

SscLatestNews.Com

HTML, CSS, Bootstrap, Javascript and jQuery



<https://onlinestudytest.com/>

Table of contents

Table of contents	i
1 HTML	1
1.1 Introduction	1
1.2 First code	1
1.3 Basic tags	2
1.4 Attributes	4
1.4.1 Attribute 'name' and 'value'	4
1.4.2 Core attributes	4
1.5 Tables	5
1.6 Text formatting	6
1.7 Images	6
1.8 Lists	8
1.9 Links	8
1.10 Forms	10
2 Cascading Style Sheets (CSS)	15
2.1 Introduction	15
2.1.1 Inline CSS	15
2.1.2 Embedded CSS	15
2.1.3 External CSS	16
2.2 Basic CSS Selectors	17
2.3 Hierarchy	18
2.4 More selectors	20
2.4.1 Attribute selector	20
2.5 More properties	21
3 Bootstrap	22
3.1 Introduction	22
3.2 Setup	22
3.2.1 Download and include files	22
3.2.2 Add CDN	23
3.2.3 Check setup	23
3.3 Grid system	24
3.3.1 Example	25
3.3.2 Nested columns	26
3.3.3 Offset	26
3.4 Components	27
3.4.1 Labels	28
3.4.2 Buttons	29
3.4.3 Forms	29
3.4.4 Horizontal form	30
3.4.5 Form elements	33

3.4.6	Control size	35
3.4.7	More buttons	37
3.4.8	Input group	39
3.4.9	Navigation bar (navbar)	42
4	JavaScript	45
4.1	Introduction	45
4.2	First code	45
4.2.1	JavaScript in HTML file	45
4.3	Keywords, Datatypes, Variables and Operators	46
4.3.1	Keywords	46
4.3.2	Datatypes	47
4.3.3	Variables	47
4.3.4	Operators	48
4.3.5	String to number conversion	49
4.3.6	Convert to integer	49
4.3.7	Convert to float	50
4.3.8	Math	50
4.3.9	String	50
4.3.10	Arrays	50
4.4	Control structure, loops and functions	51
4.4.1	If-else	51
4.4.2	Switch-case-default	52
4.4.3	For loop	52
4.4.4	While loop	52
4.4.5	do-while	53
4.4.6	for-in loop	53
4.4.7	Continue and break	53
4.4.8	Functions	54
4.5	Event handling	54
4.6	Conclusion	55
5	jQuery	56
5.1	Introduction	56
5.1.1	Requirements	56
5.1.2	Add contents	57
5.2	jQuery examples	58
5.2.1	Add jQuery code	58
5.2.2	jQuery in separate file	60
5.2.3	Get input from user	61
5.3	Selectors	64
5.3.1	Select elements	64
5.3.2	Filters	67
5.4	Operations	69
5.5	Event handling	69

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>`).

```
1 <!DOCTYPE html> <!-- tells browser above the html version -->
2 <html> <!-- beginning of the html document -->
3   <head>
4     <!-- header related tags e.g. title, links etc. -->
5   </head>
6
7   <body>
8     <!-- actual html document here -->
9
10    <!-- add JavaScript files here -->
11  </body>
12 </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>
```

(continues on next page)

(continued from previous page)

```

<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	<h2> Hi </h2>
p	paragraphs (Line changes at the end)	<p> Hi </p>
span	No line change after span	Hi Bye.
div	make division between contents	<div> ... </div>
a	hyperlink	see Section 1.9
center	Move content to center	<center> Hi </center>
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>

```

(continues on next page)

(continued from previous page)

```
<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,

Heading 2

Heading 6

This is paragraph

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

The 'div' tag can be used for formatting the tags inside it at once using 'style' and 'classes' etc.

This paragraph is inside the 'div' tag

This span is inside the 'div' tag

Heading 3 is centered

Centered span inside the paragraph.

Two horizontal line is drawn using two 'hr' tag.

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

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

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

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	<code><p id='p_1'> Hi </p></code>
class	user defined names	<code><p class='p_class'> Hi </p></code>
style	CSS styles	<code><p style="color:red; font-weight:bold;"> Hi </p></code>
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.

1.5 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

```

1 <!-- border-color, width and height -->
2 <table border="1" bordercolor="black" width="450" height="100">
3 <caption>Table 1 : Various tags of table</caption>
4   <tr bgcolor="red" > <!-- row -->
5     <th>Column 1</th> <!-- header -->
6     <th>Column 2</th>
7     <th>Column 3</th>
8   </tr>
9
10  <tr bgcolor="cyan"> <!-- background color -->
11    <td>Data 1</td> <!-- data -->
12    <td>Data 2</td>
13    <td>Data 3</td>
14  </tr>
15
16  <tr bgcolor="yellow"> <!-- row -->
17    <td colspan="2">New Data 1</td> <!-- column span -->
18    <td>New Data 2</td> <!-- data -->
19  </tr>
20 </table>
21
22
23
24 <!-- width in % -->
25 <table border="1" bordercolor="black" width="80%" height="100">
26 <caption> Table 2 : Width is 80%</caption>
27   <tr bgcolor="red" >
28     <th>Column 1</th>
29     <th>Column 2</th>
30     <th>Column 3</th>
31   </tr>
32

```

(continues on next page)

(continued from previous page)

```

33 <tr bgcolor="cyan"> <!-- row -->
34 <td>Data 1</td> <!-- data -->
35 <td>Data 2</td>
36 <td>Data 3</td>
37 </tr>
38
39 </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 attributes 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'.

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 shown in Fig. 1.5.

```
1 <!-- Images -->
2 
3
4 <br/> <br/>
5
6 
```



Fig. 1.5: Images

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

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)

```

1 <!-- Lists -->
2 <!-- unordered list -->
3 <ul> Unordered List
4   <li>Pen</li>
5   <li>Pencil</li>
6   <li>Eraser</li>
7 </ul>
8
9 <ul type="circle"> Change bullets : 'square', 'circle' or 'disc'
10  <li>Pen</li>
11  <li>Pencil</li>
12  <li>Eraser</li>
13 </ul>
14
15 <!-- ordered list -->
16 <ol> Ordered List
17   <li>Pen</li>
18   <li>Pencil</li>
19   <li>Eraser</li>
20 </ol>
21
22 <ol type='i'> Change style : 'i', 'I', '1', 'a' or 'A'
23   <li>Pen</li>
24   <li>Pencil</li>
25   <li>Eraser</li>
26 </ol>
27
28 <ol type='i' start="5"> Start from 'v'
29   <li>Pen</li>
30   <li>Pencil</li>
31   <li>Eraser</li>
32 </ol>
33
34 <!-- Definition list -->
35 <dl>
36   <dt> <h4>HTML Definition List</h4> </dt>
37   <dd> HTML is easy </dd>
38   <dd> HTML is good </dd>
39 </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>

```

(continues on next page)

Unordered List

- Pen
- Pencil
- Eraser

Change bullets : 'square', 'circle' or 'disc'

- Pen
- Pencil
- Eraser

Ordered List

1. Pen
2. Pencil
3. Eraser

Change style : 'i', 'I', '1', 'a' or 'A'

- i. Pen
- ii. Pencil
- iii. Eraser

Start from 'v'

- v. Pen
- vi. Pencil
- vii. Eraser

HTML Definition List

HTML is easy
HTML is good

Fig. 1.6: Lists

(continued from previous page)

```
<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">Feedback email</a></p>
```

Go to paragraph with id='para1'

PythonDSP

JavaScript Tutorial in same window.

JavaScript Tutorial in new Window.

Download PDF, DOC or Zip Files

Email me

Feedback email

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>

```

(continues on next page)

(continued from previous page)

```

<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,

Text input

Name :

Password :

Radio button: name should be same

☐ Male ☐ Female ☒ Infant

Check box : name should be different

☒ Male ☐ Female ☐ Infant

Select box : drop-down

Select box : scroll

Male
Female
Infant 1
Infant 2

Text area

```
Initial Text
  x = 2
  y = 3
```

Fig. 1.7: Various control inputs for creating form

Buttons and Hidden

Name :

Password :

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

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

```
1 <!-- css.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>CSS Tutorial</title>
7
8 </head>
9 <body>
10
11     <h3 style="color:blue"> Heading 1 </h3>
12     <h3 style="color:blue"> Heading 3 </h3>
13     <h3 style="color:blue"> Heading 3 </h3>
14
15 </body>
16 </html>
```

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 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

```

1 <!-- css.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <title>CSS Tutorial</title>
7
8   <style type="text/css">
9     h3.h3_blue{      /*change color to blue*/
10      color: blue;
11    }
12
13    h3.h3_red{        /*change color to red*/
14      color:red;
15    }
16  </style>
17
18 </head>
19 <body>
20
21   <h3 class='h3_blue'> Heading 1 </h3>
22   <h3 class='h3_blue'> Heading 3 </h3>
23   <h3 class='h3_blue'> Heading 3 </h3>
24
25   <h3 class='h3_red'> Heading 1 </h3>
26   <h3 class='h3_red'> Heading 3 </h3>
27   <h3 class='h3_red'> Heading 3 </h3>
28
29 </body>
30 </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;
}

```

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

```

1 <!-- css.html -->
2

```

(continues on next page)

Heading 1

Heading 3

Heading 3

Heading 1

Heading 3

Heading 3

Fig. 2.1: Embedded CSS

(continued from previous page)

```
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <title>CSS Tutorial</title>
7   <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
8 </head>
9 <body>
10
11   <h3 class='h3_blue'> Heading 1 </h3>
12   <h3 class='h3_blue'> Heading 3 </h3>
13   <h3 class='h3_blue'> Heading 3 </h3>
14
15   <h3 class='h3_red'> Heading 1 </h3>
16   <h3 class='h3_red'> Heading 3 </h3>
17   <h3 class='h3_red'> Heading 3 </h3>
18
19 </body>
20 </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,

```

1 <!-- css.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>CSS Selectors</title>
7     <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
8 </head>
9 <body>
10     <h3>CSS Selectors</h3>
11
12     <p class='c_head'> Paragraph with class 'c_head' </p>
13     <p id='i_head'> Paragraph with id 'i_head' </p>
14
15 </body>
16 </html>

```

- Below code shows the example of different selectors, and the output is shown in [Fig. 2.2](#)

```

1 /* asset/css/my_css.css */
2
3 /*element selection*/
4 h3 {
5     color: blue;
6 }
7
8
9 /*class selection*/
10 .c_head{
11     font-family: cursive;
12     color: orange;
13 }
14
15
16 /*id selection*/
17 #i_head{
18     font-variant: small-caps;
19     color: red;
20 }

```

CSS Selectors

Paragraph with class 'c_head'

PARAGRAPH WITH ID 'I_HEAD'

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,

- 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’
 - ‘p’ tag with class ‘c_head’ and id ‘i_head’

```
1 <!-- css.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>CSS Selectors</title>
7     <link rel="stylesheet" type="text/css" href="asset/css/my_css.css">
8 </head>
9 <body>
10     <p>Paragraph</p>
11
12     <p class='c_head'> Paragraph with class 'c_head' </p>
13     <p class='c_head' id='i_head'> Paragraph with class 'c_head' and id 'i_head' </p>
14
15 </body>
16 </html>
```

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 **orange** 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**.

```
1 /* asset/css/my_css.css */
2
3 /*class selection*/
4 .c_head{
5     font-family: cursive;
6     color: orange;           /*override the blue color*/
7 }
8
9
10 /*id selection*/
11 #i_head{
12     color: red;
13 }
14
15 /*element selection*/
16 p {
17     font-variant: small-caps;
18     color: blue;
19 }
20
21
```

(continues on next page)

(continued from previous page)

```

22  /*element selection*/
23  p {
24      color: green;
25  }

```

PARAGRAPH

PARAGRAPH WITH CLASS 'C_HEAD'

PARAGRAPH WITH CLASS 'C_HEAD' AND ID 'I_HEAD'

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
}

```

(continues on next page)

(continued from previous page)

```

}

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 {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.	

Chapter 3

Bootstrap

3.1 Introduction

One of the problem with basic HTML design is that the webpage may look different in different browser or device (e.g. mobile, tablet and laptop). Therefore, we may need to modify the code according to browser or device. The problem can be easily resolved by using Bootstrap.

Bootstrap is a framework which uses HTML, CSS and JavaScript for the web design. It is supported by all the major browsers e.g. Firefox, Opera and Chrome etc. Further, Bootstrap includes several predefined classes for easy layouts e.g. dropdown buttons, navigation bar and alerts etc. Lastly, it is responsive in nature i.e. the layout changes automatically according to the device e.g. mobile or laptop etc.

3.2 Setup

Bootstrap needs atleast 3 files for its operation which can be [downloaded from the Bootstrap website](#).

- **bootstrap.css (Line 7)** : This file contains various CSS for bootstrap.
- **bootstrap.js (Line 16)** : This file contains various JavaScript functionalities e.g. dropdown and alerts etc.
- **jQuery.js (Line 17)** : This file is the jQuery library which can be downloaded from the 'jQuery' website. It is required for proper working of 'bootstrap.js'.

3.2.1 Download and include files

These files are downloaded and saved inside the 'asset' folder. Next, we need to include these files in the HTML document as below,

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Bootstrap Tutorial</title>
5
6      <!-- CSS -->
7      <link href="asset/css/bootstrap.min.css" rel="stylesheet">
8      <!-- Add Custom CSS below -->
9
10 </head>
11 <body>
12
13
14      <!-- Javascript -->
15      <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery-->
```

(continues on next page)

(continued from previous page)

```
16 <script src="asset/js/jquery-3.3.1.min.js"></script>
17 <script src="asset/js/bootstrap.min.js"></script>
18 </body>
19 </html>
```

3.2.2 Add CDN

Another way to include the files is CDN. In this method, we need not to download the files, but provide the links to these files, as shown in Lines 8, 17 and 19 of below code. Note that, in this case the code will not work in offline mode.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bootstrap Tutorial</title>
5
6   <!-- CSS -->
7   <!-- Latest compiled and minified CSS -->
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
  integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg320mUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin=
  "anonymous">
9   <!-- Add Custom CSS below -->
10
11 </head>
12 <body>
13
14   <!-- Javascript -->
15   <!-- jQuery -->
16   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
17   <!-- Latest compiled and minified JavaScript -->
18   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-
  Tc5Iqib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIPG9mGCD8wGNlCPD7Txa" crossorigin="anonymous"></script>
19
20 </body>
21 </html>
```

Note: In this tutorial, we have used the first method i.e. download the files and include in the html document.

3.2.3 Check setup

Let's write our first code to check the correct setup of the Bootstrap.

Note: We need not to memorize the code for creating the 'dropdown' button. All the templates are available on the [Bootstrap website](#). Copy and paste the code from there and the modify the code according to need as shown in this tutorial.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bootstrap Tutorial</title>
5
6   <!-- CSS -->
7   <link href="asset/css/bootstrap.min.css" rel="stylesheet">
8   <!-- Add Custom CSS below -->
```

(continues on next page)

(continued from previous page)

```

9
10 </head>
11 <body>
12
13     <div class="dropdown">
14         <button class="btn btn-success dropdown-toggle" type="button" id="dropdownMenu1" data-toggle=
15         → "dropdown" aria-haspopup="true" aria-expanded="true">
16             Dropdown
17         <span class="caret"></span>
18         </button>
19         <ul class="dropdown-menu" aria-labelledby="dropdownMenu1">
20             <li><a href="#">Action</a></li>
21             <li><a href="#">Another action</a></li>
22             <li><a href="#">Something else here</a></li>
23             <li role="separator" class="divider"></li>
24             <li><a href="#">Separated link</a></li>
25         </ul>
26     </div>
27
28     <!-- Javascript -->
29     <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
30     <script src="asset/js/jquery-3.3.1.min.js"></script>
31     <script src="asset/js/bootstrap.min.js"></script>
32 </body>
</html>

```

The resultant web page is shown in Fig. 3.1,



Fig. 3.1: Click on the 'Dropdown' button

3.3 Grid system

Bootstrap divides the **each row** into **12 columns**. Then following commands can be used to specify the width the columns

- **col-lg-4** : It will select 4 columns. Choose any number between 1-12. The 'lg' stand for **large screen** (e.g. large computer screen).
- **col-md-5** : 'md' = medium screen
- **col-sm-3** : 'sm' = small screen
- **col-xs-3** : 'xs' = extra small screen
- **col-lg-offset-4** : skip first 4 columns. Simimlary use 'md', 'sm' and 'xs' with offset

3.3.1 Example

Below is an example of grid system. Read the content of the Lines 13-16 to understand it. The resultant webpage is shown in [Fig. 3.2](#).

Note:

- For easy visualization, in the below code the CSS code (Lines 9-21) is used to fill the columns with colors and border.
- The columns (Lines 28-30) should be defined inside the class 'row' (Line 27).
- Also, sum of the widths of individual columns should not be greater than 12.
- Lastly, if we use 'col-md-4' without defining the 'lg', 'sm' and 'xs', then 'md' rule will be applicable to higher size screen 'lg', but **not** on the lower size screen 'sm' and 'xs'. Similarly, if we use 'sm' without defining the 'lg', 'md' and 'xs', then rule will be applicable to higher size screen i.e. 'lg' and 'md' but not on 'xs'

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bootstrap Tutorial</title>
5
6   <!-- CSS -->
7   <link href="asset/css/bootstrap.min.css" rel="stylesheet">
8   <!-- Add Custom CSS below -->
9   <style type="text/css">
10     .col-xs-1, .col-xs-2, .col-xs-3, .col-xs-4, .col-xs-5, .col-xs-6, .col-xs-7, .col-xs-8,
11     .col-sm-1, .col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7, .col-sm-8,
12     .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8,
13     .col-lg-1, .col-lg-2, .col-lg-3, .col-lg-4, .col-lg-5, .col-lg-6, .col-lg-7, .col-lg-8 {
14       background-color: green;
15       color: white;
16       font-weight: bold;
17       border: 1px solid red;
18       height: 3em; /*height of the box*/
19       text-align: center; /*vertical center the text*/
20     }
21   </style>
22
23 </head>
24
25 <body>
26
27   <div class="row">
28     <div class="col-md-2 col-xs-4">col-md-2, col-xs-4</div>
29     <div class="col-md-6 col-xs-4">col-md-6, col-xs-4</div>
30     <div class="col-md-4 col-xs-4">col-md-4, col-xs-4</div>
31   </div>
32
33
34   <!-- Javascript -->
35   <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
36   <script src="asset/js/jquery-3.3.1.min.js"></script>
37   <script src="asset/js/bootstrap.min.js"></script>
38 </body>
39 </html>
```

col-md-2, col-xs-4	col-md-6, col-xs-4	col-md-4, col-xs-4
--------------------	--------------------	--------------------

Fig. 3.2: Different grid-size (i.e. 2, 6, 4) for 'medium' and 'large' screen

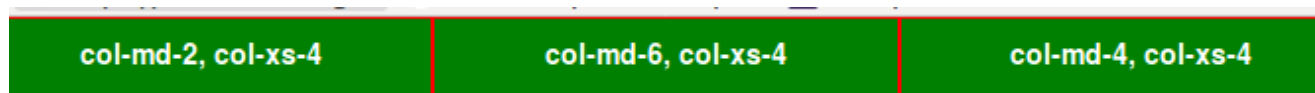


Fig. 3.3: Equal grid-size (i.e. 4) for ‘extra small’ and ‘small’ screen

3.3.2 Nested columns

We can further divide a column into small columns by defining a class ‘row’ (Line 30) inside the ‘column’ (Line 29), as shown in Lines 29-34.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bootstrap Tutorial</title>
5
6   <!-- CSS -->
7   <link href="asset/css/bootstrap.min.css" rel="stylesheet">
8   <!-- Add Custom CSS below -->
9   <style type="text/css">
10     .col-xs-1, .col-xs-2, .col-xs-3, .col-xs-4, .col-xs-5, .col-xs-6, .col-xs-7, .col-xs-8,
11     .col-sm-1, .col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7, .col-sm-8,
12     .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8,
13     .col-lg-1, .col-lg-2, .col-lg-3, .col-lg-4, .col-lg-5, .col-lg-6, .col-lg-7, .col-lg-8 {
14       background-color: green;
15       color: white;
16       font-weight: bold;
17       border: 1px solid red;
18       height: 3em; /*height of the box*/
19       text-align: center; /*vertical center the text*/
20     }
21   </style>
22
23 </head>
24
25 <body>
26
27   <div class="row">
28     <div class="col-md-2 col-xs-4">col-md-2, col-xs-4</div>
29     <div class="col-md-6 col-xs-4">
30       <div class="row">
31         <div class="col-xs-6">col-xs-6</div>
32         <div class="col-xs-6">col-xs-6</div>
33       </div>
34     </div>
35     <div class="col-md-4 col-xs-4">col-md-4, col-xs-4</div>
36   </div>
37
38
39   <!-- Javascript -->
40   <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
41   <script src="asset/js/jquery-3.3.1.min.js"></script>
42   <script src="asset/js/bootstrap.min.js"></script>
43 </body>
44 </html>

```

3.3.3 Offset

We can skip the columns using ‘offset’ as shown in Line 27; and the corresponding html page is shown in [Fig. 3.4](#),

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Bootstrap Tutorial</title>
5
6   <!-- CSS -->
7   <link href="asset/css/bootstrap.min.css" rel="stylesheet">
8   <!-- Add Custom CSS below -->
9   <style type="text/css">
10     .col-xs-1, .col-xs-2, .col-xs-3, .col-xs-4, .col-xs-5, .col-xs-6, .col-xs-7, .col-xs-8,
11     .col-sm-1, .col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7, .col-sm-8,
12     .col-md-1, .col-md-2, .col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8,
13     .col-lg-1, .col-lg-2, .col-lg-3, .col-lg-4, .col-lg-5, .col-lg-6, .col-lg-7, .col-lg-8 {
14       background-color: green;
15       color: white;
16       font-weight: bold;
17       border: 1px solid red;
18       height: 3em; /*height of the box*/
19       text-align: center; /*vertical center the text*/
20     }
21   </style>
22 </head>
23
24 <body>
25
26   <div class="row">
27     <div class="col-md-offset-2 col-md-2">col-md-2</div>
28     <div class="col-md-8">col-md-8</div>
29   </div>
30
31
32   <!-- Javascript -->
33   <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
34   <script src="asset/js/jquery-3.3.1.min.js"></script>
35   <script src="asset/js/bootstrap.min.js"></script>
36 </body>
37 </html>
```

col-md-offset-2

col-md-2

col-md-8

Fig. 3.4: Offset

3.4 Components

Once we understand the ‘html’, ‘css’ and ‘Bootstrap-grid’ understood, then the use of Bootstrap-components are straightforward. Just copy and paste the code from the [Bootstrap website](#) and modify it according to the requirement.

In this section, we will use following template.

```
<!DOCTYPE html>
<html>
<head>
  <title>Bootstrap Tutorial</title>
  <!-- CSS -->
  <link href="asset/css/bootstrap.min.css" rel="stylesheet">
  <!-- Add Custom CSS below -->
</head>
```

(continues on next page)

(continued from previous page)

```

<body>

    <!-- Javascript -->
    <!-- put jQuery.js before bootstrap.min.js; and then add custom jQuery -->
    <script src="asset/js/jquery-3.3.1.min.js"></script>
    <script src="asset/js/bootstrap.min.js"></script>
</body>
</html>

```

3.4.1 Labels

- The class 'label' is used to create the label (Line 11).
- The class 'label-info' sets the color of the label to 'blue' (Line 11).
- The size of the 'labels' can be changes using 'element' tags (e.g. 'h3' in Line 11) outside the 'span' tag (Line 11).

Note: Bootstrap provides 6 color option,

- danger
- default
- info
- primary
- success
- warning

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Bootstrap Tutorial</title>
5      <!-- CSS -->
6      <link href="asset/css/bootstrap.min.css" rel="stylesheet">
7      <!-- Add Custom CSS below -->
8  </head>
9
10 <body>
11     <h3>Example heading <span class="label label-info">New</span></h3>
12     <hr>
13
14     <!-- Javascript -->
15     <!-- put jQuery.js before bootstrap.min.js; and then add custom jQuery -->
16     <script src="asset/js/jquery-3.3.1.min.js"></script>
17     <script src="asset/js/bootstrap.min.js"></script>
18 </body>
19 </html>

```

- Fig. 3.5 is the output of above code,

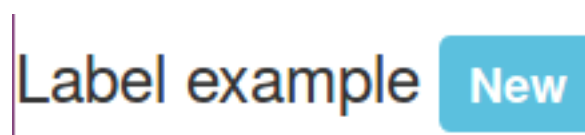


Fig. 3.5: Label

Note: All the codes will be added in the 'body' tag, and the new code will be inserted below the previous codes. Therefore full HTML code is not added in the tutorial.

3.4.2 Buttons

The element 'button' can be used to create the button in Bootstrap as shown below,

Note:

- We can use 'lg', 'md' and 'xs' etc. to set the size of the button.
- In Line 3, a glyphicon is added (i.e. trash sign). More glyphicon can be found on the [Bootstrap-webpage](#).

```
<h3> Buttons </h3>
<button type="button" class="btn btn-primary">Sign in</button>
<button type="button" class="btn btn-danger"> Delete <span class="glyphicon glyphicon-trash"><span></
→button>
<button type="submit" class="btn btn-lg btn-success">Submit</button>
<button type="submit" class="btn btn-md btn-success">Submit</button>
<button type="submit" class="btn btn-sm btn-success">Submit</button>
```

- Fig. 3.6 is the output of above code,

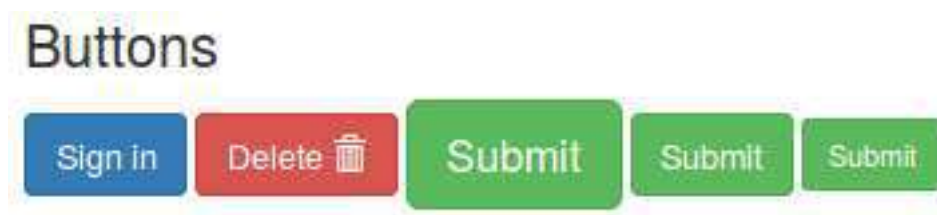


Fig. 3.6: Buttons

3.4.3 Forms

In this section, 4 types of forms are added,

3.4.3.1 Basic form

Output of below code is shown in Fig. 3.7,

```
<h2> Basic form </h2>
<div class="row">
<div class="col-sm-5" style="background: pink">
<div class="panel-body">
  <form role="form">
    <div class="form-group">
      <label for="exampleInputEmail1">Email address</label>
      <input class="form-control" id="exampleInputEmail1" placeholder="Enter email" type="email">
    </div>
    <div class="form-group">
      <label for="exampleInputPassword1">Password</label>
      <input class="form-control" id="exampleInputPassword1" placeholder="Password" type="password
→">
  </div>
```

(continues on next page)

(continued from previous page)

```

<div class="form-group">
  <label for="exampleInputFile">File input</label>
  <input id="exampleInputFile" type="file">
  <p class="help-block">Example block-level help text here.</p>
</div>
<div class="checkbox">
  <label>
    <input type="checkbox"> Check me out
  </label>
</div>
<button type="submit" class="btn btn-info">Submit</button>
</form>
</div>
</div>
</div>
<br><br>

```

Basic form

Fig. 3.7: Basic form

3.4.4 Horizontal form

Output of below code is shown in Fig. 3.8,

```

<h3> Horizontal form </h3>
<div class="row">

```

(continues on next page)

(continued from previous page)

```
<div class="col-sm-5" style="background: pink">
<div class="panel-body">
  <form class="form-horizontal" role="form">
    <div class="form-group">
      <label for="inputEmail1" class="col-lg-2 col-sm-2 control-label">Email</label>
      <div class="col-lg-10">
        <input class="form-control" id="inputEmail1" placeholder="Email" type="email">
        <p class="help-block">Example block-level help text here.</p>
      </div>
    </div>
    <div class="form-group">
      <label for="inputPassword1" class="col-lg-2 col-sm-2 control-label">Password</label>
      <div class="col-lg-10">
        <input class="form-control" id="inputPassword1" placeholder="Password" type="password">
      </div>
    </div>
    <div class="form-group">
      <div class="col-lg-offset-2 col-lg-10">
        <div class="checkbox">
          <input type="checkbox"> Remember me
        </div>
      </div>
    </div>
    <div class="form-group">
      <div class="col-lg-offset-2 col-lg-10">
        <button type="submit" class="btn btn-danger">Sign in</button>
      </div>
    </div>
  </form>
</div>
</div>
</div>
```

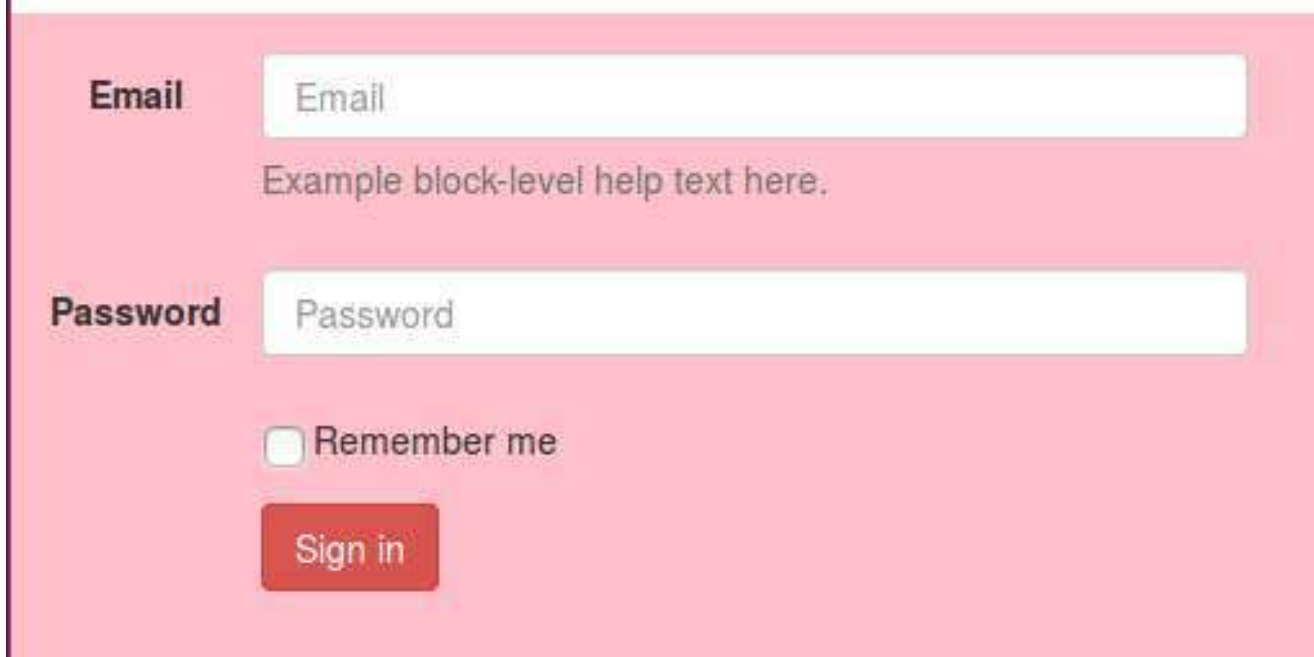
3.4.4.1 Inline form

Output of below code is shown in [Fig. 3.9](#),

```
<h2> Inline form </h2>
<div class="row">
<div class="col-sm-5" style="background: pink">
<div class="panel-body">
  <form class="form-inline" role="form">
    <div class="form-group">
      <label class="sr-only" for="exampleInputEmail2">Email address</label>
      <input class="form-control" id="exampleInputEmail2" placeholder="Enter email" type="email">
    </div>
    <div class="form-group">
      <label class="sr-only" for="exampleInputPassword2">Password</label>
      <input class="form-control" id="exampleInputPassword2" placeholder="Password" type="password"
→">
    </div>
    <div class="checkbox">
      <input type="checkbox"> Remember me
    </div>
    <button type="submit" class="btn btn-success">Sign in</button>
  </form>
```

(continues on next page)

Horizontal form



A horizontal form layout on a pink background. It features two input fields: 'Email' and 'Password', each with a label to its left. Below the 'Email' field is a line of placeholder text: 'Example block-level help text here.'. Below the 'Password' field is a checkbox labeled 'Remember me'. At the bottom is a red 'Sign in' button.

Fig. 3.8: Horizontal form

(continued from previous page)

```
</div>
</div>
</div>
<br><br>
```

Inline form



An inline form layout on a pink background. It features two input fields: 'Enter email' and 'Password', each with a label to its left. Below the 'Enter email' field is a checkbox labeled 'Remember me'. To the right of the checkbox is a green 'Sign in' button.

Fig. 3.9: Inline form

3.4.4.2 Modal form

Modal form appears in the pop-up window. Output of below code is shown in [Fig. 3.10](#),

```
<!-- modal form -->
<h3> Modal form </h3>

<div class="row">
<div class="col-sm-5" style="background: pink">
```

(continues on next page)

```

<div class="panel-body">
  <!-- button to generate model form -->
  <a href="#myModal" data-toggle="modal" class="btn btn-xs btn-success">
    Form in Modal
  </a>

  <!-- model form settings -->

  <div aria-hidden="true" aria-labelledby="myModalLabel" role="dialog" tabindex="-1" id="myModal"
  class="modal fade">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <button aria-hidden="true" data-dismiss="modal" class="close" type="button">&times;</
button>

          <h4 class="modal-title">Form Tittle</h4>
        </div>
        <div class="modal-body">

          <!-- actual form -->
          <form role="form">
            <div class="form-group">
              <label for="exampleInputEmail1">Email address</label>
              <input class="form-control" id="exampleInputEmail3" placeholder="Enter email
" type="email">
            </div>
            <div class="form-group">
              <label for="exampleInputPassword1">Password</label>
              <input class="form-control" id="exampleInputPassword3" placeholder="Password
" type="password">
            </div>
            <div class="form-group">
              <label for="exampleInputFile">File input</label>
              <input id="exampleInputFile3" type="file">
              <p class="help-block">Example block-level help text here.</p>
            </div>
            <div class="checkbox">
              <label>
                <input type="checkbox"> Check me out
              </label>
            </div>
            <button type="submit" class="btn btn-default">Submit</button>
          </form>
          <!-- actual form ends -->

        </div>
      </div>
    </div>
  </div>
</div>
</div>
</div>
<hr>

```

3.4.5 Form elements

Output of below code is shown in Fig. 3.11,

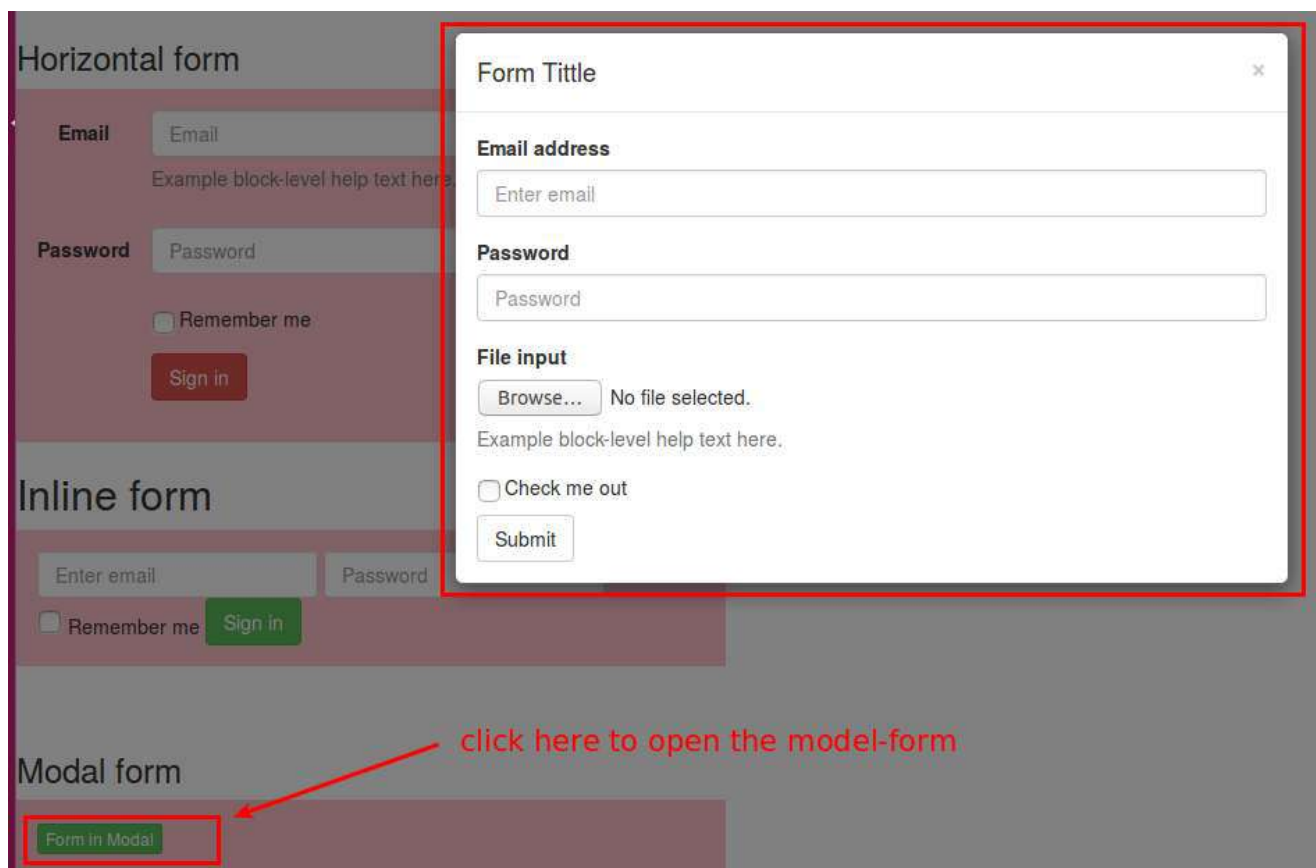


Fig. 3.10: Modal form

```

<!-- Form elements -->
<h3> Form elements </h3>
<div class="row">
<div class="col-sm-5" style="background: pink">
<div class="panel-body">
  <form class="form-horizontal tasi-form" method="get">
    <div class="form-group">
      <label class="col-sm-2 col-sm-2 control-label">Default</label>
      <div class="col-sm-10">
        <input class="form-control" type="text">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 col-sm-2 control-label">Help text</label>
      <div class="col-sm-10">
        <input class="form-control" type="text">
        <span class="help-block">A block of help text that breaks onto a new line and may
→ extend beyond one line.</span>
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 col-sm-2 control-label">Rounder</label>
      <div class="col-sm-10">
        <input class="form-control round-input" type="text">
      </div>
    </div>
    <div class="form-group">
      <label class="col-sm-2 col-sm-2 control-label">Input focus</label>

```

(continues on next page)

(continued from previous page)

```

        <div class="col-sm-10">
            <input class="form-control" id="focusedInput" value="This is focused..." type="text">
        </div>
    </div>
    <div class="form-group">
        <label class="col-sm-2 col-sm-2 control-label">Disabled</label>
        <div class="col-sm-10">
            <input class="form-control" id="disabledInput" placeholder="Disabled input here..."
→disabled="disabled" type="text">
        </div>
    </div>
    <div class="form-group">
        <label class="col-sm-2 col-sm-2 control-label">Placeholder</label>
        <div class="col-sm-10">
            <input class="form-control" placeholder="placeholder" type="text">
        </div>
    </div>
    <div class="form-group">
        <label class="col-sm-2 col-sm-2 control-label">Password</label>
        <div class="col-sm-10">
            <input class="form-control" placeholder="" type="password">
        </div>
    </div>
    <div class="form-group">
        <label class="col-lg-2 col-sm-2 control-label">Static control</label>
        <div class="col-lg-10">
            <p class="form-control-static">email@example.com</p>
        </div>
    </div>
</form>
</div>
</div>
</div>

```

3.4.6 Control size

Output of below code is shown in [Fig. 3.12](#),

```

<!-- Control size -->
<h3>Control size</h3>
<div class="row">
<div class="col-sm-5" style="background: pink">
<div class="panel-body">
    <form class="form-horizontal tasi-form" method="get">
        <div class="form-group">
            <label class="col-sm-2 control-label col-lg-2" for="inputSuccess">Control sizing</label>
            <div class="col-lg-10">
                <input class="form-control input-lg m-bot15" placeholder=".input-lg" type="text">
                <input class="form-control m-bot15" placeholder="Default input" type="text">
                <input class="form-control input-sm m-bot15" placeholder=".input-sm" type="text">

                <select class="form-control input-lg m-bot15">
                    <option selected="selected">Option 1</option>
                    <option>Option 2</option>
                    <option>Option 3</option>
                </select>
                <select class="form-control m-bot15">
                    <option selected="selected">Option 1</option>
                    <option>Option 2</option>

```

(continues on next page)

Form elements

Default	<input type="text"/>
Help text	<input type="text"/> A block of help text that breaks onto a new line and may extend beyond one line.
Rounded	<input type="text"/>
Input focus	<input type="text" value="This is focused..."/>
Disabled	<input type="text" value="Disabled input here..."/>
Placeholder	<input type="text" value="placeholder"/>
Password	<input type="password"/>
Static control	email@example.com

Fig. 3.11: Form elements

(continued from previous page)

```

        <option>Option 3</option>
    </select>
    <select class="form-control input-sm m-bot15">
        <option selected="selected">Option 1</option>
        <option>Option 2</option>
        <option>Option 3</option>
    </select>
</div>
</div>
</form>
</div>
</div>
</div>

```

Control size

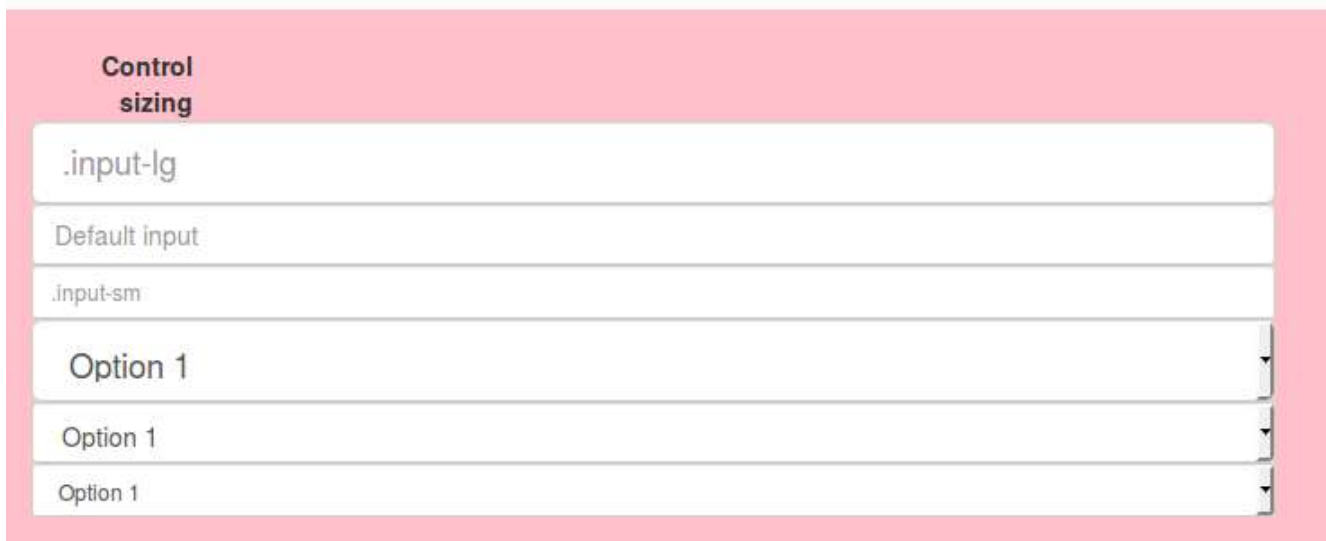


Fig. 3.12: Form elements

3.4.7 More buttons

Output of below code is shown in Fig. 3.13,

```

<h3>Buttons</h3>
<div class="row">
  <div class="col-sm-12" style="background: pink">
    <div class="panel-body">
      <form class="form-horizontal tasi-form" method="get">
        <div class="form-group">
          <label class="col-sm-2 control-label col-lg-2" for="inputSuccess">Checkboxes and
→ radios</label>
          <div class="col-lg-10">
            <div class="checkbox">
              <label>
                <input value="" type="checkbox">
                Option one is this and that-be sure to include why it's great
              </label>
            </div>
            <div class="checkbox">

```

(continues on next page)

(continued from previous page)

```

        <label>
            <input value="" type="checkbox">
            Option one is this and that-be sure to include why it's great option one
        </label>
    </div>

    <div class="radio">
        <label>
            <input name="optionsRadios" id="optionsRadios1" value="option1" checked=
→ "checked" type="radio">
            Option one is this and that-be sure to include why it's great
        </label>
    </div>
    <div class="radio">
        <label>
            <input name="optionsRadios" id="optionsRadios2" value="option2" type=
→ "radio">
            Option two can be something else and selecting it will deselect option_
→ one
        </label>
    </div>

</div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label col-lg-2" for="inputSuccess">Inline checkboxes</
→ label>
    <div class="col-lg-10">
        <label class="checkbox-inline">
            <input id="inlineCheckbox1" value="option1" type="checkbox"> 1
        </label>
        <label class="checkbox-inline">
            <input id="inlineCheckbox2" value="option2" type="checkbox"> 2
        </label>
        <label class="checkbox-inline">
            <input id="inlineCheckbox3" value="option3" type="checkbox"> 3
        </label>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label col-lg-2" for="inputSuccess">Selects</label>
    <div class="col-lg-10">
        <select id="select-dropdown-num" class="form-control m-bot15">
            <option selected="selected">1</option>
            <option>2</option>
            <option>3</option>
            <option>4</option>
            <option>5</option>
        </select>

        <select multiple="multiple" class="form-control">
            <option>1</option>
            <option>2</option>
            <option>3</option>
            <option>4</option>
            <option>5</option>
        </select>
    </div>
</div>

```

(continues on next page)

(continued from previous page)

```

<div class="form-group">
  <label class="col-sm-2 control-label col-lg-2" for="inputSuccess">Column sizing</
→label>
  <div class="col-lg-10">
    <div class="row">
      <div class="col-lg-2">
        <input class="form-control" placeholder=".col-lg-2" type="text">
      </div>
      <div class="col-lg-3">
        <input class="form-control" placeholder=".col-lg-3" type="text">
      </div>
      <div class="col-lg-4">
        <input class="form-control" placeholder=".col-lg-4" type="text">
      </div>
    </div>
  </div>
</div>
</form>
</div>
</div>
</div>

```

Buttons

The screenshot displays a pink background with several form elements:

- Checkboxes and radios:** Four options are listed. The third option, "Option one is this and thatâ€™be sure to include why it's great", is selected with a radio button.
- Inline checkboxes:** Three checkboxes labeled 1, 2, and 3 are shown. Checkboxes 1 and 2 are checked.
- Selects:** A dropdown menu is open, showing a list of options: 1, 2, 3, and 4. Option 1 is currently selected.
- Column sizing:** Three text input fields are shown with placeholders ".col-lg-2", ".col-lg-3", and ".col-lg-4".

Fig. 3.13: Buttons

3.4.8 Input group

Output of below code is shown in Fig. 3.14,

```

<!-- Input groups -->
<h3>Input groups</h3>
<div class="row">
<div class="col-sm-12" style="background: pink">
<div class="panel-body">
  <form class="form-horizontal tasi-form" method="get">
    <div class="form-group">
      <label class="col-sm-2 control-label col-lg-2">Basic examples</label>
      <div class="col-lg-10">
        <div class="input-group m-bot15">
          <span class="input-group-addon">@</span>
          <input class="form-control" placeholder="Username" type="text">
        </div>

```

(continues on next page)

(continued from previous page)

```

        <div class="input-group m-bot15">
            <input class="form-control" type="text">
            <span class="input-group-addon">.00</span>
        </div>

        <div class="input-group m-bot15">
            <span class="input-group-addon">$</span>
            <input class="form-control" type="text">
            <span class="input-group-addon">.00</span>
        </div>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label col-lg-2">Sizing</label>
    <div class="col-lg-10">
        <div class="input-group input-group-lg m-bot15">
            <span class="input-group-addon">@</span>
            <input class="form-control input-lg" placeholder="Username" type="text">
        </div>

        <div class="input-group m-bot15">
            <span class="input-group-addon">@</span>
            <input class="form-control" placeholder="Username" type="text">
        </div>

        <div class="input-group input-group-sm m-bot15">
            <span class="input-group-addon">@</span>
            <input class="form-control" placeholder="Username" type="text">
        </div>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label col-lg-2">Checkboxes and radio</label>
    <div class="col-lg-10">
        <div class="input-group m-bot15">
            <span class="input-group-addon">
                <input type="checkbox">
            </span>
            <input class="form-control" type="text">
        </div>

        <div class="input-group m-bot15">
            <span class="input-group-addon">
                <input type="radio">
            </span>
            <input class="form-control" type="text">
        </div>
    </div>
</div>
<div class="form-group">
    <label class="col-sm-2 control-label col-lg-2">Button addons</label>
    <div class="col-lg-10">
        <div class="input-group m-bot15">
            <span class="input-group-btn">
                <button class="btn btn-white" type="button">Go!</button>
            </span>
            <input class="form-control" type="text">
        </div>
    </div>
</div>

```

(continues on next page)

(continued from previous page)

```

<div class="input-group m-bot15">
  <input class="form-control" type="text">
  <span class="input-group-btn">
    <button class="btn btn-white" type="button">Go!</button>
  </span>
</div>

<div class="input-group m-bot15">
  <div class="input-group-btn">
    <button type="button" class="btn btn-white dropdown-toggle" data-toggle="dropdown
→">Action <span class="caret"></span></button>
    <ul class="dropdown-menu">
      <li><a href="#">Action</a></li>
      <li><a href="#">Another action</a></li>
      <li><a href="#">Something else here</a></li>
      <li class="divider"></li>
      <li><a href="#">Separated link</a></li>
    </ul>
  </div>
  <input class="form-control" type="text">
</div>
<div class="input-group m-bot15">
  <input class="form-control" type="text">
  <div class="input-group-btn">
    <button type="button" class="btn btn-white dropdown-toggle" data-toggle="dropdown
→">Action <span class="caret"></span></button>
    <ul class="dropdown-menu pull-right">
      <li><a href="#">Action</a></li>
      <li><a href="#">Another action</a></li>
      <li><a href="#">Something else here</a></li>
      <li class="divider"></li>
      <li><a href="#">Separated link</a></li>
    </ul>
  </div>
</div>
</div>
</div>

<div class="form-group">
  <label class="col-sm-2 control-label col-lg-2">Segmented buttons</label>
  <div class="col-lg-10">
    <div class="input-group m-bot15">
      <div class="input-group-btn">
        <button tabindex="-1" class="btn btn-white" type="button">Action</button>
        <button tabindex="-1" data-toggle="dropdown" class="btn btn-white dropdown-toggle
→" type="button">
          <span class="caret"></span>
        </button>
        <ul role="menu" class="dropdown-menu">
          <li><a href="#">Action</a></li>
          <li><a href="#">Another action</a></li>
          <li><a href="#">Something else here</a></li>
          <li class="divider"></li>
          <li><a href="#">Separated link</a></li>
        </ul>
      </div>
      <input class="form-control" type="text">
    </div>
  </div>
</div>

```

(continues on next page)

(continued from previous page)

```

<div class="input-group m-bot15">
  <input class="form-control" type="text">
  <div class="input-group-btn">
    <button tabindex="-1" class="btn btn-white" type="button">Action</button>
    <button tabindex="-1" data-toggle="dropdown" class="btn btn-white dropdown-toggle
→ " type="button">
      <span class="caret"></span>
    </button>
    <ul role="menu" class="dropdown-menu pull-right">
      <li><a href="#">Action</a></li>
      <li><a href="#">Another action</a></li>
      <li><a href="#">Something else here</a></li>
      <li class="divider"></li>
      <li><a href="#">Separated link</a></li>
    </ul>
  </div>
</div>
</div>
</div>
</div>
</form>
</div>
</div>
</div>

```

Input groups

The screenshot displays a collection of Bootstrap input group examples. The 'Basic examples' section shows a text input with a preceding '@' icon and a currency input with a '\$' icon. The 'Sizing' section shows three text inputs of different heights (large, medium, and small). The 'Checkbox and radio' section shows a text input with a preceding checkbox and a text input with a preceding radio button. The 'Button addons' section shows a text input with a preceding 'Go!' button and a text input with a preceding 'Action -' button. The 'Segmented buttons' section shows a text input with a preceding 'Action -' button.

Fig. 3.14: Input groups

3.4.9 Navigation bar (navbar)

For navbar, one more 'css' file is added at Line 8. This CSS will create the space at the top for the 'navbar' so that it will not overlap with the contents of the HTML page. Further, Lines 13-44 adds various links to the navbar. Lastly, we can add HTML content below Line 46.

Output of below code is shown in Fig. 3.15,

```

1 <!DOCTYPE html>
2 <html>
3 <head>

```

(continues on next page)

```
4 <title>Bootstrap Tutorial</title>
5 <!-- CSS -->
6 <link href="asset/css/bootstrap.min.css" rel="stylesheet">
7 <!-- Add Custom CSS below -->
8 <link href="asset/css/theme.css" rel="stylesheet"> <!-- make space at the top for nav-bar -->
9 </head>
10
11 <body>
12
13 <div class="navbar navbar-inverse navbar-fixed-top">
14 <div class="container">
15 <div class="navbar-header">
16 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
17 collapse">
18 <span class="icon-bar"></span>
19 <span class="icon-bar"></span>
20 <span class="icon-bar"></span>
21 </button>
22 <a class="navbar-brand hidden-sm" href="http://pythondsp.readthedocs.io/">PythonDSP</a>
23 </div>
24 <div class="row">
25 <div class="navbar-collapse collapse">
26 <ul class="nav navbar-nav">
27
28 <li><a href="basic.html">HTML</a></li>
29 <li><a href="css.html">CSS</a></li>
30
31 <li class="dropdown">
32 <a href="#" class="dropdown-toggle" data-toggle="dropdown">More<b class="caret"></b></a>
33 <ul class="dropdown-menu">
34 <li><a href="bootstrap.html">Bootstrap</a></li>
35 <li><a href="js.html">Javascript</a></li>
36 <li><a href="jquery.html">jQuery</a></li>
37 </ul>
38 </li>
39
40 </ul>
41 </div>
42 </div><!-- /.nav-collapse -->
43 </div>
44 </div>
45
46 <!-- other codes here -->
47
48 <!-- Javascript -->
49 <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
50 <script src="asset/js/jquery-3.3.1.min.js"></script>
51 <script src="asset/js/bootstrap.min.js"></script>
52 </body>
53 </html>
```

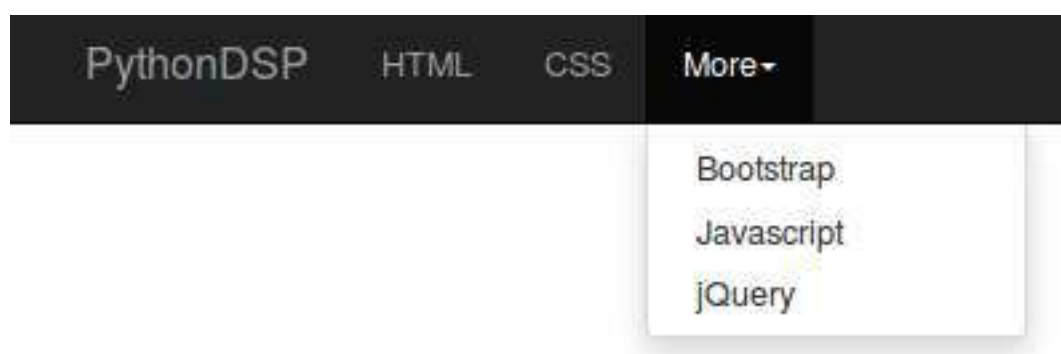


Fig. 3.15: Navigation bar (navbar)

Chapter 4

JavaScript

4.1 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.
- `;` is 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.

4.2 First code

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

4.2.1 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.

```
1  <!-- js.html -->
2
3  <!DOCTYPE html>
4  <html>
5  <head>
6      <title>JavaScript</title>
7  </head>
8  <body>
9
10
11     <script type="text/javascript">
12         document.write("Hello World from JavaScript!<br>");
13     </script>
14
15 </body>
16 </html>
```

- Below is the output on the HTML page,


```
Hello World from JavaScript!
```

4.2.1.1 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 'script tag' is used here.

```
1 // asset/js/my_javascript.js
2
3 document.write("Hello World from JavaScript!<br>");
```

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

```
1 <!-- js.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>JavaScript</title>
7 </head>
8 <body>
9
10
11
12     <!-- import JavaScript files here -->
13     <script src="asset/js/my_javascript.js"></script>
14
15 </body>
16 </html>
```

- 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.

4.3 Keywords, Datatypes, Variables and Operators

4.3.1 Keywords

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

Table 4.1: List of keywords

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

4.3.2 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’.

4.3.3 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,

```
1 <!-- js.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <title>JavaScript</title>
7 </head>
8 <body>
9
10  <p id='p_name'></p>
11
12  <!-- import JavaScript files here -->
13  <script src="asset/js/my_javascript.js"></script>
14
15 </body>
16 </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.

```

1 // asset/js/my_javascript
2
3
4 // print message on the webpage
5 document.write("Hello World from JavaScript!<br>");
6
7
8 // variable example
9 var your_name = "Meher";
10 var age = 20;
11 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

4.3.4 Operators

Various operators are shown in this section. The usage of some of these operators are shown in [Section 4.4](#).

4.3.4.1 Arithmetic operators

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

4.3.4.2 Assignment operators

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

4.3.4.3 Comparison operators

- == (compare only values not type)
- === (compare both values and type)
- !=
- >
- <
- >=
- <=

4.3.4.4 Conditional operator

- ?:

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

4.3.4.5 Logical operators

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

4.3.4.6 Bitwise operators

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

4.3.5 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
```

4.3.6 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
```

4.3.7 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
```

4.3.8 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 outputs of above code,

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

4.3.9 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](#)

4.3.10 Arrays

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

MEHER
krishna
Krishna
Krishna
~~Krishna~~
Krishna
Krishna
Krishna

Fig. 4.1: String styles

```
// 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("length of array: ", arr.length, "<br>");
```

Below are the output of above code,

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

4.4 Control structure, loops and functions

4.4.1 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>");
}
```

(continues on next page)

(continued from previous page)

```

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

```

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

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

4.4.2 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!
```

4.4.3 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
```

4.4.4 While loop

Below code prints the value from 0-4,

```

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

```

4.4.5 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
```

4.4.6 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>");
```

4.4.7 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
```

4.5 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 message, 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.

```
1 <!-- js.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>JavaScript</title>
7 </head>
8 <body>
9
10     <p id='p_name'></p>
11
12
13     <a href="JavaScript:void(0)", onclick="alert('Hello')">Say Hello</a><br> <!-- do not reload the page -->
14
15     <a href="", onclick="alert('Hello')">Say Bye</a><br> <!-- reload the page -->
16
17     <a href="JavaScript:void(0)", onclick="alertMessage('Bye via function')">Bye-function</a><br>
18
19     <a href="JavaScript:void(0)",
```

(continues on next page)

(continued from previous page)

```
20     onmouseover="this.style.color='red'",
21     onmouseleave="this.style.color='blue'",
22     onmouseup="this.text='Not clicked'",
23     onmousedown="this.text='You clicked me'">
24     Not clicked
25 </a><br>
26
27
28 <!-- import JavaScript files here -->
29 <script src="asset/js/my_javascript.js"></script>
30
31 </body>
32 </html>
```

4.6 Conclusion

In this tutorial, we have selected the element with certain ‘id’ using ‘getElementById’. Also, we saw the example of ‘event handling’. Although, we can use JavaScript for these operations, but it is better to use **jQuery** for these purposes, which is discussed in [Chapter 5](#). A good approach is to use the jQuery for ‘selectors’ & ‘event handling’ and then use various features of JavaScript which are described in [Section 4.4](#) to implement the logics.

Chapter 5

jQuery

5.1 Introduction

jQuery is the Javascript library.

5.1.1 Requirements

- First create a file 'jQuery.html'. Next, we need three files, i.e. one CSS (Line 9) and two Javascript (Lines 23-24), for using the jQuery in this tutorial. These files are saved inside the folder 'asset', which can be downloaded from the repository. Lines 9 and 24 are the bootstrap files, whereas Line 23 is the actual 'jQuery' file which is required to use the jQuery.

Note: Note that, the 'jQuery.js' files should be placed above the 'bootstrap.js' file, otherwise the bootstrap-javascript will not work.

```
1 <!-- jQuery.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>jQuery</title>
7
8     <!-- CSS -->
9     <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10    <!-- Add Custom styles below -->
11
12 </head>
13 <body>
14
15
16
17
18
19
20
21    <!-- Javascript -->
22    <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
23    <script src="asset/js/jquery-3.3.1.min.js"></script>
24    <script src="asset/js/bootstrap.min.js"></script>
25
26 </body>
27 </html>
```

- If we open this html file in the browser, the a blank page will be displayed with header 'jQuery'.

5.1.2 Add contents

Next add one jumbotron (Line 15-19) and few labels (Lines 21-26) in the html file as shown in [Fig. 5.1](#).

```
1 <!-- jQuery.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <title>jQuery</title>
7
8   <!-- CSS -->
9   <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10  <!-- Add Custom styles below -->
11
12 </head>
13 <body>
14
15   <div class="jumbotron">
16     <div class="col-md-offset-4 container">
17       <h1><b>jQuery</b></h1>
18     </div>
19   </div>
20
21   <span class="label label-lg label-default" id="l_default">Default</span>
22   <span class="label label-primary" id="l_primary">Primary</span>
23   <span class="label label-success" id="l_success">Success</span>
24   <span class="label label-info" id="l_info">Info</span>
25   <span class="label label-warning" id="l_warning">Warning</span>
26   <span class="label label-danger" id="l_danger">Danger</span>
27
28
29
30
31   <!-- Javascript -->
32   <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
33   <script src="asset/js/jquery-3.3.1.min.js"></script>
34   <script src="asset/js/bootstrap.min.js"></script>
35
36 </body>
37 </html>
```

jQuery

Default Primary Success Info Warning Danger

Fig. 5.1: Jumbotron and labels are added

5.2 jQuery examples

Following are the important parts for writing a jQuery,

- **Selectors** : to select the specific element (e.g. 'id', 'class' etc.) for operations.
- **Event handling** : perform some operations (hide, toggle etc.) on the selected items at some events (e.g. click, mouser over etc.).

In this section, we will see the working of of jQuery with the help of some examples. Then in [Section 5.3](#), we will see various methods to use the selectors to target the specific elements. Next, a list of operations are shown in [Table 5.2](#). Finally in [Section 5.5](#), a list of events are shown which can be used to initiate the operations.

5.2.1 Add jQuery code

Now, we can add the jQuery code as shown in Lines 36-40. Please note the following part,

- \$ sign is used to hit the jQuery. In the simple words, we can say that it execute the jQuery.
- The code is written between the following block. This block stops the jQuery to execute until the whole page is loaded, i.e. jQuery will be executed after completion of the page-download.

```
$(function() {
    // write code here
});
```

- The above block is a shorthand for below,

```
$(document).ready(function() {
    // write code here
});
```

Note: We can use multiple blocks, if we want to separate the codes; or we can write all the jQuerys within one block.

- Lastly, the below line hides the id "l_default" after 1000 ms, i.e. 'Default' label will not be displayed after 1000 ms as shown in [Fig. 5.2](#)

```
$("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
```

```

1  <!-- jQuery.html -->
2
3  <!DOCTYPE html>
4  <html>
5  <head>
6      <title>jQuery</title>
7
8      <!-- CSS -->
9      <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10     <!-- Add Custom styles below -->
11
12 </head>
13 <body>
14
15     <div class="jumbotron">
16         <div class="col-md-offset-4 container">
17             <h1><b>jQuery</b></h1>
18         </div>
19     </div>
20
21     <span class="label label-lg label-default" id="l_default">Default</span>
```

(continues on next page)

(continued from previous page)

```

22 <span class="label label-primary" id="l_primary">Primary</span>
23 <span class="label label-success" id="l_success">Success</span>
24 <span class="label label-info" id="l_info">Info</span>
25 <span class="label label-warning" id="l_warning">Warning</span>
26 <span class="label label-danger" id="l_danger">Danger</span>
27
28
29
30
31 <!-- Javascript -->
32 <!-- put jquery.js before bootstrap.min.js; and then add custom jquery -->
33 <script src="asset/js/jquery-3.3.1.min.js"></script>
34 <script src="asset/js/bootstrap.min.js"></script>
35
36 <script type="text/javascript">
37     $(function() {
38         $("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
39     });
40 </script>
41 </body>
42 </html>

```



Fig. 5.2: Label 'Default' is hidden after 1000 ms

Note: '#' is used to select the 'id'; whereas '.' is used to access the 'class'.

- In the below code, the label 'Danger' is removed; and "opacity='0.5'" is set for rest of the labels. Here, '.label' selects all the classes which have the name 'label' in it. Since all the entries have classes with name 'label', therefore all are selected and faded according to 'opacity='0.5'', as shown in Fig. 5.3.

```

1 <!-- jquery.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>jQuery</title>
7
8     <!-- CSS -->
9     <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10    <!-- Add Custom styles below -->
11
12 </head>
13 <body>
14
15    <div class="jumbotron">
16        <div class="col-md-offset-4 container">
17            <h1><b>jQuery</b></h1>
18        </div>
19    </div>
20
21    <span class="label label-lg label-default" id="l_default">Default</span>
22    <span class="label label-primary" id="l_primary">Primary</span>
23    <span class="label label-success" id="l_success">Success</span>

```

(continues on next page)

(continued from previous page)

```

24 <span class="label label-info" id="l_info">Info</span>
25 <span class="label label-warning" id="l_warning">Warning</span>
26 <span class="label label-danger" id="l_danger">Danger</span>
27
28
29
30
31 <!-- Javascript -->
32 <!-- put jQuery.js before bootstrap.min.js; and then add custom jquery -->
33 <script src="asset/js/jquery-3.3.1.min.js"></script>
34 <script src="asset/js/bootstrap.min.js"></script>
35
36 <script type="text/javascript">
37     $(function() {
38         $("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
39         $(".label-danger").hide(); // hide class = 'label' using . operator
40         $(".label").css({opacity:'0.5'}); // add opacity to all classes with name 'label'
41     });
42 </script>
43 </body>
44 </html>

```



Fig. 5.3: Faded labels

5.2.2 jQuery in separate file

It is better to write the jQuery in the separate files, therefore the codes are more manageable.

- For this, first write the code in the separate file 'my_jquery.js' as below,

Note: 'html' is used to change the content of the 'id' as shown in Line 5 below listing (see Fig. 5.4)

```

1 // asset/js/my_jquery.js
2
3 $(function() {
4     $("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
5     $(".label-danger").html("Shark is dangerous"); // change content using 'html'
6 });

```

- Next, import this file to 'html' as shown in Line 35 of below code,

```

1 <!-- jQuery.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>jQuery</title>
7
8     <!-- CSS -->
9     <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10    <!-- Add Custom styles below -->
11

```

(continues on next page)

(continued from previous page)

```
12 </head>
13 <body>
14
15     <div class="jumbotron">
16         <div class="col-md-offset-4 container">
17             <h1><b>jQuery</b></h1>
18         </div>
19     </div>
20
21     <span class="label label-lg label-default" id="l_default">Default</span>
22     <span class="label label-primary" id="l_primary">Primary</span>
23     <span class="label label-success" id="l_success">Success</span>
24     <span class="label label-info" id="l_info">Info</span>
25     <span class="label label-warning" id="l_warning">Warning</span>
26     <span class="label label-danger" id="l_danger">Danger</span>
27
28
29
30
31     <!-- Javascript -->
32     <!-- put jquery.js before bootstrap.min.js; and then add custom jquery -->
33     <script src="asset/js/jquery-3.3.1.min.js"></script>
34     <script src="asset/js/bootstrap.min.js"></script>
35     <script src="asset/js/my_jquery.js"></script>
36
37 </body>
38 </html>
```



Fig. 5.4: Changed content using 'html' option

5.2.3 Get input from user

Below is the complete html code, which we will use in this section,

Listing 5.1: complete HTML code

```
1 <!-- jquery.html -->
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <title>jQuery</title>
7
8     <!-- CSS -->
9     <link href="asset/css/bootstrap.min.css" rel="stylesheet">
10     <!-- Add Custom styles below -->
11
12 </head>
13 <body>
14
15     <div class="jumbotron">
16         <div class="col-md-offset-4 container">
17             <h1><b>jQuery</b></h1>
18         </div>
19     </div>
```

(continues on next page)

(continued from previous page)

```

20
21 <span class="label label-lg label-default" id="l_default">Default</span>
22 <span class="label label-primary" id="l_primary">Primary</span>
23 <span class="label label-success" id="l_success">Success</span>
24 <span class="label label-info" id="l_info">Info</span>
25 <span class="label label-warning" id="l_warning">Warning</span>
26 <span class="label label-danger" id="l_danger">Danger</span>
27
28 <br><br>
29 <span class="btn btn-sm btn-warning" id="b_warning">Toggle Warning Label</span>
30
31 <br><br>
32 <!-- label_id is custom attribute whose values are set according to id of above labels -->
33 <span class="btn btn-sm btn-default label-btn" label_id='l_default'>Toggle Default Label</span>
34 <span class="btn btn-sm btn-primary label-btn" label_id='l_primary'>Toggle Primary Label</span>
35 <span class="btn btn-sm btn-success label-btn" label_id='l_success'>Toggle Success Label</span>
36 <span class="btn btn-sm btn-info label-btn" label_id='l_info'>Toggle Info Label</span>
37 <span class="btn btn-sm btn-warning label-btn" label_id='l_warning'>Toggle Warning Label</span>
38 <span class="btn btn-sm btn-danger label-btn" label_id='l_danger'>Toggle Danger Label</span>
39
40
41 <!-- Javascript -->
42 <!-- put jquery.js before bootstrap.min.js; and then add custom jquery -->
43 <script src="asset/js/jquery-3.3.1.min.js"></script>
44 <script src="asset/js/bootstrap.min.js"></script>
45 <script src="asset/js/my_jquery.js"></script>
46
47 </body>
48 </html>

```

5.2.3.1 Toggle label on mouse click

In this section, the ‘mouse click’ on the button ‘Toggle Warning Label (Line 29)’ is read; and then the label ‘warning’ is toggled.

- The method ‘on’ waits for user’s input ‘click (i.e. mouse click)’.
- On mouse-click next query i.e. ‘toggle’ is executed which hides/unhide the ‘warning label’ as shown in [Fig. 5.5](#) - [Fig. 5.6](#).

```

1 // asset/js/my_jquery.js
2
3 // 'hide label' and 'change content'
4 $(function() {
5     $("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
6     $(".label-danger").html("Shark is dangerous"); // change content using 'html'
7 });
8
9
10 // toggle 'warning label' based on click on the button 'Toggle Warning Label'
11 $(function() {
12     $('#b_warning').on('click', function(){
13         $('#l_warning').toggle();
14     });
15 });

```

5.2.3.2 Toggle label based on the buttons

In previous section, we wrote the code which can toggle only one label. In this section, we will toggle multiple labels by writing one-generalized-jQuery as shown in [Fig. 5.7](#).



Fig. 5.5: Hide label 'warning'



Fig. 5.6: Show label 'warning'

- Lines 21-26 and 31-38 of Listing 5.1 are used for this section,
- Custom attributes "label_id=..." are added in each button which has the value which corresponds to one of the 'label's id'. e.g. 'label_id' for Line 33 (i.e. l_default) is same as 'id' at Line 21.
- Next, in the below jQuery, we saved the 'label_id' in a variable (see Line 21) and then the variable is used to toggle the labels.

Listing 5.2: Toggle the label based on the button-clicked

```

1 // asset/js/my_jquery.js
2
3 // 'hide label' and 'change content'
4 $(function() {
5     $("#l_default").hide(1000); // hide id = 'l_default' using # operator after 300 ms
6     $(".label-danger").html("Shark is dangerous"); // change content using 'html'
7 });
8
9
10 // toggle 'warning label' based on click on the button 'Toggle Warning Label'
11 $(function() {
12     $('#b_warning').on('click', function(){
13         $('#l_warning').toggle();
14     });
15 });
16
17
18 // toggle panels according to the clicked-buttons
19 $(function(){
20     $('.label-btn').on('click', function(){ // select class 'label-btn'
21         var labelId = $(this).attr('label_id'); // store the attribute value of 'label_id'

```

(continues on next page)

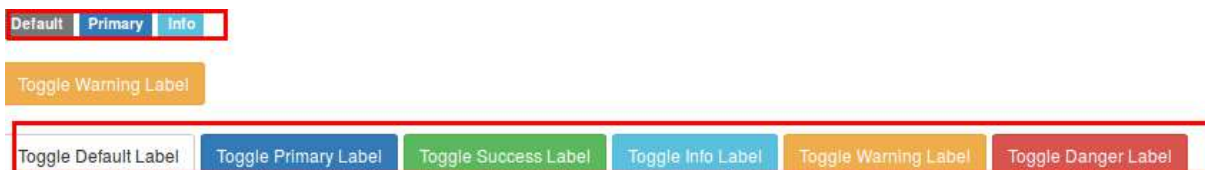


Fig. 5.7: Toggle labels based on button-clicked

(continued from previous page)

```

22     $('#'+ labelId).toggle(); // toggle the label with id stored in 'label_id'
23   });
24 });

```

5.3 Selectors

In this section, we will use following HTML to understand the various combination of the selectors which are summarized in [Table 5.1](#),

```

<!DOCTYPE html>
<html>
<head>
  <title>jQuery Selectors</title>
</head>
<body>
  <p>jQuery Selectors</p>

  <p class="foo"> Paragraph with class.</p>
  <p><span>Paragraph with span.</span></p>

  <p id="bar">Paragraph with ID. <span class="foo">Span with class and inside the paragraph-with-id.</
→span></p>

  <div>
    <p my-id="my_para">Paragraph (with custom id) inside the div</p>
    <p my-id="my_para2"><span>Span inside the paragraph (with custom id) inside the div</span></p>
  </div>

  <!-- Javascript -->
  <script src="asset/js/jquery-3.3.1.min.js"></script>

</body>
</html>

```

Note:

- Press ‘ctrl+shift+k’ (or go to Tools->Web Developer->Web Console) to open the web-console in **Firefox** and click on ‘console’ and type the ‘jQuery’ commands there.
- Press ‘ctrl+shift+J’ (or go to Tools->Javascript Console) in ‘**Chrome**’ and type the jQuery commands there.
- In this tutorial, the output of the ‘Firefox’ are shown.

5.3.1 Select elements

5.3.1.1 Select HTML elements

HTML elements, e.g. div, p, span and table etc., can be selected by writing them with the quotation mark as shown in below codes.

- Here, ‘\$(‘p’)’ is the command, whereas the lines below it are the outputs.
- The outputs show the ‘selected elements’; and the total number of selected elements i.e. ‘length: 6’.
- If we put the cursor to one of the outputs, then it will highlight the corresponding elements in the html as shown in [Fig. 5.8](#).

```
$('#p')
{...}
0: <p>
1: <p class="foo">
2: <p>
3: <p id="bar">
4: <p my-id="my_para">
5: <p my-id="my_para2">
length: 6
```

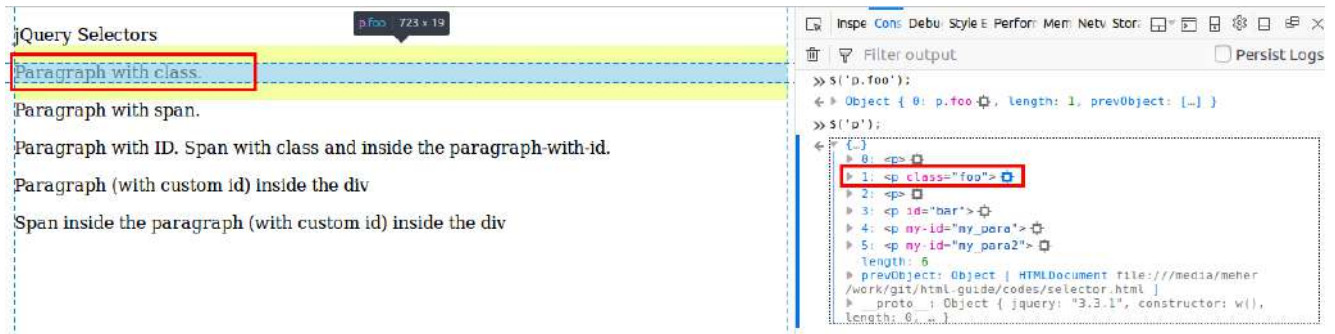


Fig. 5.8: Output of \$('#p'); (see red-squares)

- Similarly, we can select 'div' elements

```
$('#div')
{...}
0: <div>
length: 1
```

5.3.1.2 Select class

Class can be selected using '.' operator. In below code, class 'foo' is selected.

```
$(".foo")
{...}
0: <p class="foo">
1: <span class="foo">
length: 2
```

5.3.1.3 Select ID

ID can be selected using '#' operator. In the below code, id 'bar' is selected.

```
$('#bar')
[...]
```

```
0: <p id="bar">
length: 1
```

5.3.1.4 Combining selectors

We can select a element with specific class name using '.' operator as shown in below code. Here element 'span' with class 'foo' is selected,

```
$("span.foo")
{...}
0: <span class="foo">
length: 1
```

5.3.1.5 Multiple selectors

Use ',' to select different types of elements. In the below code, "paragraph with id 'bar'" and "elements with class 'foo'" are selected ,

```
$('p#bar, .foo')
{...}
0: <p class="foo">
1: <p id="bar">
2: <span class="foo">
length: 3
```

5.3.1.6 Select descendant element

Select 'span' inside the 'p' (use space between 'span' and 'p'),

```
$('p span')
{...}
0: <span>
1: <span class="foo">
2: <span>
length: 3
```

5.3.1.7 Select child element

The '>' is used to select the child of an element.

```
$('div>span')
{...}
length: 0
```

Note: There is one element inside 'div' but its path is 'div->p->span' therefore 'span' is a child of 'p' (not of 'div'). But 'span' is the descendant of 'div' therefore can be accessed using space,

```
$('div span')
{...}
0: <span>
length: 1
```

5.3.1.8 Attribute selector

Custom attributes can be selected as below,

```
$('[my-id = my_para]')
{...}
0: <p my-id="my_para">
length: 1
```

(continues on next page)

(continued from previous page)

```
$('#p[my-id != my_para]') // select p with 'my-id != my_para'
{...}
0: <p>
1: <p class="foo">
2: <p>
3: <p id="bar">
4: <p my-id="my_para2">
length: 5

$('#p[class = foo]')
{...}
0: <p class="foo">
length: 1
```

5.3.2 Filters

Various filter options (e.g. filter, first, last and has etc.) can be used after ':' as shown below,

```
$('#p:first') // select first element
{...}
0: p

$('#p:last') // last element
{...}
0: <p my-id="my_para2">
length: 1

$('#p:not(.foo)') // 'p' which does not have class 'foo'
{...}
0: <p>
1: <p>
2: <p id="bar">
3: <p my-id="my_para">
4: <p my-id="my_para2">
length: 5

$('#p:has(span)') // 'p' which has 'span' inside it
{...}
0: <p>
1: <p id="bar">
2: <p my-id="my_para2">
length: 3

$('#p:hidden') // hidden 'p'
{...}
length: 0

$('#p:visible') // visible 'p'
{...}
0: <p>
1: <p class="foo">
2: <p>
3: <p id="bar">
4: <p my-id="my_para">
5: <p my-id="my_para2">
length: 6

$('#p:contains(with class)') // paragraph which contains class inside it
```

(continues on next page)

(continued from previous page)

```

{...}
0: <p class="foo">
1: <p id="bar">
length: 2

$('p').filter(".foo") // same as p.foo
{...}
0: <p class="foo">
length: 1

$('p').not(".foo") // p with not class foo
{...}
0: <p>
1: <p>
2: <p id="bar">
3: <p my-id="my_para">
4: <p my-id="my_para2">
length: 5

$('p').last() // last element of 'p'
{...}
0: <p my-id="my_para2">
length: 1

$('p').has('span') // p with span
{...}
0: <p>
1: <p id="bar">
2: <p my-id="my_para2">
length: 3

$('p').is('.foo') // does 'p' contain 'foo'
true

$('p').children() // children of 'p'
{...}
0: <span>
1: <span class="foo">
2: <span>
length: 3

$('p').children(".foo") // child of 'p' with class 'foo'
{...}
0: <span class="foo">
length: 1

```

Table 5.1: Selectors

Command	Description	example
<code>\$(‘elementName’)</code>	select the tags (p, body, table)	<code>\$(‘p’)</code> , <code>\$(‘body’)</code>
<code>\$(‘.className’)</code>	select element with ‘className’	<code>\$(“.foo”)</code>
<code>\$(‘elementName.className’)</code>	element with className	<code>\$(“span.foo”)</code>
<code>\$(selector1, selector2)</code>	select selector1 and selector2	<code>\$(‘p#bar, .foo’)</code>
<code>\$(selector1 selector2)</code>	select selector2 inside selector1	<code>\$(‘p span’)</code>
<code>\$(selector1>selector2)</code>	select selector2 which is child of selector1	<code>\$(‘div>span’)</code>
<code>\$(attributName = someName)</code>	select attribute with someName	<code>\$(‘[my-id = my_para]’)</code>
<code>\$p(attributName = someName)</code>	select ‘p’ with attribute with someName	<code>\$(‘p[my-id = my_para]’)</code>
See Section 5.3.2 for filters		

5.4 Operations

In [Listing 5.2](#) we saw some operations e.g. ‘hide()’ and ‘toggle()’. [Table 5.2](#) shows some more such operations.

Table 5.2: Operations

Operations			
hide()	hide(speed)	fadeIn(speed)	fadeOut(speed)
show()	show(speed)	slideDown(speed)	slideToggle(speed)
slideUp(speed)	toggle()	toggle(speed)	toggle(switch)

5.5 Event handling

In [Listing 5.2](#) we saw one example of event handling, where certain operations are performed on the event ‘mouse click’. [Table 5.2](#) shows a list of some more events.

Table 5.3: Event handle

Events			
click	keydown	keypress	keyup
mousedown	mouseover		