



# DIPLOMA IN SYSTEMS ANALYSIS

## E-LEARNING GUIDE

*PRE-REQUISITE UNIT*

*HTML, CSS & JAVASCRIPT*

**Copyright © NUS, 2019.** The contents contained in this document may not be reproduced in any form or by any means, without the written permission of NUS, other than for the purpose for which it has been supplied.

## Table of Contents

<b>E-Learning Overview .....</b>	<b>6</b>
Introduction: .....	6
Study Plan:.....	<b>Error! Bookmark not defined.</b>
How does this E-Learning work? .....	7
Reference Texts and Web resources.....	8
Software Tools.....	8
Notepad & Notepad ++ .....	8
Visual Studio .....	9
Online HTML Editors .....	9
WYSIWYG Editors .....	9
<b>Lesson 01: HTML basics.....</b>	<b>10</b>
Objective .....	10
Preparing the PC.....	10
My First HTML Program .....	10
Steps for creating an HTML.....	10
Viewing the Output. ....	11
HTML Page Structure .....	12
The <!DOCTYPE> Declaration.....	12
Tags.....	13
Common Declarations.....	13
HTML Versions .....	13
More with HTML - Checking different headings .....	14
Practice Examples.....	15
Practice 01 .....	16
<b>Lesson 02: Developing a Web Site .....</b>	<b>17</b>
Webpage design rules .....	17
Page Design .....	18
HTML Documents.....	18
Structure.....	18
Creating an HTML Document.....	19

Markup Tags .....	20
HTML Element .....	20
HTML Attributes .....	22
Backgrounds .....	24
Comments in the HTML .....	25
Practice Exercises .....	26
Practice 02 .....	32
<b>APPENDIX TO LESSON 02 - HTML Tags LIST .....</b>	<b>34</b>
Basic HTML Tags .....	34
Link Tags .....	34
Formatting Tags .....	35
Images Tags .....	36
Audio / Video Tags .....	36
List Tags .....	36
Table Tags .....	37
Styles and Semantics .....	37
Meta Info .....	38
Programming Tags .....	38
<b>Lesson 03: Further HTML Elements &amp; Concepts .....</b>	<b>39</b>
Color Values .....	39
Color Names .....	39
HTML Lists .....	40
Unordered Lists .....	40
Ordered Lists .....	40
Tables .....	41
Defining Tables: .....	41
Table Size .....	44
HTML Layout - Using Tables .....	45
Practice 03 .....	47
<b>Lesson 04: Cascading Style Sheets .....</b>	<b>49</b>
Aim .....	49
Introduction .....	49

Advantages of Cascading Style Sheets .....	49
Types of Style Sheets.....	50
Creating your own Styles.....	54
More on HREF attribute & file specification .....	55
Practice 04 .....	56
<b>Lesson 05: Forms in HTML.....</b>	<b>57</b>
How does an HTML form work?.....	57
Form Examples:.....	57
Constructing a Form in HTML.....	59
Input Field Types .....	60
Input Field Examples .....	61
Checkboxes.....	61
Radio Buttons .....	62
Dropdown Lists.....	63
Submitting the Data .....	64
Practice 05 .....	66
<b>Lesson 06: Introduction to JavaScript .....</b>	<b>67</b>
Introduction.....	67
What is JavaScript?.....	67
Uses of JavaScript.....	67
Writing JavaScript.....	67
The SCRIPT Tag .....	68
Implementing JavaScript .....	68
Programming Basics .....	68
Variables .....	69
Variable declaration .....	69
Variable Names .....	69
Functions .....	70
Practice 06 .....	70
<b>Lesson 07: JavaScript Programming Constructs .....</b>	<b>75</b>
Introduction.....	75
Online Learning .....	75

---

Programming Concepts .....	75
Operators .....	75
Comparisons .....	76
String comparison .....	77
Conditional operators: if .....	77
Logical operators .....	78
Loop Constructs .....	79
WHILE Loop .....	79
FOR Loop .....	80
Functions .....	81
Invoking the Function .....	82
Functions with Arguments .....	83
Functions that return a value .....	84
Accessing HTML fields from JavaScript .....	85
Practice 07 .....	87
<b>Web Application Development .....</b>	<b>89</b>
Aim .....	89
Menu .....	89
Menu Bar .....	90
Practice 08 .....	95

## E-Learning Overview

### Introduction:

This guide facilitates self-paced learning of HTML, CSS and JavaScript. This guide does not substitute a text book. Learners are required to read up the concepts using recommended text books, online resources and vendor documents. Upon completion of the prescribed study students should attempt on a set of activities to reinforce their learning and gain competence in each topic. This guide aims to provide a roadmap for study of the topics and practice.

---

## How does this E-Learning work?

This topic is scheduled as e-Learning as it is more practice based and most topics (except for some programming) is mostly straightforward and can be read and understood by most students.

However, to use your time better and to focus a path for learning the following is the recommended steps:

1. Follow the sequence provided in this guide. Most self-learning effort may go unproductive if there is no plan or roadmap. Hence **this guide is the harness that keeps you on track** and ensures the essentials are learnt by you without distracting. If you go without this guide you may spend too much time and may go deeper than what a beginner can understand.
2. Read up **one lesson at a time**.
3. There are examples given in this guide. **Practice these examples without fail**. Yes, it may look trivial, you may already have answers in the description. But yet, doing once helps in consolidating the information provided.
4. This companion guide is not a text book – hence only brief descriptions are provided. If things are not too clear, **read the relevant pages from your chosen material** (either the web or text that you choose to use).
5. **Do the exercises at the end of each Lesson**. These ensure that you are able to implement what you learnt in the lesson. The exercises may expect you to use things which may go slightly beyond what has been described in this guide. This forces you to explore a bit more when you do the exercise and your learning gets consolidated.
6. **Complete the tasks for submission**. Pay attention to doing these to best possible. For instance you should focus on impressive layouts and clearly readable codes.
7. Then move on to next lesson.

As you may see above, a topic as this emphasis is on doing and learning rather than reading and understanding.

Happy Learning!

---

## Online videos for Beginners

HTML Tutorial for Beginners: HTML Crash Course [2021]

<https://www.youtube.com/watch?v=qz0aGYrrlhU>

Learn JavaScript - Full Course for Beginners

<https://www.youtube.com/watch?v=PkZNo7MFNFg>

JavaScript Tutorial for Beginners - Full Course in 8 Hours [2020]

[https://www.youtube.com/watch?v=Qqx\\_wzMmFeA](https://www.youtube.com/watch?v=Qqx_wzMmFeA)

## Reference Texts and Web resources

1. Sams Teach Yourself: HTML, CSS, and JavaScript, by Julie Meloni & Jenifer Kyrimin, 3<sup>rd</sup> Ed, Pearson Education. 2018.
2. Beginning HTML and CSS by Rob Larsen, O'Reilly, 2013.
3. [www.w3schools.com](http://www.w3schools.com) (online tutorials and resources)
4. <https://javascript.info>
5. HTML & CSS Quick Guide (App based Audio-Visual presentation) downloadable from Microsoft store at: <https://www.microsoft.com/en-us/p/html5-css-quicklook-guide/9wzdncrdg4rk?activetab=pivot%3Aoverviewtab>

Note: There are numerous resources on web as well as several beginners book. Just pick whichever suits your style of learning. The above need not be the only resource for you to use and there is no necessity that you use any of them as the learning in this companion guide is independent of the above references.

## Software Tools

### Notepad & Notepad ++

Notepad may appear to be a primitive tool, but we recommend that you should do at least some of your practice in this. The absence of help facility will ensure that you carefully read, understand and code your HTML, CSS & JavaScript with error free syntax.

**Notepad** usually comes with Windows.

You can download **Notepad++** from: <https://notepad-plus-plus.org/download>

The advantage of Notepad++ is in its ability to provide formatting and clarity.

If you are using Mac, then possibly you should use **TextEdit** software to write codes.



---

## Visual Studio

Visual studio is a large IDE from Microsoft. It is a profession IDE which supporting enterprise level development. You can download Visual Studio as a registered student of ISS – NUS. Contact IT Department of ISS for more information.

That said there is no urgency to switch to Visual Studio for learning HTML or CSS, so you can keep it in view for later use.

## Online HTML Editors

There are numerous online editors. The advantage is you do not need to install these on your machines.

In addition, the online editors help you in typing and running the HTML codes to see the output in one window (rather than type, save, browse mode of operation when you use other editors – more on this later).

We recommend that you can use W3 Schools online editor for learning. This can be accessed at: <https://www.w3schools.com/tryit/>

The left window will provide scope for you to type the codes and when you hit the RUN button the output is displayed on the right side.

## WYSIWYG Editors

There is a class of HTML editors called WYSIWYG editors. We recommend that you DO NOT use these at this stage as it hampers your learning. Both installable and online versions are available.

The WYSIWYG may be good option for projects after you have mastered the syntax. The WYSIWYG helps you to create Web pages using visual tools, drag and drop options and mouse based formatting (just the way you may use MS-Word or other word processing software). The HTML codes are generated automatically as you create the page layout and content.

As you see this is not a learning tool and is hence discouraged.

For information of course, you can explore an online WYSIWYG html editor available at <https://html-online.com/editor/>

## Lesson 01: HTML basics

### Objective

This lesson helps the student to create the first HTML page and to understand the various components of the HTML code. The focus of this lesson is to make the learners get familiar with various ways to create web pages using different IDEs, Online interactive learning tools. In addition, this Activity is aimed at facilitating the learners to understand the structure of the HTML code, the structure of the web page as seen from the HTML code perspective and learn the most essential code elements. This activity kickstarts the learner to set up the environment, learn the various tools and rapid build a First HTML and does not aim to provide a full detail on the HTML tags, which will be covered in the next lesson.

### Preparing the PC

Prior to working on the activities, the learner should prepare PC with appropriate software. We recommend that you install two or more editors and browsers listed below. While it is sufficient that you install on only one in each, for better learning outcome we recommend that you use one pure text editor without any help elements and one that has help elements and WYSIWYG features.

**Editor:** Notepad ++ and Visual Studio.

**Browser:** Internet Explorer (Windows PC) or Safari (Mac) and Google Chrome.

### My First HTML Program

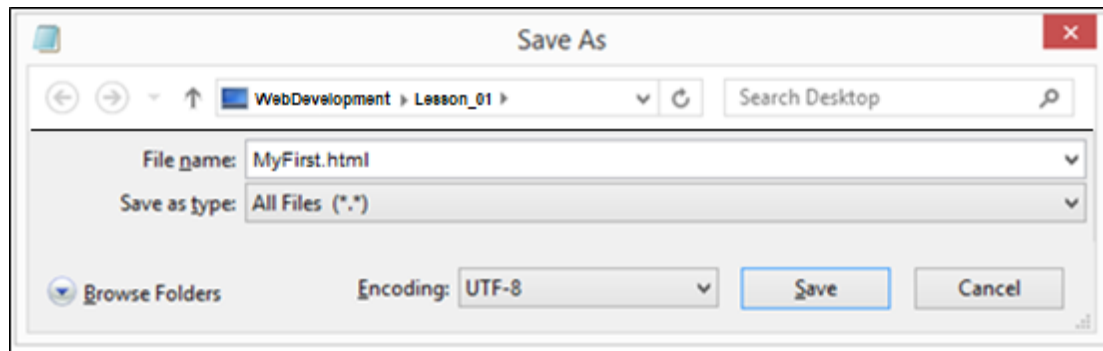
HTML files have a basic structure that you **MUST** work within. The structure of the HTML is provided in the first activity that you will perform as shown in the codes in step three below.

### Steps for creating an HTML.

1. Create a folder called **WebDevelopment** and create a subfolder within that folder and name it **Lesson\_01**.

*Note: If you are using a shared PC, as a practice save all your work to your external storage before you go home and copy back for next lesson if using a new machine.*

2. Open Notepad++ and save the blank new file as **MyFirst.html** in the Lesson\_01 folder. Use "Save As" and make sure the settings are as below (remember to use file type as *All Files* and remember to name the file with a .html)



3. Write the html code given below on the MyFirst.html notepad++ file.

```
<!DOCTYPE html>
<html>
  <head>
    <title> Page Title </title>
  </head>

  <body>

    <h1> This is a Heading </h1>
    <p> This is a paragraph. </p>

  </body>
</html>
```

4. Save the file.

## Viewing the Output.

To test the code that you have just typed, save the file and open the file in a browser of your choice. This can be any standard browser such as Internet Explorer, Google Chrome, Firefox etc.

You can view your output using one of the following approaches:

- You can simply double click the MyFirst.html file and the page will be displayed on the default browser.
- Alternately you can right click on the file and 'Open with' your preferred browser.
- A third approach is to launch your browser and type in the address bar:

file:///C:/WebDevelopment/Lesson\_1/MyFirst.html

You will see the following output:

## This is a Heading

This is a paragraph.

### HTML Page Structure

The HTML page structure is as below. Whatever you write on the white area (*i.e.*, within the body tags) only you see on the browser. Note the `<head>` and `<title>` tags. Whatever you write on the `<title>` tag will appear on the header of the browser at the top left-hand corner.

```
<html>

  <head>

    <title>Page title</title>

  </head>

  <body>

    <h1>This is a heading</h1>

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

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

  </body>

</html>
```

### The `<!DOCTYPE>` Declaration

The `<!DOCTYPE>` declaration helps the browser to display a web page correctly.

There are different document types on the web.

To display a document correctly, the browser must know both type and version.

The doctype declaration is not case sensitive as shown below. All cases are acceptable:

```
<!DOCTYPE html>
<!DOCTYPE HTML>
<!doctype html>
<!Doctype Html>
```

## Tags

- The <> bracket with text inside are called **tags**. Note they come in pairs.
- Tags are used to provide formatting (or other) information to the browsers.
- Tags should maintain the exact spelling for it to work. Tags are *not* case sensitive, hence can be written in either capital or small letters.
- Tags usually occur in pairs (there are some exceptions to this though). There is a beginning tag < > and an end tag <\ >. The beginning tag signals browser that a tag is started and the closing tag signals browser that the tag/task has ended.
- The tags are interpreted by the browser to format the text occurring in between the begin and end tags. As seen earlier, the <H1> ... <\H1> tag informs the browser to display text in *Heading 1* format.
- A list of tags is provided at the end of this activity. You are required to practice each tag and understand the the functionality of the tags.

## Common Declarations

### HTML5

```
<!DOCTYPE html>
```

### HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

### XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

**In this course we use HTML5 for all the examples.**

## HTML Versions

Since the early days of the web, there have been many versions of HTML:

Version	Year
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

## More with HTML - Checking different headings

```
<!DOCTYPE html>
<html>
  <body>
    <h1> This is heading 1 </h1>
    <h2> This is heading 2 </h2>
    <h3> This is heading 3 </h3>
    <h4> This is heading 4 </h4>
    <h5> This is heading 5 </h5>
    <h6> This is heading 6 </h6>
  </body>
</html>
```

You will see the following output:

**This is heading 1**

**This is heading 2**

**This is heading 3**

**This is heading 4**

**This is heading 5**

**This is heading 6**

## Practice Examples

Try these examples and record what you observe regarding these tags.

You should try two approaches:

- Mandatory: Firstly, create html pages using note pad with the following examples embedded. Launch them on a browser and write your observations.
- Second approach: Use an WSWIG IDE to create the examples below as part of and view the outcome AND/OR Use an online site such as W3 Schools or other Free Tutorial sites to type in the tags to observe the functionality of various tags.

### Example 1 - Paragraphs

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

### Example 2 – Displaying words in italic, bold or underline.

```
<p>  
One of the greats in sports <b>Roger Federer</b> who is a world class <i>tennis  
player </i> and hails from <u>Switzerland </u> and has won several grand slam titles.  
</p>
```

### Example 2 - HTML hyperlinks

```
<a href="http://www.google.com">This is a link</a>
```

### Example 3 – Colouring text of an entire paragraph with style attribute

```
<p style="color:blue"> This paragraph will be displayed in blue</p>
```

### Example 4– Colouring selected words in a paragraph with style attribute

```
<p > The colour of <span style="color:blue"> sky </span> is blue</p>
```

## Practice 01

Create a web page using below text with exact format style

Note: Styles that you need to fix are headings, fonts, colour, paragraphs etc. The text within the paragraph will auto wrap based on the browser width and is not a part of the requirement for HTML formatting. You may refer to the table overleaf to locate appropriate tags.

### Internet

#### From Wikipedia, the free encyclopedia

This article is about the worldwide computer network. For other uses, see Internet (disambiguation).

Not to be confused with the World Wide Web.

The **Internet** is a global system of interconnected computer networks that use the standard Internet protocol suite(TCP/IP) to link several billion devices worldwide. It is a *network of networks* that consists of millions of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries an extensive range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web(WWW), the infrastructure to support email, and peer-to-peer networks for file sharing and telephony.

The origins of the Internet date back to research commissioned by the United States government in the 1960s to build robust, fault-tolerant communication via computer networks.[2] This work, combined with work in the United Kingdom and France, led to the primary precursor network, the ARPANET, in the United States. A 1980 paper refers to "the ARPA internet". *The interconnection of regional academic networks in the 1980s marks the beginning of the transition to the modern Internet.* From the early 1990s, the network experienced sustained exponential growth as generations of institutional, personal, and mobile computers were connected to it.

For courses on this Internet Technologies visit [Institute of Systems Science](#) course page.



## Lesson 02: Developing a Web Site

### Webpage design rules

The first phase in creating a new web site is planning. This involves determining the site's navigation structure, content, and page layout. It is only after this has been accomplished that HTML coding or authoring tools like FrontPage can be used effectively to actually create the site.

The planning process involves:

#### Define site purpose

- What are the web site's objectives?
- Who is the target audience?
- What is their connectivity (dial up, broadband, network, etc.)?
- What computers (PC/Mac) and browsers (IE, Netscape, Opera, etc.) do they use?

#### Define content

- What you have and what you need.
- How it will be organized. Categorize information rather than defining categories and filling them. Build the site structure around content organization.
- What graphics are required and how many? Remember, the larger the number of graphics, the longer the page will take to download.
- Develop a file naming convention and be consistent in implementing it. Avoid underscores in files & folder names and use underscore if required.
  - Web files and folders should not have any spaces and be in lowercase.
  - Use the underscore ( \_ ) to connect words in file names.
  - File names cannot contain colons (:), asterisks (\*), questions marks (?), or other special characters.
- Determine site architecture and navigation.
- A simple flowchart-style sketch usually works best. Hierarchical models are best suited to the web.

## Page Design

With web pages, the layout design process accounts for the arrangement of text and graphics elements on the page. These elements can be broadly divided into:

**Page Header:** Located at the top of the page, it includes the page banner or title and the navigation bars.

**Page Footer:** Located at the bottom of the page, this is where you insert copyright and authoring information, the date of the most recent update, institutional affiliation.

**Side Navigation:** This is usually a rail along the side of the page that displays the global or local navigation.

## HTML Documents

HTML documents are made up of tags or angled brackets (<>) that contain an element within them to determine the layout and formatting of the web page. For example, <B> is the tag used to define text in bold where “B” is the element contained within the tag.

Elements can also include attributes, or additional information for that element. Attributes are included inside the start tag. For example, you can specify the alignment of images (top, middle, or bottom) by including the appropriate attribute with the image source HTML code.

## Structure

The basic HTML document is contained within the <HTML> </HTML> tags and comprises two sections: Head and Body.

The Head contains the page title and meta-tags within the <HEAD> </HEAD> tags. Any JavaScript code that is used, as well as Cascading Style Sheet information is also contained within the Head. This section will not be displayed on the web page.

The Body holds the actual content of the page (text, graphics, lists, etc.) contained within the <BODY> </BODY> tags. The <HTML>, <HEAD>, <TITLE>, and <BODY> tags are also referred to as document tags.

A basic HTML document would look something like this:

```
<HTML>
  <HEAD>
    <TITLE> A Simple HTML Example </TITLE>
  </HEAD>

  <BODY>
    Welcome to the world of HTML. This is the first paragraph.
  </BODY>
</HTML>
```

- Tags are case insensitive, i.e. <B> will work as well as <b>.
- Not all tags work with all web browsers. If a browser does not recognize a tag, it simply ignores it. It will display the information contained within the tags however.

## Creating an HTML Document

Since HTML documents are essentially text files with the extension **.htm** or **.html**, any basic text editor like Notepad WordPad can be used to create one. You can also the HTML view in authoring systems like FrontPage or Dreamweaver, or HTML hand coding applications like Home Site.

HTML documents do not recognize white spaces created by formatting tools such as spaces, linefeeds, and carriage returns. These are automatically compressed into a single space when the page is displayed in the browser. For instance, the HTML code in the example above can also be written out as

```
<HTML>
  <HEAD>
    <TITLE> A Simple HTML Example </TITLE>
  </HEAD>

  <BODY>
    Welcome to the world of HTML. This is the first paragraph.
  </BODY>
</HTML>
```

and it would still display in exactly the same manner in the browser. However, it is harder to read and edit in this format. Consequently, for clarity in reading and editing HTML, it is best to structure the document using carriage returns and indents.

## Markup Tags

The information contained in the Body section of an HTML document is formatted using markup tags. These tags allow us to create paragraphs, line breaks, headings, lists, etc. They can also be used to set the position of the content using alignment attributes.

HTML tags consist of ELEMENTS and ATTRIBUTES. Tags (as seen before) are enclosed within a pair of angular brackets. Tags usually occur in pairs a start tag and end tag as seen before. The content in between the start and end tags is formatted based on the meaning associated with the tag (key words within the tag). Elements (tag) may further optionally contain **attributes** to provide further qualification/enhancement to the tag.

## HTML Element

The most common markup tags are described below:

### Heading

These tags are used to highlight text by making them bigger and bolder than normal text. There are six levels of headings numbered **1** through **6**, where 1 is the biggest and 6 is the smallest heading. Headings are specified as **<H<sub>y</sub>> ... </H<sub>y</sub>>** where **H** stands for heading & **y** is the level number (1 to 6).

E.g.:

```
<HTML>
  <HEAD>
    <TITLE> Headings </TITLE>
  </HEAD>
  <BODY>
    <H1> Level 1 heading </H1>
    <H2> Level 2 heading </H2>
  </BODY>
</HTML>
```

## Paragraph

These tags, denoted by `<P>` `</P>`, are used to define paragraphs. Since HTML does not recognize carriage returns, it is especially important to use these tags to indicate paragraphs, or you will end up with as one long block of text.

E.g.:

```

<HTML>
<HEAD>
  <TITLE> Working with Paragraphs</TITLE>
</HEAD>
<BODY>
  <P>
    This is the opening paragraph for this page. It does not have much information
    at the moment, but it indicates how a paragraph is constructed in HTML.
  </P>
  <P> This is the second paragraph on this page. </P>
</BODY>
</HTML>

```

## Line Breaks

The `<br>` tag is used when you want to start a new line, but don't want to start a new paragraph. The `<br>` tag forces a line break wherever you place it. It is similar to single line spacing in a document (analogous to **SHIFT-ENTER** in MS-Word, instead of **ENTER** key).

This HTML Code:	Would Display:
<code>&lt;p&gt; This &lt;br&gt; is a para &lt;br&gt; graph with line breaks &lt;/p&gt;</code>	This is a para graph with line breaks

The `<br>` tag has no closing tag.

## HTML Attributes

Certain parameters are included within the **opening tag** to provide additional element properties (such as colorization, measurement, location, alignment, or other appearances) to the data between the markup tags. These parameters are called HTML Attributes.

Most attributes require a value. In HTML, the value can be left unquoted if it doesn't include spaces (**name=value**), or it can be quoted with single or double quotes (name='value' or name="value"). That said, it is recommended to enclose Attribute values in quotes.

We have already subtly introduced attributes in an earlier activity where we used the **href** attribute in the **<a>** tag, to provide the link parameter url.

The most common examples of using attributes are provided below:

### Aligning Paragraphs

Paragraphs can be positioned or **aligned** relative to the sides of the page by using the **ALIGN="alignment" attribute** within the opening tag where the value for alignment is left, right, center, or justified. The default alignment for a paragraph tag is **left**.

E.g.

```
<HTML>
  <HEAD>
    <TITLE>Working with Paragraphs</TITLE>
  </HEAD>

  <BODY>
    <P align= "center">
      This is the opening paragraph for this page. I have
      used the attribute align with the value center to put it
      in the middle of the page.
    </P>
    <P align="right">
      This is the second paragraph on this page and it is
      right aligned.
    </P>
  </BODY>
</HTML>
```


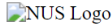
## Horizontal Rule

You can insert a rule (line) in your page by using the `<hr>` tag. The width and align attributes are used to show how multiple attributes can be used. You can specify as many number of attributes with the name-value pairs separated by a space.

This HTML Code:	Would Display:
<code>&lt;hr width="50%" align="center"&gt;</code>	<hr/>

## Images

The `<img>` tag is used to display an image. The src attribute in the img tag will point to a file. The src element can be used either with local files or network / internet located files

This HTML Code:	Would Display:
<pre>&lt;img src="http://www.nus.edu.sg/identity/images/identity/logo/fullcolorlogo.jpg" alt="NUS Logo" &gt;</pre>	<p><u><i>If the image file is available:</i></u></p>  <p>If we try to display an image that does not exist, the value of the alt attribute will be displayed instead.</p> <p><u><i>If the image file is NOT available:</i></u></p>  <p>If we try to display an image that does not exist, the value of the alt attribute will be displayed instead.</p>

## Style Attribute

The style element can be used in `<p>` and many other tags such as `<span>`, `<div>`, `<table>` etc. within which text for display can be provided. The style attribute introduced subtly in an earlier Activity is used to provide display styles such as fonts, color, emphasis or other features for visual presentation.

Styles can be either the default styles defined (applied by) the browser settings or can be explicit styles using CSS for standards and customisation within your project or context.

More about styles will be discussed in a later lesson dealing with CSS.

## HTML Fonts

The `<font>` tag in HTML is deprecated. In future versions of HTML, style sheets (CSS) will be used to define the layout and display properties of HTML elements.

## Backgrounds

The `<body>` tag has two attributes where you can specify backgrounds. The background can be a color or an image.

### Bgcolor

The **bgcolor** attribute specifies a background-colour for an HTML page. The value of this attribute can be a hexadecimal number, an RGB value, or a colour name:

```
<body bgcolor="#000000">  
<body bgcolor="rgb(0,0,0)">  
<body bgcolor="black">
```

The lines above all set the background-colour to black.

### Background Attribute

The background attribute can also specify a background-image for an HTML page. The value of this attribute is the URL of the image you want to use. If the image is smaller than the browser window, the image will repeat itself until it fills the entire browser window.

```
<body background="clouds.gif">  
<body background="https://media.giphy.com/media/10539PNHnSuPTi/giphy.gif">
```

The URL can be relative (as in the first line above) or absolute (as in the second line above). If you want to use a background image, you should consider your answers (choice) for the following questions:

- Will the background image increase the loading time too much?
- Will the background image look good with other images on the page?
- Will the background image look good with the text colors on the page?
- Will the background image look good when it is repeated on the page?
- Will the background image take away the focus from the text?



## Comments in the HTML

You can use comments to write notes to yourself or write a helpful message to someone looking at your source code.

### Comment Tag

The comment tag is used to insert a comment in the HTML source code. A comment can be placed anywhere in the document and the browser will ignore everything inside the brackets.

This HTML Code:	Would Display:
<code>&lt;p&gt; This html comment &lt;!-- This is a comment --&gt; would be displayed like this.&lt;/p&gt;</code>	This HTML comment would be displayed like this.

*Notice you don't see the text between the tags `<!--` and `-->`. If you look at the source code, you would see the comment. To view the source code for this page, in your browser window, select **View** and then select **Source**.*

## Practice Exercises

Read through the **Appendix to this Lesson** that describe various tags. You are required to focus on the following sections:

- Basic HTML Tags
- Link Tags
- Formatting Tags
- Images Tags

*(avoid sections on Table, Lists and Styles which will be discussed in later lesson)*

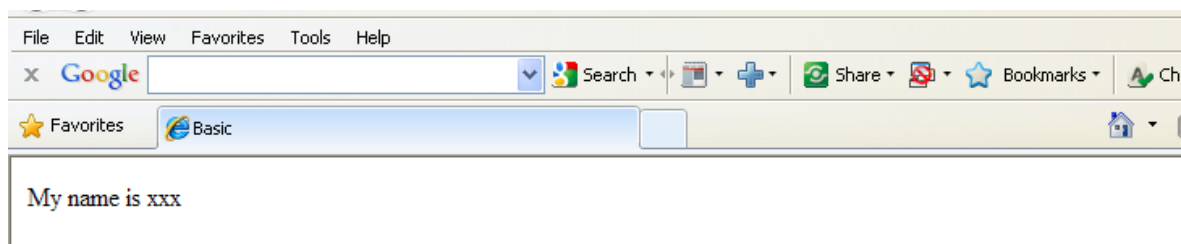
Try out these exercise **(not for submission, but must practice)**:

### Example 1 – Starting a web page

- In your PC folder in My documents, create a new folder called **HTML**
- Open Notepad++ from the Start menu
- Save this file as **basic.html** in your HTML folder
- Type in the following HTML code & tags exactly as they appear below (with your name instead of xxx):

```
<HTML>
  <HEAD>
    <TITLE>Basic</TITLE>
  </HEAD>
  <BODY>
    My name is xxx
  </BODY>
</HTML>
```

- Save your work again (Ctrl-S)
- Open My Documents, then your ICT folder, then your new HTML folder
- Open the Basic file to see it displayed as a web page in Internet Explorer
- It should look like this:



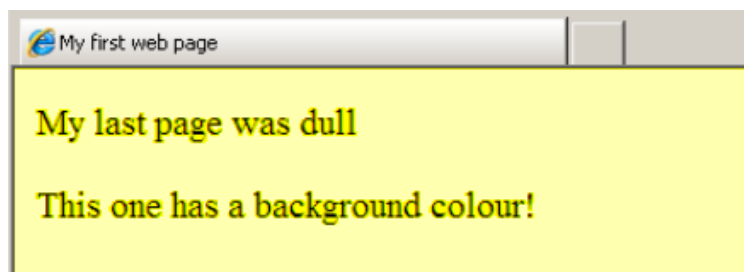
- Notice what is written in the title bar

## Exercise 2 – Changing the background colour

- Open Notepad ++
- Save this file as **background.html** in your HTML folder
- Type in the HTML tags below:

```
<HTML>
  <HEAD>
    <TITLE>Background</TITLE>
  </HEAD>
  <BODY STYLE="BACKGROUND-COLOR:YELLOW;">
    <P>My last page was dull</P>
    <P>This one has a background colour!</P>
  </BODY>
</HTML>
```

- The **<P>** tag is called the paragraph tag and anything typed between the opening **<P>** and closing **</P>** tags will display as a paragraph.
- Note 'color' is spelt in the American way
- Save your work again (Ctrl-S)
- Open your HTML folder
- Open the background.htm file to see it displayed as a web page in Internet Explorer. It should look something like ...



### Example 3 – Different sizes of text

- Open Notepad ++
- Save this file as **sizes.html** in your HTML folder
- Type in the HTML tags below:

```
<HTML>
  <HEAD>
    <TITLE>Text sizes</TITLE>
  </HEAD>
  <BODY>
    <BODY STYLE="BACKGROUND-COLOR:RED;">
      <H1>Test heading 1</H1>
      <H2>Test heading 2</H2>
      <H3>Test heading 3</H3>
      <H4>Test heading 4</H4>
      <H5>Test heading 5</H5>
      <H6>Test heading 6</H6>
    </BODY>
  </HTML>
```

- Save your work then open it in Internet Explorer to view your page. It should look like this ...



#### Example 4 – Changing font style & colour

- Open Notepad ++
- Save this file as **font.html** in your HTML folder
- Type in the HTML tags below:

```
<HTML>
<HEAD>
<TITLE>
Font style and colour
</TITLE>
</HEAD>
<BODY style="background-color: pink;">
<H1>Font style and colour</H1>
<P style="color: red;">My text should be red, </p>
<P style="font-family: comic sans MS;">This text is
in Comic Sans MS</p>
</BODY>
</HTML>
```

- Save your work then open it in Internet Explorer to view your page.

#### Example 5 – Text attributes & alignment

- Text attributes mean displaying the text as bold, italics or underlined.
- Alignment is when text is centred or right-aligned, for example
- Open Notepad++
- Save this file as **text.html** in your HTML folder
- Type in the HTML tags below:

```
<HTML>
<HEAD>
<TITLE>Text attributes and alignment</TITLE>
</HEAD>
<BODY style="background-color: aqua;">
<p style="text-align: right;">My web page will be amazing...</p>
<p style="text-align: center;">...all my friends will be impressed.</p>
<p style="text-align: left;">Hooray for HTML!</p>
</BODY>
</HTML>
```

- Save your work again then open it in Internet Explorer to view your page.
- Return to the Notepad file. You are now going to make some changes to the text
- Note that the word “center” is spelled in the American way.
- The word “amazing” need to be in bold. Use the tags <B> & </B>
- The word “impressed” needs to be underlined. Use the <U> tag, remember to add the closing tag.
- Save your work again then open its Internet Explorer window, click Refresh, and view the changes to your page.

## Example 6 – Adding an image

- Go on the Internet and find an image you would like to include on a new web page
- Click on the image and drag it into your HTML folder. Note that the image file must be saved in the same folder as the web pages. Rename it to have a short name.
- Open Notepad++
- Save this file as **picture.html** in your HTML folder
- In the example below an image called “Simpsons” was inserted into the page. You need to replace the word “Simpsons” with the name of your saved image
- You will also need to enter the type of image: jpg or gif or png. In the example below a png is used. Generally, photos are jpg, and cartoon pictures are gif or png.
- Type in the HTML tags below:

```
<HTML>
  <HEAD>
    <TITLE>Image</TITLE>
  </HEAD>
  <BODY STYLE="BACKGROUND-COLOR:AQUA;">
    <H1>This web page has a blue background</H1>
    <H2>It displays an image</H2>
    <IMG SRC="simpsonsfamily.png"> </IMG>
  </BODY>
</HTML>
```

- Save your work again then open it in Internet Explorer to view your page.
- Can you add a tag to centre the image?
- Save the file again. Then open it in Internet Explorer window, click Refresh, and view the change to your page.
- Go on the Internet again, and find an animated image. Save it to your HTML folder then insert it into your web page.
- Save the file again. Then open it in Internet Explorer window, click Refresh, and view the change to your page.

## *Changing the size of an image*

- You can set the size of your image if it appears too big or too small on your web page. To do this add the height and width attributes, for example:

```
<IMG SRC="global-warming.jpg" height="300" width="200" />
```

## Exercise 7 – Linking to a website

- Open Notepad++
- Save this file as **website.html** in your HTML folder.
- Type in the HTML tags below:

```
<HTML>
<HEAD>
<TITLE>Lists</TITLE>
</HEAD>
<BODY style="background-color:#f82;">
<p style="font-size:18pt">To the news!</p>
<p><a href="http://news.bbc.co.uk/">Tell me what's going on please...</a></p>
</BODY>
</HTML>
```

- Save your work again then open it in Internet Explorer to view your page.
- Test your hyperlink.
- Note that you have to put some text to act as the link. In the example the text 'To the news!' will be in blue & underlined to show it's a hyperlink. This text goes after the opening <A HREF...> tag, and before the closing </A>.

## Exercise 8 – Adding a horizontal line

- A horizontal line can be used to separate your main heading from the rest of the page.
- Open Notepad++
- Save this file as **horizontal.html** in your HTML folder.
- Type in the HTML tags below:

```
<HTML>
<HEAD>
<TITLE>Horizontal line</TITLE>
</HEAD>
<BODY style="background-color: black;">
<p style="color:yellow;">Hey check this out!</p>
<HR/>
<p style="color:red;">This page has lines.</p>
<HR/>
<p style="color:green;">But the colours are <B>Horrible</B></p>
</BODY>
</HTML>
```

- Save your work again then open it in Internet Explorer to view your page.

## Practice 02

Using HTML concepts learnt above and referring to further tags provided in Appendix to this lesson overleaf, create web pages as below. The specifications are:

### FIRST PAGE:

- Provides brief introduction about Computers and contains headings for the page, and Subtitles for “History of Computers” & “Components of Computers”.
- An Author name which when clicked hyperlinks to the Author Page (SECOND PAGE)
- Formatted paragraphs under each section. In one (or more if you prefer) of the paragraphs on the Components of the Computer section, images of the component (in the example we have just used a picture of keyboard).
- You may also attempt to right justify the keyboard image and wrap the text adjacent to the picture.
- You may notice some font, colour and other implementation shown for your implementation. You should attempt to get the page as near as the one that is shown in samples below.
- Name this page as IntroComputers.html.

### SECOND PAGE

- Provides the Author profile.
- Contains a brief profile of the author of the Introduction to Computers (FIRST PAGE).
- Contains a photo of the Author.
- Contains contact email, which when clicked should launch an email client.
- Name the page Author.html

Note: For your submission you may enhance these requirements trying out more tags from the appendix. However, your submission should cover all the requirements shown in the sample.

*Page Design & Contents overleaf...*



## IntroComputers.html

### Introduction to Computers

[James Brown](#)

#### History of Computers

One of the earliest machines designed to assist people in calculations was the **abacus** which is still being used some 5000 years after its invention.

In 1642 Blaise Pascal (a famous French mathematician) invented an adding machine based on mechanical gears in which numbers were represented by the cogs on the wheels.

Englishman, Charles Babbage, invented in the 1830's a "Difference Engine" made out of brass and pewter rods and gears, and also designed a further device which he called an "Analytical Engine". His design contained the five key characteristics of modern computers:-

1. An input device
2. Storage for numbers waiting to be processed
3. A processor or number calculator
4. A unit to control the task and the sequence of its calculations
5. An output device

Augusta Ada Byron (*later Countess of Lovelace*) was an associate of Babbage who has become known as the first computer programmer.

#### Computer Parts

##### Central Processing Unit

A **central processing unit (CPU)** is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logic, controlling, and input/output (I/O) operations specified by the instructions. The computer industry has used the term "central processing unit" at least since the early 1960s.<sup>[1]</sup>

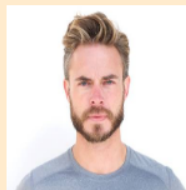
##### Keyboard

A computer keyboard is a typewriter-style device which uses an arrangement of buttons or keys to act as mechanical levers or electronic switches. Following the decline of punch cards and paper tape, interaction via teleprinter-style keyboards became the main input method for computers. Keyboard keys (buttons) typically have characters engraved or printed on them,<sup>[2]</sup> and each press of a key typically corresponds to a single written symbol. However, producing some symbols may require pressing and holding several keys simultaneously or in sequence. While most keyboard keys produce letters, numbers or signs (characters), other keys or simultaneous key presses can produce actions or execute computer commands. In normal usage, the keyboard is used as a text entry interface for typing text and numbers into a word processor, text editor or any other program. In a modern computer, the interpretation of key presses is generally left to the software. A computer keyboard distinguishes each physical key from every other key and reports all key presses to the controlling software.



## Author.html

### James Brown



[jbrown@gmail.com](mailto:jbrown@gmail.com)

**James Brown** is a Professor in the Department of Computer Science at Yorker University, where he has been since 1994. From 2013 to 2016, he served as Department Chair. He is the recipient of the Sloan Fellowship, the NSF Career Award, and was named Computer Journal's "Top 100 Technology Innovators" in 2000. He is affiliated with the Artificial Intelligence Laboratory and leads the research efforts of Woods Institute for High Performance Computing at Cherry Hill.

His research interests span across AI, computer networking and network science. Much of his work has been on improving the understanding, design, and performance of parallel and networked computer systems, mainly through the application of data mining, statistics, and performance evaluation.

## APPENDIX TO LESSON 02 - HTML Tags LIST

A near complete list of HTML tags is presented in this section. You may try to practice many of these though not explicitly explained above. Some of the tags will be discussed in a later *Activity* but are presented here for completeness. Please do not try to understand each tag in the list below as many of the tags you may never have to use in your lifetime.

### Basic HTML Tags

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Defines information about the document
<title>	Defines a title for the document
<body>	Defines the document's body
<meta>	Defines metadata about an HTML document
<base>	Specifies the base URL/target for all relative URLs in a document
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
 	Inserts a single line break
<hr>	Defines a thematic change in the content
<!--...-->	Defines a comment

### Link Tags

Tag	Description
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

## Formatting Tags

Tag	Description
<acronym>	<i>Not supported in HTML5.</i> Use <abbr> instead. Defines an acronym
<abbr>	Defines an abbreviation or an acronym
<address>	Defines contact information for the author/owner of a document/article
<b>	Defines bold text
<blockquote>	Defines a section that is quoted from another source
<center>	<i>Not supported in HTML5.</i> Use CSS instead. Defines centered text
<cite>	Defines the title of a work
<code>	Defines a piece of computer code
<del>	Defines text that has been deleted from a document
<dfn>	Represents the defining instance of a term
<font>	<i>Not supported in HTML5.</i> Use CSS instead. Defines font, color & size for text
<i>	Defines a part of text in an alternate voice or mood (italics)
<ins>	Defines a text that has been inserted into a document
<mark>	Defines marked/highlighted text
<pre>	Defines preformatted text
<progress>	Represents the progress of a task
<q>	Defines a short quotation
<rp>	Defines what to show in browsers that do not support ruby annotations
<ruby>	Defines a ruby annotation (for East Asian typography)
<s>	Defines text that is no longer correct (similar to <del> tag)
<small>	Defines smaller text
<strike>	<i>Not supported in HTML5.</i> Use <del> instead. Defines strikethrough text
<strong>	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<template>	Defines a template
<time>	Defines a date/time
<u>	Defines text that is stylistically different from normal text (underline)
<var>	Defines a variable

## Images Tags

Tag	Description
<img>	Defines an image
<map>	Defines a client-side image-map
<area>	Defines an area inside an image-map
<canvas>	Used to draw graphics, on the fly, via scripting (usually JavaScript)
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content
<picture>	Defines a container for multiple image resources

## Audio / Video Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements (<video>, <audio> and <picture>)
<track>	Defines text tracks for media elements (<video> and <audio>)
<video>	Defines a video or movie

## List Tags

Tag	Description
<ul>	Defines an unordered list
<ol>	Defines an ordered list
<li>	Defines a list item
<dir>	Not supported in HTML5. Use <ul> instead. Defines a directory list
<dl>	Defines a description list
<dt>	Defines a term/name in a description list
<dd>	Defines a description of a term/name in a description list

## Table Tags

Tag	Description
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting

## Styles and Semantics

Tag	Description
<style>	Defines style information for a document
<div>	Defines a section in a document
<span>	Defines a section in a document
<header>	Defines a header for a document or section
<footer>	Defines a footer for a document or section
<main>	Specifies the main content of a document
<section>	Defines a section in a document
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element
<data>	Links the given content with a machine-readable translation

## Meta Info

Tag	Description
<head>	Defines information about the document
<basefont>	Not supported in HTML5. Use CSS instead. Specifies a default color, size, and font for all text in a document










## Programming Tags

Tag	Description
<script>	Defines a client-side script
<noscript>	Defines an alternate content for users that do not support client-side scripts
<applet>	Not supported in HTML5. Use <embed> or <object> instead. Defines an embedded applet
<embed>	Defines a container for an external (non-HTML) application
<object>	Defines an embedded object
<param>	Defines a parameter for an object

## Lesson 03: Further HTML Elements & Concepts

### Color Values

Colors are defined using a hexadecimal notation for the combination of red, green, and blue color values (RGB). The lowest value that can be given to one light source is 0 (hex #00). The highest value is 255 (hex #FF). This table shows the result of combining red, green, and blue:

Color	Color HEX	Color RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

### Color Names

A collection of colour names is supported by most browsers. To view a table of color names that are supported by most browsers refer to W3 Consortium site or reference textbook.

**Note:** Most modern browsers support about 140 names. A small sample is given below for reference

Color	Color Name
	AliceBlue
	Aquamarine
	Black
	Blue
	Brown
	Red
	Green
	Yellow
	HoneyDew

## HTML Lists

HTML provides a simple way to show unordered lists (bullet lists) or ordered lists (numbered lists).

### Unordered Lists

An unordered list is a list of items marked with bullets (typically small black circles). An unordered list starts with the `<ul>` tag. Each list item starts with the `<li>` tag.

This HTML Code	Would Display
<pre>&lt;ul&gt; &lt;li&gt;Coffee&lt;/li&gt; &lt;li&gt;Tea&lt;/li&gt; &lt;/ul&gt;</pre>	<ul style="list-style-type: none"><li>• Coffee</li><li>• Tea</li></ul>

### Ordered Lists

An ordered list is also a list of items. The list items are marked with numbers. An ordered list starts with the `<ol>` tag. Each list item starts with the `<li>` tag.

This HTML Code	Would Display
<pre>&lt;ol&gt; &lt;li&gt;Coffee&lt;/li&gt; &lt;li&gt;Tea&lt;/li&gt; &lt;/ol&gt;</pre>	<ol style="list-style-type: none"><li>1. Coffee</li><li>2. Tea</li></ol>

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.



## Tables

Tables are useful to present information in a tabular fashion. Tables can consist of several columns and rows forming cells within which information can be presented in a clear fashion.

Tag	Description
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table


### Defining Tables:

Tables are defined with the <table>tag. A table is divided into rows (with the <tr>tag), and each row is divided into data cells (with the <td>tag). The letters td stands for table data, which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc.

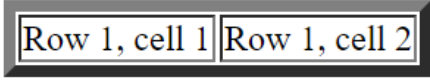
This HTML Code	Would Display				
<pre>&lt;table&gt; &lt;tr&gt; &lt;td&gt;row 1, cell 1&lt;/td&gt; &lt;td&gt;row 1, cell 2&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;row 2, cell 1&lt;/td&gt; &lt;td&gt;row 2, cell 2&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>	<table> <tr> <td>row 1, cell 1</td><td>row 1, cell 2</td></tr> <tr> <td>row 2, cell 1</td><td>row 2, cell 2</td></tr> </table>	row 1, cell 1	row 1, cell 2	row 2, cell 1	row 2, cell 2
row 1, cell 1	row 1, cell 2				
row 2, cell 1	row 2, cell 2				

## Tables and the Border Attribute

To display a table with borders, you will use the border attribute.

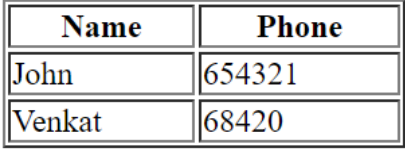
This Code	Would Display
<pre> &lt;table border="1"&gt; &lt;tr&gt; &lt;td&gt;Row 1, cell 1&lt;/td&gt; &lt;td&gt;Row 1, cell 2&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt; </pre>	

*and....*

This HTML Code	Would Display
<pre> &lt;table border="5"&gt; &lt;tr&gt; &lt;td&gt;Row 1, cell 1&lt;/td&gt; &lt;td&gt;Row 1, cell 2&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt; </pre>	

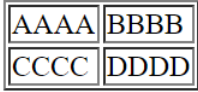
## Headings in a Table

Headings in a table are defined with the <th>tag.

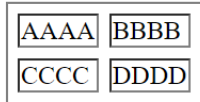
This HTML Code	Would Display
<pre> &lt;table border="1" &gt; &lt;tr&gt; &lt;th&gt;Name&lt;/th&gt; &lt;th&gt;Phone&lt;/th&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;John&lt;/td&gt; &lt;td&gt;654321&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;Venkat&lt;/td&gt; &lt;td&gt;68420&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt; </pre>	

## Cell Padding and Spacing

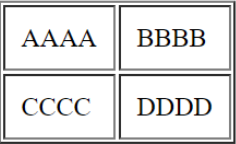
The <table>tag has two attributes known as cellspacing and cellpadding. Here is a table example without these properties. These properties may be used separately or together.

This HTML Code	Would Display
<pre>&lt;table border="1"&gt; &lt;tr&gt; &lt;td&gt;AAAA&lt;/td&gt; &lt;td&gt;BBBB&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;CCCC&lt;/td&gt; &lt;td&gt;DDDD&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>	

**Cellspacing** is the pixel width between the individual data cells in the table (The thickness of the lines making the table grid). The default is zero. If the border is set at 0, the cellspacing lines will be invisible.

This HTML Code	Would Display
<pre>&lt;table border="1" cellspacing="5"&gt; &lt;tr&gt; &lt;td&gt;AAAA&lt;/td&gt; &lt;td&gt;BBBB&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;CCCC&lt;/td&gt; &lt;td&gt;DDDD&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>	

**Cellpadding** is the pixel space between the cell contents and the cell border. The default for this property is also zero. This feature is not used often, but sometimes comes in handy when you have your borders turned on and you want the contents to be away from the border a bit for easy viewing. Cellpadding is invisible, even with the border property turned on. Cellpadding can be handled in a style sheet.

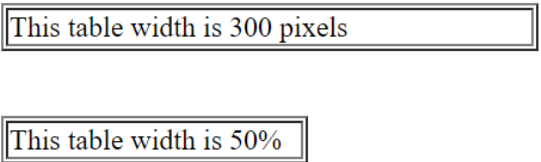
This HTML Code	Would Display
<pre>&lt;table border="1" cellpadding="10"&gt; &lt;tr&gt; &lt;td&gt;AAAA&lt;/td&gt; &lt;td&gt;BBBB&lt;/td&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;CCCC&lt;/td&gt; &lt;td&gt;DDDD&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>	

## Table Size

Table sizes by default to the browser, i.e., the width determined by the browser window width and the columns determined spaced proportionately within the table width and the text within the cell. The row height is defaulted to the line spacing and any padding that is applied. This ensures the tables are fully displayed to the best fit the browser size. However, there are instances where the table may need to be explicitly sized in absolute measurement sizing or as a relative measure to the page. In such instances we can explicitly dimension the tables.

## Table Width

The width attribute can be used to define the width of your table. It can be defined as a fixed width or a relative width. A fixed table width is one where the width of the table is specified in pixels. For example, this code, `<table width="550">`, will produce a table that is 550 pixels wide. A relative table width is specified as a percentage of the width of the visitor's viewing window. Hence this code, `<table width="80%">`, will produce a table that occupies 80 percent of the screen.

This HTML Code	Would Display
<pre>&lt;table width="300"&gt; &lt;tr&gt; &lt;td&gt;This table width is 300 pixels&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt; &lt;p&gt;&lt;p&gt; &lt;table width="50%"&gt; &lt;tr&gt; &lt;td&gt;This table width is 50% &lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>	

Note: There are arguments in favour of giving your tables a relative width because such table widths yield pages that work regardless of the visitor's screen resolution. For example, a table width of 100% will always span the entire width of the browser window whether the visitor has a 800x600 display or a 1024x768 display (etc). Your visitor never needs to scroll horizontally to read your page, something that is regarded by most people as being very annoying.

## HTML Layout - Using Tables

One very common practice with HTML, is to use HTML tables to format the layout of an HTML page. A part of this page is formatted with several columns or rows as per the design layout.

Unlike tables in other contexts and environments it should be noted that an HTML table need not be strictly 'rectangular' in the sense each row or column may contain differing number of cells. For instance, it is quite possible that row 1 may contain four columns, while row 2 may contain just 3 columns. This way it is a good tool to define page layouts.

One of the classical and frequently used page layout is as below:

HEADER	
LEFT MENU	CONTENT
FOOTER	

The HEADER section is used to provide heading information such as company name, logo and main menu at the top.

The LEFT MENU section is used to provide navigation in a *Tree* type menu. This enables the user to navigate through the website with ease.

The FOOTER section is used to provide information such as copyright, contact information etc.

The CONTENT section contains the actual page information of the various pages content. As such while the Header, Left and Footer will remain standard for each page, the Content section is where the details regarding the page is displayed.

The implementation of the above layout in HTML codes is as below:

```
<!DOCTYPE html>
<html>
<body align="center">
  <table border="1" width="400" cellspacing="0">
    <tr height="50"><td colspan="2"> HEADER SECTION </td> </tr>
    <tr height="300"><td width="25%"> LEFT MENU </td> <td> CONTENT </td></tr>
    <tr height="50"><td colspan="2"> FOOTER SECTION </td> </tr>
  </table>
</body>
</html>
```

The **colspan** attribute of the **<td>** tag is used to ensure that the Header and Footer row cells span the two columns represented by the Left and Content sections.

Likewise, if situation requires you can use the **rowspan** attribute of the **<tr>** tag to span multiple cells represented by another column in the table. The above design did not require this and hence not used.

In the above example, we have used width and height attributes appropriately to size the sections. This is up to you to design as per your page.

## Practice 03

### Question 1

Write HTML codes to create the following tables.

**Table 1**

**Time Table**

Period	Course	Teacher	Room
1	Math	Andrew	26
2	English	Varni	48
3	PE	Wong	Gym
4	Science	Schaefer	40
5	History	Rand	31
6	Computers	Rubin	46

**Table 2**

Energy expenditures for Each Region (millions of dollars)			
		Type	
		Residential Customers	Business Customers
		Expenditures	Expenditures
		Sum	Sum
Region	Division		
East	Changi	\$7,477	\$5,129
	Tampines	\$19,379	\$15,078
West	Jurong	\$5,478	\$4,729
	Clementi	\$13,959	\$12,619

## Question 2

It's time to create your own Profile page. Use your text editor to create a page which contains the following:

- the required HTML page codes
- link to another web page
- an email link
- a picture/graphic
- a list of information

## Question 3

Enhance the above introducing page layout designed with the aid of HTML Table.



---

## Lesson 04: Cascading Style Sheets

### Aim

Understand how to use CSS with HTML and practice and learn to create pages with CSS and HTML.

### Introduction

Cascading style sheets (CSS) is a standard defined by the World Wide Web

Consortium that offers designers more flexibility and accuracy when defining the appearance of text and formats than standard HTML. Essentially, CSS allows designers to manipulate the appearance of the webpage without affecting its HTML structure. For example, if you wanted to change all the text in your document to blue, and all the headlines to green, with standard HTML one would have to manually change each of the elements on the page one by one. However, by using CSS, it's possible to redefine the body elements in the entire document to turn blue with just one instruction, and likewise change the headers (h1, h2, etc.) to green. Hence it is possible to alter the standard appearance of an HTML design to a preferred project or organisational standard.

The CSS can be used in conjunction with HTML based formatting. In the above cited example while all of the body will appear in blue, you may still use tag specific color attribute to work on sections where a colour alternate to blue is preferred. Hence this gives the advantage of generic setting without interfering with HTML or Page level settings that you may require.

In addition to redefining standard attributes CSS gives facilities to create named styles which can be referred in the HTML or in another CSS definition. Named styles may define multiple formatting attributes such as font, colour, size etc. collectively. The cascading nature of style sheets lets us define a style and use it in another style.

### Advantages of Cascading Style Sheets

If used appropriately, cascading style sheets can be quite beneficial to the web designer. In addition to text, cascading style sheets can define spacing between lines, the size of the type in pixels instead of points, and define specific fonts within pages.

Formatting each individual page becomes very cumbersome for the designer, especially if their site contains hundreds of pages. With style sheets, one only needs to specify such preferences once, and the style can be applied to an entire site. And if the designer decides to change the width of the page, then he or she only needs to change this preference in one place, rather than having to search through all of the pages to change the HTML.

Style sheets offer flexibility in terms of the presentation effects that they provide. Properties such as colour, background, margin, border, and many more can be applied to all elements. With just HTML and its proprietary extensions, one must rely on attributes like BGCOLOR, which are only available for a few elements. Style sheets give the flexibility of applying a style to all paragraphs, or all level-two headings, or all emphasized text.

## Types of Style Sheets

There are several different types of style sheets, and different circumstances in which you would use them.

The first, although ad hoc in nature is what is referred as **inline style**. We had already subtly introduced this in the first lesson through the following sample:

```
<p> The colour of <span style="color:blue"> sky </span> is blue</p>
```

A more structured approach which gives code segregation and reusability is to use one of the two approaches described below.

Embedded style sheets are an internal part of the HTML document. All of the code is written inside the **<head>** tag of the document and affects only the one page. The following is an example of embedded CSS code:

```
<STYLE TYPE="text/css">
  H1 {
    color: blue;
    font-family: verdana
  }
</STYLE>
```

Embedded style sheets are useful when you want to apply this style to only a single page.

**External style sheets** are the most powerful. A single file can be used to format an infinite number of pages. Then, a single change can affect all the other pages instantly.

The contents of an external style sheet file look similar to the embedded code, except that they are not part of the HTML page. Instead, they are located in a separate file, with a .css extension as opposed to .html. This .css file contains a list of styles with no other HTML code. Instead of embedding code directly into the HTML document, create an external link to the external CSS file. This link will go under **<head>** section in html document and would point to the css file through the **href** attribute.

```
<LINK REL=stylesheet HREF="mystyles.css" TYPE="text/css">
```

### Example:

Create the following HTML page:

```
<!DOCTYPE html>
<html>

<head>

</head>
<body>

    <h1> This is my HTML Page</h1>

    <h2> Nothing new here</h2>

    <p> This page is written <span style="color:red">normally</span>
    as any other HTML page. It will display either in default
    font, color, size attributes or in customised fashion
    depending on whether the <b>head</b> section contains
    a style or not</p>
</body>
</html>
```

The above will display as:

# This is my HTML Page

## Nothing new here

The page is written **normally** as any other HTML page. It will display either in default font, color, size attributes or in customised fashion depending on whether the **head** section contains a style or not

Notice in the above, all fonts are default, except the word “normally” which has been explicitly styled in red colour and the word head which has been formatted in bold font. This is a normal HTML situation that we are familiar with.

We will now see how styles can be applied to the default text of the entire body and specifically for H2 section (i.e., the statement Nothing new here).

Create the following HTML page whose body section is the same as before but additionally has styles defined using <style> tag in the head section before the body:

```
<!DOCTYPE html>
<html>
<head>
  <style type="text/css">
    body{
      font-family:Calibri;
      color:darkblue;
      background-color:lightcyan
    }
    h2 {
      color:brown;
      font-family:verdana;
      font-style:italic
    }
  </head>
</style>
<body>
  <h1> This is my HTML Page</h1>
  <h2> Nothing new here</h2>
  <p> This page is written <span style="color:red">normally</span>
    as any other HTML page. It will display either in default
    font, color, size attributes or in customised fashion
    depending on whether the <b>head</b> section contains
    a style or not</p>
</body>
</html>
```

The above will display as:

# This is my HTML Page

## *Nothing new here*

The page is written normally as any other HTML page. It will display either in default font, color, size attributes or in customised fashion depending on whether the **head** section contains a style or not

Notice that the entire body has been changed to dark blue text and the background to light cyan. While the H2 tag is qualified further as brown superceding the body colour. Also notice that the HTML level styling of the words “normally” and “head” remains intact demonstrating the cascading effect.

The same can be achieved with External stylesheet as well by a combination of two files below:

#### MyPage.html:

```
<!DOCTYPE html>
<html>
<head>
  <link rel=stylesheet href="MyStyle.css" type="text/css">
</head>
</style>
<body>
  <h1> This is my HTML Page</h1>
  <h2> Nothing new here</h2>
  <p> This page is written <span style="color:red">normally</span>
    as any other HTML page. It will display either in default
    font, color, size attributes or in customised fashion
    depending on whether the <b>head</b> section contains
    a style or not</p>
</body>
</html>
```

#### MyStyle.css:

```
body{
  font-family:Calibri;
  color:darkblue;
  background-color:lightcyan
}

h2 {
  color:brown;
  font-family:verdana;
  font-style:italic
}
```

The output for the above will be the same as the internal styles shown before.

## Creating your own Styles

In the above section you had redefined tags to be formatted the way you desired. There may be instances where you may need to apply a particular style combination selectively on certain texts, the way we did for the word “normally”. In the case of “normally” styling you had just desired to fix the colour through the style attribute. In the event you wish to specify a variety of styling characteristics such as font, emphasis, colour etc., you may need to write tags that can get quite intrusive and lengthy. Further if the same style that you applied for normally you should reapply else where, the same set of attributes may need to be repeated each time.

To address the above requirement, CSS provides a facility called **CSS Class**. The CSS class is a named definition which can be applied repeatedly at many places in the HTML. To demonstrate we modify the previous example to introduced a named style CSS Class.

Notice that class definition is similar to the one we did for H2 or the body elements, but instead it provides a name of your choosing. To indicate that this is your own defined name, you should prefix the name with a dot as shown in example below (we have used .splText for class name, we have referred this in the span tag using the class attribute and for brevity of example removed h2 style definition ).

```
<!DOCTYPE html>

<html>
<head>
  <style type="text/css">
    body{
      font-family:Calibri;
      color:darkblue;
      background-color:lightcyan
    }
    .splText{
      color:red;
      font-style:bold
    }
  </head>
</style>
<body>
  <h1> This is my HTML Page</h1>
  <p> This page is written <span class="splText">normally</span>
    as any other HTML page. It will display either in default
    font, color, size attributes or in customised fashion
    depending on whether the <b>head</b> section contains
    a style or not</p>
  <p class="splText"> This entire paragraph will be styled in
    special text style that we created, in addition to the
    Styling the word “normally” above. </p>
  <p> Back to normal again! </p>
</body>
</html>
```

The above html and CSS codes will display the following:

## This is my HTML Page

This page is written **normally** as any other HTML page. It will display either in default font, color, size attributes or in customised fashion depending on whether the **head** section contains a style or not

This entire paragraph will be styled in special text style that we created, in addition to the Styling the word “normally” above.

Back to normal again!

### More on HREF attribute & file specification

The href attribute is used for specifying file locations. For instance the files for a css link attribute may be a **.css** file while that for image element can be a **.jpg, .png, .gif** or any accepted picture file format.

In all these cases, the files can be stored in any location either in a local pc, local network, or internet. The file hence should be qualified through the folder location as below:

#### Current directory:

If the referenced file is in the same folder as the html, then merely specifying the file name is sufficient. Example: **href="myFile.css"**.

#### Relative directory:

If the referenced file is in a sub folder to where the html is stored, then the file can be specified relative to the current folder. Example: **href=".\\mySubFolder\\myFile.css"**. Please note that the dot (.) refers to current directory.

Likewise, if the file is in a parent folder to where the html is stored, then the file can be specified relative to the current folder. Example: **href="..\\myFile.css"**. Please note that the double dot (..) refers to parent directory.

#### Absolute directory:

If the referenced file is in any folder in the local or network (mapped) folder then the absolute address as seen by the operating system should be specified. Example: **href="C:\\myFolder\\myFile.css"**. You should specify the drive letter and folder.

#### Internet stored:

If the referenced file is accessible over the internet then the url has to be specified. Example: **href="http://www.MyServer.com/myFolder/myFile.css"**. You should specify the http:// as prefix for the html to access the file over internet.

## Practice 04

### Styling your Profile Page.

Starting from the Question 2 of Task 03 where you created your Profile Page do the following enhancements:

- You should now create two CSS files to set the themes (i.e., colour, fonts etc) as per your choosing.
- You should link the CSS page successively to review and report how your page gets displayed.



## Lesson 05: Forms in HTML

A form is simply an area in the form (or the entire page itself) that can contain form fields.

Form fields are objects that allow the visitor to enter information - for example text boxes, drop-down list or radio buttons. When the visitor clicks a submit button, the content of the form is usually sent to a program that runs on the server. However, there are exceptions.

JavaScript is sometimes used in conjunction with form fields. An example could be setting the value in a form field when a choice is made from a drop down list or checking that the value entered in a text box conforms to a set of validations.

### How does an HTML form work?

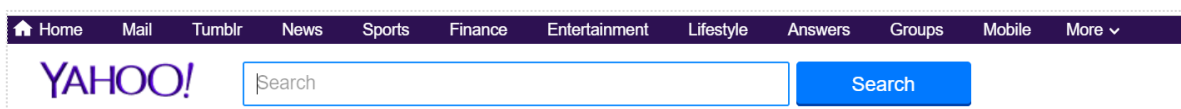
A web form consists two parts - the HTML 'front end' and a back end form processor. The HTML front end part handles the presentation while the back end handles the form submissions. The back end of the form is usually written in languages like PHP or ASP. Here is how a form normally works, step-by-step

A user visits your web page which contains a form.

- The users' web browser displays the HTML form.
- The user fills out the form and clicks submits
- The browser sends the submitted form data to the web server
- A form processor script (also known as a formmail) running on the web server processes the form data
- A response page is sent back to the browser.

### Form Examples:

One of the classic example of a form that we use virtually in day to day activity is the search form such as in Bing, Yahoo or Google. The search form typically consists a Text Box and a Button as shown in figure below.



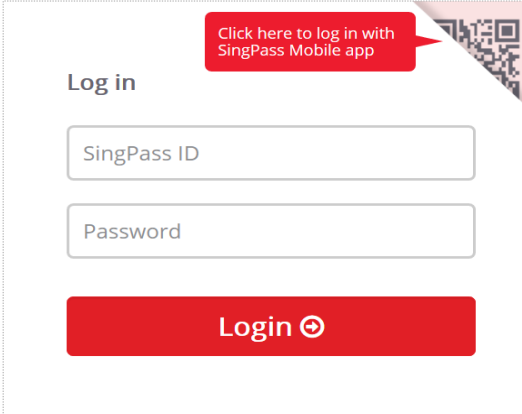
Users can enter a set of words in the text box marked 'Search' and press the **Search** button. This is what happens when the user hits the Search button:

- The search words are sent to a program on the server (*i.e., the server hosting www.yahoo.com in this case*).

- The program (at the server) will search a database for matches.
- The program creates a webpage with the results.
- The results webpage is sent back to the browser which the user views. *Typically, the server sends a HTML content to the client browser to display a formatted page.*

Another example is the login facility in any email like Hotmail or Gmail or your own university. Apart from email, the login form is used for accessing any application such as banks or your corporate services. The login form consists (apart from other information) a set of two text boxes one for name and another for password along with a Submit button. It may optionally contain a check box say to remember you for subsequent visits.

A typical login page to log into a Singapore Government portal using Singpass is shown below.



The image shows a SingPass login form. At the top right, there is a red callout box with a QR code and the text "Click here to log in with SingPass Mobile app". Below this, the text "Log in" is displayed. There are two input fields: "SingPass ID" and "Password". Below these fields is a red "Login" button with a circular arrow icon.

This is what happens when the user enters the ID and Password and presses the Login button.

- The ID and password are sent to a program on the server.
- The program will search a database for valid entries.
- If the entry is valid the visitor is sent to the access protected page.
- If the entry is invalid the visitor is sent to a "failure" page.

## Constructing a Form in HTML

The `<form>` tag is used to create the actual form – it looks like this:

```
<form>

    Input elements

</form>
```

The `<form>` tag contains the content of your form. This content is called form widgets, controls or fields – they all describe the same thing. This content is mostly different kinds (or states, which is actually the correct word to use) of input fields and a button here and there. Let's have a look at a very simple form with just two input fields and a submit-button:

```
<form>

    First name: <input type="text" name="firstname" /> <br />

    Surname: <input type="text" name="surname" /> <br />

    <input type="submit" value="Submit now" />

</form>
```

As you can see, the input fields are defined by the `<input>` element. The input element represents a typed data field, which the user can edit. The type-attribute defines which input state you are using, in this case a regular text element, and the name attribute denominates the individual input fields. The type attribute is the master switch that determines what the input element really is.

The button is also defined using the `<input>` element but the value of the type attribute is "submit" as opposed to "text" in the two previous fields and this indicates that this is a submit button. The submit button has an additional attribute, value. The value of the value attribute defines caption on the button for the example we used "Submit now". As you have probably noted by now, the `<input>` element is an empty element and therefore it is self-closing, using the `/` at the end. *You can use the standard notation `<input> ... </input>` and leave the ... as blank; but is seldom done that way for ease of readability.*

## Input Field Types

As seen the input element is a versatile element and is used for various UI controls for obtaining values from the user. The following table provides a list of input type value:

Value	Description
button	Defines a clickable button (mostly used with a JavaScript to activate a script)
checkbox	Defines a checkbox
file	Defines a file-select field and a "Browse" button (for file uploads)
hidden	Defines a hidden input field
image	Defines an image as the submit button
password	Defines a password field
radio	Defines a radio button
reset	Defines a reset button
submit	Defines a submit button
text	Default. Defines a single-line text field

The following new type values have been added in HTML 5

Value	Description
color	Defines a color picker
date	Defines a date control (year, month, day (no time))
datetime-local	Defines a date and time control (year, month, day, time (no timezone))
email	Defines a field for an e-mail address
file	Defines a file-select field and a "Browse" button (for file uploads)
month	Defines a month and year control (no timezone)
number	Defines a field for entering a number
range	Defines a range control (like a slider control)
search	Defines a text field for entering a search string
tel	Defines a field for entering a telephone number
time	Defines a control for entering a time (no timezone)
url	Defines a field for entering a URL
week	Defines a week and year control (no timezone)

## Input Field Examples

### Checkboxes

Using checkboxes is a good option when you want to give your visitors the option to choose several items from a group of choices. In that regard, the checkbox works opposite of a radio button, which only allows you to select one item from a group of choices. In its most simple form, a checkbox is simply an input element with the type property set to checkbox, like this:

```
<input type="checkbox">
```

However, as with all input elements, you need to define a name for it to be usable - without a name, the element won't be identifiable when posting the form back to a server for processing. You also want to set a value - this will be the value sent to the server if the checkbox has been checked. Here's an example:

Example:

```
<input type="checkbox" name="nameOfChoice" value="1" checked="checked">
```

Will display:



In the above example, if the checkbox has been checked and the form is submitted to a server, the server will be able to read the form element "nameOfChoice" and its value will be 1.

To provide multiple check boxes along with labelling them can be done as below:

```
Do you like: <br>
<input type="checkbox" name="drinkCB" value="Coffee" checked="checked">Coffee <br>
<input type="checkbox" name="drinkCB" value="Tea">Tea <br>
```

Will display:

Do you like:  
☒ Coffee  
☐ Tea

Noticed that the Coffee is checked as specified in the input element while Tea is not. When the form is submitted the values of the two checkboxes will be sent with the identification drinkCB with appropriate value(s) depending on whether it is checked or not. In the case of check boxes, the user can chose multiple options or none at all.

## Radio Buttons

Radio buttons should be used whenever you want to give your user a selection between two or more options. They look a lot like checkboxes, but instead of allowing zero or several selections within a group of options, a radio button force you to select just one. In its most simple form, a radio button is simply an input element with the type property set to radio, like this:

```
<input type="radio" name="nameOfChoice" value="1" checked="checked">
```

However, as with all input elements, you need to define a name for it to be usable - without a name, the element won't be identifiable when posting the form back to a server for processing.

A typical example will be:

```
Your residence status:<br />
<input type="radio" name="statusRB" value="Citizen" checked="checked">Citizen<br />
<input type="radio" name="statusRB" value="PR">PR<br />
<input type="radio" name="statusRB" value="Foreigner">Foreigner<br />
<br />
Your Gender:<br />
<input type="radio" name="genderRB" value="Male" checked="checked">Male<br />
<input type="radio" name="genderRB" value="Female">Female<br />
```

The above will display as:

Your residence status:

- ☒ Citizen
- ☐ PR
- ☐ Foreigner

Your Gender:

- ☒ Male
- ☐ Female

The salient difference between the checkbox and radio button is that user is able to choose only one of the options. In the above example, we have grouped the two sets of radio buttons through using the same value for the name attribute; hence the user can select one from the residence status and one from the gender.

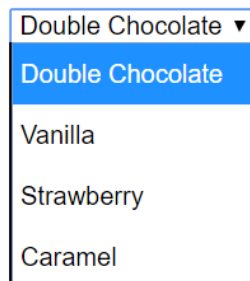
## Dropdown Lists

The drop-down lists is another way to give the user the opportunity to choose just one of a series of choices. When would you use the drop-down list? As the first item on the list is normally the default choice, using the drop-down list can be a good choice when you know that a specific option is often chosen over the other options. The drop-down list should only be used when the user has to choose between the options as the drop-down list does not gives the user opportunity not to choose anything (such as radio-buttons do).

```
<form method="post">
  <select name="Icecream Flavours">
    <option value="double chocolate">Double Chocolate</option>
    <option value="vanilla">Vanilla</option>
    <option value="strawberry">Strawberry</option>
    <option value="caramel">Caramel</option>
  </select>
</form>
```

The drop-down lists are defined by the <select> element and the values being submitted to the server as specified by the value attribute.

The above example will display



If you want another item to be selected instead of the first one on the list, you use the selected attribute.

For all these controls viz., check box, radio button, drop-down list, HTML5 allows you to shorten you mark-up when using the selected attribute, so instead of writing both name and value.

*That is, in HTML5 you can write*

```
<option value="vanilla" selected>Vanilla</option>
```

*instead of:*

```
<option value="vanilla" selected="selected" >Vanilla</option>
```

## Submitting the Data

Until now we have not discussed how to actually send the data entered by the user. You can either send it to another page to your database, or to your email address,

Whichever option you choose, you always have to define how the browser handles the user input. This is defined through the method attribute, which can have one of two values – post or get.

```
<form action="url-to-formmail-provided-by-your-ISP" method="get">  
    .... some fields here  
</form>
```

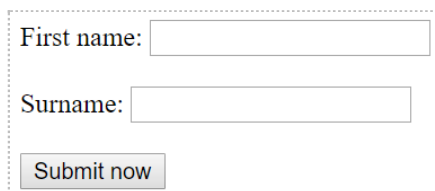
The method attribute value **get** ensures that the data is to be encoded by the browser into a url and **post** means that the data is sent to a url, database or your email. This means, that generally you will use the post value as it is used for form submissions.

The easiest way to demonstrate this to send data to your email. All you must do is add the action attribute to the form element and the method attribute. The action attribute tells the browser what to do with the content of your form and the method attribute tells the browser how to handle it. Here is an example:

```
<form action="mailto:your@email.com" method="post">  
    First name: <input type="text" name="firstname" /> <br />  
    Surname: <input type="text" name="surname" /> <br />  
    <input type="submit" value="Submit now" />  
</form>
```

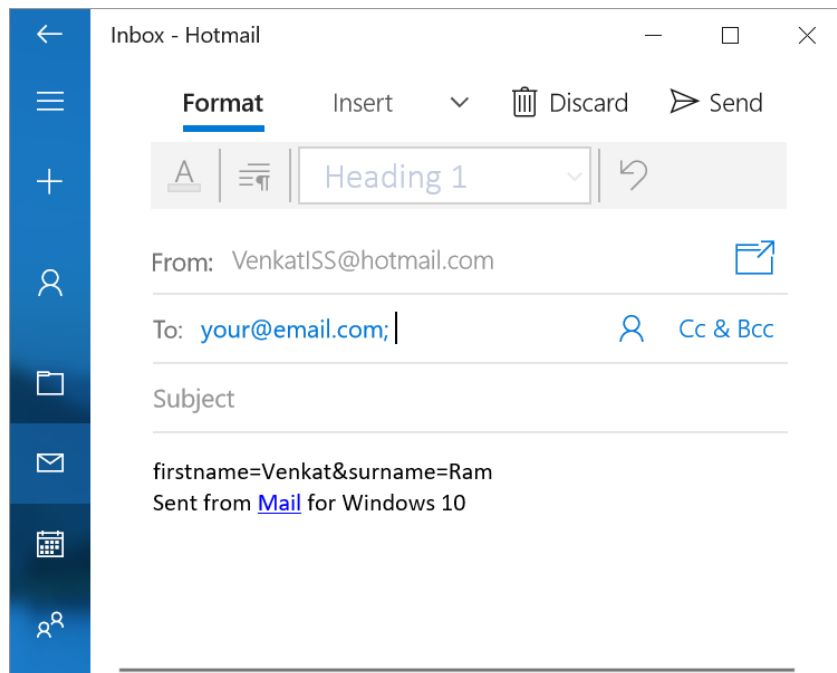
If you choose to use this method you need to be aware that the functionality depends on the email-client installed on your users' computer. This means, that not all users will be able to use your form.

The above example will display as below:





And when you enter Venkat for First name, Ram for Surname and hit the submit button is pressed, the client email application will launch with the following values for To & Body:



---

## Practice 05

You are required to design and implement a form for customers to register for a travel portal. You are required to do the following:

- Design a form identifying the design element for page layout (eg: corporate logo, header and footer information).
- Identify the fields that you require to capture for customers to input.
  - You should design a form which should consist of textboxes, multiple groups of radio buttons, at least one group of checkboxes, text box fields to take in numeric data, text box fields to take in date or other special data, submit button, reset button etc.
- Implement HTML codes for the form above.
  - You should keep in view the aesthetics of the form in terms of layout of the fields and formats.
  - You may be required to use HTML Tables for layout, stylesheets for fonts, formats and colour etc.
- You may test the form with a mail-based submission.
- Your submission should consist of HTML file, CSS file, Output (email window).

---

## Lesson 06: Introduction to JavaScript

### Introduction

JavaScript a programming language was released by Netscape and Sun Microsystems in 1995. However, JavaScript is not the same language as Java.

### What is JavaScript?

- It is a programming language.
- It is an interpreted language.
- It is object-based programming.
- It is widely used and supported
- It is accessible to the beginner
- It was created to provide functionality to the HTML page.

### Uses of JavaScript

- Use it to add multimedia elements  
With JavaScript you can show, hide, change, resize images, and create image rollovers. You can create scrolling text across the status bar.
- Create pages dynamically  
Based on the user's choices, the date, or other external data, JavaScript can produce pages that are customized to the user.
- Interact with the user  
It can do some processing of forms and can validate user input when the user submits the form.

### Writing JavaScript

JavaScript code is typically embedded in the HTML, to be interpreted and run by the client's browser. Here are some tips to remember when writing JavaScript commands.

- JavaScript code is case sensitive
- White space between words and tabs are ignored
- Line breaks are ignored except within a statement
- JavaScript statements end with a semi- colon

## The SCRIPT Tag

The <SCRIPT> tag alerts a browser that JavaScript code follows. It is typically embedded in the HTML.

```
<SCRIPT language="JavaScript">  
    statements  
</SCRIPT>
```

### Example

```
<script language="JavaScript">  
    alert("Welcome to the Java Script tag test page.")  
</script>
```

When you execute the above page, you will notice that a message box gets displayed with the above message. The alert function shows a message on the browser with an ok button. This demonstrates that the script gets executed appropriately executed when the page is launched.

## Implementing JavaScript

There are three ways to add JavaScript commands to your Web Pages.

- Embedding code
- Inline code
- External file

The above three options are very much similar to the way we worked with styles in CSS. In this instance it is programming functionality embedded into the HTML as against formatting information which was the case with CSS.

## Programming Basics

Programs are used to provide functionality such as providing a message to the user, reading the information entered by the user, performing numerical computations etc. Programs are created through a set of statements using the appropriate word and syntax (programming key word, grammar & punctuation). A collection of several such statements collectively result in a program. Each statement in JavaScript ends with a semicolon (;). When a program is executed the statements are executed one after the other top to bottom (and left to right).

## Variables

Variables are names that programmers use to refer to things that they wish to store during the course of a program. Using variables you can store numbers such as Product Price, or text such as Product Names, or dates such as Purchase Date etc.

A variable can hold several types of data. In JavaScript you don't have to declare a variable's data type before using it. Any variable can hold any JavaScript data type, including:

- String data
- Numbers
- Boolean values (T/F)

## Variable declaration

- To declare variables, use the keyword *var* and the variable name: **var username**
- To assign values to variables, add an equal sign and the value:

```
var userName = "Smith"
```

```
var price = 100
```

Incidentally text (or more technically called string) are enclosed inside double quotes, while the numbers are written plainly.

## Variable Names

There are rules and conventions in naming variables in any programming language. It is good practice to use descriptive names for variables. The following are the JavaScript rules:

- The variable name must start with a letter or an underscore.
  - firstName or \_myName
- You can use numbers in a variable name, but not as the first character.
  - name01 or tuition\$
- You can't use space to separate characters.
  - userName not user Name
- Capitalize the first letter of every word except the first
  - salesTax or userFirstNam

## Functions

With functions, you can give a name to a whole block of code, allowing you to reference it from anywhere in your program. JavaScript has built-in functions for several predefined operations. Here are three some functions.

- `alert("message")`
- `confirm("message")`
- `prompt("message")`

## Practice 06

- Write following programs on the notepad and save as html file (eg: MyJS\_Code.html).
- Open the program in a browser and record your observation.

**You should provide your conservation in you a sentence or two for submission.**

**Q1:**

```
<html>
<body>
  <script type="text/javascript">
    document.writeln('Hello World');
  </script>
</body>
</html>
```

**Q2:**

```
<html>
<body>
  <script type="text/javascript">
    alert('Hello World');
  </script>
</body>
</html>
```

**Q3:**

```
<html>
<body>
  <script type="text/javascript">
    document.writeln('Hello World');
    document.writeln('Good bye Folks!');
  </script>
</body>
</html>
```

**Q4:**

```
<html>
<body>
  <script type="text/javascript">
    document.writeln('How <i> are </i> you');
  </script>
</body>
</html>
```

**Choose the most appropriate choice for the following:**

**Q4.** Why so JavaScript and Java have similar name?

- A. JavaScript is a stripped-down version of Java
- B. JavaScript's syntax is loosely based on Java's
- C. They both originated on the island of Java
- D. None of the above

**Q5.** When a user views a page containing a JavaScript program, which machine actually executes the script?

- A. The User's machine running a Web browser
- B. The Web server
- C. A central machine deep within Netscape's corporate offices
- D. None of the above

**Q6.** What are variables used for in JavaScript Programs?

- A. Storing numbers, dates, or other values
- B. Varying randomly
- C. Causing high-school algebra flashbacks
- D. None of the above

**Q7.** What should appear at the very end of your JavaScript?

The `<script LANGUAGE="JavaScript">`tag

- A. The `</script>`
- B. The `<script>`
- C. The END statement
- D. None of the above

**Q8.** Which of the following can't be done with client-side JavaScript?

- A. Validating a form
- B. Sending a form's contents by email
- C. Storing the form's contents to a database file on the server
- D. None of the above

**Q9.** Which of the following are capabilities of functions in JavaScript?

- A. Return a value
- B. Accept parameters and Return a value
- C. Accept parameters
- D. None of the above

Ans: C



**Q10.** Which of the following is not a valid JavaScript variable name?

- A. 2names
- B. \_first\_and\_last\_names
- C. FirstAndLast
- D. None of the above

**Q11.** What is the correct JavaScript syntax to write "Hello World"?

- A. System.out.println("Hello World")
- B. println ("Hello World")
- C. document.write("Hello World")
- D. response.write("Hello World")

**Q12.** Which of the following way can be used to indicate the LANGUAGE attribute?

- A. <LANGUAGE="JavaScriptVersion">
- B. <SCRIPT LANGUAGE="JavaScriptVersion">
- C. <SCRIPT LANGUAGE="JavaScriptVersion"> JavaScript statements...</SCRIPT>
- D. <SCRIPT LANGUAGE="JavaScriptVersion"!> JavaScript statements...</SCRIPT>

**Q13.** Inside which HTML element do we put the JavaScript?

- A. <js>
- B. <scripting>
- C. <script>
- D. <javascript>

**Q14.** What is the correct syntax for referring to an external script called " abc.js"?

- A. <script href=" abc.js">
- B. <script name=" abc.js">
- C. <script src=" abc.js">
- D. None of the above

**Q15.** JavaScript entities start with \_\_\_\_\_ and end with \_\_\_\_\_.

- A. Semicolon, colon
- B. Semicolon, Ampersand
- C. Ampersand, colon
- D. Ampersand, semicolon

**Q16.** JavaScript is designed for following purpose -

- A To Style HTML Pages
- B To add interactivity to HTML Pages.
- C To Perform Server Side Scripting Opertion
- D To Execute Query Related to DB on Server

## Lesson 07: JavaScript Programming Constructs

### Introduction

This lesson deals with programming constructs.

For those who have prior programming this lesson will help as a reference to orient towards JavaScript syntax. They should be able to attempt the full range of tasks.

For those without programming background may find the learning steep. They may read through this lesson and try to grasp the rudimentary programming aspects and attempt the simpler exercises/tasks. There is no compulsion to gain mastery on this lesson as many of the programming concepts will be discussed/taught in the programming fundamentals unit and once they learn the programming concepts, they may revisit this lesson to orient towards JavaScript based programming for a later unit in the course.

### Online Learning

In addition to reading the below the students may refer to online tutorials or books mentioned at the start of this guide.

For this chapter useful online free learning environments (apart from books and notes) are:

<http://www.w3schools.com>

<https://javascript.info>

To keep your learning consistent to the above, if you choose to use the above, we align the examples and write up with adaptations and simplifications in this lesson.

#### IMPORTANT

JavaScript is case sensitive. So, you should make sure your statements are written with the correct capitalisation; else the program will not work

### Programming Concepts

#### Operators

Many **operators** are known to us from math class at primary school. They are addition +, a multiplication \*, a subtraction - and so on. In this lesson we concentrate on aspects that are not covered by school arithmetic.

An **operand** is what operators are applied to. For instance in multiplication  $5 * 2$  there are two operands: the left operand is 5, and the right operand is 2.

An expression is the phrase that consists of operators and operands as shown earlier eg:  $5 * 2$ .

The + operator sums up the values of two numeric operands.

In a JavaScript program, expressions may be used with appropriate syntax as shown below:

```
alert(5 + 3);  
var result = 5 + 4;  
document.writeln(result);
```

Likewise, the \* operator multiplies the operands, the - operator subtracts the second operand from the first while the / operator divides the first operand with the second operand.

In the second line we used **var** before the variable name result. The **var** is used to declare the variable and the data type of the variable (**result**) is determined by the righthand side value. In this instance, it is a **numeric** (*Integer* to be more specific).

For instance, if we had used the statement below, the variable answer will take a *String* (text) value:

```
var answer = 'Venkat'
```

## Comparisons

Many comparison operators we know from maths:

- Greater/less than: **a > b**, **a < b**.
- Greater/less than or equals: **a >= b**, **a <= b**.
- Equality check is written as **a == b** (please note the double equation sign =. A single symbol **a = b** would mean an assignment).
- Not equals. in JavaScript it's written as an assignment with an exclamation sign before it: **a != b**.

The outcome of a comparison is a **Boolean** i.e, an true or false.

true – means “yes”, “correct” or “the truth”.

false – means “no”, “wrong” or “a lie”.

For example:

```
alert( 2 > 1 );      is true (correct)  
alert( 2 == 1 );     is false (wrong)  
alert( 2 != 1 );     is true (correct)
```

A comparison result can be assigned to a variable, just like any value:

```
var result = 5 > 4;    // assign the result of the comparison
alert( result );      // this will display true in a pop up box
```

### String comparison

To see which string is greater than the other, the so-called “dictionary” or “lexicographical” order is used. In other words, strings are compared letter-by-letter.

For example:

```
alert( 'Z' > 'A' );           is true
alert( 'His' > 'Him' );       is true, because although Hi is same s is > than m
```

### Conditional operators: if

Sometimes we need to perform different actions based on a condition.

There is achieved by using the **if** statement along with condition (usually using the comparison operator) that yields a true or false.

Hence you can perform something if the condition is true and something else (or nothing) if condition is false.

For example:

```
var answer = prompt('What is the 5 * 3);
if (answer == 15)
{
    alert( 'You are right!' );
}
alert('End of program')
```

In the above example the program will ask for an value from the user, and if the answer is 15, then will message “You are right!” and then inform “End of program”. If the user has entered any other value other than 15, (say 23), then the program will skip the if block {} and just say ‘End of program.

In the same example we write this:

```
var answer = prompt('What is the 5 * 3');
if (answer == 15)
{
    alert( 'You are right!' );
}
else
{
    alert( 'You are wrong!' );
}
alert('End of program')
```

In this example the program will ask for an value from the user, and if the answer is 15, then will message “You are right!” and then inform “End of program”. If the user has entered any other value other than 15, (say 23), then the program will skip the if block {} but execute the else block {} hence will say “You are wrong!” and state “End of program”.

Hence unlike in the past examples where all statements were sequentially executed we are now able to selectively execute the statements by avoiding certain lines of codes depending on the context through the use of the *if ... else* statements.

## Logical operators

There are three logical operators in JavaScript: || (OR), && (AND), ! (NOT).

They are usually applied to two comparison operations.

### Example for OR

```
var hour = 9;
if (hour < 10 || hour > 18)
{
    alert( 'The office is closed.' );
}
```

In the case of OR statement if one of the operands (i.e., left or right of the || is True, the result is true. The result is false only if both the operands are false, the result is false. In the above example, 9<10 is true while 9>18 is false. Since at least one is true the result is true hence the alert message is displayed.

## Example for AND

```
var hour = 9 ;  
if (hour >= 10 && hour <= 18)  
{  
    alert( 'The office is open.' );  
}
```

In the case of AND statement if one of the operands (i.e., left or right of the && is False, the result is False. The result is True only if both the operands are True.. In the above example,  $9 \geq 10$  is false while  $9 \leq 18$  is true. Since at least one is false the result is false hence the alert message is not displayed. On the other hand if hour was specified as 12, you will notice that both comparisons are true and in that event the alert message will be displayed.

## Loop Constructs

We often have a need to perform similar actions many times in a repeatedly.

For example, when we need to output goods from a list one after another. Or just run the same code for each number from 1 to 10.

Loops are a way to repeat the same part of code multiple times.

### WHILE Loop

The while loop has the following syntax:

```
while (condition) {  
    // code  
    // i.e., "loop body"  
}
```

The loop body is executed so long as the while condition is true.

Example, the loop below outputs the value that the user enters till such time he enters QUIT.

```
var someText = "dummy"  
while (someText != "QUIT") {  
    someText = prompt("Enter some text; type QUIT to stop asking");  
    document.writeln("You entered: " + someText;  
}
```

If the user enters QUIT the program ends, else it will mention what he typed and continue prompting.

## FOR Loop

For loop is also called loops with counters. It is used in situations where you know that you need to execute a block of statements (body) **n** number of times.

The structure is like this:

```
for ( initial; condition; step )  
{  
    // ... loop body ...  
}
```

The meaning of these stages (initial, condition, step) are better understood through an example.

Example: The loop below runs alert(i) for i from 0 up to (but not including) 3:

```
for (var i=0; i < 3; i++)  
{  
    alert (i);  
}
```

Let's examine the for statement stage by stage:

Section	Statement	Action
initial	i = 0	Executes once upon entering the loop.
condition	i < 3	Checked before every loop iteration, if fails the loop stops.
step	i++	Executes after the body on each iteration, but before the condition check.
body	alert(i)	Runs again and again while the condition is truthy

The i++ operation (not discussed earlier) increments the value of i. That is if the value of i is currently 5, its value becomes 6 after the statement.

It is easily seen that the loop executes with initial value i value of 0, then messages the i value and since initially i is zero the body executes then the value is increased from zero to one and the condition i<0 is still true so the body executes again. But when value of i becomes three, the condition fails and loop terminates to proceed with rest of the code.

When the program is run, the loop executes three times and displays the values 0, 1 and 2 in each iteration.



## Functions

Quite often we need to perform a similar action in many places of the script.

For example, we need to show a confirmation message when a user performs some action say saving data or submitting form or at other key places. One way to do this is to write the programming statements every time the message has to be sent. This is ok if it is just a line. But if the activity of displaying information is lengthy with several statements, then repeating the same code everywhere becomes cumbersome and causes issues of code maintenance.

In such instances, we create functions to perform an action which can be reused several times by ‘calling’ the function in a single word to execute its multiple lines rather than writing multiple lines each time.

Functions are the main “building blocks” of the program. They allow the code to be called many times without repetition.

We’ve already seen examples of built-in functions, like `alert(message)`, `prompt(message, default)`. But we can create functions of our own as well.

### Define Function

A function can be defined using the following syntax:

```
function functionName()  
{  
    // some JavaScript code  
}
```

To declare your own function, you should use the key word **function** followed by a *name* of your choosing to refer the function. The function name is followed by simple parenthesis **()** which may optionally hold some variables called arguments.

#### Example:

```
function showMessage()  
{  
    alert('Hello Folks! ');  
}
```

In the above example, whenever this is **called** (aka invoked) the message “Hello Folks!” is displayed.

## Invoking the Function

A function can be invoked from another function or program. It can also be invoked from an HTML code either at the script level or when user performs an action such as clicking a button.

```
function showMessage()
{
    document.writeln ('Hello Folks! ');
}

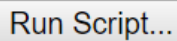
function myProgram()
{
    document.writeln('Program starts... we will call the function we created');
    showMessage();
    document.writeln ('We have called the function first time');
    document.writeln ();
    document.writeln ('We have called the function a second time; And, program ends');
}
```

In the above example if we run the program myProgram, this will call the showMessage() function twice and each time the message “Hello Folks” gets displayed.

To test the above program, you can invoke it from the HTML code. Note that JavaScript should be placed inside the **<script>...</script>** of the header section of the HTML. Typically, the program is called on a button click as per example below:

```
<html>
<head>
  <script>
    function showMessage() {
      alert('Hello Folks!');
    }
    function myProgram() {
      document.writeln('Program starts... we will call the function we created');
      showMessage();
      document.writeln ('We have called the function first time');
      showMessage();
      document.writeln ('We have called the function a second time; And, program ends');
    }
  </script>
</head>
<body>
  <input type="submit" value="Run Script..." onclick="myProgram()" />
</body>
</html>
```

When you launch the html the following screen will be displayed i.e., the input tag representing a button.



To run the JavaScript you can **click** the button.

The entire output will look as below:

```
Program starts... we will call the function we created  
  
Hello Folks!  
  
We have called the function first time  
  
Hello Folks!  
  
We have called the function a second time; And, program ends
```

Notice that the click of button calls myProgram() function; while the myProgram() function in turn calls the showMessage() function twice as instructed.

## Functions with Arguments

Functions can have arguments or parameters which can be supplied from another program. Let us take the example of a function that prints the square of a number supplied to it:

```
function squareIt(var x)  
{  
    document.writeln ( x * x );  
}
```

To test the program you can write a button with the following:

```
<input type="submit" value="Run Script..." onclick="squareIt(3)" />
```

When calling the function we supplied the value 3 in brackets called the argument. This value is assigned to the variable x in the function and the function prints 9 i.e., 3 x 3. We can use this function any number of times with different values as per our needs. For instance in another place if we had used squareIt(5) the program will print 25.

The example below shows calling the squareIt() from another program passing a variable rather than a number. The variable of course should contain a numeric value.

```
function myProgram()
{
    var n = prompt("Enter a number");
    squareIt(n);
}
```

In this program if the user enters 4, then the output on the page will be 15.

## Functions that return a value

In the previous example of squareIt, the function displayed the square of a number supplied. It is possible for the function to return back a value to the calling program. The calling program can then use the value returned to do whatever it intends. For instance if your program should compute the output of an expression  $5y^2$ .

The program should first get the value of  $y$  from the user then need to compute  $y^2$  possibly using the squareIt function and finally multiply the square of  $y$  with 5. In order to do this, we need the squareIt function to return the square of the number rather than print it out as per previous example. This can be accomplished by the following code:

```
function squareIt(var x)
{
    var sq = x * x ;
    return sq;
}
```

Please note that this function we do not print the value but use the **return** key word to send back the value (i.e., square of  $x$ ).

To now do our function to compute  $5y^2$  the following function is written:

```
function myProgram()
{
    var n = prompt('Enter a number');
    var result = 5 * squareIt(n);
    document.writeln(result);
}
```

This program will output 80 if the user enters 4 when prompted.

## Accessing HTML fields from JavaScript

While JavaScript is a programming language and the examples given above introduces you to the programming aspects, it is important in the context of web development we demonstrate how JavaScript can be integrated to web pages.

We have already seen how to run a JavaScript through button click. More importantly, JavaScripts read or write information to HTML page contents. This section describes this with the aid of a simple example.

Assume you have a page as below:

Enter a value:

The result is:

There are two textboxes and one button. Recall all the three are `<input>` element; two with type text and one with type submit.

The expected behaviour of this screen is that user should enter a number in the first text box and upon clicking the Compute button the square of the number should be displayed in the second text box.

We have done a program like that before, but we used prompt box for input and alert or `document.writeln` for displaying the result. This time we need to use the textboxes for input and display of result. So our JavaScript program should be able to read and write to textboxes.

This can be done by the following statement:

```
document.getElementById("someFieldName").value
```

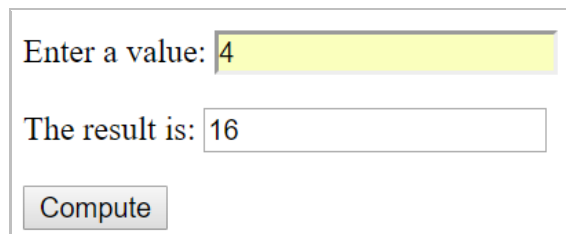
where **someFieldName** is a name that you have given to a HTML field (text box in our example). Hence to read the value of the text box we should provide an id for the control. This can be done by providing the attribute **id="someName"** in the input tag. It is similar to the way we provided the name for checkboxes.

Hence to read from a textbox and assign to a variable in JavaScript the above statement is written on the left side of an assignment statement, while to write value to the textbox the same is written on the right side of the assignment statement.

The following HTML code provides a full implementation of the page example that we have described above.

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myNewProgram()
      {
        var n = document.getElementById("inputTextBox").value;
        var result = n * n;
        document.getElementById("outputTextBox").value = result;
      }
    </script>
  </head>
  <body>
    <p> Enter a value: <input type="text" id="inputTextBox" /> </p>
    <p> The result is: <input type="text" id="outputTextBox" /> </p>
    <p> <input type="submit" value="Compute" onclick="myNewProgram()" /> </p>
  </body>
</html>
```

Upon running the program the HTML page as provided earlier is displayed. If the user enters a value 4 in the first text box and clicks the Compute button, the following output is seen (i.e., the square of 4, i.e., 16 is displayed in the second text box.



Enter a value: 4

The result is: 16

Compute

## Practice 07

Write JavaScript functions for the following:

1. Write a JavaScript that would prompt the user for a number and compute and print the value of  $3x^3 - 7x^2 + 2x + 5$ .
2. Write a JavaScript that would prompt the user for the time of the day and if the time entered by the user is  $< 12$ , the program would display 'Good Morning', else will display 'Good Afternoon'.
3. Modify the question 2 to prompt the user for his name and print 'Good Morning X' or 'Good Afternoon X', where X is the name that the user entered.
4. Write a program that would use the for loop compute the sum of numbers from 1 to n.

Write JavaScript along with HTML codes for the following:

5. Write a JavaScript function to generate the following HTML table that will give the a calculation reference table.

The table should be generated for all values of x from 1 to 5:

x	$x^2$	$\frac{1}{x}$	$\sqrt{x}$
1	1	1	1
2	4	0.5	1.414
3	9	0.333	1.732
4	16	0.25	2
5	25	0.2	2.236

Hint:

- You should use a **for loop** and within the for loop programmatically construct **<tr>** and **<td>** tags and place the values of x,  $x^2$ ,  $\frac{1}{x}$  and  $\sqrt{x}$ .
- You should use floating point values for x; that is, x should have values 1.0 rather than 1; 2.0 instead of 2 and so on.

## 6. Create the following screen in HTML

Name:

Age:

Gender

☐ Male ☐ Female

Phone:

Hobbies

☐ Reading ☐ Sports ☐ Movies ☐ Travelling

Submit

Output:

Errors:

Write JavaScript codes to perform the following validation upon clicking the Submit button. If there are errors, the errors should be displayed in the box below with appropriate message. The validations to be performed are the following:

- Name should not be blank.
- Name should be no more than 15 letters long.
- Age should be numeric.
- Age should not be negative number.
- Gender must be selected (cannot leave both blank).

If there are no errors, then the data entered by the user should be displayed in the first box.

Hint: Create the page layout using HTML. Write JavaScript functions for the various validation tasks. Write a JavaScript function to output the values or errors as the case may be.



## Lesson 08 Web Application Development

### Aim

This lesson aims to provide opportunity for the learners to explore further and prepare themselves towards web application development. The last lesson introduced JavaScript with a task to validate data entered through an HTML form. In this lesson we explore further options that a typical web interface may have.

### Menu

We had already introduced menus as part of page layout. In that instances we had provided a section on the left of the screen to provide navigation to other pages through menu. Such a simple menu can be implemented by a list control with each list item hyperlinking to an appropriate page.

Explore the following code for implementing such a menu:

```
<body>
  <ul style="list-style-type: none">
    <li><a href="http://www.nus.edu.sg">NUS</a></li>
    <li><a href="http://www.iss.nus.edu.sg">ISS</a></li>
    <li><a href="http://www.office.com">Office Mail</a></li>
  </ul>
</body>
```

Running the above code will yield the following page:



Clicking the links will navigate your page to the appropriate URL. Note that we have styled the list to override the bullets.

## Menu Bar

A better option to the left menu is the top menu with dynamism built into it. Ideally coupled with dynamic drop down the menu at the top can get very rich. It is best accomplished in combination with CSS which hides and displays the drop down menu. It consists of HTML page with list as described above. In order to get it into a menubar style, we use CSS which will be subsequently included:

### MenuSample.html

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="MenuStyles.css"/>
  <title>CSS Menu Maker</title>
</head>
<body>
  <div id='cssmenu'>
    <ul>
      <li><a href='#'><span>Home</span></a></li>
      <li class='active has-sub'>
        <a href='#'><span>Products</span></a>
        <ul>
          <li class='has-sub'>
            <a href='#'><span>Product 1</span></a>
            <ul>
              <li><a href='#'><span>Sub Product 1.1</span></a></li>
              <li class='last'><a href='#'><span>Sub Product 1.2</span></a></li>
            </ul>
          </li>
          <li class='has-sub'>
            <a href='#'><span>Product 2</span></a>
            <ul>
              <li><a href='#'><span>Sub Product 2.1</span></a></li>
              <li class='last'><a href='#'><span>Sub Product 2.2</span></a></li>
            </ul>
          </li>
        </ul>
      </li>
      <li><a href='#'><span>About</span></a></li>
      <li class='last'><a href='#'><span>Contact</span></a></li>
    </ul>
  </div>
</body>
</html>
```

Note in the above we have used the list presented earlier, but this time we use nested lists to present a multi-level menu (i.e, menu, sub-menu and sub-sub-menu). We have left out the HREF in the above sample which can be supplied in real projects that you may wish to try.

At this point assume that you have not linked or created the CSS file mentioned in the <link> element. If you run the code, you will see the below output.

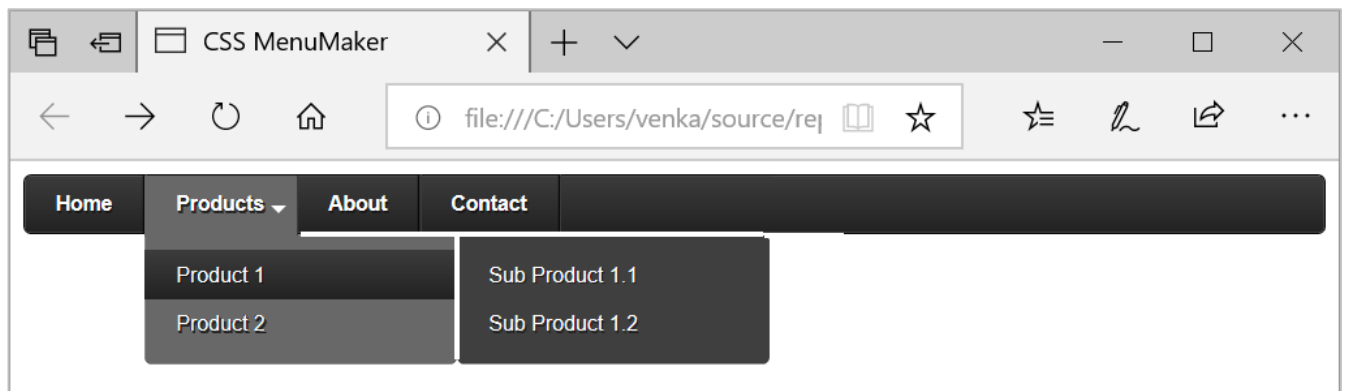


While the Home, About and Contact do not have sub menus, the Products menu has two level of sub menus below it (as labelled with appropriate index).

To get the menu bar presentation and to have a drop down menu, we now resort to CSS. The CSS below provides a horizontal layout and displays and hides list items of the sub menu based on mouse hovering on the menu item.

You may read and understand the CSS code below to learn how this works, but that is not essential. These are like frameworks and ideally unless you are a utility developer, expertise on these are not required. As a application developer you should be able to interpret these codes and edit the open source frameworks for your use. For instance, you may wish to change some color scheme or fonts, but not recreate existing frameworks.

Upon executing the above HTML, with linking to CSS provided overleaf you will be able to present the menu bar structure as below:



## MenuStyles.css

```
#cssmenu,
#cssmenu ul,
#cssmenu li,
#cssmenu a {
    border: none;
    line-height: 1;
    margin: 0;
    padding: 0; }

#cssmenu {
    height: 37px;
    display: block;
    border: 1px solid;
    border-radius: 5px;
    width: auto;
    border-color: #080808;
    margin: 0;
    padding: 0; }

#cssmenu > ul {
    list-style: inside none;
    margin: 0;
    padding: 0; }

#cssmenu > ul > li {
    list-style: inside none;
    float: left;
    display: inline-block;
    position: relative;
    margin: 0;
    padding: 0; }

#cssmenu.align-center > ul {
    text-align: center; }

#cssmenu.align-center > ul > li {
    float: none;
    margin-left: -3px; }

#cssmenu.align-center ul ul {
    text-align: left;
}

#cssmenu.align-center > ul > li:first-child > a {
    border-radius: 0;
}

#cssmenu > ul > li > a {
    outline: none;
    display: block;
    position: relative;
    text-align: center;
    text-decoration: none;
    text-shadow: 1px 1px 0 rgba(0, 0, 0, 0.4);
```

```
font-weight: 700;
font-size: 13px;
font-family: Arial, Helvetica, sans-serif;
border-right: 1px solid #080808;
color: #ffffff;
padding: 12px 20px; }

#cssmenu > ul > li:first-child > a {
  border-radius: 5px 0 0 5px; }

#cssmenu > ul > li > a:after {
  content: "";
  position: absolute;
  border-right: 1px solid;
  top: -1px;
  bottom: -1px;
  right: -2px;
  z-index: 99;
  border-color: #3c3c3c; }

#cssmenu ul li.has-sub:hover > a:after {
  top: 0;
  bottom: 0; }

#cssmenu > ul > li.has-sub > a:before {
  content: "";
  position: absolute;
  top: 18px;
  right: 6px;
  border: 5px solid transparent;
  border-top: 5px solid #ffffff; }

#cssmenu > ul > li.has-sub:hover > a:before {
  top: 19px; }

#cssmenu > ul > li.has-sub:hover > a {
  padding-bottom: 14px;
  z-index: 999;
  border-color: #3f3f3f; }

#cssmenu ul li.has-sub:hover > ul,
#cssmenu ul li.has-sub:hover > div {
  display: block; }

#cssmenu > ul > li.has-sub > a:hover,
#cssmenu > ul > li.has-sub:hover > a {
  background: #3f3f3f;
  border-color: #3f3f3f; }

#cssmenu ul li > ul,
#cssmenu ul li > div {
  display: none;
  width: auto;
  position: absolute;
  top: 38px;
  background: #3f3f3f;
```

```

border-radius: 0 0 5px 5px;
z-index: 999;
padding: 10px 0; }

#cssmenu ul li > ul {
  width: 200px; }

#cssmenu ul ul ul {
  position: absolute; }

#cssmenu ul ul li:hover > ul {
  left: 100%;
  top: -10px;
  border-radius: 5px; }

#cssmenu ul li > ul li {
  display: block;
  list-style: inside none;
  position: relative;
  margin: 0;
  padding: 0; }

#cssmenu ul li > ul li a {
  outline: none;
  display: block;
  position: relative;
  font: 10pt Arial, Helvetica, sans-serif;
  color: #ffffff;
  text-decoration: none;
  text-shadow: 1px 1px 0 rgba(0, 0, 0, 0.5);
  margin: 0;
  padding: 8px 20px; }

#cssmenu,
#cssmenu ul ul > li:hover > a,
#cssmenu ul ul li a:hover {
  background: #3c3c3c;
  background: -moz-linear-gradient(top, #3c3c3c 0%, #222222 100%);
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #3c3c3c), color-
stop(100%, #222222));
  background: -webkit-linear-gradient(top, #3c3c3c 0%, #222222 100%);
  background: -o-linear-gradient(top, #3c3c3c 0%, #222222 100%);
  background: -ms-linear-gradient(top, #3c3c3c 0%, #222222 100%);
  background: linear-gradient(top, #3c3c3c 0%, #222222 100%); }

#cssmenu > ul > li > a:hover {
  background: #080808;
  color: #ffffff; }

#cssmenu ul ul a:hover {
  color: #ffffff; }

#cssmenu > ul > li.has-sub > a:hover:before {
  border-top: 5px solid #ffffff; }

```

## Practice 08

Attempt to create a shopping cart page (refer sample below, but you can choose to use your own product and design concept).

The page should contain features such as header, menu, cart items with images. Should have buttons for Add to cart, Remove from cart, Check out etc.

While adding functionality is optional, learners may do well to refer to some standard shopping card JavaScript code abundantly available over the internet to understand the programming elements associated with the such web based application (to be taught as part of the main course)

The deliverables for this assignment are HTML and CSS source files.

