**HTML**

**HTML :** HYPER TEXT MARKUP LANGUAGE

* HTML is used to building blocks of web page

**TITLE :**

* We can add any tittle name it will displayed on the top of the browser
* Title is written in between the head

<title>HTML Tutorial</title>

**Taggings :**

* In HTML there are six heading are there [h1 to h6]
* For each heading and paragraph there is one opening tag and closing tag is required.
* The data present in the h1 tag is displayed larger in size.

Ex: <h1>My First Heading</h1>

* The paragraph is written in the paragraph tagging[p]
* In this paragraph we can the font into bold , italic

Ex: <p>My first paragraph.</p>

**Lists :**

There are two types of lists are there

1. **Ordered Lists :**

* The data present in the ordered list is displayed in the sequence number.
* In the ordered list we can use li for storing the data.

<body>

<ol>

<li>list 1</li>

<li>list 2</li>

<li>list 3</li>

</ol>

</body>

**Output:**

1. list 1
2. list 2
3. list 3

**2.Unordered List :**

* The data present in the unordered list is displayed with the bullet

<body>

<ul>

<li>list 1</li>

<li>list 2</li>

<li>list 3</li>

</ul>

</body>

**Output:**

* list 1
* list 2
* list 3

There is a combination of ordered and unordered lists are used is called nested list.

**Attributes :**

* Attributes are used to add the more information for the HTML page like adding images and links
* In the source file we can also use the website link.

1. <img src="source file location" alt=" ">
2. <a href="*url link*">*link text*</a>

**Tables :**

* HTML tables allow web developers to arrange data into rows and columns.

<table border="5">

<thead>

<th>country</th>

</thead>

<tr>

<td>india</td>

</tr>

</table>

* <th> tag is used for the headings of the data or for the rows.
* <td> tag is used for the data present for the headings or the column data.
* We can use the border in the table for different shapes and sizes of the data.
* We can add any links and images in the tables**.**

**FORMS :**

* An HTML form is used to collect user input. The user input is most often sent to a server for processing.

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname">  
</form>

* Labels are useful for the selecting radio buttons , checkboxes
* Input is used to enter the information from the user.
* For submit this information in the input we assign type as submit.
* For the radio buttons in the input we give type as radio and for the

checkbox we assign type as checkbox.

**DROP DOWN :**

* For the drop down we can use the select and option
* Select is use for giving the fields and option is used for giving the data for the fields.

<select id="cars" name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

</select>

**LINKS :**

* We can connect the css and java script folders to the HTML page with the link tag.

<link rel="stylesheet" href=" ">

* In the href we can connect the css and js folders in the html.

**CLASS AND ID :**

* The HTML class attribute is used to specify a class for an HTML element.
* One class can give for the multiple HTML elements.
* ID can give for only one element in the html.
* Class and ID can be mainly helpful in the CSS for giving styling for the particular elements or for the group of elements.

<div class="city">  
   <h2>London</h2>  
   <p>London is the capital of England.</p>  
 </div>

<h1 id="myHeader">My Header</h1>

**CSS**

**CSS :** Cascading Style Sheets

* Css is mainly used for the styling of the html page.
* First step in the css is create the css file and link the css file in the html page.
* Now we can work on the css file.

**Comment :**

**/\*** This is a comment **\*/**

**Syntax :**

Selected tag{

Property : value;

}

* For giving the color to the h1 tag

h1{

color : green;

}

* In the css we can change the background color of the website.

body{

background-color : red;

}

* If we want to give the background coloring for some information then we use the div and spans in the html page.

div{

background-color : blue;

}

* We can add the link to the background

body{

background: url( );

}

**BORDER :**

* for giving the borders for the information we use the

border-color : red;

border-width : 10px;

border-style : dotted;

**SELECTORS :**

* selectors are used to select the styling to the specific elements i.e for group of elements and for single elements.
* For single elements we use id.

#name of id{

Property : value;

}

* For group of elements we use class.

**.**name of the class{

Property : value;

}

* \* is used to change the properties of all elements except the id.

\*{

Property : value;

}

**SPECIFICITY :**

* Specificity means giving the specific styling to the specific elements.

**MARIGINS :**

* Margins are used to create space around elements, outside of any defined borders.
* We can adjust the margins for our requirements of the webpage.

  margin-top: 100px;  
  margin-bottom: 100px;  
  margin-right: 150px;  
  margin-left: 80px;

* We can give all four margin properties in the single line

margin :100px 150px 100px 80px;

* If we use the margin as auto then the information is placed at the center.

**PADDING :**

* The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

padding-top: 50px;

padding-right: 30px;

padding-bottom: 50px;

padding-left: 80px;

* We can give all four padding properties in the single line

padding :100px 150px 100px 80px;

**TEXT :**

* We can give the specific coloring to the text

body {  
  color: blue;  
 }

* We can alignment the text to the center, left, right

h1 {  
  text-align: center;  
 }

* text-alignment-last is set the last line of the information is set to the center/right/left

p.a {  
  text-align-last: right;  
 }

* text decorators are used to add a decorators for the text and also add a color to the line.

h1 {  
  text-decoration-line: overline;  
 }

* Text transformation changes the text to uppercase ,lowercase ,capitalize.

p.uppercase {  
   text-transform: uppercase;  
 }

* Text spacing.

p {  
  text-indent: 50px;  
 }

* Text shadow.

h1 {  
   text-shadow: 2px 2px;  
 }

**FONTS :**

* Giving the fonts to the text and also for elements.

  font-family: "Times New Roman", Times, serif;

* we can adjust the font size of the text.

font-size: 20px;

font-weight: bold;

font-style: oblique;

**JAVASCRIPT**

* javascript is a programming language of the web.
* we use the alter syntax to display the output in the browser.

alert(“hello world”)

* The output is displayed in the console of the web page.

console.log(“ ”)

* prompt( ) is used to type something on the browser.

prompt(“enter some information”)

* we can connect the javcascript to the html file.

**VARIABLES :**

* variables are used to store the data values
* in javascript we use four types of variables var ,let ,cons ,automatically.

Automatically:

x=5;

y=6;

Var:

* we can change the value of variable after assigning the value.

var x=5;

var y=6;

let:

* We cannot change the value of the variable after assigning the value.
* let is a block scope

let x=5;

let y=6;

constant:

* We can change the elements of the constant array.
* const is also a block scope

const cars = ["Saab", "Volvo", "BMW"];  
cars[0] = "Toyota";  
cars.push("Audi");

**OPERATORS :**

1.Arthemetic operator [+,-,\*,\*\*,/,%,++]

2.Assignment operator[=]

3.Comparision operator[==,!=,>,<.<=,>=]

4.String Operators

5.Logical operators[&&,||,!]

**DATA TYPES :**

* basic data types in javascript are strings ,numbers , bollean ,undefines and null.
* Object data types are Objects ,array ,Date

**FUNCTIONS :**

* JS functions is a block of code to design for a particular task.
* Functions are created by function followed by function name and parameters present inside the function.
* We can reuse the code in functions.

function name(para1,para2){

//code

}

**OBJECTS :**

* Objects are written in the [name : value] pair

Ex:

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};

**STRINGS :**

* strings a written in single quotes or in double quotes.
* We can write zero or more characters inside the string.

Methods:

1.length

2.slice()

3.substring()

4.replace()

5.replaceAll()

6.toUpperCase()

7.toLowerCase()

8.concat()

9.trim()

10.trimStart()

11.trimEnd()

12.charAt()

**NUMBERS :**

* Js numbers are written with decimals or without decimals.
* We can concatenate two numbers , two strings , one string and one number.
* Trying to do arithmetic with a non-numeric string will result in NaN.

**Methods :**

1.toString()

2.toExponentioal()

3.toFixed()

4.toPrecision()

**ARRAYS** **:**

* Arrays are used to store the multiple values for a single variable.

Ex:

const cars = [“BMW”,”Shift”,”volvo”]

Methods :

1.array length

2.array toString()

3.array pop()

4.array push()

5.array shift()

6.array unshift()

7.array join()

8.array delete()

9.array concat()

10.array sort()

11.array reverse()

ARRAY ITERATION :

* Array iteration methods operate on every array item.

1.forEach()

2.map()

3.flatMap()

4.filter()

5.reduce()

6.reduceRight()

7.every()

8.some()

**DATES :**

* Dates are in the form of year month date hour minute secounds

const d = new Date();

GET DATE METHODS :

1.getFullYear()

2.getMonth()

3.getDate()

4. getDay()

5. getHour()

6. getMinutes()

7. getSecounds()

8. getMillisecounds()

9. getTime()

SET DATE METHODS :

* same methods that are present in the get date but it set the date values.

**MATH :**

* math allowed to perform some mathematical operations on numbers.

METHODS :

Math.round() ------- displays the nearest value

Math.celi() -------- displays the greater value

Math.floor() --------displays the rounded down nearest value

Math.trunc() -------- display the integer value

Math.sqrt()

Math.Pow()

Math.min()

Math.max()

Math.random()

Math.log()

**CONTROL STATEMENTS :**

**if statement :**

* the condition is true then it executes the block of code

if(condition){

//code

}

**else statement :**

* else statement is used when if statement is false\

if(condition){

//code

}else{

//code

}

**else if statement :**

* if there are two or more condition ,if the first condition is false then it goes to the else if statement.

if(condition){

//code

}else if(condition){

//code

}else{

//code

}

**LOOPS :**

* Loops can execute a block of code a number of times.

**1.For loop :**

* Loop through a block of code for number of time.

for(expression1,expression2,expression3){

}

* Expression1 execute one time before the execution of code.
* Expression2 defines the condition.
* Expression3 executed after the execution of code.

**2.while loop :**

* It executed until the block of code gets true.

While(condition){

//code

}

**DJANGO**

* Django is used for the backend web framework.
* Django makes it easier to write the web frame works with python.
* First we want to install the Django and activate to the environment.
* Then we create the first project of the Django , for creating the project we want to enter the command in cmd
* django-admin startproject myproject
* for activating this project cd myproject
* After creating the project we want to create the app for the Django.

* Python manage.py startapp myapp
* After creating the app for the Django we can the write the coding.
* First we can add the project app in the settings.py.
* We can write the code in the views.py and then connect the views.py to the urls.py.
* For executing this we can write the command in cmd

Python manage.py runserver

* We can get the link in the cmd we can copy and paste the link in the browser. If there are no errors then the output will displayed.

Ex:

Views.py:

from django.shortcuts import render

from django.http import HttpResponse

def index(request):

    return HttpResponse("hello world")

url mapping :

from django.contrib import admin

from django.urls import path,include

from myapp import views

urlpatterns = [

    path('',views.index,name="index"),

    path('myapp/',include('myapp.urls')),

    path('admin/', admin.site.urls),

]

**TEMPLATES :**

* Templates are the key part in the Django for the real interaction with the websites.
* We can write the html code in the templates.
* We can create the templates folder for writing the html code.
* We can add the TEMPLATES\_DIR in the settings.py file.

TEMPLATES\_DIR = os.path.join(BASE\_DIR,'templates')

* Connect the html page to the views

return render(request,”other.html”)

**STATIC :**

* Static files are used to insert images to the web page.
* For inserting the images in the web page we use the {% load static %} in the html page and img tag.
* Static is also used to write and connect the css and js to the html page.

<link rel="stylesheet" href="{% static "page.css" % }">

**MODELS :**

* models are used to accept the user information.
* Django can do this entire process in the single command.

Python manage.py migrate

* then register the changes we pass the command

python mange.py makemigrations myapp

* then we add the created models to the admin.py file.
* Entering the admin username and password we create the super user
* For creating the super user use the command line

Python mange.py createsuperuser

* We want to give the username, password, email for login to the admin.

Models.py:

from django.db import models

# Create your models here.

class Student(models.Model):

    name=models.CharField(max\_length=256)

    age=models.IntegerField()

    def \_\_str\_\_(self):

        return self.name

class School(models.Model):

    location=models.CharField(max\_length=256)

    principal=models.CharField(max\_length=254)

    name=models.CharField(max\_length=256)

    def \_\_str\_\_(self):

        return self.name

admin.py:

from django.contrib import admin

from modela.models import Student,School

# Register your models here.

admin.site.register(School)

admin.site.register(Student)

**FORMS :**

* Default there is no forms are there in the Django but we want to create the forms.py file.
* Forms are similar to the models.

Ex:

from django import forms

class FormName(forms.Form):

    name=forms.CharField()

    age=forms.IntegerField()

    location=forms.CharField()

    text=forms.CharField(widget=forms.textarea)

* We import the forms in the views.py file and create a new view for the form.

Forms.py:

from django import forms

class FormName(forms.Form):

    name=forms.CharField(max\_length=256)

    age=forms.IntegerField()

    location=forms.CharField(max\_length=256)

views.py:

from django.shortcuts import render

from . import forms

# Create your views here.

def index(request):

    return render(request,'modela/index.html')

def form\_name\_view(request):

    form=forms.FormName()

    if request.method == 'POST':

        form = forms.FormName(request.POST)

        if form.is\_valid():

            print("validation success")

            print("Name:"+form.cleaned\_data[name])

            print("Age:"+form.cleaned\_data[age])

            print("Location:"+form.cleaned\_data[location])

    return render(request,'modela/from.html',{'form':form})

**RELATIVE URL’S WITH TEMPLATES :**

* With the help of relative url’s with templates on the web page when we click on the path it moves to the another web page.

  <a href="{% url 'myapp:other' %}">link to other page</a>

    <a href="{% url 'admin:index' %}">link to admin page</a>

    <a href="{% url 'index' %}">link to index page</a>

**URL TEMPLATE INHERITANCE :**

* Template inheritance allows us to inherit data from base html to other html files.
* There is no limit for extension of files from base.html , for multiple files it will be inherit

**base.html: other.html:**

{% block body\_block %} {% extend “myapp/base.html” %}

{% block body\_block %}

{% endblock %} {% endblock %}

**TEMPLATE FILTERS :**

* There is a data present in the models if want to make a slight edit like operators , strings template filters are used.
* General form of template filter is:

{{ value || filter:”parameter” }}

**PASSWORDS :**

* In the settings.py we required two pre-installed packages
* We can add the PASSWORD\_HASHERS in the settings.py file

**CLASS BASED VIEW[CBV] :**

* Previously we created views with the help of functions but in the class based views we create the views with the class.
* There is a slight change in the uls.py file for the CBV.

**Views.py :**

from django.shortcuts import render

from django.views.generic import view

from django.http import HttpResponse

class CBView(View):

    def get(self,request):

        return HttpResponse("HEllo world")

**urls.py :**

from django.contrib import admin

from django.urls import path

from classapp import views

urlpatterns = [

    path('',views.CBView.as\_view(),name=CBView),

    path('admin/', admin.site.urls),

]

**CLASS BASED TEMPLATE VIEW :**

class IndexView(TemplateView):

Template\_name=’index.html’

CRUD:

* We can create , update and delete the data from the models with the help of views.
* For crud we required fields and model.

**ADMIN :**

* **Ordering fields:**
* it will display the fields in the we mention in the models but we can change the fields order.

fields=[field1,field2,field3]

**Adding Search:**

We can add the search bar for the fields

search\_fields=[field3]

**Adding Filters:**

We can add the filters for the fields

list\_filters=[field1,field2,field3]

**Adding Fields:**

We currently see only one field but we can add more fields

list\_display=[field1,field2,field3]

**Editable List:**

We can also edit the data present in the fields

list\_editable=[field1]