Hyperiondev

TASK

Deployment of Static Websites

Visit our website

Introduction

In this task, you will push one of the **static web pages** (HTML or CSS) you created in the bootcamp to GitHub. From GitHub, you will use GitHub Pages to host your webpage and make it visible to the rest of the world. By the end of this task, you will have a URL that you can make available to others so that they can view your work.

STATIC VERSUS DYNAMIC WEB PAGES

Static web pages

A static web page is a website that shows the same content to every visitor. Unlike dynamic websites that generate content on the spot, static pages are pre-built HTML and CSS files that display information exactly as they were originally created. Each page remains unchanged until a web developer physically modifies the underlying code. This approach works best for websites that don't require frequent updates, such as personal portfolios, simple blogs, or straightforward informational sites where content remains relatively stable.

Key characteristics of static pages:

- No back-end or server-side processing.
- Faster to load since the content is pre-rendered.
- Easier to host (e.g., using GitHub Pages or similar services).
- Limited interactivity – no real-time data updates or personalised content.

Example: A personal portfolio site with basic text and images.

Dynamic web pages

Dynamic web pages are interactive websites that generate content in real-time. Unlike static pages, these sites use server-side programming languages like PHP, Python, or Node.js, and client-side scripts like JavaScript, to create personalised and changing content. They can retrieve information from databases, adapt to user interactions, and display unique content for each visitor. This makes dynamic websites more flexible and responsive, ideal for complex platforms like social media, e-commerce sites, or web applications that need to show personalized or constantly updated information.

Key characteristics of dynamic pages:

- Content is generated or modified in real time.
- Often requires a database and back-end framework, like Django or Laravel, and databases like MySQL to generate and manage interactive content.
- Useful for complex applications like social media platforms, e-commerce websites, or dashboards.
- Require more resources and hosting services that support server-side processing.

Example: A shopping site where product availability and pricing update dynamically based on user actions.

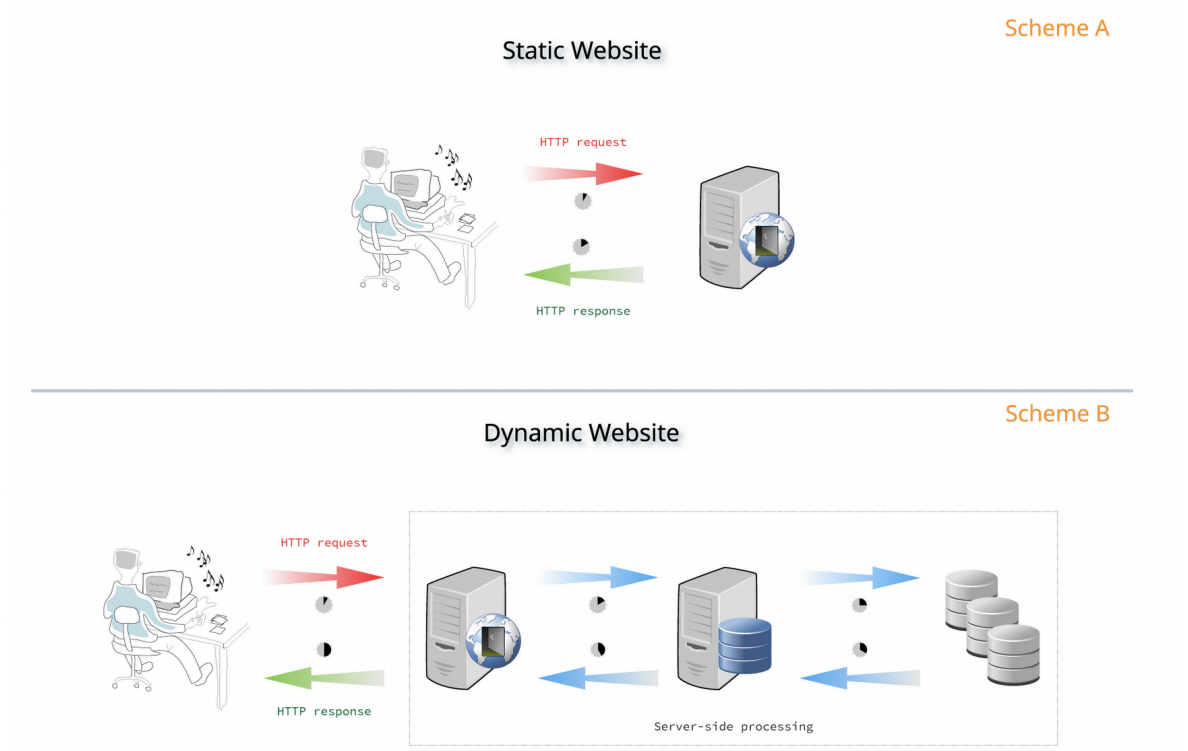

Illustration (Ramos, 2016)
https://about.gitlab.com/blog/2016/06/03/ssg-overview-gitlab-pages-part-1-dynamic-x-static/

The image just above illustrates the difference between static and dynamic websites. In a static website (Scheme A), the web server sends a pre-built HTML page directly to the user's browser when a request is made. This means the content remains the same for all users unless the website is manually updated. In a dynamic website (Scheme B), the server processes the request, often interacting with a database, before sending a tailored HTML page to the user. This allows for personalized content, real-time updates, and user interactions.

GitHub Pages

To make a webpage visible to the rest of the world, it needs to be hosted on a web server. There are various options for deploying web pages to web servers. In this task, you will be pushing your webpage to GitHub and using **GitHub Pages** to host your personal webpage.

GitHub allows you to host and easily publish web pages using GitHub Pages. GitHub Pages is available for **free** in public repositories.

There are certain restrictions regarding the use of GitHub Pages. As stated by **GitHub**, "GitHub Pages is **not intended** for or allowed to be used as a free web hosting service **to run your online business, e-commerce site**, or any other website that is primarily directed at either facilitating commercial transactions or providing commercial software as a service (SaaS)."

Besides that, GitHub lists the following usage limits for GitHub pages:

- "Published GitHub Pages sites may be no larger than 1GB.
- GitHub Pages sites have a soft bandwidth limit of 100GB per month.
- GitHub Pages sites have a soft limit of 10 builds per hour."

Practical task 1

In this task, you are going to be deploying **any static website** that you built in a previous task using GitHub Pages.

Follow these steps:

- Open GitHub and create a repository in which you will store your project. Remember to initialise your repository with a **README**. Make sure it is set to "**Public**" so that you can deploy your project to GitHub Pages.
- Ensure your main HTML file is named **index.html**, as GitHub Pages requires this to load your homepage. If it's named something else, please rename it before uploading.
- Upload your project files to the repository by clicking the "**Upload files**" button. Select all the necessary files, making sure your project structure is intact.

- To activate GitHub Pages, go to the **Settings** of your repository. In the Settings menu, click on or scroll down to the **Pages** section. Under "**Branch**", select `main` (the default branch). Save the changes to deploy your site. Note that it may take a few minutes for the deployment to be completed.
- Once your site has been deployed, you can visit it by navigating to:
    - https://**username**.github.io/**repoName**
    - (Replace **username** with your GitHub username, and replace **repoName** with the name of your repository.)
- For more information, go to **GitHub Pages** or the **GitHub documentation**.
- Take a screenshot of your deployed website that includes the URL in the image and save it as a .png or .jpg file called **static_page**.
- Create a text file named **static_url.txt** and paste the URL of your deployed site into it. Submit this file along with your screenshot.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Practical task 2

In this task, you are going to be creating a personal webpage. Make sure that this webpage acts as a CV and introduces you to the world effectively. Strike a balance in your content – this webpage should show more of your personality than a typical CV, but it should still be professional.

Follow these steps:

- Create an HTML page called **index.html**.
- On this page, add any elements you would like to use to create a webpage that acts as an online CV. This is your personal webpage, so feel free to customise it to suit your needs, but make sure that you include at least the following:
    - Write a brief description of yourself in no more than three paragraphs. Who are you? What experience do you have? What are your passions and motivations? What are your goals or aspirations?

    - Your contact details. For example: name, contact number, email address, and links to any of your (professional) social media, including LinkedIn.
    - A list of your skills and competencies.
    - Describe your education.
    - Describe your work experience.
    - **Optional:**
        - An image of yourself.
        - Add links to any blog posts or articles that you have written.
- Use CSS to style and position the elements on your webpage as attractively as possible. Feel free to use style libraries like **Bootstrap** if you wish.
- Create a new repository for this task called **MyCV**.
- Following the steps in practical task 1, deploy your webpage using GitHub Pages.
- Add the **index.html** file from this task into this task's folder, as well as a screenshot of your deployed webpage called **my_cv** (as a .png or .jpg) and a text file **my_cv.txt** with your webpage URL.

You can keep adding to this personal webpage. For example, in the future, you can detail projects that you have worked on and add links to the code (in GitHub).

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.