

This is a copy of a chat between Claude and Jono. Content may include unverified or unsafe content

(i) that do not represent the views of Anthropic. Shared snapshot may contain attachments and data not displayed here.

Report

Hey Claude! I have a grammar for a new XSS-proof HTML, safeHTML. Is it safe?

```
Doc      := <doc> EList </doc>
EList    := EList Element | Element | ε
Element  := Img | Para | Table
Table    := <table> RList </table>
RList    := RList Row | Row | ε
Row      := <tr> CList </tr>
CList    := CList Cell | Cell | ε
Cell     := <td> EList </td>
Img      := <img Src />
Src      := src= Href
Href     := /static/1.png | /static/2.png | /static/3.png
Para     := <p> Text </p>
Text     := [a-zA-Z0-9]+
```

Show less

Your grammar is a solid start and does eliminate most classic XSS vectors, but there are a few things to consider:

## What it does well

- **No `<script>` tags** — not in the grammar, so they can't appear.
- **No event handlers** — attributes like `onclick`, `onerror`, `onload` are completely absent.
- **No free-form `href` / `src`** — images are restricted to a hardcoded allowlist of three static paths. No `javascript:` URIs.
- **No raw HTML injection** — text content is strictly `[a-zA-Z0-9]+`, so no `<`, `>`, `&`, quotes, or spaces in text nodes.

## Potential concerns

1. No whitespace in `Text` `[a-zA-Z0-9]+` means paragraphs can't contain spaces, punctuation, or most real prose. This is probably unintentional and makes `<p>` nearly useless. You'd want something like `[ a-zA-Z0-9.,!?]+` — but be careful: allowing `<` or `"` opens injection paths.
2. Only 3 images are ever renderable The `Href` rule is extremely rigid. This may be intentional for a demo, but in practice you'd want a more flexible static-path pattern like `/static/[a-zA-Z0-9_\\-]+\\.\\(png|jpg|gif)` — still safe, but actually usable.
3. No `alt` text on images Minor accessibility issue, not a security one — but worth noting.
4. No CSS or `style` attributes This is actually good from a security standpoint! CSS injection (e.g. `expression()` in old IE, or exfiltration via `background: url(...)`) is a real XSS/data-leak vector, and you've avoided it entirely by omission.
5. Nesting depth is unbounded Tables can contain cells which contain tables (via `EList` → `Element` → `Table`) recursively. Not an XSS issue, but a potential DoS vector if you're parsing untrusted input — a deeply nested document could cause stack overflows in a naive recursive-descent parser.
6. No `<a>` tags No links at all, which is safe but limiting. If you ever add them, `href` would need very careful whitelisting.

## Verdict

For the XSS threat model specifically, **yes, it looks safe** — as long as your parser strictly rejects any document that doesn't match this grammar exactly, with no error-recovery that might let malformed input through. The security of a grammar-based sanitizer lives or dies by the strictness of the parser, not just the grammar itself.



Start your own conversation