WEB ENGINEERING FUNDAMENTALS

HTML Fundamentals

A Complete Beginner's Guide to Building Web Pages

Learn HTML from first principles. Understand how web pages are structured, how HTML elements work together, and how to build well-organized, semantic web pages before learning CSS and JavaScript.

haas.dev • dev-roast-app.vercel.app

Table of Contents

1. Introduction

Every website starts with HTML.

Whether you open Google, YouTube, Amazon, GitHub, or haas.dev, the browser first receives an HTML document. Without HTML, a browser has no structure to display.

Many beginners think HTML is a programming language. It is not.

HTML is a markup language used to describe the structure and meaning of content. It tells the browser:

- This is a heading.
- This is a paragraph.
- This is an image.
- This is a button.
- This is a navigation menu.

Think of HTML as the blueprint of a house. A blueprint defines where rooms, doors, and windows belong. Similarly, HTML defines where every piece of content belongs on a webpage.

2. What Is HTML?

HTML stands for HyperText Markup Language.

It is the standard language used to structure web pages.

HTML does not:

- add colors
- create animations
- perform calculations
- make websites interactive

Instead, HTML defines what each piece of content is.

Examples include:

- headings
- paragraphs
- buttons
- images
- videos
- forms
- navigation menus
- tables

CSS controls appearance.

JavaScript controls behavior.

HTML provides the foundation that both depend on.

3. Why HTML Is Important

Every frontend framework—including React, Angular, Vue, and Svelte—ultimately produces HTML that the browser renders.

Without HTML:

- browsers cannot display content
- CSS has nothing to style
- JavaScript has nothing to manipulate
- search engines cannot understand your pages
- screen readers cannot interpret your content

Learning HTML properly makes learning every frontend technology easier.

4. How HTML Works

The browser follows a simple process.

```
Developer writes HTML
↓
Browser downloads HTML
↓
Browser parses the document
↓
DOM is created
↓
CSS styles the DOM
↓
JavaScript adds behavior
↓
Complete webpage appears
```

HTML is always the first building block of a webpage.

> Learn More
> Read How Browsers Work to understand how browsers convert HTML into the DOM before rendering the page.

5. HTML Document Structure

Every HTML page follows a basic structure.

```
<!DOCTYPE html>
<html>

<head>
    Page information
</head>

<body>
    Visible webpage content
</body>

</html>
```

Each section has a specific responsibility.

DOCTYPE

Tells the browser to use the latest HTML standard.

html

The root element that contains the entire webpage.

head

Contains information about the webpage that users normally do not see.

Examples include:

- page title
- metadata
- CSS files
- fonts
- favicon

body

Contains everything visible on the webpage.

Examples:

- text
- images

- buttons
- forms
- videos
- navigation

6. Understanding HTML Elements

An HTML element consists of:

```
Opening tag
↓
Content
↓
Closing tag
```

Example:

```
<h1>Welcome to haas.dev</h1>
```

The browser understands that this is the main heading.

Different elements describe different kinds of content.

7. HTML Tags and Attributes

Tags define elements.

Attributes provide additional information.

Example:

```
<img src="profile.jpg" alt="Developer Profile">
```

Here:

- img is the element.
- src specifies the image location.
- alt provides descriptive text for accessibility.

Attributes make HTML more powerful without changing its structure.

8. Common HTML Elements

Headings

Use headings to organize content.

```
<h1>Main Title</h1>
<h2>Section</h2>
<h3>Subsection</h3>
```

Use only one <h1> per page whenever possible.

Paragraphs

Paragraphs display normal text.

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

Links

Links connect pages together.

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

Without links, websites would not be connected.

Images

```
<img src="photo.jpg" alt="Team Photo">
```

Always include meaningful alt text.

Buttons

```
<button>Submit</button>
```

Buttons allow users to perform actions.

9. Semantic HTML

Semantic HTML uses meaningful elements instead of generic containers.

Examples include:

- <header>
- <nav>
- <main>
- <section>
- <article>
- <aside>
- <footer>

Instead of writing:

```
<div>
```

Use:

```
<header>
```

when the content represents the page header.

Semantic HTML improves:

- accessibility
- SEO
- readability
- maintainability

> Related Guide
> Read Anatomy of a Modern Website to understand where semantic elements belong in real websites.

10. Lists, Links, Images, and Tables

HTML provides specialized elements for organizing information.

Ordered List

```
<ol>
<li>Install VS Code</li>
<li>Create HTML file</li>
<li>Open browser</li>
</ol>
```

Unordered List

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

Table

Tables organize structured data.

Examples:

- pricing
- schedules
- comparison charts

Avoid using tables to create page layouts.

11. HTML Forms

Forms collect user information.

Common form controls include:

- text fields
- email inputs
- password fields
- checkboxes
- radio buttons
- dropdown menus
- submit buttons

Example:

```
<form>

<input type="email">

<input type="password">

<button>Login</button>

</form>
```

Forms are the foundation of login systems, registrations, search bars, and contact pages.

12. HTML Best Practices

Follow these habits from the beginning.

- Use semantic elements.
- Write meaningful page titles.
- Keep indentation consistent.
- Use descriptive filenames.
- Add alt text to every important image.
- Use headings in the correct order.
- Keep HTML clean and readable.

Writing clean HTML makes collaboration much easier.

13. Real World Example

Imagine building the homepage of haas.dev.

The structure could be:

```
Header
↓
Navigation
↓
Hero Section
↓
Featured Learning Paths
↓
Resources Section
↓
Testimonials
↓
Newsletter
↓
Footer
```

Each section is represented using semantic HTML elements.

CSS later controls the design.

JavaScript later adds interactivity.

HTML provides the structure first.

14. Common Beginner Mistakes

- Treating HTML as a programming language.
- Using for everything.
- Skipping semantic elements.
- Using multiple <h1> elements incorrectly.
- Forgetting alt attributes on images.
- Writing messy, unindented code.
- Using tables for layouts instead of data.

15. Practical Action Plan

Create a simple webpage that includes:

- one header
- navigation
- hero section
- three feature cards
- contact form
- footer

Do not use CSS yet.

Focus only on building a clean HTML structure.

Ask yourself:

> "Can someone understand my webpage just by reading the HTML?"

If yes, your structure is likely good.

16. Mini Architecture Challenge

Challenge

Design the HTML structure for an online learning platform.

Before writing code, identify:

- page sections
- navigation
- main content
- sidebar (if needed)
- footer
- forms
- buttons

Draw the page layout first.

Then decide which semantic HTML element belongs to each section.

This follows the haas.dev framework:

```
Understand → Break → Design → Build
```

17. Key Takeaways

- HTML structures webpages.
- HTML is a markup language, not a programming language.
- Browsers create the DOM from HTML.
- Semantic HTML improves accessibility and SEO.
- CSS controls appearance.
- JavaScript controls behavior.
- Clean HTML is the foundation of professional frontend development.

18. Summary Page

HTML Cheat Sheet

HTML = Structure

CSS = Design

JavaScript = Behavior

<head> contains metadata

<body> contains visible content

Use semantic HTML

Use proper heading hierarchy

Add meaningful alt text

Keep code organized

Common HTML Workflow

```
Plan webpage
↓
Write semantic HTML
↓
Validate structure
↓
Add CSS
↓
Add JavaScript
↓
Test accessibility
↓
Deploy
```

19. HTML Cheat Sheet

A quick-reference checklist you can keep beside your editor while you build.

19. HTML Cheat Sheet

A quick-reference checklist you can keep beside your editor while you build.

- [ ] Start every page with <!DOCTYPE html>
- [ ] One <h1> per page
- [ ] Use <header>, <nav>, <main>, <footer> instead of
- [ ] Every <img> has meaningful alt text
- [ ] Links use descriptive text, not "click here"
- [ ] Forms use proper <label> and <input type=""> values
- [ ] Indentation is consistent throughout the file
- [ ] Heading order goes h1 > h2 > h3 without skipping levels
- [ ] Tables are used for data, never for layout

20. Related Resources

Anatomy of a Modern Website

Why read it: Learn how real websites are structured before writing HTML.

How Browsers Work

Why read it: Understand how browsers parse HTML and build the DOM.

Responsive Web Design

Why read it: Learn how HTML structures adapt to different screen sizes.

Frontend vs Backend

Why read it: Understand where HTML fits within the complete web application architecture.

What Is a Full Stack Developer?

Why read it: See how HTML integrates into the broader full stack development workflow.

21. Recommended Next Learning Path

```
Step 1

HTML Fundamentals (Current PDF)
↓
Step 2

Semantic HTML: Writing Accessible and SEO-Friendly Web Pages
↓
Step 3

CSS Fundamentals
↓
Step 4

CSS Box Model
↓
Step 5

Flexbox
↓
Step 6

CSS Grid
↓
Step 7

Responsive Web Design
↓
Step 8

JavaScript Fundamentals
```

---

haas.dev • dev-roast-app.vercel.app