IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING

Course Title - Web System Engineering

Topic Title - creating tables, lists, HTML forms

Presenter's Name - **M ArunKumar**
Presenter's ID - **IARE11151**
Department Name - **CSE (Artificial Intelligence, and Machine Learning)**
Lecture Number - 4
Presentation Date - 30/01/2025

Creating Table

- Creating a table in HTML is simple! Here's an example of how to structure it:

<table> creates the table.

<tr> defines a row.

<th> defines a table header cell (usually bold and centered by default).

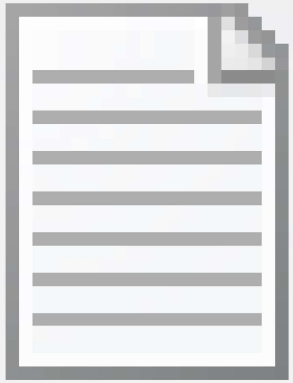
<td> defines a table data cell (regular cell).

table, tr , th , td tags & usage

```
<table border="1">   ..... </table>
 <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
 </tr>

<tr>
          <td>Row 1, Cell 1</td>
          <td>Row 1, Cell 2</td>
          <td>Row 1, Cell 3</td>
  </tr>
```

html code



CreatingTable.txt

Creating lists in HTML

There are two main types of lists:

unordered lists (bulleted lists) and

ordered lists (numbered lists)

```
<ul>
      <li>Apple</li>
      <li>Banana</li>
      <li>Orange</li>
      <li>Grapes</li>
 </ul>
```

Creating lists in HTML

There are two main types of lists: unordered lists (bulleted lists) and ordered lists (numbered lists).

unordered lists (bulleted lists):

```
<ul>  <li>Apple</li>
      <li>Banana</li>
      <li>Orange</li>
      <li>Grapes</li>
</ul>
```

- defines the unordered list.
- defines a list item.

Creating lists in HTML

Ordered List (Numbered List)

```
<ol>
<li>Boil water</li>
<li>Steep the tea leaves</li>
<li>Add milk and sugar (optional)</li>
<li>Serve and enjoy!</li>
</ol>
```

- defines the ordered (numbered) list.
- defines a list item (same as unordered, but it will be numbered in an ordered list).

Nested Lists

You can also create nested lists (a list inside another list)

```
<ul>
      <li>Groceries
        <ul>
          <li>Milk</li>
          <li>Bread</li>
          <li>Eggs</li>
</ul>   </li>
```

Creating lists in HTML

UnOrderLi.html

OrderLi.html

NestedLi.html

UnorderOrderedNestedLists.html

Creating a form in HTML

Creating a form in HTML is easy, and it allows you to gather user input. which can include different types of input fields like text fields, radio buttons, checkboxes, a submit button, etc.

Syntax : <form action="/submit" method="POST"> ... </form>

<form>: Defines the form. The action attribute specifies where the form data will be sent when submitted (in this case, to /submit), and method="POST" means the form will send data using the POST method (data sent as part of the request body).
<label>: Used to define labels for the input fields. This improves accessibility and makes forms easier to use.
<input>: Used for various types of user input.

example:
type="text" for a text input field.
type="email" for email input, which also validates that the input is in email format.
type="radio" for radio buttons (where the user can choose one option).
type="checkbox" for checkboxes (where the user can select multiple options).
type="submit" creates the submit button.
<textarea>: Allows users to enter a longer block of text (like a message).

Creating a form in HTML

```
<form action="/submit" method="POST">

<!-- Name Field -->
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required><br><br>

<!-- Email Field -->
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>

<!-- Gender Radio Buttons -->
    <label for="gender">Gender:</label><br>
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">Male</label><br>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">Female</label><br><br>
```

Creating a form in HTML

```
<!-- Hobbies Checkbox -->
    <label for="hobbies">Hobbies:</label><br>
    <input type="checkbox" id="reading" name="hobbies" value="Reading">
    <label for="reading">Reading</label><br>
    <input type="checkbox" id="sports" name="hobbies" value="Sports">
    <label for="sports">Sports</label><br><br>

<!-- Message Textarea -->
    <label for="message">Message:</label><br>
    <textarea id="message" name="message" rows="4" cols="50"></textarea><br><br>

<!-- Submit Button -->
    <input type="submit" value="Submit">
```

Creating a form in HTML

Additional Form Elements:
Select dropdown:

```
<label for="country">Country:</label>
<select id="country" name="country">
   <option value="usa">USA</option>
   <option value="uk">UK</option>
   <option value="india">India</option>
</select><br><br>
```

Styles and classes to your web pages

Inline Styles: Quick and simple, but not scalable.

Internal Styles: Best for small projects or one-page websites.

External Styles: Best for larger projects, as they allow for better organization and reusability.

Classes & IDs: Use to apply styles to multiple elements and target specific elements.

Layout Techniques (Flexbox/Grid): For more complex designs and layouts.

Styles and classes to your web pages

1. Inline Styles

You can add styles directly to an HTML element using the style attribute, but this method isn't recommended for larger projects because it makes your HTML messy and harder to maintain.

Example:

```
<p style="color: red; font-size: 20px;">
This is a red colored text with a larger font size.
</p>
```

InlineStyles.txt

InlineStyles.html

Styles and classes to your web pages

2. Internal Styles

You can place CSS within a <style> tag inside the <head> of your HTML document. This is good for small projects or one-page websites.

InternalStyle.txt

InternalStyles.html

Styles and classes to your web pages

3. External Styles

For larger websites, it's best to keep CSS in a separate file and link it to your HTML document. This keeps the HTML clean and allows you to reuse styles across multiple pages.

Syntax:

```
<link rel="stylesheet" href="styles.css">
```

ExternalStyles.txt

ExternalSyles.html

Styles and classes to your web pages

4. Using Classes

Classes allow you to apply the same styles to multiple elements.

For example, we used the .highlight class and .button class.

```
<style>
    .header-text {
        color: blue;
        font-size: 24px;
    }
    .footer-text {
        color: gray;
        font-size: 12px;
    }
</style>
```

```
<h1 class="header-text">This is the header</h1>
<p class="footer-text">This is footer text.</p>
```

StylesUsingClasses.html

Styles and classes to your web pages

5. CSS Layout Techniques

To build layouts, use Flexbox or CSS Grid for positioning elements:

```
<style>
    .container {
        display: flex;
        justify-content: space-between;
    }
    .box {
        width: 30%;
        background-color: lightblue;
        padding: 20px;
        text-align: center;
    }
</style>
```

```
<div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
</div>
```

Styles and classes to your web pages

Key Takeaways:

- **Inline Styles**: Quick and simple, but not scalable.
- **Internal Styles**: Best for small projects or one-page websites.
- **External Styles**: Best for larger projects, as they allow for better organization and reusability.

Layout Techniques (Flexbox/Grid): For more complex designs and layouts

Thank You


I A R E
EDUCATION FOR LIBERATION

Introduction to Responsive Web Design with CSS3 and HTML5

Responsive Web Design (RWD) is a design approach that ensures **websites look and function well on all devices**, from **desktops** to **tablets** and **smartphones**.

With the **increasing variety** of **screen sizes** and **resolutions**, **responsive** design has become a **standard practice in modern web development**.

•Key Principles(Core Concepts of Responsive Design):

◦Fluid Grids: Layouts are designed using relative units (like percentages) instead of fixed units (like pixels).
◦Flexible Images: Images scale to fit the size of their container.
◦Media Queries: CSS rules are applied conditionally based on the device's screen size, resolution, or orientation.

Why is Responsive Design Important?

- Improved User Experience - seamless experience across devices
- SEO Benefits - Google prioritizes mobile-friendly websites in search results
- Cost-Effective - Maintain a single website instead of separate ones for desktop and mobile
- Future-Proof - Adapts to new devices and screen sizes

Core Concepts of Responsive Design

- Fluid Grids

```
.container {
  width: 90%; /* Takes 90% of the parent container */
  margin: 0 auto; /* Centers the container */
}
.column {
  width: 48%; /* Two columns with a small gap */
  float: left;
  margin: 1%;
}
```

Core Concepts of Responsive Design

- Flexible Images - images scale with the container using max-width: 100%

```
img {
  max-width: 100%;
  height: auto; /* Maintains aspect ratio */
}
```

Core Concepts of Responsive Design

- Media Queries - Media queries allow you to apply CSS rules based on specific conditions, such as screen width, height, or orientation.