

INSTITUTE :- TOPS TECHNOLOGIES (BARODA)

NAME :- SRUJAL SOLANKI M

COURSE :- JAVA FULL STACK DEVELOPER

Module 5) Website Designing - HTML5

HTML5 Tags

1. HTML5 Tags

Definition:

HTML5 introduced many new tags to make the content more semantic and structured.

Common HTML5 Tags:

Tag Purpose

<header> -> Top section of page

<footer> -> Bottom section

<article> -> Independent content

<section> -> Logical section

<aside> -> Side content (ads, related links)

<nav> -> Navigation links

<figure>, <figcaption> -> Image + caption

Example:

<header>

<h1>My Website</h1>

</header>

<nav>

<a href="#">Home</a> | <a href="#">About</a>

</nav>

<section>

<h2>Introduction</h2>

<p>This is a section element.</p>

</section>

<footer>

&copy; 2025 SAM developer

</footer>

2) HTML5 Input and Attribute

2. HTML5 Input Types and Attributes

New Input Types:

Input Type -> Purpose

email -> Email validation

url -> Website URL

tel -> Telephone number

number -> Numeric input

range -> Slider

date, datetime-local -> Date/time

color -> Color picker

New Attributes:

Attribute -> Use

placeholder -> Hint text

required -> Field must be filled

autofocus -> Focuses on load

min, max, step -> Used with number, range

pattern -> Regex validation

Example:

<form>

<input type="email" placeholder="Enter email" required>

<input type="number" min="1" max="100" step="5">

<input type="color">

<input type="range" min="0" max="10">

</form>

3).Audio and Video, Semantic Element in HTML5

3. Audio and Video in HTML5

🔹 Audio Tag:

<audio controls>

<source src="music.mp3" type="audio/mpeg">

Your browser does not support audio.

</audio>

Video Tag:

Example :

<video controls width="320" height="240">

<source src="video.mp4" type="video/mp4">

Your browser does not support video.

</video>

Attributes: controls, autoplay, loop, muted, poster

4. Semantic Elements in HTML5

Semantic = Meaningful

Key Semantic Tags:

<header>, <footer>

<main>, <article>, <section>, <aside>

<nav>, <figure>, <figcaption>

:These tags help screen readers, SEO, and code clarity.

What are Semantic Elements in HTML5?

Definition:

Semantic elements are HTML5 tags that describe the purpose of the content they hold.

They are meaningful tags that help both developers and browsers understand the structure of a web page.

Think of semantics like labeling boxes. Instead of saying “Box A” (non-semantic), you say “Shoes Box” (semantic).

Examples of Semantic vs Non-Semantic Tags:

Non-Semantic Tags Semantic Tags

<div> <header>, <footer>, <section>, <article>, etc.

<span> <mark>, <time>, etc.

Full List of Important Semantic Tags:

Tag Description Can Contain

<header> Page or section header Logo, nav, heading

<nav> Navigation links Links, menus

<main> Main content of the page Articles, sections

<section> Thematic group of content Any block content

<article> Self-contained content Blog, news post

<aside> Sidebar content ds, related links

<footer> Footer content Copyright, links

<figure> Image/media + caption Images, diagrams

<figcaption> Caption for <figure> Text

<mark> Highlighted text Text

<time> Time/date content Date, time

Why Semantic Elements Matter?

For Developers:

Easy to read and maintain code

Clear structure without relying on IDs/classes

For Browsers:

Help with rendering, accessibility, SEO

Screen readers can better understand page structure

For SEO:

Google and other search engines prefer semantic pages for indexing

Improves ranking and visibility

1. <header>

Used for the top part of the page or section.

Can include logo, nav, titles, etc.

<header>

<h1>My Website</h1>

<nav>

<a href="#">Home</a> | <a href="#">About</a>

</nav>

</header>

2. <nav>

Contains main navigation links.

Typically appears inside <header> or <aside>.

<nav>

<ul>

<li><a href="/">Home</a></li>

<li><a href="/blog">Blog</a></li>

</ul>

</nav>

3. <main>

Represents the main content of the document.

There should be only one <main> per page.

<main>

<h2>Welcome to the Blog</h2>

<p>This is the main article.</p>

</main>

4. <section>

Groups related content thematically.

Often has its own <h2> or heading.

<section>

<h2>Features</h2>

<p>We offer fast delivery, great coffee, and cozy ambiance.</p>

</section>

5. <article>

A self-contained piece of content.

Can be shared or reused independently.

<article>

<h3>Power Rangers Return!</h3>

<p>A new Power Rangers movie is coming in 2025...</p>

</article>

6. <aside>

Represents side content not directly related to the main content.

Often used for ads, tips, or links.

<aside>

<h4>Related Articles</h4>

<ul>

<li><a href="#">How to design a comic site</a></li>

</ul>

</aside>

7. <footer>

Bottom section of a page or article.

Used for copyright, author info, links.

<footer>

<p>© 2025 Srujal Solanki | All rights reserved</p>

</footer>

8. <figure> and <figcaption>

Used for images, charts, or code with captions.

<figure>

<img src="power-rangers.png" alt="Power Rangers">

<figcaption>Power Rangers team in 2025.</figcaption>

</figure>

9. <mark>

Used to highlight part of the text.

<p>The <mark>Power Rangers</mark> are heroes.</p>

<time>

Represents time/date information.

<p>Event Date: <time datetime="2025-10-06">October 6, 2025</time></p>

Example:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Srujal's Power Rangers Site</title>

</head>

<body>

<header>

<h1>Power Rangers Universe</h1>

<nav>

<a href="#">Home</a> | <a href="#">Episodes</a> | <a href="#">Gallery</a>

</nav>

</header>

<main>

<section>

<h2>About the Rangers</h2>

<p>The Power Rangers are a team of warriors who fight to save the world.</p>

</section>

<article>

<h3>New Season in 2025</h3>

<p>Coming soon in <time datetime="2025-09-15">September 2025</time>.</p>

</article>

<figure>

<img src="power-rangers.png" alt="Power Rangers Team" width="300">

<figcaption>The Legendary Rangers return</figcaption>

</figure>

<aside>

<h4>Merch Alert!</h4>

<p>Get 10% off on Power Rangers toys using code <mark>RANGER10</mark></p>

</aside>

</main>

<footer>

<p>© 2025 Srujal Solanki | Powered by HTML5</p>

</footer>

</body>

</html>

5).Canvas, Svg

Canvas in HTML5

Definition:

<canvas> is an HTML5 element used to draw graphics, charts, animations, games, etc. using JavaScript.

It creates a blank drawing area on the web page.

Syntax:

<canvas id="myCanvas" width="300" height="150" style="border:1px solid black;"></canvas>

JavaScript Example (Draw a Circle):

<canvas id="myCanvas" width="200" height="200" style="border:1px solid black;"></canvas>

<script>

const canvas = document.getElementById("myCanvas");

const ctx = canvas.getContext("2d");

ctx.beginPath();

ctx.arc(100, 100, 50, 0, 2 \* Math.PI); // x, y, radius

ctx.fillStyle = "skyblue";

ctx.fill();

ctx.stroke();

</script>

Canvas Supports:

2D graphics (default)

WebGL for 3D graphics (advanced)

No internal content: if browser doesn't support canvas, it shows fallback text

2. SVG (Scalable Vector Graphics)

Definition:

<svg> is used to draw vector-based graphics — lines, shapes, and text. It is resolution-independent, so doesn't pixelate when zoomed.

Difference between Canvas vs SVG:

Feature Canvas SVG

Output type Bitmap (pixels) Vector (shapes)

Drawing API JavaScript only HTML/XML based

Performance Better for many objects Slower for large graphics

Interactivity Manual via JS Built-in via HTML DOM

Resolution Pixel-based calable & resolution-free

3. CSS Display Grid

Definition:

CSS Grid Layout is a 2D layout system that lets you design web pages using rows and columns — perfect for complex layouts.

Syntax:

.container {

display: grid;

grid-template-columns: 200px 1fr 1fr;

grid-template-rows: auto auto;

gap: 10px;

}

HTML + Grid Example:

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

gap: 15px;

background-color: lightgray;

padding: 20px;

}

.grid-item {

background-color: white;

padding: 20px;

font-size: 18px;

text-align: center;

border: 2px solid black;

}

</style>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

</div>

Important Properties:

Property Description

display: grid Makes the element a grid container

grid-template-columns Defines column structure

grid-template-rows Defines row structure

gap or grid-gap Adds space between grid items

grid-column, grid-row Places items in custom rows/columns

justify-content Aligns items horizontally

align-items Aligns items vertically

6.)Display Grid

6.)What is CSS Display Grid?

Definition:

CSS Grid is a two-dimensional layout system used to organize content in rows and columns. Unlike Flexbox (which works in one direction), Grid lets you design layouts both horizontally and vertically.

Think of it like creating an Excel sheet layout where you can place elements precisely into cells.

Basic Syntax:

.container {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

grid-template-rows: auto;

}

display: grid – activates grid layout on container.

grid-template-columns – defines the number and width of columns.

grid-template-rows – defines the height of rows.

1. Create Grid Container & Item

<style>

.grid-container {

display: grid;

grid-template-columns: 150px 150px 150px;

grid-template-rows: auto auto;

gap: 10px;

background-color: #eee;

padding: 10px;

}

.grid-item {

background-color: #ffa;

border: 2px solid #333;

text-align: center;

padding: 20px;

font-size: 20px;

}

</style>

<div class="grid-container">

<div class="grid-item">Box 1</div>

<div class="grid-item">Box 2</div>

<div class="grid-item">Box 3</div>

<div class="grid-item">Box 4</div>

<div class="grid-item">Box 5</div>

<div class="grid-item">Box 6</div>

</div>

This creates a 3-column grid with two rows.

2. Units You Can Use in Grid

Unit Meaning

px Fixed pixels

% Percentage of container

fr Fraction of remaining space (flexible)

auto Content size

grid-template-columns: 1fr 2fr; /\* First column 1/3, second 2/3 \*/

3. Grid Gaps (Spacing)

gap: 10px; /\* both row and column gap \*/

row-gap: 20px; /\* only vertical space \*/

column-gap: 30px; /\* only horizontal space \*/

4. Place Items in Specific Cells

.grid-item-1 {

grid-column: 1 / 3; /\* spans from column 1 to 2 (excludes 3) \*/

grid-row: 1 / 2;

}

grid-column: span 2; /\* span across 2 columns \*/

5. Naming Areas (Advanced & Powerful)

.grid-container {

display: grid;

grid-template-areas:

"header header"

"menu content"

"footer footer";

grid-template-columns: 200px 1fr;

grid-template-rows: auto 1fr auto;

}

.header { grid-area: header; }

.menu { grid-area: menu; }

.content { grid-area: content; }

.footer { grid-area: footer; }

🔹 HTML:

<div class="grid-container">

<div class="header">Header</div>

<div class="menu">Menu</div>

<div class="content">Main Content</div>

<div class="footer">Footer</div>

</div>

6. Alignment in Grid

Property Purpose

justify-items Aligns content horizontally

align-items Aligns content vertically

justify-content Aligns entire grid horizontally

align-content Aligns entire grid vertically

.grid-container {

justify-items: center;

align-items: center;

}

7. Responsive Grid with repeat() and auto-fit

.grid-container {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 15px;

}

This allows items to automatically adjust on small screens without writing media queries.

Real-Life Example (Product Cards):

<style>

.products {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

gap: 20px;

}

.product {

background: #fff;

border: 1px solid #ccc;

padding: 15px;

text-align: center;

}

</style>

<div class="products">

<div class="product"> Pizza</div>

<div class="product"> Coffee</div>

<div class="product"> Burger</div>

<div class="product"> Donut</div>

</div>