Understanding URLs

Uniform Resource Locator
The Address of the Web

A Comprehensive Guide

Git & GitHub Workshop

NEUB CSE Society

Prepared by: Mahfuz Ibne Syful

Table of Contents

1. What is a URL?
2. URL Structure
3. Protocol (Scheme)
4. Domain Name
5. Path
6. Query String (Parameters)
7. Fragment (Anchor)
8. URL Encoding
9. Absolute vs Relative URLs
10. URL Best Practices
11. Common URL Examples
12. Summary

1. What is a URL?

A **URL (Uniform Resource Locator)** is the address used to access resources on the internet. It's the text you type into your browser's address bar to visit a website or access a specific resource online.

```
https://www.github.com/Mahfuz2411/my-repo
```

URLs are used to:

- Access web pages and applications
- Load images, videos, and other media files
- Create links between different resources
- Make API requests to servers
- Download files from the internet

2. URL Structure

A URL consists of several components, each serving a specific purpose:

```
https://www.example.com:8080/path/to/page?key=value#section
```

Component	Example	Description
Protocol	`https://`	How to access the resource
Domain	`www.example.com`	Server address (hostname)
Port	`:8080`	Port number (optional, default 80/443)
Path	`/path/to/page`	Specific resource location
Query String	`?key=value`	Parameters (optional)
Fragment	`#section`	Section within page (optional)

3. Protocol (Scheme)

The protocol defines how to access the resource. It tells the browser which method to use for communication.

Common Protocols:

- `http://` - HyperText Transfer Protocol (unsecured)
- `https://` - HTTP Secure (encrypted with SSL/TLS) **✓ Recommended**
- `ftp://` - File Transfer Protocol
- `mailto:` - Email address link
- `file://` - Local file system access

> ⚠ **Security Note:** Always use `https://` when handling sensitive information. HTTPS encrypts the communication between your browser and the server, protecting data from interception.

4. Domain Name

The domain name represents the address of the server hosting the resource. It's a human-readable alternative to IP addresses.

```
www.github.com
```

Domain Structure:

- **Subdomain:** `www`, `blog`, `api`, `cdn` (optional prefix)
- **Domain:** `github`, `google`, `amazon` (main name)
- **TLD (Top-Level Domain):** `.com`, `.org`, `.io`, `.edu`

Examples:

- `github.com` - Main domain
- `api.github.com` - API subdomain
- `docs.github.com` - Documentation subdomain

5. Path

The path specifies the location of a specific resource on the server. It resembles a file system path.

```
https://github.com/Mahfuz2411/my-repo/blob/main/README.md
```

Path Characteristics:

- Separated by forward slashes (`/`)
- Case-sensitive on many servers (Linux/Unix)
- Can represent folders, files, or logical routes
- May not correspond to actual file system structure

Common Path Examples:

- `/users/profile` - User profile page
- `/products/123` - Product with ID 123
- `/api/v1/data` - API endpoint version 1
- `/blog/2024/01/post-title` - Blog post with date structure

6. Query String (Parameters)

Query strings pass additional data to the server as key-value pairs. They're commonly used for filtering, searching, and tracking.

```
https://example.com/search?q=git&lang=en&page=2
```

Query String Syntax:

- Starts with a question mark (`?`)
- Multiple parameters separated by ampersand (`&`)
- Format: `key=value`
- Special characters should be URL-encoded

Common Use Cases:

- **Search queries:** `?q=search+term`
- **Filters:** `?category=books&price=low`
- **Pagination:** `?page=2&limit=20`
- **Sorting:** `?sort=date&order=desc`
- **Tracking:** `?utm_source=email&utm_campaign=launch`

7. Fragment (Anchor)

Fragments point to a specific section within a page. They're processed by the browser, not sent to the server.

```
https://example.com/docs#introduction
```

Fragment Properties:

- Starts with a hash symbol (`#`)
- **NOT sent to the server** (client-side only)
- Jumps to element with matching `id` attribute
- Used for in-page navigation and single-page applications

> **Note:** Fragments are useful for creating table of contents, jump links, and bookmarking specific sections of long documents.

8. URL Encoding

Special characters in URLs must be encoded to ensure proper transmission. URLs can only contain a limited set of ASCII characters.

Why URL Encoding is Necessary:

URLs can only safely use specific characters from the ASCII set. Characters outside this range, or characters with special meaning in URLs (like `?`, `&`, `=`), must be converted to a percent-encoded format.

Common Encodings:

Character	Encoded	Usage
Space	`%20` or `+`	Query parameters
`?`	`%3F`	Query string delimiter
`&`	`%26`	Parameter separator
`=`	`%3D`	Key-value separator
`#`	`%23`	Fragment identifier
`/`	`%2F`	Path separator

Example:

```
Before encoding: hello world! how are you?
After encoding: hello%20world%21%20how%20are%20you%3F
```

9. Absolute vs Relative URLs

Absolute URL

A complete URL including protocol and domain. Can be used from any location.

```
https://github.com/Mahfuz2411/my-repo
```

Relative URL

A path relative to the current location. Used within the same website.

```
/about/team
../images/logo.png
contact.html
```

Relative URL Notation:

- `/` - Root of current domain
- `../` - Parent directory (go up one level)
- `./` - Current directory (can be omitted)
- `filename.html` - File in current directory

10. URL Best Practices

Do's:

- ✓ Keep URLs short, descriptive, and readable
- ✓ Use hyphens (`-`) to separate words, not underscores
- ✓ Use lowercase letters for consistency
- ✓ Make URLs meaningful and SEO-friendly
- ✓ Use HTTPS for security
- ✓ Keep URL structure consistent across your site

Don'ts:

- ✗ Don't expose sensitive data in URLs (passwords, tokens)
- ✗ Don't change URLs without proper redirects (breaks bookmarks)
- ✗ Don't use special characters unnecessarily
- ✗ Don't make URLs excessively long
- ✗ Don't use session IDs in URLs (security risk)

Good vs Bad Examples:

Good URL	Bad URL
`/blog/understanding-git`	`/page.php?id=12345`
`/products/laptop-dell-xps`	`/products?type=laptop&brand=Dell`
`/about/team`	`/about_us/our_team.html`

11. Common URL Examples

GitHub Repository:

```
https://github.com/Mahfuz2411/my-repo
```

Search Query:

```
https://www.google.com/search?q=git+tutorial&hl=en
```

API Endpoint:

```
https://api.github.com/users/Mahfuz2411/repos
```

Documentation with Anchor:

```
https://git-scm.com/docs/git-commit#_options
```

YouTube Video:

```
https://www.youtube.com/watch?v=dQw4w9WgXcQ
```

E-commerce Product:

```
https://example.com/products/electronics/laptop?color=silver&storage=512gb
```

12. Summary

Understanding URLs is fundamental to web development and internet usage. A URL is more than just an address—it's a structured format that communicates how, where, and what resource to access. By following best practices and understanding each component, you can create clean, secure, and user-friendly URLs that enhance both user experience and search engine optimization.

> **Key Takeaways:**
>
> - Always use HTTPS for secure communication
> - Keep URLs simple, readable, and descriptive
> - Understand the difference between absolute and relative URLs
> - Encode special characters properly
> - Never expose sensitive information in URLs

Understanding URLs - A Resource Document

Git & GitHub Workshop | NEUB CSE Society

Mentor: Mahfuz Ibne Syful