

```

1 // @ts-nocheck
2
3 // MIT License
4
5 // Copyright (c) 2021 Pine Wu
6
7 // Permission is hereby granted, free of charge, to any person obtaining a copy
8 // of this software and associated documentation files (the "Software"), to dea
9 // in the Software without restriction, including without limitation the rights
10 // to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 // copies of the Software, and to permit persons to whom the Software is
12 // furnished to do so, subject to the following conditions:
13
14 // The above copyright notice and this permission notice shall be included in a
15 // copies or substantial portions of the Software.
16
17 // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 // IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 // FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 // AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 // LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
22 // OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN TH
23 // SOFTWARE.
24
25 // @filename: renderer.ts
26 // source: https://github.com/shikijs/shiki/blob/ca5a6c60a255fb390776ada849e01e
27 // 1744a916fc/packages/shiki/src/renderer.ts
28 import { FontStyle } from './stackElementMetadata'
29 import { IThemedToken } from './themedTokenizer'
30
31 export interface HtmlRendererOptions {
32   langId?: string
33   fg?: string
34   bg?: string
35 }
36
37 export function renderToHtml(
38   lines: IThemedToken[][],
39   options: HtmlRendererOptions = {}
40 ) {
41   const bg = options.bg || '#fff'
42
43   let html = ''
44
45   html += `

```
${options.langId}</div>`
48   }
49   html += `

```


```

```

56     cssDeclarations.push('font-style: italic')
57   }
58   if (token.fontStyle & FontStyle.Bold) {
59     cssDeclarations.push('font-weight: bold')
60   }
61   if (token.fontStyle & FontStyle.Underline) {
62     cssDeclarations.push('text-decoration: underline')
63   }
64   html += `

```

```

115 let _themes: IThemeRegistration[] = options.themes || []
116
117 if (options.langs?.length) {
118     _languages = options.langs.map(resolveLang)
119 }
120 if (options.theme) {
121     _themes.unshift(options.theme)
122 }
123 if (!_themes.length) {
124     _themes = ['nord']
125 }
126
127 return { _languages, _themes }
128 }
129
130 export async function getHighlighter(
131     options: HighlighterOptions
132 ): Promise<Highlighter> {
133     const { _languages, _themes } = resolveOptions(options)
134     const _resolver = new Resolver(getOnigasm(), 'onigasm')
135     const _registry = new Registry(_resolver)
136
137     if (options.paths?.themes) {
138         _registry.themesPath = options.paths.themes
139     }
140
141     if (options.paths?.languages) {
142         _resolver.languagesPath = options.paths.languages
143     }
144
145     const themes = await _registry.loadThemes(_themes)
146     const _defaultTheme = themes[0]
147     let _currentTheme: IShikiTheme | undefined
148     await _registry.loadLanguages(_languages)
149
150     /**
151      * Shiki was designed for VS Code, so CSS variables are not currently support
152      * See: https://github.com/shikijs/shiki/pull/212#issuecomment-906924986
153      *
154      * Instead, we work around this by using valid hex color codes as lookups in
155      * a
156      * final "repair" step which translates those codes to the correct CSS variab
157      * les.
158      */
159     const COLOR_REPLACEMENTS: Record<string, string> = {
160         '#000001': 'var(--shiki-color-text)',
161         '#000002': 'var(--shiki-color-background)',
162         '#000004': 'var(--shiki-token-constant)',
163         '#000005': 'var(--shiki-token-string)',
164         '#000006': 'var(--shiki-token-comment)',
165         '#000007': 'var(--shiki-token-keyword)',
166         '#000008': 'var(--shiki-token-parameter)',
167         '#000009': 'var(--shiki-token-function)',
168         '#000010': 'var(--shiki-token-string-expression)',
169         '#000011': 'var(--shiki-token-punctuation)',
170         '#000012': 'var(--shiki-token-link)',
171     }
172
173     function fixCssVariablesTheme(theme: IShikiTheme, colorMap: string[]) {

```

```

172 theme.bg = COLOR_REPLACEMENTS[theme.bg] || theme.bg
173 theme.fg = COLOR_REPLACEMENTS[theme.fg] || theme.fg
174 colorMap.forEach((val, i) => {
175     colorMap[i] = COLOR_REPLACEMENTS[val] || val
176 })
177 }
178
179 function getTheme(theme?: IThemeRegistration) {
180     const _theme = theme ? _registry.getTheme(theme) : _defaultTheme
181     if (!_theme) {
182         throw Error(`No theme registration for ${theme}`)
183     }
184     if (!_currentTheme || _currentTheme.name !== _theme.name) {
185         _registry.setTheme(_theme)
186         _currentTheme = _theme
187     }
188     const _colorMap = _registry.getColorMap()
189     if (_theme.name === 'css-variables') {
190         fixCssVariablesTheme(_theme, _colorMap)
191     }
192     return { _theme, _colorMap }
193 }
194
195 function getGrammar(lang: string) {
196     const _grammar = _registry.getGrammar(lang)
197     if (!_grammar) {
198         throw Error(`No language registration for ${lang}`)
199     }
200     return { _grammar }
201 }
202
203 function codeToThemedTokens(
204     code: string,
205     lang = 'text',
206     theme?: StringLiteralUnion<Theme>,
207     options = { includeExplanation: true }
208 ) {
209     if (isPlainText(lang)) {
210         return [{ content: code }]
211     }
212     const { _grammar } = getGrammar(lang)
213     const { _theme, _colorMap } = getTheme(theme)
214     return tokenizeWithTheme(_theme, _colorMap, code, _grammar, options)
215 }
216
217 function codeToHtml(
218     code: string,
219     lang = 'text',
220     theme?: StringLiteralUnion<Theme>
221 ) {
222     const tokens = codeToThemedTokens(code, lang, theme, {
223         includeExplanation: false,
224     })
225     const { _theme } = getTheme(theme)
226     return renderToHtml(tokens, {
227         fg: _theme.fg,
228         bg: _theme.bg,
229     })
230 }
231

```

[illegible]

283