ScopeTreasure Courses

HTML COURSES · LESSON 1.7 · CODE SCOPE SERIES

Semantic HTML (header, nav, main, footer)

Semantic HTML explained simply: header, nav, main, section, article and footer — what each means, and why Google and screen readers love them.

Introduction

Until now you could build a whole page from `<div>` boxes alone — and beginners do exactly that. But a page full of anonymous boxes says nothing about itself: which box is the menu? Which is the real content? Semantic HTML fixes this by giving each part a tag whose name *is* its meaning. Every professional website you admire is built this way, and in this lesson you convert a div-soup page into a semantic page, part by part. *(Roman Urdu: Semantic tag ka naam hi us ka matlab hota hai — header matlab oopar wala hissa, footer matlab neeche wala.)*

Key points

- A **semantic tag** is a tag whose **name tells the meaning** of the content inside it. `<footer>` means "the bottom part of the page" — the name itself explains it.
- The old way: build everything with `<div>` boxes and class names like `class="header"`. This is called **div soup** — the page looks fine to us, but the browser and Google see only anonymous boxes. Nothing tells them which part is the menu and which is the main content.
- The **semantic family** from the lesson:
  - `<header>` — the top part: logo, site title, main heading area.
  - `<nav>` — the navigation menu (the main links of the site).
  - `<main>` — the main content of the page. **Only one** `<main>` per page.
  - `<section>` — one themed part of the content (a chapter of the page).
  - `<article>` — a complete piece that makes sense on its own: a blog post, a news story.
  - `<aside>` — side content, like a sidebar.
  - `<footer>` — the bottom part: copyright, small links, contact info.
- **The look does not change.** A semantic page and a div page can look exactly the same — CSS does the design. What changes is the **meaning**.
  (Roman Urdu: Semantic tags se page ki shakl nahi badalti, matlab saaf ho jata hai.)
- **Why it matters — the lesson's 3 reasons:**
  1. **Google / SEO** — Google understands which part is the menu, the main content, the footer — and ranks a clear page better.
  2. **Accessibility** — screen readers can jump straight to the `<nav>` or `<main>`, so blind users can travel the page quickly.

3. **Clean code** — any developer (including future you) can read the page structure at a glance.

- The conversion is simple: look at each div, ask "what IS this part?" and replace the div with the tag that answers the question. There is no new syntax to memorise — no new attributes, no new rules. You already know how tags work; semantic tags are simply honest names for the parts you were already building.

Code examples

Example 1 — Div soup vs semantic (the lesson's conversion)

```
<!-- Before: div soup -->
<div class="header">...</div>
<div class="menu">...</div>
<div class="content">...</div>
<div class="bottom">...</div>

<!-- After: semantic -->
<header>...</header>
<nav>...</nav>
<main>...</main>
<footer>...</footer>
```

What it does: Same page, same design — but now any person, browser, or search engine can read the structure from the tag names alone. No need to guess what `class="bottom"` means.

What you will see: on the screen — honestly, *nothing different at all*. Both versions can look identical, and that is the lesson's point: the change is not in the look, it is in the reading. Open the code of both and the second one explains itself line by line, while the first one needs you to open every div and guess.

Example 2 — A complete semantic blog page

```
<header>
  <h1>Ali's Blog</h1>
  <nav>
    <a href="index.html">Home</a>
    <a href="about.html">About</a>
  </nav>
</header>
<main>
  <article>
    <h2>My First Day at University</h2>
    <p>Today I joined BS Computer Science...</p>
  </article>
  <aside>
    <p>About me: I am Ali, a student from Lahore.</p>
  </aside>
</main>
<footer>
  <p><small>© 2026 Ali's Blog</small></p>
</footer>
```

What it does: A whole page in the semantic family: the header carries the blog's name and its menu, `<main>` holds one complete blog post (`<article>`) plus a small side note (`<aside>`), and the footer closes the page.

What you will see: a plain, honest page — the blog title big at the top, two links under it, the post heading and its paragraph, a short side note, and a small copyright line at the bottom. The beauty is invisible on screen: press View Source and the whole page reads like a table of contents.

Example 3 — Sections: chapters inside one main

```
<main>
  <section>
    <h2>Our Courses</h2>
    <p>HTML, CSS and JavaScript courses for beginners.</p>
  </section>
  <section>
    <h2>Student Reviews</h2>
    <p>Our students say the lessons are simple and practical.</p>
  </section>
</main>
```

What it does: One `<main>`, divided into two themed chapters with `<section>`. Each section has its own `<h2>` and its own topic — courses in one, reviews in the other — instead of one long undifferentiated flow.

What you will see: two clear blocks on the page, each starting with its own heading — "Our Courses", then "Student Reviews". A reader (and Google) can find the reviews chapter without reading the courses chapter first.

Quick recap

- A semantic tag's **name tells the meaning** of its content — `<nav>` is the menu, `<footer>` is the bottom.
- A page built only of `<div>` boxes is **div soup**: fine to look at, silent about its own structure.
- The family: `<header>`, `<nav>`, `<main>` (only one per page), `<section>`, `<article>`, `<aside>`, `<footer>`.
- The **look does not change** — CSS owns the look. What changes is the meaning.
- Three wins: Google understands the page (SEO), screen readers can jump between parts (accessibility), and humans read the code at a glance (clean code).

Common mistakes

1. **Using semantic tags for their look** — for example using `<header>` everywhere because its text "looks bold". The tags mostly look plain; their job is meaning, not style.
2. **More than one** `<main>` — a page has exactly one main content. Sections inside it are `<section>` or `<article>`.
3. **Semantic outside, div soup inside** — a `<main>` full of anonymous divs when the parts are really an `<article>` and an `<aside>`. Use the meaning tag at every level where one fits.

Practice

1. Take your first page from Lesson 1.1 and rebuild it with a semantic structure: `<header>` with your name, a `<nav>` with 3 links, a `<main>` with your paragraphs, and a `<footer>` with a copyright line.
2. Open any news or college website, view its page source (right-click → View Page Source), and find its `<header>`, `<nav>`, `<main>` and `<footer>`. Note which parts of the visible page each one holds.

Practice quiz

1. What does <nav> hold on a page?

- The navigation menu, the main links of the site
- The copyright line at the bottom
- A sidebar note
- The main article text

Answer: The navigation menu, the main links of the site

2. How many <main> elements may one page have?

- As many as you like
- Exactly one
- Two, one per section
- None, main is only a class name

Answer: Exactly one

3. What changes when you replace div soup with semantic tags?

- The look of the page changes completely
- Nothing changes at all, not even the meaning
- The look stays the same, but the meaning becomes clear
- The page stops working without CSS

Answer: The look stays the same, but the meaning becomes clear

---

Watch the video lesson: https://youtu.be/hH0R9qkwX30

Read online: https://scopetreasure.com/notes/semantic-html · More free tools and notes at https://scopetreasure.com