Dom ASSIGNMENT

1.Basic Level

Task 1: Select and Modify an Element

Description: Select an h1 element by its id and change its text content to "Welcome to DOM Manipulation!".

```
//html.....                                    <!-- HTML -->
➡                                              <h1 id="welcome">Old Title</h1>
```

```
 // This is_ JavaScript
const h1Element = document.getElementById('welcome');          // Selects the h1 element
h1Element.textContent = "Welcome to DOM Manipulation!";        // Changes the text content
console.log(h1Element.textContent);                            // Logs the new text content
```

Output Explanation:

- The original text of the <h1> element is "Old Title".
- After running the JavaScript code, the textContent is updated to "Welcome to DOM Manipulation!".
- The console.log statement outputs this new text, confirming the change.

Output: Welcome to DOM Manipulation!

Task 2: Create and Append an Element

Description:

Create a new span element with the text "Hello, DOM!" and append it to an existing div with a specific id.

```
//html.....                                    <!-- HTML -->
                                               <div id="myDiv"></div>
```

```
                                                // This is_ JavaScript
const newSpan = document.createElement('span');          // Creates a new span element
newSpan.textContent = "Hello, DOM!";                     // Sets the text content of the span
const myDiv = document.getElementById('myDiv');          // Selects the existing div
myDiv.appendChild(newSpan);                              // Appends the new span to the div
console.log(myDiv.inner.HTML );                         // Logs the inner of the div
```

Output Explanation:

- A new element is created and its text is set to "Hello, DOM!".
- This span is then appended to the with id myDiv.
- The innerHTMLof the now contains the new , which is confirmed by the output.

Output: Hello, DOM!

Task 3: Modify an Element's Class

Description:

Select a p element by its class name and add an additional class highlight to it.

```
//html.....                    <!-- HTML -->
                               <p class="text">This is a paragraph.</p>
```

```
// This is_ JavaScript
const pElement = document.querySelector('.text');          // Selects the p element
pElement.classList.add('highlight');                       // Adds the highlight class to the p element
console.log(pElement.className);                           // Logs the class names of the p element
```

Output Explanation:

- The <p> element initially has the class text.
- The classList.add method adds the highlight class to this element.
- The output shows the combined class names of the element, indicating the change.

Output: text highlight

Task 4: Change Element Styles

Description:

Select a div element by its id and change its background color to lightblue and its text color to darkblue.

```
//html.....                    <!-- HTML -->
                               <div id="myStyleDiv">Styled Div</div>
```

```
// This is_ JavaScript
const styleDiv = document.getElementById('myStyleDiv');    // Selects the div element
styleDiv.style.backgroundColor = 'lightblue';              // Sets the background color
styleDiv.style.color = 'darkblue';                         // Sets the text color
console.log(styleDiv.style.    Text);                      // Logs the      text of the div
```

Output Explanation:

- The backgroundColor and color properties are modified to change the visual appearance of the .
- The Text output reflects these changes, showing the applied styles.

Output: background-color: lightblue; color: darkblue;

Task 5: Create and Append a Text Node

Description:

Create a new text node with the content "This is a text node" and append it to a div with a specific class name.

```
//html.....                    <!-- HTML -->
                               <div class="textContainer"></div>
```

```
// This is_ JavaScript
const textNode = document.createTextNode("This is a text node");        // Creates a new text node
const textContainer = document.querySelector('.textContainer');         // Selects the div
textContainer.appendChild(textNode);                                    // Appends the text node to the div
console.log(textContainer.innerHTML );                                  // Logs the .inner HTML of the div
```

Output Explanation:

- A new text node containing "This is a text node" is created and appended to the div.
- The innerHTMLof the div now contains this text, which is confirmed by the output.

Output: This is a text node

Task 6: Modify an Image's Source

Description: Select an image element by its id and change its src attribute to "new-image.jpg".

```
//html.....                    <!-- HTML -->
                               <img id="myImage" src="old-image.jpg" />
```

```
// This is_ JavaScript
const imageElement = document.getElementById('myImage');        // Selects the image element
imageElement.src = "new-image.jpg";                             // Changes the src attribute
console.log(imageElement.src);                                  // Logs the new src attribute
```

Output Explanation:

- The src attribute of the <img> element is changed from "old-image.jpg" to "new-image.jpg".
- The output confirms the new src value, indicating the image source has been updated.

Output: new-image.jpg

Task 7: Change Inner HTML

Description:

Select a div by its id and set its innerHTMLto " New paragraph inside div. ".

```
//html.....                    <!-- HTML -->
                               <div id="myInnerDiv"></div>
```

```
// This is_ JavaScript
const innerDiv = document.getElementById('myInnerDiv');          // Selects the div
innerDiv.innerHTML  = "<p>New paragraph inside div.</p>";         // Sets the inner HTML
console.log(innerDiv.innerHTML );                                 // Logs the .inner HTML of the div
```

Output Explanation:

- The innerHTMLproperty of the div is directly set to a new string that includes a paragraph.
- The output reflects this change, showing the new content inside the div.

Output: New paragraph inside div.

Task 8: Retrieve and Display Attribute Value

Description:

Select a link element by its id, retrieve its href attribute value, and log it to the console.

```
//html.....                    <!-- HTML -->
                               <a id="myLink"
                               href="https://example.com">Example</a>
```

```
// This is_ JavaScript
const linkElement = document.getElementById('myLink');           // Selects the link element
console.log(linkElement.href);                                   // Logs the href attribute value
```

Output Explanation:

- The href attribute of the link is accessed and its value is logged.
- The output displays the complete URL as stored in the href attribute.

Output: https://example.com/

Task 9: Append a List Item

Description:

Create a new li element with the text "New List Item" and append it to an unordered list (ul) with a specific class name.

```
//html.....                    <!-- HTML -->
                               <ul class="myList"></ul>
```

```
 // This is_ JavaScript
const newListItem = document.createElement('li');                    // Creates a new li element
newListItem.textContent = "New List Item";                           // Sets the text content
const myList = document.querySelector('.myList');                    // Selects the ul element
myList.appendChild(newListItem);                                     // Appends the new li to the ul
console.log(myList.innerHTML );                                       // Logs the .inner HTML of the ul
```

Output Explanation:

- A new <li> element is created and its text is set.
- This list item is appended to the <ul>, updating its innerHTMLto include the new item, confirmed by the output.

Output: New List Item

Task 10: Set Element's Title Attribute

Description:

Select a button element by its id and set its title attribute to "Click me!".

```
//html.....                    <!-- HTML -->
                               <button id="myButton">Button</button>
```

```
 // This is_ JavaScript
const buttonElement = document.getElementById('myButton');           // Selects the button element
buttonElement.title = "Click me!";                                    // Sets the title attribute
console.log(buttonElement.title);                                    // Logs the title attribute value
```

Output Explanation:

- The title attribute of the button is updated with the new text "Click me!".
- The output confirms this change, showing the updated title attribute.

Output: Click me!

2.Medium Level

Task 1: Traverse to Child Elements

Description: Select a ul element by its class name, then find and log the text content of all its child li elements.

```
//html……..                    <!-- HTML -->
                              <ul class="myList">
                                <li>Item 1</li>
                                <li>Item 2</li>
                                <li>Item 3</li>
                              </ul>
```

```
 // This is_ JavaScript
const ulElement = document.querySelector('.myList');                    // Selects the ul element
const liElements = ulElement.getElementsByTagName('li');                // Retrieves all child li elements
for (let li of liElements) {
   console.log(li.textContent);                                         // Logs the text content of each li
}
```

Output Explanation:

- The querySelector method selects the <ul> element with the class myList.
- getElementsByTagName('li') retrieves all child <li> elements.
- The loop iterates through each <li>, logging their text content.

Output:

Item 1

Item 2

Item 3

Task 2: Insert an Element After Another

Description:

Create a new p element with the text "Inserted After" and insert it after an existing paragraph with a specific id.

```
//html……..                    <!-- HTML -->
                              <p id="existingParagraph">This is an existing
                              paragraph.</p>
```

```
 // This is_ JavaScript
const newParagraph = document.createElement('p');                       // Creates a new p element
newParagraph.textContent = "Inserted After";                            // Sets the text content
const existingParagraph = document.getElementById('existingParagraph'); // Selects the existing p element
existingParagraph.insertAdjacentElement('afterend', newParagraph); // Inserts the new p after the existing one
console.log(existingParagraph.nextSibling.textContent); // Logs the text content of the newly inserted paragraph
```

Output Explanation:

- A new <p> element is created and its text is set to "Inserted After".
- The insertAdjacentElement method inserts the new paragraph immediately after the existing one.
- The nextSibling property retrieves the newly inserted <p>, and its content is logged.

Output:

Inserted After

Task 3: Replace an Existing Element

Description: Create a new h2 element with the text "New Heading" and replace an existing h2 element with this one.

```
//html.....                    <!-- HTML -->
                               <h2>Old Heading</h2>

 // This is_ JavaScript
const newHeading = document.createElement('h2');                    // Creates a new h2 element
newHeading.textContent = "New Heading";                             // Sets the text content
const oldHeading = document.querySelector('h2');                    // Selects the existing h2 element
oldHeading.parentNode.replaceChild(newHeading, oldHeading); // Replaces the old h2 with the new one
console.log(newHeading.textContent);                                // Logs the text content of the new heading
```

Output Explanation:

- A new <h2> element is created with the text "New Heading".
- The replaceChild method replaces the old <h2> with the new one in the DOM.
- The content of the new heading is logged to confirm the replacement.

Output: New Heading

Task 4: Traverse to Sibling Elements

Description:

Select an h3 element by its id, traverse to its next sibling element, and change its text content to "Sibling Updated".

```
//html.....                    <!-- HTML -->
                               <h3 id="myH3">Heading 3</h3>
                               <p>Original Sibling Paragraph</p>
```

```
// This is_ JavaScript
const h3Element = document.getElementById('myH3');              // Selects the h3 element
const siblingElement = h3Element.nextElementSibling;            // Selects the next sibling element
siblingElement.textContent = "Sibling Updated";                 // Changes the text content of the sibling
console.log(siblingElement.textContent);                        // Logs the updated sibling text
```

Output Explanation:

- The nextElementSibling property retrieves the next sibling of the <h3>, which is the <p>.
- The text content of the sibling <p> is updated to "Sibling Updated".
- The updated content is logged to confirm the change.

Output: Sibling Updated

Task 5: Modify Multiple Attributes

Description: Select an input element by its id and set its type attribute to "password", placeholder to "Enter your password", and name to "user-password".

```
//html.....                    <!-- HTML -->
                               <input id="myInput" type="text" />
```

```
// This is_ JavaScript
const inputElement = document.getElementById('myInput');          // Selects the input element
inputElement.type = "password";                                   // Sets the type attribute
inputElement.placeholder = "Enter your password";                 // Sets the placeholder
inputElement.name = "user-password";                              // Sets the name attribute
console.log(inputElement outerHTML   );          // Logs the  outerHTML   of the input element
```

Output Explanation:

- The input element's attributes are modified to change its type, placeholder, and name.
- The outerHTML property shows the complete updated html of the input element, confirming the changes.

Output:<input id="myInput" type="password" placeholder="Enter your password" name="user-password">

Task 6: Move an Element to a New Parent

Description: Select an existing div by its class name and move it to a new parent section element.

```
//html.....                    <!-- HTML -->
                               <div class="myDiv">This is a div to move</div>
                               <section id="newSection"></section>
```

```
// This is_ JavaScript
const divElement = document.querySelector('.myDiv');                    // Selects the div element
const newSection = document.getElementById('newSection');               // Selects the section element
newSection.appendChild(divElement);                                     // Moves the div to the new section
console.log(newSection.innerHTML );                                     // Logs the .inner HTML of the new section
```

Output Explanation:

- The selected is appended to the <section>, effectively moving it.
- The innerHTMLof the section reflects this change, showing the moved .

Output: This is a div to move

Task 7: Add a New Element After a Sibling

Description:

After the last p element inside the div, create and insert a new p element with the text "This is an additional paragraph".

```
//html……                                    <!-- HTML -->
                                            <div class="content">
                                              <p>First paragraph</p>
                                              <p>Second paragraph</p>
                                            </div>
```

```
// This is_ JavaScript
const divContent = document.querySelector('.content');                  // Selects the div
const newParagraph = document.createElement('p');                       // Creates a new p element
newParagraph.textContent = "This is an additional paragraph";           // Sets the text content
divContent.appendChild(newParagraph);                                   // Inserts the new p as the last child
console.log(divContent.innerHTML );                                     // Logs the .inner HTML of the div
```

Output Explanation:

- A new paragraph is created and added as the last child of the .
- The innerHTMLof the div now includes this new , confirming the addition.

Output:

First paragraph
Second paragraph
This is an additional paragraph

Task 8: Remove a Specific Child Element

Description: Select a div by its id, find a specific p child element by its class name, and remove it from the div.

```
//html.....                    <!-- HTML -->
                               <div id="myDiv">
                                 <p class="removeMe">Remove this
                               paragraph</p>
                                 <p>Keep this paragraph</p>
                               </div>
```

```
 // This is_ JavaScript
const myDiv = document.getElementById('myDiv');                    // Selects the div element
const pToRemove = myDiv.querySelector('.removeMe');                // Selects the specific p element
myDiv.removeChild(pToRemove);                                      // Removes the specified p from the div
console.log(myDiv.innerHTML );                                     // Logs the .inner HTML of the div
```

Output Explanation:

- The specified paragraph with class removeMe is selected and removed from the .
- The updated innerHTMLof the div confirms that the paragraph has been successfully removed.

Output: Keep this paragraph

Task 9: Modify Multiple Styles

Description:

Select a div by its class name and change its font size to 20px, padding to 10px, and border to 2px solid black.

```
//html.....                    <!-- HTML -->
                               <div class="styledDiv">Stylish Div</div>
```

```
 // This is_ JavaScript
const styledDiv = document.querySelector('.styledDiv');            // Selects the div element
styledDiv.style.fontSize = '20px';                                 // Changes the font size
styledDiv.style.padding = '10px';                                  // Sets the padding
styledDiv.style.border = '2px solid black';                        // Sets the border
console.log(styledDiv.style.    Text);                             // Logs the      text of the div
```

Output Explanation:

- The properties of the are modified to update its styles.
- The Text property provides a string representation of the applied styles, confirming the changes.

Output: font-size: 20px; padding: 10px; border: 2px solid black;

Task 10: Insert Multiple Elements

Description:

Create two new li elements with text "Item 1" and "Item 2", and insert them into an existing ul with a specific class name.

```
//html.....                    <!-- HTML -->
                               <ul class="myList"></ul>
```

```
// This is_ JavaScript
const ulElement = document.querySelector('.myList');          // Selects the ul element
const li1 = document.createElement('li');                      // Creates the first li element
li1.textContent = "Item 1";                                    // Sets its text
const li2 = document.createElement('li');                      // Creates the second li element
li2.textContent = "Item 2";                                    // Sets its text
ulElement.appendChild(li1);                                    // Inserts the first li into the ul
ulElement.appendChild(li2);                                    // Inserts the second li into the ul
console.log(ulElement.innerHTML );                             // Logs the .inner HTML of the ul
```

Output Explanation:

- Two new <li> elements are created and their text content is set.
- Both items are appended to the existing <ul>, expanding its content.
- The updated innerHTMLof the <ul> reflects the newly added items.

Output:

```
<li>Item 1</li>
<li>Item 2</li>
```

3.Advanced Level

Task 1: Create a Complex Nested Structure

Description: Create a div with a class of "container". Inside it, create an h3 element with text "Section Header", a p element with text "Section Content", and append them all to the div. Finally, append this div to the body.

```
//html……..                                    <!-- No //html  needed, we will create this using JavaScript -->

 // This is_ JavaScript
const container = document.createElement('div');                    // Create a new div
container.className = 'container';                                   // Set the class name to "container"

const header = document.createElement('h3');                         // Create an h3 element
header.textContent = 'Section Header';                               // Set the text content

const paragraph = document.createElement('p');                       // Create a p element
paragraph.textContent = 'Section Content';                           // Set the text content

                                                  // Append the header and paragraph to the container
container.appendChild(header);
container.appendChild(paragraph);

                                                  // Append the container to the body
document.body.appendChild(container);

                                                  // Log the container's  outerHTML   to the console
console.log(container outerHTML   );
```

Output Explanation:

- A element is created and given the class container.
- An <h3> and a <p> element are created and their text content is set.
- The header and paragraph are appended to the container, which is then appended to the document body.
- The outerHTML   property logs the complete structure of the container to the console.

Output:

```
<div class="container">
  <h3>Section Header</h3>
  <p>Section Content</p>
</div>
```

Task 2: Remove All Child Elements

Description: Select a div by its id, remove all its child elements, and log a message to the console indicating that the div is now empty.

```
//html.....                                    <!-- HTML -->
                                               <div id="myDiv">
                                                 <p>Paragraph 1</p>
                                                 <p>Paragraph 2</p>
                                               </div>
```

```
 // This is_ JavaScript
const myDiv = document.getElementById('myDiv');                    // Selects the div
while (myDiv.firstChild) {                                         // While the div has children
   myDiv.removeChild(myDiv.firstChild);                            // Remove the first child
}
console.log('The div is now empty.');                              // Log the message
```

Output Explanation:

- The while loop checks if the div has any child elements. If it does, it removes the first child repeatedly until no children remain.
- A message is logged to confirm that the div is empty.

Output:

The div is now empty.

Task 3: Modify Multiple Elements in a Loop

Description: Select all li elements inside a ul by its class name and change the text content of each li to "Modified Item".

```
//html.....                                    <!-- HTML -->
                                               <ul class="myList">
                                                 <li>Item 1</li>
                                                 <li>Item 2</li>
                                                 <li>Item 3</li>
                                               </ul>
```

```
// This is_ JavaScript
const listItems = document.querySelectorAll('.myList li');                    // Selects all li elements
listItems.forEach(item => {                                                   // Loop through each li
   item.textContent = 'Modified Item';                                        // Change the text content
});
console.log(document.querySelector('.myList').innerHTML );       // Log the updated .inner HTML of the ul
```

Output Explanation:

- All <li> elements are selected using querySelectorAll.
- A forEach loop modifies the text content of each <li> to "Modified Item".
- The updated .inner HTML of the <ul> is logged to show the changes.

Output:

```
<li>Modified Item</li>
<li>Modified Item</li>
<li>Modified Item</li>
```

Task 4: Change Attributes of an Image

Description: Find the first img element within the article and change its alt attribute to "New Image Description" and set its width attribute to 300.

```
//html……..                                <!-- HTML -->
                                          <article>
                                            <img src="image.jpg" alt="Old Image
                                          Description">
                                          </article>
```

```
// This is_ JavaScript
const imgElement = document.querySelector('article img');                  // Selects the first img in the article
imgElement.alt = 'New Image Description';                                  // Change the alt attribute
imgElement.width = 300;                                                     // Set the width attribute
console.log(imgElement outerHTML   );                              // Log the updated  outerHTML   of the img
```

Output Explanation:

- The first <img> element in the <article> is selected.
- Its alt attribute is updated, and the width attribute is set to 300 pixels.
- The updated outerHTML   of the image is logged.

Output: <img src="image.jpg" alt="New Image Description" width="300">

Task 5: Add Multiple Attributes Dynamically

Description:

Select an existing img element by its class name and add three new attributes: data-id, alt, and title.

```
//html……                    <!-- HTML -->
                            <img class="myImage" src="image.jpg">

// This is_ JavaScript
const existingImg = document.querySelector('.myImage');          // Selects the img element
existingImg.setAttribute('data-id', '123');                      // Adds data-id attribute
existingImg.alt = 'Image Description';                           // Sets the alt attribute
existingImg.title = 'Image Title';                               // Sets the title attribute
console.log(existingImg outerHTML   );                           // Log the updated  outerHTML   of the img
```

Output Explanation:

- The existing <img> element is selected.
- Three new attributes (data-id, alt, and title) are added to the image element using setAttribute and property assignment.
- The updated outerHTML   is logged to show the changes.

Output:

<img class="myImage" src="image.jpg" data-id="123" alt="Image Description" title="Image Title">

Task 6: Dynamic Element Creation with Loop

Description: Create a ul element, then use a loop to create and append 5 li elements with the text "List Item 1", "List Item 2", etc., and append the ul to a div with a specific id.

```
//html……                    <!-- HTML -->
                            <div id="listContainer"></div>

 // This is_ JavaScript
const ulElement = document.createElement('ul');                  // Create a new ul element

for (let i = 1; i <= 5; i++) {                                   // Loop to create 5 li elements
   const liElement = document.createElement('li');               // Create an li element
   liElement.textContent = `List Item ${i}`;                     // Set its text content
   ulElement.appendChild(liElement);                             // Append the li to the ul
}

const container = document.getElementById('listContainer');     // Selects the div
container.appendChild(ulElement);                                // Append the ul to the div
console.log(container.innerHTML );                               // Log the .inner HTML of the div
```

Output Explanation:

- A new <ul> is created, and a loop generates five <li> elements with incrementing text.
- Each <li> is appended to the <ul>, which is then appended to the specified container div.
- The .inner HTML of the container div is logged to show the resulting list.

Output:

```
<ul>
  <li>List Item 1</li>
  <li>List Item 2</li>
  <li>List Item 3</li>
  <li>List Item 4</li>
  <li>List Item 5</li>
</ul>
```

Task 7: Replace All Occurrences of a Tag

Description: Select all p elements in the document, replace them with div elements that contain the same text content as the original p elements.

```
//html.....                                    <!-- HTML -->
                                               <p>First paragraph.</p>
                                               <p>Second paragraph.</p>
```

```
 // This is_ JavaScript
const paragraphs = document.querySelectorAll('p');                    // Selects all p elements
paragraphs.forEach(p => {                                              // Loop through each p element
   const div = document.createElement('div');                           // Create a new div
   div.textContent = p.textContent;                                     // Set the text content from the p
   p.parentNode.replaceChild(div, p);                                   // Replace the p with the div
});
console.log(document.body.innerHTML );                    // Log the updated .inner HTML of the body
```

Output Explanation:

- All <p> elements are selected, and each is replaced by a new containing the same text.
- The replaceChild method performs the replacement in the DOM.
- The updated .inner HTML of the body is logged to reflect the changes.

Output:

```
<div>First paragraph.</div>
<div>Second paragraph.</div>
```

Task 8: Complex Traversal and Manipulation

Description: Select a div with a specific class name, traverse to its parent, find its last child element, and change its background color to yellow.

```
//html.....                                    <!-- HTML -->
                                               <div class="myDiv">
                                                 <p>Paragraph 1</p>
                                                 <p>Paragraph 2</p>
                                               </div>
```

```
 // This is_ JavaScript
const myDiv = document.querySelector('.myDiv');                  // Selects the div
const parentDiv = myDiv.parentNode;                               // Get the parent of the div
const lastChild = parentDiv.lastChild;                            // Get the last child of the parent

if (lastChild.nodeType === Node.ELEMENT_NODE) {                   // Check if it's an element
   lastChild.style.backgroundColor = 'yellow';                    // Change background color
}
console.log(lastChild outerHTML   );                              // Log the  outerHTML   of the last child
```

Output Explanation:

- The div is selected, and its parent is accessed.
- The last child of the parent is retrieved, and if it's an element node, its background color is changed to yellow.
- The outerHTML   of the last child is logged to confirm the change.

Output:

```
<p style="background-color: yellow;">Paragraph 2</p>
```

Task 9: Merge and Append Multiple Elements

Description: Select two existing ul elements by their class names, merge all their li elements into a single ul, and append this new ul to a div with a specific id.

```
//html.....                                    <!-- HTML -->
                                               <ul class="list1">
                                                 <li>Item A</li>
                                                 <li>Item B</li>
                                               </ul>
                                               <ul class="list2">
                                                 <li>Item C</li>
                                                 <li>Item D</li>
                                               </ul>
                                               <div id="mergedListContainer"></div>
```

```
 // This is_ JavaScript
const list1 = document.querySelector('.list1');                    // Select the first ul

const list2 = document.querySelector('.list2');                    // Select the second ul

const mergedList = document.createElement('ul');                   // Create a new ul for merging

                                                                   // Move li elements from the first ul to the merged list

while (list1.firstChild) {
   mergedList.appendChild(list1.firstChild);
}
                                                                   // Move li elements from the second ul to the merged list
while (list2.firstChild) {
   mergedList.appendChild(list2.firstChild);
}
                                                                   // Append the merged list to the container
const container = document.getElementById('mergedListContainer');

container.appendChild(mergedList);

console.log(container.innerHTML );                                 // Log the .inner HTML of the container
```

Output Explanation:

- The two <ul> elements are selected, and a new <ul> is created to hold the merged items.
- All <li> elements from both lists are moved into the new merged list.
- The merged list is appended to the specified container div, and its .inner HTML is logged.

Output:

```
<ul>
  <li>Item A</li>
  <li>Item B</li>
  <li>Item C</li>
  <li>Item D</li>
</ul>
```

Task 10: Create and Manipulate a Document Fragment

Description:

Create a document fragment, then create and append three div elements with text

"Fragment Div 1",

"Fragment Div 2", and

"Fragment Div 3" to this fragment. Finally, append the entire fragment to the body.

```
//html……                                    <!-- No HTML needed, we will create this using JavaScript -->

 // This is_ JavaScript
const fragment = document.createDocumentFragment();          // Create a document fragment

for (let i = 1; i <= 3; i++) {                                // Loop to create 3 div elements
   const div = document.createElement('div');                 // Create a div
   div.textContent = `Fragment Div ${i}`;                     // Set its text content
   fragment.appendChild(div);                                 // Append the div to the fragment
}

                                                              // Append the entire fragment to the body
document.body.appendChild(fragment);
console.log(document.body.innerHTML );                        // Log the .inner HTML of the body
```

Output Explanation:

- A document fragment is created to hold the new elements without affecting the DOM immediately.
- Three elements are created and appended to the fragment.
- The fragment is appended to the document body, efficiently updating the DOM in one go.
- The .inner HTML of the body is logged to show the newly added elements.

Output:

```
<div>Fragment Div 1</div>
<div>Fragment Div 2</div>
<div>Fragment Div 3</div>
```