// WEB ENGINEERING FUNDAMENTALS — PDF 09

How Browsers Work.

From URL to interactive webpage — how browsers turn code into the sites you use every day.

// TABLE OF CONTENTS

Inside this guide

01 — INTRODUCTION

A website does not magically appear on your screen

Your browser downloads different files, understands their contents, combines them, and finally displays a webpage you can interact with — all in a few seconds.

Most beginners think browsers simply "open websites." In reality, browsers are powerful software applications capable of downloading resources, interpreting programming languages, executing JavaScript, managing storage, enforcing security, and rendering complex user interfaces.

// WHY THIS MATTERS

Understanding how browsers work will help you write faster, more reliable, and more efficient web applications.

02 — WHAT IS A WEB BROWSER?

Software that turns a request into something you can see

A web browser is software that allows users to access and interact with websites on the internet. It communicates with web servers, downloads website resources, and converts those resources into visual webpages.

// POPULAR BROWSERS

Google Chrome, Microsoft Edge, Mozilla Firefox, Apple Safari. Although these browsers look different, they all perform the same fundamental job.

03 — WHY ARE BROWSERS IMPORTANT?

The bridge between users and the web

Without browsers: websites could not be viewed, JavaScript could not run, users could not interact with web applications, online forms would not work, and images and videos would not load.

04 — THE MAIN PARTS OF A BROWSER

Six components, one seamless experience

USER INTERFACE

Everything you can see — address bar, back button, tabs, bookmarks.

BROWSER ENGINE

Coordinates communication between components; manages the overall workflow.

RENDERING ENGINE

Reads HTML and CSS and converts them into visible content. Differs per browser.

JAVASCRIPT ENGINE

Executes JavaScript code — without it, sites have very limited interactivity.

NETWORKING LAYER

Downloads HTML, CSS, JavaScript, images, and fonts from web servers.

STORAGE

Cookies, local storage, session storage, cache — improves performance and UX.

05 — HOW BROWSERS WORK

Eleven steps, each depending on the last

1. Type URL
2. Find server (DNS)
3. Connect to server
4. Send HTTP request
5. Receive response
6. Download HTML
7. Download CSS
8. Download JavaScript
9. Build webpage
10. Display website
11. Respond to user actions

// GO DEEPER

Read **"What Happens When You Type a URL?"** to understand the networking steps before the browser begins rendering.

06 — THE BROWSER RENDERING PROCESS

Converting code into pixels — eight stages

Only after completing every stage below does the webpage actually become visible.

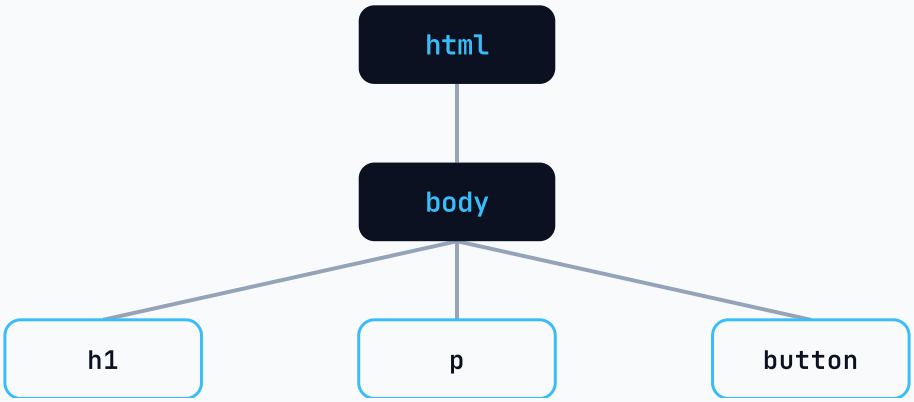
01 Download HTML	02 Parse HTML	03 Build the DOM	04 Download CSS
05 Build the CSSOM	06 Render Tree	07 Calculate Layout	08 Paint Pixels

Each stage feeds directly into the next — skip one, and the browser has nothing to paint.

07 — UNDERSTANDING THE DOM

Document Object Model — HTML becomes a structured tree

When the browser reads HTML, it does not simply display it. Instead, it creates a structured tree representing every HTML element.



This tree allows JavaScript to change text, create elements, remove elements, and update styles. The DOM is what makes webpages dynamic.

08 — UNDERSTANDING THE CSSOM

The browser builds a second tree — this time from CSS

This is called the CSS Object Model (CSSOM). The CSSOM contains all styling information.

CSSOM STORES

— Colors

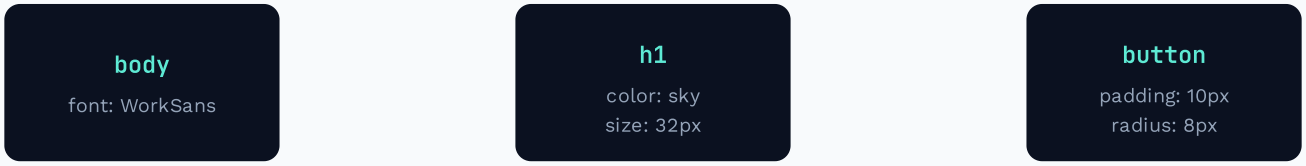
— Fonts

— Spacing

— Positioning

WITHOUT IT

Webpages would appear as plain, unstyled HTML — structure with zero visual design.



09 — THE RENDER TREE

Two trees combine into one — only what's visible survives

The Render Tree contains only the elements that should actually appear on the screen. Hidden elements are usually excluded.

// WHY IT MATTERS

This optimization improves rendering performance — the browser doesn't waste time laying out and painting things nobody will ever see.

10 — LAYOUT AND PAINTING

Two final steps before anything hits the screen

LAYOUT

The browser determines the size of every element, the position of every element, and the relationship between elements.

PAINTING

Once layout is complete, the browser paints every pixel onto the screen.

// RUNS MORE THAN ONCE

Whenever you resize a browser window or change CSS, layout and painting may run again — this is why heavy, repeated layout changes can feel janky.

11 — JAVASCRIPT AND THE BROWSER

What makes a webpage interactive, not just visible

JavaScript can update text, validate forms, display animations, communicate with servers, and load new content without refreshing the page.

```
// A TRADEOFF TO KNOW
```

Because JavaScript can modify the DOM, browsers sometimes pause rendering while executing JavaScript. This is one reason developers try to write efficient JavaScript.

12 — BROWSER CACHE

Reusing what's already been downloaded

Browsers store frequently used resources locally — images, CSS, JavaScript, fonts, and previously visited pages. The next time you visit the same website, the browser may reuse these resources instead of downloading them again.

```
// BENEFITS
```

Faster loading, reduced bandwidth usage, lower server load — all from not re-fetching what's already sitting on your device.

13 — REAL-WORLD EXAMPLE

Visiting https://dev-roast-app.vercel.app

The browser:

- Performs DNS lookup
- Connects to the server
- Downloads HTML
- Discovers linked CSS files
- Downloads JavaScript
- Builds the DOM
- Builds the CSSOM
- Creates the Render Tree
- Calculates layout
- Paints the webpage
- Waits for your interactions

Every modern website follows a similar rendering process.

14 — COMMON BEGINNER MISTAKES

Where most beginners get stuck

- × Thinking browsers only display HTML.
- × Ignoring CSS and JavaScript rendering.
- × Confusing the DOM with HTML.
- × Believing browsers download everything one file at a time.
- × Writing inefficient JavaScript that blocks rendering.

15 — PRACTICAL ACTION PLAN

Trace your favorite website's rendering pipeline

- Which files are likely downloaded first?
- Which resources might be cached?
- Which elements are probably created dynamically using JavaScript?

Draw the rendering pipeline from memory. If you can explain it without looking at this PDF, you've understood one of the most important concepts in frontend development.

16 — KEY TAKEAWAYS

What to carry forward

- Browsers are software applications that display websites.
- Browsers download HTML, CSS, JavaScript, and other resources.
- HTML becomes the DOM.
- CSS becomes the CSSOM.
- The DOM and CSSOM combine into the Render Tree.
- Layout determines positions.
- Painting displays pixels on the screen.
- JavaScript makes webpages interactive.

// 17 — SUMMARY PAGE

Cheat sheet

The whole guide, compressed to eight lines.

browser

Downloads website resources

html

Creates the DOM

css

Creates the CSSOM

dom + cssom

= Render Tree

layout

Calculates positions

painting

Displays pixels

javascript

Updates the DOM

browser cache

Improves loading speed

18 — BROWSER RENDERING CHECKLIST

Before you move to Part 10

- [ ] I know what a browser does.
- [ ] I understand the rendering pipeline.
- [ ] I know what the DOM is.
- [ ] I understand the CSSOM.
- [ ] I know why the Render Tree exists.
- [ ] I understand Layout and Painting.
- [ ] I know how JavaScript interacts with webpages.

// 19 — RELATED RESOURCES

Keep going

why read it

What Happens When You Type a URL?

Understand how browsers retrieve website resources before rendering begins.

why read it

HTTP vs HTTPS

Learn how browsers communicate securely with servers.

why read it

What Is DNS?

Understand how browsers locate websites.

why read it

What Is Web Hosting?

Learn where browsers retrieve website files from.

why read it

Frontend vs Backend

Understand which parts of a website run inside the browser and which run on the server.

// 20 — RECOMMENDED NEXT LEARNING PATH

Where to go from here

1. What Is a Website?
2. Website vs Web Application
3. How the Internet Works
4. What Happens When You Type a URL?
5. HTTP vs HTTPS
6. What Is a Domain Name?
7. What Is DNS?
8. What Is Web Hosting?
9. **How Browsers Work — you are here**
10. Cookies, Sessions & Local Storage

haas.dev

Engineering mindset over syntax memorization. Learn to think like a systems builder, one fundamental at a time.

haas.dev