Learn how websites work

🌍 1. What a Website Really Is

A **website** is a collection of **web pages** (HTML files) stored on a **web server** and accessible through the **internet**.

Example:
When you visit https://www.wikipedia.org, your browser requests the Wikipedia server for its homepage, and the server sends back HTML, CSS, JavaScript, and images.

🧩 2. The Main Parts of a Website

Component	Description
Client (Browser)	The software you use to view websites (e.g., Chrome, Edge, Firefox).
Server	The computer that stores the website's files and sends them when requested.
Domain Name	The human-friendly name (like example.com) that points to the server's IP address.
Hosting	A service that provides space on a server for your website's files.
Database	Stores dynamic data (like users, posts, products). Example: MySQL, PostgreSQL.

⚙️ 3. What Happens When You Visit a Website

Here's what happens when you type https://www.example.com:

1. **DNS Lookup**
   Your computer asks the **Domain Name System (DNS)** for the IP address of example.com.
2. **Browser Connects to the Server**
   Using the IP address, your browser sends a request (via **HTTP or HTTPS**) to the website's server.

3. **Server Processes the Request**
   The server finds the right page (or generates it dynamically using PHP, Node.js, Python, etc.).
4. **Server Sends a Response**
   The server sends back files: HTML, CSS, JS, images, etc.
5. **Browser Renders the Page**
   Your browser reads the HTML, applies CSS for styling, runs JavaScript for interactivity, and displays the page to you.

🧠 4. Frontend vs. Backend

Side	Technologies	What It Does
Frontend (Client-side)	HTML, CSS, JavaScript	What users see — layout, design, interactivity.
Backend (Server-side)	PHP, Python, Node.js, Java, etc.	Handles logic, data storage, user authentication.

🗄️ 5. Example Flow (Dynamic Website)

Imagine a **student management system** you're building:

1. The student logs in (frontend form).
2. The form data is sent to your **PHP backend**.
3. PHP checks credentials in the **MySQL database**.
4. If valid, PHP generates an HTML page showing student info.
5. The page is sent to the browser for display.

🔒 6. Extra Important Concepts

- **HTTPS:** Encrypts data between browser and server for security.
- **APIs:** Allow different apps or websites to talk to each other (like showing weather data).
- **Cookies & Sessions:** Store temporary data like login states.
- **CDN (Content Delivery Network):** Speeds up websites by serving files from nearby servers.

🚀 7. How to Start Learning Practically

Here's a roadmap you can follow:

1. **Learn Frontend**
   - HTML → structure

    - CSS → styling
    - JavaScript → interactivity
2. **Learn Backend**
    - Choose one language: **PHP**, **Python (Flask/Django)**, **Node.js**, etc.
    - Learn how to connect to databases (MySQL, PostgreSQL).
3. **Learn Deployment**
    - Use hosting platforms (e.g., Hostinger, Vercel, or AWS).
    - Buy a domain and connect it.

HTML: headings, paragraphs, images, links

1. Headings (<h1> to <h6>)

What are they?

Headings are semantic HTML elements that define the structure and outline of a webpage's content. They form a hierarchy from most important (<h1>) to least important (<h6>).

Why use headings?

- **Content structure: Headings divide content into logical sections, similar to chapters and subchapters in a book.**
- **SEO: Search engines use headings to understand your page's main topics and improve ranking.**
- **Accessibility: Screen readers provide navigation by jumping between headings, helping users quickly understand and navigate the page.**
- **Styling: Browsers apply default sizes and bold styles, which can be overridden with CSS.**

Best practices

- **Only one <h1> per page** — Usually the page's main title.
- Use headings in **sequential order**: don't skip levels (e.g., no jumping from <h1> directly to <h4>).
- Write **descriptive, concise headings** that summarize the section content.
- Avoid using headings just for styling (e.g., bold text); use CSS for appearance instead.

2. Paragraphs (<p>)

What are they?

Paragraphs represent blocks of text — the primary way to present readable content.

Why use paragraphs?

- **Browsers automatically add vertical spacing (margin) to paragraphs, making text easier to read.**

- **They maintain the semantic meaning of your content — a block of prose or explanation.**
- **Screen readers identify paragraphs to provide better voice reading flow.**
- **Helps SEO by organizing text logically.**

Important details

- **Paragraphs cannot contain block-level elements like , <section>, or <h1>. They can contain inline elements like <a>, , .**
- **Use inline tags within paragraphs to emphasize or link text.**

Example with inline elements

This is a paragraph with a link and some important text.

3. Images (<img>)

What are they?

The <img> tag embeds images in your page — visual content like photos, logos, icons, etc.

Why use images properly?

- **Images enhance user experience by providing visual context.**
- **Proper attributes make images accessible and SEO-friendly.**
- **Misuse can lead to slow page loads or inaccessible content.**

Critical attributes explained

Attribute	Details & Importance
src	URL/path to the image file (required).
alt	Alternative text describing the image (required). Crucial for screen readers & SEO.
width & height	Helps browsers reserve layout space, reducing page shifting during loading.
loading="lazy"	Defers offscreen image loading, improving page speed on large pages (modern browsers).

Why is alt so important?

- Screen readers read alt text to users who can’t see the image.
- If the image fails to load, alt text is displayed.
- Provides context to search engines about the image content.

Tip: If image is purely decorative, use alt="" so screen readers skip it.

Example:

<img src="team-photo.jpg" alt="Our team standing outside the office" width="600" height="400" loading="lazy">

4. Links (<a>)

What are they?

Anchor (<a>) tags create hyperlinks, allowing users to navigate to other pages, files, or parts of the same page.

Why are links important?

- They are the building blocks of the web.
- Help users navigate your site and the internet.
- Search engines use links to crawl and rank pages.
- Proper use helps with usability and accessibility.

Important attributes

Attribute	Purpose
href	**Target URL or anchor name. Required for navigation.**
target	**Where the link opens. _blank opens in a new tab.**
rel	**Defines relationship (e.g., noopener noreferrer to improve security with _blank).**
title	**Tooltip text shown on hover (helps users).**
download	**Triggers download of the linked file instead of navigation (useful for PDFs, images, etc.).**

Security note:

When using target="_blank", always include

rel="noopener noreferrer"

to prevent tab-napping attacks — a security vulnerability where the new tab can control the original page.

Types of links

Link Type	Example	Notes
External link	Google	Opens another website
Internal link	About Us	Goes to another page on your site
Anchor link	Contact	Jumps to a section within the same page
Email link	Email Us	Opens email client

Example:

```
<a href="https://example.com" target="_blank" rel="noopener noreferrer" title="Visit Example website">
  Visit Example
</a>
```

Putting It All Together: A Simple Webpage Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Deep HTML Explanation Example</title>
</head>
```

```
<body>

  <h1>Welcome to My Website</h1>

  <p>This website is created to explain basic HTML elements deeply. Here is an image:</p>

  <img src="landscape.jpg" alt="Beautiful mountain landscape" width="600" height="400" loading="lazy">

  <p>Feel free to visit <a href="https://www.wikipedia.org" target="_blank" rel="noopener noreferrer">Wikipedia</a> for more information.</p>

  <h2>About This Page</h2>

  <p>The heading above uses an <code>&lt;h2&gt;</code> tag, which is a subheading.</p>

</body>

</html>
```

Additional Notes

- HTML is semantic — meaning tags have meaning beyond appearance.
- Using the right tags improves accessibility, SEO, and maintainability.
- Styling and interactivity should be done with CSS and JavaScript, not by misusing HTML tags.
- Learn to validate your HTML with tools like W3C Validator to avoid errors.

Practice-create your own portfolio page