# Lab Manual for Murach's HTML5 and CSS3 (3rd Edition)

# Introduction for students

This Lab Manual is designed to be used with *Murach's HTML5 and CSS3 (3rd Edition)*. The topics that follow describe the components of this Manual and explain how to get the most from them.

## Behavioral objectives

The behavioral objectives for each chapter describe the skills that you should have when you complete the chapter. The educational theory is that if you know what the objectives are, your study will be more focused. In particular, you'll know exactly what to study as you prepare for a test because the test questions that we supply test only the skills that are described by the objectives.

If you study the objectives, you can see that the first objectives for each chapter are what we refer to as *applied objectives*. These ask you to apply what you've learned as you develop web pages. If you can develop the pages that are described by the book exercises, the exercises for the Halloween case study, and the student projects in this Manual, you will meet these objectives.

After the applied objectives for each chapter, you'll find what we refer to as *knowledge objectives*. These objectives define skills like identifying, describing, and explaining the required concepts, terms, and procedures. In general, you should be able to do the knowledge objectives, even if you have trouble with some of the applied objectives.

In some cases, it makes sense to review the objectives before you read a chapter. That way, you'll know what to focus on. In most cases, though, the objectives won't make sense until after you read the chapter. That's why it's better to review the objectives after you read a chapter. Then, if you don't think you can do one or more of the objectives, you can review the related items in the chapter summary.

## Chapter summaries

This Manual also includes summaries of the information presented in each chapter. These summaries are bulleted lists of the main concepts and programming techniques that are presented in the text. Then, if you aren't clear about one or more of these summary points, you can review the topics that present them.

Although these are the same summaries that are presented in the text, it's good to see how these summaries relate to the knowledge objectives. In many cases, in fact, the summary items have a one-to-one relationship with the knowledge objectives.

## Terms

The terms list for each chapter contains all of the new terms that are presented in the chapter. After reading a chapter, you can scan the list to make sure you have a general understanding of what each term means. Then, if necessary, you can review the terms you don't understand.

When you use these lists, please keep in mind that you don't need to be able to write definitions of the terms. In fact, you won't be tested on the meanings of terms that aren't required by the objectives. You should, however, be able use the major terms in conversation.

## Book exercises

For each chapter, the book presents one or more exercises. Some of these exercises guide you through the development of web pages, and some force you to apply what you've learned in new ways. If you can do all of the exercises, you should be able to meet the applied objectives for each chapter. These exercises are repeated in this Lab Manual so you have easy access to them while you use the book for reference.

To help you get the most practice in the least time, you start most of the exercises from code that can be downloaded from our website. Before you start the exercises, then, you need to download these files from our website and install them on your system. To do that, you can follow the procedures in appendix A of the book.

When you download the files, you will see that they also include the solutions to the exercises. That way, you can peek at a solution to get over a learning block when you're working on your own. You can also compare your solutions to the book solutions to see other ways that a feature can be coded. Of course, the more work you do on your own, the better you will master the essential web development skills.

## The Halloween case study

The case study for this book requires that you build a website for a Halloween store, one chapter at a time. For each chapter, you'll either enhance the web pages you've already built or add new pages to the site. As you do the exercises for this case study, you will build a website that has most of the features of a real-world website. And being able to do that, of course, is the ultimate proof of your HTML5 and CSS3 mastery.

In contrast to the book exercises, the Halloween exercises don't guide you through the process of adding an enhancement or a web page. Instead, the exercises give you general specifications for what needs to be done. As a result, you may have to figure out some of the details on your own. That's realistic, of course, because specifications in the real world usually don't provide all of the implementation details.

All of the files that you need for building the Halloween website are in a folder named halloween_exercises. You will get this folder from your instructor.

## Short exercises

The short exercises are designed to let you practice a specific skill that has been presented in the chapter. They are available for selected chapters, and you should be able to do each short exercise in from 5 to 45 minutes. In some cases, your instructor will assign a short exercise in lab as a quiz or test.

The files that you need for doing the short exercises are in a folder named short_exercises. And here again, you will get this folder from your instructor.

## Student projects

As an alternative to or in addition to the Halloween case study, your instructor may have you do one of the student projects. Each of these projects asks you to create a small website of your own. In other words, you create or choose the content and you decide what images to use.

You can start a student project after you complete the first eight chapters of the book. Then, you can enhance the initial version of your website as you read the chapters in sections 2 and 3 of the book. You'll find generic descriptions for the student projects at the end of this Lab Manual.

## Conclusion

Since there's so much to learn and so little time, we have included only those activities that help you master the essential skills in the most efficient way possible. So, after you read each chapter in the book, you can review the objectives, chapter summaries, and terms lists to make sure you understand the critical concepts and terms.

After you do that, the exercises, Halloween case study, and student projects will provide all the practice you need for mastering web development with HTML5 and CSS3. Although you won't have time to do all of the book exercises, Halloween exercises, and projects for this book, the more you do, the more skilled you'll become.

## Chapter 1

# Introduction to web development

### How web applications work

The components of a web application
How static web pages are processed
How dynamic web pages are processed
A survey of web browsers and server-side scripting languages
How client-side JavaScript fits into web development

### An introduction to HTML and CSS

The HTML for a web page
The CSS for a web page
A short history of the HTML and CSS standards

### Tools for web development

Text editors for HTML and CSS
IDEs for web development
FTP programs for uploading files to the web server

### How to view deployed web pages

How to view a web page
How to view the source code for a web page

### Five critical web development issues

Users and usability
Cross-browser compatibility
User accessibility
Responsive Web Design
Search engine optimization

# Objectives

### Applied

1. Load a web page from the Internet or an intranet into a web browser.
2. View the source code for a web page in a web browser.

### Knowledge

1. Describe the components of a web application.
2. Distinguish between the Internet and an intranet.
3. Describe HTTP requests and responses.
4. Distinguish between the way a web server processes static web pages and dynamic web pages.
5. Name the five major web browsers.
6. Describe the use of JavaScript.
7. Distinguish between HTML and CSS.
8. Explain how you deploy a website on the Internet.
9. Describe the components of an HTTP URL.
10. Describe these five web development issues: usability, cross-browser compatibility, user accessibility, search engine optimization, and Responsive Web Design.

## Summary

- A web application consists of clients, a web server, and a network. *Clients* use *web browsers* to request web pages from the web server. The *web server* returns the requested pages.
- A *local area network* (*LAN*) connects computers that are near to each other. This is often called an *intranet*. In contrast, a *wide area network* (*WAN*) uses *routers* to connect two or more LANs. The *Internet* consists of many WANs.
- To request a web page, the web browser sends an *HTTP request* to the web server. Then, the web server retrieves the HTML for the requested page and sends it back to the browser in an *HTTP response*. Last, the browser *renders* the HTML into a web page.
- A *static web page* is a page that is the same each time it's retrieved. The file for a static page has .html or.htm as its extension, and its HTML doesn't change.
- The HTML for a *dynamic web page* is generated by a server-side program or script, so its HTML can change from one request to another.
- JavaScript is a scripting language that is run by the JavaScript engine of a web browser. It provides for client-side processing.
- *HTML* (*HyperText Markup Language*) is the language that defines the structure and contents of a web page. *CSS* (*Cascading Style Sheets*) are used to control how the web pages are formatted.
- To develop web pages, you can use a text editor like Aptana Studio 3 or an *Integrated Development Environment* (*IDE*) like Adobe Dreamweaver CC.
- To *deploy* (or *publish*) a website on the Internet, you need to transfer the folders and files for your site from your computer to a web server with Internet access. To do that, you use an *FTP program* that uses *File Transfer Protocol*.
- To view a web page on the Internet, you can enter the *URL* (*Uniform Resource Locator*) into a browser's address bar. A URL consists of the *protocol*, *domain name*, *path*, and filename.
- To view a web page that's on your own computer or server, you can use the browser's File→Open (IE) or File→Open File (Firefox) command. With Chrome, you can use your file explorer to locate the file for the page and then drag it into the browser window.
- To view the HTML for a web page, right-click on the page and select View Source or View Page Source. Then, to view the CSS for a page, you can click on its link in the source code or enter its URL in the address bar.
- Five critical issues for web development are *usability*, *cross-browser compatibility*, *user accessibility*, *search engine optimization* (*SEO*), and *Responsive Web Design* (*RWD*). In this book, you'll learn the HTML and CSS techniques that are related to these issues.

## Terms

World Wide Web
Internet
client
web browser
web server
network
local area network (LAN)
intranet
wide area network (WAN)
router
Internet service provider (ISP)
Internet exchange point (IXP)
static web page
HTTP request
HTTP (HyperText Transfer Protocol)
HTML (HyperText Markup Language)
HTTP response
render a web page
dynamic web page
application server
database server
round trip
web application
scripting language
server-side processing
Apache web server
IIS (Internet Information Services)
JavaScript
client-side processing
image swap
image rollover
JavaScript engine
HTML document
DOCTYPE declaration
HTML element
tag
opening tag
closing tag
attribute
CSS (Cascading Style Sheets)
style sheet
external style sheet
rule set
selector
declarative block
declaration
rule
XHTML (eXtensible HTML)
W3C (World Wide Web Consortium)
WHATWG (Web Hypertext Application Technology Working Group)
IDE (Integrated Development Environment)
suite
deploy
publish
FTP program
FTP (File Transfer Protocol)
FTP plugin
ISP (Internet Service Provider)
web hosting
IP address
domain name
URL (Uniform Resource Locator)
protocol
domain name
path
usability
cross-browser compatibility
user accessibility
SEO (Search Engine Optimization)
RWD (Responsive Web Design)
native mobile applications

## About the book exercises

At the end of each chapter in the book, you'll find exercises that are designed to let you practice what you've just learned. In many cases, these exercises also force you to apply what you've just learned in a different way. That's the true test of whether you're learning the skills that are needed on the job. The book exercises are repeated in this lab manual so you can refer to them while you're using the book for reference.

## Before you do the exercises for this book…

Before you do the book exercises, you should install the Chrome and Firefox browsers and the applications and exercises for this book. If you're going to use Aptana Studio 3 as your text editor, you should also download and install that product. The procedures for doing all three are in appendix A.

# Exercise 1-1 Visit some Internet websites

In this exercise, you'll visit some Internet websites and view the source code for those sites.

### Visit the author's website with Chrome

1. Start Chrome.
2. Enter www.modulemedia.com into the address bar and press the Enter key. That should display the home page for this website. Here, JavaScript is used to rotate the images at the top of the page.
3. If you're using a Windows system, enter "modulemedia" into the address bar, hold down the Ctrl key, and press the Enter key. If this works, it will add www. and .com to your entry. This is a quick way to enter the URL for a .com address.
4. Use one of the techniques in figure 1-13 to view the source code for the home page. Here, the three link elements identify the CSS files that do the formatting for the page. This is followed by three script elements that identify JavaScript files.
5. If you scroll through this code, it probably looks overwhelming, even though this site is relatively simple. By the time you complete this book, though, you should understand the HTML and CSS that it uses.
6. Click on the underlined value of the href attribute in the first link element. That should open the first CSS file for this page. This shows how easy it is to access the HTML and CSS code for many (but not all) sites.

### Visit other websites

7. Go to www.landsend.com, find a page like the one in figure 1-5, and experiment with the image swaps and rollovers. Those are done by JavaScript after all of the images are loaded with the page.
8. Use Chrome to visit other websites and view the source code for those sites. When you're through experimenting, go to the next exercise.

## Exercise 1-2 View the application for this chapter

In this exercise, you'll visit the book page that was used as an example in figures 1-6 and 1-7.

### Open the book page in Firefox

1. Start Firefox if it isn't already open.
2. Use the File→Open File command to open this HTML file:

   **`c:\murach\html5_css3_2\book_apps\ch01\dreamweaver_book.html`**
3. Right-click on the page and choose View Page Source to display the source code for this page.
4. Click on book.css in the link element in the HTML code to display the CSS file for this page.

### Open the book page in IE (if you're using a Windows system)

5. Start Internet Explorer if it isn't already open.
6. Use the File→Open command to open the same HTML file as in step 2.
7. Use the View→Source command to display the source code for this page.
8. Click on book.css in the link element in the source code to try to display the CSS file for this page. Note that this doesn't work with IE.

## Exercise 1-3 View other applications and examples

1. Start Chrome if it isn't already open.
2. Use your file explorer to locate this file:

   **`c:\murach\html5_css3_2\book_apps\ch07\town_hall\index.html`**

   Then, drag the index.html file to the browser window to display the web page.
3. Click on the link for Scott Sampson to see that page. This is the website that's presented at the end of chapter 7, and this gives you some idea of what you'll be able to do when you complete that chapter. Note, however, that only the Scott Sampson link has been implemented for this website.
4. Open this file in the Chrome browser:

   **`c:\murach\html5_css3_2\book_examples\ch05\12_gradients.html`**

   This is the example for figure 5-12 in chapter 5. Here, the 12 in the filename refers to the figure number. As you do the exercises for this book, you may want to copy code from the examples to your exercise solution.
5. Open this file in the Chrome browser.

   **`c:\murach\html5_css3_2\book_examples\ch15\05_print_page\index.html`**

   This is the example for figure 15-5 in chapter 15. Note that the figure number is in the folder name this time. This is true for all of the examples that require more than one file.

## Exercise 1-4 Learn more about HTML5, accessibility, SEO, and RWD

1. Go to www.w3.org. This is the website for the group that develops the HTML5 and CSS3 standards. It provides all sorts of useful information including HTML5 documentation.
2. Go to www.whatwg.org. This is the website for a community that is interested in evolving HTML and its related technologies. It also provides all sorts of useful information including HTML5 documentation.
3. Go to www.html5test.com, and view the HTML5 rating for your browser. Then, review the other browser data that this site provides.
4. Go to www.webaim.org. Then, review the information about accessibility that this site provides.
5. Use Google to search for "search engine optimization". Then, click on the first google.com link for this subject to get more information about it. You may also want to download and print the PDF on SEO that Google offers.
6. Use Google to search for "responsive web design". Then, click on one or more of the links to see what type of information is available.

## Chapter 2

# How to code, test, and validate a web page

### The HTML syntax

The basic structure of an HTML document
How to code elements and tags
How to code attributes
How to code comments and whitespace

### The CSS syntax

How to code CSS rule sets and comments
How to code basic selectors

### How to use Aptana to work with HTML and CSS files

How to create a project
How to open an HTML file
How to start a new HTML file
How to edit an HTML file
How to open or start a CSS file
How to edit a CSS file
How to preview and run an HTML file

### How to test, debug, and validate HTML and CSS files

How to test and debug a web page
How to validate an HTML file
How to validate a CSS file

# Objectives

### Applied

1. Use a text editor like Aptana Studio 3 to create and edit HTML and CSS files.
2. Test an HTML document that's stored on your computer or a local server by loading it into a browser.
3. Validate an HTML document using a website like W3C Markup Validation Service or Aptana's validation feature.

### Knowledge

1. Describe the use of the head and body elements in an HTML document.
2. Describe these types of HTML tags: opening, closing, and empty.
3. Describe the use of attributes within HTML tags.
4. Describe the use of HTML comments and whitespace.
5. Describe the components of a CSS rule set.
6. Describe the use of these types of CSS selectors: type, id, and class.
7. Explain how and why you would start a new HTML or CSS file from a template.
8. Describe three ways to run a web page and one way to retest a page after you've changed the source code for the page.
9. Describe two benefits of validating HTML files.

## Summary

- An *HTML document* consists of a *DOCTYPE declaration* that indicates what version of HTML is being used and a *document tree* that contains the *HTML elements* that define the content and structure of a web page.
- The *root element* in a document tree is the html element, which always contains a head element and a body element. The head element provides information about the page, and the body element provides the structure and content for the page.
- Most HTML elements consist of an *opening tag* and a *closing tag* with *content* between these tags. When you *nest* elements with HTML, the inner set of tags must be closed before the outer set.
- *Attributes* can be coded in an opening tag to supply optional values. An attribute consists of the name of the attribute, an equal sign, and the attribute value. To code multiple attributes, you separate them with spaces.
- An *HTML comment* can be used to describe or explain a portion of code. Because comments are ignored, you can also use comments to *comment out* a portion of HTML code so it isn't rendered by the browser. This can be helpful when you're testing your HTML code.
- *Whitespace* consists of characters like tab characters, line return characters, and extra spaces that are ignored by browsers. As a result, you can use whitespace to indent and align your code.
- A *CSS rule set* consists of a selector and a declaration block. The *selector* identifies the HTML elements that are going to be formatted. Three of the common CSS selectors select by element (called a *type selector*), ID, and class.
- The *declaration block* in a CSS rule set contains one or more *declarations* that do the formatting. Each declaration (or *rule*) consists of a *property*, a colon, a *value*, and a semicolon.
- *CSS comments* work like HTML comments. However, CSS comments start with /* and end with */, and HTML comments start with <!-- and end with -->.
- Aptana is a text editor that can be used to edit HTML or CSS code. To help you read the code, Aptana displays the syntax components with different colors. It also provides auto-completion lists and error checking that detects common entry errors.
- When you start a new HTML or CSS file, it's best to start from a *template* or an old file that's similar to the new file that you're going to create.
- To *test* an HTML file, you run it on all of the browsers that your clients may use. Then, if you discover problems, you need to *debug* the code and test it again.
- To *validate* an HTML or CSS file, you can use a program or website for that purpose. Often, a validation program will detect errors in a file, even though the web page displays the way you want it to on all browsers.

## Terms

syntax
HTML document
DOCTYPE declaration
document tree
HTML element
root element
opening tag
closing tag
content of an element
empty tag
nested elements
attribute
Boolean attribute
comment
comment out
whitespace
rule set
selector
declaration block
declaration
rule
property
value
type selector
template
Aptana project
testing
debugging
HTML validation
CSS validation

# Before you do the exercises for this book…

If you haven't already done it, you should install the Chrome and Firefox browsers and the applications, examples, and exercises for this book. If you're going to use Aptana Studio 3 as your text editor, you should also download and install that product. The procedures for doing all three are in appendix A.

# Exercise 2-1 Get started right with Aptana

This exercise is for readers who are going to use Aptana Studio 3 with this book. It guides you through the process of creating projects that provide easy access to the book applications, examples, and exercise starts that you've downloaded.

### Create the projects

1. Use the procedure in figure 2-7 to create a project for the book applications that are stored in this folder:

   **`c:\murach\html5_css3_2\book_apps`**

   This project should be named HTML5 Book Apps.

2. Use the same procedure to create a project named HTML5 Examples for the book examples that are stored in this folder:

   **`c:\murach\html5_css3_2\book_examples`**

3. Use the same procedure to create a project named HTML5 Exercises for the exercises that are stored in this folder:

   **`c:\html5_css3_2\exercises`**

### Use the projects that you've created

4. Use the drop-down list in the App Explorer to select the HTML5 Book Apps project. This provides access to all of the applications that are in this book.
5. Click on the ◢ symbol before ch02 to display the files in this folder. Then, double-click on the file named dreamweaver_book.html to open that file.
6. Delete the l in the opening tag for the h1 element and note how Aptana highlights this error. Then, undo this change. (To use the keyboard to undo a change, press Ctrl+Z.)
7. Start a new element after the h1 element and note how Aptana provides auto-completion. Then, undo this change and save the file if necessary.
8. Click on the Show Preview button shown in figure 2-13 to preview the file in Aptana. Then, click on the tab for the HTML file and click on the Run button shown in figure 2-13 to run the file in your default browser.
9. Use the drop-down list in the App Explorer to select the HTML5 Exercises project. This provides access to all of the starting files for the exercises.
10. Right-click on one of the tabs in the editor window, and select Close All to close all of the tabs. Then, experiment on your own if you like.

## Exercise 2-2 Edit and test the book page

In this exercise, you'll edit and test the book page that is shown in figure 2-14 and below. You should do this exercise whether you're using Aptana or another text editor.


### Open the HTML and CSS files for the book page

1. Start your text editor.
2. Open this HTML file in your text editor:

   **`c:\html5_css3_2\exercises\ch02\dreamweaver_book.html`**

   For Aptana users, this will be in the project named HTML5 Exercises.
3. Open the CSS file named book.css that's in the same folder. This should open in a new tab.

### Test the book page in Firefox and Chrome

4. Run the HTML file in Firefox.
5. Run the HTML file in Chrome. Is there a difference in the pages? If so, this illustrates why you need to test a web page in more than one browser.

### Modify the HTML and CSS code and test again

6. Go back to your text editor and click on the tab for the book.css file. Then, change the float property for the img element from left to right, and save the file. Now, test this change in both browsers.
7. Go back to the book.css file, and change the color for the h1 element to "red" and change the font-size to 180%. Then, save the file, and test the change in one or both browsers.

8. Go back to the dreamweaver_book.html file, and add a <p> element at the bottom of the page that has this content:

   ```
   For customer service, call us at 1-555-555-5555.
   ```

   Then, save the file and test this change in both browsers. When you do, you will notice that this paragraph displays differently in the two browsers.

9. Go to the dreamweaver_book.html file and add an id attribute with the value "service" to the <p> element that you just entered. Then, go to the book.css file and enter a rule set for the <p> element with that id. This rule set should use the color property to change the text to red, and it should use the clear property with a value of both to stop the floating before the paragraph. If you need help doing this, refer to figures 2-5 and 2-6. Now, save both the html and css files, and test these changes.

### Validate the HTML and CSS files

10. In the HTML file, delete the ending > for the img tag, and save the file. Then, go to the site in figure 2-15, and use the Validate by File Upload tab to validate the file. If you scroll down the page when the validation is done, you'll see 3 error messages, even though the file contains just one error.

11. In the CSS file, delete the semicolon for the color rule in the h1 rule set, and save the file. Then, go to the site in figure 2-16, and use the Validate by File Upload tab to validate the file. This time, you'll see 1 error message.

12. If you're using Aptana, note the error marker in the HTML file. Then, validate the HTML file by using the command in figure 2-15. This shows how much easier it is to validate files within the text editor or IDE that you're using.

13. If you're using Aptana, note the error marker in the CSS file. Then, validate the CSS file by using the command in figure 2-16. Note here that the validation found no errors, which differs from the results for step 11.

14. Now, undo the errors in the files, save the files, and validate the HTML page again. This time, it should pass with one warning. Then, test the web page one last time.

## Exercise 2-3 Start a new web page

In this exercise, you'll start HTML and CSS files from templates in this folder:

```
c:\html5_css3_2\exercises\ch02
```

For Aptana users, this will be in the project named HTML5 Exercises.

### Start your files from templates

1. Create a new HTML file named testpage.html from the template named template.html. If you're using Aptana, you can use the first procedure in figure 2-9. Otherwise, you can use the second procedure in this figure.
2. Create a new CSS file named testpage.css from the template named template.css. If you're using Aptana, you can use the first procedure in figure 2-11. Otherwise, you can use the second procedure in this figure.

### Add some content to the HTML file and test it

3. In the title element, change the content to "Test page".
4. In the link element, enter "testpage.css" for the value of the href attribute.
5. Add an h1 element to the body of the HTML file that says: "This is a test page." Then, save the file, and test the page by running it in Chrome, Firefox, or IE.
6. This illustrates how fast you can get started with a new web page when you start the html and css files from templates. Now, if you want to add more elements to the HTML or more rules or rule sets to the CSS, give it a try.
7. When you're through experimenting, close the files in your editor.

## Chapter 3

# How to use HTML to structure a web page

# Objectives

### Applied

1. Code a properly structured HTML web page using the HTML5 semantic elements and any of the other elements that are presented in this chapter.
2. Given the HTML for a web page, code a relative URL that refers to any file in the folder structure for the website.

### Knowledge

1. Describe the use of the title and meta elements in the head section of an HTML document.
2. Distinguish between a block element and an inline element.
3. Describe the use of these block elements: h1, h2, h3, and p.
4. Describe the use of these inline elements: br, i, b, sup, em, q, and strong.
5. Describe the use of character entities like   or ©.
6. Describe the use of these core attributes: id, class, and title.
7. Describe the use of the div and span elements in the HTML for a modern website.
8. Describe the use of these HTML5 semantic elements: header, main, section, article, nav, aside, footer, and figure.
9. Distinguish between absolute and relative URLs, and distinguish between root-relative and document-relative paths
10. Distinguish between the use of the <a> element and the img element.
11. Describe the accessibility guidelines for <a> elements and img elements.
12. Describe the two types of lists that you can create with HTML.

## Summary

- In the head section of an HTML document, the title element provides the text that's displayed in the browser's title bar. This is an important element for search engine optimization.
- One common use of the link element in the head section is to identify a custom icon called a *favicon* that appears in the browser's address bar. This icon may also appear in the browser tab for the document or as part of a bookmark.
- The meta elements in the head section provide *metadata* that is related to the page. Here, the charset metadata is required for HTML5 validation, and the description and keywords metadata should be coded because they can affect search engine optimization.
- *Block elements* are the primary content elements of a website, and each block element starts on a new line when it is rendered by a browser. Headings and paragraphs are common block elements.
- *Inline elements* are coded within block elements, and they don't start on new lines when they are rendered. Some common inline elements like <i> (for italics) and b (for bold) can be used to format text without implying any meaning. Whenever possible, though, you should use the inline elements that imply meaning.
- *Character entities* are used to display special characters like the ampersand and copyright symbols in an HTML document. In code, character entities start with an ampersand and end with a semicolon as in   (a non-breaking space).
- The core attributes that are commonly used for HTML elements are the id, class, and title attributes. The id attribute uniquely identifies one element. The class attribute can be used to identify one or more elements. And the title attribute can provide other information about an element like its tooltip text.
- Historically, the div element has been used to divide the code for an HTML document into divisions, and the span element has been used to identify portions of text so formatting can be applied to them.
- The HTML5 *semantic elements* provide a new way to structure the content within an HTML document. This makes it easier to code and format elements. In the long run, it may also improve the way search engines rank your pages.
- When you code an *absolute URL*, you code the complete URL including the domain name. When you code a *relative URL*, you can use a *root-relative path* to start the path from the root folder for the website or a *document-relative path* to start the path from the current document.
- The <a> element (or anchor element) is an inline element that creates a link that usually loads another page. By default, the text of an <a> element is underlined. Also, an unvisited link is displayed in blue and a visited link in purple.
- Lists are block elements that can be used to display both *unordered lists* and *ordered lists*. By default, these lists are indented with bullets before the items in an unordered list and numbers before the items in an ordered list.

- The img element is used to display an image file. The three common formats for images are *JPEG* (for photographs and scans), *GIF* (for small illustrations and logos), and *PNG*, which combines aspects of JPEG and GIF.

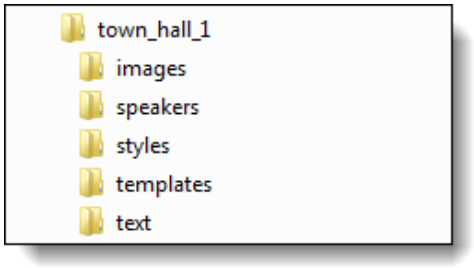
## Terms

favicon
metadata
block element
inline element
character entity
semantic elements
HTML5 semantics
absolute URL
relative URL
root-relative path
document-relative path
link
ordered list
unordered list
JPEG (Joint Photographic Experts Group)
GIF (Graphic Interchange Format)
PNG (Portable Network Graphics)

## About the book exercises for chapters 3 through 8

In the exercises for chapters 3 through 8, you'll develop a new version of the Town Hall website. This version will be like the one in the text, but it will have different content, different formatting, and different page layouts. Developing this site will give you plenty of practice, and it will also show you how similar content can be presented in two different ways.

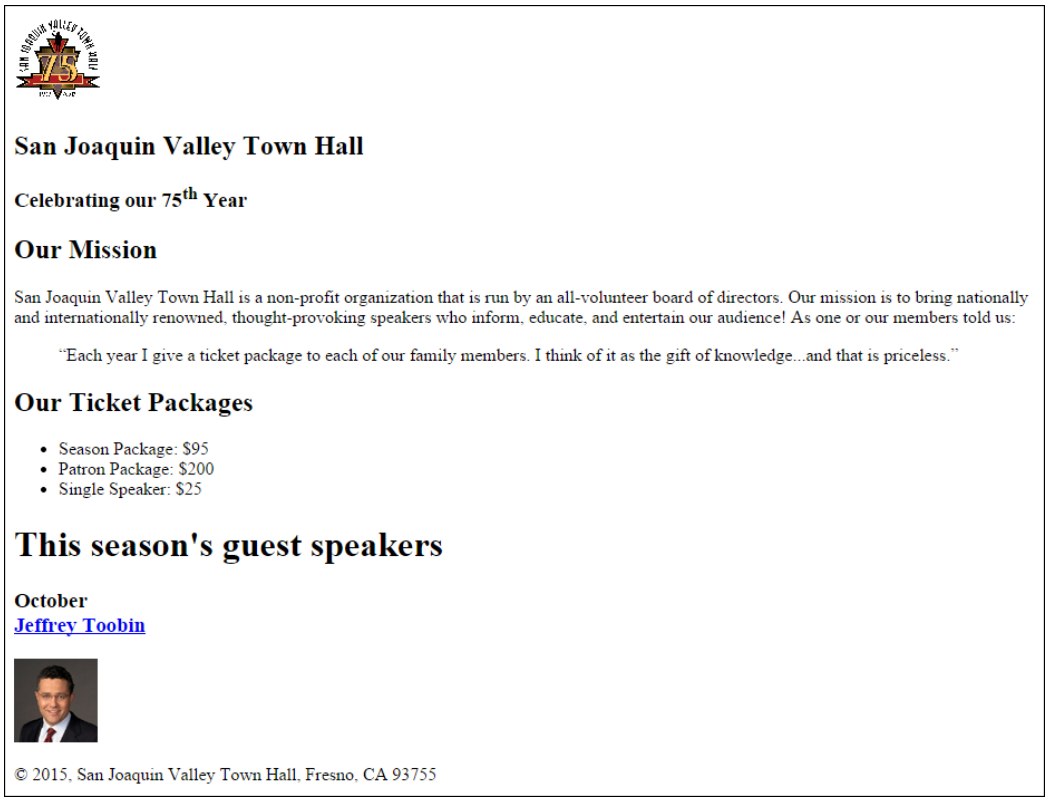
As you develop this site, you will use this folder structure:




This is a realistic structure with images in the images folder, speaker HTML pages in the speakers folder, CSS files in the styles folder, and template files in the templates folder. In addition, the text folder contains text files that will provide all of the content you need for your pages. That too is realistic because a web developer often works with text that has been written by someone else.

# Exercise 3-1 Enter the HTML for the home page

In this exercise, you'll code the HTML for the home page. When you're through, the page should look like this, but with three speakers:




### Open the starting page and get the contents for it

1. Use your text editor to open this HTML file:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

   Note that it contains the head section for this web page as well as a body section that contains header, main, and footer tags.

2. Use your text editor to open this text file:

   **`c:\html5_css3_2\exercises\town_hall_1\text\c3_content.txt`**

   Note that it includes all of the text that you need for this web page.

### Enter the header

3. Code the img element that gets the image at the top of the page from the images directory. To locate the image file, use this document-relative path: images/town_hall_logo.gif. Be sure to include the alt attribute, and set the height attribute of the image to 80.
4. Copy the text for the first two headings from the txt file in the text folder into the header of the HTML file. Next, apply the h2 and h3 elements.

5. Test this page in Chrome. If necessary, correct the HTML and test again.

### Enter the content for the main element

6. Copy all of the content for the main element from the txt file into the HTML file. Then, add an h1 tag to the heading "This season's guest speakers", and add h2 tags to these headings "Our Mission" and "Our Ticket Packages".
7. Add <p> tags to the first block of text after the "Our Mission" heading, and add blockquote tags to the second block of text as shown above.
8. Add the ul and li tags that are needed for the three items after the "Our Ticket Packages" heading. Then, test these changes and make any adjustments.
9. Format the name and month for the first speaker after the "This season's guest speakers" heading as one h2 element with a <br> in the middle that rolls the speaker's name over to a second line. Then, test and adjust.
10. When that works, do the same for the next two speakers.
11. Enclose the name for each speaker in an <a> tag. The href attribute for each tag should refer to a file in the speakers subfolder that has the speaker's last name as the filename and html as the file extension. In other words, the reference for the first speaker should be:

    ```
    speakers/toobin.html
    ```

12. After the h2 element for each speaker, code an img element that displays the image for the speaker, and be sure to include the alt attribute. The images are in the images subfolder, and the filename for each is the speaker's last name, followed by 75 (to indicate the image size), with jpg as the extension. So to refer to the first speaker's file, you need to use a document-relative path like this: images/toobin75.jpg. Now, test and adjust.

### Enter the footer

13. Copy the last paragraph in the txt file into the footer of the HTML file. Then, enclose the text in a <p> element.

### Add character entities and formatting tags

14. Use character entities to add the quotation marks at the start and end of the text in the blockquote element.
15. Use a character entity to add the copyright symbol to the start of the footer.
16. Add the sup tags that you need for raising the *th* in the second line of the header (as in $75^{th}$). Then, test these enhancements.

### Test the links, validate the HTML, and test in Firefox and IE

17. Click on the link for the speaker page. This should display a page that gives the speaker's name and says "This page is under construction". If this doesn't work, fix the href attribute in the link and test again. To return to the first page, you can click the browser's Back button.

18. Open the toobin.html file that's in the speakers subfolder. Then, add a link within a <p> element that says "Return to index page." To refer to the index.html file, you'll have to go up one level in the folder structure with a document-relative path like this: ../index.html. Now, test this link.
19. Validate the HTML for the index page as shown in figure 2-15 of chapter 2. This should indicate one warning but no errors. If any errors are detected, fix them and validate the HTML again.
20. Test the index page in the Firefox browser. If necessary, fix any problems and test again in both Chrome and Firefox.
21. Test the index page in the IE browser. If necessary, fix any problems and test again in all three browsers.

## The Halloween Case Study

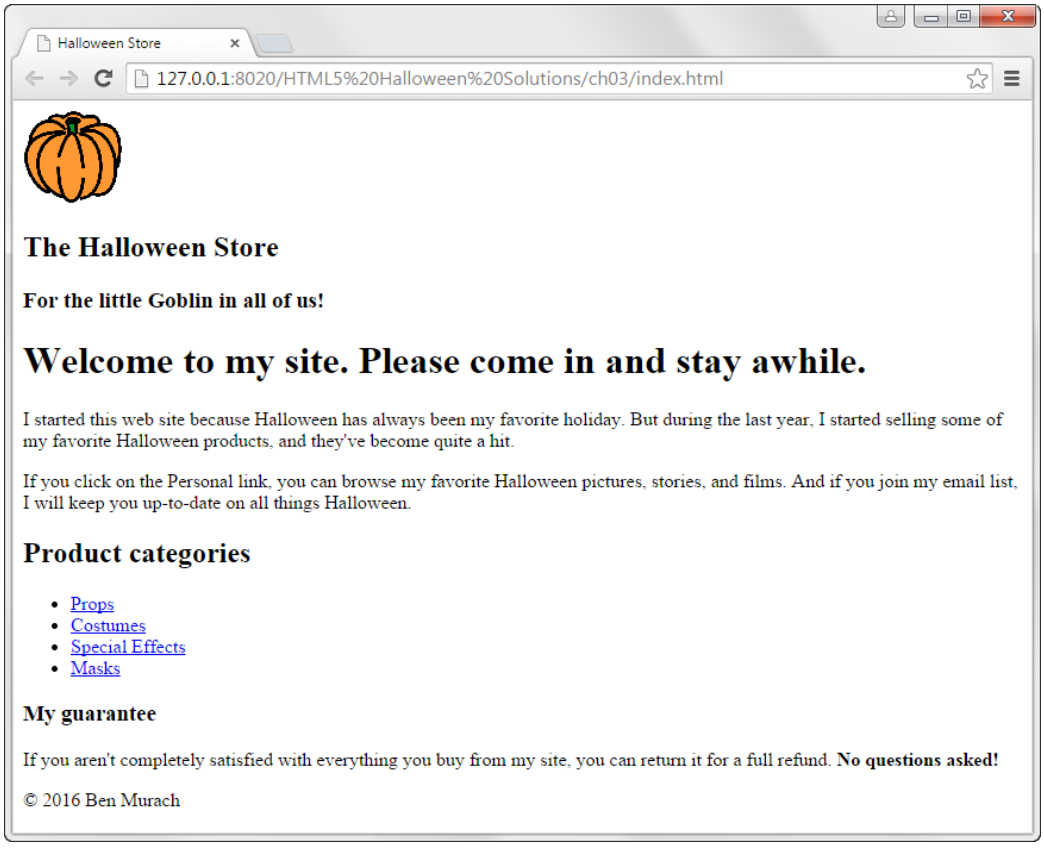
The Halloween Case Study guides you through the development of a Halloween website that consists of several pages. Unlike the exercises in the book, the Case Study exercises don't give you step-by-step instructions. Instead, the description for each exercise includes an image of how the page should appear in a browser, along with specifications that you can complete in whatever sequence you prefer.

These specifications are detailed enough for you to know what needs to be done, but you'll need to use your best judgment on how to code many of the details. To do that, write the code in the way that you think is best, based on the skills that are presented in the book.

The starting files for the Halloween website are in the folder named halloween_exercises that your instructor will provide. This folder includes starting HTML and CSS files for the home page, along with all supporting files such as image, media, JavaScript, and text files. The case study will instruct you on when to use each of these files.

# Halloween 3 Create the home page

In this exercise, you'll code the HTML for the home page. When you're through, the page should look like this:




### Specifications

- Create the home page from the index.html file that's in the root folder for the project. This file includes just the basic HTML elements.
- Add the HTML5 semantics to structure the page so it includes a header, the main content, and a footer.
- Add an image and two levels of headings to the header. You'll find the image, named pumpkin.gif, in the images folder. You'll find the text for the header as well as the for the other areas of the page in the index.txt file in the text folder.
- Add three levels of headings and the required text for the main content. Boldface the last sentence of the guarantee.
- Code the product categories as an unordered list with links to other pages. Because these pages don't exist, you can name them anything you like. However, you should assume that they will be stored in the products folder.
- Add the required text to the footer, using a character entity for the copyright symbol.

## About the short exercises

The short exercises are designed to let you practice a specific skill that has been presented in the chapter. They are available for selected chapters. You should be able to do each short exercise in from 5 to 45 minutes. In some cases, your instructor will assign a short exercise in lab as a quick quiz or test.

## Guidelines for doing the short exercises

- Do the exercise steps in sequence. That way, you will work from the most important tasks to the least important.
- Feel free to copy and paste code from the book applications, book examples, or exercises that you've already done.
- Use your book as a guide to coding.
- If you are doing an exercise in class with a time limit set by your instructor, do as much as you can in the time limit.
- All of starting files for the short exercises will be in the short_exercises folder that your instructor will provide.

**Note**

Because each short exercise is independent of the others, some of the links in the starting pages won't work. To provide for that, the href attribute for each of those links is coded as a pound sign (#). That way, you won't get a "file not found" error if you click the link. Instead, you'll just stay on the same page.

## Short 3-1 Create an HTML page for a speaker

In this exercise, you'll create a semantically correct, HTML page that looks like the one below. Estimated time: 20-30 minutes.

### Andrew Ross Sorkin, author of *Too Big to Fail*

**November 2015**

New York Times columnist and author, Andrew Ross Sorkin, has been described as "the most famous financial journalist of his generation." His book, *Too Big to Fail*, was the first behind-the-scenes account of the financial crisis that led to our current recession.

The Economist, The Financial Times, and Business Week all named *Too Big To Fail* one of the best books of the year. The book was published by Viking October 20, 2009. The book was adapted as a movie by HBO Films and premiered on HBO on May 23, 2011.

#### The cast of the movie: *Too Big to Fail*

1. William Hurt as Hank Paulson
2. Paul Giamatti as Ben Bernanke
3. Billy Crudup as Timothy Geithner

#### Movie reviews

> "Lots of heat but not very much light. It came across as a kind of daytime soap opera version showing the personalities of the players rather than dealing with the nuts and bolts of what really happened. Even that would not have been so bad but the movie tries to portray lots of these folks as sympathetic figures just trying to do the right thing. "
>
> "I am watching this for the 3rd time, trying to understand how this large group of capitalist pigs could royally screw their families, descendants, neighbors, countrymen and the rest of the whole world with absolutely no compassion."

Prepared by: Your Name

Date: Today's Date

1. Open this HTML file:

   **`short_exercises\town_hall\speakers\c03x_sorkin.html`**

   Note that it contains a head section and all the copy for the text in the body of the page, but the HTML tags haven't yet been applied to that text.

2. Apply the HTML tags to the text so the text will look like it does in the page above. Remember too that these tags should be semantically correct. The last two lines should be coded within a footer element, and everything else should be within a main element.

3. Add the image for the speaker: sorkin_desk260.jpg, which is 260 pixels wide.

4. Use character entities or inline formatting tags to add the quotation marks and italics that this page requires.

## Chapter 4

# How to use CSS to format the elements of a web page

### An introduction to CSS

Three ways to provide CSS styles for a web page
Two ways to provide for browser compatibility

### How to specify measurements and colors

How to specify measurements
How to specify colors
How to use the CSS3 color specifications

### How to code selectors

How to code selectors for all elements, element types, ids, and classes
How to code relational selectors
How to code combinations of selectors
How to code attribute selectors
How to code pseudo-class and pseudo-element selectors

### How to work with Cascading Style Sheets

How the cascade rules work
How to use the F12 tools to inspect the styles that have been applied

### How to work with text

How to set the font family and font size
How to set the other properties for styling fonts
How to set properties for formatting text
How to use CSS3 to add shadows to text
How to float an image so text flows around it

### A web page that uses external style sheets

The page layout
The HTML file
The CSS file

# Objectives

### Applied

1. Given an HTML document, create a CSS style sheet for formatting the web page using any of the types of selectors or rules that are presented in this chapter.
2. Given an HTML document with one or more CSS style sheets applied to it, use the developer tools for your browser to inspect the styles.
3. Given a selector in the CSS for an HTML document, describe what the selector applies to.

### Knowledge

1. Describe three ways to include CSS in a web page.
2. Explain why it's usually best to use an external style sheet for formatting a page.
3. Describe the changes you need to make to the HTML and CSS files if you want to use CSS to format the HTML5 semantic elements in versions of Internet Explorer before version 9.
4. Describe the purpose of the normalize.css style sheet.
5. Distinguish between absolute and relative units of measurement.
6. Describe three ways to specify color in CSS, and describe how CSS3 expands upon that.
7. Describe these types of selectors: universal, type, id, class, descendant, child, sibling, pseudo-class, and pseudo-element.
8. Describe one accessibility guideline for using pseudo-class selectors.
9. Explain how user style sheets, !important rules, and specificity are used in the cascade order for applying rule sets.
10. Describe these properties for styling fonts: font-family, font-style, font-weight, font-size, and line-height.
11. Describe these properties for formatting text: text-indent, text-align, text-decoration, and text-shadow.

## Summary

- If you're going to use a style sheet for more than one HTML document, it's usually best to use an *external style sheet*. However, you can also apply CSS by embedding a style sheet in the HTML or by using the style attribute of an element.
- To provide cross-browser compatibility with older versions of Internet Explorer when you use the HTML5 semantic elements, you can add a script element that accesses a JavaScript *shiv* to the head element of each HTML document.
- To make sure that all HTML elements are rendered the same in all modern browsers, you can include the normalize.css style sheet in your web pages.
- You can use *absolute measurements* like pixels or *relative measurements* like ems or percents to specify the CSS properties for sizes. For fonts, it's better to use relative measurements so the user can change the font sizes by changing the browser's default font size.
- Most graphic designers use *hex* for the *RGB values* that represent the colors that they want because that gives them the most control. However, most browsers also support 16 standard color names like red and blue.
- CSS3 lets you use *RGBA*, *HSL*, and *HLSA* values to specify colors. This gives the web designer more control over colors and transparency. CSS also provides 147 more keywords for colors.
- You can code CSS selectors for element types, ids, classes, relations, and attributes. You can also code selectors for combinations of these items.
- A *pseudo-class selector* can be used to apply CSS formatting when certain conditions occur or have occurred, like when the mouse hovers over an element or the focus is on an element. A *pseudo-element selector* lets you select a portion of text.
- If more than one style sheet is applied to an HTML document and two or more rule sets apply to the same element, the *cascade order* determines which rule set takes precedence. The first four levels of this order are: the important rules in a *user style sheet* for the browser; the important rules in a web page; the normal rules in a web page; and the normal rules in a user style sheet.
- If more than one rule set in a cascade level is applied to an element, the rule set with the highest specificity is used. But if the specificity is the same for two or more rule sets in a cascade level, the rule set that's specified last is used.
- The developer tools for a modern browser let you inspect all of the styles that have been applied to an element, including the styles that have been overridden.
- The default font for most browsers is a 16-pixel, serif font. However, because sans-serif fonts are easier to read in a browser, you normally change the font family. It's also good to change the font size to a relative measurement like a percent of the default font size.
- The colors and font properties of a parent element are *inherited* by the child elements. But those properties can be overridden by the rule sets for the children.
- The *shorthand property* for fonts can be used to apply all six of the font properties to an element: font-family, font-size, font-style, font-weight, font-variant, and line-height.

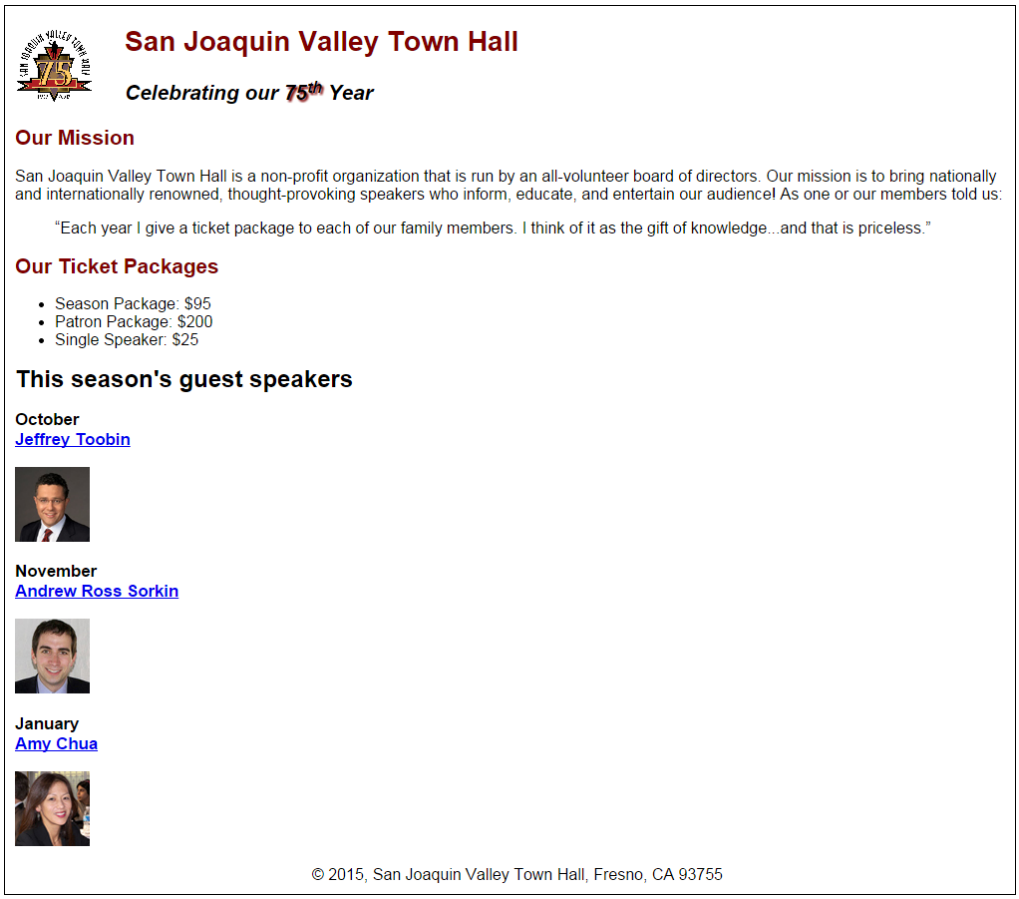
- Text properties can be used to indent, align, and decorate the text in a block element like a heading or paragraph. CSS3 also provides for adding shadows to text.
- You can use the float property of an image to float the image to the left. Then, the block elements that follow in the HTML flow to its right. To stop the flow, you can code the clear property for an element.

## Terms

external style sheet
embedded style sheet
inline style
shiv
shim
absolute unit of measure
relative unit of measure
absolute measurement
relative measurement
RGB value
hexadecimal (hex) value
inherited property
RGBA value
HSL value
HSLA value
universal selector
type selector
id selector
class selector
relational selector
descendant selector
adjacent sibling selector
child selector
general sibling selector
attribute selector
pseudo-class selector
pseudo-element selector
cascade order
user style sheet
F12 tools
font family
shorthand property

# Exercise 4-1 Format the Town Hall home page

In this exercise, you'll format the home page that you built in exercise 3-1 by using the skills that you've learned in this chapter. When you're through, the page should look like this.




## Open the index.html page and update the head section

1. Use your text editor to open the HTML file that you created for exercise 3-1:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

2. Use your text editor to open this HTML template file:

   **`c:\html5_css3_2\exercises\town_hall_1\templates\basic.html`**

   Then, copy the second and third link elements from the head section to the clipboard, switch to the index.html file, and paste these elements at the end of the head section.

3. Note that the first link element you just copied into the head is for the normalize.css style sheet in the styles subfolder. Next, complete the href attribute in the second link element so it refers to the main.css file in the styles subfolder. Then, close the template.

### Open the main.css file and format the header

4. Use your text editor to open this CSS file:

   `c:\html5_css3_2\exercises\town_hall_1\styles\main.css`

   Note that this file contains some of the CSS code that you'll need, including the rule set that specifies the font family and font size for the body, the rule set that floats the image in the header, and the rule set that clears the floating in the main element.

5. Add two rule sets for the header to the style sheet. The first one should be for the h2 element, and it should set the font size to 170%, set the color to #800000, and indent the heading 30 pixels. The second one should be for the h3 element, and it should set the font size to 130%, set the font style to italic, and indent the heading 30 pixels.

6. Test the HTML page in Chrome to make sure that the style sheets have been linked properly, the image has been floated, and the headings have been formatted correctly. If necessary, make corrections and test again.

### Format the main element and the footer

From now on, test each change right after you make it.

7. Add a rule set for the h1 elements within the main element that sets the font size to 150%.

8. Add a rule set for the h2 elements within the main element that sets the font size to 130% and the color to #800000.

9. Add a rule set for the h3 elements within the main element that sets the font size to 105%.

10. Add a rule set that italicizes any link that has the focus or has the mouse hovering over it.

11. Add a rule set that centers the <p> tag in the footer.

### Use the developer tools to review the styles for the page

12. Display the page in Chrome, and then press the F12 key to display the developer tools. Next, expand the main element in the Elements pane and click on one of the h2 elements.

13. Review the styles for the h2 element in the Styles pane, and notice how the font-family style for the html element in the normalize style sheet is overridden by the font-family style for the body element in the main style sheet. Also notice how the font-size style for the body element in the main style sheet and the h2 element in the user agent style sheet are overridden by the font-size style for the main h2 element in the main style sheet.

14. Click the icon in the developer tools toolbar that has a magnifying glass on it, and then click on the h2 element in the header to see that it's now selected in the Elements pane.

15. Review the styles for this h2 element to see that they're similar to the styles for the main h2 element. However, the font size for this element is smaller and it has a text indent.

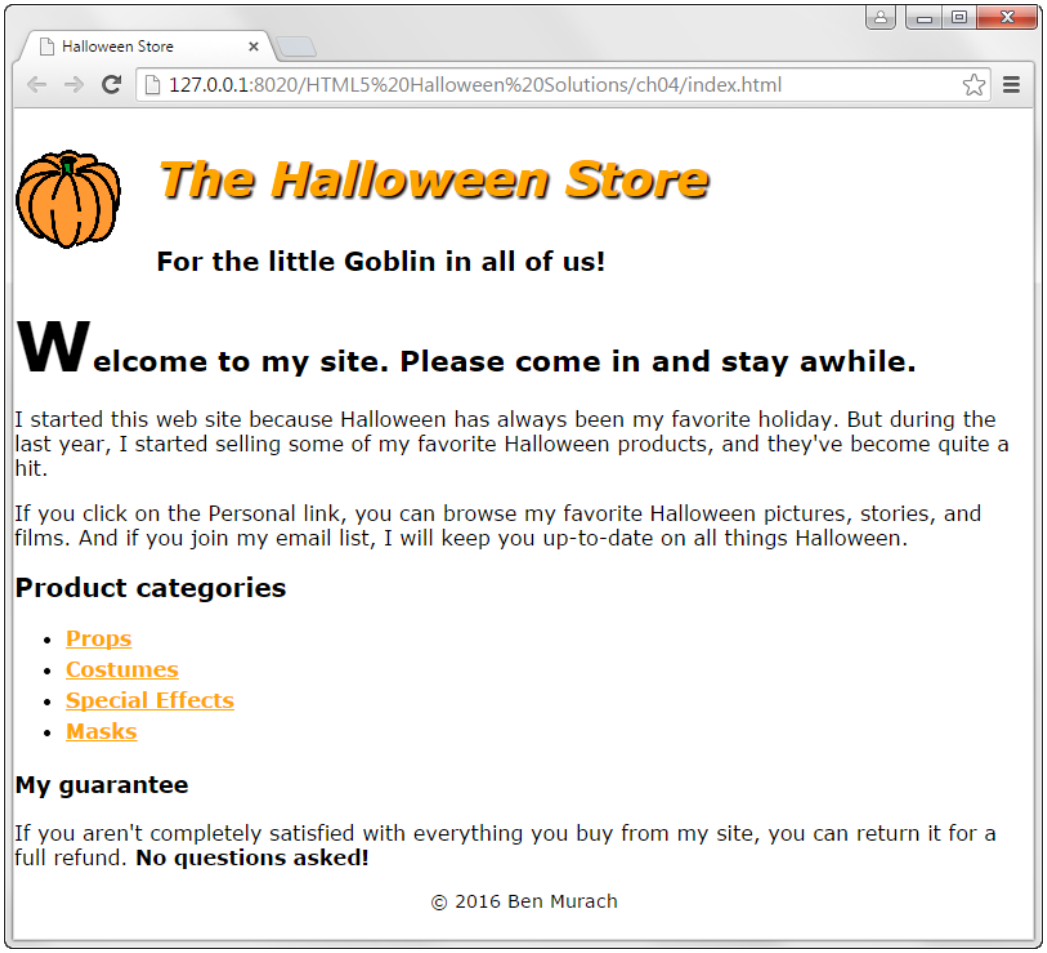
16. When you're done with the developer tools, close the panel by clicking the icon with an "X" on it in the upper right corner.

### Add the finishing touches and test in IE

17. Add a text shadow to the *75th* in the second heading in the header. To do that, enclose the *75th* in the HTML in an em or span element and give that element a class attribute with a value of "shadow". Then, create a rule set that uses a class selector (.shadow) for that class, and code a rule that adds a shadow to the text with #800000 as the color of the shadow.
18. Test the page in IE. There the text shadow won't work if you're using a version before version 10. But if you see any other formatting problems, make the corrections and test again in both Chrome and IE.

# Halloween 4 Format the home page

In this exercise, you'll create an external style sheet that you can use to format the Halloween Store home page that you created in exercise 3. When you're through, the page should look similar to this:




### Specifications

- Create the external style sheet for the home page from the main.css file that's in the styles folder. This file is currently blank.
- Set the default font for the document to a sans-serif font.
- Adjust the font sizes for all of the headings. In addition, format the first letter of the first heading in the main content so it's larger than the other letters in the heading.
- In the header, float the image to the left and indent both headings. In addition, change the color of the first heading to orange, italicize it, and add a black shadow.
- Format the links so they're displayed in boldfaced orange whether or not they've been visited. If a link has the focus or the mouse is hovering over it, though, it should be displayed in green.

- Format the list to increase the space between the list items.
- Reduce the font size for the footer and center it.
- Link the main style sheet that you just created as well as the normalize style sheet that's in the styles folder to the home page.

## Chapter 5

# How to use the CSS box model for spacing, borders, and backgrounds

# Objectives

### Applied

1. Given an HTML document, create CSS rule sets that use the CSS box model to apply spacing, borders, and backgrounds to the web page.

### Knowledge

1. Describe the use of the CSS box model.
2. Explain how the CSS box model can be used to control the spacing between the headings and paragraphs on a page.
3. Describe the effect of "collapsed margins".
4. Describe the use of a reset selector.
5. Describe these properties for a block element in a box model: height, width, margin, padding, border, background color, and background image.
6. Describe these CSS3 features for formatting boxes: rounded corners, shadows, background gradients.

## Summary

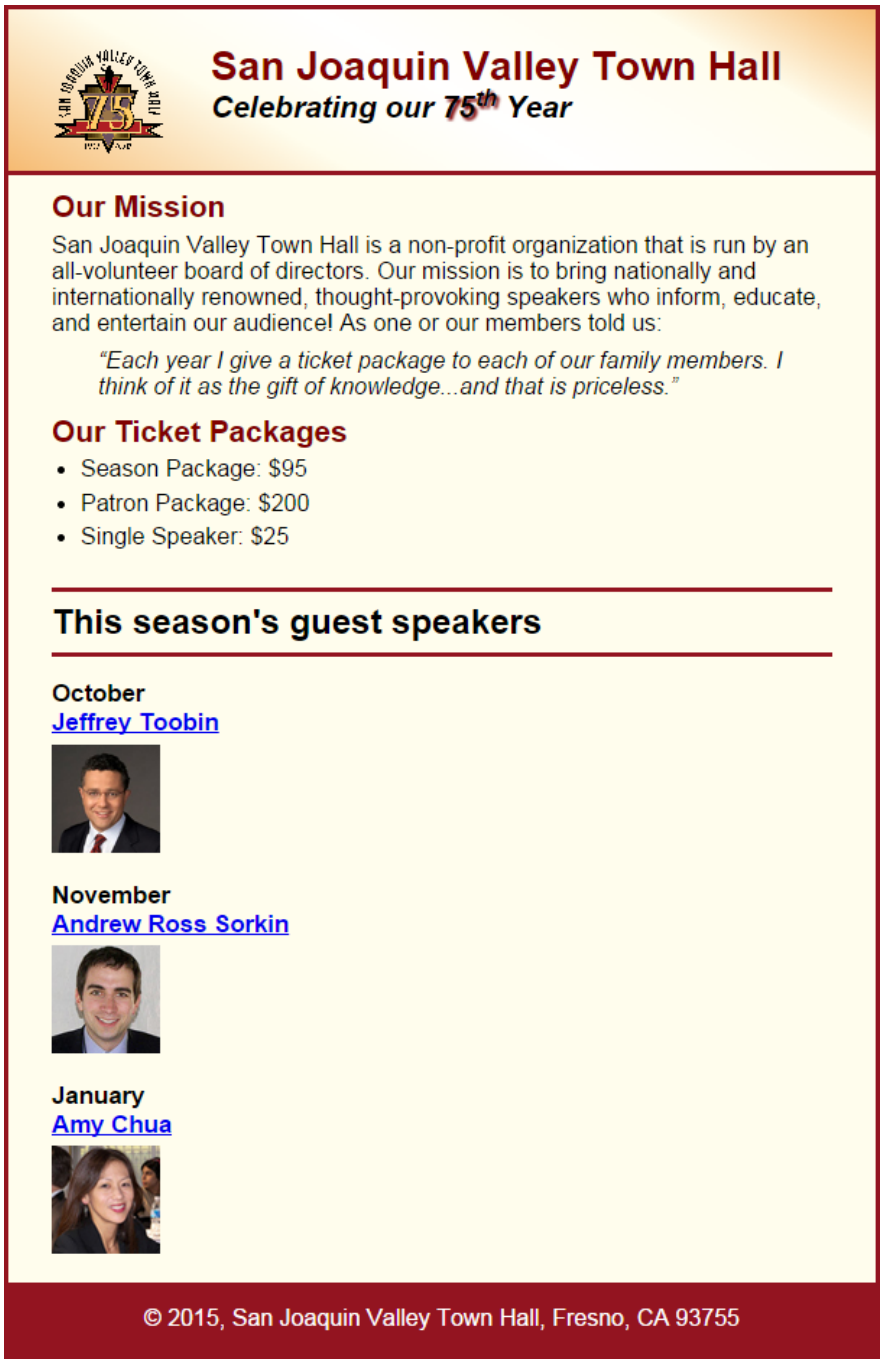
- The CSS *box model* refers to the box that a browser places around each block element as well as some inline elements. Each box includes the content of the element, plus optional padding, borders, and margins.
- To set the height and width of a content area, you can use absolute measurements like pixels or relative measurements like percents. If you use a percent, the percent applies to the block that contains the box you're formatting.
- You can set the *margins* for all four sides of a box. Because different browsers have different default values for margins, it's good to set the margins explicitly.
- If you specify a bottom margin for one element and a top margin for the element that follows, the margins are *collapsed* to the size of the largest margin.
- Like margins, you can set the *padding* for all four sides of a box. One way to avoid margin collapse is to set the margins to zero and use padding for the spacing.
- A *border* can be placed on any of the sides of a box. That border goes on the outside of the padding for the box and inside any margins, and you can set the width, style, and color for a border.
- When you set the *background* for a box, it is displayed behind the content, padding, and border for the box, but not behind the margins. The background can consist of a color, an image, or both.
- You can use CSS3 to *round corners* and add *shadows* to borders. You can also use CSS3 to provide *linear gradients* as backgrounds.

## Terms

box model
padding
margin
fixed layout
containing block
shorthand property
collapsed margins
reset selector
border
background
rounded corners
shadows
linear gradient

## Exercise 5-1 Enhance the Town Hall home page

In this exercise, you'll enhance the formatting of the Town Hall home page that you formatted in exercise 4-1. When you're through, the page should look like this:



### Open the HTML and CSS files and start enhancing the CSS

1. Use your text editor to open the HTML and CSS files that you created for exercise 4-1:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

   **`c:\html5_css3_2\exercises\town_hall_1\styles\main.css`**

2. Enhance the rule set for the body, by setting the width to 600 pixels, setting the top and bottom margins to 0 and the right and left margins to auto, and adding a 3-pixel, solid border with #931420 as its color. Then, test this change in Chrome. If the page isn't centered with a border, make the required corrections.
3. Add one more rule to the body that sets the background color to #fffded. Then, test this change, and note that the entire window is set to the background color, not just the body.
4. To fix this, code a rule set for the html element that sets the background color to white. Then, test to make sure that worked.

### Add the other borders and another background color

From this point on, test each change right after you make it. If you have any problems, use the developer tools as shown in figure 4-11 to help you debug them.

5. Add a bottom border to the header that's the same as the border around the body.
6. Add top and bottom borders to the h1 heading in the main element. Both borders should be the same as the borders for the header and footer.
7. Set the background color of the footer to the same color as the borders, and then set the font color for the paragraph in the footer to white so it's easier to read.

### Get the padding right for the header, section, and footer

At this point, you have all of the borders and colors the way they should be, so you just need to set the margins and padding. In the steps that follow, you'll start by adding a reset selector to the CSS file. Then, with one exception, you'll use padding to get the spacing right.

8. Add a reset selector like the one in figure 5-8 to the CSS file. When you test this change, the page won't look good at all because the default margins and padding for all of the elements have been removed.
9. For the header, add 1.5 ems of padding at the top and 2 ems of padding at the bottom. Then, delete the text-indent rules for the h2 and h3 elements in the header, and add 30 pixels of padding to the right and left of the image in the header. When you test these changes, you'll see that the heading looks much better.
10. For the main element, add 30 pixels of padding to the right and left.

### Get the padding right for the headings and text

11. In the main element, set the padding for the headings and text as follows:

| Element | Padding |
|---|---|
| h1 | .3em top and bottom |
| h2 | .5em top, .25em bottom |
| h3 | .25em bottom |
| img | 1em bottom |
| p | .5em bottom |
| blockquote | 2em right and left |
| ul | .25em bottom, 1.25em left |
| li | .35em bottom |

12. Set the padding for the top and bottom of the paragraph in the footer to 1em.
13. Test these changes. At this point, all of the spacing should look right.

### Add the finishing touches and test in IE

14. Italicize the blockquote element to make it stand out.
15. Add a linear gradient as the background for the header. The one that's shown uses #f6bb73 at 0%, #f6bb73 at 30%, white at 50%, #f6bb73 at 80%, and #f6bb73 at 100% as its five colors at a 30 degree angle. But experiment with this until you get it the way you want it.
16. Do one final test to make sure that the page looks like the one at the start of this exercise. Then, experiment on your own to see if you can improve the formatting. For instance, you may want to add some space between the author names and their images.
17. When you're through experimenting, test the page in IE. If you see any problems, fix them and test again in both Chrome and IE.

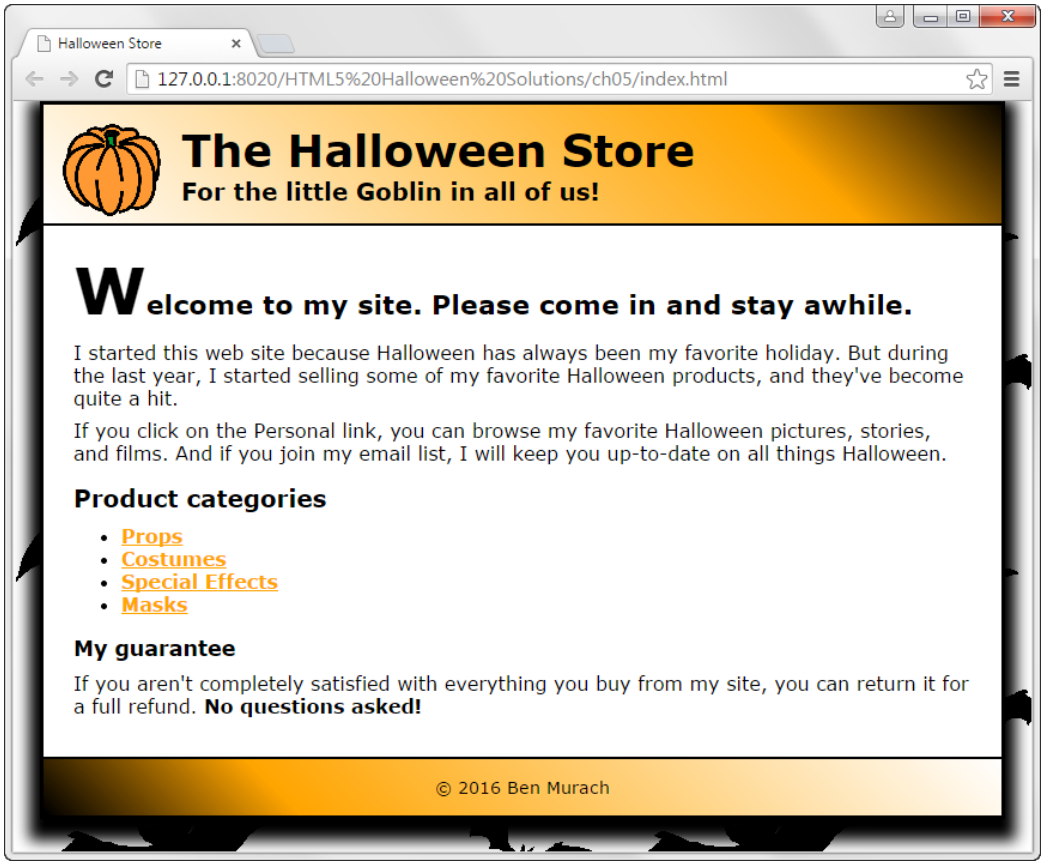
## Exercise 5-2 Add rounded corners and box shadows to the Speakers heading

Use CSS to add a double border with rounded corners and box shadows to the Speakers heading so it looks like this:


This should work in both Chrome and IE.

## Halloween 5 Enhance the home page formatting

In this exercise, you'll use the CSS box model to add spacing, borders, and backgrounds to the Halloween Store home page that you formatted for chapter 4. When you're through, the page should look similar to this:




### Specifications

- Format the body of the page so it's 800 pixels wide, has a white background, has a black border with a shadow, and is centered in the browser window.
- Add a background image behind the body of the page that uses the bats.gif file in the images folder. This image should repeat both horizontally and vertically.
- Add a black border below the header and above the footer.
- Remove the shadow, italics, and color from the first heading in the header.
- Add a 45-degree gradient to the header and footer. The gradient in the header should range from white at the left to orange ¾ of the way to the right to black at the right. The gradient in the footer should be the reverse. Be sure to provide for all modern browsers.
- Adjust the margins and padding for the elements on the page so it looks as shown above.

## Short 5-1 Apply CSS to an HTML page

In this exercise, you'll apply the CSS skills that you learned in chapters 4 and 5 to an existing HTML document. The resulting page should look something like the screen shot that follows. Estimated time: 30 to 45 minutes.


1. Open the HTML and CSS files that follow, and note that the CSS file includes one rule set:

   **`short_exercises\town_hall\speakers\c05x_sorkin.html`**

   **`short_exercises\town_hall\styles\c05x_sorkin.css`**

2. Add a link element to the head section of the HTML file for the normalize.css style sheet.

3. If you want to use a reset selector, add that to the CSS file. But feel free to code the CSS in the way that you prefer.

4. Code a rule set for the html element that sets the background color to yellow.
5. Enhance the rule set for the body so the width is 650 pixels, the body is centered in the browser window, and the body has a double blue border around it like the one above. If you need to make any other changes to the body, do that too.
6. Code a rule set for the main element that puts padding around its contents. Then, code a rule set for the footer that puts a blue border above it. Note that this border doesn't touch the border for the body.
7. Code rule sets for the h1, h2, and h3 elements. The h1 font should be 150% of the default specified in the body, the h2 font should be 125% of the default font, and the h3 font should be 115% of the default font. The h1 font should also be blue. Then, apply appropriate margins or padding to the h1, h2, and h3 elements so the spacing before and after the headings is similar to what's shown above.
8. Code the rule sets for the <p>, blockquote, ul, and li elements so the spacing before and after the elements is similar to what's shown above.
9. Code a rule set for the cite element that changes its color to blue and removes the italics from the text.
10. Code a rule set for the paragraphs that contain cite elements. This rule set should right align the paragraphs and increase the spacing below to .75em. One way to do this is to add a class attribute to these paragraphs and use that class as the selector for the rule set.
11. Float the image to the left and apply appropriate margins or padding so the text flows to its right as shown above.
12. Apply rules to the footer or the paragraphs within the footer so the font size is 90% of the default, the font weight is bold, the paragraphs are centered, and the spacing above and below is similar to what's shown above.

## Chapter 6

# How to use CSS for page layout

### How to float elements in 2- and 3-column layouts

How to float and clear elements
How to use floating in a 2-column, fixed-width layout
How to use floating in a 2-column, fluid layout
How to use floating in a 3-column, fixed-width layout
Two web pages that use a 2-column, fixed-width layout
The home page
The HTML for the home page
The CSS for the home page
The speaker page
The HTML for the speaker page
The CSS for the speaker page

### How to use CSS3 to create text columns

The CSS3 properties for creating text columns
A 2-column web page with a 2-column article

### How to position elements

Four ways to position an element
How to use absolute positioning
How to use fixed positioning
A table of contents that uses positioning

# Objectives

### Applied

1. Given an HTML document, create a CSS style sheet that uses floating to implement a fixed or fluid page layout in two- or three-column format with both header and footer.

### Knowledge

1. Describe the use of floating for page layout.
2. Describe the use of the clear property in a CSS rule set.
3. Distinguish between fixed and fluid page layout.
4. Describe the use of the CSS3 feature for text columns.
5. Describe the use of absolute, relative, and fixed positioning.

# Summary

- When you use the float property to *float* an element, any elements after the floated element will flow into the space left vacant. To make this work, the floated element has to have a width that's either specified or implied.
- To stop an element from flowing into the space left vacant by a floated element, you can use the clear property.
- In a *fixed layout*, the widths of the columns are set. In a *fluid layout*, the width of the page and the width of at least one column change as the user changes the width of the browser window.
- When you use *absolute positioning* for an element, the remaining elements on the page are positioned as if the element weren't there. Because of that, you may need to make room for positioned elements by adjusting other elements.
- When you use *relative positioning* for an element, the remaining elements leave space for the moved element as if it were still there.
- When you use *fixed positioning* for an element, the element doesn't move in the browser window, even when you scroll.

# Terms

float
fixed layout
fluid layout
absolute positioning
relative positioning
fixed positioning

# Exercise 6-1 Enhance the Town Hall home page

In this exercise, you'll enhance the formatting of the Town Hall home page that you formatted in exercise 5-1. When you're through, the page should look like this, but without the Speaker of the Month content:


## Open the HTML and CSS files

1. Use your text editor to open the HTML and CSS files that you created for exercise 5-1 or 5-2:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

   **`c:\html5_css3_2\exercises\town_hall_1\styles\main.css`**

### Enhance the HTML and CSS to provide for the two columns

2. In the main element of the HTML file, enclose the h2 elements and the elements that follow them up to the h1 element in a section element.
3. Enclose the remaining content of the main element in an aside element. Also, shorten the content of the h1 heading to just "Guest speakers" and change the h1 element to an h2 element now that it's in an aside.
4. Still in the HTML file, add the heading and image for the fourth speaker.
5. In the CSS file, enhance the rule set for the body so the width is 800 pixels. Next, set the width of the section to 525 pixels and float it to the right, and set the width of the aside to 215 pixels and float it to the right. Then, use the clear property in the footer to clear the floating. Last, delete the rule set for the h1 heading. Now, test this. The columns should be starting to take shape.
6. To make this look better, delete the left and right padding for the main element, set the left and bottom padding for the aside to 20 pixels, change the right and left padding for the section to 20 pixels, and set the bottom padding for the section to 20 pixels. Now, test again.
7. To make the CSS easier to read, change the selectors for the main elements so they refer to the section or aside element as appropriate and reorganize these style rules. Be sure to include a rule set for the h2 headings in both the section and aside. Then, test again to be sure you have this right.

### Get the headings right

8. Add a rule set for the h3 element in the aside that sets the font size to 105% and the bottom padding to .25 ems.
9. Add a rule set for the img elements in the aside so the bottom padding is set to 1em. Then, test this change.
10. At this point, the page should look good, but it won't include the Speaker of the Month content. Now, make any adjustments, test them in both Chrome and IE, use the developer tools if necessary, and then go on to the next exercise.

## Exercise 6-2 Add the Speaker of the Month

In this exercise, you'll add the Speaker of the Month to the home page. Whenever appropriate, test the changes in Chrome.

### Enhance the HTML page

1. Copy the content for the speaker of the month from the file named c6_content.txt in the text folder into the HTML file right before the heading for "Our Ticket Packages".
2. Enclose the Speaker of the Month heading in h1 tags, and enclose the rest of the content in an article element.
3. Within the article, enclose the first heading in h2 tags; enclose the second heading (the date and speaker's name) in h3 tags with a <br> tag to provide the line break; and enclose the rest of the text in <p> tags.
4. Add an <a> element within the last paragraph that goes to the sampson.html page in the speakers folder when the user clicks on "Read more." Also, add a non-breaking space and <b> tags so the rest of the line looks right.
5. Add an image element between the h2 and h3 elements in the article, and display the image named sampson_dinosaur.jpg from the images folder.

### Enhance the CSS for the home page

6. Add a rule set for the h1 element that sets the font size to 150%, sets the top padding to .5 ems and the bottom padding to .25 ems, and sets the margins to 0. (You need to explicitly set the top and bottom margins of this element to 0 because they're set to .67 ems in the normalize.css style sheet.)
7. Add .5 ems of padding above and below the article. Then, add 2 pixel top and bottom borders to the article with #800000 as the color.
8. Float the image in the article to the right, and set its top, bottom, and left margins so there's adequate space around it. Then, add a 1 pixel, black border to the image so the white in the image doesn't fade into the background.
9. Set the top padding of the h2 element in the article to 0.
10. Set the font size of the h3 element in the article to 105% and set the bottom padding to .25 ems.
11. Make any final adjustments to the margins or padding, use the developer tools if necessary, validate the HTML page, and test in both Chrome and IE.

## Exercise 6-3 Add one speaker page

In this exercise, you'll add the page for one speaker. This page will be like the home page, but the speaker information will be in the second column as shown below. As you develop this page, test it in Chrome whenever appropriate.


### Create the CSS and HTML files for the speaker

1. Copy the index.html file that you've been working with into the speakers folder and name it sampson.html.
2. Copy the main.css file that you've been working with into the styles folder and name it speaker.css.

3. In the head element of the sampson.html file, change the last link element so it refers to the speaker.css file in the styles folder. To do that, you can code a document-relative path like this:

   ```
   ../styles/speaker.css
   ```

   Update the other link elements so they also use relative paths.

### Modify the HTML file

4. Update all the image and link references in the header and aside to document-relative paths. Then, at the bottom of the HTML aside element, add a link back to the home page within an h3 element.
5. Delete all of the content from the section in the sampson.html file, but not the section tags. Then, copy the text for the speaker from the c6.sampson.txt file in the text folder into the section of the sampson.html file.
6. Enclose the first heading ("Fossil Threads in the Web of Life") within an h1 element. Then, code an image element after the first heading that displays the sampson_dinosaur.jpg file that's in the images folder.
7. Enclose the rest of the content for the speaker including the image you just added in an article element. Within the article, enclose the first heading (the date and speaker's name) in h2 tags with a <br> tag to provide the line break, and enclose the rest of the text in <p> tags.
8. Validate the file, and fix any errors.

### Modify the CSS file

9. If you test the page right now, it should look pretty good. Then, you just need to adjust the styles to get the page to look like the one shown above. For instance, you should delete the borders from the rule set for the article, switch the colors of the h1 and h2 headings in the section, adjust the space between those headings, and reduce the size of the h2 heading.
10. Test all the links to make sure they work correctly. Also, make sure the favicon and all images are displayed correctly.
11. Test the page in IE. Then, if necessary make any corrections, and test again in both Firefox and IE. Now, if you've done everything: "Congratulations!"

## Halloween 6 Use a 2-column layout

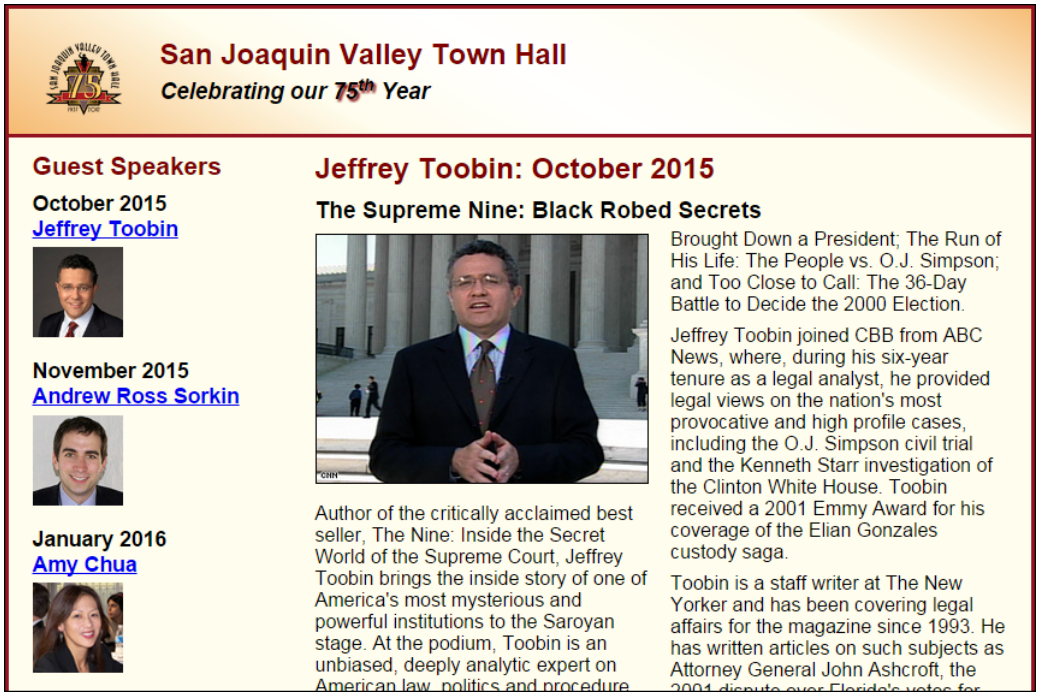
In this exercise, you'll enhance the formatting of the home page so it uses a 2-column layout. You'll also add some additional content to the page. When you're through, the page should look similar to this:


### Specifications

1. Move the unordered list to a sidebar that's floated to the left of the remaining content, which should be coded in a section. The width of the sidebar should be 160 pixels. Be sure to clear the footer so it's not displayed below the list.
2. Adjust the margins and padding for the sidebar and section as necessary. (Note: Because the unordered list is given some left padding by default, you don't need to apply any additional margin or padding to it.)
3. Modify the HTML for the first heading in the section so it's displayed on two lines.
4. Modify the second heading as shown. Then, add the image (cat1.jpg) and the link below the heading. The link should display a page named cat.html in the products folder.

## Short 6-1 Use the CSS3 columns feature

In this exercise, you'll float the speaker image to the left, adjust the formatting as needed, and apply two-column formatting to the article. Estimated time: 5-10 minutes.




1. Open these HTML and CSS files, and run the HTML file:

   **`short_exercises\town_hall\speakers\c06x_toobin.html`**

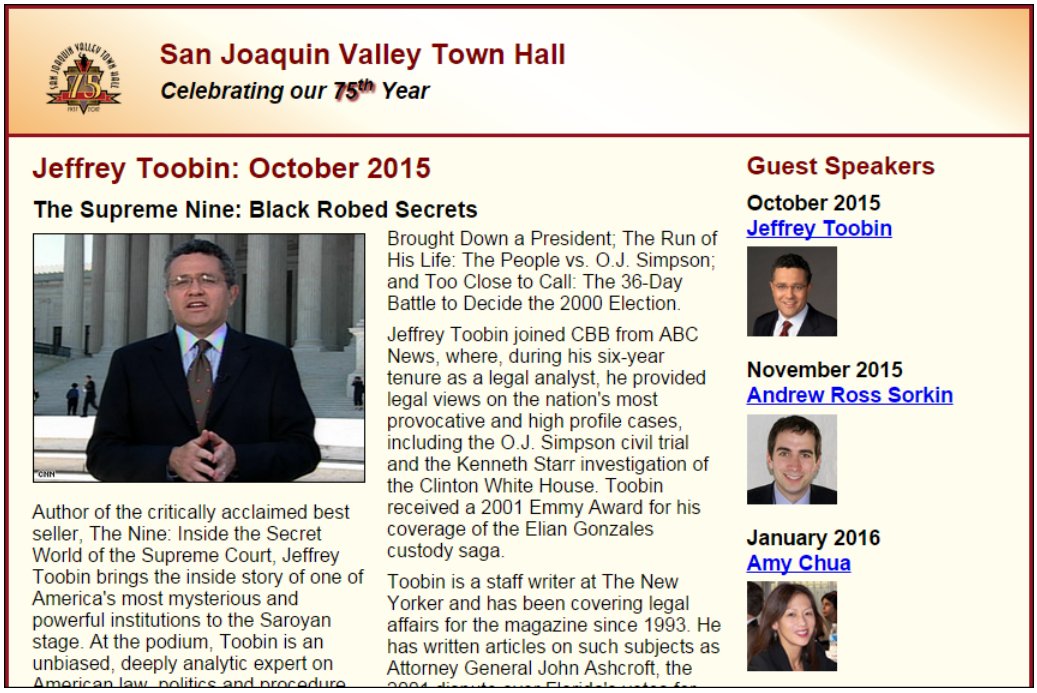
   **`short_exercises\town_hall\styles\c06x_speaker.css`**

2. Float the image to the left instead of the right and adjust the space around the image.

3. Apply two-column formatting to the article using the column-count property as in figure 6-10. If this creates any formatting problems, adjust the HTML or the CSS so the page looks like the one above. One hint: Use CSS to set the width of the image to 275 pixels so it fits in one column.

## Short 6-2 Switch the columns of a page

In this exercise, you'll switch the section and aside of a speaker page so it looks like the one below. That will demonstrate your understanding of floating, margins, and padding. Estimated time: 5-10 minutes.




1. Open the HTML and CSS files for the page:

   **`short_exercises\town_hall\speakers\c06x_toobin.html`**

   **`short_exercises\town_hall\styles\c06x_speaker.css`**

2. Change the style rules for the section and aside so the columns are switched.

3. Adjust the margins and padding for the section and aside so the formatting is similar to the formatting shown above.

## Short 6-3 Add a third column to a page

In this exercise, you'll add a third column to an index page. That will demonstrate your understanding of floating, margins, and padding. Estimated time: 10-15 minutes.


1. Open the HTML and CSS files for the page:

   **`short_exercises\town_hall\c06x_index.html`**

   **`short_exercises\town_hall\styles\c06x_main.css`**

2. Note that another aside has been added to the HTML page. That aside contains the heading and text shown in the third column above.

3. If necessary, rearrange the code in the HTML file so the page can be presented in three columns as shown above.

4. Modify the code in the CSS file so it provides for the three columns with the approximate spacing shown above.

## Chapter 7

# How to work with lists and links

### How to code lists

How to code unordered lists
How to code ordered lists
How to code nested lists
How to code description lists

### How to format lists

How to change the bullets for an unordered list
How to change the numbering system for an ordered list
How to change the alignment of list items

### How to code links

How to link to another page
How to format links
How to use a link to open a new browser window or tab
How to create and link to placeholders
How to link to a media file
How to create email, phone, and Skype links

### How to create navigation menus

How to create a vertical navigation menu
How to create a horizontal navigation menu
How to create a 2-tier navigation menu
How to create a 3-tier navigation menu
The CSS for a 3-tier navigation menu

# Objectives

### Applied

1. Create links and lists in all their variations with HTML and format them with CSS.

### Knowledge

1. Name and describe the three types of HTML lists.
2. Describe the use of <a> elements for linking to another web page, opening another web page in a new browser window, linking to placeholders on the same page, linking to media files, starting an email message, calling a phone number, or starting a Skype session.
3. Describe the use of unordered lists and <a> elements for the creation of navigation lists and navigation menus, including 2- and 3-tier menus.
4. Describe the use of these pseudo-classes for formatting links: :link, :visited, :hover, and :focus.
5. Describe the use of these CSS properties for formatting links: text-decoration and border.

## Summary

- An *unordered list* is displayed as a bulleted list, but you can change the bullets with CSS.
- An *ordered list* is displayed as a numbered list, but you can change the types of numbers that are used with CSS.
- A *description list* consists of terms and descriptions.
- You typically code an <a> element to create a *link* that loads another web page. You can use the tabindex attribute of an <a> element to change the *tab order* of a link, and you can use the accesskey attribute to provide an *access key* for activating the link.
- You can use pseudo-code selectors in CSS to change the default colors for links and to change the styles when the mouse hovers over a link.
- You can use the target attribute to load a linked page in a new browser window or tab. This attribute is no longer deprecated in HTML5.
- A *placeholder* is a location on a page that can be linked to. To create a placeholder, you use the id attribute of an <a> element. To go to the placeholder, you specify that id in the href attribute of another <a> element.
- If an <a> element links to a media file, the browser tries to display or play it by using the right *media player*. To help the browser find the right player, you can use the type attribute to specify a *MIME type*.
- You can also use an <a> element to start an email message, call a phone number, or start a Skype session.
- To create a *navigation menu*, you code a series of <a> elements within the li elements of an unordered list. Then, you can use CSS to remove the bullets and to change the display property so the list is displayed horizontally.
- To create a *multi-tier navigation menu*, you code an unordered list within an li element of another unordered list. Then, you can use CSS to position the submenu relative to the main menu, hide the submenu when the page is first displayed, and display the submenu when the mouse hovers over the list item that contains the submenu.

## Terms

unordered list
ordered list
nested list
description list
link
tab order
access key
placeholder
media player
plugin
MIME type
navigation menu
vertical navigation menu
horizontal navigation menu
navigation bar
multi-tier navigation menu

# Exercise 7-1 Enhance the Town Hall home page

In this exercise, you'll add a two-tier navigation menu to the Town Hall home page that you worked on in chapter 6. You'll also add a link that plays a video, and you'll enhance the formatting of the list on the page. When you're through, the page should look like this:


### Open the HTML and CSS files for this page

1. Use your text editor to open the index.html and main.css files for the Town Hall website:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

   **`c:\html5_css3_2\exercises\town_hall_1\styles\main.css`**

### Add the HTML for the main navigation menu

Use figure 7-16 as a guide as you complete steps 2 through 17.

2. Add a ul element with the id "nav_menu" between the header and main elements.
3. Add the first li element for this list. Then, add an <a> element to this list item with the href attribute set to index.html and the text set to "Home".
4. Copy the li element you just created and paste it four times to create four more list items. Then, change the text for the links in each list item as shown above and change the href attributes accordingly.
5. Set the class for the first link to "current".

### Add the CSS for the main navigation menu

6. Add a rule set for the ul element of the navigation menu that removes the bullets from the list and sets the margins and padding for the list to 0.
7. Add a rule set for the li elements in the unordered list for the navigation menu that floats the elements to the left so they're displayed horizontally.
8. Add a rule set for the <a> elements within the li elements that displays these elements as block elements, sets the widths of the elements to 1/5$^{th}$ the width of the body element, and aligns the text for these elements in the center of the block. This rule set should also set the padding above and below the <a> elements to 1 em, remove the underline from the links, set the background color to #800000, set the text color to white, and set the font weight to bold.
9. Add a rule set for the "current" class that sets the text color for that class to yellow. Then, test these changes in Chrome and adjust the CSS until you get it right.

### Create and format the submenu

10. Add a ul element within the last li element in the navigation menu. Then, add four li elements with <a> elements that contain the text shown above, but don't worry about setting the value for the href attribute of these elements.
11. Modify the rule set for the main navigation menu so it uses relative positioning.
12. Add a rule set for the submenu that uses the display property to keep the submenu from being displayed by default. This rule set should also position the submenu absolutely 100% from the top of the main menu.
13. Add a rule set for the list items in the submenu that keeps them from floating so they're displayed vertically.
14. Add a rule set that causes the submenu to be displayed as a block element when the mouse hovers over the list item that contains the submenu.
15. Test these changes in Chrome. When you point to the About Us menu item, notice that the first item in the submenu is displayed on top of the main menu.
16. Add one more rule set for the ::after pseudo-element of the navigation menu. This rule set should add empty space following the navigation menu, clear the content so it doesn't float, and display the content as a block element.

17. Test the page again. This time, the submenu should be displayed below the main menu.

**Play a video**

18. At the bottom of the Speaker of the Month copy, change "Or meet us there!" to a link that plays the sampson.swf file in the media folder in a new tab or window. This link should say: "Or play video." Then, test this change.

**Change the bullets in the unordered list**

19. Change the bullets in the unordered list to circles. Then, test this change.

**Validate and test in IE**

20. When you've got everything working right, validate the HTML page and test it in IE. If you have any problems, fix them and test again in both Chrome and IE.

## Exercise 7-2 Add the navigation menu to the speaker's page

1. Add the navigation menu of exercise 7-1 to the speaker's page for Scott Sampson. To do that, you need to add the HTML for the menu to the file named sampson.html in the speakers folder, and you need to add the CSS for the menu to the file named speaker.css in the styles folder.
2. Test and adjust to get the formatting right if necessary.
3. In the HTML file, make sure the paths in the links for the navigation menu are correct. Because the sampson.html page is in the speakers folder and the index.html page is in the root folder, you'll need to use a document-relative path to go to this page. Also, none of the links in the navigation menu of the sampson.html page should be current.
4. Test to be sure that you can use the link for the Home menu item to return to the home page from the speaker's page. When this works, delete the link at the bottom of the aside that returns to the home page.

# Halloween 7 Add two navigation menus

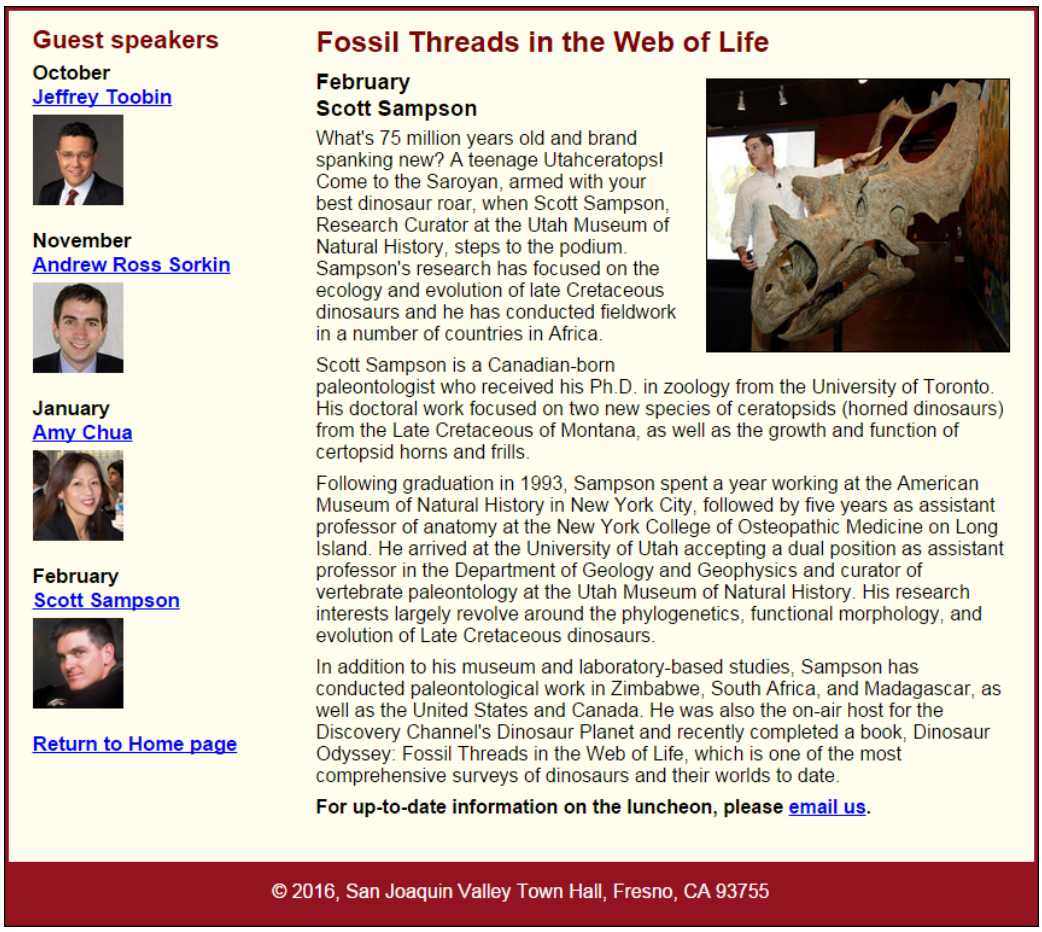
In this exercise, you'll enhance the home page you worked on in exercise 6 so it includes a vertical navigation menu, a two-tier horizontal navigation menu, and an image link. When you're through, the page should look similar to this:


## Specifications

1. Add a 2-tier navigation menu that includes the links shown above. Format the navigation menu so the link for the page that's currently displayed (in this case, home) has orange text, and so the link that the mouse is hovering over (in this case, indoor) has a gray background.
2. Format the links in the sidebar as a vertical navigation menu as shown above. Be sure to make the entire box for each link clickable.
3. Modify the image in the main content so it's a link that displays the same page as the link below it.

## Short 7-1 Start an email

In this exercise, you'll add a paragraph at the bottom of the speaker's page that will start an email. Estimated time: 5-10 minutes.




1. Open this HTML file and run it:

   **`short_exercises\town_hall\speakers\c07x_sampson.html`**

2. In the HTML file, add the boldfaced paragraph at the bottom of the article (use a <b> element to apply the boldfacing). When the user clicks on "email us", an email should be started that is populated with this data:

   To: eleanor@townhall.com

   CC: georgia@townhall.com

   Subject: Scott Sampson luncheon

   Reminder: When you populate two or more fields, you separate them with ampersands (&).

## Short 7-2 Create a vertical navigation menu

In this exercise, you'll create a vertical navigation menu in the sidebar as shown below. Note that each item in the menu includes a right-arrow symbol, which is actually a background image. Estimated time: 20-30 minutes.


1. Open the HTML and CSS files that follow and display the page in a browser:

   **`short_exercises\town_hall\speakers\c07x_sampson.html`**

   **`short_exercises\town_hall\styles\c07x_speaker.css`**

2. Modify the HTML so the sidebar doesn't include the speaker images and so both the month and name for each speaker are in an <a> element. Then, enclose the <a> elements within the li elements of an unordered list as shown in figure 7-14, and enclose the ul element in a nav element that has "nav_list" as its id.

3. Add the CSS for formatting the navigation menu. To speed that up, you may want to copy the code from the example for figure 7-13 and then make the appropriate adjustments so the links are rectangular with a black border.

4. Add rounded corners to the links as well as box shadows with #800000 as the color. The current link should also have #800000 as its color. As you do this, you may want to refer to figure 5-10.

5. Add the symbols to the links. To do that, use the background-image, background-repeat, and background-position properties as shown in figure 5-11. The symbol is in the images file, and it is named right.jpg. In this case, you don't want to repeat the image, and you want to position it in the middle vertically and about 95% from the left.

6. If necessary, adjust the margins, spacing, and font sizes for any of the elements.

## Chapter 8

# How to use Responsive Web Design

### Introduction to Responsive Web Design

The three components of a responsive design
How to test a responsive design

### How to implement a fluid design

Fluid layouts vs. fixed layouts
How to convert fixed widths to fluid widths
How to size fonts
How to scale images
A web page with a fluid design

### How to use CSS3 media queries

How to control the mobile viewport
How to code media queries
Common media queries for a responsive design
How to build responsive menus with the SlickNav plugin

### A web page that uses Responsive Web Design

The design of the web page
The HTML for the web page
The CSS for the web page

# Objectives

### Applied

1. Given an HTML document for the desktop that uses a fixed layout, convert it so it uses fluid widths, scalable images, and fonts that are inherited by child elements.
2. Given an HTML document for the desktop that uses a fluid layout, add media queries that change the appearance of the page depending on the size of the screen where it's displayed.
3. Given an HTML document with a navigation menu that uses media queries, use the SlickNav plugin to replace the navigation menu on smaller screen sizes.
4. Use the browser developer tools to test your responsive designs.

### Knowledge

1. Describe the three components of a Responsive Web Design.
2. Describe three basic ways that you can test a responsive design.
3. Explain how fluid layouts compare to fixed layouts.
4. Explain why you should use relative font sizes with a responsive design.
5. Describe the use of the meta element for setting the viewport on mobile devices.
6. Describe the basic syntax of a media query.
7. Describe two standard approaches for developing the media queries for a responsive design.
8. Explain why you might want to use the SlickNav plugin when developing a responsive design.

## Summary

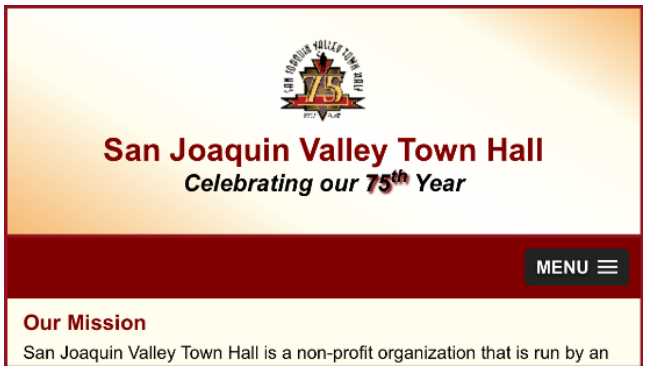
- *Responsive Web Design* refers to a technique that's used to create websites that adapt gracefully to any screen size. A *responsive design* includes fluid layouts, media queries, and scalable images.
- To create a web page with a *fluid layout*, you set the widths of the page and its main structural elements to percents so they increase and decrease depending on the width of the screen.
- To convert the fixed width for an element to a fluid width, you divide the width of the element in pixels by the width of its containing element in pixels and then multiply the result by 100 to get a percent.
- When you develop a responsive design, you should specify font sizes in ems or percents. To convert a font size from pixels to ems, you divide the size by 16 since that's the default size for most browsers. To convert the font size to a percent, you multiply the result of the division by 100.
- To create a *scalable image*, you remove the height and width property from the img element for the image, and you set the max-width property to the percent of its containing block you want it to fill.
- If you want to limit the size of an image to its native size, you can set the width property to a percent and then set the max-width property to the native width in pixels.
- A *media query* is defined by a CSS3 *@media selector* that uses conditional expressions to determine when the styles it contains are applied. You use media queries with RWD to change the appearance of a page for different screen sizes.
- When you use media queries with RWD, you can develop the design for larger devices first and then work your way down to smaller devices. Or, you can develop the design for the smallest device first and then work your way up to larger devices.
- The *viewport* on a mobile device determines the content that's displayed on the screen. When you use media queries, the viewport should be set so the page is displayed at its full size.
- SlickNav is a jQuery plugin that converts a standard navigation menu to a menu that's easier to use on smaller devices.
- An easy way to test a responsive design is to use the developer tools that are provided by most modern browsers. You can also use device emulators and browsers simulators, or you can deploy the website and then test it on various devices or use a web-based tool like ProtoFluid.

## Terms

Responsive Web Design (RWD)
responsive design
fluid design
fixed layout
fluid layout
media query
scalable image
viewport
scale
@media selector
breakpoint

# Exercise 8-1 Convert the Town Hall home page to use a responsive design

In this exercise, you'll convert the Town Hall home page that you worked on in exercise 7-1 so it uses a fluid layout, scalable images, and media queries. When you're through, the page should look like this in mobile phone portrait and landscape orientations:


### Open the HTML and CSS files for this page

1. Use your text editor to open the index.html and main.css files for the Town Hall website:

   **`c:\html5_css3_2\exercises\town_hall_1\index.html`**

   **`c:\html5_css3_2\exercises\town_hall_1\styles\main.css`**

### Add a meta element for the viewport

2. Add a meta element like the one shown in figure 8-8 to the HTML for the page.
3. Display the page in Chrome. Then, size your browser window so it's just wide enough to see all of the page.

### Convert the fixed widths to fluid widths

4. Display the CSS file for this page and note that the width of the body is set to 800 pixels. Then, change this width to 99% so there's room for the 3-pixel left and right borders, and change the maximum width to 960 pixels.
5. Change the widths of the section and aside elements and the <a> elements within the unordered list for the main navigation menu to percents by using the formula in figure 8-4. Do the same for the left and right padding for the image in the header, the left and right padding for the section, and the left padding for the aside.
6. Test these changes in Chrome. When you do, you'll notice that changing the widths of the <a> elements doesn't work.

7. Move the style rule that specifies the width of the <a> elements to the rule set for the li elements that contain the <a> elements, and then test the page again. This time, the navigation menu should look the way it did to start.
8. Notice that, unlike when you first displayed this page, it now extends beyond the right side of the browser window. That's because you set the maximum width of the body to 960 pixels, which is 160 pixels wider than the original page. Increase the width of the browser window to see that the width of the page will increase only until it reaches 960 pixels.
9. Now, decrease the width of the browser window until the sidebar no longer fits next to the section to see that the sidebar is now displayed below the section. Although this is generally what you want to happen, the formatting can be greatly improved.
10. Continue decreasing the width of the browser window to see what happens to the article. To improve this, you'll make the image scalable.

### Make the image in the article scalable

11. Display the HTML for the page and notice that the img element for the image in the article doesn't contain width or height attributes. Because of that, the image is displayed at its native width of 250 pixels, and the text that flows into the space to its left takes up the rest of the width of the article.
12. Set the max-width property of this image to 40%. In addition, set the min-width property to 150 pixels so the image doesn't get to be too small. Then, test the page.

### Add a media query for a tablet in portrait view

13. Code a media query for the screen type that checks that the viewport width is 959 pixels or less. Within this media query, reduce the font size for the h1 element in the section to 135%, and reduce the font size for the h2 elements in the section and aside to 120%.
14. Test this media query. To do that, make the browser window wide enough to see the entire page. Then, reduce the width so it's less than the width of the page. When you do, the size of the h1 and h2 headings in the section and aside should change.

### Add a media query for a phone in landscape orientation

15. Code a media query for the screen type that checks that the viewport width is 767 pixels or less. Within this media query, change the image in the header so it doesn't float and center the contents of the header.
16. Test this media query by reducing the width of the browser window until the styles you just coded are applied. Although this looks pretty good, the line length in the section is starting to get too short.
17. Change the section and aside so they don't float. Then, set the right padding for the aside so it's the same as the left padding, and set the widths of the section and aside by subtracting the left and right padding from 100%.
18. Test the page and notice that the image in the article is too big. Fix that by changing the maximum width of the image to 30% and then test again.

19. To improve the formatting for the aside, display the speakers in two columns. To do that, you'll need to add a div around everything in the aside except the h2 element. Then, you can set the -moz-column-count, -webkit-column-count, and column-count properties to 2. Test this change.

### Add a media query for a phone in portrait orientation

20. Code a media query for the screen type that checks that the viewport width is 479 pixels or less.
21. Change the base font size to 90%, then test this page. When you reduce the width of the page to less than 480 pixels, all of the font sizes should be reduced. This illustrates the advantage of using relative sizes for fonts.

### Add a mobile menu using the SlickNav plugin

22. Use figure 8-11 as a guide to add a link element for the slicknav.css file in the styles folder and a script element for the jquery.slicknav.min.js file in the js folder to the head element of the page. In addition, add a script element for the jQuery core library before the script element for the SlickNav plugin.
23. Add a nav element before the nav element for the navigation menu, and give it an id of "mobile_menu".
24. Add another script element like the one in figure 8-11 that includes the jQuery for calling the slicknav method.
25. Add a rule set outside the media queries that hides the mobile menu. Then, add three rule sets in the media query for a phone in landscape orientation. The first one should hide the standard navigation menu, the second one should display the mobile menu, and the third one should set the background color of the mobile menu to #800000 using the slicknav_menu class.
26. Test this code, and notice that the mobile menu still has its default background color of dark gray. To change that, add an !important rule to the style rule for the background color and test again.

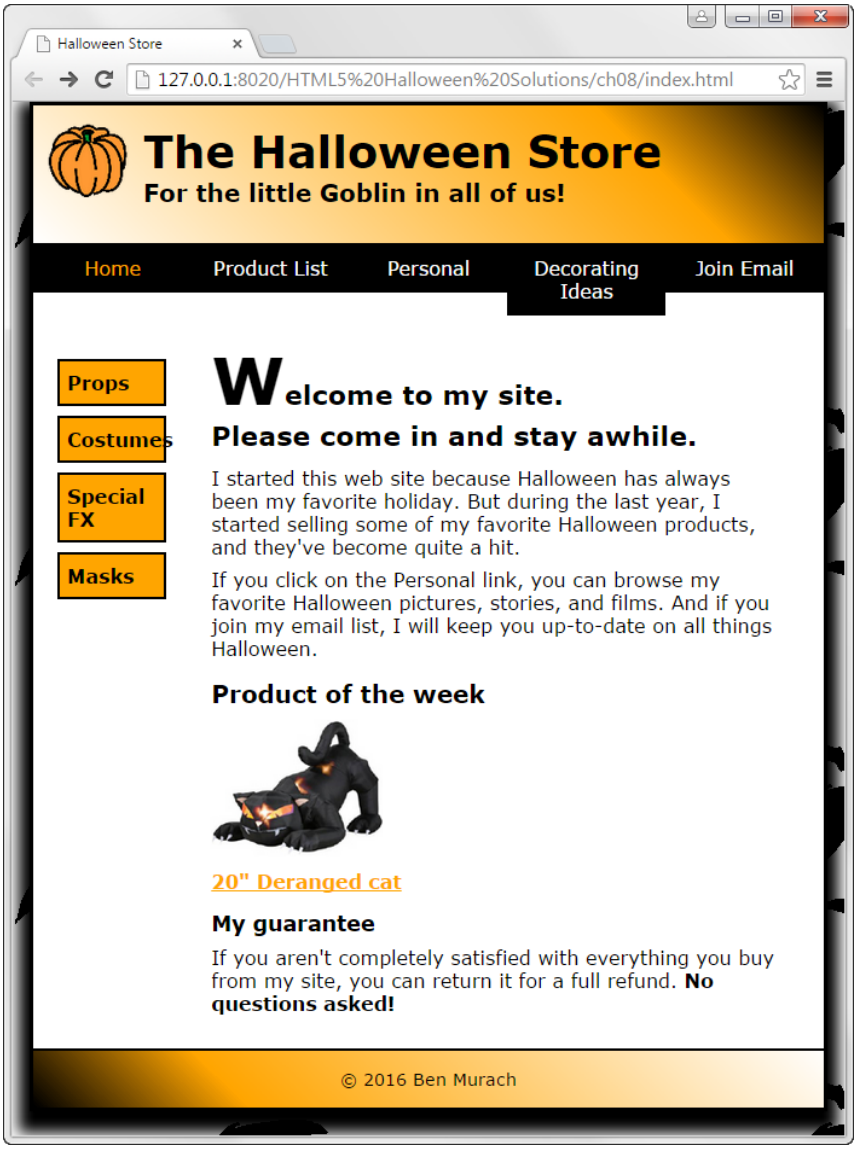
### Use the developer tools to test the page

27. With the page still displayed in Chrome, press F12 to display the developer tools in a pane at the bottom of the browser window.
28. Click the Toggle Device Mode icon to display the page in a grid in the top portion of the window that shows the device size. If a message is displayed indicating that you may need to reload the page, go ahead and do that.
29. Select Apple iPhone 6 from the first drop-down list at the top of the window, reloading the page if necessary. The page will be displayed in portrait orientation.
30. Click the Swap Dimensions icon to the right of the dimensions for the iPhone 6 to display the page in landscape orientation.
31. Continue testing to see how the page will be displayed in different devices. When you're done, click the Close button in the upper right corner of the developer tools pane and then close the browser.

# Halloween 8 Use Responsive Web Design

In this exercise, you'll enhance the home page you worked on in exercise 7 so it uses Responsive Web Design.

### Specifications for a fluid design




- Make a copy of the main.css style sheet and name it main_rwd.css. Then, modify the HTML for the page so it uses this style sheet.
- Modify the page so it uses a fluid layout. The maximum width for the page should be 960 pixels, and the page should occupy 95% of the browser window. Be sure to make the left and right margins and padding as well as the structural elements fluid. (Note: To get the boxes in the vertical navigation menu to size properly, you'll need to specify the width on the ul element instead of the li elements.)

- If you used pixels for any top or bottom margins or padding, convert them to ems. In addition, if you specified any of the font sizes using pixels, convert them to percentages or ems.
- Make both of the images on the page scalable, and make sure that both images are limited in size to the width that's specified in the HTML. In addition, make sure that the image in the header can't be reduced to a width less than 40 pixels.

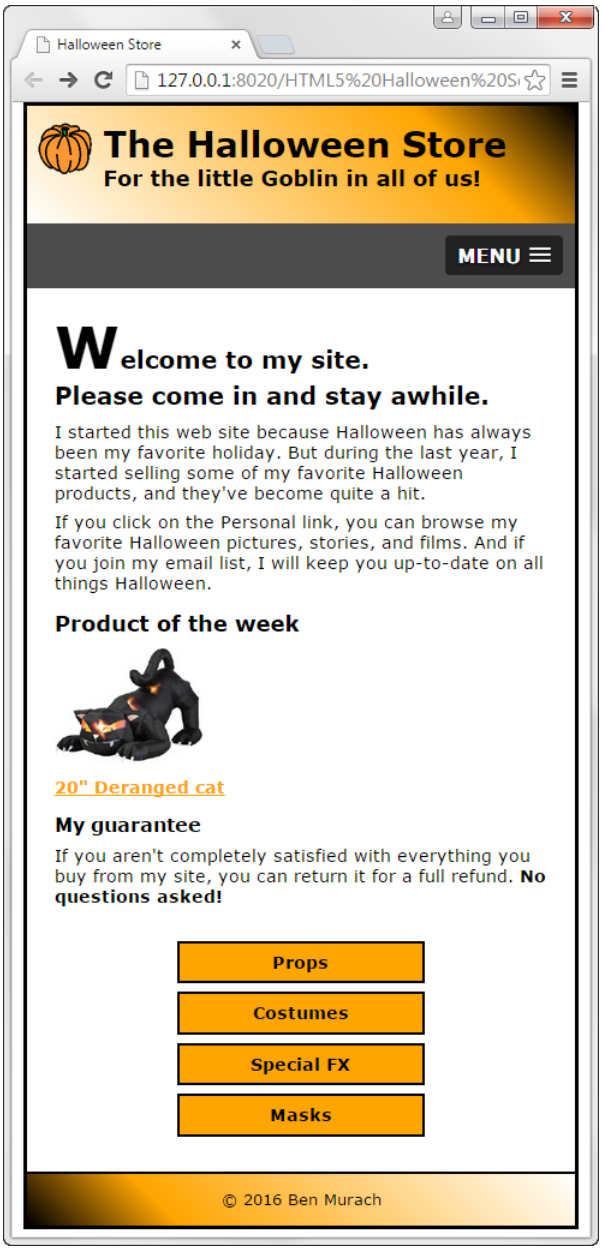
### Specifications for a media query for a tablet in portrait orientation


- Add a meta element to the page to control the viewport for media queries.
- Add a media query that will be applied to screens with a maximum width of 800 pixels. Then, add styles to remove the background image from the page, remove the shadow from the body, and reduce the size of all the fonts on the page to 90%.

### Note

- The page shown above was captured on a desktop so you can see the layout of the entire page.

**Specifications for a media query for a mobile phone in landscape orientation**

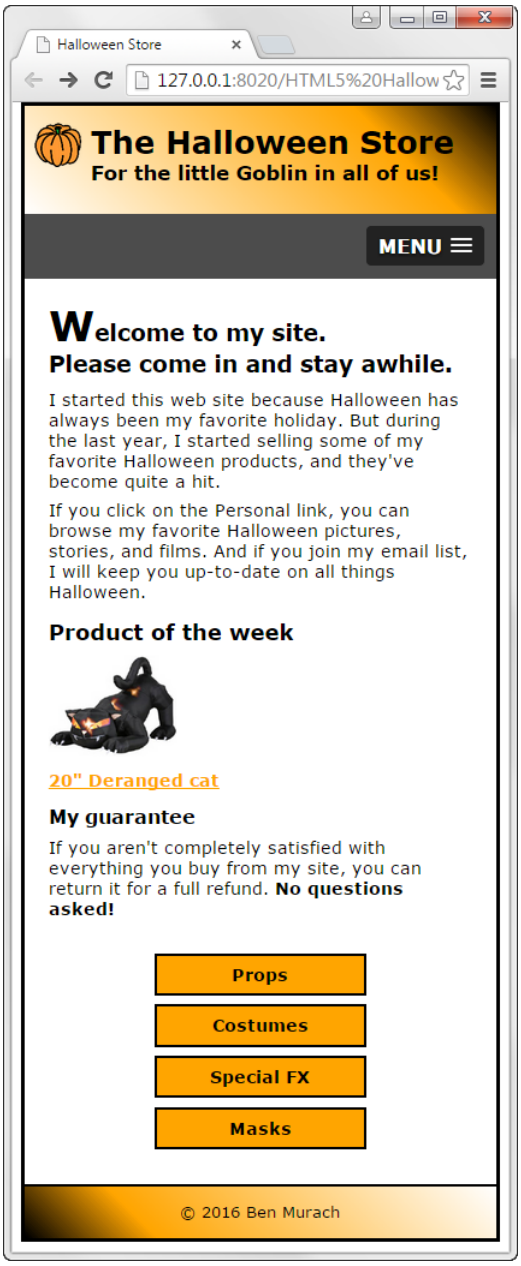



- Add a media query that will be applied to screens with a maximum width of 767 pixels. Then, add the HTML and CSS required to hide the navigation menu and display a SlivkNav menu in its place. You'll find the slicknav.css file in the styles folder, you'll find the slicknav.min.js file in the js folder, and you can get the jQuery library from the jQuery Content Delivery Network. Format the page so it appears in one column.

**Note**

- The page shown above was captured on a desktop so you can see the layout of the entire page.

**Specifications for a media query for a mobile phone in portrait orientation**




- Add a media query that will be applied to screens with a maximum width of 479 pixels. Then, reduce the font sizes of the headings in the header and the first heading in the section.

**Note**

- The page shown above was captured on a desktop so you can see the layout of the entire page.

## Short 8-1 Create a fluid design

In this exercise, you'll convert a page so it uses a fluid layout and scalable images. When you're done, the page will look as shown below when the width of the browser window is reduced. Estimated time: 20 to 30 minutes.


1. Open the HTML and CSS files that follow and display the page in a browser:

   **`short_exercises\town_hall\speakers\c08x_sampson.html`**

   **`short_exercises\town_hall\styles\c08x_speaker.css`**

2. Reduce the width of the browser window so it's less than the width of the page. That way, you'll be able to see how the page changes as you convert it to a fluid design.

3. Modify the CSS for the body so it takes up 98% of the browser window and so its maximum width is the current width.

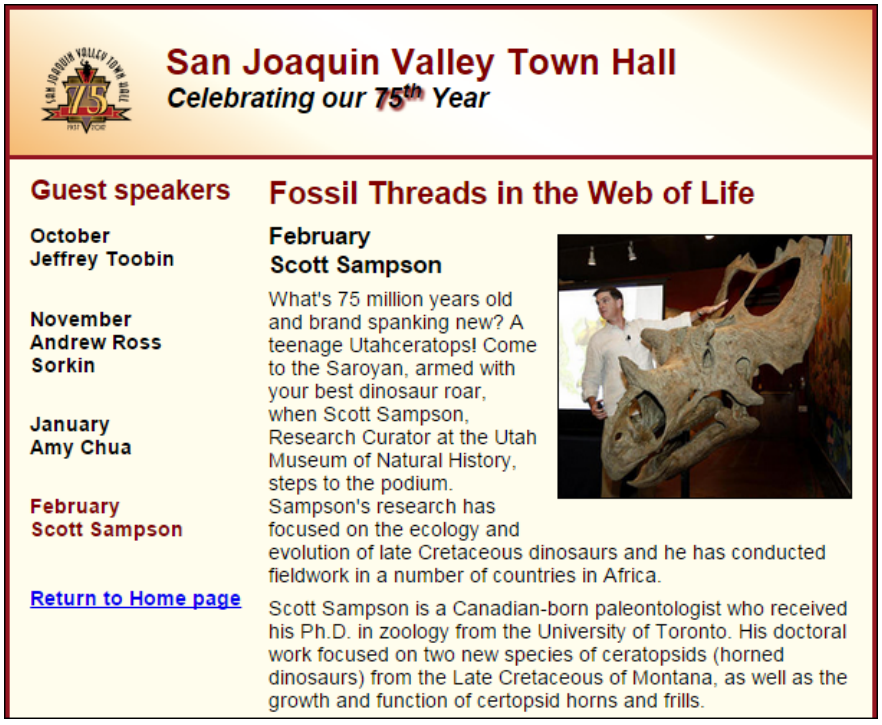
4. Convert the widths of the section, the aside, and the li elements within the navigation menu to percentages.

5. Convert any left or right margins or padding to percentages. (Hint: If any measurements are specified in ems, you can convert them to pixels by multiplying by 16 since that's the base font size.)

6. Make the image in the article scalable so it's always 50 percent of the width of the article.

## Short 8-2 Add a media query

In this exercise, you'll add a media query to a page so it provides for tablets in portrait orientation. When you're done, the page will look as shown below when the width of the browser window is reduced. Estimated time: 10 to 15 minutes.




1. Open the HTML and CSS files that follow and display the page in a browser:

   **`short_exercises\town_hall\speakers\c08x_sampson.html`**

   **`short_exercises\town_hall\styles\c08x_speaker.css`**

2. Add a meta element to the HTML that sets the width property for the viewport to the device width and the initial zoom factor to 1.
3. Code a media query for the screen media type that has a maximum width of 850 pixels.
4. Within the media query, change the base font size for the page so it's 90% of the default font size.
5. Remove the border, box shadow, and background image from the list items in the sidebar.
6. Remove the left padding from the links in the sidebar.

## Chapter 9

# How to work with images

### Basic skills for working with images

Types of images for the Web
How to include an image on a page
How to resize an image
How to align an image vertically
How to float an image

### Advanced skills for working with images

How to use the HTML5 figure and figcaption elements
How to work with thumbnails
How to do image rollovers
How to create image maps

### Related skills for working with images

When to use an image editor
How to get images and icons
How to create favicons

## Objectives

### Applied

1. Use HTML to include images on a web page and CSS to align and float images.
2. Use the HTML5 figure and figcaption elements to treat an image as a figure.
3. Use HTML to create an image map that can be used to link to more than one web page.

### Knowledge

1. List and describe the three types of images that are used with websites.
2. Describe the proper use of the img element and its related CSS.
3. Describe the use of the figure and figcaption elements with images.
4. Describe the use of thumbnails, image rollovers, and image maps.
5. Describe the use of image editors and tools for creating favicons.

## Summary

- The three common formats for images are *JPEG* (for photographs and scanned images), *GIF* (for small illustrations, logos, and animated images), and *PNG* (typically used as a replacement for still GIF images).
- You should use the height and width attributes of an <img> tag only to specify the size of the image, not to resize it. Then, the browser can reserve the right amount of space for the image and continue rendering the page, even if the image is still being loaded.
- You can use CSS to vertically align an image within the block element that contains it. You can also use CSS to float an image.
- The HTML5 figure element can be used to treat an image as a figure that's referred to outside of the figure. The HTML5 figcaption element can be used within a figure element to provide a caption for the figure.
- A *thumbnail* is a small version of an image that is often used as a link to a page that displays a larger version of the image.
- An *image rollover* occurs when the mouse hovers over an image and the image is replaced by another image.
- An *image map* defines the clickable *hotspots* for an image. To define these hotspots, you code map and area elements in the HTML.
- To resize an image so it's the right size for a web page, you can use an image editor like Photoshop CC or Photoshop Elements.
- To reduce the loading time for an image, you can use an image editor to change the image type or quality. To improve the user experience as images load, you can create *progressive JPEGs, interlaced GIFs,* and *interlaced PNGs*.
- A GIF file with two or more *frames* is an *animated image*. Then, you can use an image editor to control how that animation works.
- GIF and PNG files support *transparency*. Then, the background color that's behind the image shows through the transparent parts of the image.
- If you use an image editor to specify a *matte* for an image and then set the matte color to the same one that's used for the background color, the image will blend in better with the background.
- The Internet has many sites that offer images, stock photos, and icons that you may want to use for your site.
- A *favicon* is a small image that appears to the left of the title in the browser's tab for the page. It is typically 16 pixels wide and tall and has ico as its extension.

## Terms

JPEG file
GIF file
animated image
frame
PNG file
thumbnail
image rollover
image map
hotspot
progressive JPEG
interlaced GIF
interlaced PNG
transparency
matte
favicon

## Exercise 9-1 Use a figure on the speaker's page

In this exercise, you'll be working with a second version of the website that you developed for the chapters in section 1 of this book through chapter 7. In this exercise, you'll enhance the speaker page by adding figure and figcaption elements so the page looks like this:


1. Use your text editor to open the HTML and CSS files for this speaker's page, which will be in the exercises/town_hall_2 folder.
2. In the HTML file, enclose the img element at the top of the article in a figure element. Then, add a figcaption element below the img element with the text shown above.
3. In the CSS file, add the rule sets for formatting the figure and figcaption.
4. Test this enhancement in both Chrome and IE.

# Halloween 9 Create a product page

In this exercise, you'll create a product page that uses a variety of features for working with images. When you're through, the page should look similar to this:


### Specifications

1. To create the product page, you can copy the index.html file you worked on in exercise 7 to the products folder and rename it cat.html. Then, you can delete the content from the section, modify the URLs on the page as necessary, and add the content shown above.
2. Create a new style sheet named product.css for the product page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the cat.html file so it points to the correct style sheet.
3. Modify the horizontal navigation menu so it indicates that no page is current.
4. In the section, float the image to the left of the text. In addition, set the left margin for the text so if the product description is longer, the text won't flow below the image.
5. Assign an id to the left sidebar, and modify the CSS so it refers to the sidebar by that id.

6. Add another sidebar with a width of 180 pixels that's floated to the right of the main section, and assign an id to that sidebar. Set the background color of the right sidebar to #F5DEB3.
7. Add a heading to the right sidebar, formatted as shown above. Then, add an unordered list with three list items. Each list item should include a figure with a link that includes an image and a figure caption that includes a text link. The images are named flying_bats.jpg, rat1.jpg, and strobe1.jpg. Be sure to size the images appropriately.
8. Format the links in the right sidebar so they're displayed in black whether or not they've been visited. If a link has the focus or the mouse is hovering over it, though, it should be displayed in green.
9. Adjust the margins and padding as necessary so the page looks as shown above.
10. Add a circular image map for the pumpkin image in the header. When this image map is clicked, the home page should be displayed.
11. Add the link element for the favicon to the head element of the HTML document. You'll find the favicon in the images folder.

## Short 9-1 Do an image rollover with CSS

In this exercise, you'll do an image rollover by using background images. When you're done, the page will look as shown below before and after the rollover. Estimated time: 5-10 minutes.


1. Open these HTML and CSS files, and run the HTML file:

   **`short_exercises\town_hall\speakers\c09x_sampson.html`**

   **`short_exercises\town_hall\styles\c09x_rollovers.css`**

   Note that it displays the image for Scott Sampson (sampson_dinosaur.jpg), as shown above.

2. Modify the HTML and CSS as shown in figure 9-7 so the image rolls over to the image for Andrew Ross Sorkin (sorkin_desk260.jpg) when the user hovers the mouse over the first image. To make this work, set the height and width properties for the <p> element to 260px. Because the images aren't quite that size, you'll also need to set the background-repeat property to no-repeat;

## Chapter 10

# How to work with tables

### Basic HTML skills for coding tables

An introduction to tables
How to create a table
How to add a header and footer

### Basic CSS skills for formatting tables

How to use CSS properties to format a table
How to use the CSS3 structural pseudo-classes for formatting tables

### Other skills for working with tables

How to use the HTML5 figure and figcaption elements with tables
How to merge cells in a column or row
How to provide for accessibility
How to nest tables
How to control wrapping

# Objectives

### Applied

1. Use HTML to create tables with merged cells, captions, headers, and footers.
2. Use CSS and the CSS3 structural pseudo-classes to format the tables that you create.
3. Use the HTML5 figure and figcaption elements to treat a table as a figure.
4. Use HTML to provide accessibility for tables.

### Knowledge

1. Describe these components of a table: rows, columns, cells, header cells, data cells, table header, table footer, and table body.
2. Describe the proper use of tables, now that CSS should be used for page layout.
3. Describe the use of nesting and wrapping with tables.
4. Describe the use of the table attributes for accessibility.

## Summary

- A *table* consists of *rows* and *columns* that intersect at *cells*. *Data cells* contain the data of a table. *Header cells* identify the data in a column or row.
- The rows in a table can be grouped into a *header*, a *body*, and a *footer*.
- To define a table in HTML, you use the table, tr, th, and td elements. Then, you can use CSS to apply borders, spacing, fonts, and background colors to these elements.
- If you use the thead, tfoot, and tbody elements to group the rows in a table, it's easier to style the table with CSS.
- You can use the CSS3 *structural pseudo-classes* to format the rows or columns of a table without using id or class selectors.
- The HTML5 figure element can be used to treat a table as a figure. The HTML5 figcaption element can be used within a figure element to provide a caption for the figure.
- To make tables more accessible to visually-impaired users, you can use the HTML attributes that can be read by screen readers.
- You will often want to *merge* two or more cells in a column or row, and you may occasionally want to *nest* one table within another.
- If a table doesn't fit in the browser window, the browser will *wrap* the data so it does fit. If you don't want that, you can use CSS to turn the wrapping off.

## Terms

table
row
column
cell
data cell
header cell
header
footer
body
structural pseudo-classes
merged cells
nested tables
wrapped rows

# Exercise 10-1 Add a table to the luncheons page

In this exercise, you'll enhance the luncheons page by adding a table to it so it looks like the one that follows.


### Enter the table into the luncheons.html file

1. Use your text editor to open the luncheon.html page in the town_hall_2 folder. Then, run the page to see that everything but the table is already in the file.
2. Add the table shown above to the page. To start the table, you may want to copy the HTML for one of the tables in the book applications into the file. Then, you can modify that code and add the data for the new table. To quickly add new rows to the table, you can copy and paste earlier rows.
3. Test the page to make sure the contents are all there, even though the table won't be formatted right.

### Add the CSS for the table to the main.css file

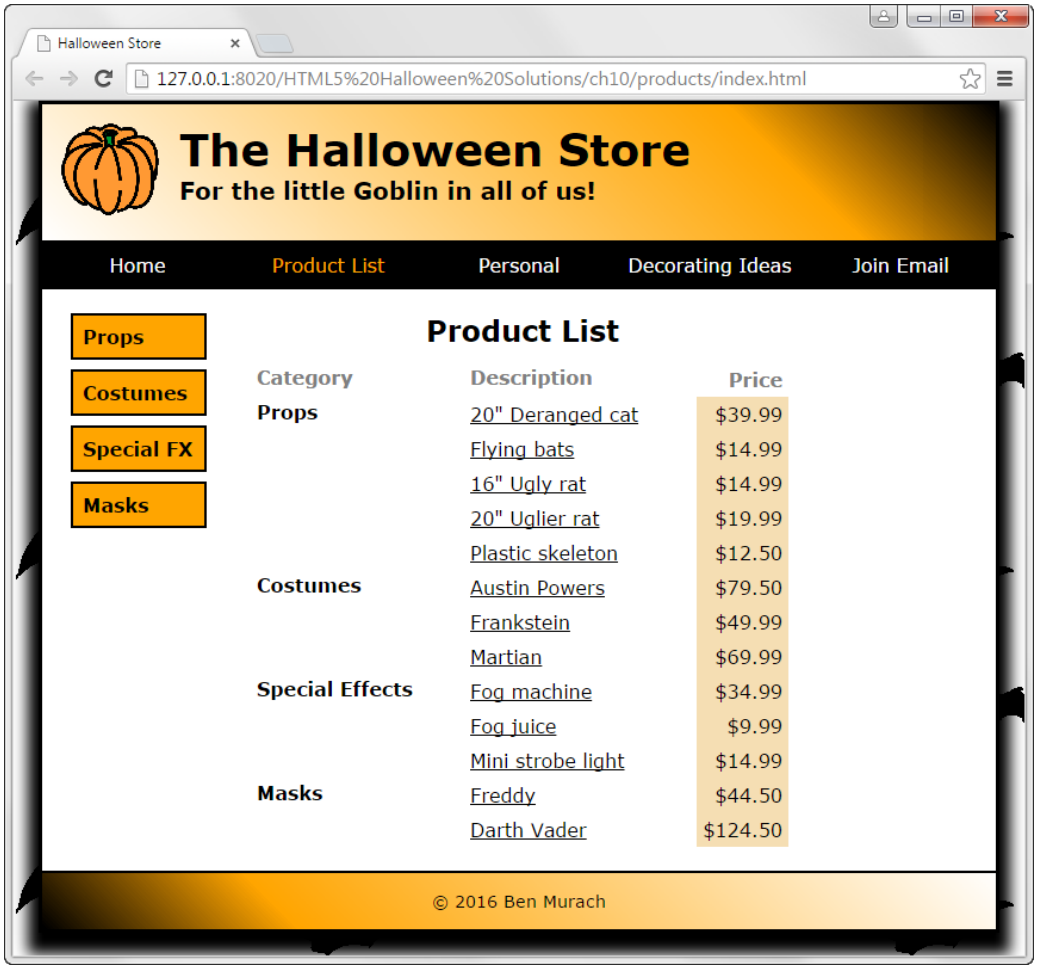
4. Use your text editor to open the main.css file in the styles folder. To start the code for the table, copy the CSS for one of the tables in the book applications into the file. Then, test to see how the table looks.
5. Modify the CSS so the table looks like the one above. Here, all of the borders and the text in the heading above the table should have #800000 as their color. To do some of the alignment and to add borders to some of the rows, you should use classes.
6. Test and adjust until the table looks the way you want it to.

### Treat the table as a figure, provide for accessibility, and experiment

7. Enclose the table within a figure element, and code a figcaption element above the table, but within the figure element. Then, copy the h2 element that says: "The luncheon schedule" into the figcaption element.
8. Test this change. The page should look the same as it did in step 6, except the table will now be centered.
9. Open the developer tools for your browser and review the styles that are applied to the figure element by the normalize style sheet. Then, adjust the CSS so the figure isn't centered.
10. Using figure 10-8 as a guide, add the attributes for user accessibility.
11. Experiment with the CSS3 structural pseudo-classes to see whether you can replace some of your class selectors with pseudo-class selectors.

## Halloween 10 Create a page that uses a table

In this exercise, you'll create a page that uses a table to present a list of products. When you're through, the page should look similar to this:




### Specifications

- To create the product list page, you can copy the index.html file you worked on in exercise 7 to the products folder. Then, you can delete the content from the main section and modify the URLs on the page as necessary.
- Modify the horizontal navigation menu so it indicates that the product list page is the current page.
- Add a table to the section with a caption, a header, and a body as shown above. Be sure to merge the rows in the first column for each category so the category name is displayed only in the first row.

- Code the description for each product as a link to the page for that product. You can use any names you want for these product pages, but the pages should be stored in the products folder. (The exception is the page for the deranged cat, which should be named cat.html as indicated in exercise 9.)
- Create a new style sheet named summary.css for the product list page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the products/index.html file so it points to the correct style sheet.
- Align the caption, headings, and data, and apply any other required formatting as shown above. Use a structural pseudo-class selector to apply a background color of #F5DEB3 to the prices in the third column.

## Short 10-1 Enhance a table

In this exercise, you'll start from a table that looks like this:

| Name | E-mail | Years of Service |
|---|---|---|
| Joel Murach | joelmurach@yahoo.com | 22 |
| Anne Boehm | anne@murach.com | 34 |
| Zak Ruvalcaba | zak@modulemedia.com | 4 |
| Judy Taylor | judy@murach.com | 39 |
| Cyndi Vasquez | cyndi@murach.com | 10 |
| Kelly Slivkoff | kelly@murach.com | 25 |
| Juliette Baylon | juliette@murach.com | 1 |

Then, you'll expand and format the table so it looks like the one that follows. Estimated time: 20-30 minutes.

**Employee Table**

| Department | Name | E-mail | Years of Service |
|---|---|---|---|
| Editorial | Joel Murach | joelmurach@yahoo.com | 22 |
| | Anne Boehm | anne@murach.com | 34 |
| | Zak Ruvalcaba | zak@modulemedia.com | 4 |
| Marketing | Judy Taylor | judy@murach.com | 39 |
| | Cyndi Vasquez | cyndi@murach.com | 10 |
| Customer Service | Kelly Slivkoff | kelly@murach.com | 25 |
| | Juliette Baylon | juliette@murach.com | 1 |
| | | **Total Years of Service** | **105** |

1. Open this HTML file and run it:

   **`short_exercises\town_hall\c10x_employee_table.html`**

   Note that this HTML file uses embedded CSS in the head section to do the formatting.

2. Expand the HTML so the table has the caption and data shown in the second table above.

3. Expand the CSS so it does the formatting shown in the second table above.

4. If you used classes to do the alignment and apply the colors to the cells of the table, try using pseudo-classes to get the same results.

## Chapter 11

# How to work with forms

### How to use forms and controls

How to create a form
How to use buttons
How to use text fields
How to use radio buttons and check boxes
How to use drop-down lists
How to use list boxes
How to use text areas
How to use labels
How to group controls with fieldset and legend elements
How to use a file upload control

### Other skills for working with forms

How to align controls
How to format controls
How to set the tab order and assign access keys

### How to use the HTML5 features for data validation

The HTML5 attributes and CSS3 selectors for data validation
How to use regular expressions for data validation
How to use a datalist to present entry options

### How to use the HTML5 controls

How to use the email, url, and tel controls
How to use the number and range controls
How to use the date and time controls
How to use the search control for a search function
How to use the color control
How to use the output element to display output data
How to use the progress and meter elements to display output data

### A web page that uses HTML5 data validation

The page layout
The HTML
The CSS

# Objectives

### Applied

1. Use HTML to create a form that includes any of the form controls, including the HTML5 controls.
2. Use CSS to format a form and its controls.
3. Use the HTML5 attributes and CSS3 selectors for data validation.
4. Use an HTML form to add a search function to a website.

### Knowledge

1. Describe the use of an HTML form for initiating client-side processing or submitting data to a web server.
2. Describe the get and post methods that can be used to submit a form.
3. Describe the use of any of the form controls.
4. Describe the use of tab order and access keys.
5. Describe the use of these HTML5 attributes: autocomplete, required, and pattern.
6. Describe the use of regular expressions.
7. Describe the use of a datalist.
8. Describe the use of a form that provides a search function for a website.

## Summary

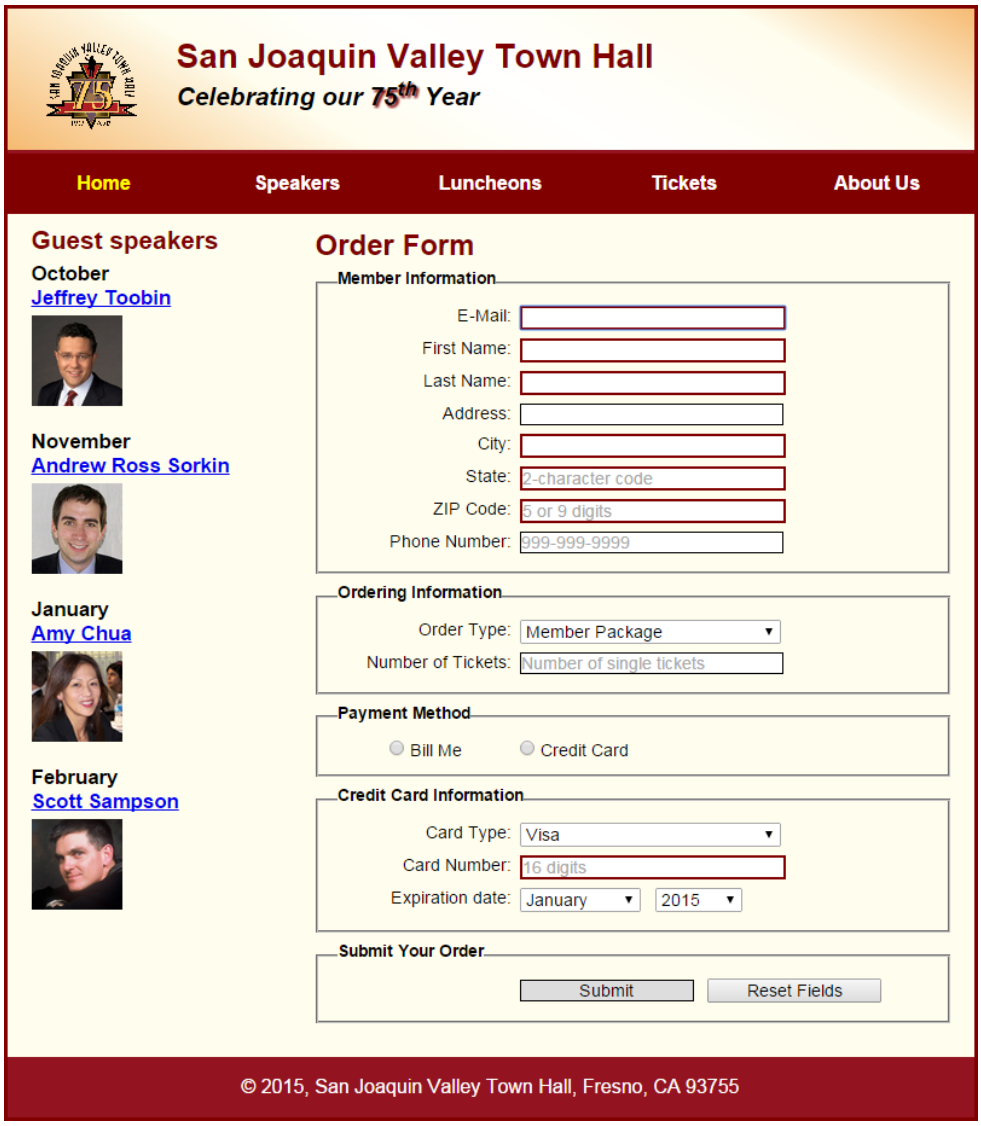
- A *form* contains one or more *controls* like text boxes, radio buttons, or check boxes that can receive data. When a form is submitted to the server for processing, the data in the controls is sent along with the HTTP request.
- When the get method is used to submit a form, the data is sent as part of the URL for the next web page. When the post method is used, the data is hidden.
- A *submit button* submits the form data to the server when the button is clicked. A *reset button* resets all the data in the form when it is clicked. Buttons can also be used to start client-side scripts when they are clicked.
- The controls that are commonly used within a form are *labels*, *text fields*, *radio buttons*, *check boxes*, *drop-down lists*, *list boxes*, and *text areas*.
- A *text field* is used to get data from a user. A *password field* also gets data from the user, but its data is obscured by bullets or asterisks. A *hidden field* contains data that is sent to the server, but the field isn't displayed on the form.
- A *label* is commonly used to identify a related control. To associate a label with a control, you use the for attribute of the label to specify the id value of the control.
- You can use a *file upload control* to upload a file from the user's system. Then, server-side code is used to store the file on the web server.
- You can use CSS to align controls by floating the labels to the left of the controls. You can also use CSS to format the controls.
- The *tab order* of a form is the order in which the controls receive the focus when the Tab key is pressed. By default, this is the sequence in which the controls are coded in the HTML. To change that, you can use tabindex attributes.
- *Access keys* are shortcut keys that the user can press to move the focus to specific controls on a form. To assign an access key to a control, you use its accesskey attribute. To let the user know that an access key is available, you can underline the access key in the label for the control.
- HTML5 introduced some attributes for *data validation*, and CSS3 introduced some pseudo-classes for formatting required, valid, and invalid fields. The HTML5 attributes for data validation include the required attribute and the pattern attribute that provides for *regular expressions*.
- HTML5 also introduced some input controls including the email, url, tel, number, range, date, time, search, and color controls. These are good semantically because they indicate what types of data the controls accept.
- HTML5 also introduced some output controls that can receive the results of client-side or server-side programming. These include the output, progress, and meter controls.

## Terms

form
control
button
submit button
reset button
image button
text field
text box
password field
hidden field
check box
radio button
drop-down list
list box
text area
label
file upload control
tab order
access key
data validation
auto-completion feature
regular expression
pattern
datalist

# Exercise 11-1 Create a form for getting tickets

In this exercise, you'll create a form like the one that follows. To do that, you'll start from the HTML and CSS code that is used for the form in the chapter application.




## Open the HTML and CSS files

1. Use your text editor to open the HTML and CSS files for the Tickets page:

   **`c:\html5_css3_2\exercises\town_hall_2\tickets.html`**

   **`c:\html5_css3_2\exercises\town_hall_2\styles\tickets.css`**

2. Run the HTML file in Chrome to see that this page contains the form and the formatting that is used in the application at the end of the chapter. Your job is to modify the form and its CSS so the page looks like the one above.

### Fix the formatting of the form and test after each change

3. Change the fieldset margins and padding so there is no top margin, the bottom margin is .5em, the top and bottom padding is .5em, and the right and left padding is 1em.
4. Change the legend formatting so the color is black.
5. Change the formatting so the required and the invalid fields have a 2-pixel border with #800000 as the font color, and the valid fields have a 1-pixel black border.

### Fix the HTML one fieldset at a time and test after each change

6. In the HTML, change the heading before the form to "Order Form". Then, combine the first two fieldsets, change the legend to "Member Information", and delete the password fields so the first fieldset looks like the one above.
7. In the Membership Information fieldset, change the legend to "Ordering Information" and modify the HTML for the fields. Here, the Order Type list should have these choices: Member Package, Donor Package, and Single Tickets. Also, the Number of Tickets field should be a text field with the placeholder shown above.
8. Add the Payment Method fieldset with two radio buttons, and use the coding method in figure 11-8 so the user can activate the buttons by clicking on the labels. Then, adjust the CSS so the buttons are side by side as shown above.

   One way to do that is to code an id for the fieldset, and then use an id selector to turn off the float for the labels, to set the width of the buttons to "auto", and to set the left margin for the buttons to 3 ems.
9. Add the Credit Card Information fieldset. Here, the Card Type list should have these choices: Visa, MasterCard, and Discover. The Card Number field should have the placeholder shown. The month list should have values from January through December. And the year list should have values from 2015 through 2019.

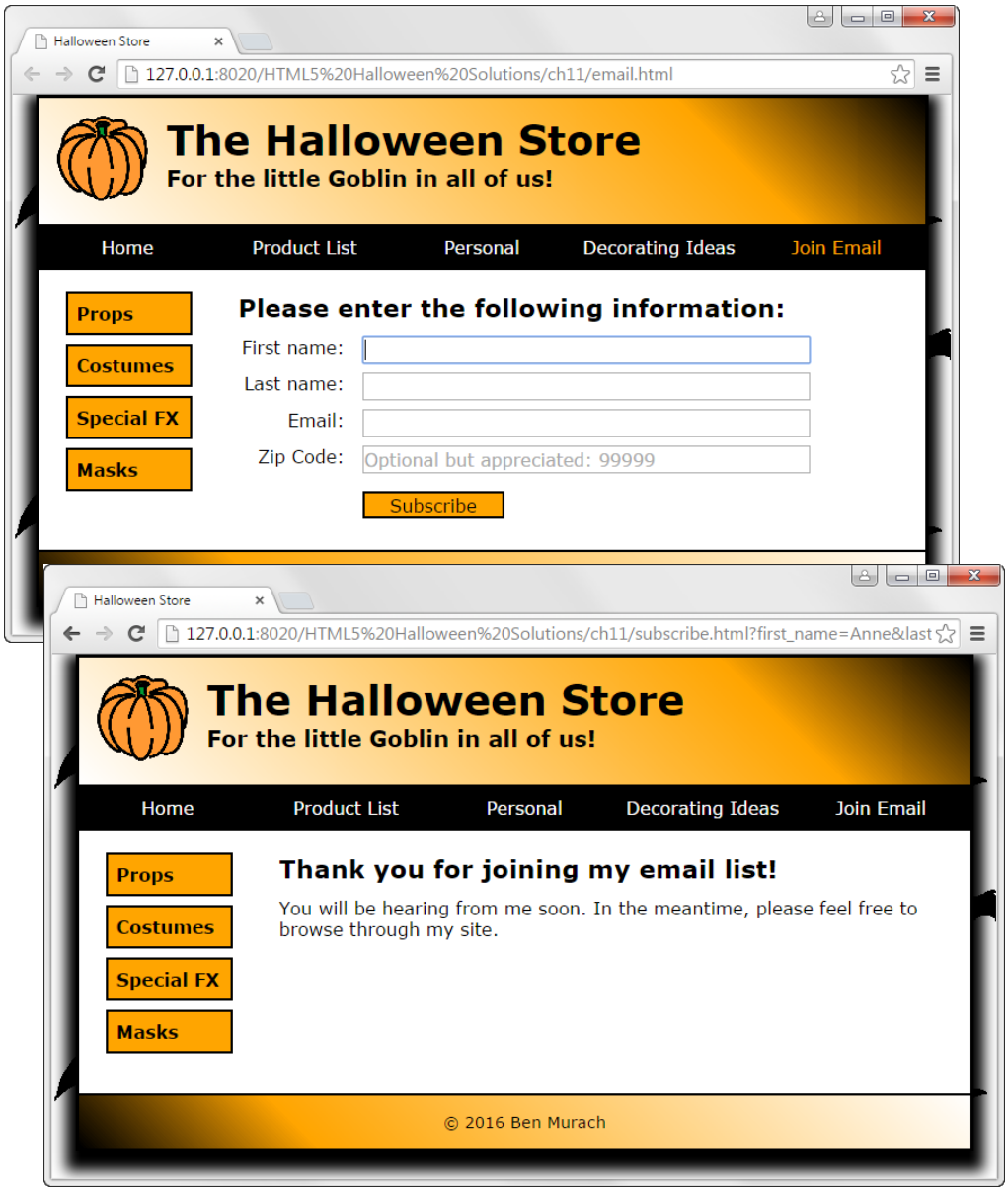
   To format these lists, you can use id selectors and set the width of the month to 7 ems and the width of the year to 5 ems. To check the validity of the card number, you can use this pattern: \d{16}.
10. For the Submit Your Order fieldset, you just need to change the legend.

### Test, validate, and test in other browsers

11. Test the validation that's done by the form by entering combinations of invalid data. Then, enter valid data for all the fields and click the Submit button. This should display a page (register_account.html) that shows the data that has been submitted.
12. Validate the HTML file, and test the form in IE. Then, if you have the time, test the form in Firefox, Opera, and Safari too.

# Halloween 11 Create a page that uses a form

In this exercise, you'll create a page that includes a form for joining an email list. In addition, you'll create a page that's displayed when the form is submitted. When you're through, the pages should look similar to this:




### Specifications for the email form

1. Create a new page named email.html in the root folder for the project. Then, copy the code for the basic page elements, as well as for the header, footer, and sidebar, from the page for exercise 7.

2. Modify the horizontal navigation menu so it indicates that the email page is the current page.
3. Create a new style sheet named email.css for the email page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the email.html file so it points to the correct style sheet.
4. Add the heading and the form element to the section. Code the form element so a form named subscribe.html in the root folder is displayed when the Subscribe button is clicked.
5. Add the label and input elements for the form. The focus should be on the first text field when the form is first displayed.
6. Align the controls as shown above, and format the button so it's orange with a black border.
7. Use the data validation attributes to require entries in the first three text fields. In addition, use a regular expression to validate an entry in the Zip Code field, and display an appropriate message in the tooltip if the entry is invalid.
8. Use a placeholder to display a message in the Zip Code field as shown above.
9. Add any other formatting as necessary so the page looks as shown above.

### Specifications for the subscribe form

10. Create a new page named subscribe.html in the root folder for the project. Then, copy the code for the basic page elements, as well as for the header, footer, and sidebar, from the email page.
11. Create a new style sheet named subscribe.css for the subscribe page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the subscribe.html file so it points to the correct style sheet.
12. Modify the horizontal navigation menu so it indicates that no page is current.
13. Add the heading and paragraph shown above, and format it as necessary.

## Short 11-1 Create a form

In this exercise, you'll complete a form so it looks as shown below. Estimated time: 20-30 minutes.


1. Open this HTML file and run it to see that it already provides the first two headings, the first three fields, the last heading, and the two buttons:

   **`short_exercises\town_hall\c11x_forms.html`**

2. Modify the attributes for the form so it uses the "get" method and submits the form to the file named survey_data.html. Then, test the submission of the completed form. This should display a page that shows the submitted data.

3. Add the autofocus attribute to the first field and the required attribute to all three of the fields. Then, test to make sure the data validation works.

4. Add the "Geographic information" heading and fields. These fields should also be required and they should have the placeholders that are shown above. To validate the state and zip code fields with regular expressions, you can use these patterns:

   **`[A-Za-Z]{2} and \d{5}`**

5. Add the "How did you hear about us" heading and checkbox fields. Use labels to provide for user accessibility as shown in figure 11-8.

6. Enhance the CSS in the HTML file so the form looks like the one above. In particular, you'll need to change the width of the checkboxes so they will align right, and you should remove the box shadow from required and invalid fields so they won't have red shadows when displayed in Firefox.

7. Do a final test to make sure everything works as it should.

## Chapter 12

# How to add audio and video to your website

# Objectives

### Applied

1. Use the HTML5 audio and video elements to add audio and video to a web page.
2. Use object and param elements or embed elements along with the audio and video elements to fall back to Flash in older browsers that don't support the audio and video elements.

### Knowledge

1. Distinguish between these terms: media type, media player, and codec.
2. Explain why more than one audio or video type is required to run an audio or video file in all browsers.
3. In general terms, describe the coding that's required for adding audio or video to a web page.

# Summary

- A browser uses a *media player* to play an audio or video *media type* when media are embedded in a web page.
- Some browsers require *plugins* for the media players that play specific media types.
- Audio and video files come in a variety of media types. MPEG, Ogg, and WebM are common types for video, and AAC and MP3 are common types for audio.
- *Codecs* are software components that are used by browsers to decipher the algorithms in a media type.
- A *MIME type* describes the contents of a file and can help a browser determine what media player to use for the file.
- To embed media types in the web page for older browsers, you can either use the object and param elements or the embed element.
- The HTML5 video and audio elements are supported by modern browsers and will play media without the need for special plugins.
- You can nest object or embed elements within video and audio elements to fall back to the Flash media type. This allows you to target older browsers that don't support the HTML5 video and audio elements.

# Terms

media type
media player
plugin
codec
MIME type

## Exercise 12-1 Add video to a speaker's page

In this exercise, you'll add video to a copy of the speaker's page for Scott Sampson. This will show you how easy it is to add video to a page. When you're through, the page should look like the one that follows.


### Open the page, copy it, and copy in the HTML for running a video

1. Use your text editor to open this page:

   `c:\html5_css3_2\exercises\town_hall_2\speakers\sampson.html`

2. Make a copy of this page in the same folder, and name it sampson_video.html. If you're using Aptana, you can use the File→Save As command to do that.

3. Open the book application for this chapter, and copy the code for running the video to the clipboard. Then, paste in into the sampson_video.html file so it replaces either the figure element (if your file has one) or the img element.

4. Test the page to see what the result is. But don't run it from Aptana because the video won't work right. Instead, run the page from your browser or file server.

### Modify the HTML to get this working right

5. Modify the references to the media files so they point to the right files. To do that, you need to use a document-relative path that goes up one level, and you need to use the right filenames. For instance, the reference in the first source element should be:

   ```
   src="../media/sampson.mp4"
   ```

   To find out what the right filenames are, you can look in the media folder.

6. You also need to modify the reference to the poster file that's in the images folder with a similar document-relative path.

7. Test this page again and it should work. This shows how easy it is to add a video to a page because you'll use the same code each time. Only the references will change.

## Halloween 12 Create a page that uses video

In this exercise, you'll create a personal page that contains a video. When you're through, the page should look similar to this:


### Specifications

- To create the personal page, you can copy the index.html file from exercise 7 to the personal folder. Then, you can delete the content from the section and modify the URLs on the page as necessary.
- Modify the horizontal navigation menu so it indicates that the personal page is the current page.
- Modify the vertical navigation menu so it provides the links shown above. Use any names you want for the pages, but assume that they're stored in the personal folder.
- Create a new style sheet named personal.css for the personal page, and copy the styles you need from the main.css file. Then, modify the link element for the style sheet in the personal/index.html file so it points to the correct style sheet.
- Add a heading and the video to the section. Provide for the MPEG-4 media type, and provide a Flash file as fallback in case a browser doesn't support the video element. You'll find the video files you need in the media folder.
- Add any other formatting as necessary so the page looks as shown above.

Note: Remember that if you use Aptana to run a page that uses an object element within a video element, it may not work correctly. If it doesn't, you can access the page directly from your browser.

## Short 12-1 Embed audio in a page

In this exercise, you'll add audio to the top of the section in a speaker's page, but the audio controls won't show so the page will look the same. However, the audio will play automatically after the page is loaded. Estimated time: 5-10 minutes.


1. Open this HTML file:

   **`short_exercises\town_hall\speakers\c12x_sampson.html`**

2. Copy the code for the audio example in figure 12-9 into the HTML file. Then, using that figure as a guide, modify the code.

3. Note that you'll have to change the relative reference for the media folder. Also, the filename that you should use is welcome.mp3. Remember too that the controls shouldn't show and the audio should play only one time after starting automatically.

## Chapter 13

# How to work with fonts and printing

# Objectives

### Applied

1. Use the CSS3 @font-face selector, Google Web Fonts, or Adobe Typekit fonts to embed fonts in a web page.
2. Use the @media print selector to add the styles for printing a page to the style sheet for displaying the page.
3. Use a second style sheet for the styles that should be used for printing a web page.

### Knowledge

1. In general terms, describe how the CSS3 @font-face selector works.
2. In general terms, describe the use of Google Web Fonts and Adobe Typekit fonts.
3. In general terms, describe the way that you format a web page for printing.
4. Describe three ways that you can provide the styles for printing a web page.
5. Describe the use of these properties for printing a page: display, page-break-before, page-break-after, orphans, and widows.
6. Explain why in (inches) is often used as the unit of measurement for printed pages.

## Summary

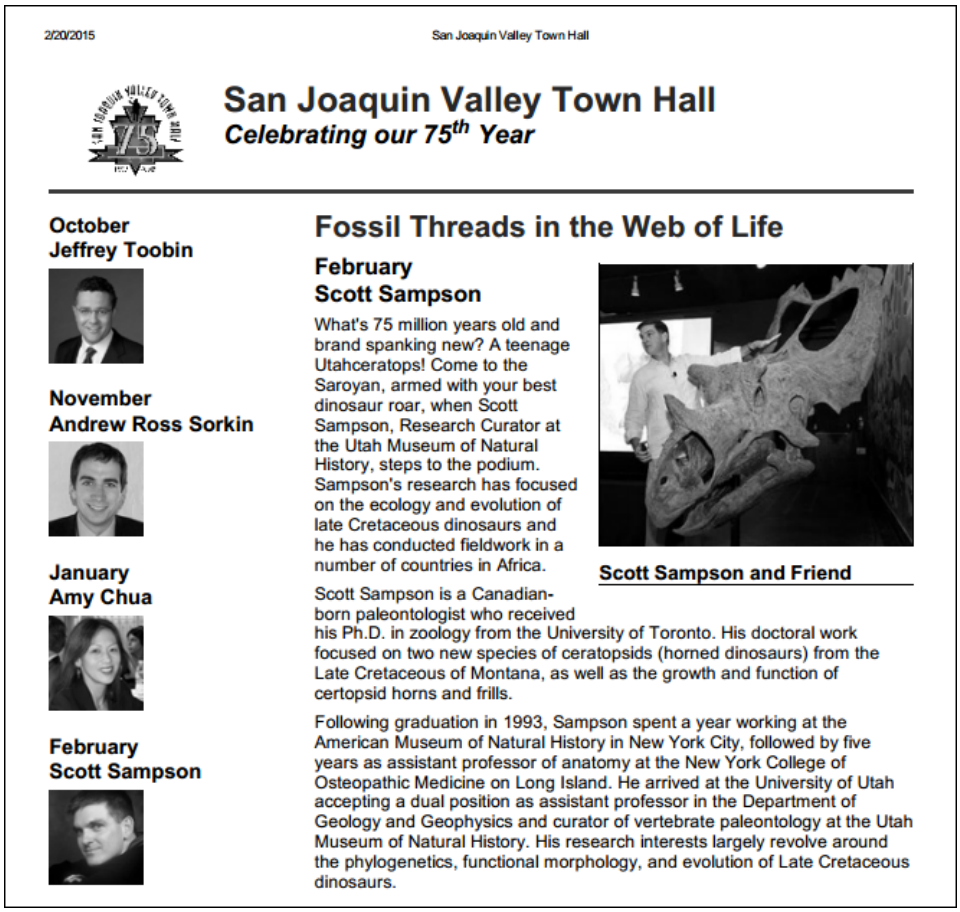
- CSS3 provides a new @font-face selector that you can use to *embed fonts* from your libraries into a web page. You can also embed fonts in your web pages by using Google Web Fonts or Adobe Edge Web Fonts.
- If you want to change the way a web page is formatted when it's printed, you can code a separate style sheet for printing. Then, you can use a link element to include the *print style sheet* after the *screen style sheet*. That way, the print styles will override the screen styles.
- Another way to provide the styles for printing is to code a *media query* for printing at the end of the screen style sheet. Here again, the styles for printing will override the styles for the screen.
- CSS provides several properties for printing, including the display property for omitting an element from a web page, several page-break properties for controlling where *page breaks* occur, and orphans and widows properties for avoiding *orphans* and *widows*.
- When you code the CSS for a page that's going to be printed, you should remove site navigation and unnecessary images, use dark print on a white background, and use a serif font for the text in a size and with a line length that's easy to read.

## Terms

embed fonts
@font-face selector
print style sheet
screen style sheet
media query
page break
widow
orphan

# Exercise 13-1 Format the printing for a page

In this exercise, you'll style the printing for the speaker's page for Scott Sampson so it prints like this:




### Open the HTML and CSS files for this page

1. Use your text editor to open these pages:

   ```
   c:\html5_css3_2\exercises\town_hall_2\speakers\sampson.html
   c:\html5_css3_2\exercises\town_hall_2\styles\speaker.css
   ```

2. Run this page in your browser, and use the Print Preview command to see what it will look like when printed.

### Add the styles for printing the page to the style sheet

3. In the style sheet for this page, add a @media print selector to the end of the file, as shown in figure 13-4. Then, add rules sets that do the following:

   Change the font family for the body to Times New Roman, Times, or the default serif font.

   Change the width of the body to "auto", the background color to white, and the border to none.

Turn off the display for the navigation bar and the first heading in the aside.

Set the width of the section to 5 inches and the width of the aside to 2 inches.

Set the right padding for the section and the left padding for the aside to zero.

Now, test these changes in Chrome to see how things are going.

4. Add the other rule sets that are needed to get the print preview to look like the one above, such as:

   Reduce the font size for the h2 elements in the aside to 100%.

   Reduce the font size for the <p> elements in the article and footer to 87.5%.

   Remove the underlines from the <a> elements in the aside and set their color to black.

   Change the top padding for the <p> element in the footer to .75 ems, and add a 3-pixel solid border with the color #931420.

   Now, test these changes. The page should be looking pretty good.

5. Make any final adjustments, and test the page in Chrome one last time.

6. Test the page in IE or Firefox and notice that the sidebar isn't aligned at the left side of the page. To fix that, remove the floating from the aside and test again.

## Exercise 13-2 Use a web font

In this exercise, you'll enhance the header of a page with a web font:


1. Use your text editor to open the HTML and CSS files for this page:

   ```
   c:\html5_css3_2\exercises\town_hall_2\speakers\sampson.html
   c:\html5_css3_2\exercises\town_hall_2\styles\speaker.css
   ```

2. Test this page in your browser to see what it looks like.

3. Using figure 13-1 as a guide, choose a web font on your system and copy it into the styles folder.

4. Use CSS3 to apply the font to the h2 element in the header. Be sure to include one or more additional font families in case the browser doesn't support the @font-face selector.

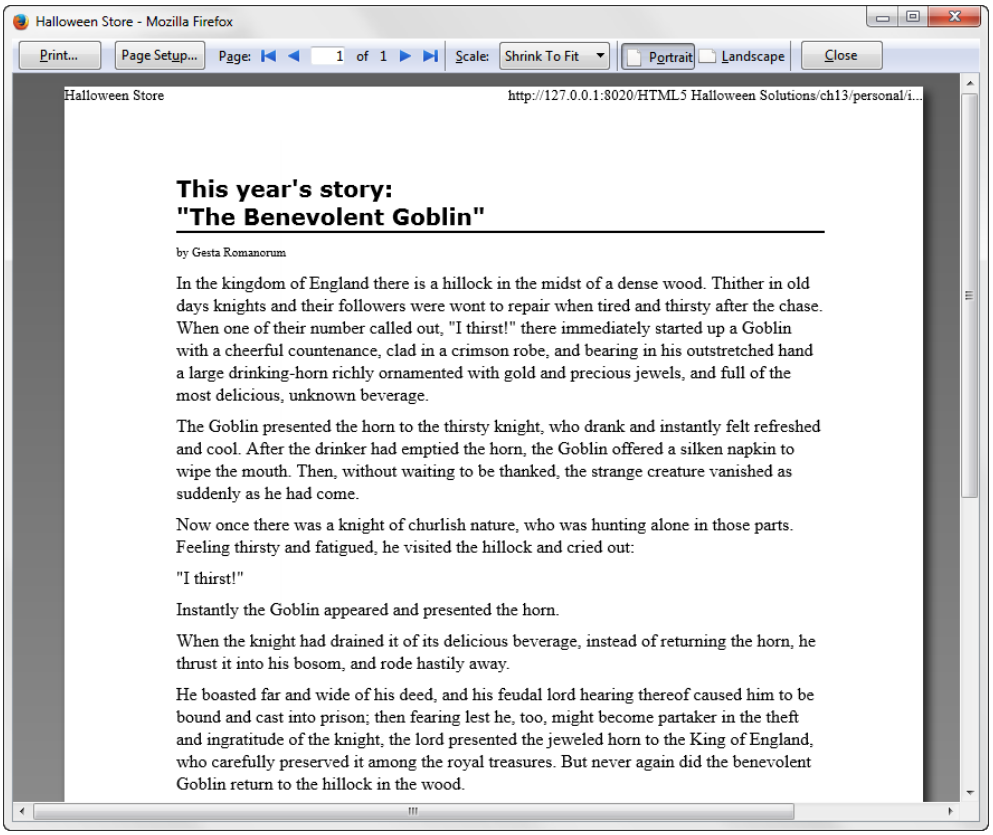
# Halloween 13 Add an embedded font and a story and provide for printing the story

In this exercise, you'll use a Google web font on the personal page that you created in exercise 12. You'll also add a story to this page and provide for printing that story. When you're done, the page should look similar to this:


## Specifications

1. Use the Google Web Fonts page (www.google.com/fonts) to get the link element for using the Creepster font. Copy this link element into the head section of the personal page.
2. Apply the Creepster font to the first heading in the header.
3. Add the story to the personal page, and format it as shown above. You'll find the text for the story in the story.txt file in the text folder. (If you didn't do exercise 12, see the information below for creating the personal page without the video.)

4. Add the required code to the personal.css style sheet to provide for printing the story. Print the author's name and the body of the story in a serif font, but print the heading in a sans-serif font. In addition, add a 3-pixel black rule below the heading.
5. Remove all other elements from the page, and remove the box shadow from around the page. When you're done, the printed output should look like this:




### If you didn't do exercise 12:

6. Start by creating the personal page. To do that, you can copy the index.html file from exercise 7 to the personal folder. Then, you can delete the content from the section and modify the URLs on the page as necessary.
7. Modify the horizontal navigation menu so it indicates that the personal page is the current page.
8. Modify the vertical navigation menu so it provides the links shown above. Use any names you want for the pages, but assume that they're stored in the personal folder.
9. Create a new style sheet named personal.css for the product list page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the personal/index.html file so it points to the correct style sheet.
10. Continue with the specifications listed above.

# Short 13-1 Format the home page for printing

In this exercise, you'll format the home page of the Town Hall website for printing.
Estimated time: 15-20 minutes.


1. Open the HTML file for this page and the CSS file for printing it:

   **`short_exercises\town_hall\c13_index.html`**

   **`short_exercises\town_hall\styles\home_print.css`**

   Note that the CSS file is a copy of the main.css file that is used to format the index page for screens.

2. In the HTML file, add a link element for the home_print.css file and be sure to specify that the medium is "print".

3. In the CSS file, delete all of the rule sets and rules that you don't need to override, and code the rule sets or rules that will stop the display of the navigation menu and the images in the sidebar.

4. Increase the font size in the body element to 100% so the page will be easier to read when it's printed.

5. Modify or add styles to the home_print.css file until the printed page looks like the one above.

## Chapter 14

# How to use CSS3 transitions, transforms, animations, and filters

# Objectives

### Applied

1. Use any of the properties presented in this chapter for working with transitions, transforms, animations, and filters.

### Knowledge

1. Explain how a transition effects the way that a change is applied to an element.
2. Explain what a timing function is and how it affects a transition or animation.
3. Explain how the rotate, scale, skew, and translate transforms along with the transform origin effect an element.
4. Describe the function of the @keyframes selector rule for an animation and two ways it can be coded.
5. Describe what you can do with filters and when they're applied.

# Summary

- When the CSS properties for an HTML element are changed, a CSS3 *transition* provides a smooth change over a specified period of time.
- CSS3 *transforms* let you rotate, scale, skew, and position HTML elements.
- CSS3 *animations* let you create frame-based animations that are similar to what you can create with programs like Flash.
- The CSS3 animation property always points to an *@keyframes selector* rule that defines the *keyframes* that are used in the *animation sequence*. When those keyframes are played, they give the impression of motion.
- CSS3 *filters* let you provide effects like blurring or grayscale to HTML elements like images and backgrounds.

# Terms

transition
timing function
speed curve
transform
animation
@keyframes selector
keyframe
animation sequence
tweening
filter

# Exercise 14-1 Use transitions, transforms, and animation

In this exercise, you'll have some fun with the CSS3 features that you learned about in this chapter. For instance, you'll animate the four images in the aside when the page is loaded, and you'll rotate the speaker image in the article when the user hovers the mouse over it. When both of those features are in progress, the page will look something like this:


## Use a transition

1. Use your text editor to open these pages:

   ```
   c:\html5_css3_2\exercises\town_hall_2\index
   c:\html5_css3_2\exercises\town_hall_2\styles\main.css
   ```

2. With figure 14-1 as a guide, add a transition to the page that moves any of the images in the aside 100 pixels to the right and 20 pixels down when the user hovers the

mouse over one of them. The duration of the transition should be 2 seconds. Then, test this change in Firefox and test all of the changes that follow using that browser.

3. Comment out the transition and test this again to see that the movement still works, but there isn't a gradual transition to the new location. Then, uncomment the transition.

### Use a transition with a transform

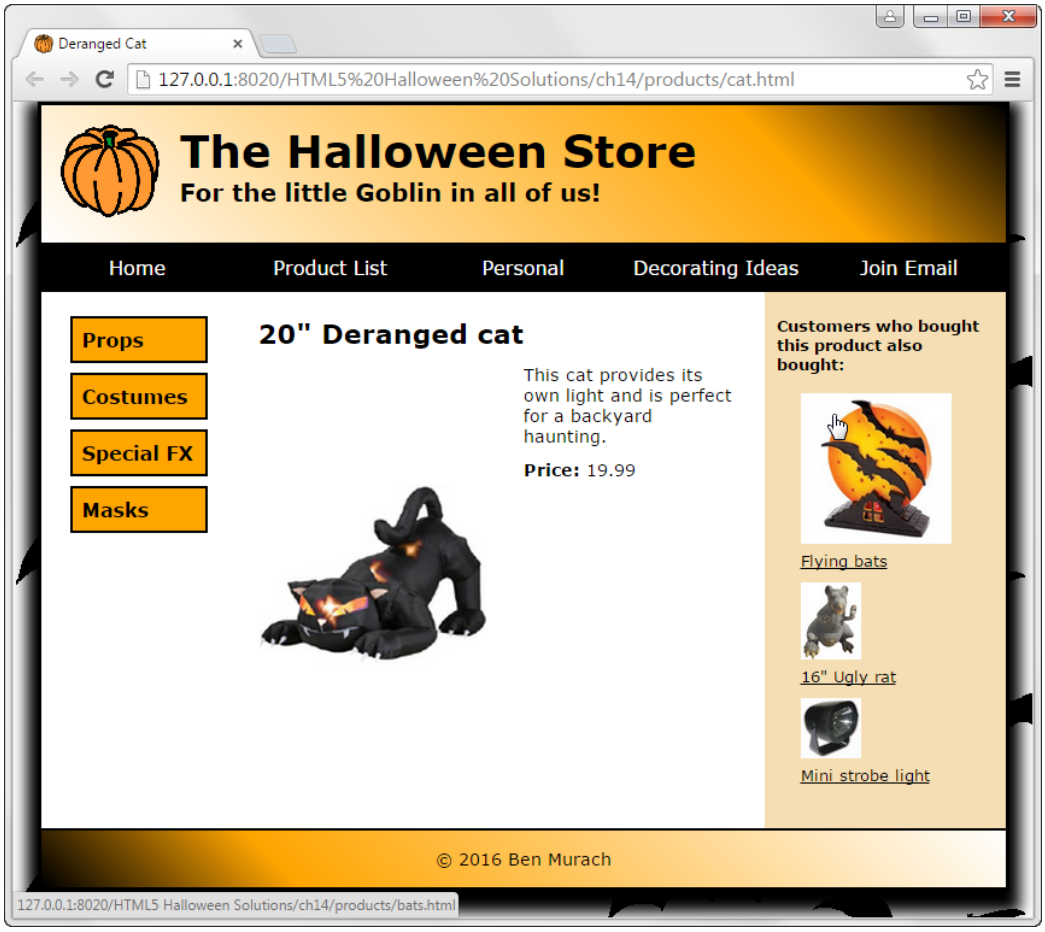
4. With figure 14-3 as a guide, add a transition and a transform that rotates the image in the article by 720 degrees when the user hovers the mouse over the image. The duration of this transition should be 3 seconds.
5. Change the origin of this transform so it is at the upper left of the image. Then, test this change and note the difference.

### Use animation with a transform

6. With figure 14-5 as a guide, animate the four images in the aside so they move to the right 100 pixels and down 20 pixels when the page is loaded into the browser. The animation should be delayed 2 seconds, last 3 seconds, be run 4 times, and alternate directions.
7. Add a transform to this animation that increases the size of the image 1.4 times. You can do that with the scale method.
8. When everything is working, test this page in Chrome to see whether everything still works.

# Halloween 14 Add a transition and an animation

In this exercise, you'll add a transition and an animation to the product page that you created in exercise 9. When you're done, the page should look similar to this:




## Specifications

1. Add a transition for the widths of the images in the right sidebar of the cat.html file in the products folder that will last for three seconds and use the linear speed curve. The transition should increase the width of an image to 125 pixels when the mouse hovers over an image.
2. Add an animation for the image in the section that will cause the image to move up and down when the page is loaded. The animation should last for half a second, use the ease-in-out speed curve, repeat six times, and alternate directions on each repetition. Be sure to provide for all browsers.

## Short 14-1 Use a CSS3 transition and transform

In this exercise, you'll use a transition and a transform to rotate elements when the mouse hovers over them as shown below. Estimated time: 10 to 15 minutes.


1. Open these HTML and CSS files, and run the HTML file:

   **`short_exercises\town_hall\speakers\c14x_sampson.html`**

   **`short_exercises\town_hall\styles\c14x_speaker.css`**

2. Add CSS for a transform that will rotate the list items in the sidebar horizontally 360 degrees when the mouse hovers over them.

3. Add CSS for a transition that will cause the transform to last for 3 seconds.

## Short 14-2 Use CSS3 animation

In this exercise, you'll animate the image in the page header so it appears to bounce from left to right. Estimated time: 10 to 15 minutes.


1. Open these HTML and CSS files, and run the HTML file:

   **`short_exercises\town_hall\speakers\c14x_sampson.html`**

   **`short_exercises\town_hall\styles\c14x_speaker.css`**

2. Add CSS to the image in the header that will use an animation named "bounce" with a duration of half a second that will repeat eight times, use the ease-in timing function, and alternate direction.

3. Add an @keyframes selector rule that defines the keyframes for the animation. In the from group, set the left and right padding so the animation starts at the image's current position on the page. In the to group, set the left and right padding so the image moves to the right 20 pixels.

## Chapter 15

# How to use JavaScript and jQuery to enhance your web pages

### Introduction to JavaScript

How JavaScript works
Three ways to include JavaScript in a web page
How DOM scripting works
Methods and properties for DOM scripting
How JavaScript handles events

### The Email List application in JavaScript

The HTML
The JavaScript

### Introduction to jQuery

How to include jQuery in your web pages
How to code jQuery selectors, methods, and event methods

### The Email List application in jQuery

The HTML
The jQuery

### How to use jQuery as a non-progammer

The Image Rollover application
The Image Swap application
A Slide Show application
Websites for JavaScript and jQuery code

# Objectives

### Applied

1. Given an HTML document, use JavaScript to enhance the page so it includes the current date or year or a Print button.
2. Given an HTML document, use jQuery to enhance the page so it includes an image rollover, image swap, or slide show.
3. Use the Internet to find JavaScript and jQuery code that you can use for other functions.

### Knowledge

1. Describe how JavaScript works.
2. Describe three ways that JavaScript can be used with an HTML document.
3. Describe the Document Object Model (DOM) and explain what happens when DOM scripting is used to change the DOM.
4. In general terms, describe how you script the DOM and how the $ sign is commonly used for DOM scripting.
5. Describe the way that JavaScript event handlers work with DOM scripting.
6. Describe jQuery and the two ways you can include it in a web page.
7. Describe the syntax for calling jQuery methods, including the use of the $ sign, selectors, and the dot operator.
8. Describe the syntax for coding a jQuery event handler, including the use of an event method and function.
9. Describe the use of the jQuery ready method.
10. In general terms, describe how you use jQuery for functions like image rollovers, image swaps, and slide shows.

## Summary

- *JavaScript* is a scripting language that is run by the *JavaScript engine* of a browser. As a result, the work is done on the client, not the server, and that takes some of the processing burden off the server.
- To embed JavaScript code or to include JavaScript files in an HTML document, you code a script element. If a script element is coded in the body of a document, it is replaced by the output of the JavaScript code.
- The *DOM* (*Document Object Model*) is a hierarchical collection of *nodes* in the web browser's memory that represents the current web page. The DOM for each page is built as the page is loaded.
- *DOM scripting* is the process of changing the DOM by using JavaScript. When the DOM changes, the browser immediately displays the results of the change.
- An *event* is an action the user performs, like clicking on a button or image. When an event occurs, it can be handled by JavaScript code known as an *event handler*.
- *jQuery* is a JavaScript library that makes JavaScript programming easier. To use jQuery, you code a script element in the head section that either includes a downloaded jQuery file or accesses it through a *Content Delivery Network* (*CDN*).
- When you code statements that use jQuery, you use *selectors* that are like those for CSS. You also use jQuery *methods* and *event methods*.

## Terms

JavaScript
JavaScript engine
client-side processing
server-side processing
DOM (Document Object Model)
node
element node
text node
attribute node
comment node
DOM scripting
scripting the DOM
document object
event
event handler
data validation
jQuery
CDN (Content Delivery Network)
jQuery selector
jQuery method
jQuery event method
image rollover
image swap

# Exercise 15-1 Enhance a page with JavaScript

In the exercises for this section, you'll be working with a third version of the Town Hall website. In this exercise, you'll make two JavaScript enhancements to the speaker's page for Scott Sampson:


### Open the HTML and CSS files for this page

1. Use your text editor to open the HTML file for this page:

   `c:\html5_css3_2\exercises\town_hall_3\speakers\sampson.html`

2. Run the page in your browser to see that it looks like the one above, but without the Print Page button.

### Add the Print Page button

3. With figure 15-5 as a guide, add the script element for the external printPage.js file that's in the javascript folder. Then, add the Print Page button to the page at the top of the aside, and be sure to set its id to "printButton".
4. Test this enhancement. If it doesn't work, make sure that you have the script element and the button id coded correctly.
5. When you get this working, note that the formatting of the button isn't the same as above. To fix that, adjust the CSS file for the page, and test again.
6. If you preview the printing for this page, you'll see that the Print Page button will be printed. If you've already read chapter 13 and know how to fix this, adjust the media query for the printed page so the button isn't printed.

### Automatically update the year in the footer

7. With figure 15-1 as a guide, use JavaScript to get the current year and put it into the copyright line in the footer. Then, test this change.

## Exercise 15-2 Use jQuery for image swaps

In this exercise, you'll enhance a page so it uses jQuery to do image swaps. That page will look like the one below, which is the page that's displayed when you click on the Speakers link in the navigation bar. Then, when the user clicks on a small image, a larger image of that speaker is displayed below the small images.


### Open the HTML and CSS files for this page

1. Use your text editor to open the HTML file for this page:

   `c:\html5_css3_2\exercises\town_hall_3\image_swaps.html`

2. Look in the head section of the HTML file to see that it contains two script elements that load the jQuery library and the JavaScript file for this application. Then, test this page to see that it looks okay, but it doesn't work.

### Modify the HTML so the image swaps work

3. Using figure 15-13 as a guide, modify the HTML so it should work. You can assume that the ul element is coded correctly, so the problem is elsewhere.

4. When the image swaps are working, try each one and note that the caption for Amy Chua is incorrect. So there actually is a problem somewhere within the ul element. Now, fix this.

# Halloween 15 Use JavaScript and jQuery to enhance the footer and add a slide show

In this exercise, you'll create a decorations page that uses jQuery to present a slide show with decorating ideas. In addition, you'll use JavaScript to add a footer that indicates when Halloween is. When you're through, the page should look similar to this:


## Specifications

1. To create the decorations page, you can copy the index.html file from exercise 7 to the decorations folder. Then, you can delete the content from the section and modify the URLs on the page as necessary.
2. Modify the horizontal navigation menu so it indicates that the decorations page is the current page.
3. Modify the vertical navigation menu so it provides the links shown above. Use any names you want for the pages, but assume that they're stored in the decorations folder.

4. Create a new style sheet named decorations.css for the decorations page, and copy the styles you need from the main.css file to this style sheet. Then, modify the link element for the style sheet in the decorations/index.html file so it points to the correct style sheet.
5. Add the JavaScript that creates the footer to the cat.html file. You'll find this script in the days_to_halloween.js file in the js folder.
6. Add the script elements that are needed to create a slide show. You'll find the slide_show.js file in the js folder, and you can get the jQuery library from the jQuery Content Delivery Network.
7. Add a heading and the slide show to the section. You'll find the image files you need in the slides/decorations folder.
8. Add any other formatting as necessary so the page looks as shown above.

## Short 15-1 Do image rollovers using jQuery

In this exercise, you'll do image rollovers by using jQuery. Estimated time: 15-20 minutes.


1. Open these HTML and CSS files, and run the HTML file:

   **`short_exercises\town_hall\speakers\c15x_sampson.html`**

   **`short_exercises\town_hall\styles\c15x_speaker.css`**

   Notice that the aside appears the same as it did before, but the speakers are now coded within an unordered list.

2. Add a script element like the one in figure 15-10 to the head section of the HTML document that loads the jQuery library from a Content Delivery Network.

3. Add another script element to the head section for the rollover.js file that's in the javascript folder.

4. Use figure 15-12 as a guide to add the code to the unordered list that's needed to use image rollovers with the speaker images. The alternate images are named toobin_court.jpg, sorkin_desk260.jpg, chua_220.jpg, and sampson_dinosaur.jpg. Because these images are larger than the initial images, you'll need to set the image heights to 75 pixels in the CSS.

## Chapter 16

# How to use jQuery UI and jQuery plugins to enhance your web pages

### Introduction to jQuery UI

What jQuery UI is and where to get it
How to download jQuery UI
How to include jQuery UI in your web pages
How to use any jQuery UI widget

### How to use four of the most popular jQuery UI widgets

How to use the Accordion widget
How to use the Tabs widget
How to use the Button and Dialog widgets

### Introduction to jQuery plugins

How to find jQuery plugins
How to use any jQuery plugin

### How to use three of the most popular plugins

How to use the Lightbox plugin for images
How to use the bxSlider plugin for carousels
How to use the Cycle 2 plugin for slide shows

# Objectives

### Applied

1. Given an HTML document, use jQuery UI to enhance the page so it includes any of the following widgets: Accordion, Tabs, Button, or Dialog.
2. Given an HTML document, enhance the page with any of the following jQuery UI plugins: Lightbox, bxSlider, or Cycle 2.
3. Use the Internet to find jQuery plugins that you can use for other functions.

### Knowledge

1. In general terms, describe jQuery UI and the four features it provides.
2. Explain why you might want to build a custom jQuery UI download.
3. Describe the three files that jQuery UI requires.
4. In general terms, describe how you use jQuery UI for functions like accordions, tabs, buttons, and dialogs.
5. In general terms, explain what a plugin is.
6. In general terms, describe the process of finding and using a plugin.
7. In general terms, describe how you use jQuery plugins for features like lightboxes, carousels, and slide shows.

## Summary

- *jQuery UI (User Interface)* is a JavaScript library that extends the jQuery library. Since the jQuery UI library uses the jQuery library, the script element for jQuery UI must come after the script element for jQuery.
- Before you download the jQuery UI library, you can build a custom download that can include a custom *theme* for styling the jQuery UI components. This theme is implemented by a CSS style sheet that's part of the download.
- As you build a jQuery UI download, you can select the components for the features that you're going to use, including widgets, interactions, and effects. As you would expect, the fewer components you select, the smaller the jQuery UI file that has to be loaded into the user's browser.
- The most widely-used jQuery UI components are the *widgets*. To use a widget, you code the prescribed HTML for it. Then, in the jQuery, you select the widget and run its primary method, often with one or more options.
- If you need a common function for a web page, chances are that a *jQuery plugin* is already available for it. By using a plugin, you can often save hours of work and do the job even better than you would have done it on your own.
- To access a plugin, you code a script element for it in the head element of the HTML document. This script element must come after the script element for the jQuery library, because all jQuery plugins use that library.
- To use a plugin, you often need to use jQuery code to call its method within the ready event handler for a page. Then, if the plugin requires options, you code the options as part of the method call. For some plugins, though, you just need to code the HTML and the required attributes correctly.

## Terms

jQuery UI (User Interface)
widget
theme
modal
jQuery plugin

# Exercise 16-1 Add an Accordion widget to a page

In this exercise, you'll add an Accordion to a page so it looks like this:


### Open and review the HTML for this page

1. Use your text editor to open the HTML file for this page:

   `c:\html5_css3_2\exercises\town_hall_3\speakers\accordion_dialog.html`

2. Review the files in the jquery-ui-1.11.2.custom folder. This is the folder for a custom jQuery UI download. It contains the images folder, the jquery-ui.min.css file for the smoothness theme, and the jquery-ui.min.js file. Then, look at the link and style elements in the accordion_dialog.html file to see that these jQuery UI files have been included for the web page.

3. Note that the last script element contains the jQuery code for implementing the accordion. Then, test this page to see that the accordion works in default mode. That means that an accordion panel is opened when you click on its heading, the height is the same for all panels, and you can't close all of the panels.

### Change the way the Accordion works and looks

4. Modify the CSS for this page so any image in an accordion panel will be floated left with appropriate margins as shown above.

5. With figure 16-4 as a guide, modify the script for the accordion so the user can close all of the panels, but note carefully the punctuation that's required.

6. Add the heightStyle option to the accordion with its value set to "content". That way, the size of each accordion panel will depend on its content.

7. Test the page to be sure these changes work.

## Exercise 16-2 Use the Cycle 2 plugin for a slide show

In this exercise, you'll enhance a page so it uses the Cycle 2 plugin to run a slide show that displays the speakers. That page will look like this:


### Open and review the HTML for this page

1. Use your text editor to open the HTML file for this page:

   `c:\html5_css3_2\exercises\town_hall_3\speakers\slideshow.html`

2. Review the link and style elements for this page. Note that there isn't a CSS file for the Cycle 2 plugin, and that the script element gets the JavaScript file for the plugin from a CDN. Note too that a second script element gets a cycle2.shuffle JavaScript file from a CDN.

3. Test this page to see that it runs a default Cycle2 slide show.

### Change the way the Slide Show works and looks

4. With figure 16-11 as a guide, add a data-cycle-timeout attribute to the div element to the slide show so each slide is shown for 2 seconds.

5. With figure 16-11 as a guide, add a data-cycle-fx attribute so the shuffle transition is used for the slides. This option requires the use of the cycle2.shuffle.js file that's included by the third script element.

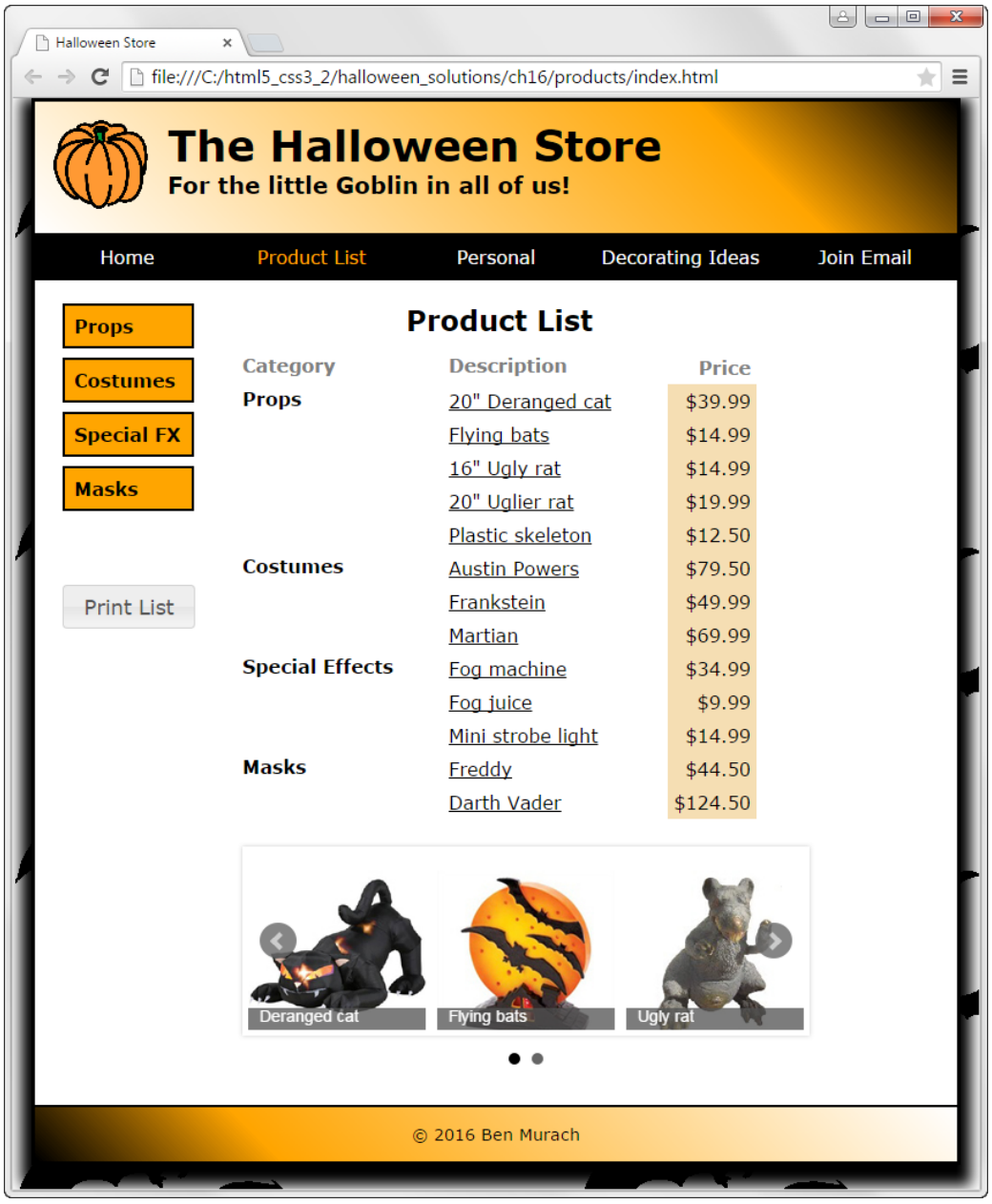
6. Add a data-cycle-pause-on-hover attribute set to a value of "true" so the slide show will pause when the user hovers the mouse over a slide.

7. Go the website for the Cycle 2 plugin, and then go to the demo for providing captions that get their text from the alt attributes of the images. Then, see if you can add the code to the HTML file that will make this work. The caption should be displayed below the images as shown above.

8. Test the page to be sure these changes work.

# Halloween 16 Use jQuery UI and jQuery plugins to create a button and a carousel

In this exercise, you'll modify the product list page that you created in exercise 10 so it includes a button widget for printing the product list and a carousel of products. When you're done, the page should look similar to this:



### Specifications

- Add the link and script element that are needed to use the jQuery UI Button widget. You'll find the jquery-ui.min.css and jquery-ui-min.js files in the jquery-ui-1.11.2.custom folder, and you can get the jQuery library from the jQuery Content Delivery Network.
- Add a link to the aside after the navigation menu that will print the product list. Don't worry about making this button work.
- Add a script element that will format this link so it looks like a button.
- Add the link and script elements that are needed to use the bxSlider plugin to create a carousel. You'll find the jquery.bxslider.css file in the styles folder, and you'll find the jQuery.bxSlider.min.js file in the js folder.
- Add the carousel after the table in the section. You'll find the product images for the slides in the slides/products folder.
- Add a script element that uses the bxSlider plugin to implement the carousel. The carousel should run automatically, it should stop running when the mouse hovers over the slider (use the autoHover option), it should display captions like the ones shown above, it should always display three slides, it should have a width of 154 pixels, and it should have a margin between slides of 10 pixels.
- Add any other formatting as necessary so the page looks as shown above.
- Note: Remember that if you use Aptana to run a page that uses the bxSlider plugin, it won't work correctly. Because of that, you'll have to access the page directly from your browser.

## Short 16-1 Use jQuery UI for tabs

In this exercise, you'll use JavaScript to format the data in a page so it appears in tabs as shown below. Estimated time: 15-20 minutes.


1. Open the HTML and CSS files for this page:

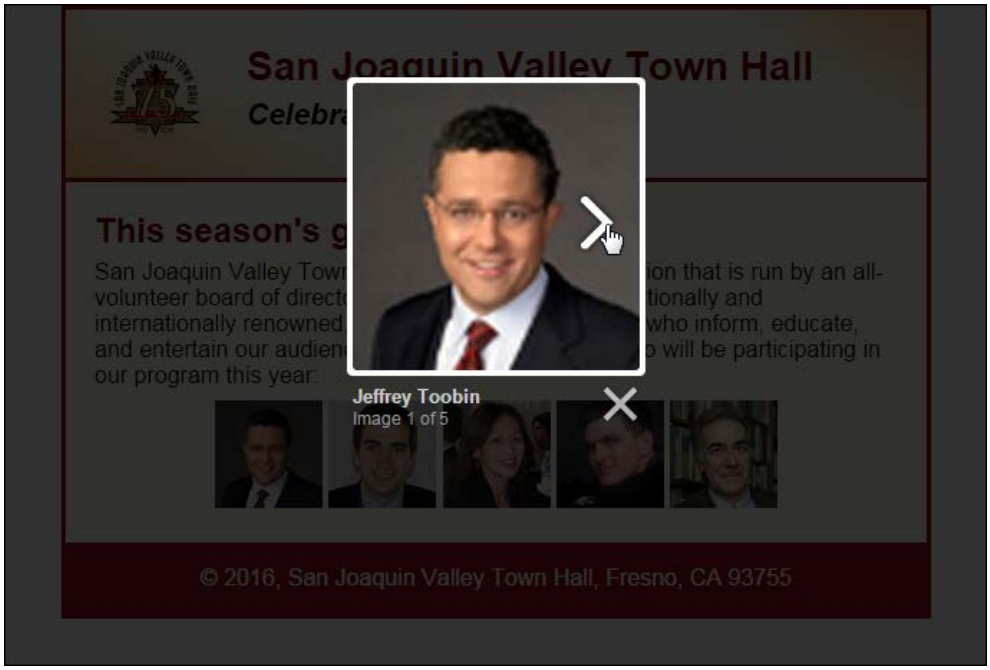
   **`short_exercises\town_hall\speakers\c16x_sorkin.html`**

   **`short_exercises\town_hall\styles\c16x_sorkin.css`**

2. Review the HTML file to see that it contains a div element with an unordered list that defines the tabs for the page.
3. Add the link and script elements that are required for using jQuery UI as shown in figure 16-2.
4. Add another script element that implements the Tabs widget as shown in figure 16-5.
5. Use figure 16-5 as a guide to code the div elements that will hold the contents of the tabs. Then, move the appropriate content into each tab.
6. Change the CSS for the image so it doesn't float.

## Short 16-2 Use the Lightbox plugin

In this exercise, you'll use the jQuery Lightbox plugin to display larger versions of thumbnail images as shown below. Estimated time: 10-15 minutes.




1. Open the HTML and CSS files for this page:

   **`short_exercises\town_hall\c16x_speakers.html`**

   **`short_exercises\town_hall\styles\c16x_speakers.css`**

2. Review the HTML file to see that the head section includes the link and script elements for the Lightbox plugin as well as a link element for the jQuery library.

3. Run the HTML file to see what you're starting with.

4. Use figure 16-9 as a guide to modify the code in the section element so a larger version of a speaker image is displayed when the user clicks on the image. Each image should be displayed with a caption and a counter as shown above. The larger images have the same names as the thumbnail images, except the "75" in each name is replaced by "200".

5. Modify the CSS so no underlines appear between the images. You can do that by removing the text decoration from the <a> elements you just added.

## Chapter 17

# How to use jQuery Mobile to build mobile websites

### How to work with mobile devices

How to provide pages for mobile devices
How to use a JavaScript plugin to redirect mobile browsers to a mobile website
How to set the viewport properties
Guidelines for designing mobile web pages
Guidelines for testing mobile web pages

### How to get started with jQuery Mobile

What jQuery Mobile is and where to get it
How to include jQuery Mobile in your web pages
How to create one web page with jQuery Mobile
How to code multiple pages in a single HTML file

### How to navigate between pages

How to use dialogs and transitions
How to create buttons
How to create a navigation bar

### How to format content with jQuery Mobile

The default styles that jQuery Mobile uses
How to apply themes to HTML elements

### How to use jQuery Mobile for page layout

How to create collapsible content blocks
How to create an accordion
How to create a list

### A mobile website that uses jQuery Mobile

The layout for the mobile website
The HTML for the mobile website

# Objectives

### Applied

1. Use the jQuery Mobile features that are presented in this chapter to build a website for mobile devices.

### Knowledge

1. Explain why you might want to create a separate website for mobile devices rather than convert an existing site to use Responsive Web Design.
2. Explain how a JavaScript plugin can be used by a full website to redirect an HTTP request from a mobile device to a mobile website.
3. Describe the use of the viewport meta element for mobile devices.
4. Explain how testing a mobile website differs from testing a website for computer screens.
5. Describe jQuery Mobile and the three files that it requires.
6. Describe two ways to include the jQuery Mobile files in your web pages.
7. In general terms, describe how you use jQuery Mobile to build mobile websites.
8. Describe how these attributes are used by jQuery Mobile: data-role, data-transition, data-dialog, data-rel, data-icon, data-theme, and class.
9. Describe two ways that you can apply the swatches in a theme to the components of a web page when you're using jQuery Mobile.

## Summary

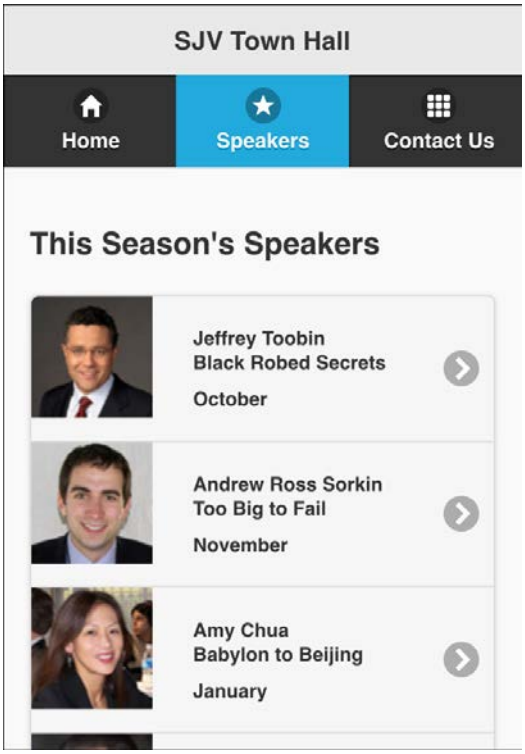
- When Responsive Web Design is impractical, it makes sense to build a separate website for mobile devices. Then, you can use client-side or server-side code to detect mobile devices and redirect them from your full website to your mobile website.
- The *viewport* on a mobile device determines the content that's displayed. To control how that works, you can code a viewport meta element in the head section of a page.
- *jQuery Mobile* is a JavaScript library that's designed for developing mobile websites. jQuery Mobile uses the jQuery library along with its own CSS file.
- To include the jQuery Mobile and jQuery libraries in a web page, you code script elements in the head section. Because jQuery Mobile uses jQuery, the script element for jQuery Mobile must be coded after the one for jQuery.
- jQuery Mobile lets you code the HTML for many mobile pages in a single HTML file. jQuery Mobile also supports the use of dialogs, transitions, buttons, navigation bars, collapsible content blocks, accordions, and more.
- By default, jQuery Mobile uses a *theme* that provides formatting that relies on a browser's native rendering engine. jQuery Mobile also provides two *swatches* that you can use to adjust the default formatting without using CSS style sheets of your own.

## Terms

viewport
jQuery Mobile
dialog
transition
navigation bar
theme
swatch
collapsible content block
accordion

# Exercise 17-1 Test and modify the mobile website for Town Hall

In this exercise, you'll first test the mobile version of the Town Hall website that's presented in this chapter. Then, you'll make some modifications to it, like changing the Speakers page so it looks like the one that follows. This should demonstrate how easy it is to build mobile websites when you use jQuery Mobile.




### Open and test the mobile website for Town Hall

1. Use your text editor to open this HTML page:

   **`c:\html5_css3_2\exercises\town_hall_mobile\index.html`**

2. Test this page and the navigation from one page to another within the site. The easiest way to do that is to run the page in Chrome, press F12 to display the developer tools, click the Toggle Device Mode icon near the left side of the toolbar, and then choose a mobile device like iPhone 6. Then, you can use your mouse to "tap" on a link. As you go from page to page, remember that all of the pages come from one HTML file.

3. Click on the image on the home page to go to the page for Scott Sampson, and then click the button at the bottom of that page to return to the home page. Next, click the Speakers button in the navigation bar and then the link for Scott Sampson to go to the page for Scott Sampson. This time, when you click on the button at the bottom of the page, you return to the Speakers page.

### Modify the way the pages work and test these changes

4. Review the code for the button on Scott Sampson's page. Then, modify the code so it returns to the Speakers page no matter how the user got there.
5. In the HTML for the Speakers page, delete the second <a> element for each speaker. Then, test this change to see how the Speakers page has changed and how the navigation works.
6. Add the small images of the speakers to the Speakers page as shown in figure 17-16. Then, adjust the formatting for the text so the page looks something like the page above, but don't use CSS to do that. Instead, experiment with HTML elements like the h4, p, and strong elements. Without using CSS, this formatting can be time-consuming.
7. Add a page for Andrew Ross Sorkin that's like the ones for the other speakers. Use "Too Big to Fail" as the heading for the page and follow that with the large image for him, but don't bother adding any content beyond that. Then, test to make sure that this works and that the navigation to and from the page works.

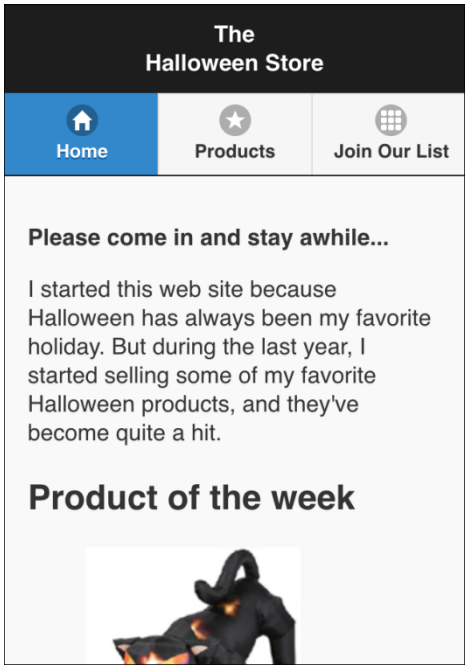
### Experiment on your own

8. Experiment with themes. Try using CSS to format the Speakers page. Add a page that uses a table. You'll soon see how quickly you can build a mobile website with jQuery Mobile. Keep in mind, though, that this chapter has presented fewer than half of the jQuery Mobile features.

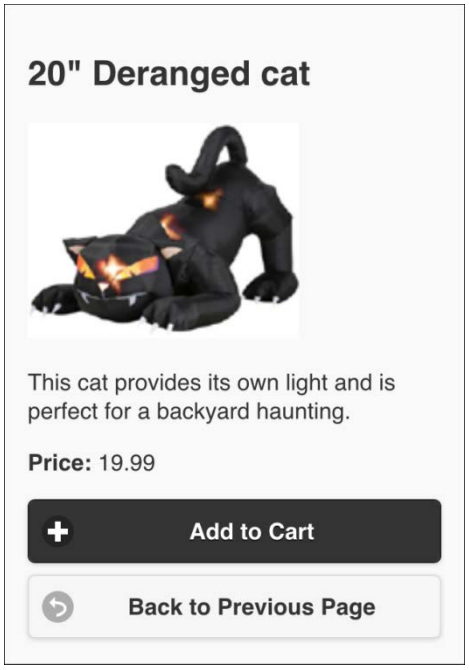
# Halloween 17 Create a mobile website

In this exercise, you'll use jQuery Mobile to create a mobile version of the Halloween Store website that you've been developing.

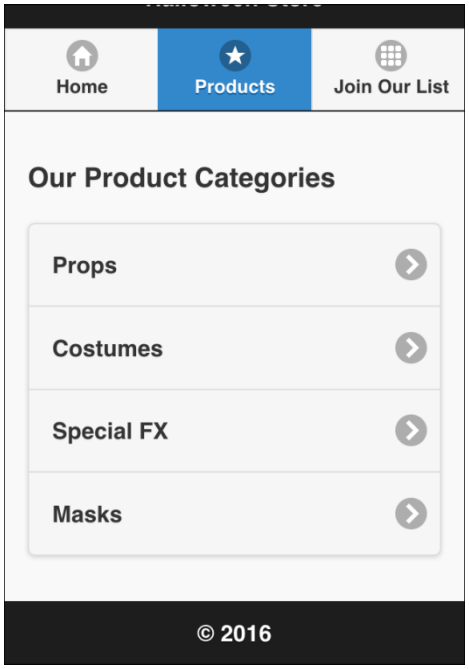
**Home page**



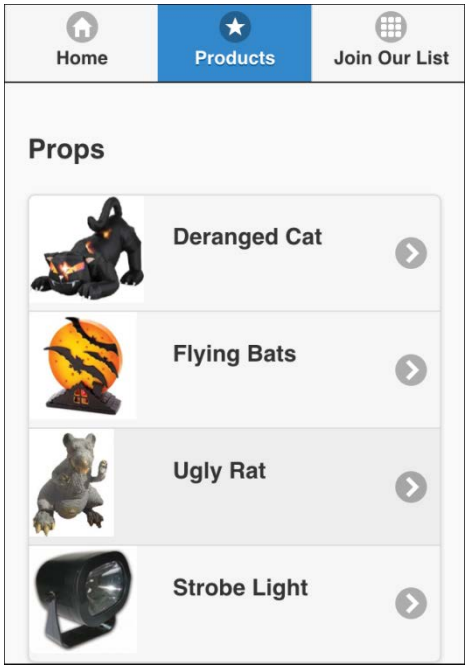

**Deranged Cat page**




**Categories page**




**Props page**



### Specifications

- In the mobile folder for the Halloween Store exercises, you'll find a file named index.html. This file contains the code for the Town Hall mobile website that's presented in the book. To create the mobile version of the Halloween Store website, you'll modify the code in this file.
- For the home page, you can copy the code from the home page that you developed for exercise 6 and delete the content that isn't needed. Here, all of the code for the product of the week and the guarantee should be included, but the text before the product should be abridged as shown above.
- For all of the pages, use the header and navigation bar shown in the home page above and the footer shown in the Categories page. Also, the theme for the header and footer should be changed to "b", and the theme for the buttons in the navigation bar should be changed to "a".
- For the Deranged Cat page, you can copy in the code for the cat page that you developed for exercise 9. At the bottom of this page, you should have Add to Cart and Back to Previous Page buttons with the plus and back icons preceding the text. The Add to Cart button should use theme "b".
- For the Product Category and Props pages, you can start from the code for the list that's already in the HTML file.
- When you finish developing the four pages for the Halloween Store website, you can delete the code for the other pages in the HTML file.

## Chapter 18

# How to design a website

### Users and usability

What web users want is usability
The current conventions for usability

### Design guidelines

Think mobile from the start
Use the home page to sell the site
Let the users know where they are
Make the best use of web page space
Divide long pages into shorter chunks
Know the principles of graphics design
Write for the web

### How to design a website

The lifecycle of a website
Step 1: Define the audience and set the goals
Step 2: Develop the site map
Step 3: Wireframe the critical pages
Step 4: Illustrate the critical pages

### Other design considerations

Development teams
Top-down design and prototyping

# Objectives

### Applied

1. Design a website using the guidelines, methods, and procedures of this chapter.

### Knowledge

1. Describe the notion of usability and explain how the current website conventions contribute to usability.
2. Describe two basic approaches that you can use to design the pages for Responsive Web Design.
3. Describe the primary principle for home-page design if you're designing a small site for a relatively unknown company.
4. Describe three ways to let the users of a website know where they are.
5. Describe the use of chunking.
6. List three guidelines for typography on a web page.
7. List three guidelines for web writing.
8. Describe the lifecycle of a website.
9. Describe the most important principle for making a website easier to maintain.
10. List the four steps in a general procedure for designing a website.
11. Describe the use of a site map for designing a website.
12. Describe the use of wireframes for designing web pages.
13. Distinguish between a web designer and a graphics designer.
14. Describe the concepts that top-down design and prototyping are based upon.

## Summary

- When you design a website, *usability* is the primary issue. That refers to how easy a website is to use, and to a large extent that determines the success of the website. To achieve usability, a website needs to implement the current conventions for website use.
- The home page of a website should be used to sell the site by emphasizing what's different about the site and why users should want to use it.
- To let the users know where they are as they navigate through a site, you can highlight the active links on the page, match the heading on the page to the link that led to it, and use *breadcrumbs*.
- *Chunking* refers to the division of information into topics (chunks) that can be presented on separate web pages or in separate components on a page.
- If you aren't a graphics designer, you should at least know the principles of *alignment*, *proximity*, *repetition*, and *contrast*. You should also know how to make the typography easy to read.
- Writing for the web isn't like writing for print. Instead, web writing should be written in inverted pyramid style, use headings and subheadings, and use bulleted and numbered lists.
- The *lifecycle* of all systems consists of design, implementation, and maintenance. Often, more time and money are spent on the maintenance of a website than on its development.
- When you design a website, you start by defining its *target audience* and goals. Then, those definitions guide you as you develop the site map and design the pages for the site.
- The *site map* identifies all of the pages of the website as well as the links in the navigation bar.
- To plan the designs of the pages for a website, you start with black-and-white *wireframes* so the focus is on function, not appearance. Then, you develop full color illustrations of those pages.
- With few exceptions, you will work with other people on the design of a site. In a large company, the design team may include web designers, writers, marketing specialists, and graphics designers.
- A *web designer* is involved with all aspects of web design. A *graphics designer* focuses on the graphics that make a website look inviting and manageable.
- When you use *top-down* design for a website, you design the most critical pages first, then the next most critical set of pages, and so forth. This implies that you don't have to complete the site map for all of the pages of the site before you design the critical pages. Instead, you work on one set of pages at a time.
- When you *prototype* the critical pages of a website, you develop a working model of those pages. Then, the reviewers can actually run the pages and see how they work. This works in combination with top-down design, and it can resolve misunderstandings early in the design process when they're easier to fix.

## Terms

usability
breadcrumbs
chunking
alignment
proximity
repetition
contrast
lifecycle
target audience
site map
wireframe
web designer
graphics designer
top-down design
prototyping

## Chapter 19

# How to deploy a website on a web server

### How to get a web host and domain name

How to find a web host
How to get a domain name

### How to transfer files to and from the web

How to install FileZilla Client
How to connect to a website on a remote web server
How to upload and download files

### Four more skills for deploying a website

How to test a website that has been uploaded to the web server
How to get your website into search engines and directories
How to control which pages are indexed and visited
How to maintain a healthy website

## Objectives

### Applied

1. Deploy a website to a web server that has Internet access.
2. Get your website into the major search engines and directories.

### Knowledge

1. Explain how to get a web host and a domain name.
2. Describe the use of an FTP program for transferring files to and from a website on a server.
3. Describe the process of testing a website after it has been deployed.
4. Describe the process for getting your website into search engines and directories.
5. Describe two ways to control which pages on your site are indexed.

## Summary

- To *deploy* (or *publish*) a website, you can use a *web host*, or *web hosting service*, that will make your website accessible from the Internet.
- You can use *File Transfer Protocol* (*FTP*) to transfer the files for your website to and from your web host.
- An *IP address* uniquely identifies a website, and so does a *domain name*.
- To find a *domain name* for your website, you search the *domain name registry*, which is a database of all registered domain names.
- Once you have a domain name, you can have one or more *subdomains* that address portions of the domain.
- You can use an FTP program to *upload* files from your computer to your Internet web server and to *download* files from your web server to your computer. FileZilla is a free FTP program that runs on the Windows, Mac, and Linux operating systems.
- To test a web page that has been uploaded to the Internet, start your web browser, navigate to the URL for the web page, and click on all links to make sure they work correctly.
- To get your website into a search engine, go to the URL for its submission page. Then, its *robot* (or *spider*) will crawl through your pages and score them for later searches that are based on keywords.
- To stop pages from being indexed by a search engine, you can use a robots.txt file or robots meta tags. To redirect the request for an inactive page on your website to another page, you can use a refresh meta tag.

## Terms

deploy
publish
web host
web hosting service
FTP (File Transfer Protocol)
ISP (Internet Service Provider)
IP address
domain name
subdomain
domain name registry
upload a file
download a file
robot
spider

# Student projects

Each of the projects that follows asks you to create a small website of your own. You can do that after you complete the first eight chapters of *Murach's HTML5 and CSS3 (Third edition)*. Then, you can enhance the initial version of your website as you read the chapters in sections 2 and 3 of the book.

## Starting specifications for all of the projects

### Prerequisites

- Chapters 1 through 8
- (Optional) Chapter 18

### General

- Develop a website that consists of at least three pages: a home page and two content pages.
- You decide what subject you're going to use for your website. It can be about you, a relative, a friend, a pet, a course, a hobby, a product, an issue…you decide.
- Be sure that you know enough about the subject of your website so it will be easy for you to create the content for your pages. If possible, you should also have three or more related image files that you can use in your web pages.

### Content

The goal of these projects is to give you a chance to develop a small website using HTML and CSS, not to create content. With that in mind, you should feel free to:

- Copy content from other websites or sources and adapt it to your own website
- Copy images from other websites or sources

### Skills

As you develop your website, you should of course demonstrate your HTML and CSS skills. To that end, the initial version of your website should show your mastery of:

- Headings, text, and character entities
- Links and lists
- Images
- Margins and padding
- Borders and backgrounds
- Floating
- Responsive Web Design

### Design, graphics, and typography

- Use the principles and guidelines presented in this book and especially in chapter 18 to make the design of your website as effective as possible.
- Implement the first two guidelines in figure 18-5 to let the users know where they are in your website.
- Implement the typographical guidelines in figure 18-8 throughout your site.
- Do your best to implement the graphics design principles in figure 18-8.
- Do your best to write for the web by implementing the guidelines in figure 18-9.

### HTML and CSS files and directories

- The HTML file for the home page should be named index.html.
- The names of the other pages should indicate what the pages represent.
- The images for the website should be stored in a directory named images, the CSS files should be stored in a directory named styles, and the HTML files should be stored in the root directory for the website.

## Section 2 enhancements

As you read the chapters in section 2, you will learn new skills that you can use to enhance your website, either by adding to an existing page or by adding another page to your site. Here are some ideas for those enhancements.

### Chapter 9: Images

- Add links that consist of images.
- Create and add a favicon.
- Add an image to a figure that includes a caption.
- Add an image rollover using background images.
- Add an image map that links to two or more different pages.

### Chapter 10: Tables

- Add a table that presents tabular data to the site.

### Chapter 11: Forms

- Add a form that lets the user enter first name, last name, email address, and comments along with Reset and Submit buttons.
- Of course, the data won't be processed because there's no client-side or server-side script for doing that. However, the data should be submitted to a page that has a link back to the home page. Also, the data that is passed to the page should be visible in the URL for the page.

### Chapter 12: Audio and video

- Add audio or video to your site by embedding the players or files in your pages.

### Chapter 13: Fonts and printing

- Use an embedded font in your site.
- Create a style sheet for printing the home page of your site in a readable form.

### Chapter 14: CSS3 transitions, transforms, filters, and animation

- Add a transition to your site or a transform that uses a transition.
- Add an animation to your site.

## Section 3 enhancements

### Chapter 15: JavaScript and jQuery

Add one or more of the following elements to your site:

- The current date
- A Print button that will print the home page of your site
- An image rollover
- Thumbnail images and image swaps
- A slide show

### Chapter 16: jQuery UI and jQuery plugins

Use jQuery UI widgets to add one or more of the following features to your site:

- An accordion
- Tabbed data
- A dialog box
- A button

Use jQuery plugins to add one or more of the following features to your site:

- Thumbnail images and lightboxes
- A carousel
- A slide show

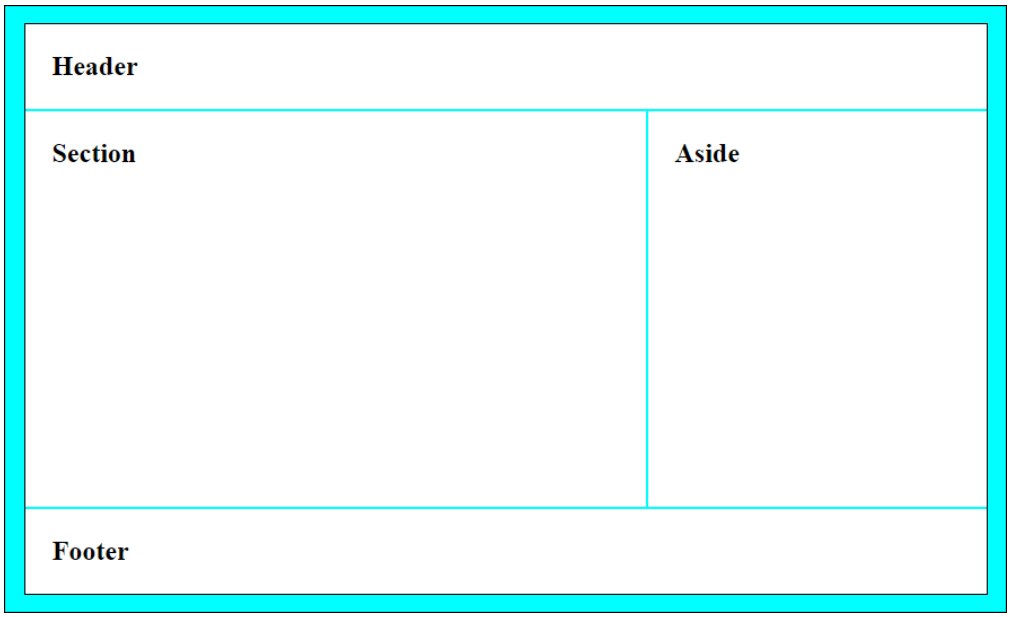
### Chapter 17: jQuery Mobile

- Use the jQuery Mobile library to develop a mobile home page for your site.

## Project 1: Two columns with header and footer

This project asks you to develop a website that consists of a home page and two or more other pages, all with a header, two columns, and a footer.

### Sketch of the general layout of the pages




### Page layout

- Each page should be fixed width and centered in the browser window.
- The header for each page should include a website title and a tag line, but a logo image is optional.
- You don't have to use borders for any of the components of a page. The sketch above is just intended to show you what the components of the pages should be.

### Page contents

- The section for the home page should include an image and text.
- The home page should include an unordered or ordered list. This list can be in the section or the aside.
- The aside for the home page should include at least two links to other pages. The asides for the other pages should include at least a link back to the home page.
- Beyond these minimal specifications, you are free to present your content in whatever way you think works the best. For instance, you can have links, images, and content in both the sections and asides. Have some fun with it.

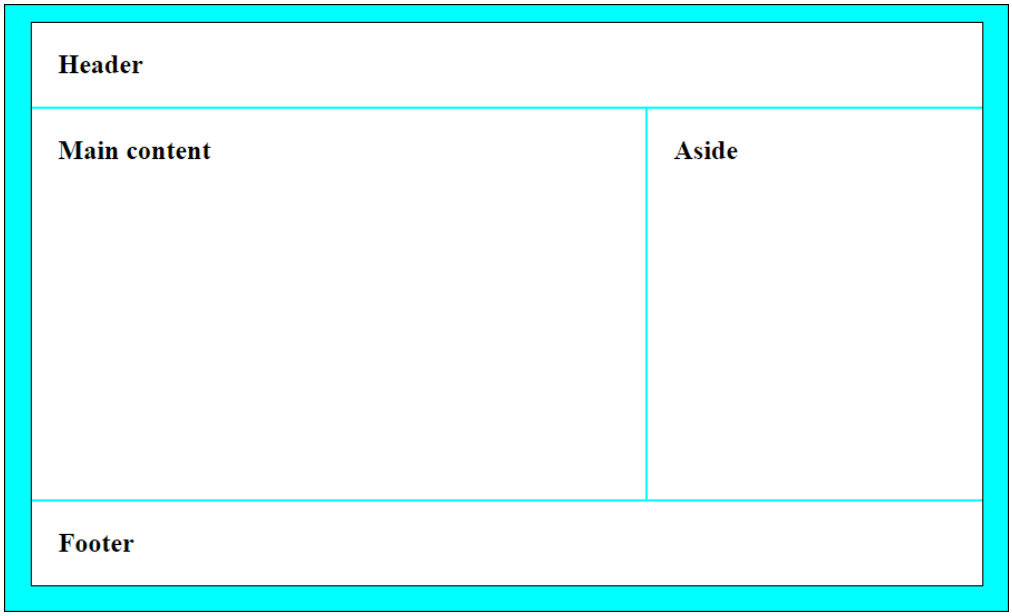
### CSS files

- You can use one style sheet named main.css for all of the pages of your site. You should also use the normalize.css style sheet for each page.
- As you enhance your website using the skills of chapters 9 through 17, you can create other style sheets if you think they're necessary.

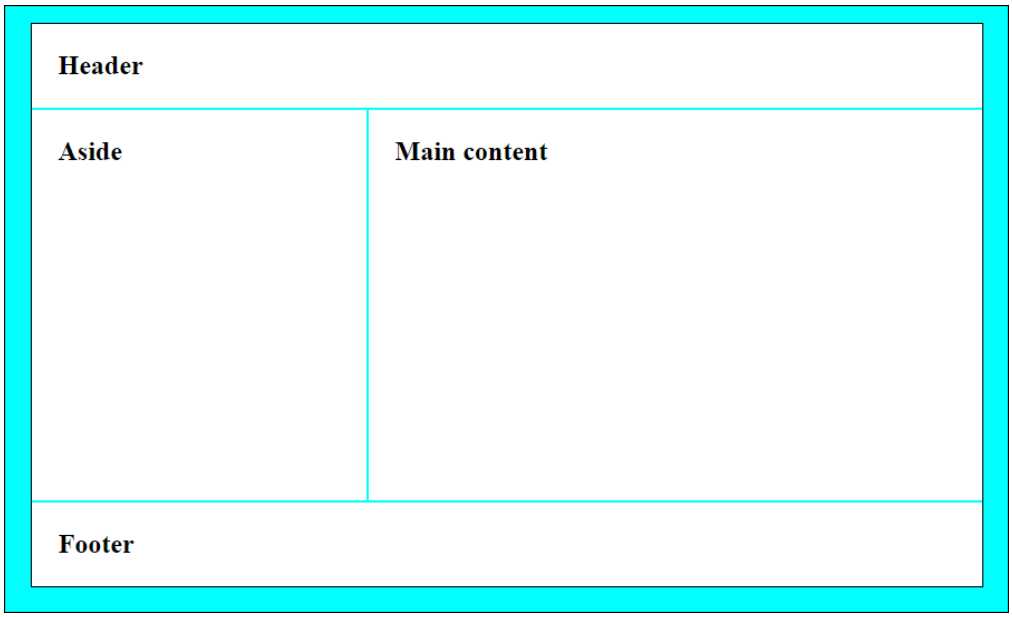
## Project 2: Two, 2-column layouts with header and footer

This project asks you to develop a website that consists of a home page and two or more other pages, all with a header, two columns, and a footer. Note, however, that the home page and the content pages have different page layouts.

**Sketch of the general layout of the home page**




**Sketch of the general layout of the content pages**



### Page layout

- Each page should be fixed width and centered in the browser window.
- The header for each page should include a website title and a tag line, but a logo image is optional.
- You don't have to use borders for any of the components of a page. The sketches above are just intended to show you what the components of the pages should be.

### Page contents

- The section for the home page should include an image and text.
- The home page should include an unordered or ordered list. This list can be in the section or the aside.
- The aside for the home page should include at least two links to other pages. The aside for the other pages should include at least a link back to the home page.
- Beyond these minimal specifications, you are free to present your content in whatever way you think works the best. For instance, you can have links, images, and content in both the sections and asides. Have some fun with it.

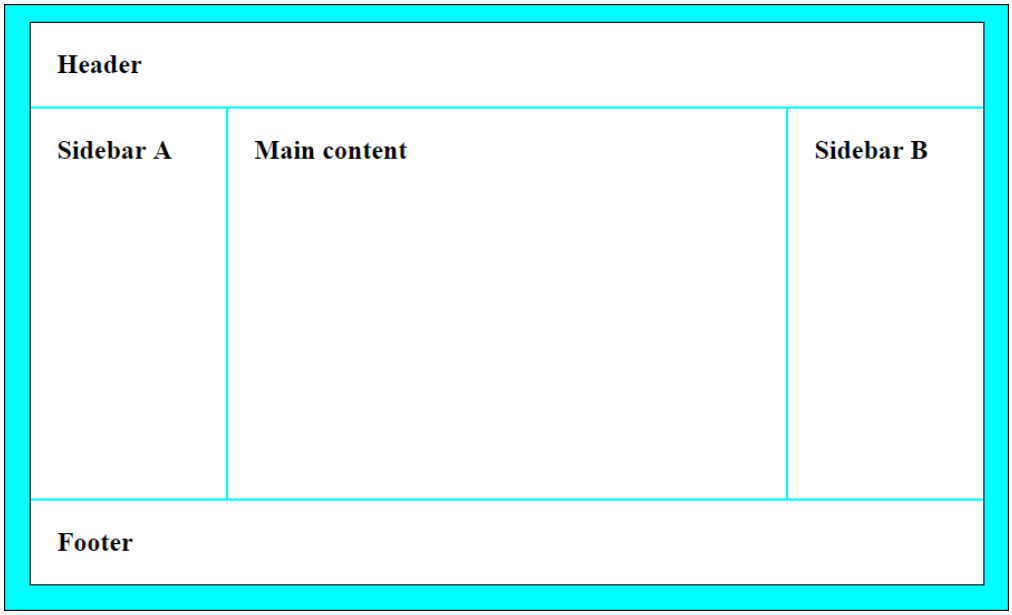
### CSS files

- You should use one style sheet named main.css for the home page and another style sheet named content.css for the other pages. You should also use the normalize.css style sheet for each page.
- As you enhance your website using the skills of chapters 9 through 17, you can create other style sheets if you think they're necessary.

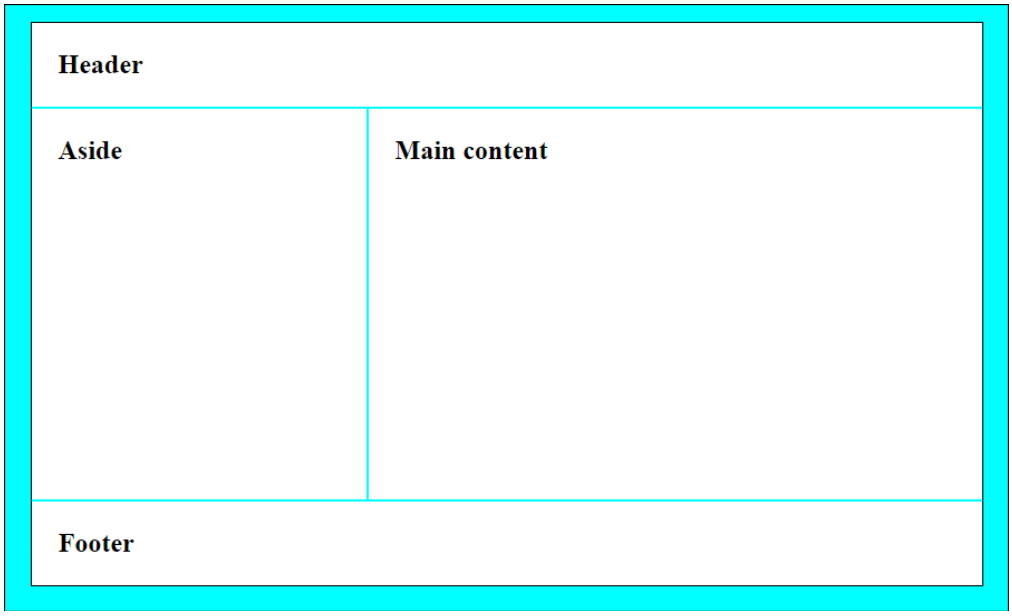
## Project 3: Three-column home page with 2-column content pages

This project asks you to develop a website that consists of a home page and two or more other pages. For this project, the home page has three columns, and the content pages have two columns.

### Sketch of the general layout of the home page




### Sketch of the general layout of the content pages



### Page layout

- Each page should be fixed width and centered in the browser window.
- The header for each page should include a website title and a tag line, but a logo image is optional.
- You don't have to use borders for any of the components of a page. The sketches above are just intended to show you what the components of the pages should be.

### Page contents

- The section for the home page should include an image and text.
- The home page should include an unordered or ordered list. This list can be in the section or either one of the asides.
- The left aside for the home page should include at least two links to other pages. The left aside for the other pages should include at least a link back to the home page.
- Beyond these minimal specifications, you are free to present your content in whatever way you think works the best. For instance, you can have links, images, and content in both the sections and asides. Have some fun with it.

### HTML and CSS files and directories

- You should use one style sheet named main.css for the home page and another style sheet named content.css for the other pages. You should also use the normalize.css style sheet for each page.
- As you enhance your website using the skills of chapters 9 through 17, you can create other style sheets if you think they're necessary.

## Sample project

The four pages that follow illustrate an acceptable implementation of the page layout described in project 2. They include features that were presented in sections 1, 2, and 3 of the book. You can use these pages to give you an idea of what you can do as you develop your own project.

### Home page

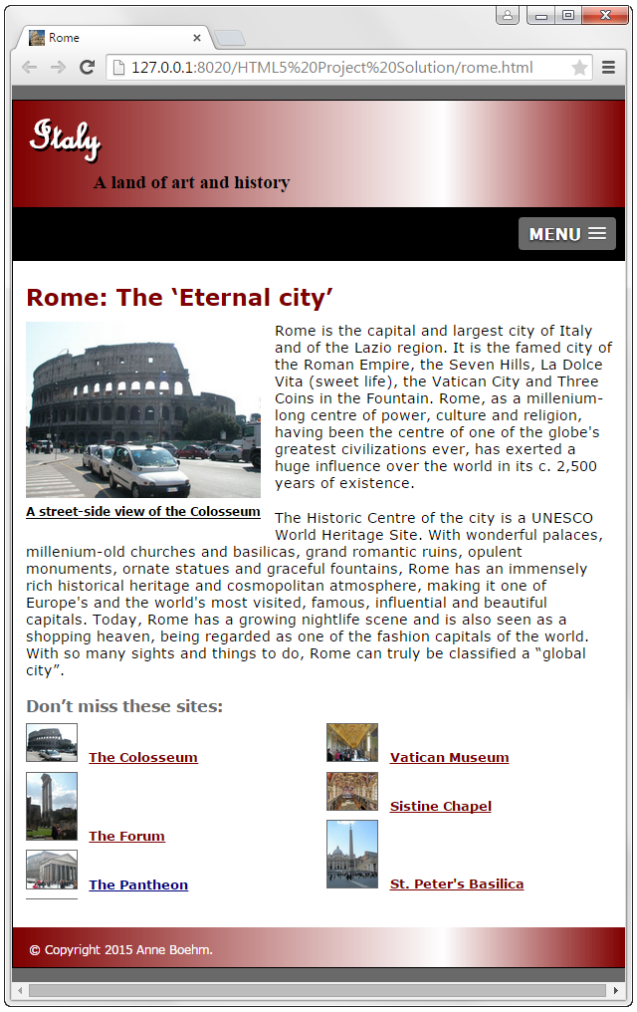

### Features

- Various heading levels and fonts including an embedded font
- 2-column layout with right sidebar
- Text with floated image in section
- List with links in sidebar
- Horizontal navigation menu
- Gradient in header and footer
- Character entity in footer

## Content page 1: Desktop and tablet landscape

## Content page 1: Tablet portrait

## Content page 1: Mobile landscape and portrait

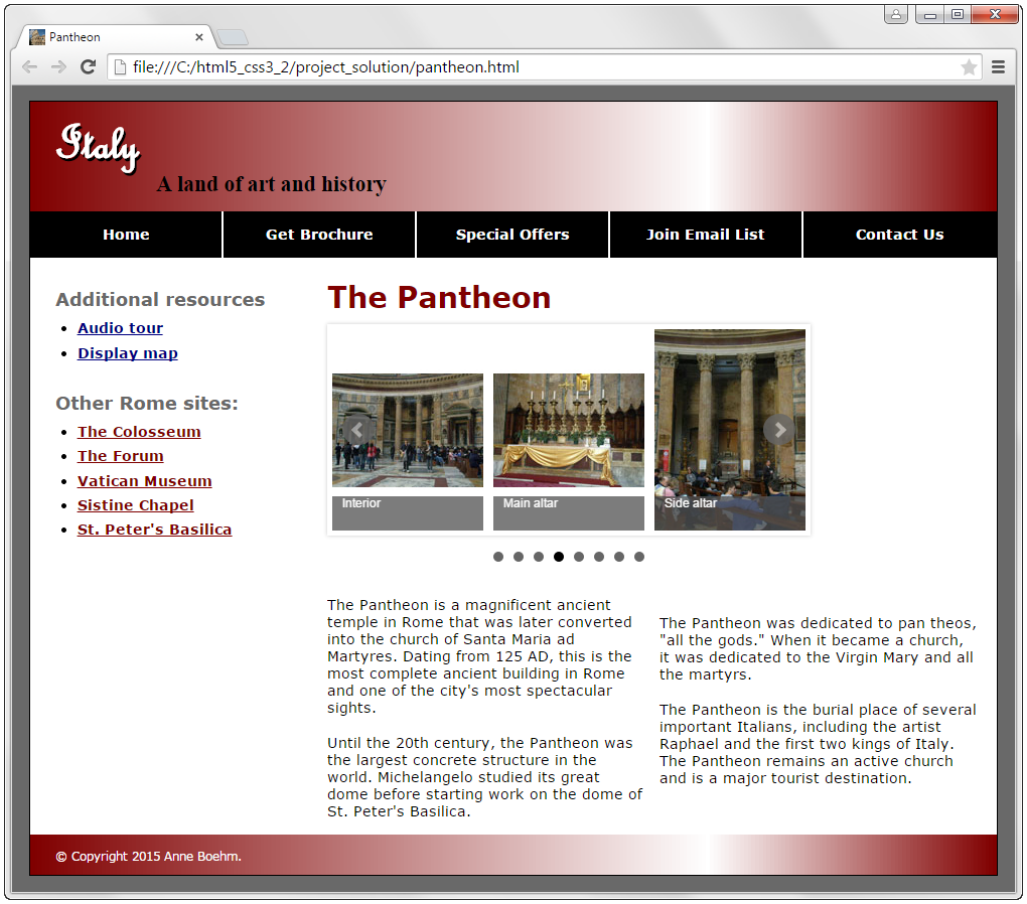




### Features

- 2-column layout with left sidebar (desktop and tablet only)
- List with image links in sidebar
- 2-column article in section (desktop and tablet only)
- Figure with image and caption
- Fluid layout
- Scalable image in article
- Media queries for screen media type

## Content page 2




## Features

- Carousel
- Link to audio file
- Link to PDF file