Getting Start with HTML5
IMED 1316 and ARTC 1317: Lecture and Lab Notes and Exercises
Instructor: Dan Dao

HTML is the main markup language for describing the structure of web pages.

HTML stands for HyperText Markup Language. HTML is the basic building block of World Wide Web.

Hypertext is text displayed on a computer or other electronic device with references to other text that the user can immediately access, usually by a mouse click or key press.

Apart from text, hypertext may contain tables, lists, forms, images, and other presentational elements. It is an easy-to-use and flexible format to share information over the Internet.

Markup languages use sets of markup tags to characterize text elements within a document, which gives instructions to the web browsers on how the document should appear.

HTML was originally developed by Tim Berners-Lee in 1990. He is also known as the father of the web. In 1996, the World Wide Web Consortium (W3C) became the authority to maintain the HTML specifications. HTML also became an international standard (ISO) in 2000. HTML5 is the latest version of HTML. HTML5 provides a faster and more robust approach to web development.

Tip: Our HTML exercise will help you to learn the fundamentals of the latest HTML5 language, from the basic to advanced topics step-by-step. If you're a beginner, start with the basic section and gradually move forward by learning a little bit every day.

What You Can Do with HTML

There are lot more things you can do with HTML.

- You can publish documents online with text, images, lists, tables, etc.
- You can access web resources such as images, videos or other HTML document via hyperlinks.
- You can create forms to collect user inputs like name, e-mail address, comments, etc.
- You can include images, videos, sound clips, flash movies, applications and other HTML documents directly inside an HTML document.
- You can create offline version of your website that work without internet.
- You can store data in the user's web browser and access later on.
- You can find the current location of your website's visitor.

The list does not end here, there are many other interesting things that you can do with HTML. You will learn about all of them in detail in upcoming exercises.

Note: HTML as described earlier is a markup language not a programming language, like Java, Ruby, PHP, etc. You need a web browser to view the HTML pages. The web browsers do not display the HTML tags, but uses the tags to interpret the content of the web pages.

What This Exercise Covers

This HTML exercise series covers all the fundamentals of HTML, including the idea of elements and attributes, way of formatting the text using HTML tags, methods of adding the style information to the document, technique of inserting images and tables, process of creating lists and forms as well as method of including other HTML documents inside the current document, and so on.

Once you're familiar with the basics, you'll move on to next level that explains the concept of doctype, methods for creating the web page layouts, importance of adding meta information to the web pages, way of adding scripts, techniques of showing special characters, anatomy of a URL, and more.

Finally, you'll explore some advanced features introduced in HTML5 such as new input types, drawing graphics on the webpage, including audios and videos in the document, storing data on client-side using web storage, caching files, performing background work with web worker, as well as getting user's geographical coordinates, creating drag and drop application, and so on.

Tip: Every exercise in this exercise contains lots of real-world examples that you can try and test using an online editor. These examples will help you to better understand the concept or topic. It also contains smart workarounds as well as useful tips and important notes.

HTML Get Started

An HTML file is simply a text file saved with an .html or .htm extension.

Getting Started

In this exercise you will learn how easy it is to create an HTML document or a web page. To begin coding HTML you need only two stuff: a simple-text editor and a web browser.

Well, let's get started with creating your first HTML page.

Creating Your First HTML Document

Let's walk through the following steps. At the end of this exercise, you will have made an HTML file that displays "Hello world" message in your web browser.

Step 1: Creating the HTML file

Open up your computer's plain text editor and create a new file.

Tip: We suggest you to use Notepad (on Windows), TextEdit (on Mac) or some other simple text editor to do this; don't use Word or WordPad! Once you understand the basic principles, you may switch to more advanced tools such as Adobe Dreamweaver.

Step 2: Type some HTML code

Start with an empty window and type the following code:

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>A simple HTML document</title>
</head>
<body>
    <p>Hello World!<p>
</body>
</html>
```

Step 3: Saving the file

Now save the file on your desktop as "myfirstpage.html ".

Note: It is important that the extension `.html` is specified — some text editors, such as Notepad, will automatically save it as `.txt` otherwise.

To open the file in a browser. Navigate to your file then double click on it. It will open in your default Web browser. If it does not, open your browser and drag the file to it.

Explanation of code

You might think what that code was all about. Well, let's find out.

- The first line `<!DOCTYPE html>` is the document type declaration. It instructs the web browser that this document is an HTML5 document. It is case-insensitive.
- The `<head>` element is a container for the tags that provides information about the document, for example, `<title>` tag defines the title of the document.
- The `<body>` element contains the document's actual content (paragraphs, links, images, tables, and so on) that is rendered in the web browser and displayed to the user.

You will learn about the different HTML elements in detail in the upcoming exercises. For now, just focus on the basic structure of the HTML document.

Note: A DOCTYPE declaration appears at the top of a web page before all other elements; however the doctype declaration itself is not an HTML tag. Every HTML document requires a document type declaration to insure that your pages are displayed correctly.

Tip: The `<html>`, `<head>`, and `<body>` tags make up the basic skeleton of every web page. Content inside the `<head>` and `</head>` are invisible to users with one exception: the text between `<title>` and `</title>` tags which appears as the title on a browser tab.

HTML Tags and Elements

HTML is written in the form of HTML elements consisting of markup tags. These markup tags are the fundamental characteristic of HTML. Every markup tag is composed of a keyword, surrounded by angle brackets, such as `<html>`, `<head>`, `<body>`, `<title>`, `<p>`, and so on.

HTML tags normally come in pairs like `<html>` and `</html>`. The first tag in a pair is often called the opening tag (or start tag), and the second tag is called the closing tag (or end tag).

An opening tag and a closing tag are identical, except a slash (`/`) after the opening angle bracket of the closing tag, to tell the browser that the command has been completed.

In between the start and end tags you can place appropriate contents. For example, a paragraph, which is represented by the `p` element, would be written as:

Example

Try this code »

```
<p>This is a paragraph.</p>
<!-- Paragraph with nested element -->
<p>
    This is <b>another</b> paragraph.
```

```
</p>
```

You will learn about the various HTML elements in upcoming exercise.

HTML Elements

In this exercise you will learn about HTML tags and elements.

HTML Element Syntax

An HTML element is an individual component of an HTML document. It represents semantics, or meaning. For example, the `title` element represents the title of the document.

Most HTML elements are written with a *start tag* (or opening tag) and an *end tag* (or closing tag), with content in between. Elements can also contain attributes that defines its additional properties. For example, a paragraph, which is represented by the `p` element, would be written as:

We will learn about the HTML attributes in the next exercise.

Note: All elements don't require the end tag or closing tag to be present. These are referred as *empty elements*, *self-closing elements* or *void elements*.

HTML Tags Vs Elements

Technically, an HTML element is the collection of start tag, its attributes, an end tag and everything in between. On the other hand an HTML tag (either opening or closing) is used to mark the start or end of an element, as you can see in the above illustration.

However, in common usage the terms HTML element and HTML tag are interchangeable i.e. a tag is an element is a tag. For simplicity's sake of this website, the terms "tag" and "element" are used to mean the same thing — as it will define something on your web page.

Case Insensitivity in HTML Tags and Attributes

In HTML, tag and attribute names are not case-sensitive (but most attribute values are case-sensitive). It means the tag `<P>`, and the tag `<p>` defines the same thing in HTML which is a paragraph.

But in XHTML they are case-sensitive and the tag `<P>` is different from the tag `<p>`.

Example

Try this code »

```
<p>This is a paragraph.</p>
<P>This is also a valid paragraph.</P>
```

Tip: We recommend using lowercase for tag and attributing names in HTML, since by doing this you can make your document more compliant for future upgrades.

Empty HTML Elements

Empty elements (also called self-closing or void elements) are not container tags — that means, you can not write `<hr>some content</hr>` or `<br>some content</br>`.

A typical example of an empty element, is the `<br>` element, which represents a line break. Some other common empty elements are `<img>`, `<input>`, `<link>`, `<meta>`, `<hr>`, etc.

Example

Try this code »

```
<p>This paragraph contains <br> a line break.</p>
<img src="images/sky.jpg" alt="Cloudy Sky">
<input type="text" name="username">
```

Note: In HTML, a self-closing element is written simply as `<br>`. In XHTML, a self-closing element requires a space and a trailing slash, such as `<br />`.

Nesting HTML Elements

Most HTML elements can contain any number of further elements (except empty elements), which are, in turn, made up of tags, attributes, and content or other elements.

The following example shows some elements nested inside the `<p>` element.

Example

Try this code »

```
<p>Here is some <b>bold</b> text.</p>
<p>Here is some <em>emphasized</em> text.</p>
<p>Here is some <mark>highlighted</mark> text.</p>
```

Tip: Placing one element inside another is called nesting. A nested element, also called a child element, can be a parent element too if other elements are nested within it.

HTML tags should be nested in correct order. They must be closed in the inverse order of how they are defined, that means the last tag opened must be closed first.

Example

Try this code »

```
<p><strong>These tags are nested properly.</strong></p>
<p><strong>These tags are not nested properly.</p></strong>
```

Writing Comments in HTML

Comments are usually added with the purpose of making the source code easier to understand. It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the HTML. Comments are not displayed in the browser.

An HTML comment begins with `<!--`, and ends with `-->`, as shown in the example below:

Example

Try this code »

```
<!-- This is an HTML comment -->
<!-- This is a multi-line HTML comment
     that spans across more than one line -->
<p>This is a normal piece of text.</p>
```

You can also comment out part of your HTML code for debugging purpose, as shown here:

Example

Try this code »

```
<!-- Hiding this image for testing
<img src="images/smiley.png" alt="Smiley">
-->
```

HTML Elements Types

Elements can be placed in two distinct groups: *block level* and *inline level* elements. The former make up the document's structure, while the latter dress up the contents of a block.

Also, a block element occupies 100% of the available width and it is rendered with a line break before and after. Whereas, an inline element will take up only as much space as it needs.

The most commonly used block-level elements are `<div>`, `<p>`, `<h1>` through `<h6>`, `<form>`, `<ol>`, `<ul>`, `<li>`, and so on. Whereas, the commonly used inline-level elements are `<img>`, `<a>`, `<span>`, `<strong>`, `<b>`, `<em>`, `<i>`, `<code>`, `<input>`, `<button>`, etc.

You will learn about these elements in detail in upcoming exercises.

Note: The block-level elements should not be placed within inline-level elements. For example, the `<p>` element should not be placed inside the `<b>` element.

HTML Attributes

In this exercise you will learn how to use attributes to give more meaning to HTML tags.

What are Attributes

Attributes define additional characteristics or properties of the element such as width and height of an image. Attributes are always specified in the start tag (or opening tag) and usually consists of name/value pairs like `name="value"`. Attribute values should always be enclosed in quotation marks.

Also, some attributes are required for certain elements. For instance, an `<img>` tag must contain a `src` and `alt` attributes. Let's take a look at some examples of the attributes usages:

Example

Try this code »

```
<img src="images/smiley.png" width="30" height="30" alt="Smiley">
<a href="https://www.google.com/" title="Search Engine">Google</a>
<abbr title="Hyper Text Markup Language">HTML</abbr>
<input type="text" value="John Doe">
```

In the above example `src` inside the `<img>` tag is an attribute and image path provided is its value. Similarly `href` inside the `<a>` tag is an attribute and the link provided is its value, and so on.

Tip: Both single and double quotes can be used to quote attribute values. However, double quotes are most common. In situations where the attribute value itself contains double quotes it is necessary to wrap the value in single quotes, e.g., `value='John "Williams" Jr.'`

There are several attributes in HTML5 that do not consist of name/value pairs but consists of just name. Such attributes are called Boolean attributes. Examples of some commonly used Boolean attributes are `checked`, `disabled`, `readonly`, `required`, etc.

Example

Try this code »

```
<input type="email" required>
<input type="submit" value="Submit" disabled>
<input type="checkbox" checked>
<input type="text" value="Read only text" readonly>
```

You will learn about all these elements in detail in upcoming exercises.

Note: Attribute values are generally case-insensitive, except certain attribute values, like the `id` and `class` attributes. However, World Wide Web Consortium (W3C) recommends lowercase for attributes values in their specification.

General Purpose Attributes

There are some attributes, such as `id`, `title`, `class`, `style`, etc. that you can use on the majority of HTML elements. The following section describes their usages.

The id Attribute

The `id` attribute is used to give a unique name or identifier to an element within a document. This makes it easier to select the element using CSS or JavaScript.

Example

Try this code »

```
<input type="text" id="firstName">
<div id="container">Some content</div>
<p id="infoText">This is a paragraph.</p>
```

Note: The `id` of an element must be unique within a single document. No two elements in the same document can be named with the same `id`, and each element can have only one `id`.

The class Attribute

Like `id` attribute, the `class` attribute is also used to identify elements. But unlike `id`, the `class` attribute does not have to be unique in the document. This means you can apply the same class to multiple elements in a document, as shown in the following example:

Example

Try this code »

```
<input type="text" class="highlight">
<div class="box highlight">Some content</div>
<p class="highlight">This is a paragraph.</p>
```

Tip: Since a `class` can be applied to multiple elements, therefore any style rules that are written to that `class` will be applied to all the elements having that `class`.

The title Attribute

The `title` attribute to is used to provide advisory text about an element or its content. Try out the following example to understand how this actually works.

Example

Try this code »

```
<abbr title="World Wide Web Consortium">W3C</abbr>
<a href="images/kites.jpg" title="Click to view a larger image">
    <img src="images/kites-thumb.jpg" alt="kites">
</a>
```

Tip: The value of the `title` attribute (i.e. title text) is displayed as a tooltip by the web browsers when the user place mouse cursor over the element.

The style Attribute

The `style` attribute allows you to specify CSS styling rules such as color, font, border, etc. directly within the element. Let's check out an example to see how it works:

Example

Try this code »

```
<p style="color: blue;">This is a paragraph.</p>
<img src="images/sky.jpg" style="width: 300px;" alt="Cloudy Sky">
<div style="border: 1px solid red;">Some content</div>
```

You will learn more about styling HTML elements in HTML styles exercise.

The attributes we've discussed above are also called global attributes. For more global attributes please check out the HTML5 global attributes reference.

A complete list of attributes for each HTML element is listed inside HTML5 tag reference.

HTML Headings

In this exercise you will learn how to create headings in HTML.

Organizing Content with Headings

Headings help in defining the hierarchy and the structure of the web page content.

HTML offers six levels of heading tags, `<h1>` through `<h6>`; the higher the heading level number, the greater its importance — therefore `<h1>` tag defines the most important heading, whereas the `<h6>` tag defines the least important heading in the document.

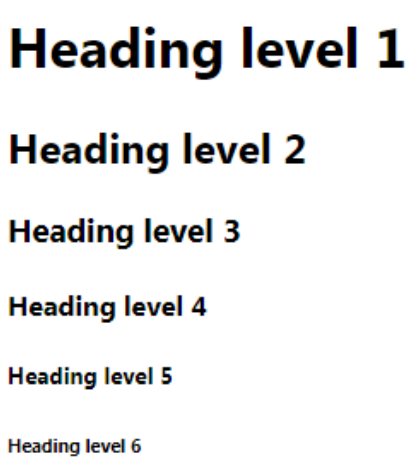
By default, browsers display headings in larger and bolder font than normal text. Also, `<h1>` headings are displayed in largest font, whereas `<h6>` headings are displayed in smallest font.

Example

Try this code »

```
<h1>Heading level 1</h1>
<h2>Heading level 2</h2>
<h3>Heading level 3</h3>
<h4>Heading level 4</h4>
<h5>Heading level 5</h5>
<h6>Heading level 6</h6>
```

— The output of the above example will look something like this:



Note: Each time you place a heading tag on a web page, the web browser built-in style sheets automatically create some empty space (called margin) before and after each heading. You can use the CSS `margin` property to override the browser's default style sheet.

Tip: You can easily customize the appearance of HTML heading tags such as their font size, boldness, typeface, etc. using the CSS font properties.

Importance of Headings

- HTML headings provide valuable information by highlighting important topics and the structure of the document, so optimize them carefully to improve user engagement.
- Don't use headings to make your text look BIG or bold. Use them only for highlighting the heading of your document and to show the document structure.
- Since search engines, such as Google, use headings to index the structure and content of the web pages so use them very wisely in your webpage.
- Use the `<h1>` headings as main headings of your web page, followed by the `<h2>` headings, then the less important `<h3>` headings, and so on.

Tip: Use the `<h1>` tag to mark the most important heading which is usually at the top of the page. An HTML document generally should have exactly one `<h1>` heading, followed by the lower-level headings such as `<h2>`, `<h3>`, `<h4>`, and so on.

HTML Paragraphs

In this exercise you will learn how to create paragraphs in HTML.

Creating Paragraphs

Paragraph element is used to publish text on the web pages.

Paragraphs are defined with the `<p>` tag. Paragraph tag is a very basic and typically the first tag you will need to publish your text on the web pages. Here's an example:

Example

Try this code »

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Note: Browsers built-in style sheets automatically create some space above and below the content of a paragraph (called margin), but you can override it using CSS.

Closing a Paragraph Element

In HTML 4 and earlier versions, it was enough to initiate a new paragraph using the opening tag. Most browsers will display HTML correctly even if you forget the end tag. For example:

Example

Try this code »

```
<p>This is a paragraph.
<p>This is another paragraph.
```

The HTML code in the example above will work in most of the web browsers, but don't rely on it. Forgetting the end tag can produce unexpected results or errors.

Note: For the purposes of forwards-compatibility and good coding practice, it's advisable to use both the opening and closing tags for the paragraphs.

Creating Line Breaks

The `<br>` tag is used to insert a line break on the web page.

Since the `<br>` is an empty element, so there is no need of corresponding `</br>` tag.

Example

Try this code »

```
<p>This is a paragraph <br> with line break.</p>
<p>This is <br>another paragraph <br> with line breaks.</p>
```

Note: Don't use the empty paragraph i.e. `<p></p>` to add extra space in your web pages. The browser may ignore the empty paragraphs since it is logical tag. Use the CSS `margin` property instead to adjust the space around the elements.

Creating Horizontal Rules

You can use the `<hr>` tag to create horizontal rules or lines to visually separate content sections on a web page. Like `<br>`, the `<hr>` tag is also an empty element. Here's an example:

Example

Try this code »

```
<p>This is a paragraph.</p>
<hr>
<p>This is another paragraph.</p>
```

Managing White Spaces

Normally the browser will display the multiple spaces created inside the HTML code by pressing the *space-bar key* or *tab key* on the keyboard as a single space. Multiple line breaks created inside the HTML code through pressing the enter key is also displayed as a single space.

The following paragraphs will be displayed in a single line without any extra space:

Example

Try this code »

```
<p>This paragraph contains  multiple   spaces    in the source code.</p>
<p>
    This paragraph
    contains multiple tabs and line breaks
    in the source code.
</p>
```

Insert ` ` for creating extra consecutive spaces, while insert `<br>` tag for creating line breaks on your web pages, as demonstrated in the following example:

Example

Try this code »

```
<p>This paragraph has multiple   spaces.</p>
<p>This paragraph has multiple<br><br>line<br><br><br>breaks.</p>
```

Defining Preformatted Text

Sometimes, using ` `, `<br>`, etc. for managing spaces isn't very convenient. Alternatively, you can use the `<pre>` tag to display spaces, tabs, line breaks, etc. exactly as written in the HTML file. It is very helpful in presenting text where spaces and line breaks are important like poem or code.

The following example will display the text in the browser as it is in the source code:

Example

Try this code »

```
<pre>
    Twinkle, twinkle, little star,
    How I wonder what you are!
    Up above the world so high,
    Like a diamond in the sky.
</pre>
```

Tip: Text within the `<pre>` element is typically rendered by the browsers in a monospace or fixed-width font, such as Courier, but you can override this using the CSS `font` property.

HTML Links

In this exercise you will learn how to create links to other pages in HTML.

Creating Links in HTML

A link or hyperlink is a connection from one web resource to another. Links allow users to move seamlessly from one page to another, on any server anywhere in the world.

A link has two ends, called anchors. The link starts at the source anchor and points to the destination anchor, which may be any web resource, for example, an image, an audio or video clip, a PDF file, an HTML document or an element within the document itself, and so on.

By default, links will appear as follow in most of the browsers:

- An unvisited link is underlined and blue.
- A visited link is underlined and purple.
- An active link is underlined and red.

However, you can overwrite this using CSS.

HTML Link Syntax

Links are specified in HTML using the `<a>` tag.

A link or hyperlink could be a word, group of words, or image.

Link text

Anything between the opening `<a>` tag and the closing `</a>` tag becomes the part of the link that the user sees and clicks in a browser. Here are some examples of the links:

Example

Try this code »

```
<a href="https://www.google.com/">Google Search</a>
<a href="https://www.exerciserepublic.com/">Exercise Republic</a>
<a href="images/kites.jpg">
    <img src="kites-thumb.jpg" alt="kites">
</a>
```

The `href` attribute specifies the target of the link. Its value can be an absolute or relative URL.

An absolute URL is the URL that includes every part of the URL format, such as protocol, host name, and path of the document, e.g., `https://www.google.com/`,

`https://www.example.com/form.php`, etc. While, relative URLs are page-relative paths, e.g., `contact.html`, `images/smiley.png`, and so on. A relative URL never includes the `http://` or `https://` prefix.

Setting the Targets for Links

The `target` attribute tells the browser where to open the linked document. There are four defined targets, and each target name starts with an underscore(_) character:

- `_blank` — Opens the linked document in a new window or tab.
- `_parent` — Opens the linked document in the parent window.
- `_self` — Opens the linked document in the same window or tab as the source document. This is the default, hence it is not necessary to explicitly specify this value.
- `_top` — Opens the linked document in the full browser window.

Try out the following example to understand how the link's target basically works:

Example

Try this code »

```
<a href="/about-us.php" target="_top">About Us</a>
<a href="https://www.google.com/" target="_blank">Google</a>
<a href="images/sky.jpg" target="_parent">
    <img src="sky-thumb.jpg" alt="Cloudy Sky">
</a>
```

Tip: If your web page is placed inside an iframe, you can use the `target="_top"` on the links to break out of the iframe and show the target page in full browser window.

Creating Bookmark Anchors

You can also create bookmark anchors to allow users to jump to a specific section of a web page. Bookmarks are especially helpful if you have a very long web page.

Creating bookmarks is a two-step process: first add the `id` attribute on the element where you want to jump, then use that `id` attribute value preceded by the hash sign (`#`) as the value of the `href` attribute of the `<a>` tag, as shown in the following example:

Example

Try this code »

```
<a href="#sectionA">Jump to Section A</a>
<h2 id="sectionA">Section A</h2>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
```

Tip: You can even jump to a section of another web page by specifying the URL of that page along with the anchor (i.e. `#elementId`) in the `href` attribute, for example, `<a href="mypage.html#topicA">Go to TopicA</a>`.

Creating Download Links

You can also create the file download link in exactly the same fashion as placing text links. Just point the destination URL to the file you want to be available for download.

In the following example we've created the download links for ZIP, PDF and JPG files.

Example

Try this code »

```
<a href="downloads/test.zip">Download Zip file</a>
<a href="downloads/masters.pdf">Download PDF file</a>
<a href="downloads/sample.jpg">Download Image file</a>
```

Note: When you click a link that points to a PDF or image file, the file is not downloaded to your hard drive directly. It will only open the file in your web browser. Further you can save or download it to your hard drive on a permanent basis.

HTML Text Formatting

In this exercise you will learn how to format the text on the web pages using HTML tags.

Formatting Text with HTML

HTML provides several tags that you can use to make some text on your web pages to appear differently than normal text, for example, you can use the tag `<b>` to make the text bold, tag `<i>` to make the text italic, tag `<mark>` to highlight the text, tag `<code>` to display a fragment of computer code, tags `<ins>` and `<del>` for marking editorial insertions and deletions, and more.

The following example demonstrates the most commonly used formatting tags in action. Now, let's try this out to understand how these tags basically work:

Example

Try this code »

```
<p>This is <b>bold text</b>.</p>
<p>This is <strong>strongly important text</strong>.</p>
<p>This is <i>italic text</i>.</p>
```

```
<p>This is <em>emphasized text</em>.</p>
<p>This is <mark>highlighted text</mark>.</p>
<p>This is <code>computer code</code>.</p>
<p>This is <small>smaller text</small>.</p>
<p>This is <sub>subscript</sub> and <sup>superscript</sup> text.</p>
<p>This is <del>deleted text</del>.</p>
<p>This is <ins>inserted text</ins>.</p>
```

By default, the `<strong>` tag is typically rendered in the browser as `<b>`, whereas the `<em>` tag is rendered as `<i>`. However, there is a difference in the meaning of these tags.

Difference between and <b> tag

Both `<strong>` and `<b>` tags render the enclosed text in a bold typeface by default, but the `<strong>` tag indicates that its contents have strong importance, whereas the `<b>` tag is simply used to draw the reader's attention without conveying any special importance.

Example

Try this code »

```
<p><strong>WARNING!</strong> Please proceed with caution.</p>
<p>The concert will be held at <b>Hyde Park</b> in London.</p>
```

Difference between and <i> tag

Similarly, both `<em>` and `<i>` tags render the enclosed text in italic type by default, but the `<em>` tag indicates that its contents have stressed emphasis compared to surrounding text, whereas the `<i>` tag is used for marking up text that is set off from the normal text for readability reasons, such as a technical term, an idiomatic phrase from another language, a thought, etc.

Example

Try this code »

```
<p>Cats are <em>cute</em> animals.</p>
<p>The <i>Royal Cruise</i> sailed last night.</p>
```

Note: Use the `<em>` and `<strong>` tags when the content of your page requires that certain words or phrases should have strong emphasis or importance. Also, in HTML5 the `<b>` and `<i>` tags have been redefined, earlier they don't have semantic meaning.

Formatting Quotations

You can easily format the quotation blocks from other sources with the HTML `<blockquote>` tag.

Blockquotes are generally displayed with indented left and right margins, along with a little extra space added above and below. Let's try an example to see how it works:

Example

Try this code »

```
<blockquote>
    <p>Learn from yesterday, live for today, hope for tomorrow. The important
thing is not to stop questioning.</p>
    <cite>— Albert Einstein</cite>
</blockquote>
```

Tip: The `cite` tag is used to describe a reference to a creative work. It must include the title of that work or the name of the author (people or organization) or an URL reference.

For short inline quotations, you can use the HTML `<q>` tag. Most browsers display inline quotes by surrounding the text in quotation marks. Here's an example:

Example

Try this code »

```
<p>According to the World Health Organization (WHO): <q>Health is a state of
complete physical, mental, and social well-being.</q></p>
```

Showing Abbreviations

An abbreviation is a shortened form of a word, phrase, or name.

You can use the `<abbr>` tag to denote an abbreviation. The `title` attribute is used inside this tag to provide the full expansion of the abbreviation, which is displayed by the browsers as a tooltip when the mouse cursor is hovered over the element. Let's try out an example:

Example

Try this code »

```
<p>The <abbr title="World Wide Web Consortium">W3C</abbr> is the main
international standards organization for the <abbr title="World Wide Web">WWW
or W3</abbr>. It was was founded by Tim Berners-Lee.</p>
```

Marking Contact Addresses

Web pages often include street or postal addresses. HTML provides a special tag `<address>` to represent contact information (physical and/or digital) for a person, people or organization.

This tag should ideally used to display contact information related to the document itself, such as article's author. Most browsers display an address block in italic. Here's an example:

Example

Try this code »

```
<address>
Mozilla Foundation<br>
331 E. Evelyn Avenue<br>
Mountain View, CA 94041, USA
</address>
```

Please check out the HTML reference section for a complete list of HTML formatting tags.

HTML Styles

In this exercise you will learn how to apply style rules to HTML elements.

Styling HTML Elements

HTML is quite limited when it comes to the presentation of a web page. It was originally designed as a simple way of presenting information. CSS (Cascading Style Sheets) was introduced in December 1996 by the World Wide Web Consortium (W3C) to provide a better way to style HTML elements.

With CSS, it becomes very easy to specify the things like, size and typeface for the fonts, colors for the text and backgrounds, alignment of the text and images, amount of space between the elements, border and outlines for the elements, and lots of other styling properties.

Adding Styles to HTML Elements

Style information can either be attached as a separate document or embedded in the HTML document itself. These are the three methods of implementing styling information to an HTML document.

- **Inline styles** — Using the `style` attribute in the HTML start tag.
- **Embedded style** — Using the `<style>` element in the head section of the document.
- **External style sheet** — Using the `<link>` element, pointing to an external CSS files.

In this exercise we will cover all these different types of style sheet one by one.

Note: The inline styles have the highest priority, and the external style sheets have the lowest. It means if you specify styles for your paragraphs in both *embedded* and *external* style sheets, the conflicting style rules in the embedded style sheet would override the external style sheet.

Inline Styles

Inline styles are used to apply the unique style rules to an element, by putting the CSS rules directly into the start tag. It can be attached to an element using the `style` attribute.

The style attribute includes a series of CSS property and value pairs. Each `property: value` pair is separated by a semicolon (`;`), just as you would write into an embedded or external style sheet. But it needs to be all in one line i.e. no line break after the semicolon.

The following example demonstrates how to set the `color` and `font-size` of the text:

Example

Try this code »

```
<h1 style="color:red; font-size:30px;">This is a heading</h1>
<p style="color:green; font-size:18px;">This is a paragraph.</p>
<div style="color:green; font-size:18px;">This is some text.</div>
```

Using the inline styles are generally considered as a bad practice. Because style rules are embedded directly inside the html tag, it causes the presentation to become mixed with the content of the document, which makes updating or maintaining a website very difficult.

To learn about the various CSS properties in detail, please check out CSS exercise section.

Note: It's become impossible to style pseudo-elements and pseudo-classes with inline styles. You should, therefore, avoid the use of `style` attributes in your markup. Using external style sheet is the preferred way to add style information to an HTML document.

Embedded Style Sheets

Embedded or internal style sheets only affect the document they are embedded in.

Embedded style sheets are defined in the `<head>` section of an HTML document using the `<style>` tag. You can define any number of `<style>` elements inside the `<head>` section.

The following example demonstrates how style rules are embedded inside a web page.

Example

Try this code »

```
<head>
    <style>
        body { background-color: YellowGreen; }
               h1 { color: blue; }
        p { color: red; }
    </style>
</head>
```

External Style Sheets

An external style sheet is ideal when the style is applied to many pages.

An external style sheet holds all the style rules in a separate document that you can link from any HTML document on your site. External style sheets are the most flexible because with an external style sheet, you can change the look of an entire website by updating just one file.

You can attach external style sheets in two ways — *linking* and *importing*:

Linking External Style Sheets

An external style sheet can be linked to an HTML document using the `<link>` tag.

The `<link>` tag goes inside the `<head>` section, as shown here:

Example

Try this code »

```
<head>
    <link rel="stylesheet" href="css/style.css">
</head>
```

Importing External Style Sheets

The `@import` rule is another way of loading an external style sheet. The `@import` statement instructs the browser to load an external style sheet and use its styles.

You can use it in two ways. The simplest way is to use it within the `<style>` element in your `<head>` section. Note that, other CSS rules may still be included in the `<style>` element.

Example

Try this code »

```
<style>
    @import url("css/style.css");
    p {
        color: blue;
        font-size: 16px;
    }
</style>
```

Similarly, you can use the `@import` rule to import a style sheet within another style sheet.

Example

Try this code »

```
@import url("css/layout.css");
@import url("css/color.css");
body {
    color: blue;
```

```
    font-size: 14px;
}
```

Note: All `@import` rules must occur at the start of the style sheet. Any style rule defined in the style sheet itself override conflicting rule in the imported style sheets. The `@import` rule may cause performance issues, you should generally avoid importing style sheets.

HTML Images

In this exercise you will learn how to include images in an HTML document.

Inserting Images into Web Pages

Images enhance visual appearance of the web pages by making them more interesting and colorful.

The `<img>` tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The syntax of the `<img>` tag can be given with:

<img src="*url*" alt="*some_text*">

The following example inserts three images on the web page:

Example

Try this code »

```
<img src="kites.jpg" alt="Flying Kites">
<img src="sky.jpg" alt="Cloudy Sky">
<img src="balloons.jpg" alt="Balloons">
```

Each image must carry at least two attributes: the `src` attribute, and an `alt` attribute.

The `src` attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the `alt` attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image.

Note: Like `<br>` , the `<img>` element is also an empty element, and does not have a closing tag. However, in XHTML it closes itself ending with "`/>`".

Tip: The required `alt` attribute provides alternative text description for an image if a user for some reason cannot able to view it because of slow connection, image is not available at the specified URL, or if the user uses a screen reader or non-graphical browser.

Setting the Width and Height of an Image

The `width` and `height` attributes are used to specify the width and height of an image.

The values of these attributes are interpreted in pixels by default.

Example

Try this code »

```
<img src="kites.jpg" alt="Flying Kites" width="300" height="300">
<img src="sky.jpg" alt="Cloudy Sky" width="250" height="150">
<img src="balloons.jpg" alt="Balloons" width="200" height="200">
```

You can also use the `style` attribute to specify width and height for the images. It prevents style sheets from changing the image size accidently, since inline style has the highest priority.

Example

Try this code »

```
<img src="kites.jpg" alt="Flying Kites" style="width: 300px; height: 300px;">
<img src="sky.jpg" alt="Cloudy Sky" style="width: 250px; height: 150px;">
<img src="balloons.jpg" alt="Balloons" style="width: 200px; height: 200px;">
```

Note: It's a good practice to specify both the `width` and `height` attributes for an image, so that browser can allocate that much of space for the image before it is downloaded. Otherwise, image loading may cause distortion or flicker in your website layout.

Using the HTML5 Picture Element

Sometimes, scaling an image up or down to fit different devices (or screen sizes) doesn't work as expected. Also, reducing the image dimension using the `width` and `height` attribute or property doesn't reduce the original file size. To address these problems HTML5 has introduced the `<picture>` tag that allows you to define multiple versions of an image to target different types of devices.

The `<picture>` element contains zero or more `<source>` elements, each referring to different image source, and one `<img>` element at the end. Also each `<source>` element has the `media` attribute which specifies a media condition (similar to the media query) that is used by the browser to determine when a particular source should be used. Let's try out an example:

Example

Try this code »

```
<picture>
    <source media="(min-width: 1000px)" srcset="logo-large.png">
    <source media="(max-width: 500px)" srcset="logo-small.png">
```

```
    <img src="logo-default.png" alt="My logo">
</picture>
```

Note: The browser will evaluate each child `<source>` element and choose the best match among them; if no matches are found, the URL of the `<img>` element's `src` attribute is used. Also, the `<img>` tag must be specified as the last child of the `<picture>` element.

Working with Image Maps

An image map allows you to define hotspots on an image that acts just like a hyperlink.

The basic idea behind creating image map is to provide an easy way of linking various parts of an image without dividing it into separate image files. For example, a map of the world may have each country hyperlinked to further information about that country.

Let's try out a simple example to understand how it actually works:

Example

Try this code »

```
<img src="objects.png" usemap="#objects" alt="Objects">
<map name="objects">
    <area shape="circle" coords="137,231,71" href="clock.html" alt="Clock">
    <area shape="poly" coords="363,146,273,302,452,300" href="sign.html"
alt="Sign">
    <area shape="rect" coords="520,160,641,302" href="book.html" alt="Book">
</map>
```

The `name` attribute of the `<map>` tag is used to reference the map from the `<img>` tag using its `usemap` attribute. The `<area>` tag is used inside the `<map>` element to define the clickable areas on an image. You can define any number of clickable areas within an image.

Note: The image map should not be used for website navigation. They are not search engine friendly. A valid use of an image map is with a geographical map.

Tip: There are many online tools available for creating image maps. Some advanced editors like Adobe Dreamweaver also provides a set of tools for easily creating image maps.

HTML Tables

In this exercise you will learn how to display tabular data using HTML tables.

Creating Tables in HTML

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on.

You can create a table using the `<table>` element. Inside the `<table>` element, you can use the `<tr>` elements to create rows, and to create columns inside a row you can use the `<td>` elements. You can also define a cell as a header for a group of table cells using the `<th>` element.

The following example demonstrates the most basic structure of a table.

Example

Try this code »

```
<table>
    <tr>
        <th>No.</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Peter Parker</td>
        <td>16</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Clark Kent</td>
        <td>34</td>
    </tr>
</table>
```

Tables do not have any borders by default. You can use the CSS `border` property to add borders to the tables. Also, table cells are sized just large enough to fit the contents by default. To add more space around the content in the table cells you can use the CSS `padding` property.

The following style rules add a 1-pixel border to the table and 10-pixels of padding to its cells.

Example

Try this code »

```
table, th, td {
    border: 1px solid black;
}
th, td {
    padding: 10px;
}
```

By default, borders around the table and their cells are separated from each other. But you can collapse them into one by using the `border-collapse` property on the `<table>` element.

Also, text inside the `<th>` elements are displayed in bold font, aligned horizontally center in the cell by default. To change the default alignment you can use the CSS `text-align` property.

The following style rules collapse the table borders and align the table header text to left.

Example

Try this code »

```
table {
    border-collapse: collapse;
}
th {
    text-align: left;
}
```

Please check out the exercise on CSS tables to learn about styling HTML tables in details.

Note: Most of the `<table>` element's attribute such as `border`, `cellpadding`, `cellspacing`, `width`, `align`, etc. for styling table appearances in earlier versions has been dropped in HTML5, so avoid using them. Use CSS to style HTML tables instead.

Spanning Multiple Rows and Columns

Spanning allow you to extend table rows and columns across multiple other rows and columns.

Normally, a table cell cannot pass over into the space below or above another table cell. But, you can use the `rowspan` or `colspan` attributes to span multiple rows or columns in a table.

Let's try out the following example to understand how `colspan` basically works:

Example

Try this code »

```
<table>
    <tr>
        <th>Name</th>
        <th colspan="2">Phone</th>
    </tr>
    <tr>
        <td>John Carter</td>
        <td>5550192</td>
        <td>5550152</td>
    </tr>
</table>
```

Similarly, you can use the `rowspan` attribute to create a cell that spans more than one row. Let's try out an example to understand how row spanning basically works:

Example

Try this code »

```
<table>
    <tr>
        <th>Name:</th>
        <td>John Carter</td>
    </tr>
    <tr>
        <th rowspan="2">Phone:</th>
        <td>55577854</td>
    </tr>
    <tr>
        <td>55577855</td>
    </tr>
</table>
```

Adding Captions to Tables

You can specify a caption (or title) for your tables using the `<caption>` element.

The `<caption>` element must be placed directly after the opening `<table>` tag. By default, caption appears at the top of the table, but you can change its position using the CSS `caption-side` property.

The following example shows how to use this element in a table.

Example

Try this code »

```
<table>
    <caption>Users Info</caption>
    <tr>
        <th>No.</th>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Peter Parker</td>
        <td>16</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Clark Kent</td>
        <td>34</td>
    </tr>
</table>
```

Defining a Table Header, Body, and Footer

HTML provides a series of tags <thead>, <tbody>, and <tfoot> that helps you to create more structured table, by defining header, body and footer regions, respectively.

The following example demonstrates the use of these elements.

Example

Try this code »

```
<table>
    <thead>
        <tr>
            <th>Items</th>
            <th>Expenditure</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Stationary</td>
            <td>2,000</td>
        </tr>
        <tr>
            <td>Furniture</td>
            <td>10,000</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <th>Total</th>
            <td>12,000</td>
        </tr>
    </tfoot>
</table>
```

Note: In HTML5, the `<tfoot>` element can be placed either before or after the `<tbody>` and `<tr>` elements, but must appear after any `<caption>`, `<colgroup>`, and `<thead>` elements.

Tip: Do not use tables for creating web page layouts. Table layouts are slower at rendering, and very difficult to maintain. It should be used only to display tabular data.

HTML Lists

In this exercise you will learn how to create different types of lists in HTML.

Working with HTML Lists

HTML lists are used to present list of information in well formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning.

- **Unordered list** — Used to create a list of related items, in no particular order.

- **Ordered list** — Used to create a list of related items, in a specific order.
- **Description list** — Used to create a list of terms and their descriptions.

Note: Inside a list item you can put text, images, links, line breaks, etc. You can also place an entire list inside a list item to create the nested list.

In the following sections we will cover all the three types of list one by one:

HTML Unordered Lists

An unordered list created using the `<ul>` element, and each list item starts with the `<li>` element.

The list items in unordered lists are marked with bullets. Here's an example:

Example

Try this code »

```
<ul>
    <li>Chocolate Cake</li>
    <li>Black Forest Cake</li>
    <li>Pineapple Cake</li>
</ul>
```

— The output of the above example will look something like this:

- Chocolate Cake
- Black Forest Cake
- Pineapple Cake

You can also change the bullet type in your unordered list using the CSS `list-style-type` property. The following style rule changes the type of bullet from the default *disc* to *square*:

Example

Try this code »

```
ul {
    list-style-type: square;
}
```

Please check out the exercise on CSS lists to learn about styling HTML lists in details.

HTML Ordered Lists

An ordered list created using the `<ol>` element, and each list item starts with the `<li>` element. Ordered lists are used when the order of the list's items is important.

The list items in an ordered list are marked with numbers. Here's an example:

Example

Try this code »

```
<ol>
    <li>Fasten your seatbelt</li>
    <li>Starts the car's engine</li>
    <li>Look around and go</li>
</ol>
```

— The output of the above example will look something like this:

1. Fasten your seatbelt
2. Starts the car's engine
3. Look around and go

The numbering of items in an ordered list typically starts with 1. However, if you want to change that you can use the `start` attribute, as shown in the following example:

Example

Try this code »

```
<ol start="10">
    <li>Mix ingredients</li>
    <li>Bake in oven for an hour</li>
    <li>Allow to stand for ten minutes</li>
</ol>
```

— The output of the above example will look something like this:

10. Mix ingredients
11. Bake in oven for an hour
12. Allow to stand for ten minutes

Like unordered list, you can also use the CSS `list-style-type` property to change the numbering type in an ordered list. The following style rule changes the marker type to roman numbers.

Example

Try this code »

```
ol {
    list-style-type: upper-roman;
}
```

Tip: You can also use the `type` attribute to change the numbering type e.g. `type="I"`. However, you should avoid this attribute, use the CSS `list-style-type` property instead.

HTML Description Lists

A description list is a list of items with a description or definition of each item.

The description list is created using `<dl>` element. The `<dl>` element is used in conjunction with the `<dt>` element which specify a term, and the `<dd>` element which specify the term's definition.

Browsers usually render the definition lists by placing the terms and definitions in separate lines, where the term's definitions are slightly indented. Here's an example:

Example

Try this code »

```
<dl>
    <dt>Bread</dt>
    <dd>A baked food made of flour.</dd>
    <dt>Coffee</dt>
    <dd>A drink made from roasted coffee beans.</dd>
</dl>
```

— The output of the above example will look something like this:

Bread

: A baked food made of flour.

Coffee

: A drink made from roasted coffee beans.

HTML Forms

In this exercise you will learn how to create a form in HTML to collect user inputs.

What is HTML Form

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like inputbox, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The `<form>` tag is used to create an HTML form. Here's a simple example of a login form:

Example

Try this code »

```
<form>
    <label>Username: <input type="text"></label>
    <label>Password: <input type="password"></label>
    <input type="submit" value="Submit">
</form>
```

The following section describes different types of controls that you can use in your form.

Input Element

This is the most commonly used element within HTML forms.

It allows you to specify various types of user input fields, depending on the `type` attribute. An input element can be of type *text field*, *password field*, *checkbox*, *radio button*, *submit button*, *reset button*, *file select box*, as well as several new input types introduced in HTML5.

The most frequently used input types are described below.

Text Fields

Text fields are one line areas that allow the user to input text.

Single-line text input controls are created using an `<input>` element, whose `type` attribute has a value of `text`. Here's an example of a single-line text input used to take username:

Example

Try this code »

```
<form>
    <label for="username">Username:</label>
    <input type="text" name="username" id="username">
</form>
```

— The output of the above example will look something like this:

Username:

Note: The `<label>` tag is used to define the labels for `<input>` elements. If you want your user to enter several lines you should use a `<textarea>` instead.

Password Field

Password fields are similar to text fields. The only difference is; characters in a password field are masked, i.e. they are shown as asterisks or dots. This is to prevent someone else from reading

the password on the screen. This is also a single-line text input controls created using an `<input>` element whose `type` attribute has a value of `password`.

Here's an example of a single-line password input used to take user password:

Example

Try this code »

```
<form>
    <label for="user-pwd">Password:</label>
    <input type="password" name="user-password" id="user-pwd">
</form>
```

— The output of the above example will look something like this:

Radio Buttons

Radio buttons are used to let the user select exactly one option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `radio`.

Here's an example of radio buttons that can be used to collect user's gender information:

Example

Try this code »

```
<form>
    <input type="radio" name="gender" id="male">
    <label for="male">Male</label>
    <input type="radio" name="gender" id="female">
    <label for="female">Female</label>
</form>
```

— The output of the above example will look something like this:

Checkboxes

Checkboxes allows the user to select one or more option from a pre-defined set of options. It is created using an `<input>` element whose `type` attribute has a value of `checkbox`.

Here's an example of checkboxes that can be used to collect information about user's hobbies:

Example

Try this code »

```
<form>
    <input type="checkbox" name="sports" id="soccer">
    <label for="soccer">Soccer</label>
    <input type="checkbox" name="sports" id="cricket">
    <label for="cricket">Cricket</label>
    <input type="checkbox" name="sports" id="baseball">
    <label for="baseball">Baseball</label>
</form>
```

— The output of the above example will look something like this:

Note: If you want to make a radio button or checkbox selected by default, you can add the attribute `checked` to the input element, like `<input type="checkbox" checked>`.

File Select box

The file fields allow a user to browse for a local file and send it as an attachment with the form data. Web browsers such as Google Chrome and Firefox render a file select input field with a Browse button that enables the user to navigate the local hard drive and select a file.

This is also created using an `<input>` element, whose `type` attribute value is set to `file`.

Example

Try this code »

```
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

— The output of the above example will look something like this:

Tip: There are several other input types. Please check out the exercise on HTML5 new input types to learn more about the newly introduced input types.

Textarea

Textarea is a multiple-line text input control that allows a user to enter more than one line of text. Multi-line text input controls are created using an `<textarea>` element.

Example

Try this code »

```
<form>
    <label for="address">Address:</label>
    <textarea rows="3" cols="30" name="address" id="address"></textarea>
</form>
```

— The output of the above example will look something like this:

Address:

Select Boxes

A select box is a dropdown list of options that allows user to select one or more option from a pull-down list of options. Select box is created using the `<select>` element and `<option>` element.

The `<option>` elements within the `<select>` element define each list item.

Example

Try this code »

```
<form>
    <label for="city">City:</label>
    <select name="city" id="city">
        <option value="sydney">Sydney</option>
        <option value="melbourne">Melbourne</option>
        <option value="cromwell">Cromwell</option>
    </select>
</form>
```

— The output of the above example will look something like this:

City: Sydney

Submit and Reset Buttons

A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's `action` attribute to process the submitted data.

A reset button resets all the forms control to default values. Try out the following example by typing your name in the text field, and click on submit button to see it in action.

Example

Try this code »

```
<form action="action.php" method="post">
    <label for="first-name">First Name:</label>
    <input type="text" name="first-name" id="first-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
</form>
```

Type your name in the text field above, and click on submit button to see it in action.

Note: You can also create buttons using the `<button>` element. Buttons created with the `<button>` element function just like buttons created with the input element, but they offer richer rendering possibilities by allowing the embedding of other HTML elements.

Grouping Form Controls

You also group logically related controls and labels within a web form using the `<legend>` element. Grouping form controls into categories makes it easier for users to locate a control which makes the form more user-friendly. Let's try out the following example to see how it works:

Example

Try this code »

```
<form>
    <fieldset>
        <legend>Contact Details</legend>
        <label>Email Address: <input type="email" name="email"></label>
        <label>Phone Number: <input type="text" name="phone"></label>
    </fieldset>
</form>
```

Frequently Used Form Attributes

The following table lists the most frequently used form element's attributes:

Attribute	Description
`name`	Specifies the name of the form.
`action`	Specifies the URL of the program or script on the web server that will be used for processing the information submitted via form.
`method`	Specifies the HTTP method used for sending the data to the web server by the browser. The value can be either `get` (the default) and `post`.
`target`	Specifies where to display the response that is received after submitting the form. Possible values are `_blank`, `_self`, `_parent` and `_top`.
`enctype`	Specifies how the form data should be encoded when submitting the form to the server. Applicable only when the value of the `method` attribute is `post`.

There are several other attributes, to know about them please see the `<form>` tag reference.

Note: The `name` attribute represents the form's name within the forms collection. Its value must be unique among the forms in a document, and must not be an empty string.

Tip: All the data sent via `get` method is visible in the browser's address bar. But, the data sent via `post` is not visible to the user. Please check out the exercise on GET vs. POST to learn about the difference between these two HTTP methods in detail.

HTML iFrame

In this exercise you will learn how to use an iframe to display a web page within another web page.

What is iframe

An iframe or inline frame is used to display external objects including other web pages within a web page. An iframe pretty much acts like a mini web browser within a web browser. Also, the content inside an iframe exists entirely independent from the surrounding elements.

The basic syntax for adding an iframe to a web page can be given with:

```
<iframe src="URL"></iframe>
```

The URL specified in the `src` attribute points to the location of an external object or a web page.

The following example display "hello.html" file inside an iframe in current document.

Example

Try this code »

```
<iframe src="hello.html"></iframe>
```

Setting Width and Height of an iFrame

The `height` and `width` attributes are used to specify the height and width of the iframe.

Example

Try this code »

```
<iframe src="hello.html" width="400" height="200"></iframe>
```

You can also use CSS to set the width and height of an iframe, as shown here:

Example

Try this code »

```
<iframe src="hello.html" style="width: 400px; height: 200px;"></iframe>
```

Please see the exercise on HTML styles to learn the methods of applying CSS to HTML elements.

Note: The `width` and `height` attribute values are specified in pixels by default, but you can also set these values in percentage, such as 50%, 100% and so on. The default width of an iframe is 300 pixels, whereas the default height is 150 pixels.

Removing Default Frameborder

The iframe has a border around it by default. However, if you want to modify or remove the iframe borders, the best way is to use the CSS `border` property.

The following example will simply render the iframe without any borders.

Example

Try this code »

```
<iframe src="hello.html" style="border: none;"></iframe>
```

Similarly, you can use the `border` property to add the borders of your choice to an iframe. The following example will render the iframe with 2 pixels blue border.

Example

Try this code »

```
<iframe src="hello.html" style="border: 2px solid blue;"></iframe>
```

Using an iFrame as Link Target

An iframe can also be used as a target for the hyperlinks.

An iframe can be named using the `name` attribute. This implies that when a link with a `target` attribute with that name as value is clicked, the linked resource will open in that iframe.

Let's try out an example to understand how it basically works:

Example

Try this code »

```
<iframe src="demo-page.html" name="myFrame"></iframe>
<p><a href="https://www.exerciserepublic.com" target="myFrame">Open
ExerciseRepublic.com</a></p>
```