**docxjs**

Docx rendering library

Demo - [**https://volodymyrbaydalka.github.io/docxjs/**](https://volodymyrbaydalka.github.io/docxjs/)

**Goal**

Goal of this project is to render/convert DOCX document into HTML document with keeping HTML semantic as much as possible. That means library is limited by HTML capabilities (for example Google Docs renders \*.docx document on canvas as an image).

**Installation**

npm install docx-preview

**Usage**

<!--lib uses jszip-->

<script src="https://unpkg.com/jszip/dist/jszip.min.js"></script>

<script src="docx-preview.min.js"></script>

<script>

var docData = <document Blob>;

docx.renderAsync(docData, document.getElementById("container"))

.then(x => console.log("docx: finished"));

</script>

<body>

...

<div id="container"></div>

...

</body>

**API**

// renders document into specified element

renderAsync(

document: Blob | ArrayBuffer | Uint8Array, // could be any type that supported by JSZip.loadAsync

bodyContainer: HTMLElement, //element to render document content,

styleContainer: HTMLElement, //element to render document styles, numbeings, fonts. If null, bodyContainer will be used.

options: {

className: string = "docx", //class name/prefix for default and document style classes

inWrapper: boolean = true, //enables rendering of wrapper around document content

ignoreWidth: boolean = false, //disables rendering width of page

ignoreHeight: boolean = false, //disables rendering height of page

ignoreFonts: boolean = false, //disables fonts rendering

breakPages: boolean = true, //enables page breaking on page breaks

ignoreLastRenderedPageBreak: boolean = true, //disables page breaking on lastRenderedPageBreak elements

experimental: boolean = false, //enables experimental features (tab stops calculation)

trimXmlDeclaration: boolean = true, //if true, xml declaration will be removed from xml documents before parsing

useBase64URL: boolean = false, //if true, images, fonts, etc. will be converted to base 64 URL, otherwise URL.createObjectURL is used

renderChanges: false, //enables experimental rendering of document changes (inserions/deletions)

renderHeaders: true, //enables headers rendering

renderFooters: true, //enables footers rendering

renderFootnotes: true, //enables footnotes rendering

renderEndnotes: true, //enables endnotes rendering

renderComments: false, //enables experimental comments rendering

debug: boolean = false, //enables additional logging

}): Promise<WordDocument>

/// ==== experimental / internal API ===

// this API could be used to modify document before rendering

// renderAsync = praseAsync + renderDocument

// parse document and return internal document object

praseAsync(

document: Blob | ArrayBuffer | Uint8Array,

options: Options

): Promise<WordDocument>

// render internal document object into specified container

renderDocument(

wordDocument: WordDocument,

bodyContainer: HTMLElement,

styleContainer: HTMLElement,

options: Options

): Promise<void>

**Thumbnails, TOC and etc.**

Thumbnails is added only for example and it's not part of library. Library renders DOCX into HTML, so it can't be efficiently used for thumbnails.

Table of contents is built using the TOC fields and there is no efficient way to get table of contents at this point, since fields is not supported yet ([**http://officeopenxml.com/WPtableOfContents.php**](http://officeopenxml.com/WPtableOfContents.php))

**Breaks**

Currently library does break pages:

* if user/manual page break <w:br w:type="page"/> is inserted - when user insert page break
* if application page break <w:lastRenderedPageBreak/> is inserted - could be inserted by editor application like MS word (ignoreLastRenderedPageBreak should be set to false)
* if page settings for paragraph is changed - ex: user change settings from portrait to landscape page

Realtime page breaking is not implemented because it's requires re-calculation of sizes on each insertion and that could affect performance a lot.

If page breaking is crutual for you, I would recommend:

* try to insert manual break point as much as you could
* try use editors like MS Word, that inserts <w:lastRenderedPageBreak/> break points

NOTE: by default ignoreLastRenderedPageBreak is set to true. You may need to set it to false, to make library break by <w:lastRenderedPageBreak/> break points

**Status and stability**

So far I can't come up with final approach of parsing documents and final structure of API. Only **renderAsync** function is stable and definition shouldn't be changed in future. Inner implementation of parsing and rendering may be changed at any point of time.

**Readme**

**Keywords**

* [**word**](https://www.npmjs.com/search?q=keywords:word)
* [**docx**](https://www.npmjs.com/search?q=keywords:docx)