**Phân biệt innerText và innerHTML**

Trong JavaScript, `innerText` và `innerHTML` là hai thuộc tính của các phần tử HTML, nhưng chúng có mục đích khác nhau:

**1. innerText**

- Mục đích: `innerText` được sử dụng để lấy hoặc thiết lập văn bản nằm giữa các thẻ của một phần tử.

- Đặc điểm: Nó chỉ trả về văn bản, không bao gồm các thẻ HTML bên trong phần tử.

- Ví dụ:

var element = document.getElementById("myElement");

var text = element.innerText; // Lấy nội dung văn bản

element.innerText = "New Text"; // Thiết lập nội dung văn bản

**2. innerHTML**

- Mục đích: `innerHTML` cũng được sử dụng để lấy hoặc thiết lập nội dung của một phần tử, nhưng nó bao gồm cả các thẻ HTML bên trong phần tử đó.

- Đặc điểm: Nếu bạn sử dụng `innerHTML` để thiết lập nội dung, nó sẽ hiểu và xử lý các thẻ HTML trong chuỗi cung cấp.

- Ví dụ:

var element = document.getElementById("myElement");

var htmlContent = element.innerHTML; // Lấy nội dung với thẻ HTML

element.innerHTML = "<b>New</b> HTML Content"; // Thiết lập nội dung với thẻ HTML

Lưu ý rằng khi sử dụng `innerHTML`, chúng ta cần cẩn thận để tránh các vấn đề liên quan đến bảo mật, nhất là khi người dùng có thể nhập dữ liệu. Nếu dữ liệu người dùng không được kiểm tra kỹ, có thể xảy ra tấn công Cross-Site Scripting (XSS).

**3. Cross-Site Scripting (XSS)**

Cross-Site Scripting (XSS) là một loại tấn công bảo mật thường xuyên xảy ra khi ứng dụng web không kiểm tra hoặc lọc đầu vào người dùng một cách đúng đắn trước khi hiển thị nó trên trang web. Trong ngữ cảnh của JavaScript và thuộc tính `innerHTML`, XSS có thể xảy ra khi người dùng có thể nhập dữ liệu và các dữ liệu này được sử dụng mà không được kiểm tra đúng cách.

**Ví dụ không an toàn:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>XSS Example</title>

</head>

<body>

    <div id="userInput"></div>

    <script>

        // Giả sử userInput là dữ liệu người dùng nhập vào từ form

        var userInput = "<img src=x onerror='alert(\"XSS Attack\")'>";

        // Sử dụng innerHTML mà không kiểm tra dữ liệu

        document.getElementById("userInput").innerHTML = userInput;

    </script>

</body>

</html>

Trong ví dụ trên, người dùng có thể nhập đoạn mã JavaScript như là một hình ảnh (`<img src=x onerror='alert(\"XSS Attack\")'>`). Khi dữ liệu này được đặt vào phần tử có `innerHTML`, nó sẽ được hiểu là một đoạn mã JavaScript và thực thi, gây ra một cửa sổ cảnh báo ("XSS Attack"). Điều này có thể làm tổn thương người dùng trang web bằng cách chèn mã độc hại hoặc thực hiện các hành động không mong muốn.

**Cách phòng tránh:**

Để ngăn chặn tấn công XSS, chúng ta nên luôn kiểm tra và làm sạch dữ liệu người dùng trước khi chèn nó vào trang web. Thay vì sử dụng `innerHTML`, có thể sử dụng `innerText` để đảm bảo rằng nội dung người dùng được hiển thị dưới dạng văn bản thuần túy mà không được hiểu là mã HTML.

var userInput = "<img src=x onerror='alert(\"XSS Attack\")'>";

document.getElementById("userInput").innerText = userInput;

Ngoài ra, nếu phải chèn HTML từ người dùng, ta cũng có thể sử dụng các thư viện bảo mật như DOMPurify để làm sạch dữ liệu trước khi chèn nó vào trang web.

var userInput = "<img src=x onerror='alert(\"XSS Attack\")'>";

var sanitizedInput = DOMPurify.sanitize(userInput);

document.getElementById("userInput").innerHTML = sanitizedInput;

Bằng cách này, ta có thể giảm thiểu rủi ro XSS và đảm bảo an toàn cho ứng dụng web của mình.