جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

College of Sciences
Intelligent Medical System Department

Lecture 2

Introduction of HTML Basics

Subject: Web Programming

Level: Third

Lecturer: Asst. Lect. Ali Saleem Haleem

HTML Tags

Now we're going to describe how to implement elements for a web page. To implement an element for a web page, you'll need to use tags. For example, if you want to implement a strong element (in order to put emphasis on the element's content and display using boldface), surround the content with tags. Here's how to implement a strong element for the word "very":

```
Caiden was <strong>very</strong> happy when her tooth finally came out.
```

The use of tags is the key characteristic of a markup language. Why is it called "markup"? A markup language "marks up" a document by surrounding parts of its content with tags. Web pages are implemented with HTML, which stands for Hypertext Markup Language. You already know what "markup" means. The "hyper" in "Hypertext" refers to HTML's ability to implement hyperlinks. A hyperlink (or link for short) is where you click on text or an image in order to jump to another web page. So, HTML is a language that supports markup tags for formatting and the ability to jump to other web pages.

Most, but not all, HTML tags come in pairs with a start tag and an end tag. For example, the following code uses an <h1>start tag and an </h1>end tag:

```
<h1>Today's Weather</h1>
```

Note that each tag is surrounded by angled brackets, and the end tag starts with "</". The ***h1*** heading tags cause the enclosed text ("Today's Weather") to display like a heading. That usually means that the enclosed text will be *boldfaced*, *large*, and *surrounded by blank lines*. Note the use of the term "usually." The official HTML standard/specification often doesn't specify precise display characteristics. Consequently, not all browsers display tags in the exact same way. Besides ***h1***, there are other heading elements h2through h6. The element h1generates the largest heading, and h6generates the smallest. Use a heading tag whenever you want to display a heading above other text. Headings are usually at the top of the page, but they can be in the middle of the page as well. There are two types of elements:

container elements and *void elements*. A container element (usually called simply a "container") has a start tag and an end tag, and it contains content between its two tags. For example, the h1element is a container. On the other hand, a void element has just one tag, and its content is stored within the tag.

Structural Elements

Take a look at FIGURE 1. It shows the HTML source code that was used to generate the Kansas City Weather web page shown earlier. What's source code, you ask? With many programming languages, two types of code are associated with a single program. There's *source code*, which the programmer enters, and there's *executable code*, which is low-level code that the computer hardware understands and executes (runs). Executable code is generated from the source code with the help of something called a compiler. As you learn HTML, there's no need to worry about executable code because it is generated automatically behind the scenes and you never see it.

Next, we'll describe the different sections of code in Figure 1. Let's start with the really important HTML constructs that you'll use as the basic framework for all your web pages: *doctype, html, head, and body*. The doctype construct is considered to be an instruction, not an element, and it goes at the top of every web page. The html, head, and body elements form the basic structure of a web page, so we'll refer to those elements as structural elements. The doctype instruction plus the structural elements form the skeleton code shown in FIGURE 2. You can use that skeleton code for all your web pages. The first construct, <!DOCTYPE html>, tells the browser what type of document the web page is. Its HTML value (in <!DOCTYPE html>) indicates that the document is an HTML document,

After the doctype instruction comes the HTML element. It's a container, and it contains/surrounds the rest of the web page. Its start tag includes ***lang="en"***, which tells the browser that the web page is written in English. The head and body elements are also containers. The head element surrounds elements that provide information associated with the web page as a whole. The body element surrounds elements that display content in the web page. Container elements must be properly nested, meaning that if you start a container inside another container, you must end the inner

container before you end the outer container. Because the body element starts inside the html element, the </body>end tag must come before the </html>end tag. In Figure 2, note how the head and body elements are properly nested within the html element.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="author" content="John Dean">
<meta name="description" content="Kansas City weather conditions">
<title>K.C. Weather</title>
<style>
  h1 {text-align: center;}
  hr {width: 75%;}
</style>
</head>

<body>
<h1>Kansas City Weather</h1>
<hr>
<p>
  It should be pleasant today with a high of 95 degrees.<br>
  With a humidity reading of 30%, it should feel like 102 degrees.
</p>
<div>
  Tomorrow's temperatures:<br>
  high 96, low 65
</div>
</body>
</html>
```

Figure 1:Source code for Kansas City Weather web page

```
<!DOCTYPE html>
<html lang="en">
<head>
  :
  :
</head>

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

Figure 2:Skeleton code using just doctype and the structural elements

Title Element

Let's now dig a little deeper into the head element. The head code in FIGURE 3 comes from the weather page, but in the interest of keeping things simple, we've omitted its style element. The head element contains two types of elements: *meta* and *title*. In your web pages, you should position them in the order shown in Figure 3, meta and then title. Remember that the head element surrounds elements associated with the web page as a whole. The web page's title pertains to the entire web page, so its title element goes within the head container. The title element's contained data (e.g., "K.C. Weather") specifies the label that appears in the browser window's title bar. With browsers that support tabbed windows, that tab is considered to be the browser's title bar. The official HTML standard requires that every head container contains a title element. Besides providing a label for your browser window's title bar, what's the purpose of the title element? (1) It provides documentation for someone trying to maintain your web page, and (2) it helps web search engines find your web page.

```
<head>
<meta charset="utf-8">
<meta name="author" content="John Dean">
<meta name="description" content="Kansas city weather conditions">
<title>K.C. Weather</title>
</head>
```

Figure 3:head container for Kansas City Weather web page

Meta Element

In Figure 3, note the meta elements within the head container. The meta elements provide information about the web page. If you look up "meta" in a standard dictionary, you'll probably see a confusing definition. In everyday speech and in HTML, "meta" means "about itself." There are many different types of meta elements—some you should always include, but most are just optional. The meta element is a void element (not a container), so it does not have an end tag. Note how

there are no end tags for the three meta elements in Figure 3. Many web programmer's end their void elements with a space and a slash. For example:

```
<meta charset="utf-8" />
```

The space and slash are a bit outdated, and we recommend omitting them. If you decide to include them, be consistent.

HTML Attributes

Before we formally introduce a few of the different types of meta elements, we first need to discuss attributes, which are used in all meta elements. Container elements provide information between their start and end tags. Void elements (including the meta element) have no end tags, so they can't provide information that way. Instead, they provide information using attributes. In the following example, charset is an attribute for a meta element:

```
<meta charset="utf-8">
```

Most attributes have a value assigned to them. In this example, charset is assigned the value "utf-8". Although most attributes have a value assigned to them, some do not. Later on, we'll see some attributes that appear by themselves, without a value. You should always surround attribute values with quotes, thus forming a string.

Attributes are more common with void elements, but they can be used with container elements as well. Here's an example of a container element that uses an attribute:

```
<html lang="fr">
  •
  •
  •
</html>
```

The Lang attribute tells the browser that the element is using a particular language for its content. You can use the Lang attribute for any element. Here we're using it for the html element, so it means that the entire web page uses French. You're not required to use the lang attribute, but we recommend that you do include it for the html element. For web pages written in English, use

```
<html lang="en">.
```

Why would you want to specify an element's language? The W3C's Internationalization Activity group (https://www.w3.org/International/questions/qa-lang-why.en) provides quite a few good reasons, and here are a few of them:

- Help search engines find web pages that use a particular language.
- Help spell-checker and grammar-checker tools work more effectively.
- Help browsers use appropriate fonts.
- Help speech synthesizers pronounce words correctly.

Meta Charset Element

Now that you've learned syntax details for element attributes, it's time to focus on semantic details. Syntax refers to the punctuation rules for code. Semantics refers to the meaning of the code. First up the semantics for the meta charset element. When a web server transmits a web page's source code to an end-user's computer, the web server doesn't transmit the source code's characters the way you see them in this book or on a keyboard. Instead, it transmits coded representations of the source code's characters. The coded representations are in binary (0, and 1). There are different encoding schemes, and in order for the receiving end of a transmission to understand the transmitted binary data, the receiver has to know the encoding scheme used by the sender. For web page transmissions, the meta charset element specifies the encoding scheme. Normally, you should use a charset value of "utf-8" because all modern browsers understand that value. If you omit the meta charset

element, your web page will usually work because most browsers assume UTF-8 encoding by default.

Meta Name Element

Most of the meta elements use the name attribute to specify the type of information that's being provided. Common values for the meta name attribute are author, description, and keywords. Here's an example with an author value for a name attribute:

```
<meta name="author" content="John Dean">
```

The name and content attributes go together. The name attribute's value specifies the type of thing that the content attribute's value specifies. So in this example, with the name attribute specifying "author," the content attribute specifies the author's name ("John Dean"). Why is knowing the author's name important? Often, the person who fixes or enhances a web page is different from the person who originally wrote the web page. By specifying the author, the fixer/enhancer knows whom to ask for help.

In the following examples, the name attribute uses the values "description" and "keywords":

```
<meta name="description" content="Kansas City weather conditions"
<meta name="keywords" content="KC, weather, meteorology, forecast"
```

The meta description element and also the meta keywords element help web search engines find your web page. In addition, the meta description element helps the person reading the code learn the purpose of the web page.

Body Elements (hr, p, br, div)

In FIGURE 4, which shows the body container code, note the ***h1, hr, p, br, and div elements.*** We've already talked about the h1heading element, so now let's focus on the other elements. The hr element is used to render a horizontal line. When a

browser renders an element, it figures out how the element's code should be displayed. To keep things simple, you can think of "render" as a synonym for "display." The "h" in hr stands for horizontal. The "r" in hr stands for rule, presumably because a rule is another name for a ruler, which can be used to make a straight line. The hr element is a void element, so it uses just one tag, <hr>.

```
<body>
<h1>Kansas City Weather</h1>
<hr>
<p>
  It should be pleasant today with a high of 95 degrees.<br>
  With a humidity reading of 30%, it should feel like 102 degrees.
</p>
<div>
  Tomorrow's temperatures:<br>
  high 96, low 65
</div>
</body>
```

Figure 4: body container for Kansas City Weather web page

The ***p element*** is a container for a group of words that form a paragraph. Normally, browsers will render a p element's enclosed text with a blank line above the text and a blank line below it. In Figure 4, note how we indented the text between the start tag and the end tag. Whenever you've got a p element whose enclosed text is greater than one line, you should put the start and end tags on separate lines and indent the enclosed text. That rule is an example of a coding-style convention rule (or style rule for short). Whenever you write a program, including an HTML program, it's important to follow standard coding-style conventions, so your program is easy to read by you and also by future programmers who need to understand your program.

A ***div element*** is also a container for a group of words, but it's more generic, so the words don't have to form sentences in a paragraph. div stands for division because a division can refer to a part of something that has been divided, and a div element is indeed a part of a web page. Normally, the div element causes its enclosed text to have single line breaks above and below it. If a div element's enclosed text is greater

than one line, then proper style suggests putting the tags on separate lines and indenting the enclosed text.

Except for single line breaks instead of blank lines, the characteristics for a div element are the same as for a p element. So when should you use a p element versus a div element? Use a p element if the enclosed text forms something that would normally be considered a paragraph. On the other hand, use a div element if the enclosed text is related in some way, but the text would not normally be considered a paragraph. If you use the p element only for bona fide paragraphs, then the rest of the web page can process p elements as paragraphs, and you avoid including non-paragraphs in that processing.

Finally, there's the ***br element,*** which is used to render a new line. In Figure 4, note this line:

```
It should be pleasant today with a high of 95 degrees.<br>
```