HTML & CSS

A Clear Guide
to Modern Websites,
Responsive Design,
Accessibility,
Testing and Publishing

How Websites Work and Workspace Setup

A website can look simple on the screen, but several systems work together behind it. The browser requests resources, reads HTML, applies CSS, creates an internal document structure, calculates layout, and draws the result. More complex websites can also communicate with servers, databases, and external services.

This chapter explains that foundation. The aim is not to perform setup tasks. The aim is to understand the parts of a web project and the roles they play.

1.1 Websites, Webpages, Web Applications, and Static and Dynamic Content

A webpage is one document or view presented in a browser. A news article, product page, contact page, and account page can each be a webpage. A website is a collection of related webpages and resources under one identity. Those resources can include HTML documents, stylesheets, images, fonts, audio, video, and scripts.

A web application is a web-based product designed mainly for tasks and ongoing interaction. Webmail, online banking, booking systems, and project-management tools are common examples. The border between a website and a web application is not absolute. A single product can contain both information pages and application-like areas.

Static delivery means the server sends a resource that already exists. For example, a stored file named about.html can be sent in the same form to every visitor.

Dynamic generation means the server creates a response when a request arrives. An online store can read product information from a database, combine it with a template, and generate the HTML for that product page.

A real website can combine several approaches. A logo may be a static image, a product page may be generated on the server, and a shopping-basket total may change in the browser with JavaScript. A useful distinction is whether the final HTML was prepared before the request or generated because of the request.

1.2 Front End, Back End, Databases, APIs, and the Roles of HTML, CSS, and JavaScript

The front end is the part of a web product handled by the browser. It includes the interface that people see and operate.

HTML, CSS, and JavaScript have different responsibilities.

```
<h1>Community Workshop</h1>
<p>Learn basic bicycle maintenance.
</p>
```

HTML describes the structure and meaning of content. In this example, the first line is a main heading and the second line is a paragraph.

```
h1 {
    color: #205c32;
}
```

CSS controls presentation. It can change colour, typography, spacing, borders, layout, and responsive behaviour. It does not change the meaning of the h1 element.

JavaScript is a programming language that can add programmed behaviour. It can react to events, update content, perform calculations, and communicate with servers.

The back end is software that runs on a server. It can process requests, check permissions, work with databases, and return responses.

A database stores structured information. An online store may keep products, customers, and orders in a database.

An API, or Application Programming Interface, is a defined way for software systems to communicate. A weather website, for example, may request forecast data from a weather API and then present the result in its own interface.

HTML and CSS are essential front-end technologies, but they are not replacements for server logic, databases, or application programming.

1.3 Browsers, Rendering, the DOM, and How Source Files Become a Visible Page

A browser does more than display a file. It interprets several resources and turns them into a rendered page.

A simplified process is:

```
HTML source
    ↓
DOM
    +
CSS rules
    ↓
layout and painting
    ↓
visible page
```

The **DOM**, or Document Object Model, is a tree-like representation of the document created by the browser.

For example:

```
<main>
    <h1>Repair Guide</h1>
    <p>Bring one item.</p>
</main>
```

can be represented conceptually as:

```
main
├── h1
│    └── text
└── p
     └── text
```

The browser also reads CSS and determines which rules apply to which elements. It then calculates sizes and positions and paints pixels to the screen.

Browsers contain error-recovery rules. Incorrect HTML can still produce a visible page because the browser repairs or interprets the source as best it can. A visible result is therefore not proof that the source is correct.