TEI Reader Documentation

Release 0.1.1

Mark Hall

May 24, 2019

Contents:

The TEI Reader is a simple custom HTML element that generates a reader interface for reading TEI documents. It supports transforming TEI into HTML both on the client and server side. It comes with basic styling for both small and large screens, but this can be customised as desired.

CHAPTER 1

Quickstart

1.1 Dependencies

The TEI Reader is distributed as an NPM package, you thus require one of the following:

- Yarn
- NPM

1.2 Installation

To install the TEI Reader either run

```
$ yarn add tei-reader
```

or

```
$ npm install tei-reader
```

depending on which package manager you use.

1.3 Embedding the Reader

The TEI Reader can be embedded into any HTML page in two steps. First the TEI Reader script and styles need to be included in the HTML `<head>` element:

```
<head>
  ...
  <link rel="stylesheet" href="node_modules/tei-reader/dist/app.css"/>
  <script src="node_modules/tei-reader/dist/app.js"></script>
```

(continues on next page)

(continued from previous page)

```
  ...
</head>
```

Then the reader itself can be embedded anywhere in the `<body>`:

```
<body>
  ...
  <tei-reader></tei-reader>
  ...
</body>
```

After loading the TEI Reader then replaces the `<tei-reader>` tag with its own content, which can then be *styled as desired*. The main tag for the content is the following:

```
<body>
  ...
  <div id="tei-reader">
    ...
  </div>
  ...
</body>
```

The detailed HTML structure is documented in the *styling* section.

1.3.1 Showing (and Hiding) the Reader

By default the TEI Reader is hidden. To show the TEI Reader, you need to set the `class` attribute of the `#tei-reader` element to `"visible"`:

```
document.querySelector('#tei-reader').classList.add('visible');
```

To hide the TEI Reader attach a click event listener to the `#tei-reader`. Any clicks that are directly on that element are clicks on the overlay, which is the moment when you should hide the TEI Reader:

```
var teiReader = document.querySelector('#tei-reader');
teiReader.addEventListener('click', function (ev) {
    if (ev.target === teiReader) {
        teiReader.classList.remove('visible');
    }
});
```

1.4 Loading Data

The TEI Reader needs three pieces of data: the main text, the navigation data, and the annotation data. These can either be loaded directly from an element in the HTML page itself or can be loaded remotely. Which loader to use and its configuration are set via the `window.teiReaderConfig` object, which **must** be defined before you use the `<tei-reader>` tag.

1.4.1 Embedded Content

For the embedded content loader, use the following configuration setting:

```
<script>
  window.teiReaderConfig = {
      loader: 'local-html-json',
      elements: {
          text: '#source .text',
          nav: '#source .nav',
          annotation: '#source .annotation'
      }
  };
</script>
```

The three settings `elements.text`, `elements.nav`, and `elements.annotation` all contain CSS selectors that mark the elements that contain the main text, navigation configuration, and annotation configuration. The text element can contain any HTML you wish to use. Details on the content for the navigation and annotation elements can be found in the *content* documentation.

1.4.2 Remote Content

For the remote content loader, use the following configuration setting:

```
<script>
  window.teiReaderConfig = {
      loader: 'remote-tei-xslt',
      urls: {
          tei: 'text.tei',
          text_extract: 'text-extract.xslt',
          nav_extract: 'nav-extract.xslt',
          annotation_extract: 'annotation-extract.xslt'
      }
  };
</script>
```

The `urls.tei` setting defines the URL from which to load the TEI content file. The `urls.text_extract`, `urls.nav_extract`, and `urls.annotation` settings define the URLs from which to load the XSLT stylesheets for transforming the TEI file into the text, navigation, and annotation settings. The text XSLT must produce HTML output, while the navigation and annotation XSLTs must produce JSON as documented *here*.

CHAPTER 2

Content

The TEI Reader uses three types of content to generate its interface. The text content contains the main text to display, the navigation content generates the navigation bar on the left-hand side, and the annotation content generates the annotations on the right-hand side.

2.1 Text

The text content can be any valid HTML, the TEI Reader defines no constraints. The only requirements are specific attributes to link the text content to the navigation and annotation:

`data-heading-id` This attribute needs to be set on all elements that are headings that are also configured in the navigation. The values do **not** have to be unique.

`data-annotation-id` This attribute needs to be set on all elements that are linked to an annotation. The values do **not** have to be unique.

Example:

```
<h1 data-heading-id="welcome">Welcome</h1>
<p>Here is an <a data-annotation-id="1">annotation</a> that is shown
  on the right, if the user clicks on the link.</p>
<h2 data-heading-id="chapter-1">Chapter 1</h2>
<h3 data-heading-id="chapter-1">A useful chapter</h3>
```

2.2 Navigation

The navigation content is a JSONAPI formatted data-structure that lists all navigation elements that are to be displayed. Each element has two attributes, the `label` to display in the navigation list, and the `level` of the heading. Each element is structured like this:

```
{
    "type": "headings",
    "id": "id-value",
    "attributes": {
        "labels": [
            {
                "label": "Label to show",
                "level": "Value 1 - 6"
            }
        ]
    }
}
```

Full example:

```
{
    "data": [
        {
            "type": "headings",
            "id": "welcome",
            "attributes": {
                "labels": [
                    {
                        "label": "Welcome",
                        "level": 1
                    }
                ]
            }
        },
        {
            "type": "headings",
            "id": "chapter-1",
            "attributes": {
                "labels": [
                    {
                        "label": "Chapter 1",
                        "level": 2
                    },
                    {
                        "label": "A useful chapter",
                        "level": 3
                    }
                ]
            }
        }
    ]
}
```

The `data` list **must** contain at least one entry for each heading that is annotated with the `data-heading-id` in the text content.

2.3 Annotations

The annotation content is also a JSONAPI formatted data-structure that lists all annotations that are referenced in the text content. Each annotation has two attributes `title` and `content`, both of which can contain HTML content. The basic structure for each annotation is as follows:

```
{
    "type": "annotations",
    "id": "id-value",
    "attributes": {
        "title": "Annotation title to display (escaped HTML)",
        "content": "Annotation content to display (escaped HTML)"
    }
}
```

Important: If you are using the embedded loader, you **must** escape your HTML so that the browser does not parse it.

For the text content example above, the annotation could look like this:

```
{
    "data": [
        {
            "type": "annotations",
            "id": "1",
            "attributes": {
                "title": "&lt;span class=\"page-line\"&gt;1,3&lt;/span&gt;&lt;span␣
↪class=\"word-range\"&gt;annotation&lt;/span&gt;",
                "content": "This is what an example annotation can look like."
            }
        }
    ]
}
```

CHAPTER 3

Styling

The TEI Reader comes with only the minimal CSS styling that defines the interface layout. This section documents the HTML structure the TEI reader generates and which can be used to style the display.

The basic structure of the TEI Reader is as follows:

```
<div id="tei-reader">
  <nav>
    <!-- Navigation content goes here -->
  </nav>
  <main>
    <!-- Main text content goes here -->
  </main>
  <aside>
    <!-- Annotation content goes here -->
  </aside>
</div>
```

3.1 Text

Any HTML is valid within this area. As mentioned in the *content* documentation, the only requirements are specific attributes to link the text content to the navigation and annotation:

data-heading-id This attribute needs to be set on all elements that are headings that are also configured in the navigation. The values do **not** have to be unique.

data-annotation-id This attribute needs to be set on all elements that are linked to an annotation. The values do **not** have to be unique.

Full example:

```
<h1 data-heading-id="welcome">Welcome</h1>
<p>Here is an <a data-annotation-id="1">annotation</a> that is shown
  on the right, if the user clicks on the link.</p>
```

(continues on next page)

(continued from previous page)

```
<h2 data-heading-id="chapter-1">Chapter 1</h2>
<h3 data-heading-id="chapter-1">A useful chapter</h3>
```

3.2 Navigation

The navigation area HTML uses the main structure:

```
<nav>
  <ul>
    <!-- Individual menu items goe here -->
  </ul>
</nav>
```

Individual navigation elements uses the following structure:

```
<li>
  <a>
    <span class="level-X">Label text</span>
  </a>
</li>
```

When two (or more) consecutive heading elements are merged into a single `<li><a>` element, with separate `<span>` elements for each merged heading elements:

```
<li>
  <a>
    <span class="level-X">Label text</span>
    <span class="level-Y">Other label text</span>
  </a>
</li>
```

As the user reads and scrolls through the text, the current heading is highlighted by applying the `aria-current="true"` attribute:

```
<li>
  <a aria-current="true">
    <span class="level-X">Current label text</span>
  </a>
</li>
```

Full example:

```
<nav>
  <ul>
    <li>
      <a>
        <span class="level-1">Welcome</span>
      </a>
    </li>
    <li>
    <li>
      <a aria-current="true">
        <span class="level-2">Chapter 1</span>
        <span class="level-3">A useful chapter</span>
```

(continues on next page)

(continued from previous page)

```
      </a>
    </li>
  <ul>
</nav>
```

3.2.1 Small devices

On small devices the navigation layout changes and moves the navigation to the top of the screen. Only the current navigation entry is shown by default. To see the other items, the user must click on the menu icon. The basic structure is as follows:

```
<nav>
  <div class="menu-toggle">
    <svg viewBox="0 0 24 24">
      <path d="..."/>
    </svg>
  </div>
  <div class="menu-container">
    <ul>
      <!-- Just the current navigation entry as described above -->
    </ul>
  </div>
</nav>
```

After the user clicks on the menu icon, the HTML structure is updated to the following:

```
<nav>
  <div class="menu-toggle">
    <svg viewBox="0 0 24 24">
      <path d="..."/>
    </svg>
  </div>
  <div class="menu-container">
    <ul>
      <!-- Just the current navigation entry as described above -->
    </ul>
    <div class="menu-overlay">
      <div class="full-menu">
        <ul>
          <-- Full list of navigation entries as above -->
        </ul>
      </div>
    </div>
  </div>
</nav>
```

3.3 Annotations

The main annotation structure is very simple:

```
<aside>
  <dl>
```

(continues on next page)

(continued from previous page)

```
    <!-- Annotations go here -->
  </dl>
</aside>
```

Each annotation is structured as follows:

```
<dt>
  <div class="title">
    <!-- Annotation title -->
  </div>
  <div class="action">
    <a>
      <svg viewBox="0 0 24 24">
        <path d="..."/>
      </svg>
    </a>
  </div>
</dt>
<dd>
  <!-- Annotation content -->
</dd>
```

The `<div class="action">` contains an `<a>` action, which when clicked by the user hides the annotation. In the structure the annotation title and content can be any valid HTML.

Full example:

```
<aside>
  <dl>
    <dt>
      <span class="page-line">1,3</span><span class="word-range">annotation</span>
      <div class="action">
        <a>
          <svg viewBox="0 0 24 24">
            <path d="..."/>
          </svg>
        </a>
      </div>
    </dt>
    <dd>This is what an example annotation can look like.</dd>
  </dl>
</aside>
```

3.3.1 Small devices

There is no difference in the generated HTML structure between large and small device sizes. However, for small devices, annotations are shown at the bottom of the display, only one annotation is visible at any time, and the annotation HTML is only generated when an annotation has been selected by the user.

CHAPTER 4

Development

The documentation in this section is aimed at developers wishing to extend/customise/bugfix the TEI Reader.

The TEI Reader is developed in TypeScript using Glimmer JS as the underlying framework.

The source-code is available here: https://gitlab.informatik.uni-halle.de/amsvu/tei-reader

The JavaScript dependencies are handled via Yarn.

The documentation uses Sphinx and all dependencies must be installed using Pipenv.

The code is also fully commented and any changes **must** be documented to be merged into the repository.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search