readme.md for @push.rocks/smartpuppeteer

simplified access to puppeteer

Install

To install `@push.rocks/smartpuppeteer` in your project, run the following command using npm:

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

Or using yarn:

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

Usage

`@push.rocks/smartpuppeteer` simplifies interaction with Puppeteer, providing easier ways to launch Puppeteer instances considering environment constraints, such as running in a CI pipeline or as root, which necessitates certain flags for Chrome.

Here, we give a comprehensive guide to using `@push.rocks/smartpuppeteer` in various scenarios, using ESM syntax and TypeScript.

Basic Setup

Firstly, let's set up the basic environment for using `@push.rocks/smartpuppeteer`:

```
import { getEnvAwareBrowserInstance, IncognitoBrowser, puppeteer } from
'@push.rocks/smartpuppeteer';

// Usually, you would initialize the browser instance at the start of your script or
application logic
const initializeBrowser = async () => {
  const browser = await getEnvAwareBrowserInstance({
```

```
    forceNoSandbox: true, // A flag useful for certain environments; use it with caution
  });
  return browser;
};
```

Opening a Page and Navigating

After obtaining a browser instance, you commonly want to open a page and navigate to a URL:

```
const openPage = async (browser: puppeteer.Browser) => {
  const page = await browser.newPage();
  await page.goto('https://www.example.com');
  const pageTitle = await page.title();
  console.log(`Page title: ${pageTitle}`);
  // Always close the browser after you are done to free resources
  await browser.close();
};

// Utilize the async function
initializeBrowser()
  .then(openPage)
  .catch(console.error);
```

Using Incognito Mode for Isolated Sessions

`@push.rocks/smartpuppeteer` offers easy management of incognito sessions, allowing isolated environments within the same browser instance:

```
const useIncognitoBrowser = async () => {
  const incognitoBrowser = new IncognitoBrowser();
  await incognitoBrowser.start(); // Initializes a new incognito browser instance
  const context = await incognitoBrowser.getNewIncognitoContext();
  const page = await context.newPage();
  await page.goto('https://www.privacyfocusedsite.com');
  // Perform actions in the isolated session
  // Tidy up
  await incognitoBrowser.stop(); // Stops the incognito browser and closes all its pages and
contexts
};
```

```
useIncognitoBrowser()
  .then(() => console.log('Incognito session used successfully'))
  .catch(console.error);
```

Advanced Configuration

`@push.rocks/smartpuppeteer` allows further customization for launching the Puppeteer browser, such as disabling the sandbox environment (not recommended for production).

Handling Browser Events

It's important to handle browser events, such as disconnections, which might occur due to various reasons:

```
const browserWithEventHandling = async () => {
  const browser = await getEnvAwareBrowserInstance();
  browser.on('disconnected', () => {
    console.log('Browser disconnected. Handling reconnection...');
    // Implement reconnection logic here
  });
  // Utilize the browser for tasks
};

browserWithEventHandling()
  .then(() => console.log('Handled browser events successfully'))
  .catch(console.error);
```

Rotation of Browsers and Pages

In scenarios such as web scraping or automated testing, you might want to rotate between browser instances or pages to manage memory usage or simulate new sessions:

```
const rotateBrowserInstances = async (incognitoBrowser: IncognitoBrowser) => {
  // Assuming incognitoBrowser is already initialized and started
  await incognitoBrowser.rotateBrowser(); // Closes the current browser and starts a new
instance
  // Now you have a fresh browser instance
};
```

```
// Example usage
const incognitoBrowser = new IncognitoBrowser();
incognitoBrowser.start()
  .then(() => rotateBrowserInstances(incognitoBrowser))
  .catch(console.error);
```

`@push.rocks/smartpuppeteer` with its encapsulated features and simplified API provides an efficient way to harness the power of Puppeteer without getting bogged down by its complexities. Whether you are handling web scraping, automated testing, or any task requiring browser automation, `@push.rocks/smartpuppeteer` streamlines the process, making it more accessible and manageable even for those new to Puppeteer.

License and Legal Information

Trademarks

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.

Revision #3
Created 2026-03-28 11:11:41 UTC by foss.global Team
Updated 2026-03-28 12:18:33 UTC by foss.global Team