* **What's in a web page?**
* The content, style, and interactive logic are separated into HTML, CSS, and JavaScript files, respectively. A design principle in modern programming is *separation of concerns*. Among the many reasons to separate concerns, two are simplicity and reuse. For example, by styling HTML elements using CSS, you can simplify your HTML code. Instead of coding the appearance within each element, you apply CSS styles to all the elements on a page, regardless of the page complexity. In addition, you can link multiple HTML pages to a single CSS file, which can help simplify a consistent look and feel across your entire website.

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Simple Website</title>

    <link rel="stylesheet" href="main.css">

</head>

<body class="dark-theme">

    <h1>Task List</h1>

    <p id="msg">Current Tasks:</p>

    <ul>

        <li class="list">Add visual styles</li>

        <li class="list">Add light and dark themes</li>

        <li>Enable switching the themes</li>

    </ul>

    <div>

        <button class="btn">Dark</button>

    </div>

    <script src="app.js"></script>

    <noscript>You need to Enable Javascript to view the full site.</noscript>

</body>

</html>

* The doctype <!DOCTYPE html> indicates this HTML document contains HTML5 code.
* The meta tag indicates metadata information that won't typically be visible to the viewer unless they view the source code in their browser. Meta elements or tags provide descriptive information about the webpage. For example, they help search engines process which information in your webpages to return in search results.
* The character set (charset) for UTF-8 may seem insignificant, but is crucial for establishing how computers interpret characters. If the metadata for the character set is missing, that can lead to compromised security. There's quite a bit of history and technical information behind the charset attribute, but important takeaway from this exercise is that the **VS Code** boilerplate provides some sensible defaults.
* The <head> element in your HTML code contains information about your website not visible inside the browser tab.
* The metadata defines data about the HTML document, such as character set, scripts, and which browser the webpage opens in.
* The title of a webpage appears at the top of a browser window, and is significant in many ways. For example, the title is used by and displayed in search engines.
* The <body> element contains the content of your website visible to your customers in their browsers.
* An ID attribute (used in the <p> element) can be used for styling one element, while the class attribute (used in the <li> element) is for styling all elements of the same class.
* The **Elements** tab in developer tools shows you the document object model (DOM) as rendered in the browser. When debugging, it's often important to see how the browser interprets your source code.

Main.css

body{

    background: var(--bg);

    color: var(--fontColor);

    font-family: helvetica;

}

li{

    list-style-type: circle;

}

.list{

    list-style-type: square;

}

.light-theme{

    --bg: var(--green);

    --fontColor: var(--black);

    --btnBg: var(--black);

    --btnFontColor: var(--white);

}

.dark-theme{

    --bg: var(--black);

    --fontColor: var(--green);

    --btnBg: var(--white);

    --btnFontColor: var(--black);

}

:root{

    --green: #00FF00;

    --white: #FFFFFF;

    --black: #000000;

}

.btn{

    position: absolute;

    top: 20px;

    left: 250px;

    height: 50px;

    width: 50px;

    border-radius: 50%;

    border: none;

    color: var(--btnFontColor);

    background-color: var(--btnBg);

}

* To apply styles to the HTML elements on the webpage, you could write the CSS code directly in the head of the webpage. Writing CSS in the HTML page is called internal CSS. However, it's a best practice to separate HTML structure and CSS styling. Having a separate CSS page is called external CSS. Code tends to be easier to read when it's concise and compartmentalized. You can use one or more external style sheets to service multiple webpages. Rather than updating each HTML page with replicated CSS code, you can make changes to a single CSS file, and have those updates applied to all of the dependent web pages. Let's link to an external stylesheet.
* Cascading Style Sheets (CSS) let you specify how your page should look. The basic idea is to define what the style should be for the elements that you use within your HTML pages. While the HTML elements define your content, CSS styles define what this content looks like.

For example, you can apply rounded corners or give a gradient background to an element. Or you can use CSS to specify how hyperlinks look and respond when you interact with them. You can also perform sophisticated page layouts and animation effects.

You can apply styles to specific elements, all elements of a specific type, or use classes to style many different elements.

* One benefit of external CSS is that multiple HTML pages can link to the same CSS file. If you make a change to the CSS, your styling will update for each page. Using an HTML file for your page content, a CSS file for styling, and a JavaScript file for interaction is called separation of concerns.
* As described previously, you can also write CSS directly in HTML, which is called internal CSS. Even for a basic website, there are so many CSS rules the HTML page can become cluttered quickly. With more than one page, the same CSS would often be repeated and challenging to manage.
* CSS rules are how you apply styles to HTML elements. CSS rules have a **selector** which is used to express which element, or elements, should the styles be applied to.
* A *selector*. body and ul are the selectors of the two rules and are used to select which element(s) the styles apply to.
* An opening curly brace ({).
* A list of style *declarations* that determine what the selected elements should look like.
* A closing curly brace (}).

For example, the ul selector selects the <ul> HTML element in the page, to apply styles to it. The declaration is font-family: helvetica and determines what the style should be. The *property name* is font-family, and the *value* is helvetica.

* ID and class selectors enable you to apply styles to custom attribute names in your HTML. An ID is used to style one element, whereas classes can be used to style multiple elements.

The preceding code contains three CSS rules, with the last two rules using custom attributes to select elements: .list and #msg.

* .list is a class selector. Each HTML element that contains a class attribute set to list will get the styles that are defined within this selector.
* #msg is an ID selector. The HTML element that has its id attribute set to msg will get the styles that are defined within this selector.

The names that you use for your selectors can be arbitrary, as long as they match what you've defined in the HTML.

App.js

'use strict';

const switcher=document.querySelector('.btn');

switcher.addEventListener('click',function(){

    document.body.classList.toggle('light-theme');

    document.body.classList.toggle('dark-theme');

    const className=document.body.className;

    if(className == 'light-theme'){

        this.textContent="Dark";

    }

    else{

        this.textContent="Light";

    }

    console.log('current class Name: '+className);

});

* n your JavaScript file, use document.querySelector to get the button reference.
* Next, add the event handler for the click event. In the following code, you add a listener for the click event and define an event handler function to be executed by the browser when the click event occurs.
* In the preceding code, you used the toggle method to modify the <body> element's class attribute. This method automatically adds or removes the light-theme and dark-theme classes. This code applies the dark styles instead of light styles on click, and then light styles instead of dark if you click again.

However, the label for the button also needs to be updated to show the correct theme, so you need to add an if statement to determine the current theme, and update the button label.

Output:-



