// WEB ENGINEERING FUNDAMENTALS — PDF 04

What Happens When You Type a URL?

A step-by-step guide to how websites load — from DNS lookup to the moment pixels hit your screen.

// TABLE OF CONTENTS

Inside this guide

01 — INTRODUCTION

Opening a website looks simple. It isn't.

You type a URL, press Enter, and within seconds the website appears. Behind those few seconds, dozens of operations happen automatically.

Your browser talks to DNS servers, finds the correct server, sends an HTTP request, waits for a response, downloads multiple files, and finally builds the webpage you see on your screen.

Understanding this process helps developers troubleshoot website problems, improve performance, and understand how frontend and backend systems actually communicate. This PDF explains every step in simple language.

02 — WHAT IS A URL?

The complete address of a resource on the internet

A URL (Uniform Resource Locator) tells your browser which server to contact, which resource to request, and which communication protocol to use. Without a URL, the browser wouldn't know where to find the webpage.

```
// EXAMPLE URL

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

This URL tells the browser exactly where the requested resource is located.
```

03 — WHY IS IT IMPORTANT?

Every website visit starts with a URL

Search on Google, open YouTube, visit haas.dev, log into Gmail — the browser first processes a URL. Understanding URLs helps developers understand website navigation, routing, APIs, SEO, deployment, and debugging.

04 — UNDERSTANDING THE PARTS OF A URL

Breaking down one address, piece by piece

```
https://dev-roast-app.vercel.app/resources
```

PROTOCOL

https

Defines how the browser communicates with the server.

DOMAIN NAME

dev-roast-app.vercel.app

The human-readable address of the server.

PATH

/resources

Tells the server which specific resource the browser wants.

Some URLs also contain query parameters, fragments, and ports — these will be covered in later PDFs.

05 — WHAT HAPPENS WHEN YOU PRESS ENTER?

Eleven steps — completed in a blink by modern browsers

1. User enters URL
2. Browser checks local cache
3. DNS lookup
4. IP address found
5. Browser connects to server
6. HTTP request sent
7. Server processes request
8. HTTP response returned
9. Browser downloads files
10. Browser builds webpage
11. Website becomes visible

06 — STEP 1: BROWSER CHECKS THE URL

Before contacting the internet, the browser checks itself first

It looks in the browser cache, DNS cache, and local resources. If the required data already exists, loading becomes much faster.

// WHY CACHE MATTERS

Without caching, every website visit would require downloading everything again from scratch. Caching reduces loading time, network usage, and server load — for everyone.

07 — STEP 2: DNS LOOKUP

Browsers understand IP addresses. Humans remember names.

DNS converts a domain name into its corresponding IP address.

```
dev-roast-app.vercel.app
↓
IP Address
— now the browser knows which server to contact.
```

// GO DEEPER

For the complete explanation, read **"What Is DNS?"** — next in this series.

08 — STEP 3: CONNECTING TO THE SERVER

The browser establishes a connection using the IP it just found

If the website uses HTTPS, a secure encrypted connection is created before any data is exchanged. This ensures information cannot easily be intercepted.

// RELATED GUIDE

Read **"What Is HTTP & HTTPS?"** to understand secure communication in depth.

09 — STEP 4: SENDING AN HTTP REQUEST

"Please send me this webpage."

The browser sends a request asking for the webpage. A request typically includes:

- Requested path
- Browser information
- Accepted file types
- Cookies (if available)

10 — STEP 5: SERVER PROCESSES THE REQUEST

The server does the actual work

Depending on the website, it may locate HTML files, run backend code, query databases, check user authentication, or generate dynamic content. Once everything is ready, the server creates a response.

11 — STEP 6: RECEIVING THE HTTP RESPONSE

The response arrives as a bundle of files and a status

The response usually contains HTML, CSS, JavaScript, images, fonts, and a status code.

200

Success

404

Page not found

500

Internal server error

The browser begins downloading these resources.

12 — STEP 7: BROWSER RENDERS THE WEBPAGE

The browser can't display a page until it understands it

1. Download HTML
2. Read HTML structure
3. Download CSS
4. Apply styles
5. Download JavaScript
6. Execute scripts
7. Download images
8. Display complete webpage

Only after these steps does the final webpage appear.

13 — REAL-WORLD EXAMPLE

Visiting the haas.dev Resources page

```
https://dev-roast-app.vercel.app/resources
```

The browser:

- Checks cache
- Performs DNS lookup
- Finds the server
- Establishes a secure HTTPS connection
- Requests the Resources page
- Receives HTML, CSS, JavaScript, and images
- Downloads all required files
- Builds the webpage
- Displays the Resources page on your screen

All of this typically happens in less than a few seconds.

14 — COMMON BEGINNER MISTAKES

Where most beginners get stuck

- × Thinking the browser immediately displays websites.
- × Believing HTML alone creates the entire webpage.
- × Ignoring DNS and server communication.
- × Confusing URLs with domain names.
- × Assuming images are included directly inside HTML.

15 — PRACTICAL ACTION PLAN

Trace three websites you use today

For each one, explain:

- What URL did you enter?
- Which server received the request?
- What files do you think were downloaded?
- Which parts were likely generated by the backend?

Try explaining the complete request lifecycle without looking at this PDF.

16 — KEY TAKEAWAYS

What to carry forward

- A URL tells the browser where to find a resource.
- Browsers first check local cache.
- DNS converts domain names into IP addresses.
- Browsers communicate with servers using HTTP or HTTPS.
- Servers process requests and send responses.
- Browsers build webpages by combining HTML, CSS, JavaScript, and other resources.

// THE TAKEAWAY

Next time a page loads slowly, you'll know exactly which of these steps to suspect first — that's the whole point of learning the lifecycle.

// 17 — SUMMARY PAGE

Cheat sheet

The whole guide, compressed to eight lines.

`url`
Identifies a web resource

`browser`
Checks cache first

`dns`
Finds the server

`connection`
Browser connects to server

`request`
HTTP request is sent

`server`
Generates a response

`download`
Browser downloads required files

`render`
Browser renders the webpage

18 — REQUEST LIFECYCLE CHECKLIST

Before you move to Part 5

- [ ] I understand what a URL is.
- [ ] I know why DNS is required.
- [ ] I understand how browsers contact servers.
- [ ] I know what an HTTP request is.
- [ ] I understand how servers generate responses.
- [ ] I can explain how browsers render webpages.

If you can check all six honestly, you're ready for the next PDF in this series.

// 19 — RELATED RESOURCES

Keep going

why read it

What Is a Website?

Learn the overall structure of websites before understanding request lifecycles.

why read it

How the Internet Works

Understand how information travels across networks.

why read it

What Is DNS?

Learn how domain names are converted into IP addresses.

why read it

What Is HTTP & HTTPS?

Understand the protocol that makes browser and server communication possible.

why read it

How Browsers Work

Explore how browsers parse HTML, apply CSS, execute JavaScript, and render webpages.

// 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? — you are here**
5. What Is HTTP & HTTPS?
6. What Is DNS?

haas.dev

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

haas.dev