FastHTML

Source > Formatting Components

Formatting Components

To expedite fast development, FastHTML comes with several built-in Javascript and formatting components.

light_media

source

```
light_media (css:str)
```

Render light media for day mode views

	Type	Details
css	str	CSS to be included in the light media query

```
light_media('.body {color: green;}')
```

```
<style>@media (prefers-color-scheme: light) {.body {color: green;}}</style>
```

dark_media

source

```
dark_media (css:str)
```

Render dark media for nught mode views

	Type	Details
css	str	CSS to be included in the dark media query

```
dark_media('.body {color: white;}')
```

```
<style>@media (prefers-color-scheme:  dark) {.body {color: white;}}</style>
```

MarkdownJS

source

```
MarkdownJS (sel='.marked')
```

Implements browser-based markdown rendering.

	Type	Default	Details
sel	str	.marked	CSS selector for markdown elements

Usage example here.

```
__file__ = '../../fasthtml/katex.js'
```

KatexMarkdownJS

source

```
KatexMarkdownJS (sel='.marked', inline_delim='$', display_delim='$$',
                 math_envs=None)
```

	Type	Default	Details
sel	str	.marked	CSS selector for markdown elements
inline_delim	str	$	Delimiter for inline math
display_delim	str	$$	Delimiter for long math
math_envs	NoneType	None	List of environments to render as display math

```
KatexMarkdownJS()[0]
```

```
<script type="module">import { marked } from "https://cdn.jsdelivr.net/npm/marked/lib/ma
import { proc_htmx } from "https://cdn.jsdelivr.net/gh/answerdotai/fasthtml-js/fasthtml.
import katex from "https://cdn.jsdelivr.net/npm/katex/dist/katex.mjs";

const renderMath = (tex, displayMode) => { return katex.renderToString(tex, {
    throwOnError: false, displayMode: displayMode, output: 'html', trust: true
}) };

const processLatexEnvironments = (content) => {
    return content.replace(/\\begin{(\w+)}([\s\S]*?)\\end{\1}/g, (match, env, innerConte
        if ([['equation','align','gather','multline']].includes(env)) { return `\$\$${ma
        return match;
}) };

proc_htmx('.marked', e => {
```

```
    let content = processLatexEnvironments(e.textContent);
    // Display math (including environments)
    content = content.replace(/\$\$([\s\S]+?)\$\$/gm, (_, tex) => renderMath(tex.trim(),
    // Inline math
    content = content.replace(/(?<!\w)\$([^\$\s](?:[^\$]*[^\$\s])?)\$(?!\w)/g, (_, tex)
    e.innerHTML = marked.parse(content);
});

</script>
```

KatexMarkdown usage example:

```
longexample = r"""
Long example:

$$\begin{array}{c}

\nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} 
= \frac{4\pi}{c}\vec{\mathbf{j}}    \nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\

\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t

\nabla \cdot \vec{\mathbf{B}} & = 0

\end{array}$$
"""

app, rt = fast_app(hdrs=[KatexMarkdownJS()])

@rt('/')
def get():
    return Titled("Katex Examples",
        # Assigning 'marked' class to components renders content as markdown
        P(cls='marked')("Inline example: $\sqrt{3x-1}+(1+x)^2$"),
        Div(cls='marked')(longexample)
    )
```

HighlightJS

source

```
HighlightJS (sel='pre code', langs:str|list|tuple='python', light='atom-
             one-light', dark='atom-one-dark')
```

Implements browser-based syntax highlighting. Usage example here.

	Type	Default	Details
sel	str	pre code	CSS selector for code elements. Default is industry standard, be careful before adjusting it
langs	str | list | tuple	python	Language(s) to highlight
light	str	atom-one-light	Light theme
dark	str	atom-one-dark	Dark theme

SortableJS

source

```
SortableJS (sel='.sortable', ghost_class='blue-background-class')
```

	Type	Default	Details
sel	str	.sortable	CSS selector for sortable elements
ghost_class	str	blue-background-class	When an element is being dragged, this is the class used to distinguish it from the rest

Report an issue