

OBJECTIVE: INTRODUCTION TO HTML

Objectives

After completion of this skill, you should be able to:

- Explain what Hyper Text Markup Language (HTML) and HTML5 is
- Explain what is HTML5
- Describe how to use HTML to build web pages
- Describe the new page layout for HTML5
- Explain the elements of HTML and HTML5
- Explain what is a Cascade Style Sheet (CSS)
- Describe how to use CSS to format HTML elements
- Describe how to use CSS in HTML documents to format web pages

Prerequisites

The prerequisites for this skill are:

- Familiarity with working with web pages in web browser
- Being able to navigate web sites

WHAT IS HTML?

Hyper Text Markup Language (HTML) can be thought of as “the language of the internet”. HTML is the standard mark-up language that is used to create web pages. It was originally designed for sharing scientific documents. Adaptations to HTML over the years made it suitable to describe several other types of documents that can be displayed as web pages on the internet.

As published by the *World Wide Web Consortium (W3C)*, which is an international community that develops open standards to ensure the long-term growth of the Web, HTML gives authors the means to complete the following tasks:

- Publish online documents with headings, text, tables, lists, photos, and so on.
- Retrieve online information by using hypertext links at the click of a button.
- Design forms for conducting transactions with remote services, to be able to search for information, making reservations, ordering products, and so on.
- Include spreadsheets, video clips, sound clips, and other applications directly in their documents.

HTML Elements

HTML Elements are the building blocks of an HTML page, and they are represented by tags. Browsers do not display the tags but use them to render the content. The elements of the language are labelled into pieces of content such as “paragraph,” “list,” and “table.

HTML Editor

To write HTML syntax and to create simple web pages, you can start by using a simple text editor, such as *Notepad* on Windows and *TextEdit* on Mac OS. Also, you can use other advanced tools, such as *Notepad++*, *Brackets*, or any text editor you feel comfortable with.

WHAT IS HTML5

In 2007, W3C formed a working group that was chartered to work with the *Web Hypertext Application Technology Working Group (WHATWG)* on the development of the HTML5 specification. The W3C published the specification under the W3C copyright, while a version with a less restrictive license was kept on the WHATWG site.

Since 2007, the W3C and WHATWG groups worked together on the development of the HTML5 standard and which made it the language of choice for mobile and hybrid web applications.

HTML5 is a language that is designed to organize web content. It is intended to make web design and development easier by creating a standardized and intuitive user interface (UI) markup language.

HTML5 features

HTML5 includes the following features:

- Provides the means to categorize web pages into different sections. It provides effective data management, drawing, video, and audio tools.
- Facilitates the development of cross-browser applications for the web and portable devices.
- Allows greater flexibility, which permits the development of exciting and interactive websites.
- Helps to create a more engaging user experience. Pages that are designed by using HTML5 can provide an experience similar to desktop applications.
- Allows for enhanced, multiple-platform development by combining the capability of an *application programming interface (API)* with the ubiquity of the browser.
- By using HTML5, developers can create a modern application experience that smoothly crosses platforms.

OBJECTIVE: GETTING STARTED WITH HTML

Unit objectives

When you finish this unit you should be able to:

- Create HTML pages
- Describe the new semantic element for page layout in HTML5

HTML PAGE LAYOUT

All HTML pages follow a pre-defined structure that is used to differentiate areas of a webpage.

The following example illustrates the basic structure of an HTML page:

```
<! DOCTYPE html>
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>HTML Tags</h1>
    <p>This is an example of a paragraph</p>
    <ul>
      <li> Sample list 1 </li>
      <li> Sample list 2 </li>
      <li> Sample list 3 </li>
    </ul>
  </body>
</html>
```

This example uses the following elements:

- The `<!DOCTYPE>` is a declaration tag that represents the document type. The `<!DOCTYPE>` declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in. Although this declaration is not required, it should be the first line of the HTML code if the developer decides to include it.
- The `<html>` tag is the root element of this tree. It contains all of the other HTML elements, except the `<!doctype>` tag. This example, contains two elements: `<head>` and `<body>`.
- The `<head>` tag contains a `<title>` tag , which contains the text “Sample page” in the example.

The `<head>` element can contain the following tags:

- title (`<title>`),
 - scripts (`<script>`),
 - style (`<style>`),
 - style sheet links (`<link>`),
 - meta information (`<meta>`),
 - browser support information and other initialization functions (`<base>`).
- The `<title>` element defines the title of the document, and it is required in all HTML documents. You can't have more than one `<title>` element.

```
<head>  
  <title>Sample page</title>  
</head>
```

- Finally, the `<body>` tag contains all content that is displayed on the webpage, as shown in the following example:

```
<body>  
  <h1>HTML Tags</h1>  
  <p>This is an example of a paragraph</p>  
  <ul>  
    <li> Sample list 1 </li>  
    <li> Sample list 2 </li>  
    <li> Sample list 3 </li>  
  </ul>  
</body>
```

HTML5 PAGE LAYOUT

The layout of an HTML page using the new tags and semantic elements was introduced in HTML5, and is as follows:

- Header
- Navigation
- Article
- Footer

The Article section is divided into three *Section* areas and one *Aside* area.

Following this layout, as shown in *Figure 1* helps create a well-formed code and elegant page design.

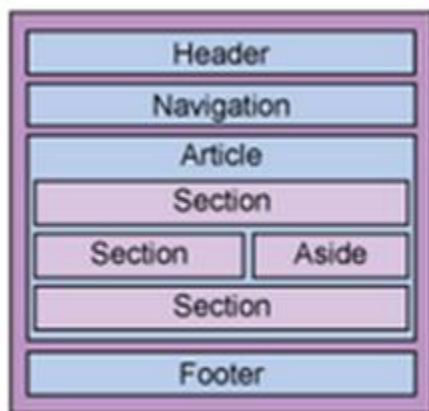


Figure 1. A typical HTML page layout

Header section

Use the `<header>` tag to create content for the Header area. The `<header>` tag can contain introductory information about a `<section>`, `<article>` and the web page, as shown in *Figure 2*:

```
1 <header>
2   <h1>Heading Text</h1>
3   <p>Text or images can be included here</p>
4   <p>Logos are frequently placed here too</p>
5 </header>
```



Figure 2. HTML header section

The `<hgroup>` tag

The `<header>` tag can also contain an `<hgroup>` tag. The `<hgroup>` tag groups the headings by using the `<h1>` - `<h6>` heading levels as shown in *Figure 3*, with Main Heading and Sub Heading

```
1 <header>
2   <hgroup>
3     <h1>Main Heading</h1>
4     <h2>Sub-heading </h2>
5   </hgroup>
6   <p> Text or images can be included here</p>
7 </header>
```

Main heading

Sub-heading

Text of images can be included here

Figure 3. `<hgroup>` tag

Navigation section

You create the Navigation area of the page by using the `<nav>` tag. The `<nav>` element defines an area that is specifically intended for navigation. The `<nav>` tag is used for the main site navigation, not to hold links that are contained in other areas of the page. The Navigation area can contain code, as shown in *Figure 4*:

```
1 <nav>
2   <ul>
3     <li><a href="#">Home</a></li>
4     <li><a href="#">About Us</a></li>
5     <li><a href="#">Our Products</a></li>
6     <li><a href="#">Contact Us</a></li>
7   </ul>
8 </nav>
```

- [Home](#)
- [About Us](#)
- [Our Products](#)
- [Contact Us](#)

Figure 4. Navigation area

Article and Sections

The differences between the use of the `<article>` and the `<section>` tags are as follows:

The `<article>` tag:

As its name implies, the `<article>` tag specifies independent, self-contained content.. The `<article>` tag is used to create an area that defines content that can be used independently of other content on the page, that is, content that can be removed and placed in another context and be completely understandable such as an article, blog, or RSS feed.

The `<section>` tag:

The `<section>` tag contains related information; however, the information cannot be placed in a different context alone because its meaning could be lost.

The `<section>` tag is used for grouping similar content.

The `<section>` tag and the `<article>` tag can contain headers, footers, or any other components that are required to complete the section.

The `<section>` tag can also contain `<article>` tags, and the `<article>` tag can contain the `<section>` tag.

The use of the `<article>` and `<section>` tags is shown in *Figure 5*:

```
1  <article>
2      <section>
3          Content
4      </section>
5      <section>
6          Content
7      </section>
8  </article>
9  <section>
10     <article>
11         Content
12     </article>
13     <article>
14         Content
15     </article>
16 </section>
```

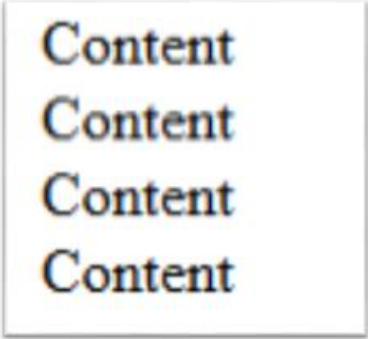


Figure 5. `<article>` and `<section>` tag

The `<aside>` tag

You create the Aside section by using the `<aside>` tag. Think of this section as holding supplementary content that is not part of the flow of the article it supplements. In magazines, asides are frequently used to highlight a point that was made in the main article.

The `<aside>` tag contains content that can be removed without affecting the information that is covered by the article, section, or page that contains it.

The use of the `<aside>` tag is shown in *Figure 6*:

```
1 | <p>My family and I visited Euro Disney last year.</p>
2 | <aside>
3 |     <h4>Disney in France</h4>
4 |     <p>Besides Euro Disney, there is a Disneyland in California.</p>
5 | </aside>
```

My family and I visited Euro Disney last year

Disney in France

Besides Euro Disney, there is a Disneyland in California

Figure 6. Aside section

Footer Section

The `<footer>` element contains information about a page, article, or a section, such as the author or date of the article. A page footer can contain copyright or other important legal information, as shown in *Figure 7*:

```
1 | <footer>
2 |     <p>Copyright 2011 Acme United. All rights reserved.</p>
3 | </footer>
```

Copyright 2011 ACME United. All rights reserved

Figure 7. Footer section

OBJECTIVE: HTML ELEMENTS

Unit objectives

When you finish this unit, you should be able to:

- Describe the use of HTML Elements.
- Provide different examples of HTML Elements.
- Describe new elements introduced in HTML5.
- Explain what HTML attributes are
- Explain how to use HTML attributes in HTML elements

HTML ELEMENTS AND TAGS

HTML elements are the building blocks for HTML pages. Elements are represented by using tags.

HTML tags play an important part in creating a web page. These tags are hidden within a web page and define how the web browser formats and displays the page content. Tags are written inside angle brackets `< >`. For example `<body>` is a start tag or opening tag and `</body>` is an end tag or closing tag.

Basic tags that are used in HTML include the following examples:

Heading tags

Heading tags define the content hierarchy by using `h1` - `h6` tags (as shown in *Figure 1a*). The `h1` tag defines the most important heading on the page; `h6` indicates the lowest-level heading, as shown in *Figure 1b*.

```
<h1>HTML Tags</h1>
<h2>HTML Tags</h2>
<h3>HTML Tags</h3>
<h4>HTML Tags</h4>
<h5>HTML Tags</h5>
<h6>HTML Tags</h6>
```

Browser view

Figure 1a. HTML heading tags



Figure 1b.

Paragraph Element

This tag defines a paragraph using the tag `<p>`, as shown in the following example:

```
<p>This is an example of a paragraph.</p>
```


Unordered list tag

It is represented using the `` tag. This tag is an example of a nested tag, which means that it is composed of a parent and a child tag, as shown in *Figure 2a*.

This tag is used for displaying unordered bulleted list (as show in *Figure 2b*). This tag should always be followed by list tag ``.

```
<ul>
  <li>Sample list 1</li>
  <li>Sample list 2</li>
  <li>Sample list 3</li>
</ul>
```

Figure 2a. Unordered listed tag

- 
- Sample list 1
 - Sample list 2
 - Sample list 3

Figure

2b. Browser view

A complete example is shown in *Figure 3*:

```
<h1>HTML Tags</h1>
<p>This is an example of a paragraph.</p>
<ul>
  <li>Sample list 1</li>
  <li>Sample list 2</li>
  <li>Sample list 3</li>
</ul>
```

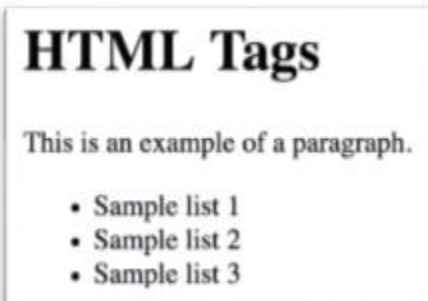


Figure 3. HTML heading and unordered bulleted list

HTML5 NEW ELEMENTS

An HTML page consist of elements that are placed inside tags. These elements help web browsers render web pages. This section describes some of the new elements that were introduced in HTML5.

Image Elements

It is represented using the `<figure>` tag. This tag is used to include images, diagrams and photos. The `<section>`, `<article>`, `<header>` and `<footer>` tags can contain the `<figure>` tag.

The `<figure>` tag can contain the `<figcaption>`, which in turn contains the caption for the figure that is in the `<figure>` tag. This configuration allows you to enter a description that can tie the figure more closely to the content.

The `<figure>` and `<figcaption>` tag structure is shown in *Figure 1*.

```
1 | <figure>
2 |   
3 |   <figcaption>Caption for the figure</figcaption>
4 | </figure>
```

Figure 1. Image elements

Media Elements

The `<section>` and `<article>` tags can also contain various media elements. HTML5 provides tags that quickly convey an understanding of their content. Media elements, such as music and video that were previously only embedded, can now be identified more precisely.

The `<audio>` tag

The `<audio>` tag identifies sound content, such as music or any other audio streams. The `<audio>` tag has attributes that control what, when, and how audio is played. The attributes are `src`, `preload`, `control`, `loop`, and `autoplay`. In *Figure 2*, the audio starts to play when the page loads and plays continuously. It also provides controls so the user can stop or restart the audio.

```
1 | <audio src="MyFirstMusic.ogg" controls autoplay loop>
2 |   Your browser does not support the audio tag.
3 | </audio>
```

Figure 2. Audio tags

The `<video>` tag

The `<video>` tag allows you to broadcast video clips or stream visual media. It features all of the attributes of the `<audio>` tag in addition to three other features: *poster*, *width*, and *height*. By using the poster attribute, you can identify an image to be used while the video is loading or not loading.

The `<video>` tag structure is shown in *Figure 3*:

```
1 | <video src="MyFirstMovie.ogg" controls="controls">
2 |     Your browser does not support the video tag
3 | </video>
```

Figure 3.

Video tags

The `<video>` and `<audio>` tags can contain the `<source>` tag, which defines multimedia resources for `<video>` and `<audio>` tags. With this element, you specify alternative video and audio files from which the browser can then choose based on its media type or codec support.

As shown in *Figure 4*, two choices are available. If the WMA version of the file cannot be played in the browser that is used, try the MP3; otherwise, display the message so that the user knows why the audio is unavailable.

```
1 | <audio>
2 |     <source src="/music/good_enough.wma" type="audio/x-ms-wma">
3 |     <source src="/music/good_enough.mp3" type="audio/mpeg">
4 |     <p>Your browser does not support the HTML 'audio' element.</p>
5 | </audio>
```

Figure 4. Source tag

The `<embed>` tag

The `<embed>` tag defines embedded content that can be brought into a page; for example, a plug-in for Adobe Flash SWF files. In *Figure 5*, the type attribute identifies the embedded source as a Flash file.

```
1 | <embed src="MyFirstVideo.swf" type="application/x-shockwave-flash" />
```

Figure 5. Embed tag

In addition to the attributes `src` and `type`, the `<embed>` tag includes `height` and `width` attributes.

HTML ATTRIBUTES

- HTML Attributes, provide extra information about the elements.
- All elements can have attributes.
- Attributes are specified in the start tag, in the format of `name = "value"` pairs.
- Example: `<input type="text" name="username">`

HTML5 ATTRIBUTE VALUES

The following new attributes are introduced in HTML5 for the Input Element tag:

- **Color chooser:** `<input type="color" />`

Provides a way to specify a color. The displayed output can vary from one browser or platform to another. It might be a textual input where the user can enter the color into a text field in its "#rrggbb" hexadecimal format, a platform-standard color picker, or a custom color picker window.

Figure 1 shows how the color chooser appears on Chrome or Firefox browser.

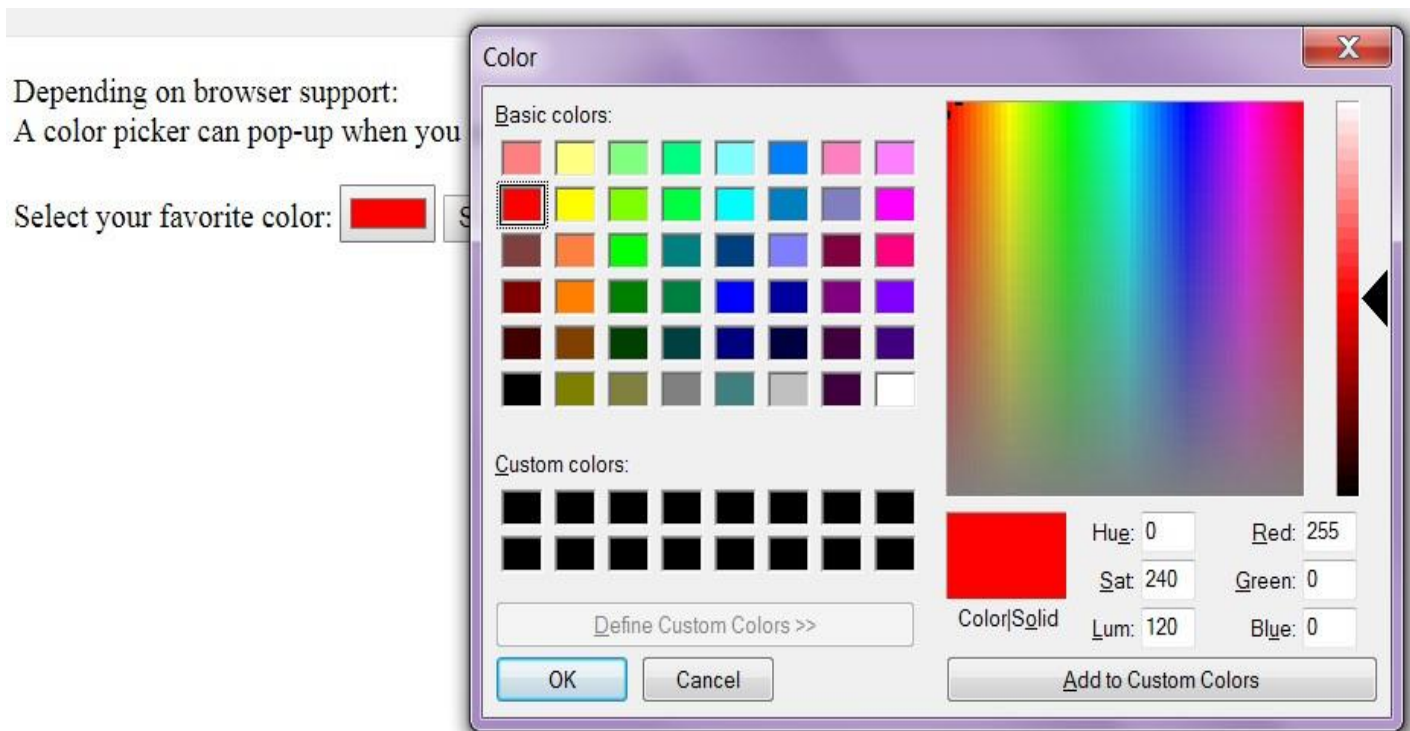


Figure 1: Color input type on browser

- **Input for date:** `<input type="date" />`

Defines a date control (year, month, or day) with no time zone. The displayed output can vary from one browser or platform to another. It might be displayed as a normal text input field “YYYY-MM-DD” or as a drop-down calendar.

Figure 2 shows how the date control appears on Chrome. It is not supported on Firefox

mm/dd/yyyy Submit

March, 2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 2: Date control input type on Chrome browser

- **Input for date with no timezone:** `<input type="datetime-local" />`

Provides input for a date and time (year, month, day, hour, minute, second, or fraction of a second) with no time zone. The date and time can be displayed as a normal text field, depending on the browser or platform that is used. The date also can be displayed as a drop-down calendar, and the time field can be displayed as a spinner control.

Figure 3 shows how the datetime-local appears on Chrome. It is *not* supported on Firefox.

mm/dd/yyyy --:-- -- Send

March, 2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 3: datetime-local input type on Chrome browser

- **Input for email:** `<input type="email" />`

Used to enter or edit an email address and displayed as a regular text input field in all other browsers. Browsers that do not support the email attribute manage this input attribute by reverting to `<input type="text" />`, which is referred to as “*graceful degradation*.” Graceful degradation means that the website continues to operate even when viewed with browsers that do not support all the features.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Figure 4a shows how the email input type appears on Chrome browser and the automatic validation is shown in Figure 4b. Figure 4c shows the validation on

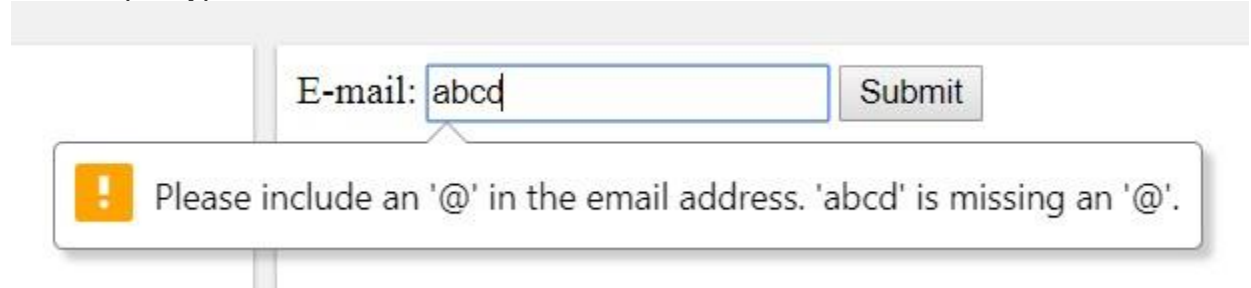
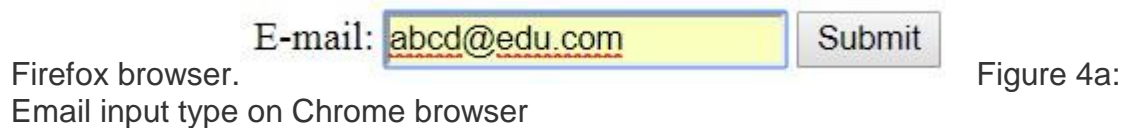


Figure 4b: Validation of email input type on Chrome browser

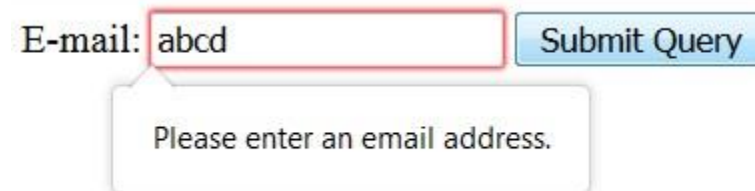


Figure 4c: Validation of email input type on Firefox browser

- **Input attribute list:** `<input type="text" list="some_list" />`

The **list** attribute refers to a `<datalist>` element that contains pre-defined options for an `<input>` element. Used for browsers that support the `<datalist>` feature. You can complete the list by nesting `<option>` elements inside the `<datalist>` tag. You also can provide compatibility with browsers that do not support the `<datalist>` tag by surrounding the `<option>` tags with a `<select>` tag. Browsers that understand the `<datalist>` tag ignore the surrounding `<select>` elements and use what is in the `<option>` tags.

Non-supporting browsers do not recognize the `<datalist>` tag. Instead, these browsers fallback to showing what they see as a standard `<select>` tag. Also, there is the autocomplete feature, which you can enter part of a list option in the input

field. For example, use “app” if one of the selectable options is “apples” or “or” if one of the selectable items is “oranges”.

Example of using `list` attribute with `datalist` tag

```
1 <input list="browsers" name="browser">
2 <datalist id="browsers">
3   <option value="Internet Explorer">
4   <option value="Firefox">
5   <option value="Chrome">
6   <option value="Opera">
7   <option value="Safari">
8 </datalist>
```

Figure 5a and Figure 5b show how the previous Example appears on Chrome and



Firefox browsers, respectively.

Figure 5a: datalist



appearing on Chrome browser
on Firefox browser

Figure 5b: datalist appearing

- **Input for number:** `<input type="number" />`

Takes a numeric value as input. You can optionally specify the minimum and maximum values. The displayed output can vary from one browser or platform to another. This attribute can be displayed as a normal text field or as a number selector. For the number selector, only the numbers between the minimum and the maximum are available for selection.

Figure 6a and Figure 6b show how the number input type appears in Chrome and Firefox browsers, respectively.



Figure 6a: number input type on Chrome browser



Figure 6b: number input type on Firefox browser

- **Input attribute range:** `<input type="range" />`

Takes a numeric range as input. The displayed output can vary from one browser or platform to another. This attribute typically is represented by using a slider or dial control. Only the numbers in the range of the minimum and the maximum are available for selecting. In some cases, extra JavaScript code is needed to display the slider's value.

Figure 7a and Figure 7b show how the range input type appears on Chrome and Firefox browsers respectively.



Figure 7a: range input type on Chrome browser



Figure 7b: range input type on Firefox browser

- **Input attribute search:** `<input type="search" />`

The differences between `<input type="search" />` and `<input type="text" />` are mostly in style. WebKit-based browsers return a history of recently searched text strings.

Figure 8a and Figure 8b show how the search input type appears on Chrome and Firefox, respectively.

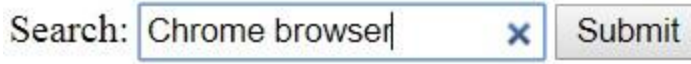
A search input field with the text "Chrome browser" and a "Submit" button. The input field has a small 'x' icon in the top right corner.

Figure 8a: search input type on

A search input field with the text "Firefox browser" and a "Submit Query" button.

Chrome browser
search input type on Firefox browser

Figure 8b:

- **Input attribute tel:** `<input type="tel" />`

Expects a telephone number as input. On its own, the `<input type="tel" />` provides nothing more than a text entry field in the browsers. It does not enforce numeric-only input, because many telephone numbers include other characters, such as the *plus* sign (+) or *hyphens* <->. You must supply your own pattern matcher if you want the browser to validate the telephone number.

In browsers not supporting tel input type, it appears as normal text input field as shown in *Figure 9*. The tel input type is only supported on Safari browser.

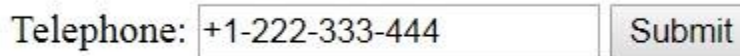
A telephone input field with the text "+1-222-333-444" and a "Submit" button.

Figure 9: tel input type on

Chrome and Firefox browsers

- **Input attribute url:** `<input type="url" />`

Used to validate that the information is entered in a properly formatted URL or web address. The Opera browser adds a prefix of `http://` to any URL that does not include a protocol.

Figure 10a show how the url input type appears on Chrome. The url validation is shown for Chrome and Firefox browsers on *Figure 10b* and *Figure 10c*, respectively

A form with the label "Add your homepage:" and an input field containing "http://www.ibm.com". A "Submit" button is next to the input field.

Figure 10a: url

input type on Chrome browser

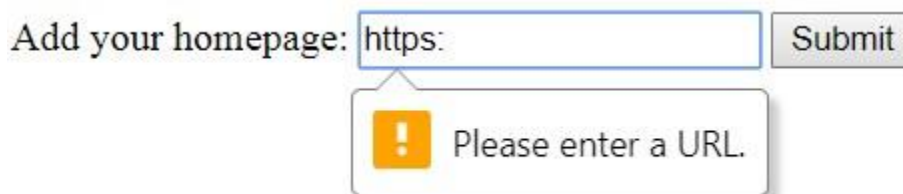
A form with the label "Add your homepage:" and an input field containing "https:". A "Submit" button is next to the input field. Below the input field, there is a warning message box with an exclamation mark icon and the text "Please enter a URL."

Figure 10b:

Validation of url input type on Chrome browser

Add your homepage:

Please enter a URL.

Figure 10c: Validation of url input type on Firefox browser

OBJECTIVE: CASCADING STYLE SHEETS (CSS)

Unit objectives

When you finish this unit you should be able to:

- Explain what CSS is.
- Write styles using CSS syntax.
- Apply style to elements.
- Bind style sheets to HTML page

CSS

CSS is the design that is layered over the top of an HTML web page. CSS is a style sheet language that describes how HTML elements are displayed. You use CSS to design (format) different aspects of your web pages.

What makes the style sheet “*cascading*” is that developers can apply CSS to create a uniform look throughout each element of each page of the website.

For websites, it is an important concept to separate the data from the design. The data is sent to the browser by using HTML, and the design is applied to that data by using a CSS. This separation allows people to render a web page without your design if they have special accessibility needs. It allows machines (such as search engines) to index a website without the design interfering.

You can use CSS to control a document’s appearance and specify style rules for the following web page elements:

- | | | |
|---------------|---------------|------------------|
| • Fonts | • Sizes | • Visual effects |
| • Text | • Borders | • Tables |
| • Colors | • Spacing | • Lists |
| • Backgrounds | • Positioning | |

Child and descendant elements often inherit styles that are defined for parent elements. However, exceptions to the rule exist.

You can code CSS as a style attribute in an HTML tag, a head section of a document, or an external document. The preference is to code CSS in external documents (referred to as *style sheets*).

CSS FORMAT

The following example shows a structure of a CSS style element:

```
html-tag-name
{
    css-property-key-1: css-value-1;
    css-property-key-2: css-value-2;
}
```

In this example, the `html-tag-name` can be one of the following elements:

- **Any of the tags you find in HTML code** (for example, `<a>`, `<div>`, ``, or `<label>`, etc.)
- **An id reference** that is displayed with a preceding hash symbol (#) in CSS code:

```
• #id-of-html-tag
• {
•     ...
• }
```

- **A class reference** that is displayed with a preceding dot/period (.) symbol in CSS code:

```
• .class-of-html-tag
• {
•     ...
• }
```

These parts of CSS (`html-tag-name`, `id-of-html-tag`, or `class-of-html-tag`) are known as **Simple Selectors** and can be nested (with a space between) to increase the granularity into HTML:

```
outer-html-tag-name inner-html-tag-name {...}
```

Simple Selectors can also be used as a list for applying one design element to many selectors:

```
1st-html-tag-name, 2nd-html-tag-name {...}
```

This example is an abstract way of understanding the formal syntax of a CSS. The remainder of this unit focuses on more concrete and useful examples that highlight the simplicity, potency, and flexibility of CSS.

BASE STYLES

When making a site design, begin by establishing the base style. Establish a base style by styling the `<body>` tag, as shown in the following example:

```
Body
{
background-color: #EEEEEE;
color: #000000;
margin: 0;
padding: 0;
text-align: left;
font-size: 100%;
font-family: sans-serif;
}
```

This example makes the following styles:

- Sets the background color: off-white (`background-color`).
- Ensures that the font color is black (`color`).
- Ensures that all content edges match the edge of the browser window frame (`margin` and `padding`).
- Horizontally aligns the textual content to the left (`text-align`).
- Sets the font size to the browser's default (`font-size`) and uses a sans-serif (a font without the little flicks around the edges) as font family (`font-family`).

These settings are simple. Generally, follow these guidelines:

- When a color is specified, use *Red-Green-Blue (RGB)* hexadecimal light values. You can find a list of RGB values at the [IBM Design site](#)
- When a size is specified, use *pixels* (indicated by a *px* after the number); an *em*, which is indicated by *em* after the number (that is, the size of the font multiplied by the specified number); or a *percentage*, which is indicated by a % after the number.
- Text can be aligned *left*, *right*, or *center*.
- Floats can also be *left* or *right*.
- Vertical alignments must be *top*, *middle*, or *bottom*.

- Fonts can be any specific font or font family (serif, sans-serif, or monospace) or even a downloadable font.

One of the most important decisions you must make when you are determining the design of your website is whether to use a **fluid** or a **fixed** layout:

- **A fluid layout:** is a layout in which the height and width of elements is flexible and can expand or contract based on the browser window, the operating system, and other user preferences. You specify these elements mostly by using *percentages* and *ems*.
- **A fixed layout:** is a layout where you specify the height and width of elements, and those values remain the same regardless of which operating system or browser you use to access the website. You specify these elements mostly by using *pixels*

When determining the layout, consider also the pros and cons for fluid and fixed layouts. The type of layout you choose depends on the type and amount of content and the target audience of the website.

APPLYING CSS TO HTML

To apply a CSS, you must tell the browser where to look for it. This step is the only true point where HTML references a CSS. You can make the reference in one of two ways—the `<style>` tag or the `<link>` tag.

Style tag

This method tends to be the quickest way to see your style applied to one web page, and it “dirty” the page with a non-HTML code. However, you apply this style to only one page (unless you are using a server-side language to include a header). If you copy and paste this style on each page, the page size increases (kilobytes, rather than width and height).

Therefore, this approach increases the load time of each page, which causes the user to wait longer. Time is crucial to the user, even if they can access fast internet speeds. So, how is it done? By adding the following markup to the HTML `<head>` tag:

```
1 <style>
2 /** Your CSS goes here **/
3 </style>
```

Linking to a CSS (Link tag)

This method is the cleaner way to apply styles. When you have a CSS in an external file, you can link to it from other pages, which ensures a clean HTML and a smaller page size (again, kilobytes rather than width and height). To use this method, add the following code to the HTML `<head>` tag section:

```
1 | <link href="http://www.example.com/styles/style.css" media="screen" rel="stylesheet" type="text/css" />
```

In this example, `style.css` is a plain-text file with your CSS code inside.

OBJECTIVE: EXERCISE 1

In this exercise, you apply some of the HTML and CSS concepts you learn in this skill

Exercise Summary Steps

The exercise includes the following steps:

- Create web page using HTML elements
- Write new styles.
- Add style to webpage and elements
- Run web page on browser

INSTRUCTIONS

In this exercise, you run code snippets for different HTML and CSS elements that were covered in the previous units. After creating different code snippets, the complete web page constructed will be as shown in *Figure 1*.

Completed code:

You can download the completed code for your use (see [Download](#)).

Running the web page on the browser

To see the output of the code after adding the HTML and CSS codes to the respective `.html` and `.css` files:

- Open the `Acme_United.html` file to view the output. The output that displays should be similar to that shown in *Figure 1*.

Acme United

A Simple HTML5 Example

[Home](#) [About Us](#) [Contact Us](#)

Article Heading

Primum non nocere ad vitam Paramus . . .

This is the first section heading

Scientia potentia est qua nocent docentp . . .

This is an aside that has multiple lines. . . .

Second section with mark, aside, menu & figure

. . . **veni, vidi, vici**. Mater . . .

[Clio](#) [Thalia](#) [Urania](#) [Calliope](#)



Figure 1. Stonehenge

This is a video section



This video will work in Mozilla Firefox or Google Chrome only.

Copyright: 2011 Acme United. All rights reserved.

Figure 1. Complete web

page

Follow these preparation steps:

1. Create a project folder named *HTML Project*.
2. Create a blank text document, and save it as html file in the HTML Project folder with the name **Acme_United.html**.

All of the HTML code snippets are stored in the **Acme_United.html** file.

3. Create another Notepad file in the same folder (HTML Project), and save it as **main-stylesheet.css**.

All the CSS code are stored in the **main-stylesheet.css** file.

COMPLETE THE HTML AND CSS CODE

Add the following HTML and CSS code described in the following steps to begin coding your web page: **Note:** To add code to `Acme_United.html` or `main-stylesheet.css` file:

Right click on the file and choose “**Open with**”, then choose your favorite text editor. For example: *Notepad ++* (<https://notepad-plus-plus.org/>) in Windows, or *Brackets* (<http://brackets.io/>) in Mac.

1. Add !doctype tag

In `Acme_United.html`, add the `<!doctype>` as the first tag. In HTML5, `<!doctype>` is simplified.

```
<!doctype html>
```

2. Add html tag

In `Acme_United.html`, below the `<!doctype>` tag, add the tag, which contains all of the other HTML elements, except the `<!doctype>` tag.

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

3. Add head tag

In `Acme_United.html`, insert the following HTML code to add the element which contains the `<title>` and `<link>` elements.

The `<title>` tag contains HTML5 Fundamentals Example, which will be the title that you see at the top of the browser when the web page is viewed.

The `<link>` tag that is shown in the following example identifies the CSS3 style sheet that is used to render the HTML5 page. The style sheet is called `main-stylesheet.css`:

```
<head>  
    <title>HTML5 Fundamentals Example</title>  
    <link rel="stylesheet" href="main-stylesheet.css"/>  
</head>
```

4. Add body tag

In `Acme_United.html`, add the `<body>` tag, followed by the `<header>` and `<hgroup>` tags. The `<h1>` tag contains the name of your fictitious company Acme United, and the `<h2>` tag contains the subtitle, “A Simple HTML5 Example.” The following code example shows this markup:

```
<body>
  <header>
    <hgroup>
      <h1>Acme United</h1>
      <h2>A Simple HTML5 Example</h2>
    </hgroup>
  </header>
```

In `main-stylesheet.css`, add the following CSS3 code, which is used to set up the page:

```
* {
  font-family: Lucida Sans, Arial, Helvetica, sans-serif;
}
body {
  width: 800px;
  margin: 0em auto;
}

header h1 {
  font-size: 50px;
  margin: 0px;
  color: #006;
}

header h2 {
  font-size: 15px;
  margin: 0px;
  color: #99f;
  font-style: italic;
}
```

You define the font for the page and then the font for the body. After the dimensions of the body are defined, you can design the header paragraph structure for first- and second-level heading tags. You use these headers for the page.

Note: All of the CSS3 code is stored to the `main-stylesheet.css` file.

5. Add nav tag

In `Acme United.html`,

add the `<nav>` tag, which is designed to handle the main site navigation, as shown in the following code:

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
</nav>
```

In `main-stylesheet.css`, add the code shown in the following example. Formatting navigation is managed by CSS3. Each `nav` tag definition that is shown in the following code represents a specific state of the `ul` and `li` elements inside the `nav` tag.

```
nav ul {
  list-style: none;
  padding: 0px;
  display: block;
  clear: right;
  background-color: #99f;
  padding-left: 4px;
  height: 24px;
}

nav ul li {
  display: inline;
  padding: 0px 20px 5px 10px;
  height: 24px;
  border-right: 1px solid #ccc;
}

nav ul li a {
  color: #006;
  text-decoration: none;
  font-size: 13px;
  font-weight: bold;
}

nav ul li a:hover {
  color: #fff;
}
```

6. Add an article tag

In `Acme_United.html`,
add the code shown in the following example.

The Article area is defined by the `<article>` tag and includes its own `<header>` information. The `<section>` that is contained in the `<article>` also contains a `<header>` tag of its own.

```
<article>
  <header>
    <h1>
      <a href="#" title="Link to this post" rel="bookmark">Article Heading </a>
    </h1>
  </header>
  <p> Primum non nocere ad vitam Paramus . . . </p>
  <section>
    <header>
      <h1>This is the first section heading </h1>
    </header>
    <p>Scientia potentia est qua nocent docentp . . . </p>
  </section>
```

In `main-stylesheet.css`, Add the following CSS code:

```
article > header h1 {
  font-size: 40px;
  float: left;
  margin-left: 14px;
}

article > header h1 a {
  color: #000090;
  text-decoration: none;
}

article > section header h1 {
  font-size: 20px;
  margin-left: 25px;
}

article p {
  clear: both;
  margin-top: 0px;
  margin-left: 50px;
}
```

Note that the definition for the paragraph, header, and section areas are all defined for the `<article>` tag in which they are contained. The `<h1>` tag that is defined in this example does not have the same format as the `<h1>` tag that is defined for the page-level `<h1>` tag.

7. Add a second section tag in the article tag

In `Acme_United.html`, add the following HTML code, which adds a second `<section>` element in the `<article>` element that contains the same basic information as the first `<section>`, except that now you use the `<aside>`, `<figure>`, `<menu>` and `<mark>` tags:

```
<section>
  <header>
    <h1>Second section with mark, aside, menu & figure</h1>
  </header>
  <p class="next-to-aside"> . . . <mark>veni, vidi, vici</mark>. Mater . . .</p>
  <aside>
    <p>This is an aside that has multiple lines. . . .</p>
  </aside>
  <menu label="File">
    <button type="button" onClick="JavaScript:alert('Clio . . .')">Clio</button>
    <button type="button" onClick="JavaScript:alert('Thalia . . .')">Thalia</button>
    <button type="button" onClick="JavaScript:alert
      ('Urania . . .')">Urania</button>
    <button type="button" onClick="JavaScript:alert
      ('Calliope . . .')">Calliope</button>
  </menu>
  <figure>
  <figcaption>Figure 1. Stonehenge</figcaption>
</figure>
</section>
```

The `<aside>` tag is used to present information that is not a part of the flow. The `<figure>` tag includes a graphic of Stonehenge. (The Stonehenge graphic can be found in the files you downloaded and named `stonehenge.jpg`. Extract the graphic and copy it to your project folder).

This `<section>` also contains the `<menu>` tag, which is used to create buttons with the names of the four Muses. When one of the buttons is clicked, it provides information about that particular Muse. The `<mark>` tag is used inside the `<p>` tag to highlight the words `veni`, `vidi`, and `vici`.

In `main-stylesheet.css`, add the following CSS code. This code includes a new definition for this section's `<p>` tag. It features a shorter width than the width set for the page. This change allows the aside to float to the right without overlapping the text.

```
article p.next-to-aside {
  width: 500px;
}

article > section figure {
  margin-left: 180px;
  margin-bottom: 30px;
}

article > section > menu {
```

```

        margin-left: 120px;
    }

    aside p {
        position: relative;
        left: 0px;
        top: -100px;
        z-index: 1;
        width: 200px;
        float: right;
        font-style: italic;
        color: #99f;
    }

```

8. Add a third section tag inside the article tag

In [Acme United.html](#), add the following HTML code to add a third `<section>` element in the `<article>` element, which is the video section. The video is a `.ogg` format video that auto-plays when the page is loaded, continuously loops, and provides controls for pausing and playing. `.ogv` files are often used for playing web page video content using the HTML5 `<video>` tag. Note that the `<audio>` tag works the same way.

```

<section>
    <header>
        <h1>This is a video section</h1>
    </header>
    <p><video src="http://people.xiph.org/~maikmerten/demos/BigBuckBunny.ogv"
controls autoplay loop>
        <div class="no-html5-video"><p>This video will work in
                                Mozilla Firefox or Google Chrome only. </p>
        </div>
    </video>
</section>
</article>

```

In [main-stylesheet.css](#), add the following CSS code, which includes the CSS definition for the video section:

```

article > section video {
    height: 200px;
    margin-left: 180px;
}

```

```
article > section div.no-html5-video{
    height: 20px;
    text-align: center;
    color: #000090;
    font-size: 13px;
    font-style: italic;
    font-weight: bold ;
    background-color: #99f;
}
```

9. Add a footer tag

In [Acme United.html](#), add the following HTML code, to add the footer and the closing of the page:

```
<footer>
    <p>Copyright: 2011 Acme United. All rights reserved.</p>
</footer>
</body>
</html>
```

In [main-stylesheet.css](#), add the following CSS code, which includes the CSS definition for the footer:

10. Follow the steps in the previous section “Running the web page on the browser”, to see how the web page appears on browser

```
footer p {
    text-align: center;
    font-size: 12px;
    color: #888;
    margin-top: 24px;
}
```


1. HTML is..?

- ☒ The standard language used to describe the structure of pages by using markup
- ☒ A programming language used to render web pages.

2. To embed sound in your page, use the tag:

- ☒ <player>
- ☐ <sound>
- ☐ <audio>
- ☐ <media>

3. In CSS, to select the element with the ID "elementId," use the selector:

- ☐ elementId
- ☐ .elementId
- ☐ #elementId

4. To reference an external CSS file, refer to it in the:

- ☐ <footer>
- ☐ <header>
- ☐ <body>
- ☐ <head>

5. For <Section> and <Article> tags:

☐ Article tag uniquely identifies the content, and the <section> tag contains related information.

☐ Article tags can contain Section tags.

☐ All of the above.

☐ Section tags can contain Article tags.

6. What is the correct tag to reference external CSS file?

☐ <link rel="stylesheet" type="text/css" src="mystyle.css">

☐ <style type="text/css" href="mystyle.css">

☐ <link rel="stylesheet" type="text/css" href="mystyle.css">

7. To make a unordered list, use the tag:

☐ <list>

☐

☐

8. In CSS, to select all <p> tags that are direct children of a <div> tag, use this tag:

☐ div > p

☐ div.p

☐ .p.div.

☐ p > div.

9. To add navigation links, use the tag:

☐ <nav>

☐ <navigation>

☐ <links>

10. What is the valid CSS syntax?

☐ p {color:black;}

☐ {p color:black;}

☐ {p:color=black;}

☐ p {color=black;}