Standards. . .

V2: Some standard standards maneuvers

Standards are built on standards which are built on standards which are . . .

There are standards, and there are meta-standards (and for more reasons than one)

Being non-standard can be standardized too

Standards are built on standards . . .

Take a close look at almost any standard: XML, ePUB, JSON, etc.

You will see that it is based on other standards

A standard markup language doesn’t need to define its own schema language,

It can specify XML Schema,

and XML Schema in turn doesn’t need to define its own character encoding,

it can specify Unicode,

and so on,

[with great complexity of breadth and depth, especially if “subsetting” takes place
i.e. only portions of the subordinate standards are allowed.
this can make production and validation complicated,
but simplify things for downstream applications]

Meta-standards . . .

Imagine trying to get everyone to agree on a document markup language

Shall it be LaTeX? Or Scribe? Or troff –ms? Or a new one?

Right. It won't be easy. And may not be possible. And you could get hurt trying.

So how about standardize not markup languages, but on *how we define* markup languages.

i.e. how we indicate what our tags are, what patterns or use are legitimate, etc.
[And do it with a processable formal grammar so schemas can be validated and configure applications.]

Then we don't have to agree on so much up front.

We can still keep *trying* to standardize the language itself, and we may succeed (e.g. TEI, JATS, etc.).

But at least we get something important, very important, and necessary in any case, by going meta.

[And that, of course, is why we have standards like XML.]

Standardizing being non-standard . . .

Standards may be extended in standard ways

- e.g. the XML internal subset lets elements and attributes be added to a schema

Completely non-standard constructs may be used

- For instance, one can use an escaped segment to have the processor skip validation

Departures from standards can be graceful

- Don't just start using a construct because you want to and *your* apps can deal with it.
    - (e.g. the infamous <blink> tag that was not in the HTML standard).
- Instead:
    - 1) add it to the schema so at least it won't break validation
    - 2) document it so everyone knows what it means or does

Standards wishing to support such departures . . . extensions . . . can specify

- That 1) and 2) above are required for conformance
- A "fallback"*:
    - Any *non-standard* construct must be mapped to a *standard* construct that approximates the meaning of the non-standard element or is an appropriate alternative action.

*An approach used in the ePUB standard (International Digital Publishing Federation)