HTML is a markup language, not a programming language. Web browsers are designed to interpret and display HTML. Therefore, all you need is a text editor to create the HTML file and a web browser to view it.

Here's a breakdown of how to create, save, and view an HTML file:

1. **Create the HTML file:**
    - **Use a text editor:** You can use any plain text editor like Notepad (Windows), TextEdit (Mac), Sublime Text, VS Code, Atom, or any other code editor. *Do not use a word processor like Microsoft Word*, as these add extra formatting that will interfere with the HTML.
    - **Write the HTML code:** Start with a basic HTML structure. Here's a simple example:

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>My1 First Web Page</title>
</head>

<body>
 <h1>Hello, World!</h1>
 <p>This is my first web page.</p>
</body>

</html>
```

Let's break down this example:

* `<!DOCTYPE html>`: Declares the document type as HTML5.
* `<html lang="en">`: The root element of the HTML document. `lang="en"` specifies the language as English.
* `<head>`: Contains meta-information about the HTML document, such as character set, viewport settings, and title. This information is generally not displayed directly on the page.
* `<meta charset="UTF-8">`: Specifies the character encoding for the document (UTF-8 is recommended).
* `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Configures the viewport for proper rendering on different devices.
* `<title>My First Web Page</title>`: Sets the title of the web page, which appears in the browser tab.
* `<body>`: Contains the visible content of the web page.
* `<h1>Hello, World!</h1>`: A level 1 heading.
* `<p>This is my first web page.</p>`: A paragraph of text.

2. **Save the HTML file:**
    - **Choose a location:** Save the file to a location you can easily access.
    - **Save as HTML:** Crucially, save the file with an .html or .htm extension. For example, my_webpage.html. Make sure your text editor is saving it as plain text, not rich text.
3. **View the HTML file in a web browser:**
    - **Open with browser:** Locate the saved HTML file in your file system. Right-click on the file and choose "Open with" and then select your preferred web browser (Chrome, Firefox, Safari, Edge, etc.). You can also simply double-click the HTML file, and your default browser should open it.

- **That's it!** Your web browser will now render the HTML code and display the web page.

Key Points:

- **No server needed (for basic HTML):** For simple HTML files, you don't need a web server. Browsers can directly open and display HTML files from your local file system.
- **Inspect the code:** In your browser, you can right-click on the page and select "Inspect" or "Inspect Element" to view the HTML source code. This is a great way to learn how HTML works.
- **Learning more:** There are many excellent online resources for learning HTML. Look for tutorials and documentation on sites like MDN Web Docs (Mozilla Developer Network)

Webserver based execution:

It sounds like you want to use the http-server package (from Node.js) to serve your HTML file. Here's how to do that:

1. Make sure you have Node.js and http-server installed

- If you don't have Node.js installed, download and install it from nodejs.org.
- Open your terminal or command prompt and install http-server globally:

```
npm install -g http-server
```

2. Navigate to your HTML file's directory

- Use the cd command in your terminal to go to the folder where your HTML file is located. For example:

```
cd my_website
```

3. Start the http-server

- In your terminal, run the following command:

```
http-server
```

- This will start the http-server in the current directory. It will usually run on port 8080. You'll see a message in your terminal like:

```
Starting http-server on port: 8080
```

4. Access your HTML file in your browser

- Open your web browser and go to http://localhost:8080.
- If your HTML file is named index.html, it will be displayed automatically.
- If your HTML file has a different name (e.g., my_page.html), you'll need to include the filename in the URL: http://localhost:8080/my_page.html

Important Notes:

- **Stopping the server:** To stop the http-server, press Ctrl+C in your terminal.
- **Serving a specific file:** If you want to serve a specific HTML file, you can provide the filename as an argument:

```
http-server my_page.html
```

- **Changing the port:** If you need to use a different port (e.g., port 80), you can specify it with the -p option:

```
http-server -p 80
```

- **More options:** http-server has other options you can explore. Run http-server --help in your terminal to see the available options.

If you have any trouble, double-check that you're in the correct directory and that your HTML file is named correctly.

Deploy Website/page

Let's walk through deploying your webpage to Surge, GitHub Pages, and Netlify. These are all excellent options, each with slightly different strengths.

1. Surge (Easiest for simple static sites):

Surge is fantastic for quickly deploying static websites (HTML, CSS, JavaScript).

- **Install Surge:** If you don't have it already, install Surge globally using npm:
```
npm install -g surge
```
- **Navigate to your project directory:** In your terminal, use cd to navigate to the directory containing your HTML file (and any other assets).
- **Deploy:** Run the surge command:
```
surge
```

Surge will guide you through the process. It will ask you:

* **Project directory:** Usually, it detects the current directory correctly. Just press Enter if it's right.
* **Domain:** It will suggest a random subdomain (e.g., `something-random.surge.sh`). You can accept this or enter a custom subdomain if it's available. If you have a custom domain (e.g., `yourdomain.com`), you can configure that later (see Surge documentation).

- **Access your site:** Once the deployment is complete, Surge will provide you with the URL. Visit it in your browser.

2. GitHub Pages (Free hosting from GitHub repositories):

GitHub Pages is ideal if your website's code is already on GitHub. It's free and integrates seamlessly.

- **Create a GitHub repository:** If you haven't already, create a new repository on GitHub and upload your website's files to it.
- **Configure GitHub Pages:**
  - Go to your repository on GitHub.
  - Click on "Settings."
  - Scroll down to the "GitHub Pages" section.
  - In the "Source" dropdown, select "Deploy from a branch" (or "GitHub Actions" for more advanced setups).
  - Choose the branch you want to deploy from (usually main or master).
  - The root folder will be selected as / (root).
  - Save your changes.
- **Access your site:** GitHub Pages will provide you with a URL, usually in the format yourusername.github.io/yourrepositoryname. It might take a few minutes for the site to become live after the initial setup.

3. Netlify (More features, easier continuous deployment):

Netlify is a powerful platform that simplifies website deployment and offers many features like continuous deployment, serverless functions, and more. It has a generous free tier.

- **Create a Netlify account:** Sign up for a free account at netlify.com.
- **Connect to GitHub (or other Git provider):** Netlify makes it easy to deploy from your GitHub repository (or GitLab, Bitbucket, etc.).
- **Choose your repository:** Select the repository containing your website's code.
- **Configure build settings (usually automatic):** Netlify will typically auto-detect that it is a static website and configure the build settings automatically. If it doesn't, you may need to specify the publish directory (the folder containing your HTML file, usually the root folder).
- **Deploy:** Netlify will deploy your site.
- **Access your site:** Netlify will provide you with a URL. You can also configure a custom domain.

Which service to choose?

- **Surge:** Best for quick, simple deployments of static websites. Very easy to use.
- **GitHub Pages:** Best if your website's code is already on GitHub and you want a free and simple hosting solution.
- **Netlify:** Best for more advanced needs, continuous deployment, and features like forms, serverless functions, and more. It's very user-friendly but offers more power than Surge or GitHub Pages.

For a simple HTML file, Surge is the fastest way to get it online. If you're using version control with Git, GitHub Pages or Netlify are excellent choices. Netlify is highly recommended as you learn more about web development because it simplifies the deployment workflow and provides many useful features.