Pinakidion

A Scripture Text Editor Powered By Proskomma

by Mark Howe

August 2021

Pinakidion

"Diminutive of *pinax* – little (wooden) tablet - especially of a writing tablet for notes (Lk 1.63)"

Pinakidion

- Proskomma Scripture Runtime Engine
- Electronite App Framework
  - Chaliki Framework for Proskomma
- SlateJS

https://github.com/Proskomma/pinakidion

<Demo Goes Here>

From Proskomma to Slate and Back

Proskomma

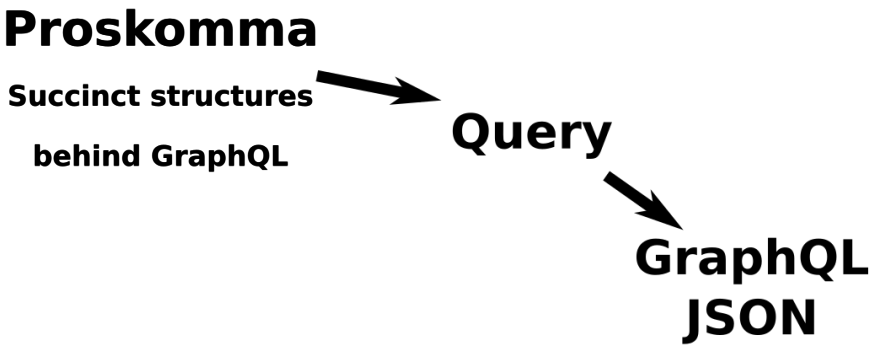
Succinct structures

behind GraphQL

Slate

Editor-centric JSON

From Proskomma to Slate and Back

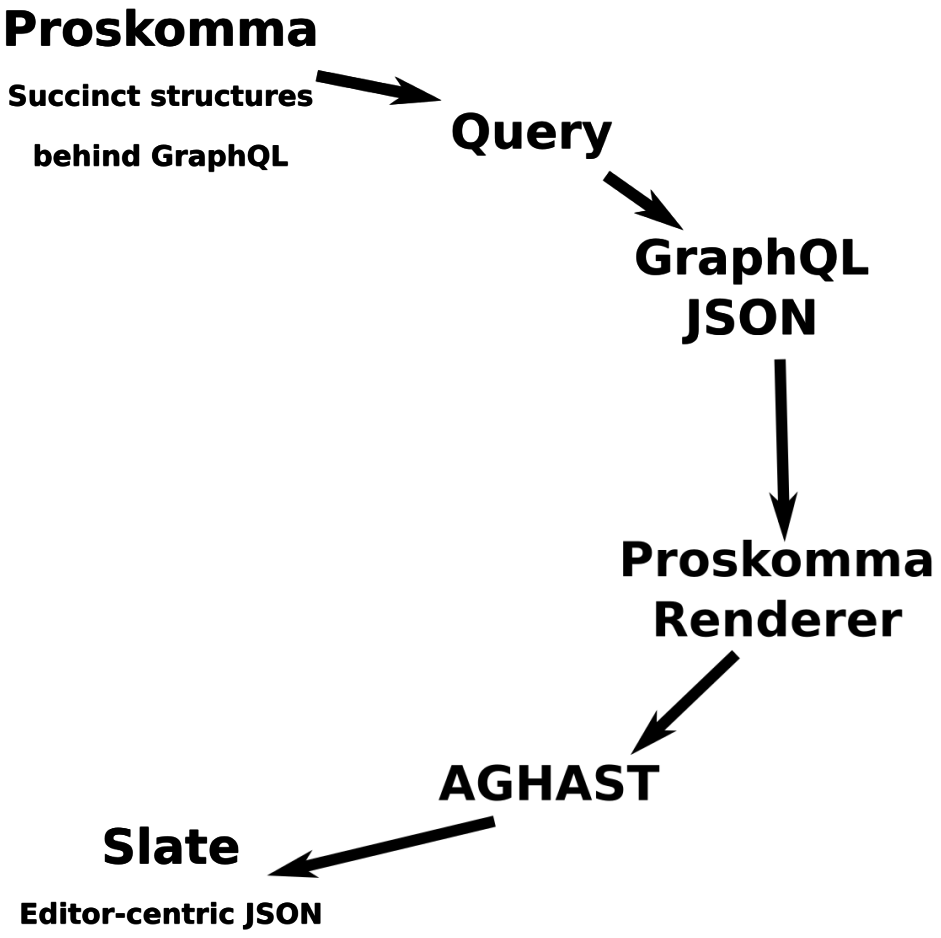


Slate

Editor-centric JSON

```
        {
          "type": "token",
          "subType": "lineSpace",
          "payload": " "
        },
        {
          "type": "token",
          "subType": "wordLike",
          "payload": "praise"
        },
        {
          "type": "token",
          "subType": "lineSpace",
          "payload": " "
        },
        {
          "type": "token",
          "subType": "wordLike",
          "payload": "Yahweh"
        },
        {
          "type": "token",
          "subType": "punctuation",
          "payload": "."
        }
      ]
    },
    {
      "bs": {
        "payload": "blockTag/q"
      },
      "bg": [],
      "items": [
        {
          "type": "token",
          "subType": "wordLike",
          "payload": "Praise"
        },
        {
          "type": "token",
          "subType": "lineSpace",
          "payload": " "
        },
        {
          "type": "token",
          "subType": "wordLike",
          "payload": "Yahweh"
        },
        {
```

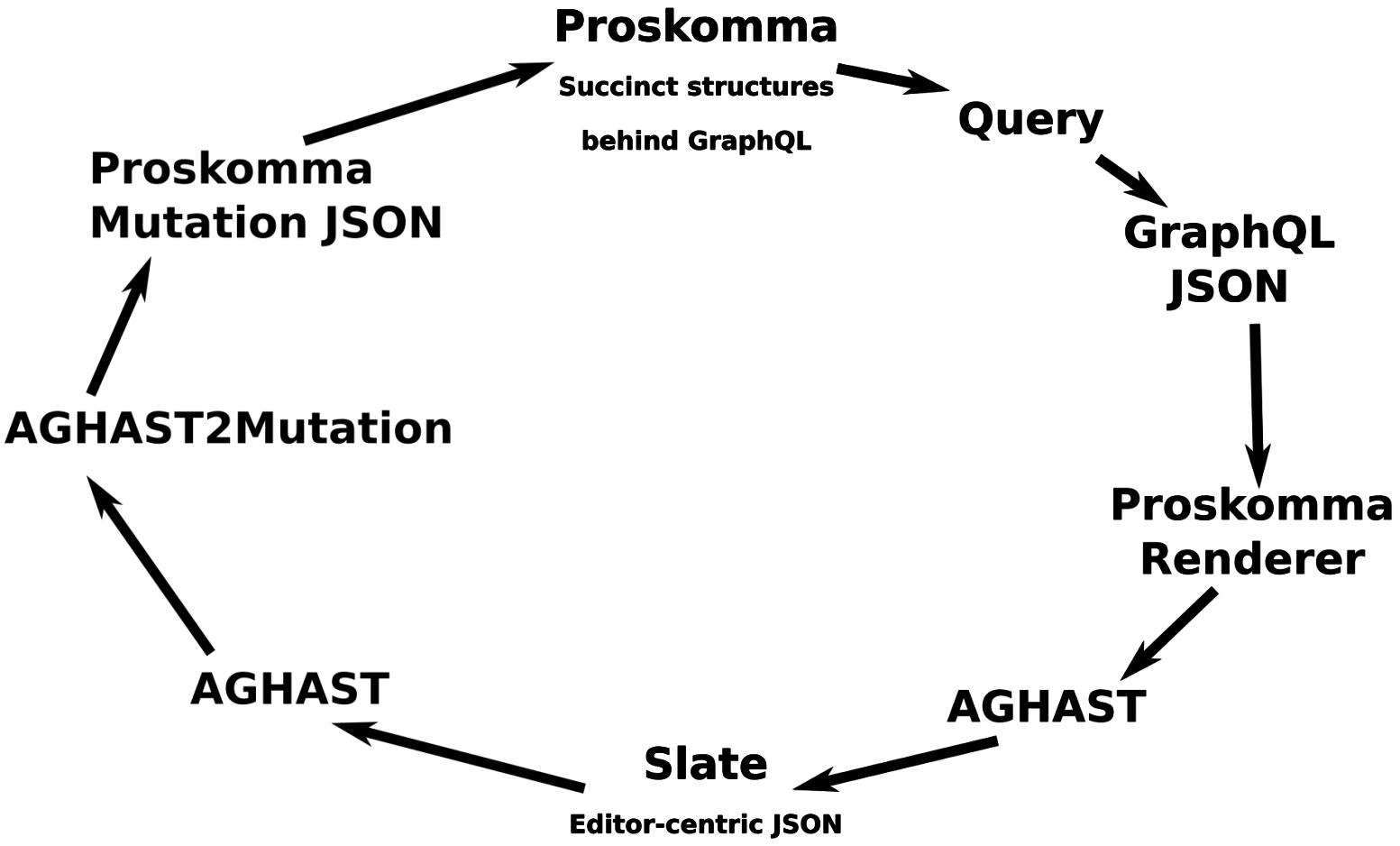
From Proskomma to Slate and Back



```
    }
);
dInstance.addAction(
    'blockGraft',
    context => true,
    (renderer, context, data) => {
        if (selectedSequence(renderer, context)) {
            renderer.config.aghast.children.push({
                type: 'blockGraft',
                subType: data.subType,
                seqId: data.payload,
                children: [{text: `>> ${data.subType[0].toUpperCase()}${data.subType.substring(1)}`}]
            });
        }
        renderer.renderSequenceId(data.payload);
    }
);
dInstance.addAction(
    'startItems',
    context => true,
    (renderer, context) => {
        if (selectedSequence(renderer, context)) {
            renderer.config.aghast.children.push({
                type: 'block',
                scope: context.sequenceStack[0].block.blockScope,
                children: [],
            });
        }
    }
);
dInstance.addAction(
    'scope',
    (context, data) => data.subType === 'start' && ['chapter', 'verses'].includes(data.payload.split('/')[0]),
    (renderer, context, data) => {
        if (selectedSequence(renderer, context)) {
            const lastBlock = renderer.config.aghast.children[renderer.config.aghast.children.length - 1];
            const markElement = {
                type: 'mark',
                scope: data.payload,
                children: [{text: `${data.payload.split('/')[0][0]} ${data.payload.split('/')[1]}`}]
```

```
[
  {
    "type": "blockGraft",
    "subType": "title",
    "seqId": "N2E1NmQ3NTct",
    "children": [
      {
        "text": ">> Title"
      }
    ]
  },
  {
    "type": "block",
    "scope": "blockTag/p",
    "children": [
      {
        "type": "mark",
        "scope": "chapter/1",
        "children": [
          {
            "text": "c 1"
          }
        ]
      },
      {
        "type": "mark",
        "scope": "verses/1",
        "children": [
          {
            "text": "v 1"
          }
        ]
      },
      {
        "text": "Start of the Good News of Jesus Christ, Son of God. "
      },
      {
        "type": "mark",
        "scope": "verses/2",
        "children": [
          {
            "text": "v 2"
          }
        ]
      },
      {
        "text": "In the book of the prophet Isaiah, it is written:"
      }
```

From Proskomma to Slate and Back



```
let aghastSequenceChildren = removeEmptyBlocks(removeEmptyText(rawAghast[0].children));
for (const blockLike of aghastSequenceChildren) {
    if (blockLike.type === 'blockGraft') {
        waitingBlockGrafts.push({
            type: 'graft',
            subType: blockLike.subType,
            payload: blockLike.seqId,
        });
    } else {
        const string2scope = str => ({
           type: 'scope',
           subType: 'start',
           payload: str,
        });
        includedScopes = new Set( values: []);
        const oss = Array.from(openScopes)
        const items = processItems(blockLike.children);
        blocks.push({
            os: oss.map(string2scope),
            is: Array.from(includedScopes).map(string2scope),
            bs: {
                type: 'scope',
                subType: 'start',
                payload: blockLike.scope,
            },
            bg: waitingBlockGrafts,
            items: items,
        });
        waitingBlockGrafts = [];
    }
}
if (currentVerses) {
    endVerses(blocks[blocks.length -1].items, currentVerses);
}
```

```
    "type": "scope",
    "subType": "start",
    "payload": "blockTag/m"
  },
  "bg": [ {
    "type": "graft",
    "subType": "title",
    "payload": "NzE0MDk1ZWIt"
  }],
  "items": [
    {
      "type": "scope",
      "subType": "start",
      "payload": "chapter/1"
    },
    {
      "type": "scope",
      "subType": "start",
      "payload": "verse/1"
    },
    {
      "type": "scope",
      "subType": "start",
      "payload": "verses/1"
    },
    {
      "type": "token",
      "subType": "wordLike",
      "payload": "Start"
    },
    {
      "type": "token",
      "subType": "lineSpace",
      "payload": " "
    },
    {
      "type": "token",
      "subType": "wordLike",
      "payload": "of"
    },
```

What Next?

- More tags
- USFM Export
- Agree editing API (sequence? block? Character?)
- Proskomma-aghast interface layer
- Schemas
- Proskomma does checking and normalization