

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:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows an untitled workspace with files: byub.html, gdrcdgch.html, hfh.html, bjh.html, hrkd.html, and byfng.html (the active file).
- Code Editor:** Displays the content of the 'byfng.html' file. The code includes:
 - Doctype declaration: `<!DOCTYPE html>`
 - HTML structure: ``
 - Head section: `` with meta charset="UTF-8" and viewport content="width=device-width, initial-scale=1.0".
 - Title: `Experiment 1: Basic HTML`
 - Body section: ``
 - Section 1: `

This is a Main Heading (h1)

`
 - Text: `

This is a standard paragraph. Paragraphs are block-level elements used for most of the text content on a webpage. They si
 - Section 2: `

This is a Sub-Heading (h2)
 - Text: `

This paragraph demonstrates various **text formatting tags**.
 - Text:

You can make text **bold** (`important`) or *emphasized* (`italic`).
 - Text:

You can also use older tags like **this bold text** and *this italic text*, but '`strong`' and '`em`' are generally
 - Text:

Here are more formatting examples:
 - List: `
 - This text is highlighted.
 - This text is underlined.
 - This is _{subscript} text (like in H_{sub}2₀).
 - This is ^{superscript} text (like in E=mc²).
 - This text is smaller.
 - This text is ~~deleted~~.
 - Text: `

Another Sub-Heading (h3)
 - Text:

Headings have different levels of importance, from '`h1`' (the most important) down to '`h6`' (the least important).
 - End of body: ``
 - End of html: ``
- Bottom Status Bar:** ShowsLn 1, Col 1, Spaces:4, UTF-8, CRLF, HTML, Go Live, and a search icon.

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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 2: Links, Images, Lists</title>
7  </head>
8  <body>
9
10     <h1>HTML Links</h1>
11     <p>This is an <a href="https://www.google.com" target="_blank">external link</a> to Google (opens in a new tab).</p>
12
13     <p>This is an <a href="about.html">internal link</a> to an About page.</p>
14
15     <h1>HTML Image</h1>
16
17     
18     <p>The 'alt' attribute provides descriptive text for screen readers or if the image fails to load.</p>
19
20     <h1>HTML Lists</h1>
21
22     <h3>Unordered List (Bulleted)</h3>
23     <ul>
24         <li>Coffee</li>
25         <li>Tea</li>
26         <li>Milk</li>
27     </ul>
28
29     <h3>Ordered List (Numbered)</h3>
30     <ol>
31         <li>First, wake up.</li>
32         <li>Second, make coffee.</li>
33         <li>Third, code.</li>
34     </ol>
35
36     <h3>Nested List</h3>
37     <ul>
38         <li>Front-End
39             <ol>
40                 <li>HTML</li>
41                 <li>CSS</li>
42                 <li>JavaScript</li>
43             </ol>
44         </li>
45         <li>Back-End
46             <ol>
47                 <li>Python</li>
48                 <li>Node.js</li>
49                 <li>PHP</li>
50             </ol>
51         </li>
52     </ul>
53
54 </body>
55 </html>
```

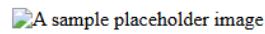
Output:

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

A sample placeholder 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:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Experiment 3: HTML Table</title>
7     <style>
8         table, th, td {
9             border: 1px solid black;
10            border-collapse: collapse;
11            padding: 8px;
12        }
13        caption {
14            font-weight: bold;
15            font-size: 1.2em;
16            margin: 8px;
17        }
18    </style>
19 </head>
20 <body>
21
22     <h1>Student Time Table</h1>
23
24     <table>
25         <caption>Weekly Class Schedule</caption>
26         <thead>
27             <tr>
28                 <th>Day</th>
29                 <th>9:00 AM</th>
30                 <th>10:00 AM</th>
31                 <th>11:00 AM</th>
32                 <th>12:00 PM</th>
33             </tr>
34         </thead>
35         <tbody>
36             <tr>
37                 <td>Monday</td>
38                 <td>Math</td>
39                 <td>Science</td>
40                 <td>English</td>
41                 <td colspan="2" align="center">Project Work</td>
42             </tr>
43             <tr>
44                 <td>Tuesday</td>
45                 <td rowspan="2" align="center">Physics Lab</td>
46                 <td>History</td>
47                 <td>Math</td>
48                 <td>PE</td>
49             </tr>
50             <tr>
51                 <td>Wednesday</td>
52                 <td>Chemistry</td>
53                 <td>Geography</td>
54                 <td>English</td>
55             </tr>
56             <tr>
57                 <td>Thursday</td>
58                 <td>Math</td>
59                 <td>Science</td>
60                 <td colspan="2" align="center">Library</td>
61             </tr>
62             <tr>
63                 <td>Friday</td>
64                 <td colspan="4" align="center">Assembly & Half Day</td>
65             </tr>
66         </tbody>
67     </table>
68
69 </body>
70 </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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 4: HTML Form</title>
7      <style>
8          body { font-family: sans-serif; }
9          div { margin-bottom: 15px; }
10         label { display: block; margin-bottom: 5px; font-weight: bold; }
11         input[type="text"] { width: 300px; padding: 5px; }
12     </style>
13 </head>
14 <body>
15
16     <h1>Event Registration Form</h1>
17
18     <form action="#" method="POST">
19
20         <div>
21             <label for="name">Full Name:</label>
22             <input type="text" id="name" name="user_name" required>
23         </div>
24
25         <div>
26             <label>Gender:</label>
27             <input type="radio" id="male" name="gender" value="male">
28             <label for="male" style="display:inline;">Male</label><br>
29             <input type="radio" id="female" name="gender" value="female">
30             <label for="female" style="display:inline;">Female</label><br>
31             <input type="radio" id="other" name="gender" value="other">
32             <label for="other" style="display:inline;">Other</label>
33         </div>
34
35         <div>
36             <label>Interests:</label>
37             <input type="checkbox" id="interest1" name="interests" value="music">
38             <label for="interest1" style="display:inline;">Music</label><br>
39             <input type="checkbox" id="interest2" name="interests" value="sports">
40             <label for="interest2" style="display:inline;">Sports</label><br>
41             <input type="checkbox" id="interest3" name="interests" value="tech">
42             <label for="interest3" style="display:inline;">Technology</label>
43         </div>
44
45         <div>
46             <label for="session">Preferred Session:</label>
47             <select id="session" name="session_choice">
48                 <option value="">--Please choose an option--</option>
49                 <option value="morning">Morning (9 AM - 12 PM)</option>
50                 <option value="afternoon">Afternoon (1 PM - 4 PM)</option>
51                 <option value="evening">Evening (6 PM - 9 PM)</option>
52             </select>
53         </div>
54
55         <div>
56             <button type="submit">Submit Registration</button>
57             <button type="reset">Clear Form</button>
58         </div>
59
60     </form>
61
62 </body>
63 </html>
```

Output:

Event Registration Form

Full Name:

Gender:

- Male
- Female
- Other

Interests:

- Music
- Sports
- Technology

Preferred Session:

Experiment – 5

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

Code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 5: Semantic HTML</title>
7      <style>
8          body { font-family: sans-serif; background: #f4f4f4; }
9          .container { max-width: 960px; margin: auto; background: #fff; }
10         header { background: #333; color: #fff; padding: 10px; text-align: center; }
11         nav { background: #555; padding: 10px; }
12         nav a { color: #fff; padding: 10px; text-decoration: none; }
13         .main-content { display: flex; }
14         article { flex: 3; padding: 20px; }
15         aside { flex: 1; background: #f9f9f9; padding: 20px; }
16         footer { background: #333; color: #fff; text-align: center; padding: 20px; margin-top: 20px; }
17     </style>
18 </head>
19 <body>
20
21     <div class="container">
22
23         <header>
24             <h1>My Blog</h1>
25         </header>
26
27         <nav>
28             <a href="#">Home</a>
29             <a href="#">About</a>
30             <a href="#">Contact</a>
31         </nav>
32
33         <div class="main-content">
34             <section>
35                 <article>
36                     <h2>What is Semantic HTML?</h2>
37                     <p>Posted on October 30, 2025</p>
38                     <p>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.</p>
39                 </article>
40
41                 <article>
42                     <h2>Why use It?</h2>
43                     <p>It's great for accessibility (screen readers) and SEO (search engines)!</p>
44                 </article>
45             </section>
46
47             <aside>
48                 <h3>Related Links</h3>
49                 <ul>
50                     <li><a href="#">W3C Specification</a></li>
51                     <li><a href="#">HTML5 Tutorial</a></li>
52                 </ul>
53             </aside>
54         </div>
55
56         <footer>
57             <p>Copyright © 2025 My Blog</p>
58         </footer>
59
60     </div>
61
62 </body>
63 </html>
```

Output:

The screenshot shows a blog website with a dark header bar containing the title "My Blog". Below the header is a navigation bar with links to "Home", "About", and "Contact". The main content area features a section titled "What is Semantic HTML?" with a timestamp "Posted on October 30, 2025". It includes a descriptive paragraph about semantic HTML elements. To the right of the main content is a sidebar titled "Related Links" with two items: "W3C Specification" and "HTML5 Tutorial". Below the main content is a footer bar with the copyright notice "Copyright © 2025 My Blog".

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)!

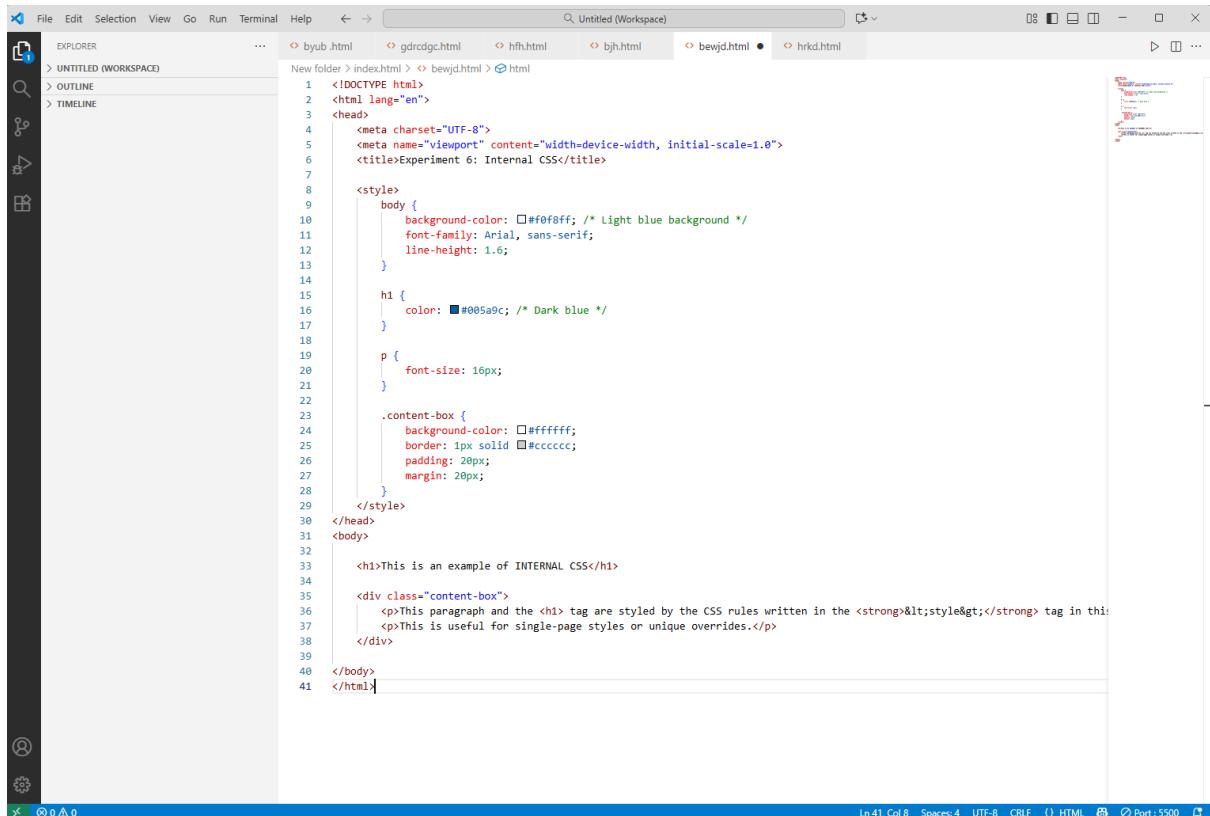
Copyright © 2025 My Blog

Experiment – 6

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

1. Internal css:-

Code:



The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Standard file operations like Open, Save, Print, etc.
- Explorer:** Shows an Untitled workspace with files: byub.html, gdrcdgc.html, hfh.html, bjh.html, bewjd.html (selected), and hrkd.html.
- Code Editor:** The main area contains an HTML document with internal CSS. The code is as follows:

```
<!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: #f0f0ff; /* 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 page. This is useful for single-page styles or unique overrides.</p>
    </div>
</body>
</html>
```

Status Bar: Ln 41, Col 8 Spaces: 4 UTF-8 CRLF HTML Port: 5500

Output:

This is an example of INTERNAL CSS

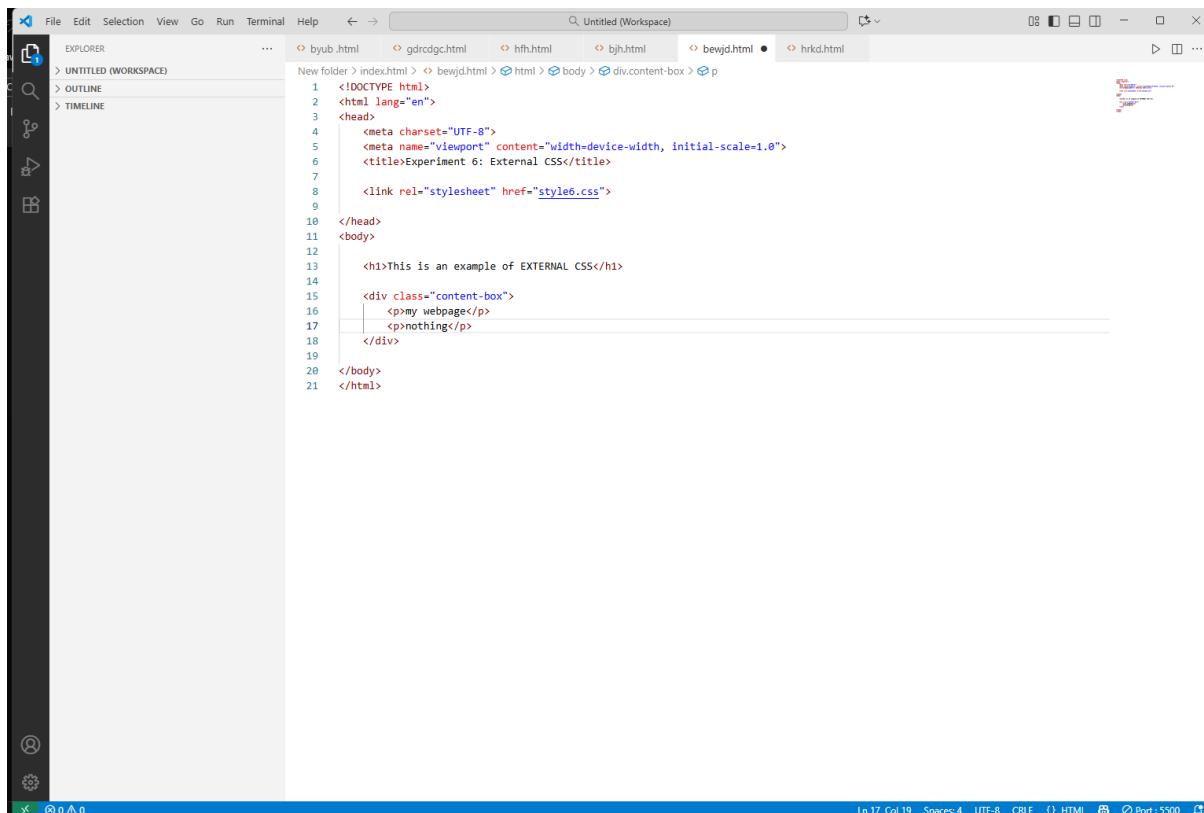
This paragraph and the

tag are styled by the CSS rules written in the <style> tag in this file's head section.

This is useful for single-page styles or unique overrides.

2. External css:-

Code:



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows a workspace named "UNTITLED (WORKSPACE)" containing files: byub.html, gdrcdgc.html, hfh.html, bjh.html, bewjd.html, and hrkd.html.
- Code Editor:** An HTML file with the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Experiment 6: External CSS</title>
<link rel="stylesheet" href="style6.css">
</head>
<body>
<h1>This is an example of EXTERNAL CSS</h1>
<div class="content-box">
<p>my webpage</p>
<p>nothing</p>
</div>
</body>
</html>
```
- Status Bar:** Shows "Ln 17, Col 19" and "Port : 5500".

Output:

This is an example of EXTERNAL CSS

my webpage

nothimg

3. Inline css:-

Code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Experiment 6: Inline CSS</title>
7 </head>
8 <body style="font-family: 'Courier New', monospace; background-color: #ffffaf0;">
9
10  <h1 style="color: #ff4500; text-decoration: underline;">
11    This is an example of INLINE CSS
12  </h1>
13
14  <div style="background-color: #fafad2; border: 2px dashed #ff8c00; padding: 20px;">
15    <p style="font-size: 18px; color: #8b4513;">
16      Every style on this page is applied "inline".
17    </p>
18    <p style="font-weight: bold;">
19      Notice how the `style="..."` attribute is inside the `<h1>`, `<div>`, and `<p>` tags themselves. This is very specific and overrides most other styles.
20    </p>
21  </div>
22
23 </body>
24 </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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 7: Box Model</title>
7      <style>
8          body {
9              /* Faint background image, set to cover */
10             background-image: url('https://www.toptal.com/designers/subtlepatterns/uploads/web-player.png');
11             background-size: cover;
12             background-attachment: fixed;
13             font-family: sans-serif;
14         }
15
16         .box-model-demo {
17             width: 300px;
18             background-color: #ffffff; /* White content area */
19
20             /* 1. PADDING: Space INSIDE the border */
21             padding: 25px;
22
23             /* 2. BORDER: The line around the padding/content */
24             border: 10px solid #4CAF50; /* Green border */
25
26             /* 3. MARGIN: Space OUTSIDE the border */
27             margin: 40px auto; /* 40px top/bottom, 'auto' left/right centers it */
28
29             box-shadow: 0 4px 8px rgba(0,0,0,0.2);
30             text-align: center;
31         }
32
33         /* The total width of the .box-model-demo element will be:
34         Width (300px)
35         + Padding (25px left + 25px right = 50px)
36         + Border (10px left + 10px right = 20px)
37         = 370px
38
39         To make 'width: 300px' define the TOTAL width,
40         you can add 'box-sizing: border-box';
41         */
42
43         .box-model-border-box {
44             box-sizing: border-box; /* This is the modern, easier way */
45             width: 300px;
46             background-color: #e3f2fd;
47             padding: 25px;
48             border: 10px solid #2196F3; /* Blue border */
49             margin: 40px auto;
50             text-align: center;
51         }
52         /* This box's total width is exactly 300px */
53
54     </style>
55 </head>
56 <body>
57
58     <h1>CSS Box Model</h1>
59
60     <div class="box-model-demo">
61         This box demonstrates the standard box model. Its content is 300px wide, but its total width is 370px.
62     </div>
63
64     <div class="box-model-border-box">
65         This box uses 'border-box'. Its total width is 300px, and the padding/border are subtracted from the content area.
66     </div>
67
68 </body>
69 </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:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 8: CSS Selectors</title>
7      <style>
8          /* 1. Universal Selector: Applies to ALL elements */
9          * {
10              font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
11          }
12
13          /* 2. ID Selector: Applies to ONE unique element */
14          /* Targets the element with id="main-title" */
15          #main-title {
16              color: #4a148c; /* Dark purple */
17              border-bottom: 2px solid #4a148c;
18          }
19
20          /* 3. Class Selector: Applies to all elements with this class */
21          /* Targets any element with class="highlight" */
22          .highlight {
23              background-color: #ffff9c4; /* Light yellow */
24              padding: 2px 5px;
25              border-radius: 3px;
26          }
27
28          /* 4. Descendant Selector: Applies to elements INSIDE another */
29          /* Targets all <li> elements that are inside a <ul> */
30          ul li {
31              color: #555;
32              margin-bottom: 5px;
33          }
34
35          /* 5. Group Selector: Applies one set of rules to multiple selectors */
36          /* Targets all <h2> AND all <h3> elements */
37          h2, h3 {
38              color: #80095c; /* Teal */
39          }
40
41          /* Bonus: Child Selector (>) */
42          /* Targets only <p> elements that are DIRECT children of <article> */
43          article > p {
44              font-style: italic;
45          }
46
47      </style>
48  </head>
49  <body>
50
51      <h1 id="main-title">CSS Selectors Demo</h1>
52
53      <h2>Main Section</h2>
54
55      <article>
56          <p>This paragraph is a direct child of 'article', so it is italic.</p>
57          <span class="highlight">This span has the 'highlight' class.</span>
58
59          <div>
60              <p>This 'p' is NOT a direct child, so it is not italic. But this <span class="highlight">span is also highlighted</span>.</p>
61          </div>
62      </article>
63
64      <h2>Shopping List</h2>
65      <ul>
66          <li>Apples</li>
67          <li class="highlight">Milk (this 'li' is also highlighted)</li>
68          <li>Bread</li>
69      </ul>
70
71      <h3>Todo List</h3>
72      <ol>
73          <li>This 'ol' is not targeted by the 'ul li' selector.</li>
74          <li>Finish experiment.</li>
75      </ol>
76
77  </body>
78  </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.

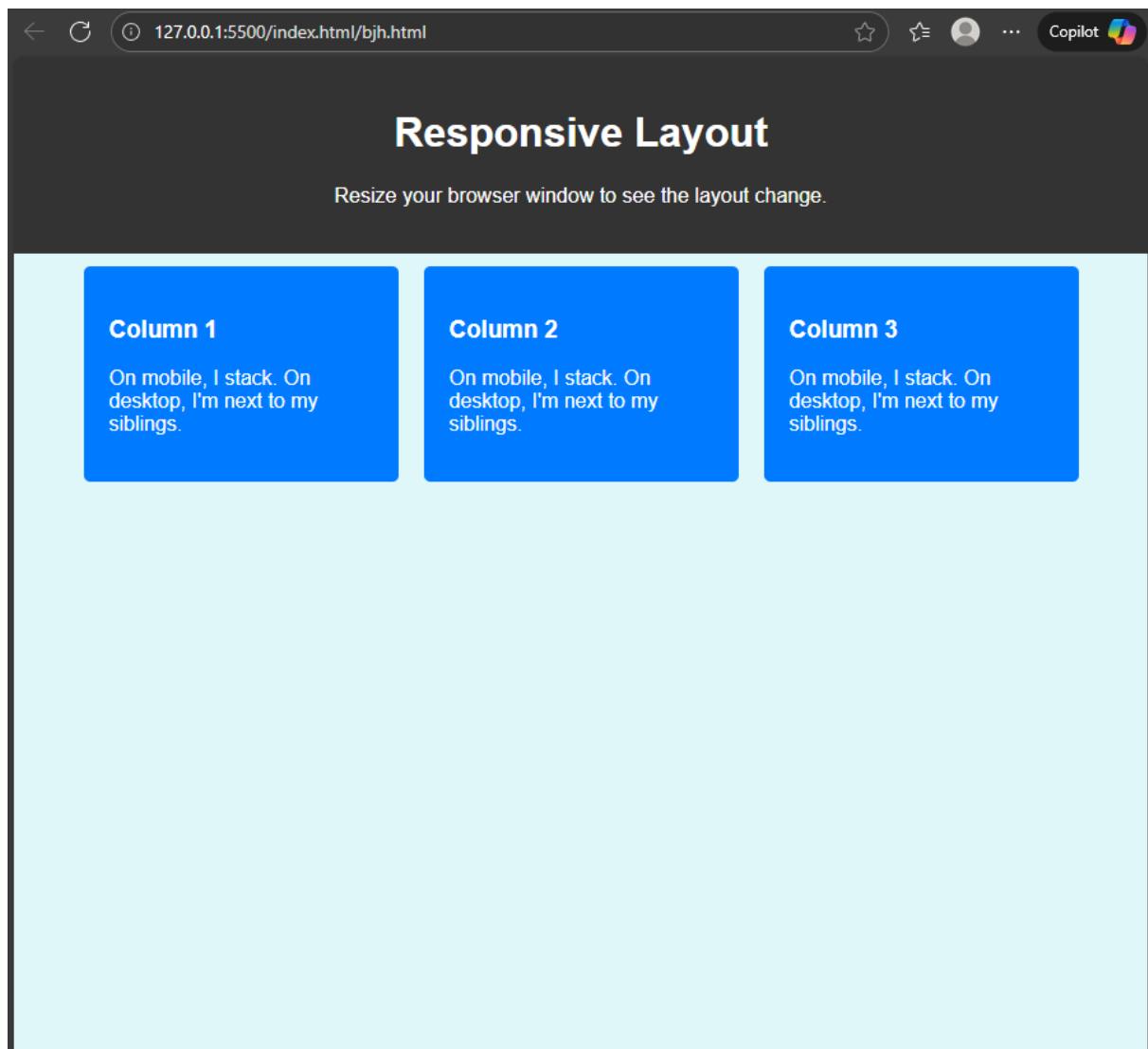
Experiment – 9

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

Code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 9: Responsive Design</title>
7      <style>
8          /* This is a MOBILE-FIRST design */
9
10     body {
11         font-family: Arial, sans-serif;
12         margin: 0;
13         padding: 0;
14         background-color: #f4f4f4;
15     }
16
17     .container {
18         width: 100%; /* Flexible unit */
19     }
20
21     header {
22         background: #333;
23         color: #FFF;
24         padding: 20px;
25         text-align: center;
26     }
27
28     /* On mobile (default), items stack vertically */
29     .grid-container {
30         display: block; /* Default for divs */
31     }
32
33     .grid-item {
34         background: #007BFF;
35         color: white;
36         padding: 20px;
37         margin: 10px;
38         border-radius: 5px;
39     }
40
41     /* --- This is the Media Query --- */
42
43     /* When the viewport is 768px WIDE or MORE, apply these styles */
44     @media (min-width: 768px) {
45         body {
46             background-color: #e0f7fa; /* Change background on larger screens */
47         }
48
49         /* On desktop, use flexbox to create a horizontal layout */
50         .grid-container {
51             display: flex;
52             width: 90%; /* Use 90% of the screen */
53             margin: auto;
54         }
55
56         .grid-item {
57             flex: 1; /* Makes all items share the space equally */
58         }
59     }
60     </style>
61 </head>
62 <body>
63
64     <header>
65         <h1>Responsive Layout</h1>
66         <p>Resize your browser window to see the layout change.</p>
67     </header>
68
69     <div class="container">
70         <div class="grid-container">
71             <div class="grid-item">
72                 <h3>Column 1</h3>
73                 <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
74             </div>
75             <div class="grid-item">
76                 <h3>Column 2</h3>
77                 <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
78             </div>
79             <div class="grid-item">
80                 <h3>Column 3</h3>
81                 <p>On mobile, I stack. On desktop, I'm next to my siblings.</p>
82             </div>
83         </div>
84     </div>
85
86 </body>
87 </html>
```

Output:



Experiment – 10

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

Code:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Experiment 10: Div and Span</title>
7      <style>
8          body {
9              font-family: Arial, sans-serif;
10             background-color: #f4f4f4;
11             line-height: 1.6;
12         }
13
14     /* --- STYLING <div> ELEMENTS (for Layout) --- */
15
16     /* This <div> acts as the main page wrapper */
17     .page-wrapper {
18         width: 80%;
19         max-width: 800px;
20         margin: 20px auto; /* 'auto' centers the block */
21         background-color: #ffffff;
22         border: 1px solid #ccc;
23         border-radius: 8px;
24         padding: 20px;
25     }
26
27     /* This <div> acts as a "header" section */
28     .header-box {
29         background-color: #2c3e50;
30         color: white;
31         padding: 15px;
32         text-align: center;
33         border-radius: 5px;
34     }
35
36     /* This <div> acts as a "content" section */
37     .content-box {
38         padding-top: 20px;
39     }
40
41     /* --- STYLING <span> ELEMENTS (for Inline Text) --- */
42
43     /* This <span> will be used for special labels */
44     .label {
45         font-family: 'Courier New', Courier, monospace;
46         background-color: #e0f7fa;
47         color: #00796b;
48         padding: 3px 6px;
49         border-radius: 4px;
50         font-weight: bold;
51     }
52
53
54     /* This <span> will be used for important alerts */
55     .alert {
56         color: #d9534f; /* Red */
57         font-weight: bold;
58         font-style: italic;
59         text-decoration: underline;
60     }
61
62     </style>
63 </head>
64 <body>
65
66     <div class="page-wrapper">
67
68         <div class="header-box">
69             <h1>Using &lt;div&gt; and &lt;span&gt;</h1>
70         </div>
71
72         <div class="content-box">
73             <p>
74                 A &lt;div&gt; element is a <span class="label">block-level</span> container.
75                 We've used divs to create the main page wrapper, the header, and this content area.
76                 Each &lt;div&gt; starts on a new line and creates a structural "box".
77             </p>
78
79             <p>
80                 A &lt;span&gt; element, on the other hand, is an <span class="label">inline</span> container.
81                 You use it to style a <span class="alert">small piece of text</span>
82                 inside a block element, just like this. Notice how the &lt;span&gt; elements
83                 don't break the flow of the text or start a new line.
84             </p>
85         </div>
86
87     </div>
88
89 </body>
90 </html>
```

Output:

Using <div> and

A <div> element is a **block-level** container. We've used divs to create the main page wrapper, the header, and this content area. Each <div> 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 > X
New folder > index.html > < fhh.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Exp 11: Arithmetic Operations</title>
5  </head>
6  <body>
7      <h2>Basic Arithmetic</h2>
8      <input type="number" id="num1" placeholder="Enter number 1">
9      <input type="number" id="num2" placeholder="Enter number 2">
10     <br><br>
11     <button onclick="calculate('+')">Add</button>
12     <button onclick="calculate('-')">Subtract</button>
13     <button onclick="calculate('*')">Multiply</button>
14     <button onclick="calculate('/')">Divide</button>
15
16     <h3 id="result">Result: </h3>
17
18     <script>
19         function calculate(operator) {
20             // Get values and convert to numbers
21             let n1 = parseFloat(document.getElementById('num1').value);
22             let n2 = parseFloat(document.getElementById('num2').value);
23             let res;
24
25             if (isNaN(n1) || isNaN(n2)) {
26                 alert("Please enter valid numbers");
27                 return;
28             }
29
30             switch(operator) {
31                 case '+': res = n1 + n2; break;
32                 case '-': res = n1 - n2; break;
33                 case '*': res = n1 * n2; break;
34                 case '/': res = n2 !== 0 ? n1 / n2 : "Cannot divide by zero"; break;
35             }
36
37             document.getElementById('result').innerText = "Result: " + res;
38         }
39     </script>
40 </body>
41 </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:

```
fh.html  X
New folder > index.html > fh.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h2>Loops and Conditions Demo</h2>
5      <div id="output"></div>
6
7      <script>
8          let outputDiv = document.getElementById("output");
9          let html = "";
10
11         // 1. Conditional (If/Else)
12         let hour = new Date().getHours();
13         html += "<strong>Conditional:</strong> ";
14         if (hour < 12) {
15             html += "Good Morning!<br>";
16         } else {
17             html += "Good Afternoon/Evening!<br>";
18         }
19
20         // 2. For Loop
21         html += "<br><strong>For Loop (0 to 4):</strong> ";
22         for (let i = 0; i < 5; i++) {
23             html += i + " ";
24         }
25
26         // 3. While Loop
27         html += "<br><strong>While Loop (Count down 5 to 1):</strong> ";
28         let j = 5;
29         while (j > 0) {
30             html += j + " ";
31             j--;
32         }
33
34         // 4. Do...While Loop
35         html += "<br><strong>Do...While (Runs at least once):</strong> ";
36         let k = 0;
37         do {
38             html += "Executed! ";
39             k++;
40         } while (k < 1);
41
42         outputDiv.innerHTML = html;
43     </script>
44 </body>
45 </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  
1  <!DOCTYPE html>  
2  <html>  
3  <body>  
4  |   <h2>Functions and Scope (Check Console)</h2>  
5  |   <p>Press F12 to see the console logs for scope demonstration.</p>  
6  |   <button onclick="demoScope()">Run Demo</button>  
7  
8  <script>  
9      function demoScope() {  
10         console.log("--- Scope Demo Started ---");  
11  
12         // Function Scope  
13         var functionScoped = "I am var (Function Scoped)";  
14  
15         if (true) {  
16             // Block Scope  
17             let blockScoped = "I am let (Block Scoped)";  
18             const constantVar = "I am const (Cannot Change)";  
19             var leakVar = "I am var (I leak out of blocks)";  
20  
21             console.log("Inside Block: " + blockScoped);  
22             console.log("Inside Block: " + constantVar);  
23         }  
24  
25         // Demonstrating accessibility  
26         console.log("Outside Block: " + functionScoped);  
27         console.log("Outside Block: " + leakVar); // Works because var ignores block scope  
28  
29         try {  
30             console.log(blockScoped); // This will throw error  
31         } catch (e) {  
32             console.log("Error: 'let' cannot be accessed outside its block.");  
33         }  
34     }  
35  </script>  
36 </body>  
37 </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:

```
<!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:

Form Validation

Password (min 5 chars):

Experiment-15

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

Code:

```
↳ fhh.html  X
New folder > index.html > ↳ fhh.html > html
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h2 id="header">Original Header</h2>
5      <p class="text-content">Original paragraph text.</p>
6      <button onclick="changeContent()">Manipulate DOM</button>
7
8      <script>
9          function changeContent() {
10              // Using getElementById
11              let header = document.getElementById("header");
12              header.innerHTML = "<em>Header Changed!</em>";
13              header.style.color = "blue";
14
15              // Using querySelector (selects first element with class .text-content)
16              let para = document.querySelector(".text-content");
17              para.innerText = "Paragraph text changed using querySelector.";
18          }
19      </script>
20  </body>
21  </html>
```

Output:

Original Header

Original paragraph text.

[Manipulate DOM](#)

Experiment-16

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

Code:

```
fhh.html •
New folder > index.html > fhh.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style>
5          .highlight {
6              background-color: yellow;
7              font-weight: bold;
8              padding: 10px;
9          }
10         #box {
11             width: 100px;
12             height: 100px;
13             background-color: lightblue;
14             margin-top: 10px;
15         }
16     </style>
17 </head>
18 <body>
19     <h2>Show/Hide & Toggle</h2>
20
21     <button onclick="toggleVisibility()">Show/Hide Box</button>
22     <button onclick="toggleClass()">Toggle CSS Class</button>
23
24     <div id="box">I am a box</div>
25     <p id="text">I am text for class toggling.</p>
26
27     <script>
28         function toggleVisibility() {
29             let box = document.getElementById("box");
30             // Check current display state
31             if (box.style.display === "none") {
32                 box.style.display = "block";
33             } else {
34                 box.style.display = "none";
35             }
36         }
37
38         function toggleClass() {
39             let text = document.getElementById("text");
40             // Toggle the 'highlight' class defined in CSS
41             text.classList.toggle("highlight");
42         }
43     </script>
44 </body>
45 </html>
```

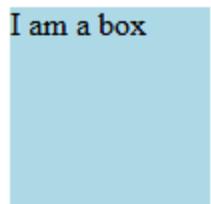
Output:

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  
1  <!DOCTYPE html>  
2  <html>  
3  <body>  
4    <h2>Event Listeners</h2>  
5    <div id="mouseBox" style="border:1px solid black; padding:20px; width:200px;">  
6      Hover over me!  
7    </div>  
8    <br>  
9    <input type="text" id="keyInput" placeholder="Type something here...">  
10   <p id="status">Status: Waiting...</p>  
11  
12  <script>  
13    let box = document.getElementById("mouseBox");  
14    let input = document.getElementById("keyInput");  
15    let status = document.getElementById("status");  
16  
17    // Mouse Event  
18    box.addEventListener("mouseover", function() {  
19      box.style.backgroundColor = "lightgreen";  
20      status.innerText = "Status: Mouse is inside!";  
21    });  
22  
23    box.addEventListener("mouseout", function() {  
24      box.style.backgroundColor = "white";  
25      status.innerText = "Status: Mouse is outside!";  
26    });  
27  
28    // Keyboard Event  
29    input.addEventListener("keyup", function(event) {  
30      status.innerText = "You pressed: " + event.key;  
31    });  
32  </script>  
33 </body>  
34 </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 X
New folder > index.html > fhh.html > html
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h2>Node Manipulation</h2>
5      <input type="text" id="newItem" placeholder="New Item Name">
6      <button onclick="addItem()">Add Item</button>
7      <button onclick="removeLastItem()">Remove Last Item</button>
8
9      <ul id="itemList">
10         <li>Item 1</li>
11         <li>Item 2</li>
12     </ul>
13
14     <script>
15         function addItem() {
16             let list = document.getElementById("itemList");
17             let val = document.getElementById("newItem").value;
18
19             if(val) {
20                 // Create new element
21                 let newLi = document.createElement("li");
22                 // Modify content
23                 newLi.innerText = val;
24                 // Append to parent
25                 list.appendChild(newLi);
26             }
27         }
28
29         function removeLastItem() {
30             let list = document.getElementById("itemList");
31             // Check if there are children to remove
32             if (list.lastElementChild) {
33                 list.removeChild(list.lastElementChild);
34             }
35         }
36     </script>
37 </body>
38 </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  X
New folder > index.html > fhh.html > html > body > script
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h2>Canvas Drawing</h2>
5      <canvas id="myCanvas" width="300" height="200" style="border:1px solid black;#000000;">
6          Your browser does not support the HTML canvas tag.
7      </canvas>
8
9      <script>
10         var c = document.getElementById("myCanvas");
11         var ctx = c.getContext("2d");
12
13         // Draw a Rectangle (Red)
14         ctx.fillStyle = "#FF0000";
15         ctx.fillRect(20, 20, 150, 100);
16
17         // Draw a Circle (Blue)
18         ctx.beginPath();
19         ctx.arc(240, 70, 40, 0, 2 * Math.PI);
20         ctx.fillStyle = "blue";
21         ctx.fill();
22         ctx.stroke();
23
24         // Draw a Line
25         ctx.moveTo(0, 0);
26         ctx.lineTo(300, 200);
27         ctx.stroke();
28     </script>
29 </body>
30 </html>
```

Output:

