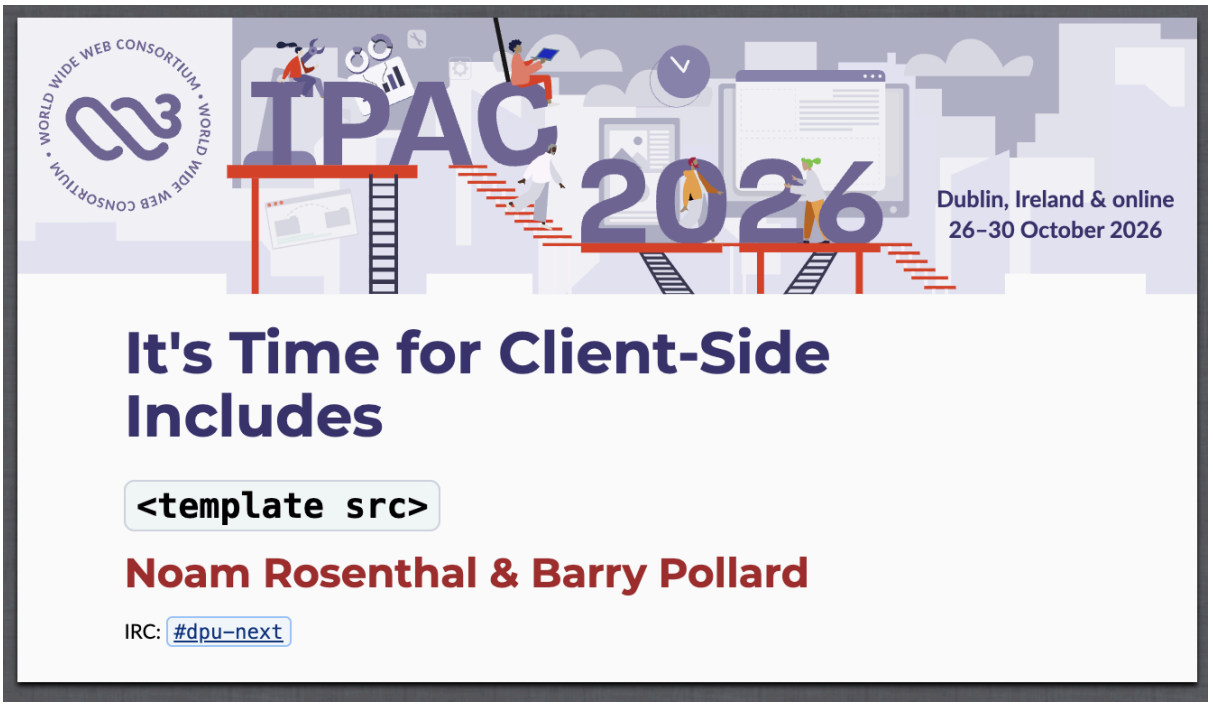
It’s time for client-side includes

Breakout proposal: https://github.com/w3c/tpac2026-breakouts/issues/41
Breakout: https://www.w3.org/events/meetings/5ec70748-3fcf-411e-8a58-aaca280e05dd/

Slides: https://noamr.github.io/tpac-2026-presentations/fragment-include/

Attendees: approx 40 people

Started scribing late. Filled in previous parts from memory. Please feel free to add what I missed!



Noam: Deliberately provocative title, looking for lots of discussion, push-back or better ways of doing this!!

TPAC 2026

30 Years of Trying

- **1995–2001:** Server-only (SSI, PHP, ESI)
- **2008–2018:** `<iframe seamless>` and `<link rel="import">` — not really it
- **2017–Today:** `whatwg/html#2791` — 280+ comments, 225+ 👍

TPAC 2026

Developers Still Want This

"PHP was literally created to solve this problem… Imagine if the client could cache the entire header and footer and only need to download the main content."

— @grepsedawk (`whatwg/html#2791`)

"DRY HTML authoring without running code server-side or requiring JS. It's actually what I naively expected HTML imports to do at first."

— @brandondees (74 👍) · GitHub has used `<include-fragment>` in prod for 10+ years (@josh)

TPAC 2026

Benefits

- Buildless DRY HTML
- Split caching
- Async subsections
- Composition

TPAC 2026

Why It Stalled

1. Parser-blocking
2. Layout shift / FOUC
3. XSS

TPAC 2026

What Changed?

1. Parser blocking → `<template for>`
2. Layout shift / FOUC → render blocking & `buffer` (see later...)
3. XSS → sanitizer

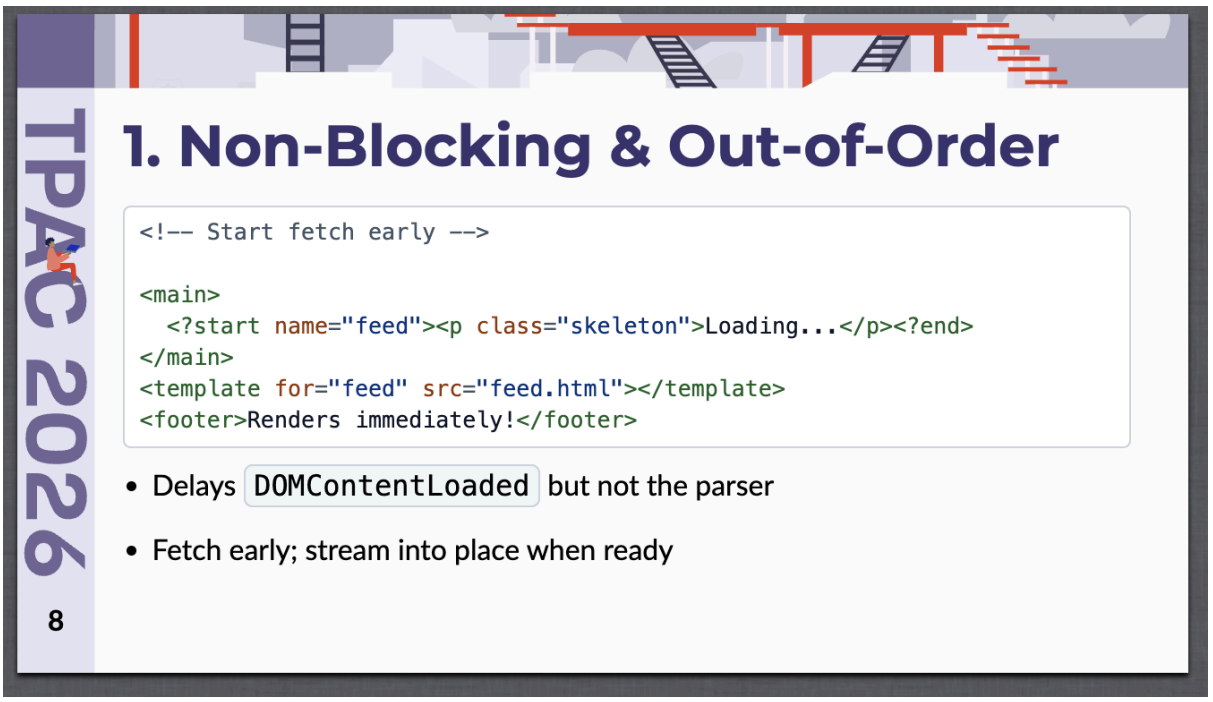
TPAC 2026

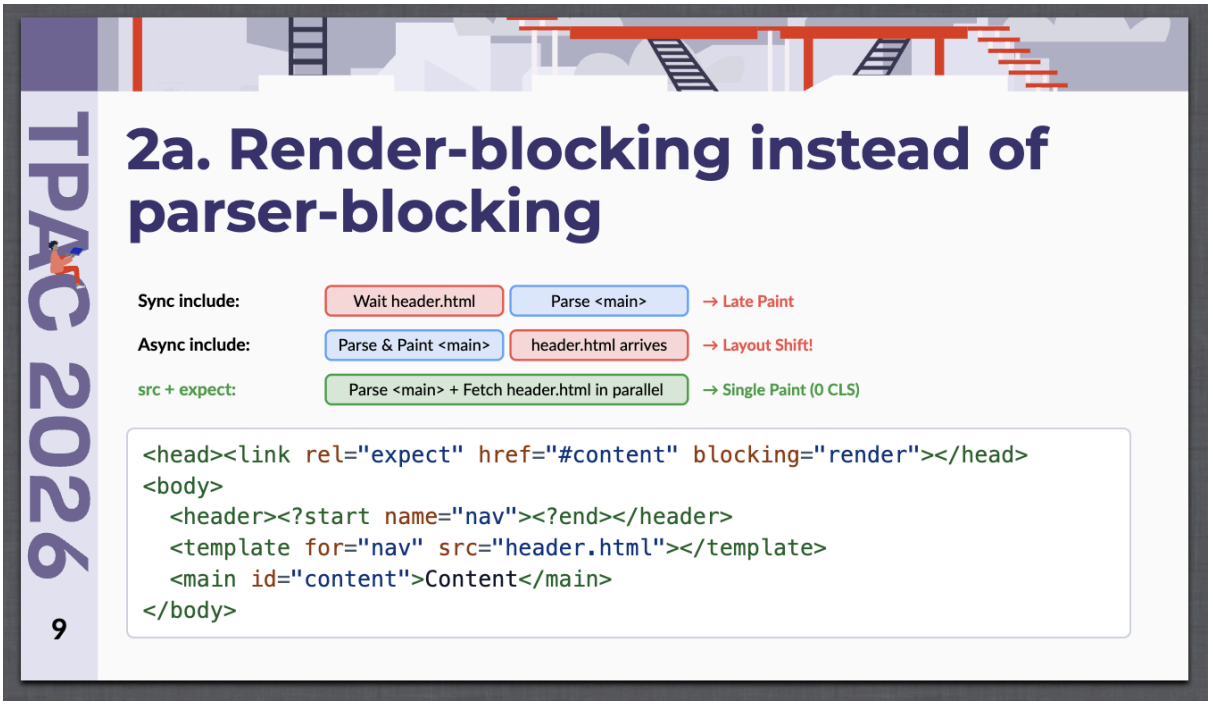
Proposal: `<template for src>`

```
<main>
  <?start name="feed">Loading...<?end>
</main>

<!-- Out-of-order stream from an external URL -->
<template for="feed" src="feed.html"></template>
```

Streams from a URL into a location in the DOM





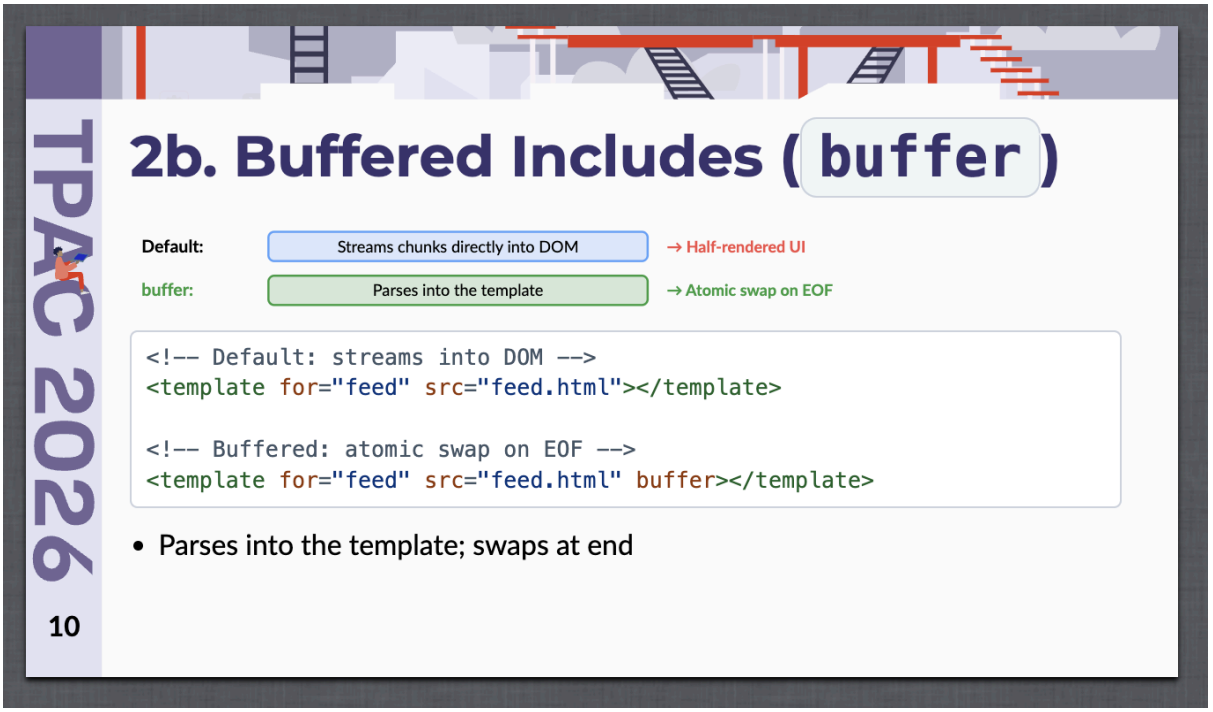
Lots of discussion on this slide. Confusion as to what the slide shows (3 options? 1 option)? Is 2a related? Answer: This slides shows option 3 as options 1 and 2 are non-runners.

Is href="#content" right to be on element after the template? Should it be on the template?

Noam: the `id=content` could be in the template - doesn't need to be in there main document.

Emeeli Why do you have a `<template>` tag?

Keith: What if expect href does not exist?
Noam: What's for DCL
Barry: But if blocked will wait forever
Keith: And if in document will resolve then?
Noam: Yes





TPAC 2026

`buffer` Waits for Subresources

```
<template for="widget" buffer>
  <link rel="stylesheet" href="widget.css">
  <link rel="preload" as="image" href="avatar.jpg">
  <div class="widget"><?start name="icon"><?end></div>
  <template for="icon" src="star.svg"></template>
</template>
```

- Scan links and includes while buffering
- Wait for their contents to fetch before applying buffer

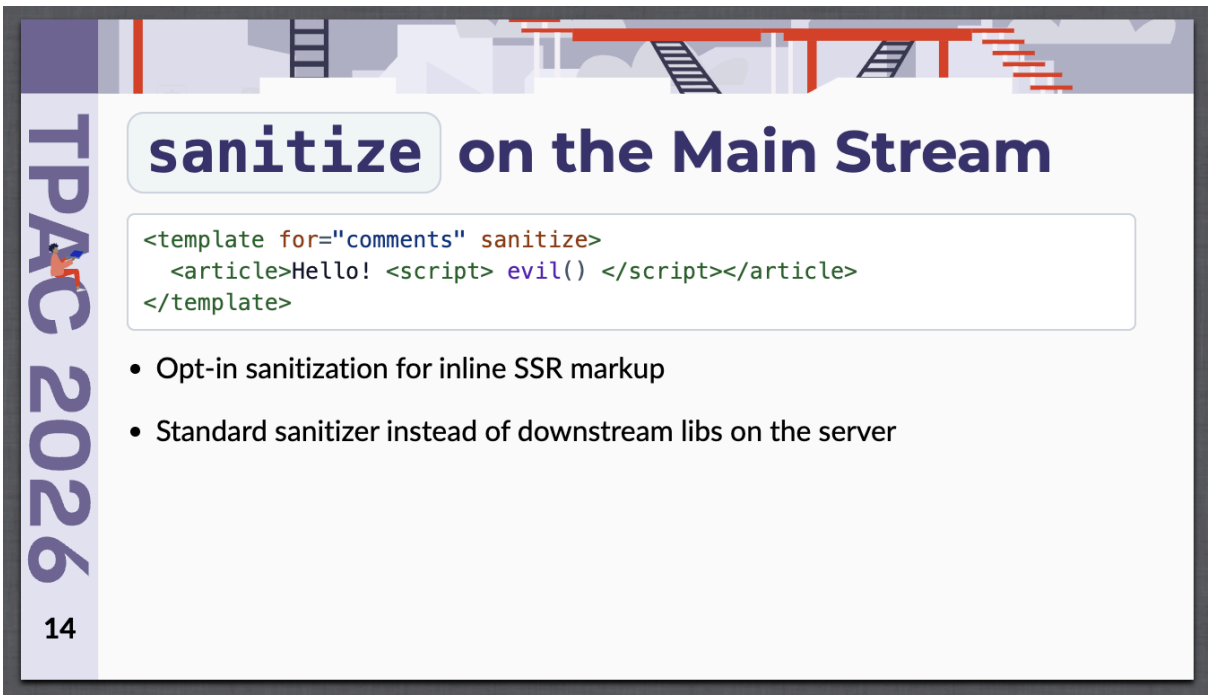
TPAC 2026

3. Safe by Default (`sanitize`)

```
<!-- Safe by default: sanitized with Sanitizer API -->
<template for="promo" src="https://ads.example/banner.html"></template>

<!-- Opt-out for trusted code -->
<template for="app" src="/trusted-app.html" sanitize="unsafe"></template>
```

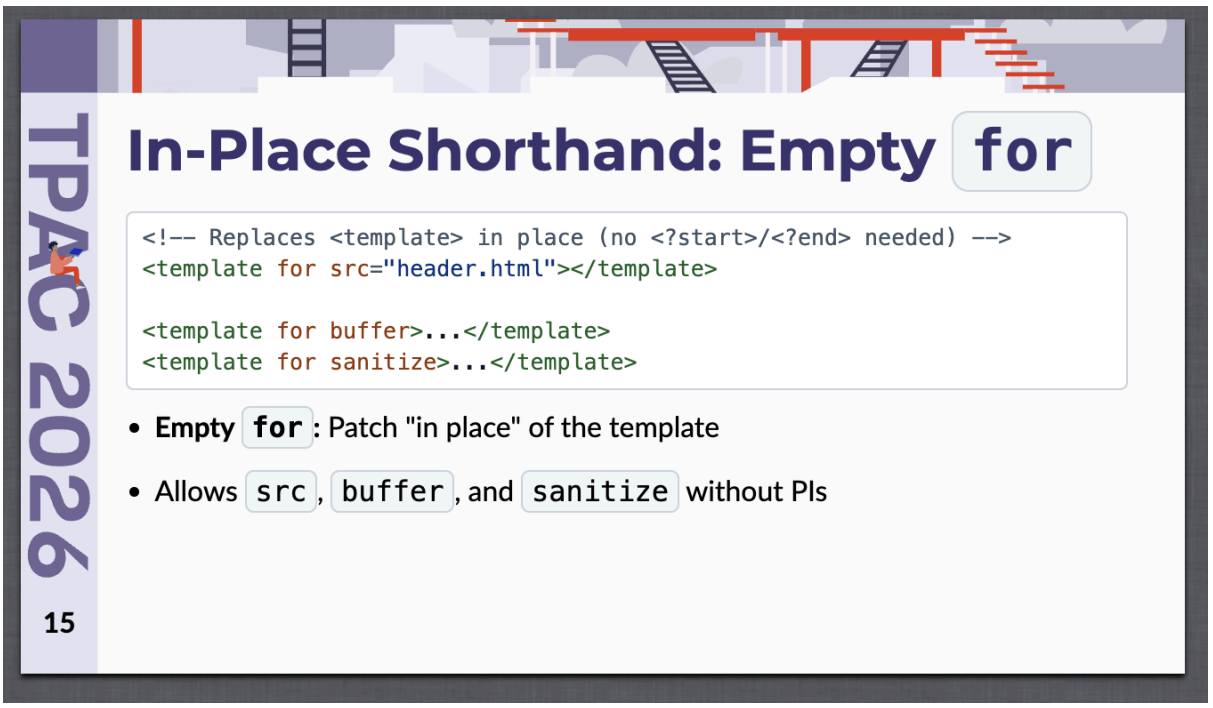
- `src` is sanitized by default
- `sanitize="unsafe"` opts out for trusted fragments



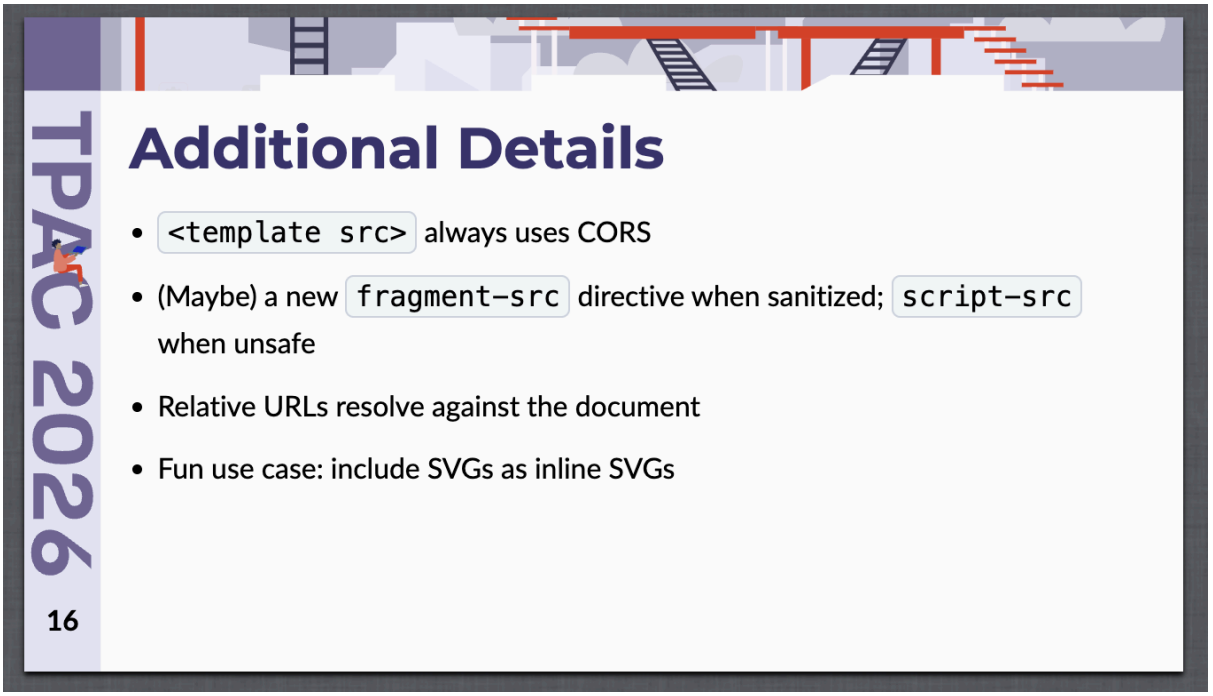
Keith: What if you wanted a different sanitizer config
Noam: Sanitzier allows named configs
Keith: Would need to wait for that to resolve?
Noam: It would probably have to be there before it was used in `<template>`.

Artur: Could you break nesting and change sanitizing?
Bas: Can the content unnest
Noam: No
Simon: Inline sanitization, practically too hard as too easy to break the nesting.

??: Possible to limit data urls to Base 64?
Keith: Isn't that an existing issue? Servers can already inline arbitrary HTML
Artur: The `sanitize` argument makes it look like this safe, when it might not be.
Frederick: Wouldn't sanitizer be dependent of where the content is inserted?
Noam: We don't have that context now?
Rysoke: If content is inserted into a `<table>` it should sanitizer and parse in contest of `<table>`.
Noam: Yes, it does that already.



Noam: Also include inline processing with empty `for`.
??: Could you do self-referencing id?
Noam: No as the for name is the processing instruction.
Noam could have a keyword (for=_self), or wrap template in processing instructions. We'll figure it out.

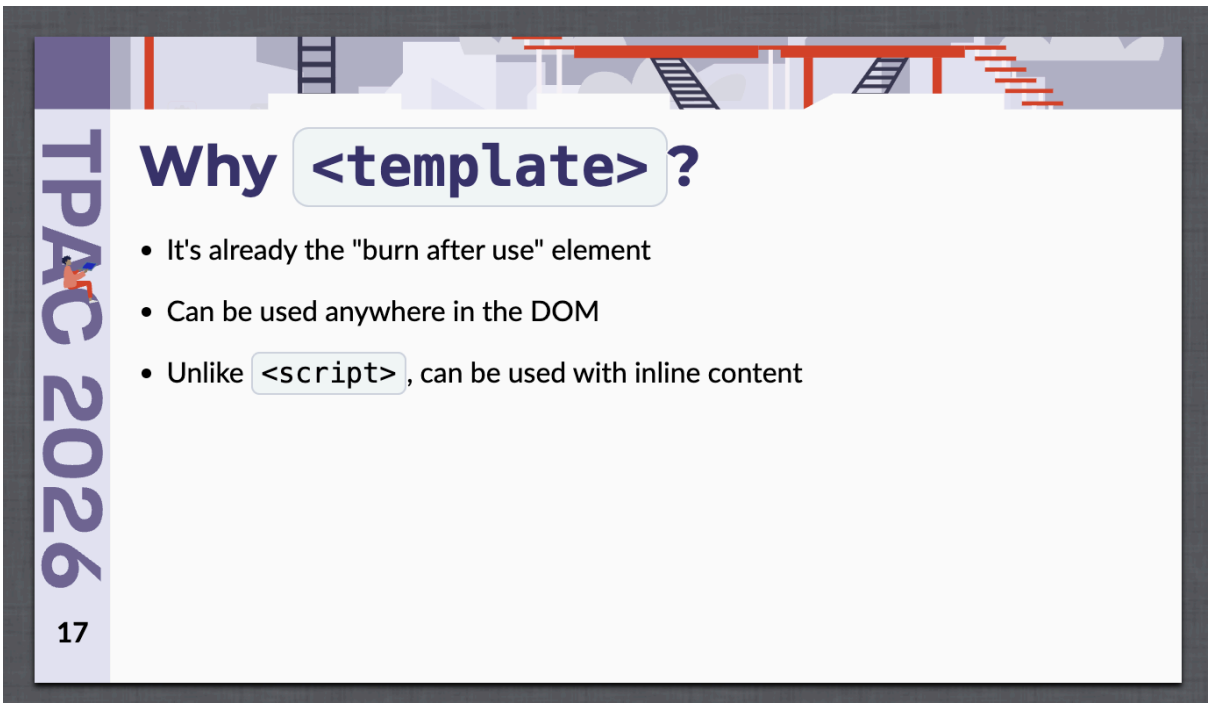


Noam: It's twenty-twenty something so we have CORS
(laughs)
Noam: Still need to think through CSP.

Artur: I would love to talk about this, maybe not now. I like the fact you're thinking about this, even for the unsafe version. It was a problem for HTML imports
Frederick: Unsafe didn't work for HTML imports so maybe not best precedence.
Noam: Relative URLs - for simplicity, href is relative to document importing.

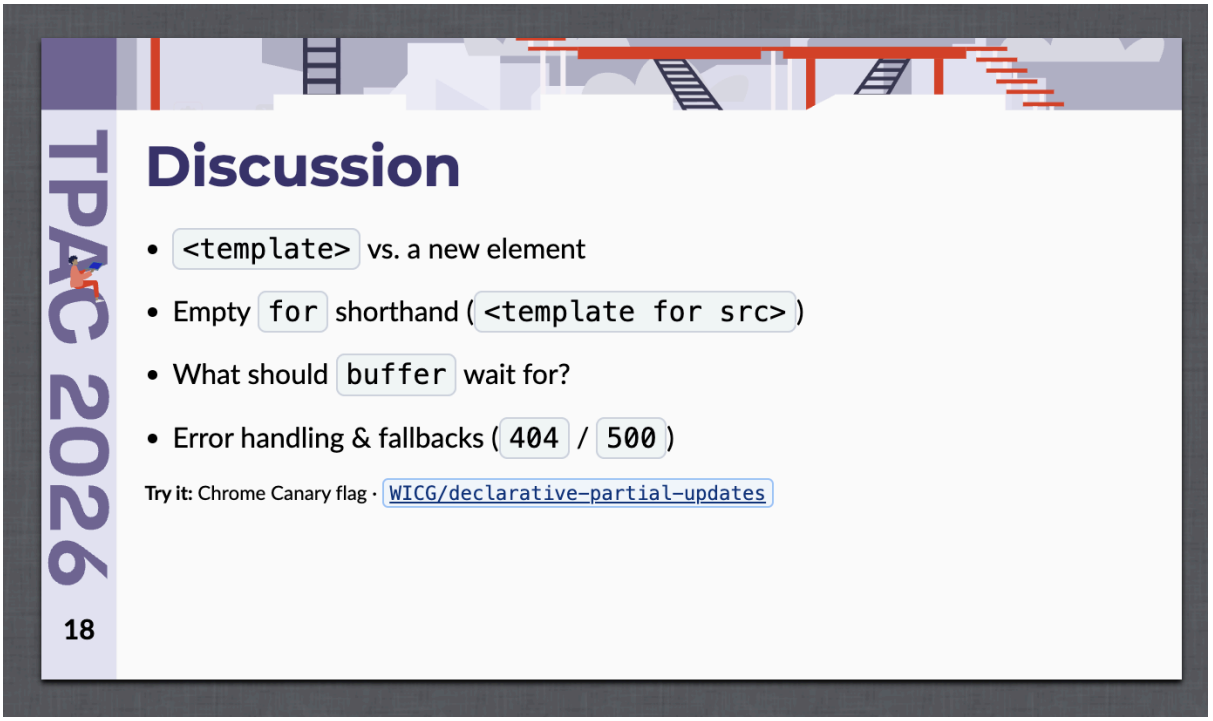
Noam: Inline SVG
Keith: Default Sanitzer excludes ??
Noam: Only if JavaScript URLS
Keith: I wonder if a naive developer would disable Sanitizer to enable more SVG use case.

??: What is the includes? HTML fragments? Or full document?
Noam: Yes fragments, including SVG, MathML. Totally valid HTML, but don't parse as document.
Keith: Linked to important concern: need some some sort of content-negotiation to avoid having 404 included in the page.
Noam: Sec-Fetch-Dest
Barry: Also 404 wouldn't be included
Bas: And any non-success
Noam: Yes, needs to be defined.
Simon: Also SVG mimettype can cause security issues. Should ONLY allow text/html.
Noam: Good point.
Frederick: What if add (self-referencing?) PI?
Noam: Yeah you can't do that.
??: *something about self-refer*
Noam: Default sanitizer doesn't allow processing instructions
Keith: I presume 1-1 with PIs?
Noam: Yes
Keith: Might be nice to support that.
??: *something about self-referencing or in-place updates.*
Noam: At the moment it is replaced. We have future considerations to mutate just the attributes…etc.
Keith: Full HTMX?
Noam: Inspired but one step at a time.
Luke: The sanitizer use is the fairly lax one?
Noam: It's the default baseline sanitizer
Frederick: It's the stricter one.
Luke: I think I'd prefer it to be more lax, or it won't be used.
Noam: Actually I think that's the one I meant. Sorry



Jas: Conversation about module stylesheets and how to make sure <link> is consistent. Might bleed into this conversation.

Noam: Can stream into <tr>
Emeeli: Should you?
Noam: Yeah, why not! If you want to stream in a bunch of rows, that's a good use case.
Keith: Can you have an `</template>` in a template?
Noam: you can nest them.



Noam: Discussed most of this already.

??: One thing discussed was lazy loading?
Keith: It would be nice to have template lazy-loading based on intersection observer.
Noam: loading=lazy sounds cool for this.
Barry: PIs might be further up and need the loading=lazy attribute rather than the template.
Simon: Look at parent of PIs
Michal: Does it make sense to put template higher than PIs to preload
Noam: Just use `<link preload>` if want that.
Emeeli: Would you support a fragment identified on the `src`?
Simon: Just have that content be the template!
Noam: Would have to parse the document.
Emeeli: Wouldn't you have to do that for the sanitizer?
Noam: No it doesn't…
Barry: There's no end fragment
Emeeli: Yes but just take the end of that element. I don't know if this is a good idea?
Simon: No
Noam: No
Keith: Nice use case would be for multiple PI replacements.
Noam: We should look at that
Keith: One piece of feedback for template shadowrootmode. Maybe could solve both at once here.
Noam: And then handlebars and then, and then…
(laughs)

??: Is default for render blocking correct. I can see developers all reaching for that, and the need to have three elements.
Noam:
??: Is the render blocking intuitive?
Noam: Simon, was suggesting render blocking id on the template. We could do that.
Emeeli: This seems complementary (and not against) DOM localisation. That's good.
Noam: Thanks.

Same presentation made at WHATWG

Slides: https://noamr.github.io/tpac-2026-presentations/fragment-include/

Minutes: https://docs.google.com/document/d/1D90QugzZZM6ga8k0ChMA7llmGFHcH9quDj4oCrgqtkE/edit?tab=t.0#heading=h.87sjwnyu5auh

Additional notes specific to this session in this doc.

Attendees: approx 30 people

Jake: Can you template inside of a template, both with buffer.
Noam: Yes
Jake: You reduce layout shifts to 1, rather than avoiding layout shifts completely. I'd like to see some sort of inline render-blocking.
Noam: render-blocking is whole page, you wouldn't want that
Jake: Yes, I'm asking for that
Noam: Can look at that in CSS separate
Anne: Buffer use cases not given, and no need to couple with client-side includes since it also applies to non-src
Noam: The reason why coupled is client-side includes needs this more.
Anne: That's render-blocking no?
Noam: Yes, but only get that once though.
Anne: Other thing is it's not really text/html - no DOCTYPE…etc. It should have it's own type.
Noam: Fine with that, and with exploring more types for buffer.
Anthony: The empty `for` attribute doesn't have fallback as no PI. Is that a problem?
Noam: Effectively the PI is just before
Rniwa: Not blocker, but how does this interact with module scripts?
Noam: Can explore that more
Rniwa: It's similar to HTML imports. Need to make it clear when what is what.
Dan: Not clear how this works. Possible these can be integrated together later. I do seem them as complimentary.
Noam: Can we move to stage 1?
Olli: I think stage 1 is fine. We should support lazy loading.