**Chapter 7: Tables**

* **Basic Table Structure**: Understand the <table>, <tr>, <th>, and <td> elements.
* **Styling Tables**: CSS properties like border, padding, text-align, and border-collapse.
* **Responsive Tables**: Techniques for making tables adapt to different screen sizes.
* **Terms to Review**: <caption>, <colgroup>, <col>, etc.

<table>

<caption>Student Grades</caption>

<tr><th>Name</th><th>Grade</th></tr>

<tr><td>Alice</td><td>A</td></tr>

<tr> <td>Bob</td><td>B</td></tr>

</table>

table {

width: 100%;

border-collapse: collapse;}

th, td {

padding: 8px;

text-align: left;

border: 1px solid #ddd;}

**Chapter 8: Forms**

* **Form Elements**: Key elements like <form>, <input>, <label>, <textarea>, <select>, and <button>.
* **Input Types**: Understand various types (text, password, email, number, date, etc.).
* **Form Validation**: Required fields, pattern matching, and validation types.
* **Terms to Review**: <fieldset>, <legend>, form submission methods (GET, POST).

<form action="/submit" method="post">

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

<input type="text" id="name" name="name" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="age">Age:</label>

<input type="number" id="age" name="age" min="18" max="100">

<button type="submit">Submit</button>

</form>

**Chapter 18: CSS Transitions and Animations**

* **Transitions**: How to create smooth transitions using transition property.
* **Animations**: Keyframe animations (@keyframes) and properties like animation-duration, animation-timing-function.
* **Practical Uses**: Examples of enhancing UI interactivity and visual appeal.
* **Terms to Review**: transform, opacity, scale, rotate.

button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

transition: background-color 0.3s ease;}

button:hover {background-color: #45a049;}

@keyframes slideIn {

from { transform: translateX(-100%); }

to { transform: translateX(0); }}

.div-slide-in {

animation: slideIn 0.5s ease-in;}

**UI vs. UX**

* **UI (User Interface)**: Design of the site elements, layout, colors, typography, and responsiveness.
* **UX (User Experience)**: User journey, accessibility, usability, and satisfaction in interactions.
* **Comparison**: UI focuses on visuals, UX on user experience and functionality.

<div class="header">Header</div>

<div class="navbar">Navigation Bar</div>

<div class="content">Main Content</div>

<div class="footer">Footer</div>

body {

display: flex;

flex-direction: column;

align-items: center;

font-family: Arial, sans-serif;}

.header, .navbar, .content, .footer {

width: 90%;

padding: 20px;

margin: 10px 0;

border: 1px solid #ddd;}

.header { background-color: #f1f1f1; }

.navbar { background-color: #ddd; }

.content { background-color: #fff; }

.footer { background-color: #f1f1f1; }

**Website Development Process**

* **1. Planning and Research**: Define goals, target audience, and project scope.
* **2. Design and Wireframing**: Creating layout designs and wireframes to outline structure and flow.
* **3. Development**: Coding using HTML, CSS, and JavaScript to build functionality.
* **4. Testing and Launch**: Debugging, testing compatibility, and launching.
* **5. Maintenance**: Ongoing updates, security patches, and improvements.

**Wireframe Basics**

* **Definition**: A skeletal framework for the website layout.
* **Types**: Low-fidelity (basic structure) and high-fidelity (detailed design with UI elements).
* **Tools and Techniques**: Use tools like Figma, Sketch, or basic sketching.
* **Purpose**: Helps in planning layout, navigation, and functionality without visual distractions.

**Streamlit Basics**

* **Introduction**: A Python framework for building web applications with a focus on data apps.
* **Key Components**:
  + **Text Elements**: st.title, st.header, st.write, etc.
  + **Widgets**: st.button, st.slider, st.selectbox for interactive controls.
  + **Media**: Display images, videos, and audio using st.image, st.video.
* **Running Streamlit Apps**: streamlit run your\_app.py.

import streamlit as st

st.title("Simple Streamlit App")

st.write("This is a simple app to demonstrate Streamlit basics.")

# Input widget

name = st.text\_input("Enter your name:")

if name:

st.write(f"Hello, {name}!")

# Slider widget

age = st.slider("Select your age:", 0, 100, 25)

st.write(f"You selected age: {age}")

import streamlit as st

st.header("Interactive Controls")

if st.button("Click Me"):

st.write("Button clicked!")

# Selectbox for choices

option = st.selectbox("Choose a number:", [1, 2, 3, 4, 5])

st.write(f"You selected: {option}")

**Python Web Development Basics**

* **Web Frameworks**: Introduction to Flask, Django, and Streamlit.
* **Routing and Views**: Basics of URL routing and connecting views (HTML pages) to Python functions.
* **Data Handling**: Sending and receiving data through forms, handling JSON data.
* **Terms to Review**: HTTP methods (GET, POST), JSON, server vs. client-side rendering.

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/submit', methods=['POST'])

def submit():

name = request.form['name']

return f"Hello, {name}!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Simple Flask Form</title>

</head>

<body>

<form action="/submit" method="post">

<label for="name">Enter your name:</label>

<input type="text" id="name" name="name" required>

<button type="submit">Submit</button>

</form>

</body>

</html>