TR-102

MASTERING THE SEMANTIC WEB

DAY-2

❖ INTRODUCTION TO GITHUB

➢ What is GitHub?

- GitHub is a web-based platform used for version control and collaborative software development.
- It leverages Git, a distributed version control system, to track changes in code, facilitate collaboration among developers, and manage various projects.
- GitHub provides a user-friendly interface and robust features such as issue tracking, project management, and integration with other tools and services.

➢ What is a repository?

- A repository (or "repo") is a storage space where your project's files, including the codebase, documentation, and other essential resources, are kept.
- Each repository can be thought of as a project folder that contains all the necessary files and their revision history.
- Repositories can be public (accessible to everyone) or private (restricted access).

➢ How to create a repository using GitHub Desktop?

GitHub Desktop is a graphical interface that simplifies the use of Git and GitHub. Below are the steps to create one:

- **Download and Install GitHub Desktop:**
  - Visit the GitHub Desktop website.
  - Download the appropriate version for your operating system.
  - Follow the installation instructions.
- **Set Up GitHub Desktop:**
  - Open GitHub Desktop after installation.
  - Sign in to your GitHub account or create one if you don't have an account.
- **Create a New Repository:**
  - Click on the "File" menu and select "New Repository" or click the "Create New Repository" button on the main screen.
  - Fill in the repository details:
    - **Name:** Choose a name for your repository.
    - **Description (optional):** Provide a brief description of your project.
    - **Local Path:** Select the location on your computer where you want to store the repository.

        - **Git Ignore Template:** Choose a template to ignore specific files (optional).
        - **License:** Select a license for your project (optional).
- **Create the Repository:**
    - Click the "Create Repository" button.
    - Your new repository is now created and available on both your local machine and GitHub.
- **Add Files to the Repository:**
    - You can drag and drop files into the repository folder or use the "Add File" button.
    - Commit your changes by providing a commit message and clicking the "Commit to main" button.
- **Publish the Repository to GitHub:**
    - Click the "Publish repository" button.
    - Choose the visibility (public or private) and click "Publish Repository.

➢ Importance of using GitHub and repositories

- **Version Control:** GitHub allows tracking changes in the code, making it easier to manage different versions and collaborate without conflicts.
- **Collaboration:** Multiple developers can work on the same project simultaneously, review each other's code, and merge changes seamlessly.
- **Backup and Security:** Repositories serve as a backup of your project, ensuring that your code is not lost and can be accessed from anywhere.
- **Community and Open Source:** GitHub hosts millions of open-source projects, allowing developers to contribute, learn, and share their work with a global community.
- **Integration:** GitHub integrates with various development tools and services, enhancing productivity and streamlining workflows.

❖ INTRODUCTION TO CSS (CASCADING STYLE SHEETS)

➢ What is CSS?

CSS, which stands for Cascading Style Sheets, is a language used to describe the presentation of a document written in HTML or XML. CSS controls the layout, colors, fonts, and overall appearance of a web page, enabling developers to create visually appealing and well-structured websites.

➢ Key Features of CSS

- **Separation of Content and Presentation**: CSS allows developers to separate the content (HTML) from the presentation (CSS). This makes it easier to maintain and update the design without altering the content.
- **Reusability**: CSS can be used across multiple web pages, allowing for consistent styling and easier maintenance. A single CSS file can control the appearance of an entire website.

- **Flexibility and Control**: CSS provides fine-grained control over the layout and design of web pages, including positioning, spacing, colors, fonts, and responsive design for different devices.

➢ Basic Syntax

A CSS rule consists of a selector and a declaration block. The selector targets the HTML elements to be styled, and the declaration block contains one or more declarations separated by semicolons.

Syntax:

```
selector {
   property: value;
   property: value;
}
```

Example:

```
p {
   color: blue;
   font-size: 16px;
}
```

➢ How to apply CSS?

- **Inline CSS**: Directly within HTML elements using the style attribute.

Syntax:

```
<p style="color: blue; font-size: 16px;">This is a paragraph.</p>
```

- **Internal CSS:** Within a <style> tag inside the <head> section of an HTML document.

```
<head>
<style>
p {
color: blue;
font-size: 16px;
}
</style>
</head>
```

- **External CSS:** In an external .css file, linked to the HTML document using the <link> tag.

```
<head>
   <link rel="stylesheet" href="styles.css">
</head>
/* styles.css */
p {
   color: blue;
   font-size: 16px;
}
```

➢ CSS Selectors

CSS selectors are patterns used to select the elements you want to style. Some of the most common CSS selectors with their syntax and examples are:

- **Element Selector:** Selects HTML elements based on the element name.

  Syntax:

```
element {
   property: value;
}
```

  Example:

```
/* Selects all <p> elements */
p {
color: blue;
}
```

- **Class Selector:** Selects elements with a specific class attribute. Classes can be reused across multiple elements.

  Syntax:

```
.classname {
property: value;
}
```

Example:

```
/* Selects all elements with the class "example" */
.example {
  background-color: yellow;
}
```

- **ID Selector:** Selects a single element with a specific id attribute. IDs should be unique within a page.

  Syntax:

```
#idname {
  property: value;
}
```

  Example:

```
/* Selects the element with the id "unique" */
#unique {
  font-size: 20px;
}
```

- **Universal Selector:** Selects all elements on the page.

  Syntax:

```
* {
  property: value;
}
```

  Example:

```
/* Applies a border to all elements */
* {
  border: 1px solid black;
}
```

❖ The ' ' tag in HTML

➢ What is a Tag?

- The tag, short for "division," is a block-level container in HTML that is used to group and organize other elements.

- It serves as a generic container for content with no inherent styling or semantic meaning, but it is essential for applying CSS styles and JavaScript functionality to specific sections of a web page.

➢ Key Characteristics

- **Block-level Element**: The `<div>` element creates a block of content that takes up the full width of its parent container and starts on a new line.
- **Generic Container**: It can contain any other HTML elements, including text, images, links, and even other `<div>` tags.

➢ Basic Syntax

```
<div>
   <!-- Content goes here -->
</div>
```