1. HTML का परिचय (Introduction to HTML)

1.1 HTML क्या है? (What is HTML?)

HTML (HyperText Markup Language) एक markup language है, जिसका उपयोग वेबपेज बनाने के लिए किया जाता है। यह वेब ब्राउज़र को बताता है कि किसी पेज पर टेक्स्ट, इमेज, लिंक और अन्य एलिमेंट्स कैसे दिखने चाहिए।

उदाहरण (Basic HTML Structure)

```
<!DOCTYPE html>
<html>
<head>
   <title>My First HTML Page</title>
</head>
<body>
   <h1>Welcome to HTML</h1>
   <p>This is my first web page.</p>
</body>
</html>
```

यह एक साधारण HTML पेज का उदाहरण है, जो हेडिंग और पैराग्राफ दिखाएगा।

1.2 वेब डेवलपमेंट में HTML की भूमिका (Role of HTML in Web Development)

HTML वेब डेवलपमेंट का **बेसिक ब्लॉक** है। यह अन्य वेब टेक्नोलॉजीज जैसे **CSS और JavaScript** के साथ मिलकर काम करता है।

Technology	Role
HTML	पेज का स्ट्रक्चर बनाता है
CSS	पेज को स्टाइल और लेआउट देता है
JavaScript	पेज में इंटरेक्टिव फीचर्स जोड़ता है

1.3 HTML और अन्य वेब टेक्नोलॉजीज (Relation of HTML with Other Web Technologies)

1. **CSS (Cascading Style Sheets): HTML एलिमेंट्स को स्टाइलिंग देने के लिए उपयोग किया जाता है।**
2. **JavaScript: HTML पेज को डायनामिक और इंटरेक्टिव बनाने में मदद करता है।**
3. **Backend Technologies (PHP, Node.js, Python): HTML को डेटा प्रोसेसिंग और स्टोरेज के लिए इन तकनीकों के साथ जोड़ा जाता है।**

2. HTML की संरचना और बेसिक टैग्स (HTML Structure & Basic Tags)

2.1 HTML डॉक्यूमेंट का स्ट्रक्चर (HTML Document Structure)

HTML डॉक्यूमेंट का एक निश्चित स्ट्रक्चर होता है, जो ब्राउज़र को बताता है कि वेबपेज को कैसे रेंडर करना है।

Basic Structure Example:

```
<!DOCTYPE html>
<html>
<head>
   <title>My Web Page</title>
</head>
<body>
   <h1>Hello, World!</h1>
   <p>This is a sample webpage.</p>
</body>
</html>
```

मुख्य तत्व (Main Elements):

- `<!DOCTYPE html>`: HTML5 डॉक्यूमेंट को डिफाइन करता है।
- `<html>`: पूरा HTML डॉक्यूमेंट इसमें लिखा जाता है।
- `<head>`: मेटा इनफार्मेशन, टाइटल, और स्टाइल शीट्स शामिल करता है।
- `<body>`: मुख्य वेबपेज कंटेंट को होस्ट करता है।

2.2 HTML के मुख्य टैग्स (Important HTML Tags)

टैग	उद्देश्य
`<h1> - <h6>`	हेडिंग्स (Headings) को डिफाइन करता है
`<p>`	पैराग्राफ को परिभाषित करता है
` `	लाइन ब्रेक डालता है
`<hr>`	पेज में एक क्षैतिज रेखा जोड़ता है
`<strong>`	टेक्स्ट को बोल्ड बनाता है
`<em>`	टेक्स्ट को इटैलिक बनाता है
`<a>`	हाइपरलिंक जोड़ने के लिए
`<img>`	इमेज जोड़ने के लिए

Example:

```
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<a href="https://www.example.com">Visit Example</a>
```

```
<img src="image.jpg" alt="Sample Image">
```

2.3 DOCTYPE, `<html>`, `<head>`, `<body>` टैग्स की व्याख्या (Explanation of DOCTYPE, `<html>`, `<head>`, `<body>` Tags)

DOCTYPE टैग (DOCTYPE Tag)

- `<!DOCTYPE html>` HTML5 डॉक्यूमेंट को दर्शाने के लिए उपयोग किया जाता है।

HTML टैग (`<html>`)

- पूरे HTML डॉक्यूमेंट को `<html>` टैग में लिखा जाता है।

Head टैग (`<head>`)

- `<head>` सेक्शन में मेटाडेटा, CSS फाइल्स, और पेज टाइटल शामिल होता है।

Body टैग (`<body>`)

- `<body>` टैग के अंदर वे सभी कंटेंट होते हैं जो यूज़र को दिखते हैं।

3. टेक्स्ट फॉर्मेटिंग टैग्स (Text Formatting Tags)

3.1 टेक्स्ट स्टाइलिंग (Text Styling)

HTML में टेक्स्ट को फॉर्मेट करने के लिए कई टैग्स होते हैं:

टैग	उद्देश्य
`<b>`	टेक्स्ट को बोल्ड बनाता है
`<i>`	टेक्स्ट को इटैलिक बनाता है
`<u>`	टेक्स्ट को अंडरलाइन करता है
`<mark>`	टेक्स्ट को हाइलाइट करता है
`<small>`	टेक्स्ट को छोटा करता है
`<del>`	टेक्स्ट को स्ट्राइकथ्रू करता है
`<ins>`	टेक्स्ट को अंडरलाइन जोड़ता है
`<sub>`	टेक्स्ट को सबस्क्रिप्ट बनाता है
`<sup>`	टेक्स्ट को सुपरस्क्रिप्ट बनाता है

Example:

```
<p><b>Bold Text</b></p>
<p><i>Italic Text</i></p>
<p><u>Underlined Text</u></p>
<p><mark>Highlighted Text</mark></p>
<p><del>Deleted Text</del></p>
<p><ins>Inserted Text</ins></p>
<p>H<sub>2</sub>O (Subscript)</p>
<p>10<sup>2</sup> (Superscript)</p>
```

3.2 टेक्स्ट अलाइनमेंट और कलर (Text Alignment & Color)

HTML में टेक्स्ट को CSS का उपयोग करके स्टाइल किया जाता है।

Example:

```
<p style="text-align: center; color: blue;">This is a centered blue text.</p>
```

---

4. लिस्ट्स और लिंक (Lists & Links)

4.1 ऑर्डर्ड और अनऑर्डर्ड लिस्ट (Ordered & Unordered Lists)

HTML में लिस्ट दो प्रकार की होती हैं:

1. **Ordered List (`<ol>`) – जिसमें आइटम्स क्रमबद्ध (numbered) होते हैं।**
2. **Unordered List (`<ul>`) – जिसमें आइटम्स unordered (bulleted) होते हैं।**

Example:

```
<h3>Ordered List Example:</h3>
<ol>
   <li>Apple</li>
   <li>Banana</li>
   <li>Cherry</li>
</ol>

<h3>Unordered List Example:</h3>
<ul>
   <li>Red</li>
   <li>Green</li>
   <li>Blue</li>
</ul>
```

4.2 नेस्टेड लिस्ट (Nested Lists)

लिस्ट के अंदर एक और लिस्ट डालना Nested List कहलाता है।

Example:

```
<ul>
   <li>Fruits
      <ul>
         <li>Apple</li>
         <li>Banana</li>
      </ul>
   </li>
   <li>Vegetables
      <ul>
         <li>Carrot</li>
```

```
            <li>Spinach</li>
        </ul>
    </li>
</ul>
```

4.3 HTML लिंक्स (HTML Links)

लिंक जोड़ने के लिए `<a>` टैग का उपयोग किया जाता है।

Basic Link Example:

```
<a href="https://www.google.com">Visit Google</a>
```

Target Attribute का उपयोग:

```
<a href="https://www.example.com" target="_blank">Open in New Tab</a>
```

Anchor Link (Same Page Navigation):

```
<a href="#section2">Go to Section 2</a>
<h2 id="section2">This is Section 2</h2>
```

---

5. इमेजेज और मीडिया (Images & Media)

5.1 इमेज टैग का उपयोग (Using the `<img>` Tag)

वेबपेज पर इमेज जोड़ने के लिए `<img>` टैग का उपयोग किया जाता है।

Basic Image Example:

<img src="image.jpg" alt="Sample Image" width="300" height="200">

- `src`: इमेज का URL या फाइल पाथ सेट करता है।
- `alt`: इमेज का विवरण (SEO और Accessibility के लिए)।
- `width` और `height`: इमेज का साइज़ सेट करने के लिए।

5.2 HTML में ऑडियो जोड़ना (Adding Audio in HTML)

वेबपेज में ऑडियो फाइल जोड़ने के लिए `<audio>` टैग का उपयोग किया जाता है।

Basic Audio Example:

<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>

- `controls`: प्लेयर बटन दिखाने के लिए।
- `source`: ऑडियो फाइल का पाथ और टाइप सेट करने के लिए।

5.3 HTML में वीडियो जोड़ना (Adding Video in HTML)

वेबपेज में वीडियो दिखाने के लिए `<video>` टैग का उपयोग किया जाता है।

Basic Video Example:

<video width="400" controls>

<source src="video.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

- `width`: वीडियो की चौड़ाई सेट करने के लिए।
- `controls`: प्लेयर बटन दिखाने के लिए।
- `source`: वीडियो फाइल और उसका फॉर्मेट।

5.4 YouTube वीडियो Embed करना (Embedding YouTube Videos)

YouTube वीडियो को डायरेक्ट वेबपेज में जोड़ने के लिए `<iframe>` टैग का उपयोग किया जाता है।

YouTube Embed Example:

<iframe width="560" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0" allowfullscreen></iframe>

---

6. HTML टेबल्स (HTML Tables)

6.1 टेबल का परिचय (Introduction to Tables)

HTML में टेबल्स का उपयोग डेटा को स्ट्रक्चर और प्रेजेंटेशन देने के लिए किया जाता है।

Basic Table Structure:

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>City</th>
    </tr>
    <tr>
        <td>Rahul</td>
        <td>25</td>
        <td>Delhi</td>
    </tr>
    <tr>
        <td>Anjali</td>
        <td>22</td>
        <td>Mumbai</td>
    </tr>
</table>
```

यह एक साधारण टेबल है जिसमें नाम, उम्र और शहर की जानकारी दी गई है।

6.2 टेबल के मुख्य टैग्स (Important Table Tags)

टैग	उद्देश्य
`<table>`	टेबल को परिभाषित करता है
`<tr>`	एक रो (Row) जोड़ता है
`<th>`	टेबल हेडर जोड़ता है
`<td>`	टेबल डेटा सेल जोड़ता है
`border`	टेबल की बॉर्डर सेट करता है

6.3 कॉलस्पैन और रोस्पैन (Colspan & Rowspan)

कभी-कभी किसी सेल को मर्ज करना आवश्यक होता है, जिसके लिए `colspan` और `rowspan` का उपयोग किया जाता है।

Colspan Example:

```
<table border="1">
```

```
    <tr>
        <th colspan="2">Full Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Rahul</td>
        <td>Sharma</td>
        <td>25</td>
    </tr>
</table>
```

यहां पहली रो (row) में दो कॉलम्स को मर्ज किया गया है।

Rowspan Example:

```
<table border="1">
    <tr>
        <th>Name</th>
        <th rowspan="2">Age</th>
    </tr>
    <tr>
        <td>Rahul</td>
    </tr>
</table>
```

इसमें दूसरी रो (row) की दो सेल्स को मर्ज किया गया है।

6.4 टेबल को स्टाइल देना (Styling Tables)

HTML में CSS का उपयोग करके टेबल को अधिक आकर्षक बनाया जा सकता है।

Example:

```
<style>
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid black;
    padding: 10px;
    text-align: center;
}
th {
    background-color: #f2f2f2;
}
</style>
```

इस कोड से टेबल अधिक व्यवस्थित और आकर्षक दिखेगी।

7. HTML फॉर्म्स और इनपुट टैग्स (HTML Forms & Input Tags)

7.1 HTML फॉर्म का परिचय (Introduction to HTML Forms)

HTML फॉर्म का उपयोग **यूज़र इनपुट लेने** के लिए किया जाता है, जैसे कि **लॉगिन, रजिस्ट्रेशन, सर्च बॉक्स**, आदि।

Basic Form Structure:

```
<form action="submit.php" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <br>
    <input type="submit" value="Submit">
</form>
```

- `<form>`: फॉर्म को डिफाइन करता है।
- `action`: डेटा सबमिट करने वाली फ़ाइल का URL।
- `method`: डेटा भेजने का तरीका (`GET` या `POST`)।
- `<label>`: इनपुट फ़ील्ड के लिए लेबल जोड़ता है।

- `<input>`: यूज़र इनपुट लेता है।

7.2 HTML इनपुट टाइप्स (HTML Input Types)

HTML में विभिन्न प्रकार के **इनपुट टैग्स** होते हैं:

Input Type	Description
`text`	साधारण टेक्स्ट इनपुट
`password`	पासवर्ड इनपुट (●●●)
`email`	ईमेल इनपुट
`number`	केवल नंबर इनपुट के लिए
`date`	डेट सिलेक्टर
`file`	फाइल अपलोड इनपुट
`checkbox`	मल्टीपल ऑप्शन सिलेक्शन

`radio`	**सिंगल ऑप्शन सिलेक्शन**
`submit`	**फॉर्म सबमिट करने के लिए**
`reset`	**फॉर्म रीसेट करने के लिए**

Example:

```
<form>
    Name: <input type="text" name="name"><br>
    Password: <input type="password" name="password"><br>
    Email: <input type="email" name="email"><br>
    Date: <input type="date" name="dob"><br>
    <input type="submit" value="Submit">
</form>
```

7.3 रेडियो बटन और चेकबॉक्स (Radio Buttons & Checkboxes)

Radio Button Example:

```
<form>
    <p>Select your gender:</p>
```

```
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">Female</label>
</form>
```

Checkbox Example:

```
<form>
    <p>Select your hobbies:</p>
    <input type="checkbox" id="reading" name="hobby" value="reading">
    <label for="reading">Reading</label>
    <input type="checkbox" id="sports" name="hobby" value="sports">
    <label for="sports">Sports</label>
</form>
```

7.4 ड्रॉपडाउन मेनू (Dropdown Menu)

ड्रॉपडाउन लिस्ट बनाने के लिए `<select>` टैग का उपयोग किया जाता है।

Dropdown Example:

```
<form>
    <label for="city">Choose your city:</label>
    <select id="city" name="city">
        <option value="delhi">Delhi</option>
```

```
        <option value="mumbai">Mumbai</option>
        <option value="bangalore">Bangalore</option>
    </select>
</form>
```

7.5 फॉर्म वैलिडेशन (Form Validation)

फॉर्म वैलिडेशन से गलत डेटा को सबमिट होने से रोका जा सकता है।

Example (Required Field Validation):

```
<form>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br>
    <input type="submit" value="Submit">
</form>
```

- `required`: **इनपुट को भरना अनिवार्य बनाता है।**

8. HTML5 के नए फीचर्स (New Features in HTML5)

8.1 HTML5 का परिचय (Introduction to HTML5)

HTML5, HTML का सबसे नया संस्करण है, जिसमें कई **नए फीचर्स** जोड़े गए हैं, जिससे वेब डेवलपमेंट को आसान और प्रभावी बनाया जा सकता है।

HTML5 के मुख्य फीचर्स:

- **Semantic Elements** – `<header>`, `<article>`, `<section>` आदि
- **Multimedia Support** – `<audio>`, `<video>` टैग्स
- **New Input Types** – `email`, `date`, `range`, आदि
- **Canvas & SVG** - Graphics बनाने के लिए
- **Geolocation API** - लोकेशन ट्रैकिंग के लिए

8.2 सेमांटिक एलिमेंट्स (Semantic Elements)

HTML5 में Semantic Elements का उपयोग पेज को अधिक **स्ट्रक्चर्ड और SEO फ्रेंडली** बनाने के लिए किया जाता है।

Element	Usage
`<header>`	वेबपेज का हेडर सेक्शन
`<nav>`	नेविगेशन लिंक के लिए

`<section>`	पेज के सेक्शन को परिभाषित करता है
`<article>`	इंडिपेंडेंट कंटेंट के लिए
`<footer>`	पेज के नीचे का भाग

Example:

```
<header>
  <h1>Welcome to My Website</h1>
</header>
<nav>
  <a href="#home">Home</a>
  <a href="#about">About</a>
</nav>
<section>
  <article>
    <h2>HTML5 Features</h2>
    <p>HTML5 provides many new semantic elements.</p>
  </article>
</section>
<footer>
  <p>Copyright © 2024</p>
</footer>
```

8.3 ऑडियो और वीडियो (Audio & Video)

HTML5 में **मल्टीमीडिया सपोर्ट** को बेहतर बनाया गया है। अब हम `<audio>` और `<video>` टैग का उपयोग करके आसानी से मीडिया फाइल्स जोड़ सकते हैं।

Audio Example:

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

Video Example:

```
<video width="500" controls>
  <source src="video.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

8.4 नए इनपुट टाइप्स (New Input Types in HTML5)

HTML5 में कई नए फॉर्म इनपुट टाइप्स जोड़े गए हैं:

Input Type	Usage
`email`	ईमेल एड्रेस इनपुट करने के लिए
`url`	वेब एड्रेस इनपुट करने के लिए
`date`	डेट सिलेक्ट करने के लिए
`range`	स्लाइडर इनपुट
`number`	केवल नंबर इनपुट के लिए

Example:

```
<form>
```

```
    Email: <input type="email" name="email"><br>
    Birth Date: <input type="date" name="dob"><br>
    Volume: <input type="range" name="volume" min="0" max="100"><br>
    <input type="submit" value="Submit">
</form>
```

8.5 कैनवास और एसवीजी (Canvas & SVG)

HTML5 में Graphics और Animations के लिए `<canvas>` और `<svg>` का उपयोग किया जाता है।

Canvas Example:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000;"></canvas>
<script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "red";
    ctx.fillRect(20, 20, 150, 75);
</script>
```

---

9. CSS का बेसिक परिचय (Basic Introduction to CSS)

9.1 CSS क्या है? (What is CSS?)

CSS (Cascading Style Sheets) एक **स्टाइलिंग लैंग्वेज** है, जिसका उपयोग HTML एलिमेंट्स को **डिज़ाइन और लेआउट** देने के लिए किया जाता है।

CSS के फायदे:

- **HTML और डिज़ाइन को अलग करता है**
- **वेबपेज को आकर्षक बनाता है**
- **Responsive Design को आसान बनाता है**

9.2 CSS जोड़ने के तरीके (Ways to Add CSS)

CSS को HTML में जोड़ने के **तीन तरीके** होते हैं:

तरीका	विवरण
Inline CSS	`style` attribute का उपयोग
Internal CSS	`<style>` टैग का उपयोग (HTML के अंदर)
External CSS	`.css` फ़ाइल को `<link>` टैग से जोड़ना

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body { background-color: lightblue; }
    h1 { color: darkblue; }
  </style>
</head>
<body>
  <h1>Welcome to CSS</h1>
</body>
</html>
```

9.3 CSS Selectors (CSS सेलेक्टर्स)

CSS में Selectors का उपयोग HTML एलिमेंट्स को स्टाइल देने के लिए किया जाता है।

Selector	Description

`element`	**सभी टैग्स को सेलेक्ट करता है (e.g., `p {}`)**
`#id`	**किसी खास ID वाले एलिमेंट को स्टाइल करता है**
`.class`	**किसी खास Class वाले एलिमेंट को स्टाइल करता है**

Example:

```
p {
    color: red;
}
#header {
    background-color: yellow;
}
.text-bold {
    font-weight: bold;
}
```

9.4 CSS Properties (CSS प्रॉपर्टीज)

CSS में कई प्रॉपर्टीज होती हैं, जो वेबपेज को स्टाइल करने में मदद करती हैं।

Common CSS Properties:

Property	Usage
`color`	टेक्स्ट का रंग बदलने के लिए
`background-color`	बैकग्राउंड का रंग सेट करने के लिए
`font-size`	टेक्स्ट का साइज़ बदलने के लिए
`margin`	बाहरी स्पेस जोड़ने के लिए
`padding`	अंदरूनी स्पेस जोड़ने के लिए

Example:

```
body {
    background-color: #f0f0f0;
}
h1 {
    color: blue;
    font-size: 24px;
```

}

---

10. CSS का एडवांस परिचय (Advanced CSS)

10.1 Box Model (CSS बॉक्स मॉडल)

CSS में हर एलिमेंट एक **बॉक्स** के रूप में माना जाता है, जिसमें चार मुख्य हिस्से होते हैं:

- **Content:** असली टेक्स्ट या इमेज
- **Padding:** कंटेंट और बॉर्डर के बीच का स्पेस
- **Border:** बॉक्स की बॉर्डर
- **Margin:** बॉक्स के बाहर का स्पेस

Box Model Example:

css

Copy code

```
div {
    width: 200px;
    padding: 10px;
    border: 2px solid black;
```

```
    margin: 20px;
}
```

10.2 Flexbox Layout (CSS Flexbox)

Flexbox का उपयोग लेआउट को फ्लेक्सिबल और रिस्पॉन्सिव बनाने के लिए किया जाता है।

Flexbox Properties:

Property	Usage
`display: flex;`	Flexbox कंटेनर को एक्टिवेट करता है
`justify-content`	आइटम्स को एक्सिस पर अलाइन करता है
`align-items`	आइटम्स को वर्टिकल अलाइन करता है

Example:

css

Copy code

```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 200px;
    border: 1px solid black;
}
```

10.3 Grid Layout (CSS Grid)

CSS Grid का उपयोग कंप्लेक्स वेब लेआउट बनाने के लिए किया जाता है।

Grid Properties:

Property	Usage
`display: grid;`	**Grid सिस्टम को एक्टिव करता है**
`grid-template-columns`	**कॉलम्स की संख्या सेट करता है**
`gap`	**कॉलम्स और रो के बीच का स्पेस**

Example:

css

Copy code

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 10px;
}
```

10.4 Animation & Transitions (CSS एनिमेशन और ट्रांज़िशन)

CSS में animations और transitions का उपयोग वेबपेज को डायनामिक बनाने के लिए किया जाता है।

Transition Example:

css

Copy code

```
div {
    width: 100px;
    height: 100px;
```

```
    background-color: blue;
    transition: background-color 0.5s ease;
}
div:hover {
    background-color: red;
}
```

Animation Example:

css

Copy code

```
@keyframes example {
    from {background-color: red;}
    to {background-color: yellow;}
}
div {
    width: 100px;
    height: 100px;
    animation: example 2s infinite;
}
```

---

11. JavaScript का बेसिक परिचय (Basic Introduction to JavaScript)

11.1 JavaScript क्या है? (What is JavaScript?)

JavaScript एक scripting language है, जिसका उपयोग वेबपेज को **डायनामिक और इंटरैक्टिव** बनाने के लिए किया जाता है।

JavaScript के उपयोग:

- HTML एलिमेंट्स को बदलना (DOM Manipulation)
- यूज़र इंटरेक्शन हैंडल करना (Event Handling)
- फॉर्म वैलिडेशन
- एनिमेशन और इफेक्ट्स

11.2 JavaScript जोड़ने के तरीके (Ways to Add JavaScript)

JavaScript को HTML में जोड़ने के **तीन तरीके** होते हैं:

Method	Description

Inline JavaScript	`onclick` **और** `onmouseover` **इवेंट्स का उपयोग**
Internal JavaScript	`<script>` **टैग का उपयोग (HTML के अंदर)**
External JavaScript	`.js` **फ़ाइल को** `<script src="">` **से जोड़ना**

Example:

```
<!DOCTYPE html>
<html>
<head>
   <script>
      function showMessage() {
         alert("Hello, JavaScript!");
      }
   </script>
</head>
<body>
   <button onclick="showMessage()">Click Me</button>
</body>
</html>
```

11.3 JavaScript में वेरिएबल्स और डेटा टाइप्स (Variables & Data Types in JavaScript)

JavaScript में डेटा स्टोर करने के लिए var, let, और const का उपयोग किया जाता है।

Example:

```
var name = "Rahul"; // Global Variable
```

```
let age = 25; // Block Scope Variable
const country = "India"; // Constant Variable
```

Common Data Types:

Data Type	Example
String	"Hello World"
Number	100, 3.14
Boolean	true, false
Array	["Apple", "Banana"]
Object	`{name: "John", age: 30}`

11.4 JavaScript में फंक्शन्स (Functions in JavaScript)

JavaScript में फंक्शन्स का उपयोग कोड को रियूज़ेबल और मैनेजेबल बनाने के लिए किया जाता है।

Function Example:

```
function greet(name) {
    return "Hello, " + name + "!";
}
console.log(greet("Rahul"));
```

11.5 इवेंट्स और DOM मैनिपुलेशन (Events & DOM Manipulation)

JavaScript के माध्यम से हम HTML एलिमेंट्स को बदल सकते हैं।

Event Handling Example:

```
<button onclick="changeText()">Click Me</button>
<p id="demo">Old Text</p>

<script>
function changeText() {
    document.getElementById("demo").innerHTML = "New Text!";
}
</script>
```

---

12. JavaScript का एडवांस परिचय (Advanced JavaScript)

12.1 ECMAScript और ES6+ (ECMAScript and ES6+)

ECMAScript, JavaScript का **स्टैंडर्ड वर्जन** है। ES6+ में कई नए फीचर्स जोड़े गए हैं जो कोड को और अधिक प्रभावी बनाते हैं।

ES6 के मुख्य फीचर्स:

- `let` और `const` (Block Scope Variables)
- Arrow Functions (`=>`)
- Template Literals
- Destructuring Assignment
- Default Parameters

Example:

```
const greet = (name) => `Hello, ${name}!`;

console.log(greet("Rahul"));
```

12.2 Promises और Async/Await (Promises & Async/Await)

JavaScript में Asynchronous Programming को बेहतर बनाने के लिए Promises और `async/await` का उपयोग किया जाता है।

Promise Example:

```
let myPromise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("Data received!"), 2000);
});

myPromise.then(data => console.log(data));
```

Async/Await Example:

```
async function fetchData() {
    let response = await fetch("https://api.example.com/data");
    let data = await response.json();
    console.log(data);
}

fetchData();
```

12.3 JavaScript Modules (JS Modules)

JavaScript में कोड को अलग-अलग फाइलों में बांटने के लिए `import` और `export` का उपयोग किया जाता है।

Module Export Example:

```
// math.js
export function add(a, b) {
    return a + b;
}
```

Module Import Example:

```
// main.js
import { add } from "./math.js";
console.log(add(2, 3));
```

12.4 JavaScript में API कॉल करना (Fetching APIs in JavaScript)

वेब एप्लिकेशन में डेटा प्राप्त करने के लिए API का उपयोग किया जाता है।

Fetch API Example:

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
```

```
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.log("Error:", error));
```

12.5 JavaScript में लोकल स्टोरेज और सेशन स्टोरेज (Local Storage & Session Storage)

डाटा को ब्राउज़र में स्टोर करने के लिए `localStorage` और `sessionStorage` का उपयोग किया जाता है।

Local Storage Example:

```
localStorage.setItem("username", "Rahul");
console.log(localStorage.getItem("username"));
```

Session Storage Example:

```
sessionStorage.setItem("sessionData", "Active");
console.log(sessionStorage.getItem("sessionData"));
```

---

13. JavaScript Projects & Best Practices (JavaScript प्रोजेक्ट्स और बेहतरीन अभ्यास)

13.1 Basic JavaScript Projects (बेसिक JavaScript प्रोजेक्ट्स)

नीचे कुछ सिंपल JavaScript प्रोजेक्ट्स दिए गए हैं, जो आपकी प्रैक्टिस में मदद करेंगे।

1. डिजिटल घड़ी (Digital Clock):

```
<p id="clock"></p>

<script>

function updateClock() {

    document.getElementById("clock").innerHTML = new
Date().toLocaleTimeString();

}

setInterval(updateClock, 1000);

</script>
```

2. टोडो लिस्ट (To-Do List):

```
<input type="text" id="task" placeholder="Enter task">
```

```
<button onclick="addTask()">Add</button>
<ul id="taskList"></ul>
<script>
function addTask() {
    let task = document.getElementById("task").value;
    let li = document.createElement("li");
    li.innerText = task;
    document.getElementById("taskList").appendChild(li);
}
</script>
```

13.2 JavaScript Best Practices (JavaScript के बेहतरीन अभ्यास)

बेहतर और एरर-फ्री कोड लिखने के लिए निम्नलिखित बेस्ट प्रैक्टिसेस को अपनाएं:

1. `let` और `const` का उपयोग करें, `var` नहीं।

```
const name = "Rahul";
let age = 25;
```

2. Arrow Functions का उपयोग करें

```
const add = (a, b) => a + b;
console.log(add(2, 3));
```

3. Strict Mode का उपयोग करें

```
"use strict";
x = 10; // Error: x is not defined
```

4. Code Readability बढ़ाएं

- फ़ाइल और फ़ंक्शन के नाम अर्थपूर्ण रखें।
- Consistent Indentation का उपयोग करें।

5. Try-Catch का उपयोग करें (Error Handling)

```
try {
    let result = x / 0;
} catch (error) {
    console.log("Error Occurred: ", error);
}
```

---

14. HTML वेब डिप्लॉयमेंट (HTML Web Deployment)

14.1 HTML वेबसाइट को होस्ट कैसे करें? (How to Host an HTML Website?)

वेब डिप्लॉयमेंट का मतलब है कि आपकी HTML वेबसाइट को **इंटरनेट पर लाइव** करना। इसके लिए निम्नलिखित तरीके उपलब्ध हैं:

1. **GitHub Pages (फ्री होस्टिंग के लिए)**
2. **Netlify & Vercel (Static Websites के लिए)**
3. **Shared Hosting & cPanel (व्यवसायिक वेबसाइट के लिए)**
4. **Cloud Hosting (AWS, Firebase, DigitalOcean)**

14.2 GitHub Pages पर HTML वेबसाइट होस्ट करना (Hosting on GitHub Pages)

GitHub Pages एक **फ्री और आसान तरीका** है वेबसाइट को होस्ट करने का।

स्टेप्स:

1. GitHub पर एक नया रिपॉजिटरी बनाएं।
2. अपनी HTML, CSS, और JavaScript फ़ाइलें उसमें अपलोड करें।
3. `Settings` → `Pages` पर जाएं।
4. `Branch` से `main` चुनें और `Save` करें।
5. कुछ सेकंड्स में आपकी वेबसाइट लाइव होगी!

14.3 Netlify & Vercel पर HTML वेबसाइट होस्ट करना (Deploying on Netlify & Vercel)

Netlify और Vercel **स्टैटिक वेबसाइट होस्टिंग** के लिए बेहतरीन विकल्प हैं।

Netlify पर डिप्लॉय करने के स्टेप्स:

1. Netlify पर साइन अप करें।
2. GitHub से अपना प्रोजेक्ट कनेक्ट करें।
3. `Deploy` बटन पर क्लिक करें।
4. आपकी वेबसाइट कुछ ही सेकंड्स में लाइव होगी!

14.4 cPanel का उपयोग करके वेबसाइट अपलोड करना (Uploading Website Using cPanel)

यदि आपके पास **शेयर होस्टिंग** है, तो आप cPanel का उपयोग कर सकते हैं।

स्टेप्स:

1. cPanel में लॉग इन करें।
2. `File Manager` खोलें।
3. `public_html` फ़ोल्डर में अपनी HTML फाइल अपलोड करें।
4. आपकी वेबसाइट अब डोमेन से एक्सेस की जा सकती है।

14.5 Cloud Hosting (AWS, Firebase, DigitalOcean)

यदि आपकी वेबसाइट **एडवांस फीचर्स** वाली है, तो क्लाउड होस्टिंग का उपयोग करें।

Firebase पर होस्टिंग:

npm install -g firebase-tools

firebase login

firebase init

firebase deploy

Firebase आपकी HTML वेबसाइट को फास्ट और सिक्योर होस्टिंग प्रदान करता है।

---

15. SEO और HTML (SEO & HTML)

15.1 मेटा टैग्स (Meta Tags)

SEO (Search Engine Optimization) में HTML के `<meta>` टैग महत्वपूर्ण भूमिका निभाते हैं।

<meta name="description" content="यह एक SEO फ्रेंडली वेबपेज है">

<meta name="keywords" content="HTML, SEO, Meta Tags">

<meta name="author" content="Your Name">

15.2 Open Graph और Twitter Cards

सोशल मीडिया पर लिंक शेयरिंग को बेहतर बनाने के लिए Open Graph और Twitter Cards का उपयोग किया जाता है।

<meta property="og:title" content="My Webpage">

<meta property="og:description" content="यह मेरी वेबसाइट का विवरण है">

<meta property="og:image" content="image.jpg">

15.3 ऑन-पेज SEO

- **Heading Tags (`<h1>` - `<h6>`) का सही उपयोग करें।**
- **Alt Attributes का उपयोग इमेजेज के लिए करें।**
- **SEO फ्रेंडली URLs बनाएं।**

16. HTML & Accessibility (HTML और एक्सेसिबिलिटी)

16.1 ARIA (Accessible Rich Internet Applications)

ARIA टैग्स स्क्रीन रीडर यूजर्स को बेहतर अनुभव देते हैं।

<button aria-label="Close">X</button>

16.2 Semantic HTML

Semantic HTML, स्क्रीन रीडर्स और SEO के लिए बेहतर होता है।

```
<header>
  <h1>Welcome</h1>
</header>
<nav>
  <a href="#home">Home</a>
</nav>
```

---

17. Progressive Web Apps (PWA) और HTML

PWA ऑफलाइन सपोर्ट और फास्ट लोडिंग के लिए उपयोग होता है।

```
{
  "name": "My PWA",
  "short_name": "PWA",
  "start_url": "/index.html"
}
```

---

18. HTML और वेब सिक्योरिटी (Web Security in HTML)

18.1 XSS (Cross-Site Scripting) से बचाव

```
<input type="text" oninput="sanitizeInput(this.value)">
```

18.2 Content Security Policy (CSP)

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'">
```

---

19. HTML और Backend Integration

19.1 PHP या Node.js के साथ HTML

```
<form action="server.php" method="POST">
  <input type="text" name="username">
</form>
```

19.2 REST API और AJAX के साथ HTML

```
fetch('https://api.example.com/data')
  .then(response => response.json())
```

```
.then(data => console.log(data));
```

---

20. HTML और वेब परफॉर्मेंस (HTML & Web Performance)

20.1 Lazy Loading Images

```
<img src="image.jpg" loading="lazy" alt="Lazy Loaded Image">
```

20.2 Minification और Compression

- **HTML, CSS और JavaScript फाइलों को Minify करें।**
- **Gzip और Brotli Compression को उपयोग करें।**

---

21. HTML और माइक्रोफ्रंटेंड्स (HTML & Microfrontends)

Microfrontend आर्किटेक्चर का उपयोग बड़ी वेब एप्लिकेशन को छोटे, स्वतंत्र मॉड्यूल में बांटने के लिए किया जाता है।

```
<iframe src="https://microfrontend.com" width="600" height="400"></iframe>
```

---

22. HTML और वेब 3.0 (HTML & Web 3.0)

Web 3.0 नई तकनीकों को HTML में शामिल करने का तरीका है, जैसे:

- **Blockchain Integration**
- **Decentralized Applications (DApps)**
- **Smart Contracts**

<button onclick="connectToBlockchain()">Connect Wallet</button>

---

23. HTML और वर्चुअल रियलिटी (HTML & Virtual Reality)

23.1 WebVR और WebXR

WebVR और WebXR API का उपयोग करके वर्चुअल और ऑगमेंटेड रियलिटी एक्सपीरियंस बनाए जा सकते हैं।

<a-scene>

<a-box position="0 2 -5" color="red"></a-box>

</a-scene>

23.2 3D Objects और HTML

- **`<model-viewer>` टैग का उपयोग करके 3D मॉडल रेंडर करें।**

```
<model-viewer src="model.glb" auto-rotate camera-controls></model-viewer>
```

---

24. HTML और IoT (Internet of Things)

24.1 HTML के साथ IoT डैशबोर्ड बनाना

IoT डिवाइसेस से डेटा दिखाने के लिए HTML का उपयोग किया जा सकता है।

```
<div id="temperature">Loading...</div>

<script>

  fetch("https://api.iotdevice.com/temp")

    .then(response => response.json())

    .then(data => document.getElementById("temperature").innerText = data.temp + "°C");

</script>
```

24.2 WebSockets और IoT

WebSockets का उपयोग रियल-टाइम कम्युनिकेशन के लिए किया जाता है।

```
const socket = new WebSocket("wss://iotserver.com");

socket.onmessage = (event) => {

  console.log("New Data: ", event.data);

};
```

25. HTML और Machine Learning

25.1 TensorFlow.js के साथ HTML

TensorFlow.js का उपयोग करके वेब ब्राउज़र में Machine Learning Models चलाए जा सकते हैं।

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<script>
  async function predict() {
    const model = await tf.loadLayersModel('model.json');
    const output = model.predict(tf.tensor2d([5], [1, 1]));
    output.print();
  }
  predict();
</script>
```

25.2 Face Recognition और HTML

वेबकैम का उपयोग करके फेस डिटेक्शन किया जा सकता है।

```
<video id="webcam" autoplay></video>
<script>
```

```
  navigator.mediaDevices.getUserMedia({ video: true }).then(stream => {
    document.getElementById("webcam").srcObject = stream;
  });
</script>
```

---

26. HTML के महत्वपूर्ण टिप्स (Important Tips for HTML)

26.1 कोड को हमेशा वैलिडेट करें

HTML को W3C Validator (validator.w3.org) से चेक करें।

26.2 Semantic HTML का उपयोग करें

`<div>` और `<span>` के बजाय semantic tags (`<header>`, `<section>`, `<article>`, `<footer>`) का उपयोग करें।

26.3 वेब एक्सेसिबिलिटी (Web Accessibility) को प्राथमिकता दें

- `alt` एट्रिब्यूट का उपयोग करें।

- स्क्रीन रीडर्स के लिए ARIA टैग (`aria-label`, `aria-hidden`) जोड़ें।

26.4 वेबपेज की लोडिंग स्पीड ऑप्टिमाइज़ करें

- इमेजेज को compressed format में सेव करें।
- CSS और JavaScript को minify करें।
- Lazy Loading (`loading="lazy"`) का उपयोग करें।

26.5 Mobile-Friendly Design बनाएं

- Responsive Design के लिए CSS Flexbox और Grid का उपयोग करें।
- Viewport Meta Tag जोड़ें:

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

26.6 External CSS और JavaScript का उपयोग करें

Inline स्टाइल से बचें और CSS और JavaScript को अलग फ़ाइलों में रखें।

26.7 SEO के लिए HTML को ऑप्टिमाइज़ करें

- `<title>` और `<meta description>` का सही उपयोग करें।
- Heading Structure (`<h1>`, `<h2>`, `<h3>`) सही रखें।
- Internal Linking का सही उपयोग करें।

26.8 ब्राउज़र कंपैटिबिलिटी चेक करें

- HTML को Cross-Browser Compatible बनाने के लिए **Polyfills और Modernizr** का उपयोग करें।
- `<DOCTYPE html>` सही से डिफाइन करें।

26.9 वेब फॉर्म्स को सुरक्षित बनाएं

- **Client-side और Server-side Validation** दोनों करें।
- CSRF और XSS अटैक्स से बचने के लिए सुरक्षा उपाय लागू करें।

26.10 HTML5 के नए फीचर्स का उपयोग करें

- `<audio>`, `<video>`, `<canvas>` और `<svg>` टैग्स का सही उपयोग करें।

26.11 HTML कोड को साफ-सुथरा और संरचित रखें

- उचित **Indentation और Commenting** का उपयोग करें।

26.12 वेब स्टोरेज (LocalStorage & SessionStorage) का सही उपयोग करें

- डेटा स्टोर करने के लिए `localStorage` और `sessionStorage` का उपयोग करें।

26.13 वेब होस्टिंग से पहले सुरक्षा उपाय अपनाएं

- HTTPS का उपयोग करें।

- Security Headers को सही से सेट करें।

26.14 HTML और CSS को अलग रखें

- HTML में `style` टैग के बजाय External Stylesheets का उपयोग करें।

26.15 वेबपेज की स्पीड बढ़ाने के लिए CDN का उपयोग करें

- CSS, JavaScript और इमेजेज के लिए CDN का उपयोग करें।

26.16 HTML टेबल्स का सही उपयोग करें

- डेटा दिखाने के लिए `<table>` टैग का सही उपयोग करें, न कि वेब लेआउट के लिए।

26.17 सही फोंट और कलर स्कीम चुनें

- Google Fonts और Contrast Ratio का सही उपयोग करें।

26.18 वेब फॉर्म्स में Placeholder और Labels का सही उपयोग करें

- Placeholder को **Label के रिप्लेसमेंट** के रूप में उपयोग न करें।

26.19 Open Graph और Meta Tags का सही उपयोग करें

- **सोशल मीडिया शेयरिंग के लिए Open Graph (`og:title`, `og:image`) और Twitter Cards का उपयोग करें।**

26.20 Favicon और Touch Icons को सही से सेट करें

- **`<link rel="icon" type="image/png" href="favicon.png">` टैग का उपयोग करें**