HTML5 (Exam 98-375 Certification)

Instructor Lesson Plans with Textbook Solutions and Code Fragments

Rev. 06/25/19

Table of Contents

Lesson 1: Managing the Application Life Cycle

Learning Objectives

Students will learn about:

- Platform fundamentals
- Application state management
- Touch interfaces and gestures
- Debugging and testing an HTML5-based touch-enabled application
- Publishing an application to an online store

ODN Skills

Understand the platform fundamentals.	1.1
Manage the state of an application.	1.2
Debug and test an HTML5-based touch-enabled application.	1.3
Publish an application to a store.	1.4

Lesson Summary — Lecture Notes

General Notes: Microsoft Internet Explorer 9 was used to show examples in Lesson 1. The authors used Visual Studio 11 Express Beta for Web to create most samples in the textbook. Microsoft Visual Studio 12 and Visual Studio 12 Express for Windows 8 were not available when the book was being written. Those tools are now needed to publish apps to the Windows Store.

Lesson 1 introduces students to platform fundamentals and the application life cycle. Begin by asking students for a show of hands regarding who has worked with HTML markup and who has created a Web page from scratch.

If some students have never created a Web page, you should start by briefly describing Hypertext Markup Language (HTML) and that HTML 4.01 is the current standard. Beginners sometimes confuse HTML with Hypertext Transport Protocol (HTTP), so explain the difference.

Then, explain that this course focuses on HTML5, the latest HTML draft standard. Several HTML5 features are already in use on many Web sites. HTML5 is also the name of a family of technologies that includes HTML, CSS, and JavaScript. CSS3 is the latest draft standard for CSS.

Important: Be sure students understand that "draft" means that the standards can change, and in fact are changing constantly. Students will learn about

Briefly cover some of the new HTML5 family markup tags and technologies, such as media queries and geolocation. Students will learn more about them and other features later in the course.

Explain that students can work with the HTML5 family using many different tools from many different companies. Even a simple text editor like Notepad or Notepad++ will work, although an HTML editor is more convenient because it colors tags so you can easily distinguish tags from content. Many HTML editors include intellisense, which provides closing tags when the user enters an opening tag, and so on. Students need more comprehensive tools, like Microsoft Visual Studio, when debugging a lot of code, packaging apps for distribution, and similar tasks. Free development tools for creating Metro style apps are available at http://bit.ly/K8nkk1.

Walk students through the general the steps involved in creating an app, from planning and designing through validation and deployment.

Briefly describe the concept of Metro style. (Per news articles available when this instructor guide was being written, the term “Metro style” might change in the near future. Microsoft might change it to Windows 8 UI or something similar.)

Discuss the Document Object Model (DOM), which is a topic that’s hard for some students to grasp. Explain that it’s designed for HTML and Extensible Markup Language (XML), and it allows programs and scripts to update content, structure, and styles on the fly. The DOM is neither HTML nor JavaScript—explain that it’s actually an application programming interface—but it ties them together. Provide a visual DOM example, if possible.

Students must understand the essentials of the host process, the app package, and the app container. Describe the Windows Runtime (WinRT) environment, the foundation of the Windows 8 operating system. Mention that it provides functionality to Metro-style apps.

In the next section, ensure students understand the concepts of a session state and an application state, and how they differ. Students must also understand `localStorage` and `sessionStorage`, JavaScript methods to deal efficiently with state data. In addition, explain that AppCache enables a user to load data ordinarily stored on a server even when the user is offline.

In the next section, introduce touch interfaces and the concept of gestures. Most students should be highly familiar with these topics! Explain that a finger move on a touch-screen device is called a gesture, and the response by the app to that gesture is called an event. Developing touch-enabled apps requires thorough knowledge of how fingers interact with the screen and planning for different sizes of fingers. Students can use JavaScript to create touch-enabled apps, primarily using the `touchstart`, `touchend`, and `touchmove` events. Remind students that they will learn more about touch and gestures in Lesson 10.

The next section briefly covers testing and debugging. Debugging an application involves detecting, finding, and correcting logical or syntactical errors. A syntax error is a typo in the code or a similar error, which is usually revealed during runtime for interpreted apps. A logic error results in the app behaving differently than expected.

Explain that a normal part of HTML document creation is validation. Many Web developers use the W3C’s Markup Validation Service Web page at http://validator.w3.org/ to validate markup. When a student needs to test or validate an app that’s going to be distributed, they can use the Windows App Certification Kit provided by Microsoft.

For the last section, Publishing an Application to a Store, review the general steps for publishing an app to the Windows Store. The store was not generally available at the time the book was written.

Note: To publish an app to the Windows Store, students must use Visual Studio 12 or Visual Studio 12 Express for Windows 8.

Key Terms

app container - An app container is a separate memory space within a system in which an application runs. An app container prevents corruption of the operating system if the application fails for some reason and enables a user to cleanly uninstall the app.

app package - An app package is a file that contains an app's files and folders. The purpose of an app package is for ease of distribution and deployment.

AppCache - The Application Cache, or AppCache, is an HTML5 feature that enables Web data to be stored locally when a user is offline. AppCache stores resources like images, HTML pages, CSS files, and JavaScript—data that would ordinarily be stored on a server. Because the resources are stored on the client's hard disk or device, the resources load faster when requested.

application programming interface (API) - An application programming interface (API) is a list of instructions letting a program communicate with another program.

application state - The application state is the phase of a running application at any point in time. For Web applications, the application state is created when the Web browser sends the first request for a Web page to the Web server, and it ends when the user closes the browser.

Cascading Style Sheets (CSS) - Cascading Style Sheets (CSS) is a style sheet language that defines styles for HTML. CSS styles are usually saved in a separate file from the HTML file. This enables you to easily change fonts, font sizes, and other attributes in the CSS file and the changes are reflected across all HTML files that reference the CSS file.

cookies - Cookies are small files that contain information about the user and the Web site visited and are saved on the user's computer.

debugging - Debugging is the process of detecting, finding, and correcting logical or syntactical errors in an application.

gesture - A gesture is any finger move, which can involve a single finger (one-touch, such as press, tap, press and hold, slide to pan, and so on) or a finger and a thumb (two-touch, such as a pinch and stretch or a turn to rotate).

Hypertext Markup Language (HTML) - Hypertext Markup Language (HTML) is the language used to describe Web pages. It is a markup language, not a programming language, which means HTML uses markup tags such as <body> and <h1> to describe parts of a Web page.

Hypertext Transport Protocol (HTTP) - Hypertext Transport Protocol (HTTP) is the protocol that transfers data on the World Wide Web.

HTML5 - HTML5 is the latest HTML standard and a family of technologies that includes HTML, CSS, and JavaScript. The HTML5 standard won't be finalized for a few years.

identity permissions - Identity permissions are sets of characteristics that identify an assembly. Identity permissions protect assemblies (compiled code libraries) based on evidence, or credentials, which is information about the assembly that an assembly must have in order to run.

JavaScript - JavaScript is a scripting language that adds interactivity to Web pages.

launcher icon - A launcher icon is a small image that represents an app.

localStorage - localStorage is a JavaScript method that allows users to save relatively large amounts of data from session to session (persistent data), and there's no time limit as to how long the data exists.

markup language - A markup language is a set of symbols and rules to describe the parts of a markup document, like an HTML Web page.

media queries - A media query is a CSS3 feature that detects the user's type of screen and sizes the output accordingly.

Metro-style user interface (UI) - A Metro-style user interface (UI) is the UI used in Microsoft Windows 8. The Metro style UI includes features like a clean, uncluttered look and feel, use of the full screen, large hubs (graphical buttons), and a focus on lateral scrolling.

namespace - A namespace is a sort of work area or abstract container for related objects (pages, code, etc.). A single app package can have a lot of functionality. To keep all of the components separated so they don't conflict, a package defines a namespace.

permission sets - A permission set is a group of permissions. In coding, transparent code executes commands that don't exceed the limitations of a permission set.

persistent state information - Persistent state information is data that an application needs after the session ends. Many Web applications need to store data (make it persistent) so that users can pick up where they left off when they return to the site.

platform-independent - The term platform-independent describes an application that can run on different desktop and mobile device operating systems, such as Microsoft Windows, Internet Explorer, Windows Phone, Mac OS X, Android, iOS, and Blackberry OS.

scripting language - A scripting language is a programming language that uses scripts and requires no compiler.

session state - The session state is an application's working set of data. When a user first requests access to an application, the session state is created. The state ends when the user closes the session.

sessionStorage - sessionStorage is a JavaScript method that keeps data only for one session (until the browser is closed), which is also referred to as "per-tab storage."

touch event - A touch event is the action an application takes in response to a gesture.

touch-screen emulator or simulator - A touch-screen emulator or simulator is an application that imitates a system that only has touch capabilities.

validator - A validator is an application that looks for anything that could cause code to be interpreted incorrectly, such as missing or unclosed tags, an improper DOCTYPE declaration, a trailing slash, deprecated code, and so on.

Windows Runtime (WinRT) - The Windows Runtime (WinRT) is an application architecture, or framework, that sits on top of the Windows 8 kernel. Developers test Windows applications and users run Windows 8 apps within WinRT.

Windows Store - The Windows Store is an online global marketplace for Metro-style apps. Publishing your app for distribution through the store can possibly turn a good idea into a lucrative venture.

World Wide Web Consortium (W3C) - The World Wide Web Consortium (W3C) is the main standards body developing specifications for HTML5, CSS3 and other Web technologies.

Lesson 1: Managing the Application Life Cycle- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. HTML is a __markup__ language, not a programming language, which means HTML uses markup tags such as <body> and <h1> to describe parts of a Web page.
2. __Cascading Style Sheets (CSS)_ defines styles for HTML in a separate file, so you can easily change fonts, font sizes, and other attributes.
3. Windows 8 users the _Metro style__ user interface (UI).
4. The _Windows Runtime (WinRT)_ is the foundation of the Windows 8 operating system, and is made up of layers that provide functionality to Metro style apps and the Windows shell.
5. __Application packaging___ is the process of bundling an application and its various files into an app container, making it easy to distribute and deploy the app. The app package is the result of this process.
6. The _application__ state is created when the Web browser sends the first request for a Web page to the Web server and it ends when the user closes the browser.
7. The _sessionStorage__ method keeps data only for one session (until the browser is closed), which is also referred to as "per-tab storage."
8. Any finger move is referred to as a _gesture__, which can involve a single finger (one-touch) or a finger and a thumb (two-touch.
9. A __validator__ looks for anything that could cause code to be interpreted incorrectly, such as missing or unclosed tags, an improper DOCTYPE declaration, a trailing slash, deprecated code, and so on.
10. The _Windows Store__ is an online global marketplace for Metro style apps.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which three components are the primary elements of the HTML5 family?
   a. XML
   b. HTML
   c. CSS
   d. JavaScript
2. JavaScript is a type of:
   a. Program compiler
   b. Markup language
   c. Scripting language
   d. Validator
3. All of the following are true of HTML5 except:
   a. It requires Windows 8

b. It can be used to create Web apps and PC and device apps

c. It is platform-independent

d. It is built on an open standard

4. Which operating system environment allows a developer to access a camera or webcam?

   a. localStorage

   b. WinRT

   c. the session state

   d. Metro

5. You are developing a Metro style app and want the app to access another app. Where do you declare the interaction?

   a. App manifest

   b. CSS

   c. At the top of the HTML file

   d. Nowhere; you do not have to declare the interaction

6. Which of the following is used to create an app package?

   a. JavaScript

   b. CSS

   c. DOM

   d. An app development tool

7. Which API allows programs and scripts to update content, structure, and styles on the fly?

   a. JavaScript

   b. WinRT

   c. The DOM

   d. RTE

8. AppCache, localStorage, and sessionStorage are forms of:

   a. Web storage

   b. HTML commands

   c. Standards

   d. Namespaces

9. Which of the following does *not* usually work well with multi-touch environments and should be disabled? (Choose two.)

   a. Tracking

   b. Zooming

   c. Scrolling

   d. Gesturing

10. Which tool is a type of validator that tests your app on your computer before you attempt to package and publish it to the Windows Store?

    a. WinRT

    b. Windows 8

c. W3C Markup Validation Service

[✓ d.] Windows App Certification Kit

True / False

Circle T if the statement is true or F if the statement is false.

[✓ T] F 1. An application programming interface (API) is a list of instructions letting a program communicate with another program.

T [✓ F] 2. A best practice is to publish your app without validation to perform live online testing.

T [✓ F] 3. An emulator searches HTML and CSS documents, looking for errors.

[✓ T] F 4. It’s a best practice to design touch-enabled apps for wider rather than narrower digits.

[✓ T] F 5. A platform-independent app can run on different desktop and mobile device operating systems.

Competency Assessment

Scenario 1-1: Understanding New Features in the HTML5 Family

Your manager, Marylyne, wants to learn about the HTML5 family to decide if the company should begin using it on new projects. She asks you to provide her a list of five or six new features. What items do you include in the list?

Some of the new features in HTML5 are:

- **Audio and video tags:** Embeds audio and video multimedia using the HTML5 markup tags <audio> and <video>.
- **Canvas:** An HTML5 element that creates a container for graphics, and uses JavaScript to draw the graphics as needed.
- **Media queries:** A CSS3 feature that detects the user’s type of screen and sizes the output accordingly.
- **New application programming interfaces (APIs):** Give apps access to a plethora of resources, such as files, webcams, and hardware-accelerated animations.
- **Geolocation:** Uses JavaScript to detect the location (geographic positioning) of a client device, whether it’s a Windows Phone, Android phone, or a PC.
- **Modernizr:** A JavaScript library that helps you deliver the new capabilities of HTML5 and CSS3 in older browsers.

Scenario 1-2: Creating an App

Marylyne approaches you again, this time wanting to know what is involved in creating an HTML5 app. She asks you to provide an outline. What steps do you include in the outline?

The general steps for creating an app are: plan the project, design a UI, update the app manifest, write code, build the app, debug and test the app, package the app, validate the app, and deploy the app.

Proficiency Assessment

Scenario 1-3: Sharing Touch-Enabled App Development Tips

Antoine is working on a touch-enabled app and asks you for development tips and items he should be sure to test on his tablet. What do you tell him?

When designing apps for a touch-screen environment, gesture responsiveness is key. Slow performance will frustrate most users. Incorporate physics effects such as acceleration and inertia to create a more fluid interaction between the user and screen.

Visual feedback for successful interactions and other notifications is highly important. This allows the user to understand whether he or she is using the touch landscape appropriately. Snap points help users stop at a location within the interface where intended, even if a gesture is a little off the mark.

You should also keep in mind that users have different size fingers, and it's a best practice to design for wider rather than narrower digits. And of course, users will be either right- or left-handed, so a well-designed app uses vertically symmetric navigation and provides for flipping the screen 90 degrees to go from portrait to landscape or vice versa.

When developing any touch-enabled app, be sure to test for the following:

Overall responsiveness and fluidity

Tapping, pinching, rotating, and other common gestures

Controlled scrolling

Controlled panning

Ability to disabled scrolling and panning

Accuracy of snap points

Unintended zooming or scrolling, especially in a multi-touch environment

Proper touch event reaction, especially in a multi-touch environment

Scenario 1-4: Publishing an App to the Windows Store

Sammy created his first app and wants to publish it to the Windows Store. What are three preparatory steps he should take?

Sammy should follow these steps; the student may list three of them:

1. Sign up and pay for a Windows Store developer account, reserve a name for the app, and edit the app's manifest file.
2. Go through the app submission checklist at http://bit.ly/HAPmbk. The checklist includes tasks such as naming apps, choosing selling details such as selecting appropriate pricing and a release date, assigning an age rating, describing an app, and more.
3. Use the Windows App Certification Kit to test his app, if he hasn't done so already.
4. Capture some screen shots of significant or unique features of his app to showcase in the store.
5. Have other testers or developers test his app on as many different devices and platforms as possible, especially if Sammy tested the app only in a simulator or emulator.

6. Include a privacy statement if his app gathers personal information or uses copyrighted software to run.

Lesson 1: Managing the Application Life – Code Fragment

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

AppCache for Offline Files

Using AppCache, a developer uses a text file called a “cache manifest” to specify the files a Web browser should cache offline. Even if a user presses the Refresh button offline, the app will load and work correctly. A cache manifest file looks similar to the following:

```
index.html

stylesheet.css

images/dot.png

scripts/main.js
```

Lesson 2: Building the User Interface by Using HTML5: Text, Graphics, and Media

Learning Objectives

Students will learn about:

- HTML5 essentials
- Displaying text content
- Displaying graphics
- Playing audio and video files

ODN Skills

Choose and configure HTML5 tags to display text content	2.1
Choose and configure HTML5 tags to display graphics	2.2
Choose and configure HTML5 tags to play media	2.3

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to test and show all of the examples in Lesson 2. It's suggested that you provide examples of the `audio` and `video` elements at the end of this lesson, so you should find suitable audio and video files prior to class.

Lessons 2 and 3 focus on building HTML5 user app interfaces. Lesson 2 introduces students to incorporating text, graphics, and media, whereas Lesson 3 covers organization, input, and validation.

Lesson 2 begins with a lengthy section on basic HTML markup and terms, along with simple Web page creation. It's important that students who haven't created Web pages in the past thoroughly understand all of the basics. The course provides the basics as a review and quickly builds upon them.

Start by briefly discussing HTML documents, and that a Web page is a kind of HTML document. Cover tag pairs and empty tags, and provide examples. Mention that every Web page (not necessarily every HTML document) requires the `<html>`, `<head>`, `<title>`, and `<body>` tags. Explain that a tag pair or an empty tag is also called an element, which can describe content, insert graphics, and create hyperlinks. In an HTML document, a well-formed element is one that is opened and then closed, or an empty element, which must be terminated and a slash.

Discuss attributes and global attributes. Show an example of nesting and describe why it's important. Proper nesting is also part of a well-formed document.

Explain that entities are special characters on a Web page and most often require character encoding to be displayed properly in current and legacy browsers. Point out that students should use UTF-8 encoding in HTML5 documents because most browsers use UTF-8. That means they should add the following declaration to the head element:

<meta charset="UTF-8">

That brings you to the doctype declaration. Discuss this declaration and its purpose. The new HTML5 doctype is very simple compared to HTML 4: `<!doctype html>`. Be sure to show the students some markup examples of simple Web pages.

In the next section, explain that HTML5 uses most of the same elements and attributes specified in HTML 4, and has introduced some new tags, modified the preferred usage of others, and no longer supports certain elements. New text-related elements include `command`, `mark`, `time`, `meter`, and `progress`. Give an example of at least one of the new elements.

Explain that some HTML 4 text-related elements now have slightly different meaning or functionality in HTML5. The elements include `<b>`, `<i>`, `<strong>`, `<em>`, and `<small>`. The `<b>` element should now be used to offset text without conveying importance, such as for keywords or product names. The `<i>` element now indicates content in an alternate voice or mood, like spoken text. The `<strong>` element indicates strong importance, whereas the `<em>` element indicates emphatic stress. The `<small>` element should be used for small print, like a copyright line. The intended functionality for some of these elements in HTML5 can be confusing, such as knowing when to use the italic element. The best approach is to strive for consistency within a page or Web site, and watch how other developers use the same elements as HTML5 use evolves.

Explain deprecation but emphasize that some deprecated elements are still widely used in Web pages. A few of the deprecated elements are `basefont`, `center`, `font`, and `strike`.

The next section covers graphics. Begin by explaining the difference between raster and vector images. Then explain use of the `img` element to display linked images in a Web page, and that the images can be located with the Web pages HTML files, usually in an images subfolder, or on a different server or Web site.

The `figure` and `figcaption` elements are new to HTML5 and offer more control over the type of image being displaying and the ability to include captions. Show how these work.

Students with a penchant for graphics creation might really like the canvas and SVG sections. Explain that the `canvas` element is used for drawing, rendering, and manipulating images and graphics dynamically in HTML5. Scalable Vector Graphics (SVG) enables developers to create scalable objects that resize to best fit the screen on which they're viewed, whether a PC screen or a smartphone. Students will see both canvas and SVG again in the course.

Be sure to address fallback content—alternate images or text for canvas drawings. Explain that some older browsers cannot render canvas drawings or animation, so students should reference an image, include a text note, or include some other HTML content within the canvas element that will display if the drawing cannot. The "backup" content, also referred to as fallback content, won't display if canvas is supported.

The last section covers the HTML5 `audio` and `video` elements. The elements don't require plug-ins or media players to enable users to listen to music or watch videos via a Web browser. Be sure to show an example of each, emphasizing how easy they are to add to HTML documents.

Key Terms

attribute - An attribute is a modifier of HTML elements that provides additional information.

`audio` element - The `audio` element enables you to incorporate audio, such as music and other sounds, in HTML documents.

`canvas` element - The `canvas` element is new in HTML5 and creates a container for graphics, and uses JavaScript to draw the graphics dynamically.

codec - A codec is a technology used for compressing data.

compression - Compression reduces the amount of space needed to store a file, and it reduces the bandwidth needed to transmit the file.

deprecation - Deprecation is the state of an element or attribute that's been removed from the list of available HTML elements because its functionality is no longer useful.

`doctype` - The `doctype` is an instruction, or declaration, found at the very top of almost every HTML document that associates the document with a Document Type Definition (DTD). When a Web browser reads a `doctype` declaration, the browser assumes that everything on the Web page uses the language or rules specified in the declaration.

element - An element is the combination of tags and the content they enclose. An element can describe content, insert graphics, and create hyperlinks. A tag pair or an empty tag is called an element.

empty tag - An empty tag is a single tag that doesn't require an end tag, like `<br />` for a line break and `<hr />` for a horizontal line.

entity - An entity is a special character, such as the dollar symbol, the registered trademark (a capital R within a circle), and accented letters.

`figcaption` element - The `figcaption` element adds a caption to an image on a Web page, and you can display the caption before or after the image.

`figure` element - The `figure` element specifies the type of figure you're adding, such as an image, diagram, photo, and so on. This element provides a major benefit: the ability to easily add multiple images side by side.

global attribute - A global attribute is one you can use with any HTML5 element. Examples of global attributes include `id`, `lang`, and `class`, among many others.

nesting - Nesting means to place one element inside another.

raster image - A raster image is an image made up of pixels, such as a photograph. Raster images are most often in JPG format. Other raster file formats that work well on Web pages are PNG, GIF, and BMP.

render - Render means to interpret or reproduce. When a Web browser or mobile device such as a smartphone opens an HTML file, it renders the content of the page.

Scalable Vector Graphics (SVG) - Scalable Vector Graphics (SVG) is a language for describing 2D graphics in Extensible Markup Language (XML). SVG technology is not new, but HTML5 now enables SVG objects to be embedded in Web pages without using the `<object>` or `<embed>` tags.

tags - Tags are keywords that help to give an HTML page structure.

valid - Valid means logically correct. If a Web page adheres to the specifications perfectly, it is considered valid.

vector image - A vector image is an image made up of lines and curves based on mathematical expressions. A vector image is an illustration, such as a line drawing. Developers often convert vector file formats from programs like Adobe Illustrator or CorelDRAW, which aren't supported by Web browsers, into PNG or GIF for Web display.

video compression - Video compression reduces the size of video images while retaining the highest quality video with the minimum bit rate.

`video` element - The `video` element enables you to incorporate videos in HTML documents using minimal code.

Lesson 2: Building the User Interface by HTML5: Text, Graphics, and Media Life Cycle- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. An HTML tag that doesn't require an end tag is called a(n) __empty__ tag.
2. A(n) __attribute__ works with an element to describe data in enough detail for rendering.
3. The __`doctype`__ is a declaration that is found at the very top of almost every Web page.
4. A __deprecated__ element or attribute has been removed from the list of available HTML elements according to the W3C.
5. A __raster__ image is made up of pixels, whereas a __vector__ image is made up of lines and curves based on mathematical expressions.
6. New to HTML5, the __`figure`__ element specifies the type of figure you're adding, such as an image, diagram, photo, and so on.
7. The __`figcaption`__ element adds a caption to an image on a Web page, and you can display the caption before or after the image.
8. Using the __`canvas`__ element, the Web page becomes a drawing pad, and you use JavaScript commands to draw pixel-based shapes on a canvas that include color, gradients, and pattern fills.
9. __SVG__ is a language for describing 2D graphics in Extensible Markup Language (XML).
10. The HTML5 __`audio`__ element and __`video`__ element enable you to provide multimedia from a Web browser without the need for plug-ins.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which of the following tags are required on every Web page? (Choose all that apply.)
   a. <html>
   b. <head>
   c. <title>
   d. <body>
2. Which of the following is the syntax for creating a hyperlink in HTML?
   a. `<link href="http://www.example.com">link</a>`
   b. <a href="http://www.example.com"> link text</a>
   c. `<link>http://www.example.com</link >`
   d. `<http://www.example.com>`

3. Which HTML5 element defines a command button that users click to invoke a command?
    a. `<objectbut>`
    b. `<combutton>`
    c. <command>
    d. `<cbutton>`
4. Which HTML5 element enables you to highlight blocks of text in an HTML document?
    a. <mark>
    b. `<highlight>`
    c. `<emphasis>`
    d. `<yellow>`
5. Which of the following tags are deprecated in HTML5? (Choose all that apply.)
    a. <big>
    b. <center>
    c. <font>
    d. `<time>`
6. Which tag is used with the `<figure>` tag to display an image?
    a. <img>
    b. `<src>`
    c. `<fig>`
    d. `<a>`
7. Both canvas and SVG require which of the following?
    a. Microsoft Silverlight
    b. An external drawing program, such as Microsoft Paint
    c. A large amount of storage space or bandwidth
    d. JavaScript
8. When deciding whether to use canvas or SVG, which of the following considerations are true?
    a. If the drawing is relatively small, use SVG.
    b. Generally, use canvas for small screens and SVG for larger screens.
    c. If the drawing requires a large number of objects, use SVG.
    d. If you must create highly detailed vector documents that must scale well, go with canvas.
9. Which of the following is the general format of the `video` element?
    a. `<movie src="file.mp4" width="X" height="Y">`
    b. `<movie href="file.mp4" width="X" height="Y">`
    c. <video src="file.mp4" width="X" height="Y">
    d. `<video href="file.mp4" width="X" height="Y">`
10. Which of the following is the general format of the `audio` element?

[✓ a.] <audio src="sample.mp3" controls="controls">

b. `<audio href="sample.mp3" controls>`

c. `<sound src="sample.mp3" controls>`

d. `<sound href="sample.mp3" controls="controls">`

True / False

Circle T if the statement is true or F if the statement is false.

[✓ T] F 1. The `canvas` element requires JavaScript to create shapes.

T [✓ F] 2. Creating an SVG object in HTML5 does *not* require JavaScript.

[✓ T] F 3. The `audio` element can provide playback controls with a single attribute.

T [✓ F] 4. Deprecated elements cannot render in an HTML5-supported browser.

T [✓ F] 5. The most popular format for audio files is MP4.

Competency Assessment

Scenario 2-1: Correcting Simple Markup Errors

Geraldine, the assistant to the company owner, is learning HTML. Her markup as shown below isn't rendering as she expected. The boldface doesn't stop after "Thursday." The image of the company logo doesn't display, even though it's saved in her images subfolder like all of her other images. The alternate text doesn't display either when she hovers her mouse pointer over the image placeholder. What do you tell her?

```
<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>Internal</title>

</head>

<body>

<h1>Staff Meeting</h1>

<img src="cologo.jpg" olt="Company logo" />

<p>Report to the <strong>Blue Conference Room</strong> at
<strong>10:00 a.m.</strong> on <strong>Thursday<strong> for
an emergency staff meeting.</p>

</body>

</html>
```

Geraldine needs to insert "images/" just before "cologo.jpg" so the browser can find her JPG file. Tell her that unless her JPG file is sized appropriately, she should add width and height attributes to the `img` element. She also needs to correct the typo "olt" by changing it to "alt" so the alternate text will appear. Geraldine is missing a slash in the last `<strong>` tag, which is why the boldface continued after "Thursday." Suggest to Geraldine that she use an HTML editor or application development tool, or the W3C Markup Validation Service at http://validator.w3.org, to help her find errors in her markup.

Scenario 2-2: Working with Symbols

Petra is formatting some accounting-related documents to be hosted on the company's intranet. She says the dollar signs and percent symbols look fine when she views them in one browser, but only garbage characters appear when she views the documents on a different browser. What should she do?

Tell Petra to use the &dollar; or &#36; syntax to render the dollar symbol. She should use the &percnt; or &#37; syntax to render the percent symbol.

Proficiency Assessment

Scenario 2-3: Canvas or SVG?

M.A. is a graphic artist at ClickTick Watches, an upscale wristwatch manufacturer. She has been asked to refresh the company logo and create it using a tool that scales well whether the image is viewed on laptops or smartphones. She has also been tasked with creating interactive graphs for sales staff to use on their slate or tablet devices. She wants to keep her skillset current by learning as much as possible about HTML5 technologies, but doesn't know whether to focus on canvas or SVG for these projects. What do you suggest?

Tell M.A. that she should learn both canvas and SVG. SVG is the better tool to use for the company logo because SVG is vector-based and highly scalable. She should consider canvas for the interactive graphs because they will be viewed mainly on smaller screens, and canvas is ideal for displaying real-time data output.

Scenario 2-4: Selecting Appropriate Web Video Formats and Codecs

Sammy is responsible for setting up meetings for employees of Clear Blue Resorts. He wants to post a video from the CEO, who is overseas reviewing possible locations for new resorts, to the intranet for the upcoming employee appreciate party. He knows Clear Blue standardized on Internet Explorer 9, and he has heard that he can easily display video in HTML5 but doesn't know where to start. What do you tell Sammy?

Tell Sammy that because Clear Blue has standardized on Internet Explorer 9, he can create a simple Web page and use the HTML5 `<video>` element.

Sammy needs to find out the video format of the video file, and then determine the best codec. The format should ideally be MP4 or H.264, OGG, or WebM, with Theora and Vorbis codecs for OGG, and the VP8 codec for WebM.

The markup would look similar to the following:

```
<video src="/videos/intro.mp4"
    width="400" height="300
    poster="image.jpg"
    autoplay="autoplay"
    controls="controls">
</video>
```

Sammy can choose to include the `poster` attribute or not, as well as `autoplay`.

Lesson 2: Building the User Interface by Using HTML5: Text, Graphics, and Media – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Understanding the Doctype

In HTML 4, the doctype declaration specifies the HTML page's language and DTD, and looks quite complex. Here's an example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.example.com/TR/xhtml11/DTD/
  xhtml11.dtd">
```

The new HTML5 doctype, in comparison, is very simple:

```
<!doctype html>
```

Exploring the Markup of a Simple Web Page

An example of markup and content for a simple HTML5 Web page looks like this:

```
<!doctype html>


<html>
  <head>
    <title>78704 Pet Services</title>
  </head>

<body>
  <p>Your dog is a friend for life. Why not
  provide the best care possible?</p>
</body>

</html>
```

The blank lines between parts of the page, such as between the doctype

Create a Simple Web Page

GET READY. To create a simple Web page and see the effect of missing tags, nesting, and entities, perform the following steps:

2. Open a Web page editor, app development tool, or even a simple text editor like Notepad and type the following:

```
<!doctype html>
<html>
  <head>
    <title>78704 Pet Services</title>
  </head>
<body>
```

```
<h1>Care and Feeding</h1>
  <p>Your dog is a friend for life. Why not provide the
   best care possible?</p>
  <p>Make sure your pet has plenty of <i><b>fresh
   water</b></i> during hot weather. When taking your dog
   on long walks, bring along a collapsible water dish and
   bottled water. You can find specialty water dishes at
   many pet supply stores for $10 or less.</p>
</body>
</html>
```

New Text Elements in HTML5

Let's take a look at some of the new text elements in HTML5 along with some examples:

- **`<command>`:** The command element is used to define a command button that users click to invoke a command. The command element has many attributes you can use, such as type, label, title, icon, disabled, checked, and radiogroup. For example:

```
<menu label="Music Genre">

   <command type="radio" radiogroup="musicgenre"
label="Art">

   <command type="radio" radiogroup="musicgenre"
label="Popular">

   <command type="radio" radiogroup="musicgenre"
label="Traditional">

</menu>
```

- **`<mark>`:** The mark element is very handy for highlighting text on a page. You could use it on a search results page, for example, or to set off a block of text that you want to draw to the reader's attention. For example:

```
<p>Since I started jogging last fall, I have <mark
style="background-color:yellow;">lost 35 pounds</mark>.</p>
```

- **`<time>`:** The time element indicates content that is a time or date, which can be made machine-readable with the `datetime` attribute. The time element defines time on a 24-hour clock and a date in the Gregorian calendar. One benefit of making times and dates machine-readable on your Web page is that it helps search engines produce better search results. For example:

`<time datetime="2013">` means the year 2013

`<time datetime="2013-04">` means April 2013

`<time datetime="04-15">` means 15 April (any year)

Using the figure and figcaption Elements

The following markup uses the figure element, specifies the width and height of the image, and adds a caption. The result is shown in Figure 2-10:

```
<figure>

  <img src="doghappy.jpg" alt="Happy dog"
  width="100" height="125" />
  <figcaption>Happy dogs are good dogs</figcaption>

</figure>
```

The following markup is for a figure with multiple images that share a single caption, the results of which are shown in Figure 2-11:

```
<figure>

  <img src="doghappy.jpg" alt="Happy dog"
  width="100" height="125" />

  <img src="dogpaws.jpg" alt="Happy dog"
  width="100" height="125" />

  <img src="dogwalk.jpg" alt="Happy dog"
  width="100" height="125" />

  <figcaption>Happy dogs are good dogs</figcaption>

</figure>
```

Use the figure and figcaption Elements

GET READY. To display an image in a Web page, perform the following steps:

3. Replace the markup for the figure that follows the h1 element with the following, replacing the image file names (doghappy.jpg, dogpaws.jpg, and dogwalk.jpg) with your image file names:

```
<figure>

  <img src="doghappy.jpg" alt="Happy dog"
  width="100" height="125" />

  <img src="dogpaws.jpg" alt="Dog paws"
  width="100" height="125" />

  <img src="dogwalk.jpg" alt="Walking a dog"
  width="100" height="125" />

  <figcaption>Happy dogs are good dogs</figcaption>

</figure>
```

Use the Canvas to Create a Shape

GET READY. To use the canvas element to create a shape, perform the following steps:

1. In your editing tool, type the following markup:

```
<!doctype html>

<html>

  <head>

    <meta charset="UTF-8">

    <title>Canvas Test</title>

<script>
```

```
   function f1() {
       var canvas =
       document.getElementById("smlRectangle");
       context = canvas.getContext("2d");
       context.fillStyle = "rgb(0,0,255)";
       context.fillRect(10, 20, 200, 100);
   }
  </script>
  </head>
<body onload = "f1();">
<canvas id="smlRectangle" height="100" width="200 ">
</canvas>
</body>
</html>
```

Providing an Alternate Image or Text for Older Browsers

Some older browsers cannot render canvas drawings or animation. Therefore, you should add an image, text, or some other HTML content within the canvas element that will display if the drawing cannot. The “backup” content, also referred to as fallback content, won’t display if canvas is supported. This example displays an image (smlRectangle.jpg) similar to a filled rectangle canvas would create:

```
<canvas id="smlRectangle" height="100" width="200">
  <img src="http://www.example.com/images/smlRectangle.jpg"
alt="A blue rectangle" />
</canvas>
```

To display text instead of an image, you would insert text in place of the <img> tag.

Creating Graphics with SVG

You can include attributes such as color, rotation, stroke color and size, and so on, to each SVG object. The following markup can be included in an HTML file to create a purple ball:

```
<svg id="svgpurpball" height="200"
  xmlns="http://www.w3.org/2000/svg">
  <circle id="purpball" cx="40" cy="40"
  r="40" fill="purple" />
</svg>
```

Create an SVG Vector Graphic

GET READY. To create a simple SVG vector graphic, perform the following steps:

1. In your editing tool, type the following markup:

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>SVG Star</title>
  </head>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polygon points="100,10 40,180 190,60 10,60 160,180"
   style="fill:aqua;stroke:orange;stroke-width:5;fill-
   rule:evenodd;"/>
</svg>
</body>
</html>
```

Understanding and Using Video Tags

The ***`video`* element** enables you to incorporate videos in HTML documents using minimal code. The structure for embedding video is simple. The following is an example of the markup for adding an MP4 file to a Web page:

```
<video src="intro.mp4" width="400" height="300">

</video>
```

The `src` attribute points to the name of the video file (in this case, video.mp4) to be played. The height and width attributes specify the size of window in which the video will display.

Other attributes are available that you can add for control of the video:

- **`poster:`** Displays a static image file before the video loads
- **`autoplay:`** Start playing the video automatically upon page load
- **`controls:`** Displays a set of controls for playing, pausing, and stopping the video, and controlling the volume
- **`loop:`** Repeats the video

Using all of the controls listed above, the markup would look similar to this:

```
<video src="/videos/intro.mp4"
    width="400" height="300
    poster="78704-splash.jpg"
    autoplay="autoplay"
    controls="controls"
    loop="loop">

</video>
```

Notice that this markup refers to an MP4 video file. Other popular Web video formats also include H.264, OGG, and WebM, although WebM is used less than 10 percent of the time. Along with a video format, you should also specify the ***codec,*** which is a technology used for compressing data. ***Compression*** reduces the amount of space needed to store a file, and it reduces the bandwidth needed to transmit the file. ***Video compression*** reduces the size of video images while retaining the highest quality video with the minimum bit rate. All of this makes for better performance.

In a nutshell, the main video formats along with codecs (for the last two) are:

- MP4 or H.264
- OGG + Theora with Vorbis audio
- WebM + VP8

A best practice is to use the type attribute to specify the video format. You should also use the codecs attribute to specify the codec(s), if applicable. Sample markup is shown as follows:

```
<video
  width="400" height="300"
  poster="78704-splash.jpg"
  autoplay="autoplay"
  controls="controls"
  loop="loop">

  <source src="intro.mp4" type="video/mp4" />

  </video>
```

The `<source>` tag is being used as content of the video element so that the `type` attribute can be set and so that the multiple format option is available.

Not all video formats are supported by all browsers, although MP4/H.264 is the most widely used by both Web browsers and mobile devices. (The HTML5 Video Web page at http://www.w3schools.com/html5/html5_video.asp displays a table showing which video formats work for what browser. The table is updated regularly.) To help make your video viewable by the majority of browsers and devices, you can use the source attribute to include multiple formats in your markup. This example shows the same video available in two formats, and the OGG format specifies codecs:

```
<video
  width="400" height="300" poster="image.png"
  autoplay="autoplay"
  controls="controls"
  loop="loop">

  <source src="video.mp4" type="video/mp4">

  <source src="video.ogg" type='video/ogg;
    codecs="theora, vorbis"'>

</video>
```

Work with the video Element

GET READY. To work with the HTML5 video element, perform the following steps:

1. In your editing tool, create an HTML file with the following markup. Substitute appropriate file names for your image file and video clip. Change

the type attribute, if necessary, and replace sample.mp4 with the name of your video file.

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Video Test</title>
  </head>
<body>
  <video
    width="400" height="300"
    poster="sample.jpg"
    autoplay="autoplay"
    controls="controls">
    <source src="sample.mp4" type="video/mp4" />
  </video>
</body>
</html>
```

Lesson 3: Building the User Interface by Using HTML5: Organization, Input, and Validation

Learning Objectives

Students will learn about:

- Organizing content
- Creating forms
- Restricting input
- Validating markup

ODN Skills

Choose and configure HTML5 tags to organize content and forms.	2.4
Choose and configure HTML5 tags for input and validation.	2.5

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used for most examples in Lesson 3. Mozilla Firefox was used to show placeholders in the Create a Simple Web Form exercise and validation error messages in the "Understanding Validation" section. Internet Explorer 9 does not support the `placeholder` or `required` attributes of the `input` element; you could use Internet Explorer 10 to demonstrate these attributes. If an element or attribute doesn't display in Internet Explorer 9, you can use the When Can I Use Web page at caniuse.com, which lists many HTML5 elements and attributes and indicates which Web browsers support them.

Lesson 3 introduces organization, input, and validation elements and techniques for building HTML5 user app interfaces.

Begin by explaining that, for years, developers have used the `<div>` tag along with a class or ID attribute to provide structure to HTML documents. Explain how to use `<div>` and show an example.

Introduce some of the new HTML5 elements for structuring and organizing content in an HTML document, such as `header`, `footer`, `section`, `nav`, `article`, and `aside`. Explain that the new tags are considered "semantic" markup because the tag names are intuitive, making it easier to build and modify HTML documents, and for Web browsers and other programs to interpret. Developers can still use `<div>` in HTML5 documents but should use the new structure elements whenever appropriate.

Discuss the `header` and `footer` elements in detail and show an example of each. The `header` element defines a header for a document, section, or article. HTML 4.01 uses the header div (<div id="header">). The `footer` element defines a footer for a document or section, and typically contains information about the document or section, such as the author name, copyright data, links to related documents, and so on. Emphasize that the `footer` element doesn't automatically appear at the bottom (or foot) of the document—you need to use CSS to instruct the browser where to display the footer.

Discuss the `section` element and show an example. The `section` element defines a section in a document, such as a chapter, parts of a thesis, or parts of a Web page whose content is distinct from each other. According to the W3C, the `section` element must contain at least one heading and define something that would ordinarily appear in the document's outline.

Next, discuss the `nav` element and show an example. The `nav` element defines a block of navigation links and is useful for creating a set of navigation links as a document's primary navigation, a table of contents, breadcrumbs in a footer, or Previous-Home-Next links.

Discuss the `article` element and provide an example. The `article` element defines a part of an HTML document that consists of a "self-contained composition" that is independent from the rest of the content in the document. Content set off by `<article>` tags can be distributed in syndication, so think of it as content that makes sense on its own.

Finally, discuss the `aside` element and provide an example. This element is used for sidebars and notes—content that's related to the current topic but would interrupt the flow of the document if left inline.

The next section covers HTML tables. Explain that basic structure of HTML tables: the `<table>` tag, the `<tr>` tag for rows, the `<th>` tag for column headers, and the `<td>` tag for cells. The `<caption>` tag adds a caption above or below the table. To apply inline styles using HTML rather than CSS, developers use the `<col>` tag to apply styles to an entire column. The `<colgroup>` tag groups columns within a table so you can apply formatting to the group rather than just a column.

When creating a long table that requires scrolling within a browser, use the `<thead>`, `<tfoot>`, and `<tbody>` tags.

Be sure to briefly explain how to apply background color and other styles to cells, and mention that the CSS chapters will cover styles in more detail.

Next, discuss ordered and unordered lists, and the various symbols you can use for unordered lists. You can mention the HTML5 `menu` element, which presents a list (or menu) of commands, usually with buttons, if the element is supported by Internet Explorer 9 or 10 at the time of course delivery.

The last major section addresses input and validation. Explain that developers use Web forms as the interface for collecting information from Web site and client application users. Ask students if they've used Web forms and for what types of input.

Web forms have been around for a long time and still use the same main element: `input`. The input element uses many attributes, which are listed in Table 3-5. Point out and briefly describe the new HTML5 `input` attributes. Because Web forms can be a bit challenging, show students how to construct a simple Web form, perhaps with the `pattern` attribute.

Finally, explain that validation ensures that information entered into a Web form is in the correct format and usable before sending the data to the server. Explain the concepts of automatic validation of input, client-side validation, and server-side validation. If possible, show how the `required` and `pattern` attributes of the `input` element work when entering text into an input field.

Key Terms

`article` element - The `article` element defines a part of an HTML document that consists of a “self-contained composition” that is independent from the rest of the content in the document.

`aside` **element** - The `aside` element is used to set off content that’s related to the current topic but would interrupt the flow of the document if left inline.

`autofocus` attribute - The `autofocus` attribute moves the focus to a particular input field when a Web page loads. An example of autofocus is when you open a search engine Web page and the insertion point automatically appears in the input box so you can type search terms without first clicking in the box.

automatic validation - Automatic validation of input means the browser checks the data the user inputs.

client-side validation - Client-side validation is the process of validating user input before submission to the server. A browser is often used to validate user input locally.

`datalist` element - The `datalist` element enables you to present the user with a drop-down list of options to select from. Only the options in the list may be selected.

`email` attribute - The `email` attribute requires the user to enter an email address into an input field.

`footer` element - The `footer` element defines a footer for a document or section, and typically contains information about the document or section, such as the author name, copyright data, links to related documents, and so on.

form input - Form input is the information a user enters into fields in a Web or client application form.

global attribute - A global attribute can be used with any HTML element; in other words, it’s permitted globally.

`header` element - The `header` element defines a header for a document, section, or article.

`menu` element - The `menu` element presents a list (or menu) of commands, usually with buttons. The W3C prefers that you use the menu element only for context menus, lists of form controls and commands, toolbars, and similar items.

`nav` element - The `nav` element defines a block of navigation links. The nav element is useful for creating a set of navigation links as your document’s primary navigation, a table of contents, breadcrumbs in a footer, or Previous-Home-Next links.

ordered list - A list that orders the list entries using numbers, by default.

`pattern` attribute - The `pattern` attribute provides a format (a regular expression) for an input field, which is used to validate whatever is entered into the field.

placeholder text - Placeholder text is text displayed inside an input field when the field is empty.

`required` attribute - The `required` attribute requires information in a field when the form is submitted.

`section` element - The `section` element defines a section in a document, such as a chapter, parts of a thesis, or parts of a Web page whose content is distinct from each other. The W3C specifies uses for the section element to differentiate it from other structure-related elements, mainly that it contain at least one heading and that it define something that would appear in the document’s outline.

semantic markup - Semantic markup is intuitive markup that gives better meaning or definition to several tags so they make more sense to humans, programs, and Web browsers.

server-side validation - Server-side validation is the process of a server validating data received from a user input form.

table - An HTML table contains rows and columns, and is used to organize and display information in a grid format.

unordered list - A list that displays list entries in a bulleted list.

validation - Validation is the process of verifying that information entered or captured in a form is in the correct format and usable before sending the data to the server.

Web form - A Web form is a Web page that provides input fields for a user to enter data, which is sent to a server for processing. From there, the information is stored in a database or forwarded to a recipient.

Lesson 3: Building the User Interface by Using HTML5: Organization, Input, and Validation- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. An HTML __table__ contains rows and columns, and is used to display information in a grid format.
2. Class and ID are __global__ attributes, which means they can be used with any HTML element.
3. An __ordered__ list orders the list entries using numbers, by default.
4. An __unordered__ list displays list entries in a bulleted list.
5. The HTML5 __menu__ element presents a list (or menu) of commands, usually with buttons.
6. Form __input__ is the information a user enters into fields in a Web or client application.
7. The __`required`__ attribute requires information in a field when the form is submitted.
8. __Placeholder__ text is displayed inside an input field when the field is empty. It helps users understand the type of information they should enter or select.
9. __Validation__ is the process of verifying that information entered or captured in a form is in the correct format and usable before sending the data to the server.
10. The __`autofocus`__ attribute moves the focus to a particular input field when a Web page loads.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which HTML5 element defines subdivisions in a document, such as chapters, parts of a thesis, or parts of a Web page whose content is distinct from each other?
   a. `aside`
   b. section
   c. `header`
   d. `article`
2. Which HTML5 element defines a part of an HTML document that consists of a "self-contained composition" that's independent from the rest of the content in the document and may be syndicated?
   a. `aside`
   b. `section`
   c. `header`
   d. article
3. Which HTML5 element is used to set off content that's related to the current topic but would interrupt the flow of the document if left inline?

a. aside

b. `section`

c. `header`

d. `article`

4. Which HTML5 attribute provides a format (a regular expression) for an input field, which is used to validate whatever is entered into the field?

a. pattern

b. `autofocus`

c. `required`

d. `placeholder`

5. Which of the following does validation *not* check and return an error for, by default, if invalid?

a. Required fields are empty

b. Valid email addresses

c. Email address to the wrong recipient

d. Text in a numeric field or vice versa

6. Which of the following is a practical use for a Web form?

a. To collect contact information from a user

b. To capture user comments after an article on a Web site

c. Both a and b

d. Neither a nor b

7. Which of the following are new `form` attributes in HTML5? (Choose all that apply.)

a. autocomplete

b. `target`

c. `method`

d. novalidate

8. What pattern attribute expression would you use to enter a product code that consists of three digits, separated by a hyphen, and then a single lowercase letter?

a. `[a-z]{1}-[0-9]{3}`

b. [0-9]{3}-[a-z]{1}

c. `[A-Z]{3}-[0-9]{1}`

d. `[0-9]{1}-[a-z]{3}`

9. Which of the following displays a key word or short phrase that describes the expected value of an input field, and then disappears when a user enters data?

a. label

b. placeholder

c. title

d. email

10. What is the format for the HTML5 tag that validates an email address?

a. `<input label="email" name="URL">`

b. `<form id="email">`

c. `<label for="email">Email</label>`

d. <input type = "email" name = "email">

True / False

Circle T if the statement is true or F if the statement is false.

[✓ T] F 1. In a table, the `tfoot` element must appear before the `tbody` element.

[✓ T] F 2. You can use numbers or letters for each item in an ordered list.

T [✓ F] 3. You can specify the height of an input element using the size attribute.

T [✓ F] 4. The `label` element displays the caption, or title, for a table.

[✓ T] F 5. The `nav` element defines a block of navigation links.

Competency Assessment

Scenario 3-1: Markup for a Newsletter Article

Sally Rowe, the document controller at Malted Milk Media, wants to publish a series of articles on the company intranet regarding document security and versioning. She needs to create a skeleton of the HTML5 markup for an article that will appear in the monthly online newsletter created by one of the Web developers. Each article will have a title and subtitle, several paragraphs of text, and her name and the article date in the footer. What should her article markup look like?

One possible example of markup for Sally Rowe’s article would look like the following:

```
<article>
  <header>
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
  </header>
  <p>content</p>
  <p>content</p>
  <p>content</p>
  <footer>
    <p>by Sally Rowe, date</p>
  </footer>
</article>
```

Scenario 3-2: Displaying Long Tables in HTML

Vince generates accounting reports for the VP of Finance at Momentum Strategies, a PR firm geared toward political campaigns. Vince regularly prints table that are two- or three-pages long and delivers hard copies to senior management staff. He wants to publish them to a secure area of the company intranet, but the rows of data separate from the column headings and totals line at the end. He wants to know how to present the tables properly in HTML5. What do you tell him?

Tell Vince that a long table that requires scrolling within a browser requires the `<thead>`, `<tfoot>`, and `<tbody>` tags. The content within the table header and footer will remain on the page while the content marked by `<tbody>` will scroll between them.

Proficiency Assessment

Scenario 3-3: Creating a Glossary of Terms

Waylon is a student working on a term paper. His instructor requires each student to format the paper for display on the Web. Waylon wants to include a glossary of terms at the end of the paper but can't produce the right "look" using an unordered list. Which markup would be better suited for Waylon's glossary?

Waylon should use a definition list. It displays items with their definitions below the list item and indented. The `<dl>` tag defines the list, the `<dt>` tag marks each term in the item, and the `<dd>` tag defines each description.

Scenario 3-4: Using Proper Input Types in a Web Form

Margie is creating and testing a Web form that includes an email field, a Web address field, and a zip code field, among others. When she has a few co-workers test the form, she finds they often enter the email address in the Web address field by mistake, and sometimes enter too many or too few numbers in the zip code field. She doesn't want to use a pattern expression because she says it's too complicated. What other input types can Margie use?

In HTML5, several input element types offer automatic validation of input, which means the browser checks the data the user inputs. If the user enters the wrong type of data into a field, such as an email address in a field with the `url` attribute, the browser instructs the user to enter a valid URL. Some of the elements include `<input type="email">`, `<input type="url">`, and `<input type="number">`. Using placeholder text could also help.

Lesson 3: Building the User Interface by Using HTML5: Organization, Input, and Validation – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Understanding Semantic HTML

In HTML 4.01 and prior specifications, a developer creating the structure of an HTML document uses the `<div>` tag frequently throughout. The `<div>` tag often includes a class or ID attribute, which may also include CSS styles such as `background-color`, `height`, and `width`. A simple example of a `<div>` tag is:

```
<div id="header" > This is a header </div>
```

The `div` element alone doesn't have much meaning without the `id` or `class` attribute. Even the ID can be assigned a value of your choice, such as "`header`", "`header_inner`", "`slogan`", "`content`", "`style`", and many more. An example from an HTML 4.01 document is shown as follows:

```
<div id="header">

  <div id="header_inner">

    <img src="images/doghappy.jpg"
      alt="Attaboy Pet Services" />

      <div id="slogan">Happy dogs are good dogs</div>

  </div>

</div>
```

Using Tags to Add Structure to an HTML Document

The header and footer Elements

An example of an article with a `header` tag and a `footer` tag is as follows:

```
<article>

  <header>

    <h1>Learning HTML5</h1>

    <h2>The New Elements</h2>

  </header>

  <p>New HTML5 tags make Web page and application
  development easier than ever.</p>

  <footer>

    <p>Published: <time datetime="2012-09-
    03"September 3, 2012</time></p>

  </footer>
```

```
</article>
```

The section Element

When defining a section header, which may contain h1 through h6 headings, you can use the `hgroup` element to group headings. The `hgroup` element affects organization but not presentation. Consider using `hgroup` when you have a heading and a subheading one right after the other, as follows.

```
<section>

   <hgroup>

       <h1>Hip-Hop Dance Routines</h1>

         <h3>The Eight-Count Method</h3>

   </hgroup>

   <article>

     <p>Hip-hop dance instructors often teach
     moves that have eight counts per set.</p>

   </article>

</section>
```

Create an HTML Document with a Header, Sections, and a Footer

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>My Page</title>
</head>
<body>
<header>
   <h1>Selecting a Concert Style</h1>
</header>
<section>
   <h1>Symphonies</h1>
   <p>A symphony is a type of musical composition generally
   performed by a full orchestra.</p>
</section>
<section>
   <h1>Raves</h1>
  <p>A rave is a gathering of people who listen and dance
   to music, especially electronic music, usually performed
   by a live band or live DJs.</p>
</section>
<footer>
   <p>Author: Nathaniel Becker</p>
</footer>
```

```
</body>
</html>
```

The nav Element

The W3C mentions that you don't have to use `<nav>` tags for all navigation links, just major blocks of links. Because `<nav>` tags are interpreted by screen reader software for the visually challenged, the software can determine if it should make the navigation links available to the user immediately or not, depending on their importance.

The following example shows the `<nav>` tag in use:

```
<nav>

  <a href="/hiphop/">Hip-Hop</a>

  <a href="/modern/">Modern</a>

  <a href="/swing/">Swing</a>

  <a href="/tap/">Tap</a>

</nav>
```

An example of markup for Previous-Home-Next links follows, with vertical bars after each navigation item to separate it from the others visually:

```
<nav>
  <a href="http://www.example.com/Services">Previous</a> |
  <a href="http://www.example.com">Home</a> |
  <a href="http://www.example.com/About">Next</a>
</nav>
<br />
```

The aside Element

The ***`aside`* element** is used to set off content that's related to the current topic but would interrupt the flow of the document if left inline. Essentially, the `aside` element is used for information that lends itself to sidebars and notes. This content might give a more detailed look at a topic, offer related reading links, or display definitions for keywords in the paragraph. The `aside` element doesn't change the position of content or how the content displays; it simply lets the browser and search engines know that it's related content.

```
<article>

  <header>

    <h1>Learning HTML5</h1>

    <h2>The New Elements</h2>

  </header>

  <p>New HTML5 tags make Web page and application
  development easier than ever. One of the very
  handy new features of HTML5 is the use of
  semantic markup.</p>

  <aside>

      <h4><b>semantic markup</b></h4>
```

```
        <p> gives better meaning, or definition,
        to tags so they make more sense to humans,
        programs, and Web browsers</p>

    </aside>

  <p>Not all HTML tags have been replaced or updated
  for HTML5, but some new tags introduced in HTML5
  make the work of creating Web pages a lot
  easier.</p>

<footer>

    <p>Published: <time datetime="2012-09-
    03">September 3, 2012</time></p>

  </footer>

</article>
```

Using Tags to Create Tables and Lists

Creating Tables

The markup for a very basic two-column, five-row table is as follows. Comments have been added to indicate columns and rows, which are informational only and don't appear when the document is viewed in a browser, is shown in Figure 3-11:

```
<table border="1">

  <tr> <!--first row-->

    <th>Quarter</th> <!--first column in first row-->

    <th>Total Sales</th> <!--first row, second column-->

  </tr>

  <tr> <!--second row-->

    <td>Q1</td>

    <td>$4,349</td>

  </tr>

  <tr> <!--third row-->

    <td>Q2</td>

    <td>$2,984</td>

  </tr>

  <tr> <!--fourth row-->

    <td>Q3</td>

    <td>$3,570</td>

  </tr>

  <tr> <!--fifth row-->

    <td>Q4</td>
    <td>$7,215</td>

  </tr>

</table>
```

The following is an example of the markup for a table with three columns and five rows, the first row being the column headings and the last row the table foot. The markup also includes a caption above the table. The markup is shown rendered by a browser in Figure 3-12:

```
<table>

  <caption>Sales for Employee ID 2387</caption>

    <colgroup
      span="2"
      style="background-color:#EEE8AA;">
    </colgroup>

    <colgroup
      style="background-color:#00FA9A;">
    </colgroup>


  <thead>

    <tr>

      <th scope="col">Quarter</th>

      <th scope="col">Total Sales</th>

      <th scope="col">Goal Met?</th>

    </tr>

  </thead>

<tfoot>

    <tr>

      <th scope="col">Total</th>

      <th scope="col">$18,118</th>

    </tr>

</tfoot>

<tbody>

    <tr>

      <td>Q1</td>

      <td>$4,349</td>

      <td>Yes</td>

    </tr>

    <tr>

      <td>Q2</td>
      <td>$2,984</td>
      <td>No</td>
    </tr>
    <tr>
      <td>Q3</td>
      <td>$3,570</td>
      <td>Yes</td>
    </tr>
    <tr>
      <td>Q4</td>
      <td>$7,215</td>
      <td>Yes</td>
    </tr>
  </tbody>
```

Create a Table

GET READY. To create a table, perform the following steps:

1. Using an HTML editor or app development tool and a Web browser, create a file named **L3-PracTable.html** with the following markup:

```
<!doctype html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>High-grossing Movies</title>
</head>
<body>
<table border="1">
  <tr>
    <th>Movie</th>
    <th>Gross Proceeds</th>
  </tr>
  <tr>
    <td>Avatar</td>
    <td>$2.7 billion</td>
  </tr>
  <tr>
    <td>Titanic</td>
    <td>$2.1 billion</td>
  </tr>
  <tr>
    <td>The Dark Knight</td>
    <td>$1.0 billion</td>
  </tr>
</table>
</body>
</html>
```

Creating Lists

- ***Unordered list:*** Displays list entries in a bulleted list. It uses a `<ul>` tag.

Items in a list are marked by `<li>`, which indicates an ordinary list item. Let’s look at some examples.

Here is an ordered list, shown in a browser in Figure 3-14:

```
<p>Favorite cupcakes:</p>

<ol>

  <li>Chocolate chip cheesecake</li>

  <li>Strawberry delight</li>

  <li>Italian creme</li>

</ol>
```

```
<p>Cupcake flavors:</p>

<ul>

  <li>Strawberry delight</li>

  <li>Chocolate chip cheesecake</li>

  <li>Italian creme</li>

</ul>
```

You can change the round bullet symbols in an unordered list by simply adding an attribute to change the nature of the bullets. For square symbols, add `type="square"` to the `<ul>` tag, and for empty circles add `type="circle"`. You can also add the attributes to individual list items (marked by `<li>`) to affect individual points. For example, to display all bullet symbols as filled-in squares:

```
<p>Cupcake flavors:</p>

<ul type="square">

  <li>Strawberry delight</li>

  <li>Chocolate chip cheesecake</li>

  <li>Italian creme</li>

</ul>
```

Another type of list is the definition list. It displays items with their definitions below the list item and indented. The `<dl>` tag defines the list, the `<dt>` tag marks each term in the item, and the `<dd>` tag defines each description. Here's an example of the markup, and Figure 3-17 shows the rendered list.

```
<dl>

  <dt>Strawberry delight</dt>

    <dd>Strawberry meringue buttercream with
    tiny wild strawberries</dd>

  <dt>Chocolate chip cheesecake</dt>

    <dd>Mini chocolate chips blended with creamy
    cheesecake and a chocolate brownie bottom,
    topped with cream cheese frosting</dd>

  <dt>Italian creme</dt>

    <dd>Italian cream cake topped with cream
    cheese frosting and toasted coconut</dd>

</dl>
```

Understanding Input and Forms

Exploring Form Creation, Input Attributes, and Values

If the form is included in an HTML document with other items, you can use the `<div>` tag at the beginning and end of the form to separate it from other content. Using the `<div>` tag also lets you include inline formatting, if the form uses tags to align fields vertically short and simple and you don't want to create a CSS style sheet. The `<div>` tag uses the `id` attribute and appears before the first `<form>` tag. The `label` element displays the label for each field. An example of the markup for a very simple form is:

```
<div id="contact-form"

  style="font-family:'Arial Narrow','Nimbus Sans
  L',sans-serif;">

   <form id="contact" method="post" action="">

    <fieldset>

      <label for="name">Name</label>

      <input type="text" name="name" />

    </fieldset>

    <fieldset>

      <label for="email">Email</label>

      <input type="email" name="email" />

    </fieldset>

 </form>

</div> <!-- end of contact-form -->
```

The ***`datalist`* element** enables you to present the user with a drop-down list of options to select from. Only the options in the list may be selected. Alternately, you could insert `type="text"` into the `input` element to create a text box in which the user enters text. The following example lets the user select from one of three countries:

```
<input id="country" name="country"
size="30" list="countries" />

  <datalist id="countries">

    <option value="United States">

    <option value="Canada">

    <option value="United Kingdom">

  </datalist>
```

The `search` value for the `type` attribute enables you to create a search feature for a Web page. An example of the markup is:

```
<form>

  <input name="search" required>

  <input type="submit" value="Search">

</form>
```

Create a Simple Web Form

GET READY. To create a simple Web form, perform the following steps:

1. Using an HTML editor or app development tool and a Web browser, create a simple Web form with the following markup:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Contact Us</title>
</head>
<body>
  <div id="contact-form">

  <form id="contact" method="post" action="">
  <fieldset>

  <label for="custname">Name</label>
  <input type="text" id="custname" />

  <label for="email">Email</label>
  <input type="email" id="email" />

  <label for="phone">Phone</label>
  <input type="text" id="phone" />
  <label for="message">Questions or
     Comments</label>
  <textarea name="message"></textarea>

  <input type="submit" name="submit" id="submit"
     value="Submit" />

</fieldset>
</form>

</div><!-- End of contact-form -->

</body>
</html>
```

Lesson 4: Understanding CSS Essentials: Content Flow, Positioning, and Styling

Learning Objectives

Students will learn about:

- CSS essentials
- Separating content from style
- Understanding selectors and declarations
- Understanding fonts and font families
- Managing content flow
- Positioning individual elements
- Managing content overflow

ODN Skills

Understand the core CSS concepts.	4.1

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to test and show all of the examples in Lesson 4.

Lesson 4 introduces students to content flow, positioning, and styling using Cascading Style Sheets (CSS).

Begin the lecture by discussing presentation (or style) versus content. To drive home the point, perhaps show a Web page with and without CSS styling. Then segue into a discussion of CSS, what it's used for, and what the "cascading" part of CSS means.

Explain that, like HTML, all you need for composing CSS code is a simple editor like Notepad. However, you can also use Microsoft Visual Studio, Microsoft Expression Blend, Expression Studio, Notepad++ for Windows or TextWrangler for Mac OS, Microsoft Web Matrix, or a number of other tools. You need a Web browser to check the effect of CSS code on the HTML markup.

Next, show how CSS links to HTML, both inline and from a separate CSS file. This is also a good time to introduce concepts of basic CSS styling, such as CSS selectors, declarations, properties, and attributes. Discuss the significance of curly brackets, the pound (#) sign, and class selectors. You'll need to provide examples for all of the concepts and topics covered in this part of the lecture.

Then, discuss fonts and font families. Point out the difference between serif and sans serif fonts, and the purpose of monospace fonts. Briefly mention the concept of Web-safe fonts and, alternatively, use of the @font-face rule. Students should come out of the lecture knowing how to manipulate an HTML page using fonts and font families. They will learn more about Web-safe fonts and be introduced to the Web Open Font Format (WOFF) in Lesson 7.

The next section addresses inline flow and block flow. Explain that inline flow forces no new lines before or after the inlined element, but simply places the element between the content before and after the inlined element. In block flow, in contrast to inline flow, an element is separated from other elements by new lines above and below, and fills from left to right the horizontal extent where it appears. Show examples of inline flow and block flow applied to the same content.

Next, cover float positioning and absolute positioning. Explain that float positioning is useful when a layout is in columns, at least in part. To float an element is to have it move as far as possible either to the right or left; text then "wraps" around the element. With absolute positioning is more exact and uses geometric placement. Show the two types of positioning techniques applied to the same content.

The next section covers overflow. To begin the discussion, describe the concept of the bounding box. Then, discuss the `overflow` property in the context of scrolling overflow, visible overflow, and hidden overflow. Again, show the effects of the `scroll`, `visible`, and `hidden` attributes applied to the same content.

Key Terms

absolute positioning - Absolute positioning is the placement of an element at a geometric position in the display, relative to the first parent element that has a non-static position.

block flow - Block flow is a positioning method in which an element is separated from other elements by new lines above and below, and fills from left to right the horizontal extent where it appears.

bounding box - A bounding box is an invisible rectangle, the smallest perimeter of which surrounds a word.

Cascading Style Sheets (CSS) - Cascading Style Sheets (CSS) is a style sheet language that defines styles for HTML. CSS styles are usually saved in a separate file from the HTML file. This enables you to easily change fonts, font sizes, and other attributes in the CSS file and the changes are reflected across all HTML files that reference the CSS file.

class - Class is an attribute a Web author uses to provide structure to a document beyond the meaning HTML builds in with elements such as paragraph, header, and so on.

declaration - The declaration is the style for a specific selector. A declaration has a property, which is a style attribute, and a value.

float positioning - Float positioning is the flexible placement of elements, enabling them to move as far as possible either to the right or left; text then "wraps" around the element.

font - A font is a set of characters of a particular size and style.

`font-family` property - The `font-family` property can declare either a specific font, like Garamond or Arial, or a wider family that includes many different fonts, such as "serif."

hidden overflow - Hidden overflow is a feature that makes overflow invisible.

inline flow - Inline flow is a feature that forces no new lines before or after the inlined element, but simply places the element between the content before and after the inlined element.

monospace - Monospace is a type of font family in which each character is the same width. Monospace is often used for technical material such as formulas, numbers, codes, and so on.

rules - Rules are statements that tell Web browsers how to render particular elements on an HTML page or how to apply CSS styles to Web pages.

sans serif - Sans serif is a font style drawn without serifs, such as the Arial font. Serifs are the details at the ends of particular letters; look at the "d," "p," and "t" in this sentence for examples of serif characters.

scrolling overflow - Scrolling overflow is a feature that prevents content that can overflow its box from appearing outside the box. The content appears to be clipped.

selector - A selector is a feature that defines which HTML elements will be affected by CSS code. For example, in CSS, the `p` selector means a particular style will be applied to paragraphs. The general syntax for a selector is `selector {property: value}`.

visible overflow - Visible overflow is a feature that enables content that overflows it box to appear in the display rather than be clipped or hidden. Visible overflow also writes over the content that follows it.

Lesson 4: Understanding CSS Essentials: Content Flow, Positioning, and Styling- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. HTML has responsibility for content, and CSS for _style__.
2. An HTML source file refers to an external CSS source file with the __link__ element.
3. A CSS source file consists of zero or more individual _rules__.
4. An individual CSS rule has two parts: a __selector__ and one or more declarations.
5. An individual declaration within a CSS rule consists of a _property_, followed by a colon, then a value, and a semi-colon terminator.
6. The most common CSS selectors are: element or type, id, and __class__.
7. The two visible HTML content flows are __inline__ and block.
8. To make HTML elements appear in columns, it is common to apply __float__ positioning.
9. Suppose an element is subject to overflow: it might grow beyond the size allocated for it on the screen. To give it scrollbars which make it possible for an end user to see the entire element, declare the overflow property to have the __scroll__ value.
10. The most common values for the float property are _left_ and _right_.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which of the following best summarizes a useful pattern for commercial development?
   a. Web pages are written in HTML
   b. Designers need to learn Java or Ruby to layout displays
   c. CSS takes responsibility for visual style
   d. CSS defines structure, and HTML assigns colors and fonts
2. Which of the following codes a comment within CSS?
   a. `<!-- ... -->`
   b. /* ... */
   c. `# ... d.`
   d. `// ...`
3. How many different rules within a single valid CSS source file can declare the style of a paragraph `<p>` element?
   a. 0
   b. 1
   c. 1 or 2, depending on whether HTML5 is used

d. 0 or more

4. Sometimes colors are expressed with English words and sometimes they are expressed with symbolic numbers. Which of the following means “blue”?

   a. `009`

   b. #0000FF

   c. `!008000`

   d. `(128, 128, 128)`

5. A paragraph appears on an important display coded as “`<p id = 'introduction'>Trusty Lawn Care takes . . .`”. You’ve been told this paragraph must appear in the Tahoma font. Which of the following will best help you define an appropriate rule?

   a. `p {font: Tahoma;}`

   b. #introduction {font-family: Tahoma;}

   c. `.introduction {font: Tahoma;}`

   d. `.p {font-family: Tahoma;}`

6. Someone has set up a Web page with HTML that links to three different CSS source files. The name of one of the source files is misspelled in the HTML. Which of the following is a Web browser most likely to display?

   a. It shows a display as though the link to the CSS with a misspelled name is simply missing.

   b. It shows the misspelled name of the CSS source along with an error message.

   c. It shows as much of the display as possible, using the last CSS correctly linked in place of the CSS with a misspelled name.

   d. It displays a warning that the CSS can’t be found and asks whether you want to continue anyway.

7. The anchor tag `<a >` is the HTML element defined for definition of hyperlinks, among other things. Name the default content flow for an anchor.

   a. Inline

   b. Block

   c. Hidden

   d. Visible

8. Your team is constructing an application that embeds a long license statement end-users must have a chance to read and approve. You want to limit the amount of space on the screen the license fills up, at the same time as you make every word of it available to end users who choose to read all of it. Which of the following is most likely to help you code this?

   a. `{position: scrolling;}`

   b. `{fixed: scrolling;}`

   c. `{overflow: scrolling;}`

   d. {overflow: scroll;}

9. The latest CSS standard that is still under development but widely used is:

   a. CSS8

   b. CSS5

   c. CSS3

d. CSS2

10. When HTML links to CSS you have written, which is most likely to be a useful part of the link?

    [✓ a.] type = 'text/css'

    b. CSS = “SOME_NAME.css”

    c. type = "style/CSS"

    d. Web = "style/css"

True / False

Circle T if the statement is true or F if the statement is false.

T [✓ F] 1. A CSS source file includes two different rules for the font of an `h1` element. Your Web browser applies the rule one closer to the top of the source file, and ignores the one that appears closer to the bottom.

[✓ T] F 2. Overflow for a particular element is defined through CSS to scroll. There happens to be no overflow, because the content of the element is unusually short at the moment. Even in this case, the scrollbar is visible.

[✓ T] F 3. You’ve been asked to lay out a design with columns. Float positioning is more likely than absolute positioning to be useful for this situation.

T [✓ F] 4. If you use the latest CSS standard in your coding, end users who rely on old Web browsers will be warned that your pages present security risks.

T [✓ F] 5. Before you can test the CSS you write, you need to make sure a Python compiler is installed on your computer.

Competency Assessment

Scenario 4-1: Basic Workflow

Your team is developing an application. You have responsibility for “styling the layout.” What files are you likely to update?

The CSS source(s) for the project. Sketch for the team the name of the CSS source file; if you think you need more than one, explain the reason, and how they are related. As a style coder, you should not change any HTML or “active” pages (in Java, PHP, and so on). You might, however, provide a model HTML source that illustrates how your CSS is linked and used.

Scenario 4-2: Client Consultation

A customer has defined a rather rigid layout in which one news item area occupies a fixed location within the display. The news item has the potential to overrun the area assigned to it. Quickly create sample displays that show for the customer how the layout works with scrollbars for the news item, or with the news item simply truncated if it overruns its assigned “box.”

Quickly put together a sample HTML source with several distinct segments of text. Use CSS positioning and geometry to place the news item where it should be. Show how the display looks when overflow is scroll and hidden, respectively.

Proficiency Assessment

Scenario 4-3: Theatrical Dialogue

An application already in use displays dialogue for different actors. The display seen by a particular actor has his or her lines in normal font, while everything spoken by others is italicized. This was originally implemented by putting each speech in its own paragraph, with successive paragraphs labeled “paragraph1”, “paragraph2”, and so on. CSS then styled the paragraphs by speaker. Show your team a more efficient and easily maintained way to use CSS to achieve the same result.

The existing implementation has many dozens of CSS rules, one for each paragraph of the dialogue. All of these rules are copies of other rules. By selection on class, rather than id, all these rules can be collapsed down to just two. Call these two classes “self” and “others” and then demonstrate how the old and new stylesheets might look.

Scenario 4-4: International Cooperation

You’re a member of a distributed development team. A public-relations firm in Virginia is responsible for copy that will appear on a Web site; at the same time, a Belgian consultancy is providing fashionable typography, while coders in Egypt make sure that the lay-out has the correct appearance and color scheme. How would you outline the file structure of this project?

Create main.html. Within main.html, write:

```
<link href = "typography.css" rel = "stylesheet" type =
"text/css">
<link href = "layout.css" rel = "stylesheet" type =
"text/css">
```

Now each locale can confine its updates and edits to a single source file: main.html for Virginia, typography.css for Belgium, and layout.css for Egypt. While creation and naming of separate files of this sort might seem obvious, trivial, or unnecessary, many real-life situations similar to this suffer for lack of good, simple organization.

Lesson 4: Understanding CSS Essentials: Content Flow, Positioning, and Styling – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Exploring the Link between HTML and CSS

A Simple Use of CSS with HTML

GET READY. There are a number of ways to style an HTML page with CSS styles. Here is a basic way to start:

1. Use a text editor or app development tool to create a file in your home directory called **e1.html** with the following content:

```
<!doctype html>
<html>
  <head>
    <title>Trusty Lawn Care, Inc.</title>
        <link href = "e1.css" rel = "stylesheet"
           type = "text/css">
  </head>
<body>
  <h1>Trusty Lawn Care, Inc.</h1>
  <p id = "slogan">We keep you in green.</p>
  <p>Trusty Lawn Care can keep your lawn looking lush and
   vigorous all season. We use only natural fertilizers,
   mulches, and soils to boost the health of your turf.</p>
</body>
</html>
```

2. Create a second file, in the same folder as the **`.html`** source; name it as **e1.css** and use the following content:

```
#slogan {
  font-size: 20px;
  color: green;
  font-style: italic;
}
```

Understanding Selectors and Declarations

This is the content of an HTML file, named e2.html:

```
<!doctype html>
<html>
  <head>
```

```
    <title>A class example</title>
    <link href = "e2.css" rel = "stylesheet"
    type = "text/css">

  </head>

<body>

  <h1>About States</h1>

  <p class = "fact">Alaska is the largest U.S. state
  in area.</p>

  <p class = "opinion">New Jersey deserves its
  nickname “Garden State.”</p>

  <p class = "fact">A single congressman represents
  Wyoming in the national House of
  Representatives.</p>

</body>

</html>
```

This is the content of the associated CSS file, named e2.css:

```
p {color: black;}

/* The prefix for a class selector is a period: ‘.’ */

.opinion {color: gray;}
```

Understanding Fonts and Font Families

Managing Content Flow

Explore Inline Flow and Block Flow

GET READY. To explore inline flow and block flow, perform the following steps:

1. Create the file **e4.html** with the following contents:

```
<!doctype html>
<!-- This is the content of the file e4.html.-->
<html>
  <head>
    <title>Block and inline flow</title>
       <link href = "e4.css" rel = "stylesheet"
       type = "text/css">
<style type = 'text/css'>
   .toolbar li {
   }
</style>
  </head>
<body>
<h1>Block and inline flow</h1>
```

```
<p>Here are choices you can make:</p>
<ul class = "toolbar">
   <li>Automobile</li>
   <li>Bicycle</li>
   <li>Scooter</li>
   <li>Taxi</li>
   <li>Walk</li>
</ul>
</body>
</html>
```

2. When you display this source in your browser, you see something like Figure 4-9.
3. Update the source of e4.html so that the `<style>` segment looks like

```
<style type = "text/css">
   .toolbar li {
      display:inline;
      background-color: #EEE;
      border: 1px solid;
      border-color: #F3F3F3 #BBB #BBB #F3F3F3;
      margin: 2px;
      padding: .5em;
   }
</style>
```

Positioning Individual Elements

Use Float Positioning with Multi-Columns

GET READY. To apply float positioning to multiple columns, perform the following steps:

1. Create the file **e5.html** with the following contents:

```
<!doctype html>
<!-- This is e5.html. -->
<html>
<head>
<title>Float positioning</title>
<style type = 'text/css'>
#col1 {
    float: left;
    width: 150px;
    background-color: lightskyblue;
}
#col2 {
    float: left;
    width: 120px;
    background-color: yellow;
}

</style>
</head>
<body>
<h1>Float positioning</h1>
<p id = "col1">Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultricesviverra velit.

<p id = "col2">Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultrices   viverra
   velit.

<p id = "col3">Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultrices viverra velit.

</body>
</html>
```

Use Absolute Positioning with Multi-Columns

GET READY. To apply absolute positioning to multiple columns, perform the following steps:

1. Create **e6.html** by opening **e5.html** saving a copy as **e6.html**.
2. Replace the comment at the top with:

```
<!-- This is e6.html. -->
```

Replace the content within the `<head>` tags with the following:

```
<title>Absolute positioning</title>
<style type = 'text/css'>
#col1 {
    position: absolute;
    bottom: 100px;
    right:  100px;
    background-color: lightskyblue;
}
#col2 {
    background-color: yellow;
}

</style>
```

Managing Content Overflow

Work with Scrolling Overflow

GET READY. To practice scrolling overflow, perform the following steps:

1. Create **e7.html** with the following contents:

```
<!-- This is the content of e7.html. -->
<!doctype html>
<html>
<head>
<title>Scroll overflow</title>
<style type = "text/css">
#col1 {
width: 200px;
height: 200px;
background-color: lightskyblue;
overflow: scroll;
}
#col3 {
background-color: yellow;
}

</style>
</head>
<body>
<h1>Scroll overflow</h1>

<p id = 'col1'>Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultrices viverra velit.
   Pellentesque habitant morbi tristique senectus et netus
   et malesuada fames ac turpis egestas. Vestibulum tortor
   quam, feugiat vitae, ultricies eget, tempor sit amet,
   ante.  Donec eu libero sit amet quam egestas semper.
</p>
<p id = 'col2'>Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultrices viverra velit.
</p>
<p id = 'col3'>Lorem ipsum dolor sit amet, consectetuer
   adipiscing elit. Integer pretium dui sit amet felis.
   Integer sit amet diam. Phasellus ultrices viverra velit.
</p>
</body>
</html>
```

Lesson 5: Understanding CSS Essentials: Layouts

Learning Objectives

Students will learn about:

- Arranging user interface content with CSS
- Using a flexible box to establish content alignment, direction, and orientation
- Using grid layouts to establish content alignment, direction, and orientation

ODN Skills

Arrange user interface (UI) content by using CSS.	3.2

Lesson Summary — Lecture Notes

General Notes: This lesson focuses mainly on flexible boxes and grid layouts. The topics can be quite challenging for both the beginning student and the seasoned developer. However, the Microsoft 98-375 exam tests on both flexible boxes and grid layouts, so students must learn the basics.

When the textbook was written, Internet Explorer 9 supported some flexbox properties and attributes but did not display grid layouts. Google Chrome displayed flexboxes most reliably, and was used to display the `flex-wrap`, `flex-flow`, and `flex-order` examples in the book. Internet Explorer 10 provided some support for grid layouts. You can use the When Can I Use Web page at caniuse.com, which lists many CSS3 properties and indicates which Web browsers support them.

The CSS Flexible Box Layout Module and the CSS3 Grid Layout specifications have been undergoing significant changes. You should browse the CSS Flexible Box Layout Module Web page at http://www.w3.org/TR/css3-flexbox/ and the CSS3 Grid Layout Web page at http://www.w3.org/TR/css3-grid-layout/ before teaching this lesson.

Lesson 5 introduces students to CSS layouts. Begin the lecture by explaining general layouts of HTML pages—from simple with just search box and a button to highly complex. Show a few examples, such as a comparison of the Bing.com Web page and the Microsoft.com main Web page. Point out that although Bing.com and Microsoft.com look very different, they follow the elements of good design—clean, easy to use, well-structured, and so on. Part of the appeal of both pages is the use of excellent positioning of elements.

Next, discuss the CSS Box model, which includes margin, border, padding, and content. This model was one of the original layout models. Explain block-level elements and inline elements, and then discuss parent/child relationships.

Explain that developers have used floats for many years to build flexibility into their page layouts. However, floats don't always render pages well in smaller screens like tablets and smartphones. Now they can use the CSS3 Flexbox Box model, which is ideal for items that should resize or reposition themselves (horizontally or vertically) depending on the size of the screen. Describe the general benefits of flexboxes.

Before you begin discussing flexbox properties, explain that the CSS3 specification is still in draft format and undergoing modifications. Developers use vendor prefixes with some CSS3 properties to help ensure they work during the transition phase. For example, the Internet Explorer vendor prefix is -ms-, and the Mozilla Firefox prefix is -moz-. Emphasize that students must use vendor prefixes with most flexbox properties.

Then discuss details of the flexbox model. It's important that students understand the following; providing examples will be critical to students understanding the concepts:

- You define an element as a flexbox using the CSS properties `display:flexbox` or `display:inline-flexbox`.
- The `flexbox` attribute sets the flexbox as a block-level element
- The `inline-flexbox` attribute sets the flexbox as an inline-level element
- A box within a box is a child box, which can be flexible or not. A child box is referred to as a flexbox item.
- The `flex` property controls the height and width of flexbox items.
- Whereas the `display:flexbox` property creates a flexible parent box, the `flex` property is what gives the flexible nature to child boxes.

Next, discuss the values the `flex` property can use: a positive and/or negative flex value, a preferred size, and the `none` keyword, and provide an example of at least one of the values.

Discuss some of the significant `flexbox` properties, such as `flex-wrap`, `flex-pack`, and `flex-align`:

- The `flex-wrap` property determines whether child boxes automatically create a new line and wrap onto it.
- The `flex-pack` property justifies the alignment of child boxes within a flexbox and minimizes whitespace in the parent box. This property accepts one of four values: `start`, `end`, `justify`, or `center`.
- The `flex-align` property sets the default alignment for child boxes, but with a twist. If the orientation of the parent box is horizontal, `flex-align` determines the vertical alignment of the child boxes, and vice versa.

Next, discuss properties that affect the direction and order of child boxes (flex items) within flexboxes:

- The `flex-direction` property affects the direction of child boxes in the parent box. It uses the row, row-reverse, column, and column-reverse values.
- The `flex-flow` property sets the `flex-direction` and `flex-wrap` properties at the same time.
- The `flex-order` property controls the order and arrangement of child boxes in a flexbox by placing the child boxes in ordered groups.

The next section addresses the CSS3 Grid Layout model, which gives developers greater control over complex layouts than the flexbox model. The Grid Layout model lets you control the design of sections or entire HTML-based documents using CSS3. Explain that grid layouts are similar to spreadsheets in that they use columns, rows, and cells, but you can easily move blocks of content from one part of a page or application to another by moving code lines in CSS.

Explain that the basic CSS property of a grid layout is either `display:grid` or `display:inline-grid`, which creates the container for the layout. Grids also use the `grid-columns` and `grid-rows` properties. Child elements of a grid are called grid items.

Discuss the concept of defining columns and rows with a fixed size versus a fractional size relative to the grid. Fractional sizes are defined using fr (short for "fraction"), so a row defined as 2fr will be twice the size of a row defined as 1fr.

Show an example of a grid layout with fixed-size elements and one with fractional-size elements.

The last section covers grid templates, part of the W3C's CSS Grid Template Layout Module. Explain that a grid template is like an empty table into which data can be flowed. A grid template uses the `grid-position` property along with alphabetical characters to represent the position of items in a grid. Provide an example.

Key Terms

block-level element - A block-level element creates boxes that contribute to the layout of an HTML document. Sections, articles, paragraphs, lists, and images are examples of block-level elements.

border - A border is a colored or transparent line, which can be thin or thick, that surrounds a box. The border is a part of the CSS Box model.

content - Content is whatever is displayed on a Web page, such as text and images. Regarding the CSS Box model, content is text or images contained within a box. You use the `border`, `margin`, `padding`, `height`, and `width` CSS properties to modify various parts of the CSS Box model.

flexbox - A flexbox is a type of layout that enables relative sizes and positioning of boxes. Flexbox takes available space into account when defining box dimensions. A box can include child boxes that are flexible by height and width.

flexbox item - A flexible item is a child box. A child box can be flexible or static. The `flex` property makes child boxes flexible.

Flexbox Box model - The CSS Flexbox Box model is a layout mode for using flexible boxes in user interfaces.

grid item - A grid item is a child element of a grid.

grid layout - A grid layout is a way to structure complex HTML documents using rows and columns to make the design look cleaner and structured.

Grid Layout model - The CSS Grid Layout model is a model for structuring HTML layouts. This model lets you control the design of sections or entire HTML-based documents using CSS3.

grid template - A grid template, an approach to grid layouts, is like an empty table into which data can be flowed. A grid template uses alphabetical characters to represent the position of items in a grid.

inline element - Inline elements are elements designed for laying out text and don't disrupt the flow of the document. Applying boldface and the new HTML5 `mark` element are examples of inline elements.

margin - A margin is the outermost edge of a box, providing space between the box and other boxes in an HTML document. Margins are transparent. The margin is a part of the CSS Box model.

media queries - A media query is a CSS3 feature that detects the user's type of screen and sizes the output accordingly.

padding - Padding is the space between the border of a box and its content. Padding generally takes on the same color as the box's background color. Padding is a part of the CSS Box model.

parent/child relationship - The parent/child relationship describes how a parent box can contain one or more child boxes. Boxes contained within a parent box are referred to as child boxes. Child boxes inherit attributes applied to parent boxes unless coded otherwise.

user interface (UI) - A user interface (UI) is the portion of a Web site or application with which a user interacts.

vendor prefix - A vendor prefix is a keyword surrounded by dashes, added to the front of a CSS3 property name. The Microsoft Internet Explorer Web browser recognizes the
-ms- prefix. Vendor prefixes help to ensure CSS3 styles work properly in Web pages during the transition from CSS2 to CSS3.

Lesson 5: Understanding CSS Essentials: Layouts- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. A __user interface, or UI,__ is the portion of a Web site or application with which a user interacts.
2. In the original W3C CSS box model, the __padding__ is the space between border and the content of the box.
3. In the W3C CSS box model, a _block_-level element creates boxes that contribute to the layout of the document.
4. Flexbox children are called _flexbox items_ and are laid out using the flexbox model.
5. Child elements of a grid are called __grid items__.
6. __Flexbox__ offers flexible layouts for UI design, mainly to create controls, toolbars, menus, and forms that resize and reposition automatically when the user changes the size of the browser window.
7. The __parent/child__ relationship describes how a parent box can contain one or more child boxes.
8. A __grid template__ is like an empty table into which data can be flowed.
9. An _inline_ element is designed for laying out text and doesn't disrupt the flow of a document. Examples include boldface and the new HTML5 mark element.
10. Where the flexbox model is suitable for simple things like buttons, toolbars, and many forms, you can use the _Grid Layout_ model for more complex layouts.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. The original W3C CSS Box model does *not* include which of the following?
   a. margin
   b. border
   [✓ c. toolbar]
   d. padding
2. Which of the following is best suited for buttons and toolbars?
   [✓ a. Flexbox Box model]
   b. CSS Box model
   c. Grid Layout model
   d. none of the above
3. You are using CSS to create a flexbox in an HTML document for work. Everyone at work uses the Internet Explorer Web browser. Which prefix should be used with the CSS property names to ensure compatibility while viewing the HTML document?

    a. -moz-
    b. -ms-
    c. -webkit-
    d. -o-

4. Which flexbox property makes child boxes flexible by height and width?
    a. flex
    b. flex-child
    c. flex-wrap
    d. flex-align
5. You want to ensure that extra space in a browser window is distributed equally to the size of all child boxes in a flexbox. Which CSS property should be used?
    a. flex-align
    b. flex-wrap
    c. flex-order
    d. flex-pack
6. Which flexbox property assigns child items to groups to control arrangement within a flexbox?
    a. flex
    b. flex-group
    c. flex-direction
    d. flex-order
7. Which of the following places child items in a grid?
    a. grid-columns
    b. grid-column
    c. grid-flow
    d. grid-pack
8. Which of the following enables you to adapt an HTML document to end-user devices?
    a. Media queries
    b. The CSS Box model
    c. The grid-template property
    d. @import
9. Which of the following is the best use of a grid layout?
    a. Menu
    b. Toolbar
    c. Footer
    d. Game interface
10. What is the primary purpose of a grid template?
    a. To style a grid
    b. To create a table that will hold data

c. To ensure your grid has equal numbers of columns and rows

d. none of the above

True / False

Circle T. if the statement is true or F if the statement is false.

T [✓ F] 1. A flexbox is defined by an element with the CSS properties `display:boxflex` or `display:inline-boxflex`.

[✓ T] F 2. A parent box can contain one or more child boxes.

T [✓ F] 3. You cannot reverse the order of child boxes within a flexbox.

T [✓ F] 4. A flexbox requires an outline or background color.

[✓ T] F 5. An appropriate use for a grid layout is for an online newspaper or a game.

Competency Assessment

Scenario 5-1: Distinguishing between the Flexbox Box Model and the Grid Layout Model

A co-worker named Cynthia is confusing the W3C CSS Flexbox Box model with the Grid Layout model. What do you tell her to clarify both?

Tell Cynthia that the CSS Flexbox Box model is part of the CSS3 draft specification. A flexbox offers flexible layouts for UI design. You can create Web pages and mobile applications with elements, controls, toolbars, menus, and forms that resize and reposition automatically when the user changes the size of the browser window. The browser takes the available space into account and calculates the dimensions for the user, which enables relative sizes and positioning.

Where the flexbox model is suitable for simple things like buttons, toolbars, and many forms, you can use the CSS Grid Layout model for more complex layouts. The grid layout model lets you control the design of sections or entire HTML-based documents using CSS3. As the name implies, a grid layout uses rows and columns to make the design look cleaner and structured.

Scenario 5-2: Understanding Flexboxes and Flexbox Items

Miss Takeet is a teacher at Barely Tall Academy, a private pre-school. She wants to develop a memory game to help her students learn about African animals. The game will include a lot of boxes that contain images and questions. She has decided to use a flexible layout but has no experience with HTML5 or CSS, and she's growing frustrated trying to understand flexboxes and flexbox items. What do you tell Miss Takeet?

Explain to Miss Takeet that the `display: flexbox` CSS property creates the parent box—the flexbox. The `flex` property applies flexibility to child boxes. When the size of a browser window changes, child boxes flex, expanding or contracting along with the parent box.

Proficiency Assessment

Scenario 5-3: Working Around Browser Incompatibility

Ed says that no matter how carefully he checks his CSS code and HTML markup, and has validated his document at the W3C Markup Validation Service Web page, the document doesn't render as expected in his Web browser. Some of the flexbox properties simply don't work. What do you tell Ed?

Explain CSS3 flexbox browser incompatibility to Ed. Like HTML5, the CSS3 specification is still in draft format and undergoing modifications. The names of some properties can change from one version of the CSS3 draft specification to the next, and new property values can be introduced while others are removed.

To help ensure that CSS3 styles work during this transition phase, many of the major Web browsers offer alternative property names. These workarounds simply add a vendor prefix, which is keyword surrounded by dashes, to the front of a CSS3 property name, as follows:

- Internet Explorer uses the -ms- prefix.
- Firefox supports the -moz- prefix.
- Opera uses the -o- prefix.
- Chrome and Safari support the -webkit- prefix.

A good way to determine whether a Web browser can render flexbox properties is to use the controls on the Flexin Web page at http://ie.microsoft.com/testdrive/HTML5/Flexin/Default.html. Click each control to see if the sample flexbox and flexbox items change. For example, some browsers render horizontal justification correctly and others don't. The same applies to flexing child items.

Scenario 5-4: Understanding the Flex Property

Ed is back with another issue. He is experimenting with the flex property to create portions of a Web page. He wants one child box to be double the size of the other child box on the same row. He says he's using a flex value of 2 for the second box but it doesn't render twice as large as the first child box. What do you tell him in order to help him better understand the flex property?

The flex property can be difficult to understand. When Ed applies a flex value of 2 to the second child box, that means the box is twice as flexible as the first box. This doesn't necessarily mean that the second child box will be two times as wide as the first child box. The flex value is a calculation based on available space for stretching or shrinking; the change is assigned based on the portion of flexibility compared to the other child boxes.

Lesson 5: Understanding CSS Essentials: Layouts – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Work with Flexboxes and Flexbox Items

Applying Proportional Scaling within a Flexbox

In the following CSS code and HTML markup, the flexbox contains four flexbox items. Each child has a flex value of `1` and is set to `auto`. When the user changes the size of the browser window, the child boxes should expand and contract along with the parent box.

```
<!doctype html>

<html>

<head>

  <meta charset="utf-8">

  <title>Flexible Child Box Example</title>

  <style>

      div { display: flexbox;
            outline: 2px solid silver

          }

      p {  flex: 1 auto; margin: 1em;

          font-family: sans-serif;

          color: white;

          background: tomato;

          font-weight: bold;

          text-align: center;

        }

  </style>

</head>


<body>

  <div>

    <p>This is the child1 box.</p>

    <p>This is the child2 box.</p>

    <p>This is child3.</p>

    <p>This is child4.</p>

  </div>

</body>
```

```
</html>
```

Figure 5-7 shows the before and after effects of resizing the browser window.

Create a Flexbox with Flexbox Items

GET READY. To learn how to create a flexbox with flexbox items that have a fixed height but a flexible width, perform the following steps:

1. In an editing tool or app development tool, create an HTML file that includes the following CSS code and markup:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Flexible Child Box Example</title>
  <style>
    div { display: flexbox;
          outline: 2px solid silver }
    p { flex: 1 auto; margin: 1em;
        font-family: sans-serif;
        color: white;
        background: limegreen;
        height: 25px;
        padding: 1em;
        font-weight: bold;
        font-size: xx-large;
        text-align: center;
      }
  </style>
</head>

<body>
  <div>
    <p>This is the child1 box.</p>
    <p>This is the child2 box.</p>
    <p>This is child3.</p>
  </div>
</body>
</html>
```

Create Flexbox Items with the flex Function

GET READY. To create flexbox items with the `flex` function and use the `flex-wrap` property, perform the following steps:

1. In an editing tool or app development tool, create an HTML document with the following markup:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Flex Function Example</title>
<style>
    div {
    display: flexbox;
    display: -ms-flexbox;
    display: -moz-flexbox;
    display: -o-flexbox;
    display: -webkit-flexbox;
    flex-wrap: wrap;
    -ms-flex-wrap: wrap;
    -moz-flex-wrap: wrap;
    -o-flex-wrap: wrap;
    -webkit-flex-wrap: wrap;
    height: 200px;
    padding: 1em;
    color: white;
    outline: 2px solid silver;
    }

   div>div {
    width: 75px;
    width: -ms-flex(1 75px);
    width: -moz-flex(1 75px);
    width: -o-flex(1 75px);
    width: -webkit-flex(1 75px);
    margin: 1em;
    height: 100px;
    background-color: #b200ff;
    font-family: sans-serif;
    text-align: center;
    line-height: 100px;
    font-size: xx-large;
   }
</style>
</head>
```

```
<body>
  <div>
    <div>Service 1</div>
    <div>Service 2</div>
    <div>Service 3</div>
  </div>
</body>
</html>
```

Changing the Direction of Child Items in a Flexbox

The `flex-flow` property sets the `flex-direction` and `flex-wrap` properties at the same time. The following example uses the `flex-flow` property with the `column` value.

```
<!doctype html>

<html>

<head>

  <meta charset="utf-8">

  <title>Flex-flow Example</title>

  <style>

   div {

    display: flexbox;

    display: -ms-flexbox;

    display: -moz-flexbox;

    display: -o-flexbox;

    display: -webkit-flexbox;

    flex-flow: column;

    -ms-flex-flow: column;

    -moz-flex-flow: column;

    -o-flex-flow: column;

    -webkit-flex-flow: column;

    height: 400px;

    padding: 1em;

    outline: 2px solid silver;

    color: white;

    font-family: sans-serif;

    font-weight:bold;

  }
```

```
    p {
     width: 100px;
     margin: 1em;
     height: 100px;
     background-color: dodgerblue;
     text-align: center;
     line-height: 100px;
    }
  </style>
 </head>

 <body>
   <div>
     <p>Child1</p>
     <p>Child2</p>
     <p>Child3</p>
   </div>
 </body>
</html>
```

To reverse the order of the child boxes, change each of the `flex-flow` column values to `column-reverse`, as follows:

```
flex-flow: column-reverse;
-ms-flex-flow: column-reverse;
-moz-flex-flow: column-reverse;
-o-flex-flow: column-reverse;
-webkit-flex-flow: column-reverse;
```

Reverse the Order of Flexbox Items

GET READY. To create a flexbox that reverses the order of the flexbox items, perform the following steps:

1. In an editing tool or app development tool, create an HTML document with the following markup:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Flexbox Items Reverse Order Example</title>
  <style>
    div {
      display: flexbox;
      display: -ms-flexbox;
      display: -moz-flexbox;
      display: -o-flexbox;
      display: -webkit-flexbox;
      flex-flow: column;
      -ms-flex-flow: column;
      -moz-flex-flow: column;
      -o-flex-flow: column;
      -webkit-flex-flow: column;
      height: 400px;
      padding: 1em;
      outline: 2px solid silver;
      color: white;
      font-family: sans-serif;
      font-weight: bold;
    }

    p {
      width: 300px;
      margin: 1em;
      height: 100px;
      background-color: olive;
      text-align: center;
      line-height: 100px;
    }
  </style>
  </head>

  <body>
    <div>
      <p>Rock</p>
      <p>Paper</p>
      <p>Scissors</p>
    </div>
  </body>
</html>
```

4. In the HTML file, reverse the order of the columns by using the `flex-flow: column-reverse` value, as follows:

```
flex-flow: column-reverse;

-ms-flex-flow: column-reverse;

-moz-flex-flow: column-reverse;
```

```
-o-flex-flow: column-reverse;
-webkit-flex-flow: column-reverse;
```

5. Resave the file and open it in the Chrome Web browser. The display should look similar to Figure 5-15.

Let’s see how the `flex-order` property works. The following CSS code and markup creates three child boxes in a flexbox:

```
<!doctype html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Flexible Order Example</title>
  <style media="screen">
     div {
      display: flexbox;
      display: -ms-flexbox;
      display: -moz-flexbox;
      display: -o-flexbox;
      display: -webkit-flexbox;
      flex-flow: row;
      -ms-flex-flow: row;
      -moz-flex-flow: row;
      -o-flex-flow: row;
      -webkit-flex-flow: row;
      height: 200px;
      padding: 1em;
      background-color: palegoldenrod;
      font: bold 100%/1 sans-serif;
     }
   div>div {
     width: 100px;
     margin: 1em;
     height: 100px;
     background-color: dodgerblue;
     text-align: center;
     color:  white;
```

```
      font-size:  x-large;
      line-height: 100px;
      }
   </style>
</head>

<body>
  <div>
    <div>Keys</div>
    <div>Phone</div>
    <div>Wallet</div>
  </div>
</body>
</html>
```

The `flex-order` property places child boxes into ordered groups. The default group is 0. You declare groups and assign a number to them in CSS using the `flex-order` property, and any child items not explicitly assigned to a group remain in group 0, and declared groups appear before group 0. So, to reorder the child boxes so that the Keys and Wallet boxes appear before the Phone box, add this code to the bottom of the style section:

```
div>div:first-child,
   div>div:last-child {
      flex-order: 1;
      -ms-flex-order: 1;
      -moz-flex-order: 1;
      -o-flex-order: 1;
      -webkit-flex-order: 1;
      }
```

Creating a Grid Using CSS Properties for Rows and Columns

The following CSS code and HTML markup provide an example of a grid layout. The `–ms–` vendor prefix is included at the beginning of all grid-related constructs because, as of this writing, only Microsoft Internet Explorer 10 supports grid layouts. Internet Explorer 10 is currently available only in the Windows 8 Release Preview operating system. The rendered version is shown in Figure 5-19.

```
<!doctype html>
<html>
<head>
```

```
  <meta charset="utf-8">
  <title>Grid Example</title>
  <style type="text/css">
    #grid {
      background: palegoldenrod;
      border: silver;
      display: -ms-grid;
      color: white;
      font-family: sans-serif;
      font-weight: bold;
      -ms-grid-columns: 150px auto 2fr;
      -ms-grid-rows: 50px 6em auto;
    }
    #logo {
      background: dodgerblue;
      -ms-grid-row: 1;
      -ms-grid-column: 1;
    }
    #item {
      background: olive;
      -ms-grid-row: 2;
      -ms-grid-column: 2;
    }
  </style>
</head>

<body>
  <div>
    <div id="grid">
      <div id="logo">Logo</div>
      <div id="item">Item</div>
    </div>
  </div>
</body>
</html>
```

Lesson 6: Managing Text Flow by Using CSS

Learning Objectives

Students will learn about:

- Managing the flow of text content by using CSS
- Using regions to flow text content between multiple sections

ODN Skills

Manage the flow of text content by using CSS.	3.3

Lesson Summary — Lecture Notes

Note: See specific notes throughout this section regarding which Web browsers were used to display HTML5 text flow features.

Lesson 6 introduces students to the concepts of text/content flow using CSS.

Begin by asking students if they have used page layout software such as Microsoft Publisher or Adobe InDesign. Also ask if they have used columns in a word processor. Then describe the challenges of flowing content between areas of an HTML document similar to the functionality of page layout software or columns. Explain that many of the new HTML5 features covered in this lesson help developers gain that kind of content flow in HTML documents.

The first section covers CSS Regions. Explain the concepts of a content source, content containers, and a named flow, and the `flow-from` and `flow-into` CSS properties. Point out that Microsoft uses its own implementation of CSS Regions, which uses the `-ms-flow-from` construct. Also describe the purpose and functionality of an iframe.

Note: The CSS Regions specification is a work in progress and has not been widely adopted by browser vendors. As of this writing, Internet Explorer 9 does not support CSS Regions; Internet Explorer 10 provides some support for CSS Regions using the `-ms-flow-from` construct. Chrome partially supports CSS Regions with the `-webkit-` prefix, but you must first enable the feature in the browser's Properties dialog box per the CSS Regions exercise in the textbook. Consult the When Can I Use Web page at caniuse.com to check the status of Web browser support.

Next, discuss multi-column layouts. Explain how to use the `column-count`, `column-gap`, and `column-rule` CSS properties. The textbook used Mozilla Firefox to show examples of multi-column layouts.

Then discuss the purpose of hyphenation and the CSS3 `hyphens` property. Explain the values `none`, `manual`, and `auto`. Emphasize that the W3C requires you to declare a language using the HTML `lang` or XML `xml:lang` attributes for correct automatic hyphenation to occur. Be sure to review the Microsoft-specific hyphenation properties:

`-ms-hyphenate-limit-zone`, `-ms-hyphenate-limit-chars`, and `-ms-hyphenate-limit-lines`.

Note: Because of support issues for hyphenation in Internet Explorer 9 and 10, the examples in the book were tested using the Firefox Web browser.

In the next section, discuss CSS Exclusions, which were formerly referred to as positioned floats. Students might get excited about CSS Exclusions because of the unique nature in which they can display text set off from the rest of the content. The book shows a square and a circle in the examples, but CSS Exclusions can take on almost any type of shape. Search for CSS Exclusions on the Web and show the students some examples.

Then, explain that CSS Exclusions use the `wrap-flow:both` property to display content on all sides of the exclusion. The `wrap-flow:clear` property displays content above and below the exclusion but leaves the sides blank. You declare an exclusion shape using the `shape-inside` and `shape-outside` properties, which define the content and the general shape of an exclusion, respectively. All properties require the `-ms-` vendor prefix to be viewed in Internet Explorer 10.

Note: The CSS Exclusions specification is preliminary as of this writing. The authors were not able to reliably display a CSS Exclusion in any of the major Web browsers. The Microsoft Exclusions Web site states that “the properties and syntax defined in the CSS Exclusions specification might not correspond exactly to those described in this topic. As development on the specification continues, this might change.”

Key Terms

content container - Using CSS Regions, a content container is an area into which content is flowed.

content source - Using CSS Regions, a content source may be one or more blocks of text in the same or a separate HTML document that holds the content you want to flow through a layout. The content is referred to as a “content stream.”

CSS Exclusions - CSS Exclusions are another name for positioned floats.

CSS Regions - CSS Regions are defined areas (regions) of an HTML document where content can flow. Similar to a page layout program, when there’s too much content to fit in one region, the remaining content automatically flows into the next region.

`flow-from` - The `flow-from` property specifies one or more content containers.

`flow-into` - The `flow-into` property specifies a content source.

hyphenation - Hyphenation is the process of connecting two words with a hyphen mark (-) or breaking words between syllables at the end of a line.

iframe - Iframes are like mini boxes on a Web page that contain external content embedded in an HTML document, as the content source.

multi-column layout - Multi-column layout lets you create columns, divide text across multiple columns, specify the amount of space that appears between columns (the gap), make vertical lines (rules) appear between columns, and define where columns break. You create a multi-column layout using CSS3 properties.

named flow - A named flow is a set of elements taken from the source and to be flowed into a content container.

positioned float - A positioned float is a CSS construct that enables you to position images, text, and boxes anywhere in an HTML document and then wrap text completely around these elements.

Lesson 6: Managing Text Flow by Using CSS- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. _CSS Regions_ are defined areas of an HTML document where content can flow. They're used instead of multiple columns in more complex layouts.
2. CSS3 properties for _multi-column layout_ let you create columns by dividing text across multiple columns, specify the amount of space that appears between columns (the gap), make vertical lines (rules) appear between columns, and define where columns break.
3. A __content source__ may be one or more blocks of text in the same or a separate HTML document that holds the content you want to flow through a CSS Regions layout.
4. __Content containers__ are the part of CSS Regions into which content is flowed.
5. __Hyphenation__ is the process of connecting two words with a hyphen mark (-) or breaking words between syllables at the end of a line.
6. A __CSS Exclusion__ is a positioned float that enables you to position images, text, and boxes anywhere in an HTML document and then wrap text completely around these elements.
7. The __`flow-from`__ CSS property creates a content container for CSS Regions.
8. The __`flow-into`__ CSS property identifies the content source for CSS Regions.
9. A(n) __iframe__ is a mini HTML document embedded in an HTML document.
10. The value of the `flow-into` property is called a __named flow__.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. You are creating a CSS Regions content source named "main." Which of the following is the correct syntax?

   a. `flow-from: main`

   b. flow-into: main

   c. `main: flow-into`

   d. `main: flow-from`

2. You are creating a CSS Regions content container to be associated with a content source named "main." Which of the following is the correct syntax?

   a. flow-from: main

   b. `flow-into: main`

   c. `main: flow-into`

   d. `main: flow-from`

3. What are the options for handling overflow text in the last container of a CSS region? (Choose all that apply.)

   a. Truncation

b. Continue overflowing and be visible

c. Continue overflowing but be hidden

d. Duplication

4. How does Microsoft's implementation of CSS Regions differ from the W3C's specification?

a. Microsoft uses the `flow-into` property.

b. Microsoft uses the flow-from property.

c. Microsoft does not use iframes.

d. Microsoft uses iframes.

5. Which CSS3 property creates scalable columns?

a. column-count

b. `add-columns`

c. `wrap-columns`

d. none of the above

6. Which CSS3 property creates a line between columns in a multi-column layout?

a. `break-inside`

b. `column-fill`

c. `column-gap`

d. column-rule

7. Which of the following is *not* a legal value for the CSS3 hyphens property?

a. `none`

b. lines

c. `manual`

d. `auto`

8. Which of the following specifies the width of the trailing whitespace that can be left in a line before hyphenation occurs?

a. `-ms-hyphenate-limit-chars`

b. `-ms-hyphenate-limit-lines`

c. -ms-hyphenate-limit-zone

d. none of the above

9. Which of the following is the formerly used term for CSS Exclusions?

a. left/right floats

b. positioned floats

c. shape changer

d. the DOM

10. Which CSS3 property creates a CSS exclusion?

a. wrap-flow

b. `flow-wrapper`

c. `shape-wrapper`

d. `wrapper-shape`

True / False

Circle T if the statement is true or F if the statement is false.

T [✓ F] 1. A CSS Exclusion must be either rectangular or circular in shape.

[✓ T] F 2. You must declare a language using the HTML `lang` or XML `xml:lang` attributes for correct automatic hyphenation to occur.

[✓ T] F 3. You can center a heading across multiple columns using the `column-span: all` property.

T F 4. You combine CSS Regions with CSS layout techniques, such as columns, flexboxes, and grid layouts.

[✓ T] F 5. In CSS Regions, the value for the `flow-from` property must match the value of the `flow-into` property.

Competency Assessment

Scenario 6-1: Flowing Content in a Newsletter

Changpu is fellow intern at Malted Milk Media. He was asked by his manager to create a newsletter layout using HTML5 and CSS3. The newsletter will be viewed by employees using PCs, tablets, and possibly smartphones. Changpu created a template with sections, articles, a header and footer, and asides. He wants some items to automatically move overflow content to different areas of the layout, but he's not sure how to do it. How do you advise him?

Tell Changpu that in a typical HTML document, he can display content in different sections or areas, but each area is independent. If he wants overflow text to move from one area to another, he generally has to do so manually. This approach doesn't work well when a user resizes the screen or uses accessibility tools such as a screen magnifier. This method also doesn't lend itself to automatic switching from portrait to landscape orientation on tablets and smartphones. Changpu should use CSS Regions or multi-column layout, depending on the complexity of the project.

Scenario 6-2: Distinguishing between Content Source and Content Containers

Changpu decided to use CSS Regions to provide dynamic content flow in his newsletter, but he's confused about content sources and content containers. How do you explain the two features to Changpu?

A content source may be one or more blocks of text in the same or a separate HTML document that holds the content you want to flow through a layout. The content is referred to as a "content stream."

You also need content containers, which are the areas into which content is flowed. An HTML document with content containers acts as a master page, like a template, in which each container is sized and positioned where you want content to appear, but each container is initially empty.

Proficiency Assessment

Scenario 6-3: Understanding Hyphenation Requirements

Salih is a friend of yours from college. He is creating an HTML document that has content in Arabic for his language skills group, and he wants the document to be hyphenated. He used the hyphens property and is viewing the document in a supported Web browser, but the hyphenation isn't working. Is there something else Salih needs to do to the document?

The W3C points out that you must declare a language using the HTML `lang` or XML `xml:lang` attributes for correct automatic hyphenation to occur. That means if Salih's entire HTML document is in Arabic, he must add the appropriate attribute to his html element or `doctype` declaration.

The language code for Arabic is `ar`, so Salih would use:

```
<!doctype html>
<html lang="ar">
```

or

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ar"
lang="ar">
```

Scenario 6-4: Understanding CSS Exclusions Essentials

Gladys just transferred to the Web and application development team from the Web site maintenance team. She is just beginning to research CSS Exclusions and saw you walking past her work area. She stops and asks you for a jumpstart on basic information on CSS Exclusions properties.

Tell Gladys that to create a simple CSS exclusion, she can use the `wrap-flow:both` property to display content on all sides of an exclusion. Another option is to use `wrap-flow: clear`, which displays content above and below the exclusion but leaves the sides blank.

An exclusion shape is declared using the `shape-inside` and `shape-outside` properties, which define the content and the general shape of an exclusion, respectively.

Lesson 6: Managing Text Flow by Using CSS – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Flowing Content through Containers Dynamically

The following example begins with some CSS code for a content source. The `flow-into` value is "main," which is the named flow. Because the content will actually appear in another place (in the content containers), this source element itself will not be displayed on the HTML page.

```
<style>
#source {
   flow-into: main;
}

.region {
    flow-from: main;
    background: #9ACD32;
}
</style>

<body>
<div id="source">
    <p>Lorem ipsum dolor ...</p>
</div>
  <div id="region1" class="region"></div>
  <div id="region2" class="region"></div>
  <div id="region3" class="region"></div>
</body>
```

Create CSS Regions

GET READY. To create CSS Regions, perform the following steps:

1. In an editing tool or app development tool, create an HTML document that includes the following content:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>CSS Regions Example</title>
<style type="text/css">

   body, html { height:100%; width:100%; }

   body{
      font-family: serif;
      color: black;
      font-size: large;
}

   #source{
      -webkit-flow-into: main;
   }

   .region{
      -webkit-flow-from: main;
      margin: 0 25px 0 0;
      background: #EEE8AA;
      padding: 20px;
   }

   #region1{
      width: 20%;
      height: 50%;
      float: left;
   }

   #region2{
      width: 20%;
      height: 50%;
      float: left;
   }

   #workarea{
      position: relative;
      padding: 25px;
   }

   </style>
</head>

<body>

<div id="source">
```

```
  <p>Lorem ipsum dolor . . . mollis a ipsum.</p>
</div>

<div id="workarea">
  <div id="region1" class="region"></div>
  <div id="region2" class="region"></div>
</div>

</body>
</html>
```

Using Columns and Hyphenation to Optimize the Readability of Text

Creating Columns

Now let's look at how the CSS code works. This code uses the column-count property to create three columns using the text in the HTML markup that follows:

```
<head>
  <style>
    .tricolumn { column-count: 3; }
  </style>
</head>


<body>
<h2>My Three Columns</h2>
<div class="tricolumn">
Lorem ipsum . . . orci.
</div>
</body>
```

Because CSS3 columns are still a work in progress, you may need to add vendor prefixes to column-related property names. In this case, we modified the code as follows for all four of the major browsers:

```
  <style>
    .tricolumn {
      -ms-column-count: 3;
      -moz-column-count: 3;
      -o-column-count: 3;
      -webkit-column-count: 3;
    }
</style>
```

Create a Multi-Column Layout

GET READY. To create a multi-column layout, perform the following steps:

1. In an editing tool or app development tool, create a proper HTML document that includes the following content:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Three Columns</title>
  <style>
    .tricolumn {
    -moz-column-count: 3;
    }
  </style>
</head>

<body>
<h2>My Three Columns</h2>
<div class="tricolumn">
Lorem ipsum . . . orci.
</div>
</body>
</html>
```

Enable Automatic Hyphenation

GET READY. To enable automatic hyphenation, perform the following steps:

1. In an editing tool or app development tool, create an HTML document that includes the following content. Notice that we’re using the Mozilla vendor prefix as an example in this exercise. Replace the paragraph text that begins with “Hyphenation is” with a four-line biography about yourself, your instructor, a classmate, or a celebrity:

```
<!doctype html>
<html lang="en-us">
<head>
    <meta charset="utf-8" />
    <title>Hyphenation Example</title>
</head>
<body>
```

```
  <div style="width: 200px;
  border: 2px solid orange;">
  <p style="-moz-hyphens: auto;
   text-align: justify;
   font-size: 14pt;">
   Hyphenation is . . . professional.</p>
</body>
</html>
```

Lesson 7: Managing the Graphical Interface by Using CSS

Learning Objectives

Students will learn about:

- Creating graphics effects
- Applying transparency
- Applying background gradients
- Understanding typography and the Web Open Font Format
- Applying 2D and 3D transformations
- Creating transitions and animations
- Applying SVG filter effects
- Using canvas to enhance the GUI

ODN Skills

Manage the graphical interface by using CSS.	3.4

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to show about half of the examples in Lesson 7. A Webkit browser was used to show perspective, transition, animation, and SVG examples, which didn't render properly in Internet Explorer 9 or 10.

Lesson 7 introduces students to graphical user interface elements created with CSS, from rounded box corners and drop shadows, to 2D and 3D transformations like rotations and scaling, to transitions and animations.

Begin the lesson by discussing the `border-radius` property, which creates rounded corners. Point out that `border-radius` is a length, which is usually expressed in pixels or ems but can be a percentage.

Next, explain how to use the `box-shadow` property to create drop-shadows. , The `h-shadow` and `v-shadow` attributes indicate the horizontal and vertical position of the shadow in relation to the box. Both of these attributes are required. The `blur`, `spread`, `color`, and `inset` properties are optional.

Discuss opacity and transparency. An opaque item does not let light pass through, whereas you can see through a transparent item. The syntax for applying a transparency to an image or other element is `opacity: value`. The value is a floating-point value between 0.0 (100% transparent) and 1.0 (100% opaque).

Next, discuss CSS3 gradients. The four methods are `linear-gradient`, `radial-gradient`, `repeating-linear-gradient`, and `repeating-radial-gradient`. You use these methods with the `background` property. Show students how to create gradients using the same color and different colors. Be sure to discuss color interpolation in the alpha color space, which is part of the red blue green alpha (RGBA) color model. Explain color stops and how to use multiple color stops with an RGBA color.

The next section addresses fonts. Fonts and font families were introduced in Lesson 4. For this lesson, explain Web-safe fonts and their design limitations. Then explain the purpose of the Web Open Font Format (WOFF) and how to use WOFF fonts in an HTML document.

In the next section, discuss 2D and 3D transformations. This section can be a lot of fun for students. Explain that a transform is an effect that lets you change the size, shape, and position of an element. You should also explain some of values listed in Table 7-1 for the `transform` property. Be sure to mention that to see the "action" of a transformation requires JavaScript; using only CSS shows the before and after effects of properties and their values.

Next, explain translations and how to use the `translate` method with the `transform` property. Continue by explaining scaling, rotation, and skewing. Provide examples for each method.

Move on to perspective by explaining that the CSS3 3D `perspective` property defines how a browser renders the depth of a 3D transformed element. The `perspective` property takes on a number value: lower values (in the range of 1 to 1000) create a more pronounced effect than higher values. Demonstrate a perspective example where you change the value three or four times to see the differences in the affected object or text.

Discuss transitions, explaining that it's a change from one thing to another. In CSS3, the action of a transition renders onscreen—no JavaScript is needed! Explain that the `transition` property requires the CSS property to be acted upon during the transition, and describe some of the transition properties listed in Table 7-2.

Next, discuss animations. Point out that transitions and animations are similar in that they both use timings. An important difference is that animations use keyframes, which give you control over parts of the animation. Explain how to use the @keyframes rule, examine some of the properties in Table 7-3, and then demonstrate an animation.

The last two sections in the lesson address SVG filters and canvas, which were introduced in Lesson 2.

Explain that an SVG filter is a set of operations that use CSS to style or otherwise modify an SVG graphic. The enhanced graphic is displayed in a browser while the original graphic is left alone. Describe how to define an SVG filter using the `filter` element and the `id` attribute to name the filter. Demonstrate two or three of the filters listed in Lesson 7.

Finally, describe how students can apply stylistic effects to canvas drawings. The code must include instructions for drawing the canvas object and apply effects such as color, gradients, rotations, and so on.

Key Terms

animation - An animation is the display of a sequence of static images at a fast enough speed to create the illusion of movement.

`border-radius` property - The CSS3 `border-radius` property creates rounded corners around layout elements, like headers, footers, sidebars, graphics boxes, and outlines around images. `border-`

`radius` is a length, which is usually expressed in pixels or ems but can be a percentage. The length is the radius of the circle that defines the “roundedness” of each box corner

drop shadow - A drop shadow is a visual effect in which an object is repeated behind and slightly below itself to create the illusion that the object floats over its background.

gradient - A gradient is a smooth change of colors, either within the same hue, such as from dark green to light green, or starting with one color and ending with a different color, such as starting with blue and ending with yellow.

keyframe - A keyframe is a construct that enables you to change values anywhere within an animation. You can also pause, resume, and reverse animations.

linear gradient - A linear gradient is a horizontal, vertical, or diagonal gradient.

opacity - Opacity is the quality of an object, like an image or rectangle, that does not allow light to shine through; an opaque image is not transparent.

perspective - Perspective, in terms of drawings and illustrations, is the convergence of lines that give the illusion of depth.

radial gradient - Radial gradients start from a central point and radiate color out to the edges of a container.

rotate - To rotate an element turns it clockwise by a specified number of degrees. To rotate an element, you use the `rotate()` method in CSS and specify the degrees of rotation.

scale - To scale an element is to increase or decrease its size.

skew - To skew an element is to stretch it in one or more directions. To skew an element using CSS, you use the `skew()` method and provide x-axis and y-axis values, in degrees, to create an angular shape.

SVG filter - An SVG filter is a set of operations that use CSS to style or otherwise modify an SVG graphic. The enhanced graphic is displayed in a browser while the original graphic is left alone.

transform - In HTML5/CSS3, a transform is an effect that lets you change the size, shape, and position of an element.

transition - A transition is a change from one thing to another; in CSS, a transition is the change in an element from one style to another.

translate - To translate an element means to move it, without rotating, skewing, or otherwise turning the image.

transparency - Transparency is reduced opacity.

Web Open Font Format (WOFF) - The Web Open Font Format (WOFF) allows Web developers to use custom fonts instead of being limited to the standard Web fonts. WOFF files are compressed True Type, OpenType, or Open Font Format fonts that contain additional metadata.

Web safe - Web safe refers to a set of standard fonts that are typically located on a user’s computer and render consistently in the majority of Web browsers.

Lesson 7: Managing the Graphical Interface by Using CSS- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. A __gradient__ is a smooth change of colors, either within the same hue or starting with one color and ending with another.
2. To __scale__ an element is to increase or decrease its size.
3. A __drop shadow__ is a visual effect in which an object is repeated behind and moved slightly below itself to create the illusion that the object floats over its background.
4. A __transition _ is a change from one thing to another; in CSS, it is the change in an element from one style to another.
5. The __Web Open Font Format (WOFF)_ enables Web developers to use custom fonts—pretty much any font—instead of being limited to the standard Web fonts.
6. The __`border-radius`__ CSS3 property enables you to create rounded corners around layout elements, such as headers, footers, sidebars, graphics boxes, and outlines around images.
7. __Transparency__ is reduced opacity.
8. To __translate__ an element means to move it, without rotating, skewing, or otherwise turning the image.
9. In HTML5/CSS3, a __transform__ is an effect that lets you change the size, shape, and position of an element.
10. An SVG __filter__ is a set of operations that use CSS to style or otherwise modify an SVG graphic.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which of the following creates a gradient from top to bottom, left to right, or from corner to corner, without reiterating colors?
   a. linear-gradient
   b. `radial-gradient`
   c. `repeating-linear-gradient`
   d. `repeating-radial-gradient`
2. Which of the following is *not* true of the border-radius property?
   a. It creates rounded corners around layout elements.
   b. It can be expressed in pixels.
   c. It can be expressed as a percentage.
   d. It can animate an object.
3. To apply a 60% transparency to an image or element, which property do you use?

   a. `opacity: 40`
   b. opacity: 0.4
   c. `transparency: 40`
   d. `transparency: 0.4`
4. Which of the following are disadvantages of Web-safe fonts? (Choose all that apply.)
   a. They must be loaded on a Web server.
   b. They are limited in number and variety.
   c. They make brand identity difficult to achieve on the Web.
   d. They are expensive.
5. Keyframes are associated with which of the following?
   a. Rounded corners
   b. Transitions
   c. Animations
   d. None of the above
6. When creating a transition, which of the following must be specified?
   a. A start delay
   b. The CSS property to be acted upon during the transition
   c. The transition timing function
   d. The keyframe
7. What is a primary advantage to using color interpolation in the alpha color space?
   a. It produces smoother color transitions in gradients.
   b. It enables you to add color to SVG drawings.
   c. Both a and b
   d. Neither a nor b
8. Which of the following do you use to add color to canvas text?
   a. fillStyle
   b. `strokeStyle`
   c. `textColor`
   d. `strokeColor`
9. What are the two play states of an animation?
   a. started
   b. running
   c. paused
   d. resumed
10. In the following code sample, what controls the amount of blur?

```
<defs>
<filter id="a1" x="0" y="0">
<feGaussianBlur in="SourceGraphic"
stdDeviation="20" />
```

```
</filter>
</defs>
```

a. `feGaussianBlur`

b. `SourceGraphic`

[✓ c. stdDeviation]

d. none of the above

True / False

Circle T if the statement is true or F if the statement is false.

[✓ T] F 1. An opaque item does *not* let light pass through, whereas you can see through a transparent item.

T [✓ F] 2. In CSS, to rotate an element turns it counterclockwise by a specified number of degrees.

[✓ T] F 3. An animation is the display of a sequence of static images at a fast enough speed to create the illusion of movement.

[✓ T] F 4. Radial gradients start from a central point and radiate color out to the edges of a container.

[✓ T] F 5. Perspective, in terms of drawings and illustrations, is the convergence of lines that give the illusion of depth.

Competency Assessment

Scenario 7-1: Troubleshooting CSS3 Code

Ali is a vintage car enthusiast who restores old cars and sells them for a profit. He posts photos and descriptions on his Web site, which he maintains himself. Ali is transitioning to HTML5 and CSS3 as much as possible. He has been trying to apply the translate and scale transformations to a 2D image on his site but neither transformation is working. This is the code he's using:

```
<style>

img { transform: translate(100px,50px);

transform: scale(2,4); }

</style>
```

What advice do you give to Ali?

The code looks properly written, so suggest that Ali add vendor prefixes to the transform property, like this:

```
<style>
img { transform: translate(100px,50px);
      -moz-transform: translate(100px,50px);
      -ms-transform: translate(100px,50px);
      -o-transform: translate(100px,50px);
      -webkit-transform: translate(100px,50px);
      transform: scale(2,4);
      -moz-transform: scale(2,4);
      -ms-transform: scale(2,4);
```

```
        -o-transform: scale(2,4);
        -webkit-transform: scale(2,4);
      }
</style>
```

Scenario 7-2: Displaying Before and After Images

Ali's wife Linda thinks his vintage car Web site could use some enhancements to attract more visitors. One of her suggestions is to show before and after photos of the cars that Ali has restored. They ask you what options they have using CSS3 to make it easy on users to see the photos. What do you tell them?

Tell Ali and Linda that they can use transitions or animations to show before and after photos of the vintage cars. A transition would be triggered by a mouse-over or click; the before photo could fade away as the photo of the restored car comes into view. An animation would work similarly but could loop continuously.

Proficiency Assessment

Scenario 7-3: Creating Buttons with Enhancements

Edward is creating a set of buttons for a Web application and wants the buttons to have rounded corners and a small drop shadow that's slightly blurred. He asks you which CSS3 properties to use. What do you tell him?

Suggest that Edward use the CSS3 `border-radius` property to achieve rounded corners and the `box-shadow` property for the drop shadow with blur. The code would look similar to the following, which Edward could modify as needed:

```
border-radius: 8px;

box-shadow: 5px 5px 5px #999;
```

Remind Edward that he should use vendor prefixes to ensure the widest browser compatibility. You could also suggest that he consider using SVG to achieve the same enhancements, which would easily scale to varying screen sizes.

Scenario 7-4: Understanding 3D Perspective

Meghan is a university student who is studying for a fine arts degree. She's learning about digital graphics in one of her courses, and a fellow student said he's interested in the CSS3 3D perspective. She hadn't heard of it before, so she asks you to briefly explain it to her. What do you tell her?

The CSS3 3D perspective property defines how a browser renders the depth of a 3D transformed element. The `perspective` property takes on a number value: lower values (in the range of 1 to 1000) create a more pronounced effect than higher values. Another important thing to remember is that perspective applies only to 3D transformed elements.

Lesson 7: Managing the Graphical – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Creating Graphics Effects

Creating Rounded Corners

To create a box with a rounded border, the CSS code and markup might look like this:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
<title>Rounded Corners</title>
<style type="text/css">
div {
  padding: 40px 40px;
  background: dodgerblue;
  width: 400px;
  color: #fff;
  font-family: sans-serif;
  font-size: xx-large;
  border-radius: 25px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 100px;
}
</style>
</head>

<body>
  <div>A box with rounded corners</div>
</body>
</html>
```

You can also round a single corner of a box using the following properties:

- `border-top-left-radius`

- `border-top-right-radius`
- `border-bottom-right-radius`
- `border-bottom-left-radius`

If you plan to use single rounded corners on several elements in an HTML document, you can save time by creating a separate class for each (top left, top right, bottom left, and bottom right). The syntax would look similar to the following:

```
.top-left-corner { border-top-left-radius:25px; }
```

Create a Box with Rounded Corners and a Shadow

GET READY. To create a box with rounded corners and a shadow, perform the following steps:

1. In an editing or app development tool, create an HTML document that includes the following content:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Rounded Corners</title>
<style type="text/css">
div {
  border: 3px solid #000;
  background-color: #000;
  padding: 1em;
  width: 300px;
  border-radius: 8px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 100px;
  color: #fff;
  font-family: sans-serif;
  font-size: large;
  text-align: center;
}
</style>
</head>

<body>
  <div>A box example</div>
</body>
</html>
```

2. Save the file as **L7-box-exercise.html**. View the file in a Web browser.

Apply a Background Gradient to a Box

GET READY. To apply a background gradient to a box, perform the following steps:

2. Add the following lines to the style element:

```
background: linear-gradient(black,white);
background: -ms-linear-gradient(black,white);
background: -moz-linear-gradient(black,white);
background: -o-linear-gradient(black,white);
background: -webkit-linear-gradient(black,white);
```

Translate and Scale a 2D Shape

GET READY. To translate and scale a 2D shape, perform the following steps:

1. In an editing or app development tool, create an HTML document that includes the following content:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
<style type="text/css">
div
{
padding: 20px 20px;
background: tomato;
width: 150px;
height: 75px;
color: #fff;
font-family: sans-serif;
font-size: x-large;
}
</style>
</head>

<body>
<div>This element can move</div>
</body>
```

3. Add the following lines to the style element:

```
transform: translate(200px,100px);
-ms-transform: translate(200px,100px);
-moz-transform: translate(200px,100px);
-o-transform: translate(200px,100px);
-webkit-transform: translate(200px,100px);
```

5. To scale the box so that it's twice as wide and twice as tall as the original box, add **`scale(2,2)`** to the transform lines, as follows:

```
transform: translate(200px,100px) scale(2,2);
-ms-transform: translate(200px,100px) scale(2,2);
-moz-transform: translate(200px,100px) scale(2,2);
```

```
-o-transform: translate(200px,100px) scale(2,2);
-webkit-transform: translate(200px,100px) scale(2,2);
```

Rotate and Skew a 2D Shape

GET READY. To rotate and skew a 2D shape, perform the following steps:

2. Modify the transform lines to replace scale with a 30-degree rotation, as follows:

```
transform: translate(200px,100px) rotate(30deg);

-ms-transform: translate(200px,100px) rotate(30deg);

-moz-transform: translate(200px,100px) rotate(30deg);

-o-transform: translate(200px,100px) rotate(30deg);

-webkit-transform: translate(200px,100px) rotate(30deg);
```

Understanding 3D Perspective, Transitions, and Animations

The general syntax for perspective is:

perspective: *number*;

An example of perspective with a 3d rotation applied is:

```
perspective: 600; margin: 100px 0 0 50px;

transform: rotate3d(0, 1, 1, 45deg);
```

The following is a complete markup example for perspective and is shown in Figure 7-21 using a webkit-supported browser.

```
<!doctype html>

<html>

<head>

<style type="text/css">

div

{

padding: 40px 40px;

background: #B8860B;

width: 150px;

color: #fff;

font-family: sans-serif;

font-size: xx-large;

}

div#div2

{
```

```
margin: 100px;
perspective: 600; margin: 100px 0 0 50px;
-ms-perspective: 600; margin: 100px 0 0 50px;
-moz-perspective: 600; margin: 100px 0 0 50px;
-o-perspective: 600; margin: 100px 0 0 50px;
-webkit-perspective: 600; margin: 100px 0 0 50px;

transform: rotate3d(0, 1, 1, 45deg);
-ms-transform: rotate3d(0, 1, 1, 45deg);
-moz-transform: rotate3d(0, 1, 1, 45deg);
-o-transform: rotate3d(0, 1, 1, 45deg);
-webkit-transform: rotate3d(0, 1, 1, 45deg);
}
</style>
</head>

<body>
  <div>Original element</div>
  <div id="div2">Transformed element</div>
</body>
</html>
```

The following is an example of a simple transition that displays text in a box. When a user hovers the mouse pointer over the box, the text changes. Figure 7-22 shows the before and after boxes.

```
<style type="text/css">
#wrapper { transition-property: opacity;
           transition-duration: 3s;
           transition-delay: 1s;
           transition-timing-function: linear; }
#wrapper #before, #wrapper:hover #after {
  opacity: 1; }
#wrapper:hover #before, #wrapper #after {
  opacity: 0; }
</style>
</head>
```

```
<body>
<div id="wrapper">
  <div id="before">Now you see me</div>
  <div id="after">Now you don't</div>
</div>
</body>
```

Create a Transition Using CSS

GET READY. To create a transition using CSS, perform the following steps:

1. In an editing or app development tool, create an HTML document that includes the following content:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Transition and Opacity Example</title>
<style type="text/css">
#wrapper {
padding: 40px 40px;
background: dodgerblue;
width: 200px;
font-family: sans-serif;
font-size: xx-large;
margin-left: auto;
margin-right: auto;
margin-top: 100px;
color: #fff;
}

#wrapper #front, #wrapper:hover #back {
opacity:1;
-ms-opacity: 1;
-moz-opacity: 1;
-o-opacity: 1;
-webkit-opacity: 1;

transition-property: opacity;
-ms-transition-property: opacity;
-moz-transition-property: opacity;
-o-transition-property: opacity;
-webkit-transition-property: opacity;
transition-duration: 1s;
-ms-transition-duration: 1s;
-moz-transition-duration: 1s;
-o-transition-duration: 1s;
-webkit-transition-duration: 1s;

transition-timing-function: linear;
-ms-transition-timing-function: linear;
-moz-transition-timing-function: linear;
-o-transition-timing-function: linear;
-webkit-transition-timing-function: linear;
}

#wrapper:hover #front, #wrapper #back {
opacity: 0;
-ms-opacity: 0;
-moz-opacity: 0;
-o-opacity: 0;
-webkit-opacity: 0;
}
</style>
</head>

<body>
<div id="wrapper">
  <div id="front">Knock knock</div>
  <div id="back">Who's there?</div>
</div>

</body>
</html>
```

To create an animation, you begin by specifying a CSS style within the `@keyframes` rule. For example, a rule for a fadeout might look like the following:

```
@keyframes fadeout {
from { opacity: 1; }
to { opacity: 0; }
}
```

The following is a snippet of code that configures animation properties for a fadeout:

```
div { animation-duration: 3s;
      animation-delay: 0s;
      animation-timing-function: ease; }
div:hover { animation-name: fadeout; }
```

This fadeout starts immediately when the user hovers the mouse pointer over a `div` element, lasts for three seconds, and uses and ease timing function.

Create an Animation Using CSS

GET READY. To create an animation using CSS, perform the following steps:

1. In an editing or app development tool, create an HTML document that includes the following content:

```
<style type="text/css">
div { width: 200px;
      height: 200px;
      background: limegreen;
      animation: a1 3s; }

@keyframes a1 { from {background: limegreen;}
                to {background: dodgerblue;} }
</style>
</head>

<body>
<div></div>
</body>
```

Applying SVG Filter Effects

You use the filter element to define SVG filters; you must include the id attribute to name the filter. For example, the following code is an example `feGaussianBlur` filter, the results of which are shown in Figure 7-24:

```
<body>
<svg>
  <defs>
    <filter id="a1" x="0" y="0">
      <feGaussianBlur in="SourceGraphic"
       stdDeviation="20" />
    </filter>
  </defs>
  <rect width="150" height="150" stroke="plum"
   stroke-width="3" fill="plum" filter="url(#a1)" />
</svg>
</body>
```

In another example, the following `feOffset` filter creates a drop shadow under a rectangle. A shadow fits the "action" of the offset filter because a shadow is merely a box that has been moved down and to the right of the source image. The `dx` and `dy` attributes specify the amount to move the image along the x-axis and y-axis, respectively. The rendered image is shown in Figure 7-25.

```
<body>

<svg>

  <defs>

    <filter id="i1" x="0" y="0">

    <feOffset dx="5" dy="5" />

    </filter>

  </defs>

  <rect width="150" height="150" fill="grey"
   filter="url(#i1)" />

  <rect width="150" height="150" fill="plum" />

</svg>

</body>
```

Apply an Offset and Gaussian Blur to an SVG Drawing

GET READY. To apply an offset and Gaussian blur to an SVG drawing, perform the following steps:

1. In an editing or app development tool, create an HTML document with the following content:

```
<!doctype html>

<html>

<head>

    <meta charset="utf-8" />

    <title>SVG Offset and Gaussian Blur Example</title>

</head>

</style>


<body>

<svg>

  <defs>

    <filter id="i1" x="0" y="0">

    <feOffset dx="5" dy="5" />

    </filter>

  </defs>
  <rect width="150" height="150" fill="grey"
   filter="url(#i1)" />

  <rect width="150" height="150" fill="springgreen"
  />

</svg>

</body>

</html>
```

Using Canvas to Enhance the GUI

The following code creates a basic canvas box:

```
<script>

   function f1() {

       var canvas =

       document.getElementById("smlRectangle");

       context = canvas.getContext("2d");

       context.fillStyle = "blue";

       context.fillRect(10, 20, 200, 100);

   }

  </script>


<body onload = "f1();">

<canvas id="smlRectangle" height='300' width='500'>

</canvas>

</body>
```

This sample already uses the `fillStyle` attribute for the `getContext("2d")` object. Let's change the color to apply a gradient with coral as the start color and khaki as the end color. Replace the `fillStyle` line with the following, which render as shown in Figure 7-28:

```
var grd=context.createLinearGradient(0,0,150,0);

grd.addColorStop(0.3,"coral");

grd.addColorStop(0.7,"khaki");

context.fillStyle=grd;
```

To rotate the canvas, you use the formula `*degrees*Math.PI/180`. You must also add the rotation before the rectangle is generated. So, to rotate our canvas 20 degrees, add the following line before the `context-fillRect` line:

```
context.rotate(20*Math. PI/180);
```

The result is shown in Figure 7-29. You can also translate (move), scale, and skew the object similar to transforming HTML elements.

Finally, let's see how to generate text by using canvas. You use the `fillText` and font methods:

```
<body>

<canvas id="myText" width="400" height="250"
style="border:3px solid #0000FF;">

</canvas>

<script type="text/javascript">

    var canvas = document.getElementById("myText");

    context = canvas.getContext("2d");

    context.font = "30px Arial";

    context.fillText("Canvas-generated text", 40, 120);

</script>

</body>
```

Enhance a Canvas Object

GET READY. To enhance a canvas object, perform the following steps:

1. In an editing or app development tool, create an HTML document that includes the following content:

```
<!doctype html>

<html>

<head>

    <meta charset="utf-8" />

    <title>Canvas Exercise</title>

<script>

   function f1() {

       var canvas =

       document.getElementById("smlRectangle");

       context = canvas.getContext("2d");

       context.fillStyle = "coral";

       context.fillRect(10, 20, 200, 100);

   }

  </script>

  </head>

<body onload = "f1();">

<canvas id="smlRectangle" height='300' width='500'>

</canvas>

</body>
```

```
</html>
```

3. Replace the solid color with a gradient that starts with light blue and ends with dark blue. To do so, replace the current `fillStyle` line with the following:

```
var grd = context.createLinearGradient(0, 0, 150, 0);
        grd.addColorStop(0.3, "lightblue");
        grd.addColorStop(0.7, "darkblue");
        context.fillStyle = grd;
```

Lesson 8: Understanding JavaScript and Coding Essentials

Learning Objectives

Students will learn about:

- Managing and maintaining JavaScript
- Creating and using functions
- Updating the UI by using JavaScript
- Locating and accessing elements
- Listening and responding to events
- Showing and hiding elements
- Updating the content of elements
- Adding elements

ODN Skills

Manage and maintain JavaScript.	4.1
Update the UI by using JavaScript.	4.2

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to test and show all of the examples in Lesson 8.

Lesson 8 introduces students to JavaScript. Begin by explaining that JavaScript is a programming language that provides action in applications. Describe interactivity and the concept of a dynamic application.

The first section has students create a simple JavaScript application. Emphasize that JavaScript can be disabled in a Web browser, and that it's important for students to know how to enable JavaScript if necessary. In Internet Explorer, you can enable JavaScript in the Internet Options dialog box.

Explain that alert boxes are commonly used to test the operation of JavaScript programs, and that students will see alert boxes used in many examples in this course. Walk through the code for the first example program in the lesson and then demonstrate the alert box. Point out that `onClick` is an event that responds to something, in this case, a mouse click. Mention that events will be discussed later in the lesson.

The next section focuses on functions. Define a function and its purpose, and distinguish between the expression of a function and its execution. Mention that JavaScript programmers sometimes identify functions that return no value as subroutines. Then, explain the purpose of `<script>` tags and where they are found in HTML documents. You should also define and describe variables and identifiers.

In the next section, introduce JavaScript libraries and jQuery. Consider taking students on a quick tour of where to find JavaScript libraries on the Internet. Demonstrate a simple application using jQuery.

Next, introduce the `getElementById()` method for accessing display elements. Ask students for examples of display elements, and then demonstrate the `getElementById()` method in an application. Consider demoing the validation example from the textbook, revisiting the purpose of validation.

Move on to the next section, which covers events. Ask students to define "event" and provide an example. Explain the purpose of an event handler and define callback, and how an application "listens" for event. Then, explore the `onLoad` event. It's important for students to understand that JavaScript programs can be erratic—they give different results at different times. Explain why and offer an example. Tell students that one solution is to begin calculations only after `onLoad` has fired.

Explain show/hide functionality in JavaScript using the HTML `display` attribute. Then explore the `innerHTML` property to change the current content of HTML elements (referred to as "inner content") or to insert new content.

Finally, describe how to use the `createElement` command and the `appendChild` method in JavaScript to add elements to the display and demonstrate their use.

Key Terms

callback - A callback is a response to an event, such as a script execution in response to a mouse click.

computer program - A computer program is a recipe we direct the computer to execute that results in a particular display or action.

dynamic application - A dynamic application is one that adjusts and responds to end particular users or user actions.

event handler - An event handler is an optional script or executable that handles input received in a program. Handlers are JavaScript code inside the `<html>` tags (rather than the `<script>` tags) that execute other JavaScript code in response to an event.

events - An event is an action that triggers another action to occur.

function - A function is a segment of a program defined and performed in isolation from other parts.

identifier - An identifier is the name of a variable or function. Identifiers cannot be the same as words already used in the language; for example, "if " has a special meaning in JavaScript statements and is not available as a variable name.

interactivity - Interactivity enables an end user to take an action in an application, usually by clicking a button or pressing a key.

JavaScript library - A JavaScript library is pre-written JavaScript code.

jQuery - jQuery is the leading JavaScript library.

library - A library is collection of resources, like pre-written function code and subroutines, that developers use to create programs. JavaScript builds in a library of useful functions for many common operations.

methods - Methods are JavaScript functions that belong to objects. Methods differ from functions only in that methods are always associated and used with a particular object.

subroutines - A subroutine is a function that returns no value.

validation - Validation is the process of verifying that information entered or captured is in the correct format.

variable - A variable is a symbolic abbreviation, or name, that stands for a piece of data.

Lesson 8: Understanding JavaScript and Coding Essentials- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. A __computer program__ is a recipe we direct the computer to execute to result in a particular display or action
2. You use the `var` syntax to define a new __variable__ in JavaScript.
3. Many application requirements involve __events__, which are actions, such as a mouse click, that trigger other actions to occur.
4. You notice that a part of your JavaScript program represents a sequence of actions that is logically separate from other parts of program. It likely will be useful to define a __function__ to perform that specific sequence.
5. The __onload__ event associated with `<body>` constitutes a guarantee that all HTML has been displayed.
6. A common coding pattern is to attach an id to a specific HTML element, then access that element through JavaScript with _`getElementById()`_.
7. JavaScript __identifiers__ are the names of variables and functions.
8. A __library__ is collection of resources, like pre-written code and subroutines, that developers use to create programs.
9. A __JavaScript library__ is pre-written JavaScript code.
10. More than half of leading Web sites use the __jQuery__ JavaScript library.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. JavaScript programming makes applications which of the following? (Choose all that apply.)
   a. static
   [✓ b. dynamic]
   c. syntactically correct
   [✓ d. interactive]
2. Which of the following are JavaScript libraries? (Choose all that apply.)
   [✓ a. Dojo]
   [✓ b. MooTools]
   [✓ c. YUI]
   [✓ d. jQuery]
3. Which of the following names a valid JavaScript variable?
   a. `my.variable`
   b. `1st-variable`

c. `ord['a']`

d. [TBC]var1_$

4. When can JavaScript *not* be used?

   a. With HTML 4.01 and previous

   b. When the user has set a browser preference to disable JavaScript

   c. When the user hasn't installed JavaScript on his or her desktop

   d. none of the above

5. JavaScript uses which property to change the current content of HTML elements?

   a. `changeHTML`

   b. `modInnerHTML`

   c. innerHTML

   d. `HTMLinner`

6. A particular Web page has a single `<form>`. How does JavaScript best reach this?

   a. document. getElementsByTagName("form")[0]

   b. `document. getElementsByTagName("form")[1]`

   c. `document. getElementsByTagName("form")`

   d. `[document. getElementsByTagName("form")]`

7. The user has checked a box in a form indicating that he has not traveled recently in a country with an elevated incidence of hepatitis. How would you use JavaScript to hide an advisory paragraph?

   a. warning.style.display = "none"

   b. `warning.style.display = 0`

   c. `hide(warning)`

   d. `warning.style.hide()`

8. An individual statement in JavaScript ends in a ________.

   a. hash mark

   b. closing parenthesis

   c. period

   d. semicolon

9. Which of the following does JavaScript use to add new elements to a program display? (Choose all that apply.)

   a. createElement

   b. appendChild

   c. `getElement`

   d. `addChild`

10. Which of the following can you use to show and hide elements in a JavaScript program?

    a. display attribute

    b. `show-hide attribute`

    c. `show command`

d. `innerHTML`

True / False

Circle T if the statement is true or F if the statement is false.

T [✓ F] 1. The names of functions are listed in the JavaScript standard.

[✓ T] F 2. In HTML source, JavaScript typically appears inside a `<script>` element.

[✓ T] F 3. It is possible to write JavaScript code in such a way that it executes before all images are loaded.

T [✓ F] 4. If function `f2()` uses function `f1()`, and the definitions for both functions appear in the same `<script>`, then the definition for `f1()` must appear first.

T [✓ F] 5. The `getElementByElement()` method returns a reference to the first object with the specified `id` or `NAME` attribute.

Competency Assessment

Scenario 8-1: Validating Input

You've been asked to code a JavaScript function that judges whether a user entry might be a valid U.S. Social Security number or not. How would you go about this? What would you deliver back to your team?

You don't need the whole application to complete this assignment. Instead, write a tiny demonstration: a small stand-alone HTML source with a single entry field, an also invokes a JavaScript function kept in a separate source that validates the value of that entry according to the specifications given you. Test different values to ensure that your validator returns correct results.

More sophisticated solutions automatically test different values against specifications.

There are a couple of possibilities for delivery of results to the larger team: first is the `<link . . .>` which provides the validating function's source. A second possibility is to provide that `<link . . .>`, along with an application that tests it.

Scenario 8-2: Understanding Function Names

Raymond is creating a program that includes several functions. He has listed his function names and their descriptions in a table for easy reference while he develops his application. Some of his applications aren't working and he asks you for advice. You notice that the functions in question begin with hash marks or slashes. What do you tell Raymond about naming functions?

While the function name is under our control, there are a few restrictions on our choice: the name must be made up of letters, digits, underscores, and dollar signs, and the first character of the name must be a letter, underscore, or dollar sign. `example$1` is a possible JavaScript function name, but `not.with.periods`, `1wrong`, and `first name/last name` are not.

A full description of valid JavaScript names is beyond the scope of this language, because of the way letters are handled in languages other than English. For an English-speaker, though, a name is:

- a sequence of characters, where

- each character is a letter (‘A’-‘Z’, ‘a’-‘z’), a digit (‘0’-‘9’), an underscore (‘_’), or dollar sign (‘$’), and
- the first character is a letter, underscore, or dollar sign.

Proficiency Assessment

Scenario 8-3: Picking a JavaScript Library

CiCi wants to begin using JavaScript libraries to speed the production time of her programs, but she’s not sure where to start. There are so many libraries available that it’s hard to choose. What do you tell her?

jQuery is the leading JavaScript library. Over half of the 10,000 most-visited Web sites in the world use jQuery. jQuery is available for use with no fee and minimal restriction. Microsoft and other industry leaders make copies of jQuery available for anyone to download and use. Aside from jQuery, other popular libraries include Dojo, MooTools, and YUI.

Scenario 8-4: Distinguishing between Methods and Functions

Andre is new to JavaScript programming and is struggling to understand the difference between methods and functions. What do you tell him?

Methods differ from functions only in that they’re always associated and used with a particular object. `isNaN()` is an example of a JavaScript function, which tests for “not a number.” If `isNaN()` returns a value of 0 (false), the value is a number. `document.getElementById()` is an example of a JavaScript method; you can effectively only use `getElementById` with the special document object.

Lesson 8: Understanding JavaScript and Coding Essentials – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Create a Simple JavaScript Program

GET READY. To create a simple JavaScript program, perform the following steps:

1. Using an editing or app development tool, create a file with the following content:

```
<!doctype html>
<html>
<head>
<title>My first JavaScript program</title>
</head>
<body>
<h1>My first JavaScript program</h1>
<p>This is text.
<button type = 'button' onclick = "alert('You clicked the
   button');">I'm a button; click me</button>
</body>
</html>
```

Use a JavaScript Function

GET READY. To learn how to use a JavaScript function, perform the following steps:

1. In an editing or app development tool, create **L8-js2.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>First use of a function</title>
<script type = "text/javascript">
function example1() {
    alert("This is the first alert.");
    alert("This is the second alert.");
}
</script>
</head>
<body>
<h1>First use of a function</h1>
```

```
<p>This is text.
<button type = 'button' onclick = "example1();">I'm a
   button; click me</button>
</p>
</body>
</html>
```

Use a Variable in a JavaScript Program

GET READY. To use a variable in a JavaScript program, perform the following steps:

1. In an editing or app development tool, create **L8-js3.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>First use of a variable</title>
<script type = "text/javascript">
function example1() {
    var version_name = "serial number X358-AA-3T601-22"
    alert("This is the first alert of " + version_name);
    alert("This is the second alert of " + version_name);
}
</script>
</head>
<body>
<h1>First use of a variable</h1>
<p>This is text.
<button type = 'button' onclick = "example1();">I'm a
   button; click me</button>
</body>
</html>
```

Using jQuery and Other Third-Party Libraries

Use jQuery

GET READY. To use jQuery, perform the following steps:

1. In an editing or app development tool, create **L8-js4.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>First use of jQuery</title>
```

```
<script type = "text/javascript"
        src =
   "http://ajax.aspnetcdn.com/ajax/jquery/jquery-
   1.5.js"></script>
<script type = "text/javascript">

    // Once the HTML document loads, execute
    // the function within ready().
$(document).ready(function() {
        // Each paragraph receives a "click" action:
           // hide that particular paragraph.
    $("p").click(function() {
        $(this).hide();
    });
});

</script>
</head>
<body>
<p>This is the first paragraph. Click on me to make me
   disappear.
<p>This is the second paragraph.
<p>This is the third paragraph.
</body>
</html>
```

Updating the UI by Using JavaScript

The Bottom Line

.

Create an In-Browser Calculator Using JavaScript

GET READY. To create an in-browser calculator using JavaScript, perform the following steps:

1. In an editing or app development tool, create **L8-js5.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>In-browser calculator</title>
</head>
</body>

<h1>In-browser calculator</h1>
```

```
<form name="calculator">
<table border=4>
<tr>
<td>
<input type="text" name="Input" Size="20">
<br>
</td>
</tr>
<tr>
<td>
<input type="button" name="one"   value="  1  "
   OnClick="calculator.Input.value
+= '1'">
<input type="button" name="two"   value="  2  "
   OnCLick="calculator.Input.value
+= '2'">
<input type="button" name="three" value="  3  "
   OnClick="calculator.Input.value
+= '3'">
<input type="button" name="plus"  value="  +  "
   OnClick="calculator.Input.value
+= ' + '">
<br>
<input type="button" name="four"  value="  4  "
   OnClick="calculator.Input.value
+= '4'">
<input type="button" name="five"  value="  5  "
   OnCLick="calculator.Input.value
+= '5'">
<input type="button" name="six"   value="  6  "
   OnClick="calculator.Input.value
+= '6'">
<input type="button" name="minus" value="  -  "
   OnClick="calculator.Input.value
+= ' - '">
<br>
<input type="button" name="seven" value="  7  "
   OnClick="calculator.Input.value
+= '7'">
<input type="button" name="eight" value="  8  "
   OnCLick="calculator.Input.value
+= '8'">
<input type="button" name="nine"  value="  9  "
   OnClick="calculator.Input.value
   += '9'">
<input type="button" name="times" value="  x  "
   OnClick="calculator.Input.value
```

```
+= ' * '">
<br>
<input type="button" name="clear" value="  c  "
   OnClick="calculator.Input.value
= ''">
<input type="button" name="zero"  value="  0  "
   OnClick="calculator.Input.value
+= '0'">
<input type="button" name="DoIt"  value="  =  "
   OnClick="calculator.Input.value
= eval(calculator.Input.value)">
<input type="button" name="div"   value="  /  "
   OnClick="calculator.Input.value
+= ' / '">
</td>
</tr>
</table>
</form>

</body>
   </html>
```

Locating and Accessing Elements

Use the getElementById() Method for User Input

GET READY. To learn how to use the `getElementByID()` method to gather user input, perform the following steps:

1. In an editing or app development tool, create **L8-js6.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Validation of a user entry</title>

<script type = "text/javascript">
function validate() {
    var value = document.getElementById("input1").value;
    if (isNaN(value)) {
        modifier = "not ";
    } else {
        modifier = "";
    }
    var report = "You entered '" + value + "'; this is " +
   modifier + "a valid number.";    alert(report);
}
```

```
</script>
</head>
<body>

<h1>Validation of a user entry</h1>
<form name="calculator">
<input type = "text" id = "input1"></input>
<button type = "button" onclick = "validate();">Click me to
   see what I think of your entry</button>
</form>

</body>
</html>
```

Listening and Responding to Events

Use the onLoad Event Handler

GET READY. To use the `onLoad` event handler, perform the following steps:

1. In a text editor or app development tool, create **L8-js7.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>The onLoad event handler</title>

<script type = "text/javascript">
function validate() {
    var value = document.getElementById("input1").value;
    if (isNaN(value)) {
        modifier = "not ";
    } else {
        modifier = "";
    }
    var report = "You entered '" + value + "'; this is " +
   modifier + "a valid number.";
    alert(report);
}

function init() {
    alert("At this point, it is guaranteed that all HTML
   elements have loaded.")
;
}
</script>
</head>
```

```
<body onload = "init();">

<h1>The onLoad event handler</h1>
<form name="calculator">
<input type = "text" id = "input1"></input>
<button type = "button" onclick = "validate();">Click me to
   see what I think of your entry</button>
</form>

</body>
</html>
```

Showing and Hiding Elements

Hide Parts of the Display Based on User Action

GET READY. To create an application that shows and hides a paragraph based on the value an end user enters, perform the following steps:

1. In a text editor or app development tool, create **L8-js8.html** with the following contents:

```
<!doctype html>
<html>
<head>
<title>Show/hide responsively</title>

<script type = "text/javascript">
function check_range() {
    var value = document.getElementById("price1").value;
    var paragraph_list =
   document.getElementsByTagName("p");
    var first_paragraph = paragraph_list[0];
    if (value >= 80) {
        display = "block";
    } else {
        display = "none";
    }
    first_paragraph.style.display = display;
}
</script>
</head>
<body>

<h1>Show/hide responsively</h1>
<form>
Enter a price: <input type = "number" id = "price1" min =
   "1" max = "100"
```

```
       oninput = "check_range();"
  ></input>
</form>
<p style = "display:none;">Warning: you are within 20% of
   your limit.</p>

</body>
</html>
```

Updating the Content of Elements

Update Content Visible on the Screen

GET READY. To create an application that updates content visible on the screen, perform the following steps:

1. In an editing or app development tool, create **L8-js9.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Compute element</title>

<script type = "text/javascript">
  // check_range assigns the display style
  // of the first paragraph as a function of
  // the displayed price—whether it's 80 and
  // above, or 79 and below.
function check_range() {
    var value = document.getElementById("price1").value;
    var paragraph_list =
   document.getElementsByTagName("p");
    var first_paragraph = paragraph_list[0];
    if (value >= 80) {
        display = "block";
    } else {
        display = "none";
    }
    first_paragraph.style.display = display;
}
  // compute_total() has the responsibility
  // to update the display with the total of
  // the price and its tax.
function compute_total() {
    var value = document.getElementById("price1").value;
    if (isNaN(value)) {
        total = "INDETERMINATE";
    } else {
            // Assume an 8% tax.
        total = 1.08 * value;
        total = total.toFixed(2);
    }
    var total_slot = document.getElementById("total");
    total_slot.innerHTML = total;
}
</script>
</head>
<body><h1>Compute element</h1>
<form>
Enter a price: <input type = "number" id = "price1" min =
   "1" max = "100"
       oninput = "check_range(); compute_total();"
  ></input> The total price, including tax,
            is <span id = "total">INDETERMINATE</span>.
</form>
<p style = "display:none;">Warning: you are within 20% of
   your limit.</p>
</body>
</html>
```

Adding Elements

Add an Element to the Display

GET READY. To create an application that adds elements to the display, perform the following steps:

1. In an editing or app development tool, create **L8-js10.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Create a new element</title>

<script type = "text/javascript">
function add_paragraph() {
    var original = document.getElementById("original");
    var new_paragraph = document.createElement("p");
    var current_time = new Date()
    var this_text = "This new paragraph appeared at " +
   current_time + ".";
      // Even after the browser has rendered
      // all HTML, it's possible to add *new*
      // HTML elements.  createTextNode()-
      // appendChild()-insertBefore() is one
      // typical pattern for adding new textual
      // content.
    var new_content = document.createTextNode(this_text);
    new_paragraph.appendChild(new_content);
    document.body.insertBefore(new_paragraph, original);
}
</script>
</head>
<body>

<h1>Create a new element</h1>
<p id = "original">This is the text that appears when the
   display first loads.</p>
<button type = "button" onclick = "add_paragraph();">Click
   on me to
     add new content</button>
<p style = "display:none;">Warning: you are within 20% of
   your limit.</p>

</body>
</html>
```

Lesson 9: Creating Animations, Working with Graphics, and Accessing Data

Learning Objectives

Students will learn about:

- Creating animations
- Working with images, shapes, and other graphics
- Manipulating the canvas with JavaScript
- Sending and receiving data
- Transmitting complex objects and parsing
- Loading and saving files
- Using the Application Cache (AppCache)
- Understanding and using data types
- Using JavaScript to validate user form input
- Understanding and using cookies
- Understanding and using local storage

ODN Skills

Code animations by using JavaScript.	4.3
Access data access by using JavaScript.	4.4

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to show most of the examples in Lesson 9. The authors used the Mozilla Firefox browser to show the local file access example; you can also use Internet Explorer 10 to display the feature.

Lesson 9 introduces students to animations, graphics, and data access using JavaScript. Begin by reminding students that they worked with animations in Lesson 7, and that JavaScript can provide animation as well. Explain the concept of recursion, and how to use `setTimeout()` recursively to introduce timing into JavaScript.

The next section focuses on images, shapes, and other graphics. Begin by exploring how to display an image when a button is clicked or in response to some other event using the `createElement()` method. Explain that JavaScript can display different types of graphics, from JPG and PNG files to shapes like boxes and circles. The `createElement()` method creates a reusable function to display an image.

The lesson revisits canvas, which students learned about in Lessons 2 and 7. Explain that students can use the `getElementById()` function and `canvas.getContext` to work with a canvas object in JavaScript. The `getElementById()` function finds the canvas element, and `canvas.getContext` (sometimes abbreviated to `c.getContext`) creates the canvas object. Students can then use a variety of methods to draw shapes, include images, and so on. Demonstrate the analog clock example or the animated boxes exercise from the textbook.

The next major section covers sending and receiving data. Explain that one of the most essential techniques for data transfer involves the XMLHttpRequest API (sometimes abbreviated as XHR). `XMLHttpRequest` enables you to use JavaScript to pass data in the form of text strings between a client and a server. Walk through the example in the textbook and explain how the application works.

Note: In order for students to try some of the data exchange applications created in the lesson, such as one involving AppCache, you'll need to set up student access to a Web server. To enable them to send and receive data using `XMLHttpRequest` requires a dynamic Web server and server-side programming. The Access Data exercise in the "Sending and Receiving Data" section of the textbook does not require Web server access.

Move on to parsing and describe situations in which it is used, such as extracting information from a data stream of stock quotes. Be sure to explain each of the lines in the Parse Complex Data exercise.

Next, discuss how JavaScript can access files on a local computer and, using HTML5, validate the file type before loading, which greatly reduces errors. Walk students through the Access a Local File exercise, using Internet Explorer 10 as the Web browser, if available.

The next section covers AppCache, which saves a copy of Web site files locally, in a structured form. Emphasize that AppCache is not the same as a browser's cache. Whereas a browser's cache saves all Web pages visited, AppCache saves only the files listed in the cache manifest. You can apply AppCache to a single Web page or an entire site. Have the students work through the AppCache exercise in class.

Note: You must first configure the Web server with the correct MIME type, which is text/cache-manifest. In addition, the preferred file extension for manifest files is .appcache.

The next brief section covers data types. Explain what a data type is and provide examples.

Move on to user form input validation. Remind students that they studied input validation in Lesson 3, using the HTML `input` element and the HTML5 `pattern` attribute. Lesson 9 uses JavaScript to achieve the same goal. Walk students through the Manage a Form with JavaScript exercise, pointing out any parallels to the `input pattern` method of validation.

The next section focuses on cookies. Ask students for a show of hands regarding who knows what a cookie is. Most students should have a basic understanding of them because of the prevalence of cookies in everyday Internet use. Explain the purpose of cookies and that, although they have their disadvantages, they are still widely used. Have students work through the Use Cookies exercise in class.

Note: Cookies can only be enabled for Web pages accessed through the network, not locally. You must first set up Web server with folders unique to each user. Students should have read and write access to their folder. They will use a file transfer program to upload an HTML file to the Web server to test cookies.

Move on to local storage. Explain that HTML5's Local Storage feature has better security and makes programming easier than with cookies. Have students work through the Save to Local Storage exercise in class. Tell them they can use the same unique folder and Web address they used for the cookies exercise. Describe differences between the `localStorage` method and cookie method of saving persistent data.

Key Terms

animation - An animation is the display of a sequence of static images at a fast enough speed to create the illusion of movement.

AppCache - The Application Cache, or AppCache, is an HTML5 feature that enables Web data to be stored locally when a user is offline. AppCache stores resources like images, HTML pages, CSS files, and JavaScript—data that would ordinarily be stored on a server. Because the resources are stored on the client's hard disk or device, the resources load faster when requested.

canvas element - The canvas element is a drawing area under programmatic control.

cookies - Cookies are small text files that Web sites save to a computer's hard disk that contain information about the user and his or her browsing preferences.

data type - A data type is JavaScript's interpretation of the kind of data a program can work with. Data types include string, number, array, Boolean, null, object, and undefined.

encapsulate - To encapsulate is to group data and the methods that operate on it under a single name.

JSON - JSON (JavaScript Object Notation) is a subset of JavaScript, providing a way to store information in an organized, easy-to-access manner.

Local Storage - Local Storage is an HTML5 feature that uses the `localStorage` method to allow users to save relatively large amounts of data from session to session (persistent data). It is an improvement on data storage using cookies.

parsing - Parsing is a term used to describe analysis of complex information into constituent parts.

recursion - Recursion is a programming technique in which a function calls itself.

XMLHttpRequest API - The XMLHttpRequest API, sometimes abbreviated XHR, enables you to use JavaScript to pass data in the form of text strings between a client and a server.

Lesson 9: Creating Animations, Working with Graphics, and Accessing Data- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. __Animation__ is the display of a sequence of static images at a fast enough speed to create the illusion of movement.
2. You need to draw a complex diagram as part of an HTML5 display. One way is with the __canvas__ element.
3. Before HTML5, the most common way to keep information on the client side of a Web application—that is, on the user's computer—was with __cookies__. [Possible alternative answer: "hidden form variables".]
4. __`XMLHttpRequest`__ enables you to use JavaScript to pass data in the form of text strings, but not objects, between a client and a server.
5. __Recursion__ is a programming technique in which a function calls itself.
6. __Parsing__ is the label generally used for analysis of complex information into constituent parts.
7. The __AppCache__ API saves a copy of your Web site files locally, in a structured form.
8. Values in JavaScript appear in special different appearances, called __data types__, which are most often strings and numbers.
9. __JSON__ is a subset of JavaScript that enables you to exchange JavaScript objects with a server.
10. __`localStorage`__ is an alternative to cookies.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which of the following is the most common way to code JavaScript with a delayed effect?
   a. `sleep()`
   b. `delay()`
   c. `wait()`
   d. setTimeout()
2. JavaScript can display different types of graphics, from JPG and PNG files to shapes like boxes and circles. One method you can use to display graphics using JavaScript is:
   a. createElement
   b. `move_paragraph`
   c. JSON
   d. `display`
3. What are the two primary constructs used to draw a canvas object?

   a. getElementById()
   b. `getCanvasContext`
   c. `getElementByCanvas()`
   d. canvas.getContext
4. Sending and receiving data in JavaScript requires a dynamic Web server and:
   a. client-side validation
   b. server-side programming
   c. CSS
   d. none of the above
5. The canvas element builds in which set of methods?
   a. `drawRect(), outlineRect(), eraseRect()`
   b. fillRect(), strokeRect(), clearRect()
   c. `beginPath(), fillPath(), endPath()`
   d. `beginPath(), fillPath(), closePath()`
6. Which JSON API converts a JavaScript object to string data for exchange with a server?
   a. `JSON.parsify`
   b. `XMLHttpRequest`
   c. JSON.stringify
   d. `getObjectString`
7. How does AppCache differ from browser cache?
   a. AppCache saves copies of Web pages.
   b. You first have to visit a Web page for it to be included in the cache.
   c. AppCache saves only those files listed in the cache manifest.
   d. AppCache and browser cache are the same thing.
8. Which of the following is not a data type used by JavaScript?
   a. composite
   b. string
   c. number
   d. Boolean
9. Which of the following poses a threat to data privacy?
   a. AppCache
   b. `localStorage`
   c. cookies
   d. animation
10. Which API enables you to work on remote files offline?
   a. XMLHttpRequest
   b. AppCache
   c. JSON.parse

d. JSON.stringify

True / False

Circle T if the statement is true or F if the statement is false.

T [✓ F] 1. JavaScript doesn't allow recursion.

T [✓ F] 2. You use `XMLHttpRequest` to create animations.

[✓ T] F 3. It is possible to write JavaScript code in such a way that it executes before all images are loaded.

[✓ T] F 4. You can use `localStorage` to store a user's personal data.

[✓ T] F 5. A common technique in JavaScript animation is to use `setTimeout` recursively.

Competency Assessment

Scenario 9-1: Understanding Animation Basics

Roan works as an administrative assistant at a nonprofit organization that is producing an encyclopedia on wild plants. The outreach volunteer asked Roan if he could animate the graphic on the homepage so that it appears as though the seed grows to a full-grown swatch of prairie grass in 5 or 6 seconds. Roan knows nothing about animation and asks you for a quick summary.

Tell Roan that animation is the display of a sequence of static images at a fast enough speed to create the illusion of movement. He probably would not need recursion with this type of animation, but he should look into canvas to re-create the graphic dynamically.

Scenario 9-2: Creating a Warehouse Floor Plan App

Trudy is a warehouse manager. She wants a Web application that sketches an outline of the floor plan of the interior of a warehouse. She's supposed to submit a request to the development team soon but doesn't know the appropriate technology to include in her description. Trudy wants your opinion. What do you tell her?

Suggest that Trudy ask for an HTML5 canvas application that represents the warehouse interior. The app should be created so that she can draw the shelves as rectangles on the map.

Proficiency Assessment

Scenario 9-3: Enhancing the Warehouse Application

Trudy has decided to expand her request to the development team, and now wants to include an entry form that accepts stock-keeping unit (SKU) codes. The warehouse server has photographs of merchandise, indexed by SKU, as well as information on where in the warehouse items of that description are currently located. Trudy wants the program to place a photograph of the item on the aisle and shelf where it is currently located. Briefly, how can this be done? She needs to add the description to her request.

Tell Trudy to include this in her request:

Define an HTML form that invites entry of a SKU. Use JavaScript to validate the entry. Use the SKU value to request the photograph and location of the corresponding item, and place the photograph on the map.

Scenario 9-4: Reviewing Documents Offline

An existing online application collects and scores reviewers' comments about different budget items in a highly structured process: each item links to proposal details, personnel involved, and so on. The application is successful. The reviewers complain, though, that it can only be used when online; they can't, for example, fill out the score sheets while on an airplane flight and without an Internet connection. They propose to download all the linked materials and fill out spreadsheets of their comments. What is your response?

The spreadsheets are not an ideal solution. The users might not realize how many problems spreadsheets introduce—essentially, spreadsheets are too flexible and will be difficult to aggregate.

Lesson 9: Creating Animations, Working with Graphics, and Accessing Data – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Creating Animations

Create a Simple Animation

GET READY. To create a simple application, perform the following steps:

1. Create **L9-js1.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Animate with JavaScript</title>

<script type = "text/javascript">
  // Create a “ticker-tape” effect by sliding
  // the paragraph of text one pixel to the
  // right, over and over, until the right-hand
  // limit of 300 pixels is reached.  At that
  // point, restart the animation all the way
  // to the left.
function move_paragraph() {
    next = current + "px";
    current += 1;
    if (current > 300) {
        current = 0;
    }
    paragraph.style.left = next;
      // Pause for 18 milliseconds before
      // the next move.
    var rate = 18;
    setTimeout(move_paragraph, rate);
}
function init() {
    paragraph = document.getElementById("original");
    paragraph.style.position = "absolute";
    current = 0;
    move_paragraph();
}
</script>
</head>
```

```
<body onload = "init();">

<h1>Animate with JavaScript</h1>
<p id = "original">Do you see me scrolling across the
   screen?</p>

</body>
</html>
```

Create an Interactive Animation

GET READY. To create an animation responsive to the action of a user, perform the following steps:

1. Create **L9-js2.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Animate with JavaScript</title>

<script type = "text/javascript">
  // This page acts much the same as L9-js1.html
  // did, with the exception that the frequency
  // of moving the paragraph depends on the numeric
  // value the end-user has entered in the input
  // field.
function move_paragraph() {
    next = current + "px";
    current += 1;
    if (current > 300) {
        current = 0;
    }
    paragraph.style.left = next;
    var rate = document.getElementById("rate").value;
    setTimeout(move_paragraph, rate);
}
function init() {
    paragraph = document.getElementById("original");
    paragraph.style.position = "absolute";
    current = 0;
    move_paragraph();
}
</script>
</head>
<body onload = "init();">
```

```
<h1>Animate with JavaScript</h1>
  <!-- The number <input> tag is new with HTML5.
  It provides a convenient way to enforce that
  the end-user inputs a valid numeral within a
  specified range. -->
<form>
<input id = "rate" type = "number" value = "18" min = "5"
   max = "100"></input>
</form>
<p id = "original">Do you see me scrolling across the
   screen?</p>

</body>
</html>
```

Working with Images, Shapes, and Other Graphics

The Bottom Line

You can use JavaScript to display an image when a button is clicked or some other event occurs. The `createElement` method works well for this use.

JavaScript can display different types of graphics, from JPG and PNG files to shapes like boxes and circles. One method is to use the `createElement()` method. This method creates a reusable function to display an image:

```
function show_image(src, width, height, alt) {
    var img = document.createElement("img");
    img.src = src;
    img.width = width;
    img.height = height;
    img.alt = alt;

    // Adds it to the <body> tag
    document.body.appendChild(img);
}
```

To display the image, include this code:

```
<button onclick="show_image
('path/filename', 276,110, 'Logo');">
 Display logo</button>
```

Manipulating the Canvas with JavaScript

.

Use Canvas to Create a Clockface with Moving Hands

GET READY. To demonstrate use of HTML5's canvas, perform the following steps:

1. Create **L9-js3.html** with following content:

```
<!doctype html>
<html>
<head>
<title>Analogue clockface illustrates JavaScript's control
   of canvas</title>

<script type = "text/javascript">
function draw_leg(fraction) {
    dctx.lineTo(center_x + length * Math.sin(2 * Math.PI *
   fraction),
                center_y - length * Math.cos(2 * Math.PI *
   fraction));
}

function init() {
    var canvas = document.getElementById("clockface");
      // The following variables are created as
      // globals, so they can conveniently be
      // accessed by other functions.
    dctx = canvas.getContext('2d');
    dctx.fillStyle = "black";
    center_x = 100;
    center_y = 100;
    length = 100;
    show_hands();
}

// A single hand is drawn as an isosceles triangle
   // from the center to the edge of the clockface.
function show_hand(fraction, width) {
    dctx.beginPath();
    dctx.moveTo(center_x, center_y);
    draw_leg(fraction - width);
    draw_leg(fraction + width);
    dctx.fill();
}

function show_hands() {
      // Erase anything already present in the area
      // that represents the clock face.
    dctx.clearRect(0, 0, 200, 200);
```

```
        // What is the time *now*?
      var now = new Date();
      seconds = now.getSeconds();
      minutes = now.getMinutes() + seconds / 60;
      hours = now.getHours() + minutes / 60;
        // The second hand is “skinniest” of the three.
      show_hand(seconds / 60, 0.002);
      show_hand(minutes / 60, 0.005);
        // The hour hand is twice as wide as the minute
        // hand.
      show_hand(hours / 12, 0.01);
      var rate = 1000;
      setTimeout(show_hands, rate);
}
</script>
</head>
<body onload = "init();">

<h1>Analogue clockface illustrates JavaScript's control of
   canvas</h1>
<canvas id = "clockface" width = "200" height =
   "200"></canvas>

</body>
</html>
```

Create Animated Boxes Using Canvas

GET READY. To create animated shapes using the canvas element, perform the following steps:

1. Create **L9-js4.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Blocks</title>

<script type = "text/javascript">
// This page is “for play”—it just places blocks of
// color on the screen with a bit of randomization,
// to achieve mildly interesting visual effects.
// Recursion is used in a couple of distinct ways
// below:  place_blocks() calls draw_spiral(), and
// draw_spiral() calls either place_blocks() or
// draw_spiral(), depending on how much of a spiral
// has been most recently drawn.
```

```
function init() {
    var canvas = document.getElementById("drawing_area");
    dctx = canvas.getContext('2d');
    place_blocks();
}

function draw_spiral() {
      // Once a block moves outside the drawing
      // area, stop the current spiral, and create
      // a new one.
    if (x > 500 || y > 500 || x < 0 || y < 0) {
        place_blocks();
    }
    ratio = 1.6;
    newx = x;
    newy = y;
    dx = size;
    dy = size;
      // Each block is turned 90 degrees away
      // from the last one.
    switch (direction) {
        case "up":
            dy = -size;
            newy += dy;
            direction = "left";
            break;
        case "left":
            dx = -size;
            dy = -size;
            newx += dx;
            direction = "down";
            break;
        case "down":
            dx = -size;
            newy += dy;
            direction = "right";
            break;
        case "right":
            newx += dx;
            direction = "up";
            break;
    }
    dctx.fillRect(x, y, dx, dy);
      // Each successively-drawn block is larger
      // than the last.
    size *= ratio;
    x = newx;
    y = newy;
    setTimeout(draw_spiral, delay);
}


function place_blocks() {
    dctx.fillStyle =
   '#'+Math.floor(Math.random()*16777215).toString(16);
    x = 100 + 300 * Math.random();
    y = 100 + 300 * Math.random();
    delay = 100 + 2000 * Math.random();
    size = 3 + 7 * Math.random();
    direction = "up";
    draw_spiral();
}
</script>
</head>
<body onload = "init();">

<h1>Blocks</h1>
<canvas id = "drawing_area" width = "500" height =
   "500"></canvas

</body>
</html>
```

Sending and Receiving Data

The Bottom Line

Access Data

GET READY. As a minimal example of JavaScript's ability to access data, perform the following steps:

1. Create **L9-js5.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>JavaScript accesses data</title>

<script type = "text/javascript">

function init() {
    var paragraph_object =
   document.getElementById('paragraph');
    message = "Notice that the title of this page is '" +
   document.title + "'.";
    paragraph_object.innerHTML = message;
}

</script>
</head>
<body onload = "init();">

<h1>JavaScript accesses data</h1>
<p id = "paragraph"></p>

</body>
</html>
```

Transmitting Complex Objects and Parsing

Parse Complex Data

GET READY. To demonstrate JavaScript's ability to parse complex data, perform the following steps:

1. Create **L9-js6.html** with the following contents:

```
<!doctype html>
<html>
<head>
<title>Parsing complex data</title>

<script type = "text/javascript">

  // The next statement needs to appear
  // on a single line.
sample_data = "Mobil-17: 3.49; Kroger-03: 3.36; Exxon-01:
   3.59; Kroger-04: 3.49;
   Valero-A: 3.41; Chevron-01: 3.52";
of_interest = "Kroger-04";

// sample_data is an example of a typical piece of
// structured data:  a string with parts separated
// by semi-colons, where each of the parts has two
// subparts separated by a colon.  Many other formats
// are possible.  To parse this particular format,
// init() splits on two distinct separators.
function init() {
    var paragraph_object = 
   document.getElementById("paragraph");
    var data_list = sample_data.split(';');
    for (j = 0; j < data_list.length; j++) {
        parts = data_list[j].split(':');
        var site = parts[0].trim()
        if (site == of_interest) {
           var message = "Given the sample data '" + 
   sample_data + "', this program parsed out the price $" + 
   parts[1].trim() + " for the " + site + " site.";
           paragraph_object.innerHTML = message;
        }
    }
}

</script>
</head>
<body onload = "init();">

<h1>Parsing complex data</h1>
<p id = "paragraph">Welcome.</p>

</body>
</html>
```

For example, you can also use a subset of JavaScript called ***JSON*** (JavaScript Object Notation) to exchange JavaScript objects with a server. Using the JSON.parse and JSON.stringify APIs, the code might look like the following:

```
function loadJSON(url, data, callback) {
    var xhr = new XMLHttpRequest();
    xhr.open("GET", url, true);
    xhr.onload = function() {
        callback( JSON.parse(xhr.responseText) );
    }
    xhr.send( JSON.stringify(data) );
}


loadJSON("my.json", { id : 1 }, function(response) {
    setTitle(response.title);
});
```

Loading and Saving Files

The Bottom Line

JavaScript can access files on your local computer and, with HTML5, can validate the file type before

Access a Local File

GET READY. To demonstrate JavaScript's ability to access a local file, perform the following steps:

1. Create **L9-js7.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>JavaScript accesses local files</title>

<script type = "text/javascript">
function acknowledge(file_handle) {
    var size = file_handle.size;
    var fname = file_handle.name;
    var message = "You have chosen the file '" + fname +
   "'.  This appears to be a recognizable image, totaling "
   + size + " bytes in size.";
    alert(message);

}

function complain(fname) {
    var message = "The file named '" + fname + "' does not
   appear to have an acceptable extension.";
    alert(message);
}

function handle_file_selection(item) {
    var f = item.files[0];
    var fname = f.name;
    var last_index = fname.lastIndexOf('.');
    if (last_index == -1) {
        complain(fname);
        return;
    }
    var ext = fname.substr(last_index + 1);
    if (ext.toLowerCase() in {'gif': '',
                              'jpg': '',
                              'png': '',
                              'tif': ''
                              }) {
        acknowledge(f);
    } else {
        complain(fname);
    }
}

</script>
</head>
<body>

<h1>JavaScript accesses local files</h1>
<input type = 'file'
       onchange = 'handle_file_selection(this);' />

</body>
</html>
```

Using JavaScript to Validate User Form Input

The Bottom Line

Manage a Form with JavaScript

GET READY. To demonstrate JavaScript's ability to manage a form, perform the following steps:

1. Create **L9-js8.html** with contents:

```
<!doctype html>
<html>
<head>
<title>Form management</title>

<script type = "text/javascript">

                  // The action of the correct() function
   is to
                  // test the characters the user has
   entered
                  // against the pattern ‘XXX-XXX-XX-X’.
   If the
                  // user’s entry does not match this
   pattern,
                  // remove the last character. This gives
   the
```

```
                  // user the impression that she or he can
   only
                  // enter valid characters.
function correct() {
    var input_object = document.getElementById("serial");
    var value = input_object.value;
    var current_length = value.length;
    if (current_length) {
        var last_character = value.substring(current_length
   - 1);
        switch (current_length) {
            case 4:
            case 8:
            case 11:
                if (last_character != '-') {
                    value = value.substring(0,
   current_length - 1);
                }
                break;
            default:
                if (!/\d/.test(last_character)) {
                    value = value.substring(0,
   current_length - 1);
                }
        }
        if (current_length > 12) {
            value = value.substring(0, current_length - 1);
        }
        current_length = value.length;
        switch (current_length) {
            case 3:
            case 7:
            case 10:
                value += "-";
        }
        input_object.value = value;
    }
}

</script>
</head>
<body>

<h1>Form management</h1>
<form>
```

```
Enter here a serial number in the pattern XXX-XXX-XX-X,
   where each X is a digit: <input id = "serial" type =
   'text' size = '12'
                                  onkeyup = "correct();">.
</form>

</body>
</html>
```

Understanding and Using Cookies

The Bottom Line

Use Cookies

GET READY. To demonstrate JavaScript's ability to retain information on the desktop computer even with the browser shut down, perform the following steps:

1. Create **L9-js9.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Use of cookies</title>

<script type = "text/javascript">

function getCookie(c_name) {
    var i,x,y,ARRcookies=document.cookie.split(";");
    for (i=0;i<ARRcookies.length;i++)
    {
      x=ARRcookies[i].substr(0,ARRcookies[i].indexOf("="));
      y=ARRcookies[i].substr(ARRcookies[i].indexOf("=")+1);
      x=x.replace(/^\s+|\s+$/g,"");
      if (x==c_name)
        {
        return unescape(y);
        }
      }
}

function init() {
    var message;
    level_object  = document.getElementById("level");
    var welcome = document.getElementById("welcome");
    var level = getCookie("level");
    if (level == null || level == '') {
        message = "It appears this is your first time to
   play. You will start at level 1.";
        level = 1;
     } else {
        message = "When you last played, you reached level "
   + level +".  You will start there now.";
     }
    welcome.innerHTML = message;
    level_object.value = level;
}

function save_level() {
    setCookie("level", level_object.value, 10);
}

function setCookie(c_name,value,exdays) {
    var exdate=new Date();
    exdate.setDate(exdate.getDate() + exdays);
    var c_value=escape(value) + ((exdays==null) ? "" : ";
   expires="+exdate.toUTC
String());
    document.cookie=c_name + "=" + c_value;
}

</script>
</head>
<body onload = "init();">

<h1>Use of cookies</h1>
<p id = "welcome">Welcome.</p>
<form>
You can update your level at any time.  It is currently set
   at
<input id = "level" type = "number" min = "1" max = "100"
       oninput = "save_level();" />.
</form>

</body>
</html>
```

Understanding and Using Local Storage

The Bottom Line

Save to Local Storage

GET READY. To demonstrate JavaScript's ability to save information with*out* cookies, perform the following steps:

1. Create **L9-js10.html** with the following content:

```
<!doctype html>
<html>
<head>
<title>Use of local storage</title>

<script type = "text/javascript">
// This page demonstrates a simple model for
// saving a player’s “level” in local storage.
function init() {
    var message;
    level_object  = document.getElementById("level");
    var welcome = document.getElementById("welcome");
      // “localStorage” is a keyword for an HTML5-
      // capable browser.  “localStorage.level” is
      // a variable name chosen for use by this
      // particular page.
    var level = localStorage.level;
    if (level == null || level == '') {
       message = "It appears this is your first time to
   play. You will start at level 1.";
       level = 1;
    } else {
       message = "When you last played, you reached level "
   + level +".  You will start there now.";
    }
    welcome.innerHTML = message;
    level_object.value = level;
}

function save_level() {
    localStorage.level = level_object.value;
}

</script>
</head>
<body onload = "init();">
```

```
<h1>Use of local storage</h1>
<p id = "welcome">Welcome.</p>
<form>
You can update your level at any time.  It is currently set
   at
<input id = "level" type = "number" min = "1" max = "100"
       oninput = "save_level();" />.
</form>

</body>
</html>
```

Lesson 10: JavaScript Coding for the Touch Interface, Device and Operating System Resources, and More

Learning Objectives

Students will learn about:

- Touch interface
- Gestures
- Capturing geolocation data
- Using Web Workers, WebSockets, and the File API
- Accessing in-memory resources
- Hardware capabilities: GPS, accelerometer, camera

ODN Skills

Respond to the touch interface.	4.5
Code additional HTML5 APIs.	4.6
Access device and operating system resources.	4.7

Lesson Summary — Lecture Notes

General Notes: Internet Explorer 9 was used to show most of the examples in Lesson 10. The authors used the Mozilla Firefox browser to show Web Worker and WebSocket examples; Internet Explorer browsers do not support these API as of this writing.

Lesson 10 introduces students to JavaScript techniques for responding to the touch interface, coding HTML5 APIs, and accessing device and operating system resources.

The first section revisits the touch interface. Remind students that they were introduced to the touch interface and gestures in Lesson 1. In this lesson, students learn how to code for gestures using JavaScript. Begin by briefly discussing how touch screens work, and the differences between resistive and capacitive touch screens. Emphasize that resistive screens respond well to finger touches, whereas capacitive screens respond more precisely to a stylus or some other conductive item. Review the gestures listed in Table 10-1.

Explain the `touchstart`, `touchmove`, `touchend`, and `touchcancel` touch events, and define touch object and touchlist. Briefly discuss the `touches`, `targetTouches`, `changedTouches` touchlists.

Describe the `addEventListener` event and walk through the example in the textbook.

Next, discuss the `gesturestart`, `gesturechange`, and `gestureend` gesture events, including the `scale` and `rotation` properties.

The next section covers the Geolocation, Web Workers, WebSockets, and File APIs. Begin by mentioning the Web Hypertext Application Technology Working Group (WHATWG), which maintains a living HTML specification that includes APIs that were not originally part of the HTML5 specification.

The Geolocation API gets a user's geographical coordinates (latitude and longitude). The API can also show a map with a marker showing the user's location based on the coordinates. Explain the purpose of the `getCurrentPosition` and `watchPosition` functions, and then explain each line of the `getCurrentPosition` example in the textbook. Also describe geodetic data versus civic data. Ask students why they think geolocation doesn't work consistently for desktop computers.

Next, discuss Web Workers. Explain the benefits of these APIs and the kinds of tasks you can accomplish with them. Explain that Web Workers objects run in isolated threads—they do not act directly on the main HTML document or the DOM. That means you don't use `getElementById` in your script. (You can use `setTimeout`, `setInterval`, and `XMLHttpRequest`.) Instead, Web Workers pass information through messages, executing code from a JavaScript file separate from the main HTML document. Walk through the example in the textbook that references the doWork.js file.

Next, discuss WebSockets, an API that offers full-duplex communication through a single socket over the Internet. Developers use WebSockets mainly for real-time Web applications like chat, multiplayer online gaming, and stock quotes. Discuss the three primary events associated with WebSocket communications: `onopen`, `onmessage`, and `onclose`.

The JavaScript that opens a WebSocket connection is:

```
var host = 'ws://example.com';
```

Point out the use of ws rather than http in the URL. You can also use wss for secure WebSocket connections, just like you use https for secure HTTP connections.

Next, discuss the File API, which allows a browser or application to upload files from local storage to a remote server without the need for a plug-in. Discuss some of the interfaces for accessing files from local storage: `File`, `FileList`, `Blob`, and `FileReader`. Mention that an easy way to load a file is to use the `input type="file"` element, which returns the list of selected File objects as a `FileList`.

The next section addresses device and operating system resources access, which is a function of the Windows Runtime (WinRT). Discuss the Web Storage API, the `localStorage` method, and the `sessionStorage` method.

Next, discuss the concept of platform independence. Explain that some device hardware capabilities you can access with device-independent applications are global positioning system (GPS), Accelerometer, and camera. Briefly discuss each capability.

Key Terms

accelerometer - An accelerometer is a device that measures acceleration, which is a change in speed or direction over a period of time. The Accelerometer sensor detects forces applied to the device, such as movement (up, down, sideways) and gravity.

Blob - A Blob is a data type that can store binary data, like images or multimedia files.

capacitive touch screen - A capacitive touch screen uses electrodes to sense objects touching the screen. Because the object must have conductive properties, a finger works but something like a stylus does not. Most touch-screen smartphones and computer monitors are capacitive.

civic data - Civic data is location data that's more easily understood by humans, such as a map or an address like 637 Park Street.

device-independent - A program or interface that runs software that produces similar results on a wide variety of hardware is also called device-independent.

File API - The File API allows a browser or application to upload files from local storage to a remote server without the need for a plug-in.

geodetic data - Geodetic data provides raw location data, such as longitude and latitude, or meters.

Geolocation API - The Geolocation API defines an interface that provides a device's location, usually using latitude and longitude coordinates. The API exposes the latitude and longitude to JavaScript in a Web page using the geolocation object.

gesture event - A gesture event is a type of touch event triggered by a multi-finger gesture. Gesture events receive event objects that contain touch properties.

local storage - Local storage is persistent data and is useful for things like online to-do lists, contact lists, calendars, and saved shopping cart data. You want this information to be available to the user after the browser closes and the user reopens it at some point. The information is held in persistent memory of Web applications and mobile devices.

platform-independent - The term platform-independent describes an application that can run on different desktop and mobile device operating systems, such as Microsoft Windows, Internet Explorer, Windows Phone, Mac OS X, Android, iOS, and Blackberry OS.

polling - Polling is the process in which a browser contacts a Web server periodically (sometimes constantly) to see if new information is available to present to the user. Technologies such as Comet refresh only a portion of a Web page. Comet and similar "push" technologies introduced polling.

resistive touch screen - A resistive touch screen is made up of several layers, the topmost of which flexes when pressed and pushes into the layer underneath. Sensors detect the pressure, which is how the system knows which part of the screen has been pressed. The touch screens used in hospitals and restaurants are often resistive.

session storage - Session storage is temporary data that's kept for only one session, until the browser is closed. All of the data in a session is saved in session storage and then erased from session storage when you close the browser tab or window.

touch event - A touch event is the action an application takes in response to a gesture.

touch object - In JavaScript, the touch object detects input from touch-enabled devices.

touchlist - A touchlist references touch objects; the touchlist includes all of the points of contact with a touch screen.

Web Hypertext Application Technology Working Group (WHATWG) - The Web Hypertext Application Technology Working Group (WHATWG) maintains a living HTML specification that includes APIs that were not originally part of the HTML5 specification. These include Geolocation, Web Workers, WebSockets, and File API.

Web Worker API - Web Workers are APIs that allow scripts to run in the background as parallel threads. These background threads can connect to multiple Web pages, fetch real-time data like stock updates, make network requests, or access local storage while the main HTML document responds to the user input like tapping, scrolling, and typing. Web Workers help keep your user interface responsive for users.

WebSocket API - WebSocket is an API that offers full-duplex communication, which is simultaneous two-way communication, through a single socket over the Internet. Developers use WebSockets mainly for real-time Web applications like chat, multiplayer online gaming, and stock quotes.

Lesson 10: JavaScript Coding for the Touch Interface, Device and Operating System Resources, and More- Chapter Exercise Answers

Knowledge Assessment

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. The action an application takes in response to a gesture is called a __touch__ event.
2. __Geodetic__ data provides raw location data, such as longitude and latitude, or meters.
3. __Web Workers__ are APIs that allow scripts to run in the background as parallel threads.
4. The __WebSocket__ API offers full-duplex, two-way communication through a single socket over the Internet.
5. The __File__ API allows a browser or application to upload files from local storage to a remote server without the need for a plug-in.
6. A program or interface that runs software that produces similar results on a wide variety of hardware is also called __device-independent__.
7. An __accelerometer__ is a device that measures acceleration, which is a change in speed over a period of time.
8. A __resistive__ touch screen is made up of several layers, the topmost of which flexes when pressed and pushes into the layer underneath. Sensors detect the pressure, which is how the system knows which part of the screen has been pressed.
9. A __Blob__ is a data type that can store binary data, like images or multimedia files.
10. Comet and similar "push" technologies introduced __polling__, in which a browser would contact the Web server periodically (sometimes constantly) to see if new information was available to present to the user.

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Which type of touch screen requires conductive properties?
   a. Capacitive
   b. Resistive
   c. Electronic
   d. none of the above
2. In JavaScript, which of the following contains a reference to all points of contact with a touch screen?
   a. Touch object
   b. Identifier
   c. Touchlist
   d. Manifest
3. Which API defines an interface that provides a device's location, usually using latitude and longitude coordinates?

   a. WebSocket
   b. Geolocation
   c. Web Workers
   d. File
4. Web Workers do *no*t use which of the following?
   a. `setTimeout`
   b. `setInterval`
   c. `XMLHttpRequest`
   d. getElementById
5. Which of the following are good examples of Web applications that benefit from WebSockets? (Choose all that apply.)
   a. Chat
   b. Address book
   c. Multiplayer online gaming
   d. Stock quotes
6. Which API enables you to upload images and immediately display thumbnails in HTML documents?
   a. WebSocket
   b. Geolocation
   c. Web Workers
   d. File
7. Which API uses ws rather than http when referencing URLs?
   a. WebSocket
   b. Geolocation
   c. Web Workers
   d. File
8. Which method allows users to save relatively large amounts of data that persists from browser session to browser session?
   a. localStorage
   b. `sessionStorage`
   c. `postMessage`
   d. `addEventListener`
9. Which method accesses a device's local camera and microphone stream?
   a. getUserMedia
   b. `sessionStorage`
   c. `addEventListener`
   d. `getCameraSound`
10. Which mobile device sensor detects the force of gravity along with any forces resulting from the movement of the device?
   a. GPS

b. Compass

[✓ c. Accelerometer]

d. Gyroscope

True / False

Circle T if the statement is true or F if the statement is false.

T [✓ F] 1. The W3C was formed by Apple, the Mozilla Foundation, and Opera Software to define and document the HTML5 specification.

[✓ T] F 2. In JavaScript, the touch object detects input from touch-enabled devices.

[✓ T] F 3. Civic data is location data that's more easily understood by humans, such as a map or an address like 637 Park Street.

[✓ T] F 4. Developers test their applications and users run the apps in the WinRT environment.

T [✓ F] 5. The press and tap gesture is an equivalent to a left-mouse click.

Competency Assessment

Scenario 10-1: Understanding Gestures

Jerome is a co-worker and budding developer who is experimenting with a touch-enabled application. He wants to know which gesture mimics a mouse click. What do you tell him?

Tell Jerome that several touch gestures accomplish the same tasks as mouse clicks. The tap is equivalent to a left-click, and the double tap mimics a left double-click. Both the press and tap, and the press and hold, are similar to right-clicks.

Scenario 10-2: Gathering Customer Location Data

Austin Energy and Light wants customers to log in to their Web site and use an application that pinpoints their exact location. The application must be very responsive to the user experience, and allow users to continue filling in form data. The location data will be sent to handheld devices that technicians use in the field to quickly locate customer homes and businesses. Which technologies do you suggest the company explore?

Tell your contact at Austin Energy and Light to explore geolocation and Web Workers. Geolocation will provide latitude and longitude coordinates, and Web Workers will help make the application run without delay.

Proficiency Assessment

Scenario 10-3: Understanding Device Motion

Vong is a programmer for a major smartphone manufacturer. She was recently assigned to work on a device motion project with a development team. She wants to quickly understand the difference between the Accelerometer, Compass, and Gyroscope sensors. What do you tell her?

Tell Vong that an accelerometer is a device that measures acceleration, which is a change in speed or direction over a period of time. The Accelerometer sensor detects forces applied to the device, such as movement (up, down, sideways) and gravity. The Compass sensor determines a device's orientation relative to the Earth's magnetic north pole. She can use the Compass sensor along with the appropriate APIs to create apps for geocaching and navigation, for example. The Gyroscope sensor uses motion (rotational forces) to detect the rotational velocity of the device along its three primary axes.

Scenario 10-4: Exploring Web Storage

Vong has returned with a new question about Web storage. She believes Web storage refers to saving files to a cloud service over the Web, but that doesn't make sense in relation to her smartphone project. How do you clarify Web Storage for her?

Although saving files to a cloud service is, technically, a kind of Web storage, Vong needs to learn about the Web Storage API. This API provides a client-side method for saving session information locally within the browser or device memory. The `localStorage` method allows users to save larger amounts of data from session to session (persistent data), whereas the `sessionStorage` method keeps data only for one session (until the browser is closed). The data is stored in key/value pairs for both types of Web storage.

Local storage is useful for things like online to-do lists, contact lists, calendars, and saved shopping cart data. You want this information to be available to the user after the browser closes and the user reopens it at some point. The information is held in persistent memory of Web applications and mobile devices. An example of the proper use of session storage is a ZIP code lookup program.

Lesson 10: JavaScript Coding for the Touch Interface, Device and Operating System Resources, and More – Code Fragments

Select code segments from the textbook have been included here to allow you to avoid retyping lengthy code segments. Each code segment herein is listed under its heading in the book.

Responding to the Touch Interface

The Bottom Line

Capturing and Responding to Gestures

For example, in the following code for a canvas drawing program, the startup function is called when the Web page loads. The program listens for a user touching the screen (the `touchstart` event), moving a finger (`touchmove`), and so on:

```
function startup() {
  var el = document.getElementsByTagName("cdraw")[0];
  el.addEventListener("touchstart", handleStart, false);
  el.addEventListener("touchmove", handleMove, false);
  el.addEventListener("touchend", handleEnd, false);
  el.addEventListener("touchcancel", handleCancel, false);
}
```

Let's look more closely at the `handleStart` function. It's declared in the following code, the same way other functions are declared in JavaScript except that the listener function must have one argument to represent the event. You can use any identifier as the event argument, but developers often use the letter "e" or an identifier that starts with the letter "e."

```
function handleStart(evt) {
  evt.preventDefault();
  var el = document.getElementsByTagName("cdraw")[0];
  var context = el.getContext("2d");
  var touches = evt.changedTouches;
for (var i=0; i<touches.length; i++) {
    ongoingTouches.push(touches[i]);
    var color = colorForTouch(touches[i]);
    context.fillStyle = color;
    context.fillRect(touches[i].pageX, touches[i].pageY, 4,
4);
  }
}
```

Detect Touch Screen Capability

GET READY. To check whether a user's device is touch-enabled, perform the following steps:

1. In an editing or app development tool, create a file named **L10-touch.html** with the following content:

```
<!doctype html>

<html>
  <head>
    <title>Detect Touch Screen</title>
    <meta charset="utf-8" />

    <style type="text/css">
      #canvas{background-color: dodgerblue;}
    </style>

    <script type="text/javascript">

        document.addEventListener("DOMContentLoaded", init,
   false);

        function init() {
            var canvas = document.getElementById("canvas");

            if ("ontouchstart" in document.documentElement)
   {
                canvas.addEventListener("touchstart",
   detect, false);
            }
            else {
                canvas.addEventListener("mousedown",
   detect, false);
            }
        }

        function detect() {
            if ("ontouchstart" in document.documentElement)
   {
                alert("Touch screen device detected!");
            }
            else {
                alert("No touch screen device detected.");
            }
        }
    </script>

  </head>

  <body>
    <canvas id="canvas" width="100" height="100"></canvas>
   <br />
   <p>Click the box to start touch screen detection.</p>
  </body>
</html>
```

Coding Additional HTML5 APIs

The Bottom Line

Create a Geolocation Application

GET READY. To create a geolocation application that displays the user's current latitude and longitude, perform the following steps:

1. In an editing or app development tool, create a file named **L10-geolocation.html** with the following content:

```
<!doctype html>
<html>
<head>
   <title>Geolocation Example</title>
   <meta charset="utf-8" />
   <script>
       var messageDiv = document.getElementById('message');

       function initLocation() {

           var geolocation = navigator.geolocation;

           if (geolocation) {
               try {

   navigator.geolocation.getCurrentPosition(
                              successCallback,
                              errorCallback
                        );

               } catch (err) {
                   messageDiv.innerHTML = 'Error';
```

```
                }
            } else {
                messageDiv.innerHTML = 'Your browser does
   not support geolocation.';
            }

        }

        function successCallback(location) {
            message.innerHTML = "<p>Latitude: " +
   location.coords.latitude + "</p>";
            message.innerHTML += "<p>Longitude: " +
   location.coords.longitude + "</p>";
        }

        function errorCallback() {
            messageDiv.innerHTML = 'There was an error
   looking up your position';
        }
   </script>
</head>

<body onload="initLocation()">
   <div id="message">Looking Up Location</div>
</body>
</html>
```

Understanding Web Workers

As a simple example of message passing with a Web Worker, the following is a script in the main HTML document:

```
var worker = new Worker("doWork.js");
// Watch for messages from the worker
worker.onmessage = function(e){
  // The message from the worker
  e.data
};

worker.postMessage("start");
```

The following code might appear in the doWork.js file:

```
onmessage = function(e){
  if ( e.data === "start" ) {
    // Perform an action or computation
    done()
  }
};
function done(){
  // Send results to main document
  postMessage("Hello, I am done");
}
```

Instead of using the `onmessage` event handler, you could use `addEventListener()` in the main HTML document, like this:

```
worker.addEventListener('message', function(e) {
```

Create and Run a Web Worker

GET READY. To create and run a Web Worker, perform the following steps:

1. In an editing or app development tool, create a file named **L10-worker.html** with the following content:

```
<!doctype html>
<html lang="en">
<head>
<script>
var worker = new Worker('doWork.js');

// Send a message to start the worker and pass a variable
   to it
var info = 'Web Workers';
worker.postMessage(info);

// Receive a message from the worker
worker.onmessage = function (event) {
  // Do something
  alert(event.data);
};
</script>
<title>Web Workers Example</title>
</head>
<body>
</body>
</html>
```

2. Create a JavaScript file named **doWork.js** in the same folder as worker.html, with the following content:

```
onmessage = function(event) {
  var info = event.data;
```

```
                  var result = 'Hello ' + info + ' everywhere';
  postMessage(result);

};
```

Understanding WebSockets

Create a WebSocket to Test Browser Compatibility

GET READY. To create a WebSocket that tests whether your browser supports WebSockets, perform the following steps:

1. In an editing or app development tool, create a file named **L10-WebSocket.html** with the following content:

```
<!doctype html>
<html>
<head>
<script type="text/javascript">
    function WebSocketTest() {
        if ("WebSocket" in window) {
            alert("Your browser supports WebSockets.");
            // Open a WebSocket
            var socket = new
   WebSocket("ws://localhost:9998/echo");
            socket.onopen = function () {
                // Connected, send data
                socket.send("Connected");
                alert("Connected.");
            };
            socket.onmessage = function (e) {
                var received_msg = e.data;
                alert("Message received.");
            };
            socket.onclose = function () {
                // WebSocket closed
                alert("Connection closed.");
            };
        }
        else {
            // Browser doesn't support WebSockets
            alert("Your browser does not support
   WebSockets.");
        }
    }
</script>
</head>
<body>
```

```
<div>
   <a href="javascript:WebSocketTest()">Click to run
   WebSocket demo</a>
</div>
</body>
</html>
```

Using File API for File Uploads

Check Browsers for File API Compatibility

GET READY. To check the major browsers for File API compatibility, perform the following steps:

1. In an editing or app development tool, create an HTML file with the following content:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>File API Browser Support Check</title>
<script>
if (window.File && window.FileReader && window.FileList &&
   window.Blob) {
    alert('Hurray! This browser supports File APIs.');
} else {
  alert('This browser does not fully support File APIs.');
}
</script>
</head>

<body>
</body>
</html>
```

Accessing Device and Operating System Resources

The Bottom Line

Use the localStorage Object

GET READY. To save a value to local storage, perform the following steps:

1. In an editing or app development tool, create a file named **L10-localStorage.html** with the following content:

```
<!doctype>
<html>
<head>
```

```
<title>localStorage Example</title>
    <script type="text/javascript">
      function load() {
        var value = localStorage.getItem("myKey");

          if (!value) {
            alert("Item not found, adding to
   localStorage");
            localStorage.setItem("myKey", "myValue");
            }
          else {
            alert(value + " found!");
          }
      }
</script>
</head>
<body onload="load()">
</body>
</html>
```