

Agenda

- HTML Tables
- HTML Forms
- Semantic HTML

In the last lecture, we have learned about the basics of HTML.

Now, In this lecture we are going to look into the advance topics of HTML, starting with HTML Tables.

HTML Tables:

Introduction:

We daily come across with several types of information that need to be displayed in the form of grid or tables, like train timetables, stock reports etc.

We can display information in form of tables on our web page using HTML Tables.

Let's first see what is actually a table is:

We can say that a table represents the information in a grid format.

we can look at the example shown below:

ID	Name	Weight	Price/lb	lbs. ordered	Total Price
1	Broccoli	1 lb	\$1.50	1	\$1.50
4	Asparagus	1 lb	\$1.00	2	\$2.00
7	Peas	1 lb	\$3.00	1	\$3.00
8	Spinach	1 lb	\$1.50	2	\$3.00
10	Carrots	1 lb	\$1.00	3	\$3.00

Let's now start with HTML Tables

Creating a HTML Table:

The `<table>` element is used to create a table.

The contents of the table are written out row by row.

```
<table>
  <!-- Your table -->
</table>
```



Table Rows:

As we have seen above that the contents of a table are written out row by row.

So, You indicate the start of each row using the opening `<tr>` tag.

At the end of the row you use a closing `</tr>` tag.

We add data in the row between these tags.

Table Data:

In order to define data properly in a table, cells must also be defined along the rows.

In HTML tables, cells can be defined using `<td>`

tags and in each row we have created three cells or columns using

The table heading element is used just like a table data element, except with a relevant title.

Just like table data, a table heading must be placed within a table row.

Scope attribute in Table headings:

Scope attribute can take two values:

- 1. **row** - this value makes it clear that the heading is for a row.
- 2. **col** - this value makes it clear that the heading is for a column.



```
<table>
  <tr>
    <th></th>
    <th scope="col">Saturday</th> <th scope="col">Sunday</th>
  </tr>
  <tr>
    <th scope="row">Tickets sold:</th> <td>120</td>
    <td>135</td>
  </tr>
  <tr>
    <th scope="row">Total sales:</th> <td>$600</td>
    <td>$675</td>
  </tr>
</table>
```

Explanation:

In the above code, we have created a table with both row and column headings.

we also have used a empty tag in the first row of the table as, Even if a cell has no content, you should still use a element, it only contains three elements even though there are four columns in the result below. This is because the movie in the element above it uses the **rowspan** attribute to stretch down and take over the cell below.

Output of the code:

	ABC	BBC	CNN
6pm - 7pm	Movie	Comedy	News
7pm - 8pm		Sport	Current Affairs

Long Tables:

Over time, a table can grow to contain a lot of data and become very long. When this happens, the table can be sectioned off so that it is easier manage.

Long tables can be sectioned off using the *table body* element: **<tbody>** .

Table Head:

The headings of the table should sit inside the `<thead>` element. When a table's body is sectioned off, however, it also makes sense to section off the table's column headings using the

```
<thead>
```



element.

Table Body:

The body should sit inside the `<tbody>` element. The

```
<tbody>
```



element should contain all of the table's data, excluding the table headings.

Table Footer:

The footer belongs inside the `<tfoot>` element. The bottom part of a long table can also be sectioned off using the

```
<tfoot>
```



element.

```
<table>
  <thead>
    <tr>
      <th>Date</th>
      <th>Income</th>
      <th>Expenditure</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>1st January</th>
      <td>250</td>
      <td>36</td>
    </tr>
    <tr>
      <th>2nd January</th>
      <td>285</td>
      <td>48</td>
    </tr>
    <!-- additional rows as above -->
    <tr>
      <th>31st January</th>
      <td>129</td>
      <td>64</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td></td>
      <td>7824</td>
      <td>1241</td>
    </tr>
  </tfoot>
</table>
```



Output:

DATE	INCOME	EXPENDITURE
1st January	250	36
2nd January	285	48
31st January	129	64
	7824	1241

HTML Forms:

Introduction:

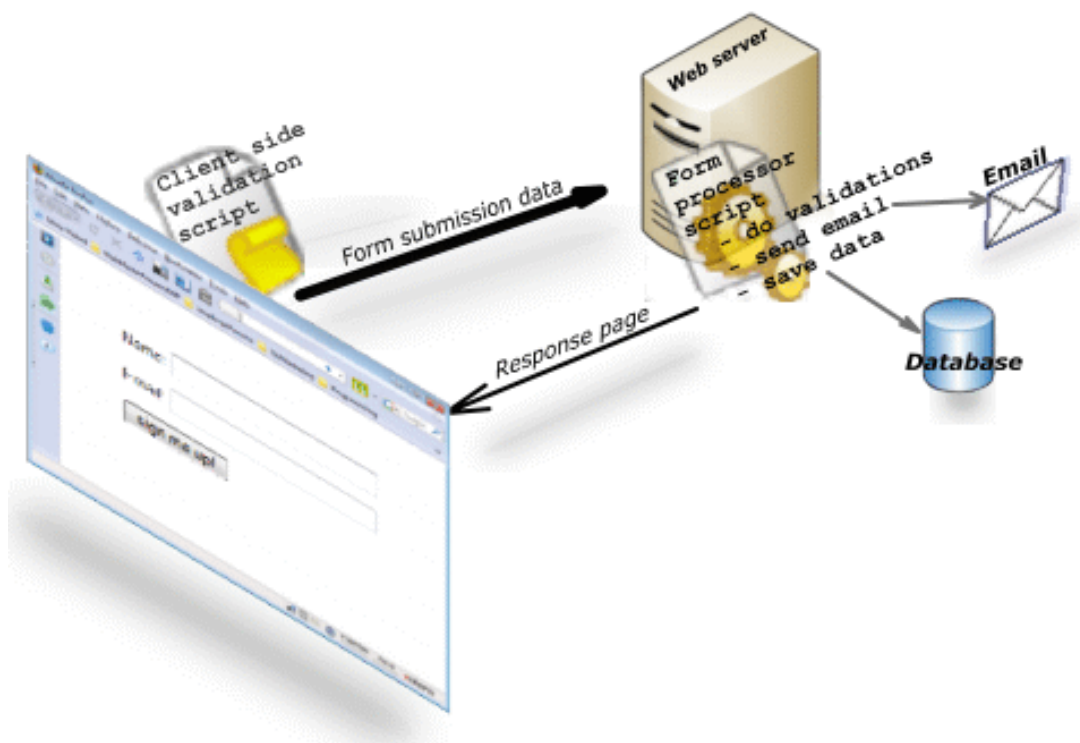
Forms are a part of everyday life. When we use a physical form in real life, we write down information and give it to someone to process.

Just like a physical form, an HTML Forms element is responsible for collecting information to send somewhere else.

Whether you are adding a simple search box to your website or you need to create more complicated insurance applications, HTML forms give you a set of elements to collect data from your users.

Working of HTML Forms:

A web form has two parts: the HTML 'front end' and a back end form processor. The HTML front end part handles the presentation while the back end handles the form submissions (like saving the form submissions, sending emails etc).



Working of a HTML Form can be stated as:

- A visitor visits a web page that contains a form.
- The web browser displays the HTML form.
- The visitor fills in the form and submits
- The browser sends the submitted form data to the web server
- A form processor script running on the web server processes the form data
- A response page is sent back to the browser.

Let's start learning to create HTML Forms:

Creating a Form:

The `<form>` element is a great tool for collecting information, but then we need to send that information somewhere else for processing. We need supply the `<form>` element with both the location of where the `<form>` 's information goes and what HTTP request to make.

As we need to make a HTTP request in order to send data to the proper server and the information regarding the HTTP request is taken care by sever attribute that this form element can have, which are as:

action:

Every

element requires an **action** attribute. Its **value** is the URL for the page on the server that will receive the information in the form when it is submitted.

method:

Forms can be sent using one of two methods: **get** or **post**.

With the **get** method, the values from the form are added to the end of the URL specified in the **action** attribute.

With the **post** method the values are sent in what are known as HTTP headers. If the method attribute is not used, the form data will be sent usir the **get** method.

```
<form action="http://www.example.com" method="get">
  <p>This is where the form controls will appear.</p>
</form>
```

Explanation:

In the above example, we have created a simple form in which we can add our form controls and which will send a get request to the url specified in tr action attribute.

Now let's learn about the Form controls that we can add in the HTML Form:

Text Input:

If we want to create an input field in our `<form>` , we'll need the help of the `<input>` element.

The `<input>` element has a **type** attribute which determines how it renders on the web page and what kind of data it can accept.

The first value for the **type** a ttribute we're going to explore is **"text"** . When we create an `<input>` element with **type="text"** , it renders a te field that users can type into.

When users enter information into a form, the server needs to know which form control each piece of data was entered into. Therefore, each form contr requires a **name** ****attribute. The value of this attribute identifies the form control and is sent along with the information they enter to the server.

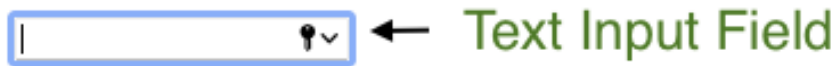
```
<form>
  <h1>Creating First Text Field!</h1>
  <input type="text" name="username" id="username">
</form>
```

Explanation:

In the above example, we are creating a text input field with name as username.

Output:

Creating First Text Field!



Adding a Label:

The `<label>` element has an opening and closing tag and displays text that is written between the opening and closing tags. To associate a `<label>` and an `<input>`, the `<input>` needs an `id` attribute. We then assign the `for` attribute of the `<label>` element with the value of the `id` attribute of `<input>`. As shown in the code below:

```
<form action="/example.html" method="POST">
  <label for="username">Write a Username:</label>
  <br>
  <input type="text" name="food" id="username">
</form>
```

Output:



Password Input:

An `<input type="password">` element will replace input text with another character like an asterisk (*) or a dot (•). Even though the password field obscures the text of the password, when the form is submitted, the value of the text is sent.

```
<form>
  <label for="user-password">Password: </label>
  <input type="password" id="user-password" name="user-password">
</form>
```

Output:



Textarea Element:

There are cases where users need to write in more information, like a blog post. In such cases, instead of using an `<input>` , we can use `<textarea>` .

The `<textarea>` element is used to create a bigger text field for users to write more text. We can add the attributes `rows` and `cols` to determine the amount of rows and columns for the `<textarea>` .



```
<form>
  <label for="blog">New Blog Post: </label>
  <br>
  <textarea id="blog" name="blog" rows="5" cols="30">
</textarea>
</form>
```

Output:

New Blog Post:

Textarea field which can be resized

Radio Button:

Radio button in the HTML form can be created using

element, with type = "radio".

type="radio"

Radio buttons allow users to pick just one of a number of options.

Attributes:

name:

The **name** attribute is sent to the server with the value of the option the user selects. **value:**

The **value** attribute indicates the value that is sent to the server for the selected option.

checked: The **checked** attribute can be used to indicate which value (if any) should be selected when the page loads. The value of this attribute is **checked**. Only one radio button in a group should use this attribute.



```
<form action="http://www.example.com/profile.php">
  <p>Please select your favorite genre:
  <br />
  <input type="radio" name="genre" value="rock"
    checked="checked" /> Rock
  <input type="radio" name="genre" value="pop" />
    Pop
  <input type="radio" name="genre" value="jazz" />
    Jazz
  </p>
</form>
```

Output:

Please select your favorite genre:

☐ Rock ☐ Pop ☒ Jazz

Checkbox Input:

Checkbox input in the HTML Form can be created using `<input type="checkbox">` element with `type="checkbox"`.

type="checkbox"

Checkboxes allow users to select (and unselect) one or more options

Attributes:

name:

The **name** attribute is sent to the server with the value of the option(s) the user selects.

value:

The **value** attribute indicates the value sent to the server if this checkbox is checked.

checked:

The **checked** attribute indicates that this box should be checked when the page loads. If used, its value should be **checked**.

```
<form action="http://www.example.com">
  <p>Please select your favorite music service(s):
  <br />
  <input type="checkbox" name="service"
    value="itunes" checked="checked" /> iTunes
  <input type="checkbox" name="service"
    value="lastfm" /> Last.fm
  <input type="checkbox" name="service"
    value="spotify" /> Spotify
</p>
</form>
```

Output:

Please select your favorite music service(s):

☒ iTunes ☐ Last.fm ☐ Spotify

Dropdown List:

▼ element is used to create a drop down list box. It contains two or more elements.

The **name** attribute indicates the name of the form control being sent to the server, along with the value the user selected.

The element is used to specify the options that the user can select from. The words between the opening and closing tags will be shown to the user in the drop down box. The element uses the **value** attribute to indicate the value that is sent to the server along with the name of the control if this option is selected. The **selected** attribute can be used to indicate the option that should be selected when the page loads. The value of this attribute should be **selected**.

```
<form action="http://www.example.com">
  <p>What type of OS you use?</p>
```



```

<select name="systems">
  <option value="macos">macos</option>
  <option value="windows">windows</option>
  <option value="linux">linux</option>
</select>
</form>

```

Output:

What type of OS you use?

Submit Button:

Submit button is used to send the form to the server.

Submit button can be created in HTML Forms by using

element with type="submit".

It can use a **name** attribute but it does not need to have one.

The **value** attribute is used to control the text that appears on a button. It is a good idea to specify the words you want to appear on a button because the default value of buttons on some browsers is 'Submit query' and this might not be appropriate for all kinds of form.

```

<form action="http://www.example.com">
  <p>Subscribe to us with your username:</p>
  <input type="text" name="username" />
  <input type="submit" name="subscribe" value="Subscribe" />
</form>

```



Output:

Subscribe to us with your username:

Submit button

username text field

Let's now start with last topic in our HTML module i.e., semantic HTML.

Semantic HTML:

When building web pages, we use a combination of non-semantic HTML and *Semantic HTML*. The word semantic means "relating to meaning," and semantic elements provide information about the content between the opening and closing tags.

By using Semantic HTML, we select HTML elements based on their meaning, not on how they are presented. Elements such as `<div>` and `` are not semantic elements since they provide no context as to what is inside of those tags.

For example, instead of using a `<div>` element to contain our header information, we could use a `<header>` element, which is used as a header section. By using a `<header>` tag instead of a `<div>`, we provide context as to what information is inside of the opening and closing tag.

Now, let's see some of the semantic elements that assist in structuring the web page:

Header Element:

A `<header>` is a container usually for either navigational links or introductory content containing `<h1>` to `<h6>` headings.



```
<header>
  <h1>
    Starting out with HTML semantic tags!
  </h1>
</header>
```

Explanation:

By using a `<header>` tag, our code becomes easier to read. It is much easier to identify what is inside of the `<h1>` 's parent tags, as opposed a `<div>` tag which would provide no details as to what was inside of the tag.

Nav Element:

A `<nav>` is used to define a block of navigation links such as menus and tables of contents. It is important to note that `<nav>` can be used inside the `<header>` element but can also be used on its own.



```
<header>
  <nav>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
    </ul>
  </nav>
</header>
```

Explanation:

In the above code, we have used a `<nav>` element inside of the `<header>` element.

By using `<nav>` as a way to label our navigation links, it will be easier for not only us, but also for web browsers and screen readers to read the code.

Main Element:

The element `<main>` is used to encapsulate the dominant content within a webpage.

By using `<main>` as opposed to a `<div>` element, screen readers and web browsers are better able to identify that whatever is inside of the tag is the bulk of the content.



```
<main>
  <header>
    <h1>Operating Systems</h1>
  </header>
  <article>
    <h3>macOs</h3>
    <p>
      MacOS is the operating system of Macbooks.
    </p>
  </article>
</main>
```

Section Element:

`<section>` defines elements in a document, such as chapters, headings, or any other area of the document with the same theme.



```
<section>
  <h2>Facts about Operating systems.</h2>
</section>
```

Article Element:

The `<article>` element holds content that makes sense on its own. `<article>` can hold content such as articles, blogs, comments, magazine etc.



```
<section>
  <h2>Facts about Operating systems</h2>
  <article>
    <p>Macos is used in Macbooks developed by Apple.</p>
  </article>
</section>
```

In the code above, the `<article>` element containing a fact about operating system was placed inside of the `<section>` element. It is important note that a `<section>` element could also be placed in an `<article>` element depending on the context.

Aside Element:

The `<aside>` element is used to mark additional information that can enhance another element but isn't required in order to understand the main content. This element can be used alongside other elements such as `<article>` or `<section>`.



```
<aside>
  <section>
    <h2>Popular Recipes</h2>
    <a href="">Yakitori (grilled chicken)</a>
    <a href="">Tsukune (minced chicken patties)</a>
    <a href="">Okonomiyaki (savory pancakes)</a>
    <a href="">Mizutaki (chicken stew)</a>
  </section>
</aside>
```

Figure Element:

`<figure>` is an element used to encapsulate media such as an image, illustration, diagram, code snippet, etc, which is referenced in the main flow of the document.



```
<figure>
  
</figure>
```

In the above code, we created a `<figure>` element so that we can encapsulate our `` tag. In `<figure>` we used the `` tag to insert an image onto the webpage.

Figcaption Element:

`<figcaption>` is an element used to describe the media in the `<figure>` tag. Usually, `<figcaption>` will go inside `<figure>`.



```
<figure>
  
```

```
<figcaption>This picture shows a macbook.</figcaption>
</figure>
```

In the example above, we added a `<figcaption>` into the `<figure>` element to describe the image. This helps group the `<figure>` content with the `<figcaption>` content.

Footer Element:

The content at the bottom of the subject information is known as the *footer*, indicated by the `<footer>` element. The `<footer>` element contains copyright information, along with links to the privacy policy and terms and conditions.

```
<footer>
  &copy; 2011 Yoko's Kitchen
</footer>
```

Now, let's take a look at a bigger example of table in HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <link
      href="https://fonts.googleapis.com/css?family=Lato: 100,300,400,700|Luckiest+Guy|Oxygen:300,400"
      rel="stylesheet"
    />
    <link href="style.css" type="text/css" rel="stylesheet" />
  </head>
  <body>

    <table>
      <thead>
        <tr>
          <th></th>
          <th scope="col">Home starter hosting</th>
          <th scope="col">Premium business hosting</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <th scope="row">Disk space</th>
          <td>250mb</td>
          <td>1gb</td>
        </tr>
        <tr>
          <th scope="row">Bandwidth</th>
          <td>5gb per month</td>
          <td>50gb per month</td>
        </tr>
        <!-- more rows like the two above here -->
      </tbody>
      <tfoot>
        <tr>
          <td></td>
          <td colspan="2">Sign up now and save 10%!</td>
        </tr>
      </tfoot>
    </table>

  </body>
</html>
```

Output:

	HOME STARTER HOSTING	PREMIUM BUSINESS HOSTING
Disk space	250mb	1gb
Bandwidth	5gb per month	50gb per month
Sign up now and save 10%!		

Now let's take a look at a bigger example of HTML Forms:



```
<!DOCTYPE html>
<html>
  <head>
    <title>Forms</title>
  </head>
  <body>
    <form action="http://www.example.com" method="get">
      <fieldset>
        <legend>Your Details:</legend>
        <label>
          Name:
          <input type="text" name="name" size="30" maxlength="100" />
        </label>
        <br />
        <label>
          Email:
          <input type="email" name="email" size="30" maxlength="100" />
        </label>
        <br />
      </fieldset>
      <br />
      <fieldset>
        <legend>Your Review:</legend>
        <p>
          <label for="hear-about"> How did you hear about us? </label>
          <select name="referrer" id="hear-about">
            <option value="google">Google</option>
            <option value="friend">Friend</option>
            <option value="advert">Advert</option>
            <option value="other">Other</option>
          </select>
        </p>
        <p>
          Would you visit again?
          <br />
          <label>
            <input type="radio" name="rating" value="yes" />
            Yes
          </label>
          <label>
            <input type="radio" name="rating" value="no" />
            No
          </label>
        </p>
      </fieldset>
    </form>
  </body>
</html>
```

```

        </label>
        <label>
            <input type="radio" name="rating" value="maybe" />
Maybe
        </label>
    </p>
    <p>
        <label for="comments">
Comments:
        </label>
        <br />
        <textarea rows="4" cols="40" id="comments">
        </textarea>
    </p>
    <label>
        <input type="checkbox" name="subscribe" checked="checked" />
        Sign me up for email updates
    </label>
    <br />
    <input type="submit" value="Submit review" />
</fieldset>
</form>
</body>
</html>

```

Output:

Your Details:

Name:
Email:

Your Review:

How did you hear about us?
Google
Would you visit again?
☐ Yes ☐ No ☐ Maybe
Comments:

☒ Sign me up for email updates
Submit review

Now, let's take a look at a bigger example in which we are using most of our semantic html elements to structure our web page:

```

!DOCTYPE HTML
<html>
<body>
    <header>
        <h1>Yoko's Kitchen</h1>
        <nav>
            <ul>
                <li><a href="">home</a></li>
                <li><a href="">catering</a></li>
            </ul>

```



```

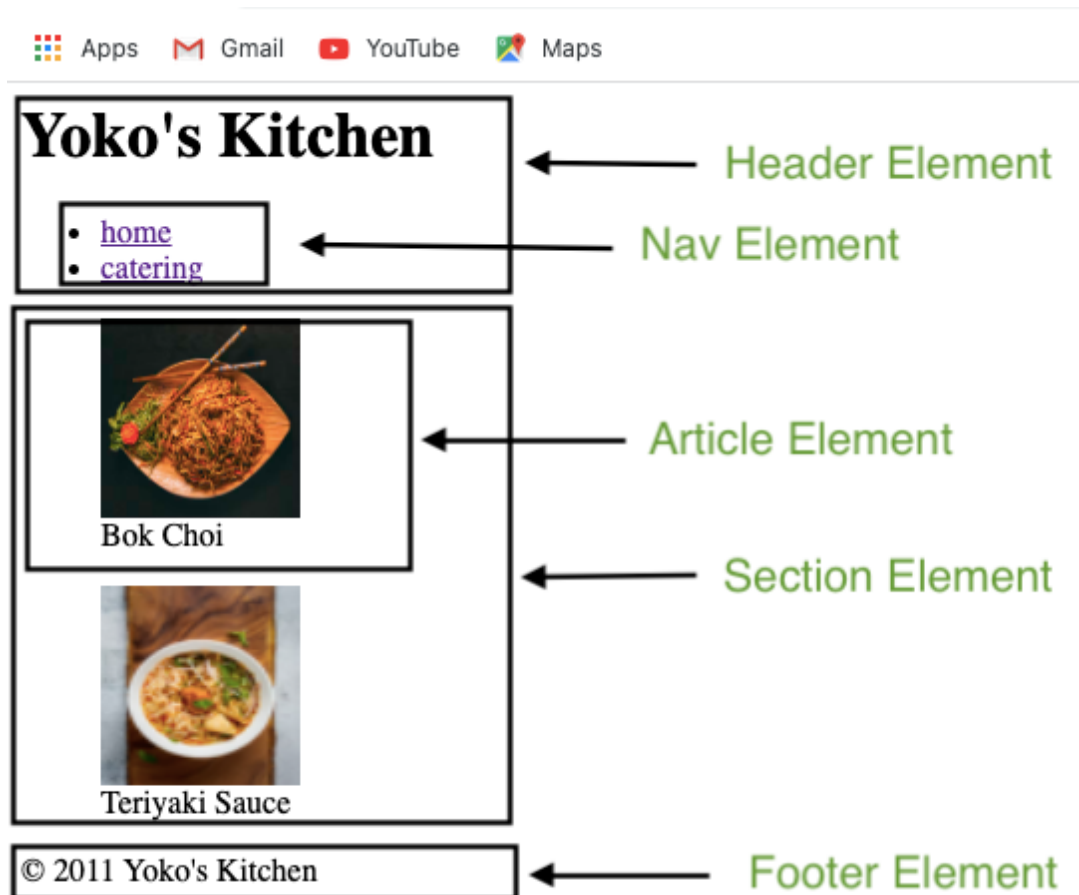
</nav>
</header>
<section>
  <article>
    <figure>
      
      <figcaption>Bok Choi</figcaption>
    </figure>

  </article>
  <article>
    <figure>
      
      <figcaption>Teriyaki Sauce</figcaption>
    </figure>
  </article>
</section>

<footer>
  &copy; 2011 Yoko's Kitchen
</footer>
</body>
</html>

```

Output:



Conclusion:

In this module, we have learnt about HTML from basic to the advanced level, we have learnt:

- About Web Development.
- About Frontend Web Development.

- About Web Page and it's working.
- About all the main HTML Tags.
- About HTML Tables.
- About HTML Forms.
- About Semantic HTML.

So, till now we have completed the HTML module and from the next module we are going to start designing our page with CSS.

If there isn't any text in between the HTML tags, what will happen?

If no text is present in between the tags, there would be nothing to format. As a result, nothing will show up on the screen. Some tags, such as those th don't have a closing tag, like the

tag, don't need any text in between them.

Describe the HTML5 layout structure i.e., using semantic tags ?

Every web page has different components to display the intended content and a specific UI. But still, there are few things which are templated and a globally accepted way to structure the web page, such as:


- : Stores the starting information about the web page.
- : Represents the last section of the page.
- : The navigation menu of the HTML page.
- : It is a set of information.
- : It is used inside the article block to define the basic structure of a page.
- : Sidebar content of the page.

How can we club two or more rows or columns into a single row or column in an HTML table?

HTML provides two table attributes "rowspan" and "colspan" to make a cell span to multiple rows and columns respectively.

Thank You !

Thank You !

	<p>tag.</p> <p>We can put the data we want to show in a cell in between these tags.</p> <div></div> <pre><table> <!-- First row - <tr> <td>15</td> <td>15</td> <td>30</td> </tr> <!-- second row <tr></pre>	
--	---	--


```
<td>45</td>
<td>60</td>
<td>45</td>
</tr>
</table>
```

In the above code, we have two rows which we have created using

tags.

Output of the above code:




Table Headings:


In order to make sense that what information is given in the rows and columns of a table we need to give headings to rows and columns.

In HTML tables this can be achieved using Table Heading tags.

or

element to represent the presence of an empty cell otherwise the table will not render correctly.

Output of the code:




Spanning columns in HTML Tables:

Sometimes we need the entries in a table to stretch across more than one column.


The **colspan** attribute can be used on a

or

element and indicates how many columns that cell should run across.



```
<table>
  <tr>
    <th></th>
    <th>9am</th>
    <th>10am</th>
    <th>11am</th>
    <th>12am</th>
  </tr>
  <tr>
    <th>Monday</th>
    <td colspan="2">Geograp
    <td>Math</td>
    <td>Art</td>
  </tr>
  <tr>
    <th>Tuesday</th>
    <td colspan="3">Gym</td>
    <td>Home Ec</td>
  </tr>
</table>
```



Explanation:

In the example, you can see a timetable with five columns; the first column contains the heading for that row (the day), the remaining four represent one hour time slots.

If you look at the table cell that contains the words 'Geography' you will see that the value of the **colspan** attribute is **2**, which indicates that the cell should run across two columns. In the third row, 'Gym' runs across three columns.

Output of the code:

	9am	10am	11am	12am
Monday	Geography		Math	Art
Tuesday	Gym			Home Ec

Spanning Rows in HTML Tables:

or


element and that cell shou

```
<table>
  <tr>

</tr>

<tr>

</tr>
</table>
```



Explanation:

In the example that ABC is sh 8pm, wherea channels are b during this tir lasts one hour) If you look at th

					Sometimes we need the entries in a table to stretch across more than one row.	
--	--	--	--	--	---	--

The **colspan attribute** can be used on a