codingninjas

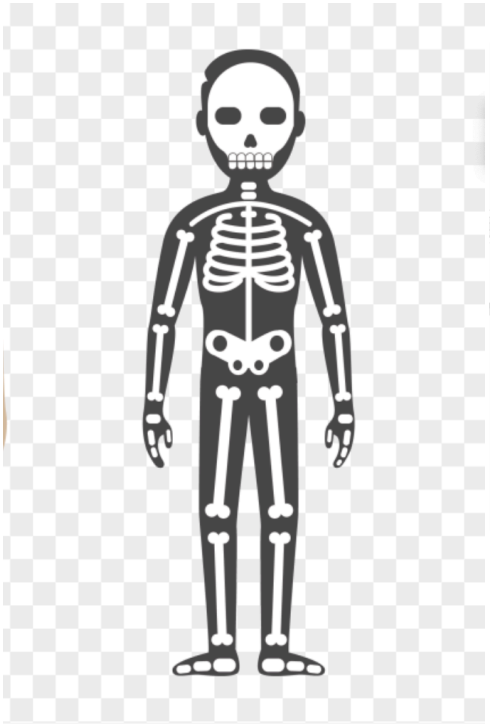
# Class 1: Fundamentals of HTML and CSS

## Session Overview

1. Introduction to HTML
2. HTML Document Structure
3. Basic of the HTML Tags
4. Introduction to CSS
5. Selectors and properties

## Introduction to HTML

HyperText Markup Language, or HTML, is the standard markup language for describing the structure of documents displayed on the web.



The way the skeletal system provides structure to our body in the similar way HTML provides structure to our website.

HTML consists of a series of elements and attributes that are used to mark up all the components of a document to structure it in a meaningful way.

HTML documents are a tree of nodes, including HTML elements and text nodes.

## HTML Document Structure

HTML documents include a document type declaration and the <html> root element. Nested in the <html> element are the document head and document body.

```
<!DOCTYPE html>
<html lang="en-US">
    <head>
    </head>
    <body>
    </body>
</html>
```

a. **<!doctype html>**

   The doctype tells the browser to use standards mode. The only purpose of <!DOCTYPE html> is to activate no-quirks mode.

**Quirks Mode**

- Quirks mode refers to a compatibility mode in web browsers that allows older web pages to be displayed correctly even if they don't strictly follow modern web standards.
- History: Earlier webpages were written in two browsers: Netscape Navigator and Microsoft Internet Explorer. When the web standards were made at W3C, browsers could not just start using them, as doing so would break most existing sites on the web. Browsers therefore introduced two modes to treat new standards-compliant sites differently from old legacy sites.
- There are now three modes used by the layout engines in web browsers: quirks mode, limited-quirks mode, and no-quirks mode.
- In quirks mode, the layout emulates behaviour in Navigator 4 and Internet Explorer 5. It supports older browsers that do not follow new web standards.

- In **no-quirks mode**, the behaviour is the desired behaviour described by modern HTML and CSS specifications.
- In **limited-quirks mode**, there are only a very small number of quirks implemented.

**b. <html>**

The <html> element is the root element for an HTML document. It is the parent of the <head> and <body>, containing everything in the HTML document other than the doctype.

**Content Language**

The “lang” language attribute added to the <html> tag defines the main language of the document. The value of the lang attribute is a two- or three-letter ISO language code followed by the region.

**c. <head>**

The <head>, or document metadata header, contains all the metadata for a site or application. All the tags added inside the head tag aren’t visible on the website.

Tags that can be added inside the head tag:
The document metadata, including the document title, character set, viewport settings, description, base URL, stylesheet links, and icons, are found in the <head> element. While you may not need all these features, always include character set, title, and viewport settings.

**d. <body>**

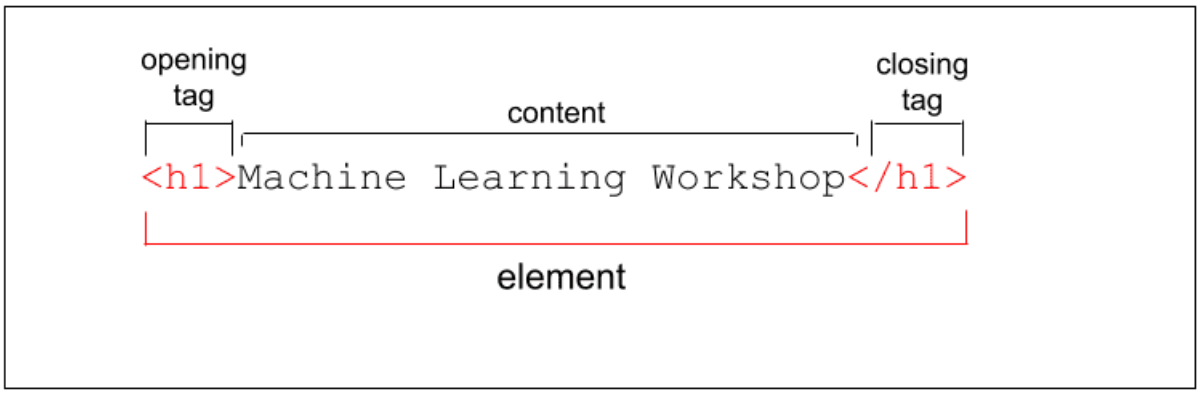
The body contains all the elements that you want to be visible on the website.

## Basics of HTML Tags

### Elements:

- The HTML element consists of both the opening and closing tags as well as the content inside the tag.
- Elements and tags aren't the exact same thing, though many people use the terms interchangeably. The tag name is the content in the brackets. The tag includes the

brackets. In this case, <h1>. An "element" is the opening and closing tags, and all the content between those tags, including nested elements.




## Tags:

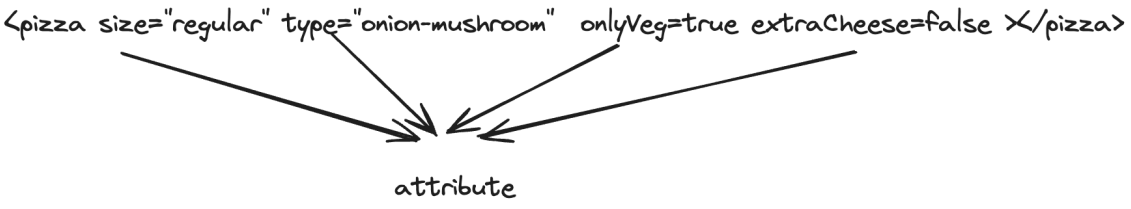
- HTML Tags are the opening and closing parts of an HTML element. They begin with < symbol and end with > symbol. Whatever is written inside < and > are called tags.

## Attributes:

**Syntax:**

```
<element attribute_name="value">content</element>
```

- HTML attributes are used to describe the characteristics of an HTML element in detail.
- HTML attributes can only be added in the starting tag.
- Consider an example where you are ordering a pizza, how do you specify which pizza you want? You consider the following features: the size of the pizza, toppings to be added, veg/non-veg, extra-cheese etc. If we write that in the form of an element it would look like this:
  Here size, type, onlyVeg and extraCheese are attributes.



- Similarly, let's consider an element now:

```
<img src="codingninjaslogo.jpg" height="400" width="600">
```

Here src, height and width are attributes.

## Introduction to CSS

Cascading Style Sheets (CSS) are used to style web pages.

**CSS Syntax:**

- The selector is used to target the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.


**Using CSS**

There are three ways in which you can add CSS to your document:

- **Inline** - here we use the style attribute inside HTML elements
- **Internal** - here we use a <style> element in the <head> section
- **External** - here we use a <link> element to link to an external CSS file

**Inline CSS:**

Inline CSS is used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant HTML tag. This method is not recommended for large projects because it mixes style with HTML documents. Eg:

```
<p style="color: blue; font-size: 16px;">This is a blue paragraph.</p>
```

**Internal CSS**

Internal CSS is used when a single HTML document needs to be styled uniquely. Internal styles are defined within the <style> tag in the <head> section of the HTML document.

```
<html>
<head>
    <style>
        body {
            background-color: lightblue;
        }
        h1 {
            color: navy;
            margin-left: 20px;
        }
    </style>
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

**External CSS:**

External CSS is used when you need to apply styles to multiple HTML documents. External styles are defined in a separate CSS file, which is linked to the HTML document using the <link> tag within the <head> section.

Eg:

***HTML file (index.html)***

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
```

```
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

***CSS File (style.css)***

```
body {
    background-color: lightblue;
}
h1 {
    color: navy;
    margin-left: 20px;
}
```

External CSS is the best way to add CSS to your project because:

1. **Single responsibility:** External CSS allows for a clear separation between the structure/content of a web page (HTML) and its presentation/styling (CSS). This improves the maintainability and readability of both HTML and CSS files.
2. **DRY:** External CSS files can be linked to multiple HTML documents, enabling you to apply the same styles across different pages of a website. This reduces redundancy and follows the don't repeat yourself principle(DRY).

## Selectors in CSS

CSS selectors are patterns used to select the elements you want to style. They can be divided into 5 categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

**Universal Selector (*):** Selects all elements on the page.

```
* {
    margin: 0;
    padding: 0;
}
```

## Simple Selectors:

1. **Type Selector:** A type selector matches an HTML element directly.

```
section {
  padding: 2em;
}
```

This rule causes every <section> element to have 2em of padding on all sides.

2. **Class Selector (.):** A HTML element can have one or more classes defined in its class attribute. The class selector in CSS matches any element that includes that class.
   Eg:

```
<div class="my-class"></div>
<button class="my-class"></button>
<p class="my-class"></p>
```

Any element with the class my-class applied will be styled according to the following CSS rule, which colours the text red:

```
.my-class {
  color: red;
}
```

Note that the “.” character is only present in CSS, not in the HTML. In CSS, the “.” character specifies that the selector is targeting a class attribute. This is a common pattern in CSS, where special characters are used to define different types of selectors.

An HTML element with multiple classes will still match the CSS rule if one of its classes is my-class. For instance:

```
<div class="my-class another-class some-other-class"></div>
```

In this example, the div element will be styled with the colour red because it includes the my-class among its classes. CSS matches any class attribute that contains the defined class, rather than requiring an exact match of the class list.

3. **Id selector (#):** An HTML element with an id attribute should be the only element on a page with that ID value. You select elements with an ID selector like this:

```
#my-id {
  // styles
}
```

This CSS would apply a blue border to the HTML element that has an id of my-id, like this:

```
#my-id {
  border: 1px solid blue;
}
```

Similarly to the class selector ., use a # character to instruct CSS to look for an element that matches the id that follows it.

**CSS Group Selector**

The grouping selector is used to select all the elements to which you want to give the same style.
Eg:

```
h1, .my-class, #my-id {
  text-align: center;
  color: red;
}
```

This would apply the same style on h1, my-class, my-id making the text central aligned and text color red.

# Combinator Selectors

Combinator selectors are used in CSS to select elements based on the relationship between them. There are four types of combinators in CSS that are listed as follows:

1. General sibling selector (~)
2. Adjacent sibling selector (+)
3. Child selector (>)
4. Descendant selector (space)

## Descendant Selector:

The descendant selector in CSS is used to apply styles to elements that are nested within a specified parent element. It matches all elements that are descendants of a specified element, no matter how deep the nesting goes.

**Syntax**

The syntax for the descendant selector is straightforward:

```
ancestor-descendant {
    /* styles */
}
```

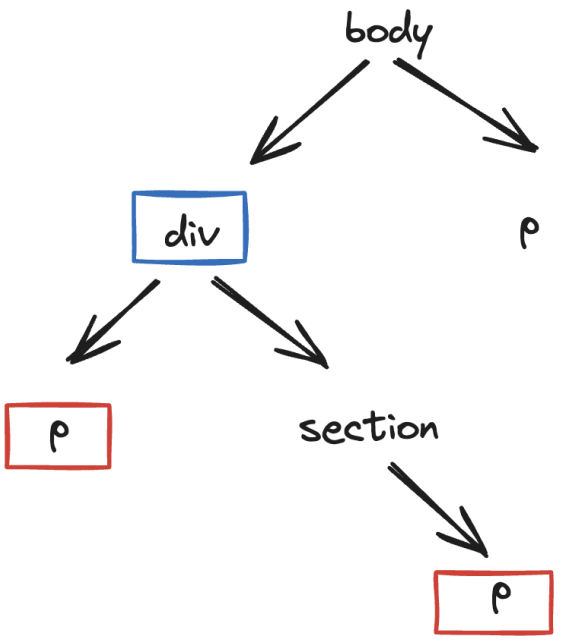
- descendant: The child element that is nested inside the ancestor element.
- ancestor: The parent element.

**Example:**

```
div p {
    color: blue;
}

<div>
    <p>This paragraph inside a div will be blue.</p>
    <section>
        <p>This paragraph inside a section within the div will also be blue.</p>
    </section>
</div>
<p>This paragraph outside the div will not be blue.</p>
```

See the tree representing the HTML document below,




In this example:

- The CSS rule div p targets all <p> elements that are descendants of any  element.
- The <p> elements inside the  will be styled with color: blue;, regardless of how deeply nested they are.
- The <p> element outside the  will not be affected by this rule.

## Child Selector:

The child selector in CSS is used to select and style elements that are direct children of a specified parent element. This means it only targets elements that are immediately nested within the parent element, not deeper nested descendants.

**Syntax**

The syntax for the child selector is:

```
parent > child {
   /* styles */
}
```

- parent: The parent element.
- child: The direct child element of the specified parent element.

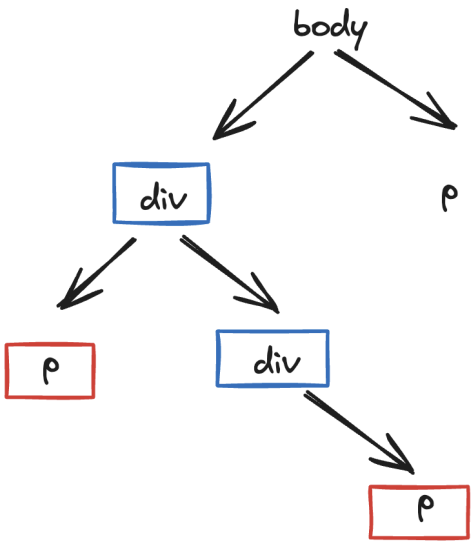
**Example**

Let's go through an example to see how the child selector works.

```
div > p {
        color: blue;
        font-weight: bold;
    }

<div class="container">
    <p>This paragraph is a direct child of div and will be blue and bold.</p>
    <div>
        <p>This paragraph is nested inside another div also child of this div so will be blue and bold.</p>
    </div>
  </div>
  <p>This paragraph is outside of the div and will not be blue or bold.</p>
```

In this example:




- The CSS rule div > p targets all p elements that are direct children of any div element.
- The direct child p elements of the top-level div will be styled with color blue and text bold.

- The nested p inside the nested div will be affected as it is the direct child of the div element.

## General Sibling Selector in CSS

The general sibling selector in CSS allows you to select elements that are siblings of a specified element and share the same parent, regardless of their position in the HTML structure. It targets elements that come after the specified element in the HTML markup.

**Syntax**

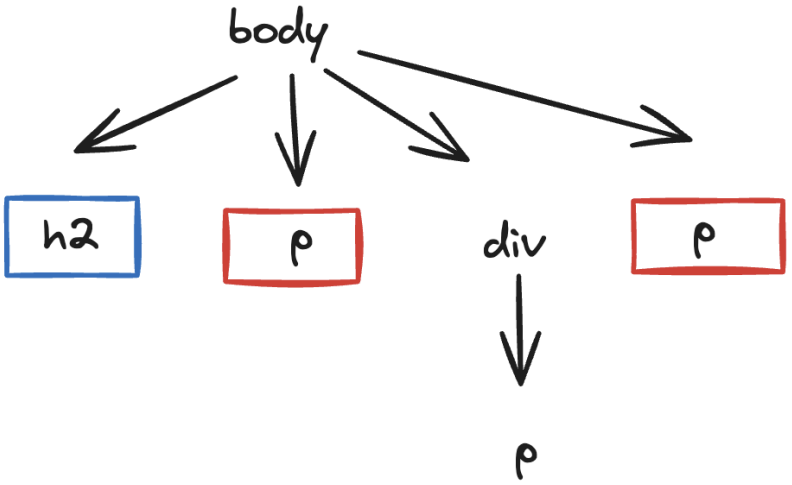
The syntax for the general sibling selector is:

```
element1 ~ element2 {
 /* styles */
}
```

- element1: The reference element.
- ~: The general sibling combinator.
- element2: The sibling element that comes after element1.

**Example**

Let's go through an example to illustrate how the general sibling selector works.




```
h2 ~ p {
    color: purple;
```

```
}
<h2>Heading</h2>
<p>This paragraph is styled because it comes after the h2.</p>
<div>
  <p>This paragraph is not styled because it's not a sibling of the h2.</p>
</div>
<p>This paragraph is also styled because it's a sibling of the h2.</p>
```

In this example:

- The CSS rule h2 ~ p selects all p elements that are siblings of h2 elements and come after them.
- The first p element is styled because it is a sibling of the h2 element.
- The second p element inside the div is not styled because it's not a sibling of the h2 element.
- The third p element after the div is styled because it's a sibling of the h2 element.

### Adjacent sibling selector (+)

The adjacent sibling selector in CSS, denoted by the plus sign (+), allows you to select elements that are immediately preceded by a specified element, and both share the same parent. This selector targets elements that are adjacent to each other in the HTML markup, with no other elements in between.

**Syntax**

The syntax for the adjacent sibling selector is:

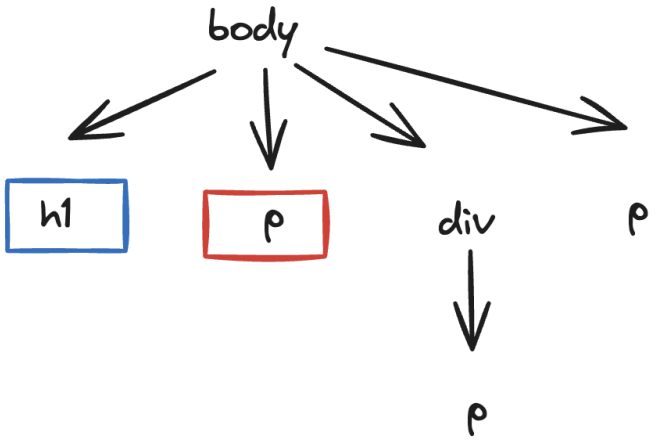
```
element1 + element2 {
  /* styles */
}
```

- **element1**: The reference element.
- **+**: The adjacent sibling combinator.
- **element2:** The sibling element that directly follows element1.

Example:

```
h1 + p {
        color: blue;
        font-weight: bold;
    }
<h1>Title</h1>
  <p>This paragraph is styled because it immediately follows the h1.</p>
  <div>
    <p>This paragraph is not styled because it's not an adjacent sibling of the h1.</p>
  </div>
  <p>This paragraph is also styled because it's an adjacent sibling of the h1.</p>
```

In this example:




- The CSS rule h1 + p selects all p elements that are immediately preceded by h1 elements.
- The first and third p elements are styled because they immediately follow an h1 element.
- The second p element inside the div is not styled because it's not an adjacent sibling of any h1 element.

## Pseudo-class selectors

Pseudo-class selectors in CSS allow you to style elements based on various states or conditions that cannot be targeted using regular selectors alone. These states or conditions include user interaction, element position, and element type. Pseudo-classes are prefixed with a colon (:).

**Syntax**

The syntax for pseudo-class selectors is:

```
selector:pseudo-class {
   /* styles */
}
```

- selector: The element or elements to which the pseudo-class should apply.
- pseudo-class: The state or condition to be targeted.

### 1. :hover

The ":hover" pseudo-class targets an element when the user hovers over it with the cursor.

```
button:hover {
 background-color: #ffcc00;
}
```

### 2. :active

The ":active" pseudo-class targets an element when it is being activated or clicked by the user.

```
a:active {
 color: red;
}
```

### 3. :focus

The ":focus" pseudo-class targets an element when it has keyboard focus.

```
input:focus {
 border-color: blue;
}
```

### 4. :nth-child(n)

The ":nth-child(n)" pseudo-class targets elements based on their position within a parent element.

```
ul li:nth-child(odd) {
 background-color: lightgray;
}
```

### 5. :first-child and :last-child

These pseudo-classes target the first and last child elements of a parent, respectively.

```
ul li:first-child {
 font-weight: bold;
}

ul li:last-child {
 color: #333;
}
```

### 6. :not(selector)

The ":not(selector)" pseudo-class selects all elements that do not match the specified selector.

```
a:not(.external) {
 color: blue;
}
```

### 7. :checked

The ":checked" pseudo-class targets radio buttons or checkboxes are checked.

```
input[type="checkbox"]:checked {
 background-color: green;
}
```

## Pseudo-elements selectors

Pseudo-elements selectors in CSS allow you to style specific parts or elements of an element. Unlike pseudo-classes, which target the state of an element, pseudo-elements target parts of an element that do not exist in the HTML structure, such as the first line of text or the first letter of a paragraph. Pseudo-elements are prefixed with two colons (::).

Syntax

The syntax for pseudo-elements selectors is:

```
selector::pseudo-element {
 /* styles */
}
```

- selector: The element to which the pseudo-element should apply.
- pseudo-element: The part or element of the selected element to be targeted.

## Common Pseudo-elements and Examples

### 1. ::first-line

The “::first-line” pseudo-element targets the first line of text within an element.

```
p::first-line {
 font-weight: bold;
}
```

### 2. ::first-letter

The “::first-letter” pseudo-element targets the first letter of text within an element.

```
p::first-letter {
 font-size: larger;
 color: red;
}
```

### 3. ::before and ::after

The “::before” and “::after” pseudo-elements allow you to insert content before or after an element's content, respectively.

```
blockquote::before {
 content: "\201C"; /* Insert opening quotation mark */
}

blockquote::after {
 content: "\201D"; /* Insert closing quotation mark */
}
```

### 4. ::selection

The “::selection” pseudo-element targets the portion of text selected by the user.

```
::selection {
 background-color: #ffff00; /* Yellow background */
 color: #333; /* Dark text color */
}
```

## Attribute Selector

Attribute selectors in CSS allow you to target elements based on the presence of specific attributes or attribute values within their HTML markup.

### 1. Presence Selector

Targets elements that have a specified attribute, regardless of its value.

**Example**:

```
[target] {
    border: 2px solid #ccc;
}}
```

This CSS rule applies a border to all elements with a target attribute, regardless of its value.

### 2. Exact Value Selector

Targets elements that have a specific attribute value.

Example:

```
[lang="en"] {
    font-family: "Arial", sans-serif;
};
}
```

This rule applies a specific font to all elements with a lang attribute set to "en".

### 3. Partial Value Selector

Targets elements where the attribute value contains a specific substring.

Example:

```
[href*="example"] {
    color: blue;
}
```

This rule applies a blue color to all elements with an href attribute containing the substring "example".

### 4. Begins with Value Selector

Targets elements where the attribute value begins with a specific substring.

**Example**:

```
[src^="https://"] {
    border: 1px solid #ccc;
}
```

This rule adds a border to all elements with a src attribute that starts with "https://".

### 5. Ends with Value Selector

Targets elements where the attribute value ends with a specific substring.

**Example**:

```
[src$=".pdf"] {
    background-color: #f0f0f0;
}
```

This rule applies a background color to all elements with a src attribute that ends with ".pdf".

### 6. Space-separated Value Selector

Targets elements where the attribute value contains a specific word in a space-separated list.

Example:

```
[class~="active"] {
    color: green;
}
```

This rule applies a green color to all elements with a class attribute containing the word "active".

### 7. Prefix-matched Value Selector

Targets elements where the attribute value is prefixed by a specific substring, followed by a hyphen.

**Example**:

```
[lang|="en"] {
    font-weight: bold;
}
```

This rule applies bold font weight to all elements with a lang attribute starting with "en-".