LESSON 4

98-361 Software Development Fundamentals

4.1 Understand Web Page Development

4.2 Understand Microsoft ASP.NET Web Application Development

4.3 Understand Web Hosting

4.4 Understand Web Services

MTA Software Fundamentals 4 Test

LESSON 4.1

98-361 Software Development Fundamentals

Understand Web Page Development

LESSON 4.1

98-361 Software Development Fundamentals

Lesson Overview

Students will understand Web page development.

In this lesson, you will learn about:

- HTML
- CSS
- JavaScript

Review Terms

- Cascading Style Sheets (CSS) —a Hypertext Markup Language (HTML) specification ,developed by the World Wide Web Consortium (W3C), that allows authors of HTML documents and users to attach style sheets to HTML documents.
- HTML—acronym for Hypertext Markup Language, the markup language used for documents on the World Wide Web. A tag-based notation language used to format documents that can then be interpreted and rendered by an Internet browser.
- JavaScript—a scripting language developed by Netscape Communications and Sun Microsystems that is loosely related to Java. JavaScript, however, is not a true object-oriented language, and it is limited in performance compared with Java because it is not compiled.

What is HTML?

- The language that Web servers and browsers use to define the elements of a Web page.
- HTML uses tags to mark elements in a document, such as text and graphics, to indicate how Web browsers should display these elements to the user and should respond to user actions such as activation of a link by means of a key press or mouse click.
- Examples of tags:
  - `<html>`
  - `<body>`

What is HTML? (continued)

- Used primarily to format data on a page.
- Doesn’t contain any advanced support for doing complex operations; it just serves to lay out the contents in a readable way on the Web page.
- When the browser receives an HTML document, it converts the HTML description to a screen display.

What is HTML? (continued)

- A text-based language that you can view and edit in a standard text editor like Notepad.
- Consists of the text on the Web page, along with “markup tags” that indicate to the browser how that text should be displayed.
- Specifies items such as the font to use for sections of text, where to display embedded images, and of course, hyperlinks that enable you to link to other Web pages.

HTML Example Code

- HTML code consists of a series of tags denoted by angle brackets around a tag name, such as the `<html>` tag in the example below.
- At the end of the document is another tag, `</html>`. The forward slash indicates that this is the end tag that corresponds to the `<html>` tag at the start of the document. Everything between the two tags is called the `<html>` element.

```
<html>
<body>
Content
here
</body>
</html>
```

What is a cascading style sheet (CSS)?

- A language/code that completely separates the text displayed on a Web page (which is created in HTML code) and information that describes how to display that text.
- Includes typographical information on how the page should appear, such as the font of the text in the page.
- Directs the manner in which the style sheets for an HTML document and the user's style will blend.

Why use CSS?

- Improves content accessibility by allowing the same content to be displayed in different styles depending on the rendering method: on screen, by voice, or using Braille-based devices.
- Allows more flexibility and control over the way the content is presented.
- Provides a more efficient way for multiple pages to share the same formatting.

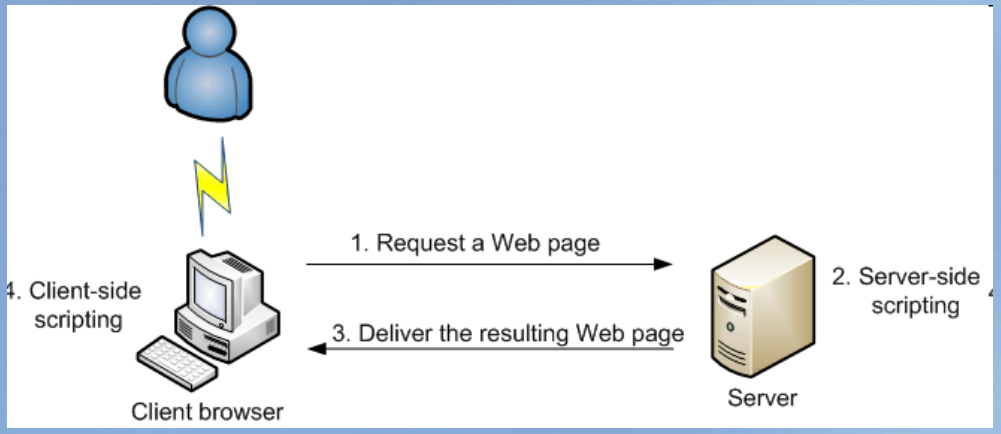
CSS Example Code

- The first section of the style sheet code here defines that all content within the HTML **body** element will use the Verdana font with a point size of 9 and will align it to the right.

```
body
{
    font-family: Verdana;
    font-size: 9pt;
    text-align: right;
}
div
{
    font-family: Georgia;
}
```

Client-side vs. Server-side

- A Uniform Resource Locator (URL) is entered, or a link is clicked, and the browser requests that particular page from the server (1)
- The server finds the page, opens it, and runs server-side scripts within it (2).
- After all the server-side scripts are processed, the results are sent back to the browser (3).
- The result is a page created from the HTML and CSS code, and optionally, client-side scripting code. If there is client-side scripting code on the page (for example, JavaScript), the browser will process it and display it to the user (4).



Server-side Scripting

- ASP.NET, ASP, or PHP.
- Used for generating Web pages by request.
- All data on the pages will be current because it was created moments before being sent back to the browser.
- Examples: online shops, auction sites, online forums, and bulletin boards.

Client-side Scripting

- Used to make pages interactive after they are sent to the browser.
- A common usage is to check the data that the user has entered in a form on the page.
  - If the user forgets to enter his or her full name or misspells the e-mail address, the client-side script will warn the user.
- More responsive and faster for the user, unlike server-side scripts, which depend on the server and Internet traffic between the user's computer and the server.

What is JavaScript?

- Client-side scripting language used on millions of Web pages
- Adds interactivity to the browser and Web pages
- Complements very popular server-side programming languages and platforms, like ASP.NET

Why use JavaScript?

- Allows for dynamic content (while HTML is static).
- Adds functionality, interactivity, and dynamic effects.
- For example, JavaScript allows you to show the current date on the Web page without having to edit the code and upload it to the server every night.

Example: JavaScript Code

```
<html>
<head>
   <title>My page</title>
   <script type="text/javascript" language="javascript">
   <!-
   function say(text)
   {
      alert(text);
   }
   say("The page is loading!");
   //-->
   </script>
</head>
 <body>

</body>
</html>
```

LESSON 4.1

98-361 Software Development Fundamentals

No Student Lab 4.1

Microsoft

LESSON 4.2

98-361 Software Development Fundamentals

Understand Microsoft ASP.NET Web Application Development

Lesson Overview

Students will understand ASP.NET Web application development.

In this lesson, you will learn about:

- The page life cycle
- The event model
- State management
- Client-side vs. server-side programming

Review Terms

- client-side program—a program that is run on a client rather than on a server.
- event model—allows the developer to program Web pages using an event-based model that is similar to the event model used in client applications.
- page life cycle—when an ASP.NET page runs, the page performs a series of processing steps. These steps include initialization, instantiating controls, restoring and maintaining state, running event handler code, and rendering.

Review Terms (continued)

- server-side program—a program that is run on a server rather than on a client.
- state management—the process by which you maintain state and page information over multiple requests for the same page or different pages.
- Web application—software on a set of clients and servers that cooperates to provide the solution to a problem.

Page Life Cycle

- The ASP.NET page life cycle refers to the series of steps that a page goes through from creation to disposal.
- Important to understand the page life cycle so that code can be written at the appropriate life cycle stage for the intended effect.

Page Life Cycle Stages

- Page request—happens before the page life cycle begins.
  - Once a page is requested, ASP.NET determines whether the page needs to be parsed and compiled (therefore beginning the life of a page), or whether a cached version of the page can be sent
- Start—page properties such as *Request* and *Response* are set. At this stage, the page also determines whether the request is a postback or a new request, and sets the *IsPostBack* property accordingly.

Page Life Cycle Stages (continued)

- Initialization—controls on the page are available and each control's *UniqueID* property is set. Any themes are also applied to the page.
  - If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state.
- Load—if the current request is a postback, control properties are loaded with information recovered from view state and control state.

Page Life Cycle Stages (continued)

- Validation—the *Validate* method of all validator controls is called, which sets the *IsValid* property of individual validator controls and of the page.
- Postback event handling—if the request is a postback, any event handlers are called.
- Rendering
  - Before rendering, page and control view states are saved.
  - During rendering, the page calls the *Render* method for each control, providing a text writer that writes its output to the *OutputStream* of the page's *Response* property.

Page Life Cycle Stages (continued)

- Unload—called after the page has been fully rendered, has been sent to the client, and is ready to be discarded. Page properties such as *Response* and *Request* are unloaded, and any cleanup is performed.

Event Model

- The event-based model in ASP.NET is similar to that in client applications.
  - Example: add a button to an ASP.NET Web page and write an event handler for the button's click event.
- In ASP.NET Web pages, events associated with server controls originate on the client but are handled on the server.
  - It is different from client-based applications, where the events are raised and handled on the client.

Event Model Steps

1. The event information is captured on the client and an event message is transmitted to the server through a Hypertext Transfer Protocol (HTTP) post.
2. The page interprets the post to determine what event occurred and calls the appropriate method in the code on the server to handle the event.

- ASP.NET handles the task of capturing, transmitting, and interpreting the event.

State Management

- A new instance of the *Webpage* class is created each time the page is posted to the server.
- In traditional Web programming, all information associated with the page and the controls on the page would be lost with each round trip to and from the server.
  - Example: If a user enters information into a text box, that information would be lost in the round trip from the browser to the server.

State Management (continued)

- In ASP.NET, state management refers to the several client-side and server-side options that are available to help you preserve data on both a per-page and an application-wide basis.
- Client-side examples:
  - View state, control state, hidden fields, cookies
- Server-side examples:
  - Application state, session state, profile properties, database support

State Management: View State

- Use when storing small amounts of information for a page that will post back to itself.
- The *ViewState* property provides a structure for retaining values between multiple requests for the same page.
- When the page is processed, the current state of the page and controls is hashed into a string and saved in the page as a hidden field.
- When the page is posted back to the server, the page parses the view-state string at page initialization and restores property information in the page.

State Management: Control State

- Use when storing small amounts of state information for a control between round trips to the server.
- You need to store control-state data for a control to work properly.
- The *ControlState* property allows you to persist property information that is specific to a control, and it cannot be turned off like the *ViewState* property can.

State Management: Application State

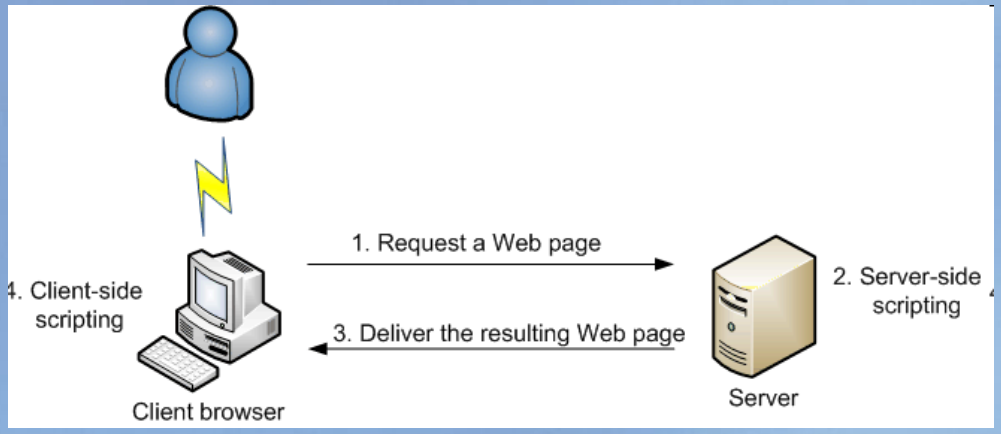
- Use when storing infrequently changed, global information that is used by many users and security is not an issue. Do not store large quantities of information in application state.
- ASP.NET provides application state via the *HttpApplicationState* class as a method of storing global application-specific information that is visible to the entire application.
- Store application-specific values in application state, which is then managed by the server.

State Management: Session State

- Use when storing short-lived information that is specific to an individual session and security is an issue.
- ASP.NET provides a session state, which is available as the *HttpSessionState* class, as a method of storing session-specific information that is visible only within the session.
- Store session-specific values and objects in session state, which is then managed by the server and available to the browser or client device.

Client-side vs. Server-side

- A Uniform Resource Locator (URL) is entered, or a link is clicked, and the browser requests that particular page from the server (1).
- The server finds the page, opens it, and runs all server-side scripts within it (2).
- After all the server-side scripts are processed, the results are sent back to the browser (3).
- The result is a page that has Hypertext Markup Language (HTML) and cascading style sheets (CSS) code, and optionally, client-side scripting code. If there is client-side scripting code on the page (for example, JavaScript), the browser will process it and display it to the user (4)



Server-side Scripting

- Server-side scripting (like ASP.NET, ASP, or PHP) is used for generating Web pages by request.
- All data on such pages will be current, as they are created moments before being sent back to the browser.
- Examples: online shops, auction sites, online forums and bulletin boards.

Client-side Scripting

- Used to make pages interactive after they are sent to the browser.
- A common use of client-side scripting is to check the data that the user has entered in a form on the page.
  - If the user forgot to enter his or her full name or misspelled the e-mail address, the client-side script will warn the user.
- Client-side scripts are also much more responsive and faster for the user, unlike the server-side scripts, which depend on the server and Internet traffic between the user's computer and the server, as the request has to be sent to the server and then back again.

No Student Lab 4.2

LESSON 4.3

98-361 Software Development Fundamentals

Understand Web Hosting

LESSON 4.3

98-361 Software Development Fundamentals

Lesson Overview

Students will understand Web hosting.

In this lesson, you will learn about:

- Creating virtual directories for Web sites
- Deploying Web applications
- The role of Internet Information Services (IIS)

Web Hosting

- The process involved with making a website accessible via the World Wide Web.
- While you can host a site on your own server, it is better to use an Internet service provider (ISP).
- An ISP will host your website and will usually provide:
  - Space on a server
  - Maintenance and support
  - E-mail capabilities
  - Security and stability

Review Terms

- HTTP—acronym for Hypertext Transfer Protocol, the protocol used to carry requests from a browser to a Web server and to transport pages from Web servers back to the requesting browser.
- ISP—acronym for Internet service provider, a business that supplies Internet connectivity services to individuals, businesses, and other organizations.
- World Wide Web—the total set of interlinked hypertext documents residing on HTTP servers all around the world.

Virtual Directory

- A directory name (path) that maps to a physical directory on a local or remote server
- Creates a path to the physical directory that does not necessarily have to use the actual directory path name
- Allow multiple websites and Uniform Resource Locators (URLs) to map to the same content without having to change the path structure of the content directory in each application

Deploying a Web Application

- Deployment makes a Web application available to other users through the Internet.
- Requires deployment of the application from the development server to the production server.
- On the production server, the application will have access to live databases (if needed).
- When deployed, the final version of the website is copied to a private folder on the production Web server.

Internet Information Services (IIS)

- Windows-based Web service that delivers content, like Web pages, using HTTP over the World Wide Web
- Delivers Hypertext Markup Language (HTML) documents to clients, primarily Web browsers
- Supports server-side scripting, such as ASP.NET
- Is installed on the server and is the backbone of the Microsoft Windows Server operating system

LESSON 4.3

98-361 Software Development Fundamentals

No Student Lab 4.3

LESSON 4.4

98-361 Software Development Fundamentals

Understand Web Services

Lesson Overview

Students will understand Web services.

In this lesson, you will learn about:

- Web services consumed by client applications
- Accessing Web services from a client application
- SOAP and WSDL

Guiding Questions

1. How are Web services accessed and consumed by client applications?
2. What is SOAP?
3. What is WSDL?

Activator

1. How would you communicate with someone who did not speak your language?
2. How is this similar to how applications interact with Web services?

Review Terms

- SOAP—acronym for Simple Object Access Protocol, a simple, XML-based protocol for exchanging structured and type information on the Web
- WSDL—acronym for Web Services Description Language, an Extensible Markup Language (XML) format that allows for better interoperability among Web services and development tools

Web Services

- Extend the World Wide Web infrastructure to provide a means for software to connect to other applications
- Allow applications to access Web services via Web protocols and data formats such as Hypertext Transfer Protocol (HTTP), XML, and SOAP
  - No need to worry about how each Web service is implemented
- Combine the best aspects of component-based development and the Web
- A cornerstone of the Microsoft .NET Framework programming model

What is a Web service?

- Used when a program wants to interact with another program through the Internet.
- A program is required to process the communication.
- The program might change its implementation from one language to another, but the functionality of the Web service stays the same.
- The Web service is the abstract functionality.
  - The software and hardware is the concrete implementation.

What do Web services do?

- Provide reusable operations for other applications
- Can serve as a component of another application
  - Examples: weather reports, currency conversion
- Provide interoperability between different platforms by giving applications a way to communicate

How does a Web service work?

1. A company (the requester) wants to use the Web service of another company (the provider) .
2. Before they can pass information to each other, the requester and provider must agree on both the meaning and method of the information exchange.
3. The method of exchange is documented in an interface described in a machine-processable format such as WSDL.
4. Outside systems interact with the Web service in a manner prescribed by WSDL using SOAP.

WSDL

- It is written in XML and is used to locate a Web service and describe its functions.
- It defines the information formats, data types, transport protocols, and transport serialization formats.

Element	Meaning
<*types*>	The data types used by the Web service
<*message*>	The messages used by the Web service
<*portType*>	The operations performed by the Web service
<*binding*>	The communication protocols used by the Web service

SOAP

- A generalized XML messaging framework for exchanging structures and type information on the Web (over HTTP).
- It contains no application or transport semantics, which makes it highly modular and extensible.
- To access a Web service, you use SOAP.

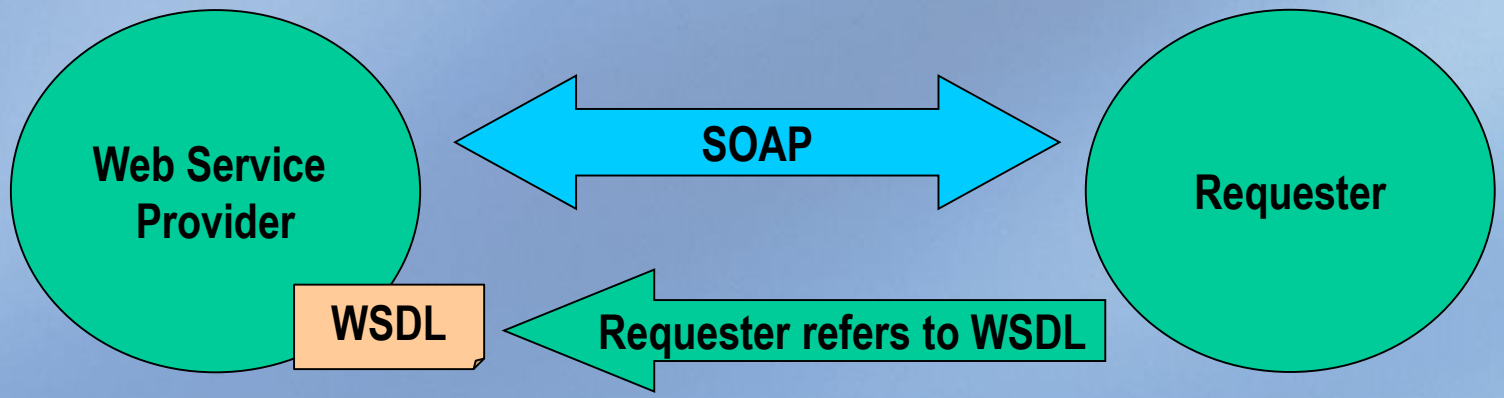
SOAP (continued)

- A SOAP message is an XML document containing the elements in the table.

Element	Meaning
<*soap:Envelope*>	Identifies the XML document as a SOAP message
<*soap:Header*>	Header information
<*soap:Body*>	Call and response information
<*soap:Fault*>	Errors and status information

Lesson Review

A company wants to provide a Web service that allows outside applications to access sports statistics. Draw a diagram showing the interaction of WSDL and SOAP in the context of this Web service. Include labels for the requester and the provider, and use arrows to illustrate the flow of information.


Web Service Provider
SOAP
Requester
WSDL
Requester refers to WSDL

LESSON 4

98-361 Software Development Fundamentals

Complete the QUIA Test

MTA Software Fundamentals 4 Test