Full Stack Web Development with Python and DJango

HTML

Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

Agenda

1. HTML
2. JS
3. CSS
4. Bootstrap
5. JQuery
6. Python
7. Django
8. SQLite|Oracle|MySQL

Web Application:

The applications which will provide services over the web are called web applications.

Eg gmail.com, facebook.com, durgasoftvideos.com et

Every web application contains 2 main components

1. Front-End
2. Back-End

1) Front-End:

It represents what user is seeing on the website
We can develop Front-End content by using the following technologies:
HTML, JS, CSS, JQuery and BootStrap

JQuery and Bootstrap are advanced front-end technologies, which are developed by using HTML, CSS and JavaScript only.

HTML: HyperText Markup Language
Every web application should contain HTML. i.e HTML is the mandatory technology for web development.It represents structure of web page

CSS: Cascading Style Sheets
It is optional technology; still every web application contains CSS.
The main objective of CSS is to add styles to the HTML Pages like colors,fonts,borders etc.

Java Script:

It allows to add interactivity to the web application including programming logic.

The main objective of Java Script is to add functionality to the HTML Pages. ie to add dynamic nature to the HTML Pages.

HTML===>Meant for Static Responses
HTML+JS==>Meant for Dynamic Responses

Eg 1: To display "Welcome to DURGASOFT" response to the end user only HTML is enough, because it is static response.

Eg 2: To display current server date and time to the end user, only HTML is not enough we required to use some extra technology like JavaScript,JSP,ASP,PHP etc as it is dynamic response.

Static Response vs Dynamic Response:

If the response is not varied from time to time and person to person then it is considered as static response.

Eg login page of gmail
home page of icici bank

If the response is varied from time to time and person to person then it is considered as dynamic response.

Eg inbox page of gmail
balance page of icicibank

2) Back End:

It is the technology used to decide what to show to the end user on the Front-End.
ie Backend is responsible to generate required response to the end user,which is displayed by the Front-End.

Back-End has 3 important components:

1. The Language like Java, Python etc
2. The Framework like DJango, Pyramid, Flask etc
3. The Database like SQLite, Oralce, MySQL etc

For the Backend language Python is the best choice because of the following reasons: Simple and easy to learn, libraries and concise code.

For the Framework DJango is the best choice because it is Fast, Secure and Scalable. Django is the most popular web application framework for Python.

DJango provides inbuilt database which is nothing but SQLite, which is the best choice for database.

The following are various popular web applications which are developed by using Python and DJango

YouTube, Dropbox, Quora, Instagram, Reddit, Yahoo Maps etc

HTML Basics:

HTML stands for HyperText Markup language.
This is the most basic building block of every web application.Without using HTML we cannot build web applications. It is the mandotory technology.

We can use CSS to sstyle HTML Pages.
We can use Java Script to add functionality to the HTML pages.

In general we will add django template tags to HTML for generating dynamic content based on our requirement.

Strcutre of HTML Page:

Every HTML page contains 2 parts
1. Head
2. Body

Head contains meta data like title of the page, keywords etc. CSS files and Java Script files information we have to specify in the Head Part only.

Body contains actual content.

```
1)  <!doctype html>// to indicate that it is HTML page
2)  <html >
3)   <head>
4)     Meta Data like keywords,author,title...
5)     css files information
6)     js files information
7)   </head>
8)   <body>
9)      Actual Data
10) </body>
11) </html>
```

HTML Comment:

<!-- Anything here is considered as Comment -->

Title:

```
1)  <head>
2)    <title>Basic HTML for DJango Classes</title>
3)  </head>
```

Heading Tags:

HTML supports 6 heading tags
<h1>,<h2>,....

<h1>This is Heading1</h1>

Paragraph tag: :

We can use this tag to represent paragraph of text.
 This is first paragraph

case-1:

```
1)  <p>This is First Line
2)      This is Second Line
3)      This is Third Line
4)      This is Four Line
5)   </p>
```

Total Data will come in a single line, because we are using only one paragraph tag.

Case-2:

```
1)    <p>This is First Line</p>
2)   <p>This is Second Line</p>
3)   <p>This is Third Line</p>
4)   <p>This is Fourth Line</p>
```

Output will come in 4 lines

case-3:

This is First Line This is Second Line

output will come in multiple lines

Note: In HTML indentation is not important but tags are important. Blocking also takes care by html tags only.

```
1)  <body>
2)    <h1>This is HTML Demo Class</h1>
3)            <p>This is First Line</p>
4)     <p>This is Second Line</p>
5)            <p>This is Third Line</p>
6)     <p>This is Fourth Line</p>
7)  </body>
```

Bold and Italic:

legacy tags:

 for bold
 for italic
These are old (lEgacy)html tags and not recommended to use.

Eg This is First Line

advanced tags:

We can use the following HTML 5 advanced tags for bold and italic

  for strong text(bold)
 for emphasis (italic)

Eg
This is Second Line

HTML Lists:

There are 2 types of lists

1. ordered list
2. unordered list

1. ordered list:

All list items will be displayed with numbers

```
1)  <ol>
2)        <li>Chicken</li>
3)        <li>Mutton</li>
4)        <li>Fish</li>
```

```
5)        <li>Beer</li>
6)  </ol>
```

Output:

1. Chicken
2. Mutton
3. Fish
4. Beer

2. unordered list:

Instead of numbers bullet symbol will come. Here order is not important.

```
1)      <ul>
2)         <li>Chicken</li>
3)         <li>Mutton</li>
4)         <li>Fish</li>
5)         <li>Beer</li>
6)  </ul>
```

Nested Lists:

We can take list inside list, which are nothing but nested lists.

Eg

```
1)      <ol>
2)         <li>Chicken</li>
3)         <li>Mutton</li>
4)         <li>Fish</li>
5)         <li>Beer</li>
6)         <ul>
7)            <li>KF</li>
8)            <li>KO</li>
9)            <li>RC</li>
10)        </ul>
11) </ol>
```

Note: For outer and inner lists we can take both ordered and unordered lists

Eg we can take unordered list inside ordered list

Div and Span Tags:

We can use div and span tags to group related tags into a single unit.

In general we can use these tags with CSS.

Eg

```
1)  <div class="groupone">
2)     <h1> Group One Content</h1>
3)     <p>This division tag helpful to style a group of
4)        html tags with css</p>
5)  </div>
```

div means division

 tag is exactly same as division tag except that it is inline. i.e to define group within the line of text we can use tag.

This division tag helpful to style a group of html tags with css

 will work for group of lines where as will work within the line.

Note: and tags are helpful only for styling html. Hence they will always work with css only.

Attributes:

HTML Attributes will provide extra information to HTML tags.

To insert image in the html page, src attribute specify location of the image to the <img> tag.

Setting Image inside HTML:

<img src="guido.jpg" alt="OOPS Image is missing">

src means source where we have to specify the image source(complete location). We can take image address from the google also.

alt means alternate. If image is missing then broken link image will display.In this case if we want to display some meaningful text information then we should go for alt attribute.

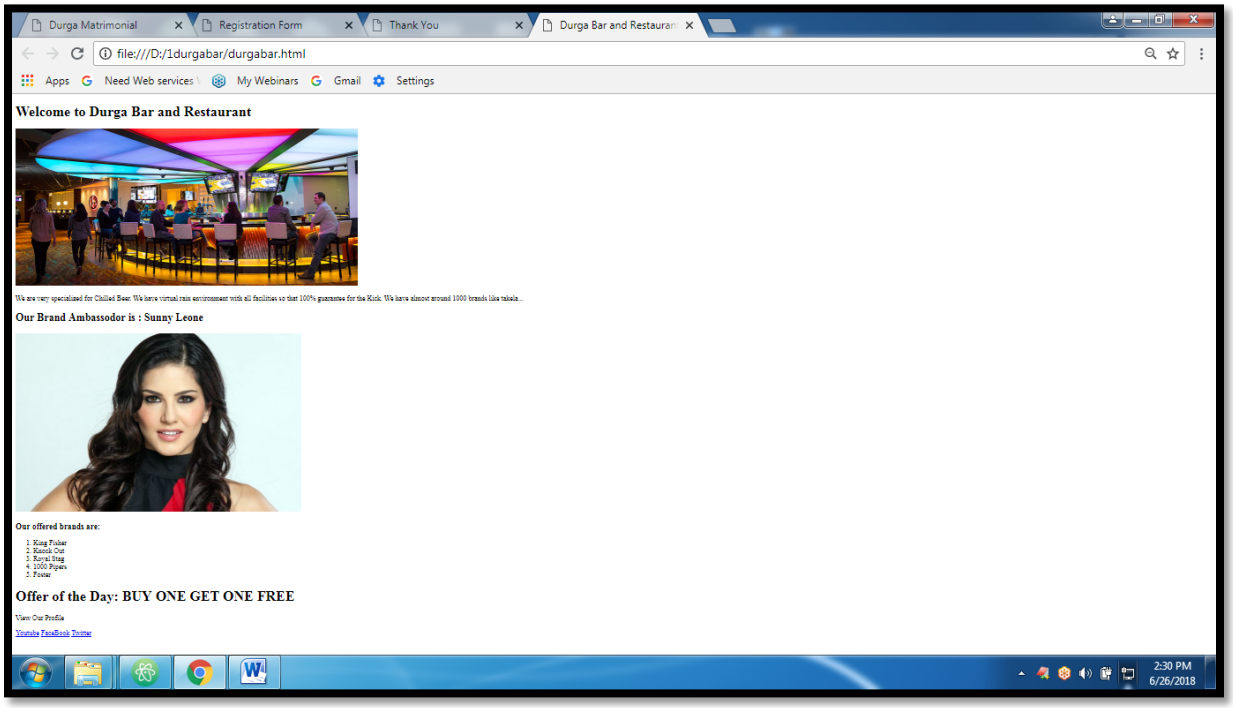
Note: We have to open the tag and we are not responsible to close the tag,such type of tags are called self closing tags.

Eg <img> tag

Creating Hyperlinks by using anchor tag: <a>:

```
<a href="second.html">Click Here to go Second Page</a>
<a href="https://facebook.com">FaceBook</a>
```

Durga Bar and Restaurant Application:




```
1)  <!doctype html>
2)  <html lang="en">
3)   <head>
4)    <meta charset="UTF-8">
5)    <meta name="Generator" content="EditPlus®">
6)    <meta name="Author" content="">
7)    <meta name="Keywords" content="">
8)    <meta name="Description" content="">
9)    <title>Durga Bar and Restaurant</title>
10)  </head>
11)  <body>
12)   <h1>Welcome to Durga Bar and Restaurant</h1>
13)   <img src="dbar.jpg" alt="image missing upload soon">
14)   <p>We are very specialized for Chilled Beer. We have virtual rain environment with all fa
      cilities so that 100% guarantee for the Kick</p>
15)   <h2>Our Brand Ambassodor is : Sunny Leone</h2>
16)   <img src="sunny.jpg" alt="missing image"/>
17)   <h3>Our offered Products:</h3>
18)   <ol>
19)     <li>KF</li>
20)     <li>KO</li>
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
21)     <li>RC</li>
22)     <li>FO</li>
23)    </ol>
24)    <h1>Offer of the Day: Buy One Get One FREE</h1>
25)    <p>View Our Profile:</p>
26)    <a href="http://youtube.com">YouTube</a>
27)    <a href="http://twitter.com">Twitter</a>
28)    <a href="https://facebook.com">Facebook</a>
29)  </body>
30) </html>
```

Table Creation:

We can use
<table> to create table
<thead> to specify head row
<th> to specify column data in head row(column name)
<tr> to insert row in the table
<td> to specify column data in the row/record

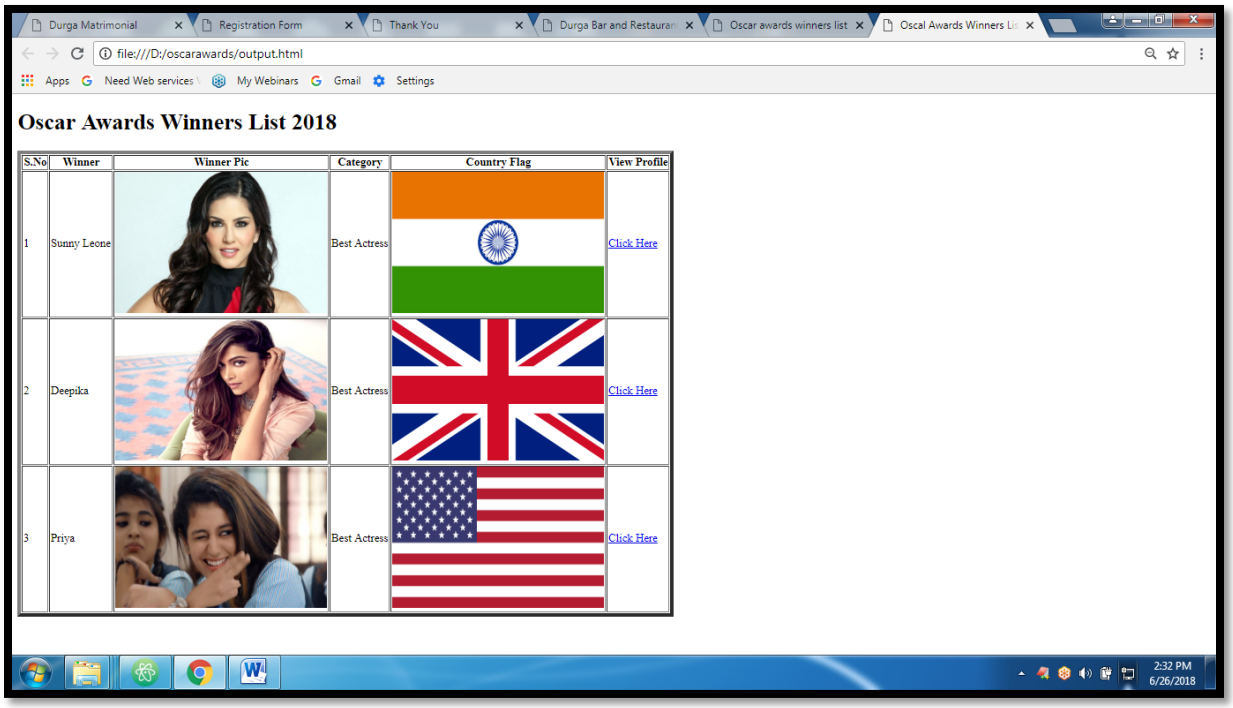
We can use border attribute inside <table> tag

Eg

```
1)  <table border="1">
2)      <thead>
3)         <th>ENO</th>
4)         <th>ENAME</th>
5)         <th>ESAL</th>
6)         <th>EADDR</th>
7)      </thead>
8)      <tr>
9)        <td>100</td>
10)       <td>DURGA</td>
11)       <td>1000</td>
12)       <td>Hyd</td>
13)     </tr>
14)     <tr>
15)       <td>200</td>
16)       <td>SUNNY</td>
17)       <td>2000</td>
18)       <td>Mumbai</td>
19)     </tr>
20)     <tr>
21)       <td>300</td>
22)       <td>Bunny</td>
23)       <td>3000</td>
24)       <td>Chennai</td>
25)     </tr>
```

```
26)  </table>
```

Oscar Awards Winners List 2018 Application(Assignment on tables):




```
1)  <!doctype html>
2)  <html lang="en">
3)   <head>
4)    <title>Oscar awards winners list</title>
5)   </head>
6)   <body>
7)     <h1> Oscar Awards Winners List 2018</h1>
8)    <table border="1">
9)       <thead>
10)       <th>S.No</th>
11)       <th>Winner</th>
12)       <th>Winner Pic</th>
13)       <th>CatEgory</th>
14)       <th>Country Log</th>
15)       <th>View Profile</th>
16)      </thead>
17)      <tr>
18)       <td>1</td>
19)       <td>Sunny</td>
20)       <td><img src="sunny.jpg" alt="Image not available"></td>
21)       <td>Best Actress</td>
22)       <td><img src="https://upload.wikimedia.org/wikipedia/en/thumb/4/41/Flag_of_Indi
      a.svg/255px-Flag_of_India.svg.png" alt="Image not available"></td>
23)       <td><a href="http://youtube.com/durgasoftware">View Info</a></td>
```

```
24)     </tr>
25)   </table>
26)  </body>
27) </html>
```

Creation of HTML Forms:

As the part of web application development, we have to develop several forms like login form, registration form etc

We can create Html form by using <form> tag.

```
<form  class="" action="" method="">
  .....
</form>
```

Within the form to collect end user input, we have to use <input> tag.
This <input> tag will play very important role in form creation.

syntax: <input type="" name="" value=""/>

type attribute can be used to specify the type of input end user has to provide. The main important types are:

- **text**
- **email**
- **password**
- **color**
- **submit**
- **checkbox**
- **radio**
- **etc**

name attribute represents the name of input tag. By using this name,in the next target page we can access end user provided input value.

value attribute represents default value will be displayed in the form.

Eg

```
<input type="text" name="username" value="Enter User Name"/>
<input type="email" name="mailid" value=""/>
<input type="password" name="pwd" value=""/>
<input type="checkbox" name="course" value=""/>
<input type="radio" name="married" value=""/>
```

To provide default value it is highly recommended to use placeholder attribute because end user is not required to delete defualt value while entering data.

Eg: <input type="text" name="username" placeholder="Enter User Name"/>

Creation of Labels for HTML Elements:

We can define Label Text for our HTML Elements like Radio Buttons, Text Box etc by using <label> tag.

Syntax:<label for="name">Enter Name:</label>

Eg1:

```
<p>Enter Name:</p>
<input type="text" name="username" placeholder="Enter Name">
```

The result looks like
diagram

In this case there is no relation between text box and data.

To link data to text box,we have to use <label> tag.

```
<label for="name">Enter Name:</label>
<input id="name" type='text' name='username' placeholder='Name to Contact' >
```

The result looks like
Diagram

required attribute:

If end user compulsory should required to provide input value then we should go for required attribute.

Eg

```
<input id="name" type='text' name='username' placeholder='Name to Contact' required>
```

action attribute:

once we fill the form and click submit, then to which page it will go is decided by action attribute.The value of action attribute can be either local resource or web url.

Eg

```
<form action="target.html" >
<form action="https://facebook.com" >
```

Demo program:

Diagram

test.html

```
1)  <!doctype html>
2)  <html lang="en">
3)   <head>
4)    <title>Form Input</title>
5)   </head>
6)   <body>
7)    <h1>User Contact Form</h1>
8)    <form action="target.html">
9)     <label for="name">Enter Name:</label>
10)    <input id="name" type='text' name='username' placeholder='Name to Contact' required
      >
11)    <input type="submit" value="ClickToContact">
12)   </form>
13)  </body>
14) </html>
```

target.html:

```
1)  <!doctype html>
2)  <html lang="en">
3)   <head>
4)    <title>Target HTML</title>
5)   </head>
6)   <body>
7)   <h1>Thanks for Confirmation</h1>
8)   </body>
9)  </html>
```

Implementing Radio Buttons:

```
1)  <h3>Are you Married:</h3>
2)  <label for='ma'>Yes</label>
3)  <input id='ma' type='radio' name="married" value="">
4)  <label for='ma1'>No</label>
5)  <input id='ma1' type='radio' name="married" value="">
```

Eg2:

```
1)  <h3>Gender:</h3>
2)  <label for='m'>Male</label>
3)  <input id='m' type='radio' name="gender" value="Male">
```

```
4)  <label for='f'>Female</label>
5)  <input id='f' type='radio' name="gender" value="Female">
6)  <label for='t'>Transgender</label>
7)  <input id='t' type='radio' name="gender" value="Transgender">
```

How to implement Drop down box/select box:

```
1)  <h3>Please Select Your Payment Mode:</h3>
2)  <select name='pmode'>
3)     <option value='ccard'>Credit Card</option>
4)     <option value='dcard'>Dedit Card</option>
5)     <option value='ppal'>Paypal</option>
6)     <option value='otransfer'>OnlineTransfer</option>
7)  </select>
```

textarea element:

```
1)  <h2>Enter Your FeedBack:</h2>
2)  <textarea name="feedback" rows="8" cols="80">
```

checkbox:

```
1)  <form>
2)        <h2>Choose Your Known Languages:</h2>
3)     <input type="checkbox"  name="languages" value="english" checked> English<br>
4)     <input type="checkbox"  name="languages" value="telugu"> Telugu<br>
5)     <input type="checkbox"  name="languages" value="hindi"> Hindi<br>
6)     <input type="checkbox"  name="languages" value="tamil"> Tamil<br>
7)  </form>
```

How to include spaces in html:

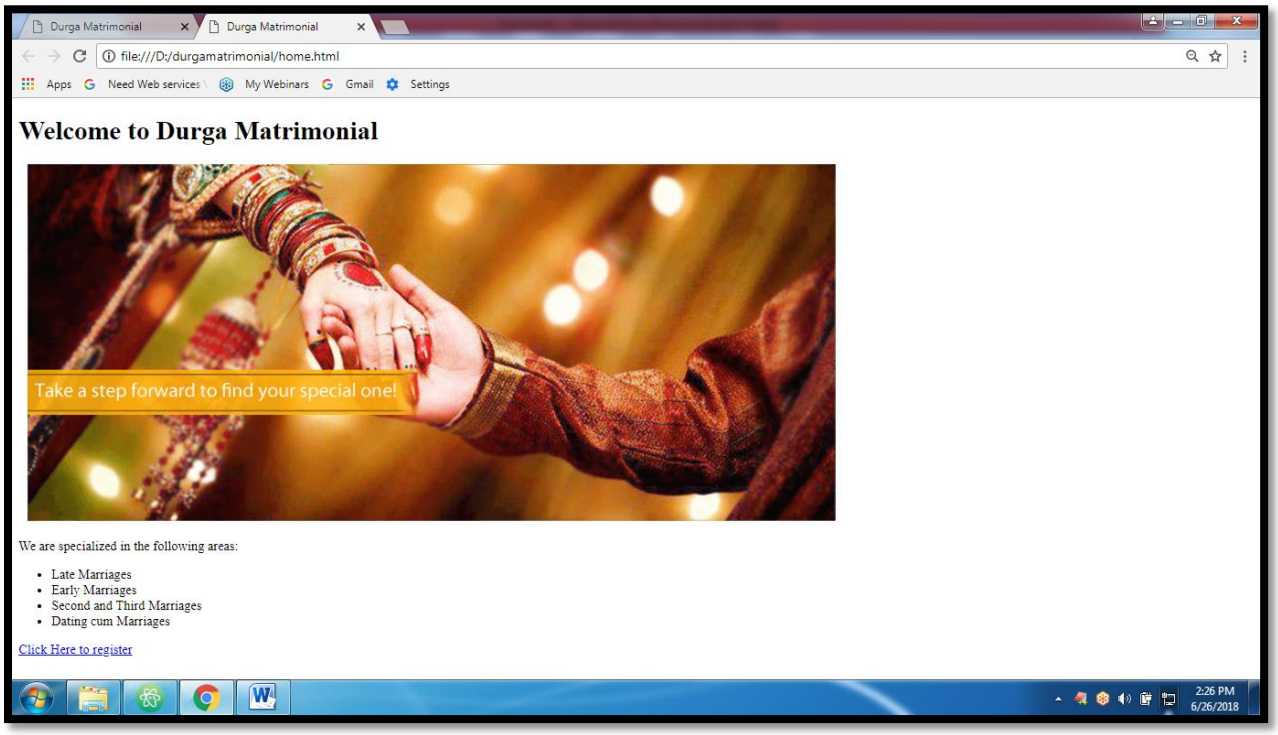
1. Type "nbsp" to add a single space.
2. Type "ensp" to add 2 spaces.
3. Type "emsp" to add 4 spaces.
4. Use the non-breaking space (nbsp) 4 times to insert a tab.
5. Use "br" to add a line break.

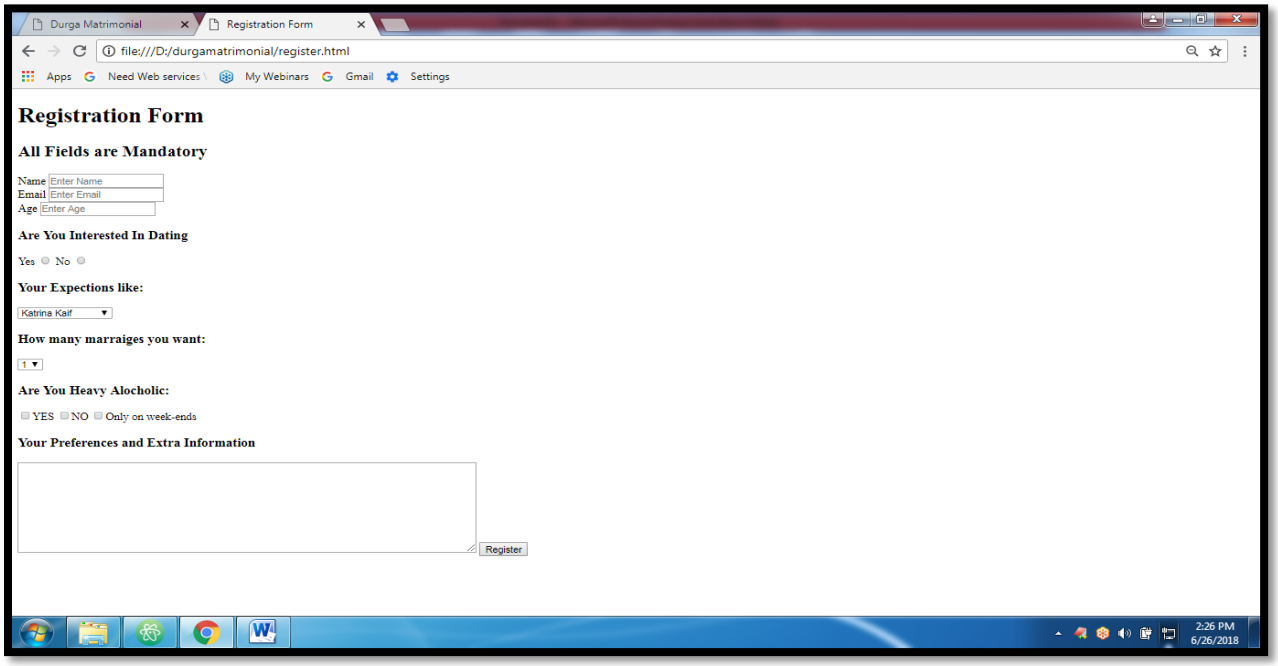
ATOM IDE/Editor:

https://atom.io/

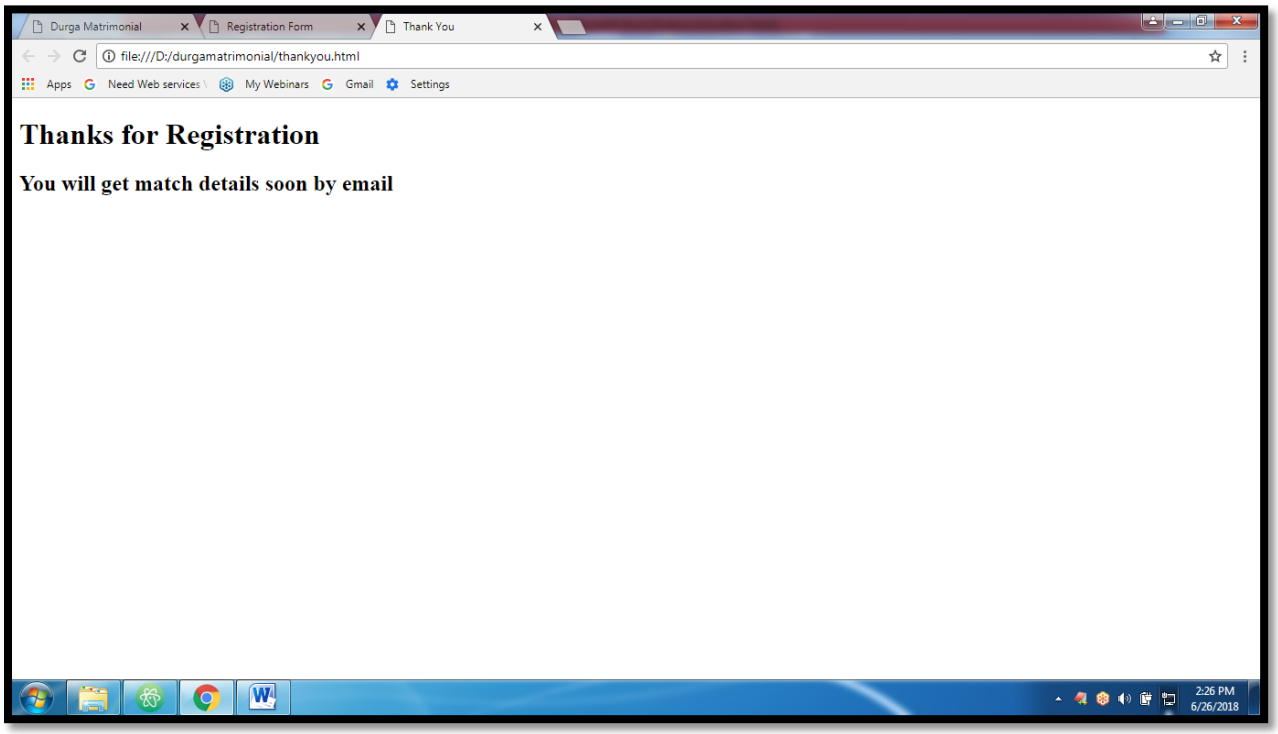
freeware
open source
supports cross platform
several auto completion short-cuts
etc

Durga Matrimonial Application:





DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com



home.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <link rel="stylesheet" href="form.css">
6)      <title>Durga Matrimonial</title>
7)    </head>
8)    <body>
9)      <h1>Welcome to Durga Matrimonial</h1>
10)     <img src="mat.png" alt="">
11)     <p>We are specialized in the following areas:</p>
12)     <ul>
13)       <li>Late Marriages</li>
14)       <li>Early Marriages</li>
15)       <li>Second and Third Marriages</li>
16)       <li>Dating cum Marriages</li>
17)     </ul>
18)     <a href="rEgister.html">Click Here to rEgister</a>
19)   </body>
20) </html>
```

register.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>REgistration Form</title>
6)    </head>
7)    <body>
8)      <h1>Registration Form</h1>
9)      <h2>All Fields are Mandatory</h2>
10)     <form class="" action="thankyou.html" method="post">
11)       <label for="name">Name</label>
12)       <input id="name" type="text" name="name" placeholder="Enter Name" required><br
      >
13)       <label for="mail">Email</label>
14)       <input id="mail" type="email" name="mail" placeholder="Enter Email" required><br>
15)       <label for="age">Age</label>
16)       <input id="age" type="text" name="age" placeholder="Enter Age" required>
17)       <h3>Are You Interested In Dating</h3>
18)       <label for="dat">Yes</label>
19)       <input id="dat" type="radio" name="dating" value="">
20)       <label for="dat1">No</label>
21)       <input id="dat1" type="radio" name="dating" value="">
22)       <h3>Your Expections like:</h3>
23)       <select class="" name="expe">
24)         <option value="katrina">Katrina Kaif</option>
25)         <option value="kareena">Kareena Kapoor</option>
26)         <option value="sunny">Sunny Leone</option>
27)         <option value="mallika">Mallika Sherawat</option>
28)       </select>
29)       <h3>How many marraiges you want:</h3>
30)       <select class="" name="nom">
31)         <option value="One">1</option>
32)         <option value="Two">2</option>
33)         <option value="Three">3</option>
34)         <option value="Four">4</option>
35)       </select>
36)       <h3>Are You Heavy Alocholic:</h3>
37)       <input type="checkbox" name="Alocholic" value="yes">YES
38)       <input type="checkbox" name="Alocholic" value="no">NO
39)       <input type="checkbox" name="Alocholic" value="week-end">Only on week-ends
40)       <h3>Your Preferences and Extra Information</h3>
41)        <textarea name="pref" rows="8" cols="80"></textarea>
42)       <input type="submit" name="" value="Register">
43)     </form>
44)   </body>
45) </html>
```

thankyou.html:

```
1)      <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Thank You</title>
6)    </head>
7)    <body>
8)      <h1>Thanks for REgistration</h1>
9)      <h2>You will get match details soon by email</h2>
10)   </body>
11) </html>
```

form.css:

```
1)  h1{
2)    color:red;
3)  }
4)  ul{
5)    color:blue;
6)  }
```

google drive link: https://goo.gl/6iq64Y

How to declare multiple submit buttons in the form:

Within the form we can define multiple submit buttons. But based on value we can implement corresponding action in the back-end.

Eg

```
1)  <form action="target">
2)    ......
3)    <input type="submit" name="action" value="update"/>
4)    <input type="submit" name="action" value="edit"/>
5)    <input type="submit" name="action" value="delete"/>
6)  </form>
```

In the target, based on submit button value we have to implement corresponding action.

```
1)  choice = req.getParameter("action")
2)  if choice=="update":
3)     perform update releated action
4)  elif choice=="edit":
5)     perform edit related action
6)  elif choice=="delete":
7)    perform delete related action
```

HTML Validations:

By using HTML, we can perform the following validations

1. required
2. email
3. min and max length
Eg password should be minimum 5 characters and maximum 10 characters

Eg

```
<label for="password">Password:</label>
<input type="password" name="password" id="password" pattern=".{5,10}" placeholder="your password" required title="5 to 10 characters">
```

Pattern for only digits:[0-9]{5,10}
Pattern for only word characters:\w{5,10}

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

CSS

Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

Module-2: CSS (Cascading Style Sheets)

The main objective of CSS to add styles to HTML. CSS describes how HTML elements are displayed on a page.
Styling includes colors,fonts,size,borders etc

We can define CSS styling inside HTML. But it is highly recommended to define styling inside a seperate CSS file(.css extension) and link that file to HTML.

The power of CSS demo link:
https://www.w3schools.com/Css/css_intro.asp

Various ways to define Style for HTML Elements:

We can define style in 3 ways

1. In Line
2. By using style tag
3. By using css file

1. In Line:

```
<h1 style="color:red">This is My First Part of Data</h1>
```

define style at tag level is not recommended b'z it increases complexity as every html page contains 1000s of tags

2. By using style tag:

```
1)  <html >
2)    <head>
3)      ...
4)      <style type="text/css">
5)      h1{
6)        color:blue;
7)      }
8)      </style>
9)    </head>
10)   ...
11) </html>
```

This way of defining style is not recommended because it is applicable only for current page but not for remaining pages.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

3. By using css file:

style1.css

```
1)  h1{
2)    color:red;
3)  }
4)
5)  <head>
6)     <link rel="stylesheet" href="style1.css">
7)  </head>
```

We can reuse same css file for every html page so that we can promote code reusability. Hence this approach is highly recommended to use.

Basic Structure of CSS File:

```
1)  tagname{
2)    property:value;
3)  }
```

Eg:

```
1)  h1{
2)   color:red;
3)  }
```

Once we defined css file, we can link it to html by using <link> tag inside <head> part of html.

```
1)  <head>
2)     <link rel="stylesheet" href="form.css">
3)    </head>
```

How to define comments in css file:

```
/*
  This is CSS comment
*/
```

Demo Application-1:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <link rel="stylesheet" href="style1.css">
6)      <title>CSS Demo</title>
```

```
7)   </head>
8)   <body>
9)     <h1>This is First CSS Example</h1>
10)   <p>Here we are going to discuss just basics of css like how to define and how to link etc<
    /p>
11)   <h2>The List of Topics are:</h2>
12)   <ul>
13)     <li>Topic-1</li>
14)     <li>Topic-2</li>
15)     <li>Topic-3</li>
16)     <li>Topic-4</li>
17)     <li>Topic-5</li>
18)   </ul>
19)   <h3>Soon we will discuss remaining things</h3>
20)  </body>
21) </html>
```

style1.css:

```
1)  h1{
2)    color: red;
3)  }
4)   h2{
5)    color: blue;
6)  }
7)  h3{
8)    color:green;
9)  }
10) p{
11)   color:blue;
12) }
13) ul{
14)   color:cyan;
15) }
```

Various possible ways to specify color:

1. color:red;

2. color:rgb(244,66,220);
we have to collect values from google color picker
The allowed values are: 0 to 255
(0,0,0)--->black
(255,255,255)-->white

3. color:#f44e42
This 6-digit number represents hexa code of color

4. color:rgba(244,66,220,0.9)
a means alpha
The allowed values for a attribute are : 0.0 to 1.0
1.0 means full dark and 0.0 means full light(transparent)

http://www.december.com/html/spec/colorrgbadec.html

Note: The most commonly used approach is: hexa code

Setting Background and Borders:

In CSS, we can set Background as follows:

```
1) body{
2)   background-color:red;
3) }
```

We can set Border as follows:

```
1) div{
2)   border-color:blue;
3)   border-width:thick;
4)   border-style:double;
5) }
```

The allowed values for border-width are:

medium,thin,thick.
We can also set our own width with pixels.

Eg: border-width:10px;

The allowed values for border-style are:
dashed,dotted,groove,double etc

Eg:

```
1) span{
2)   color:yellow;
3)   background: green;
4) }
```

color vs background:

color attribute meant for text color
background attribute meant for background color

```
1) h1{
2)   color:white;
3)   background:blue;
4) }
```

How to Set Background Image:

```
1) body {
2)   background:url(https://image.freepik.com/free-psd/abstract-background-design_1297-73.jpg);
3) }
```

Various properties while setting image:

```
1) body{
2)   color:white;
3)   background:url(https://images.pexels.com/photos/257360/pexels-photo-257360.jpeg?auto=compress&cs=tinysrgb&h=350);
4)   background-repeat: no-repeat;
5)   background-size: cover;
6) }
```

By default background image will be repeated. If we don't want to repeat then we should specify: no-repeat

How to set border:

Normal way:

```
1) h1{
2)   border-color: orange;
3)   border-width: 5px;
4)   border-style: solid;
5) }
```

short-cut way:

```
1) h1{
2)   border: solid red 5px;    order is not important
3) }
```

To set border for the image:

```
1) img{
2)   border:  groove 10px blue;
3) }
```

Basic CSS Selectors:

1. Element Selectors:

select all instances of given element. i.e style is applicable for every tag of the specified type.

```
1) h1{
2)   color:red;
3) }
```

This style applicable for every h1 tag of the html page.

2. ID Selectors:

Selects an element with the given Id. But with in the HTML Page ID is always unique.

html:

```
<h1 id="specialh1">Hello This is First Line</h1>
```

css:

```
1) #specialh1{
2)   color:white;
3)   background: blue;
4) }
```

3. Class Selector:

Select all elements with the given class.

html:

```
1)    <body>
2)     <h1 class="specialh1">Hello This is First Line</h1>
3)     <h1>Hello This is Second Line</h1>
4)     <h1 class="specialh1">Hello This is Third Line</h1>
5)    </body>
```

css:

```
1)  specialh1{
2)    color:white;
3)    background: blue;
4)  }
```

Note: element,id and class selectors are considered as basic css selectors.

Advanced CSS Selectors:

The following are main important advanced selectors

1. * selector
2. Descendant Selector
3. Adjacent Selector
4. Attribute Selector
5. nth of type selector

Demo HTML Page for CSS Selectors:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>CSS Demo</title>
6)      <link rel="stylesheet" href="style1.css">
7)    </head>
8)    <body>
9)      <h1>Advanced Selectors Demo</h1>
10)     <a href="http://google.com">Click Here to go to Google</a>
11)     <ul>
12)       <li>CAT</li>
13)       <li>RAT</li>
14)       <li>DOG</li>
15)     </ul>
16)     <h4>List of Top Movies</h4>
17)     <ul>
18)       <li>Bahubali</li>
19)       <li>MAHANATI</li>
20)       <li>RANGASTALAM</li>
21)     </ul>
22)     <h4>List of Top Websites</h4>
23)     <ul>
24)       <li><a href="http://amazon.com">AMAZON</a> </li>
25)       <li><a href="http://flipkart.com">FlipKart</a> </li>
26)       <li><a href="http://paytm.com">PAYTM</a> </li>
```

```
27)   </ul>
28)  </body>
29) </html>
```

1. * selector:

*** means everything. This style is applicable for every thing in the web page.**

```
1)  *{
2)    color:blue;
3)  }
```

2. Descendant Selector:

```
1)  li a{
2)    color:white;
3)    background: blue;
4)  }
```

li is considered as parent tag and a is considered as child tag.
For every anchor tag present in li tag this style is applicable

3. Adjacent Selector:

```
1)  a+ul{
2)    color: red;
3)  }
```

For every ul tag which is adjacent to a tag,this style is applicable.

```
1)  div+p{
2)    color:blue;
3)    }
```

For every paragraph tag, which is adjacent to div tag this style is applicable.

4. Attribute Selector:

We can define style based on attributes.

Eg 1:

```
1)  a[href]{
2)    color:red;
3)    background: yellow;
4)  }
```

For all href attributes of anchor tag this style is applicable.

Eg 2:

```
1)  a[href="http://amazon.com"]{
2)    color:red;
3)    background: yellow;
4)  }
```

If the value of href attribute is http://amazon.com then only this style is applicable.

```
1)  input[type="password"]
2)  {
3)     background:red;
4)  }
```

This style is applicable for all password fields of input tag.

5. nth of type selectors:

```
1)  li:nth-of-type(2){
2)    color:red;
3)  }
```

For every 2nd li tag this style is applicable.

```
1)  ul:nth-of-type(2){
2)    color:red;
3)    background: yellow;
4)  }
```

For every 2nd ul tag this style is applicable.

```
1)  li:nth-of-type(even){
2)    color:red;
3)  }
```

For every even numbered li this style is applicable.

CSS Inheritance:

All properties of the parent are by default available to the child and we are not required to redefine. This property is called inheritance.

Inheritance concept applicable for css styles also. i.e what every styles are defined for the parent automatically avilable to the child tags also.

Eg:

```
1) body{
2)   color:red;
3) }
```

This style is applicable for all elements present in side body tag.

Eg:

```
1) ul{
2)  color:red;
3) }
```

This style is also applicable for all <li> tags inside <ul> tag.

CSS Specificity:

If multiple styles are available for element then most specific style will be considered.This property is called Specificity of CSS.

Eg:

```
1) body{
2)   color:red;
3) }
4) ul{
5)   color:blue;
6) }
7) li{
8)   color:green;
9) }
```

For <li> tag 3 styles are avilable but css will consider most specific style from <li> tag which is nothing but green color.

CSS Styles Assignment-1:

assignment.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>CSS Assignment</title>
6)      <link rel="stylesheet" href="assignment.css">
7)    </head>
8)    <body>
9)      <h1>This is h1 Data</h1>
10)     <h2>This is h2 tag</h2>
11)     <h2>This is h2 tag</h2>
12)     <h2>This is h2 tag</h2>
13)     <h3>The important Brands are:</h3>
14)     <ul>
15)       <li>King Fisher</li>
16)       <li>Royal Challenge</li>
17)       <li>Foster</li>
18)     </ul>
19)     <p class="hello">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmo
        d tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nost
        rud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
        dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Except
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
    eur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id e
    st laborum.</p>
20)
21)   <p id="special">Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
    tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostr
    ud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure d
    olor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepte
    ur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id es
    t laborum.</p>
22)   <label for="name">Name</label>
23)   <input type="text" id="name" name="username" value=""><br><br>
24)   <label for="pwd">Password</label>
25)   <input id="pwd" type="password" name="password" value=""><br><br>
26)
27)   <a href="http://www.gmail.com">Gmail</a>
28)   <a href="http://www.youtube.com">Youtube</a>
29)   <a href="http://www.facebook.com">Facebook</a>
30)
31)   <h3>Select Your Favourate Heroine:</h3>
32)   <ul>
33)     <li>Sunny:  <input type="checkbox" name="" value="" checked> </li>
34)     <li>Mallika:  <input type="checkbox" name="" value="" checked> </li>
35)     <li>Veena Malik:  <input type="checkbox" name="" value="" > </li>
36)   </ul>
37)  </body>
38) </html>
```

assignment.css:

```
1)  body{
2)  background:#bdc646;
3)  }
4)  h1{
5)  color:#9b79b8;
6)  }
7)  h2{
8)  color:yellow;
9)  }
10) li{
11) color:#0518c4;
12) }
13) p{
14) background:white;
15) }
16) input{
17) border:5px red solid;
18) }
19) .hello{
20)   background:blue;
```

```
21)   color:white;
22)   }
23)
24)  #special{
25)    border: 10px orange solid;
26)    }
27)    input[type="text"]{
28)    background:green;
29)    }
30)    input:checked{
31)    margin-left:50px;
32)    }
33)
34) label{
35)   text-transform: uppercase;
36) }
37) #special:first-letter{
38)  color:green;
39)  font-size:100px;
40) }
41) h1:hover{
42) color:white;
43) border:10px red solid;
44) }
45) a:visited{
46) color:yellow;
47) }
```

Notes:

1. Set body tag background color with #bdc646

```
1.  body{
2.  background:#bdc646;
3.  }
```

2. Set h1 text color with #9b79b8

```
1)  h1{
2)  color:#9b79b8;
3)  }
```

3. Make all h2 tags with yellow color

```
1)  h2{
2)  color:yellow;
3)  }
```

4. Make all <li> elements with blue color,but consider hexa code

```
li{
color:#0518c4;
}
```

5. Change background for every paragraph with white color

```
p{
background:white;
}
```

6. Make all inputs have a 5px border with red color

```
input{
border:5px red solid;
}
```

7. Set blue background with class attribute hello and text color as white

```
.hello{
background:blue;
color:white;
}
```

8. Give 10px orange solid border for id attribute "special"

```
#special{
 border: 10px orange solid;
 }
```

9. Make only inputs with type 'text' have a green background

```
input[type="text"]{
background:green;
}
```

10. Make all "checked" check boxes have a left margin of 50px

```
input:checked{
margin-left:100px;
}
```

11. Make the <label> elements all UPPERCASE without changing in html

```
label{
text-transform:uppercase;
}
```

12. Make the first letter of the element with id "special" with green color and 100 px font-size

```
#special:first-letter{
 color:green;
 font-size:100px;
}
```

13. Make <h1> element color change to white when hovered and give 10px solid red border

```
h1:hover{
color:white;
border:10px red solid;
}
```

14. Make all <a> elements that have been visited with yellow color

```
a:visited{
color:yellow;
}
```

CSS Styles Assignment-2:

assignment2.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <link rel="stylesheet" href="assignment2.css">
6)      <title></title>
7)    </head>
8)    <body>
9)      <p>This Para is outside of div. <em> Duis aute irure dolor in reprehenderit</em> in volu
      ptate velit esse cillum dolore eu fugiat nulla pariatur. </p>
10)
11)     <div>
12)       <p>This Para is inside of first div. <em> em tag inside first div</em> in voluptate velit e
      sse cillum dolore eu fugiat nulla pariatur. </p>
13)
14)       <p>This Para is inside of first div. <em> em tag inside first div</em> in voluptate velit e
      sse cillum dolore eu fugiat nulla pariatur. </p>
15)
16)       <p>This Para is inside of first div. <em> em tag inside first div</em> in voluptate velit e
      sse cillum dolore eu fugiat nulla pariatur. </p>
17)     </div>
18)
19)     <div>
20)       <p>This Para is inside of Second div. <em> em tag inside Second div</em> in voluptate
      velit esse cillum dolore eu fugiat nulla pariatur. </p>
21)
22)       <p>This Para is inside of Second div. <em> em tag inside Second div</em> in voluptate
      velit esse cillum dolore eu fugiat nulla pariatur. </p>
23)
24)       <p>This Para is inside of Second div. <em> em tag inside Second div</em> in voluptate
      velit esse cillum dolore eu fugiat nulla pariatur. </p>
25)     </div>
26)
27)     <div>
28)       <p>This Para is inside of Third div. <em> em tag inside Third div</em> in voluptate velit
       esse cillum dolore eu fugiat nulla pariatur. </p>
29)
30)       <p>This Para is inside of Third div. <em> em tag inside Third div</em> in voluptate velit
       esse cillum dolore eu fugiat nulla pariatur. </p>
31)
32)       <p>This Para is inside of Third div. <em> em tag inside Third div</em> in voluptate velit
       esse cillum dolore eu fugiat nulla pariatur. </p>
33)     </div>
34)     <p>This Para is outside of div. <em> Duis aute irure dolor in reprehenderit</em> in volu
      ptate velit esse cillum dolore eu fugiat nulla pariatur. </p>
35)   </body>
36) </html>
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

assignment2.css:

```
1)  div p{
2)    font-size: 25px;
3)
4)  }
5)
6)  div:nth-of-type(3) p{
7)  background:blue;
8)  color:white;
9)  }
10)
11) div:nth-of-type(3)  p:nth-of-type(2){
12) border: 5px red solid;
13) }
14)
15) div:nth-of-type(2) em{
16)  color: white;
17)  font-size:30px;
18)  background:green;
19)  }
```

Notes:

Case-1: Make all <p>s That are nested inside div with 25px font-size

```
1)  div p{
2)    font-size: 25px;
3)
4)  }
```

Case-2: Give all <p>s in side 3rd div blue background and text color is white

```
1)  div:nth-of-type(3) p{
2)  background:blue;
3)  color:white;
4)  }
```

Case-3: Give 2nd <p> inside 3rd div, 5px red solid border

```
1)  div:nth-of-type(3)  p:nth-of-type(2){
2)  border: 5px red solid;
3)  }
```

Case-4: Make tag in the 2nd with white color 30 px font-size and green background

```
1) div:nth-of-type(2) em{
2)  color: white;
3)  font-size:30px;
4)  background:green;
5)  }
```

Fonts and Text in CSS:

The following are very important properties related to fonts and text in css

1. font-family
2. font-size
3. font-weight
4. line-height
5. text-align
6. text-decoration

1. font-family:

We can select desired font from default css system fonts in the following link

https://www.cssfontstack.com/

Eg:

```
1) h1{
2)   font-family: Arial Black;
3) }
```

Note: If we are not satisfied with default css system fonts, then we can use external fonts also.

2. font-size:

```
1) p{
2)   font-size: 20px;
3) }
```

We can also specify font-size in em units, which is also known as dynamic font-size (relative font-size)

Eg:

```
1) span{
2)   font-size: 2.0em;
3) }
```

2.0em means double of parent tag font-size

3. font-weight:

```
p{
  font-weight: 600;
}
```

something like bold font,light font etc
The different allowed values are:
bold,bolder,lighter,normal
100 to 900 where 100 means light and 900 means too much bold.

4. line-height:

The space between 2 lines is called line height.

```
p{
  line-height: 1.5;
}
```

5. text-align:

```
p{
text-align:center;
}
```

The allowed values are: left,rigth,center,justify

6. text-decoration:

like underlined,strike through

Eg:

```
p{
  text-decoration: line-through;
}
```

The allowed values are: underline, overline, line-through

How to use Custom Fonts in CSS:

Most of the times we can select custom fonts from the google.

https://fonts.google.com/

select required fonts and add that link in html header part.
Inside css file specify that font wint font-family attribute.

demo.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Demo for CSS</title>
    <link rel="stylesheet" href="demo.css">
    <link href="https://fonts.googleapis.com/css?family=Eater|Great+Vibes|Indie+Flower" rel="stylesheet">
  </head>
  <body>
   <h1>This is Heading</h1>
   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incid idunt ut labore et dolore magna aliqua. <span>Ut enim ad minim veniam, quis nostrud exe rcitation ullamco laboris nisi ut aliquip ex ea commodo consequat</span>. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Except eur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id e st laborum.</p>
  </body>
</html>
```

demo.css:

```
body{
  font-family: 'Great Vibes', cursive;
  font-size:30px;
}
```

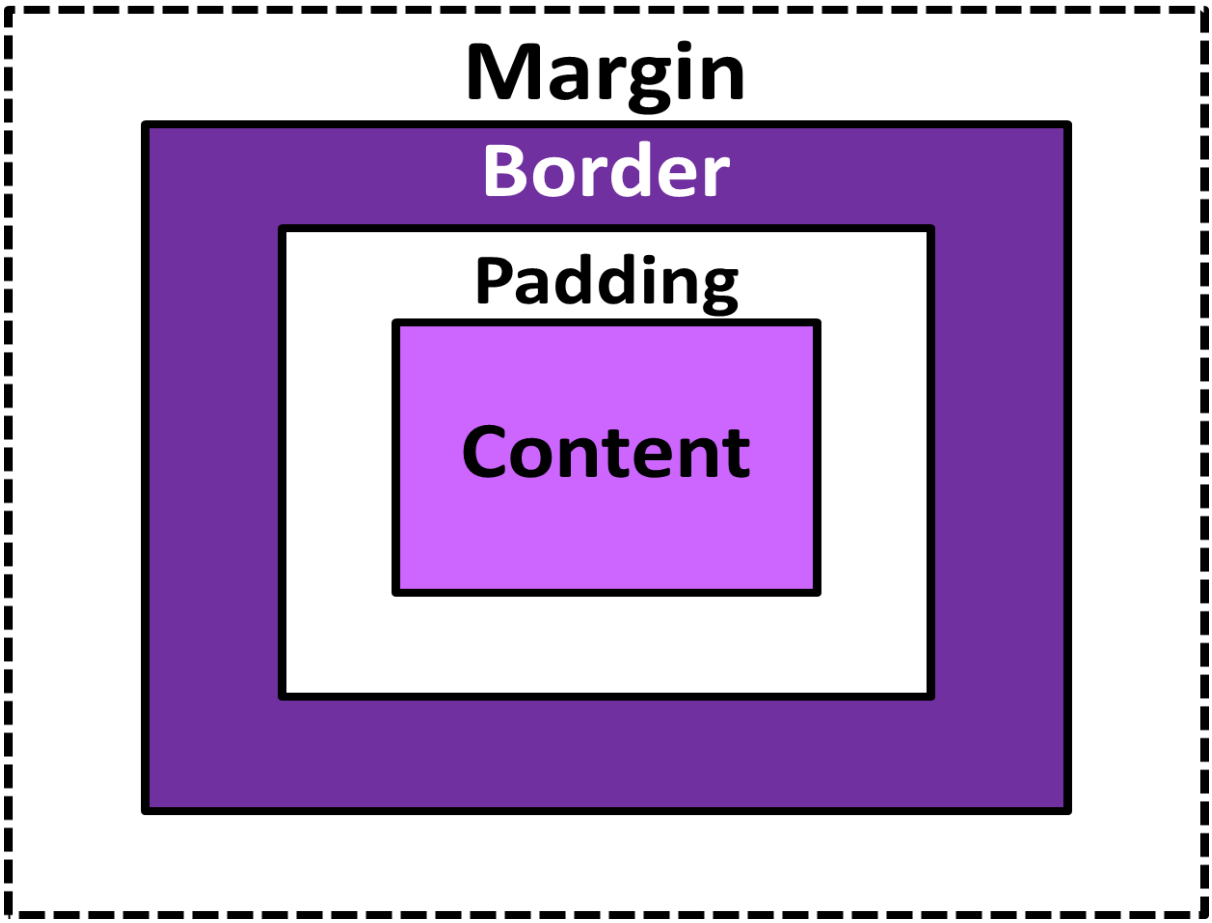
The Box Model:

In a document, each element is represented as a rectangular box. In CSS, each of these rectangular boxes is decribed by using the stanadard box model.

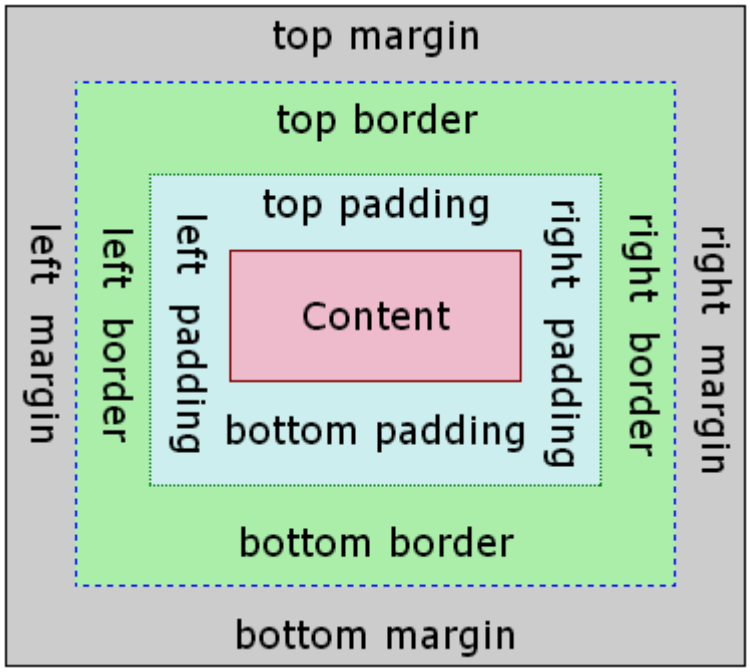
Each box has 4 edges:

1. Content Edge
2. Border Edge
3. Padding Edge
4. Margin Edge

CSS Box Model Diagram -1



CSS Box Model Diagram -2



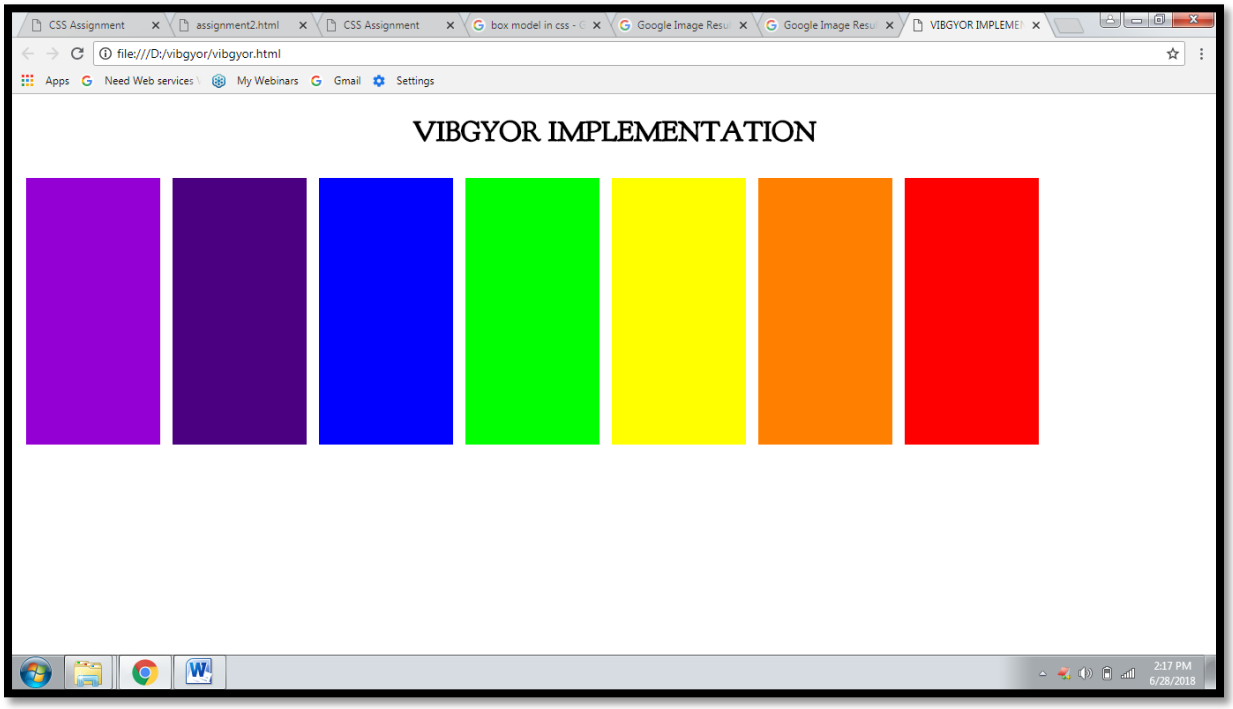
Demo Program:

```
1)  <p>This is Paragraph</p>
2)  <p>This is Paragraph</p>
```

css:

```
1)  p{
2)     width: 200px;
3)     border: 3px solid blue;
4)     padding: 40px 50px 60px 70px;
5)     margin:100px 5px 300px 400px;
6)
7)   }
```

Vibgyor Implementation:



vibgyor.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>VIBGYOR IMPLEMENTATION</title>
6)      <link rel="stylesheet" href="vibgyor.css">
7)      <link href="https://fonts.googleapis.com/css?family=Goudy+Bookletter+1911" rel="styl
    esheet">
8)    </head>
```

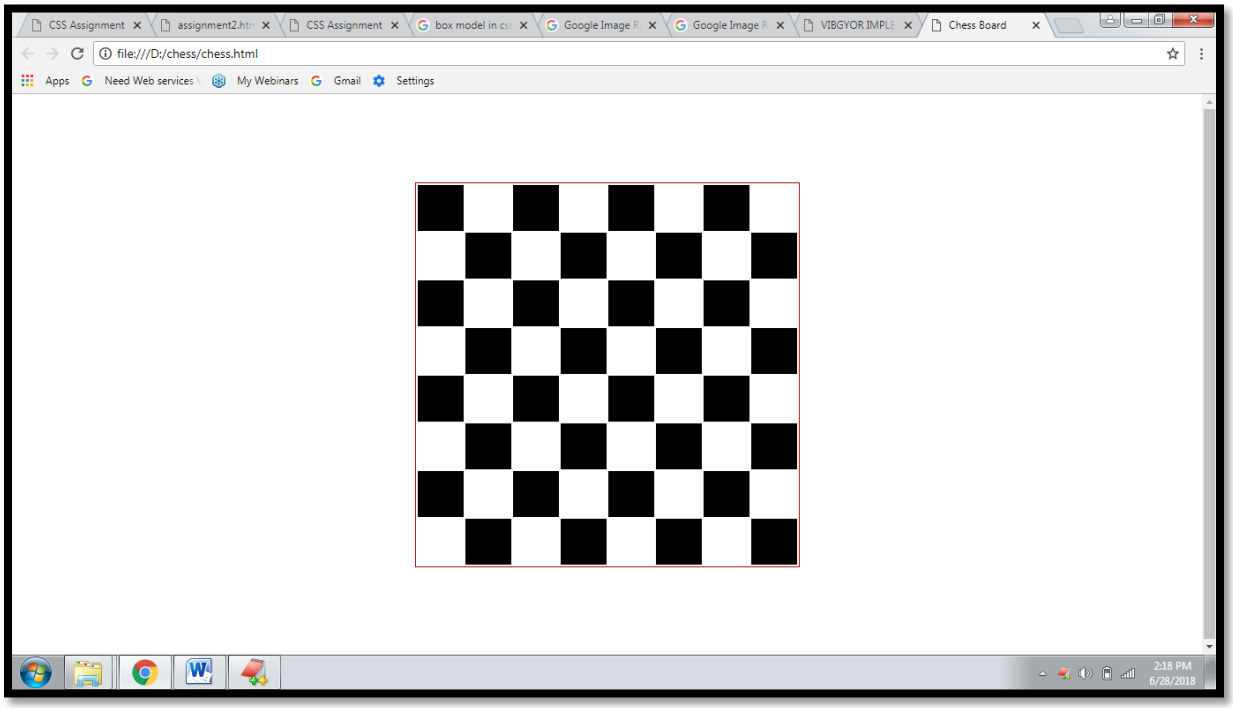
```
9)   <body>
10)   <h1>VIBGYOR IMPLEMENTATION</h1>
11)   <table class='tab'>
12)    <tr>
13)     <td id="one"></td>
14)     <td id="two"></td>
15)     <td id="three"></td>
16)     <td id="four"></td>
17)     <td id="five"></td>
18)     <td id="six"></td>
19)     <td id="seven"></td>
20)    </tr>
21)   </table>
22)
23)  </body>
24) </html>
```

vibgyor.css:

```
1)  h1{
2)   font-family: 'Goudy Bookletter 1911', serif;
3)   text-align: center;
4)  }
5)
6)  td{
7)   height:300px;
8)   width:150px;
9)   border: 6px solid white;
10)
11) }
12) #one{
13)  background: #9400D3;
14) }
15) #two{
16)  background: #4B0082;
17) }
18) #three{
19)  background: #0000FF  ;
20) }
21) #four{
22)  background: #00FF00  ;
23) }
24) #five{
25)  background: #FFFF00;
26) }
27) #six{
28)  background: #FF7F00;
29) }
30) #seven{
```

```
31)  background: #FF0000;
32) }
```

Chess Board Implementation:



chess.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Chess Board</title>
6)      <link rel="stylesheet" href="chess.css">
7)    </head>
8)    <body>
9)      <table>
10)       <tr>
11)         <td id="black"></td>
12)         <td id="white"></td>
13)         <td id="black"></td>
14)         <td id="white"></td>
15)         <td id="black"></td>
16)         <td id="white"></td>
17)         <td id="black"></td>
18)         <td id="white"></td>
19)       </tr>
20)       <tr>
21)
```

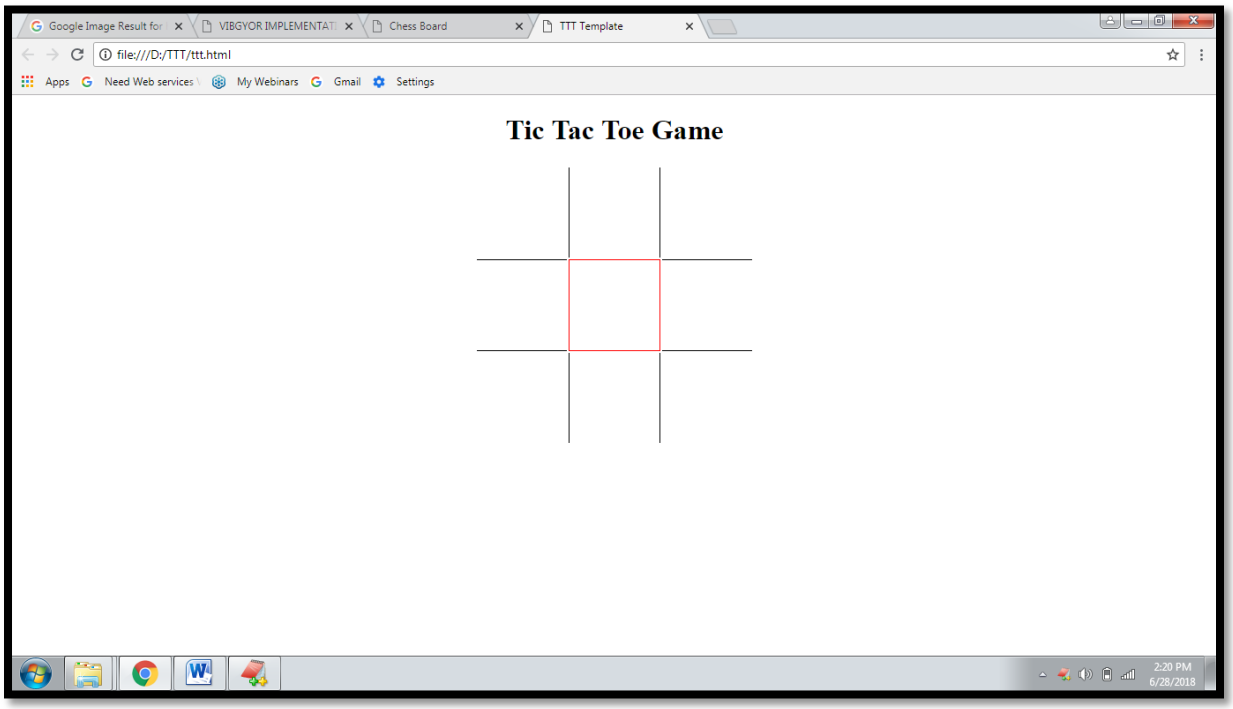
```
22)     <td id="white"></td>
23)     <td id="black"></td>
24)     <td id="white"></td>
25)     <td id="black"></td>
26)     <td id="white"></td>
27)     <td id="black"></td>
28)     <td id="white"></td>
29)     <td id="black"></td>
30)    </tr>
31)    <tr>
32)     <td id="black"></td>
33)     <td id="white"></td>
34)     <td id="black"></td>
35)     <td id="white"></td>
36)     <td id="black"></td>
37)     <td id="white"></td>
38)     <td id="black"></td>
39)     <td id="white"></td>
40)    </tr>
41)    <tr>
42)
43)     <td id="white"></td>
44)     <td id="black"></td>
45)     <td id="white"></td>
46)     <td id="black"></td>
47)     <td id="white"></td>
48)     <td id="black"></td>
49)     <td id="white"></td>
50)     <td id="black"></td>
51)    </tr>
52)    <tr>
53)     <td id="black"></td>
54)     <td id="white"></td>
55)     <td id="black"></td>
56)     <td id="white"></td>
57)     <td id="black"></td>
58)     <td id="white"></td>
59)     <td id="black"></td>
60)     <td id="white"></td>
61)    </tr>
62)    <tr>
63)
64)     <td id="white"></td>
65)     <td id="black"></td>
66)     <td id="white"></td>
67)     <td id="black"></td>
68)     <td id="white"></td>
69)     <td id="black"></td>
70)     <td id="white"></td>
```

```
71)       <td id="black"></td>
72)     </tr>
73)     <tr>
74)       <td id="black"></td>
75)       <td id="white"></td>
76)       <td id="black"></td>
77)       <td id="white"></td>
78)       <td id="black"></td>
79)       <td id="white"></td>
80)       <td id="black"></td>
81)       <td id="white"></td>
82)     </tr>
83)     <tr>
84)
85)       <td id="white"></td>
86)       <td id="black"></td>
87)       <td id="white"></td>
88)       <td id="black"></td>
89)       <td id="white"></td>
90)       <td id="black"></td>
91)       <td id="white"></td>
92)       <td id="black"></td>
93)     </tr>
94)   </table>
95)
96)  </body>
97) </html>
```

chess.css:

```
1)  table{
2)    border:1px groove red;
3)    margin: 100px auto;
4)  }
5)  td{
6)    height:50px;
7)    width:50px;
8)  }
9)  #white{
10)   background: white;
11) }
12) #black{
13)   background: black;
14)  }
```

Tic-Tac-Toe Implementation:



ttt.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <link rel="stylesheet" href="ttt.css">
5)      <meta charset="utf-8">
6)      <title>TTT Template</title>
7)    </head>
8)    <body>
9)      <h1>Tic Tac Toe Game</h1>
10)     <table>
11)       <tr>
12)         <td></td>
13)         <td class="vertical"></td>
14)         <td></td>
15)       </tr>
16)       <tr>
17)         <td class="horizontal"></td>
18)         <td class="vertical horizontal"></td>
19)         <td class="horizontal"></td>
20)       </tr>
21)       <tr>
22)         <td></td>
23)         <td class="vertical"></td>
24)         <td></td>
```

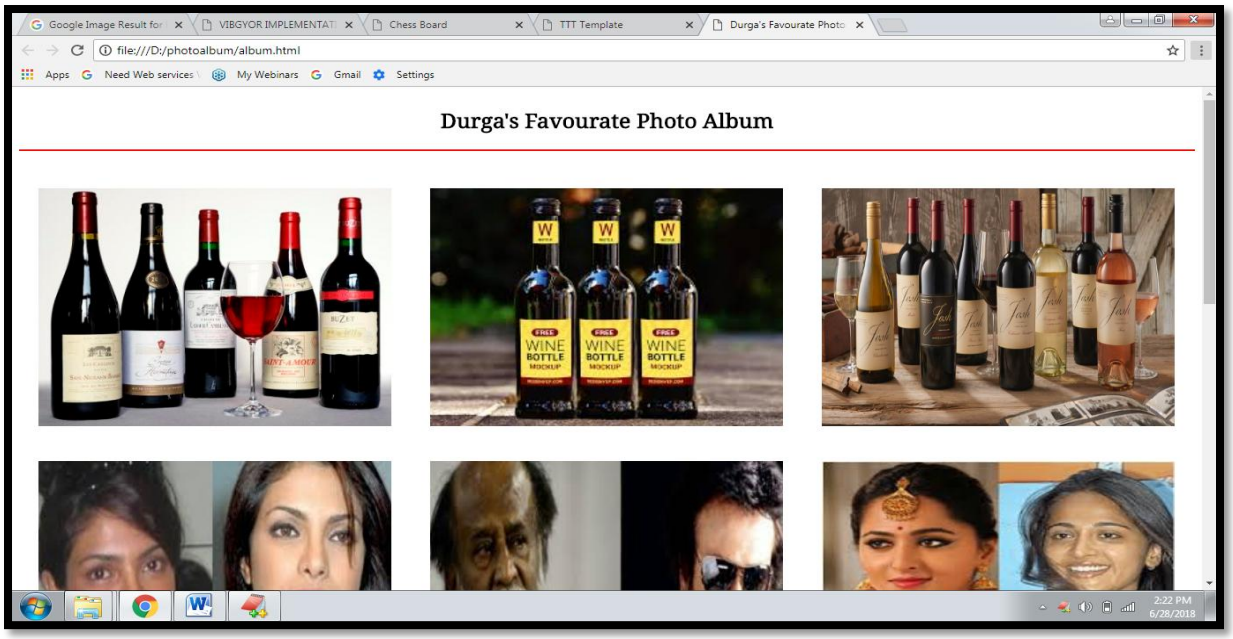

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
25)     </tr>
26)    </table>
27)   </body>
28) </html>
```

ttt.css:

```
1)  td{
2)    width: 100px;
3)    height: 100px;
4)  }
5)  .vertical{
6)    border-left: 1px solid black;
7)    border-right: 1px solid black;
8)  }
9)  .horizontal{
10)   border-top: 1px solid black;
11)   border-bottom: 1px solid black;
12) }
13) td:hover{
14)   border: 1px solid red;
15) }
16) table{
17)   margin:auto;
18) }
19) h1{
20)   text-align: center;
21) }
```

Durga's Favourate Photo Album:



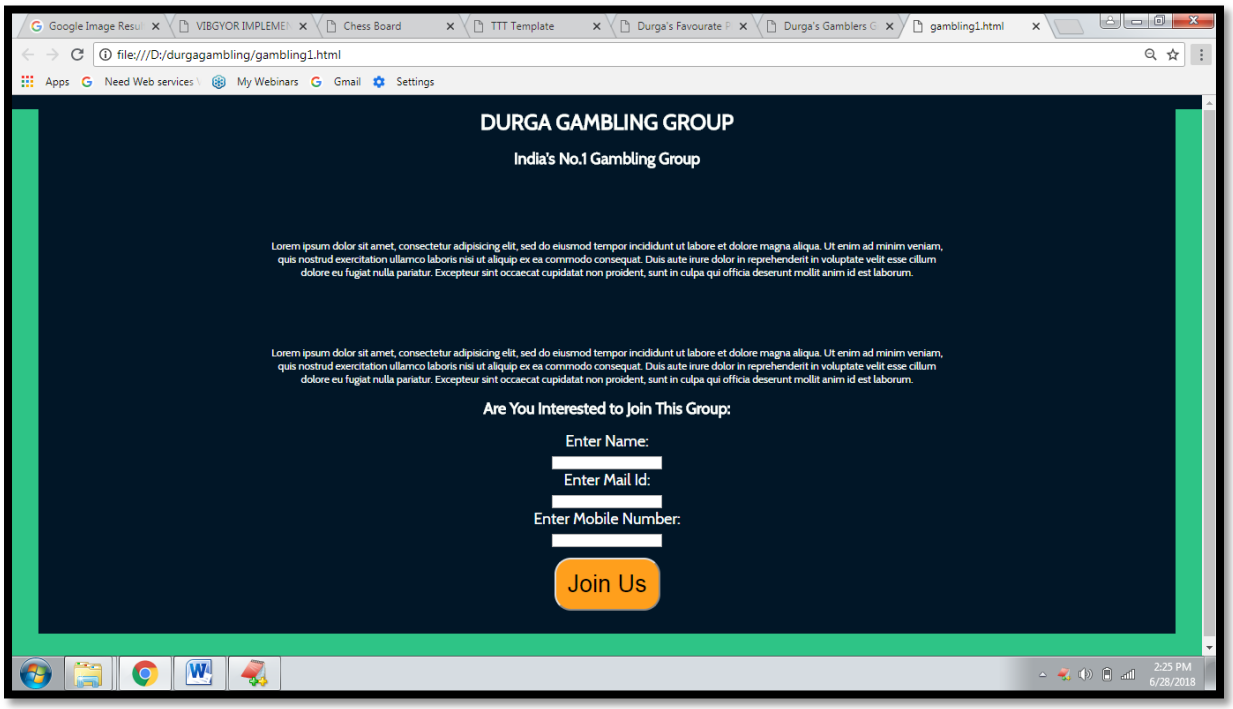
album.html:

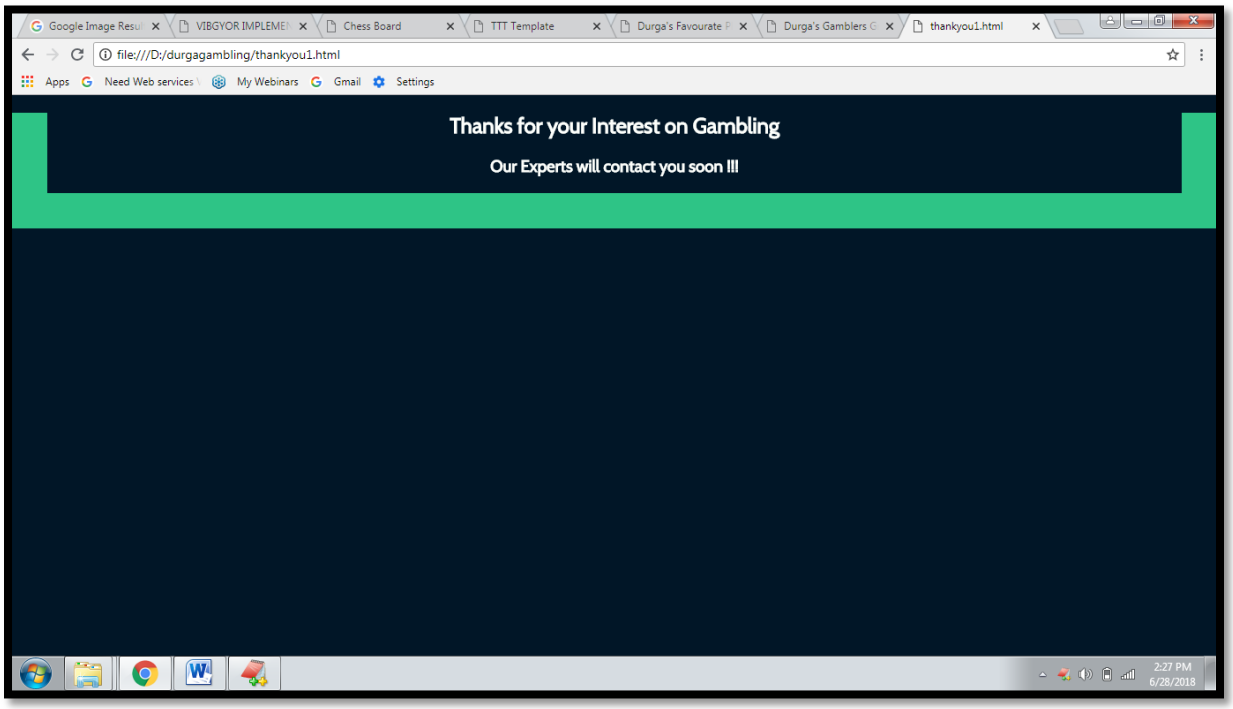
```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Durga's Favourate Photo Album</title>
6)      <link rel="stylesheet" href="album.css">
7)
8)  <link href="https://fonts.googleapis.com/css?family=Goudy+Bookletter+1911|Noto+Serif
    " rel="stylesheet">
9)
10)   </head>
11)   <body>
12)     <p>Durga's Favourate Photo Album</p>
13)     <img src="divine1.jpg">
14)     <img src="divine2.jpg">
15)     <img src="divine3.jpg">
16)     <img src="original1.jpg">
17)     <img src="original2.jpg">
18)     <img src="original3.jpg">
19)     <img src="original4.jpg">
20)     <img src="original5.jpg">
21)     <img src="original6.jpg">
22)     <img src="original7.jpg">
23)     <img src="original8.jpg">
24)     <img src="original9.jpg">
25)   </body>
26) </html>
```

album.css:

```
1)  img{
2)    width: 30%;
3)    height:300px;
4)    float: left;
5)    margin: 1.66%;
6)  }
7)  p{
8)    font-family: 'Noto Serif', serif;
9)    text-align: center;
10)   font-size: 25px;
11)   font-weight: 800;
12)   border-bottom: 2px solid red;
13)   padding-bottom: 20px;
14) }
```

Durga Gambling Website:





DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

gambling.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Durga's Gamblers Group</title>
6)      <link rel="stylesheet" href="gambling.css">
7)      <link href="https://fonts.googleapis.com/css?family=Josefin+Sans" rel="stylesheet">
8)    </head>
9)    <body>
10)     <h1>DURGA GAMBLING GROUP</h1>
11)     <h2>India's No.1 Gambling Group</h2>
12)     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor inci
        didunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitati
        on ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in repre
        henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaec
        at cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p
        >
13)     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor inci
        didunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitati
        on ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in repre
        henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaec
        at cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p
        >
14)     <h2>Are You Interested to Join This Group:</h2>
15)     <form class="" action="thanks.html" method="post">
16)       <label for="name">Enter Name:</label><br>
17)       <input id="name" type="text" name="name" value=""><br>
18)       <label for="mail">Enter Mail Id:</label><br>
19)       <input id="mail" type="email" name="" value=""><br>
20)       <label for="mobile">Enter Mobile Number:</label><br>
21)       <input id="mobile" type="text" name="" value=""><br>
22)       <input id="sub" type="submit" name="" value="Join Us">
23)     </form>
24)   </body>
25) </html>
```

thanks.html:

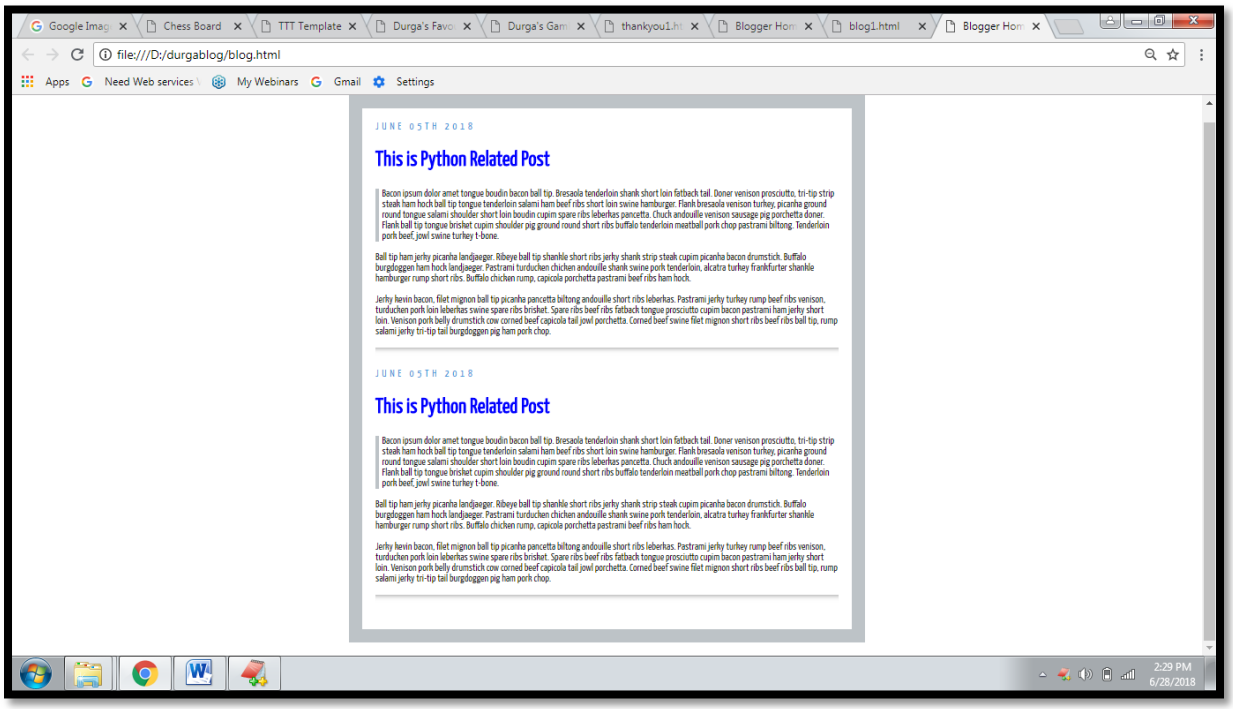
```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Thanking Page</title>
6)      <link href="https://fonts.googleapis.com/css?family=Josefin+Sans" rel="stylesheet">
7)      <link rel="stylesheet" href="gambling.css">
8)    </head>
9)    <body>
```

```
    <h2>Thanks for Your Interest on Gambling</h2>
    <h3>Our Experts will contact You soon !!!!</h3>
  </body>
</html>
```

gambling.css:

```
body{
  font-family: 'Josefin Sans', sans-serif;
  background: #011627;
  color:#fdfffc;
  text-align: center;
  border: 40px solid #2EC486;
  border-top: 0px;
  margin: 0px;
}
p{
  padding-top: 5%;
  padding-left: 20%;
  padding-right: 20%;
}
form{
  font-size: 1.5em;
  margin: 20px;
}
#sub{
  background: #FF9F1C;
  height: 80px;
  width: 150px;
  margin: 10px;
  font-size: 1.5em;
  border-radius: 25px;
}
```

Blog design by using HTML and CSS:



blog.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>Blogger Home Page</title>
6)      <link rel="stylesheet" href="blog.css">
7)      <link href="https://fonts.googleapis.com/css?family=Indie+Flower|Lobster|Yanone+Kaf
    feesatz" rel="stylesheet">
8)    </head>
9)    <body>
10)     <div class="post">
11)       <div class='date'> June 05th 2018</div>
12)     <h2>This is Python Related Post</h2>
13)     <p class="mainp">Bacon ipsum dolor amet tongue boudin bacon ball tip. Bresaola tende
    rloin shank short loin fatback tail. Doner venison prosciutto, tri-
    tip strip steak ham hock ball tip tongue tenderloin salami ham beef ribs short loin swine ha
    mburger. Flank bresaola venison turkey, picanha ground round tongue salami shoulder sh
    ort loin boudin cupim spare ribs leberkas pancetta. Chuck andouille venison sausage pig po
    rchetta doner. Flank ball tip tongue brisket cupim shoulder pig ground round short ribs buf
    falo tenderloin meatball pork chop pastrami biltong. Tenderloin pork beef, jowl swine turk
    ey t-bone.
14)     </p>
15)
16)     <p>
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
17)     Ball tip ham jerky picanha landjaeger. Ribeye ball tip shankle short ribs jerky shank stri
    p steak cupim picanha bacon drumstick. Buffalo burgdoggen ham hock landjaeger. Pastra
    mi turducken chicken andouille shank swine pork tenderloin, alcatra turkey frankfurter sh
    ankle hamburger rump short ribs. Buffalo chicken rump, capicola porchetta pastrami beef
    ribs ham hock.
18)    </p>
19)
20)    <p>
21)     Jerky kevin bacon, filet mignon ball tip picanha pancetta biltong andouille short ribs leb
    erkas. Pastrami jerky turkey rump beef ribs venison, turducken pork loin leberkas swine sp
    are ribs brisket. Spare ribs beef ribs fatback tongue prosciutto cupim bacon pastrami ham j
    erky short loin. Venison pork belly drumstick cow corned beef capicola tail jowl porchetta.
    Corned beef swine filet mignon short ribs beef ribs ball tip, rump salami jerky tri-
    tip tail burgdoggen pig ham pork chop.
22)    </p>
23)    <hr>
24)   </div>
25)   <div class="post">
26)    <div class='date'> June 05th 2018</div>
27)   <h2>This is Python Related Post</h2>
28)   <p class="mainp">Bacon ipsum dolor amet tongue boudin bacon ball tip. Bresaola tender
    loin shank short loin fatback tail. Doner venison prosciutto, tri-
    tip strip steak ham hock ball tip tongue tenderloin salami ham beef ribs short loin swine ha
    mburger. Flank bresaola venison turkey, picanha ground round tongue salami shoulder sh
    ort loin boudin cupim spare ribs leberkas pancetta. Chuck andouille venison sausage pig po
    rchetta doner. Flank ball tip tongue brisket cupim shoulder pig ground round short ribs buf
    falo tenderloin meatball pork chop pastrami biltong. Tenderloin pork beef, jowl swine turk
    ey t-bone.
29)   </p>
30)
31)   <p>
32)    Ball tip ham jerky picanha landjaeger. Ribeye ball tip shankle short ribs jerky shank strip
    steak cupim picanha bacon drumstick. Buffalo burgdoggen ham hock landjaeger. Pastrami
    turducken chicken andouille shank swine pork tenderloin, alcatra turkey frankfurter shankl
    e hamburger rump short ribs. Buffalo chicken rump, capicola porchetta pastrami beef ribs
    ham hock.
33)   </p>
34)
35)   <p>
36)    Jerky kevin bacon, filet mignon ball tip picanha pancetta biltong andouille short ribs lebe
    rkas. Pastrami jerky turkey rump beef ribs venison, turducken pork loin leberkas swine spa
    re ribs brisket. Spare ribs beef ribs fatback tongue prosciutto cupim bacon pastrami ham je
    rky short loin. Venison pork belly drumstick cow corned beef capicola tail jowl porchetta. C
    orned beef swine filet mignon short ribs beef ribs ball tip, rump salami jerky tri-
    tip tail burgdoggen pig ham pork chop.
37)   </p>
38)   <hr>
39) </div>
40)   </body>
```

```
41) </html>
```

blog.css:

```
1)  body{
2)    border: 20px solid #bdc3c7;
3)    padding:20px;
4)    width:700px;
5)    margin:20px auto;
6)    font-family: 'Yanone Kaffeesatz', sans-serif;
7)  }
8)  .mainp{
9)    border-left: 5px solid #bdc3c7;
10)   padding-left: 5px;
11) }
12) .date{
13)   text-transform: uppercase;
14)   color: #3498db;
15)   letter-spacing: 5px;
16) }
17) h2{
18)   color: blue;
19)   font-size: 2.0em;
20) }
21) .post{
22)   margin-bottom: 20px;
23) }
24) hr {
25)   height: 12px; border: 0; box-shadow: inset 0 12px 12px -12px rgba(0, 0, 0, 0.5);
26)  }
```

Bootstrap

Bootstrap is the most commonly used framework for Front-End Development. [Django is the most commonly used web framework for back-end development with Python]

Bootstrap providing several pre defined libraries for css and java script.

Current version of Bootstrap is: 4.1.1

Bootsrap 3 vs Bootstrap 4:

1. Panels are replaced with Cards
2. Larger default font sizes
3. New Grid Tier(XL)
4. Use of Flexbox
5. Moved from Less to Sass

website: getbootstrap.com

How to connect Bootstrap with HTML:

We can connect Bootstrap with HTML by using the following 2 ways

1. By using CDN
2. Locally

1. By using CDN:

Content Delivery Network

just add the following in the <head> part of our html

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css">
```

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css">
5)      <meta charset="utf-8">
6)      <title></title>
7)    </head>
8)    <body>
9)      <h1>This is First Bootstrap Demo</h1>
10)   </body>
11) </html>
```

2. Locally:

Download bootstrap.css file from the link getbootstrap.com and download button

https://github.com/twbs/bootstrap/releases/download/v4.1.1/bootstrap-4.1.1-dist.zip

zip file contains bootstrap.css file.
copy this file in our application folder and add the following link in html

<link rel="stylesheet" href="Bootstrap.css">

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <link rel="stylesheet" href="Bootstrap.css">
5)      <meta charset="utf-8">
6)      <title></title>
7)    </head>
8)    <body>
9)      <h1>This is First Bootstrap Demo</h1>
10)
11)   </body>
12) </html>
```

Agenda:

1. Buttons
2. Forms
3. Nav
4. Grid

Bootstrap- Buttons:

Most of the styles in Bootstrap are implemented as class styles.(ie class selectors)

We can use these classes directly in our web application to improve look and feel.

The following are commonly used classes

1. container class
2. button related classes
2. jumbotron classes

1. container class:

To center our elements we should use container class.

eg:

```
1)  <div class="container">
2)      <h1>This is First Bootstrap Demo</h1>
3)  </div>
```

2. Button classes:

We can use button related classes for <a>,<button> and <input> tags.

How to use bootstrap success style for our buttons:

We have to use btn btn-success

<button type="button" name="button" class="btn btn-success ">Login</button>

To make our button as large: btn-lg

<button type="button" name="button" class="btn btn-success btn-lg ">Login</button>

To make button as active:

We have to use active class

<button type="button" name="button" class="btn btn-success btn-lg active">Login</button>

To make button as disabled:

<button type="button" name="button" class="btn btn-success"
disabled="disabled">Login</button>

Usage of button related classes for anchor tag:

Click Here for Bootstrap Documentation

Usage of button related classes for input tag:

We can use button related classes for input tag also,but not recommended to use.

Name: <input type="text" class="btn btn-success btn-lg" name="" value="">

How to change default styles of Bootstrap:

Based on our requirement we can customize/change bootstrap default styles.

eg:

```
1)  <html lang="en" dir="ltr">
2)    <head>
3)      <link rel="stylesheet" href="Bootstrap.css">
4)       <style type="text/css">
5)      .btn-success{
6)        background: orange;
7)        color: blue;
8)        }
9)      </style>
10)   </head>
11)   <body>
12)    <div class="container">
13)      <button type="button" name="button" class="btn btn-success ">Login</button>
14)    </div>
15)   </body>
16) </html>
```

Jumbotron classes:

A lightweight, flexible component that can optionally extend the entire viewport to showcase key content on our site.

eg:

```
1)  <div class="jumbotron">
2)     <h1>Hello, world!</h1>
3)     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, dsfsafd.</p>
4)     <p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>
5)    </div>
6)   </div>
```

Note: Several scenarios with HTML,CSS and JS https://www.w3schools.com/howto/

Bootstrap Forms:

Bootstrap provides several classes for forms. The main important classes are

1. form-group:

It is responsible to maintain proper space between elements so that elements will be arranged properly.

2. form-control:

This class is responsible to make width as 100% for elements <input>, <textarea>, and <select>. It is also responsible for border styling.

Basic Example Form :

```
1)  <form>
2)    <div class="form-group">
3)      <label for="exampleInputEmail1">Email address</label>
4)      <input type="email" class="form-
    control" id="exampleInputEmail1" placeholder="Email">
5)    </div>
6)    <div class="form-group">
7)      <label for="exampleInputPassword1">Password</label>
8)      <input type="password" class="form-
    control" id="exampleInputPassword1" placeholder="Password">
9)    </div>
10)   <div class="form-group">
11)     <label for="exampleInputFile">File input</label>
12)     <input type="file" id="exampleInputFile">
13)     <p class="help-block">Example block-level help text here.</p>
14)   </div>
15)   <div class="checkbox">
16)     <label>
17)       <input type="checkbox"> Check me out
18)     </label>
19)   </div>
20)   <button type="submit" class="btn btn-default">Submit</button>
21) </form>
```

Inline Form:

```
1)  <form class="form-inline">
2)    <div class="form-group">
3)      <label for="exampleInputName2">Name</label>
```

```
4)     <input type="text" class="form-
    control" id="exampleInputName2" placeholder="Jane Doe">
5)    </div>
6)    <div class="form-group">
7)     <label for="exampleInputEmail2">Email</label>
8)     <input type="email" class="form-
    control" id="exampleInputEmail2" placeholder="jane.doe@example.com">
9)    </div>
10)   <button type="submit" class="btn btn-default">Send invitation</button>
11) </form>
```

Form Development by using Bootstrap elements:

```
<form>

  <!-- EMAIL SUBMISSION -->

 <div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp" placeholder="Enter email">
  <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone
else.</small>
 </div>

 <!-- PASSWORD -->


 <div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
 </div>

 <!-- DROPDOWN SELECT -->


 <div class="form-group">
  <label for="exampleSelect1">Example select</label>
  <select class="form-control" id="exampleSelect1">
   <option>1</option>
   <option>2</option>
   <option>3</option>
   <option>4</option>
   <option>5</option>
  </select>
 </div>
```

```
<!-- MULTIPLE SELECT OPTIONS -->

<div class="form-group">
  <label for="exampleSelect2">Example multiple select</label>
  <select multiple class="form-control" id="exampleSelect2">
    <option>1</option>
    <option>2</option>
    <option>3</option>
    <option>4</option>
    <option>5</option>
  </select>
</div>


<!-- TEXT AREA -->

<div class="form-group">
  <label for="exampleTextarea">Example textarea</label>
  <textarea class="form-control" id="exampleTextarea" rows="3"></textarea>
</div>


<!-- FILE UPLOAD INPUT -->

<div class="form-group">
  <label for="exampleInputFile">File input</label>
  <input type="file" class="form-control-file" id="exampleInputFile" aria-describedby="fileHelp">
  <small id="fileHelp" class="form-text text-muted">This is some placeholder block-level help text
for the above input. It's a bit lighter and easily wraps to a new line.</small>
</div>

<!-- RADIO BUTTONS -->

<fieldset class="form-group">
  <legend>Radio buttons</legend>
  <div class="form-check">
    <label class="form-check-label">
      <input type="radio" class="form-check-input" name="optionsRadios" id="optionsRadios1"
value="option1" checked>
      Option one is this and that&amp;mdash;be sure to include why it's great
    </label>
  </div>


  <div class="form-check">
  <label class="form-check-label">
      <input type="radio" class="form-check-input" name="optionsRadios" id="optionsRadios2"
value="option2">
      Option two can be something else and selecting it will deselect option one
```

```
    </label>
   </div>



   <div class="form-check disabled">
   <label class="form-check-label">
      <input type="radio" class="form-check-input" name="optionsRadios" id="optionsRadios3"
value="option3" disabled>
      Option three is disabled
    </label>
   </div>


 </fieldset>


 <!-- CHECK BUTTON -->

 <div class="form-check">
  <label class="form-check-label">
   <input type="checkbox" class="form-check-input">
   Check me out
  </label>
 </div>


 <button type="submit" class="btn btn-primary">Submit</button>


</form>
```

Note:

The <fieldset> tag is used to group related elements in a form.

Bootstrap: Navbars:

Navbars are nothing but Navigation Bars. Usually the navbars present on the top of the web page,which can be used for navigation purposes.

How to create Navbar Template:

```
<nav class="navbar navbar-default">

</nav>
```

How to add Brand to the Navbar:

```
1)  <nav class="navbar navbar-default">
2)      <div class="navbar-header">
3)          <a href="http://durgasoft.com" class="navbar-brand">DURGASOFT</a>
4)      </div>
5)  </nav>
```

How to add remaining items to the Navbar:

The remaining items will be added in the form of unordered list. Each List item acts as Navbar item

```
1)  <nav class="navbar navbar-default">
2)      <div class="navbar-header">
3)          <a href="http://durgasoft.com" class="navbar-brand">DURGASOFT</a>
4)      </div>
5)      <ul class="nav navbar-nav">
6)        <li> <a href="#">Home</a></li>
7)        <li> <a href="#">About Us</a></li>
8)        <li> <a href="#">Gallary</a></li>
9)        <li> <a href="#">Services</a></li>
10)     </ul>
11) </nav>
```

All these items will be added from Left Hand side

How to add items to the Right Hand Side:

These items also will be added in the form of unordered list.

```
1)  <nav class="navbar navbar-default">
2)      <div class="navbar-header">
3)        <a href="http://durgasoft.com" class="navbar-brand">DURGASOFT</a>
4)      </div>
5)      <ul class="nav navbar-nav">
6)        <li> <a href="#">Home</a></li>
7)        <li> <a href="#">About Us</a></li>
8)        <li> <a href="#">Gallary</a></li>
9)        <li> <a href="#">Services</a></li>
10)     </ul>
11)     <ul class="nav navbar-nav navbar-right">
12)       <li> <a href="#">Contact Us</a></li>
```

```
13)       <li> <a href="#">Logout</a></li>
14)     </ul>
15)  </nav>
```

How to position navbar items properly:

We have to take all items inside container class.

```
1)   <nav class="navbar navbar-default">
2)     <div class="container">
3)        header/brand, items..
4)      </div>
5)    </nav>
```

Note: **It is not recommended to take total navbar inside container class**

```
<div class="container">
    <nav class="navbar navbar-default">
     ...
    </nav>

 </div>
```

How to fix navbar always at top even in scroll down:

```
<nav class="navbar navbar-default    navbar-fixed-top">
```

How to inverse color of Navbar:

```
<nav class="navbar navbar-default    navbar-fixed-top   navbar-inverse">
```

How to implement Hamburger to Navbar:

We have to enclose the collapsed items inside the following div tag.

```
1)   <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
2)                    <ul class="nav navbar-nav">
3)                      <li><a href="#">Home</a></li>
4)                      <li><a href="#">About Us</a></li>
5)                      <li><a href="#">Services</a></li>
6)                      <li><a href="#">Services</a></li>
7)                      <li><a href="#">Services</a></li>
8)                    </ul>
9)                    <ul class="nav navbar-nav navbar-right">
10)                     <li><a href="#">Contact Us</a></li>
11)                     <li><a href="#">Logout</a></li>
12)                   </ul>
13) </div>
```

To add hamburger,we have to add the following tag inside

eg:

```
<div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
              <span class="sr-only">Toggle navigation</span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
              <span class="icon-bar"></span>
            </button>
            <a href="http://durgasoftonline.com" class="navbar-brand">DURGASOFT</a>
</div>
```

*****Note:**

To work Bootstrap navbar hamburger,compulsory Bootstrap Javascript must be required. But to work Bootstrap Javascript, jQuery must be required. Hence we have to include Bootstrap Java script and jQuery CDNs in our html.But order is important first jquery cdn followed by javascript cdn

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

We can write inside <body> tag anywhere.

Demo HTML:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
   <nav class="navbar  navbar-default navbar-fixed-top  navbar-inverse">
    <div class="container">
```

```
12)          <div class="navbar-header">
13)            <button type="button" class="navbar-toggle collapsed" data-
    toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
14)              <span class="sr-only">Toggle navigation</span>
15)              <span class="icon-bar"></span>
16)              <span class="icon-bar"></span>
17)              <span class="icon-bar"></span>
18)              <span class="icon-bar"></span>
19)              <span class="icon-bar"></span>
20)            </button>
21)            <a href="http://durgasoftonline.com" class="navbar-brand">DURGASOFT</a>
22)          </div>
23)          <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
24)                  <ul class="nav navbar-nav">
25)                    <li><a href="#">Home</a></li>
26)                    <li><a href="#">About Us</a></li>
27)                    <li><a href="#">Services</a></li>
28)                    <li><a href="#">Services</a></li>
29)                    <li><a href="#">Services</a></li>
30)                  </ul>
31)                  <ul class="nav navbar-nav navbar-right">
32)                    <li><a href="#">Contact Us</a></li>
33)                    <li><a href="#">Logout</a></li>
34)                  </ul>
35)          </div>
36)       </div>
37)
38)    </nav>
39)
40)
41)    <p></p><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br
    ><br>
42)    <nav class="navbar navbar-default">
43)   <div class="container-fluid">
44)    <!-- Brand and toggle get grouped for better mobile display -->
45)    <div class="navbar-header">
46)      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
    target="#bs-example-navbar-collapse-1" aria-expanded="false">
47)        <span class="sr-only">Toggle navigation</span>
48)        <span class="icon-bar"></span>
49)        <span class="icon-bar"></span>
50)        <span class="icon-bar"></span>
51)      </button>
52)      <a class="navbar-brand" href="#">Brand</a>
53)    </div>
54)
55)    <!-- Collect the nav links, forms, and other content for toggling -->
56)    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
57)      <ul class="nav navbar-nav">
```

```
58)       <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a></li>
59)       <li><a href="#">Link</a></li>
60)       <li class="dropdown">
61)         <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-
      haspopup="true" aria-expanded="false">Dropdown <span class="caret"></span></a>
62)         <ul class="dropdown-menu">
63)           <li><a href="#">Action</a></li>
64)           <li><a href="#">Another action</a></li>
65)           <li><a href="#">Something else here</a></li>
66)           <li role="separator" class="divider"></li>
67)           <li><a href="#">Separated link</a></li>
68)           <li role="separator" class="divider"></li>
69)           <li><a href="#">One more separated link</a></li>
70)         </ul>
71)       </li>
72)     </ul>
73)     <form class="navbar-form navbar-left">
74)       <div class="form-group">
75)         <input type="text" class="form-control" placeholder="Search">
76)       </div>
77)       <button type="submit" class="btn btn-default">Submit</button>
78)     </form>
79)     <ul class="nav navbar-nav navbar-right">
80)       <li><a href="#">Link</a></li>
81)       <li class="dropdown">
82)         <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-
      haspopup="true" aria-expanded="false">Dropdown <span class="caret"></span></a>
83)         <ul class="dropdown-menu">
84)           <li><a href="#">Action</a></li>
85)           <li><a href="#">Another action</a></li>
86)           <li><a href="#">Something else here</a></li>
87)           <li role="separator" class="divider"></li>
88)           <li><a href="#">Separated link</a></li>
89)         </ul>
90)       </li>
91)     </ul>
92)   </div><!-- /.navbar-collapse -->
93)  </div><!-- /.container-fluid -->
94) </nav>
95) <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
      q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="
      anonymous"></script>
96) <!-- Latest compiled and minified JavaScript -->
97) <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integ
      rity="sha384-
      Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossor
      igin="anonymous"></script>
98)
99)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
100)          </body>
101)         </html>
```

Bootstrap: Grid System

Grid System is the most fundamental and commonly used concept of Bootstrap.
To display our application layout properly on multiple devices of multiple screen sizes like desktop,laptop,tab,mobile etc,we should go for grid system.

In Bootstrap we can split total screen into 12 columns. Within this 12 columns length we can take any number of elements of same size or different sizes.

eg:
12X1==>12 elements and each element is of 1 column length
6X2===>6 elements and each element is of 2 column length
4X3===>4 elements and each element is of 3 column length
e1X3+e2X4+e3X5==>total 3 elements
First element is of 3 columns length
Second element is of 4 columns length
Third element is of 5 columns length

How to implement Grid:

We can implement grid by using 2 classes
1. row class: to define row

```
<div class="row">
```

2. Within the row we can define columns by using the following class
col-screenSize-noOfColumns

The allowed screen sizes in Bootstrap 3 are:
lg-->Large Size (like Desktop screens)
md-->Medium Size (like Laptop screens)
sm-->Small Size (like Tab screens)
xs-->Extra Small Size (like Mobile screens)

But in Bootstrap-4 Extra Large(xl) is also allowed.

eg:

```
1)    <div class="row">
2)       <div class="col-lg-3">Element-1</div>
3)       <div class="col-lg-3">Element-1</div>
4)       <div class="col-lg-3">Element-1</div>
5)       <div class="col-lg-3">Element-1</div>
6)    </div>
```

Here row contains 4 elements and each element taking 3 columns length.

demo.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <link rel="stylesheet" href="Bootstrap.css">
    <meta charset="utf-8">
    <title></title>
    <style>
    .box{
      background:orange;
      border:1px solid blue;
    }

    </style>

  </head>
  <body>
    <div class="container">
            <div class="row">
              <div class="col-lg-3 box">Element-1</div>
              <div class="col-lg-3 box">Element-2</div>
              <div class="col-lg-3 box">Element-3</div>
              <div class="col-lg-3 box">Element-4</div>
            </div>
    </div>

    </div>
  </body>
</html>
```

We can take elements of different column lengths also

```
<div class="row">
              <div class="col-lg-3 box">Element-1</div>
              <div class="col-lg-6 box">Element-2</div>
              <div class="col-lg-3 box">Element-3</div>
</div>
```

Note: The total column length per row should be:12

case-1: For medium or large screens each row should contains 4 elements

```
<div class="row">
              <div class="col-md-3 box">Element-1</div>
              <div class="col-md-3 box">Element-2</div>
              <div class="col-md-3 box">Element-3</div>
```

```
5)          <div class="col-md-3 box">Element-4</div>
6)  </div>
```

Case-2: On every screen(ls,md,sm,xs) row should contain 3 elements

```
1)  <div class="row">
2)          <div class="col-xs-4 box">Element-1</div>
3)          <div class="col-xs-4 box">Element-2</div>
4)          <div class="col-xs-4 box">Element-3</div>
5)  </div>
```

xs means either extra small or higher

How to define Grid for multiple screens simultaneously:

eg:

For Large screens 6 Elements
For Medium Screens 4 Elements
For Small Screens 3 Elements
For Extra Small Screens 2 Elements

```
1)  <div class="row">
2)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-1</div>
3)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-2</div>
4)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-3</div>
5)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-4</div>
6)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-5</div>
7)      <div class="col-lg-2 col-md-3 col-sm-4 col-xs-6 box">Element-6</div>
8)  </div>
```

Nested Grids:

Inside a grid,we can define another grid,which is nothing but Nested Grid.
Nested Grid: Grid inside Grid

Demo:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <link rel="stylesheet" href="Bootstrap.css">
5)      <meta charset="utf-8">
6)      <title></title>
7)      <style>
8)      .box{
9)        background:orange;
10)       border:1px solid blue;
```

```
11)   }
12)
13)   </style>
14)  </head>
15)  <body>
16)   <div class="container">
17)    <div class="row">
18)     <div class="col-lg-6 box">
19)      <div class="row">
20)       <div class="col-lg-4 box">Nested Element-1</div>
21)       <div class="col-lg-4 box">Nested Element-2</div>
22)       <div class="col-lg-4 box">Nested Element-3</div>
23)
24)      </div>
25)
26)     </div>
27)     <div class="col-lg-6 box">Element-2 </div>
28)    </div>
29)
30)   </div>
31)
32)  </body>
33) </html>
```

Bootstrap Photo Gallery Application:

step-1: Add css,jquery and java script cdn files to our html inside the <head> tag

```
1)  <head>
2)  <!-- To link our css to HTML -->
3)     <link rel="stylesheet" href="gallery2.css">
4)
5)  <!-- Latest compiled and minified CSS -->
6)     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
    otstrap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
    gin="anonymous">
7)
8)  <!-- For Font Awesome (icons) -->
9)     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.0/css/all.css" i
    ntegrity="sha384-
    lKuwvrZot6UHsBSfcMvOkWwlCMgc0TaWr+30HWe3a4ltaBwTZhyTEggF5tJv8tbt" crossorigi
    n="anonymous">
10) </head>
```

step-2: Add jumbotron to our HTML inside <body> tag

```
<div class="container">
    <div class="jumbotron">
      <h1> Heaven Photo Gallery</h1>
      <p>This is short cut to go to heaven... already crores of people tested... why don't yo
u test</p>
    </div>
  </div>
```

step-3: Add camera icon(fontawesome) inside <h1> tag of jumbotron

<h1> Heaven Photo Gallery</h1>

step-4: Add Navigation Bar to the html at the top

```
<nav class="navbar  navbar-default  navbar-fixed-top " id="xxx">
   <div class="container">
     <div class="navbar-header">
       <a href="http://durgasoftonline.com" class="navbar-
brand" id="yyy"> DURGABAR</a>
     </div>
     <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
       <ul class="nav navbar-nav">
     <li><a href="#">Home</a></li>
     <li><a href="#">About Us</a></li>
     <li><a href="#">Services</a></li>
       </ul>
       <ul class="nav navbar-nav navbar-right">
         <li><a href="#">Contact Us</a></li>
         <li><a href="#">Logout</a></li>
       </ul>
     </div>
   </div>
</nav>
```

step-5: Add picture icon(fontawesome) inside <a> tag of navbar for our brand

 DURGABAR

step-6: Add hamburger button to our navbar inside navbar-header class tag at the top for responsive navbar

```
<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#bs-example-navbar-collapse-1" aria-expanded="false">
   <span class="sr-only">Toggle navigation</span>
     <span class="icon-bar"></span>
```

```
4)        <span class="icon-bar"></span>
5)        <span class="icon-bar"></span>
6)        <span class="icon-bar"></span>
7)    </button>
```

step-7: Add Grid after jumbotron's tag

```
1)  <div class="row">
2)     <div class="col-lg-4 col-sm-6">
3)       <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
4)     </div>
5)     <div class="col-lg-4 col-sm-6">
6)       <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
7)     </div>
8)     <div class="col-lg-4 col-sm-6">
9)       <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
10)    </div>
11)    <div class="col-lg-4 col-sm-6">
12)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
13)    </div>
14)    <div class="col-lg-4 col-sm-6">
15)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
16)    </div>
17)    <div class="col-lg-4 col-sm-6">
18)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
19)    </div>
20)    <div class="col-lg-4 col-sm-6">
21)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
22)    </div>
23)    <div class="col-lg-4 col-sm-6">
24)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
25)    </div>
26)    <div class="col-lg-4 col-sm-6">
27)      <div class="thumbnail"><img src="heaven.jpg" alt=""> </div>
28)    </div>
29) </div>
```

gallery2.css

step-8: Add padding in between navbar and jumbotron

```
1.  body{
2.    padding-top: 70px;
3.  }
```

step-9: Add css styles to navbar

```
1)  .navbar{
2)    /* just remove navbar-default from navbar */
```

```
    background: blue;
}
.navbar a{
color:white;
}
```

step-10: Add css styles to jumbotron

```
.container .jumbotron {
    background:green;
    color:white;
}
```

==========

How to add Bootstrap Glyphicons to html:

Select the following tag and add to the html where ever it is required.

```
<span class="glyphicon glyphicon-camera" aria-hidden="true"></span>

<span class="glyphicon glyphicon-picture
" aria-hidden="true"></span>
```

class names will be changed from icon to icon

How to change jumbotron background color and text color:

```
.jumbotron{
  background-color: red;
  color: blue;
}
```

If it is not working then right click and inspect element on the element and choose the corresponding style.

```
.container .jumbotron, .container-fluid .jumbotron {
    padding-right: 15px;
    padding-left: 15px;
    border-radius: 6px;
    background:red;
}
```

How to add icons from fontawesome.com:

https://fontawesome.com/icons?from=io

Step-1: Add CDN to our html:

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.0/css/all.css" integrity="sha384-lKuwvrZot6UHsBSfcMvOkWwlCMgc0TaWr+30HWe3a4ltaBwTZhyTEggF5tJv8tbt" crossorigin="anonymous">

Step-2: Add i tag where ever icon is required

user-plus
icon to icon only class name will be changed (eg camera)

Durga Dating App:

dating.html

step-1: Add all CDNs(Bootstrap,jQuery,JavaScript) and css to our html

```
1) <head>
2)    <link rel="stylesheet" href="dating.css">
3)
4)    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.0/css/all.css" i
   ntegrity="sha384-
   lKuwvrZot6UHsBSfcMvOkWwlCMgc0TaWr+30HWe3a4ltaBwTZhyTEggF5tJv8tbt" crossorigi
   n="anonymous">
5)
6)    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
   otstrap.min.css" integrity="sha384-
   BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
   gin="anonymous">
7) </head>
```

step-2: Add Grid to our HTML with h1,h3 and hr tag

```
1)  <div class="container">
2)     <div class="row">
3)       <div class="col-lg-12">
4)         <div id="content">
5)           <h1 id="tc1">DURGA Dating App</h1>
6)           <h3 id="tc">Only for Seniour Citizens and Kids... Just for fun...</h3>
7)           <hr>
8)         </div>
9)       </div><!-- End col-lg-12 -->
10)    </div><!-- End row -->
11) </div><!-- End container -->
```

step-3: Add button to grid after <hr> tag

<button type="button" name="button" class="btn btn-default btn-lg"> Get Started</button>

step-4: Add fontawesome smile before "Get Started"

<button type="button" name="button" class="btn btn-default btn-lg"> Get Started</button>

Here   --> for one space

step-5: Add navbar to HTML at the top

```
1)  <nav class="navbar navbar-default">
2)      <div class="container">
3)       <!-- Brand and toggle get grouped for better mobile display -->
4)       <div class="navbar-header">
5)        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
    target="#bs-example-navbar-collapse-1" aria-expanded="false">
6)          <span class="sr-only">Toggle navigation</span>
7)          <span class="icon-bar"></span>
8)          <span class="icon-bar"></span>
9)          <span class="icon-bar"></span>
10)       </button>
11)       <a class="navbar-brand" href="#">DURGA Dating App</a>
12)      </div>
13)      <!-- Collect the nav links, forms, and other content for toggling -->
14)      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
15)       <ul class="nav navbar-nav">
16)        <li class="active"><a href="#">Home</a></li>
17)        <li><a href="#">About us</a></li>
18)        <li><a href="#">Services</a></li>
19)       </ul>
20)       <ul class="nav navbar-nav navbar-right">
21)        <li><a href="#">Sign up</a></li>
22)        <li><a href="#">Login</a></li>
23)       </ul>
24)      </div><!-- /.navbar-collapse -->
25)    </div><!-- /.container-fluid -->
26) </nav>
```

step-6: Add fontawesomes for Sign up and Login

Sign up
Login

dating.css:

step-7: Add css styles for Grid's content

```
#content {
  text-align: center;
  padding-top: 25%;
}
```

step-8: Add background image and it's css styles

```
body{
  background: url(https://images.unsplash.com/photo-1529610453335-db84df315b29?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=1336a860de7a33d2f9b4281a13a48b68&auto=format&fit=crop&w=400&q=60);
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
}
```

step-9: Add css styles for <h1> tag using tc1 id

```
#tc1{
  color:white;
  font-weight: 700;
  font-size: 5em;
}
```

step-10: Add css styles for <h3> tag using tc id

```
#tc{
  color:white;
}
```

step-11: Add 100% height for remove bottom space while xs size

```
html{
  height: 100%
}
```

step-12: Add css styles for hr tag

```
hr{
  width: 500px;
  border-bottom: 10px solid blue;
  border-top: 10px solid red;
}
```

==========

To get images
unsplash.com

****

Full Stack Web Development with Python and DJango

JavaScript

Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

JavaScript

HTML is for Nouns
like Anushka,input fields,buttons,forms etc

CSS is for Adjectives
Like Styling with colors,borders,background images etc

Java Script is for Verbs/Actions
Like Dance,Eat....

Java Script is Full pledged Programming Language.
The main purpose of java script is to add functionality(actions) to the HTML.

Usually we can Java Script in the Front-end and we can use Node JS for Back-end.

Agenda:

1. Java Script Developer's Console
2. The 5 Basic Javascript Primitive Data Types
3. Declaring variables with var keyword
4. The 3 most commonly used Java Script Functions

1)JavaScript Developer's Console

We can use this developer's console to test our java script coding snippets. This is just for testing purpose only and usaully not recommended for main coding.

How to launch Java Script Console:

Browser-->Right Click-->Inspect-->Console

Short-cut: Ctrl+Shift+j

Note:

1. To clear console we have to use clear() function.
2. ; at end of statement is not mandatory in newer versions

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

2)The 5 Basic JavaScript Primitive Data Types

Java Script defines the following 5 primitive data types

1. Numbers:

10
-10
10.5

All these are of "number" type
Java Script never cares whether it is integral or float-point or signed and unsigned.

General Mathematical operators are applicable for numbers

10+20
10-20
10/20
10*20
10%3
102**

General Operator precedence also applicable.
10+20*3===>70
10*(2+3)===>50

Note: We can check the type of variable by using typeof keyword

typeof x;

2. string:

Any sequence of characters within either single quotes or double quotes is treated as string.

'durga'
"durga"

We can apply + operator for Strings also and it acts as concatenation operator.

Rule: If both arguments are number type then + operator acts as arithmetic addition operator.
If atleast one argument is of string type then + operator acts as concatenation operator.

10+20==>30
'durga'+10==>durga10
'durga'+true==>durgatrue

We can use esacape characters also in the string.

Eg: 'durga\nsoft'
'durga\tsoft'
'This is \' symbol'
'This is \" symbol'
'This is \\ symbol'

Q. How to find the number of characters present in the string?

We can find by using length variable

Eg:
'durgasoft'.length

Q. How to access characters of the String?

By using index

index of first character is zero

'durga'[2]==>r
'durga'[200]===>undefined but no error
'durga'[-1] ==>undefined

Note: If we are trying to access string elements with out of range index or negative index then we will get undefined value and we won't get any Error.

3. boolean:

The only allowed values are: true and false (case sensitive)

3)JavaScript Variables

Variables are containers to store values.

Syntax:
var variableName=variableValue

Eg:
var name="durga"
var age=60
var isMarried=false

Note:
Java script variables follow CamelCase Convention

studentMobileNumber--->Camel case(Java,JavaScript etc)
student_mobile_number-->Snake Case(Python)
student-mobile-number-->Kebab Case(Lisp)
Based on provided value automatically type will be considered for variables.

Eg:
var x =10
typeof x ===>number
x = false
typeof x===>boolean

Hence Java Script is Dynamically Typed Programming Language like Python

null and undefined:

Variables that are declared but not initialized, are considered as undefined

Eg:

var x;
typeof x==>undefined

null value means nothing.
If the value of a variable null means that variable not pointing to any object.

var currentplayer='durga'
currentplayer=null //game over

The 3 most commonly used methods of Java Script:

1. alert():

To display alerts to the end user

```
alert('Hello there')
alert(100000)
alert(10.5)
```

2. console.log():

To print messages to the developer's console

Eg:

```
console.log('Hello there')
console.log(10*20)
```

These console message not meant for end user.

3. prompt():

To get input from the end user

```
prompt('What is Your Name:')
```

Here we are not saving the value for the future purpose. But we can save as follows

```
var name= prompt('What is Your Name:')
```

Based on our requirement we can use this name

```
console.log('Hello '+name+' Good Evening')
alert('Hello '+name+' Good Evening')
```

How to write Javascript to a seperate file and connect to html:

demo.js:

```
alert('Hello everyone good evening')
```

html:

We can link javascript file to html by using the following <script> tag.

```
<script type="text/javascript" src="demo.js"></script>
```

We can take this script tag either inside head tag or body tag. If we are taking inside head tag then javascript code will be exeucted before processing body.

If we are taking inside body tag then javascript code will be executed as the part of body execution.

Demo Application: Age and Death Calculator:

demo.js:

```
1) var name=prompt('Enter Your Name:');
2) var age=prompt('Enter Your Age:');
3) agedays=age*365.25
4) remainingdays=(60-age)*365.25;
5) alert("Hello "+name+"...\nYour Current Age:"+agedays+" days\nYou will be there on the e
   arth only "+remainingdays+" days. No one can change including God also");
```

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>The power of Java Script</h1>
9)      <script type="text/javascript" src="demo.js"> </script>
10)   </body>
11) </html>
```

Operators:

1. Arithmetic Operators:

+,-,*,/,%,**

2. Comparison Operators:

<,<=,>,>=,==,!=,===, !==

10<20===>true
10<=20 ==>true
10>20 ===>false
10>=20 ==>false
10==20==>false
10 != 20 ==>true

Difference between == and ===:

In the case of == operator internally type coersion will be performed. Hence if arguments are different types first both arguments will be converted to same type and then comparison will be performed. Hence both arguments need not be same type.

Eg:

10 == "10" ====>true
10==10 ==>true

Here only values are important but not types.

But in the case === operator,type coersion won't be performed. Hence argument types must be same,otherwise we will get false.

10 === 10 ==>true
10 === "10" ===>false

Here both content and type are important.

Note:

== -->Normal equality operator where types are not important but values must be same

=== --->Strict equality operator where both types and values must be same

It is recommended to use === operator because it is more safer and more specific.

Example:

true=="1" ===>true
false=="0" ===>true
null==undefined ===>true
true==="1" ===>false
false==="0" ===>false
null===undefined ===>false

NaN(Not a Number):

If the result is undefined then we will get NaN

Eg: 0/0 ===>NaN

for any x value including NaN the following expressions returns false

x<NaN
x<=NaN
x>NaN
x>=NaN
x==NaN

For any x value including NaN the following expression returns true

x != NaN

Eg:

NaN==NaN ===>false
NaN != NaN ===>true

Logical Operators:

&& -->AND
|| -->OR
! --->Not

X && Y==>If both arguments are true then only result is true. i.e if atleast one argument is false then the result is always false

X || Y ===>If atleast one argument is true then the result is true. i.e if both arguments are false then only result is false.

Note: For Logical Operators

1. zero value always treated as false
non-zero value always treated as true

2. empty string treated as false where as non-empty string treated as true

3. null,undefined,NaN are treated as false

Examples:

```
var x =10;
var y =20;

x<10 && x != 5  ===>false

y>9 || x== 10 ==>true

!(x==y)  ==>true

!(x=="10" || x=== y) && !( y!= 8 && x<=y)) ==>false

!(x !==1)&& y === "20" ==>false
```

Conditional Statements:

Based on available options, one option will be selected and executed in conditional statements/selection statements.

1. if
2. if else
3. else if

Syntax:

```
if(b){
  action if b is true;
}
else{
 action if b is false;
}
```

Eg 1: Write Java Script code to check given number is even or not?

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
```

```
5)      <script type="text/javascript" src="demo.js"></script>
6)
7)     <title></title>
8)    </head>
9)    <body>
10)    <h1>The power of Java Script</h1>
11)
12)   </body>
13)
14) </html>
```

demo.js:

```
1)  var num = Number(prompt('Enter Any Number:'))
2)  if(num%2===0){
3)    console.log('Given Number is Even')
4)    alert('Given Number is Even')
5)  }
6)  else{
7)    console.log('Given Number is Odd')
8)    alert('Given Number is Odd')
9)  }
```

Q2. Write Java Script code to print proper meaningful message based on provided age regarding matrimonial website?

demo.js:

```
1)  var age = Number(prompt('Enter Your Age:'))
2)  if(age>60){
3)    alert('Plz wait some more time..Defenitely You will get Best Match')
4)  }
5)  else if(age<18){
6)    alert("Your age already crossed marriage age..No chance of getting marriage")
7)  }
8)  else{
9)      alert('Thanks for registration..You will get match details soon by email')
10) }
```

Q3. Write Java Script code to print proper meaningful message based on provided brand regarding beer application?

```
1)  var brand = prompt('Enter Your Favourate Brand:')
2)  if(brand=="KF"){
3)    alert("It is Children's Brand")
4)  }
5)  else if(brand=="KO"){
6)    alert("It is too light")
```

```
}
else if(brand=="RC"){
  alert("It is not that much kick")
}
else if(brand=="FO"){
  alert("Buy One get One FREE")
}
else{
   alert('Other brands are not recommended')
}
```

Q. Number Guess Application:

```
var sno=4
var num = Number(prompt('Enter your guess between 1 to 9:'))
if(num>sno){
  alert("It is too high..Guess again")
}
else if(num<sno){
  alert("It is too low guess again")
}
else{
   alert('Your Guess Matched')
}
```

Iterative Statements:

If we want to execute a group of statements iteratively, then we should go for iterative statements.

DRY Principle: Don't Repeat Yourself
It is highly recommended to follow DRY principle, otherwise it increases development time
It increases length of the code and reduces readability

In JavaScript there are 2 types of iterative statements
1. While Loop
2. For Loop

1. While Loop:

As long as some condition is true execute code then we should go for while loop.

Syntax:

```
while(condition){
        body
}
```

Eg 1: To print Hello 10 times to the console

demo.js:

```
1)  var count=1
2)  while(count<=10){
3)    console.log("Hello")
4)    count++
5)  }
```

Eg 2: To print first 10 numbers

demo.js:

```
1)  var count=1
2)  while(count<=10){
3)    console.log(count)
4)    count++
5)  }
```

Eg 3: To print each character present inside a string?

demo.js:

```
1)  var s="durga"
2)  var i =0
3)  while(i<s.length){
4)    console.log(s[i])
5)    i++
6)  }
```

Eg 3: To print all numbers divisible by 3 AND 5 between 5 and 100?

demo.js:

```
1)  var n=5
2)  while(n<=100){
3)    if(n%3==0 && n%5==0){
4)      console.log(n)
5)    }
6)    n++
7)  }
```

Eg 4: Write program to read actress name from the end user until entering 'sunny' by using while loop.

```
1)  var name=prompt("Enter Your Favourite Actress:")
2)  while(name !== "sunny"){
```

```
3)   name=prompt("Enter Your Favourite Actress:")
4)  }
5)  alert("Thanks for Confirmation as your Favourite actress: Sunny")
```

Note: If we don't know the number of iterations in advance and if we want to execute body as long as some condition is true then we should go for while loop.

Iterative Statements : For Loop:

If we know the number of iterations in advance then we should use for loop.

Syntax:

```
for(initialization section; conditional check; increment/decrement section)
{
  body;
}
```

Eg 1: To print "Hello" 10 times

```
for(var i=0;i<10;i++){
 console.log("Hello");
}
```

Eg 2: To print First 1 to 10

```
for(var i=1;i<=10;i++){
 console.log(i);
}
```

Eg 3: To print all numbers which are divisible by 7 from 1 to 100:

```
for(var i=1;i<=100;i++){
 if(i%7==0){
  console.log(i)
 }
}
```

Eg 4: To print each character from the given string

```
var word=prompt("Enter Some Word:")
for(var i=0;i<word.length;i++){
 console.log(word[i])
}
```

while vs for loop:

If we don't know the number of iterations in advance and as long as some condition is true keep on execute body then we should go for while loop.

If we know the number of iterations in advance then we should use for loop.

Secret Agent Application:

Rules:

1. The first character of Name should be 'd'
2. The Last character of Favourite Actor should be 'r'
3. The lucky number should be 7
4. The length of the dish should be >=6

If the above conditions are satisfied then user is valid secret agent and share information about operation, otherwise just send thanks message.

demo.js:

```
1)  var name=prompt("Enter Your Name:")
2)  var actor=prompt("Enter Your Favourite Actor:")
3)  var lucky=prompt("Enter Your Lucky Number:")
4)  var dish=prompt("Enter Your Favourite Dish:")
5)
6)  var nameConition=false
7)  var actorCondition=false
8)  var luckyConition=false
9)  var dishConition=false
10)
11) if(name[0]=="d") {
12)   nameConition=true
13) }
14) if(actor[actor.length-1]=="r"){
15)   actorCondition=true
16) }
17) if(lucky==7){
18)   luckyConition=true
19) }
20) if(dish.length>=6){
21)   dishConition=true
22) }
23) alert("Hello:"+name+"\nThanks For Your Information")
24) if(nameConition && actorCondition && luckyConition && dishConition){
25)   console.log("Hello Secret Agent our next operation is:")
26)   console.log("We have to kill atleast 10 sleeping students in the class room b'z these are b
    urdent to country")
27) }
```

Functions

If any piece of code repeatedly required in our application,then it is not recommended to write that code seperately every time. We have to seperate that code into a function and we can call that function where ever it is required.

Hence the main advantage of functions is code Reusability.

Syntax of Java Script function:

```
function functionName(arguments){
  body
  return value;
}
```

Eg 1: To print "Good Morning" message

```
1)  function wish(){
2)    console.log("Good Morning!!!")
3)  }
4)
5)  wish()
6)  wish()
7)  wish()
```

Functions with Arguments:

A function can accept any number of arguments and these are inputs to the function.
Inside function body we can use these arguments based on our requirement.

Eg: Write a function to accept user name as input and print wish message.

```
1)  function wish(name){
2)    console.log("Hello "+name+" Good Morning!!!")
3)  }
4)
5)  var name= prompt("Enter Your Name:")
6)  wish(name)
```

Functions with default arguments:

We can provide default values for arguments. If we are not passing any value then only default values will be considered.

Eg:

```
1)  function wish(name="Guest"){
2)    console.log("Hello "+name+" Good Morning!!!")
3)  }
4)
5)  wish("Durga")
6)  wish()
```

Function with return values:

A function can return values also.

Eg: Write a Javascript function to take a number as argument and return its square value

```
1)  function square(num){
2)    return num*num;
3)  }
4)  var result=square(4)
5)  console.log("The Square of 4:"+result)
6)  console.log("The Square of 5:"+square(5))
```

Eg: Write a Javascript function to take 2 numbers as arguments and return sum.

```
1)  function sum(num1,num2){
2)    return num1+num2;
3)  }
4)  var result=sum(10,20)
5)  console.log("The sum of 10,20 :"+result)
6)  console.log("The sum of 100,200 :"+sum(100,200))
```

Eg: Write a Javascript function to take a string as argument and return Capitalized string.

```
1)  function capitalize(str){
2)    return str[0].toUpperCase()+str.slice(1);
3)  }
4)  console.log(capitalize('sunny'))
5)  console.log(capitalize('bunny'))
```

Eg: Write a Javascript function to check whether the given number is even or not?

```
1)  function isEven(num){
2)    if(num%2==0){
```

```
3)     return true;
4)    }
5)    else{
6)     return false;
7)    }
8)  }
9)  console.log(isEven(15))
10) console.log(isEven(10))
```

Eg: Write javascript function to find factorial of given number?

```
1)  function factorial(num){
2)    result=1;
3)    for (var i = 2; i <= num; i++) {
4)     result=result*i;
5)    }
6)    return result;
7)  }
8)  console.log("The Factorial of 4 is:"+factorial(4))
9)  console.log("The Factorial of 5 is:"+factorial(5))
```

Eg: Write a javascript function to convert from Snake case to Kebab case of given string.

snake case: total_number
Kebab case: total-number

```
1)  function snakeToKebab(str){
2)    var newstring=str.replace('_','-')
3)    return newstring;
4)  }
5)  console.log(snakeToKebab('total_number'))
```

Note: Inside function if we are writing any statement after return statement,then those statements won't be executed, but we won't get any error.

Eg:

```
1)  function square(n){
2)    return n*n;
3)    console.log("Function Completed!!!")
4)  }
5)  console.log(square(4));
```

Output: 16

JavaScript Scopes:

In Javascript there are 2 scopes.

1. Global Scope
2. Local Scope

1. Global Scope:

The variables which are declared outside of function are having global scope and these variables are available for all functions.

Eg 1:

```
1)  var x=10
2)  function f1(){
3)    console.log(x);
4)  }
5)  function f2(){
6)    console.log(x);
7)  }
8)  f1();
9)  f2();
```

2. Local Scope:

The variables which are declared inside a function are having local scope and are available only for that particular function. Outside of the function we cannot these local scoped variables.

Eg 2:

```
1)  function f1(){
2)    var x=10
3)    console.log(x);//valid
4)  }
5)  f1();
6)  console.log(x);//Uncaught ReferenceError: x is not defined
```

Eg 3:

```
1)  var x=10
2)  function f1(){
3)    x=777;
4)    console.log(x);
5)  }
6)  function f2(){
7)    console.log(x);
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
8)  }
9)  f1();
10) f2();
```

Output:

```
777
777
```

Eg 4:

```
1)  var x=10
2)  function f1(){
3)    x=777;
4)    console.log(x);
5)  }
6)  function f2(){
7)    console.log(x);
8)  }
9)  f2();
10) f1();
```

Output:

```
10
777
```

Eg 5:

```
1)  var x=10
2)  function f1(){
3)    var x=777;
4)    console.log(x);
5)  }
6)  function f2(){
7)    console.log(x);
8)  }
9)  f1();
10) f2();
```

Output:

```
777
10
```

Q. If local and global variables having the same name then within the function local variable will get priority.How to access global variable?

Higher Order Functions:

We can pass a function as argument to another function.A function can return another function. Such type of special functions are called Higher Order Functions.

Eg: setInterval()

setInterval(function, time_in_milliseconds)
The provided function will be executed continously for every specified time.

setInterval(singAsong, 3000)

singAsong function will be executed for every 3000 milli seconds.

We can stop this execution by using clearInterval() function.

Eg: demo.js

```
1)  function singAsong(){
2)    console.log('Rangamma...Mangamma..')
3)    console.log('Jil..Jil...Jigel Rani..')
4)  }
```

On developer's console:

```
setInterval(singAsong,3000)
1
demo.js:2 Rangamma...Mangamma..
demo.js:3 Jil..Jil...Jigel Rani..
demo.js:2 Rangamma...Mangamma..
demo.js:3 Jil..Jil...Jigel Rani..
demo.js:2 Rangamma...Mangamma..
demo.js:3 Jil..Jil...Jigel Rani..
clearInterval(1)
undefined
```

Anonymous Functions:

Some times we can define a function without name, such type of nameless functions are called anonymous functions.

The main objective of anonymous functions is just for instant use (one time usage)

Eg:

```
setInterval(function(){console.log("Anonymous Function");},3000);
8
Anonymous Function
Anonymous Function
Anonymous Function
Anonymous Function
..
clearInterval(8);
```

Coding Examples from codebat.com:

Problem-1: sleep_in

Write a function called sleep_in that takes 2 boolean parameters: weekday and vacation.

The parameter weekday is True if it is a weekday, and the parameter vacation is True if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return True if we sleep in.

```
sleep_in(false, false) --> true
sleep_in(true, false) --> false
sleep_in(false, true) --> true
```

```
1)  function sleep_in(weekday,vacation) {
2)   return !weekday || vacation;
3)  }
4)  console.log("Is Employee Sleeping:"+sleep_in(true,true))
5)  console.log("Is Employee Sleeping:"+sleep_in(true,false))
6)  console.log("Is Employee Sleeping:"+sleep_in(false,true))
7)  console.log("Is Employee Sleeping:"+sleep_in(false,false))
```

Problem-2: monkey_trouble

We have two monkeys, a and b, and the parameters a_smile and b_smile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return True if we are in trouble.

```
monkey_trouble(true, true) --> true
monkey_trouble(false, false) --> true
```

monkey_trouble(true, false) --> false

Solution:

```
1)  function monkey_trouble(aSmile,bSmile){
2)   return (aSmile && bSmile) || (!aSmile && !bSmile)
3)  }
4)  console.log("Is Person In Trouble:"+monkey_trouble(true,true))
5)  console.log("Is Person In Trouble:"+monkey_trouble(true,false))
6)  console.log("Is Person In Trouble:"+monkey_trouble(false,true))
7)  console.log("Is Person In Trouble:"+monkey_trouble(false,false))
```

Output:

```
Is Person In Trouble:true
demo.js:5 Is Person In Trouble:false
demo.js:6 Is Person In Trouble:false
demo.js:7 Is Person In Trouble:true
```

Problem-3: Warmup-2 > string_times

Given a string and a non-negative int n, return a larger string that is n copies of the original string.

string_times('Hi', 2) --> 'HiHi'
string_times('Hi', 3) --> 'HiHiHi'
string_times('Hi', 1) --> 'Hi'

Solution:

```
1)  function string_times(str,n){
2)   result="";
3)   var count=1;
4)   while(count<=n){
5)    result=result+str;
6)    count++;
7)   }
8)   return result;
9)  }
10) console.log(string_times("durga",3))
11) console.log(string_times("hello",2))
```

Output:

```
durgadurgadurga
hellohello
```

Problem-4: Logic-2 > lucky_sum

Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.

lucky_sum(1, 2, 3) --> 6
lucky_sum(1, 2, 13) --> 3
lucky_sum(1, 13, 3) --> 1

Solution:

```
1)  function lucky_sum(a,b,c){
2)    if(a==13){
3)      return 0;
4)    }
5)    if(b==13){
6)      return a;
7)    }
8)    if(c==13){
9)      return a+b;
10)   }
11) }
12) console.log(lucky_sum(13,10,5))//0
13) console.log(lucky_sum(5,13,6))//5
14) console.log(lucky_sum(7,5,13))//12
```

Problem-5:Logic-1 > caught_speeding

You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases.

caught_speeding(60, False) --> 0
caught_speeding(65, False) --> 1
caught_speeding(65, True) --> 0

Solution:

```
1)  function caught_speeding(speed,isBirthday){
2)    if (isBirthday) {
3)      speed=speed-5;
4)    }
5)    if (speed<=60) {
6)      return 0;
7)    }
8)    else if (speed>=61 && speed<=80) {
```

```
9)      return 1;
10)   }
11)   else{
12)     return 2;
13)   }
14) }
15) console.log("Getting Ticket With Number:"+caught_speeding(60, false))//0
16) console.log("Getting Ticket With Number:"+caught_speeding(65, false))//1
17) console.log("Getting Ticket With Number:"+caught_speeding(65, true))//0
```

JavaScript Arrays:

An array is an indexed collection of elements.

The main advantage of arrays concept is we can represent multiple values by using a single variable so that length of the code will be reduced and readability will be improved.

Without arrays:

```
var n1=10;
var n2=20;
var n3=30;
var n4=40;
```

With arrays:

```
var numbers=[10,20,30,40]
```

Accessing Array Elements by using index:

By using index we can access array elements. Javascript arrays follow 0-based index. i.e The index of first element is 0

Eg:

```
var friends=["durga","sunny","bunny","chinny"];
console.log(friends[0]); //durga
console.log(friends[3]); //chinny
console.log(friends[30]); //undefined
```

Note: If we are trying to access array elements by using out of range index then we will get undefined value and we won't get any error.

Updating array elements by using index:

```
var friends=["durga","sunny","bunny","chinny"];
friends[1]="mallika"
console.log(friends)// ["durga", "mallika", "bunny", "chinny"]
```

Adding new elements to the array by using index:

```
var friends=["durga","sunny","bunny","chinny"];
friends[4]="vinny";
console.log(friends)//["durga","sunny","bunny","chinny","vinny"]
friends[40]="pinny";
console.log(friends)// ["durga", "sunny", "bunny", "chinny", "vinny", empty × 35, "pinny"]
```

Note: By using index we can retrieve,update and add elements of array. But in general we can use index to access array elements.

How to create an empty array:

1st way: var numbers=[];
2nd way: var numbers=new Array();

How to find length of array:

By using length variable

```
var friends=["durga","sunny","bunny","chinny"];
console.log(friends.length)//4
```

Is Javascript array can hold only homogeneous elements?

Javascript array can hold heterogeneous elements also.

Eg:

```
var random_collection=["durga",10000,true,null]
```

Important Methods related to Javascript arrays:

Java script defines several methods which are applicable on arrays.
Being a programmer we can use these methods directly and we are not responsible to implement so that our life will become very easy.

The following are important methods

1. push()
2. pop()

3. unshift()
4. shift()
5. indexOf()
6. slice()

1. push():

We can use push() method to add elements to the end of array. After adding element this method returns length of the array.

Eg:

```
var numbers=[10,20,30,40]
numbers.push(50)
console.log(numbers)// [10, 20, 30, 40, 50]
```

2. pop():

We can use pop() method to remove and return last element of the array

```
var numbers=[10,20,30,40]
console.log(numbers.pop())// 40
console.log(numbers.pop())// 30
console.log(numbers)// [10,20]
```

3. unshift():

We can use unshift() method to add element in the first position. It is counter part of push() method.

```
var numbers=[10,20,30,40]
numbers.unshift(50)
console.log(numbers)//[50, 10, 20, 30, 40]
```

4. shift():

We can use shift() method to remove and return first element of the array. It is counter part to pop() method.

Eg:

```
var numbers=[10,20,30,40]
numbers.shift()
console.log(numbers)//[20, 30, 40]
```

5. indexOf():

We can use indexOf() to find index of specified element.
If the element present multiple times then this method returns index of first occurrence.

If the specified element is not available then we will get -1.

Eg:

```
var numbers=[10,20,10,30,40];
console.log(numbers.indexOf(10))//0
console.log(numbers.indexOf(50))// -1
```

6. slice():

We can use slice operator to get part of the array as slice.
slice(begin,end)===>returns the array of elements from begin index to end-1 index.
slice()==>returns total array.This can be used for cloning purposes.

Eg:

```
var numbers=[10,20,30,40,50,60,70,80]
var num1=numbers.slice(1,5)
console.log(num1)// [20, 30, 40, 50]
num2=numbers.slice()
console.log(num2)// [10, 20, 30, 40, 50, 60, 70, 80]
```

Multi dimensional Arrays:

Sometimes array can contain arrays as elements.i.e array inside array is possible. Such type of arrays are considered as multi dimensional arrays or nested arrays.

Eg:

```
var nums=[[10,20,30],[40,50,60],[70,80,90]]
console.log(nums[0])//[10,20,30]
console.log(nums[0][0])//10
```

Book Management Application:

demo.js:

```
1)  var books=[]
2)  var input=prompt("Which operation You want to perform [add|list|exit]:")
3)  while (input != "exit") {
4)    if (input=="add") {
5)      var newBook= prompt("Enter Name of the Book:")
6)      books.push(newBook);
7)    }
8)    else if (input=="list") {
9)      console.log("List Of Available Books:");
10)     console.log(books);
11)   }
12)   else {
13)     console.log("Enter valid option");
```

```
14)  }
15)  input=prompt("What operation You want to perform [add|list|exit]:")
16) }
17) console.log("Thanks for using our application");
```

Retrieving Elements of Array:

We can retrive elements of array by using the following ways

1. while loop
2. for loop
3. for-of loop
4. forEach method

1. while loop:

```
1)  var nums=[10,20,30,40,50]
2)  var i=0;
3)  while (i<nums.length) {
4)    console.log(nums[i]);
5)    i++;
6)  }
```

2. for loop:

```
1)  var nums=[10,20,30,40,50]
2)  for (var i = 0; i < nums.length; i++) {
3)    console.log(nums[i]);
4)    //alert(nums[i]);
5)  }
```

3. for-of loop:

It is the convinient loop to retrieve elements of array.

Eg:

```
1)  var colors=["red","blue","yellow"]
2)  for (color of colors) {
3)    console.log('******************');
4)    console.log(color);
5)    console.log('******************');
6)  }
```

4. forEach Method:

forEach() is specially designed method to retrieve elements of Array.

Syntax:
arrayobject.forEach(function)

For every element present inside array the specified function will be applied.

Eg 1:

```
1) var heroines=['sunny','mallika','samantha','katrina','kareena']
2) function printElement(element){
3)   console.log('**********************');
4)   console.log(element);
5)   console.log('**********************');
6) }
7) heroines.forEach(printElement)
```

Eg 2:

```
1) var heroines=['sunny','mallika','samantha','katrina','kareena']
2) heroines.forEach(function (element) {
3)   console.log('**********************');
4)   console.log(element);
5)   console.log('**********************');
6) })
```

Note: The following are also valid

```
heroines=['sunny','mallika','samantha','katrina','kareena']
heroines.forEach(console.log)
heroines.forEach(alert)
```

for loop vs forEach function:

1. for loop is general purpose loop and applicable everywhere. But forEach() function is applicable only for arrays

2. By using for loop, we can move either forward direction or backward direction. But by using forEach() function we can move only forward direction.

Eg: By using for loop we can print array elements either in original order or in reverse order.But by using forEach() function we can print only in original order.

How to delete array elements based on index:

We have to use splice() function.

Syntax:

arrayobject.splice(index,numberofElements)
It deletes specified number of elements starts from the specified index.

Eg:

```
var heroines=['sunny','mallika','samantha','katrina','kareena']
heroines.splice(3,1)
console.log(heroines);//["sunny", "mallika", "samantha", "kareena"]
```

Immutability vs Mutability:

Once we creates an array object,we are allowed to change its content.Hence arrays are Mutable.

Eg:

```
var numbers=[10,20,30,40]
numbers[0]=777
console.log(numbers)//[777,20,30,40]
```

Once we creates string object,we are not allowed to change the content.If we are trying to change with those changes a new object will be created and we cannot change content of existing object. Hence string objects are immutable.

Eg:

```
var name='Sunny'
name[0]='B'
console.log(name)// Sunny
```

Note: Mutability means changeable where as Immutable means Non-Changeable.

Q1. Write a Javascript function to take an array as argument and print its elements in reverse order?

```
1)  function reverse(array){
2)    console.log('Elements in Reverse Order:')
3)    for (var i = array.length-1; i >=0 ; i--) {
4)      console.log(array[i])
5)    }
6)  }
7)  reverse([10,20,30,40,50])
8)  reverse(['A','B','C','D','E'])
```

Output:

```
Elements in Reverse Order:
50
40
30
20
10
Elements in Reverse Order:
E
D
C
B
A
```

Q2. Write a Javascript function to check whether the elements of given array are identical(same) or not?

```
1)  function identical(array){
2)  var first=array[0]
3)  for (var i = 1; i < array.length; i++) {
4)    if (array[i] != first) {
5)      return false;
6)    }
7)  }
8)  return true;
9)  }
10) console.log(identical([10,10,10,10]));//true
11) console.log(identical([10,20,30,40]));//false
```

Q3. Write a Javascript function to find maximum value of the given array?

```
1)  function max(array){
2)  var max=array[0]
3)  for (var i = 1; i < array.length; i++) {
4)    if (array[i] > max) {
5)      max=array[i]
6)    }
7)  }
8)  return max
9)  }
10) console.log(max([10,20,30,40]));//40
```

Q4. Write a Javascript function to find the sum of elements present in given array?

```
1)  function sum(array){
2)  var sum=0
3)  for (num of array) {
4)    sum+=num
5)  }
6)  return sum
7)  }
8)  console.log(sum([10,20,30,40]));//100
```

Book Management Application:

```
1)  var books=[]
2)  var input=prompt("Which operation You want to perform [add|delete|list|exit]:")
3)  while (input != "exit") {
4)    if (input=="add") {
5)      addBook();
6)    }
7)    else if (input=="list") {
8)      listBooks()
9)    }
10)   else if(input=="delete"){
11)     deleteBook()
12)   }
13)   else {
14)     console.log("Enter valid option");
15)   }
16)   input=prompt("What operation You want to perform [add|delete|list|exit]:")
17) }
18) console.log("Thanks for using our application");
19)
20) function addBook(){
21)   var newBook= prompt("Enter Name of the Book:")
22)   books.push(newBook);
23) }
24) function listBooks(){
25)   console.log("List Of Available Books:");
26)   for (book of books) {
27)     console.log(book);
28)   }
29) }
30) function deleteBook(){
31)   var name=prompt("Enter Book Name to delete:")
32)   var index=books.indexOf(name)
33)   if(index== -1){
34)     console.log("Specified book is not available");
```

```
35)  }
36)  else{
37)    books.splice(index,1)
38)    console.log("Specified Book Deleted");
39)  }
40) }
```

JavaScript Objects:

By using arrays we can store a group of individual objects and it is not possible to store key-value pairs.

If we want to represent a group of key-value pairs then we should go for Objects.

Array: A group of individual objects
Object: A group of key-value pairs

JavaScript objects store information in the form of key-value pairs.
These are similar to Java Map objects and Python Dictionary objects.

Syntax:

var variableName={ key1:value1,key2:value2,...};

Eg:

```
1)  var movie={
2)            name:'Bahubali',
3)          year: 2016,
4)          hero:'prabhas'
5)       };
```

In the case of JavaScript objects, no guarentee for the order and hence index conept is not applicable.

How to access values from Object:

We can access values by using keys in the following 2 ways

1. obj["key"]
Here quotes are mandatory
Eg: movie["hero"] ===>valid
movie[hero]====>Uncaught ReferenceError: hero is not defined

2. obj.key
Here we cannot take quotes
Eg: movie.hero

How to create and initialize JavaScript objects:

1. To create empty object

```
var nums={}   or
var nums=new Object()
```

Once we creates empty object we can add key-value pairs as follows

1st way:

```
nums["fno"]=100
nums["sno"]=200
```

2nd way:

```
nums.fno=100
nums.sno=200
```

How to update values:

```
nums["fno"]=999 or
nums.fno=999
```

Iterating Objects:

To access all key-value pairs we can use for-in loop

Eg:

```
1)  var nums={fno=100,sno=200,tno=300}
2)     for(key in nums){
3)        console.log(key); //To print only keys
4)        console.log(nums[key]); //To print only values
5)        console.log(key+":"+nums[key]); //To print both key and values
6)     }
```

Differences between Arrays and Objects:

Arrays	Object
1) Arrays can be used to represent individual values	1) Objects can be used to represent key-value pairs
2) Order will be maintained in Arrays	2) Order concept not applicable for Objects
3) By using index we can access and update data in arrays	3) By using key we can access and update data in Objects

Nested Objects and Arrays:

Inside Array, we can take objects. Similarly inside Objects we can take array.
Hence nesting of objects and arrays is possible.

Eg 1:

```
1)  var movies=[{name:'Bahubali',year:2016,hero:'Prabhas'},
2)          {name:'Sanju',year:2018,hero:'Ranveer'},
3)          {name:'Spider',year:2017,hero:'Mahesh'}
4)       ]
```

movies[0]["hero"] ===>Prabhas
movies[2]["year"] ===>2017

Eg 2:

```
1)  var numbers={
2)            fg:[10,20,30],
3)            sg:[40,50,60],
4)            tg:[70,80,90]
5)       }
```

numbers.sg[2] ==>60
numbers.tg[1] ==>80

Object Methods:

Java script object can contain methods also.

Eg:

```
1)  var myobj={
2)           A:'Apple',
3)        B:'Banana',
4)        m1:function(){console.log("Object Method");}
5)       }
```

We can invoke this method as follows:

myobj.m1()

this keyword:

Inside object methods,if we want to access object properties then we should use 'this' keyword.

```
1)  var movie={
2)              name:'Bahubali',
3)          year: 2016,
4)          hero:'prabhas',
5)          getInfo:function(){
6)          console.log('Movie Name:'+this.name);
7)          console.log('Released Year:'+this.year);
8)          console.log('Hero Name:'+this.hero);
9)          }
10)     };
11)
12) movie.getInfo()
```

Output:

Movie Name:Bahubali
Released Year:2016
Hero Name:prabhas

It is possible to refer already existing function as object method.

Eg 1:

```
1)  function demo(){
2)    console.log('Demo Function');
3)    }
4)
5)  var movie={
6)              name:'Bahubali',
7)          year:2016,
8)          hero:'Prabhas',
9)          getInfo:demo
10)       };
11) movie.getInfo()
```

Output:

Demo Function

Eg 2:

```
1)  function demo(){
2)    console.log('Demo Function:'+this.name);
3)    }
4)
5)  var movie={
```

```
6)          name:'Bahubali',
7)        year:2016,
8)        hero:'Prabhas',
9)        getInfo:demo
10)      };
11)  movie.getInfo()
```

Output:

Demo Function:Bahubali

If we call demo() function direclty then output is:

Demo Function

We can use named functions also.

```
1)  var movie={
2)          name:'Bahubali',
3)        year:2016,
4)        hero:'Prabhas',
5)        getInfo: function demo(){
6)     console.log('Demo Function:'+this.name);
7)     }
8)        };
```

Even we are not required to use function keyword also for object methods inside object and we can declare function directly without key.[But outside of object compulsory we should use function keyword to define functions]

```
1)  var movie =
2)  {
3)     name:"Rockstar",
4)     hero:"Ranbeer Kapoor",
5)     year:"2012",
6)     myFunction(){
7)        console.log("kjdf");
8)     }
9)  }
```

Even we can pass parameters also

```
1)  var movie =
2)  {
3)     name:"Rockstar",
4)     hero:"Ranbeer Kapoor",
5)     year:"2012",
6)     myFunction(a){
7)        console.log("kjdf:"+a);
8)     }
9)  }
```

```
10)
11) var a =10;
12) movie.myFunction(a)
```

Mini Application:

```
1)  var movies=[{name:'Bahubali',isWatched:'true',isHit:'true'},
2)          {name:'Sanju',isWatched:'false',isHit:'true'},
3)          {name:'Spider',isWatched:'true',isHit:'false'},
4)       ]
5)
6)  movies.forEach(function(movie){
7)   var result=""
8)   if(movie.isWatched=="true"){
9)   result=result+"I Watched "
10) }
11) else{
12) result=result+"I have not seen "
13) }
14) result=result+movie.name
15) if(movie.isHit=="true"){
16) result=result+" and Movie is Hit!!!"
17) }
18) else{
19) result=result+" and Movie is Flop!!!"
20) }
21) console.log(result)
22) });
```

Output:

I Watched Bahubali and Movie is Hit!!!
I have not seen Sanju and Movie is Hit!!!
I Watched Spider and Movie is Flop!!!

Full Stack Web Development with Python and DJango

DOM

Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

Document Object Model (DOM)

DOM acts as interface between JavaScript and HTML, CSS.
i.e By using DOM, javascript can communicate/connect with HTML and CSS.

Diagram

Browser will construct DOM.All HTML tags will be stored as JavaScript objects.

Eg:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html>
3)    <head>
4)      <title>This is My Titile</title>
5)    </head>
6)    <body>
7)      <h1>This is DOM Demo</h1>
8)      <a href="#">HeperlinkText</a>
9)    </body>
10) </html>
```

The corresponding DOM Structure is:

Diagram

To display Document to the console:

just type on javascript console: document
Then we will get total HTML Page/Document.

To Display DOM objects on the console:

console.dir(document)

Observe that Root element of DOM is document.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

How to grab HTML Elements from the DOM:

Important DOM Attributes:

1. document.URL===>This is original URL of the website
2. document.body===>It returns everything inside body
3. document.head===>It returns head of the page
4. document.links===>It resturns list of all links of the pagae

Important methods of DOM:

DOM defines the following methods to grab HTML elements

1. document.getElementById()

Returns element with the specified id

2. document.getElementsByClassName()

Returns list of all elements belongs to the specified class

3. document.getElementsByTagName()

Returns list of all elements with the specified tag

4. document.querySelector()

Returns the first object matching CSS style selector

5. document.querySelectorAll()

Returns all objects Matches the CSS Style Selector

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>DOM important Methods and Attributes</h1>
9)      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor inci
    didunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitati
    on ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in repre
    henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaec
    at cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p
    >
```

```
10)   <h2>Favourate Food:</h2>
11)   <ul>
12)     <li class="first">Chicken</li>
13)     <li>Mutton</li>
14)     <li>Fish</li>
15)     <li id="special">Any animal including Human being</li>
16)   </ul>
17)
18)   <h2>Favourate Drink:</h2>
19)   <ul>
20)     <li class="first">KingFisher</li>
21)     <li>KnockOut</li>
22)     <li>Milk</li>
23)     <li>Thumsup</li>
24)   </ul>
25)   <a href="http://youtube.com/durgasoftware">View Profile</a>
26)  </body>
27) <script type="text/javascript" src="demo.js"></script>
28) </html>
```

on the Javascript developer's console:

1. document.URL
2. document.body
3. document.header
4. document.links

5. document.getElementById('special')
6. document.getElementsByClass('first')
7. document.getElementsByTagName('h2')
8. document.querySelector('#special')
9. document.querySelectorAll('.first')

querySelector() vs querySelectorAll():

If the specified CSS mapped with only one html element then we should use querySelector() method. If multiple html elements matched, then querySelector() method returns only first matched element.

We can use querySelectorAll() method to grab all matched html elements. If multiple html elements are there then we can use this method.

Q. Write DOM based Javascript code to change color of h1 tag.

```
var myh1=document.querySelector('h1');
myh1.style.color='red';
```

Demo application to grap html elements into javascript:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)    <h1>This is h1 Tag Data</h1>
9)    <p id='first' class="special">This is First Paragraph Data</p>
10)   <p class="special">This is Second Paragraph Data</p>
11)   <p>This is Third Paragraph Data</p>
12)   <p id='last'>This is Fourth Paragraph Data</p>
13)   </body>
14)  </html>
```

The following are various possible ways to grab first paragraph

1. document.getElementById('first')
2. document.getElementsByClassName('special')
3. document.getElementsByClassName('special')[0]
4. document.getElementsByTagName('p')
5. document.getElementsByTagName('p')[0]
6. document.querySelector('#first')
7. document.querySelectorAll('.special')[0]
8. document.querySelector('h1+p')

Color Changer Application:

To generate Random number from 0 to 15 java code is:

```
1)  class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        System.out.println((int)(Math.random()*16));
6)        System.out.println((int)(Math.random()*16));
7)        System.out.println((int)(Math.random()*16));
8)        System.out.println((int)(Math.random()*16));
9)        System.out.println((int)(Math.random()*16));
10)       System.out.println((int)(Math.random()*16));
11)       System.out.println((int)(Math.random()*16));
12)       System.out.println((int)(Math.random()*16));
13)    }
14) }
```

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>DOM important Methods and Attributes</h1>
9)      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor inci
        didunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitati
        on ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in repre
        henderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaec
        at cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p
        >
10)     <h2>Favourate Food:</h2>
11)     <ul>
12)       <li class="first">Chicken</li>
13)       <li>Mutton</li>
14)       <li>Fish</li>
15)       <li id="special">Any animal including Human being</li>
16)     </ul>
17)
18)     <h2>Favourate Drink:</h2>
19)     <ul>
20)       <li class="first">KingFisher</li>
21)       <li>KnockOut</li>
22)       <li>Milk</li>
23)       <li>Thumsup</li>
24)     </ul>
25)
26)     <a href="http://youtube.com/durgasoftware">View Profile</a>
27)   </body>
28)  <script type="text/javascript" src="demo.js"></script>
29) </html>
```

demo.js:

```
1)  var h1=document.querySelector('h1')
2)  var p=document.querySelector('p')
3)  var h2=document.querySelector('h2')
4)  var ul=document.querySelector('ul')
5)
6)  function getRandomColor(){
7)    var letters='0123456789ABCDEF';
8)    var color='#';
9)    for (var i = 0; i < 6; i++) {
10)     var r=Math.floor(Math.random()*16);
```

```
11)    color=color+letters[r]
12)   }
13)   return color
14) }
15) function changeColor(){
16)   var rc=getRandomColor()
17)   h1.style.color=rc;
18)   p.style.color=getRandomColor();
19)   h2.style.color=getRandomColor();
20)   ul.style.color=getRandomColor();
21) }
22) setInterval(changeColor,1000)
```

DOM: Content Interaction:

By using DOM, we can change Text,HTML Code and attributes.

How to Change Text:

```
var h1=document.querySelector('h1')
h1.textContent="This is Modified Content"
```

How to change HTML Code:

If we use textContent property then HTML changes won't be effected.Hence we have to use innerHTML property

```
var p= document.querySelector('p')

p.textContent='<strong><em>'+p.textContent+'</em></strong>'//invalid
p.innerHTML='<strong><em>'+p.textContent+'</em></strong>' //valid
```

How to change Attributes:

We can use the following methods

element.getAttribute('attributeName')
returns the value associated with specified attribute

element.setAttribute('attributeName','attributeValue')
To set new value to the attribute

Eg:

```
var a=document.querySelector('a')
// console.log(a.getAttribute('href'))
a.setAttribute('href','https://facebook.com')
```

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)    <h1>This h1 text can be replaced</h1>
9)    <p>This data can be converted into bold</p>
10)     <a href="http://youtube.com/durgasoftware">View Profile</a>
11)   </body>
12)  <script type="text/javascript" src="demo.js"></script>
13) </html>
```

demo.js:

```
1)  var h1=document.querySelector('h1')
2)  h1.textContent="This is Modified Content"
3)
4)  var p= document.querySelector('p')
5)  p.innerHTML='<strong><em>'+p.textContent+'</em></strong>'
6)
7)  var a=document.querySelector('a')
8)  // console.log(a.getAttribute('href'))
9)  a.setAttribute('href','https://facebook.com')
```

Changing All Links Properties of google.com:

```
1)  var links=document.querySelectorAll('a');
2)  for(link of links){
3)  link.style.background='green';
4)  link.style.color='white';
5)  link.setAttribute('href','http://durgasoftonline.com');
6)  link.textContent='Sunny Leone';
7)  }
```

To replace google logo without customized logo:

```
1)  var logo=document.querySelector('#hplogo');
2)  logo.setAttribute('src','http://www.durgasoftonline.com/wp-
    content/uploads/logo_durgasoft_online-1.png')
```

Event Handling by using DOM:

In our web application there may be a chance of occuring several events like mouse hover,single click and double click etc

Whenever a particular event occurs if we want to perform certain operation automatically then we should go for Listeners.

Listener listens events and will perform certain operation automatically.

How to implement event handling:

element.addEventListener(event,function);

Eg: myh1.addEventListener('click',colorChanger)

To change color of h1 on single click operation:

```
var myh1=document.querySelector('h1')
myh1.addEventListener('click',changeColor)
```

To change color of h1 on double click operation:

```
var myh1=document.querySelector('h1')
myh1.addEventListener('dblclick',changeColor)
```

To change color of h1 on mouse over operation:

```
var myh1=document.querySelector('h1')
myh1.addEventListener('mouseover',changeColor)
```

To change color of h1 on mouse out operation:

```
var myh1=document.querySelector('h1')
myh1.addEventListener('mouseout',changeColor)
```

To change content and color as blue on mouse over operation:

```
1)  var myh1=document.querySelector('h1')
2)  myh1.addEventListener('mouseover',function(){
3)     myh1.textContent='HYDERABAD';
4)     myh1.style.color='blue';
5)     });
6)
7)  myh1.addEventListener('mouseout',function(){
8)        myh1.textContent='BANGALORE';
9)        myh1.style.color='red';
10)    });
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Body Color Changer Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)    <h1>Body Color Changer Application</h1>
9)    <button type="button" name="button">Click Here to change Body Color</button>
10)
11)   </body>
12)  <script type="text/javascript" src="demo.js"></script>
13) </html>
```

demo.dss:

```
1)  function getRandomColor(){
2)    var letters='0123456789ABCDEF';
3)    var color='#';
4)    for (var i = 0; i < 6; i++) {
5)      var r=Math.floor(Math.random()*16);
6)      color=color+letters[r]
7)    }
8)    return color
9)  }
10) var b=document.querySelector('button');
11) b.addEventListener('click',function(){
12)   document.body.style.background=getRandomColor();
13) });
```

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)    <h1>Event Hanling by using DOM</h1>
9)    </body>
10)  <script type="text/javascript" src="demo.js"></script>
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
11) </html>
```

demo.js:

```
1)  function getRandomColor(){
2)    var letters='0123456789ABCDEF';
3)    var color='#';
4)    for (var i = 0; i < 6; i++) {
5)      var r=Math.floor(Math.random()*16);
6)      color=color+letters[r]
7)    }
8)    return color
9)  }
10) function changeColor(){
11)   var rc=getRandomColor()
12)   h1.style.color=rc;
13)   p.style.color=getRandomColor();
14)   h2.style.color=getRandomColor();
15)   ul.style.color=getRandomColor();
16) }
17) var h1=document.querySelector('h1')
18) h1.addEventListener('mouseover',changeColor)
19)  h1.addEventListener('mouseout',changeColor)
```

Demo Application for Random Names and Random Colors:

demo.html

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)    <h1>Event Hanling by using DOM</h1>
9)    </body>
10)  <script type="text/javascript" src="demo.js"></script>
11) </html>
```

demo.js:

```
1)  var myh1=document.querySelector('h1')
2)  function getRandomColor(){
3)    var letters='0123456789ABCDEF';
4)    var color='#';
5)    for (var i = 0; i < 6; i++) {
6)      var r=Math.floor(Math.random()*16);
```

```
7)     color=color+letters[r]
8)    }
9)    return color
10) }
11)
12) function getRandomName(){
13)   var names=['SUNNY LEONE','MALLIKA','VENNA','KATRINA','PRIYANKA','DEEPIKA','KAREE
     NA'];
14)   var r=Math.floor(Math.random()*7);
15)   return names[r];
16) }
17)
18) myh1.addEventListener('mouseover',function(){
19)    myh1.textContent=getRandomName();
20)    myh1.style.color=getRandomColor();
21)    });
22)    myh1.addEventListener('mouseout',function(){
23)     myh1.textContent=getRandomName();
24)        myh1.style.color=getRandomColor();
25)      });
```

Q. Names should be displayed with the specified color only like Katrina always with red color etc

TIC TAC TOE Implementation:

1. Create html,css,js files
2. Add css,bootstrap links to html inside head part
3. Add javascript file to html at bottom of Body tag
4. Add Jumbotron to the html

```
1)   <div class="container">
2)     <div class="jumbotron">
3)       <h1>Welcome to DURGASOFT TIC TAC TOE Game!!!</h1>
4)       <p>Be Ready To play to improve logical thinking...</p>
5)       <button id="b" type="button" class="btn btn-primary btn-
     lg" name="button">Restart Game!!!</button>
6)     </div>
7)   </div>
```

5. Define Jumbotron related styles in css file

```
1)   .container .jumbotron{
2)  background: green;
3)  color:white;
```

6. Add table after jumbotron inside container

```
<table>
    <tr>
      <td></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td></td>
      <td></td>
      <td></td>
    </tr>
</table>
```

7. Define Table related styles in css file

```
td{
  height: 150px;
  width: 150px;
  border: 5px solid blue;
  text-align: center;
  font-size: 100px;
}
```

8. Implementing Restart button functionality inside js file

```
var restartb=document.querySelector('#b')
var cells=document.querySelectorAll('td')
function clearAllCells(){
  for (var i = 0; i < cells.length; i++) {
    cells[i].textContent=''
  }
}
restartb.addEventListener('click',clearAllCells)
```

9. Add content change functionality of the cells inside javascript file

```
function changeContent(){
  if (this.textContent==='') {
    this.textContent='X'
  }
  else if (this.textContent=='X') {
    this.textContent='O'
```

```
7)    }
8)    else {
9)      this.textContent=''
10)   }
11) }
12) for (var i = 0; i < cells.length; i++) {
13)   cells[i].addEventListener('click',changeContent)
14) }
```

ttt.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <link rel="stylesheet" href="ttt.css">
5)      <!-- Latest compiled and minified CSS -->
6)  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/boots
    trap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
    gin="anonymous">
7)      <meta charset="utf-8">
8)      <title>Tic Tac Toe Game</title>
9)    </head>
10)   <body>
11)     <div class="container">
12)       <div class="jumbotron">
13)         <h1>Welcome to DURGASOFT TIC TAC TOE Game!!!</h1>
14)         <p>Be Ready To play to improve logical thinking...</p>
15)         <button id="b" type="button" class="btn btn-primary btn-
    lg" name="button">Restart Game!!!</button>
16)       </div>
17)       <table>
18)         <tr>
19)           <td></td>
20)           <td></td>
21)           <td></td>
22)         </tr>
23)         <tr>
24)           <td></td>
25)           <td></td>
26)           <td></td>
27)         </tr>
28)         <tr>
29)           <td></td>
30)           <td></td>
31)           <td></td>
32)         </tr>
33)       </table>
34)
```

```
35)    </div>
36)    <script src="ttt.js"></script>
37)   </body>
38) </html>
```

ttt.css:

```
1)   .container .jumbotron{
2)     background: green;
3)     color:white;
4)   }
5)   td{
6)     height: 150px;
7)     width: 150px;
8)     border: 5px solid blue;
9)     text-align: center;
10)   font-size: 100px;
11) }
```

ttt.js:

```
1)   var restartb=document.querySelector('#b')
2)   var cells=document.querySelectorAll('td')
3)   function clearAllCells(){
4)     for (var i = 0; i < cells.length; i++) {
5)       cells[i].textContent=''
6)     }
7)   }
8)   restartb.addEventListener('click',clearAllCells)
9)
10) function changeContent(){
11)   if (this.textContent===''){
12)     this.textContent='X'
13)   }
14)   else if (this.textContent=='X') {
15)     this.textContent='O'
16)   }
17)   else {
18)     this.textContent=''
19)   }
20) }
21) for (var i = 0; i < cells.length; i++) {
22)   cells[i].addEventListener('click',changeContent)
23) }
```

Full Stack Web Development with Python and DJango

jQuery

Study Material

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Full Stack Web Development with Python and DJango

Module-6: jQuery

Official website: jquery.com

jQuery is a fast, small, and rich JavaScript library.
We can use jQuery Library to grab and maniulate html elements,to perform event handing and ajax.
jQuery supports multiple browsers.i.e it provides support for cross browser compatibility.

The main advantage of jquery is it provides several methods and objects in the form of javascript file, so that we can use these directly and developer's life got simplified.

Note: In Plain old java script(also known as Vanilla Java Script),we have to write every thing manually. But if we jQuery,we are not required to write much code and we can use its library directly.

Sample code to change color of every h1 tag:

Vanilla JS code:

```
1) var myh1=document.querySelectorAll('h1')
2) for( h1 of myh1){
3) h1.style.color='red';
4) }
```

jQuery Code:

```
$('h1').css('color','red')
```

Advantages of jQuery:

1. It provide several built in methods and objects. We can use these directly so that development will become very easy.

2. Clear and Shorter code

3. Ease of use
4. Cross Browser support
5. AJAX support

Limitations of jQuery:

1. What ever jQuery doing, we can implement everything without jQuery also. ie jQuery won't do any things extra.

2. The total library should be loaded compulsory, which creates performance problems.

Note: Because of above limitations, some part of the developer's community won't recommend jQuery usage.

Eg: youmightnotneedjquery.com

How to connect with jQuery:

We can make jQuery library available to our application in the following 2 ways

1. By Locally
2. By CDN

1. By Locally:

Download jQuery.js file from jquery.com

https://code.jquery.com/jquery-3.3.1.js

Download and place in application folder.
Inside head of html we have to write <script> tag as follows .

<script type="text/javascript" src="jquery.js"></script>

2. By using CDN:

www.code.jquery.com

```
1)  <script
2)    src="http://code.jquery.com/jquery-3.3.1.js"
3)    integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
4)    crossorigin="anonymous"></script>
```

We can get several jquery CDNs from the google

Eg:
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

How to check whether jQuery available or not in our application:

From the console just type
> jQuery or $
We should not get any error

jQuery Selectors:

In Vanilla Javascript we have several methods to select/grab html elements like
getElementById()
getElementsByClassName()
getElementsByTagName()
querySelector()
querySelectorAll()
etc
But in jQuery we have only one way to select html elements.i.e to use $ symbol

In jQuery $ symbol is equivalent to querySelectorAll() in Vanilla Javascript

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <script
5)    src="http://code.jquery.com/jquery-3.3.1.js"
6)    integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
7)    crossorigin="anonymous"></script>
8)      <meta charset="utf-8">
9)      <title></title>
10)   </head>
11)   <body>
12)     <h1>Hello First H1</h1>
13)     <h2>Favourate Drink:</h2>
14)     <ul>
15)       <li>KingFisher</li>
16)       <li>KnockOut</li>
17)       <li>Foster</li>
18)       <li id='special'>HumanBlood</li>
19)     </ul>
20)     <a href="https://google.com">Click Here to go to Google</a>
21)   </body>
22)  </html>
```

jQuery code:

```
$('h1')
$('li')
$('#special')
$('body')
etc...
```

Manipulating HTML Elements:

Once we grab elements by using $ symbol,we can manipulate by using css() method.

Syntax:

```
$(selector).css(property,value)
```

Examples:

```
$('h1').css('color','white');
$('h1').css('background','red');
$('h1').css('border','5px solid green');
```

We can save selected html element by using variable

```
var x = $('h1')
x.css('color','white');
x.css('background','red');
x.css('border','5px solid green');
```

Instead of passing parameters one by one,we can create Object and pass that object directly.

```
1)  var x = $('h1')
2)  var myCSS={
3)     color:'white',
4)     background:'green',
5)     border:'red 5px solid'
6)     }
7)  x.css(myCSS);
```

Note: **We can use $ symbol to select and css() method to manipulate html elements.**

How to select a particular html element instead of all matched elements:

By default $('element') selects all matched html elements.
But by using the following ways we can get particular matched element.

$('element:first')
$('element:last')
$('element:first-of-type')
$('element:nth-of-type(n)')
etc...

Eg:
1. $('h1')==>Selects all h1 tags
2. $('h1:first')==>Selects only first h1 tag
3. $('h1').first()==>selects only first h1 tag
4. $('h1:first-of-type')==>Selects only first h1 tag
5. $('h1:nth-of-type(2)')==>Selects second h1 tag
6. $('h1:last')==>Selects only last h1 tag
7. $('h1').last()==>selects only last h1 tag

Q1. Write Vanilla Javascript and jQuery codes to change all h1 tags text color as white and background as red.

Vanilla Javascript code:

```
1)  var allh1s=document.querySelectorAll('h1');
2)  for(h1 of allh1s){
3)     h1.style.color='white';
4)     h1.style.background='red';
5)  }
```

jQuery Code:

```
1)  var mystyles={
2)              color:'white',
3)        background:'red'
4)     };
5)  $('h1').css(mystyles)
6)
7)  Instead of this we can write directly as follows
8)
9)  $('h1').css({
10)             color:'white',
11)       background:'red'
12)       });
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Q2. Write Vanilla Javascript and jQuery codes to set all li tags font size as 20px.

Vanilla Javascript code:

```
1)  var x=document.querySelectorAll('li');
2)  for(li of x){
3)     li.style.fontSize='20px';
4)  }
```

jQuery Code:

```
$('li').css('fontSize','20px')
```

Note: The biggest advantage of jQuery is we can do more things with less code(Write less,do More)

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <script
5)    src="http://code.jquery.com/jquery-2.2.4.js"
6)    integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
7)    crossorigin="anonymous"></script>
8)
9)      <meta charset="utf-8">
10)     <title></title>
11)   </head>
12)   <body>
13)     <p>This is First Paragraph</p>
14)     <p id="second">This is Second Paragraph</p>
15)     <p class="remaining">This is Third Paragraph</p>
16)     <p class="remaining">This is Fourth Paragraph</p>
17)   </body>
18) </html>
```

Case-1: Select all <p> tags and set background as green

```
$('p').css('background','green')
```

Case-2: Select all <p> tags with class 'remaining' and make them 200px wide(width)

```
$('.remaining').css('width','200px')
```

Case-3: Select all <p> tags with id 'second' and give red solid 10px border.
$('#second').css('border','10px solid red')

case-4: Select only first <p> tag and change text color as white
$('p:first').css('color','white')

case-4: Select only third <p> tag and change font-size as 30px
$('p:nth-of-type(3)').css('fontSize','30px')

The Most Commonly used jQuery Methods:

The following are the most commonly used jQuery methods

1. text()
2. html()
3. attr()
4. val()
5. addClass()
6. removeClass()
7. toggleClass()

1. text():

We can use this method to get and set text of the matched elements. i.e this method acts as both getter and setter method.

text()===>To get text of all matched elements including child tags.In this case this method acts as getter method

text(content)====>To set provided text content for every matched element. In this case this method acts as setter method.

Note: If any method developed to acts as both getter and setter method then it is said to be this method follows getter and setter paradigm.

Eg:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <script
5)    src="http://code.jquery.com/jquery-2.2.4.js"
6)    integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
7)    crossorigin="anonymous"></script>
8)      <meta charset="utf-8">
9)      <title></title>
10)   </head>
11)   <body>
```

```
12)   <h1>This is H1 Data</h1>
13)   <h2>Choose Your Favourate Subject:</h2>
14)   <ul>
15)     <li>HTML</li>
16)     <li>JavaScript</li>
17)     <li>CSS</li>
18)     <li>jQuery</li>
19)    </ul>
20)  </body>
21) </html>
```

1. To get text content of h1:
$('h1').text()

2. To get text of all li tags
$('li').text() or
$('ul').text()

3. To get text of only first li
$('li').first().text()

4. To set h1 text content as : DURGA SOFTWARE SOLUTIONS
$('h1').text('DURGA SOFTWARE SOLUTIONS')

5. To set every li tag text content as : KINGFISHER
$('li').text('KINGFISHER')

2. html():

We can use this method to get and set html content.
If we are not passing any argument then this method acts as getter method,which returns HTML contents of the first matched element.

Eg: $('h1').html()

If we are passing argument then this method acts as setter method,which sets the HTML contents of every matched element.

Eg: $('li').html('AMAZON')

3. attr():

We can use this method to get and set attribute values.
attr(attributename)--->To get the value of specified attribute of the first matched element.

attr(attributename,attributevalue)
 -->If the specified attribute already there then old value replaced with new value. If the specified attribute not already available then a new attribute will be added for every matched tag.

Demo for getting and setting src attribute of img tag:

```
<img src="https://images.unsplash.com/photo-1462953491269-9aff00919695?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=e2dea8d19793b1474b4c4962a98f0b47&auto=format&fit=crop&w=400&q=60" alt="">
 <img src="https://images.unsplash.com/photo-1460999158988-6f0380f81f4d?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=5a982cdff8b38d0758caa758e71f50e0&auto=format&fit=crop&w=400&q=60" alt="">
 <img src="https://images.unsplash.com/photo-1504350440606-81847d413a13?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=bdf251e852bd65abcc522fc4e903da02&auto=format&fit=crop&w=400&q=60" alt="">
```

1. To set width and height of every image properly

```
$('img').css({width:'150px',height:'150px'});
```

2. To get src attribute value of the first image:

```
$('img').attr('src')
```

3. To set src attribute of all images with our new image

```
$('img').attr('src',"https://images.unsplash.com/photo-1484406566174-9da000fda645?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=a439fe04a06e1457297643e219add900&auto=format&fit=crop&w=400&q=60");
```

4. To change only first image src attribute value

```
$('img').first().attr('src',"https://images.unsplash.com/photo-1504350440606-81847d413a13?ixlib=rb-0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=bdf251e852bd65abcc522fc4e903da02&auto=format&fit=crop&w=400&q=60")
```

Demo for getting and setting type attribute of input tag:

User Input: <input type="text" name="" value="">

1. To get the value of type attribute of input tag

```
$('input').attr('type')
```

2. To set the value of type attribute with 'color'

```
$('input').attr('type','color')
```

3. To set the value of type attribute with 'checkbox'

```
$('input').attr('type','checkbox')
```

4.val() method:

We can use val() method to get value of the first matched element.

Eg: value entered in the text box
which radio button selected
which value selected from dropdown box

We can also use val() method to set value for every matched element.

val()===>getter method
val(text)===>setter method

Eg 1: with input tag

Enter Name:<input type="text" name="user1" value="" placeholder="Your Name">
Enter Name:<input type="text" name="user2" value="" placeholder="Your Name">

$('input').val()==>It returns the value entered by the end user in the first text box.

$('input').val('durga')==>To set value durga for every text box.

Usecase: While implementing reset button functionality to clear all input fields this method is helpful.

Eg 1: with dropdown menu

Choose Your Required Course:

```
1)      <select class="" name="">
2)       <option value="CorePython">Core Python</option>
3)       <option value="AdvPython">Adv Python</option>
4)       <option value="DJango">DJango</option>
5)       <option value="Flask">Flask</option>
6)      </select>
```

$('select').val()===>Returns Selected Value
$('select').val('DJango') ===>To set value as DJango

addClass(),removeClass() &toggleClass():

addClass(): We an use this method to add specified class/classes to the set of matched elements.

removeClass(): Remove a single class/classes from each matched element.

toggleClass(): We can use this method to add and remove classes from the matched elements.

If the specified class already set then it will be removed. It is not already set then the class will be added for every matched element.

Eg:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)     <head>
4)       <script
```

```
5)    src="http://code.jquery.com/jquery-2.2.4.js"
6)    integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
7)    crossorigin="anonymous"></script>
8)     <meta charset="utf-8">
9)     <title></title>
10)    <style >
11)    .high{
12)      color:white;
13)      background:red;
14)    }
15)    .low{
16)      color:white;
17)      background: blue;
18)    }
19)    .completed{
20)      text-decoration: line-through;
21)    }
22)    </style>
23)  </head>
24)  <body>
25)    <h1>List of Activities: </h1>
26)    <ul>
27)      <li>Meeting Sunny</li>
28)      <li>Reading DJango</li>
29)      <li>Taking Classes</li>
30)      <li>Visiting US</li>
31)      <li>Watching Movie</li>
32)      <li>Playing Cricket</li>
33)      <li>Swimming </li>
34)    </ul>
35)  </body>
36) </html>
```

1. $('li').addClass('high')

It will add 'high' class to every li tag

2. $('li:nth-of-type(even)').removeClass('high')

It will remove 'high' class from every even numbered li tag.

3. $('li:nth-of-type(2)').addClass('completed')

It will add 'completed' class only for 2nd li

4. $('li').toggleClass('low')

It will add 'low' class for every li tag(because this class is not already set)

5. $('li').toggleClass('low')

It will remove 'low' class from every li tag(because this class is already set)

Event Handling by using jQuery:

We can implement event handling to make our html elements interactive.

jQuery defines several methods for event handling. We can get complete jquery event handling related methods from below link:

https://api.jquery.com/category/events/

The top 3 most commonly used jQuery event methods:

1. click()
2. keypress()
3. on()

1. click():

jQuery click() method can be used to add a click listener to the elements.

Eg 1: To raise alert message when ever we are clicking h1 tag.

```
1)  $('h1').click(function(){
2)    alert('h1 tag got clicked');
3)  });
```

Eg 2: To raise alert message and to change background color of last button on click event

```
1)  $('button:last').click(function(){
2)    alert('Hello Stupid Dont Sleep I will Kill You!!!');
3)    $(this).css('background','red')
4)  });
```

Note:
In Vanilla Java Script, 'this' always represent current element. But in jQuery we have to use $(this)

Demo Application:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <script
5)    src="http://code.jquery.com/jquery-2.2.4.js"
6)    integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
7)    crossorigin="anonymous"></script>
8)     <meta charset="utf-8">
9)     <title></title>
10)  </head>
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
<body>
  <h1>jQuery Event Handler Demo </h1>
  <button type="button" name="button">Dont Sleep First Warning</button>
  <button type="button" name="button">Dont Sleep Second Warning</button>
  <button type="button" name="button">Dont Sleep Third Warning</button>
  <script type="text/javascript" src="demo.js">
  </script>
 </body>
</html>
```

demo.js:

```
$('h1').click(function(){
  alert('h1 tag clicked')
})
$('button:first').click(function(){
  alert('Hello Dont Sleep');
  $(this).css('background','yellow')
});
$('button:nth-of-type(2)').click(function(){
  alert('Dont Sleep I will beat you');
  $(this).css('background','orange')
});
$('button:last').click(function(){
  alert('Hello Stupid Dont Sleep I will Kill You!!!');
  $(this).css('background','red')
});
```

2. keypress():

We can use this method to add keypress listener to elements.
i.e whenever we are pressing the key if we want to do any activity automatically then we should go for keypress() method.

Eg: To raise alert message for every character typed in text box

Enter Name: `<input type='text'>`

jQuery code:

```
$('input').keypress(function(){
  alert('Inserted one character!!!!');
})
```

To know the pressed character:

In the case of keypress event,the total information is avaialable in the event object. For every character the corresponding key-code is available.
char-code, unicode,ascii code, which all represents same key-code

The complete key-code information is avilable in the following link
https://www.cambiaresearch.com/articles/15/javascript-char-codes-key-codes

If we use event.which, then we will get the corresponding key-code

To raise alert message whenever we are pressing x or X:

```
1)  $('input').keypress(function(event){
2)    if (event.which==88 || event.which==120)
3)    {
4)      alert('Hello You are pressing x or X. You are under monitoring !!!!')
5)    }
6)  })
```

Note: **The key-code for 'x' is :120 where as for 'X' is :88**

To raise alert message with typed content whenever we are pressing enter key:

```
1)  $('input').keypress(function(event){
2)    if (event.which==13)
3)    {
4)      alert('Hello just pressed enter key and your current content is:'+$(this).val())
5)    }
6)  })
```

Difference between keypress and keydown, keyup events:

keydown and keyup provides a code indicating which key is pressed, where as keypress indicates that which character was entered.

Eg: **For shift+a**

1. keydown for 'shift'
2. keydown for 'a'
3. keyup for 'a'
4. keyup for 'shift'
5. key press for 'A'

Event handling by using on() method:

on() is the most commonly used method to perform event handling in jQuery.

It is similar to Vanilla Java Script addEventListener() method.

click()-->applicable only for click event
keypress()-->applicable only for keypress event
on()-->applicable for all events including click and keypress.

Eg 1: Whenever mouseover event happend, the text of h1 tag should be changed to 'Bangalore City' with red background and white text.

Whenever mouseout event happend, the text of h1 tag should be changed to 'Hyderabad City' with green background and white text.

```
1)  $('h1').on('mouseover',function(){
2)    $(this).text('BANGALORE CITY')
3)    $(this).css({background:'red',color:'white'});
4)  })
5)  $('h1').on('mouseout',function(){
6)    $(this).text('HYDERABAD CITY')
7)    $(this).css({background:'green',color:'white'});
8)  })
```

Eg 2: For button
Single Click: alert message as Hello Stupid dont click
Double Click: alert message as Hello Animal I will kill you

html:

```
1)  <button type="button" name="button">Click Here to get Greeting Message</button><br>
    <br>
2)  <button type="button" name="button">Double Click Here to get Greeting Message</butt
    on>
```

jQuery:

```
1)    $('button:first').on('click',function(){
2)    alert('Hello Stupid Dont Click!!!')
3)  })
4)
5)  $('button:last').on('dblclick',function(){
6)    alert('Hello Animal I Will Kill You!!!')
7)  })
```

jQuery Effects:

jQuery provides several in-built effects.But main important effects are:

1. Fading Effects
2. Sliding Effects

https://api.jquery.com/category/effects/

1. Fading Effects:

jQuery defines the following methods for fading purposes

1. fadeOut():

Hide the matched elements by fading them to transparent.

2. fadeIn():

Display the matched elements which are fadeout

3. fadeToggle():

Display or hide the matched elements
if already fadeOut then fadeIn will be performed.
If already fadeIn then fadeOut will be performed.

Various sample codes:

```
1)  $('div').fadeOut();
2)  $('div').fadeOut(2000);
3)  $('div').fadeOut(2000,function(){
4)    console.log('Fadeout of the element completed');
5)    });
```

2000 is the time in milliseconds
Similarly we can use these combinations for fadeIn() and fadeToggle() also.

Demo Application:

demo.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)
4)    <head>
5)      <style>
6)        div{
7)          height: 100px;
8)          width: 100px;
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

```
9)          background: red;
10)         color:white;
11)         margin:20px;
12)         text-align: center;
13)         float: left;
14)        }
15)      </style>
16)      <script
17)   src="http://code.jquery.com/jquery-2.2.4.js"
18)   integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
19)   crossorigin="anonymous"></script>
20)    <meta charset="utf-8">
21)    <title></title>
22)   </head>
23)   <body>
24)    <button type="button" name="button">Click Here</button>
25)    <div>First Div</div>
26)    <div>Second Div</div>
27)    <div>Third Div</div>
28)    <div>Fourth Div</div>
29)   </body>
30)   <script type="text/javascript" src="demo.js">
31)
32)   </script>
33) </html>
```

demo.js:

```
1)  $('button').on('click',function(){
2)    $('div').fadeToggle(2000);
3)  })
```

Note: Whenever we perform fadeout just elements will be hidden but won't be removed. But based on requirement we can remove the matched elements also.

```
1)  $('button').on('click',function(){
2)    $('div').fadeToggle(2000,function(){
3)    $(this).remove()}
4)  );
5)  })
```

Sliding Effects:

jQuery defines the following methods for sliding effects purposes.

1. slideUp():
Hide the matched elements with a sliding motion.

2. slideDown():
Display the matched elements with a sliding motion.

3. slideToggle()
Display or hide the matched elements with a sliding motion.

Eg:

```
$('button').on('click',function(){
  $('div').slideToggle(2000);
})
```

Note: passing function as argument, remove matched elements are exactly same as fading effects.

Assignments:

1. Connect4 Game
2. Todo List Application
3. Implement atleast 2 case studies from w3schools howto
https://www.w3schools.com/howto/

How to configure Atom for Python:

1) Terminal Installation:

File->Settings->Install->in the searchbox just type terminal-->platform-ide-terminal

2) Python AutoCompletion:

File->Settings->Install->in the searchbox just type python-->autocomplete-python

3) django:

File->Settings->Install->in the searchbox just type djanngo-->atom-django

4) How to change terminal from powershell to normal command prompt:

File->Settings->Install->in the searchbox just type terminal-->platform-ide-terminal-->settings-->Shell Override

C:\Windows\System32\cmd.exe

DJango:

- **Django is a free and open-source web framework.**
- **It is written in Python.**
- **It follows the Model-View-Template (MVT) architectural pattern.**
- **It is maintained by the Django Software Foundation (DSF)**

- **It is used by several top websites like Youtube,Google,Dropbox,Yahoo Maps, Mozilla,Instagram,Washington Times,Nasa and many more**

- **https://www.shuup.com/blog/25-of-the-most-popular-python-and-django-websites/**

- **Django was created in 2003 as an internal project at Lowrence Journal-World News Paper for their web development.**

- **The Original authors of Django Framework are: Adrian Holovaty, Simon Willison**

- **After Testing this framework with heavy traffics, Developers released for the public as open source framework on July 21st 2005.**

- **The Django was named in the memory of Guitarist Django Reinhardt.**

- **Official website: djangoproject.com**

Top 5 Features of Django Framework:

Django was invented to meet fast-moving newsroom deadlines, while satisfying the tough requirements of experienced Web developers.

The following are main important features of Django

1) Fast:

Django was designed to help developers take applications from concept to completion as quickly as possible.

2) Fully loaded:

Django includes dozens of extras we can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks.

3) Security:

Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

4) Scalability:

Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

5) Versatile:

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

Note:

1) **As Django is specially designed web application framework, the most commonly required activities will takes care automatically by Django and Hence Developer's life will be simplified and we can develop applications very easily.**

2) **As Django invented at news paper,clear documentation is avilable including a sample polling application.**

3) **https://docs.djangoproject.com/en/2.1/contents/**

How to install django:

1. Make sure Python is already installed in our system

python --version

2. Install django by using pip

pip install django
pip install django == 1.11.9

```
D:\>pip install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/51/1a/
6153103322/Django-2.1-py3-none-any.whl (7.3MB) 100% || 7.3MB 47kB/s

Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/30/4e/
53b898779a/pytz-2018.5-py2.py3-none-any.whl (510kB)
    100% || 512kB 596kB/s

Installing collected packages: pytz, django
Successfully installed django-2.1 pytz-2018.5
You are using pip version 9.0.3, however version 18.0 is ava
You should consider upgrading via the 'python -m pip install
```

3. To check django version:

py -m django --version

Django Project vs Django Application:

A Django project is a collection of applications and configurations which forms a full web application.
Eg: Bank Project

A Dango Application is responsible to perform a particular task in our entire web application.
Eg: loan app
registration app
polling app etc

Diagram

Project = Several Applications + Configuration Information

Note:

1) **The Django applications can be plugged into other projects.ie these are reusable. (Pluggable Django Applications)**
2) **Without existing Django project there is no chance of existing Django Application. Before creating any application first we required to create project.**

How to create Django Project:

Once we installed django in our system, we will get 'django-admin' command line tool, which can be used to create our Django project.

```
django-admin startproject  firstProject

D:\>mkdir djangoprojects

D:\>cd djangoprojects

D:\djangoprojects>django-admin start-project firstProject
```

The following project structure will be created

```
D:\djangoprojects>
 |
 +---firstProject
    ¦
    ¦---manage.py
    ¦
    +---firstProject
        ¦---settings.py
        ¦---urls.py
        ¦--wsgi.py
        ¦-- __init__.py
```

__init__.py:

It is a blank python script.Because of this special file name, Django treated this folder as python package.

Note: If any folder contains __init__.py file then only that folder is treated as Python package.But this rule is applicable until Python 3.3 Version.

settings.py:

In this file we have to specify all our project settings and and configurations like

installed applications, middileware configurations, database configurations etc

urls.py:

Here we have to store all our url-patterns of our project.
For every view (web page), we have to define separate url-pattern. End user can use url-patterns to access our webpages.

wsgi.py:

wsgi → Web Server Gateway Interface.
We can use this file while deploying our application in production on online server.

manage.py:

The most commonly used python script is manage.py
It is a command line utility to interact with Django project in various ways like to run development server, run tests, create migrations etc.

How to run Django Development server:

We have to move to the manage.py file location and we have to execute the following command.

```
py manage.py runserver

D:\djangoprojects\firstProject>py manage.py startserver

Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you apply the
migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 03, 2018 - 15:38:59
Django version 1.11, using settings 'firstProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Now the server started.

How to send first request:

Open browser and send request:

http://127.0.0.1:8000/

The following should be response if everything goes fine.

```
-------------------------------------------------------------------------------
It worked!
```

Congratulations on your first Django-powered page.
Next, start your first app by running python manage.py startapp [app_label].

You're seeing this message because you have DEBUG = True in your Django settings file and you haven't configured any URLs. Get to work!

Role of Web Server:

Web Server provides environment to run our web applications.
Web Server is responsible to receive the request and forward request to the corresponding web component based on url-pattern and to provide response to the end user.
Django framework is responsible to provide development server. Even Django framework provides one inbuilt database sqlite. Special Thanks to Django.

Note: Once we started Server a special database related file will be generated in our project folder structure.

db.sqlite3

Creation of First web application:

Once we creates Django project, we can create any number of applications in that project.

The following is the command to create application.

python manage.py startapp firstApp

D:\djangoprojects\firstProject>python manage.py startapp firstApp

The following is the folder structure got created.

```
D:\djangoprojects>
└──firstProject
   │  db.sqlite3
   │  manage.py
   │
   ├──firstApp
   │  │  admin.py
   │  │  apps.py
   │  │  models.py
   │  │  tests.py
   │  │  views.py
   │  │  __init__.py
   │  │
   │  └──migrations
   │        __init__.py
```

```
|
└──firstProject
   | settings.py
   | urls.py
   | wsgi.py
   | __init__.py
   |
```

Note: Observe that Application contains 6 files and project contains 4 files+ one special file: manage.py

1) __init__.py:
It is a blank Python script. Because of this special name,Python treated this folder as a package.

2) admin.py:
We can register our models in this file. Django will use these models with Django's admin interface.

3) apps.py:
In this file we have to specify application's specific configurations.

4) models.py:
In this file we have to store application's data models.

5) tests.py:
In this file we have to specify test functions to test our code.

6) views.py:
In this file we have to save functions that handles requests and return required responses.

7) migrations folder:
This directory stores database specific information related to models.

Note: The most important commonly used files in every project are views.py and models.py

Activities required for Application:

Activity-1: Add our application in settings.py,so that Django aware about our application.

In settings.py:

```
1) INSTALLED_APPS = [
2)     'django.contrib.admin',
3)     'django.contrib.auth',
4)     'django.contrib.contenttypes',
5)     'django.contrib.sessions',
```

```
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'firstApp'
]
```

Activity-2: Create a view for our application in views.py.
View is responsible to prepare required response to the end user. i.e view contains business logic.
There are 2 types of views.

1. Function Based Views
2. Class Based Views

In this application we are using Function based views.

views.py:

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def display(request):
    s='<h1>Hello Students welcome to DURGASOFT Django classes!!!</h1>'
    return HttpResponse(s)
```

Note:
1. Each view will be specified as one function in views.py.
In the above example display is the name of function which is nothing but one view.
2. Each view should take atleast one argument (request)
3. Each view should return HttpResponse object with our required response.

Diagram

View can accept request as input and perform required operations and provide proper response to the end user.

Activity-3: Define url-pattern for our view in urls.py file.
This url-pattern will be used by end-user to send request for our views.
The 'urlpatterns' list routes URLs to views.

For functional views we have to do the following 2 activities:
1. Add an import: from firstApp import views
2. Add a URL to urlpatterns:
   url(r'^greeting/', views.display)

urls.py:

```
1)  from django.conf.urls import url
2)  from django.contrib import admin
3)  from firstApp import views
4)
5)  urlpatterns = [
6)      url(r'^admin/', admin.site.urls),
7)      url(r'^greetings/', views.display),
8)  ]
```

Whenever end user sending the request with urlpattern: greeting then disply() function will be executed and provide required response.

Activity-4: Start Server and Send the request

py manage.py runserver

http://127.0.0.1:8000/greetings

Http Request flow in Django Application:

Diagram

1. Whenever end user sending the request first Django development server will get that request.

2. From the Request django will identify urlpattern and by using urls.py, the corresponding view will be identified.

3. The request will be forwared to the view. The corresponding function will be executed and provide required response to the end user.

Summary of sequence of activities related to Django Project:

1) Creation of Django project
django-admin startproject firstProject

2) Creation of Application in that project
py manage.py startapp firstApp

3) Add application to the Project
(inside settings.py)

4) Define view function inside views.py
5) Define url-pattern for our view inside urls.py

6) Start Server
py manage.py runserver

7) Send the request

How to change Django Server Port:

By default Django develoment server will run on port number: 8000. But we can change port number based on our requirement as follows.

py manage.py runserver 7777

Now Server running on port number: 7777
We have to send the request with this port number only

http://127.0.0.1:7777/greetings/
http://127.0.0.1:8000/time/

Defining urlpatterns at application level instead of project level:

A Django project can contain multiple applications and each application can contain multiple views.Defining url-patterns for all views of all applications inside urls.py file of project creates maintenance problems and reduces reusability of applications.

We can solve this problem by defining url-patterns at application level instead of project level. For every application we have to create a separate urls.py file and we have to define all that application specific urls in that file. We have to link this application level urls.py file to project level urls.py file by using include() method.

Demo Application:

1. Creation of Project
django-admin startproject urlProject

2. Creation of Application
py manage.py startapp urlApp

3. Add our application to the Project inside settings.py file

```
INSTALLED_APPS=[
      .........
          'urlApp'
          ]
```

4. Define View Function in views.py

```
1)  from django.http import HttpResponse
2)  def appurlinfo(request):
3)
4)     s='<h1>Application Level urls Demo</h1>'
5)     return HttpResponse(s)
```

5. Create a seperate urls.py file inside application

```
1)  from django.conf.url import url
2)   from urlApp import views
3)   urlpatterns=[
4)      url(r'^test/',views.appurlinfo)
5)      ]
```

6. Include this application level urls.py inside project level urls.py file.
from django.conf.urls. import include

```
urlpatterns=[
  ....
  url(r'^urlApp/',include('urlApp.urls'),
  ]
```

6. Run Server
py manage.py runserver

7. Send Request
http://127.0.0.1:8000/urlApp/test

Advantages:

The main advantages of defining urlpatterns at application level instead of project level are

1. It promotes reusability of Django Applications across multiple projects
2. Project level urls.py file will be clean and more readable

Django Templates:

It is not recommended to write html code inside python script (views.py file) because:

1. It reduces readability because Python code mixed with html code

2. No seperation of roles. Python developer has to concentrate on both python code and html code.

3. It does not promote reusability of code

We can overcome these problems by seperating html code into a seperate html file.This html file is nothing but template.
From the Python file (views.py file) we can use these templates based on our requirement.

We have to write templates at project level only once and we can use these in multiple applications.

Python stuff required to develop Template Based Application:

1. To know the current Python file name

```
print(__file__)    #test.py
```

2. To know absolute path of current Python File Name

```
import os
print(os.path.abspath(__file__))
```

Output: D:\durgaclasses\test.py

3. To know Base Directory name of the current file

```
print(os.path.dirname(os.path.abspath(__file__)))
```

Output: D:\durgaclasses

4. Inside D:\durgaclasses there is one folder named with templates. To know its absolute path

```
import os
BASE_DIR=os.path.dirname(os.path.abspath(__file__))
TEMPLATE_DIR=os.path.join(BASE_DIR,'templates')
print(TEMPLATE_DIR)
```

Output: D:\durgaclasses\templates

Note: **The main advantage of this approach is we are not required to hard code system specific paths(locations) in our python script.**

Steps to develop Template Based Application:

1. Creation of Project
django-admin startproject templateProject

2. Creation of Application
py manage.py startapp testApp

3. Add this application to the project in settings.py file,so that Django aware of application

4. Create a 'templates' folder inside main project folder.
In that templates folder create a seperate folder named with testApp to hold that particular application specific templates.

5. Add templates folder to settings.py file so that Django can aware of our templates.

```
TEMPLATES = [
  {
     ...,
     'DIRS': ['D:\djangoprojects\templateProject\templates'],

   },]
```

It is not recommended to hard code system specific locations in settings.py file. To overcome this problem, we can generate templates directory path programatically as follows.

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
TEMPLATE_DIR=os.path.join(BASE_DIR,'templates')
```

Specify this TEMPLATE_DIR inside settings.py as follows

```
TEMPLATES = [
  {
     ...,
     'DIRS': [TEMPLATE_DIR,],

   },]
```

6. Create html file inside templateProject/templates/testApp folder. This html file is nothing but template.

wish.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>First Template Page</title>
6)    </head>
```

```
7)  <body>
8)    <h1>Hello welcome to Second Hero of MVT: Templates</h1>
9)  </body>
10) </html>
```

7. Define Function based view inside views.py file

```
1)  from django.shortcuts import render
2)  def wish(request):
3)    return render(request,'testApp/wish.html')
```

8. Define URL Pattern either at application level or at project level

9. Run server and send request

Template Tags:

From Python views.py we can inject dynamic content to the template file by using template tags.

Template Tags also known as Template Variables.

Take special care about Template tag syntax it is not python syntax and not html syntax. Just it is special syntax.

Template tag syntax for inserting text data:
{ {insert_date} }

This template tag we have to place inside template file (ie html file) and we have to provide insert_date value from python views.py file.

Diagram

Demo Application to send date and time from views.py to template file:

wish.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title>First Template Page</title>
```

```
6)    <style >
7)    h1{
8)      color:white;
9)      background: red;
10)   }
11)   </style>
12)  </head>
13)  <body>
14)   <h1>Hello Server Current Date and Time : <br>
15)    {{insert_date}}
16)   </h1>
17)  </body>
18) </html>
```

views.py:

```
1)  from django.shortcuts import render
2)  import datetime
3)  def wish(request):
4)     date=datetime.datetime.now()
5)     my_dict={'insert_date':date}
6)     return render(request,'testApp/wish.html',context=my_dict)
```

Note: The values to the template variables should be passed from the view in the form of dictionary as argument to context.

Application to wish end user based on time:

wish.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)     <meta charset="utf-8">
5)     <title>First Template Page</title>
6)     <style >
7)     #h11{
8)       color:red;
9)     }
10)    #h12{
11)      color:green;
12)    }
13)    </style>
14)  </head>
15)  <body>
16)    <h1 id=h11>{{insert_msg}}</h1>
17)    <h1 id=h12>Current Date and Time : {{insert_date}}</h1>
18)  </body>
```

```
19) </html>
```

views.py:

```
1)  from django.shortcuts import render
2)  import datetime
3)
4)  # Create your views here.
5)  def wish(request):
6)      date=datetime.datetime.now()
7)      msg='Hello Guest !!!! Very Very Good '
8)      h=int(date.strftime('%H'))
9)      if h<12:
10)         msg +='Morning!!!'
11)     elif h<16:
12)         msg +='AfterNoon!!!'
13)     elif h<21:
14)         msg +='Evening!!!'
15)     else:
16)         msg='Hello Guest !!!! Very Very Good Night!!!'
17)     my_dict={'insert_date':date,'insert_msg':msg}
18)     return render(request,'testApp/wish.html',context=my_dict)
```

Working with Static Files:

Up to this just we injected normal text data into template by using template tags.

But sometimes our requirement is to insert static files like images,css files etc inside template file.

Process to include static files inside template:

1. Create a folder named with 'static' inside main project folder. It is exactly same as creating 'templates' folder.
In that 'static' folder create 'images' folder to place image files.

2. Add static directory path to settings.py file, so that Django can aware of our images.

settings.py:

```
1)  STATIC_DIR=os.path.join(BASE_DIR,'static')
2)
3)  ..
4)  STATIC_URL = '/static/'
5)
6)  STATICFILES_DIRS=[
7)  STATIC_DIR,
8)  ]
```

3. Make sure all paths are correct or not

http://127.0.0.1:8000/static/images/divine3.jpg

4. Use Template Tags to insert image
At the beginning of html just after <!DOCTYPE html> we have to include the following template tag

{% load staticfiles %}

Just we are conveying to the Django to load all static files.

We have to include image file as follows
<img src="{% static "images/divine3.jpg"% }">

wish.html:

```
1)  <!DOCTYPE html>
2)  {% load staticfiles %}
3)  <html lang="en" dir="ltr">
4)    <head>
5)      <meta charset="utf-8">
6)      <title>First Template Page</title>
7)      <style >
8)      #h11{
9)        color:red;
10)     }
11)     #h12{
12)       color:green;
13)     }
14)     </style>
15)   </head>
16)   <body>
17)     <h1 id=h11>{{insert_msg}}</h1>
18)     <h1 id=h12>Current Date and Time : {{insert_date}}</h1>
19)     <h1>This climate preferable image is:</h1>
20)     <img src="{% static "images/divine3.jpg" %}" alt="">
21)   </body>
22) </html>
```

views.py:

```
1)  from django.shortcuts import render
2)  import datetime
3)
4)  # Create your views here.
5)  def wish(request):
6)      date=datetime.datetime.now()
7)      msg=None
```


```
8)    h=int(date.strftime('%H'))
9)    if h<12:
10)      msg='Hello Guest !!!! Very Very Good Morning!!!'
11)   elif h<16:
12)      msg='Hello Guest !!!! Very Very Good AfterNoon!!!'
13)   elif h<21:
14)      msg='Hello Guest !!!! Very Very Good Evening!!!'
15)   else:
16)      msg='Hello Guest !!!! Very Very Good Night!!!'
17)   my_dict={'insert_date':date,'insert_msg':msg}
18)   return render(request,'testApp/wish.html',context=my_dict)
```

How to include css file:

1. Create a folder 'css' inside static and place our demo.css file in that css folder.

demo.css:

```
1)  img{
2)    height: 500px;
3)    width: 500px;
4)    border: 10px red groove;
5)    margin:0% 20%;
6)  }
7)  h1{
8)    color:blue;
9)    text-align: center;
10) }
```

2. In the template html file we have to include this css file. We have to do this by using link tag inside head tag.

<link rel="stylesheet" href="{% static "css/demo.css" %}" >

DURGASOFT NEWS PORTAL:

index.html:

```
1)  <!DOCTYPE html>
2)  {% load staticfiles%}
3)  <html lang="en" dir="ltr">
4)   <head>
5)    <meta charset="utf-8">
6)    <title></title>
7)    <link rel="stylesheet" href="{% static "css/demo.css"%}">
8)   </head>
9)   <body>
10)   <h1>Welcome to DURGASOFT NEWS PORTAL</h1>
11)   <ul>
12)    <li> <a href="/movies">Movies Information</a> </li>
13)    <li> <a href="/sports">Sports Information</a> </li>
14)    <li> <a href="/politics">Politics Information</a> </li>
15)   </ul>
16)  </body>
17) </html>
```

news.html:

```
1)  <!DOCTYPE html>
2)  {% load staticfiles %}
3)  <html lang="en" dir="ltr">
4)   <head>
5)    <meta charset="utf-8">
6)    <title></title>
7)    <link rel="stylesheet" href="{% static "css/demo.css"%}">
8)   </head>
9)   <body>
10)    <h1>{{head_msg}}</h1>
11)    <ul>
12)     <li> <h2>{{sub_msg1}}</h2> </li>
13)     <li> <h2>{{sub_msg2}}</h2> </li>
14)     <li> <h2>{{sub_msg3}}</h2> </li>
15)    </ul>
16)    <img src="{% static "images/sunny.jpg" %}" alt="">
17)    <img src="{% static "images/guido.jpg" %}" alt="">
18)    <img src="{% static "images/divine3.jpg" %}" alt="">
19)  </body>
20) </html>
```

views.py:

```
1)  from django.shortcuts import render
2)
3)  # Create your views here.
```

```
def moviesInfo(request):
    my_dict={'head_msg':'Movies Information',
             'sub_msg1':'Sonali slowly getting cured',
             'sub_msg2':'Bahubali-3 is just planning',
             'sub_msg3':'Salman Khan ready to marriage',
             'photo':'images/sunny.jpg'}
    return render(request,'news.html',context=my_dict)
def sportsInfo(request):
    my_dict={'head_msg':'Sports Information',
             'sub_msg1':'Anushka Sharma Firing Like anything',
             'sub_msg2':'Kohli updating in game anything can happend',
             'sub_msg3':'Worst Performance by India-Sehwag',
             }
    return render(request,'news.html',context=my_dict)
def politicsInfo(request):
    my_dict={'head_msg':'Politics Information',
             'sub_msg1':'Achhce Din Aaa gaya',
             'sub_msg2':'Rupee Value now 1$:70Rs',
             'sub_msg3':'In India Single Paisa Black money No more',
             }
    return render(request,'news.html',context=my_dict)
def index(request):
    return render(request,'index.html')
```

settings.py:

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
TEMPLATES_DIR=os.path.join(BASE_DIR,'templates')
STATIC_DIR=os.path.join(BASE_DIR,'static')

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'newsApp'
]

TEMPLATES = [
    {
        '
        'DIRS': [TEMPLATES_DIR,],

        ],
    },
```

```
    },
] 

STATIC_URL = '/static/'
STATICFILES_DIRS=[
STATIC_DIR,
]
```

urls.py:

```
from django.conf.urls import url
from django.contrib import admin
from newsApp import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^movies/', views.moviesInfo),
    url(r'^sports/', views.sportsInfo),
    url(r'^politics/', views.politicsInfo),
    url(r'^$', views.index),
]
```

Single Project with Multiple applications:

urls.py:

```
from django.conf.urls import url
from django.contrib import admin
from firstApp import views as v1
from secondApp import views as v2

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^firstwish/', v1.wish1),
    url(r'^secondwish/', v2.wish2),
]
```

Working with Models and Databases:

As the part of web application development, compulsory we required to interact with database to store our data and to retrieve our stored data.

Django provides a big in-built support for database operations. Django provides one inbuilt database sqlite3.
For small to medium applications this database is more enough. Django can provide support for other databases also like oracle, mysql,postgresql etc

Database configuration:

Django by default provides sqlite3 database. If we want to use this database,we are not required to do any configurations.

The default sqllite3 configurations in settings.py file are declared as follows.

settings.py:

```
1)  DATABASES = {
2)      'default': {
3)          'ENGINE': 'django.db.backends.sqlite3',
4)          'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
5)      }
6)  }
```

If we don't want sqlite3 database then we have to configure our own database with the following parameters.

1. ENGINE: Name of Database engine
2. NAME: Database Name
3. USER: Database Login user name
4. PASSWORD: Database Login password
5. HOST: The Machine on which database server is running
6. PORT: The port number on which database server is running

Note: Most of the times HOST and PORT are optional.

How to check django database connection:

We can check whether django database configurations are properly configured or not by using the following commands from the shell

```
D:\djangoprojects\modelProject>py manage.py shell
>>> from django.db import connection
>>> c=connection.cursor()
```

If we are not getting any error means our database configurations are proper.

Configuration of mysql database:

First we have to create our own logical database in the mysql.

```
mysql> create database employeedb;
```

We have to install mysqlclient by using pip as follows

pip install --only-binary :all: mysqlclient

settings.py:

```
1)  DATABASES = {
2)     'default': {
3)        'ENGINE': 'django.db.backends.mysql',
4)        'NAME': 'employeedb',
5)        'USER':'root',
6)        'PASSWORD':'root'
7)     }
8)  }
```

Checking configurations:

```
D:\djangoprojects\modelProject>py manage.py shell
>>> from django.db import connection
>>> c=connection.cursor()
```

Configuration of Oracle database:

```
1)  DATABASES = {
2)     'default': {
3)        'ENGINE': 'django.db.backends.oracle',
4)        'NAME': 'XE',
5)        'USER':'scott',
6)        'PASSWORD':'tiger'
7)     }
8)  }
```

Note: We can find oracle database name by using the following command.

```
SQL> select * from global_name;
```

Model Class:

- A Model is a Python class which contains database information.
- A Model is a single, definitive source of information about our data. It contains fields and behavior of the data what we are storing.
- Each model maps to one database table.
- Every model is a Python class which is the child class of (django.db.models.Model)
- Each attribute of the model represents a database field.
- We have to write all model classes inside 'models.py' file.

1. Create a project and application and link them.

django-admin startproject modelProject
cd modelProject/
python manage.py startapp testApp

After creating a project and application, in the models.py file, write the following code:

models.py:

```
1)  from django.db import models
2)
3)  # Create your models here.
4)
5)  class Employee(models.Model):
6)      eno=models.IntegerField()
7)      ename=models.CharField(max_length=30)
8)      esal=models.FloatField()
9)      eaddr=models.CharField(max_length=30)
```

Note: This model class will be converted into Database table. Django is responsible for this.

table_name: appName_Employee
fields: eno, ename, esal and eaddr. And one extra field: id
behaviors: eno is of type Integer, ename is of type Char and max_length is 30 characters.

Hence,

Model Class = Database Table Name + Field Names + Field Behaviors

Converting Model Class into Database specific SQL Code:

Once we write Model class, we have to generate the corresponding SQL Code. For this, we have to use "makemigrations" command.

Python manage.py make migrations

It results the following :
Migrations for 'testApp':
testApp/migrations/0001_initial.py
- Create model Employee

How to see corresponding sql code of migrations:

To see the generated SQL Code, we have to use the following command "sqlmigrate"

python manage.py sqlmigrate testApp 0001

```
1)  BEGIN;
2)  --
3)  -- Create model Employee
4)  --
5)  CREATE TABLE "testApp_employee" ("id" integer NOT NULL PRIMARY KEY AUTOINCREME
    NT, "eno" integer NOT NULL, "ename" varchar(30) NOT NULL, "esal" real NOT NULL, "eadd
    r" varchar(30) NOT NULL);
6)  COMMIT;
```

Note: **Here 0001 is the file passed as an argument**

"id" field:
1. For every table(model), Django will generate a special column named with "id".
2. ID is a Primary Key. (Unique Identifier for every row inside table is considered as a primary key).
3. This field(id) is auto increment field and hence while inserting data, we are not required to provide data for this field.
4. This id field is of type "AutoField"
5. We can override the behavior of "id" field and we can make our own field as "id".
6. Every Field is by default "NOT NULL".

How to execute generated SQL code (migrate command):

After generating sql code, we have to execute that sql code to create table in database. For this, we have to use 'migrate' command.

python manage.py migrate

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, testApp
Running migrations:
 Applying contenttypes.0001_initial... OK
 Applying auth.0001_initial... OK
 Applying admin.0001_initial... OK
 Applying admin.0002_logentry_remove_auto_add... OK
 Applying contenttypes.0002_remove_content_type_name... OK
 Applying auth.0002_alter_permission_name_max_length... OK
 Applying auth.0003_alter_user_email_max_length... OK
 Applying auth.0004_alter_user_username_opts... OK
 Applying auth.0005_alter_user_last_login_null... OK
 Applying auth.0006_require_contenttypes_0002... OK
 Applying auth.0007_alter_validators_add_error_messages... OK
 Applying auth.0008_alter_user_username_max_length... OK
 Applying sessions.0001_initial... OK
 Applying testApp.0001_initial... OK
```

Note: **Now tables will be created in the database.**

What is the advantage of creating tables with 'migrate' Command

If we use 'migrate' command, then all Django required tables will be created in addition to our application specific tables. If we create table manually with sql code, then only our application specific table will be created and django my not work properly. Hence it is highly recommended to create tables with 'migrate' command.

How to check created table in django admin interface:

We have to register model class in 'admin.py' file.

admin.py:

```
1)  from django.contrib import admin
2)  from testApp.models import Employee
3)
4)  # Register your models here.
5)
6)  admin.site.register(Employee)
```

Creation of super user to login to admin interface:

We can create super user with the following command by providing username, mailid, password.

python manage.py createsuperuser

We can login to admin interface → Start the server and login to admin interface using the created credentials.

python manage.py runserver

Open the following in browser:
http://127.0.0.1:8000/admin/

Difference between makemigrations and migrate:

"makemigrations" is responsible to generate SQL code for Python model class whereas "migrate" is responsible to execute that SQL code so that tables will be created in the database.

To display data in admin interface in browser:

models.py:

```
1)  from django.db import models
2)
3)  # Create your models here.
```

```
4)
5)  class Employee(models.Model):
6)      eno=models.IntegerField()
7)      ename=models.CharField(max_length=30)
8)      esal=models.FloatField()
9)      eaddr=models.CharField(max_length=30)
10)
11)     def __str__(self):
12)         return 'Employee Object with eno: +str(self.no)'
```

admin.py:

```
1)  from django.contrib import admin
2)  from testApp.models import Employee
3)
4)  # Register your models here.
5)
6)  class EmployeeAdmin(admin.ModelAdmin):
7)      list_display=['eno','ename','esal','eaddr']
8)
9)  admin.site.register(Employee,EmployeeAdmin)
```

Now we can write views to get data from the database and send to template.
Before writing views.py file, create “templates” and “static” folder with respective application folders and HTML and CSS files and link them in settings.py file.

Views.py:

```
1)  from django.shortcuts import render
2)  from testApp.models import Employee
3)
4)  # Create your views here.
5)  def empdata(request):
6)      emp_list=Employee.objects.all()
7)      my_dict={'emp_list':emp_list}
8)      return render(request, 'testApp/emp.html', context=my_dict)
```

emp.html

```
1)  <!DOCTYPE html>
2)  {% load staticfiles %}
3)  <html lang="en" dir="ltr">
4)    <head>
5)      <meta charset="utf-8">
6)      <title></title>
7)      <link rel="stylesheet" href="{% static '/css/demo.css'%}">
8)    </head>
9)
10)   <body>
```

```
11)   <h1> The employees list is : </h1>
12)
13)   {% if emp_list %}
14)   <table>
15)     <thead>
16)       <th> eno </th>
17)       <th> ename </th>
18)       <th> esal </th>
19)       <th> eaddr </th>
20)     </thead>
21)
22)     {% for emp in emp_list %}
23)     <tr>
24)       <td> {{emp.eno}}</td>
25)       <td>{{emp.ename}}</td>
26)       <td>{{emp.esal}}</td>
27)       <td> {{emp.eaddr}}</td>
28)     </tr>
29)     {% endfor %}
30)
31)   </table>
32)   {%else%}
33)   <p> No records found </p>
34)   {% endif %}
35)
36)  </body>
37) </html>
```

MVT Diagram:

FAQs:

How to configure database inside settings.py?
How to check connections?
How to define Model class inside models.py
How we can perform makemigrations?
How we can perform migrate?
How to add our model to admin interface inside admin.py
To display total data how to write ModelAdmin class inside admin.py
how to createsuperuser?
How to login to admin interface and add data to our tables?
How to see generated sqlcode b'z of makemigrations

Faker Module:

We can use Faker Module to generate fake data for our database models.

```
1) from faker import Faker
2) from random import *
3) fakegen=Faker()
4) name=fakegen.name()
5) print(name)
6) first_name=fakegen.first_name()
7) last_name=fakegen.last_name()
8) print(first_name)
9) print(last_name)
10) date=fakegen.date()
11) print(date)
12) number=fakegen.random_number(5)
13) print(number)
14) email=fakegen.email()
15) print(email)
16) print(fakegen.city())
17) print(fakegen.random_int(min=0, max=9999))
18) print(fakegen.random_element(elements=('Project Manager', 'TeamLead', 'Software Engineer')))
```

Note: **Working with mysql db(studentinfo project)**

Django Forms:

It is the very important concept in web development.
The main purpose of forms is to take user input.

Eg: **login form, registration form, enquiry form etc**

From the forms we can read end user provided input data and we can use that data based on requirement. We may store in the database for future purpose. We may use just for validation/authentication purpose etc

Here we have to use Django specific forms but not HTML forms.

Advantages of Django Forms over HTML forms:

1. We can develop forms very easily with python code
2. We can generate HTML Form widgets/components (like textarea, email, pwd etc) very quickly
3. Validating data will become very easy
4. Processing data into python data structures like list, set etc will become easy
5. Creation of Models based on forms will become easy etc.

Process to generate Django forms:

Step-1: Creation of forms.py file in our application folder with our required fields.

forms.py:

```
1)  from django import forms
2)  class StudentForm(forms.Form):
3)      name=forms.CharField()
4)      marks=forms.IntegerField()
```

Note: name and marks are the field names which will be available in html form

Step-2: usage of forms.py inside views.py file:

views.py file is responsible to send this form to the template html file

views.py:

```
1)  from django.shortcuts import render
2)  from . import forms
3)
4)  # Create your views here.
5)  def studentinputview(request):
6)      form=forms.StudentForm()
7)      my_dict={'form':form}
8)      return render(request,'testapp/input.html',context=my_dict)
```

 # alternative short way:

```
1)  def studentinputview(request):
2)      form=forms.StudentForm()
3)      return render(request,'testapp/input.html',{'form':form})
```

Note: context parameter is optional.We can pass context parameter value directly without using keyword name 'context'

Step-3: Creation of html file to hold form:

Inside template file we have to use template tag to inject form

{{form}}

It will add only form fields. But there is no <form> tag and no submit button.
Even the fields are not arranged properly.It is ugly form.

We can make proper form as follows

```
<h1>Registration Form</h1>
 <div class="container" align="center">
   <form   method="post">
  {{form.as_p}}
  <input type="submit" class="btn btn-primary" name="" value="Submit">
   </form>

 </div>
```

input.html:

```
<!DOCTYPE html>
{%load staticfiles%}
<html lang="en" dir="ltr">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="{%static "css/bootstrap.css"%}">
   <link rel="stylesheet" href="{%static "css/demo2.css"%}">
   <title></title>
 </head>
 <body>
   <h1>Registration Form</h1>
   <div class="container" align="center">
     <form   method="post">
        {{form.as_p}}
        <input type="submit" class="btn btn-primary" name="" value="Submit">
     </form>
   </div>
 </body>
</html>
```

If we submit this form we will get 403 status code response
Forbidden (403)
CSRF verification failed. Request aborted.

Help
Reason given for failure:

CSRF token missing or incorrect.

Every form should satisfy CSRF (Cross Site Request Forgery) Verification, otherwise Django won't accept our form.
It is meant for website security. Being a programmer we are not required to worry anything about this. Django will takes care everything.
But we have to add csrf_token in our form.

```
<h1>Registration Form</h1>
   <div class="container" align="center">
       <form   method="post">
```

```
4)          {{form.as_p}}
5)          {% csrf_token %}
6)          <input type="submit" class="btn btn-primary" name="" value="Submit">
7)         </form>
8)     </div>
```

If we add csrf_token then in the generate form the following hidded field will be added,which makes our post request secure

```
<input type='hidden' name='csrfmiddlewaretoken'
value='1ZqIJJqTLMVa6RFAyPJh7pwzyFmdiHzytLxJIDzAkKULJz4qHcetLoKEsRLwyz4h'/>
```

The value of this hidden field is keep on changing from request to request.Hence it is impossible to forgery of our request.

If we configured csrf_token in html form then only django will accept our form.

How to process input data from the form inside views.py file:

We required to modify views.py file. The end user provided input is available in a dictionary named with 'cleaned_data'

views.py:

```
1)  from django.shortcuts import render
2)  from . import forms
3)
4)  # Create your views here.
5)  def studentinputview(request):
6)      form=forms.StudentForm()
7)      if request.method=='POST':
8)          form=forms.StudentForm(request.POST)
9)          if form.is_valid():
10)             print('Form validation success and printing data')
11)             print('Name:',form.cleaned_data['name'])
12)             print('Marks:',form.cleaned_data['marks'])
13)     return render(request,'testapp/input.html',{'form':form})
```

project: formproject

Student FeedBack Form Project

forms.py:

```
1)  from django import forms
2)
3)  class FeedBackForm(forms.Form):
4)      name=forms.CharField()
5)      rollno=forms.IntegerField()
6)      email=forms.EmailField()
7)      feedback=forms.CharField(widget=forms.Textarea)
```

views.py:

```
1)  from django.shortcuts import render
2)  from . import forms
3)
4)  def feedbackview(request):
5)      form=forms.FeedBackForm()
6)      if request.method=='POST':
7)          form=forms.FeedBackForm(request.POST)
8)          if form.is_valid():
9)              print('Form Validation Success and printing information')
10)             print('Name:',form.cleaned_data['name'])
11)             print('Roll No:',form.cleaned_data['rollno'])
12)             print('Email:',form.cleaned_data['email'])
13)             print('FeedBack:',form.cleaned_data['feedback'])
14)     return render(request,'testapp/feedback.html',{'form':form})
```

feedback.html:

```
1)  <!DOCTYPE html>
2)  {% load staticfiles%}
3)  <html lang="en" dir="ltr">
4)    <head>
5)      <meta charset="utf-8">
6)      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
    otstrap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
    gin="anonymous">
7)      <link rel="stylesheet" href="{% static "css/demo4.css" %}">
8)      <title></title>
9)    </head>
10)   <body>
11)     <div class="container" align='center'>
12)       <h1>Student Feedback Form</h1><hr>
```


```
13)       <form class="" action="index.html" method="post">
14)         {{form.as_p}}
15)         {% csrf_token %}
16)         <input type="submit"  class="btn btn-primary" value="Submit Feedback">
17)       </form>
18)     </div>
19)   </body>
20) </html>
```

Form Validations:

Once we submit the form we have to perform validations like
1. Length of the field should not be empty
2. The max number of characters should be 10
3. The first character of the name should be 'd' etc

We can implement validation logic by using the following 2 ways.

1. Explicitly by the programmer by using clean methods
2. By using Django inbuilt validators

Note: All validations should be implemented in the forms.py file

1. Explicitly by the programmer by using clean methods:

The syntax of clean method:

clean_fieldname(self)

In the FormClass for any field if we define clean method then at the time of submit the form, Django will call this method automatically to perform validations. If the clean method won't raise any error then only form will be submitted.

```
1)  from django import forms
2)
3)  class FeedBackForm(forms.Form):
4)      name=forms.CharField()
5)      rollno=forms.IntegerField()
6)      email=forms.EmailField()
7)      feedback=forms.CharField(widget=forms.Textarea)
8)
9)      def clean_name(self):
10)         inputname=self.cleaned_data['name']
11)         if len(inputname) < 4:
```

```
12)         raise forms.ValidationError('The Minimum no of characters in the name field should
    be 4')
13)       return inputname
```

The returned value of clean method will be considered by Django at the time of submitting the form.

forms.py:

```
1)  from django import forms
2)  from django.core import validators
3)
4)  class FeedBackForm(forms.Form):
5)     name=forms.CharField()
6)     rollno=forms.IntegerField()
7)     email=forms.EmailField()
8)     feedback=forms.CharField(widget=forms.Textarea)
9)
10)    def clean_name(self):
11)       print('validating name')
12)       inputname=self.cleaned_data['name']
13)       if len(inputname) < 4:
14)          raise forms.ValidationError('The Minimum no of characters in the name field should
    be 4')
15)       return inputname+'durga'
16)    def clean_rollno(self):
17)       inputrollno=self.cleaned_data['rollno']
18)       print('Validating rollno field')
19)       return inputrollno
20)    def clean_email(self):
21)       inputemail=self.cleaned_data['email']
22)       print('Validating email field')
23)       return inputemail
24)
25)    def clean_feedback(self):
26)       inputfeedback=self.cleaned_data['feedback']
27)       print('Validating feedback field')
28)       return inputfeedback
```

server console:

```
validating name
Validating rollno field
Validating email field
Validating feedback field
Form Validation Success and printing information
Name: Durgadurga
Roll No: 101
Email: durgaadvjava@gmail.com
FeedBack: This is sample feedback
```

Note:

1 .Django will call these filed level clean methods automatically and we are not required to call explicitly.

2. Form validation by using clean methods is not recommended.

2. Django's inbuilt core validators:

Django provides several inbuilt core validators to perform very common validations. We can use these validators directly and we are not required to implement.

Django's inbuilt validators are available in the django.core module.

from django.core import validators

To validate Max number of characters in the feedback as 40,we have to use inbuilt validators as follows.

forms.py:

```
1)  from django import forms
2)  from django.core import validators
3)
4)  class FeedBackForm(forms.Form):
5)      name=forms.CharField()
6)      rollno=forms.IntegerField()
7)      email=forms.EmailField()
8)      feedback=forms.CharField(widget=forms.Textarea,validators=[validators.MaxLengthVali
    dator(40)])
```

Note: **We can use any number of validators for the same field**

feedback=forms.CharField(widget=forms.Textarea,validators=[validators.MaxLengthValidator(40), validators.MinLengthValidator(10)])

Note: **Usage of built in validators is very easy when compared with clean methods.**

How to implement custom validators by using the same validator parameter:

The value of name parameter should starts with 'd' or 'D'. We can implement this validation as follows

```
1)  def starts_with_d(value):
2)      if value[0].lower() != 'd':
3)          raise forms.ValidationError('Name should be starts with d | D')
4)
5)  class FeedBackForm(forms.Form):
```

```
6)    name=forms.CharField(validators=[starts_with_d])
7)    rollno=forms.IntegerField()
```

Validation of total form directly by using a single clean method:

Whenever we are submitting the form Django will call clean() method present in our Form class. In that method we can implement all validations.

forms.py:

```
1)  from django import forms
2)  from django.core import validators
3)
4)  class FeedBackForm(forms.Form):
5)     name=forms.CharField()
6)     rollno=forms.IntegerField()
7)     email=forms.EmailField()
8)     feedback=forms.CharField(widget=forms.Textarea)
9)
10)    def clean(self):
11)       print('Total Form  Validation...')
12)       total_cleaned_data=super().clean()
13)       inputname=total_cleaned_data['name']
14)       if inputname[0].lower() != 'd':
15)          raise forms.ValidationError('Name parameter should starts with d')
16)       inputrollno=total_cleaned_data['rollno']
17)       if inputrollno <=0:
18)          raise forms.ValidationError('Rollno should be > 0')
```

How to check original pwd and reentered pwd are same or not:

forms.py:

```
1)  from django import forms
2)  from django.core import validators
3)
4)  class FeedBackForm(forms.Form):
5)     name=forms.CharField()
6)     password=forms.CharField(widget=forms.PasswordInput)
7)     rpassword=forms.CharField(label='Re Enter Password',widget=forms.PasswordInput)
8)     rollno=forms.IntegerField()
9)     email=forms.EmailField()
10)    feedback=forms.CharField(widget=forms.Textarea)
11)
12)    def clean(self):
13)       print('validating passwords match...')
14)       total_cleaned_data=super().clean()
15)       fpwd=total_cleaned_data['password']
```

```
16)     spwd=total_cleaned_data['rpassword']
17)     if fpwd != spwd:
18)         raise forms.ValidationError('Both passwords must be matched')
```

How to prevent requests from BOT:

Generally form requests can be send by end user. Sometimes we can write automated programming script which is responsible to fill the form and submit. This automated programming script is nothing but BOT.

The main objectives of BOT requests are:

1. To create unnecessary heavy traffic to the website, which may crash our application

2. To spread malware (viruses)

Being web developer compulsory we have to think about BOT requests and we have to prevent these requests.

How to prevent BOT Requests:

In the form we will place one hidden form field, which is not visible to the end user. Hence there is no chance of providing value to this hidden field.

But BOT will send the value for this hidden field also.If hidden field got some value means it is the request from BOT and prevent that form submission.

```
1)  from django import forms
2)  from django.core import validators
3)
4)  class FeedBackForm(forms.Form):
5)      name=forms.CharField()
6)      password=forms.CharField(widget=forms.PasswordInput)
7)      rpassword=forms.CharField(label='Re Enter Password',widget=forms.PasswordInput)
8)      rollno=forms.IntegerField()
9)      email=forms.EmailField()
10)     feedback=forms.CharField(widget=forms.Textarea)
11)     bot_handler=forms.CharField(required=False,widget=forms.HiddenInput)
12)
13)     def clean(self):
14)         print('validating passwords match...')
15)         total_cleaned_data=super().clean()
```

```
16)      fpwd=total_cleaned_data['password']
17)      spwd=total_cleaned_data['rpassword']
18)      if fpwd != spwd:
19)         raise forms.ValidationError('Both passwords must be matched')
20)      bot_handler_value=total_cleaned_data['bot_handler']
21)      if len(bot_handler_value)>0:
22)         raise forms.ValidationError('Request from BOT...cannot be submitted!!!')
```

Notes with bot field:

```
1)  class FeedBackForm(forms.Form):
2)     normal fields..
3)     bot_handler=forms.CharField(required=False,widget=forms.HiddenInput)
4)
5)     def clean(self):
6)        total_cleaned_data=super().clean()
7)        bot_handler_value=total_cleaned_data['bot_handler']
8)        if len(bot_handler_value)>0:
9)           raise forms.ValidationError('Request from BOT...cannot be submitted!!!')
```

Note: Other ways to prevent BOT requests
1. By using Captchas
2. By using image recognizers
   (like choose 4 images where car present)

Note:
Other way to create project
py -m django startproject modelformproject

Model Forms (Forms based on Model):

Sometimes we can create form based on Model, such type of forms are called model based forms or model forms.

The main advantage of model forms is we can grab end user input and we can save that input data very easily to the database.

Django provides inbuilt support to develop model based forms very easily.

How to develop Model based forms:

1. While develop FormClass instead of inheriting forms.Form class,we have to inherit forms.ModelForm class.

```
class RegisterForm(forms.ModelForm):
    ....
```

2. We have to write one nested class (Meta class) to specify Model information and required fields.

```
class RegisterForm(forms.ModelForm):
  # field declarations if we are performing any custom validations.If we are not defining any custom validations then here we are not required to specify any field.

  class Meta:
    # we have to specify Model class name and requied fields
    model=Student
    fields='__all__'
```

Case-1: Instead of all fields if we want only selected fields,then we have to specify as follows

```
class Meta:
  model=Student
  fileds=('field1','field2','field3')
```

In the form only 3 fields will be considered.
If Model class contains huge number of fields and we required to consider very less number of fields in the form then we should use this approach.

Case-2:

Instead of all fields if we want to exclude certain fields,then we have to specify as follows

```
class Meta:
  model=Student
  exclude=['field1','field2']
```

In the form all fields will be considered except field1 and field2.

If the Model class contains huge number of fields and if we want to exclude very few fields then we have to use this approach.

Q. In Model based forms, how many ways are there to specify fields information

Ans: 3 ways
1. All fields
2. Include certain fields
3. Exclude certain fields

Note: The most commonly used approach is to include all fields.

How to save user's input data to database in Model based forms:

We have to use save() method.

```
def student_view(request):
    ...
    if request.method=='POST':
        form=RegisterForm(request.POST)
            if form.is_valid():
                form.save(commit=True)
            ..
```

Demo project-1 (modelformproject):

models.py:

```
1)  from django.db import models
2)
3)  # Create your models here.
4)  class Student(models.Model):
5)      name=models.CharField(max_length=30)
6)      marks=models.IntegerField()
```

forms.py:

```
1)  from django import forms
2)  from testapp.models import Student
3)  class StudentForm(forms.ModelForm):
4)      #fields with validations
5)      class Meta:
6)          model=Student
7)          fields='__all__'
```

views.py:

```
1)  from django.shortcuts import render
2)  from . import forms
3)
4)  # Create your views here.
5)  def student_view(request):
6)      form=forms.StudentForm
7)      if request.method=='POST':
8)          form=forms.StudentForm(request.POST)
9)          if form.is_valid():
10)             form.save(commit=True)
```

```
11)    return render(request,'testapp/studentform.html',{'form':form})
```

Demo Project-2: movieproject

Advanced Templates:

1. Template Inheritance
2. Template Filters
3. Template tags for relative urls

1. Template Inheritance:

If multiple template files have some common code,it is not recommended to write that common code in every template html file. It increases length of the code and reduces readability. It also increases development time.

We have to seperate that common code into a new template file,which is also known as base template. The remaining template files should required to extend base template so that the common code will be inherited automatically.

Inheriting common code from base template to remaining templates is nothing but template inheritance.

How to implement template inheritane:

base.html:

```
1)  <!DOCTYPE html>
2)   html,css,bootstrap links
3)   <body>
4)     common code required for every child tempalte
5)     {% block child_block%}
6)       Anything outside of this block available to child tag.
7)       in child template the specific code should be in this block
8)     {%endblock%}
9)   </body>
10)  </html>
```

child.html:

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)    {% block child_block %}
4)       child specific extra code
5)    {%endblock%}
```

Demo program: advtempproject

base.html:

```
1)  <!DOCTYPE html>
2)  {%load staticfiles%}
3)  <html lang="en" dir="ltr">
4)    <head>
5)      <meta charset="utf-8">
6)      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
    otstrap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
    gin="anonymous">
7)      <link rel="stylesheet" href="{%static "css/advtemp.css"%}">
8)      <title></title>
9)    </head>
10)   <body>
11)     <nav class="navbar navbar-default navbar-fixed-top navbar-inverse">
12)       <div class="container-fluid">
13)
14)         <div class="navbar-header">
15)           <a class="navbar-brand" href="/">DURGA NEWS</a>
16)         </div>
17)           <ul class="nav navbar-nav">
18)             <li class="active"><a href="/">Home <span class="sr-
    only">(current)</span></a></li>
19)             <li><a href="/movies">Movies</a></li>
20)             <li><a href="/sports">Sports</a></li>
21)             <li><a href="/politics">Politics</a></li>
22)           </ul>
23)       </div><!-- /.container-fluid -->
24)  </nav>
25) <div class="container">
26)   {%block body_block%}
27)     <!-- outside of this block everything available to child tempaltes -->
28)   {%endblock%}
29) </div>
30) </body>
31) </html>
```

index.html:

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)    {% block body_block %}
4)      <br><br><br><br><br><br>
5)      <h1>Welcome to DURGA NEWS PORTAL</h1>
6)    {%endblock%}
```

sports.html:

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)   {% block body_block %}
4)   <br><br><br>
5)    <h1>Sports Information</h1>
6)   {%endblock%}
```

Advantages of Template Inheritance:

1. What ever code available in base template is by default available to child templates and we are not required to write again.Hence it promotes Code Reusability (DRY)

2. It reduces length of the code and improves readability

3. It reduces development time

4. It provides unique and same look and feel for total web application.

Note: Based on our requirement we can extend any number of base templates.i.e Multiple Inheritance is applicable for templates.

Tempalte Filters:

In the template file, the injected data can be displayed by using template tags.

{{emp.eno}}

Before displaying to the end user if we want to perform some modification to the injected text, like cut some information or converting to title case etc,then we should go for Template filters.

Syntax of Template Filter:

{{value|filtername:"argument"}}

Filter may take or may not take arguments.i.e arguments are optional.

Eg:

```
<li>{{msg1|lower}}</li>
```

msg1 will be displayed in lower case

```
<li>{{msg3|add:"Durga"}}</li>
```

"Durga" will be added to msg3 and then display the result to the end user

{{ msg|title }}

{{ my_date|date:"Y-m-d" }}

Note: There are tons of built in filters are available.

https://docs.djangoproject.com/en/2.1/ref/templates/builtins/#ref-templates-builtins-filters

How to create our own filter:

Based on our requirement we can create our own filter.

Steps:

1. Create a folder 'templatetags' inside our application folder

2. Create a special file named with __init__.py inside templatetags folder, so that Django will consider this folder as a valid python package

3. Create a python file inside templatetags folder to define our own filters

cust_filters.py -->any name

cust_filters.py:

```
1)  from django import template
2)  register=template.Library()
3)
4)  def first_eight_upper(value):
5)      '''This is my own filter'''
6)      result=value[:8].upper()
7)      return result
8)
9)  register.filter('f8upper',first_eight_upper)
```

Note: We can also register filter with the decorator as follows.

```
1)  from django import template
2)  register=template.Library()
3)
4)  @register.filter(name='f8upper')
5)  def first_eight_upper(value):
6)      '''This is my own filter'''
7)      result=value[:8].upper()
8)      return result
```

f8upper is the name of the filter which can be used inside template file.

4. Inside template file we have to load the filter file as follows(In the child template bunot in base template)

{%load cust_filters%}

5. We can invoke the filter as follows

{{msg|f8upper}}

movies.html:

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)   {% block body_block %}
4)     <h1>Movies Information</h1><hr>
5)     {%load cust_filters%}
6)     <ul>
7)       <li>{{msg1|lower}}</li>
8)       <li>{{msg2|upper}}</li>
9)       <li>{{msg3|add:"--Durga"}}</li>
10)      <li>{{msg4|f8upper}}</li>
11)      <li>{{msg5}}</li>
12)  </ul>
13)  {%endblock%}
```

Eg 2: Custom Filter with argument

```
@register.filter(name='c_and_c')
def cut_and_concate(value,arg):
   result=value[:4]+str(arg)
   return result
```

Note: The main advantage of template filters is we can display the same data in different styles based on our requirement.

advtemp project

Template Tags for URLs:

```
<a href='testapp1/thankyou'>Thank You </a>

<a href="{% url 'thankyou' %}">Thank You </a>

<a href="{% url 'testapp.views.thankyou' %}">Thank You </a>

<a href="{% url 'testapp:thankyou' %}">Thank You </a>
```

Session Management:

Client and Server can communicate with some common language which is nothing but HTTP.

The basic limitation of HTTP is, it is stateless protocol. i.e it is unable to remember client information for future purpose across multiple requests. Every request to the server is treated as new request.

Hence some mechanism must be required at server side to remember client information across multiple requests.This mechanism is nothing but session management mechanism.

The following are various session management mechanisms.

1. Cookies
2. Session API
3. URL Rewriting
4. Hidden Form Fields
etc

Session Management By using Cookies:

Cookie is a very small amount of information created by Server and maintained by client.

Diagram

Whenever client sends a request to the server,if server wants to remember client information for the future purpose then server will create cookie object with the required information. Server will send that Cookie object to the client as the part of response. Client will save that cookie in its local machine and send to the server with every consecutive request. By accessing cookies from the request server can remember client information.

How to test our browser supports Cookies or not:

We have to use the following 3 methods on the request object.

1. set_test_cookie()
2. test_cookie_worked()
3. delete_test_cookie()

views.py:

```
1)  from django.shortcuts import render
2)  from django.http import HttpResponse
3)
4)  # Create your views here.
5)  def index(request):
6)      request.session.set_test_cookie()
7)      return HttpResponse('<h1>index Page</h1>')
8)
9)  def check_view(request):
10)     if request.session.test_cookie_worked():
11)         print('cookies are working properly')
12)         request.session.delete_test_cookie()
13)         return HttpResponse('<h1>Checking Page</h1>')
```

Note: **Before executing this program compulsory we should perform migrate.**

Session Management by using Cookies:

views.py:

```
1)  from django.shortcuts import render
2)
3)  # Create your views here.
4)  def count_view(request):
5)      if 'count' in request.COOKIES:
6)          newcount=int(request.COOKIES['count'])+1
7)      else:
8)          newcount=1
9)      response=render(request,'testapp/count.html',{'count':newcount})
10)     response.set_cookie('count',newcount)
11)     return response
```

count.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
```

```
4)     <meta charset="utf-8">
5)     <title></title>
6)     <style >
7)     span{
8)       font-size: 200px;
9)       font-weight: 900;
10)    }
11)
12)    </style>
13)   </head>
14)   <body>
15)    <h1>Page Count is: <span> {{count}}<span></h1>
16)   </body>
17) </html>
```

Session Management by using Cookies Demo Program-3:

```
1)  from django.shortcuts import render
2)  from testapp.forms import LoginForm
3)  import datetime
4)
5)  # Create your views here.
6)  def home_view(request):
7)     form=LoginForm()
8)     return render(request,'testapp/home.html',{'form':form})
9)
10) def date_time_view(request):
11)    # form=LoginForm(request.GET)
12)    name=request.GET['name']
13)    response=render(request,'testapp/datetime.html',{'name':name})
14)    response.set_cookie('name',name)
15)    return response
16)
17) def result_view(request):
18)    name=request.COOKIES['name']
19)    date_time=datetime.datetime.now()
20)    my_dict={'name':name,'date_time':date_time}
21)    return render(request,'testapp/result.html',my_dict)
```

home.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
```

```
8)     <h1>Welcome to DURGASOFT</h1>
9)     <form  action="/second">
10)     {{form.as_p}}
11)     {%csrf_token%}
12)     <input type="submit" name="" value="Enter Name">
13)    </form>
14)   </body>
15) </html>
```

datetime.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>Hello {{name}}</h1> <hr>
9)      <a href="/result">Click Here to get Date and Time</a>
10)   </body>
11) </html>
```

result.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>Hello {{name}}</h1><hr>
9)      <h1>Current Date and Time:{{date_time}}</h1>
10)     <a href="/result">Click Here to get Updated Date and Time</a>
11)   </body>
12) </html>
```

Cookie Example-4:

Diagram

views.py:

```
1)  from django.shortcuts import render
2)
3)  # Create your views here.
4)  def name_view(request):
5)      return render(request,'testapp/name.html')
6)
7)  def age_view(request):
8)      name=request.GET['name']
9)      response=render(request,'testapp/age.html',{'name':name})
10)     response.set_cookie('name',name)
11)     return response
12)
13) def gf_view(request):
14)     age=request.GET['age']
15)     name=request.COOKIES['name']
16)     response=render(request,'testapp/gf.html',{'name':name})
17)     response.set_cookie('age',age)
18)     return response
19)
20) def results_view(request):
21)     name=request.COOKIES['name']
22)     age=request.COOKIES['age']
23)     gfname=request.GET['gfname']
24)     response=render(request,'testapp/results.html',{'name':name,'age':age,'gfname':gfnam
    e})
25)     response.set_cookie('gfname',gfname)
26)     return response
```

name.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)    </head>
7)    <body>
8)      <h1>Welcome to DURGASOFT</h1>
9)      <form action='/age'>
10)       Enter Name: <input type="text" name="name" value=""><br><br>
11)       <input type="submit" name="" value="Submit Name">
12)     </form>
13)
14)   </body>
15) </html>
```

age.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Hello {{name}}..</h1><hr>
      <form action='/gf'>
        Enter Age: <input type="text" name="age" value=""><br><br>
        <input type="submit" name="" value="Submit Age">
      </form>
  </body>
</html>
```

gf.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Hello {{name}}..</h1><hr>

      <form action='/results'>
        Enter Girl Friend Name: <input type="text" name="gfname" value=""><br><br>
        <input type="submit" name="" value="Submit GFName">
      </form>

  </body>
</html>
```

results.html:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Hello {{name}} Thanks for providing info</h1>
    <h2>Please cross check your data and confirm</h2><hr>
    <ul>
      <li>Name:{{name}}</li>
      <li>Age:{{age}}</li>
      <li>Girl Friend Name:{{gfname}}</li>
```

```
14)    </ul>
15)  </body>
16) </html>
```

Limitations of Cookies:

1. By using cookies we can store very less amount of information. The size of the cookie is fixed.Hence if we want to store huge amount of information then cookie is not best choice.

2. Cookie can hold only string information. If we want to store non-string objects then we should not use cookies.

3. Cookie information is stored at client side and hence there is no security

4. Everytime with every request, browser will send all cookies related to that application,which creates network traffic problems.

5. There is a limit on the max number of cookies supported by browser.

To overcome all these limitations we should go for sessions.

Temporary vs Permanent Cookies:

If we are not setting any max_age for the cookie,then the cookies will be stored in browsers cache.Once we closed browser automatically the cookies will be expired.Such type of cookies are called temporary Cookies.

We can set temporary Cookie as follows:
response.set_cookie(name,value)

If we are setting max_age for the cookie,then cookies will be stored in local file system permanently.Once the specified max_age expires then only cookies will be expired.Such type of cookies are called permanent or persistent cookies. We can set Permanent Cookies as follows

response.set_cookie(name,value,max_age=180)
response.set_cookie(name,value,180)

The time unit for max_age is in seconds.

Demo Program-3:

views.py:

```
1)  from django.shortcuts import render
2)  from testapp.forms import ItemAddForm
3)
```

```
4)  # Create your views here.
5)  def index(request):
6)      return render(request,'testapp/home.html')
7)  def additem(request):
8)      form=ItemAddForm()
9)      response=render(request,'testapp/additem.html',{'form':form})
10)     if request.method=='POST':
11)         form=ItemAddForm(request.POST)
12)         if form.is_valid():
13)             name=form.cleaned_data['itemname']
14)             quantity=form.cleaned_data['quantity']
15)             response.set_cookie(name,quantity,180)
16)             # return index(request)
17)     return response
18) def displayitem_view(request):
19)     return render(request,'testapp/showitems.html')
```

home.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <!-- Latest compiled and minified CSS -->
6)  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/boots
    trap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
    gin="anonymous">
7)      <title></title>
8)    </head>
9)    <body>
10)     <div class="container" align='center'>
11)       <div class="jumbotron">
12)         <h1>DURGASOFT ONLINE SHOPPING APP</h1>
13)         <a class="btn btn-primary btn-lg" href="/add" role="button">ADD ITEM</a>
14)         <a class="btn btn-primary btn-lg" href="/display" role="button">Display ITEMS</a>
15)       </div>
16)
17)     </div>
18)   </body>
19) </html>
```

additem.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
```

```
6)    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
      otstrap.min.css" integrity="sha384-
      BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossori
      gin="anonymous">
7)   </head>
8)   <body>
9)    <div class="container" align='center'>
10)    <h1>Add Item Form</h1>
11)    <form  method="post">
12)     {{form.as_p}}
13)     {%csrf_token%}
14)     <input type="submit" name="" value="Add Item">
15)    </form><br><br><br>
16)    <a class="btn btn-primary btn-lg" href="/display" role="button">Display ITEMS</a>
17)   </div>
18)  </body>
19) </html>
```

showitems.html:

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)   <head>
4)    <meta charset="utf-8">
5)    <title></title>
6)   </head>
7)   <body>
8)    <h1>Total Cookies Information:</h1>
9)    {%if request.COOKIES %}
10)   <table border=2>
11)    <thead>
12)     <th>Cookie Name</th>
13)     <th>Cookie Value</th>
14)    </thead>
15)
16)    {% for key,value in request.COOKIES.items %}
17)    <tr>
18)     <td>{{key}}</td>
19)     <td>{{value}}</td>
20)    </tr>
21)    {% endfor %}
22)    </table>
23)   {%else%}
24)   <p>Cookie Information is not available</p>
25)  {%endif%}
26) </body>
27) </html>
```

Session Management By using Session API:

(Django Session Framework)

Diagram

Once client sends request to the server,if server wants to remember client information for the future purpose then server will create session object and store required information in that object. For every session object a unique identifier available which is nothing but sessionid. Server sends the corresponding session id to the client as the part of response. Client retrieves the session id from the response and save in the local file system. With every consecutive request client will that session id. By accessing that session id and corresponding session object server can remember client. This mechanism is nothing but session management by using session api.

Note: Session information will be stored in one of following possibilities

1. Inside a File
2. Inside a database
3. Inside Cache

The most straight forward approach is to use django.contrib.sessions application to store session information in a Django Model/database.

The Model Name is:
django.contrib.sessions.models.Session

Note: To use this approach compulsory the following applicaiton should be configured inside INSTALLED_APPS list of settings.py file.

```
django.contrib.sessions
```

If it is not there then we can add,but we have to synchronize database

```
python manage.py syncdb
```

Note:

```
INSTALLED_APPS = [
  ....
  'django.contrib.sessions',
```

```
    ...
]

MIDDLEWARE = [
    ..
    'django.contrib.sessions.middleware.SessionMiddleware',
    ....
]
```

Useful Methods for session Management:

1. request.session['key']=value
   To add data to the session

2. value=request.session['key']
   To get data from the session

3. request.session.set_expiry(seconds)
   Sets the expiry time for the session

4. request.session.get_expiry_age()
   returns the expiry age in seconds(the number of seconds until this session expire)

5. request.session.get_expiry_date()
   Returns the data on which this session will expire

Note: Before using session object in our application,compulsory we have to migrate. otherwise we will get the following error.
no such table: django_session

Session Demo1:

views.py:

```
1)  from django.shortcuts import render
2)
3)  # Create your views here.
4)  def page_count_view(request):
5)      count=request.session.get('count',0)
6)      newcount=count+1
7)      request.session['count']=newcount
8)      print(request.session.get_expiry_age())
9)      print(request.session.get_expiry_date())
10)     return render(request,'testapp/pagecount.html',{'count':newcount})
```

pagecount.html:


```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
4)      <meta charset="utf-8">
5)      <title></title>
6)      <style >
7)      span{
8)        font-size: 300px;
9)      }
10)
11)     </style>
12)   </head>
13)   <body>
14)     <h1>The Page Count:<span>{{count}}</span></h1>
15)   </body>
16) </html>
```