Pubhub Reader Component (Web)

Installation

This component is built on strict TypeScript 3.8, and should be installed using NPM or Yarn.

The package is found in our GitLab Package Registry and a token is required to access it. The registry is set up in `.npmrc`:

```
@publizon:registry=https://gitlab.com/api/v4/projects/30187012/packages/npm/
//gitlab.com/api/v4/projects/30187012/packages/npm/:_authToken=<your_token>
```

After which the `@publizon/pubhub-reader` package can be installed through yarn or npm

```
yarn add @publizon/pubhub-reader
```

We highly recommend that you use TypeScript when working with the Pubhub Reader to get full typing support.

Check out the Pubhub Reader Web app for a complete example of installation and usage.

Usage

The component is used by creating a single instance of `PubhubReader` which contains methods for loading publications, managing options, attaching various event listeners and many other things.

If you embed the reader directly in a single page application, please ensure that `PubhubReader.unloadAll()` is called when closing the reader.

Configuration

`PubhubReader` takes an optional `Partial<IReaderConfiguration>` argument which allows customizing loading text, highlight colors, etc.

```
const pubhubReader = new PubhubReader({
    contentOnLoading: "<h1>Loading...</h1>",
    highlightColors: {
        [HighlightColor.blue]: "#06b5e1",
        ...
    }
    ...
});
```

After creating the reader, it must be attached to a view container, in which the reader will render the book content. It is up to you to set the width and height of this element. The reader will not expand/style this element:

```
const viewElement = document.getElementById("reader-view");
pubhubReader.renderTo(viewElement);
```

Loading a publication

The publication is loaded using `pubhubReader.loadByOrderId()`.

To load a sample, use `pubhubReader.loadSample()`.

Please note that bookmarks, reading position and annotations are not saved in sample mode, and should not be enabled in the UI.

Options

Options can be retrieved using `pubhubReader.options.getOptions()`. This returns an element containing the current options as well as a list of supported options. Only reflowable EPUBs support options such as font size, line height and palettes, so these options should not be shown when reading PDFs and fixed format EPUBs.

Options can be set using `pubhubReader.options.setOptions`, which takes a `Partial<IReaderOptions>` argument. All undefined options remain unchanged, and `palette` may be reset to publication default by passing a `null` value.

These options are stored in LocalStorage by the reader, and are loaded automatically next time the reader is opened.

The `optionsChanged` event may be used to track options changes, e.g. to update the UI when the reader loads stored options.

Navigation

Navigation collections are retrieved using `pubhubReader.navigation.getNavigationCollections()`. This list contains all the publication's navigation collections such as TOC, page list and landmarks.

Navigation items contain a `selector` property, which defines the specific location in the publication that the item points to.

To go to a specific selector, use `pubhubReader.navigation.goToSelector(selector)`, and to go to next/previous page use `pubhubReader.navigation.nextPage()` / `pubhubReader.navigation.previousPage()`.

Markings

`pubhubReader.markings` contains methods to add, remove and toggle bookmarks, and to retrieve a list of all bookmarks.

The `visibleMarkingsChanged` event may be used to update the UI when user navigates to a page that contains markings.

Annotations

`pubhubReader.annotations` contains methods to add, remove and toggle annotations/highlights, and to retrieve a list of all annotations. Custom annotation highlight colors may be defined in the reader's `Partial<IReaderConfiguration>`.

A `selectionChanged` event is emitted when the user selects or deselects text or images. This event's `selector` value must be used when creating an annotation using `pubhubReader.annotations.setAnnotation()`. The `highlightColor` argument must be a numeric value of a `HighlightColor`. Optionally a note can be saved with the annotation.

When calling `setAnnotation` multiple times using the same selector, the annotation is updated.

The reader supports a few other annotation events that may be used: `annotationClick`, `annotationPointerEnter`, `annotationPointerLeave`.

User interaction events

The reader embeds EPUB/PDF pages in iframes, and pointer and keyboard events in iframes are not bubbled to the parent window.

This means that you're not able to detect these events when the book is focused, e.g. a keyboard arrow-key event won't be detectable in your app after the user clicks on the book.

To help this, the reader contains multiple events that may be used to improve user interaction: `click`, `pointerdown`, `pointerup`, `keydown`, `keyup`.

Confidentiality

The SDK (including as integrated in, or utilized by any Application) is the confidential and proprietary information of Publizon, and you may not, during the term or thereafter, disclose them to any third party, or to use them for any purpose other than as expressly provided herein, without a separate written agreement with Publizon authorizing you to do so.