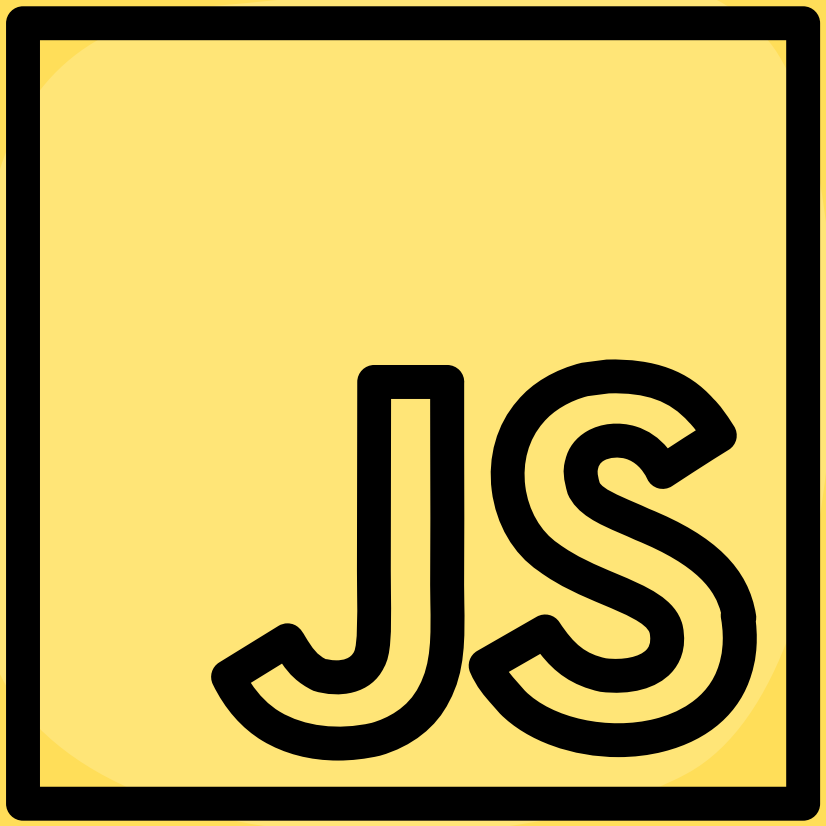

# DOM

## Insert, Replace, Remove

#

# Create & Insert

innerHTML: Inserts HTML but removes existing content and creates new elements, which is not optimal for performance.

```
const root = document.getElementById('root-el');
root.innerHTML = `
    <div>
        <h2>Welcome!</h2>
        <p>This is all created & rendered automatically!
</p></div>
`;
// Output:
// <div id="root-el">
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
// </div>
```

#

insertAdjacentHTML: Inserts HTML at a specific position without removing existing content.

```
const root = document.getElementById('root-el');
root.insertAdjacentHTML('afterbegin', `
    <div>
        <h2>Welcome Again!</h2>
        <p>This is also created & rendered automatically!</p>
    </div>
`);
// Output:
// <div id="root-el">
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
// </div>
```

#

Positions for insertAdjacentHTML:

- beforebegin: Before the element itself.
- afterbegin: Just inside the element, before its first child.
- beforeend: Just inside the element, after its last child.
- afterend: After the element itself.

# Creating & Inserting DOM Objects Manually

#

createElement and append: Creates and appends an element at the end of the parent element.

```
const someParagraph = document.createElement('p');
someParagraph.textContent = 'New Paragraph';
const root = document.getElementById('root-el');
root.append(someParagraph);
// Output:
// <div id="root-el">
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
// </div>
```

#

**createElement and appendChild**: Creates and appends an element at the end of the parent element. (Better browser support than append)

```
const anotherParagraph = document.createElement('p');
anotherParagraph.textContent = 'Another New Paragraph';
const root = document.getElementById('root-el');
root.appendChild(anotherParagraph);
// Output:
// <div id="root-el">
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
// </div>
```

# Insertion Methods: #

append: Appends at the end.

```
const newDiv = document.createElement('div');
newDiv.textContent = 'Appended Div';
root.append(newDiv);
// Output:
// <div id="root-el">
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
// </div>
```

# prepend: Inserts at the beginning.

```
const prependedDiv = document.createElement('div');
prependedDiv.textContent = 'Prepended Div';
root.prepend(prependedDiv);
// Output:
// <div id="root-el">
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
// </div>
```

## before and after: Inserts before or after a reference element. (Not supported in Safari)

```
const referenceElement = document.querySelector('#root-el div'); // First div
inside #root-el
const beforeDiv = document.createElement('div');
beforeDiv.textContent = 'Before Div';
referenceElement.before(beforeDiv);
// Output:
// <div id="root-el">
//   <div>Before Div</div>
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
// </div>
```

```
const afterDiv = document.createElement('div');
afterDiv.textContent = 'After Div';
referenceElement.after(afterDiv);
// Output:
// <div id="root-el">
//   <div>Before Div</div>
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>After Div</div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
// </div>
```

# insertAdjacentElement: Inserts an element at a specific position.

```
const newElement = document.createElement('div');
newElement.textContent = 'New Element';
root.insertAdjacentElement('beforeend', newElement);
// Output:
// <div id="root-el">
//   <div>Before Div</div>
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>After Div</div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
//   <div>New Element</div>
// </div>
```

#

**cloneNode:** Creates a copy of a node. If true is passed, it performs a deep clone, including all child nodes.

```
// Original element
const originalElement = document.getElementById('original-el');
// <div id="original-el">Original Content</div>

// Clone the element
const clonedElement = originalElement.cloneNode(true); // Deep clone

// Modify cloned element (optional)
clonedElement.id = 'cloned-el';
clonedElement.textContent = 'Cloned Content';

// Append cloned element to the DOM
document.body.appendChild(clonedElement);

// Output:
// <div id="original-el">Original Content</div>
// <div id="cloned-el">Cloned Content</div>
```

# Replace

## replaceWith: Replaces an element with a new one

```
const newElement = document.createElement('div');
newElement.textContent = 'Replaced Element';
const oldElement = document.querySelector('#root-el div');
oldElement.replaceWith(newElement);
// Output:
// <div id="root-el">
//   <div>Replaced Element</div>
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>After Div</div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
//   <div>New Element</div>
// </div>
```

# replaceChild: Replaces a child element with a new one

```
const newElement = document.createElement('div');
newElement.textContent = 'Another Replaced Element';
const parentElement = document.getElementById('root-el');
const oldElement = parentElement.querySelector('div');
parentElement.replaceChild(newElement, oldElement);
// Output:
// <div id="root-el">
//   <div>Another Replaced Element</div>
//   <div>Prepended Div</div>
//   <div>
//     <h2>Welcome Again!</h2>
//     <p>This is also created & rendered automatically!</p>
//   </div>
//   <div>After Div</div>
//   <div>
//     <h2>Welcome!</h2>
//     <p>This is all created & rendered automatically!</p>
//   </div>
//   <p>New Paragraph</p>
//   <p>Another New Paragraph</p>
//   <div>Appended Div</div>
//   <div>New Element</div>
// </div>
```

# Remove #

innerHTML = " ": Clears all content inside the element, removing all child nodes.

```
root.innerHTML = '';
// Output:
// <div id="root-el"></div>
```

remove: Removes the element from the DOM.

```
const elementToRemove = document.getElementById('remove-el');
elementToRemove.remove();
// Output: The element with id 'remove-el' is removed from the DOM.
```

**removeChild:** Removes a specific child element.

```
const parentElement = document.getElementById('parent-el');
const elementToRemove = document.getElementById('remove-el');
parentElement.removeChild(elementToRemove);
// Output: The element with id 'remove-el' is removed from the DOM.
```

THANK
YOU!