Maharaja Education Trust ®
MIT First Grade College
(Affiliated to University of Mysore)
Industrial Suburb, Manandavadi Road, Mysuru -570008

HTML, CSS, JavaScript

Table of contents

Chapter 1

HTML

1.1 Introduction

In this chapter, various component of HTML are discussed to design a web page.

The basic structure for an HTML page is shown below.

- Entries inside the /< ... /> are known as tags. Most of the tags has an opening and closing e.g. <head> (opening head) and </head> (closing head). Some of the tags do not have closing tags e.g. <!DOCTYPE ...> and
. We need to write the HTML codes inside the tags.
- The comments are written between '<!–' and '–>'.
- Here Line 1 gives the details of the 'HTML version' to the web-browser. The 'html' tells it is version 5.
- The 'head' tag (Lines 3-5) contains the header related tags e.g. 'title for the page' and 'links for the css files' etc.
- The 'body' tag (7-11) contains the actual HTML code which is displayed on the web-browser. Also, we add all the JavaScript related codes just before the closing body tag (</body>).

```
<!DOCTYPE html> <!-- tells browser above the html version --> <html> <!--
beginning of the html document --> <head>
                <!-- header related tags e.g. title, links etc. -->
    </head>

    <body>
                <!-- actual html document here -->

                <!-- add JavaScript files here --> </body>
</html>
```

1.2 First code

In below code, the message "Hello World" is displayed on the HTML page. The Fig. 1.1 is the resultant HTML page.

- The title (Line 4) appears on the top of the browser.
- The tag <h1> is called 'header' tag, which has the larger size than the normal text (see the size of 'Hello World!').
- The tag <p> is called the 'paragraph' tag, which can be used to write the paragraphs.

```
<!DOCTYPE html> <html>
```

```
<head>
    <title>HTML Tutorial</title>
</head>
<body>
    <h1> Hello World! </h1>
    <p> This is the first HTML code </p>
</body>
</html>
```

Fig. 1.1: First code

1.3 Basic tags

- The Table 1.1 shows the list of tags which are required for writing the basic 'HTML' codes i.e. without any style e.g. bold, italics and numbering etc.

Table 1.1: List of basic tags

Tag	Description	Example
h1, ..., h6	Header tag h1 to h6	Hi
p	paragraphs (Line changes at the end)	Hi
span	No line change after span	Hi Bye.
div	make division between contents	...
a	hyperlink	see Section 1.9
center	Move content to center	Hi
br	Line break (no closing tag)	 or
hr	horizontal line (no closing tag)	<hr /> or <hr>
pre	preserve formatting	<pre> .... </pre>
table	insert table	see Section 1.5

- Let's see the example of each of these tags,

Note: All the new codes are added below the previous codes in the 'body' tag. Therefore only newly added codes are shown in the tutorial.

```
<h2> Heading 2 </h2>
    <h6> Heading 6 </h6>
```

1.3. Basic tags

```
    <p> This is paragraph </p>

    <span> This is span.</span>
    <span> The 'br' tag is used after span to break the line </span> <br/>

    <div style="color:blue;">
        The 'div' tag can be used for formatting the tags inside it at once using 'style' and 'classes' ␣
↪etc.

        <p> This paragraph is inside the 'div' tag </p>
        <span> This span is inside the 'div' tag </span> <br/>

    </div>

    <center>
        <h3> Heading 3 is centered</h3>
        <p><span> Centered span inside the paragraph.</span><p>
    </center>

    Two horizontal line is drawn using two 'hr' tag.
    <hr />
    <hr>


    <pre> 'pre' tag preserve the formatting (good for writing codes)

            # Python code
            x = 2 y
            = 3
            print(x+y)

    </pre>
```

- Fig. 1.2 is the output of above code. Read the text to understand each tag,

Fig. 1.2: Basic tags : Attribute ‘style’ is used in ‘div’ tag

1.3. Basic tags

1.4 Attributes

In Fig. 1.2, we saw an example of attribute (i.e. style) which changed the color of all the elements to ‘blue’ inside the ‘div’ tag.

1.4.1 Attribute ‘name’ and ‘value’

- Attribute is defined inside the opening part of a ‘tag’. For example, in the below code, the attribute ‘style’ is defined inside the ‘div’ tag.

```
<div style="color:blue;">

</div>
```

- An attribute has two parts i.e. ‘name’ and ‘value’. For example, in the above code, name and value of the attribute are ‘style’ and ‘blue’ respectively.

1.4.2 Core attributes

Below are the three core attributes which are used frequently in web design.

- id : The ‘id’ is the unique name which can be given to any tag. This is very useful in distinguishing the element with other elements.

```
<p id='para1'> This is paragraph with id 'para1' </p>
<p id='para2'> This is paragraph with id 'para2' </p>
```

- class : The attribute ‘class’ can be used with multiple tags. This is very useful in making groups in HTML design.

```
<p class="c_blue"> This is paragraph with class 'blue'</p>
```

```
<span class="c_blue"> This is span with class 'blue'</span>
```

- style : We already see the example of style attribute, which can be used to change the formatting of the text in HTML design. We can specify various styles which are discussed in Chapter 2.

```
<p style="font-weight:bold; color:red;">Style attribute is used to bold and color</p>
```

Note: Above three attributes are used with 'CSS (cascading style sheet)' and JavaScript/jQuery, which are the very handy tools to enhance the look and functionalities of the web-page respectively. The CSS is discussed in Chapter 2, whereas JavaScript and jQuery are discussed in Chapter 4 and Chapter 5 respectively.

- Also we can define multiple attributes for one tag as shown below,

```
<p class="my_class" id="para_with_class" style="color:green"> Multiple attributes </p>
```

- The other useful attributes are listed in Table 1.2

Table 1.2: List of attributes

Name	Values	Description
id	user defined names	Hi
class	user defined names	Hi
style	CSS styles	Hi
align	left, right, center	horizontal alignment
width	numeric value or % value	width of images and tables etc.
height	numeric value	height of images and tables etc.

1.4. Attributes

1.5 Tables

In this section, we will learn to draw tables along with some attributes which are discussed in Table 1.2. Table 1.3 shows the list of tags available to create the table, which are used in Listing 1.1.

Table 1.3: Tags and attributes for creating tables

Tag	Description
table	beginning and end of table
tr	row of table
th	header cell
td	data cell
Attributes	
rowspan	number of rows to merge
colspan	number of columns to merge
border	width of border
cellpadding	width of whitespace between two border
cellspacing	width of whitespace within a border
bgcolor	background color
bordercolor	color of border
width	width of table (numeric or %)
height	height of table (numeric)
caption	caption for table

- Some of the attributes of Table 1.3 are used in below example,

Listing 1.1: Table with border and color

```
<!-- border-color, width and height -->
<table border="1" bordercolor="black" width="450" height="100">
<caption>Table 1 : Various tags of table</caption> <tr bgcolor="red" > <!-- row --
>
        <th>Column 1</th> <!-- header -->
        <th>Column 2</th>
        <th>Column 3</th>
    </tr>

    <tr bgcolor="cyan"> <!-- background color -->
        <td>Data 1</td> <!-- data -->
        <td>Data 2</td>
        <td>Data 3</td>
    </tr>

    <tr bgcolor="yellow"> <!-- row -->
        <td colspan="2">New Data 1</td> <!-- column span -->
        <td>New Data 2</td> <!-- data --> </tr>
</table>



<!-- width in % -->
<table border="1" bordercolor="black" width="80%" height="100"> <caption>
Table 2 : Width is 80%</caption> <tr bgcolor="red" >
        <th>Column 1</th>
        <th>Column 2</th>
        <th>Column 3</th> </tr>
```

32

1.5. Tables

```
    <tr bgcolor="cyan"> <!-- row -->
        <td>Data 1</td> <!-- data -->
        <td>Data 2</td>
        <td>Data 3</td>
    </tr>

</table>
```

- Fig. 1.3 is the output of above code,

Table 1 : Various tags of table

Column 1	Column 2	Column 3
Data 1	Data 2	Data 3
New Data 1		New Data 2

Table 2 : Width is 80%

Column 1	Column 2	Column 3
Data 1	Data 2	Data 3

Fig. 1.3: Table generated by Table 1.3

1.6 Text formatting

In this section, we will see some of the text formatting options (see Table 1.4) e.g. bold, italic, subscript and strike etc.

Table 1.4: Text formatting

Tag	Description
b	bold
i	italic
u, ins	underline
strike, del	strike
sup	superscript
sub	subscript
big	big size text
small	small size text

- Below are the some of the examples of text formatting, whose results are shown in Fig. 1.4,

```
<!-- Text formatting -->
<p>This is <b>bold</b> text</p>
<p>This is <strike>striked</strike> text</p> <p>This is <sub>subscript</sub> text</p>
```

1.7 Images

Image tag has two important attribues i.e. 'src' and 'alt' as described below,

- src : tells the location of 'image' file e.g. in Line 2 the image 'logo.jpg' will be searched inside the folder 'img'.

1.6. Text formatting

This is **bold** text

This is ~~striked~~ text

This is subscript text

Fig. 1.4: Text formatting

- alt : is the 'alternate text' which is displayed if image is not found. For example, in Line 6, the name of the image is incorrectly written i.e. 'logoa' (instead of 'logo'), therefore the value of 'alt' i.e. 'Missing Logo.jpg' will be displayed as

```
<!-- Images -->
<img src="img/logo.jpg" alt="Logo.jpg" width="20%"/>

<br/> <br/>

<img src="img/logoa.jpg" alt="Missing Logo.jpg" width="20%"/>
```

shown in Fig. 1.5.

Fig. 1.5: Images

Note: We can use other attributes as well e.g. ‘height’, ‘align’ and ‘border’ etc.

1.7. Images

1.8 Lists

There are three type of lists in HTML,

- Unordered list : bullet are used in it (see Lines 2 and 9)
- Ordered list : numbers are used in it (see Lines 15, 22 and 28)
- Definition list : This can be used for writing definitions in HTML (see Line 35)

```
<!-- Lists -->
<!-- unordered list -->
<ul> Unordered List <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
</ul>


<ul type="circle"> Change bullets : 'square', 'circle' or 'disc'
        <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
</ul>

<!-- ordered list -->
<ol> Ordered List
        <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
</ol>

<ol type='i'> Change style : 'i', 'I', '1', 'a' or 'A'
        <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
</ol>

<ol type='i' start="5"> Start from 'v'
        <li>Pen</li>
        <li>Pencil</li>
        <li>Eraser</li>
</ol>

<!-- Definition list -->
<dl>
        <dt> <h4>HTML Definition List</h4> </dt>
        <dd> HTML is easy </dd>
        <dd> HTML is good </dd> <dl>
```

The outputs of above codes are shown in Fig. 1.6,

1.9 Links

```
<!-- links -->
<p>Go to paragraph with<a href="#para1"> id='para1'</a></p>
<a href="http://pythondsp.readthedocs.io"> PythonDSP </a>

<br>
<p><a href="js.html" target="_self"> JavaScript Tutorial</a> in same window.</p>
<p><a href="js.html" target="_blank"> JavaScript Tutorial</a> in new Window.</p>
```

1.8. Lists

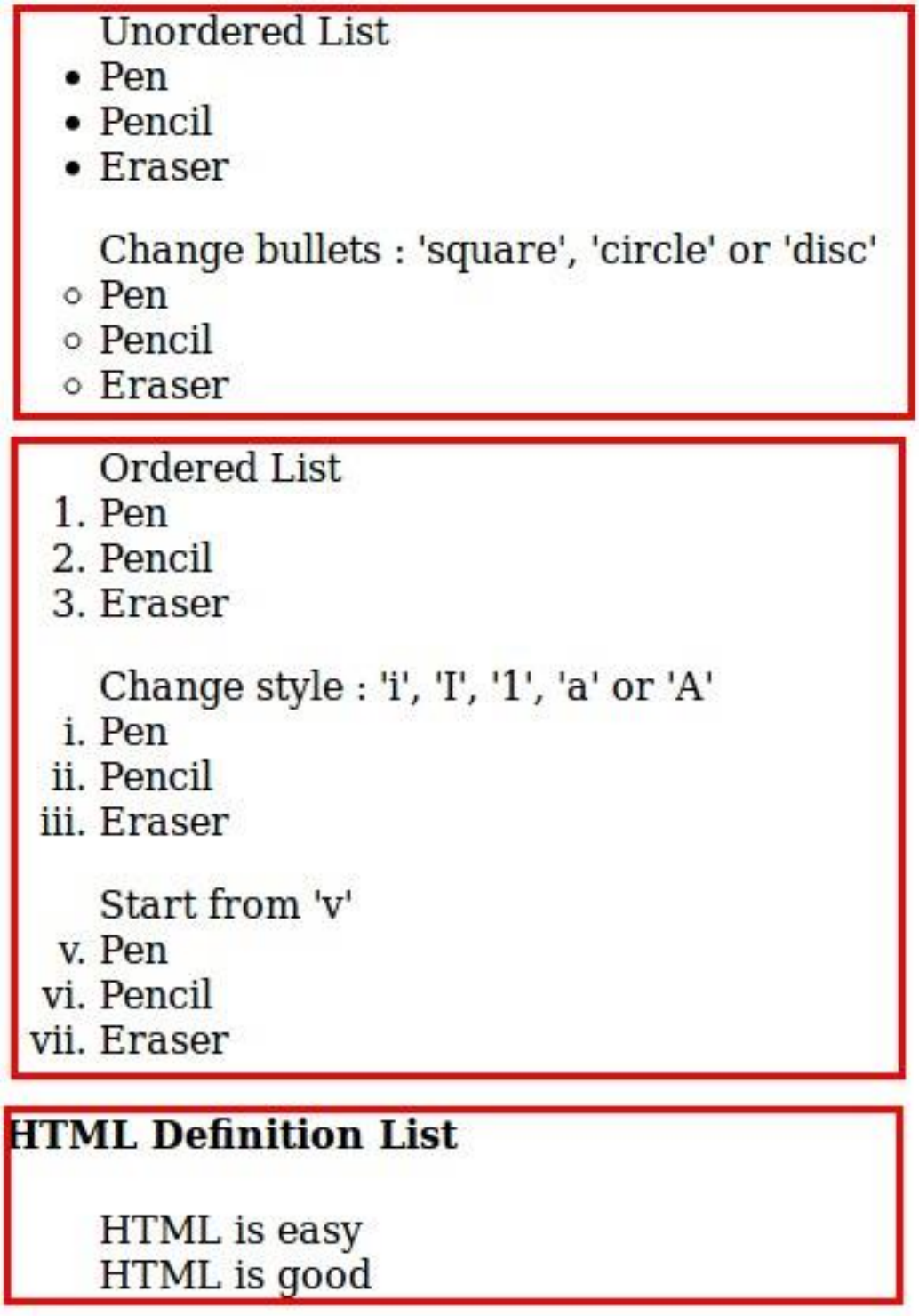

Fig. 1.6: Lists

1.9. Links

```
<p><a href="http://pythondsp.readthedocs.io/pdf">Download PDF, DOC or Zip Files</a></p>


<p><a href="mailto:pythondsp@gmail.com">Email me</a></p>
<p><a href="mailto:pythondsp@gmail.com?subject=Feedback&body=Your feedback here">Feeback email</a></p>
```

Note: We can change the color of the links using ‘alink (active link)’, ‘link’ and ‘vlink (visited link’, by defining these attributes in the ‘body tag’ as shown below,

```
<body alink="green" link="blue" vlink="red">
```

1.10 Forms

Forms can have different types of controls to collect the input-data from users, which are listed below and shown in Table 1.5,

- Text input
- Text area
- Radio button
- Checkbox
- Select box
- File select
- Buttons
- Submit and reset buttons
- Hidden input

Table 1.5: List of control inputs and their attributes

Control	Attributes	Values	Description
Input : text	type	text, password	
	value	user-defined	initial value in the area
	name	user-defined	name send to server
	size	numeric value	width of the text area
	maxlength	numeric value	maximum number of characters
Input : radio	type	radio	
	name	user-defined	name send to server
	value	user-defined value	value of the button if selected
	checked		check the button by default
Input : check box	type	checkbox	
	name	user-defined	name send to server
	value	user-defined value	value of the box if selected
	checked		check the box by default
Input : button	type	button	trigger client side script
		submit	submit the form and run ‘action’

		reset	reset form
		image	create image button
	method	get, post	get or post method
	action	user-defined	action to perform on submit
Input : hidden	type	hidden	will not display on html, but can be used
			for sending information to server
Selection box	name	user-defined	name send to server
	size	numeric value	enables scroll (default dropdown)
	multiple	numeric value	select multiple items
	value	user-defined value	value of the item if selected
	selected		select item by default
Text area	rows, cols	numeric value	number of rows and cols
	name	user-defined	name send to server

- Below are the exmaple of the control inputs described in Table 1.5

```
<!-- Forms --> <form>
    <h4>Text input </h4>
    Name : <input type="text" name="user_name" size="4" value="e.g. meher21" maxlength="10"><br> Password : <input
    type="password" name="user_pass" ><br>


    <h4> Radio button: name should be same</h4> <input
    type="radio" name="r_gender"> Male
    <input type="radio" name="r_gender"> Female
    <input type="radio" name="r_gender" checked> Infant

    <h4> Check box : name should be different</h4>
    <input type="checkbox" name="c_male" checked> Male
    <input type="checkbox" name="c_female"> Female
    <input type="checkbox" name="c_infant"> Infant

    <h4> Select box : drop-down</h4> <select
    name="s_box">
        <option value="s_male">Male</option>
        <option value="s_female" selected>Female</option>
        <option value="s_infant">Infant</option>
    </select>

    <h4> Select box : scroll</h4>
```

```
    <select name="s_box" size="4" multiple>
        <option value="s_male" selected>Male</option>
        <option value="s_female" selected>Female</option>
        <option value="s_infant">Infant 1</option>
        <option value="s_infant" selected>Infant 2</option>
        <option value="s_infant">Infant 3</option>
        <option value="s_infant">Infant 4</option>
    </select>

    <h4> Text area</h4>
    <textarea rows="10" cols="80" name="txt_area">Initial Text x = 2 y = 3
    </textarea> <!-- formatting work as pre -->

</form>
```

Fig. 1.7 is the output of above code,

- Below is the code which shows the working of various buttons. Note that method and action are defined in this form, which will be triggered on 'submit' button. Lastly, 'hidden' option is used in this example.

```
<form method="get|post" action="jquery.html">
    <h4> Buttons and Hidden</h4>

    Name : <input type="text" name="user_name" size="4" value="Meher" maxlength="16"><br> Password : <input
    type="password" name="user_pass" ><br>

    <input type="button" onclick="alert('Hello')" name="b_alert" value="Say Hello"/><br>
    <input type="submit" name="b_submit" value="Go to jQuery"/> <input
    type="reset" name="b_reset" value="Reset"/><br>

    <input type="hidden" name="h_data" value="html_tutorial"> </form>
```

Fig. 1.8 is the output of above code,

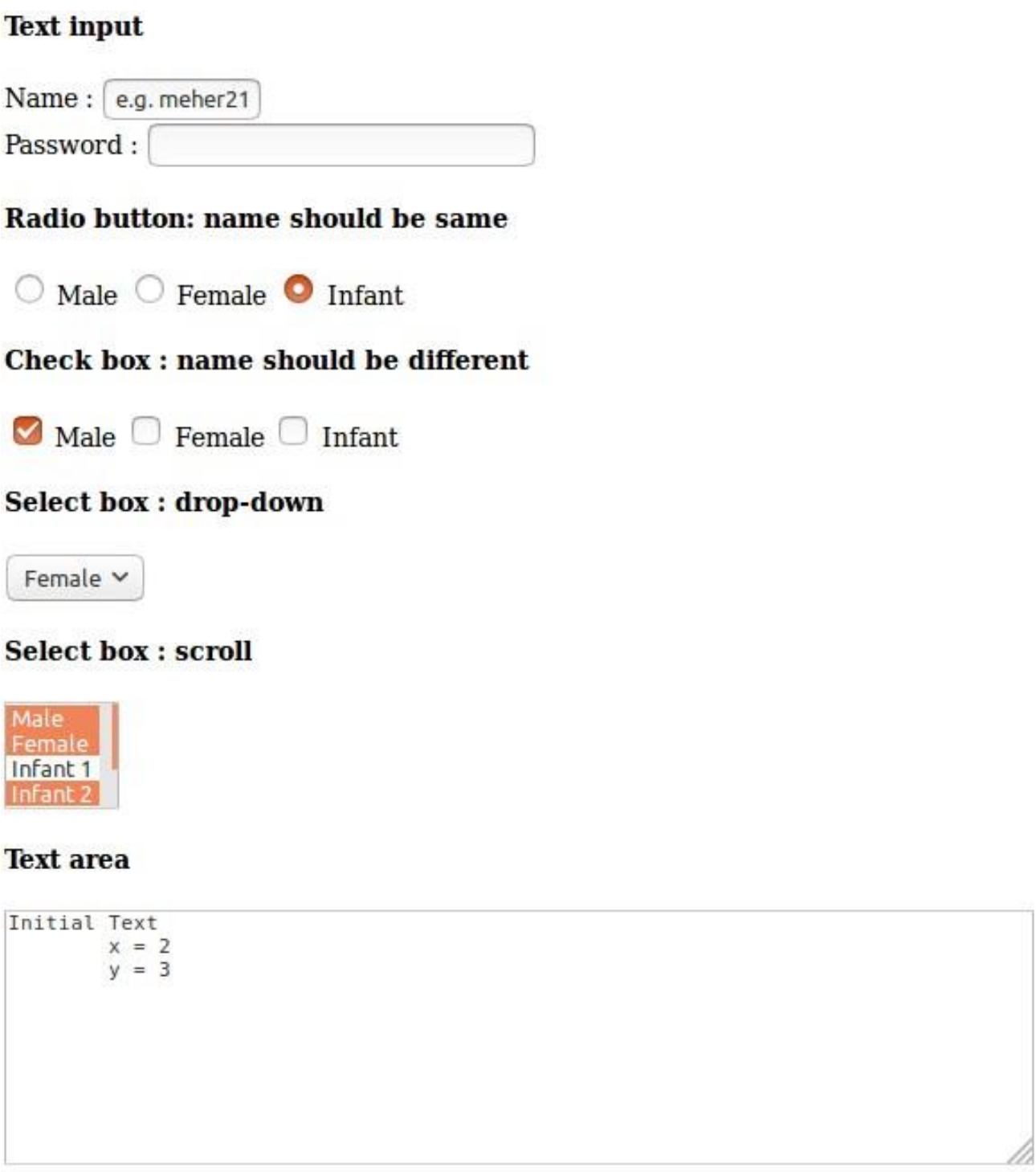

Fig. 1.7: Various control inputs for creating form

Fig. 1.8: Various buttons and hidden-input in the form

Chapter 2

Cascading Style Sheets (CSS)

2.1 Introduction

CSS is used to enhance the look of the web page. In Section 1.4.2, we saw the attribute 'style', which is used for changing the color of the text. Let's rewrite the example of 'style' as shown in next section.

2.1.1 Inline CSS

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
    <title>CSS Tutorial</title>

</head>
<body>

    <h3 style="color:blue"> Heading 1 </h3>
    <h3 style="color:blue"> Heading 3 </h3>
    <h3 style="color:blue"> Heading 3 </h3>

</body>
</html>
```

- Below code is an example of 'inline CSS', where the styles are defined inside the individual tags.

In the above code, we have three 'headings' with font-color as 'blue'. Suppose, we want to change the color to red, then we must go to to individual 'h3' tag and then change the color. This is easy in this case, but if we have 100 headings in 5 different 'html' files, then this process is not very handy. In such cases, CSS can be quite useful as shown in next section.

2.1.2 Embedded CSS

In the below code, the style is embedded inside the 'style' tag as shown in Lines 8-17. Here, we have defined two classes i.e. 'h3_blue (Lines 21-23)' and 'h3_red (Lines 26-28)'. Then, the selectors at Lines 9 and 13 targets the class 'h3_blue' & 'h3_red', and change the color to blue and red respectively. In this chapter, we will discuss the selectors (e.g. h3.h3_blue) in more details.

Note:

- In CSS, the comments are written between /* and */.
- CSS has three parts,
  - Selectors e.g. p, h3.h3_blue
  - Properties e.g. color
  - Values of properties e.g. red

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
	<title>CSS Tutorial</title>

	<style type="text/css"> h3.h3_blue{		/*change
		color to blue*/ color: blue;
		}

		 h3.h3_red{		/*change color to red*/
			color:red;
		}
	</style>


</head>
<body>

	<h3 class='h3_blue'> Heading 1 </h3>
	<h3 class='h3_blue'> Heading 3 </h3>
	<h3 class='h3_blue'> Heading 3 </h3>


	<h3 class='h3_red'> Heading 1 </h3>
	<h3 class='h3_red'> Heading 3 </h3>
	<h3 class='h3_red'> Heading 3 </h3>

</body>
</html>
```

- Below is the output of above code,

2.1.3 External CSS

We can write the 'CSS' code in different file and then import the file into 'html' document as shown in this section. In this way, we can manage the files easily.

- The 'CSS' code is saved in the file 'my_css.css' which is saved inside the folder 'asset/css'.

```
/* asset/css/my_css.css */

h3.h3_blue{ color:
	blue;
}

h3.h3_red{ color:red;
}
```

```
<!-- css.html -->
```

- Next, we need to import the CSS file into the 'html' file as shown in Line 7.

2.1. Introduction

Fig. 2.1: Embedded CSS

```
<!DOCTYPE html>
<html>
<head>
	<title>CSS Tutorial</title>
	<link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>

	<h3 class='h3_blue'> Heading 1 </h3>
	<h3 class='h3_blue'> Heading 3 </h3>
	<h3 class='h3_blue'> Heading 3 </h3>


	<h3 class='h3_red'> Heading 1 </h3>
	<h3 class='h3_red'> Heading 3 </h3>
	<h3 class='h3_red'> Heading 3 </h3>

</body>
</html>
```

2.2 Basic CSS Selectors

There are three types of selectors in CSS,

- Element : can be selected using it's name e.g. 'p', 'div' and 'h1' etc.
- Class : can be selected using '.className' operator e.g. '.h3_blue'.
- ID : can be selected using '#idName' e.g. '#my_para'.

We will use following HTML for understanding the selectors,

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
	<title>CSS Selectors</title>
	<link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>
	<h3>CSS Selectors</h3>

	<p class='c_head'> Paragraph with class 'c_head' </p>
	<p id='i_head'> Paragraph with id 'i_head' </p>

</body>
</html>
```

2.2. Basic CSS Selectors

```
/* asset/css/my_css.css */

/*element selection*/ h3 {
	color: blue;
}


/*class selection*/
.c_head{ font-family: cursive;
	color: orange;
}


/*id selection*/
#i_head{ font-variant: small-caps;
	color: red;
}
```

- Below code shows the example of different selectors, and the output is shown in Fig. 2.2

Fig. 2.2: Selectors : element, class and id

2.3 Hierarchy

In previous section, we saw the example of selectors. In this section, we will understand the hierarchy of the styling-operations.

Important: Below is the priority level for the CSS,

2.3. Hierarchy

- Priority level :
  - ID (highest priority)
  - Class
  - Element
- If two CSS has same priority, then CSS rule at the last will be applicable.

- Below is the html code with following tags,
  - 'p' tag
  - 'p' tag with class 'c_head'

```
<!-- css.html -->

<!DOCTYPE html>
<html>
<head>
    <title>CSS Selectors</title>
    <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
</head>
<body>
    <p>Paragraph</p>

    <p class='c_head'> Paragraph with class 'c_head' </p>
    <p class='c_head' id='i_head'> Paragraph with class 'c_head' and id 'i_head' </p>

</body>
</html>
```

  - 'p' tag with class 'c_head' and id 'i_head'

Below is the CSS code. Let's understand the formatting of all three 'p' tags individually. The results are shown in Fig. 2.3.

- 'p' tag at Line 13 of html : Since, 'id' has highest priority, therefore CSS rule for #i_head' (Line 12) will not be overridden by Line 24; hence the color is red. Line 13 has 'p' tag, therefore 'font-variant' rule will be applied by Line 17. Also, this tag has class 'c_head', therefore 'font' will be set to 'cursive'. Hence, the line is "all-caps with font-cursive in red color".
- 'p' tag at Line 12 of html : Similarly, the 'head' tag has higher priority than 'element' therefore color of this line is oranage and font-family is 'cursive'. Also, Line 17 will make it all caps
- 'p' tag at Line 10 of html : Color defined at Line 18 will be overridden by Line 24; hence the color will be blue. Also, Line 17 will make it all caps.

```
/* asset/css/my_css.css */

/*class selection*/
.c_head{ font-family: cursive;
         color: orange;                    /*override the blue color*/
}


/*id selection*/
#i_head{ color:
       red;
}

/*element selection*/
p {
       font-variant: small-caps; color: blue;
}
```

2.3. Hierarchy

```
/*element selection*/
p {
       color: green; }
```

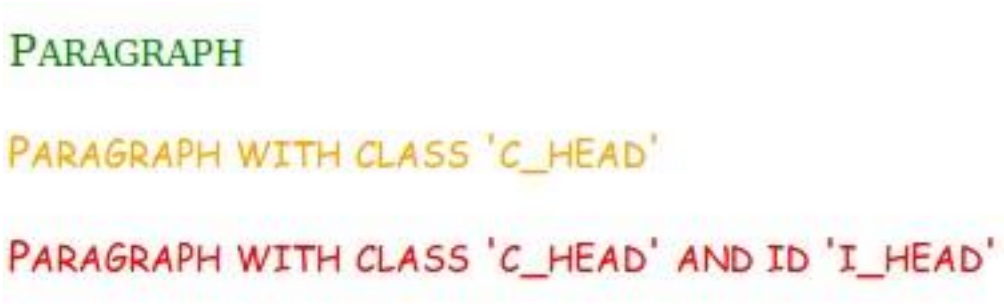

Fig. 2.3: Priority level for CSS rule

2.4 More selectors

Table 2.1 shows the combinations of selectors to target the various elements of the HTML. Also, some of the example of 'Attribute selector' is shown in this section.

Table 2.1: List of selectors

Selectors	Description
h1, p, span etc.	element selector
.className	class selector
#idName	id selector
*	Universal selector (selects everything)
h1.className	select h1 with class 'className'
h1#idName	select h1 with id 'idName'
p span	descendant selector (select span which is inside p)

p > span	child selector ('span' which is direct descendant of 'p')
h1, h2, p	group selection (select h1, h2 and p)
span[my_id]	select 'span' with attribute 'my_id'
span[my_id=m_span]	select 'span' with attribute 'my_id=m_span'

2.4.1 Attribute selector

- Add below code at the end of the html file. In these lines 'custom attributes' are added (i.e. my_id).

```
<!-- css.html -->

<span my_id='m_span'> Span with attribute 'my_id' with value 'm_span' </span> <br>
<span my_id='m_span2'> Span with attribute 'my_id' with value 'm_span2' </span>
```

- These custom attributes can be selected as below,

```
/*attribute selection*/
span[my_id] { /* select 'span' with attribute 'my_id' */ color: green; font-weight: bold
```

2.4. More selectors

```
}


span[my_id=m_span] { /* select 'span' with attribute 'my_id = m_span' */ color: red; }
```

2.5 More properties

Table 2.2 shows the some more important properties which can be used in CSS,

Table 2.2: More CSS properties

Property	Syntax	Description/possible values
size	20%	size = 20%
	20px	20 pixel
	2em	2*font-size
	2mm, 2cm, 2in	2 mm, cm and inch
color	names	e.g. red, blue, green
	hex code (#RRGGBB or #RGB)	#FFF000 or #FFF
	rgb(num, num, num)	rgb(0, 0, 255) or rgb(20%, 10%, 70%)
link	a:link	a:link {color: red}
	a:hover	
	a:visited	
	a:active	
Font	font-family	serif, cursive
	font-style	normal, italic, oblique
	font-variant	normal, small-caps

	font-weight	normal, bold, bolder, lighter, 100-900
	font-size	10px, small, medium, large etc.
Text	color	red, #FFF
	letter-spacing	10px
	word-spacing	10 px
	text-align	right, left, center
	text-decoration	underline, overline, line-through, none
	text-transform	capitalize, uppercase, lowercase, none
	white-space	pre, normal, nowrap
	text-shadow	text-shadow:5px 5px 8px red;
Image	border	'1px', or '1px solid blue'
	height, width	100px, 20%
Border	border-style	solid, dashed, dotted, double, none etc.
	border-top-style	
	border-bottom-style	
	border-left-style	
	border-right-style	
	border-width	4px, 4pt
	border-bottom-width	similarly use 'top', 'left', 'right'
	border (shortcut)	1px solid blue'
Margin	margin, margin-left etc.	
Padding	padding (top, bottom, left, right)	'10px 10px 2px 2px' or '10px 2px'
	padding-right, padding-left etc.	

2.5. More properties

Chapter 3

JavaScript

3.1 Introduction

- JavaScript is a dynamic language which is used for designing the web pages on the client side.
- It is case sensitive language.
- It is untyped language i.e. a variable can hold any type of value.
- // is used for comments.
- ; i used for line termination.
- JavaScript code should be added at the end i.e. just above the closing-body-tag.
- It is better to write the JavaScript code in separate file as shown in next section.

3.2 First code

The JavaScript code can be written in the 'html' file or in the separate 'JavaScript file (.js)' as shown in this section,

3.2.1 JavaScript in HTML file

In HTML file, the JavaScript codes can be written inside the 'script' tag as shown in Lines 11-13 of below code. The code will write the message "Hello World from JavaScript!" on the web page. Open the 'js.html' in the browser to see the message.

```
<!-- js.html -->

<!DOCTYPE html>
<html>
<head>
	<title>JavaScript</title>
</head>
<body>


	<script type="text/javascript"> document.write("Hello World from
		JavaScript!<br>");
	</script>

</body>
</html>
```

- Below is the output on the HTML page,

```
Hello World from JavaScript!
```

3.2.1.1 JavaScript in separate file

- The JavaScript code is saved in the file ‘my_javascript.js’ which is located inside the folders ‘asset/js’. Note that, no

```
// asset/js/my_javascript.js

document.write("Hello World from JavaScript!<br>");
```

‘script tag’ is used here.

```
<!-- js.html -->

<!DOCTYPE html>
<html>
<head>
    <title>JavaScript</title>
</head>
<body>



    <!-- import JavaScript files here -->
    <script src="asset/js/my_javascript.js"></script>

</body>
</html>
```

Next we need to import the .js file in the HTML file as shown below,

- Now, open the ‘js.html’ file in the browser and it will display the message.

Note: We will use the second method in this tutorial.

3.3 Keywords, Datatypes , Variables and Operators

3.3.1 Keywords

- Below are the reserved keywords in the JavaScript which can not be used as ‘variable’ and ‘function’ names etc.

Keywords			
abstract	boolean	break	byte
case	catch	char	class
const	continue	debugger	default
delete	do	double	else
enum	export	extends	false
final	float	for	function
goto	if	implements	import
in	int	interface	long
native	new	null	package
private	protected	public	return
short	static	super	switch
this	throw	throws	transient
true	try	typeof	var
void	volatile	while	with

3.3.2 Datatypes

JavaScript has three types of data,

- Numbers : 123, 32.32
- Strings : “Meher”, “Krishna Patel”, “123”
- Boolean : true, false

Note: All the numbers are considered as ‘floating point’.

3.3.3 Variables

Variables can be define using keyword ‘var’. Further, JavaScript is untyped language i.e. a variable can hold any type of value.

- In the below HTML, ‘p’ tag with id ‘p_name’ is defined at Line 10. This id will be used to write some text using JavaScript,

```
<!-- js.html -->

<!DOCTYPE html>
<html>
<head>
	<title>JavaScript</title>
</head>
<body>

	<p id='p_name'></p>

	<!-- import JavaScript files here -->
	<script src="asset/js/my_javascript.js"></script>

</body>
</html>
```

- Two variables are defined at Lines 9-10 of type ‘string’ and ‘float’ respectively. Then ‘getElementById’ is used to locate the tag with id ‘p_name’ and the text is inserted as shown in Line 11. Lastly, the ‘+’ sign is used to concatenate the values as Line 11.

```
// asset/js/my_javascript


// print message on the webpage
document.write("Hello World from JavaScript!<br>");


// variable example var
your_name = "Meher"; var
age = 20;
document.getElementById("p_name").innerHTML= "Hello "+ your_name + "<br>Age : " + age;
```

- Below is the output of above codes. Note that the message ‘Hello World from JavaScript!’ is added at the end as ‘JavaScript’ file is imported at the end.

Warning: If we import the 'JavaScript' file between the 'ending-head-tag' and 'start-body-tag', then Message 'Hello Meher ...' will not be displayed as 'JavaScript' will execute before the loading of the page; and JavaScript can not find the id 'p_name'.

```
Hello Meher Age
: 20

Hello World from JavaScript!
```

Note: All the JavaScript/HTML codes will be added below the existing codes, therefore only the newly added code will be shown in the rest of the tutorial.

- 'prompt' can be used to read the values from the user.

```
// prompt
var x = prompt("enter a number"); document.write("2 * ", x, " = ", 2*x + "<br>");
```

A pop-up window will appear due to prompt. If we provide the input value as 3, then below output will be generated,

```
2 * 3 = 6
```

Note: The ',' and '+' can be used to combine various outputs as shown in above example

3.3.4 Operators

Various operators are shown in this section. The usage of some of these operators are shown in Section 4.4.

3.3.4.1 Arithmetic operators

- +
- • *
- /
- % : modulus i.e remainder of the division
- ++ (increment)
- – (decrement)

3.3.4.2 Assignment operators

- =
- +=
- -=
- *=
- /=
- %=

3.3.4.3 Comparison operators

- == (compare only values not type)
- === (compare both values and type)

- !=
- >
- <
- >=
- <=

3.3.4.4 Conditional operator

- ?:

e.g. '((a > b) ? a/b : b/a))' i.e if 'a>b' then do 'a/b' else do 'b/a'

3.3.4.5 Logical operators

- && (logical and)
- || (logical or)
- ! (logical not)

3.3.4.6 Bitwise operators

- & (and)
- | (or)
- ^ (xor)
- ~ (not)

3.3.5 String to number conversion

'Number' is used to convert the string into numbers.

```
// string to number conversion document.write("2 + Number('3.4') = ", 2 +
Number('3.4'), "<br>");
```

Below is the output of above code,

```
2 + Number('3.4') = 5.4
```

3.3.6 Convert to integer

A string or float value can be converted into integer using 'parseInt'

```
// int conversion
document.write("2 + parseInt('3.4') = ", 2 + parseInt('3.4'), "<br>"); // string to int document.write("2 + parseInt(3.4) = ", 2 +
parseInt(3.4), "<br>"); // float to int
```

Below are the outputs of above code,

```
2 + parseInt('3.4') = 5 2 +
parseInt(3.4) = 5
```

3.3.7 Convert to float

The 'parseFloat' or 'Number' can be used to convert the value into float values.

```
document.write("2 + parseFloat('3.4') = ", 2 + parseFloat("3.4"), "<br>"); // parseFloat
```

3.3.8 Math

'Math' library contains various useful function as shown below,

```
// math
document.write("pi = ", Math.PI, "<br>"); document.write("e = ", Math.E,
"<br>");
document.write("similarly we can use 'abs', 'floor', 'ceil' and 'round' etc. <br>")
document.write("random number : ", Math.ceil(Math.random()*20), "<br>"); // enter random number
```

Below are the ouputs of above code,

```
pi = 3.141592653589793 e = 2.718281828459045 similarly we can use 'abs',
'floor', 'ceil' and 'round' etc. random number : 16
```

3.3.9 String

Below are the some of the useful 'string-styles',

```
// string
document.write("meher".toUpperCase(), "<br>") // uppercase

w = "Krishna"
document.write(w.toLowerCase(), "<br>") // lowercase document.write(w.small(), "<br>") // small document.write(w.bold(), "<br>")
// bold document.write(w.strike(), "<br>") // strike document.write(w.fontsize("5em"), "<br>") // strike
document.write(w.link("http://pythondsp.readthedocs.io"), "<br>") // link document.write(w.fontcolor("red").fontsize("12em"),
"<br>") // multiple
```

The outputs are shown in Fig. 4.1

3.3.10 Arrays

In JavaScript, the arrays can store different types of values as shown in below code,


```
// arrays
arr = [15, 30, "Meher"] for(a in arr)
document.write(arr[a], " ");
document.write("<br>");

document.write(arr.pop(), "<br>"); // remove last element arr.push("Krishna"); // add
element to end document.write(arr.pop(), "<br>");
document.write("lenght of array: ", arr.length, "<br>");
```

Fig. 4.1: String styles

Below are the output of above code,

```
15 30 Meher
Meher
Krishna lenght of array: 2
```

3.4 Control structure, loops and functions

3.4.1 If-else

In the below code, three conditions are checked for the variable 'age'; and corresponding message is printed.

```
// asset/js/my_javascript

// if-else age = 10; if (age > 3 && age < 6){ document.write("Age : " + age + "<b> go
to kindergarten</b>");
}
else if ( age >=6 && age < 18){ document.write("Age : " + age + "<b> go to
      school</b>");
} else{ document.write("Age : " + age + "<b> go to college</b>");
```

```
} document.write("<br>");
```

- Since age is 10, therefore 'else if' statement is satisfied and we will get below output,

```
Age : 10 go to school
```

3.4.2 Switch-case-default

Below is an example of Switch-case-default statement,

```
// asset/js/my_javascript

// switch-case var
grade = 'A';
document.write("Grade " + grade + " : ");
switch(grade){ case 'A': document.write("Very good
grade!"); break;
      case 'B': document.write("Good grade!");
            break;
      default: // if grade is neither 'A' nor 'B' document.write("Enter correct grade");
}
document.write("<br>");
```

- Below is the output of above code,

```
Grade A : Very good grade!
```

3.4.3 For loop

Below code prints the value from 5-0,

```
// For loop for (i=5; i>=0; i--){
document.write(i + " ");
} document.write("<br>");
```

- Below is the output,

```
5 4 3 2 1 0
```

3.4.4 While loop

Below code prints the value from 0-4,

```
// While loop x=0; while(x < 5){ document.write(x + " "); x++;
}
document.write("<br>");
```

3.4.5 do-while

Below code prints the value from 0-2,

```
// do-while x=0; do{
document.write(x + " "); x++;
}while(x < 3);
document.write("<br>");
```

Below is the output,

```
0 1 2
```

3.4.6 for-in loop

The 'for-in' loop can be used to iterate over the array as shown below,

```
// for-in loop
arr = [10, 12, 31]; // array for (a in arr){ document.write(arr[a] + " ");
}
document.write("<br>");
```

3.4.7 Continue and break

Continue and break statements are used inside the 'loop' statements.

- Continue will skip the loop and go to next iteration, if the condition is met. In the below code, 3 will not be printed at the output.

```
// continue for (i=5; i>=0; i--){ if
(i==3){ // skip 3
            continue;
      }
      document.write(i + " ");
}
document.write("<br>");
```

Below is the output where 3 is not printed in the output.

```
5 4 2 1 0
```

- 'Break' statement will quit the loop if the condition is met. In the below code, the for loop will be terminated at 'i=3', therefore only 5 and 4 will be printed,

```
// break
for (i=5; i>=0; i--){ if (i==3){ // exit loop when
      i=3
            break;
      }
      document.write(i + " ");
}
document.write("<br>");
```

Below is the output of the above code,

```
5 4
```

4.4.8 Functions

In the below code a function ‘add2Num’ is defined which adds the two numbers and returns the result.

```
// function
function add2Num(num1, num2){ // function definition return num1 + num2;
}
sum = add2Num(2, 3); // function call document.write("2 + 3 = " + sum); document.write("<br>");
```

Below is the output,

```
2 + 3 = 5
```

3.5 Event handling

One of the main usage of JavaScript is ‘event handling’. “Mouse click” and ‘pressing keys on keyboard’ are the example of ‘events’. With the help of JavaScript, we can perform certain actions at the occurrence of the events, as shown below.

- ‘alert’ is used to display message in the ‘pop up’ window.

```
function alertMessage(message){ alert(message)
}
```

- In Line 13, message ‘Hello’ is shown in the pop-up window on the event ‘onclick’. Note that “JavaScript:void(0)” is used here, which does not refresh the page.
- Line 15 is same as Line 13, but “JavaScript:void(0)” is not used, therefore page will be refreshed.
- Line 17 calls the function ‘alertMessage’ to display the mesage, which is defined in the above code.
- Lines 19-25 use different events i.e. ‘onmouseover’, ‘onmouseleave’, ‘onmouseup’ and ‘onmousedown’; and the color and the text of the string changes based on these operations.

```
<!-- js.html -->

<!DOCTYPE html>
<html>
<head>
	<title>JavaScript</title>
</head>
<body>

	<p id='p_name'></p>


<a href="JavaScript:void(0)", onclick="alert('Hello')">Say Hello</a><br> <!-- do not reload the page
-->


	<a href="", onclick="alert('Hello')">Say Bye</a><br> <!-- reload the page -->


	<a href="JavaScript:void(0)", onclick="alertMessage('Bye via function')">Bye-function</a><br>


	<a href="JavaScript:void(0)",
```

3 .5. Event handling

```
            onmouseover="this.style.color='red'", onmouseleave="this.style.color='blue'",
            onmouseup="this.text='Not clicked'", onmousedown="this.text='You clicked
            me'">
            Not clicked </a><br>



    <!-- import JavaScript files here -->
    <script src="asset/js/my_javascript.js"></script>

</body>
</html>
```