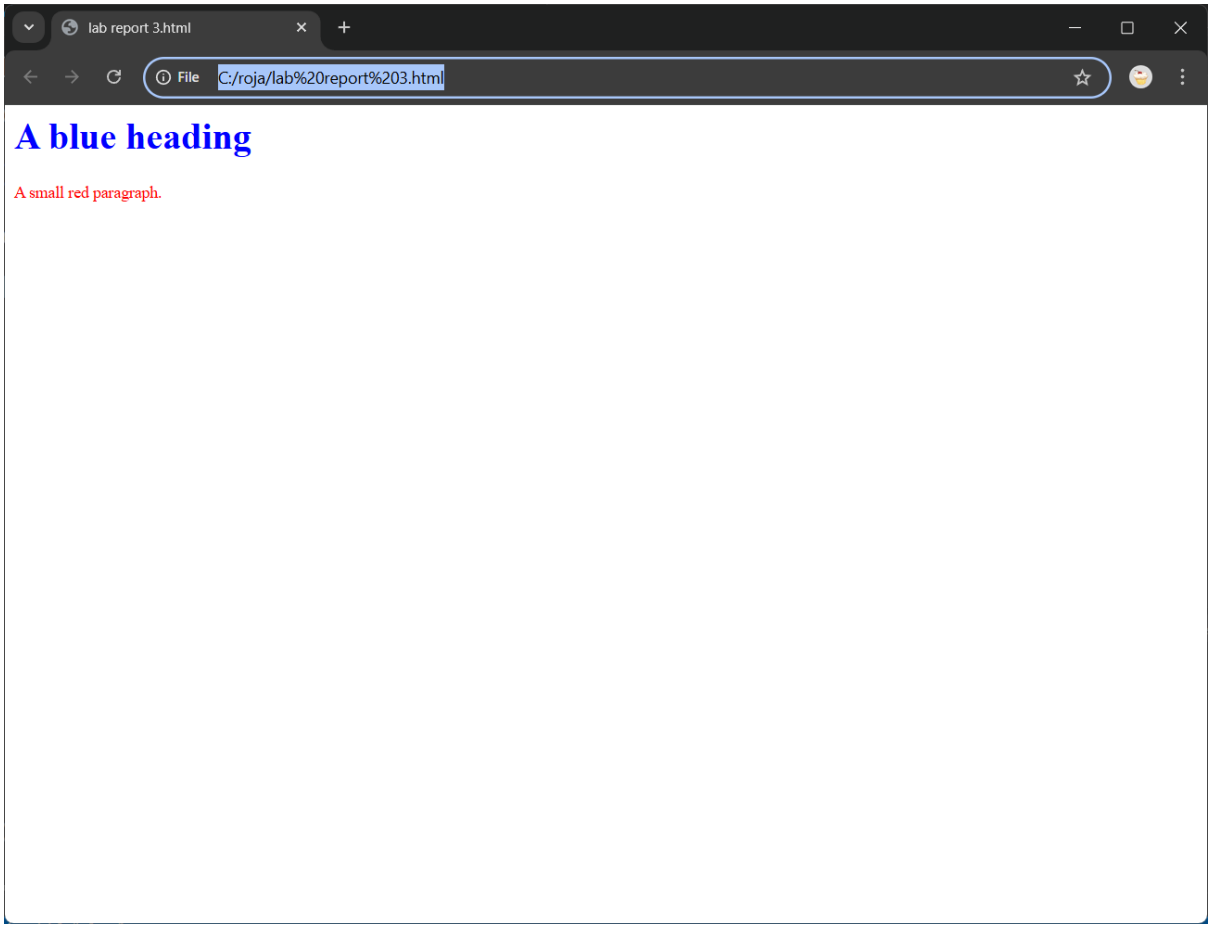
STUDENT NAME	**R.ROJA**	
STUDENT REGISTRATION NUMBER	**251U1R2064**	**CLASS: CSE AIML (C)**
PROGRAM	**UG**	**YEAR and TERM:** **1st year & 1st term**
SUBJECT NAME	**HTML**	
NAME OF THE ASSESSMENT	**Reflective lab journal-3**	
DATE OF SUBMISSION	**23.10.25**	

1.Inline style sheet:

```
<html>
<body>

    <h1 style="color: blue;">A blue heading</h1>

    <p style="color: red; font-size: 14px;">A small red paragraph.</p>

</body>
</html>
```

Output:



Explanation:

The style attribute is added directly to the <h1> tag to make only this heading blue.

Another style attribute is used on the <p> tag to make only this specific paragraph red and adjust its font size.

An inline stylesheet applies CSS rules directly to a single HTML element using the style attribute.

It is the most specific way to apply a style, meaning it will override rules defined in internal or external stylesheets, except for the !important declaration.

The syntax involves adding the style attribute to an element, followed by a series of semicolon-separated property-value pairs within double quotes.

The main advantage of inline CSS is the ability to make quick, one-off style adjustments to an element without needing to modify a separate stylesheet.

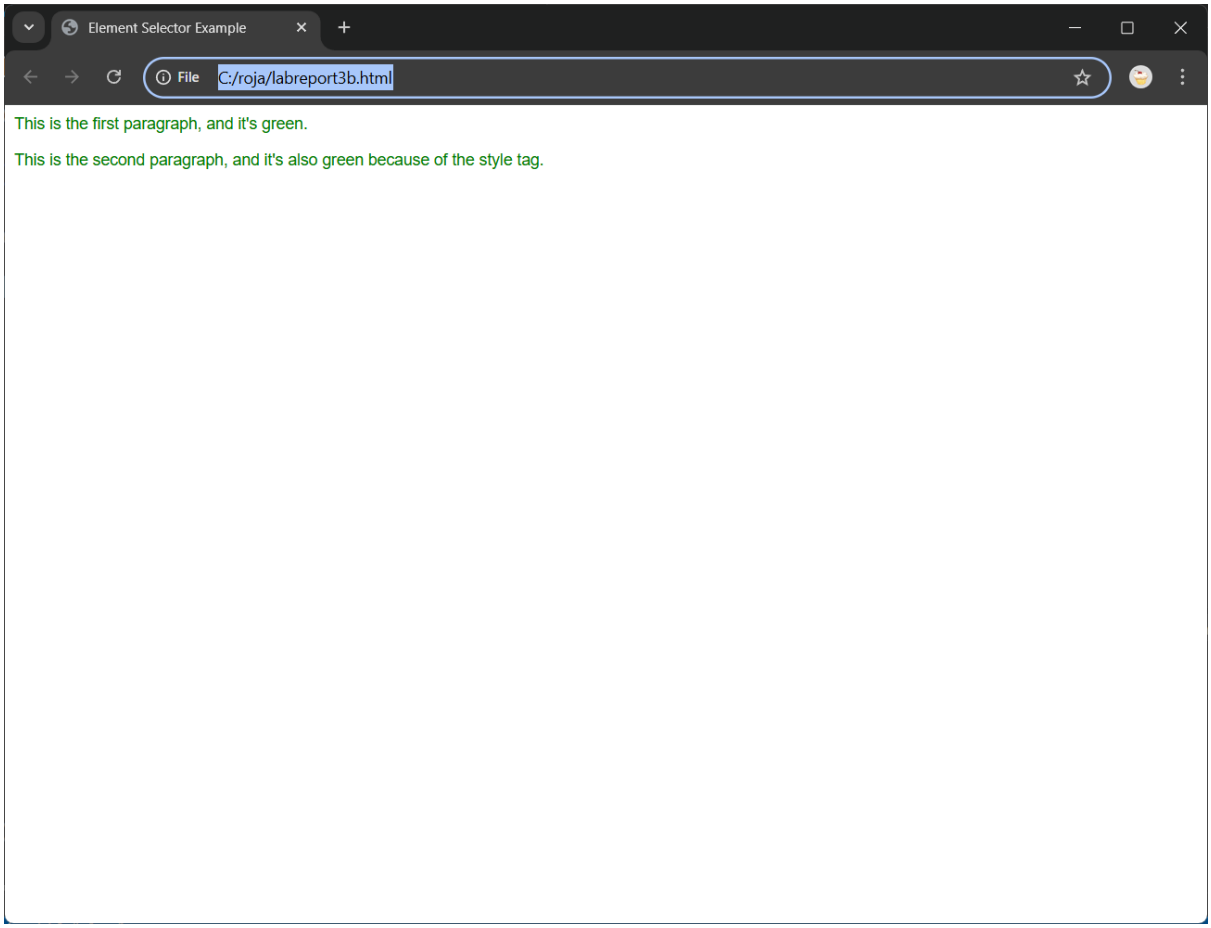
This can be useful for testing or for creating very simple pages or email templates, where broader styling methods may not be supported.

It is also valuable when a JavaScript script needs to dynamically change the styling of an element based on user interaction.

2.Internal style sheet.

```
html>
<head>
    <title>Element Selector Example</title>
    <style>
        p {
            color: green;
            font-family: Arial, sans-serif;
            font-size: 14px;
        }
    </style>
</head>
<body>
    <p>This is the first paragraph, and it's green.</p>
    <p>This is the second paragraph, and it's also green because of the style
tag.</p>
</body>
</html>
```

Output:



Explanation:

The program above uses an internal stylesheet to apply consistent styling across a single HTML document.

An internal stylesheet is defined by placing <style> tags within the <head> section of the HTML file.

This approach keeps all the presentation rules contained within one document, making it suitable for single-page applications or when a page requires unique styling that is not shared with other pages on a website.

Inside the <style> tag, CSS rules are defined using selectors to target specific HTML elements.

For instance, the body selector styles the entire web page by setting a font-family, background color, and default text color, which all elements within the body will inherit.

The h1 selector specifically targets all <h1> headings, centering the text and adding a stylized border at the bottom.

By using CSS selectors, including element selectors (p, ul, li), class selectors (.container, .highlight), and pseudo-classes (:hover), we can apply styles efficiently and consistently to multiple elements at once.

The container class is applied to a to create a content box with a white background, padding, and a subtle shadow, demonstrating how to group styles for reusable components.

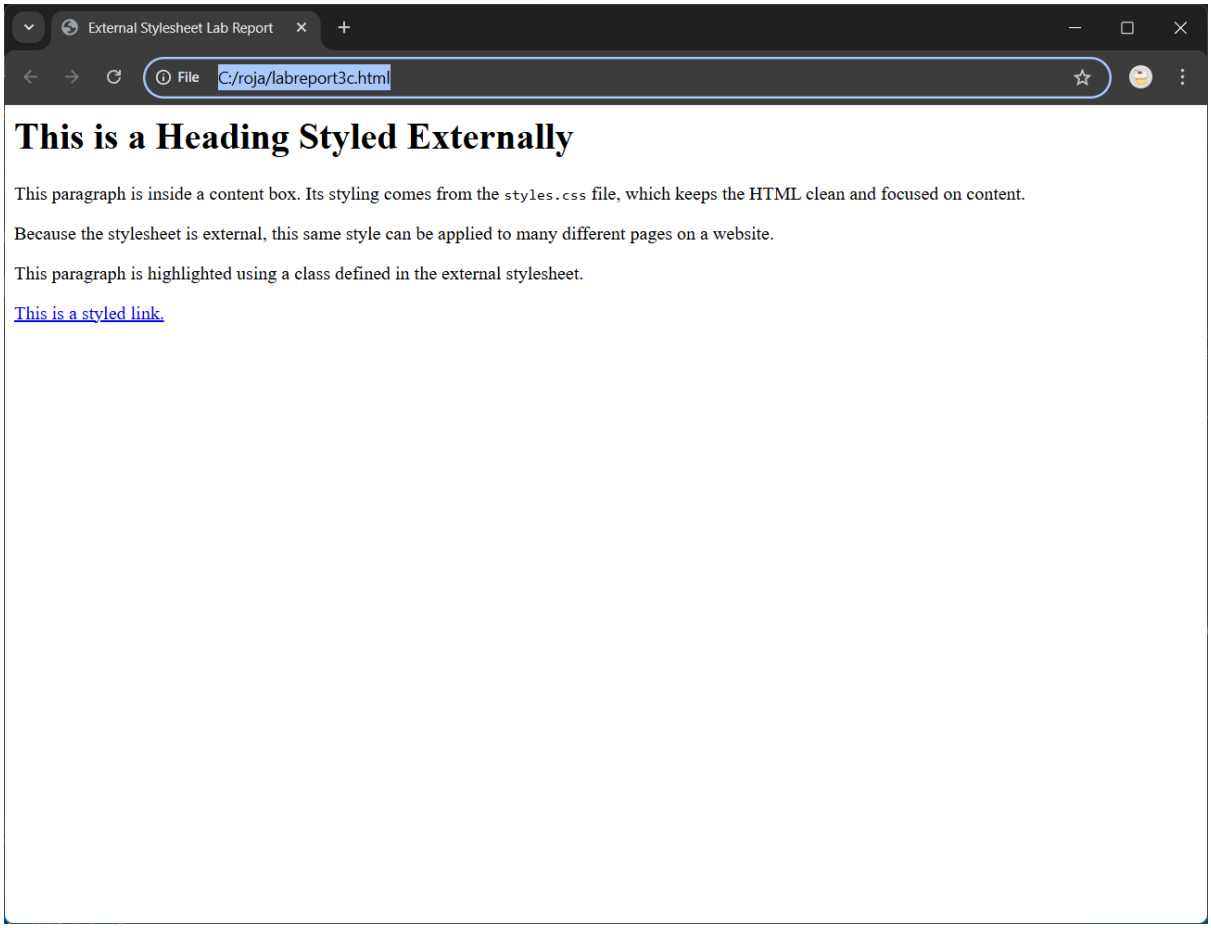
The li:hover selector adds a dynamic effect, causing a list item to shift slightly when the mouse hovers over it.

This shows the versatility of internal stylesheets for interactive styling.

3.External style sheet:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>External Stylesheet Lab Report</title>

    <!-- This links the external stylesheet -->
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <h1>This is a Heading Styled Externally</h1>

    <div class="content-box">
        <p>This paragraph is inside a content box. Its styling comes from
the <code>styles.css</code> file, which keeps the HTML clean and focused on
content.</p>
        <p>Because the stylesheet is external, this same style can be
applied to many different pages on a website.</p>
    </div>

    <p class="highlight">This paragraph is highlighted using a class
defined in the external stylesheet.</p>
    <a href="#">This is a styled link.</a>
</body>
</html>
```

Output:



Explanation:

The program above demonstrates the most efficient and scalable method for styling a website: using an external stylesheet.

This approach involves separating the HTML content into one file (index.html) and all the CSS rules into another, separate file (styles.css).

The HTML file then references the CSS file using the <link> tag within the <head> section.

The rel="stylesheet" attribute specifies the relationship, and href="styles.css" provides the path to the stylesheet file.

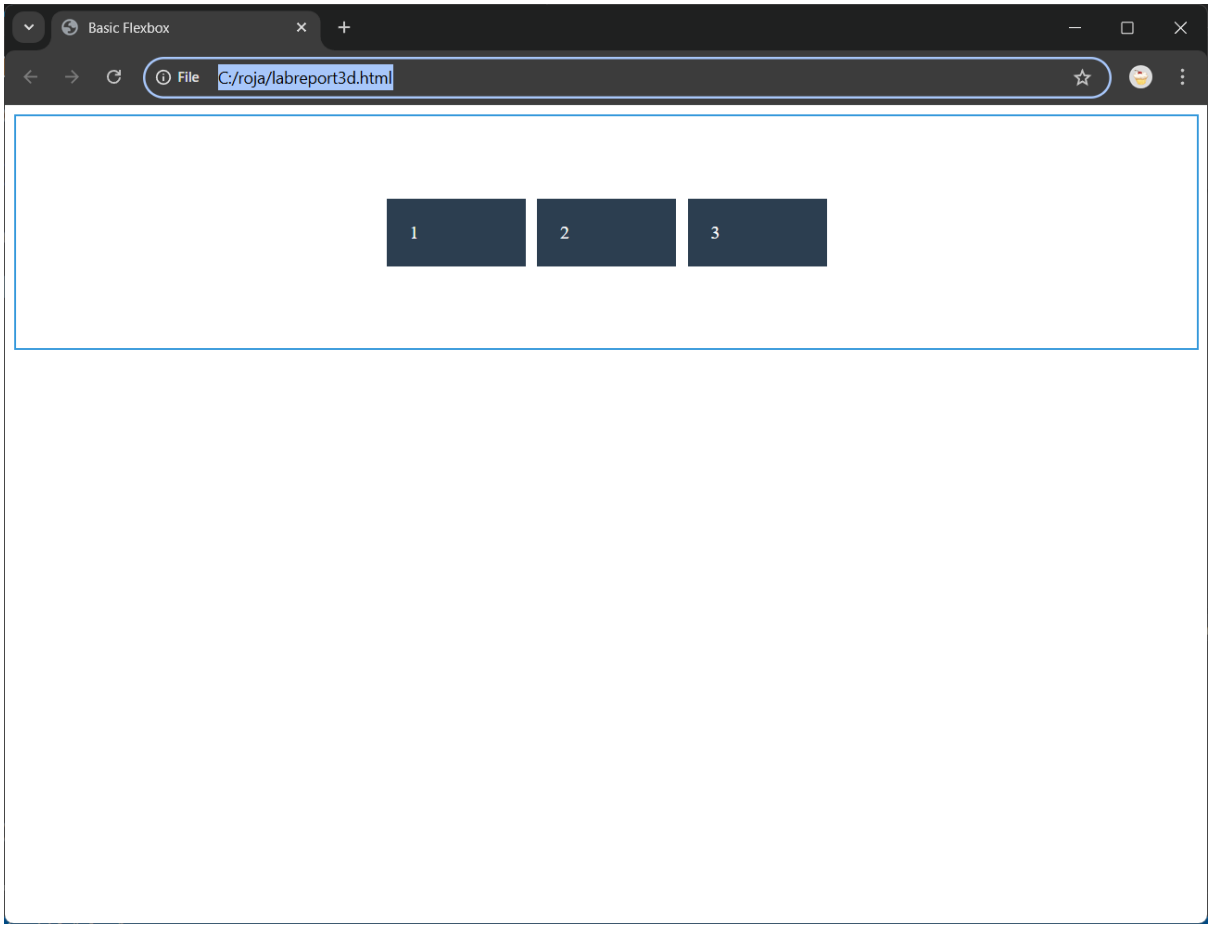
This complete separation of structure (HTML) from presentation (CSS) is considered a best practice in modern web development.

It ensures that the HTML code remains clean, readable, and focused on its content and semantic meaning, rather than being cluttered with styling information.

The true power of an external stylesheet becomes apparent when developing a website with multiple pages.

4.Flexbox and its properties:

```
<html>
<head>
    <title>Basic Flexbox</title>
    <style>
        .flex-container {
            display: flex;
            justify-content: center; /* Horizontally center items */
            align-items: center; /* Vertically center items */
            height: 200px;
            border: 2px solid #3498db;
        }
        .flex-item {
            width: 80px;
            padding: 20px;
            background-color: #2c3e50;
            color: white;
            margin: 5px;
        }
    </style>
</head>
<body>
<div class="flex-container">
        <div class="flex-item">1</div>
        <div class="flex-item">2</div>
        <div class="flex-item">3</div>
    </div>
</body>
</html>
```



This program uses CSS Flexbox, a one-dimensional layout system, to arrange a series of div elements within a container.

The core of the flexbox functionality begins by applying display: flex to the .flex-container, turning its direct children into flex items.

The layout is governed by properties applied to both the container and the items themselves.

The container property flex-direction: row arranges the items horizontally, from left to right.

When the viewport is resized and space becomes limited, the flex-wrap: wrap property ensures that items will wrap onto a new line instead of overflowing or shrinking excessively.

The justify-content: space-between property distributes the flex items evenly along the main axis (the row), placing the first item at the start, the last at the end, and the remaining space in between.

On the cross-axis, align-items: center aligns all flex items to the vertical center of the container, regardless of their individual sizes.

This demonstrates the powerful vertical alignment capabilities of flexbox.

Individual flex items can also have their own properties.

In this program, the second flex item is given flex-grow: 1, allowing it to expand and fill any available extra space, while the other items retain their fixed base width.