



Prepared by Mohamed Saied Hassan

From 1991's original tags to today's Living Standard

Lecture01: HTML "THE LANGUAGE OF THE WEB"



Chapter

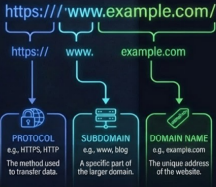
01

HTML — What We Will Learn

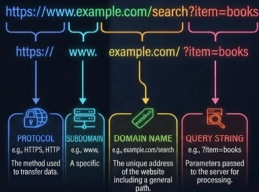
- 01 Web & HTML foundations
- 02 History of HTML
- 03 HTML document structure
- 04 Elements & attributes
- 05 Text, links & images
- 06 Lists, tables & forms
- 07 Semantic HTML & accessibility
- 08 Metadata, multimedia
- 09 Best practices & practical page

What is URL" Uniform Resource Locator"?

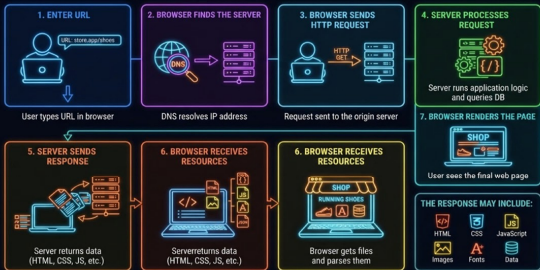
SIMPLIFIED URL STRUCTURE



URL WITH QUERY STRING



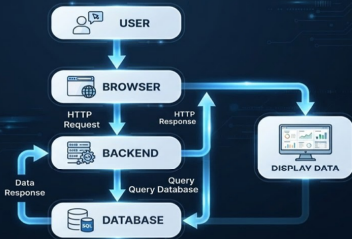
What Happens When You Visit a Website?



How Does a Web Application Work?

Web Application Data Flow

(uss ASCII Diagram)



What Is HTML?

HTML = HyperText Markup Language

A markup language, not a programming language

Describes structure and meaning of web content

Works with CSS and JavaScript

History of HTML

HTML emerged from the early World Wide Web

1989 → Tim Berners-Lee proposes a hypertext system at CERN

Early HTML focused on documents and hyperlinks

HTML evolved as the Web became more interactive

Chapter

02

Tim Berners-Lee & the Web

Core Web ideas:

- HTML — document structure
- HTTP — communication protocol
- URL — resource identification

Together they formed the foundation of the World Wide Web.

Chapter

02

From Early HTML to Modern HTML

Early HTML → simple documents

HTML 2.0 → standardized foundations

HTML 3.2 → broader browser features

HTML 4.01 → mature document markup

XHTML → stricter XML-style rules

HTML5 / Living Standard → modern Web

Chapter

02

Why HTML Became More Than Documents

Web pages evolved into applications

Forms → user input

Multimedia → audio & video

Semantic elements → meaningful structure

APIs + JavaScript → rich interactions

Responsive images → many devices

Chapter

03

Create Your First HTML File

File: index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
</html>
```

An HTML Element

`<p>Hello World</p>`

Opening tag → `<p>`

Content → Hello World

Closing tag → `</p>`

The complete structure is an element.

Nested Elements

```
<p>
```

```
  Hello <strong>students</strong>
```

```
</p>
```

Elements can contain other elements.
The strong element is nested inside p.

Correct Nesting

Correct:

```
<p><strong>Hello</strong></p>
```

Incorrect:

```
<p><strong>Hello</p></strong>
```

Rule: close nested elements in reverse order.

Attributes

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

href = attribute

"https://example.com" = value

Attributes provide additional information or behavior.

Chapter

03

Common HTML Attributes

Global: id, class, title, lang, hidden, data-*

Element-specific examples: src, href, alt, name, value, type

Choose attributes according to the element's purpose.

The HTML Document

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>...</head>
```

```
<body>...</body>
```

```
</html>
```

Chapter

04

DOCTYPE

```
<!DOCTYPE html>
```

Not an HTML element.

It is a declaration that tells the browser to use standards-oriented HTML parsing.

The <html> Element

```
<html lang="en">
```

Root element of the document.

The lang attribute identifies the document language.

Examples: en, ar, fr, de

The <head> Element

```
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>My Website</title>  
</head>
```

Contains metadata and resource references.

The `<body>` Element

```
<body>  
  <h1>Welcome</h1>  
  <p>My first webpage.</p>  
</body>
```

Contains the document content users interact with.

Headings

`<h1>Main Heading</h1>`

`<h2>Section</h2>`

`<h3>Subsection</h3>`

...

`<h6>Lowest heading level</h6>`

Use headings for document hierarchy, not just visual size.

Paragraphs

`<p>HTML is used to structure web content.</p>`

Use `p` for paragraphs.

Do not use `div` elements just because they can contain text.

Text Meaning & Emphasis

Important

Emphasized

Highlighted

Small print

~~Deleted~~

Inserted

_{H₂O} ^{x²}

Chapter

05

Line Breaks & Thematic Breaks

Line break:

`<br>`

Thematic break:

`<hr>`

Do not use repeated `<br>` elements for page layout.
Use CSS for spacing and layout.

Comments

```
<!-- This is a comment -->
```

Comments are not rendered as visible page content.
Use them to document code or temporarily explain sections.

Hyperlinks

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

`<a>` is the anchor element.
`href` specifies the destination.

Types of Links

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

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

```
<a href="#contact">Section</a>
```

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

```
<a href="tel:+201000000000">Call</a>
```

Images

```

```

img is a void element.

It has no closing tag.

Important: src + meaningful alt text.

Image Attributes

```

```

Responsive Images

`<picture>` can provide different image sources.

Use cases:

- Different screen sizes
- Art direction
- Different formats

`srcset` and `sizes` are also important for responsive images.

Unordered Lists

```
<ul>  
  <li>HTML</li>  
  <li>CSS</li>  
  <li>JavaScript</li>  
</ul>
```

Use when order does not matter.

Ordered Lists

```
<ol>  
  <li>Install VS Code</li>  
  <li>Create index.html</li>  
  <li>Open in browser</li>  
</ol>
```

Use when sequence or ranking matters.

Description Lists

```
<dl>
```

```
<dt>HTML</dt>
```

```
<dd>Structures web content.</dd>
```

```
<dt>CSS</dt>
```

```
<dd>Styles web content.</dd>
```

```
</dl>
```

Tables

```
<table>
<tr>
  <th>Name</th><th>Grade</th>
</tr>
<tr>
  <td>Ahmed</td><td>A</td>
</tr>
</table>
```

Tables are for tabular data — not page layout.

Semantic Table Structure

```
<table>  
  <thead>...</thead>  
  <tbody>...</tbody>  
  <tfoot>...</tfoot>  
</table>
```

<th> = header cell

<td> = data cell

colspan & rowspan

```
<td colspan="2">Total</td>
```

```
<td rowspan="2">Computer Science</td>
```

Use when a cell spans multiple columns or rows.

Why Forms?

Forms collect user input:

- Login
- Registration
- Search
- Contact
- Surveys
- Checkout

`<form>` groups related controls.

Basic Form

```
<form>  
  <label>Name</label>  
  <input type="text">  
  <button type="submit">Submit</button>  
</form>
```

Labels & Inputs

```
<label for="email">Email</label>  
<input id="email" type="email">
```

The label for attribute points to the input id.
This improves usability and accessibility.

Chapter

08

Input Types

text , password , email , number
date ,time ,file
checkbox , radio
color , range , search

Choose the type that matches the data.

Form Validation

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

```
<input type="text" minlength="3">
```

```
<input type="number" min="1" max="100">
```

HTML provides basic client-side validation.
Server-side validation is still mandatory.

Select & Textarea

```
<select id="department">  
  <option>Computer Science</option>  
  <option>AI</option>  
</select>
```

```
<textarea rows="5"></textarea>
```

Buttons

```
<button type="submit">Submit</button>
```

```
<button type="reset">Reset</button>
```

```
<button type="button">Normal</button>
```

Inside forms, be explicit about button type.

Form Submission

```
<form action="/login" method="post">
```

```
...
```

```
</form>
```

GET → commonly requests data/resources

POST → commonly submits data

Chapter

09

Semantic HTML

Semantic = meaningful structure

<header>

<nav>

<main>

<section>

<article>

<aside>

<footer>

A Semantic Page

- <header> Site identity
- <nav> Navigation
- <main> Main content
- <section> Thematic section
- <article> Independent content
- <aside> Related content
- <footer> Footer information

<div> &

<div> → generic block container

 → generic inline container

Use them when no more meaningful semantic element fits.
Do not build every page from anonymous divs.

Chapter

09

Accessibility

Good HTML helps:

- Screen readers
- Keyboard users
- People with visual impairments
- People with motor/cognitive disabilities

Accessibility starts with correct HTML.

Accessibility Examples

Good image:

```

```

Good form:

```
<label for="name">Name</label>
```

```
<input id="name" name="name">
```

Accessible Links & Headings

Use meaningful link text:

- ✓ Read the HTML documentation
- ✗ Click here

Use a logical heading hierarchy:

h1 → h2 → h3

Do not choose headings only because they look large.

Metadata in <head>

```
<title>My Website</title>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The head describes the document and references resources.

Character Encoding

```
<meta charset="UTF-8">
```

Supports a wide range of characters:

English • العربية • 日本語 • 中文

Use UTF-8 in modern HTML documents.

Responsive Viewport

```
<meta name="viewport"  
content="width=device-width, initial-scale=1.0">
```

Helps mobile browsers interpret the page viewport correctly.

Chapter

10

SEO Basics in HTML

`<title>` → page title

`<meta name="description">` → summary

Headings → document hierarchy

Semantic HTML → meaning

alt → image alternative text

Descriptive links → context

Audio & Video

```
<audio controls>
```

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

```
</audio>
```

```
<video controls>
```

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

```
</video>
```

iframe

```
<iframe src="https://example.com"></iframe>
```

Common uses:

- Maps
- Videos
- External documents

Consider security, privacy and performance.

Canvas & SVG

`<canvas>` → drawing surface controlled mainly by JavaScript

`<svg>` → vector graphics represented as elements

Useful for graphics, charts, games and visualizations.

HTML → DOM

```
<body>  
  <h1>Hello</h1>  
  <p>Welcome</p>  
</body>
```

Browser creates a tree-like DOM representation.

Chapter

11

DOM Tree

Document



Nodes form a tree.

JavaScript & the DOM

```
document.querySelector("h1");
```

JavaScript can:

- Find elements
- Change text
- Change attributes/classes
- Add/remove elements
- Respond to events

Chapter

11

HTML + CSS + JavaScript

HTML → structure

CSS → presentation

JavaScript → behavior

A professional frontend combines all three while keeping responsibilities clear.

Chapter

12

Global Attributes

id → unique identifier

class → reusable classification

title → advisory information

lang → language

hidden → hidden state

data-* → custom data attributes

id vs. class

```
<div id="student"></div>
```

```
<div class="card"></div>
```

```
<div class="card"></div>
```

id → identifies one element

class → reusable grouping/styling hook

Chapter

12

Relative & Absolute Paths

Relative:

`images/logo.png`

Root-relative:

`/images/logo.png`

Absolute:

`https://example.com/images/logo.png`

Paths determine where the browser looks for a resource.

Chapter

12

Browser Developer Tools

F12 / Right-click → Inspect

Elements → inspect DOM

Styles → inspect CSS

Console → JavaScript

Network → requests & resources

Accessibility → inspect semantics

Common HTML Mistakes

- ✗ Wrong nesting
- ✗ Duplicate IDs
- ✗ Missing alt text
- ✗ Missing form labels
- ✗ Tables for layout
- ✗ Repeated `<br>` for spacing
- ✗ Styling with obsolete HTML elements
- ✗ Using `div` for everything

HTML Best Practices

1. Use semantic elements
2. Keep structure meaningful
3. Use accessible labels and alt text
4. Keep nesting valid
5. Use CSS for presentation
6. Validate and inspect markup
7. Write maintainable, readable HTML

Chapter

13

Mini Project — Student Profile

Build a university student profile page containing:

- Header + navigation
- Profile information + image
- Skills list
- Courses table
- Contact form
- Footer

Focus on semantic HTML before styling.

Project — Page Skeleton

```
<body>  
  <header>...</header>  
  <nav>...</nav>  
  <main>  
    <section>...</section>  
    <section>...</section>  
  </main>  
  <footer>...</footer>  
</body>
```

Project — Profile & Skills

```
<section>
```

```
  <h1>Student Name</h1>
```

```
  
```

```
  <p>Computer Science Student</p>
```

```
</section>
```

```
<h2>Skills</h2>
```

```
<ul><li>HTML</li><li>CSS</li><li>JavaScript</li></ul>
```

Project — Courses Table

```
<table>
  <thead>
    <tr><th>Course</th><th>Grade</th></tr>
  </thead>
  <tbody>
    <tr><td>Web Development</td><td>A</td></tr>
    <tr><td>Database</td><td>B+</td></tr>
  </tbody>
</table>
```

Project — Contact Form

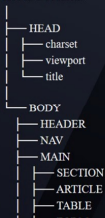
```
<form>  
  <label for="name">Name</label>  
  <input id="name" name="name" required>  
  
  <label for="email">Email</label>  
  <input id="email" name="email" type="email" required>  
  
  <button type="submit">Send</button>  
</form>
```

Chapter

13

Final HTML Architecture

HTML Document



```
graph TD; HTML[HTML Document] --- HEAD[HEAD]; HTML --- BODY[BODY]; HEAD --- charset[charset]; HEAD --- viewport[viewport]; HEAD --- title[title]; BODY --- HEADER[HEADER]; BODY --- NAV[NAV]; BODY --- MAIN[MAIN]; BODY --- SECTION[SECTION]; BODY --- ARTICLE[ARTICLE]; BODY --- TABLE[TABLE];
```

HEAD

charset

viewport

title

BODY

HEADER

NAV

MAIN

SECTION

ARTICLE

TABLE

Chapter

14

The Most Important Idea

Do not ask:

“Which tag makes this look good?”

Ask:

“What is this content?”

HTML → meaning & structure

CSS → appearance

JavaScript → behavior

Chapter

14

What You Should Know Now

- ✓ What HTML is
- ✓ HTML history
- ✓ Document structure
- ✓ Elements & attributes
- ✓ Text, links & images
- ✓ Lists & tables
- ✓ Forms
- ✓ Semantic HTML
- ✓ Accessibility
- ✓ Metadata & multimedia
- ✓ DOM
- ✓ Best practices

Chapter

14

Next: CSS

HTML → Structure

CSS → Design

Responsive layouts → Flexbox & Grid

Components → reusable UI

Next lecture: turning our HTML structure into a professional interface.

THANK YOU