@push.rocks/smartmustache

A library for creating and applying templates using Handlebars.

- readme.md for @push.rocks/smartmustache

readme.md for @push.rocks/smartmustache

templates done right

Install

To install `@push.rocks/smartmustache`, use npm:

```
npm install @push.rocks/smartmustache
```

Ensure you have Node.js and npm installed on your system before running the install command.

Usage

To use `@push.rocks/smartmustache` in your project, follow these examples to understand how to integrate and utilize the library with its core functionalities. `@push.rocks/smartmustache` leverages Handlebars for templating, providing a seamless and efficient way to render dynamic content.

Setting Up Your Project

First, ensure your project is set up to use ECMAScript modules (ESM) and TypeScript. Your `tsconfig.json` should include:

```
{
  "compilerOptions": {
    "module": "ESNext",
    "target": "es2020",
    "moduleResolution": "node",
    "esModuleInterop": true,
    "allowSyntheticDefaultImports": true
  }
```

```
}
```

This setup will allow you to use ESM syntax in your TypeScript files efficiently.

Basic Templating

Let's start with a basic example of using `@push.rocks/smartmustache` to apply data to a template string.

1. **Create a New Template Instance:**

First, import `SmartMustache` and create an instance by passing a template string to its constructor.

```
import { SmartMustache } from '@push.rocks/smartmustache';

const template = 'Hello, {{name}}! Welcome to {{location}}.';
const myTemplate = new SmartMustache(template);
```

2. **Apply Data to the Template:**

Next, apply data to your template using the `applyData` method.

```
const result = myTemplate.applyData({
  name: 'John Doe',
  location: 'SmartMustache World'
});

console.log(result); // Outputs: Hello, John Doe! Welcome to SmartMustache World.
```

Advanced Usage

Conditional Statements and Loops:

Handlebars syntax allows for conditionals and loops. `@push.rocks/smartmustache` fully supports these Handlebars features.

- **Conditionals:**

Consider you have a template that should render content based on certain conditions.

```
const conditionalTemplate = new SmartMustache(`
  {{#if isAdmin}}
    <p>Welcome, admin.</p>
  {{else}}
    <p>Welcome, user.</p>
  {{/if}}
`);

console.log(conditionalTemplate.applyData({ isAdmin: true }));
```

- **Loops:**

Similarly, if you'd like to iterate over an array of items:

```
const loopTemplate = new SmartMustache(`
  <ul>
    {{#each items}}
      <li>{{this}}</li>
    {{/each}}
  </ul>
`);

console.log(loopTemplate.applyData({
  items: ['Item 1', 'Item 2', 'Item 3']
}));
```

Providing Helpers

`@push.rocks/smartmustache` allows integrating Handlebars helpers to extend templating capabilities.

```
import Handlebars from 'handlebars';

// Register a Handlebars helper
Handlebars.registerHelper('uppercase', function(aString) {
  return aString.toUpperCase();
});

const helperTemplate = new SmartMustache('Hello, {{uppercase name}}!');
console.log(helperTemplate.applyData({ name: 'john' })); // Outputs: Hello, JOHN!
```

Reusing Templates

You can reuse an instance of `SmartMustache` by setting a new template string.

```
myTemplate.setTemplate('Goodbye, {{name}}. See you in {{location}}.');
console.log(myTemplate.applyData({
  name: 'Jane Doe',
  location: 'the virtual world'
}));
```

Conclusion

`@push.rocks/smartmustache` provides a powerful and intuitive way to work with templates in your TypeScript projects. By following these guides, you'll be able to implement dynamic content rendering seamlessly. The flexibility of Handlebars within a TypeScript-friendly wrapper accelerates development, ensuring your templates are both manageable and scalable.

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.