Unit-1 Scripting

Intoduction

Web programming, also known as web development or web application development, is the process of creating and maintaining websites and web applications that are accessible via the internet.

It involves a combination of programming, design, and other skills to build websites and web-based software.

Web programming can range from simple static web pages to complex dynamic web applications.

Static web pages display the same content, while dynamic web pages display content that changes based on a user's location or actions.

Here are some key aspects of web programming:

- **Languages and Technologies:** Web programming involves using a variety of programming languages and technologies to create web content. Common languages and technologies include HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), JavaScript, server-side languages like PHP, Python, Ruby, Java, or Node.js, and databases like MySQL or MongoDB.
- **Front-End Development:** Front-end developers work on the user interface and user experience (UI/UX) of a website. They use HTML, CSS, and JavaScript to create web pages that are visually appealing and interactive. Front-end development is responsible for what users see and interact with in their web browsers.
- **Back-End Development:** Back-end developers work on the server-side of web applications. They handle tasks like database management, server configuration, and server-side scripting. Back-end development is responsible for the server's logic, data storage, and security.
- **Web Frameworks:** Developers often use web frameworks or libraries that provide pre-built components and tools for building web applications more efficiently. For example, web frameworks like Ruby on Rails, Django, Express.js, and AngularJS can accelerate development.
- **Database Integration:** Web programming often involves integrating databases to store and retrieve data. Common database management systems include MySQL, PostgreSQL, SQLite, and NoSQL databases like MongoDB.
- **Security:** Security is a critical aspect of web programming. Developers must implement security measures to protect against common web vulnerabilities, such as cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF).
- **Responsive Design:** Creating web content that is responsive and works well on various devices and screen sizes, from desktop computers to smartphones and tablets, is crucial for a positive user experience.

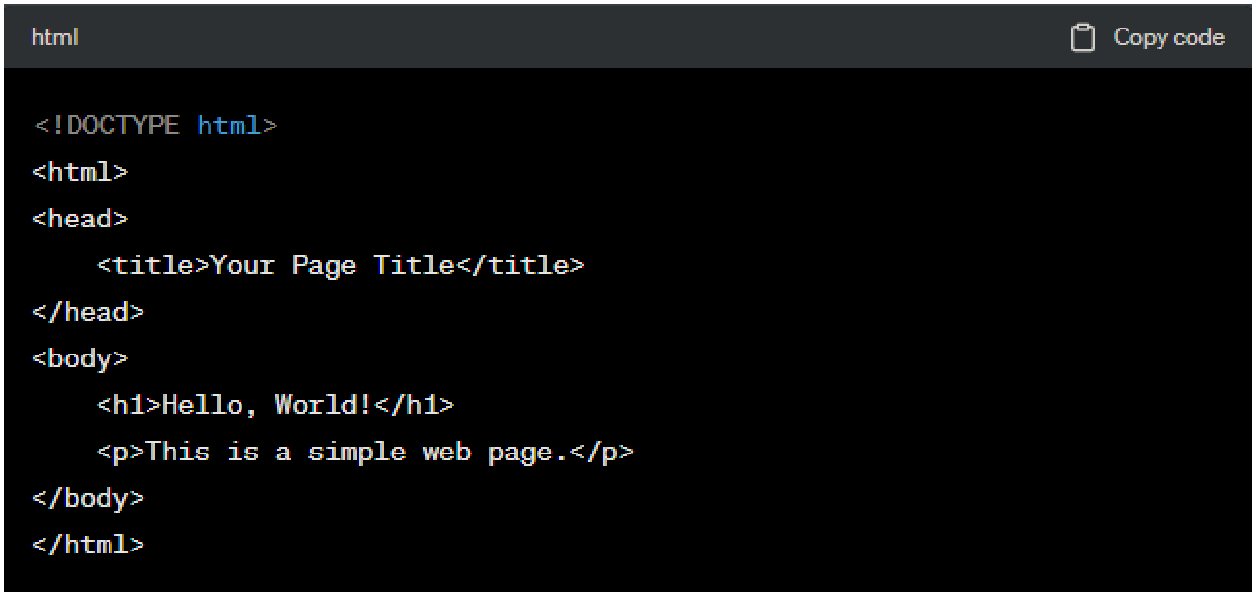
- **Testing and Debugging:** Web programmers test their code to identify and fix bugs and ensure that the website or application functions as intended. Various testing tools and techniques are used for this purpose.
- **Deployment and Hosting:** After development, web applications need to be deployed on web servers. Choosing a hosting platform and configuring servers is an essential part of web programming.
- **Maintenance and Updates:** Ongoing maintenance and updates are required to keep websites and web applications secure and up to date. This includes fixing issues, adding new features, and ensuring compatibility with the latest web technologies and standards.
- **Version Control:** Developers often use version control systems like Git to track changes to their code, collaborate with others, and manage codebase history.

Webpage Designing using HTML

HTML provides the structure and content of a web page.

The steps involved are:

1. **Create the HTML document:** Start by creating an HTML document. It typically begins with the **`<!DOCTYPE>`** declaration, followed by the **`<html>`**, **`<head>`**, and **`<body>`** elements. Here's a basic template:



```html
<!DOCTYPE html>
<html>
<head>
    <title>Your Page Title</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a simple web page.</p>
</body>
</html>
```

- **DOCTYPE declaration:** This declaration defines the document type and version of HTML you are using. In this case, it's HTML5.
- **`<html>` element:** This element is the root of the HTML document.

    - **`<head>` element:** The head section contains metadata about the document, such as the page title, character encoding, and links to external resources (e.g., stylesheets or scripts).
    - **`<title>` element:** The title of the page, displayed in the browser's title bar or tab.
    - **`<body>` element:** The body of the HTML document contains the visible content of the web page.
    - **Content within the `<body>`:** You can add various HTML elements within the **`<body>`** to structure and display your content. In the example, we've included an **`<h1>`** heading and a **`<p>`** paragraph.
2. **Save your HTML file:** Save the file with the ".html" extension (e.g., "index.html").
3. **Open in a web browser:** Double-click the HTML file you just created to open it in your web browser. You should see your simple web page.

Scripting

"Scripting" refers to the process of writing scripts, which are sets of instructions or code that automate tasks, perform specific functions, or control the behavior of software applications.
Scripts are usually written in scripting languages that are designed for specific purposes. Some commonly used scripting languages are: Javascript, Typescript, Python, Perl, etc.

Types of Scripting:

1)Client-Side Scripting: Client-side scripting refers to the execution of code on the user's device (typically in their web browser) as a response to user interactions with a web page. The primary language for client-side scripting is JavaScript. Here are the key characteristics and use cases of client-side scripting:

- **Execution Location:** Code is executed on the client's device, such as their web browser.
- **User Interface Interactivity:** Client-side scripts are commonly used to create interactive web applications. For example, validating form input, handling user interactions like button clicks, and updating the content of a web page without requiring a full server request.
- **Performance:** Client-side scripts can provide a more responsive user experience by offloading some processing tasks to the user's device, reducing the need for frequent server requests.
- **Security:** Since code is executed on the client side, it's important to be mindful of security concerns, such as protecting sensitive data and preventing malicious code injection.
- **Examples:** Dynamic web pages, web form validation, animations, and client-side data manipulation are examples of client-side scripting use cases.

2)Server-Side Scripting: Server-side scripting, on the other hand, involves executing code on the web server before sending the resulting HTML, CSS, and JavaScript to the client's browser. Various server-side scripting languages and frameworks are used, including PHP, Python (Django, Flask), Ruby (Ruby on Rails), Java (JavaServer Pages), and Node.js. Here are the key characteristics and use cases of server-side scripting:

- **Execution Location:** Code is executed on the web server that hosts the website.
- **Data Processing:** Server-side scripts are used to process and generate dynamic content, communicate with databases, and handle user authentication. They are ideal for tasks that require access to server resources and databases.
- **Security:** Server-side scripts are less vulnerable to client-side attacks because they run on the server. They can control access to sensitive data and protect against common web vulnerabilities like SQL injection.
- **SEO (Search Engine Optimization):** Server-side scripting can be beneficial for SEO since search engine crawlers can easily access and index server-generated content.
- **Examples:** Content management systems (CMS), e-commerce websites, user authentication, and server-side data processing are examples of server-side scripting use cases.

JavaScript

Events:

The Event Generating Components, such as button, checkbox, textfield, scrollbar, etc... are termed as the sources.

The changing state of a source is known as an event.

An event can be occurred with or without the user interaction with the sources.

An Event occurred with user interaction, is due to the action performed by the user, by using I/O devices, such as key-press, mouse hover, mouse click, etc...

An Event occurred implicitly (without user interaction), can be due to the pre-loaded instructions, such as when timer expires, a counter reaches a value, etc...

Windows:

Documents:

Frames:

Build-in Functions:

Browser Object Model:

Forms: (Verifying Forms)

HTML & CSS

HTML5:

CSS3:

HTML5 Canvas:

Tools used for Web-site creation: