TEI Reader

Release 0.6.1

Mark Hall

Jan 06, 2021

CONTENTS:

The TEI Reader is a JavaScript web component that lets users read TEI documents in a modern, user-friendly interface. It is primarily aimed at reading for pleasure, but is probably also applicable to other settings such as education.

CHAPTER

ONE

FEATURES

The TEI Reader has the following features:

- Support for reading text, annotations, and metadata
- Responsive design works on phones, tablets, and larger screens
- Interaction design focused on usability

CHAPTER

TWO

CONFIGURATION

The TEI Reader is highly configurable and stylable.

2.1 Main Configuration

The configuration must be made available via a `<script>` tag with the `id` attribute set to "TEIReaderConfig" and the `type` set to "application/json":

```
<script id="TEIReaderConfig" type="application/json">
  {
    "sections": ...
  }
</script>
```

The configuration itself is provided as one large JSON object, which is documented here. The documentation uses a semi-formal format, mainly showing the individual JSON objects that make up the configuration, with some additional markers:

- Optional keys are marked with "?".
- Keys that can be repeated are marked with "+".
- Nested JSON objects are indicated with a camel-case value (e.g. "ParserElement"). When creating an actual configuration object, these are replaced with the nested JSON object.
- Customisable values are indicates by the values "AnyString" (any string content allowed), "AnyHTMLString" (any HTML content is allowed), and "Boolean" (true or false).
- Choices are marked using the pipe "|" symbol.
- Fixed values are indicated using values that start with a lower-case letter.
- Where lists are shown with a single value, this value can be repeated as often as required.

The top-level configuration object is structured as follows:

```
{
  "sections": {
    "+SectionIdentifier": "TextSection | NestedListSection | MetadataSection"
  }
  "ui": "ReaderUI"
}
```

It consists of one or more section identifier keys, each of which has either a `TextSection` or a `MetadataSection` object as its value. The TEI Reader can handle any number of `TextSection`, however there may be at most one `MetadataSection`.

2.1.1 TextSection

The `TextSection` represents a section in which the user can read a TEI text block:

```
{
  "label": "AnyString",
  "type": "Text",
  "parser": "ParserElement",
  "schema": [
    "TagElement"
  ]
}
```

The `label` can be any string value and is used in the UI to allow the user to navigate to this section. The `type` must be `"Text"`. The `parser` configuration is used to identify the root element in the TEI document that contains the text to be edited in this section. The `schema` contains a list of `TagElement` objects that identify the various markup elements that have been used to annotate the text in this section.

TagElement

The `TagElement` represents one markup tag that is used in the TEI text edited in the section it is specified in.

```
{
  "name": "AnyString",
  "type": "block | wrapping | nested | inline | mark",
  "attrs": {
    "+AttributeName": "ElementAttribute"
  },
  "?parser": "ParserElement",
  "?parsers": ["ParserElement"],
  "?content": "ElementName",
  "?reference": "NestedReferenceElement",
  "?navigation": "NavigationElement"
}
```

The `name` can be any value, but each `name` **must** be unique within the `TextSection`. The `type` defines the type of markup the `TagElement` represents:

- *block*: A basic block-level element.
- *wrapping*: A block-level element that contains another block-level element. The name of the inner block-level element **must** be specified in the `content` key.
- *nested*: The root element for a nested document. Nested documents **must** have an `"xml:id"` attribute that specifies the unique identifier for each nested document. This must be in the format `nestedDocumentElementName-UniqueNumber`.
- *inline*: An inline element.
- *mark*: A formatting mark that is attached either to text or to an inline element.

The distinction between inline and mark elements is fluid, but in general you should prefer mark elements for formatting and styling markup and inline elements to mark semantic content.

The `attrs` object maps attribute names (which can by any string value) to `ElementAttribute` configurations that specify how the attribute is parsed and serialised.

Each `TagElement` **must** specify either a single `parser` or a list of `parsers` that specify which TEI tags are mapped to this `TagElement`.

The `content` **must** and **may only** be specified for a `TagElement` that has the `type` `"wrapping"`. In that case it **must** be set to the `name` of the `TagElement` that may be contained by the wrapping `TagElement`.

The `reference` is specified for any `TagElement` that represents the reference to a nested document and specifies how the two are linked together.

ElementAttribute

The `ElementAttribute` specifies the default value for the attribute and how it is parsed:

```
{
  "default": "AnyString",
  "?parser": "ParserElement",
  "?parsers": ["ParserElement"]
}
```

As with the `TagElement`, either a single `parser` or multiple `parsers` **must** be provided to specify how the attribute is parsed from the TEI document.

The `default` specifies the default value that is set for the attribute if no valid value can be parsed from the TEI document.

NestedReferenceElement

Editing nested documents consists of two steps. First, the user needs to mark up the text that represents the reference to the nested document. Then they need to edit the nested document. The `NestedReferenceElement` specifies the link from the reference element to the nested document.

```
{
  "type": "ElementName",
  "attr": "AttributeName",
  "display": "sidebar | footnote"
}
```

The `type` specifies the `name` of the `TagElement` that represents the nested documents. The `attr` specifies the attribute on the reference element that contains the nested document's unique identifier. The `display` configures whether the nested document is to be shown in the sidebar or in the footnote area of the user interface. However, if the device used by the user is small, then all nested documents are shown in the footnote area.

NavigationElement

The `NavigationElement` configures which attribute to use to generate the in-text navigation elements.

```
{
  "attr": "AttributeName"
}
```

The `attr` to use for navigation.

ParserElement

The `ParserElement` specifies how a `TagElement` or `ElementAttribute` is parsed from the TEI document.

```
{
  "selector": "XpathSelector",
  "?type": "static",
  "?value": "AnyString",
  "?text": "xpath-text-selector"
}
```

The `selector` contains an XPath selector. The selector is configured to require the "tei" prefix on all TEI nodes, for example "tei:head[@type="level-1"]".

When used in the `TagElement` for inline or mark elements, the `text` **may** be used and contains a further XPath selector that specfifies how the text content is to be parsed, relative to the TEI element selected via the `selector` XPath.

When used in the `ElementAttribute`, the attribute's value by default is set to the result of the `selector`. However, if the `type` is specified with the value `"static"`, then if the `selector` matches, the attribute's value is set to the value specified in `value`.

2.1.2 NestedListSection

The `NestedListSection` represents a section where the user can read a list of nested documents.

```
{
  "label": "AnyString",
  "type": "NestedList",
  "source": "SectionName",
  "nodeName": "TagElementName"
}
```

The `label` can be any string value and is used in the UI to allow the user to navigate to this section. The `type` must be `"NestedList"`. The `source` is the name of the `TextSection` that contains the document from which to show the nested documents. The `nodeName` specifies the name of the nested `TagElement` to show the individual nested documents for.

2.1.3 MetadataSection

The `MetadataSection` configures the Metadata reader.

```
{
  "label": "AnyString",
  "type": "Metadata",
  "schema": ["MetadataReaderElement"],
  "ui": ["MetadataReaderUISection"]
}
```

The `label` can be any string value and is used in the UI to allow the user to navigate to this section. The `type` must be `"Metadata"`. The `schema` specifies how the metadata is parsed from the TEI document. The `ui` specifies how the metadata is displayed to the user.

MetadataReaderElement

The `MetadataReaderElement` is used to convert the TEI header into a tree-structure that can then be viewed via the UI.

```
{
  "tag": "AnyString",
  "?children": ["MetadataReaderElement"],
  "?multiple": "Boolean"
}
```

The `tag` specifies the TEI tag that this `MetadataReaderElement` matches. If it matches, then if any `children` are specified, the matching is applied recursively.

If `multiple` is set to `true`, then a list of all matching TEI tags is generated, otherwise the first matching TEI tag is stored.

MetadataReaderUISection

The `MetadataReaderUISection` is used to visually separate sections of the metadata to edit.

```
{
  "label": "AnyString",
  "entries": ["MetadataReaderUIElement"]
}
```

The `label` is used as the heading that is displayed to the user. The `entries` define the editable UI elements.

MetadataReaderUIElement

The `MetadataReaderUIElement` is used to create the actual interface for editing the metadata.

```
{
  "type": "single-text | multi-field | multi-row",
  "label": "AnyString",
  "path": "DottedPath",
  "?entries": ["MetadataReaderUIElement"]
}
```

The `type` specifies how the element is displayed and **must** be one of `"single-text"`, `"multi-field"`, or `"multi-row"`. The `label` is used to label the input element. The `path` is a dotted path that specifies the location in the tree of the metadata to edit. The optional `entries` allow nesting `MetadataReaderUIElement` to enable complex displays

If the `type` is `"single-text"`, then the value specified by the `path` is displayed. If the `type` is `multi-row`, then the `entries` **must** be specified and define the `MetadataReaderUIElements` that make up one row. If the `type` is `multi-field` then the `entries` **must** be specified and define the `MetadataReaderUIElements` that conceptually belong together.

In general the `multi-field` `MetadataReaderUIElement` are contained within `multi-row` `MetadataReaderUIElements`.

The full path for accessing the metadata from the tree structure is calculated by concatenating all the `path` values for the nested `MetadataReaderUIElements`.

2.2 Callbacks

In addition to the static configuration, the TEI Reader supports twp JavaScript callbacks. To add callbacks, create a new `TEIReader` object on the `window` object:

```
<script type="application/javascript">
  window.TEIEditor = {
      callbacks: {
      }
  }
</script>
```

There are two optional callbacks that can be defined on the `callback` object:

```
<script type="application/javascript">
  window.TEIEditor = {
      callbacks: {
          autoLoad: function(callback) {
          },
          close: function() {
          }
      }
  }
</script>
```

The `autoLoad` callback is called once after the TEI Reader has initialised itself. Use this to load a TEI document. The TEI document **must** be passed as a string to the `callback` function parameter.

The `close` callback is called when the user closes the reader.

2.3 Styling

The core TEI editor comes with the minimal styling needed to layout the editor.

2.3.1 Text

All `block` and `wrapping` `TagElements` are rendered as `<div>` elements, with the `class` attribute set to `"node-{TagElementName}"`. `inline` `TagElements` are rendered as `<span>` elements, with the `class` attribute set to `"node-{TagElementName}"`.

All `mark` `TagElements` are rendered as `<span>` elements, with the `class` attribute set to `"mark-{TagElementName}"`.

All `TagElements` attributes are added to the respective `<div>` or `<span>` elements as `data-attributeName="attributeValue"`.

2.3.2 Metadata

For each `MetadataReaderUISection` a `<section>` tag is generated. For `MetadataReaderUIElements` that have a `multi-row` or `multi-field` `type` an `<ol>` tag is generated with the `class` attribute set to the `type`.

Additionally for the `multi-row` elements there is an `<ul role="menubar">` that contains the buttons for adding, removing, and re-ordering the multi-row children.

CHAPTER

THREE

DEMO

Below is a demo showing all features provided by the TEI editor:

The reader above uses the following configuration:

```
{
    "sections": {
        "body": {
            "label": "Text",
            "type": "Text",
            "parser": {
                "selector": "tei:text/tei:body"
            },
            "schema": [
                {
                    "name": "paragraph",
                    "type": "block",
                    "parser": {
                        "selector": "tei:p"
                    }
                },
                {
                    "name": "title-page",
                    "type": "block",
                    "parser": {
                        "selector": "tei:titlePage"
                    }
                },
                {
                    "name": "title-part",
                    "type": "block",
                    "parser": {
                        "selector": "tei:titlePart"
                    }
                },
                {
                    "name": "doc-edition",
                    "type": "block",
                    "parser": {
                        "selector": "tei:docEdition"
                    }
                },
                {
                    "name": "doc-imprint",
                    "type": "block",
```

(continues on next page)

(continued from previous page)

```
                "parser": {
                    "selector": "tei:docImprint"
                }
            },
            {
                "name": "byline",
                "type": "block",
                "parser": {
                    "selector": "tei:byline"
                }
            },
            {
                "name": "head",
                "type": "block",
                "parser": {
                    "selector": "tei:head"
                },
                "attrs": {
                    "id": {
                        "parser": {
                            "selector": "@xml:id"
                        }
                    }
                },
                "navigation": {
                    "attr": "id"
                }
            },
            {
                "name": "line-group",
                "type": "block",
                "parser": {
                    "selector": "tei:lg"
                }
            },
            {
                "name": "line",
                "type": "block",
                "parser": {
                    "selector": "tei:l"
                }
            },
            {
                "name": "paragraph",
                "type": "block",
                "parser": {
                    "selector": "tei:p"
                }
            },
            {
                "name": "text",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:seg",
                        "text": "text()"
                    },
```

(continues on next page)

(continued from previous page)

```
                    {
                        "selector": "tei:hi",
                        "text": "text()"
                    }
                ]
            },
            {
                "name": "page-break",
                "type": "inline",
                "parser": {
                    "selector": "tei:pb"
                }
            },
            {
                "name": "line-break",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:lb"
                    }
                ]
            },
            {
                "name": "pub-place",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:pubPlace"
                    }
                ]
            },
            {
                "name": "publisher",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:publisher"
                    }
                ]
            },
            {
                "name": "doc-date",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:docDate"
                    }
                ]
            },
            {
                "name": "choice-ref",
                "type": "inline",
                "parser": {
                    "selector": "tei:ref[@type='choice']",
                    "text": "text()"
                },
                "attrs": {
```

(continues on next page)

(continued from previous page)

```
                    "target": {
                        "parser": {
                            "selector": "@target"
                        }
                    }
                },
                "reference": {
                    "type": "choice",
                    "attr": "target",
                    "display": "sidebar"
                }
            },
            {
                "name": "choice",
                "type": "nested",
                "parsers": {
                    "selector": "tei:choice"
                },
                "attrs": {
                    "id": {
                        "parser": {
                            "selector": "@xml:id"
                        }
                    }
                }
            },
            {
                "name": "sic",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:sic",
                        "text": "text()"
                    }
                ]
            },
            {
                "name": "corr",
                "type": "inline",
                "parsers": [
                    {
                        "selector": "tei:corr",
                        "text": "text()"
                    }
                ]
            },
            {
                "name": "letter-spacing",
                "type": "mark",
                "parsers": [
                    {
                        "selector": "contains(@style, 'letter-spacing')"
                    }
                ]
            },
            {
                "name": "initial-letter",
```

(continues on next page)

(continued from previous page)

```
                "type": "mark",
                "parsers": [
                    {
                        "selector": "contains(@style, 'initial-letter')"
                    }
                ]
            }
        ]
    },
    "changes": {
        "label": "Edits",
        "type": "NestedList",
        "source": "body",
        "nodeName": "choice"
    },
    "metadata": {
        "label": "About",
        "type": "Metadata",
        "schema": [
            {
                "tag": "tei:fileDesc",
                "children": [
                    {
                        "tag": "tei:titleStmt",
                        "children": [
                            {
                                "tag": "tei:title"
                            },
                            {
                                "tag": "tei:author",
                                "children": [
                                    {
                                        "tag": "tei:persName"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "tag": "tei:publicationStmt",
                        "children": [
                            {
                                "tag": "tei:publisher",
                                "children": [
                                    {
                                        "tag": "tei:orgName",
                                        "multiple": true
                                    }
                                ]
                            },
                            {
                                "tag": "tei:availability",
                                "children": [
                                    {
                                        "tag": "tei:licence",
                                        "children": [
                                            {
```

(continues on next page)

(continued from previous page)

```
                                                    "tag": "tei:p"
                                                }
                                            ]
                                        }
                                    ]
                                }
                            ]
                        }
                    ],
                    "ui": [
                        {
                            "label": "Bibliography",
                            "entries": [
                                {
                                    "type": "single-text",
                                    "label": "Title",
                                    "path": "fileDesc.titleStmt.title._text"
                                },
                                {
                                    "type": "single-text",
                                    "label": "Author",
                                    "path": "fileDesc.titleStmt.author.persName._text"
                                }
                            ]
                        },
                        {
                            "label": "Digital Version",
                            "entries": [
                                {
                                    "type": "single-text",
                                    "label": "License",
                                    "path": "fileDesc.publicationStmt.availability.licence.p._
↪text"
                                },
                                {
                                    "type": "multi-row",
                                    "path": "fileDesc.publicationStmt.publisher.orgName",
                                    "entries": [
                                        {
                                            "type": "single-text",
                                            "label": "Publisher",
                                            "path": "._text"
                                        }
                                    ]
                                }
                            ]
                        }
                    ]
                }
            },
            "ui": {
                "closeLabel": "Close"
            }
}
```

The reader also uses the following callbacks:

```
<script type="application/javascript">
  window.TEIReader = {
      callbacks: {
          autoLoad: function(callback) {
              const request = window.fetch('text.tei');
              request.then((response) => {
                  response.text().then((text) => {
                      callback({
                          content: text,
                          identifier: 'tei-reader-demo',
                      });
                  })
              });
          },
          close: function() {
              alert('There is nothing else to see here');
          }
      }
  }
</script>
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search