readme.md for @push.rocks/smartbrowser

A simplified Puppeteer wrapper for easy browser automation, PDF generation, screenshots, and page evaluation.

Install

To install `@push.rocks/smartbrowser`, use the following command with your preferred package manager:

```
npm install @push.rocks/smartbrowser
# or
pnpm install @push.rocks/smartbrowser
```

This module works with Node.js and requires a Chromium-compatible browser available in the environment. It uses `@push.rocks/smartpuppeteer` under the hood, which automatically detects your environment and configures the browser accordingly (including CI/Docker scenarios).

Usage

`@push.rocks/smartbrowser` provides a high-level `SmartBrowser` class that wraps Puppeteer for common browser automation tasks: generating PDFs, capturing screenshots, and evaluating JavaScript on web pages.

Getting Started

Import and initialize a `SmartBrowser` instance:

```
import { SmartBrowser } from '@push.rocks/smartbrowser';

const smartBrowser = new SmartBrowser();
```

```
await smartBrowser.start();
```

Generating a PDF from a Webpage

Generate a full-page PDF from any URL. The result includes a `Buffer` with the PDF contents:

```
const pdfResult = await smartBrowser.pdfFromPage('https://example.com');
console.log(pdfResult.buffer); // PDF file buffer
console.log(pdfResult.name);   // Generated name identifier
```

The PDF generation is powered by `@push.rocks/smartpdf`, which is lazily initialized on first use. This means the SmartPdf server is only started when you actually call `pdfFromPage()`, keeping resource usage minimal.

Capturing a Screenshot of a Webpage

Capture a PNG screenshot of any webpage:

```
const screenshotResult = await smartBrowser.screenshotFromPage('https://example.com');
console.log(screenshotResult.buffer); // Screenshot buffer (PNG)
console.log(screenshotResult.name);   // Short unique identifier
console.log(screenshotResult.id);     // Identifier with extension
```

Evaluating JavaScript on a Webpage

Run arbitrary JavaScript inside a page context and retrieve the result:

```
const pageTitle = await smartBrowser.evaluateOnPage('https://example.com', async () => {
  return document.title;
});
console.log(pageTitle); // "Example Domain"
```

The `evaluateOnPage` method supports generic return types:

```
const metrics = await smartBrowser.evaluateOnPage<{ width: number; height: number }>(
  'https://example.com',
  async () => {
    return {
      width: window.innerWidth,
```

```
      height: window.innerHeight,
    };
  }
);
console.log(metrics.width, metrics.height);
```

Pages are automatically closed after evaluation, even if an error occurs.

Accessing the Underlying Puppeteer Browser

For advanced use cases, you can access the Puppeteer browser instance directly:

```
const page = await smartBrowser.headlessBrowser.newPage();
await page.goto('https://example.com');
// ... custom Puppeteer operations
await page.close();
```

You can also import the `smartpuppeteer` module directly for lower-level browser management:

```
import { smartpuppeteer } from '@push.rocks/smartbrowser';

const browser = await smartpuppeteer.getEnvAwareBrowserInstance();
```

Shutting Down

Always stop the browser instance when done to free resources:

```
await smartBrowser.stop();
```

This cleanly shuts down the SmartPdf server (if it was initialized) and closes the browser.

Full Example

```
import { SmartBrowser } from '@push.rocks/smartbrowser';

async function main() {
```

```
  const smartBrowser = new SmartBrowser();
  await smartBrowser.start();

  // Generate a PDF
  const pdfResult = await smartBrowser.pdfFromPage('https://example.com');
  console.log('PDF size:', pdfResult.buffer.length, 'bytes');

  // Take a screenshot
  const screenshot = await smartBrowser.screenshotFromPage('https://example.com');
  console.log('Screenshot size:', screenshot.buffer.length, 'bytes');

  // Evaluate JavaScript
  const title = await smartBrowser.evaluateOnPage('https://example.com', async () => {
    return document.title;
  });
  console.log('Page title:', title);

  await smartBrowser.stop();
}

main();
```

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 11:10:01 UTC by foss.global Team
Updated 2026-03-28 12:16:39 UTC by foss.global Team