Free Website Guide

June 12, 2025

Contents

1 Preface **2**

1.1 Introduction . . . 2

1.2 Purpose of this Document . . . 2

1.3 Target Audience . . . 2

1.4 Scope of the Document . . . 3

1.5 Document Structure . . . 3

1.6 Assumptions and Prerequisites . . . 4

2 Setting up files and services **4**

2.1 Choosing a Hosting Service . . . 4

2.1.1 What is a Hosting Service? . . . 4

2.1.2 GitHub Pages . . . 4

2.2 Setting Up Your Local Development Environment . . . 5

2.2.1 Installing Git . . . 5

2.2.2 Installing a Code Editor . . . 6

2.2.3 Creating Your Project Files . . . 6

2.3 Connecting Your Local Repository to GitHub . . . 6

3 How Static Websites Work **7**

3.1 Definitions and Concepts . . . 7

3.2 How Static Websites Work . . . 7

3.3 Making your First Static Website . . . 8

3.3.1 Explanation of the Code . . . 9

1 Preface

This document will walk you through the process of creating a free, publicly accessible website. The guidance provided is intended to be straightforward and accessible, ensuring that even those with minimal technical expertise can follow along. Anyone with a basic understanding of how to use a computer and navigate the internet should be able to follow the instructions without too much difficulty. Readers may range from absolute beginners—those who have never written a single line of HTML—to experienced programmers who wish to streamline their workflow by leveraging free resources.

1.1 Introduction

This document serves as a comprehensive guide for anyone interested in creating a free website. It covers the essential steps and considerations involved in setting up a website without incurring any costs. The focus is on providing clear, step-by-step instructions that are easy to follow, regardless of the reader's prior experience with web development or hosting.

1.2 Purpose of this Document

This document aims to empower readers with the knowledge and tools necessary to establish an online presence without any initial financial investment. It will cover the fundamental aspects of web hosting, domain registration, and website creation, ensuring that readers can successfully launch their own websites using free resources available online.

1.3 Target Audience

This document is intended for a broad audience of individuals who wish to create a website without a budget for premium development tools or hosting. Specifically, it is suitable for:

- Students and educators who require a platform to showcase projects, portfolios, or learning materials.
- Hobbyists and content creators aiming to share writing, art, photography, or other personal work online.
- Small organizations, clubs, and community groups looking to establish a web presence.
- Aspiring developers who wish to gain practical experience with web development tools and workflows.
- Freelancers and independent professionals who seek to demonstrate services through a self-built website.

1.4 Scope of the Document

The scope of this document encompasses all phases of free website creation, placing primary emphasis on development tasks. It includes:

- **Planning and Requirements Gathering:** Defining purpose, target audience, content inventory, and user experience goals.
- **Technology Selection:** Comparing static site generators (e.g., Jekyll, Hugo), lightweight JavaScript frameworks (e.g., Vue, Svelte), or pure HTML/CSS approaches.
- **Development Environment Setup:** Installing and configuring free code editors (e.g., Visual Studio Code), package managers (e.g., npm), and version control systems (e.g., Git).
- **Design and Layout:** Crafting responsive page layouts using free CSS frameworks (e.g., Tailwind CSS, Bootstrap) or custom CSS methodologies (e.g., Flexbox, Grid).
- **Writing Code:** Authoring HTML templates, modular CSS, and JavaScript for interactive elements; structuring files and directories for scalability.
- **Content Management:** Incorporating markdown or other plain-text formats to author content, enabling straightforward updates without proprietary software.
- **Asset Optimization:** Compressing images, minifying scripts and styles, and lazy-loading resources to ensure rapid page load times on free hosting.
- **Testing and Debugging:** Utilizing free browser development tools (e.g., Chrome DevTools, Firefox Developer Edition) and open-source testing frameworks (e.g., Lighthouse, ESLint).
- **Deployment Overview:** Brief discussion of free deployment platforms (e.g., GitHub Pages, Netlify, Vercel), Continuous Integration/Continuous Deployment (CI/CD) pipelines, and basic DNS configuration for free custom domain services.
- **Maintenance and Updates:** Strategies for version control, content updates, security patches, and performance monitoring using free analytics tools.
- **Transition to Paid Services (Optional):** Guidelines for when and how to upgrade to a paid service for domain name registration, premium hosting, or advanced features, including:
  - Criteria for upgrading based on traffic, functionality requirements, or security concerns.
  - Comparison of paid hosting providers and domain registrars.
  - Migration steps from free to paid environments, ensuring minimal downtime.

1.5 Document Structure

The document is structured to guide readers through the process of creating a free website, starting from the initial planning phase to the final deployment and maintenance. Each section builds upon the previous one, ensuring a logical flow of information. Terms and concepts are introduced progressively, with practical examples and exercises to reinforce learning. The document also includes references to additional resources for readers who wish to explore specific topics in greater depth.

1.6 Assumptions and Prerequisites

This document assumes that readers have basic computer literacy, including the ability to navigate the internet, install software, use a web browser, and manage files on their local machine. Familiarity with basic programming concepts is helpful but not required. The document will introduce necessary technical terms and concepts as they arise, ensuring that all readers can follow along regardless of their prior knowledge.

2 Setting up files and services

The first step in creating a free website is to set up the necessary files and services. This section will guide you through the process of preparing your development environment, selecting a hosting service, and organizing your project files.

2.1 Choosing a Hosting Service

2.1.1 What is a Hosting Service?

A hosting service is a company that provides the infrastructure and technology needed to make your website accessible on the internet. They store your website's files on their servers and ensure that they are available to users who visit your site. Hosting services can vary widely in terms of features, performance, and cost. It is possible to host a website yourself, but for most users, especially those new to web development, using a hosting service is the most practical option. There are several free hosting services available that allow you to host your website without any cost. Some popular options include:

- **GitHub Pages:** Ideal for simple websites, it allows you to host your site directly from a GitHub repository.
- **Netlify:** Offers free hosting with continuous deployment from Git repositories, making it easy to update your site.
- **Vercel:** Similar to Netlify, it provides free hosting with a focus on performance and ease of use.
- **Firebase Hosting:** Great for dynamic web applications, it offers free hosting with a generous quota.
- **Surge:** A simple command-line tool for publishing static sites, ideal for quick deployments.

For this guide, we will focus on using GitHub Pages as it is widely used and integrates well with version control. However, the principles outlined here can be applied to any of the mentioned services. Make sure to sign up for an account with your chosen hosting service before proceeding.

2.1.2 GitHub Pages

GitHub Pages is a free hosting service provided by GitHub that allows you to host websites directly from a GitHub repository. It is designed to work seamlessly with Git, making it easy to deploy your site by simply pushing changes to your repository. GitHub Pages is an excellent choice for personal projects, portfolios, and documentation sites, as it supports custom domains and out-of-the-box HTTPS. It is well suited for developers, as it allows you to leverage Git's version control capabilities while providing a straightforward way to publish your work online.

Creating a GitHub Account To get started with GitHub Pages, you'll need a GitHub account. If you don't have one already, follow these steps to create an account:

1. Go to the GitHub website.
2. Click on the "Sign up" button in the upper right corner.
3. Fill out the registration form with your details and click "Create account".
4. Verify your email address by clicking the link sent to your inbox.
5. Once your account is created, you can log in to GitHub.

Creating a Repository A repository is where your website's files will be stored on GitHub. To create a new repository for your website, follow these steps:

1. Log in to your GitHub account here.
2. Click on the "+" icon in the upper right corner and select "New repository".
3. Enter a name for your repository (e.g., "my-website") and add a description if desired.
4. Leave the repository as "Public" as it is required for GitHub Pages to work for free hosting.
5. Everything else can be left as default, but you can choose to initialize the repository with a README file if you wish.
6. Click the "Create repository" button to finish.

2.2 Setting Up Your Local Development Environment

Before you can start building your website, you'll need to set up your local development environment. This involves installing the necessary software and tools that will allow you to create and manage your website files effectively.

2.2.1 Installing Git

Git is a version control system that allows you to track changes in your code and collaborate with others. It is essential for managing your website's source code, especially when using GitHub Pages. To install Git, follow these steps:

1. Go to the Git website.
2. Download the appropriate version for your operating system (Windows, macOS, or Linux).
3. Follow the installation instructions for your platform. During installation, you can choose the default options unless you have specific preferences.
4. Once installed, open a terminal or command prompt and verify the installation by typing `git -version`. You should see the installed version of Git displayed.

2.2.2 Installing a Code Editor

A code editor is a software application that allows you to write and edit your website's code. For this guide, we recommend using Visual Studio Code (VS Code) due to its popularity and extensive features. To install VS Code, follow these steps:

1. Go to Visual Studio Code website.
2. Download the version for your operating system (Windows, macOS, or Linux).
3. Follow the installation instructions for your platform.
4. Once installed, open VS Code and familiarize yourself with its interface. You can customize the settings and install extensions to enhance your coding experience.

2.2.3 Creating Your Project Files

Now that you have Git and a code editor installed, you can create the initial files for your website project. Follow these steps to set up your project structure:

1. Create a new folder on your local machine where you want to store your development files (e.g., `DEV`).
2. Inside the `DEV` folder, right-click and open VS Code. This will open the folder in the editor, allowing you to create and manage your project files easily.

2.3 Connecting Your Local Repository to GitHub

To connect your local repository to the GitHub repository you created earlier, you'll need to follow these steps:

1. Open a terminal or command prompt in your project folder (you can do this directly in VS Code by selecting "Terminal" from the top menu and then "New Terminal").
2. Type the following command to configure your Git username and email (replace with your own details):

   ```
   git config --global user.name "Your Name"
   git config --global user.email "your.email@example.com"
   ```

3. Next, clone your GitHub repository to your local machine by going to source control in VS Code, clicking on the "Clone Repository" button, and entering the URL of your GitHub repository
   (e.g., `https://github.com/yourusername/my-website.git`).
4. Once the repository is cloned, you will have a local copy of your GitHub repository in your project folder.

3 How Static Websites Work

3.1 Definitions and Concepts

Definition: *Static Website*

A static website is a type of website that delivers the same content to every user. It is built using HTML, CSS, and JavaScript, and the files are served directly to the user's browser without any server-side processing. Static websites are typically faster and more secure than dynamic websites, as they do not rely on databases or server-side scripts.

Definition: *HTML*

HTML (HyperText Markup Language) is the standard markup
language used to create the structure and content of web pages. It defines elements such as headings, paragraphs, links, images, and other multimedia content. HTML files are static and do not change unless manually edited.

Definition: *CSS*

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of HTML documents. It allows you to control the layout, colors, fonts, and overall appearance of your website. CSS files are also static and are linked to HTML files to apply styles.

Definition: *JavaScript*

JavaScript is a programming language that enables interactive features on web pages. It can manipulate HTML and CSS, respond to user actions, and perform calculations. While JavaScript can be used in static websites, it is often associated with dynamic websites where server-side processing is involved.

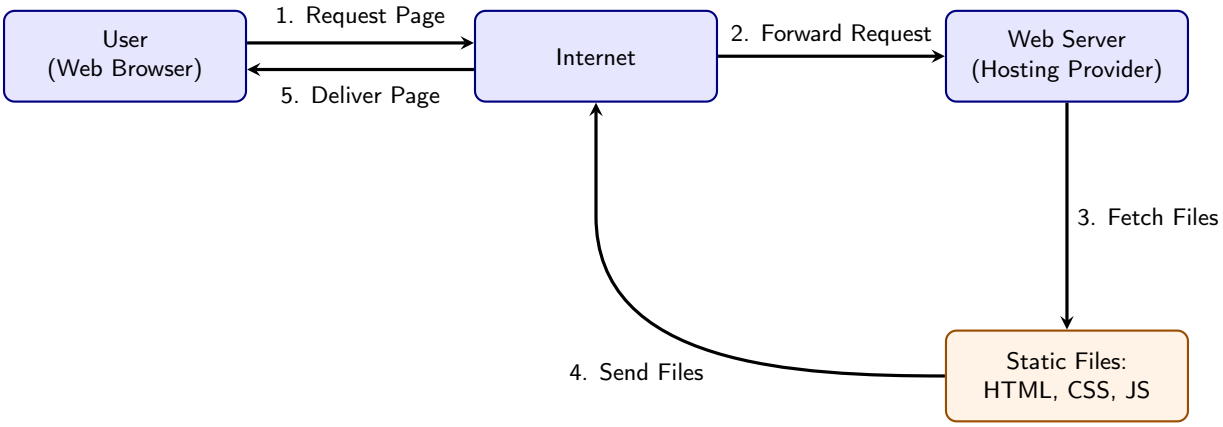
Concept: *Local Development Versus Live Website*

When developing a static website, you typically work on your local machine using a code editor. This allows you to create and test your website files before deploying them to a live server. Once you are satisfied with your local version, you can upload the files to a hosting service, making them accessible to users on the internet.

3.2 How Static Websites Work

Static websites work by serving pre-built HTML, CSS, and JavaScript files directly to the user's browser. When a user requests a static web page, the server retrieves the corresponding HTML file and sends it to the browser.
The browser then interprets the HTML, applies any linked CSS styles, and executes any JavaScript code to render the page. This process is straightforward and does not require any server-side

processing, making static websites fast and efficient.



This diagram illustrates the flow of a static website request:

1. The user opens a web browser and requests a specific page by entering a URL.
2. The request is sent over the internet to the web server hosting the static files.
3. The server retrieves the requested HTML file along with any associated CSS and JavaScript files.
4. The server sends these files back to the user's browser.
5. The browser renders the page, applying styles and executing scripts as needed.

3.3 Making your First Static Website

Before you can create your first static website, you need to understand some basic concepts. Below is a simple HTML structure that you can use as a starting point for your static website. This example includes the essential elements of an HTML document, such as the doctype declaration, head section, and body content. I will also provide a brief explanation of each part of the code.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>My First Static Website</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Welcome to My First Static Website</h1>
    </header>
    <main>
```

```
        <p>This is a simple static website created using HTML and
        CSS.</p>
    </main>
    <footer>
        <p>&copy; 2023 My First Static Website</p>
    </footer>
</body>
</html>
```

Listing 1: Basic HTML Structure

3.3.1 Explanation of the Code

Notice that all the lines so far start with a $<$ and end with a $>$. This is a fundamental characteristic of HTML, where elements are defined by tags.

Concept: *Tags in HTML*

Tags are the building blocks of HTML. They are used to define elements on a web page. Each tag typically consists of an opening tag (e.g., `<tagname>`) and a closing tag (e.g., `</tagname>`). The content between the opening and closing tags is what will be displayed on the web page. Some tags, like `<img>`, are self-closing and do not require a closing tag. example: `<img src="image.jpg" alt="Description">` where `src` is the source of the image and `alt` is the alternative text that describes the image.