Return

of the

succinct

Syntax Tree

and tsv and...

Proskomma (Scripture) Document

Sequence

- Block
  - Array of content => Items
  - Type => bs
  - Child blocks to prepend => bg
  - Fields to track and index scope context across sequence
- Block
- Block

Multiple Document Types

ScriptureText ||

SyntaxTree ||

Commentary ||

...

(More) Generic Document

Sequence

- SequenceElement
  - Cast to Block || Tree *// could add more options*
- SequenceElement
- SequenceElement

Tree

- Array of content => nodes
- Tree type
- Child elements to prepend => ‘bg’
- Referenced sequences (for faster GC)

Node

- nodeLength, for rapid scanning
- nodeType (branch or leaf)
- Index of parent
- Indexes of children
- Attributes:
  - Token ==> docSet enum (very efficient for one symbol)
  - Block ==> sequence.blockN (a paragraph of text)
  - Sequence ==> sequence (a flow of blocks or trees… graph!)

GraphQL

```
sequences { blocks { text } }        // any trees are ignored
sequences { trees { nNodes } }   // any blocks are ignored
sequences {
  elements                                  // Cast
    … on block { text }
    … on tree { nNodes }
}
```

Succinct

- Tree uses docSet enums (no duplication of tokens)
- Use existing succinct primitives:
  - Nbyte for variable-length integers (indexes etc)
  - Counted Strings
- Node structure similar to item structure
- Whole tree in one block of working memory
  - =~ 20-50x smaller than naïve nested objects or DOM
  - Potentially very fast in Go!
- Load/Save as base64 of typed arrays, as for blocks

Document Types Cancelled?

- Certainly less urgent
- See what can be done at sequenceElement level
  - TSV => sequence of row elements
- Scripture Burrito types become a metadata issue
- Revisit if/when necessary