readme.md for @push.rocks/smartssr

a smart server side renderer supporting shadow dom

Install

To install `@push.rocks/smartssr`, use the following command with npm:

```
npm install @push.rocks/smartssr --save
```

Or with yarn:

```
yarn add @push.rocks/smartssr
```

This will add `@push.rocks/smartssr` to your project's dependencies and make it available for import in your TypeScript files.

Usage

Introduction

`@push.rocks/smartssr` is a powerful package designed to facilitate server-side rendering with support for shadow DOM, making it an ideal choice for projects where SEO or initial load performance is critical while maintaining the benefits of Web Components. Below, we'll dive deep into how you can leverage `@push.rocks/smartssr` in your TypeScript projects.

Getting Started

Before you can use `@push.rocks/smartssr`, ensure you have a TypeScript environment set up and configured to support ECMAScript modules (ESM).

Setting Up Your Project

1. Initialize your project (if you haven't already):

```
npm init
```

2. Add TypeScript to your project:

```
npm install typescript --save-dev
```

3. Initialize TypeScript in your project:

```
npx tsc --init
```

Make sure your `tsconfig.json` is configured to use ECMAScript modules by setting `"module": "ESNext"`.

Basic Usage

First, import `SmartSSR` from the package:

```
import { SmartSSR } from '@push.rocks/smartssr';
```

Create an instance of `SmartSSR`. You can pass options to the constructor to customize behavior, for instance, enabling debug mode:

```
const smartSSR = new SmartSSR({
  debug: true // Enables debug mode for additional logging
});
```

Rendering a Web Page

With `SmartSSR`, you can render any webpage, including those utilizing shadow DOM, and retrieve its HTML content as a string. This feature is particularly useful for SEO purposes and can be used to pre-render contents of single-page applications (SPAs) or web components.

The `renderPage` method accepts a URL as a parameter and returns a Promise that resolves with the rendered HTML content of the page.

```
(async () => {
  try {
    const renderedHTML = await smartSSR.renderPage('https://example.com');
    console.log(renderedHTML); // Outputs the rendered HTML content
  } catch (error) {
    console.error(`Error rendering page: ${error}`);
  }
})();
```

Advanced Usage

Customizing Rendering Behavior

The `SmartSSR` constructor accepts various options allowing you to customize the behavior of the rendering process. For instance, setting the `debug` option to `true` enables additional logging, which can be helpful for development and debugging purposes.

Handling Complex Scenarios

In more complex scenarios, such as when rendering web applications that rely heavily on dynamic content or client-side scripts, you might need to customize your rendering process further. This could involve manipulating the page or waiting for specific elements or data before considering the rendering process complete.

`smartssr` leverages Puppeteer under the hood, which means you can use any Puppeteer functionality within your rendering logic if needed. For example, you could wait for a specific element to appear on the page before capturing the rendered content:

```
// Custom rendering logic
await page.waitForSelector('#dynamic-content-loaded', { timeout: 5000 });
const dynamicContentHTML = await smartSSR.renderPage('https://dynamic-content-example.com');
```

Conclusion

`@push.rocks/smartssr` offers a streamlined and effective solution for server-side rendering with shadow DOM support, enabling improved SEO and performance for web applications. Whether you're rendering simple static pages or complex SPAs with dynamic content, `smartssr` provides the tools necessary to achieve your objectives efficiently.

For further information and advanced configurations, refer to the official documentation and explore the full capabilities of `@push.rocks/smartssr`.

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.

Revision #3
Created 2026-03-28 11:12:36 UTC by foss.global Team
Updated 2026-03-28 12:19:21 UTC by foss.global Team