HTML

1.What is HTML?

* HTML stands for Hyper Text Markup Language
* HTML is the standard markup language for creating Web pages

2. HTML Element type?

|  |  |  |
| --- | --- | --- |
| **Start tag** | **Element content** | **End tag** |
| <h1> | My first Heanding | </h1> |
| <h6> | My last Heanding | </h6> |
| <p> | My first paragraph | </p> |
| <br> | Line break | none |
| <hr> | underline | none |
| <Img> | Img | none |
| <href> | link | none |

1. <nav>

<ul> unordered and <ol>ordered

<li><a href=”#”></li></a>

</nav>

Html Simple Program

<!DOCTYPE html>

<html>

<head>

<tittle>

</title>

</head>

<body>

<!-- This is a comment -->

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

<!-- Comments are not displayed in the browser -->

</body>

</html>

CSS

1.What is CSS?

* CSS stands for Cascading Style Sheets.
* CSS is the language we use to style an HTML document.

2. How To Add CSS?

There are three type of css.

1: Inline CSS

2: Internal CSS

3: External CSS

1: Inline CSS

* An inline style may be used to apply a unique style for a single element.

## Syntax:-

## <h1 style="color:blue;text-align:center;"></h1>

## 

## Example:-

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>

</body></html>

2: Internal CSS

* An internal style sheet may be used if one single HTML page has a unique style.

## Syntax:-

## <head>

## <style>

## H1{

## Color: red;

## }

## </style>

## </head>

3: External CSS

* the external style sheet file inside the <link> element, inside the head section.

<link rel=”stylesheet” href=”mystyle.css”>

## Example:-

<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

"mystyle.css"

body {

background-color: lightblue;

}

h1 {

color: navy;

margin-left: 20px;

}

CSS Comments:-

/\* This is a single-line comment \*/

/\* This is a multi-line comment \*/

**CSS TAG**

<style>

Color:red;

background-color: lightblue;

margin: 70px;( margin-top, margin-right, margin-bottom, margin-left)

border: 1px solid #4CAF50;

padding: 70px; ;( margin-top, margin-right, margin-bottom, margin-left)

height: 200px;

width: 50%;

Box(width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;)

## Example

<!DOCTYPE html>

<head>

<title>HTML and CSS Example</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

padding: 20px;

background-color: #f4f4f4;

}

h1 {

color: #333;

}

p {

line-height: 1.5;

color: #666;

}

ul {

list-style-type: none;

padding: 0;

}

li {

margin-bottom: 5px;

}

a {

text-decoration: none;

color: #007bff;

}

a:hover {

text-decoration: underline;

}

</style>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a sample HTML and CSS document that includes various tags and basic styling.</p>

<h2>HTML Tags</h2>

<p>Here are some common HTML tags:</p>

<ul>

<li><strong>&lt;h1&gt; - &lt;h6&gt;:</strong> Headings</li>

<li><strong>&lt;p&gt;:</strong> Paragraphs</li>

<li><strong>&lt;a&gt;:</strong> Links - <a href="#">Example Link</a></li>

<li><strong>&lt;ul&gt; and &lt;ol&gt;:</strong> Unordered and Ordered Lists

</body>

</html>

HTML and CSS templates

<html>

<head>

<style>

.html{

text-align: center;

color: white;

}

.color{

background-color: black;

}

.main{

background-color: chocolate;

}

.n12{

}

li{

display: inline;

font-size: 30px;

}

.searchsection{

text-align: right;

font-size: 20px;

color: white;

}

.main1{

text-align: center;

}

.haning1{

text-align: center;

font-size: 30px;

}

.text1{

text-align: center;

font-size: 20px;

}

.button1{

text-align: center;

}

.main2{

}

.img1{

float: left;

width: 33.33%;

text-align: center;

}

.img2{

float: left;

width: 33.33%;

text-align: center;

}

.img3{

float: left;

width: 33.33%;

text-align: center;

}

.main3{

}

.img4{

float: left;

width: 33.33%;

text-align: center;

background-color: green;

}

.img5{

float: left;

width: 33.33%;

text-align: center;

background-color: green;

}

.img6{

float: left;

width: 33.33%;

text-align: center;

background-color: green;

}

.s12{

text-align: center;

}

.main{

background-color: chocolate;

}

</style>

<title>taxt</title>

</head>

<body>

<div class="color">

<div class="html">

<h1>Html/cssTemplate</h1>

</div>

<div class="searchsection">

<input type="search" id="search" name="search">

<label for="search">search</label>

</div>

</div>

<div class="main">

<div class="n12">

<ul>

<li><a href="#home">home</a></li>

<li><a href="#about">about</a></li>

<li><a href="#servis">servis</a></li>

</ul>

</div>

<div class="main1">

<div class="haning1">

<h1>Dramatically Engage</h1>

</div>

<div class="text1">

<P>Objective innovate empowered manufatuer platforms.</P>

</div>

<div class="button1">

<button>Engage...</button>

</div>

</div>

</div>

<div class="main2">

<div class="haning1">

<h1>Superior collaboration</h1>

</div>

<div class="text1">

<P>Objective innovate empowered manufatuer platforms.</P>

</div>

<div class="img1">

<img src="d:\img\abcd.jpg.jpg">

<h1>Efficiently Unless</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

</div>

<div class="img2">

<img src="d:\img\abcd.jpg.jpg">

<h1>Completey synergize</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

</div>

<div class="img3">

<img src="d:\img\abcd.jpg.jpg">

<h1>Dynamically procrastinate</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

</div>

</div>

<div class="main3">

<div class="img4">

<img src="d:\img\images.jpg">

<h1>Efficiently Unless</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

<button>Unless...</button>

</div>

<div class="img5">

<img src="d:\img\images.jpg">

<h1>Completey synergize</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

<button>synergize...</button>

</div>

<div class="img6">

<img src="d:\img\images.jpg">

<h1>Dynamically procrastinate</h1>

<p>Through this connection, you can access and.</p>

<p>Objective innovate empowered manufatuer.</p>

<p>Sets or returns the default database name.</p>

<p>Sets or returns the isolation level.</p>

<button>procrastinate...</button>

</div>

</div>

<div>

<ul class="s12">

<li><a href="#Terms&conditons">Terms&conditons</a></li>

<li><a href="#privcy">privcy</a></li>

<li><a href="#policy">policy</a></li>

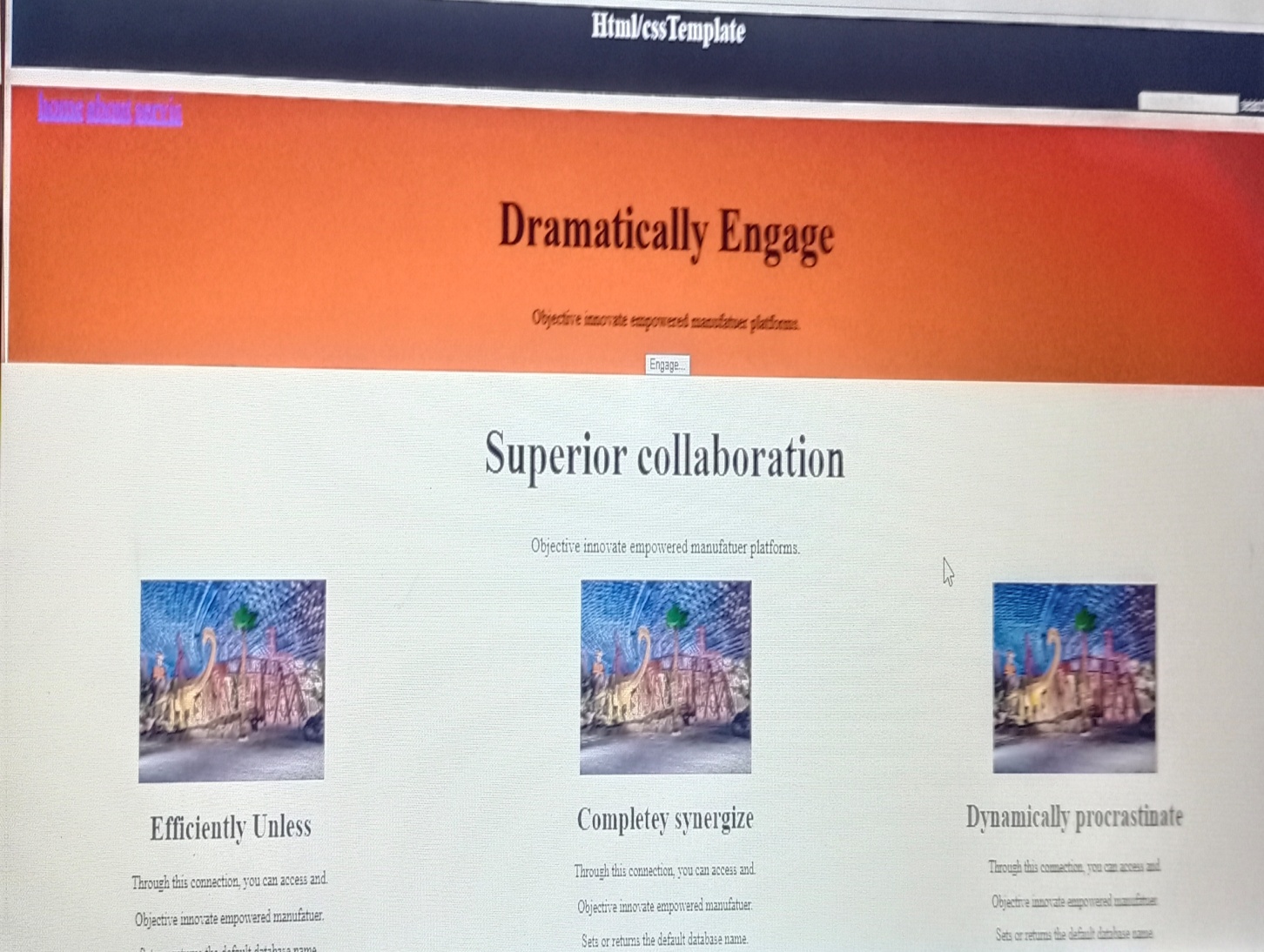
</ul>

</div>

</body>

</html>

OUTPUT:-



JavaScript

1.What is JavaScript?

* JavaScript is the world's most popular programming language.
* JavaScript is the programming language of the Web.
* JavaScript is easy to learn.

TAG

<script>

</script>

External JavaScript

An external script can be referenced in 3 different ways:

1. With a full URL (a full web address)
2. With a file path (like /js/)
3. Without any path

## Example

<!DOCTYPE html>

<html>

<body>

// Single Line Comments

<h2>Demo External JavaScript</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p>This example links to "myScript.js".</p>

<p>(myFunction is stored in "myScript.js")</p>

<script src="myScript.js"></script>

</body

</html>

**Variables**

* Variables are Containers for Storing Data
* JavaScript Variables can be declared in 4 ways:-

Variables type

1:Automatically

2: Using var

3: Using let

4:Using const

1: The var keyword was used in all JavaScript code from 1995 to 2015.

2: The let and const keywords were added to JavaScript in 2015.

3: The var keyword should only be used in code written for older browsers.

Using var

## Example:-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are variables.</p>

<p id="demo"></p>

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML =

"The value of z is: " + z;

</script></body></html>

Using let

* The value total can be changed.

## Example:-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are variables.</p>

<p id="demo"></p>

<script>

let x = 5;

let y = 6;

let z = x + y;

document.getElementById("demo").innerHTML =

"The value of z is: " + z;

</script></body></html>

Using const

1:Variables defined with const cannot be Redeclared

2:Variables defined with const cannot be Reassigned

3:Variables defined with const have Block Scope

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript const</h2>

<p id="demo"></p>

<script>

try {

const PI = 3.141592653589793;

PI = 3.14;

}

catch (err) {

document.getElementById("demo").innerHTML = err;}</script></body></html>

# {Objects}

## Example:-

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Objects</h2>

<p id="demo"></p>

<script>

// Create an object:

const car = {type:"Fiat", model:"500", color:"white"};

// Display some data from the object:

document.getElementById("demo").innerHTML = "The car type is " + car.type;

</script></body></htm>

OUTPUT:-

JavaScript ObjectsThe car type is Fiat

# Events

An HTML web page has finished loading

An HTML input field was changed

An HTML button was clicked

## Common HTML Events

**Event Description**

onchange An HTML element has been changed

onclick The user clicks an HTML element

onmouseover The user moves the mouse over an HTML element

onmouseout The user moves the mouse away from an HTML element

onkeydown The user pushes a keyboard key

onload The browser has finished loading the page

String Methods

JavaScript strings are for storing and manipulating text.

String length

String slice()

String substring()

String substr()

String replace()

String replaceAll()

String toUpperCase()

String toLowerCase()

String concat()

String trim()

String trimStart()

String trimEnd()

String padStart()

String padEnd()

String charAt()

String charCodeAt()

# [Arrays]

An array is a special variable, which can hold more than one value:

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Arrays</h1>

<p id="demo"></p>

<script>

const cars = ["Saab", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = cars;

</script>

</body>

</html>

OUTPUT:-

Saab,Volvo,BMW

# Date Objects

Date objects are static. The "clock" is not "running".

The computer clock is ticking, date objects are not.

There are generally 3 types of JavaScript date input formats:

**Type Example**

ISO Date "2015-03-25" (The International Standard)

Short Date "03/25/2015"

Long Date "Mar 25 2015" or "25 Mar 2015"

The ISO 8601 syntax (YYYY-MM-DD) is also the preferred JavaScript date format:

Short dates are written with an "MM/DD/YYYY" syntax like this

Long dates are most often written with a "MMM DD YYYY" syntax like this:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript ISO Dates</h2>

<p id="demo"></p>

<script>

const d = new Date("2015-03-25");

document.getElementById("demo").innerHTML = d;

</script>

</body>

</html>

OUTPUT:-

Wed Mar 25 2015 05:30:00

# if, else, and else if

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

## Example:-

<html>

   <head>

      <title>if</title>

   </head>

   <body>

      <div>

         <h1>if</h1>

      </div>

   </body>

   <script>

      var i=17;

      if(i>=18){

           document.write("yes");

      }else{

         document.write("not vote")

      }

   </script>

</html>

OUTPUT:- not vote

# For Loop

Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

## Example:-

<html>

   <head>

      <title>if</title>

   </head>

   <body>

      <div>

         <h1>if</h1>

      </div>

   </body>

   <script>

     for(var a= 0 ; a<=10; a++){

          document.write(a+"<br>")

     }

   </script>

</html>

OUTPUT:-

0 1 2 3 4 5 6 7 8 9 10

# While Loop

Loops can execute a block of code as long as a specified condition is true.

## Example:-

<html>

   <head>

      <title>if</title>

   </head>

   <body>

      <div>

         <h1>if</h1>

      </div>

   </body>

   <script>

      var a=0

       while(a<10){

             document.write(a+"<br>")

             a++;

       }

   </script>

</html>

OUTPUT:-

0 1 2 3 4 5 6 7 8 9

# Break and Continue

Break:-

<html>

   <head>

      <title></title>

   </head>

   <body>

      <script>

         for(var i=0; i<10; i++){

            if(i==5){

               break;

            }

            document.write(i+"<br>");

         }

      </script>

   </body>

</html>

OUTPUT:-

1  
2  
3  
4

Continue:-

<html>

   <head>

      <title></title>

   </head>

   <body>

      <script>

         for(var i=0; i<10; i++){

            if(i==5){

               continue;

            }

            document.write(i+"<br>");

         }

      </script>

   </body>

</html>

OUTPUT:-

1  
2  
3  
4  
6  
7  
8  
9

ODD:-

<html>

   <head>

      <title></title>

   </head>

   <body>

      <script>

         for(var i=0; i<10; i++){

            if(i%2!=0){

               continue;

            }

            document.write(i+"<br>");

         }

      </script>

   </body>

</html>

OUTPUT:-

2  
4  
6  
8

Html and css and javascriptv cdue from

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>CRUD with JavaScript</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      margin: 20px;

      background-color: bisque

    }

    table{

      border-collapse: collapse;

      width: 100%;

      margin-bottom: 20px;

    }

    th, td{

      border: 5px solid #ddd;

      padding: 10px;

      text-align: left;

      text-align: center;

    }

    label {

        display: block;

        margin-bottom: 5px;

        font-size: 30px;

        color: darkgray;

      }

    input {

      margin-bottom: 10px;

      padding: 5px;

      width: 100%;

      font-size: 20px;

      color: white;

      background-color: green;

      border-radius: 10px;

    }

    button {

      padding: 8px;

      margin-left: 50px;

      cursor: pointer;

      background-color: #4CAF50;

      color: white;

      border: none;

      border-radius: 4px;

      margin-left: 20px;

    }

    button:hover {

      background-color: #45a049;

    }

    button.delete {

      background-color: #f44336;

    }

    button.delete:hover {

      background-color: #d32f2f;

    }

    input:hover{

          background-color: cadetblue;

    }

  </style>

</head>

<body>

  <div>

    <label for="firstName">First Name:</label>

    <input type="text" id="firstName" placeholder="Enter first name">

  </div>

  <div>

    <label for="lastName">Last Name:</label>

    <input type="text" id="lastName" placeholder="Enter last name">

  </div>

  <div>

  <button class="edit" onclick="editOrSavePerson()">Edit</button>

  <button class="delete" onclick="deletePerson()">Delete</button>

  <button class="cancel" onclick="cancelEdit()" style="display:none;">Cancel</button>

  <table id="personTable">

    <thead>

      <tr>

        <th>First Name</th>

        <th>Last Name</th>

        <th>Action</th>

      </tr>

    </thead>

    <tbody>

      <!-- Persons will be displayed here -->

    </tbody>

  </table>

  <script>

    function editOrSavePerson() {

      let firstName = document.getElementById('firstName').value;

      let lastName = document.getElementById('lastName').value;

      if (!firstName || !lastName) {

        alert('Plese Enter first name and last name.');

        return;

      }

      let persons = JSON.parse(localStorage.getItem('persons')) || [];

      let editingIndex = localStorage.getItem('editingIndex');

      if (editingIndex !== null) {

        // Update existing person

        persons[editingIndex] = { firstName, lastName };

        localStorage.removeItem('editingIndex');

      } else {

        // Save new person

        persons.push({ firstName, lastName });

      }

      localStorage.setItem('persons', JSON.stringify(persons));

      displayPersons();

      clearPersonInputs();

      cancelEdit();

    }

    function displayPersons() {

      let persons = JSON.parse(localStorage.getItem('persons')) || [];

      let personTable = document.getElementById('personTable');

      let tbody = personTable.querySelector('tbody');

      // Clear previous rows

      tbody.innerHTML = '';

      persons.forEach((person, index) => {

        let row = tbody.insertRow();

        let cell1 = row.insertCell(0);

        let cell2 = row.insertCell(1);

        cell1.textContent = person.firstName;

        cell2.textContent = person.lastName;

        let editButton = document.createElement('button');

        editButton.textContent = 'Edit';

        editButton.className = 'edit';

        editButton.addEventListener('click', () => editPerson(index));

        row.appendChild(editButton);

        let deleteButton = document.createElement('button');

        deleteButton.textContent = 'Delete';

        deleteButton.className = 'delete';

        deleteButton.addEventListener('click', () => deletePerson(index));

        row.appendChild(deleteButton);

      });

    }

    function editPerson(index) {

      let persons = JSON.parse(localStorage.getItem('persons')) || [];

      // Enable the input fields for editing

      document.getElementById('firstName').value = persons[index].firstName;

      document.getElementById('lastName').value = persons[index].lastName;

      document.getElementById('firstName').readOnly = false;

      document.getElementById('lastName').readOnly = false;

      // Show the Save button and hide the Edit button

      document.querySelector('.edit').textContent = 'Save';

      document.querySelector('.edit').classList.remove('edit');

      document.querySelector('.edit').classList.add('save');

      // Save the index of the person being edited

      localStorage.setItem('editingIndex', index);

    }

    function deletePerson(index) {

      let persons = JSON.parse(localStorage.getItem('persons')) || [];

      persons.splice(index, 1);

      localStorage.setItem('persons', JSON.stringify(persons));

      displayPersons();

      clearPersonInputs();

      cancelEdit();

    }

    function cancelEdit() {

      clearPersonInputs();

      // Hide the Cancel button and show the Edit button

      document.querySelector('.cancel').style.display = 'none';

      document.querySelector('.edit').textContent = 'Edit';

      document.querySelector('.edit').classList.remove('save');

      document.querySelector('.edit').classList.add('edit');

      // Disable the input fields after clearing

      document.getElementById('firstName').readOnly = true;

      document.getElementById('lastName').readOnly = true;

    }

    function clearPersonInputs() {

      document.getElementById('firstName').value = '';

      document.getElementById('lastName').value = '';

    }

    // Initial display of persons

    displayPersons();

  </script>

</body>

</html>

MYSQL

1.What is MY Sql?

SQL is a standard language for storing, manipulating and retrieving data in databases.

SQL stands for Structured Query Language

SELECT \* FROM Customers;

# SQL SELECT

The following SQL statement returns all records from a table named "Customers":

# Syntax:-

SELECT \* FROM Customers;

SQL WHERE

The WHERE clause is used to filter records.

# Syntax:-

SELECT \* FROM Customers

WHERE Country='Mexico';

SQL ORDER BY

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

# Syntax:-

SELECT column1, column2, ...

FROM table\_name

ORDER BY column1, column2, ... ASC|DESC;

SQL INSERT INTO

The INSERT INTO statement is used to insert new records in a table.

# Syntax:-

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

SQL UPDATE

The UPDATE statement is used to modify the existing records in a table.

# Syntax:-

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

SQL DELETE

The DELETE statement is used to delete existing records in a table.

# Syntax:-

DELETE FROM table\_name WHERE condition;

CREATE DATABASE:-

CREATE DATABASE databasename;

DROP DATABASE:-

DROP DATABASE databasename;

SQL CREATE TABLE:-

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype,

....

);

ALTER TABLE:-

ALTER TABLE table\_name

ADD column\_name datatype;

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

# Syntax:-

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (ID)

);

SQL FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

# Syntax:-

CREATE TABLE Orders (

OrderID int NOT NULL,

OrderNumber int NOT NULL,

PersonID int,

PRIMARY KEY (OrderID),

FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

);

SQL Joins

A JOIN clause is used to combine rows from two or more tables.

Types of SQL JOINS:-

# INNER JOIN

# LEFT JOIN

# RIGHT JOIN

# FULL OUTER JOIN

INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables.

# Syntax:-

SELECT ProductID, ProductName, CategoryName

FROM Products

INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;



LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1),

# Syntax:-

SELECT column\_name(s)

FROM table1

LEFT JOIN table2

ON table1.column\_name = table2.column\_name;



RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2).

# Syntax:-

SELECT column\_name(s)

FROM table1

RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;



FULL JOIN

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

# Syntax:-

SELECT column\_name(s)

FROM table1

FULL OUTER JOIN table2

ON table1.column\_name = table2.column\_name

WHERE condition;



VUE.JS

version of Vue is v3.3.13.

1.What is Vue.Js?

* Vue is a JavaScript framework.
* Vue extends HTML attributes with Directives, and binds data to HTML with Expressions.

vue js installion

1:node js install

2:npm install -g @vue/cli(cmd)

3:vue create my-project(project create)

4:run npm run serve

vue js Directives

Different Vue Directives:-

1. v-bind
2. V-IF
3. v-show
4. v-for
5. v-on
6. v-model

Vue v-bind Directive

* The v-bind directive lets us bind an HTML attribute to data in the Vue instance.

Example

 <template>

    <h2 v-if="show">if conional</h2>

    <h2 v-else> not if conional</h2>

    <button v-on:click="show=!show">show</button>

    </template>

      <script>

    export default{

              name:"HomeComponet",

                    data()

                          {

                    return{

                  show :false

        }

    }

}

      </script>

         <style scoped>

h2{ color: aquamarine;}</style>

Vue v-If Directive

* Conditional rendering in Vue is done by using the v-if, v-else-if and v-else directives.

Example

<template>

  <div>

    <div v-if="condition1">

      Content for condition 1

    </div>

    <div v-else-if="condition2">

      Content for condition 2

</div>

    <div v-else>

      Default content if conditions are not met

    </div>

  </div>

</template>

<script>

export default {

  name:'HelloWorld',

  data(){

    return{

      condition1: false,  // Set your condition for the first case

      condition2: true  // Set your condition for the second case

    }

  }

}

</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->

<style>

.hello{

     font-size: 20px;

     border: 5px solid rgb(0, 83, 128);

     margin-top: 10px;

     margin-bottom: auto;

     margin-left: 5px;

     margin-right: 10px;

}

li{

        display: inline;

}

Vue v-For Directive

* List rendering in Vue is done by using the v-for directive, so that several HTML elements are created with a for-loop.

Example

<template>

    <h2 v-if="show">if conional</h2>

    <h2 v-else> not if conional</h2>

    <button v-on:click="show=!show">show</button>

    <ul>

        <li v-for="item in tecnology" :key="item">

            {{ item }}

        </li>

    </ul>

</template>

<script>

export default{

    name:"HomeComponet",

    data()

    {

        return{

           tecnology:["java","css","html","javascript"]

        }

    }

}

</script>

<style>

</style>

Vue v-On Directive

* Like event handling in plain JavaScript, the v-on directive in Vue tells the browser.

Example

<template>

    <h2 v-if="show">if conional</h2>

    <h2 v-else> not if conional</h2>

    <button v-on:click="show=!show">show</button>

    <ul>

        <li v-for="item in tecnology" :key="item">

            {{ item }}

        </li>

    </ul>

    <button v-on:click="getData('button1 on click')">click me</button>

    <button v-on:click="getData('button2 on click')">click me</button>

    </template>

      <script>

       export default{

               name:"HomeComponet",

       methods:{

              getData(data)

                     {

                   {

                    alert(data);

               }

             }

           }

         }

</script>

<style>

  </style>

Vue v-model Directive

* Two-way Binding.

Example

<template>

    <p>{{from}}</p>

<form>

    <label>Email :</label>

    <input  type="text" placeholder="enter email" v-model="from.email">

    <bR/><br/>

        <label>Password : </label>

        <input type="password" placeholder="enter password" v-model="from.password">

        <br/><br/>

        <button v-on:click="login" type="button">login</button>

</form>

</template>

<script>

export default{

    name:"HomeComponet",

    data(){

        return{

            from:{

                email:'',

                password:''

            }

        }

    },

    methods:{

        login(){

            console.warn("login data", this.from)

        }

    }

}

</script>

<style>

</style>

Vue Templates

All \*.vue files only consist of three parts:

<template> where the HTML content is.

<script> for our Vue code.

<style> where we write the CSS styling.

Vue Props

* Props is a configuration option in Vue.
* With props we can pass data to the components via custom attributes to the component tag.

Vue $emit()

* With the built-in $emit() method in Vue we can create a custom event in the child component that can be captured in the parent element.

<script>

export default {

props: ['foodName','foodDesc','isFavorite'],

emits: ['toggle-favorite'],

methods: {

toggleFavorite() {

this.$emit('toggle-favorite', this.foodName);

} }};</script>

Vue Scoped Styling

* Styling defined inside the <style> tag in a component, or in App.vue, is actually available globally in all components.

Example

<template>

    <h2 v-if="show">if conional</h2>

    <h2 v-else> not if conional</h2>

    <button v-on:click="show=!show">show</button>

    </template>

      <script>

    export default{

              name:"HomeComponet",

                    data()

                          {

                    return{

                  show :false

        }

    }

}

      </script>

         <style scoped>

h2{ color: aquamarine;}</style>

Vue Slots

* We use slots in Vue to send content from the parent into the <template> of a child component.
* So far we have just used components inside, <template> as self-closing tags like this.
* Before using v-slot and named slots, let's see,what happens if we use two slots in the component.

Python

python version:- 3.12.0

What is Python?

* Python is a popular programming language.

It is used for:-

web development (server-side),

software development,

mathematics,

system scripting.

# Syntax:-

Print(“Hello”)

#This is a comment

# Python Lists

* Lists are used to store multiple items in a single variable.
* Lists are created using square brackets:[]

Example

thislist = ["apple", "banana", "cherry", "apple", "cherry"]

print(thislist)

Additem=thislist.append(“orange”)

thislist.remove("banana")

# Python Tuples()

* Tuples are used to store multiple items in a single variable.
* A tuple is a collection which is ordered and unchangeable.
* Tuples are written with round brackets.

Example

thistuple = ("apple", "banana", "cherry")

print(thistuple)

# Python Sets {}

* Sets are used to store multiple items in a single variable.
* A set is a collection which is unordered, unchangeable\*, and unindexed.
* Set items are unordered, unchangeable, and do not allow duplicate values.

Example

thisset = {"apple", "banana", "cherry"}

print(thisset)

# Python Dictionary {}

* Dictionaries are used to store data values in key:value pairs.

Example

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

print(thisdict)

# Python If ... Else

* Python supports the usual logical conditions from mathematics.
* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

Example

a = 33

b = 200

if b > a:

print("b is greater than a")

# For Loops

* A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Example

1:- fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

2:-

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

break/continue

print(x)

# while Loops

Example

1:- i = 1

while i < 6:

print(i)

i += 1

2:-

i = 0

while i < 6:

i += 1

if i == 3:

continue

print(i)

# Functions

* A function is a block of code which only runs when it is called.

Example

def my\_function(fname):

print(fname + " Refsnes")

my\_function("Emil")

my\_function("Tobias")

my\_function("Linus")

# Arrays

* Arrays are used to store multiple values in one single variable:

Example

cars = ["Ford", "Volvo", "BMW"]

print(cars)

# Classes and Objects

* Python is an object oriented programming language.
* Almost everything in Python is an object, with its properties and methods.
* A Class is like an object constructor, or a "blueprint" for creating objects.

# Create a Class

class MyClass:

x = 5

print(MyClass)

# Try Except

* The try block lets you test a block of code for errors.
* The except block lets you handle the error.

Example

try:

print(x)

except:

print("An exception occurred")

File Handling

* File handling is an important part of any web application.
* Python has several functions for creating, reading, updating, and deleting files.

## Syntax

f = open("Fill name.")

print(f.read())

## 

## Delete a File;-

## import os

## os.remove("demofile.txt")

# File Write

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

## Syntax

f = open("demofile2.txt", "a")

f.write("Now the file has more content!")

f.close()

#open and read the file after the appending:

f = open("demofile2.txt", "r")

print(f.read())

# User Input

Python 3.6 uses the input() method.

Python 2.7 uses the raw\_input() method

Python 3.6

username = input("Enter username:")

print("Username is: " + username)

Python 2.7

username = raw\_input("Enter username:")

print("Username is: " + username)

progress report

[*https://www.w3profile.com/skpatel*](https://www.w3profile.com/skpatel)

<https://sk-patel.netlify.app/> (my website)