HTML Notes

Introduction

HTML (Hypertext Markup Language) is the standard language used to create web pages. It allows linking from one location to another using hyperlinks.

Purpose of HTML

HTML defines the structure of a document using various tags. (Make up Language)

Make up kisko? Content ko

Make up kaise lagayenge ? using Tags

Tags in HTML

Tags are the basic elements used to structure content in HTML.

Example of a Header Tag
<h1>This is our Title</h1>
Converts ordinary text into header text.

Paragraph Tag
This is a paragraph.

Horizontal Rule (hr) Tag
<hr/>

Line Break (br) Tag

Note: Tags like <hr/>,
, and <img/> are self-closing tags that do not require content.

Div Tag

The tag is a container used to group HTML elements.

HTML Elements

An HTML element consists of a tag and content:

Tag + Content = HTML Element

Is <hr/> an HTML Element?
Yes, <hr/> is an HTML element even though it does not contain content.

List Tags

Unordered List (ul)

```
<ul>
<li>Item 1</li>
<li>Item 2</li>
</ul>
```

Ordered List (ol)

```
<ol>
<li>Item 1</li>
<li>Item 2</li>
</ol>
```

Anchor Tag

The <a> tag creates hyperlinks. Attributes are necessary for it to work.

Example:

```
<a href="https://example.com">Link</a>
```

Attributes in HTML Elements

Attributes are written inside the opening tag of an element.

Common Attributes for <a> Tag:
- href: Specifies the URL.

Common Attributes for <img> Tag:
- alt: Provides alternative text for images.
- width and height: Set dimensions for images.

Figure and Figcaption

Instead of using <img> directly, <figure> can group an image and a caption:

```
<figure>
<img src="image.jpg" alt="Description">
<figcaption>Image Caption</figcaption>
</figure>
```

Boilerplate Template

A boilerplate is a basic structure required for HTML documents. In VS Code, typing "!" and pressing Enter generates a complete boilerplate.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Document</title>
</head>
<body>
<!-- Content goes here -->
</body>
</html>
```

Setting Background in Body

```
<body background="background.jpg">
</body>
```

```
//Add Background img without css
// You can use some css inside HTML Element
// using attribute style=" CSS CODE";
```

Example : **<body style="background-repeat: no-repeat; background-size: cover; "**

For CSS we have syntax
Property_name : Propety_value;

CSS

There are 3 ways to apply CSS.

Case 1: Inline CSS

In this type, CSS is directly applied to the HTML element using the style attribute.

Example:

```
<div style="border: 1px solid
black;"></div>
```

Attribute format:

attribute-name = "attribute-value"
CSS format:

property-name : property-value;

Case 2: Internal CSS

In this CSS, we write properties directly in the same HTML file using the style tag.
Here, HTML and CSS are in the same file.

CSS selectors are needed here.

Example:

```
h2 {
margin-bottom: 20px;
}
```

Case 3: External CSS

In this CSS, styles are written in a separate CSS file and linked to the HTML file.

HTML and CSS are in different files.

A link tag is used inside the head section to connect the CSS file.

Example (HTML file):

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

Example (CSS file – style.css):

```
h2 {
color: blue;
margin-bottom: 20px;
}
```

CSS POSITIONS

---

position: static;

In this position, the location is always default.
We cannot use offsets here.
Static position = default position.

---

position: relative;

In this position property, we can use offsets.
Relative position = default position + offset.

---

position: fixed;

In this position, the offset is given with reference to the viewport address.
Fixed position = viewport address + offset
(works from origin)

---

position: absolute;

When there is no positioned parent, then the formula is same as fixed.
When there is a positioned parent, the formula is:

Absolute position = parent address + offset.

CSS DISPLAY PROPERTIES

---

display: inline;

These elements do not start on a new line.
Their width and height cannot be changed.

---

display: block;

Every element with display block will start on a new line.
Their width and height can be adjusted.

---

display: inline-block;

These elements do not start on a new line.
Their width and height can be adjusted.
It takes advantages of both inline and block.

FLEX

Flexbox is a layout model that allows elements to be arranged in rows or columns, providing more flexibility.
We have to change the display property of the parent container into flex.
Then the child containers are called flex items.

Flex Properties

1. flex-direction:

It determines the direction of the main axis.
Its values are:
row, column, row-reverse, column-reverse.

2. justify-content:

Justify-content aligns the items along the main axis.
Values are:
flex-start, flex-end, center, space-between, space-around, space-evenly.

3. gap:

It adds space between each flex item.

4. align-items:

It is used to align flex items along the cross axis.

---

5. order:

It is used for changing the position of flex items.

---

6. flex-wrap:

It defines the behavior when there is not enough space.

i) nowrap:

Single line only.

ii) wrap:

Flex items move to the next line.

iii) wrap-reverse:

Flex items move from bottom to top.