A
Lab Records
of
Interactive Web Interface

School of Computer Science Engineering- 1st Sem

RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Submitted to:-
Prof. Ishita Gupta

Submitted by:-
Pulkit aryan singh
ERP ID: 11058

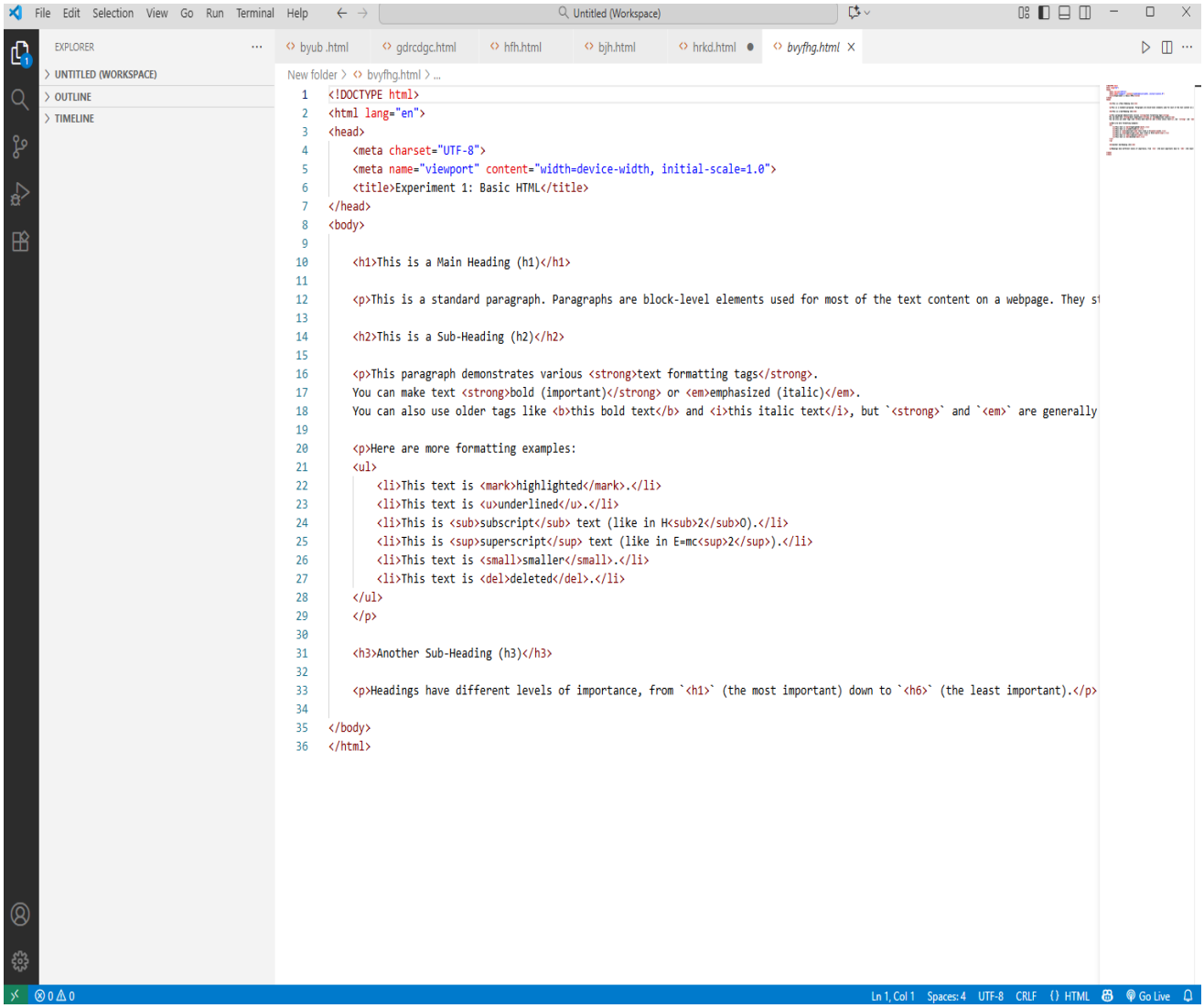
RUNGTA INTERNATIONAL SKILLS UNIVERSITY,CG
SCHOOL OF COMPUTER SCIENCE ENGINEERING

S.No	Name of Practicals	Submission date	Remarks
1.	Create a basic HTML web page using headings, paragraphs, and text formatting tags		
2.	Design a webpage demonstrating HTML links, images, and lists (ordered, unordered, nested)		
3.	Create a table-based layout with merged cells, alignment, and caption using HTML		
4.	Design a webpage with an HTML form having text input, radio buttons, checkboxes, select menus, and submit/reset buttons		
5.	Use semantic elements (<article>, <section>, <nav>, <aside>, <footer>) to build a structured web page		
6.	Apply Inline, Internal, and External CSS styles to HTML elements		
7.	Style a web page using advanced CSS: Box Model, background images, borders, margins, and padding		
8.	Demonstrate the use of different CSS selectors (class, id, descendant, group, universal)		
9.	Create a responsive web page using media queries and mobile-first design with flexible units		
10.	Use div and span elements for layout and apply styling with CSS		

Experiment – 1

AIM - 1: Create a basic HTML web page using headings, paragraphs, and text formatting tags.

Code:



Output:

This is a Main Heading (h1)

This is a standard paragraph. Paragraphs are block-level elements used for most of the text content on a webpage. They start on a new line.

This is a Sub-Heading (h2)

This paragraph demonstrates various **text formatting tags**. You can make text **bold (important)** or *emphasized (italic)*. You can also use older tags like **this bold text** and *this italic text*, but `` **and** `` ***are generally preferred because they add semantic meaning.***

Here are more formatting examples:

- ***This text is highlighted.***
- ***This text is underlined.***
- ***This is $_{subscript}$ text (like in H_2O).***
- ***This is ^{superscript} text (like in $E=mc^2$).***
- ***This text is smaller.***
- ***This text is ~~deleted~~.***

Another Sub-Heading (h3)

Headings have different levels of importance, from `

` (the most important) down to `

` (the least important).

Experiment – 2

AIM – 2 : Design a webpage demonstrating HTML links, images, and lists (ordered, unordered, nested)

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 2: Links, Images, Lists</title>
</head>
<body>

    <h1>HTML Links</h1>
    <p>This is an <a href="https://www.google.com" target="_blank">external link</a> to Google (opens in a new tab).</p>

    <p>This is an <a href="about.html">internal link</a> to an About page.</p>

    <h1>HTML Image</h1>

    <img src="https://via.placeholder.com/300x150.png?text=Sample+Image" alt="A sample placeholder image">
    <p>The 'alt' attribute provides descriptive text for screen readers or if the image fails to load.</p>

    <h1>HTML Lists</h1>

    <h3>Unordered List (Bulleted)</h3>
    <ul>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
    </ul>

    <h3>Ordered List (Numbered)</h3>
    <ol>
        <li>First, wake up.</li>
        <li>Second, make coffee.</li>
        <li>Third, code.</li>
    </ol>

    <h3>Nested List</h3>
    <ul>
        <li>Front-End
            <ol>
                <li>HTML</li>
                <li>CSS</li>
                <li>JavaScript</li>
            </ol>
        </li>
        <li>Back-End
            <ol>
                <li>Python</li>
                <li>Node.js</li>
                <li>PHP</li>
            </ol>
        </li>
    </ul>

</body>
</html>
```

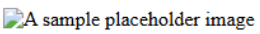
Output:

HTML Links

This is an external link to Google (opens in a new tab).

This is an internal link to an About page.

HTML Image



The 'alt' attribute provides descriptive text for screen readers or if the image fails to load.

HTML Lists

Unordered List (Bulleted)

- Coffee
- Tea
- Milk

Ordered List (Numbered)

1. First, wake up.
2. Second, make coffee.
3. Third, code.

Nested List

- Front-End
    1. HTML
    2. CSS
    3. JavaScript
- Back-End
    1. Python
    2. Node.js
    3. PHP

Experiment – 3

AIM – 3 : Create a table-based layout with merged cells, alignment, and caption using HTML

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 3: HTML Table</title>
    <style>
        table, th, td {
            border: 1px solid black;
            border-collapse: collapse;
            padding: 8px;
        }
        caption {
            font-weight: bold;
            font-size: 1.2em;
            margin: 8px;
        }
    </style>
</head>
<body>

    <h1>Student Time Table</h1>

    <table>
        <caption>Weekly Class Schedule</caption>
        <thead>
            <tr>
                <th>Day</th>
                <th>9:00 AM</th>
                <th>10:00 AM</th>
                <th>11:00 AM</th>
                <th>12:00 PM</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Monday</td>
                <td>Math</td>
                <td>Science</td>
                <td>English</td>
                <td colspan="2" align="center">Project Work</td>
            </tr>
            <tr>
                <td>Tuesday</td>
                <td rowspanS="2" align="center">Physics Lab</td>
                <td>History</td>
                <td>Math</td>
                <td>PE</td>
            </tr>
            <tr>
                <td>Wednesday</td>
                <td>Chemistry</td>
                <td>Geography</td>
                <td>English</td>
            </tr>
            <tr>
                <td>Thursday</td>
                <td>Math</td>
                <td>Science</td>
                <td colspan="2" align="center">Library</td>
            </tr>
            <tr>
                <td>Friday</td>
                <td colspan="4" align="center">Assembly & Half Day</td>
            </tr>
        </tbody>
    </table>

</body>
</html>
```

Output:

Student Time Table

Weekly Class Schedule

Day	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Monday	Math	Science	English	Project Work
Tuesday	Physics Lab	History	Math	PE
Wednesday	Chemistry	Geography	English	
Thursday	Math	Science	Library	
Friday	Assembly & Half Day			

Experiment – 4

AIM – 4 : Design a webpage with an HTML form having text input, radio buttons, checkboxes, select menus, and submit/reset buttons

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 4: HTML Form</title>
    <style>
        body { font-family: sans-serif; }
        div { margin-bottom: 15px; }
        label { display: block; margin-bottom: 5px; font-weight: bold; }
        input[type="text"] { width: 300px; padding: 5px; }
    </style>
</head>
<body>

    <h1>Event Registration Form</h1>

    <form action="#" method="POST">

        <div>
            <label for="name">Full Name:</label>
            <input type="text" id="name" name="user_name" required>
        </div>

        <div>
            <label>Gender:</label>
            <input type="radio" id="male" name="gender" value="male">
            <label for="male" style="display:inline;">Male</label><br>
            <input type="radio" id="female" name="gender" value="female">
            <label for="female" style="display:inline;">Female</label><br>
            <input type="radio" id="other" name="gender" value="other">
            <label for="other" style="display:inline;">Other</label>
        </div>

        <div>
            <label>Interests:</label>
            <input type="checkbox" id="interest1" name="interests" value="music">
            <label for="interest1" style="display:inline;">Music</label><br>
            <input type="checkbox" id="interest2" name="interests" value="sports">
            <label for="interest2" style="display:inline;">Sports</label><br>
            <input type="checkbox" id="interest3" name="interests" value="tech">
            <label for="interest3" style="display:inline;">Technology</label>
        </div>

        <div>
            <label for="session">Preferred Session:</label>
            <select id="session" name="session_choice">
                <option value="">--Please choose an option--</option>
                <option value="morning">Morning (9 AM - 12 PM)</option>
                <option value="afternoon">Afternoon (1 PM - 4 PM)</option>
                <option value="evening">Evening (6 PM - 9 PM)</option>
            </select>
        </div>

        <div>
            <button type="submit">Submit Registration</button>
            <button type="reset">Clear Form</button>
        </div>

    </form>

</body>
</html>
```

Output:

Event Registration Form

Full Name:

Gender:

- ○ **Male**
- ○ **Female**
- ○ **Other**

Interests:

- ☐ **Music**
- ☐ **Sports**
- ☐ **Technology**

Preferred Session:

--Please choose an option--

Submit Registration Clear Form

Experiment – 5

AIM – 5 : Use semantic elements (<article>, <section>, <nav>, <aside>, <footer>) to build a structured web page

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 5: Semantic HTML</title>
    <style>
        body { font-family: sans-serif; background: #f4f4f4; }
        .container { max-width: 960px; margin: auto; background: #fff; }
        header { background: #333; color: #fff; padding: 20px; text-align: center; }
        nav { background: #555; padding: 10px; }
        nav a { color: #fff; padding: 10px; text-decoration: none; }
        .main-content { display: flex; }
        article { flex: 3; padding: 20px; }
        aside { flex: 1; background: #f9f9f9; padding: 20px; }
        footer { background: #333; color: #fff; text-align: center; padding: 20px; margin-top: 20px; }
    </style>
</head>
<body>

    <div class="container">

        <header>
            <h1>My Blog</h1>
        </header>

        <nav>
            <a href="#">Home</a>
            <a href="#">About</a>
            <a href="#">Contact</a>
        </nav>

        <div class="main-content">
            <section>
                <article>
                    <h2>What is Semantic HTML?</h2>
                    <p>Posted on October 30, 2025</p>
                    <p>Semantic HTML elements are those that clearly describe their meaning in a human- and machine-readable way. Elements like &lt;header&gt;, &lt;footer&gt;, &lt;article&gt;, and &lt;section&gt; are all semantic because they accurately describe the purpose of the element and the type of content that is inside them.</p>
                </article>

                <article>
                    <h2>Why Use It?</h2>
                    <p>It's great for accessibility (screen readers) and SEO (search engines)!</p>
                </article>
            </section>

            <aside>
                <h3>Related Links</h3>
                <ul>
                    <li><a href="#">W3C Specification</a></li>
                    <li><a href="#">HTML5 Tutorial</a></li>
                </ul>
            </aside>
        </div>

        <footer>
            <p>Copyright &copy; 2025 My Blog</p>
        </footer>

    </div>

</body>
</html>
```

Output:

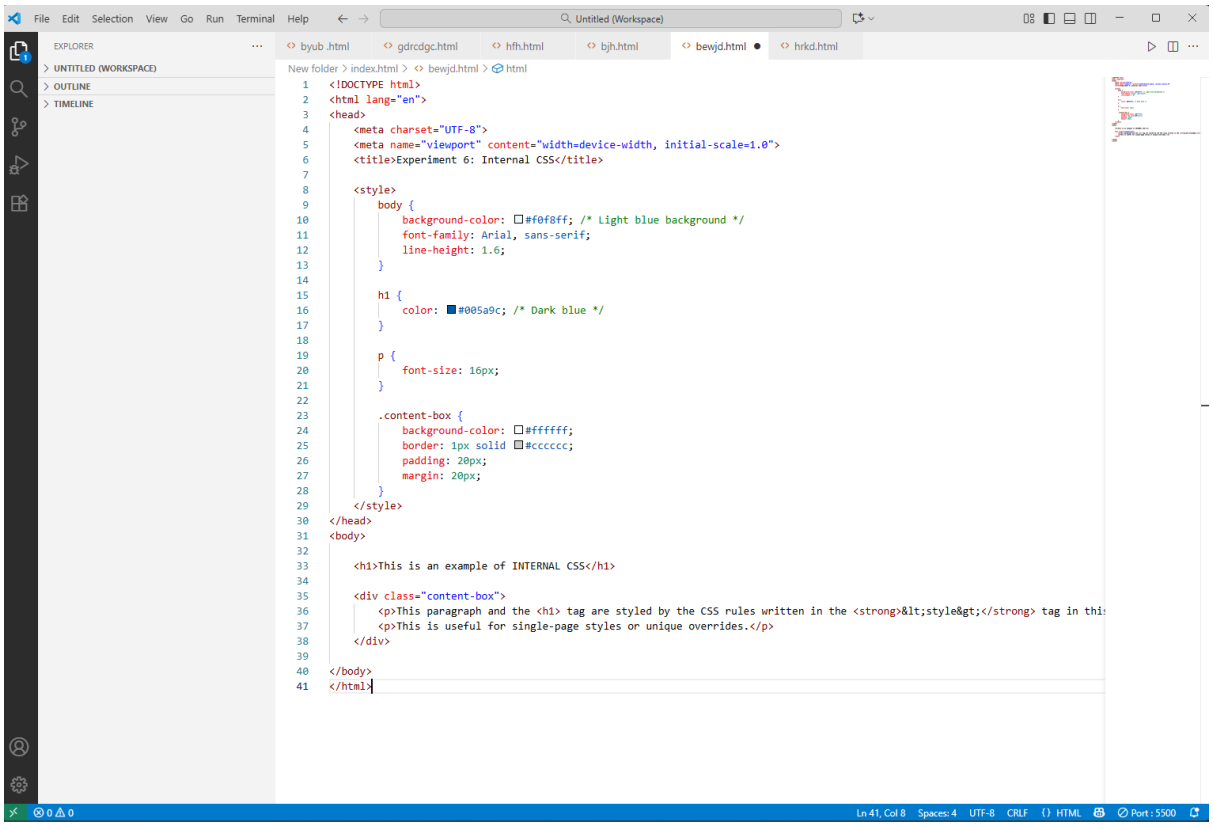

My Blog
Home
About
Contact
What is Semantic HTML?
Posted on October 30, 2025
Semantic HTML elements are those that clearly describe their meaning in a human- and machine-readable way. Elements like <header>, <footer>, <article>, and <section> are all semantic because they accurately describe the purpose of the element and the type of content that is inside them.
Why Use It?
It's great for accessibility (screen readers) and SEO (search engines)!
Related Links
W3C Specification
HTML5 Tutorial
Copyright © 2025 My Blog

Experiment – 6

AIM – 6 : Apply Inline, Internal, and External CSS styles to HTML elements

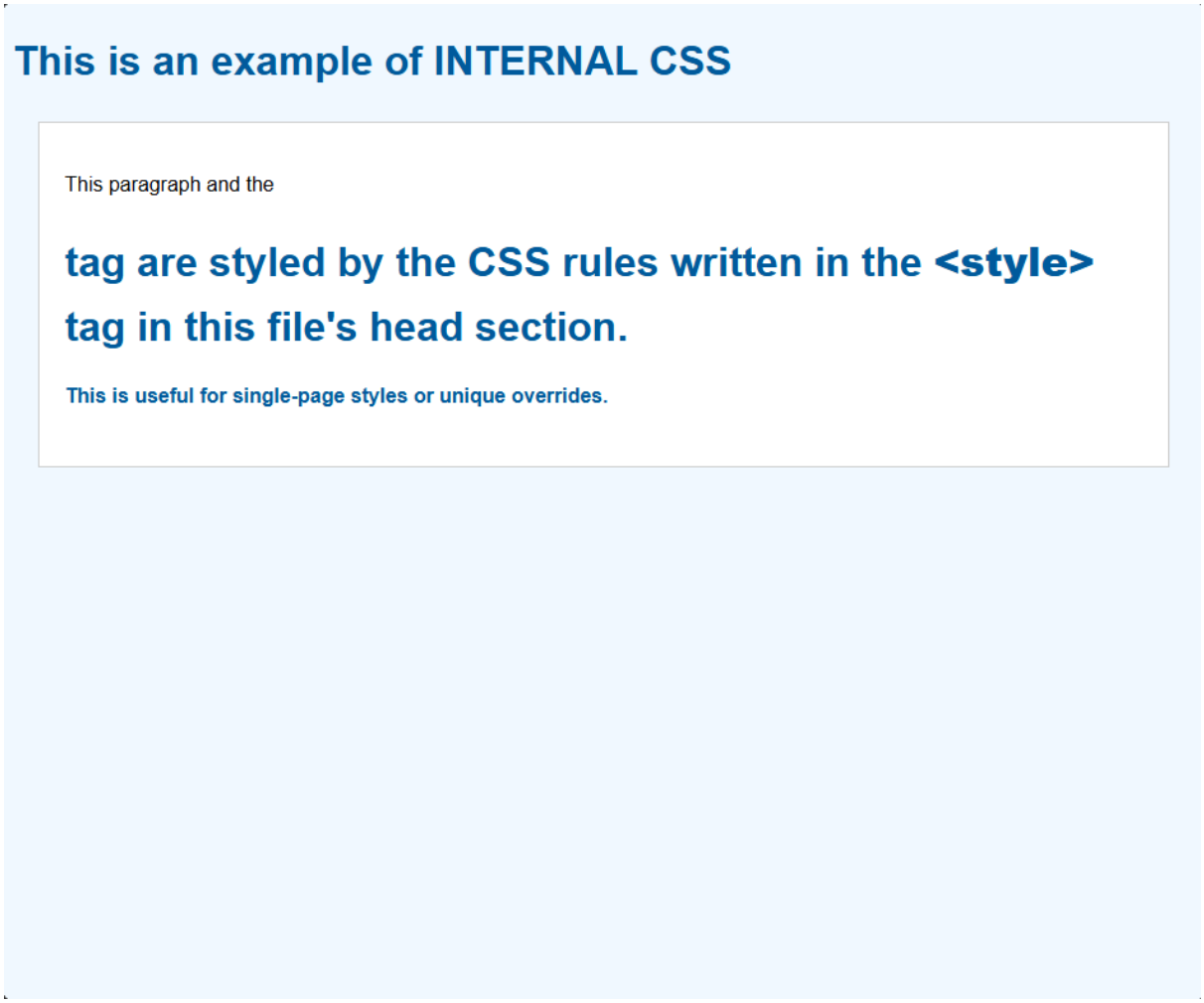
1. Internal css:-

Code:



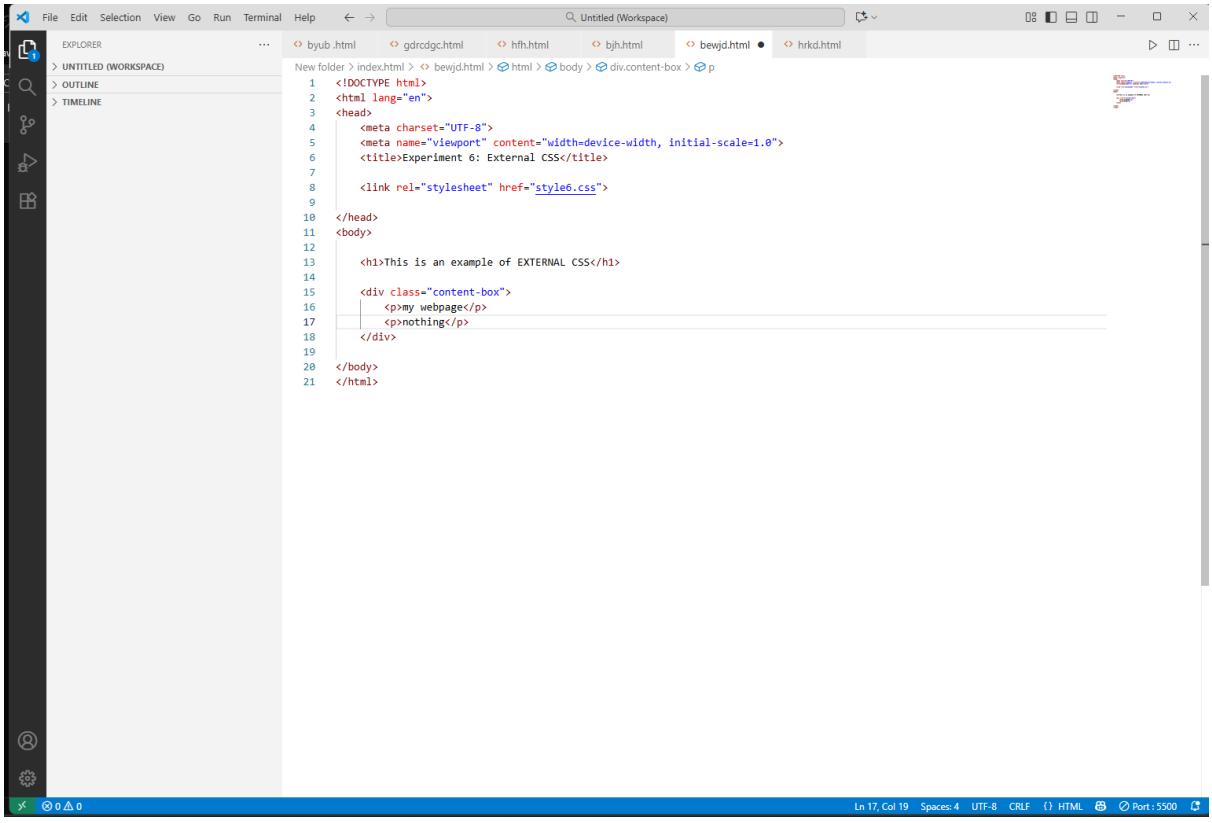
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 6: Internal CSS</title>

    <style>
        body {
            background-color: #f0f8ff; /* Light blue background */
            font-family: Arial, sans-serif;
            line-height: 1.6;
        }

        h1 {
            color: #005a9c; /* Dark blue */
        }

        p {
            font-size: 16px;
        }

        .content-box {
            background-color: #ffffff;
            border: 1px solid #cccccc;
            padding: 20px;
            margin: 20px;
        }
    </style>
</head>
<body>

    <h1>This is an example of INTERNAL CSS</h1>

    <div class="content-box">
        <p>This paragraph and the <h1> tag are styled by the CSS rules written in the <strong>&lt;style&gt;</strong> tag in this
        <p>This is useful for single-page styles or unique overrides.</p>
    </div>

</body>
</html>
```

Output:

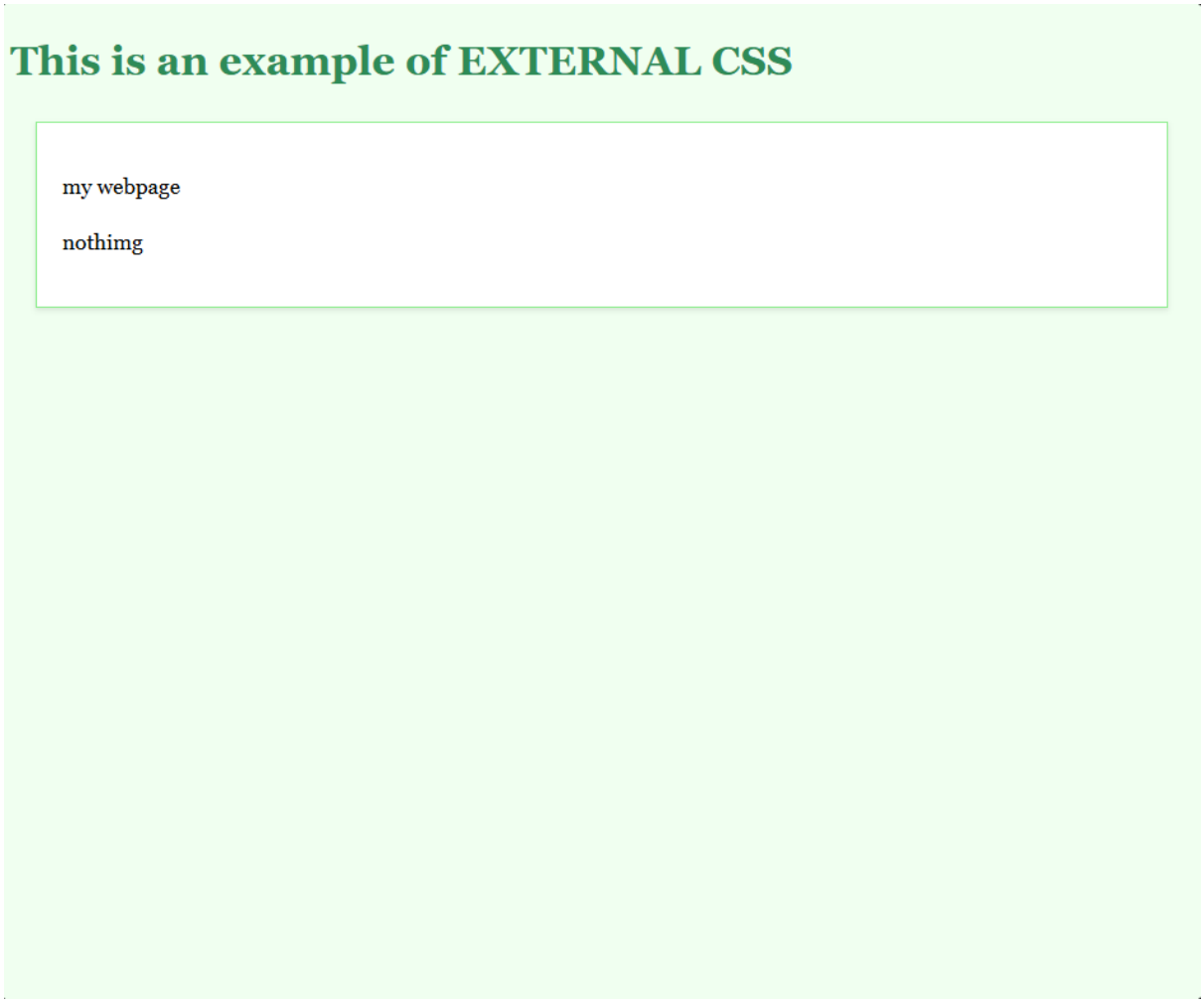


2. External css:-

Code:



Output:

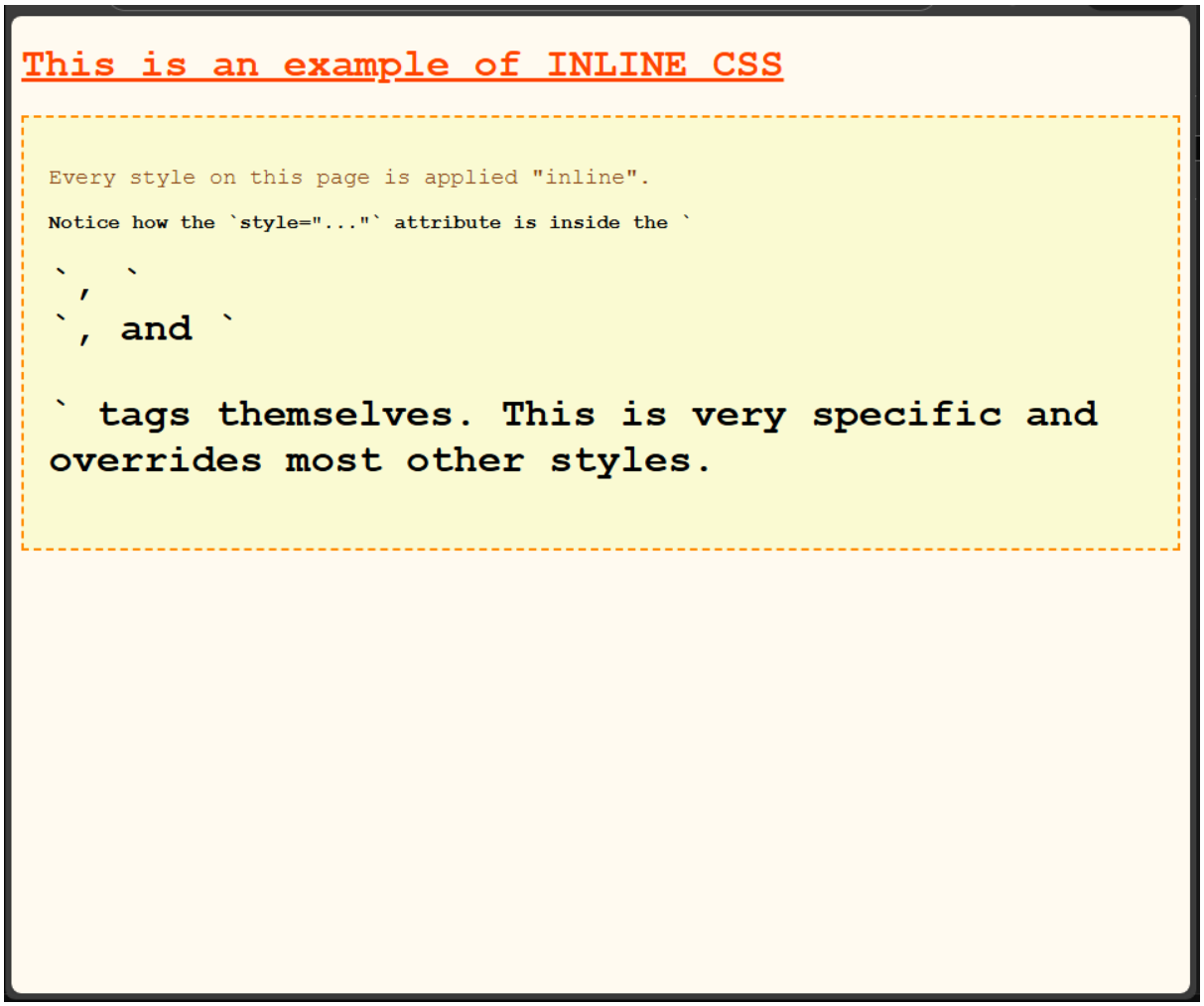


3. Inline css:-

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 6: Inline CSS</title>
</head>
<body style="font-family: 'Courier New', monospace; background-color: #fffaf0;">

    <h1 style="color: #ff4500; text-decoration: underline;">
        This is an example of INLINE CSS
    </h1>

    <div style="background-color: #fafad2; border: 2px dashed #ff8c00; padding: 20px;">
        <p style="font-size: 18px; color: #8b4513;">
            Every style on this page is applied "inline".
        </p>
        <p style="font-weight: bold;">
            Notice how the `style="..."` attribute is inside the `<h1>`, `<div>`, and `<p>` tags themselves. This is very specific and overrides most other styles.
        </p>
    </div>

</body>
</html>
```

Output:


This is an example of INLINE CSS
Every style on this page is applied "inline".
Notice how the `style="..."` attribute is inside the `
` , `
`, and `
` tags themselves. This is very specific and overrides most other styles.

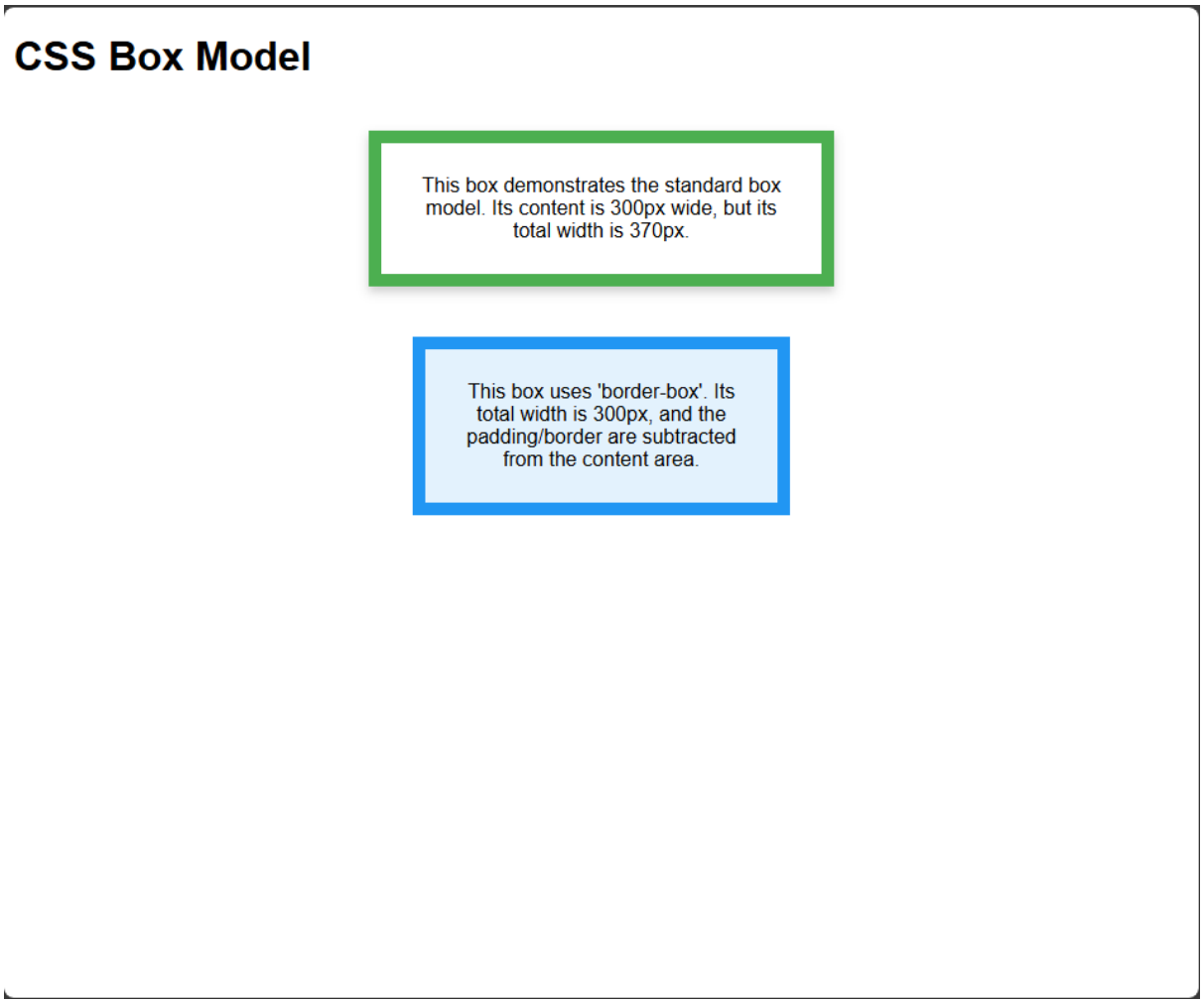
Experiment – 7

AIM – 7 : Style a web page using advanced CSS: Box Model, background images, borders, margins, and padding

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 7: Box Model</title>
    <style>
        body {
            /* Faint background image, set to cover */
            background-image: url('https://www.toptal.com/designers/subtlepatterns/uploads/web-player.png');
            background-size: cover;
            background-attachment: fixed;
            font-family: sans-serif;
        }

        .box-model-demo {
            width: 300px;
            background-color: #ffffff; /* White content area */

            /* 1. PADDING: Space INSIDE the border */
            padding: 25px;

            /* 2. BORDER: The line around the padding/content */
            border: 10px solid #4CAF50; /* Green border */

            /* 3. MARGIN: Space OUTSIDE the border */
            margin: 40px auto; /* 40px top/bottom, 'auto' left/right centers it */

            box-shadow: 0 4px 8px rgba(0,0,0,0.2);
            text-align: center;
        }

        /* The total width of the .box-model-demo element will be:
        Width (300px)
        + Padding (25px left + 25px right = 50px)
        + Border (10px left + 10px right = 20px)
        = 370px

        To make 'width: 300px' define the TOTAL width,
        you can add 'box-sizing: border-box;'
        */

        .box-model-border-box {
            box-sizing: border-box; /* This is the modern, easier way */
            width: 300px;
            background-color: #e3f2fd;
            padding: 25px;
            border: 10px solid #2196F3; /* Blue border */
            margin: 40px auto;
            text-align: center;
        }
        /* This box's total width is exactly 300px */

    </style>
</head>
<body>

    <h1>CSS Box Model</h1>

    <div class="box-model-demo">
        This box demonstrates the standard box model. Its content is 300px wide, but its total width is 370px.
    </div>

    <div class="box-model-border-box">
        This box uses 'border-box'. Its total width is 300px, and the padding/border are subtracted from the content area.
    </div>

</body>
</html>
```

Output:


CSS Box Model
This box demonstrates the standard box model. Its content is 300px wide, but its total width is 370px.
This box uses 'border-box'. Its total width is 300px, and the padding/border are subtracted from the content area.

Experiment – 8

AIM – 8 : Demonstrate the use of different CSS selectors (class, id, descendant, group, universal)

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 8: CSS Selectors</title>
    <style>
        /* 1. Universal Selector: Applies to ALL elements */
        * {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        /* 2. ID Selector: Applies to ONE unique element */
        /* Targets the element with id="main-title" */
        #main-title {
            color: #4a148c; /* Dark purple */
            border-bottom: 2px solid #4a148c;
        }

        /* 3. Class Selector: Applies to all elements with this class */
        /* Targets any element with class="highlight" */
        .highlight {
            background-color: #fff9c4; /* Light yellow */
            padding: 2px 5px;
            border-radius: 3px;
        }

        /* 4. Descendant Selector: Applies to elements INSIDE another */
        /* Targets all <li> elements that are inside a <ul> */
        ul li {
            color: #555;
            margin-bottom: 5px;
        }

        /* 5. Group Selector: Applies one set of rules to multiple selectors */
        /* Targets all <h2> AND all <h3> elements */
        h2, h3 {
            color: #00695c; /* Teal */
        }

        /* Bonus: Child Selector (>) */
        /* Targets only <p> elements that are DIRECT children of <article> */
        article > p {
            font-style: italic;
        }

    </style>
</head>
<body>

    <h1 id="main-title">CSS Selectors Demo</h1>

    <h2>Main Section</h2>

    <article>
        <p>This paragraph is a direct child of 'article', so it is italic.</p>
        <span class="highlight">This span has the 'highlight' class.</span>

        <div>
            <p>This 'p' is NOT a direct child, so it is not italic. But this <span class="highlight">span is also highlighted</span>.</p>
        </div>
    </article>

    <h2>Shopping List</h2>
    <ul>
        <li>Apples</li>
        <li class="highlight">Milk (this 'li' is also highlighted)</li>
        <li>Bread</li>
    </ul>

    <h3>Todo List</h3>
    <ol>
        <li>This 'ol' is not targeted by the 'ul li' selector.</li>
        <li>Finish experiment.</li>
    </ol>

</body>
</html>
```

Output:

CSS Selectors Demo

Main Section

This paragraph is a direct child of 'article', so it is italic.

This span has the 'highlight' class.

This 'p' is NOT a direct child, so it is not italic. But this span is also highlighted .

Shopping List

- Apples
- Milk (this 'li' is also highlighted)
- Bread

Todo List

1. This 'ol' is not targeted by the 'ul li' selector.
2. Finish experiment.

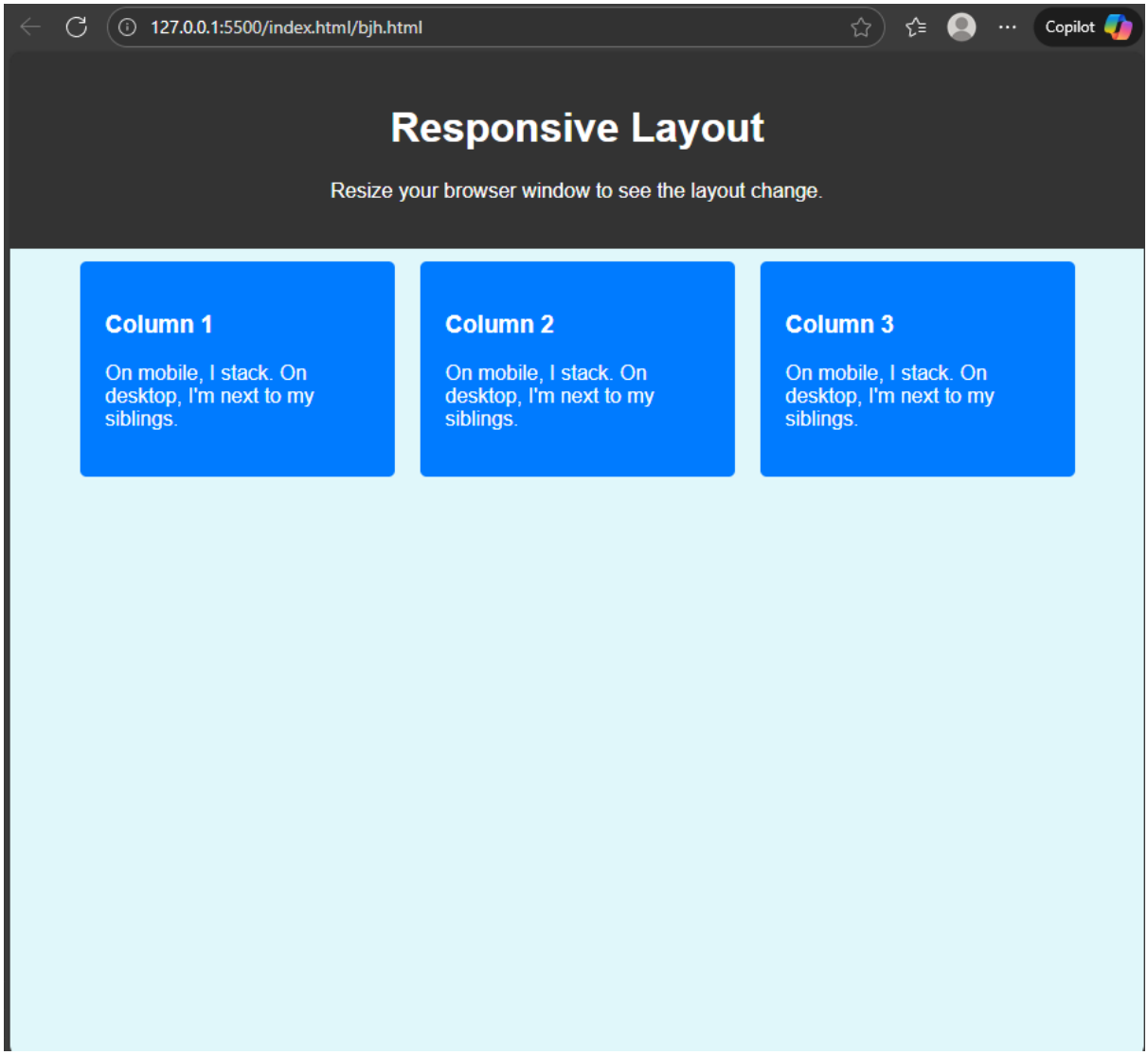
Experiment – 9

AIM – 9 : Create a responsive web page using media queries and mobile-first design with flexible units

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 9: Responsive Design</title>
    <style>
        /* This is a MOBILE-FIRST design */

        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }

        .container {
            width: 100%; /* Flexible unit */
        }

        header {
            background: #333;
            color: #fff;
            padding: 20px;
            text-align: center;
        }

        /* On mobile (default), items stack vertically */
        .grid-container {
            display: block; /* Default for divs */
        }

        .grid-item {
            background: #007BFF;
            color: white;
            padding: 20px;
            margin: 10px;
            border-radius: 5px;
        }

        /* --- This is the Media Query --- */

        /* When the viewport is 768px WIDE or MORE, apply these styles */
        @media (min-width: 768px) {
            body {
                background-color: #e0f7fa; /* Change background on larger screens */
            }

            /* On desktop, use flexbox to create a horizontal layout */
            .grid-container {
                display: flex;
                width: 90%; /* Use 90% of the screen */
                margin: auto;
            }

            .grid-item {
                flex: 1; /* Makes all items share the space equally */
            }
        }
    </style>
</head>
<body>

    <header>
        <h1>Responsive Layout</h1>
        <p>Resize your browser window to see the layout change.</p>
    </header>

    <div class="container">
        <div class="grid-container">
            <div class="grid-item">
                <h3>Column 1</h3>
                <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
            </div>
            <div class="grid-item">
                <h3>Column 2</h3>
                <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
            </div>
            <div class="grid-item">
                <h3>Column 3</h3>
                <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
            </div>
        </div>
    </div>

</body>
</html>
```

Output:


127.0.0.1:5500/index.html/bjh.html
Copilot
Responsive Layout
Resize your browser window to see the layout change.
Column 1
On mobile, I stack. On desktop, I'm next to my siblings.
Column 2
On mobile, I stack. On desktop, I'm next to my siblings.
Column 3
On mobile, I stack. On desktop, I'm next to my siblings.

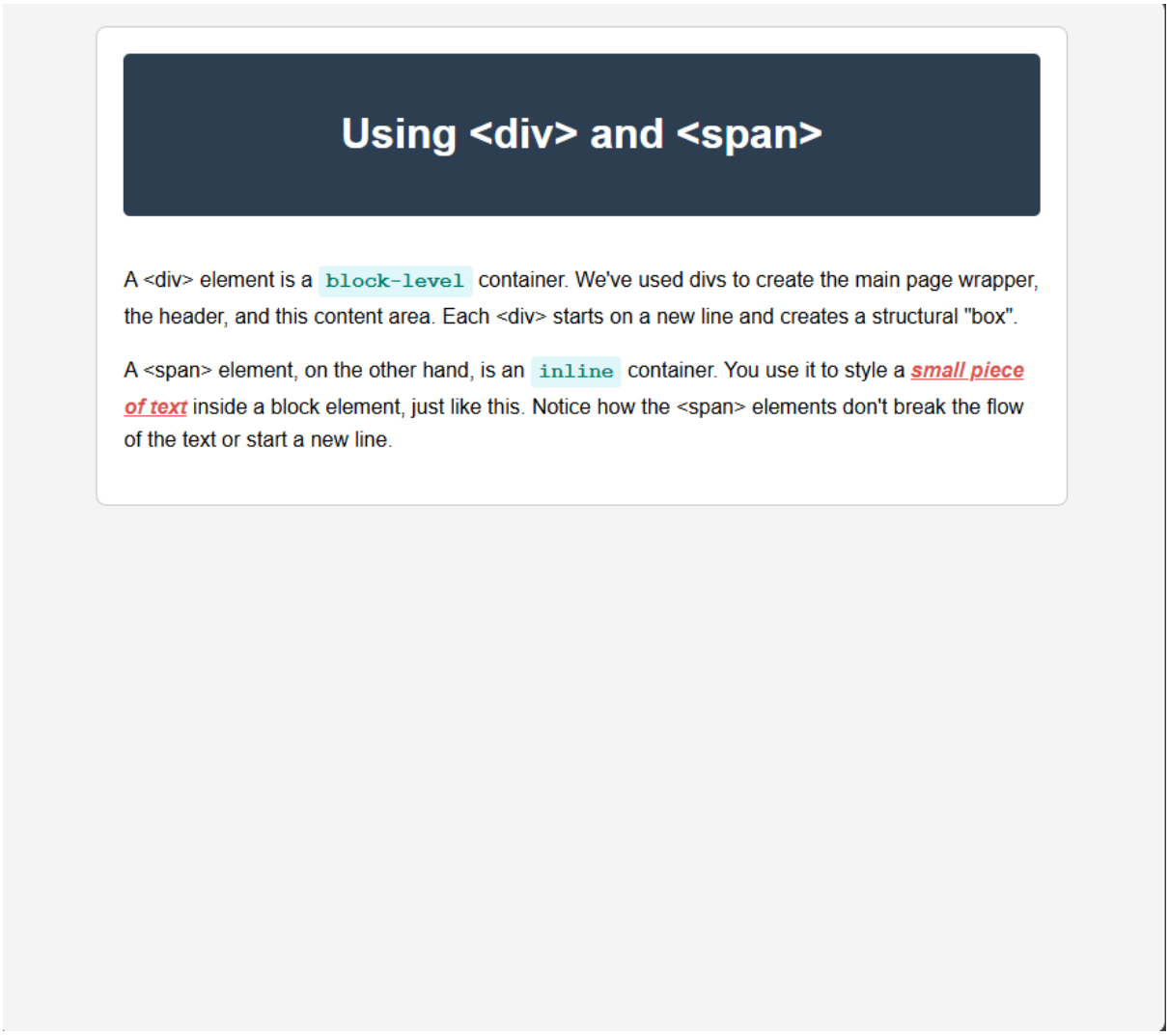
Experiment – 10

AIM – 10 : Use div and span elements for layout and apply styling with CSS

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Experiment 10: Div and Span</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            line-height: 1.6;
        }

        /* --- STYLING <div> ELEMENTS (for Layout) --- */

        /* This <div> acts as the main page wrapper */
        .page-wrapper {
            width: 80%;
            max-width: 800px;
            margin: 20px auto; /* 'auto' centers the block */
            background-color: #ffffff;
            border: 1px solid #ccc;
            border-radius: 8px;
            padding: 20px;
        }

        /* This <div> acts as a "header" section */
        .header-box {
            background-color: #2c3e50;
            color: white;
            padding: 15px;
            text-align: center;
            border-radius: 5px;
        }

        /* This <div> acts as a "content" section */
        .content-box {
            padding-top: 20px;
        }


        /* --- STYLING <span> ELEMENTS (for Inline Text) --- */

        /* This <span> will be used for special labels */
        .label {
            font-family: 'Courier New', Courier, monospace;
            background-color: #e0f7fa;
            color: #00796b;
            padding: 3px 6px;
            border-radius: 4px;
            font-weight: bold;
        }

        /* This <span> will be used for important alerts */
        .alert {
            color: #d9534f; /* Red */
            font-weight: bold;
            font-style: italic;
            text-decoration: underline;
        }

    </style>
</head>
<body>

    <div class="page-wrapper">

        <div class="header-box">
            <h1>Using &lt;div&gt; and &lt;span&gt;</h1>
        </div>

        <div class="content-box">
            <p>
                A &lt;div&gt; element is a <span class="label">block-level</span> container.
                We've used divs to create the main page wrapper, the header, and this content area.
                Each &lt;div&gt; starts on a new line and creates a structural "box".
            </p>

            <p>
                A &lt;span&gt; element, on the other hand, is an <span class="label">inline</span> container.
                You use it to style a <span class="alert">small piece of text</span>
                inside a block element, just like this. Notice how the &lt;span&gt; elements
                don't break the flow of the text or start a new line.
            </p>
        </div>

    </div>

</body>
</html>
```

Output:


Using and
A element is a block-level container. We've used divs to create the main page wrapper, the header, and this content area. Each starts on a new line and creates a structural "box".
A element, on the other hand, is an inline container. You use it to style a small piece of text inside a block element, just like this. Notice how the elements don't break the flow of the text or start a new line.

Experiment-11

AIM-11 : Create a JavaScript-enabled web page that performs basic arithmetic operations using input from users.

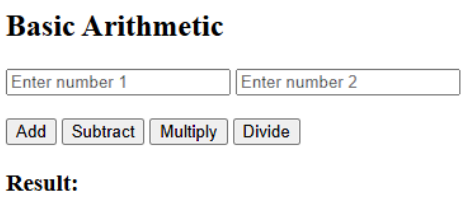
Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Exp 11: Arithmetic Operations</title>
</head>
<body>
    <h2>Basic Arithmetic</h2>
    <input type="number" id="num1" placeholder="Enter number 1">
    <input type="number" id="num2" placeholder="Enter number 2">
    <br><br>
    <button onclick="calculate('+')">Add</button>
    <button onclick="calculate('-')">Subtract</button>
    <button onclick="calculate('*')">Multiply</button>
    <button onclick="calculate('/')">Divide</button>

    <h3 id="result">Result: </h3>

    <script>
        function calculate(operator) {
            // Get values and convert to numbers
            let n1 = parseFloat(document.getElementById('num1').value);
            let n2 = parseFloat(document.getElementById('num2').value);
            let res;

            if (isNaN(n1) || isNaN(n2)) {
                alert("Please enter valid numbers");
                return;
            }

            switch(operator) {
                case '+': res = n1 + n2; break;
                case '-': res = n1 - n2; break;
                case '*': res = n1 * n2; break;
                case '/': res = n2 !== 0 ? n1 / n2 : "Cannot divide by zero"; break;
            }

            document.getElementById('result').innerText = "Result: " + res;
        }
    </script>
</body>
</html>
```

Output:


Basic Arithmetic
Enter number 1
Enter number 2
Add
Subtract
Multiply
Divide
Result:

Experiment-12

AIM-12: Write JavaScript to demonstrate conditional statements and looping (for, while, do...while).

Code:

fhh.html

New folder > index.html > fhh.html > html > body > script

```
<!DOCTYPE html>
<html>
<body>
    <h2>Loops and Conditions Demo</h2>
    <div id="output"></div>

    <script>
        let outputDiv = document.getElementById("output");
        let html = "";

        // 1. Conditional (If/Else)
        let hour = new Date().getHours();
        html += "<strong>Conditional:</strong> ";
        if (hour < 12) {
            html += "Good Morning!<br>";
        } else {
            html += "Good Afternoon/Evening!<br>";
        }

        // 2. For Loop
        html += "<br><strong>For Loop (0 to 4):</strong> ";
        for (let i = 0; i < 5; i++) {
            html += i + " ";
        }

        // 3. While Loop
        html += "<br><strong>While Loop (Count down 5 to 1):</strong> ";
        let j = 5;
        while (j > 0) {
            html += j + " ";
            j--;
        }

        // 4. Do...While Loop
        html += "<br><strong>Do...While (Runs at least once):</strong> ";
        let k = 0;
        do {
            html += "Executed! ";
            k++;
        } while (k < 1);

        outputDiv.innerHTML = html;
    </script>
</body>
</html>
```

Output:

Loops and Conditions Demo

Conditional: Good Afternoon/Evening!

For Loop (0 to 4): 0 1 2 3 4
While Loop (Count down 5 to 1): 5 4 3 2 1
Do...While (Runs at least once): Executed!

Experiment-13

AIM-13: *Create and invoke user-defined functions; use var, let, and const for scope demonstration.*

HTML

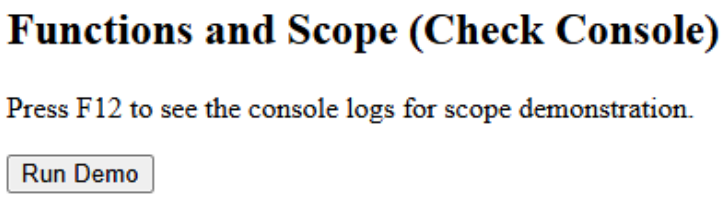
Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<body>
    <h2>Functions and Scope (Check Console)</h2>
    <p>Press F12 to see the console logs for scope demonstration.</p>
    <button onclick="demoScope()">Run Demo</button>

    <script>
        function demoScope() {
            console.log("--- Scope Demo Started ---");

            // Function Scope
            var functionScoped = "I am var (Function Scoped)";

            if (true) {
                // Block Scope
                let blockScoped = "I am let (Block Scoped)";
                const constantVar = "I am const (Cannot Change)";
                var leakVar = "I am var (I leak out of blocks)";

                console.log("Inside Block: " + blockScoped);
                console.log("Inside Block: " + constantVar);
            }

            // Demonstrating accessibility
            console.log("Outside Block: " + functionScoped);
            console.log("Outside Block: " + leakVar); // Works because var ignores block scope

            try {
                console.log(blockScoped); // This will throw error
            } catch (e) {
                console.log("Error: 'let' cannot be accessed outside its block.");
            }
        }
    </script>
</body>
</html>
```

Output:


Functions and Scope (Check Console)
Press F12 to see the console logs for scope demonstration.
Run Demo

Experiment-14

AIM-14: Handle HTML form validation using onsubmit, oninput, and alert().

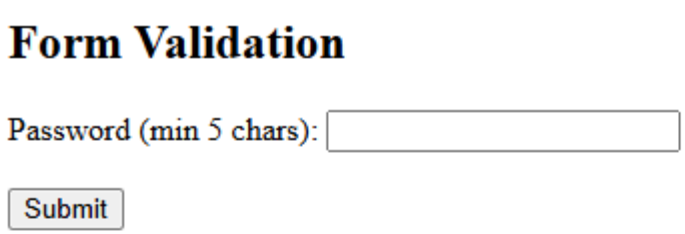
Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<body>
    <h2>Form Validation</h2>
    <form onsubmit="return validateForm()">
        Password (min 5 chars):
        <input type="password" id="pass" oninput="checkLength()">
        <span id="msg" style="color: red"></span>
        <br><br>
        <button type="submit">Submit</button>
    </form>

    <script>
        // Check input while typing (oninput)
        function checkLength() {
            let val = document.getElementById('pass').value;
            let msg = document.getElementById('msg');
            if(val.length < 5) {
                msg.innerText = "Too short!";
            } else {
                msg.innerText = "Good length.";
                msg.style.color = "green";
            }
        }

        // Check on submission (onsubmit)
        function validateForm() {
            let val = document.getElementById('pass').value;
            if (val == "" || val.length < 5) {
                alert("Validation Failed: Password must be at least 5 characters.");
                return false; // Prevents form submission
            }
            alert("Form Submitted Successfully!");
            return true;
        }
    </script>
</body>
</html>
```

Output:



Experiment-15

AIM-15: Demonstrate DOM manipulation using getElementById(), querySelector(), and innerHTML.

Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<body>
    <h2 id="header">Original Header</h2>
    <p class="text-content">Original paragraph text.</p>
    <button onclick="changeContent()">Manipulate DOM</button>

    <script>
        function changeContent() {
            // Using getElementById
            let header = document.getElementById("header");
            header.innerHTML = "<em>Header Changed!</em>";
            header.style.color = "blue";

            // Using querySelector (selects first element with class .text-content)
            let para = document.querySelector(".text-content");
            para.innerText = "Paragraph text changed using querySelector.";
        }
    </script>
</body>
</html>
```

Output:

Experiment-16

AIM-16: Show or hide HTML elements and toggle CSS classes dynamically.

Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        .highlight {
            background-color: yellow;
            font-weight: bold;
            padding: 10px;
        }
        #box {
            width: 100px;
            height: 100px;
            background-color: lightblue;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <h2>Show/Hide & Toggle</h2>

    <button onclick="toggleVisibility()">Show/Hide Box</button>
    <button onclick="toggleClass()">Toggle CSS Class</button>

    <div id="box">I am a box</div>
    <p id="text">I am text for class toggling.</p>

    <script>
        function toggleVisibility() {
            let box = document.getElementById("box");
            // Check current display state
            if (box.style.display === "none") {
                box.style.display = "block";
            } else {
                box.style.display = "none";
            }
        }

        function toggleClass() {
            let text = document.getElementById("text");
            // Toggle the 'highlight' class defined in CSS
            text.classList.toggle("highlight");
        }
    </script>
</body>
</html>
```

Output:


Show/Hide & Toggle
Show/Hide Box
Toggle CSS Class
I am a box
I am text for class toggling.

Experiment-17

AIM-17: Add interactivity using event listeners (addEventListener) for mouse/keyboard events.

Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<body>
    <h2>Event Listeners</h2>
    <div id="mouseBox" style="border:1px solid black; padding:20px; width:200px;">
        Hover over me!
    </div>
    <br>
    <input type="text" id="keyInput" placeholder="Type something here...">
    <p id="status">Status: Waiting...</p>

    <script>
        let box = document.getElementById("mouseBox");
        let input = document.getElementById("keyInput");
        let status = document.getElementById("status");

        // Mouse Event
        box.addEventListener("mouseover", function() {
            box.style.backgroundColor = "lightgreen";
            status.innerText = "Status: Mouse is inside!";
        });

        box.addEventListener("mouseout", function() {
            box.style.backgroundColor = "white";
            status.innerText = "Status: Mouse is outside!";
        });

        // Keyboard Event
        input.addEventListener("keyup", function(event) {
            status.innerText = "You pressed: " + event.key;
        });
    </script>
</body>
</html>
```

Output:


Event Listeners
Hover over me!
Type something here...
You pressed: PrintScreen

Experiment-18

AIM-18: Create and manipulate elements using append, remove, or modify child nodes.

Code:

fhh.html

New folder > index.html > fhh.html > html

```html
<!DOCTYPE html>
<html>
<body>
    <h2>Node Manipulation</h2>
    <input type="text" id="newItem" placeholder="New Item Name">
    <button onclick="addItem()">Add Item</button>
    <button onclick="removeLastItem()">Remove Last Item</button>

    <ul id="itemList">
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>

    <script>
        function addItem() {
            let list = document.getElementById("itemList");
            let val = document.getElementById("newItem").value;

            if(val) {
                // Create new element
                let newLi = document.createElement("li");
                // Modify content
                newLi.innerText = val;
                // Append to parent
                list.appendChild(newLi);
            }
        }

        function removeLastItem() {
            let list = document.getElementById("itemList");
            // Check if there are children to remove
            if (list.lastElementChild) {
                list.removeChild(list.lastElementChild);
            }
        }
    </script>
</body>
</html>
```

Output:


Node Manipulation
New Item Name
Add Item
Remove Last Item
Item 1
Item 2

Experiment-19

AIM-19: Create a simple drawing using the canvas element: draw shapes and fill colors.

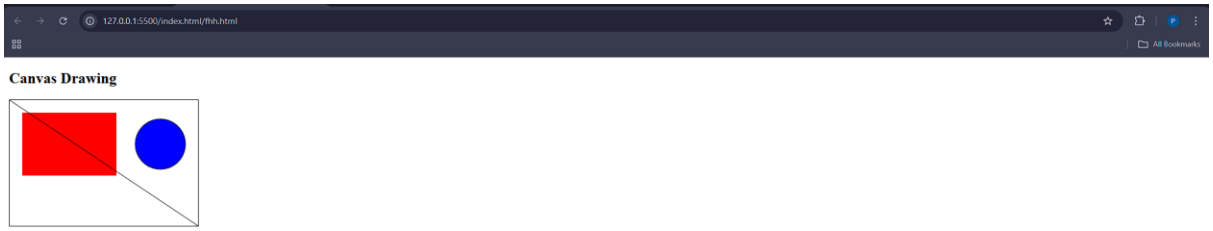
Code:

fhh.html

New folder > index.html > fhh.html > html > body > script

```
<!DOCTYPE html>
<html>
<body>
    <h2>Canvas Drawing</h2>
    <canvas id="myCanvas" width="300" height="200" style="border:1px solid #000000;">
        Your browser does not support the HTML canvas tag.
    </canvas>

    <script>
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");

        // Draw a Rectangle (Red)
        ctx.fillStyle = "#FF0000";
        ctx.fillRect(20, 20, 150, 100);

        // Draw a Circle (Blue)
        ctx.beginPath();
        ctx.arc(240, 70, 40, 0, 2 * Math.PI);
        ctx.fillStyle = "blue";
        ctx.fill();
        ctx.stroke();

        // Draw a Line
        ctx.moveTo(0, 0);
        ctx.lineTo(300, 200);
        ctx.stroke();
    </script>
</body>
</html>
```

Output:


127.0.0.1:5500/index.html/fhh.html
All Bookmarks
Canvas Drawing