Day 1- HTML Basic Tags + Inline and Block elements + HTML Lists + Html Tables.

**What is HTML?**

HTML (HyperText Markup Language) was created by Tim Berners-Lee in 1991 as a standard for creating web pages. It's a markup language used to structure content on the web, defining elements like headings, paragraphs, links, and images. HTML forms the backbone of web content. In layman's terms, HTML is like the skeleton of a website. It's a set of instructions that tells a web browser how to display text, images, videos, and other elements on a webpage. Think of it as the building blocks that create the structure and look of a website, similar to how bricks and mortar are used to build a house.

In a nutshell:

- HTML is the language of the web, used to create websites.
- HTML defines the barebone structure or layout of web pages that we see on the Internet.
- HTML consists of a set of tags contained within an HTML document, and the associated files typically have either a **.html** or **.htm** extension.
- There are several versions of HTML, with HTML5 being the most recent version.

**Quick Exercise:**

Open a webpage of your choice, right-click on the browser, and select 'View Page Source,' and then you will see the HTML code for that page.

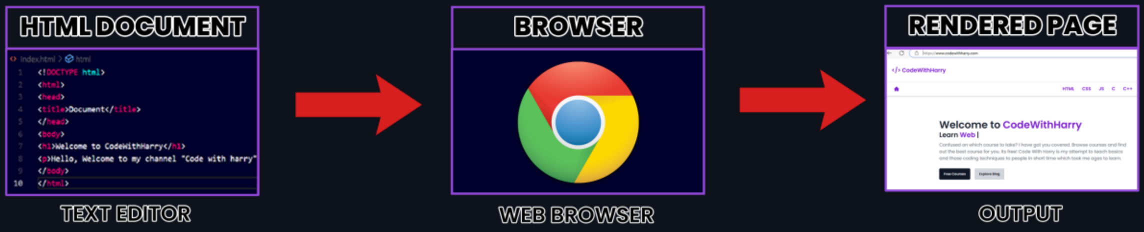
**A beautiful analogy to understand HTML, CSS, and JavaScript:**

Car Skeleton ( only body ) is HTML
Car Painted or Decorated is CSS
Car Engine and Internal logic is JS

**How do websites work?**

When we want to access any information on the internet, we search for it using a web browser. The web browser retrieves the content from web servers, where it is stored in the form of HTML documents.

An HTML document is created by writing code with specific tags in a code editor of your choice. The document is then saved with the **'.html'** extension. Once saved, the browser interprets the HTML document, reads it, and renders the web page.



```
<!DOCTYPE html>
<html>
<head>
<title>Document</title>
</head>
<body>
<h1>Welcome to CodeWithHarry</h1>
<p>Hello, Welcome to my channel "Code with harry"
</body>
</html>
```

**HTML Tags**

If you want to build a beautiful website, tags are essential elements that help you achieve that.

An HTML tag acts as a container for content or other HTML tags. Tags are words enclosed within < and > angle brackets.

They serve as keywords that instruct the web browser on how to format and display the content.

## Commonly used tags in HTML

Here are some commonly used tags in HTML. These are the only tags used 70% of the time.

### Document Structure Tags

1. `<DOCTYPE html>`: Specifies the document type.
2. `<html>`: Encloses the entire HTML document.
3. `<head>`: Contains meta-information and links to scripts and stylesheets.
4. `<body>`: Contains the content of the web page.

### Metadata Tags

1. `<title>`: Sets the title of the web page.
2. `<meta>`: Provides metadata such as character set, author, and viewport settings.
3. `<link>`: Links external resources like stylesheets.

### Text Formatting Tags

1. `<p>`: Paragraph.
2. `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`: Headings.
3. `<strong>`: Strong emphasis (typically bold).
4. `<em>`: Emphasis (typically italic).
5. `<br>`: Line break.
6. `<hr>`: Horizontal rule.

### List Tags

1. `<ul>`: Unordered list.
2. `<ol>`: Ordered list.
3. `<li>`: List item.

## Hyperlink and Media Tags

1. `<a>`: Anchor (used for links).
2. `<img>`: Image.
3. `<audio>`: Audio content.
4. `<video>`: Video content.

## Form Tags

1. `<form>`: Form.
2. `<input>`: Input field.
3. `<textarea>`: Text area.
4. `<button>`: Button.
5. `<select>`: Dropdown list.
6. `<option>`: Options within a `<select>` or `<datalist>`.

## Table Tags

1. `<table>`: Table.
2. `<tr>`: Table row.
3. `<td>`: Table data cell.
4. `<th>`: Table header cell.
5. `<thead>`: Table header group.
6. `<tbody>`: Table body group.
7. `<tfoot>`: Table footer group.

## Semantic Tags

1. `<header>`: Header section.
2. `<footer>`: Footer section.
3. `<article>`: Article.
4. `<section>`: Section.
5. `<nav>`: Navigation.
6. `<aside>`: Sidebar content.

## Paired and Unpaired HTML Tags

Well, that was a really long list. Don't worry, we will study these in detail. In HTML, tags can be broadly categorized into two types:

### 1. Paired Tags (Container Tags)

These are tags that come in pairs, consisting of an opening tag and a corresponding closing tag. The content goes between these two tags.

- **Opening Tag**: The opening tag starts with `<` and ends with `>`. For example, `<p>`.
- **Closing Tag**: The closing tag also starts with `<` but includes a forward slash `/` before the tag name, and ends with `>`. For example, `</p>`.

**Examples:**

- Paragraphs: `<p>This is a paragraph.</p>`
- Headings: `<h1>This is a heading.</h1>`

### 2. Unpaired Tags (Self-Closing Tags or Stand-Alone Tags)

These are tags that don't require a closing tag. They are self-contained, encapsulating all the information within a single tag.

- **Self-Closing Tag**: A self-closing tag starts with `<` and ends with `/>` (though the `/` is optional in HTML5). For example, `<img />` or `<br>`.

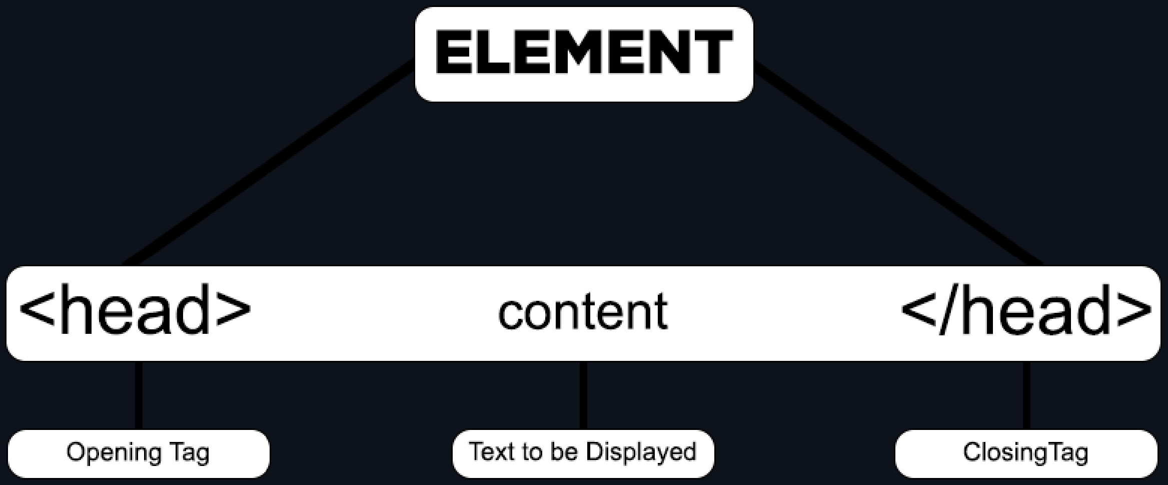
**Note:** Later if you happen to use React or a framework like Next.js, you will have to close the tag like this `<br/>` `<hr/>`. So it is better to cultivate the habit!

**Examples of self-closing tags:**

- Line Break: `<br/>`
- Horizontal Rule: `<hr/>`
- Image: `<img src="image.jpg" alt="An example image"/>`

## Pictorial Representation of Tags

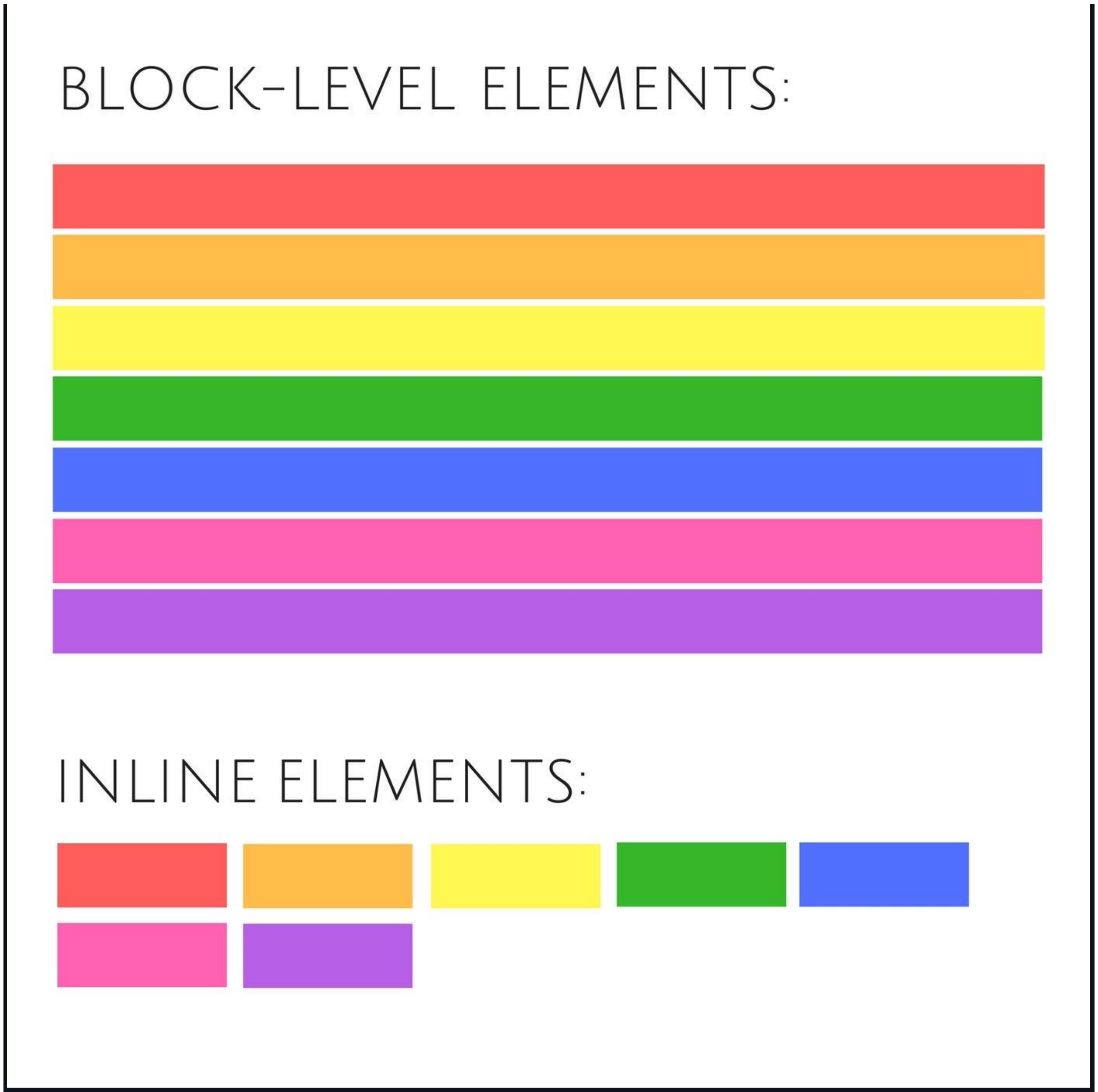
The image below offers a visual representation of how tags are structured in HTML. As you can see, an element can contain other elements, which may also contain additional elements, forming a tree-like structure. This hierarchy can include self-closing tags as well as nested tags, as illustrated in the picture.



## HTML Inline Elements

Inline Elements don't start on a new line. It only takes the width required to cover the content.

HTML elements are generally divided into two categories: **Block-level** and **Inline** elements.



No matter what the width is, block elements will always start on a new line and take up the full available width of their container by default. This means they essentially claim all the horizontal space for themselves, pushing any content that comes after them to a new line. But the inline elements will fit snugly within the flow of other elements, taking up only as much width as their content requires.

**What are Inline Elements?**

Inline elements do not start on a new line and only take up as much width as necessary. They are part of the flow within other elements.

Inline elements can contain other inline elements, but they generally should not contain block-level elements. For example, you could nest a `<strong>` (strong emphasis) element within a `<span>` (a generic inline container) element, like so:

**Common Inline Elements**

- `<span>`: A generic inline container for text
- `<a>`: Defines a hyperlink
- `<strong>`: Defines important text
- `<em>`: Defines emphasized text
- `<img>`: Embeds an image
- `<input>`: Defines an input control

**HTML Block Elements**

You already know about HTML inline elements. All HTML tags have specific display behavior: they are either block-level elements or inline elements.

**What are Block-level Elements?**

Block-level elements are those that start on a new line and take up the entire width of their container by default. They essentially claim all the horizontal space for themselves, pushing any content that comes after them to a new line.

**Characteristics of Block-level Elements:**

- Always start on a new line.
- Take up the full width available.

- Width and height can be controlled via CSS.
- Can contain other block-level as well as inline elements.

**Common Block-level Elements:**

- `<h1>,<h2>,<h3>,<h4>,<h5>,<h6>` - Headings
- `<p>` - Paragraphs
- `<hr>` - Horizontal rule
- `<address>` - Address information
- `<article>` - Article content
- `<aside>` - Sidebar content
- `<blockquote>` - Block quotations
- `<canvas>` - Drawing area
- `<dd>` - Description in a description list
- `<div>` - Generic container
- `<dl>` - Description list
- `<dt>` - Term in a description list
- `<fieldset>` - Group of related form elements
- `<figcaption>` - Caption for a figure
- `<figure>` - Image or media with a caption
- `<footer>` - Footer of a section or page
- `<form>` - HTML form
- `<header>` - Header of a section or page
- `<li>` - List item
- `<main>` - Main content of a document
- `<nav>` - Navigation links
- `<noscript>` - Alternate content when JavaScript is not enabled
- `<ol>` - Ordered list
- `<ul>` - Unordered list
- `<pre>` - Preformatted text
- `<section>` - Standalone section in a document
- `<table>` - Table
- `<video>` - Video content

## Types of HTML Lists

HTML provides different types of lists to display data in various forms. Each list contains one or more list items.

- **Unordered List:** Displays items using bullets.
- **Ordered List:** Displays items in a numerical sequence, and supports various numbering styles like Arabic numerals, Roman numerals, and so on.
- **Definition List:** Organizes items in a format similar to a dictionary, with terms and their corresponding definitions.

# HTML Tables

HTML tables allow you to arrange data like text, images, and links in rows and columns. You use the `<table>` tag to start and end a table.

### Syntax of HTML Table

```
<table>  // table content</table>
```

### Key Elements of HTML Table

- `<table>`: Defines the table itself.
- `<tr>`: Used for table rows.
- `<th>`: Used for table headings.
- `<td>`: Used for table cells (data).

### Basic Table Structure

```
<table>  <tr>    <th>Name</th>    <th>Age</th>  </tr>  <tr>
<td>Harry</td>    <td>100</td>  </tr></table>
```

## rowspan and colspan Attributes

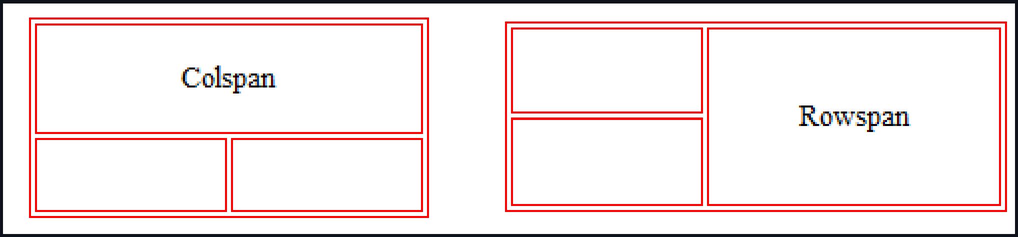
**Rowspan:** If you want a table cell to span multiple rows, you can use the `rowspan` attribute.

```
<td rowspan="value">
```

**Colspan:** If you want a table cell to span multiple columns, you can use the `colspan` attribute.

```
<td colspan="value">
```

### Visual Representation of Rowspan and Colspan



### Examples

Here are simple examples to demonstrate the use of `rowspan` and `colspan` in HTML tables.

**Example for Colspan:**

```
<table border="1">  <tr>    <td colspan="2">Merged Columns</td>  </tr>
<tr>    <td>Column 1</td>    <td>Column 2</td>  </tr></table>
```

**Example for Rowspan:**

```
<table border="1">  <tr>    <td>Row 1, Column 1</td>    <td
rowspan="2">Merged Rows</td>  </tr>  <tr>    <td>Row 2, Column 1</td>
</tr></table>
```

## HTML Attributes

HTML attributes are used to define the characteristics of an HTML element. They are placed within the element's opening tag and consist of two parts: the **name** and the **value**.

- **Name**: Specifies the property for that element.
- **Value**: Sets the value of that property for the element.

## Types of HTML Attributes

There are three main types of HTML attributes:

1. **Core Attributes**: These are basic attributes that can be applied to most HTML elements. Examples include `id`, `class`, and `style`.
2. **Internationalization Attributes**: These attributes help adapt the document to different languages and regions. Examples include `lang` and `dir`.
3. **Generic Attributes**: These attributes provide additional information about the element but don't necessarily affect its appearance or behavior. Examples include `data-*` attributes for storing custom data private to the page or application.

Core attributes are some of the most widely used attributes in HTML. There are four main types:

- `id`
- `class`
- `title`
- `style`

## ID Attribute

The ID attribute is used to assign a unique identifier to an HTML element. Each element with an ID has its own unique identity, similar to how each individual has a unique identity. Multiple elements cannot have the same ID.

Example:

```
<p id="html">This is an HTML tutorial</p><p id="python">This is a
Python tutorial</p>
```

In this example, the ID attribute helps to distinguish between two paragraphs by having different values: "html" and "python".

**Class Attribute**

The class attribute is used to associate an HTML element with a particular class, typically for styling or JavaScript manipulation. Unlike the ID attribute, the class attribute is not unique, and multiple elements can share the same class.

**Title Attribute**

The title attribute provides additional information about an element and is often displayed as a tooltip when the mouse hovers over it.

Example:

```
<h4 title="hello, motto">Title attribute</h4>
```

Output:

**Style Attribute**

The style attribute allows for inline styling of HTML elements. It is used in conjunction with CSS properties to directly style individual elements within the HTML code.

**Case Sensitivity**

The HTML standard is flexible about the case of attribute names, allowing them to be written in either uppercase or lowercase, such as "title" or "TITLE." However, for best practices and compatibility with stricter document types like XHTML, the W3C recommends using lowercase attributes.

## HTML Comments

Comments in HTML are like little notes you leave in your code for yourself or other people. These notes help make the code easier to understand but don't show up on the actual website. The web browser just skips over them!

### Key Points About HTML Comments

- Comments are ignored by web browsers.
- They aid in code readability and documentation.
- HTML comments are denoted by `<!-- content -->`.
- The shortcut key for commenting out code is `Ctrl + /`.
- HTML supports both single-line and multi-line comments.

## Types of Comments in HTML

HTML primarily supports two types of comments:

### Single-line Comments

Single-line comments are contained within one line. They are useful for short annotations.

Example:

```html
<!-- This is a single-line comment -->
```

As you can see in the video below, the text inside the comment is not rendered.

### Multi-line Comments

Multi-line comments span across multiple lines, making them ideal for detailed explanations or temporarily disabling blocks of code.

Example:

```
<!--  This is a multi-line comment.  It spans multiple lines.-->
```

**HTML Id & Classes**

HTML offers multiple ways to select and style elements. Two of the most commonly used selectors are IDs and Classes. This blog explores what they are, how to use them, and their differences.

**What is an ID?**

An ID is an attribute, a unique identifier assigned to only one HTML element within a page. It is often used for unique styling and JavaScript manipulations.

```
<div id="myUniqueID">This is a div with an ID.</div>
```

**What are Classes?**

The `class` attribute lets you give the same name to multiple HTML elements. That way, you can easily change their look or behavior all at once. Classes are not unique and can be assigned to multiple elements. They are generally used for applying the same styles or behaviors to a group of elements.

```
<div class="myClass">This is a div with a class.</div><p
class="myClass">This is a paragraph with the same class.</p>
```

**The Style Tag**

The `style` tag in HTML is used to include embedded CSS (Cascading Style Sheets) within an HTML document. It is commonly placed within the `<head>` section of the HTML file, although it can technically be used in the `<body>` as well. The `style` tag allows you to define the look and feel of various HTML elements on the page, including their colors, sizes, margins, and other visual styles.

Here's a simple example:

```
<!DOCTYPE html><html><head>  <style>    /* CSS rules go here */    p
{      color: blue;      font-size: 18px;    }    .highlight
{      background-color: yellow;    }  </style></head><body>  <p>This
is a blue paragraph.</p>  <p class="highlight">This paragraph has a
yellow background.</p></body></html>
```

In this example, we have targeted the second paragraph by its class name in CSS. The style tag is used to add CSS right into HTML. We will learn about CSS and selectors later in the CSS tutorial.

**Using IDs and Classes in CSS**

In CSS, elements with IDs are selected using a hash (`#`) symbol before the ID, and elements with classes are selected using a dot (`.`) before the class name.

```
/* CSS for ID */#myUniqueID {  background-color: yellow;} /* CSS for
Class */.myClass {  font-size: 18px;}
```

**Differences Between IDs and Classes**

- **Uniqueness:** IDs are unique, and classes can be reused.
- **JavaScript:** IDs are often used for JavaScript operations.
- **Styling:** Both can be used for styling, but IDs have higher specificity.
- 
- **Conclusion**

Understanding the difference between IDs and Classes is crucial for effective web development. While IDs are for unique elements, classes are for grouping elements.