



# HTML



This is all about HTML

## ▼ Introduction to HTML

### The Story of HTML

#### ◆ Before HTML (The World Without It)

Imagine it's the **1980s**. Computers existed, but the internet was not like today.

- People shared documents through **floppy disks, emails, or local networks**.
- Most documents were **plain text files** or PDFs — no clickable links, no images, no interactive elements.
- If a scientist in India wanted to share research with a scientist in the US, it was a **long, manual process** (download → save → open → maybe not even compatible with the software).

## 👉 The Struggle:

- Documents were isolated islands.
  - No “bridge” to jump from one file to another.
  - No standard format → everyone used their own style.
  - The idea of “browsing” knowledge didn’t exist.
- 

## ◆ The Turning Point (Problem Recognized)

At CERN (a big physics research lab in Switzerland), thousands of scientists worked together.

- They had tons of research papers, but **sharing and connecting them was chaotic**.
- Each department stored files in different systems.
- Finding information was like searching for a needle in a haystack.

## 👉 Example you can relate to:

Imagine if your college notes were stored in 100 different lockers, each requiring a different key.

If you wanted to revise for exams, you’d spend hours just opening lockers instead of studying.

That’s exactly what scientists faced.

---

## ◆ The Fix (The Birth of HTML)

Enter **Sir Tim Berners-Lee** in **1991**.

- He thought: *“What if we could connect documents together, like a web of knowledge?”*
- He created **HTML (HyperText Markup Language)** as a simple way to structure documents.
- Along with it, he invented **HTTP (HyperText Transfer Protocol)** and the first browser.

## 👉 Why HTML was genius:

- It allowed **hyperlinks** (clickable text that jumps to another document).
  - It gave a **universal language** that all computers could understand.
  - It made information **connected and accessible**.
- 

## ◆ Real-Life Analogy

Before HTML → Think of a **library where every book is locked in a separate safe**.

- If you want to read three books, you need three different keys.

After HTML → Imagine the librarian says:

*"Here's one single map. Every book is connected with threads. Just follow the thread, and you'll reach the book you need."*

That's what HTML did → it created the **World Wide Web (WWW)** by connecting documents like a spider's web.

---

## ◆ Evolution of HTML (Why Updates Were Needed)

- **HTML 1.0 (1991)** → Very basic, just text and links.
- **HTML 3.2 / 4.0 (late 90s)** → Added tables, forms, better layouts.
- **Struggle:** Webpages were still ugly and limited. For videos/animations, people needed extra tools like **Flash or Java Applets**.

👉 Example:

Think of an old Nokia phone — you could call and text, but no WhatsApp, no videos, no emojis.

- **HTML5 (2014)** → Big revolution.
  - Added `<video>` , `<audio>` → no need for Flash.
  - Added `<canvas>` for graphics.
  - Introduced **semantic tags** ( `<header>` , `<footer>` ) to make pages more meaningful.
  - Added APIs for storage, geolocation, offline apps.

👉 Example:

It's like moving from Nokia 1100 → iPhone. Same purpose (communication/website), but much smarter, user-friendly, and powerful.

---

## ◆ Why HTML Survived & Won

- Simple to learn.
- Open standard (not owned by a company, but by the **W3C** community).
- Every browser in the world supports it.

👉 Without HTML, there would be no **Google, Facebook, YouTube, Amazon**.  
It's the **skeleton of the internet** — every site you use daily is built on top of it.

---

## ✅ In One Line:

**HTML is the language that transformed disconnected text files into the connected World Wide Web we live in today.**

---

## Simple Definition

"HTML, which stands for HyperText Markup Language, is the core technology of the web used to define the structure and layout of web pages. It uses a system of tags and elements to represent headings, paragraphs, links, images, and multimedia. HTML provides the skeleton of a webpage."

## ▼ HTML Elements, Tags, Attributes

## 📖 HTML Elements, Tags, and Attributes

### ◆ Before This Concept (The Struggle)

When Tim Berners-Lee first created HTML, the goal was simple:

- Scientists needed a way to **organize content** on the web.
- But plain text had no structure → the browser couldn't tell what was a **heading**, what was a **paragraph**, or what was just **normal text**.

👉 Example:

Imagine writing a resume on paper with no bold headings, no bullet points, no formatting. It's all just one long essay — nobody can read it properly.

That was the problem before HTML introduced **tags, elements, and attributes**.

---

## ◆ The Fix (How HTML Solved It)

HTML introduced 3 building blocks to bring order:

### 1 Tags

- Tags are the **labels** that tell the browser how to treat content.
- Example in real life: Think of **labels on food jars**.
  - If the jar says "Jam", you know it's sweet.
  - If the jar says "Pickle", you know it's spicy.
  - Similarly, `<h1>` tells the browser "This is a heading", `<p>` says "This is a paragraph".

👉 **Struggle before tags:** Browsers couldn't differentiate text.

👉 **Fix:** Tags gave **meaning and structure**.

---

### 2 Elements

- An element is the **complete package**: an opening tag + content + closing tag.
- Example in real life: Think of a **sandwich**.
  - You have the **top bread** ( `<tag>` ),
  - The **filling** (content),
  - And the **bottom bread** ( `</tag>` ).

👉 `<p>Hello</p>` = one element.

👉 **Without elements**, web pages would just be unorganized chunks of text.

---

### 3 Attributes

- Attributes are **extra details** that you add to an element to describe it better.
- Example in real life: Think of a **car number plate**.
  - All cars look similar, but the **number plate (attribute)** tells you *which car belongs to whom*.
  - In HTML, `<img>` shows an image, but the `src="image.jpg"` attribute tells *which* image to show.

👉 **Struggle before attributes:**

- You could say "Here is an image", but not *which* image, its size, or description.

👉 **Fix:** Attributes made elements **customizable** and **more powerful**.

---

## ◆ Real-Life Analogy (Bringing it Together)

Think of a **school notebook**:

- **Tags** = Labels you use (Heading, Subheading, Bullet point).
- **Element** = The whole block (Heading + actual written notes).
- **Attribute** = Extra details (Page number, margin notes, highlighter color).

Without them, the notebook would just be a mess of random words. With them, anyone can understand your notes easily.

---

## ◆ Evolution Over Time

- Early HTML (1990s): Used basic tags like `<b>` for bold, `<i>` for italic.
  - **Struggle:** These were only about *style*, not *meaning*.
- Modern HTML5: Introduced **semantic elements** like `<strong>` (important text), `<em>` (emphasized text).
  - **Fix:** Made pages more **meaningful** for search engines, accessibility tools, and users.

👉 Example: Instead of just saying "make text bold" ( `<b>` ), HTML5 says "make text strong/important" ( `<strong>` ). Now even a screen reader can understand the *purpose*, not just the look.

---

### ✓ Summary (Easy to Remember)

- **Tags** = Labels (start and end markers).
- **Elements** = The complete sandwich (tag + content + closing tag).
- **Attributes** = Extra details (like an ID card for that element).

Together, they form the **grammar of the web**.

---

### Definition

👉 "In HTML, a tag is a keyword inside angle brackets that defines how the content should appear, an element is the complete structure formed by tags and content, and an attribute provides additional properties or information about that element."

## 🟢 Practice Questions – HTML Tags, Elements & Attributes

---

### Q1. Identify the Tag, Element, and Attribute

```
<h1>Welcome to My Website</h1>
```

- Tag = ?
- Element = ?
- Attribute = ?

#### ▼ Answer

- Tag = `<h1>` and `</h1>`
  - Element = `<h1>Welcome to My Website</h1>`
  - Attribute = None
- 

### Q2. Identify the Tag, Element, and Attribute

```

```

- Tag = ?
- Element = ?
- Attribute = ?

▼ Answer

- Tag = `<img>`
- Element = ``
- Attribute = `src`, `alt`, `width`

### Q3. Identify the Tag, Element, and Attribute

```
<a href="https://google.com">Go to Google</a>
```

- Tag = ?
- Element = ?
- Attribute = ?

▼ Answer

- Tag = `<a>` and `</a>`
- Element = `<a href="https://google.com">Go to Google</a>`
- Attribute = `href`

### Q4. Self-Closing Element

```
<br>
```

- Is this a Tag or Element?
- Does it have Attributes?



▼ Answer

- It is a **self-closing Element**.
- It does not have attributes (but attributes can be added in some self-closing tags, e.g., `<img>` ).

## Q5. Identify from Mixed Example

```
<p style="color:red;">This is a red paragraph.</p>
```

- Tag = ?
- Element = ?
- Attribute = ?

▼ Answer

- Tag = `<p>` and `</p>`
- Element = `<p style="color:red;">This is a red paragraph.</p>`
- Attribute = `style`

## ◆ Part A: Multiple Choice Questions (MCQs)

Q1. Which of the following is a self-closing tag?

- a) `<h1>`
- b) `<p>`
- c) `<br>`
- d) `<div>`

▼ Answer

`<br>`

Q2. Which part of an HTML element provides extra information?

- a) Tag
- b) Element

- c) Attribute
- d) Content

▼ Answer

**c) Attribute**

---

**Q3. In the code `` , which is the attribute?**

- a) `img`
- b) `src` and `alt`
- c) `car.jpg`
- d) `Car`

▼ Answer

**b) `src` and `alt`**

---

**Q4. Which of the following is NOT a valid HTML tag?**

- a) `<table>`
- b) `<span>`
- c) `<section>`
- d) `<heading>`

▼ Answer

**d) `<heading>`**

---

**Q5. An HTML element usually consists of:**

- a) Opening tag + Content + Closing tag
- b) Opening tag only
- c) Closing tag only
- d) Content only

▼ Answer

**a) Opening tag + Content + Closing tag**

---

## ◆ Part B: Fill in the Blanks

Q1. In `<p>Hello World</p>`, the word **Hello World** is the \_\_\_\_\_.

▼ Answer

Content

Q2. The extra information written inside the opening tag is called \_\_\_\_\_.

▼ Answer

**Attribute**

Q3. `<h1>HTML Quiz</h1>` is an example of an HTML \_\_\_\_\_.

▼ Answer

Element

Q4. The tag used to insert an image is \_\_\_\_\_.

▼ Answer

`<img>`

Q5. In `<a href="https://google.com">Google</a>`, `href` is an \_\_\_\_\_.

▼ Answer

**Attribute**

## ▼ HTML Structure, Headings, Paragraphs, Text Formatting

### ◆ 1. HTML Document Structure

Every HTML page follows a **hierarchy** — like a family tree.

```
<!DOCTYPE html> → Defines HTML version (HTML5 here).  
<html>           → Root of the page  
  <head>         → Metadata (info *about* the page, not visible directly)  
  <body>         → Actual content (what user sees)  
</html>
```

👉 Think of it as:

- **Head** = "Backstage" (title, SEO, CSS, JS).
- **Body** = "Stage" (the visible performance).

If you miss proper structure, your page may still run, but it will **break in SEO, accessibility, or future scaling**.

---

## ◆ 2. Headings ( `<h1>` to `<h6>` )

Headings give **hierarchy to content**.

- `<h1>` = Most important heading.
- `<h6>` = Least important heading.

👉 Rules & Best Practices:

- Use only **one** `<h1>` **per page** → It defines the main topic.
- Sub-sections should use `<h2>`, `<h3>`, etc. (like chapters and subchapters).
- Don't use headings for styling (use CSS for size). Use them **only for meaning**.

👉 Real-life analogy:

Think of a **book**:

- Title → `<h1>`
- Chapter name → `<h2>`
- Section → `<h3>`
- Subsection → `<h4>`

This structure makes content **SEO-friendly** and easier for screen readers.

---

## ◆ 3. Paragraphs ( `<p>` )

- A `<p>` defines a **block of text**.
- Browsers automatically add **margin (spacing)** before and after paragraphs.

- You should never replace `<p>` with `<br>` (line break) for spacing.

👉 Why important?

Because `<p>` is semantic → search engines and accessibility tools understand it as **paragraphs of meaning**, not just random text.

---

## ◆ 4. Text Formatting Tags

HTML provides **inline tags** to style or add meaning to text.

### Common Formatting Tags:

- `<b>` → Bold (just makes text thick, no meaning).
- `<strong>` → Important (semantic emphasis, better for SEO).
- `<i>` → Italic (slanted text, visual only).
- `<em>` → Emphasis (semantic, conveys stress/importance).
- `<u>` → Underline.
- `<mark>` → Highlighted text (yellow by default).
- `<small>` → Smaller text.
- `<sup>` → Superscript (e.g.,  $X^2$ ).
- `<sub>` → Subscript (e.g.,  $H_2O$ ).
- `<del>` → Deleted/strikethrough text.
- `<ins>` → Inserted/added text (often underlined).

👉 Golden Rule:

Use **semantic tags** (`<strong>`, `<em>`) over presentational ones (`<b>`, `<i>`).

Because **semantic tags carry meaning**, which helps SEO and accessibility.

---

## ◆ 5. Real-Life Example of Proper Use

Imagine you're writing a **news article** on a website:

- Title of news → `<h1>`

- Subtitle (category: "Sports") → `<h2>`
- Article text → `<p>`
- Important quote → `<strong>`
- Date published → `<small>`
- Player's chemical formula for sports drink → `H<sub>2</sub>O`

👉 This makes content not only look good but also **organized and meaningful** to Google, screen readers, and users.

---

## ◆ 6. Common Mistakes Beginners Make

- ✗ Using `<br>` everywhere instead of `<p>`.
- ✗ Using `<h1>` multiple times just for bigger fonts.
- ✗ Using `<b>` everywhere instead of `<strong>`.
- ✗ Using headings for styling instead of structure.

👉 Fix: **Always use tags for their purpose, not for looks.** Use CSS for styling.

---

### ✅ Summary (Quick Recall)

- **Structure:** `<head>` (meta info), `<body>` (visible content).
  - **Headings:** `<h1>` to `<h6>`, one `<h1>` only.
  - **Paragraphs:** `<p>` = blocks of text.
  - **Text Formatting:** Use semantic tags (`<strong>`, `<em>`) for meaning, not just looks.
- 

## 🟢 Interactive HTML Quiz (With Answers)

? Q1. What does `<!DOCTYPE html>` do?

▼ Answer

✅ **Answer:** It tells the browser that the document is written in **HTML5**.

---

? Q2. Which tag shows the text in the **browser tab** (top)?

- `<head>`
- `<title>`
- `<body>`

▼ Answer

✓ Answer: `<title>`

---

? Q3. Where do we write the content that is **visible to users**?

👉 Example: headings, paragraphs, images, etc.

▼ Answer

✓ Answer: Inside the `<body>` tag.

---

? Q4. Which is the **biggest heading** in HTML?

👉 `<h1>` to `<h6>` — kaunsa sabse bada hai?

▼ Answer

✓ Answer: `<h1>`

---

? Q5. How do we write a **paragraph** in HTML?

▼ Answer

```
<p>This is a paragraph</p>
```

---

? Q6. Which tag highlights text like a **marker pen**?

▼ Answer

✓ Answer: `<mark>`

---

? Q7. Write the correct tag for **Water Formula (H<sub>2</sub>O)**

▼ Answer

H<sub>2</sub>O

? Q8. Which tag shows **deleted text with strike-through**?

▼ Answer

✓ Answer: `<del>`

? Q9. What is the difference between `<b>` and `<strong>` ?

▼ Answer

✓ Answer:

- `<b>` → Sirf text ko bold karta hai (styling).
- `<strong>` → Text ko bold + meaningful importance deta hai (SEO & screen readers ke liye useful).

? Q10. Fill in the blanks:

1. The `<__>` tag is the root of an HTML document.

▼ Answer

✓ Answer: `<html>`

1. The `<__>` tag is used for paragraphs.

▼ Answer

✓ Answer: `<p>`

1. The smallest heading is `<__>`.

▼ Answer

✓ Answer: `<h6>`

▼ **Links, Anchors, Navigation Structure**



# Links, Anchors & Navigation Structure

---

## ◆ 1. Links ( `<a>` ) – The Soul of the Web

The `<a>` tag (anchor) is what makes the web a “web”.

- Without links, websites would just be disconnected pages.
- With links, you can jump between pages, sections, downloads, emails, and even other websites.

👉 Real-life analogy:

Think of **roads between cities**. A city without roads is isolated. A website without links is the same. Links are the **roads of the internet**.

---

## ◆ 2. Anatomy of an Anchor Tag

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

- `<a>` → Defines an anchor (a clickable link).
- `href` (**hyperlink reference**) → The destination.
- **Link text** → The clickable part users see.

👉 If `href` is missing, the anchor is meaningless (like a signboard with no road).

---

## ◆ 3. Types of Links

### 1 Absolute Links (Full URL)

```
<a href="https://google.com">Google</a>
```

- Used for **external websites**.

👉 Example: A road to another city.

---

### 2 Relative Links (Within the same website)

```
<a href="/about.html">About Us</a>
```

- Used for linking **pages of the same project**.

👉 Example: A road to another area inside your city.

### 3 Anchor/Jump Links (Within the same page)

```
<a href="#section2">Go to Section 2</a>
...
<h2 id="section2">This is Section 2</h2>
```

- Helps users **jump to a section** of the same page.

👉 Example: A lift inside a building that takes you directly to the 5th floor.

### 4 Email Links

```
<a href="mailto:info@example.com">Email Us</a>
```

- Opens email client with a pre-filled address.

👉 Like pressing a “call button” for a specific person.

### 5 Download Links

```
<a href="resume.pdf" download>Download Resume</a>
```

- Forces file download instead of opening it.

👉 Like a **download button** in an app.

## ◆ 4. Navigation Structure ( `<nav>` )

- The `<nav>` element defines the **navigation section** of a website.
- Usually contains a list of `<a>` links to other pages or sections.
- Helps both **users** and **search engines** understand site structure.

👉 Example of proper navigation:

```
<nav>
  <a href="index.html">Home</a>
  <a href="about.html">About</a>
  <a href="services.html">Services</a>
  <a href="contact.html">Contact</a>
</nav>
```

👉 Real-life analogy:

Think of **signboards at a highway intersection** telling you where to go.

Without a `<nav>`, users are like drivers with no map.

## ◆ 5. Best Practices for Links & Navigation

- ✅ Use **meaningful text** ("Contact Us") → ❌ Never "Click Here".
- ✅ Always use **relative paths** for internal links (portable between servers).
- ✅ For external links, add `target="_blank"` (open in new tab).
- ✅ Keep `<nav>` clean — only main links, not random stuff.
- ✅ Use IDs + anchors for **Table of Contents** (great for long pages).

## ◆ 6. Real-World Example

Imagine an **e-commerce website**:

- Top navigation ( `<nav>` ) → Home | Shop | Cart | Contact.
- Product details page → "Buy Now" link to checkout page.
- Footer → Email link for support.
- Long product description → Jump links to "Specs", "Reviews", "FAQs".

👉 Without these links, the site would feel like **a mall without signboards**.

### ✅ Summary (Quick Recall)

- `<a>` = Anchor tag for links.

- `href` = Destination (URL, file, ID, mailto).
- Types = Absolute, Relative, Anchors, Email, Download.
- `<nav>` = Defines website's navigation section.
- Best practice → Always use **clear, meaningful link text**.

```
<!DOCTYPE html>
<head>
  <title>Links & Navigation Example</title>
</head>
<body>

  <!-- Navigation Section →
  <header>
    <h1>My Demo Website</h1>
    <nav>
      <a href="index.html">Home</a> |
      <a href="about.html">About</a> |
      <a href="services.html">Services</a> |
      <a href="contact.html">Contact</a>
    </nav>
    <hr>
  </header>

  <!-- Absolute Link →
  <section>
    <h2>1. Absolute Link</h2>
    <p>Visit <a href="https://www.google.com" target="_blank">Google</a>
  > (opens in new tab).</p>
  </section>

  <!-- Relative Link →
  <section>
    <h2>2. Relative Link</h2>
    <p>Go to our <a href="about.html">About Us</a> page (within same pr
```

```

object).</p>
</section>

<!-- Jump / Anchor Link →
<section>
  <h2>3. Jump Link</h2>
  <p>Scroll down to <a href="#faq">FAQ Section</a> directly.</p>
</section>

<!-- Email Link →
<section>
  <h2>4. Email Link</h2>
  <p>Contact us: <a href="mailto:support@example.com">Email Support
</a></p>
</section>

<!-- Download Link →
<section>
  <h2>5. Download Link</h2>
  <p><a href="resume.pdf" download>Download Resume (PDF)</a></p>
</section>

<hr>

<!-- Long Content for Jump Link Demo →
<section>
  <h2>Article</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
cumsan ...</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
cumsan ...</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
cumsan ...</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
cumsan ...</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac

```

```
cumsan ...</p>
</section>

<!-- FAQ Section (target of jump link) →
<section id="faq">
  <h2>FAQ Section</h2>
  <p><strong>Q:</strong> What is this site?<br>
    <strong>A:</strong> A demo of links and navigation.</p>
</section>

</body>
</html>
```



## Quiz: Links, Anchors & Navigation

### Part A: Multiple Choice Questions

**Q1.** In HTML, which tag is used to create a hyperlink?

- a) `<link>`
- b) `<a>`
- c) `<href>`
- d) `<nav>`

▼ Answer

- b) `<a>`

---

**Q2.** What does the `href` attribute represent in the `<a>` tag?

- a) The clickable text
- b) The style of the link
- c) The destination URL or location
- d) Opens the link in a new tab

▼ Answer

c) The destination URL or location

---

**Q3.** Which `target` value is used to open a link in a new tab?

- a) `_new`
- b) `_self`
- c) `_blank`
- d) `_tab`

▼ Answer

- c) `_blank`
- 

**Q4.** The `<nav>` element in HTML is mainly used for:

- a) Inserting an image gallery
- b) Creating navigation links
- c) Adding comments
- d) Embedding audio

▼ Answer

- b) Creating navigation links
- 

**Q5.** Which attribute is used to jump to a specific section of the same page?

- a) `class`
- b) `id`
- c) `style`
- d) `target`

▼ Answer

- b) `id`
- 

## Part B: Practical Coding Questions

**Q6.** Write an HTML link that opens **Google** in the same tab.

▼ Answer

```
<a href="https://www.google.com">Google</a>
```

**Q7.** Write a link that opens **Facebook** in a new tab.

▼ Answer

```
<a href="https://www.facebook.com" target="_blank">Facebook</a>
```

**Q8.** Create a navigation bar with links → Home, Services, Contact.

▼ Answer

```
<nav>  
  <a href="index.html">Home</a> |  
  <a href="services.html">Services</a> |  
  <a href="contact.html">Contact</a>  
</nav>
```

**Q9.** Create a link on a page that jumps directly to a section with `id="about"`.

▼ Answer

```
<a href="#about">Go to About Section</a>  
  
<h2 id="about">About Section</h2> ✓ Answer:
```

**Q10.** Make a link that sends an email to `hello@example.com`.

▼ Answer

```
<a href="mailto:hello@example.com">Send Email</a>
```

## ▼ Ordered & Unordered Lists



## Ordered & Unordered Lists



---

## ◆ 1. Unordered Lists ( `<ul>` )

- Represent a **collection of items without sequence**.
- Default: bullets (•) as markers.

👉 Real-life analogy:

Think of a **grocery shopping list**:

- Milk
- Bread
- Eggs

The order doesn't matter — you can pick in any sequence.

```
<ul>
  <li>Milk</li>
  <li>Bread</li>
  <li>Eggs</li>
</ul>
```

---

## ◆ 2. Ordered Lists ( `<ol>` )

- Represent items in a **specific order**.
- Default: numbers (1, 2, 3).

👉 Real-life analogy:

Think of a **recipe**:

1. Boil water
2. Add pasta
3. Drain and serve

The sequence matters, or else it's wrong.

```
<ol>
  <li>Boil water</li>
  <li>Add pasta</li>
  <li>Drain and serve</li>
</ol>
```

### ◆ 3. Types of Ordered Lists

You can control numbering style using the `type` attribute.

- `type="1"` → Numbers (1, 2, 3...) (default)
- `type="A"` → Uppercase letters (A, B, C...)
- `type="a"` → Lowercase letters (a, b, c...)
- `type="I"` → Roman numerals (I, II, III...)
- `type="i"` → Lowercase Roman numerals (i, ii, iii...)

👉 Example:

```
<ol type="I">
  <li>Introduction</li>
  <li>Body</li>
  <li>Conclusion</li>
</ol>
```

### ◆ 4. Nested Lists

You can nest lists inside one another for subcategories.

👉 Example:

```
<ul>
  <li>Fruits
    <ul>
      <li>Apple</li>
    </ul>
  </li>
</ul>
```

```
<li>Banana</li>
</ul>
</li>
<li>Vegetables
  <ul>
    <li>Carrot</li>
    <li>Tomato</li>
  </ul>
</li>
</ul>
```

👉 Real-life analogy:

- Main shopping categories → Fruits, Vegetables.
- Sub-items → Apple, Banana, Carrot, Tomato.

## ◆ 5. Attributes You Should Know

- **start** → Define where an ordered list starts.

```
<ol start="5">
  <li>Step Five</li>
  <li>Step Six</li>
</ol>
```

👉 Output: 5. Step Five, 6. Step Six.

- **reversed** → Counts backward.

```
<ol reversed>
  <li>Last Step</li>
  <li>Middle Step</li>
  <li>First Step</li>
</ol>
```

👉 Output: 3. Last Step, 2. Middle Step, 1. First Step.

## ◆ 6. Real-World Uses of Lists

### 1. Unordered Lists ( `<ul>` )

- Navigation menus.
- Features list on product pages.
- FAQs with bullets.

### 2. Ordered Lists ( `<ol>` )

- Step-by-step guides (recipes, instructions).
- Ranking items (Top 10 movies).
- Processes where sequence matters.

---

### ✓ Summary (Quick Recall)

- `<ul>` = unordered list (bullets, no sequence).
- `<ol>` = ordered list (numbers/letters/roman, sequence matters).
- `<li>` = list item (used in both).
- Attributes: `type` , `start` , `reversed` .
- Supports nesting → create sub-lists.

```
<!DOCTYPE html>
<head>
  <title>HTML Lists Example</title>
</head>
<body>

  <h1>HTML Lists Demo</h1>

  <!-- Unordered List →
  <h2>Shopping List (Unordered)</h2>
  <ul>
    <li>Milk</li>
    <li>Bread</li>
```

```

    <li>Eggs</li>
</ul>

<!-- Ordered List →
<h2>Recipe Steps (Ordered)</h2>
<ol>
    <li>Boil water</li>
    <li>Add pasta</li>
    <li>Drain and serve</li>
</ol>

<!-- Ordered List with Roman numerals →
<h2>Exam Sections (Ordered with Roman numerals)</h2>
<ol type="I">
    <li>Maths</li>
    <li>Science</li>
    <li>English</li>
</ol>

<!-- Reversed Ordered List →
<h2>Countdown (Reversed)</h2>
<ol reversed>
    <li>Three</li>
    <li>Two</li>
    <li>One</li>
</ol>

<!-- Nested List →
<h2>Categories (Nested List)</h2>
<ul>
    <li>Fruits
        <ul>
            <li>Apple</li>
            <li>Banana</li>
        </ul>
    </li>

```

```
<li>Vegetables
  <ul>
    <li>Carrot</li>
    <li>Tomato</li>
  </ul>
</li>
</ul>

</body>
</html>
```

## Student Task

👉 Create a webpage using **lists** that looks like this:

### Website Outline

#### 1. Home

#### 2. About

- Our Team
- Our Story

#### 3. Services

- Web Development
- Mobile Development
- Cloud Solutions

#### 4. Contact

✓ Use **ordered list** for main sections (Home, About, Services, Contact).

✓ Use **unordered list** for sub-sections (Team, Story, etc.).

▼ Answer

```
<!DOCTYPE html>
<head>
  <title>Website Outline with Lists</title>
</head>
<body>

  <h1>Website Outline</h1>

  <ol>
    <li>Home</li>
    <li>About
      <ul>
        <li>Our Team</li>
        <li>Our Story</li>
      </ul>
    </li>
    <li>Services
      <ul>
        <li>Web Development</li>
        <li>Mobile Development</li>
        <li>Cloud Solutions</li>
      </ul>
    </li>
    <li>Contact</li>
  </ol>

</body>
</html>
```

## ▼ 🟢 Mini Project

### 🎯 Mini Project Idea → **Personal Portfolio (Basic Version)**

---

# ◆ Requirements for the Project

## 1. Page Structure

- Title: "My Portfolio"
- A heading ( `<h1>` ) with student's name.
- A paragraph introducing themselves.

## 2. Navigation Menu (using links `<a>` and list `<ul>` )

- Home
- About Me
- Skills
- Contact

## 3. About Me Section

- A heading ( `<h2>` )
- Few lines in paragraph form.

## 4. Skills Section

- Use an **unordered list** for skills.
- Example: HTML, CSS, JavaScript.

## 5. Experience Section

- Use an **ordered list** (timeline of steps or jobs).

## 6. Contact Section

- Add an **email link** ( `mailto:` ).
- Add a **download link** (fake CV file like `resume.pdf` ).

## 7. Extra Challenge (Optional)

- Add a **jump link** in nav → "Back to Top".

---

### ▼ Answer



```

<!DOCTYPE html>
<head>
  <title>My Portfolio</title>
</head>
<body>

  <!-- Navigation →
  <header>
    <h1>John Doe - Portfolio</h1>
    <nav>
      <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About Me</a></li>
        <li><a href="#skills">Skills</a></li>
        <li><a href="#experience">Experience</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </nav>
    <hr>
  </header>

  <!-- Home →
  <section id="home">
    <h2>Welcome!</h2>
    <p>Hello! I'm John Doe, a beginner web developer learning HTML and web design.</p>
  </section>

  <!-- About →
  <section id="about">
    <h2>About Me</h2>
    <p>I am passionate about coding and building creative projects. I enjoy solving real-world problems through technology.</p>
  </section>

```

```

<!-- Skills →
<section id="skills">
  <h2>My Skills</h2>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
  </ul>
</section>

<!-- Experience →
<section id="experience">
  <h2>My Journey</h2>
  <ol>
    <li>Started learning coding in 2024</li>
    <li>Built small HTML projects</li>
    <li>Now practicing full-stack development</li>
  </ol>
</section>

<!-- Contact →
<section id="contact">
  <h2>Contact Me</h2>
  <p>Email: <a href="mailto:johnndoe@example.com">Send me an Email</a></p>
  <p><a href="resume.pdf" download>Download My Resume</a></p>
  <p><a href="#home">Back to Top</a></p>
</section>

</body>
</html>

```

## ▼ Tables: Creation, Nesting

# Topic: HTML Tables

## ◆ What is a Table?

A **table** in HTML is used to display data in rows and columns (just like an Excel sheet).

---

## 1. Basic Table Elements

- `<table>` → the main container for the table.
- `<tr>` → **table row** (creates a row).
- `<th>` → **table header cell** (used for headings, by default bold & centered).
- `<td>` → **table data cell** (used for normal data).

### 👉 Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Basic Table Example</title>
</head>
<body>
  <h2>Student Marks Table</h2>

  <table border="1" cellpadding="8" cellspacing="0">
    <tr>
      <th>Name</th>
      <th>Subject</th>
      <th>Marks</th>
    </tr>
    <tr>
      <td>Karan</td>
      <td>Math</td>
      <td>90</td>
    </tr>
```

```

<tr>
  <td>Riya</td>
  <td>Science</td>
  <td>85</td>
</tr>
</table>
</body>
</html>

```

✅ Here we created a **3-column table** with students' names, subjects, and marks.

## 2. Merging Cells: Colspan & Rowspan

- **colspan** → merges cells horizontally (across columns).
- **rowspan** → merges cells vertically (across rows).

👉 **Example:**

```

<table border="1" cellpadding="8" cellspacing="0">
  <tr>
    <th>Name</th>
    <th colspan="2">Subjects</th>
  </tr>
  <tr>
    <td>Karan</td>
    <td>Math</td>
    <td>Science</td>
  </tr>
  <tr>
    <td rowspan="2">Riya</td>
    <td>English</td>
    <td>History</td>
  </tr>
  <tr>
    <td>Biology</td>

```

```
<td>Geography</td>
</tr>
</table>
```

✓ In this example:

- The heading "Subjects" covers **two columns** ( `colspan="2"` ).
- Riya's name covers **two rows** ( `rowspan="2"` ).

### 3. Nesting Tables

Sometimes you can put a **table inside another table cell** → this is called **nested table**.

👉 Example:

```
<table border="1" cellpadding="8" cellspacing="0">
  <tr>
    <th>Employee</th>
    <th>Details</th>
  </tr>
  <tr>
    <td>John</td>
    <td>
      <table border="1" cellpadding="5" cellspacing="0">
        <tr>
          <th>Age</th>
          <th>Department</th>
        </tr>
        <tr>
          <td>28</td>
          <td>HR</td>
        </tr>
      </table>
    </td>
```

```
</tr>
</table>
```

✓ Here, the **inner table** is placed inside the **Details cell** of the main table.

## ▼ Forms: Input Fields, Buttons, Select, Textarea

### Topic: HTML Forms (Inputs, Buttons, Select, Textarea)

#### ◆ What is a Form?

A **form** is used to **collect data from the user** and send it to a server.

Example: Google login, Amazon search box, Feedback form → all are HTML forms.

The form is created using the `<form>` tag.

👉 Basic Structure:

```
<form action="submit.php" method="post">
  <!-- form elements go here -->
</form>
```

- `action` → where the form data will be sent (server file).
- `method` → how data is sent:
  - `GET` → shows data in the URL (use for search, filters).
  - `POST` → hides data (use for login, signup, secure info).

## 1. Input Fields

Input fields are created with `<input>` tag.

### Types of Inputs:

- `type="text"` → single-line text

- `type="password"` → hidden characters (••••)
- `type="email"` → email format validation
- `type="number"` → only numbers allowed
- `type="date"` → calendar input
- `type="radio"` → select **one option** from a group
- `type="checkbox"` → select **multiple options**
- `type="file"` → upload a file

👉 Example:

```
<form>
  Name: <input type="text" name="username"><br><br>
  Password: <input type="password" name="pass"><br><br>
  Email: <input type="email" name="mail"><br><br>
  Age: <input type="number" name="age"><br><br>
  Birthday: <input type="date" name="dob"><br><br>

  Gender:
  <input type="radio" name="gender" value="male"> Male
  <input type="radio" name="gender" value="female"> Female <br><br>

  Hobbies:
  <input type="checkbox" name="hobby" value="reading"> Reading
  <input type="checkbox" name="hobby" value="sports"> Sports <br><br>
  >

  Upload Resume: <input type="file" name="resume"><br><br>
</form>
```

## 2. Buttons

Buttons are used to submit or reset form data.

- `<input type="submit">` → submits form

- `<input type="reset">` → resets form values
- `<button>` → customizable button

👉 Example:

```
<form>
  <input type="submit" value="Submit">
  <input type="reset" value="Clear">
  <button type="button">Click Me</button>
</form>
```

### 3. Select (Dropdown Menu)

Dropdowns allow users to choose one option from a list.

👉 Example:

```
<form>
  Country:
  <select name="country">
    <option value="india">India</option>
    <option value="usa">USA</option>
    <option value="uk">UK</option>
  </select>
</form>
```

✅ You can also add `multiple` attribute to allow multiple selections.

### 4. Textarea

For large multi-line text input (like feedback or address).

👉 Example:

```
<form>
  Address:<br>
```



```
<textarea name="address" rows="4" cols="30"></textarea>
</form>
```

## Real-life Connection for Students

- **Text field** → writing your name while signing up.
- **Password** → login screen.
- **Radio button** → choosing gender, payment option.
- **Checkbox** → selecting multiple hobbies/interests.
- **Dropdown** → selecting country/state while creating account.
- **Textarea** → writing feedback on a shopping website.
- **Submit button** → "Login" / "Sign up" / "Search" button.

## ▼ **Form Validation (Required, Patterns)**

### Topic: Form Validation (Required, Patterns)

#### ◆ What is Form Validation?

Form validation means **checking user input before sending it to the server**.

👉 Example:

- If user leaves the "Email" empty → show error.
- If user types wrong phone number format → reject.

This prevents **wrong data entry** and improves **user experience**.

## 1. Required Attribute

- If you add `required` to any input field, the user **must fill it before submitting**.

👉 Example:

```
<form>
  Name: <input type="text" name="username" required><br><br>
  Email: <input type="email" name="mail" required><br><br>
  <input type="submit" value="Submit">
</form>
```

✅ If the user tries to submit without filling → browser shows a warning.

## 2. Pattern Attribute

- The `pattern` attribute allows us to set a **regular expression (Regex)** → meaning, data must match that pattern.

👉 Example: Phone number (10 digits only)

```
<form>
  Phone: <input type="text" name="phone" pattern="[0-9]{10}" required>
  <input type="submit" value="Submit">
</form>
```

- `pattern="[0-9]{10}"` → only numbers, and exactly 10 digits.

## 3. More Useful Patterns

- Email (basic check)

```
<input type="email" name="mail" required>
```

- Password (at least 6 chars, 1 number)

```
<input type="password" name="pass" pattern="(?=.*\d).{6,}" required>
```

✅ Explanation:

- `(?=.*\d)` → must contain a digit

- `{6,}` → minimum 6 characters
- **PIN code (6 digits)**

```
<input type="text" name="pin" pattern="[0-9]{6}" required>
```

- **Username (only letters, min 3 chars)**

```
<input type="text" name="uname" pattern="[A-Za-z]{3,}" required>
```

## 4. Real-life Connection

- Required field = *Teacher won't accept empty answer sheet.*
- Pattern = *You must write your roll number in correct format (only digits, fixed length).*
- Email/Phone = *Apps won't let you register with wrong info.*
- Password validation = *Forcing you to keep a strong password.*

✓ By default, modern browsers show error messages.

👉 You can also add `title` attribute to explain what's expected.

Example:

```
<input type="password"
      pattern="(?=.*\d){6,}"
      title="Must contain at least 6 characters including a number" required
>
```

## ▼ Mini Project



## Practice Questions (Form Validation)

Q1.

Create a **Login Form** with:

- Email (required, valid email format)
  - Password (required, min 6 characters)
  - Submit button
- 

## Q2.

Create a **Contact Form** with:

- Name (only letters, at least 3 chars, required)
  - Phone (10-digit number only, required)
  - Message (textarea, required)
  - Submit & Reset buttons
- 

## Q3.

Create a **Job Application Form** with:

- Full Name (letters only, required)
  - Email (required)
  - Resume Upload (required file input)
  - Gender (radio buttons, required)
  - Skills (checkboxes, at least one should be checked)
  - Submit button
- 

## Q4.

Create a **Student Feedback Form** with:

- Student Name (required)
- Class Dropdown (choose from 6th–12th, required)
- Feedback Textarea (at least 20 characters using `pattern` )
- Submit button

▼ Answer

## ✓ Q1: Login Form

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
</head>
<body>
  <h2>Login Form</h2>
  <form>
    Email:
    <input type="email" name="email" required><br><br>

    Password:
    <input type="password" name="password"
      pattern=".{6,}"
      title="Password must be at least 6 characters"
      required><br><br>

    <input type="submit" value="Login">
  </form>
</body>
</html>
```

## ✓ Q2: Contact Form

```
<!DOCTYPE html>
<html>
<head>
  <title>Contact Form</title>
</head>
```

```

<body>
  <h2>Contact Form</h2>
  <form>
    Name:
    <input type="text" name="name"
      pattern="[A-Za-z]{3,}"
      title="At least 3 letters, alphabets only"
      required><br><br>

    Phone:
    <input type="text" name="phone"
      pattern="[0-9]{10}"
      title="Enter a valid 10-digit phone number"
      required><br><br>

    Message:<br>
    <textarea name="message" rows="4" cols="30" required></textare
a><br><br>

    <input type="submit" value="Send">
    <input type="reset" value="Clear">
  </form>
</body>
</html>

```

## Q3: Job Application Form

```

<!DOCTYPE html>
<html>
  <head>
    <title>Job Application Form</title>
  </head>
  <body>
    <h2>Job Application Form</h2>

```

```

<form>
  Full Name:
  <input type="text" name="fullname"
    pattern="[A-Za-z ]{3,}"
    title="Only letters, at least 3 characters"
    required><br><br>

  Email:
  <input type="email" name="email" required><br><br>

  Upload Resume:
  <input type="file" name="resume" required><br><br>

  Gender:
  <input type="radio" name="gender" value="male" required> Male
  <input type="radio" name="gender" value="female"> Female<br><br>

  Skills:
  <input type="checkbox" name="skills" value="html"> HTML
  <input type="checkbox" name="skills" value="css"> CSS
  <input type="checkbox" name="skills" value="js"> JavaScript<br><br>

  <input type="submit" value="Apply">
</form>
</body>
</html>

```

## Q4: Student Feedback Form

```

<!DOCTYPE html>
<html>
<head>

```

```

<title>Student Feedback Form</title>
</head>
<body>
  <h2>Student Feedback Form</h2>
  <form>
    Student Name:
    <input type="text" name="sname" required><br><br>

    Class:
    <select name="class" required>
      <option value="">--Select Class--</option>
      <option value="6">6th</option>
      <option value="7">7th</option>
      <option value="8">8th</option>
      <option value="9">9th</option>
      <option value="10">10th</option>
      <option value="11">11th</option>
      <option value="12">12th</option>
    </select><br><br>

    Feedback:<br>
    <textarea name="feedback" rows="4" cols="30"
      pattern=".{20,}"
      title="Feedback must be at least 20 characters"
      required></textarea><br><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>

```

✅ These four cover almost **all validation cases**:

- `required`
- `pattern` with RegEx



- `type="email"`
- Dropdowns, radio, checkbox, textarea

## ▼ Media Tags: Audio, Video, Embeds

### Topic: Media Tags (Audio, Video, Embed)

#### ◆ Why Media Tags?

Earlier, we needed plugins (like Flash) to play songs or videos.

But **HTML5 introduced** `<audio>` and `<video>` tags, making it super easy to add media.

### 1. Audio Tag

Used to play sound/music.

👉 Syntax:

```
<audio controls>
  <source src="song.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

- `controls` → adds play, pause, volume options.
- `autoplay` → plays automatically (not always allowed by browsers).
- `loop` → repeats the audio.

✅ Example:

```
<audio controls loop>
  <source src="song.mp3" type="audio/mpeg">
</audio>
```

## 2. Video Tag

Used to display videos.

👉 Syntax:

```
<video width="400" controls>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

- `width` / `height` → sets size.
- `controls` → play, pause, volume, fullscreen.
- `autoplay` → plays automatically.
- `loop` → repeats video.
- `poster="image.jpg"` → shows an image before video starts.

✅ Example:

```
<video width="400" height="250" controls poster="thumbnail.jpg">
  <source src="trailer.mp4" type="video/mp4">
</video>
```

## 3. Embed Tag

Used to embed **any external content** (like PDFs, YouTube, maps).

👉 Example 1: Embed a PDF

```
<embed src="document.pdf" width="600" height="400" type="application/pdf">
```

👉 Example 2: Embed a YouTube Video

```
<iframe width="560" height="315"
  src="https://www.youtube.com/embed/dQw4w9WgXcQ">
```

```
    frameborder="0"
    allowfullscreen>
</iframe>
```

✓ `iframe` is often used for YouTube, Google Maps, etc.

---

## Real-life Connection for Students

- **Audio** → Music player in a website 🎵
- **Video** → YouTube-like video streaming 🎬
- **Embed/Iframe** → Embedding maps, tutorials, presentations, or Google Forms inside a site.

## ▼ **HTML5: Header, Footer, Section, Article, Aside**

### Topic: HTML5 Structural / Semantic Tags

(Semantic = tags that *describe their meaning/purpose*)

---

### ◆ Why Semantic Tags?

Earlier, we used only `<div>` everywhere → messy & unclear.

HTML5 introduced **semantic elements** that tell both **browsers** and **developers** what the content is about.

Example:

- `<header>` → Page or section heading
  - `<footer>` → Bottom info / copyright
  - `<section>` → Grouping related content
  - `<article>` → Independent, reusable content (like a blog post or news)
  - `<aside>` → Side info, ads, extra notes
-

# 1. Header

Represents the **top section** of a webpage (logo, navigation, title).

```
<header>
  <h1>My Blog</h1>
  <nav>
    <a href="#">Home</a> |
    <a href="#">Articles</a> |
    <a href="#">Contact</a>
  </nav>
</header>
```

# 2. Footer

Represents the **bottom section** (copyright, contact links).

```
<footer>
  <p>&copy; 2025 My Blog. All rights reserved.</p>
</footer>
```

# 3. Section

Represents a **thematic grouping** of content (like chapters, categories, page sections).

```
<section>
  <h2>About Me</h2>
  <p>I am a web developer who loves teaching HTML & CSS.</p>
</section>
```

# 4. Article

Represents an **independent, self-contained piece** of content.

(Think: news article, blog post, product card).

```
<article>
  <h2>How to Learn HTML</h2>
  <p>Start with basics like tags, attributes, and forms...</p>
</article>
```

## 5. Aside

Represents **extra content** (sidebar, ads, related links).

```
<aside>
  <h3>Related Articles</h3>
  <ul>
    <li><a href="#">Learn CSS Basics</a></li>
    <li><a href="#">JavaScript for Beginners</a></li>
  </ul>
</aside>
```

## ✓ Full Example (All Together)

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 Layout Example</title>
  </head>
  <body>

    <!-- Header →
    <header>
      <h1>🌐 My Tech Blog</h1>
      <nav>
        <a href="#">Home</a> |
        <a href="#">Articles</a> |
```

```

    <a href="#">Contact</a>
  </nav>
</header>
<hr>

<!-- Main Content →
<section>
  <article>
    <h2>🚀 HTML Basics</h2>
    <p>HTML is the foundation of all web pages...</p>
  </article>

  <article>
    <h2>🎨 CSS Styling</h2>
    <p>CSS is used to style and beautify your webpage...</p>
  </article>
</section>

<!-- Sidebar →
<aside>
  <h3>🔗 Quick Links</h3>
  <ul>
    <li><a href="#">Learn JavaScript</a></li>
    <li><a href="#">Frontend Projects</a></li>
  </ul>
</aside>

<!-- Footer →
<footer>
  <p>© 2025 My Tech Blog | Made with ❤️ in HTML5</p>
</footer>

</body>
</html>

```

## Real-life Connection for Students


- **Header** = Cover page title of a book.
- **Footer** = Publisher info at bottom.
- **Section** = Different chapters in a book.
- **Article** = One article in a newspaper.
- **Aside** = Side notes, advertisements, references.

## ▼ **Canvas API Basics (Drawing)**

### Topic: HTML5 Canvas API (Basics of Drawing)

---

#### ◆ What is Canvas?

- `<canvas>` is an HTML5 element that lets you **draw graphics using JavaScript**.
- Think of it like a **blank sheet of paper**  where you can paint using JS code.
- It is widely used for **games, charts, animations, image editing tools**.

👉 Syntax:

```
<canvas id="myCanvas" width="400" height="300" style="border:1px solid black;"></canvas>
```

✅ The **canvas is empty by default** → we must use JavaScript to draw on it.

---

## 1. Getting the Canvas Context

To draw, we need the **context** (a toolset that allows drawing).

```
<script>  
let canvas = document.getElementById("myCanvas");
```

```
let ctx = canvas.getContext("2d"); // 2D drawing
</script>
```

## 2. Drawing Shapes

### (a) Draw a Rectangle

```
<canvas id="rectCanvas" width="300" height="200" style="border:1px solid black;"></canvas>

<script>
let c = document.getElementById("rectCanvas");
let ctx = c.getContext("2d");

ctx.fillStyle = "blue"; // Fill color
ctx.fillRect(50, 50, 150, 100); // (x, y, width, height)
</script>
```

✓ Output → A **blue rectangle**.

### (b) Draw a Circle

```
<canvas id="circleCanvas" width="300" height="200" style="border:1px solid black;"></canvas>

<script>
let c2 = document.getElementById("circleCanvas");
let ctx2 = c2.getContext("2d");

ctx2.beginPath();
ctx2.arc(150, 100, 60, 0, 2 * Math.PI); // (x, y, radius, startAngle, endAngle)
ctx2.fillStyle = "green";
```



```
ctx2.fill();  
</script>
```

✓ Output → A **green circle**.

### (c) Draw a Line

```
<canvas id="lineCanvas" width="300" height="200" style="border:1px solid black;"></canvas>  
  
<script>  
  let c3 = document.getElementById("lineCanvas");  
  let ctx3 = c3.getContext("2d");  
  
  ctx3.beginPath();  
  ctx3.moveTo(50, 50); // starting point  
  ctx3.lineTo(250, 150); // ending point  
  ctx3.strokeStyle = "red";  
  ctx3.lineWidth = 3;  
  ctx3.stroke();  
</script>
```

✓ Output → A **red diagonal line**.

## 3. Adding Text

```
<canvas id="textCanvas" width="400" height="150" style="border:1px solid black;"></canvas>  
  
<script>  
  let c4 = document.getElementById("textCanvas");  
  let ctx4 = c4.getContext("2d");  
  
  ctx4.font = "30px Arial";  
  ctx4.fillStyle = "purple";
```

```
ctx4.fillText("Hello Canvas!", 50, 80); // (text, x, y)
</script>
```

✓ Output → "Hello Canvas!" written in purple.

## 4. Real-life Connection

- **Rectangle** = buttons, UI blocks.
- **Circle** = game characters, profile pictures.
- **Line** = graphs, charts.
- **Text** = game scores, labels.
- **Canvas** = used in **games (Flappy Bird, Tetris)** and **data visualizations**.

## ✓ Full Example (All Together)

```
<!DOCTYPE html>
<html>
<head>
  <title>Canvas Basics</title>
</head>
<body>
  <h2>🎨 Canvas API Basics</h2>
  <canvas id="myCanvas" width="500" height="400" style="border:1px solid black;"></canvas>

  <script>
    let canvas = document.getElementById("myCanvas");
    let ctx = canvas.getContext("2d");

    // Rectangle
    ctx.fillStyle = "blue";
    ctx.fillRect(50, 50, 150, 100);
```

```

// Circle
ctx.beginPath();
ctx.arc(350, 100, 60, 0, 2 * Math.PI);
ctx.fillStyle = "green";
ctx.fill();

// Line
ctx.beginPath();
ctx.moveTo(50, 250);
ctx.lineTo(450, 300);
ctx.strokeStyle = "red";
ctx.lineWidth = 3;
ctx.stroke();

// Text
ctx.font = "25px Arial";
ctx.fillStyle = "purple";
ctx.fillText("Canvas is Fun!", 150, 370);
</script>
</body>
</html>

```

## ▼ SVG Basics (Vector Graphics)

### Topic: SVG Basics (Vector Graphics)

#### ◆ What is SVG?

- **SVG** stands for **Scalable Vector Graphics**.
- It uses **XML-based code** to create shapes, text, and images.
- Unlike Canvas (which is pixel-based), **SVG is vector-based** → it never loses quality, even when zoomed in.
- Great for **logos, icons, charts, maps, and illustrations**.

## ◆ Real-life Connection

- **Canvas** = great for **games, animations** (because you can redraw fast).
- **SVG** = great for **logos, icons, maps, charts** (because it scales perfectly).
- Example: All **company logos (Nike, Twitter, YouTube)** are usually in **SVG format** so they don't blur.

```
<!DOCTYPE html>
<html>
<head>
  <title>SVG Basics</title>
</head>
<body>
  <h2> ◆ SVG Examples</h2>

  <!-- Rectangle →
  <svg width="300" height="200">
    <rect x="50" y="30" width="200" height="100" fill="lightblue" stroke
    ="blue" stroke-width="3" />
  </svg>

  <!-- Circle →
  <svg width="200" height="200">
    <circle cx="100" cy="100" r="70" fill="lightgreen" stroke="darkgreen" st
    roke-width="4" />
  </svg>

  <!-- Line →
  <svg width="300" height="200">
    <line x1="20" y1="20" x2="280" y2="180" stroke="red" stroke-width
    ="4" />
  </svg>

  <!-- Polygon →
  <svg width="300" height="200">
```

```
<polygon points="150,10 200,190 100,190" fill="orange" stroke="black" s
troke-width="3" />
</svg>

<!-- Text →
<svg width="400" height="100">
  <text x="50" y="50" font-size="30" fill="purple">SVG is Scalable!</text
>
</svg>

</body>
</html>
```

## ▼ Geolocation API

### Topic: Geolocation API in HTML5

#### ◆ What is Geolocation API?

- Geolocation API allows a webpage to **get the geographical location of the user**.
- It can provide:
  - ✓ Latitude & Longitude
  - ✓ Accuracy (meters)
  - ✓ Altitude (height above sea level, if available)
  - ✓ Speed & Heading (if moving, like in a car with GPS)

👉 It needs **user permission**. Browser will ask:

| "This site wants to know your location – Allow / Block"

#### ◆ Basic Usage

```

<button onclick="getLocation()">Get My Location</button>
<p id="output"></p>

<script>
  let output = document.getElementById("output");

  function getLocation() {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(showPosition, showError);
    } else {
      output.innerHTML = "Geolocation is not supported by this browser.";
    }
  }

  function showPosition(position) {
    output.innerHTML =
      "Latitude: " + position.coords.latitude +
      "<br>Longitude: " + position.coords.longitude;
  }

  function showError(error) {
    switch(error.code) {
      case error.PERMISSION_DENIED:
        output.innerHTML = "User denied the request for Geolocation.";
        break;
      case error.POSITION_UNAVAILABLE:
        output.innerHTML = "Location information is unavailable.";
        break;
      case error.TIMEOUT:
        output.innerHTML = "The request to get user location timed out.";
        break;
      default:
        output.innerHTML = "An unknown error occurred.";
    }
  }

```

```
}  
</script>
```

✓ When clicked, it shows **Latitude & Longitude** of the user.

---

## ◆ Display Location on Google Maps

```
<button onclick="getMap()">Show My Location on Map</button>  
<div id="mapLink"></div>  
  
<script>  
  function getMap() {  
    navigator.geolocation.getCurrentPosition(function(position) {  
      let lat = position.coords.latitude;  
      let lon = position.coords.longitude;  
      let url = "https://www.google.com/maps?q=" + lat + "," + lon;  
      document.getElementById("mapLink").innerHTML =  
        "<a href='" + url + "' target='_blank'> 📍 Open in Google Maps</a>";  
    });  
  }  
</script>
```

👉 This will give a clickable **Google Maps link** with your location.

---

## ◆ Real-life Connection

- **Delivery Apps (Swiggy, Zomato, Amazon)** → to find your delivery location.
  - **Ride apps (Ola, Uber)** → to match driver with passenger.
  - **Weather apps** → to show forecast for your city.
  - **Fitness apps** → to track running/walking routes.
- 

## ✓ Full Example (Location + Map Integration)

```

<!DOCTYPE html>
<html>
<head>
  <title>Geolocation Example</title>
</head>
<body style="font-family: Arial; text-align:center;">

  <h2>🌐 Geolocation API Demo</h2>
  <button onclick="getLocation()">Get My Location</button>
  <p id="output"></p>
  <div id="mapLink"></div>

  <script>
    let output = document.getElementById("output");

    function getLocation() {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition, showError);
      } else {
        output.innerHTML = "Geolocation not supported.";
      }
    }

    function showPosition(position) {
      let lat = position.coords.latitude;
      let lon = position.coords.longitude;
      output.innerHTML = "Latitude: " + lat + "<br>Longitude: " + lon;

      let url = "https://www.google.com/maps?q=" + lat + "," + lon;
      document.getElementById("mapLink").innerHTML =
        "<a href='" + url + "' target='_blank'>📍 View on Google Maps</a>";
    }

    function showError(error) {
      switch(error.code) {

```



```

    case error.PERMISSION_DENIED:
        output.innerHTML = "❌ User denied location access.";
        break;
    case error.POSITION_UNAVAILABLE:
        output.innerHTML = "❌ Location unavailable.";
        break;
    case error.TIMEOUT:
        output.innerHTML = "⌚ Request timed out.";
        break;
    default:
        output.innerHTML = "⚠️ Unknown error.";
    }
}
</script>
</body>
</html>

```

## ▼ Drag & Drop API

### Topic: HTML5 Drag & Drop API

#### ◆ What is Drag & Drop?

- HTML5 has a built-in **Drag & Drop API** that allows users to **grab an element with the mouse and drag it to another location**.
- Used in **file uploaders, Trello boards, drag-sort lists, puzzle games**.

#### ◆ Steps in Drag & Drop

##### 1. Make an element draggable

```
<div draggable="true">Drag me</div>
```

## 2. Handle drag events with JavaScript

- `ondragstart` → When dragging starts
- `ondragover` → When dragged over a drop area
- `ondrop` → When dropped

### ◆ Example 1: Simple Drag & Drop Box

```
<!DOCTYPE html>
<html>
<head>
  <title>Two-Way Drag & Drop</title>
  <style>
    body { font-family: Arial; text-align: center; }

    .container {
      display: flex;
      justify-content: space-around;
      margin-top: 30px;
    }

    .zone {
      width: 300px; height: 250px;
      border: 3px dashed gray;
      border-radius: 10px;
      padding: 10px;
      text-align: center;
      font-weight: bold;
    }

    .item {
      width: 80px; height: 80px;
      background: lightgreen;
      margin: 10px auto;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="zone">
      <div class="item"></div>
    </div>
  </div>
</body>
</html>
```

```

    line-height: 80px;
    cursor: grab;
    border: 2px solid green;
    border-radius: 8px;
  }
</style>
</head>
<body>

<h2>🍏🍌🍊 Drag Fruits Into the Cart and Back</h2>

<div class="container">
  <!-- Fruits Section →
  <div id="fruits" class="zone">
    Fruits Section
    <div id="apple" class="item" draggable="true">🍏</div>
    <div id="banana" class="item" draggable="true">🍌</div>
    <div id="orange" class="item" draggable="true">🍊</div>
  </div>

  <!-- Cart Section →
  <div id="cart" class="zone">
    Cart Section
  </div>
</div>

<script>
  // Make all items draggable
  let items = document.querySelectorAll(".item");
  items.forEach(item => {
    item.ondragstart = function(event) {
      event.dataTransfer.setData("text", event.target.id);
    };
  });

  // Both zones (fruits + cart) can accept drops

```

```

let zones = document.querySelectorAll(".zone");
zones.forEach(zone => {
  zone.ondragover = function(event) {
    event.preventDefault(); // Allow dropping
  };

  zone.ondrop = function(event) {
    event.preventDefault();
    let data = event.dataTransfer.getData("text");
    let element = document.getElementById(data);
    zone.appendChild(element); // Move element into new zone
  };
});
</script>

</body>
</html>

```

✓ You can **drag the blue box** into the drop zone.

## ◆ Real-life Connection

- **Google Drive / File uploaders** → Drag files into upload area.
- **Trello, Jira** → Move tasks between "To Do", "In Progress", "Done".
- **E-commerce** → Drag items into a shopping cart.
- **Games** → Puzzle games, chess pieces, etc.

## ▼ 🎯 Final HTML Project



# Project 1: Student Registration Form Website

## 🎯 Goal

Create a **multi-section student registration page** for a coaching institute.

## ✅ Requirements

1. Use a **Header** (site title: "ABC Coaching") and a **Navigation Bar** (Home | About | Register).
2. Add a **form** with:
  - Full Name (text, required)
  - Email (text, pattern for email)
  - Phone Number (pattern for 10 digits)
  - Gender (radio buttons)
  - Course Selection (dropdown: HTML, CSS, JavaScript)
  - Address (textarea)
  - Submit & Reset buttons
3. Add a **Table** below the form showing "Available Courses" (Course Name, Duration, Fees).
4. Add a **Footer** with contact info.

👉 **Challenge:** Use `required` & `pattern` attributes in form fields.

---



## Project 2: Personal Portfolio (HTML Only)

### 🎯 Goal

Build a **basic personal portfolio site** with multiple sections.

## ✅ Requirements

1. Use a **Header** → Your Name + Navigation links (About | Projects | Contact).
2. **Section: About Me**
  - Heading + paragraph about yourself.
  - Use **text formatting** (bold, italic, underline).

### 3. Section: My Projects

- Create an **unordered list** of your favorite projects.
- Embed a **YouTube video** or **audio introduction** about yourself.

### 4. Section: Contact Me

- Create a **contact form** with Name, Email, Message (textarea).
- Add a **Submit button** (with validation).

5. **Aside** → Add a list of your hobbies (using ordered list).

6. **Footer** → Add your email, phone number, and copyright notice.

👉 **Challenge:** Use at least **one table** (like project list with Title, Description, Status).

▼ Answer



## Project 1: Student Registration Form Website

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    header, footer { background: #f0f0f0; padding: 15px; text-align: center; }
    nav a { margin: 0 10px; text-decoration: none; color: blue; }
    table { width: 100%; border-collapse: collapse; margin-top: 20px; }
    table, th, td { border: 1px solid black; }
    th, td { padding: 10px; text-align: center; }
    form { margin-top: 20px; background: #f9f9f9; padding: 20px; border-radius: 8px; }
```

```

    label { display: block; margin-top: 10px; }
    input, select, textarea { width: 100%; padding: 8px; margin-top: 5px;
}
    input[type="radio"] { width: auto; }
    .btn { margin-top: 15px; }
</style>
</head>
<body>

<!-- Header & Navigation →
<header>
  <h1>ABC Coaching Institute</h1>
  <nav>
    <a href="#">Home</a> |
    <a href="#">About</a> |
    <a href="#">Register</a>
  </nav>
</header>

<!-- Registration Form →
<h2>Student Registration Form</h2>
<form>
  <label>Full Name:</label>
  <input type="text" name="fullname" required>

  <label>Email:</label>
  <input type="email" name="email" required>

  <label>Phone Number:</label>
  <input type="text" name="phone" pattern="[0-9]{10}" placeholder
="10 digits" required>

  <label>Gender:</label>
  <input type="radio" name="gender" value="Male"> Male
  <input type="radio" name="gender" value="Female"> Female

```

```

<label>Course Selection:</label>
<select name="course">
  <option value="html">HTML</option>
  <option value="css">CSS</option>
  <option value="js">JavaScript</option>
</select>

<label>Address:</label>
<textarea name="address" rows="3"></textarea>

<div class="btn">
  <input type="submit" value="Register">
  <input type="reset" value="Clear">
</div>
</form>

<!-- Courses Table -->
<h2>Available Courses</h2>
<table>
  <tr>
    <th>Course Name</th>
    <th>Duration</th>
    <th>Fees</th>
  </tr>
  <tr>
    <td>HTML</td>
    <td>1 Month</td>
    <td>₹2000</td>
  </tr>
  <tr>
    <td>CSS</td>
    <td>1 Month</td>
    <td>₹2500</td>
  </tr>
  <tr>
    <td>JavaScript</td>

```





```

    label { display: block; margin-top: 8px; }
    input, textarea { width: 100%; padding: 8px; margin-top: 5px; }
  </style>
</head>
<body>

  <!-- Header →
  <header>
    <h1>My Portfolio</h1>
    <nav>
      <a href="#">About</a> |
      <a href="#">Projects</a> |
      <a href="#">Contact</a>
    </nav>
  </header>

  <!-- About Section →
  <section id="about">
    <h2>About Me</h2>
    <p>Hello! My name is <b>Karan</b>. I am passionate about <i>web
development</i> and <u>coding</u>.</p>
  </section>

  <!-- Projects Section →
  <section id="projects">
    <h2>My Projects</h2>
    <ul>
      <li>Portfolio Website</li>
      <li>Student Registration Form</li>
      <li>Online Quiz App</li>
    </ul>

  <!-- Media →
  <h3>My Intro Video</h3>
  <iframe width="300" height="200" src="https://www.youtube.com/
embed/tgbNymZ7vqY"></iframe>

```

```

<!-- Projects Table →
<h3>Projects Table</h3>
<table>
  <tr>
    <th>Project Title</th>
    <th>Description</th>
    <th>Status</th>
  </tr>
  <tr>
    <td>Portfolio Website</td>
    <td>Personal website with all details</td>
    <td>Completed</td>
  </tr>
  <tr>
    <td>Quiz App</td>
    <td>MCQ-based online quiz</td>
    <td>In Progress</td>
  </tr>
</table>
</section>

<!-- Aside →
<aside>
  <h3>My Hobbies</h3>
  <ol>
    <li>Reading</li>
    <li>Gaming</li>
    <li>Traveling</li>
  </ol>
</aside>

<!-- Contact Section →
<section id="contact">
  <h2>Contact Me</h2>
  <form>

```

```
<label>Name:</label>
<input type="text" name="name" required>

<label>Email:</label>
<input type="email" name="email" required>

<label>Message:</label>
<textarea rows="4"></textarea>

<input type="submit" value="Send">
</form>
</section>

<!-- Footer →
<footer>
  <p>Email: karan@example.com | Phone: +91 9876543210</p>
  <p>&copy; 2025 My Portfolio</p>
</footer>

</body>
</html>
```