# \*\* what is the difference between innerHTML and document.createElement :

## 1) **createElement is more performant**

Suppose that you have a div element with the class container:

**<div class="container"></div>**Code language: HTML, XML (xml)

You can new elements to the div element by creating an element and appending it:

let div = document.querySelector('.container');

let p = document.createElement('p');

p.textContent = 'JS DOM';

div.appendChild(p);Code language: JavaScript (javascript)

You can also manipulate an element’s HTML directly using innerHTML like this:

let div = document.querySelector('.container');

div.innerHTML += '<p>JS DOM</p>';Code language: JavaScript (javascript)

Using innerHTML is cleaner and shorter when you want to add attributes to the element:

let div = document.querySelector('.container');

div.innerHTML += '<p class="note">JS DOM</p>';Code language: JavaScript (javascript)

However, using the innerHTML causes the web browsers to reparse and recreate all DOM nodes inside the div element. Therefore, it is less efficient than creating a new element and appending to the div. In other words, creating a new element and appending it to the DOM tree provides better performance than the innerHTML.

## **#2) createElement is more secure**

As mentioned in the innerHTML tutorial, you should use it only when the data comes from a trusted source like a database.

If you set the contents that you have no control over to the innerHTML, the malicious code may be injected and executed.

## **#3) Using DocumentFragment for composing DOM Nodes**

Assuming that you have a list of elements and you need in each iteration:

let div = document.querySelector('.container');

for (let i = 0; i < 1000; i++) {

let p = document.createElement('p');

p.textContent = `Paragraph ${i}`;

div.appendChild(p);

}Code language: JavaScript (javascript)

This code results in recalculation of styles, painting, and layout every iteration. This is not very efficient.