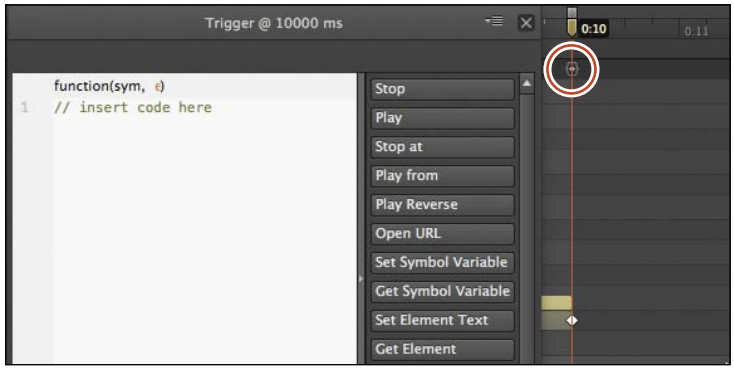
2 Move the playhead to the very end of the slideshow, at 0:10 seconds.



3 In the Timeline panel, click the Insert Trigger button on the top Actions row. You can also choose Timeline > Insert Trigger or press Ctrl+T (Windows) or Command+T (Mac OS).



An icon that appears as a diamond enclosed by curly braces appears on the Timeline at 0:10 seconds. The icon represents a trigger.

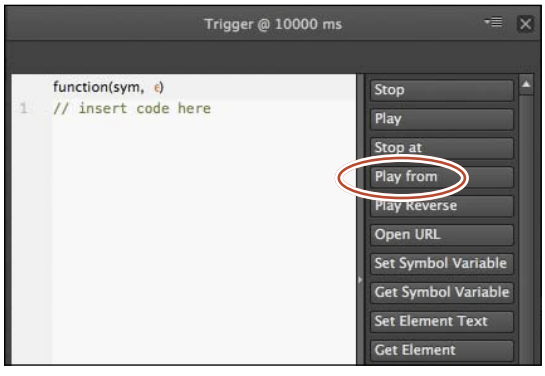


A panel appears with a large white text-entry field and a column of code snippet options on the right side. The panel is titled Trigger@10000ms, referring to the trigger's position at 10,000 milliseconds, or 10 seconds.

4 View the content of the current script.

The current script, `//insert code here`, is known as a comment. Comments always begin with two backslash characters, and are descriptions of the code. Comments are non-functional, and don't significantly add to the file size of your composition.

5 From the menu of code snippets on the right-hand side of the panel, select the Play from option.



New code appears in the panel after the existing comment. The new code comes with its own comments that describe its function. The code, `sym.play(1000);`, moves the playhead to a particular point in time on the Timeline and begins playing.

6 Replace the 1000 within the parentheses in the code with **0**.

The number in between the parentheses of the `play()` command represents the time to which the playhead will move. Since you want the playhead to move to the beginning of the Timeline at 0:00 seconds, enter **0** in the parentheses of the `play()` command.

7 Close the panel and preview your project in a browser by pressing Ctrl+Enter (Windows) or Command+Return (Mac OS).

The slideshow plays through and repeats when it reaches the end at 10 seconds.

Note: The number in between the parentheses of the `play()` command is called the *argument*. It gives the command additional information to make it more specific. In this case, it tells the command at what millisecond point in time to start playing. Commands can have multiple arguments, which are separated by commas. As you learn more commands, you also learn what arguments they require.

Editing triggers

Editing the script for your triggers is simple and easy. The panel that appeared when you added the trigger is always available for modifications, additions, or deletions.

- **To edit a trigger**, double-click the trigger icon on the Timeline.

  The script panel opens to display the trigger, and you can modify the argument, delete the code, or add new code from the snippet options on the right side of the panel.

- **To move a trigger**, click and drag the trigger icon on the Timeline to a new position.

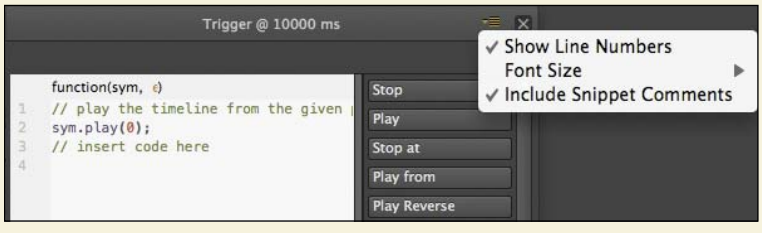
  The trigger moves to a different position, so the actions are executed when the playhead reaches a new time.

- **To delete a trigger**, select the trigger on the Timeline and press the Delete key.

  The trigger is removed from the Timeline.

Script panel viewing options

The script panel that opens when you add a trigger has several options to help you make viewing the code easier. The top-right options menu has three choices that control what's displayed.



- **Show Line Numbers** displays sequential numbers next to each line of code so you and other developers who share the script can pinpoint code.
- **Font Size** controls the display size (Small, Medium, or Large) of the text in the script. The default option is Small.
- **Include Snippet Comments** automatically adds comments to the code that you add from the snippet options on the right side of the panel.

In addition to these display options, you can click the vertical border that divides the white script area with the menu of code snippets to collapse the menu to make more room for your code. Click the divider again to expand the menu.

Minding your syntax

Let's examine the code that you added in the trigger more closely to learn about JavaScript *syntax*, or the way words and punctuation work together. Syntax is the grammar of a programming language.

If you're unfamiliar with program code or scripting, the JavaScript code that Edge Animate inserts may be challenging to decipher. However, once you understand the basic syntax, you'll find it easier to follow a script.

The code that is in your trigger at 10000 milliseconds appears as follows:

```
sym.play(0);
```

- The first word in the statement is `sym`. When the statement is on the main Timeline, the word `sym` represents the whole Edge Animate composition. Edge Animate is organized around the concept of "symbols," and the root, or base-level symbol, is the Edge Animate Stage. This root symbol contains all the elements and animations in your Edge Animate composition—everything on the Stage or Timeline. In JavaScript, when we want to do something, you first identify the object that you want to control. In this case, since you want to affect the Timeline of your Edge Animate composition, the first thing that is written in the script is `sym`.
- The *dot* operator (`.`) provides a way to access different commands for the object that you've identified (in this case, `sym`). Objects can be animations, text, or abstract concepts such as the date or particular data. Objects have *properties*, which describe them, and *methods*, which are the things that they can do. In your trigger, the method for the `sym` object is `play()`. The dot, or period, separates the object and its method.
- As in English, every open *parenthesis* must have a corresponding close parenthesis, and the same is true for JavaScript. If you open something, you must close it. All methods have parentheses. You method, `play()`, has an open and close parenthesis.
- Each method can have different *arguments* in between the parentheses, which provide additional information. The `play()` method requires a single argument, which tells Edge Animate at what point in time (in milliseconds) to begin playing. Methods can have multiple arguments, which are separated by commas.
- Some arguments require a number, some may need a name, and others may need the words `true` or `false`. Whenever you're entering the name of a file or a URL, use *single or double quotation marks*. Quotation marks distinguish a *String* value, which represent a sequence of characters, with other kinds of values.
- You can add *comments* to remind you or others of what you are accomplishing with different parts of the script. To add a comment for a single line, start it

with two slashes (`//`). To type a multiline comment, start it with `/*` and end it with `*/`. Comments are ignored by JavaScript and won't affect your code at all.

- The *semicolon* at the end of the line tells JavaScript that it has reached the end of a complete statement and the end of a line of code. A semicolon is like a period in a sentence.

  That's a lot of information packed into a single line of code! But getting comfortable with the syntax is your first step in getting out from behind the steering wheel and looking under the hood of your car.

Events and actions

So far, you've seen how Edge Animate uses triggers to automatically execute JavaScript when the playhead reaches a certain point on the Timeline. You added a trigger at the end of the animation to create a loop. The other two ways with which Edge Animate adds JavaScript is with events and actions.

Events are occurrences that happen in Edge Animate that it can detect and respond to. When an event is detected, you provide *actions* as a response.

It's useful to think of interactions in terms of events and actions. When you click on a menu button (event), more options may expand (actions). When you roll over a button (event), a triangular play icon may appear on it (actions). In the next section, you'll add thumbnail images to the Stage. When the user clicks on each thumbnail (event), the playhead will move to a new position on the Timeline (actions) to show a particular image from the travel slideshow.

Creating the buttons

A button is a basic visual indicator of what the user can interact with. The user usually clicks a button, but many other types of interactions are possible. For example, rollovers, double-clicks, and rollouts are all possible. Edge Animate also provides events unique to mobile devices, such as touches.

You'll start with the simplest, and most common event, which is the single-click.

Adding the thumbnails

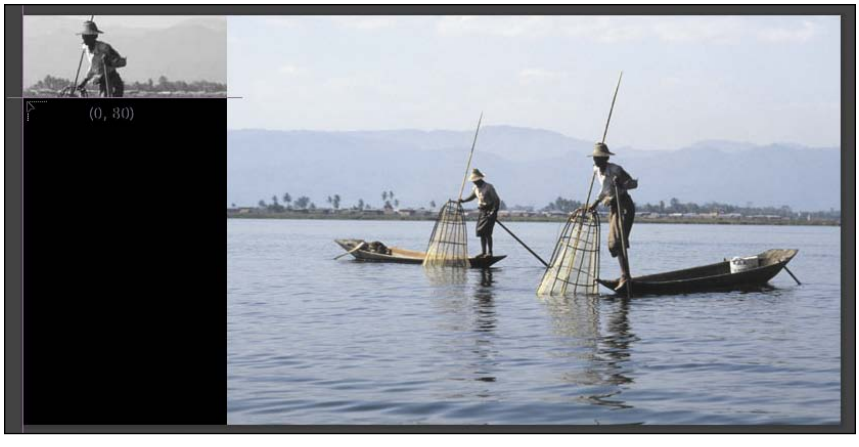
Small, cropped versions of the larger Myanmar images are provided for you in the images folder.

1. In the Library panel, expand the images folder within the Assets folder.
2. Drag the file called button1_gray.jpg from the Library panel to the Stage.

A grayscale thumbnail of a fisherman appears on the Stage, Timeline, and Elements panel.

3 Position the thumbnail so that its top-left corner is at the top-left corner of the Stage. The coordinates should be at X=0, Y=0.

4 Drag the file called button2_gray.jpg from the Library panel to the Stage, and position it just below the first thumbnail. You can use the smart guides to help automatically snap the images in place. The coordinates should be at X=0, Y=80.



Two grayscale thumbnails are at the left side of the Stage, one above the other.

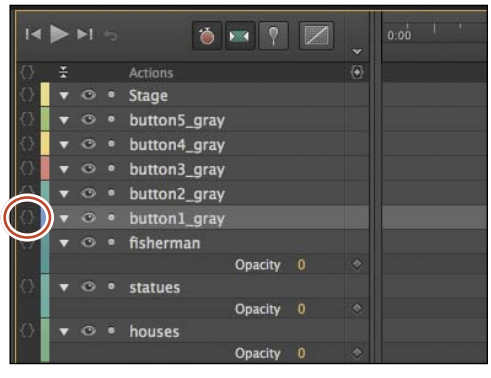
5 Drag the remaining three grayscale thumbnails from the Library panel onto the Stage, positioning each successive one below the previous one.

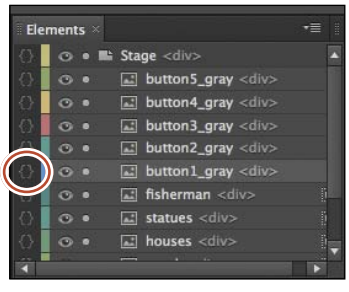
There should be a total of five thumbnail images vertically stacked on the left side of the Stage.

Adding the events

Each element on the Stage can have its own events and actions. Insert code for individual elements from the far-left column of the Timeline or Elements panel. The Open Actions button is indicated by a set of curly braces.

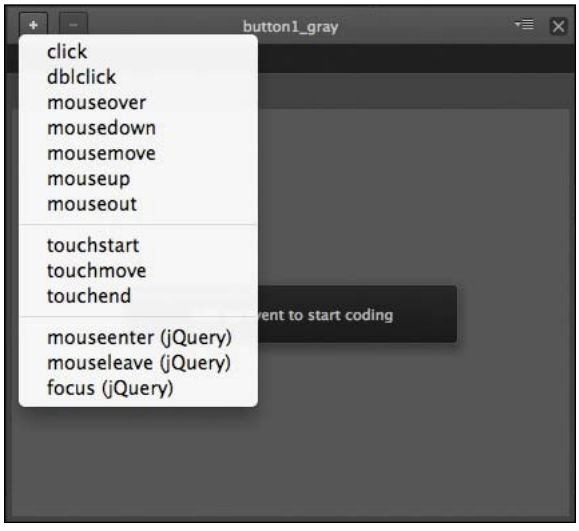
1 In the Timeline or the Elements panel, click the Open Actions button for the first thumbnail element, button1_gray.





The script panel for button1_gray opens.

A menu of options automatically opens, displaying the events that are possible for the button1_gray element.

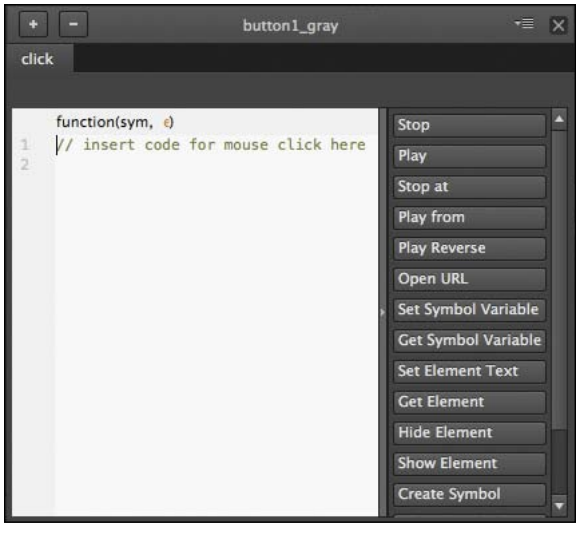


Note: As you add more events to the same element, additional tabs appear at the top of the script panel.

Note: To delete an event (and any code associated with it), select the particular tab for the event and click the Minus button at the top-left corner of the script panel.

2 Select the first option: click.

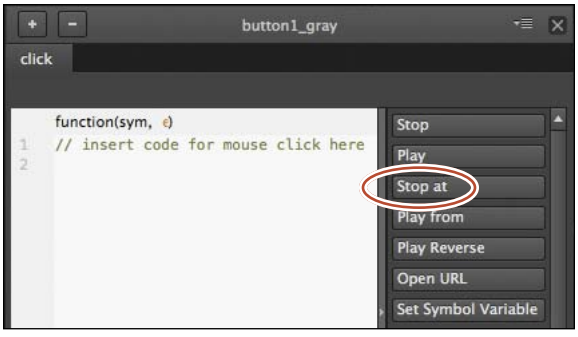
Edge Animate adds a click tab at the top of the panel with an empty script pane and available snippets on the right.



Adding the actions

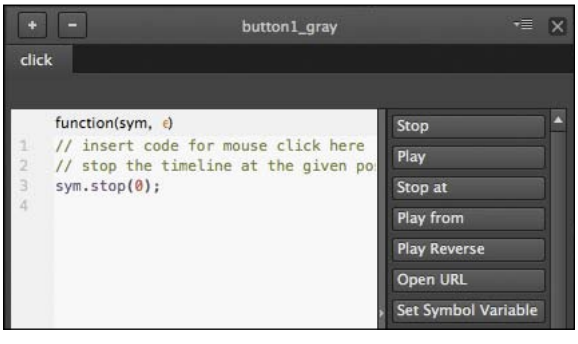
Each event must have an accompanying action, or a response, to the event.

1 Position your cursor on the second line of the script pane (after the first line of comments), and choose the Stop at option.



Edge Animate adds the code to stop the playhead at a particular position on the Timeline.

2 In between the parentheses of the `stop()` method, replace the 1000 number with **0**.

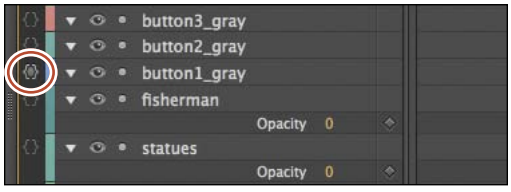


The `stop()` method requires a number, in milliseconds, of the point on the Timeline at which it will move to and stop. Since this first grayscale thumbnail is of the fisherman, you want the playhead to stop at 0:00 seconds, the point at which we see the full image of the fishermen on the Stage.

Note: You can actually use any number between 0 and 1500 for the stop() method for button1_gray, since the image of the fishermen remains on the Timeline until 1.5 seconds, but it's simpler and easier to be consistent to pick the time when the image first appears.

3 Close the script panel.

The Open Actions icon for the button1_gray element becomes filled in, indicating that there is currently script attached to that element.



4 Preview your Edge Animate composition in a browser by selecting File > Preview in browser, or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS).

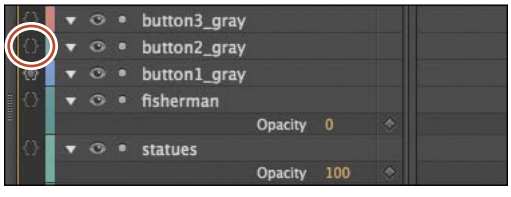
5 At any point during the slideshow, click the first grayscale thumbnail.

The slideshow stops and shows the image of the fishermen.

Completing the interactivity

Now that you've completed the interactivity for the first button, add the same functionality to the remainder.

1 In the Timeline or the Elements panel, click the Open Actions button for the second thumbnail element, button2_gray.



The script panel for button2_gray opens.

2 Click on the Plus button on the upper-left corner.

A menu of options opens, displaying the events that are possible for the button2_gray element.

3 Select the first option: click.

Edge Animate adds a click tab at the top of the panel with an empty script pane and available snippets on the right.

4 Position your cursor on the second line of the script pane (after the first line of comments), and choose the Stop at option.

Edge Animate adds the JavaScript code to stop the playhead at a particular position on the Timeline.

5 In between the parentheses of the `stop()` method, replace the 1000 number with **2000**.

The `stop()` method requires a number, in milliseconds, of the point on the Timeline at which it will move to and stop. Since the second grayscale thumbnail is of the statues, you want the playhead to stop at 0:02 seconds, the point at which we see the full image of the statues on the Stage.

6 Add similar click events to all the other grayscale thumbnail images, with the Stop at option. Be sure to change the arguments for each `stop()` method as follows, so the playhead stops at different times to display a unique larger image on the Stage:

- The `stop()` method for button1_gray should go to 0 milliseconds.
- The `stop()` method for button2_gray should go to 2000 milliseconds.
- The `stop()` method for button3_gray should go to 4000 milliseconds.
- The `stop()` method for button4_gray should go to 6000 milliseconds.
- The `stop()` method for button5_gray should go to 8000 milliseconds.

Navigating the Code panel

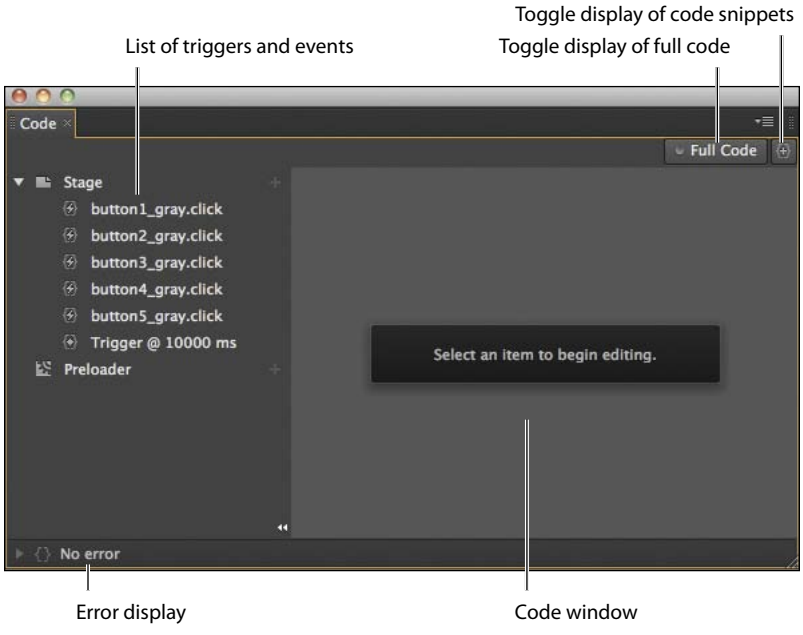
Your travel slideshow is now interactive, where users can click to see any of the images. But your code appears to be scattered among many different elements. How can you view all the code for your Edge Animate composition together? The answer is in the Code panel, which you can open by choosing Window > Code, or pressing Ctrl+E (Windows) or Command+E (Mac OS).

Viewing your code

The Code panel displays all the code in your Edge Animate composition—both the code that is automatically generated for every project, and the code that you insert yourself.

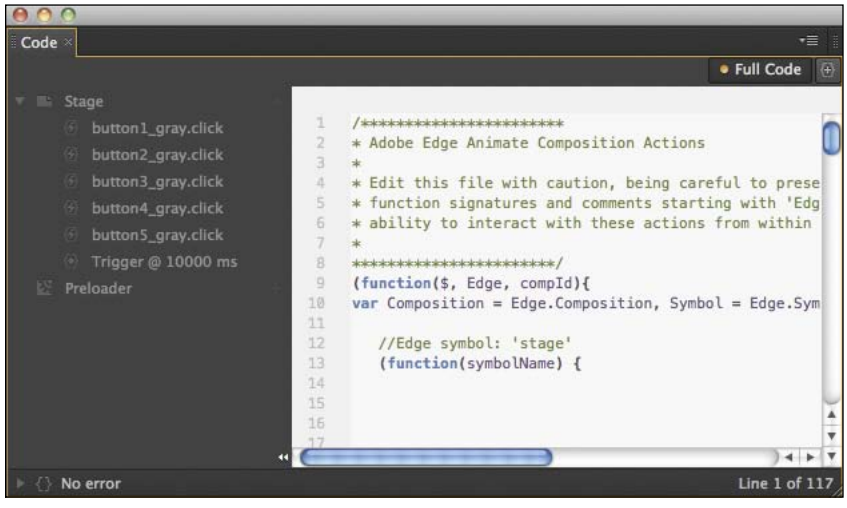
1 Choose Window > Code, or press Ctrl+E (Windows)/Command+E (Mac OS).

The Code panel opens. The Code panel is similar to the other script panels for triggers, events, and actions.

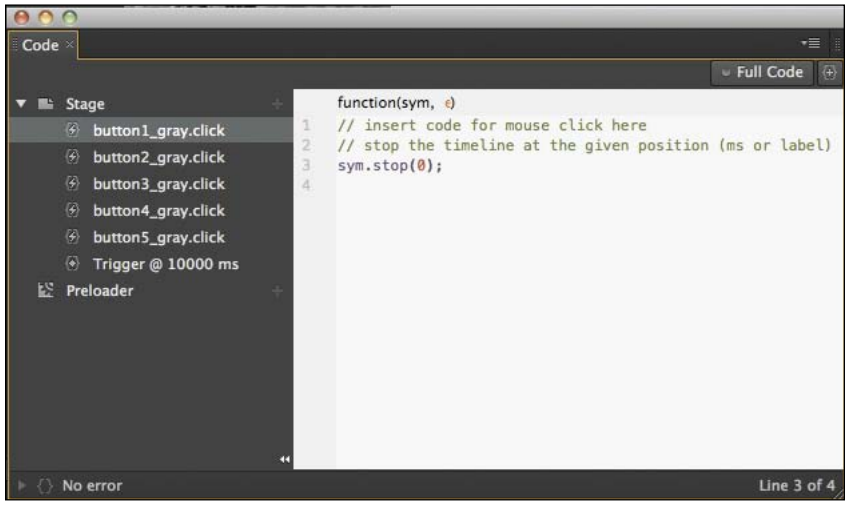


2 Click on the Full Code button on the far-right side of the panel to toggle between Full Code mode and non-Full Code mode.

In Full Code mode, Edge Animate displays the entire code for the JavaScript file for the Edge Animate composition. Scroll down to see the script for all your thumbnail elements as well as the trigger. The code that this represents is contained in the file 05_workingcopy_edgeActions.js.

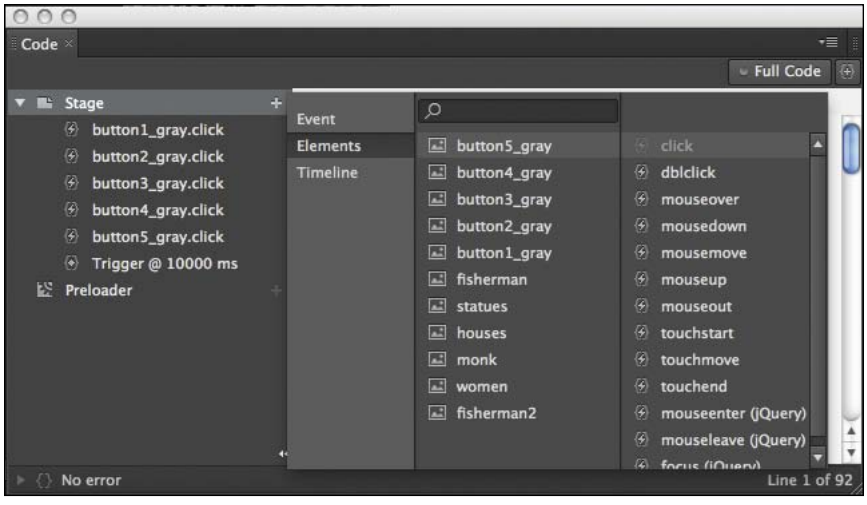


In non-Full Code mode, you can select the code for the individual elements or triggers on the Stage on the left side of the panel. In addition, there is an option to see the code for the Preloader, which is currently disabled because you haven't yet added one. In later lessons, you'll learn about adding a Preloader.

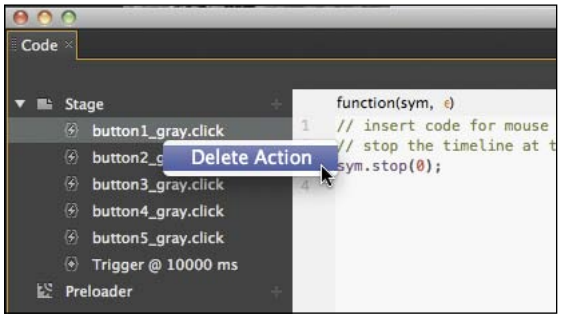


3 While in non-Full Code mode, click on the Plus button in line with the Stage element.

A hierarchical menu appears that allows you to add an event to the Stage itself, an event to any of the elements on the Stage, or an event to the Timeline.



4 If you want to delete an event or trigger from the Code panel, right-click on the event or trigger from the list and select Delete Action.

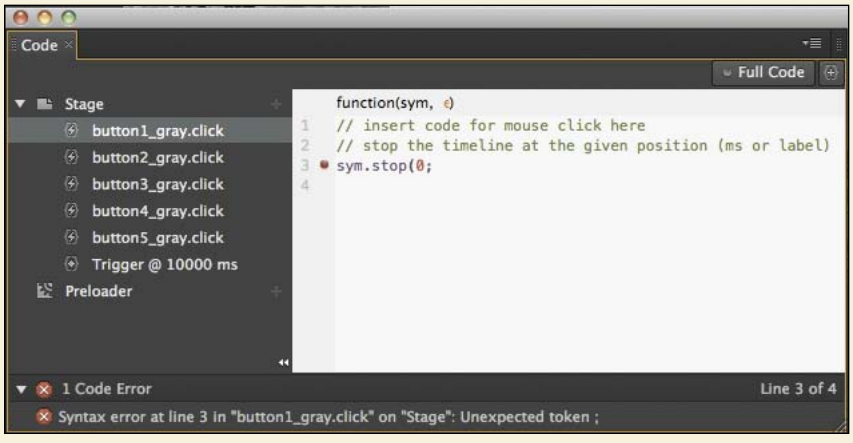


5 In either Full Code or non-Full Code mode, when you make additions and edits to the script, and they are saved in the composition.

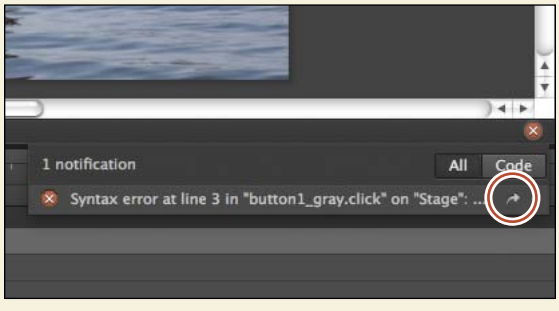
Dealing with code errors

Using the provided code snippets makes adding code relatively easy because the script is already formatted correctly. All you have to do is replace key values. However, bugs and typos do invariably creep in, and dealing with code errors is a common struggle for any developer, whether a novice or an expert.

Edge Animate immediately alerts you to code errors, which make finding and correcting them easy. When there is a mistake in syntax in any of the code, Edge Animate displays an error message at the bottom-left corner of the Code panel. For example, if you were to accidentally delete the closing parenthesis of the `stop()` method, the error display tells you where the error occurs. In addition, a red dot appears next to the line of code in question.



The error notification is also displayed at the bottom-left corner of the Stage.



Click on the arrow icon after the error description to jump directly to the source of the error in the Code panel so you can fix it. The All or Code options in the error display determines whether all errors are displayed (including warnings of feature incompatibilities with various devices, such as text shadows in IE9), or only code errors are displayed.

Creating labels

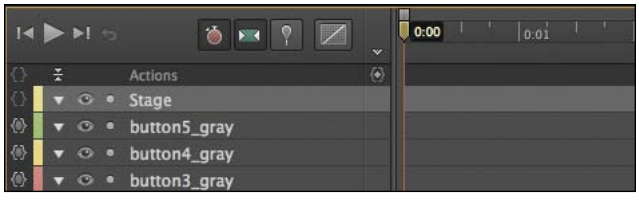
When the user clicks each thumbnail, Edge Animate moves the playhead to a new time on the Timeline, according to the argument in the `stop()` method. However, imagine that the client who has commissioned you to develop this slideshow wants the whole sequence to run a little slower. That's an easy task to do because you can select all of the elements on the Timeline and move all the keyframes and animations forward to lengthen the total amount of time. But doing so causes the times that each image appears on the Stage to change, which would require you to change all the millisecond values in the `stop()` methods.

There is an alternate approach that would save you time and effort. Instead of using fixed-millisecond times in the `stop()` methods, you can use labels, which refer to points on the Timeline. Labels can move with your animation, so increasing or decreasing the length of your animation can move the labels proportionately.

Adding labels

Labels appear on the Timeline panel, below the time markers and just above the Actions layer.

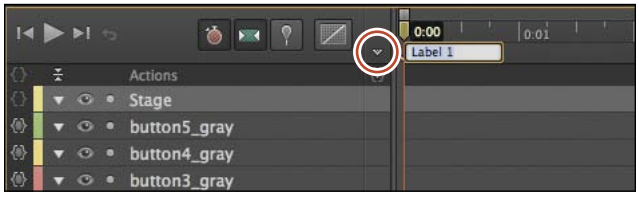
1 Move the playhead to 0:00 seconds.



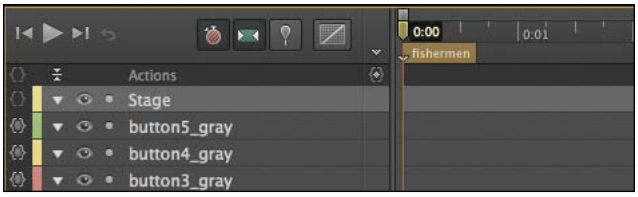
The image of the fishermen appears at 0:00 seconds.

2 Click the Insert Label button, or press Ctrl+L (Windows)/Command+L (Mac OS).

A label appears on the Timeline, named Label 1.



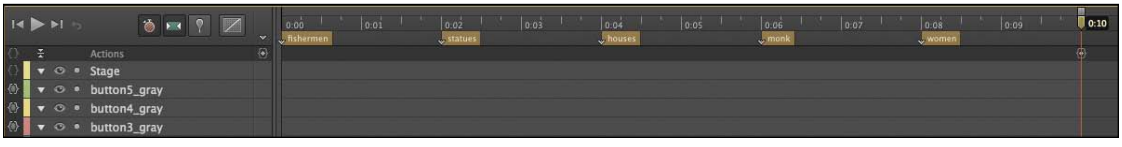
3 Rename the label **fisherman**, and press Enter to exit the text editing of the label.



The label called fisherman is now associated with 0:00 seconds.

4 Add four more labels to the Timeline, each marking the starting point at which an image appears on the Stage.

- Insert the label **statues** at 0:02 seconds.
- Insert the label **houses** at 0:04 seconds.
- Insert the label **monk** at 0:06 seconds.
- Insert the label **women** at 0:08 seconds.



Editing labels

There are several ways you can edit labels once you've inserted and named them:

- **To rename a label**, double-click the label name on the Timeline.
- **To move a label**, click and drag the label on the Timeline to a new position.
- **To delete a label**, select the label on the Timeline so it is highlighted and press the Delete key.
- **To cut, copy, or paste a label**, right-click on a label and choose your desired option, or use the standard keyboard commands for cut (Ctrl/Command+X), copy (Ctrl/Command+C), and paste (Ctrl/Command+V).

Changing the references to the Timeline

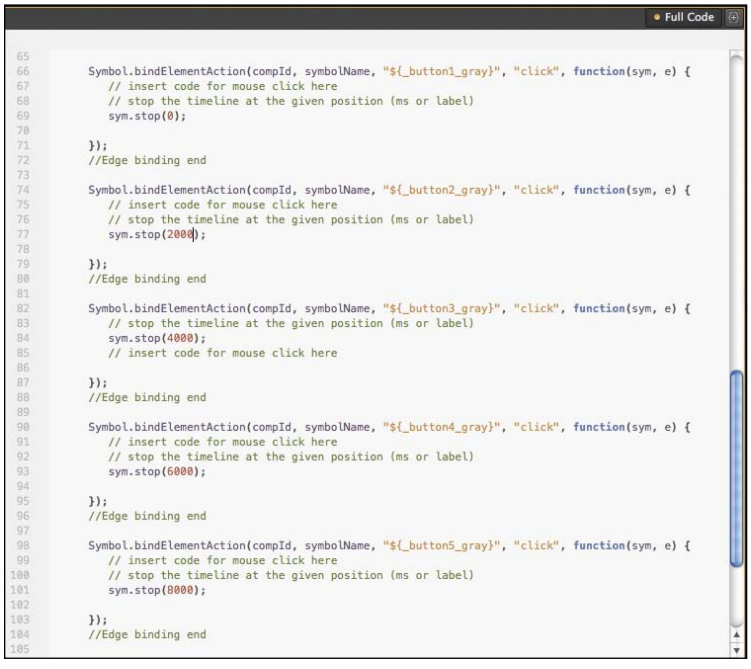
Now that the Timeline contains labels, you can change the references in the JavaScript code.

1 Choose Window > Code, or press Ctrl+E (Windows)/Command+E (Mac OS).

The Code panel opens.

2 If it is not already selected, click the Full Code button to display the panel in Full Code mode.

All the code for the thumbnail events and actions are displayed in the single script pane.



Note: Make sure that you are using straight quotation marks around your label names. The quotation marks are essential for JavaScript to identify the names as a String (and not a variable). Straight quotes and curly quotes (or smart quotes) are different characters in HTML and JavaScript and they are not interchangeable.

3 Replace all the millisecond times in the `stop()` methods with your labels. The following lines of code should be changed:

- Change `sym.stop(0);` to `sym.stop("fisherman");`
- Change `sym.stop(2000);` to `sym.stop("statues");`
- Change `sym.stop(4000);` to `sym.stop("houses");`
- Change `sym.stop(6000);` to `sym.stop("monk");`
- Change `sym.stop(8000);` to `sym.stop("women");`

4 Preview your Edge Animate composition in a browser by choosing File > Preview in the browser, or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS).

5 At any point during the slideshow, click the thumbnail images.

The slideshow stops and shows the selected image.

6 Return to Edge Animate and select all the elements on the Timeline by choosing Select > All, or by pressing Ctrl+A (Windows)/Command+A (Mac OS). Make sure none of the elements are locked.

7 Click and drag the last keyframe of the last animated element on the Timeline. Drag the keyframe to the left to decrease the total amount of time of the slideshow.

As you decrease the length of time for all the animated elements, the labels also move proportionally, preserving their identification of what's displayed on the Stage. Return the total time of the composition to 0:10 seconds.

Adding visual feedback

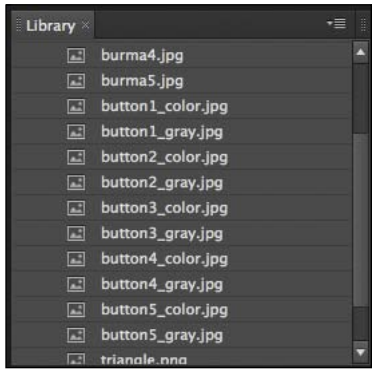
Most interactive elements on the Web feature visual feedback, which is important to provide clues to the reader that the particular item is interactive. For example, a simple hyperlink on a Web page often will change color when you move your mouse over it. A button will highlight when you move your mouse over it, and may appear depressed when you click on it.

You can create these interactions with a combination of events and actions in Edge Animate. You'll add these events and actions to the thumbnails for visual feedback next.

Adding the mouseover thumbnails

The first question you should ask is, what visual effect do you want to see when a user moves their mouse over the thumbnail images? For this project, you'll make the grayscale thumbnails become colorized and a highlight appear around the borders. The first step is to bring colorized versions of the thumbnails on to the Stage.

1 In the images folder in the Assets folder of the Library panel, you'll find color versions of each of the five thumbnail images, indicated by the _color appended to the file name.



2 Drag button1_color.jpg from the Library on to the Stage.

3 Use the Smart Guides to position the button1_color element at the upper-left corner, exactly on top of its grayscale version. Its location should be at X=0, Y=0.

4 Drag all four of the other colorized versions of the thumbnails to the Stage, positioning them exactly on top of their grayscale partners. All the colored thumbnails should be at the top of the Element panel stack.

The grayscale and colorized versions of the thumbnails are exactly the same dimensions.

Hiding the mouseover thumbnails

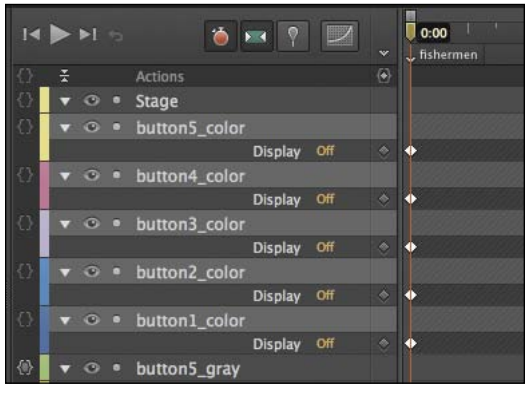
Since you want to show the colorized version only when the mouse cursor moves over the thumbnail, you must first hide the colorized thumbnails. You can hide the elements by changing their Display property to Off.

1 Move the playhead to 0:00 seconds.

2 Hold down the Shift key and select all the colorized thumbnail elements.

3 In the Properties panel, change the Display property from Always On to Off.

Edge Animate inserts a new keyframe on the Timeline for all the selected elements at 0:00 seconds and the selected elements disappear from the Stage.

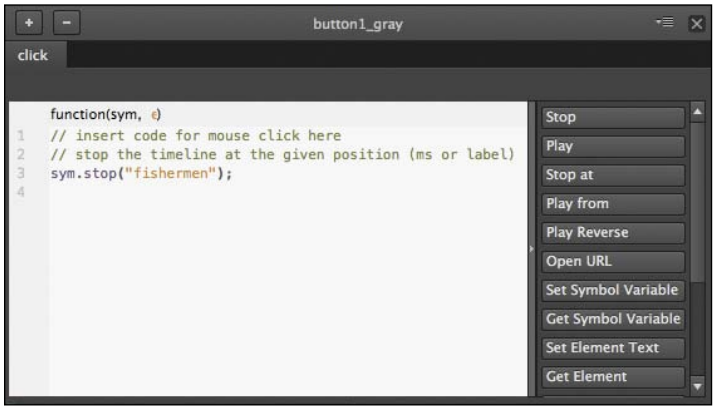


Inserting the mouseover event

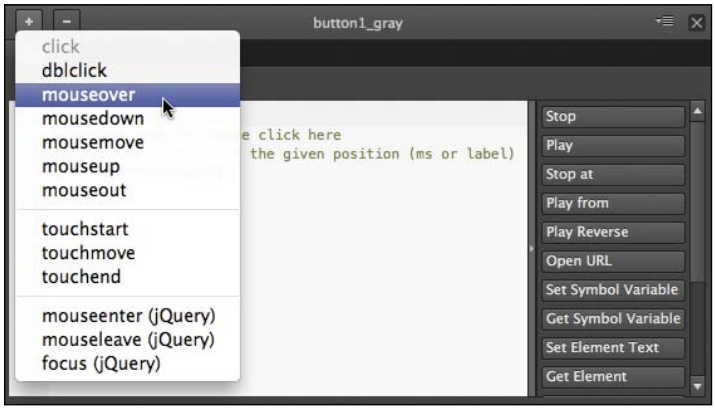
Each grayscale thumbnail contains a click event. You'll have to edit each of those elements to incorporate a mouseover event. A *mouseover event* happens when the user moves their mouse cursor over the selected element. When the mouseover event happens, you'll show the colorized thumbnails.

1 In the Timeline or the Elements panel, click the Open Actions button for the first thumbnail element, button1_gray.

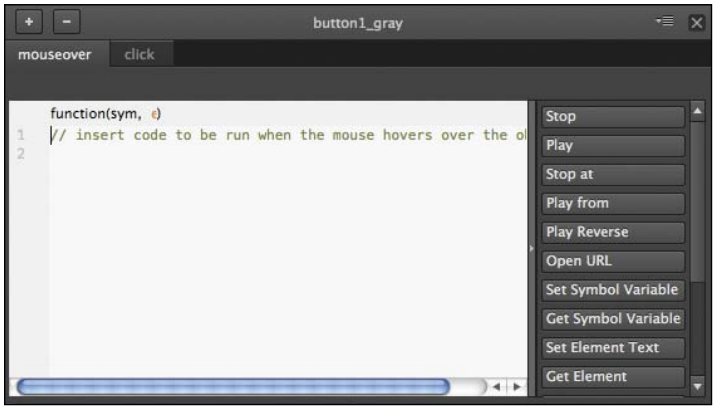
The script panel for button1_gray opens. The current click event and actions appear.



2 Click on the Plus button on the upper-left corner and choose mouseover.

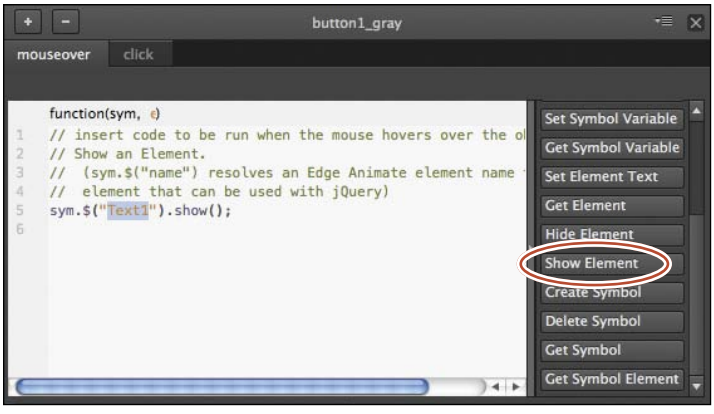


Edge Animate adds a mouseover tab.

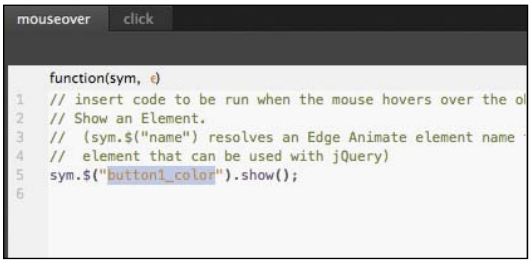


3 Position your cursor on the second line of the script pane (after the first line of comments), and choose the Show Element option.

Edge Animate adds the JavaScript code to display an element. The highlighted portion of the code is the name of the element to display.



4 Replace the highlighted code with **button1_color**. Make sure that the double straight quotation marks remain around your element name.



The full statement appears as follows:

```
sym.$("button1_color").show();
```

The dollar sign and parentheses is jQuery syntax, and it tells the browser what element to select. In this statement, the element called button1_color in the current Edge Animate composition is selected, and the method called `show()` is executed.

5 Preview your Edge Animate composition in a browser by choosing File > Preview in your browser or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS). Move your mouse over the first grayscale thumbnail image.

As soon as your mouse cursor moves over the grayscale thumbnail image, the colorized version appears. Since the colorized version is above the grayscale version, we see only the colorized image.

6 Return to Edge Animate and insert the mouseover event with the Show Element action to the remaining four grayscale thumbnail buttons. Make sure to replace the highlighted code portion with the correct colorized version of the thumbnail.

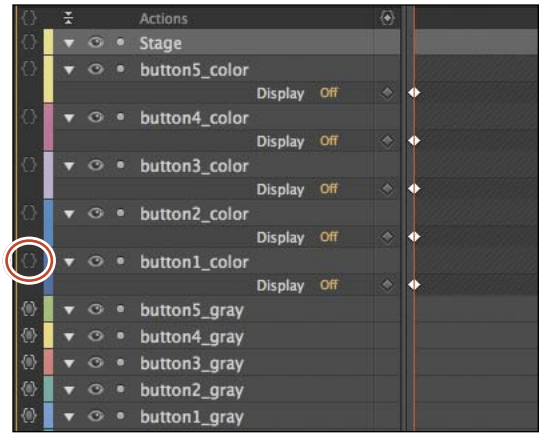
Inserting the mouseout event

When you preview the Edge Animate project, you'll notice that the thumbnails become colorized when you move your mouse over them, but they remain in color even after you move your mouse off them. In order to make the thumbnails revert to their grayscale versions, you need to add one additional event: the mouseout event.

The *mouseout* event happens when the cursor moves off an element. You'll add the mouseout event to the *colorized* thumbnails (not the grayscale thumbnails) and pair the event with a command that hides the colorized versions. The result: The colorized versions disappear, leaving the grayscale version visible again.

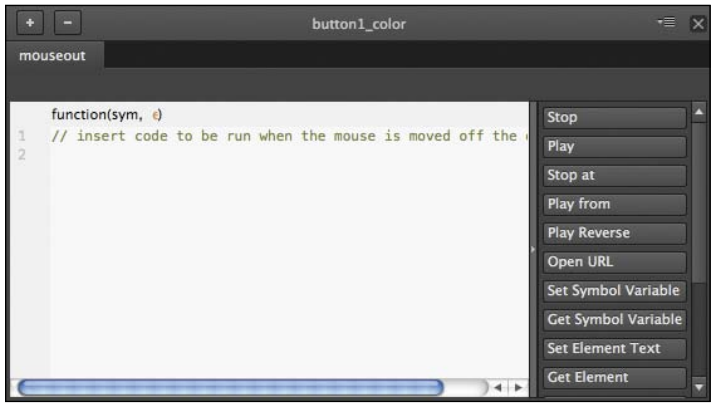
1 In the Timeline or the Elements panel, click the Open Actions button for the thumbnail element button1_color. The element is currently hidden on the Stage, but you can still add script to it.

The script panel for button1_color opens.

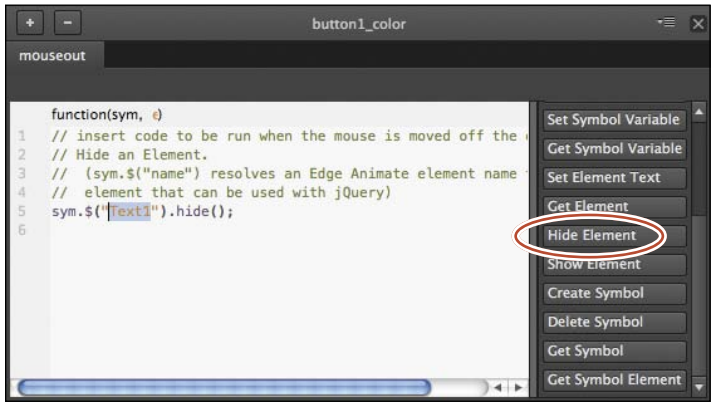


2 In the popup menu that appears, choose mouseout for the event.

Edge Animate adds a mouseout tab.

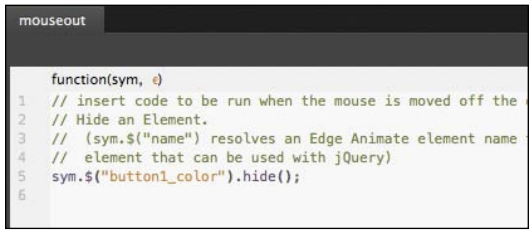


3 Position your cursor on the second line of the script pane (after the first line of comments), and choose the Hide Element option.



Edge Animate adds the JavaScript code to display an element. The highlighted portion of the code is the name of the element to display.

4 Replace the highlighted code with **button1_color**. Make sure that the double straight quotation marks remain around your element name.



The full statement appears as follows:

```
sym.$("button1_color").hide();
```

Note the similarities between the actions for this mouseout event and the previous script for the mouseover event.

5 Preview your Edge Animate composition in a browser by choosing File > Preview in your browser or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS). Move your mouse over the first grayscale thumbnail image.

As soon as your mouse cursor moves over the grayscale thumbnail image, it becomes colorized. When you move your mouse cursor off the image, the button appears to revert back to grayscale.

6 Return to Edge Animate and insert the mouseout event with the Hide Element action to the remaining four colorized thumbnail buttons. Make sure to replace the highlighted code portion with the correct colorized version of the thumbnail.

Editing the click event

One final fix is needed before all the events and actions work together. You may have noticed that clicking on the buttons doesn't move the playhead as you intend. The reason it no longer works is because the colorized thumbnails overlap their grayscale counterparts, which block the click events. Your final step is to remove the click event from the grayscale thumbnails and add them to the colorized thumbnails instead.

1 In the Timeline or the Elements panel, click the Open Actions button for each of the grayscale thumbnail elements.

2 Choose the click event tab on the script panel, and click the Minus button.

Edge Animate deletes the click event and all of its actions.

3 In the Timeline or the Elements panel, click the Open Actions button for each of the colorized thumbnail elements.

4 Click on the Plus button on the upper-left corner and choose click for the event.

Edge Animate adds a click event tab.

5 Choose the Stop at option, and as you did before, replace the millisecond argument with the corresponding label on the Timeline.

6 Preview your Edge Animate composition in a browser by choosing File > Preview in your browser or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS).

Your buttons are complete. When you move your mouse over them, they become colorized. When you move your mouse off them, they revert to grayscale, and when you click on them, Edge Animate displays the corresponding image from the slideshow.

Note: Use the Code panel and choose the Full Code mode to make global edits to your script easier. You can save time and effort if you're careful about selecting and editing code.

Customizing the mouse cursor

In addition to the visual feedback that you can provide by changing the appearance of the button when the user interacts with it, you can also change the appearance of the cursor itself. Often, the default arrow cursor on a desktop or laptop browser changes to a hand (known as the *pointer* cursor) when it hovers over an interactive element or hyperlink. You can choose to change the cursor to a pointer, or choose from among dozens of other cursor types.

Using the pointer

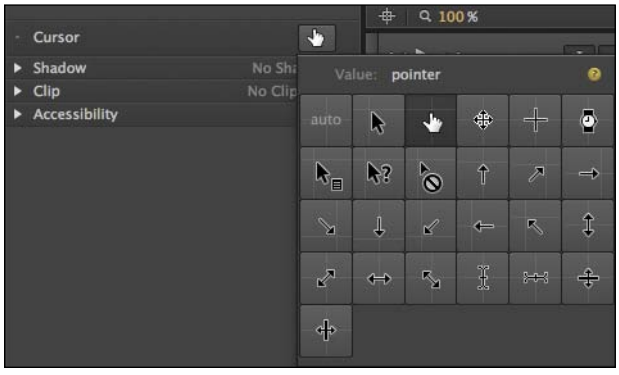
The Properties panel controls the cursor appearance and allows you to select a custom icon for each element.

1 In the Timeline panel, temporarily turn the Display property for the five colorized thumbnail elements to On.

Turning on the Display property allows you to select them on the Stage.

2 Select all five colorized thumbnail elements, button1_color through button5_color.

3 In the Properties panel, click the Cursor option and choose the pointer icon.



4 Turn the Display property for the five colorized thumbnail elements back to Off.

The colorized thumbnail elements are hidden again.

5 Preview your Edge Animate composition in a browser by choosing File > Preview in your browser or pressing Ctrl+Enter (Windows)/Command+Return (Mac OS).

The pointer cursor appears whenever you move your mouse over or click on the buttons.

Note: You can only change the appearance of the cursor for each element, and not for every event of an element.

Controlling animated elements

So far, you've added JavaScript that controlled the behavior of the playhead and the hiding or displaying of particular elements. You can also add code to control the playback of animated symbols.

Symbols, as you learned in the previous lesson, are independent objects that you create that can have their own internal animation. With JavaScript, you can control the symbol animations to create more sophisticated interactions. For example, you can create a button that controls a dramatic animated unfurling or closing of a map. Or, you can create a button that controls the animated expansion or collapse of a more info box. The map and the more info box would be symbols that behave independently on the main Timeline.

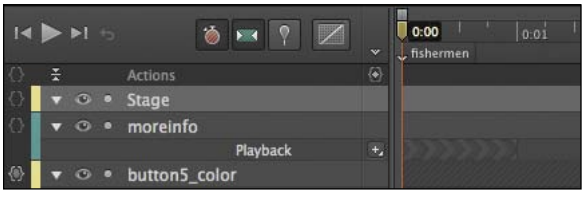
For your interactive travel slideshow, you'll add a button at the top of the Stage that, when rolled over, elegantly animates to reveal information about the images and Myanmar.

Adding the button and animated symbol

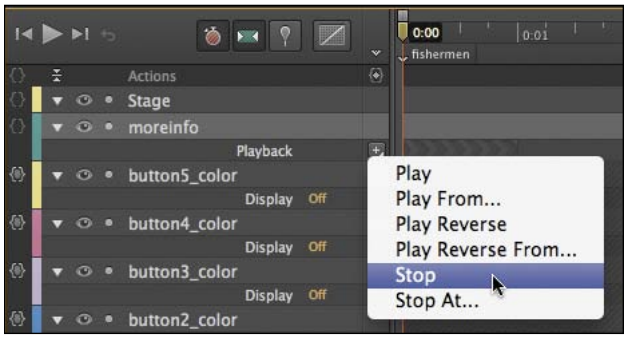
The button and the animated symbol have already been created for you, and are in the Library ready to use.

1 In the Library panel, expand the Symbols section, and drag the moreinfo symbol to the Stage. Position the moreinfo symbol at X=200, Y=0.

The moreinfo symbol appears in the Elements and Timeline panels. The short playback arrowheads on the Timeline show how long the animation within the symbol lasts (1 second long).



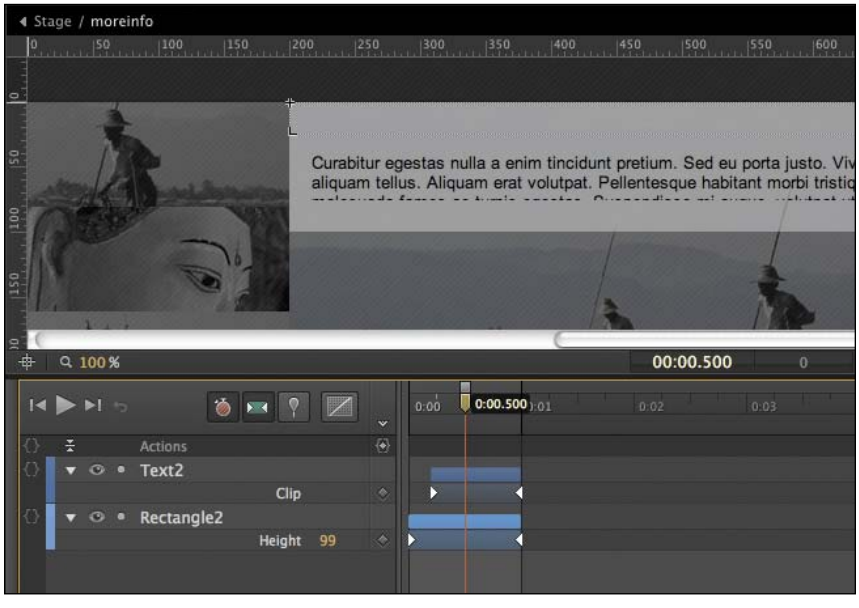
2 In the Timeline, click the Playback options for the moreinfo element and choose Stop.



The playback of the internal symbol animation does not play on the main Timeline.

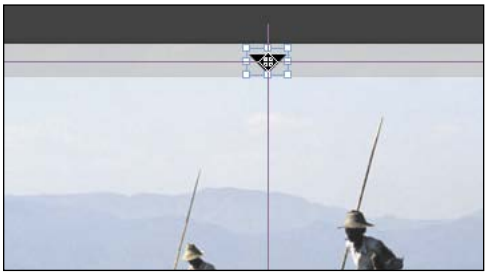
3 Double-click the moreinfo symbol on the Stage and press the spacebar to view the animation within the symbol.

The symbol consists of two short animations. The long horizontal gray rectangle expands, and at the same time, the clipping box of some informational text expands to reveal it.



4 Click the Stage button at the top of the Stage to exit your symbol.

5 Drag the triangle.png image from the Library Assets folder to the Stage. Position the triangle element at X=484, Y=3, or use the Smart Guides to center the element over the moreinfo element.

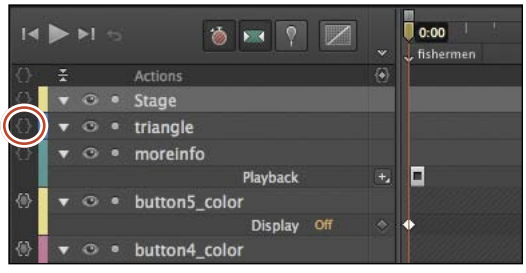


Play a symbol animation

The symbol is currently stopped at 0 seconds. You'll add a mouseover event to the triangle button that tells the symbol to begin playing.

1 In the Timeline or the Elements panel, click the Open Actions button for the triangle element.

The script panel for triangle opens.

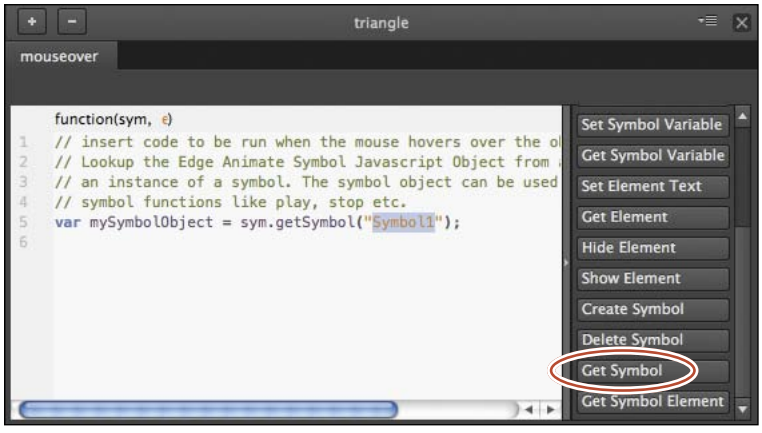


2 Choose mouseover for the event.

Edge Animate adds a mouseover event tab.

3 Choose the Get Symbol option.

Edge Animate adds the JavaScript code to select a particular symbol on the Stage. The highlighted portion of the code is the name of the symbol that you want to select.

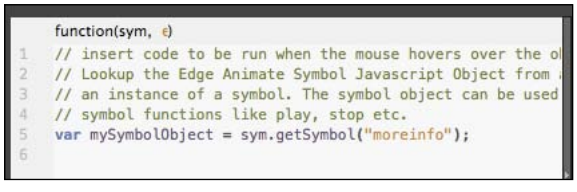


4 Replace the highlighted code with **moreinfo**, to match the moreinfo element on the Stage. Make sure that the double straight quotation marks remain around your element name.

The full statement appears as follows:

```
var mySymbolObject = sym.getSymbol("moreinfo");
```

The first part of this statement, `var mySymbolObject`, creates a variable for the reference to your symbol, so you can control it.



5 On the next line in the script panel, choose the Play option.

Edge Animate adds a statement that plays the `sym` object, or the main Timeline. However, you want the symbol to play its animation, not the animation on the main Timeline.

6 Replace `sym` with the variable, `mySymbolObject`, which refers to your symbol.

```
function(sym, e)
// insert code to be run when the mouse hovers over the ol
// Lookup the Edge Animate Symbol Javascript Object from
// an instance of a symbol. The symbol object can be used
// symbol functions like play, stop etc.
var mySymbolObject = sym.getSymbol("moreinfo");
mySymbolObject.play();
```

The next statement appears as follows:

```
mySymbolObject.play();
```

Note: You can combine the two statements into one line as follows:

```
sym.getSymbol("moreinfo").play();
```

Reset the symbol animation

Now, you'll add a mouseout event for the triangle element to move its playhead back to 0 seconds to reset the animation.

1 In the Timeline or the Elements panel, click the Open Actions button for the triangle element.

The script panel for moreinfo button opens.

2 Click on the Plus button on the upper-left corner and choose mouseout for the event.

Edge Animate adds a mouseout event tab.

3 Choose the Get Symbol option, and replace the highlighted code with **moreinfo**.

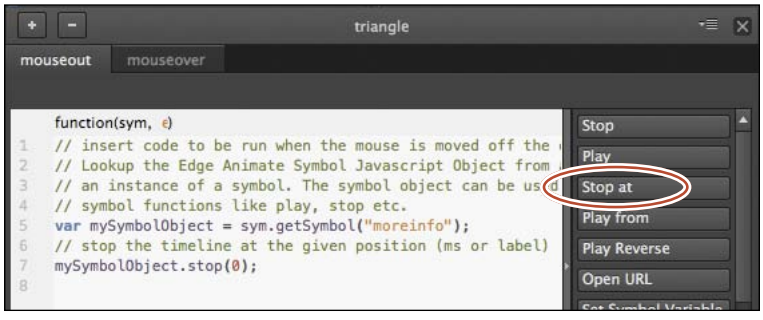
The full statement appears as follows:

```
var mySymbolObject = sym.getSymbol("moreinfo");
```

4 On the next line in the script panel, choose the Stop at option.

Edge Animate adds a statement that stops the `sym` object, or the main Timeline. However, you want the symbol to stop its animation, not the animation on the main Timeline.

5 Replace `sym` with the variable, `mySymbolObject`, which refers to your symbol. Replace the 1000 default millisecond argument with **0**.

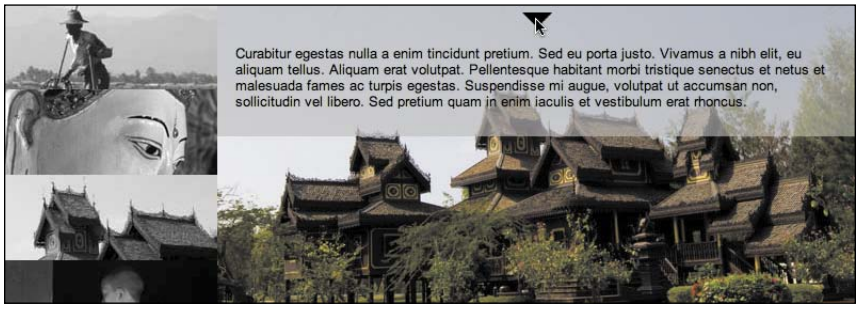


The next statement appears as follows:

```
mySymbolObject.stop(0);
```

6 Preview your Edge Animate composition in a browser by choosing File > Preview in your browser or pressing Ctrl+Enter (Windows)/ Command+Return (Mac OS).

When you roll over the triangular button at the top of the Stage, the moreinfo symbol plays its animation, which reveals the text box and text. When you roll off the button, the text box and text collapse.

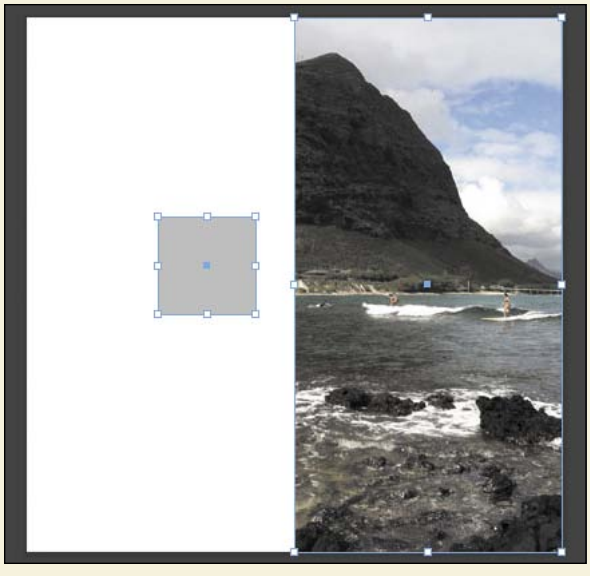


Using jQuery Effects

The Edge Animate API offers a nice balance of power, flexibility, and ease of use to incorporate interactivity to your designs and animations. Inserting script by simply clicking a button in the script panel is (mostly) idiot-proof. However, adding a bit of jQuery to your scripts can often make your job easier. As you learned earlier in this lesson, jQuery is a JavaScript library that was written specifically to make it simple to select elements on a Web page and creating animations and transitions. There are many jQuery methods for animating elements, such as a fade-in, fade-out, or a slide-in and slide-out. Since Edge Animate is fully compatible with JavaScript and jQuery, you can use these methods wherever you see fit.

Let's examine one particular jQuery method, `fadeToggle()`. The method `fadeToggle()` animates an element's transparency to fade up or fade down, depending on its current state. If the element is transparent, it will become opaque. If the element is opaque, it will become transparent.

1 In a new Edge Animate composition, add a small rectangle and an image on the Stage. Name the rectangle **Rectangle**, and the image **Image**.

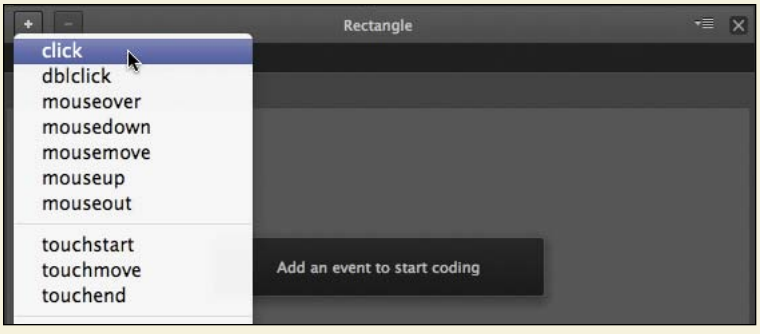


2 In the Timeline or the Elements panel, click the Open Actions button for the Rectangle element.

(continues on next page)

Using jQuery Effects *(continued)*

3 Choose click for the event.



4 Add the following statement for the click event:

```
sym.$("Image").fadeToggle();
```

5 That's all! Preview your composition in a browser.

When you click on the rectangle, Edge Animate uses jQuery to select the image and executes the `fadeToggle()` method. The image fades in and out with alternate clicks. jQuery does all the work of creating two inverted animations without requiring you to manually create any symbols, keyframes, or mechanisms to remember the state of the image. jQuery is powerful and makes a good addition to your designer-developer toolkit. You can view the 05JQuery.an file in the 05End_JQuery folder to see the completed example.

Review questions

1 What's the difference between actions, triggers, and events, and how are they used to create interactivity in Adobe Edge Animate?

2 What's the relationship between the Edge Animate API, jQuery, and JavaScript?

3 How do you create a button?

4 Why would you use a label, and where are they located?

5 What does the code `sym` mean in the Edge Animate API, and how do you use it?

Review answers

1 Actions, triggers, and events are all JavaScript code that you use to create interactivity in Edge Animate. Actions are commands that tell Edge Animate to do something, such as hide or display an element, or load a hyperlink. Triggers are actions that are placed on the Timeline so they are executed at a specific time. Events are things that happen in a composition that Edge Animate can respond to with actions.

2 Edge Animate uses JavaScript to power its animation and interactivity. JavaScript is the standard scripting language for Web browsers. jQuery is a library of well-written JavaScript functions that make it easier to select and animate elements on a Web page. The Edge Animate API provides additional functions (based on JavaScript and jQuery) to control elements in your composition.

3 A button is a visual indicator of what the user can interact with. You can create a button by creating an element on the Stage, then clicking the Open Actions button in the Timeline or Elements panel to add an event. In the event tab that opens, insert actions that you want triggered when the event happens.

4 Labels are located at the top of the Timeline. Labels identify specific points in time so that you can refer to label names, rather than fixed milliseconds, in your JavaScript code.

5 The word `sym` represents the whole Edge Animate composition, when the statement is on the main Timeline. Edge Animate is organized around the concept of "symbols," and the root, or base-level symbol, is the Edge Animate Stage. This root symbol contains all the elements and animations in your Edge Animate composition—everything on the Stage or Timeline. In JavaScript, when you want to do something, you first identify the object that you want to control. If you want to affect the Timeline of your Edge Animate composition, the first thing that is written in the script is `sym`.

INDEX

SYMBOLS

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z