Assignment 10 - Joh Dikhta Hai Wahi Bikta Hai

Q1 Explore all the ways of writting CSS?

⇒ CSS can be added to HTML document in 3 ways:

1) Inline Css
An inline CSS is used to apply a unique style to a single HTML element. An inline CSS uses the style attribute of an HTML element

Eg
```
<h1 style="color:blue;"> A Blue Heading </h1>
```

2) Internal Css
An internal CSS is used to define a style for a single HTML page. An internal CSS is defined in the <head> section of an HTML page, within a <style> element

Eg
```
<html>
    <head>
        <style>
            body {background-color: blue;}
        </style>
    </head>
    <body>
        <h1> This is a heading </h1>
    </body>
</html>
```

3) External Css
An external stylesheet is used to define the style for many HTML pages

Code
```
<head>
    <link rel="stylesheet" href="style.css"/>
</head>
```

The external stylesheet can be written in any text edition. This file must not contain any HTML code, and must be saved with a .css extension.

Q2 How do we configure tailwind CSS?

⇒ Step1: Create a new project
Step2: Install Tailwind CSS
You can install Tailwind CSS using npm or yarn. Open your terminal or command prompt and navigate to your project's root directory.

Run one of the following command :

Using npm:

⇒ npm install tailwindcss

Step 3: Create a Configuration File

You can generate a basic configuration file using the following command:

⇒ npm tailwind init

This command creates a tailwind.config.js file in your project's root directory

Step 4: Customize the Configuration (Optional)

Open the generated tailwind.config.js file, and you customize various aspects of Tailwind CSS according to your project's needs. This file includes options for colors, fonts, spacing, breakpoints, and more.

```
//Code
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#3490dc',
        secondary: '#ffed4a',
      },
    },
  },
};
```

Step 5: Create CSS File

Create a CSS file where you will import Tailwind CSS and any additional styles. Typically, this file is named styles.css or similar. Import Tailwind CSS using the @import directive

```
/* styles.css */
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';
```

Step 6: Build Your Styles

Include your CSS file in your HTML or import it in your JavaScript file if you are using a bundler like Webpack

Step 7: Use Tailwind CSS Classes in HTML

You can start using Tailwind CSS classes in your HTML files to apply styles.

```
<div class="bg-primary text-white p-4">
    This is a primary-colored box with white text and padding.
</div>
```

Step 8: Build Your Project

Depending on your setup, you might need to build your project to apply the Tailwind CSS styles. If you're using a bundler like Webpack, make sure to run the appropriate build command.

With npm:

npm run build or npm start

Q3 In tailwind.config.js, what does all the keys mean (content, theme, extend, plugins)?

⇒ In tailwind.config.js, the various keys serve different purposes and allow you to customize and configure different aspects of Tailwind CSS.

1) content Key:

It specifies the files that Tailwind CSS should analyze to generate its utility classes.

Usage:

```
// Code
module.exports = {
  content: [
    './src/**/*.html',
    './src/**/*.js',
  ],
}
```

The content key helps Tailwind CSS identify which files to process and extract utility classes from. It is particularly useful when working with frameworks like React or Vue

2) theme Key:

It defines the default styles and configurations for various aspects of Tailwind CSS, such as colors, spacing, fonts, and more.

Usage:

```
// Code
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: '#3490dc',
        secondary: '#ffed4a',
      },
    },
  },
};
```

The theme key allows you to customize default styles and extend or override the default configuration provided by Tailwind CSS. It is where you can define your project-specific design system.

3) extend Key:
It extends or overrides the default configuration provided by Tailwind CSS. The extend key is often used to add new styles or extend existing ones. It is especially useful for adding project-specific utility classes or modifying existing ones.

4) plugins Key:
It allows you to use or define custom plugins to extend or modify Tailwind CSS functionality.
Usage:

```
// Code
module.exports = {
  plugins: [
    require('@tailwindcss/forms'),
  ],
};
```

The plugins key lets you incorporate third-party plugins or create your own custom plugins. Plugins can add new features, styles, or utilities to Tailwind CSS.

Q4 Why do we have .postcssrc file?

⇒ The .postcssrc file, often named postcss.config.js, is a configuration file for PostCSS. PostCSS is a tool for transforming styles with JavaScript plugins, and it is commonly used in conjunction with build tools like webpack or parcel for processing and optimizing CSS.

Uses of .postcssrc

1) To configure the plugins that PostCSS use. These plugins can handle tasks such as autoprefixing, minification, and syntax enhancements

2) To behavior of PostCSS beyond the default settings provided by the build tool (eg. webpack). This allows you to have fine-grained control over the PostCSS transformations.

3) It is a convenient place to define these options and presents.

4) Separating the PostCSS configuration into its own file makes the build configuration more maintainable and organized. It allows you to centralize PostCSS-related settings and keep them distinct from other build tool configurations