My Whole Journey in Bytewise Fellowship Program:

Assalamualaikum!

This is Parihan Ayyaz Front-end Bytewise Fellow. The Bytewise Fellowship Journey has been very challenging for me not just because of the complexity of tasks but because of the pressure of a lot of things like fear of losing the competition, exams preparation for the Top position, fighting with circumstances, keep myself motivated, had to do job for meeting my financial needs, have been working as former GDSC-UE Documentation Lead and currently working as Documentation Lead for UE IT Society. Due to above reasons I couldn't perform 100% but I tried to complete all the tasks. But in the Bytewise fellowship program, I had a chance to test myself to what extent I can work hard and is that field suitable for me or not. I gained a lot of knowledge as I had hands-on experience. A fun fact, I did basic HTML and CSS from my university through a bootcamp and thought I have become a developer but when I entered the Bytewise Fellowship program, I realized there are a lot of things that I have not learned earlier. So in my opinion, this fellowship program was really amazing and worthy in mentorship of our mentor Sir Zubair, a very humble and nice person who understood us, led in a polite way and motivated us. May Allah shower all his blessings upon him (Amen).

The summary of the knowledge I gained throughout the fellowship program is listed below:

- **HTML(HyperText Markup Language):**

  In HTML, we have to structure websites using tags. Each tag has its own purpose. Some important and commonly used tags are listed below:

HTML:-
The <html> tag represents the root of an HTML document.
The <html> tag is the container for all other HTML elements (except for the <!DOCTYPE> tag).

HTML <!--...--> Tag
The comment tag is used to insert comments in the source code. Comments are not displayed in the browsers.
HTML <!DOCTYPE> Declaration
All HTML documents must start with a <!DOCTYPE> declaration.
The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.
In HTML 5, the declaration is simple:
<!DOCTYPE html>

HTML <abbr> Tag
The <abbr> tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

HTML <blockquote> Tag

The <blockquote> tag specifies a section that is quoted from another source.Browsers usually indent <blockquote> elements

HTML <body> Tag

The <body> tag defines the document's body.

The <body> element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

Note: There can only be one <body> element in an HTML document.

Headings in HTML:

There are six heading tags available. Each tag has different size. <h1> is the biggest heading size while <h6> is the smallest heading size.

HTML
 Tag:

The
 tag inserts a single line break.

The
 tag is useful for writing addresses or poems.

The
 tag is an empty tag which means that it has no end tag.

HTML <button> Tag

- The <button> tag defines a clickable button.
- Inside a <button> element you can put text (and tags like <i>, <b>, ,
, <img>, etc.). That is not possible with a button created with the <input> element!

HTML <cite> Tag

- The <cite> tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).
- **Note:** A person's name is not the title of a work.

The text in the <cite> element usually renders in *italic*.

Tag:

- The tag is used to define emphasized text. The content inside is typically displayed in *italic*.
- A screen reader will pronounce the words in with an emphasis, using verbal stress.

<form> tag:

The <form> tag is used to create an HTML form for user input.

The <form> element can contain one or more of the following form elements:

- **<input>**
- **<textarea>**
- **<button>**
- **<select>**
- **<option>**
- **<optgroup>**
- **<fieldset>**
- **<label>**
- **<output>**

<hr> Tag:

The <hr> tag defines a thematic break in an HTML page (e.g. a shift of topic).

The <hr> element is most often displayed as a horizontal rule that is used to separate content (or define a change) in an HTML page.

<img> tag:

The <img> tag is used to embed an image in an HTML page.

Images are not technically inserted into a web page; images are linked to web pages.
The <img> tag creates a holding space for the referenced image.

The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed

Note: Also, always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

<input> tag:

The <input> tag specifies an input field where the user can enter data.

The <input> element is the most important form element.

The <input> element can be displayed in several ways, depending on the type attribute.

The different input types are as follows:

- <input type="button">

- <input type="checkbox">
- <input type="color">
- <input type="date">

<li>,<ol>, <ul> Tag:

The **<li>** tag defines a list item.

The **<li>** tag is used inside ordered lists(<ol>), unordered lists (<ul>), and in menu lists (<menu>).

In <ul> and <menu>, the list items will usually be displayed with bullet points.

In <ol>, the list items will usually be displayed with numbers or letters.

SEMANTIC TAGS:-

In HTML there are some semantic elements that can be used to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

CSS (Cascading Style Sheet):-

CSS is used for adding beauty to the structure of website developed by HTML. Adding CSS is like painting the walls of a house.

Main Things in CSS:

Main things in CSS are:

- **Methods for Adding CSS**
- **CSS Selectors**
- **Colors**

- **Padding & Margin**
- **Flexbox**
- **Grid**
- **Alignment**
- **Positioning**

Methods for Adding CSS:

There are three methods to add CSS:

i. **Inline CSS:**
Adding CSS using style attribute in the html tags.
ii. **Internal CSS:**
Adding CSS in HTML file within <style> tag.
iii. **External CSS:**
Adding CSS in a separate file named 'style.css' and linking it with HTML file.

CSS Selectors:

CSS selectors are patterns used to select and target specific HTML elements in a web page. They allow you to apply styles or perform actions on those selected elements.

E.g: element selectors, class , id, child selectors , etc

Colors:

CSS provides various ways to specify colors for elements such as:

- **Keyword Colors**: red, green, blue
- **Hexadecimal Colors: #FF0000**
- **RGB Colors:** RGB stands for Red, Green, Blue. It allows you to define colors by specifying the intensity of these primary colors using decimal values ranging from 0 to 255. For example, **rgb(255, 0, 0)**

Padding & Margin:

Padding is the inner space between the content and the border **while margin** is the space outside the border.

Flexbox:

The Flexible Box Layout Module (**one dimensional system**), makes it easier to design flexible responsive layout structure without using float or positioning.

To start using the Flexbox model, you need to first define a flex container. A flex container must have flex elements. The flex container becomes flexible by setting the display property to flex.

Grid:

Grid layout is a two dimensional system for creating responsive complex layouts adaptable to different screen sizes with rows and columns, allowing you to manage and position rows and columns simultaneously.

Alignment:

CSS Position Property:
The position property specifies the type of positioning method used for an element (static, relative, absolute, fixed, or sticky).

Note: the default value for position is ***Static.***

Value	**Description**
static	Default value. Elements render in order, as they appear in the document flow.
absolute	The element is positioned relative to its first positioned (not static) ancestor element.
fixed	The element is positioned relative to the browser window.
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position.
sticky	The element is positioned based on the user's scroll position A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

- **BOOTSTRAP:**

Bootstrap is **a free, open source front-end development framework(**Bootstrap is the most popular **CSS** Framework**) for the creation of websites and web apps**. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

Why to use?

- Bootstrap **helps build websites and web pages quickly and with a better UI**
- it saves time from writing a lot of CSS code and gives developers more time to design web pages.
- "Responsiveness", "UI components" and "Consistent" are the key factors why developers consider Bootstrap.

- Bootstrap is based on the CSS grid system, the CSS flexbox, to improve responsiveness.

COLORS IN BOOTSTRAP:-

In bootstrap, we can have following colors:

- Primary---blue
- Warning---yellow
- Danger---red
- Light---white
- Dark---black
- Success—green
- Info---blueish green

JAVASCRIPT:-

JavaScript is a programming language unlike HTML and CSS and is used for adding functionality to the website. Its example is like installing taps in a house and enabling them to give water by wiring the water pipes.

Following are some key concepts of JavaScript:

1. **Variables:** Variables are used to store and manipulate data in JavaScript. They hold values of different types such as numbers, strings, booleans, objects, or functions.
2. **Control Flow:** Control flow refers to the order in which statements are executed in a program. JavaScript provides control flow structures like if statements, loops (such as for and while loops), and switch statements to control the execution flow based on conditions.
3. **Functions:** Functions are blocks of reusable code that perform specific tasks. They can be declared using the function keyword or defined as function expressions. Functions can accept parameters and return values.
4. **Scope and Closures:** Scope determines the accessibility and visibility of variables in different parts of a program. JavaScript has function scope and block scope. Closures refer to the ability of inner functions to access variables from their outer functions, even after the outer functions have finished executing.
5. **DOM Manipulation:** The Document Object Model (DOM) represents the HTML structure of a webpage. JavaScript allows developers to interact with and manipulate the DOM, selecting and modifying elements, handling events, and dynamically creating or removing elements.

Tailwind CSS:

Tailwind CSS is a highly popular utility-first CSS framework that provides a set of pre-built utility classes to rapidly build user interfaces. It is a CSS framework that offers extensive

customization options and provides a wide range of low-level building blocks. With this framework, you have the flexibility to create unique designs without being constrained by pre-defined and difficult-to-override styles.

WHY TAILWIND CSS:

- Highly Customizable
- Responsive Designs
- Minimum lines of code in CSS
- No more silly names for CSS classes and Id
- Can be used with any JavaScript

React JS:

React is an open-source JavaScript library developed by Facebook for building user interfaces (UIs). It has gained widespread popularity due to its efficiency, reusability, and declarative approach to building UI components.

SOME PROPERTIES OF REACT:

1. **Component-Based Architecture:** React follows a component-based architecture, where the user interface is broken down into reusable components. Each component encapsulates its own logic, state, and rendering, making it easy to manage and update.
2. **Virtual DOM:** React uses a virtual DOM (Document Object Model) to efficiently update and render components. The virtual DOM is a lightweight representation of the actual DOM, allowing React to perform efficient diffing and update only the necessary parts of the UI, leading to improved performance.
3. **Declarative Syntax**: React uses a declarative syntax, allowing developers to describe how the UI should look based on the state of the application. Instead of manually manipulating the DOM, developers define the desired UI state, and React takes care of updating the actual DOM accordingly.
4. **JSX:** React introduces JSX (JavaScript XML), which is an extension to JavaScript that allows developers to write HTML-like syntax within JavaScript code. This makes it easier to define the structure and appearance of components directly within the JavaScript code.

WHAT IS A COMPONENT?

A component is a reusable building block that encapsulates logic and rendering to create a specific part of the user interface (UI). It represents a self-contained piece of the UI, which can be composed together to create complex user interfaces. Components can be class or functional components.

Render ( ):

The **render()** method is a fundamental part of both functional and class components. It is responsible for defining what the component should render on the screen and is not explicitly defined because the component itself is the render function.

Git Commands:

- git init
- git add .
- git status
- git commit -m "repo name"
- git add remote origin "https://address of your github repo"
- git push -u origin master

Sources for Hosting:

- GitHub live pages
- Netlify

Sources for Learning:

- NetNinja
- JavaScript Mastery
- Programming with Mosh
- Traversy Media
- codewithsadsee

The above information is a summary of knowledge, I got throughout the fellowship. I hope you will like it. Thanks for your time, dedication and providing me this opportunity.