# Element: insertAdjacentHTML() method


- Syntax
  - Parameters
  - Return value
  - Exceptions
- Description
  - Security considerations
- Examples
  - Inserting HTML
    - HTML
    - CSS
    - JavaScript
    - Result
- Specifications
- Browser compatibility
- See also


> **WARNING:** This method parses its input as HTML or XML, writing the result into the DOM. APIs like this are known as injection sinks, and are potentially a vector for cross-site scripting (XSS) attacks, if the input originally came from an attacker.

> You can reduce the risk by assigning TrustedHTML objects instead of strings, and enforcing trusted types using the `require-trusted-types-for` CSP directive. This ensures that the input is passed through a transformation function, which has the chance to sanitize the input to remove potentially dangerous markup, such as script elements and event handler attributes.

The **`insertAdjacentHTML()`** method of the Element interface parses the specified input as HTML or XML and inserts the resulting nodes into the DOM tree at a specified position.

# Syntax

```
insertAdjacentHTML(position, input)
```

# Parameters

- `position`
  - : A string representing the position relative to the element. Must be one of the following strings:
    - `"beforebegin"`
      - : Before the element. Only valid if the element is in the DOM tree and has a parent element.
    - `"afterbegin"`
      - : Just inside the element, before its first child.
    - `"beforeend"`
      - : Just inside the element, after its last child.
    - `"afterend"`
      - : After the element. Only valid if the element is in the DOM tree and has a parent element.
- `input`
  - : A TrustedHTML instance or string defining the HTML or XML to be parsed.

# Return value

None (undefined).

# Exceptions

This method may raise a DOMException of one of the following types:

- `NoModificationAllowedError` DOMException
  - : Thrown if `position` is `"beforebegin"` or `"afterend"` and the element either does not have a parent or its parent is the `Document` object.
- `SyntaxError` DOMException
  - : Thrown if:
    - `position` is not one of the four listed values.
    - The input is XML that is not well-formed.
- TypeError
  - : Thrown if the property is set to a string when Trusted Types are enforced by a CSP and no default policy is defined.

# Description

The `insertAdjacentHTML()` method does not reparse the element it is being used on, and thus it does not corrupt the existing elements inside that element. This avoids the extra step of serialization, making it much faster than direct innerHTML manipulation.

Where `<p>` is the element, we can visualize the possible positions for the inserted content "foo" as follows:

```
<!-- beforebegin -->
<p>
  <!-- afterbegin -->
  foo
  <!-- beforeend -->
</p>
<!-- afterend -->
```

The method does not include any special handling for template elements. In most cases developers should use `insertAdjacentHTML()` on the template's content property instead of directly manipulating the child nodes of a template element.

# Security considerations

The method does not perform any sanitization to remove XSS-unsafe elements such as script, or event handler content attributes.

When inserting HTML into a page using `insertAdjacentHTML()`, you should pass TrustedHTML objects instead of strings, and enforce trusted types using the `require-trusted-types-for` CSP directive. This ensures that the input is passed through a transformation function, which has the chance to sanitize the input to remove potentially dangerous markup before it is injected.

The Element.insertAdjacentText() method or Node.textContent should be used when you know that the user-provided content should be plain text. This inserts the input as raw text instead of parsing it as HTML.

# Examples

## Inserting HTML

This example demonstrates the four insertion positions. All inserted text is bold, while text inserted inside the element is further styled as red monotype (code).

### HTML

```
<select id="position">
  <option>beforebegin</option>
  <option>afterbegin</option>
  <option>beforeend</option>
  <option>afterend</option>
</select>

<button id="insert">Insert HTML</button>
<button id="reset">Reset</button>

<p>
  Some text, with a <code id="subject">code-formatted element</code> inside it.
</p>
```

### CSS

```
code {
  color: red;
}
```

# JavaScript

Trusted types are not yet supported on all browsers, so first we define the trusted types tinyfill. This acts as a transparent replacement for the Trusted Types JavaScript API:

```
if (typeof trustedTypes === "undefined")
  trustedTypes = { createPolicy: (n, rules) => rules };
```

Next we define a policy named `some-content-policy` to create TrustedHTML objects from the input (we should also enforce the `some-content-policy` using CSP). The code implements no-op policy in order to allow this example to work without a third-party dependency. Your own application code should use a third-party library such as the "DOMPurify" library to return sanitized content from the untrusted input.

```
const policy = trustedTypes.createPolicy("some-content-policy", {
  createHTML(input) {
    return input; // Do not do this in your own code!
    // Instead do something like:
    // return DOMPurify.sanitize(input);
  },
});

const unsafeText = "<strong>inserted text</strong>";
const trustedHTML = policy.createHTML(unsafeText);
```

The remaining code inserts the trusted HTML at the selected position relative to the element with id `subject`.

```
const insert = document.querySelector("#insert");
insert.addEventListener("click", () => {
  const subject = document.querySelector("#subject");
  const positionSelect = document.querySelector("#position");
  subject.insertAdjacentHTML(positionSelect.value, trustedHTML);
});

const reset = document.querySelector("#reset");
reset.addEventListener("click", () => {
  document.location.reload();
});
```

Result

# Specifications

# Browser compatibility

# See also

- Element.insertAdjacentElement()
- Element.insertAdjacentText()
- XMLSerializer: Serialize a DOM tree into an XML string
- Trusted Types API

---

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:34:25 UTC