HTML & CSS Essentials for Beginners

FULL HTML STRUCTURE

Every HTML document must start with `<!DOCTYPE html>` and contain two main sections: `<head>` and `<body>`.

HEAD SECTION (Not Visible on the Page)

Used to include metadata, styles, and links.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Page</title>
    <link rel="stylesheet" href="style.css">
    <style>
      body { background-color: #f0f0f0; }
    </style>
    <script src="head-script.js"></script> <!-- Loads before body -->
  </head>
</html>
```

Head Tags:

- `<!DOCTYPE html>` – Declares HTML5 document.
- `<html>` – Root container.
- `<head>` – Metadata and resource links.
- `<meta charset="UTF-8">` – Sets character encoding.
- `<meta name="viewport">` – Makes the page responsive.
- `<title>` – Sets browser tab text.
- `<link>` – Links to external CSS.
- `<style>` – Adds internal CSS.
- `<script>` – Links or includes JavaScript.
- `<base>` – Specifies base URL for relative links.
- `<meta name="description">` – Page description (SEO).

Where to Insert `<script>` Tags:

- **In the `<head>`** – Loads before content, can block rendering unless `defer` or `async` is used.

  ```
  <script src="script.js" defer></script>
  ```

- **At the end of `<body>`** – Preferred for loading after HTML content.

  ```
  <body>
    <!-- content -->
  ```

```
  <script src="script.js"></script>
</body>
```

🧱 BODY SECTION (Visible Content)

Contains everything shown on the screen.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Visible Page</title>
  </head>
  <body>
    <h1>Welcome</h1>
    <p>This is a paragraph.</p>
    <a href="https://example.com">Go to Example</a>
    <img src="image.jpg" alt="Example Image">
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
    </ul>
    <button>Click me</button>
    <script src="body-script.js"></script>
  </body>
</html>
```

Common Body Tags:

- `<body>` – Main content area.
- `<h1>` to `<h6>` – Headings.
- `<p>` – Paragraph.
- `<a href="">` – Hyperlink.
- `<img src="" alt="">` – Image.
- `<ul>`, `<ol>`, `<li>` – Lists.
- `<div>` – Block container.
- `<span>` – Inline container.
- `<br>` – Line break.
- `<hr>` – Horizontal line.
- `<form>`, `<input>`, `<button>`, `<label>`, `<textarea>`, `<select>`, `<option>` – User input elements.
- `<video>`, `<audio>` – Multimedia elements.
- `<section>`, `<article>`, `<nav>`, `<header>`, `<footer>`, `<main>` – Semantic structure.

🏷️ ATTRIBUTES

Modify and describe elements.

- `id="uniqueId"` – Identifies element uniquely.

- `class="group"` – Applies CSS class.
- `src=""` – Source of image or script.
- `href=""` – Hyperlink destination.
- `alt=""` – Alternative text for images.
- `title=""` – Tooltip on hover.
- `style=""` – Inline CSS.
- `type`, `name`, `value`, `placeholder`, `required`, `checked`, `disabled`, `readonly` – Used in forms.

CSS STYLING

CSS controls the appearance and layout.

3 Ways to Apply CSS:

- Inline:

```
<p style="color:red">Text</p>
```

- Internal:

```
<head>
<style>p { color:blue; }</style>
</head>
```

- External:

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

CSS SELECTORS

- Element: `p {}` – Targets all `<p>`.
- ID: `#id {}` – Targets element with ID.
- Class: `.class {}` – Targets all with class.
- Universal: `* {}` – Targets all elements.
- Group: `h1, p {}` – Targets multiple.
- Descendant: `div p {}` – Targets `<p>` inside `<div>`.
- Child: `div > p {}` – Direct child.
- Adjacent: `h1 + p {}` – Next sibling.

COMMON CSS PROPERTIES

- `color`, `background-color` – Text and background color.
- `font-size`, `font-family`, `font-weight`, `font-style` – Typography.
- `text-align`, `text-decoration`, `line-height`, `letter-spacing` – Text appearance.
- `margin`, `padding`, `border`, `box-shadow` – Box model.
- `width`, `height`, `max-width`, `min-height` – Sizing.
- `display`, `visibility`, `position`, `z-index`, `overflow` – Layout behavior.
- `cursor`, `opacity`, `transition` – User interaction.

Example:

```css
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  background-color: #fafafa;
  color: #333;
}
h1 {
  color: blue;
  font-size: 32px;
  text-align: center;
}
p {
  font-size: 16px;
  line-height: 1.5;
}
a {
  color: green;
  text-decoration: none;
}
```

BOX MODEL

Explains spacing around elements: `Content → Padding → Border → Margin`

FLEXBOX (Layout)

Aligns items in one direction.

```css
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 20px;
}
```

GRID (Layout)

Creates two-dimensional layouts.

```css
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
```

RESPONSIVE DESIGN

Changes layout based on screen size.

```
@media (max-width: 600px) {
  body {
    background-color: lightgray;
    font-size: 14px;
  }
}
```

BEST PRACTICES

- Use semantic HTML for accessibility and SEO.
- Separate structure (HTML), style (CSS), and behavior (JS).
- Use clear, consistent naming for classes/IDs.
- Comment your code for clarity.
- Avoid inline styles when possible.
- Use relative units (%, em, rem) for responsiveness.
- Minimize repetition with reusable classes.

Next topics to explore:

- HTML Forms and input validation
- CSS transitions, transforms, and animations
- Responsive images and media
- JavaScript basics and DOM manipulation
- Accessibility and semantic roles