# readme.md for @push.rocks/websetup

setup basic page properties

# Install

To install @push.rocks/websetup, you can use npm (Node Package Manager). Simply run the following command in your terminal:

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

This will download the package and add it to your project's `node_modules` directory, as well as save it as a dependency in your project's `package.json` file.

# Usage

To use @push.rocks/websetup in your web applications to set up basic page properties efficiently, follow these steps and guidelines. The examples are provided using ECMAScript modules (ESM) syntax along with TypeScript for type safety and better tooling support.

First, ensure you import the necessary modules from @push.rocks/websetup in your TypeScript files:

```
// Import the main class `WebSetup` from the package
import { WebSetup } from '@push.rocks/websetup';
```

# Basic Setup

Begin by creating an instance of `WebSetup` with basic page properties. This example demonstrates setting up page title, description, and canonical URL.

```
// Create a WebSetup instance with basic meta information
const webSetup = new WebSetup({
  metaObject: {
    title: 'Example Page Title',
    description: 'This is a description of the example page.',
    canonicalDomain: 'https://www.example.com',
  },
});

// Wait for the setup to complete
webSetup.readyPromise.then(() => {
  console.log('WebSetup is ready.');
});
```

# Advanced Usage

## Setting up Structured Data with JSON-LD

Structured data is crucial for SEO and enhancing your page's appearance in search results. @push.rocks/websetup allows you to easily setup company, product, or news article information using JSON-LD.

```
// Define company information
const companyInfo = {
  name: 'Example Company Inc.',
  contact: {
    website: 'https://www.example.com',
    logoUrl: 'https://www.example.com/logo.png',
    phone: '+1234567890',
    facebookUrl: 'https://facebook.com/example',
    twitterUrl: 'https://twitter.com/example',
  },
};

// Pass the company information to WebSetup
const webSetupWithCompanyInfo = new WebSetup({
  metaObject: {
    title: 'Example Page Title',
    ldCompany: companyInfo,
```

```
  },
});

// For products
const productInfo = {
  name: 'Example Product',
  description: 'A detailed description of the example product.',
  os: 'Web',
  category: 'SaaS',
  logoLink: 'https://www.example.com/product-logo.png',
};

// Pass the product information along with company information for structured data setup
const webSetupWithProductInfo = new WebSetup({
  metaObject: {
    title: 'Product Page Title',
    ldProduct: productInfo,
    ldCompany: companyInfo,
  },
});

// Wait for setup completion
webSetupWithProductInfo.readyPromise.then(() => {
  console.log('WebSetup with product info is ready.');
});
```

## Dynamic Subpage Levels

You might have a web application where different "subpages" or sections require different meta tags (e.g., title, description). You can dynamically set these properties without recreating the WebSetup instance.

```
// Assume `webSetup` is an existing instance of WebSetup

// Define meta object for a new subpage
const subPageMeta = {
  title: 'Subpage Title',
  description: 'Description for the subpage.',
};
```

```
// Dynamically set the subpage level
webSetup.setSubLevel(subPageMeta).then(() => {
  console.log('Subpage meta tags are set.');
});

// Revert to base level when leaving the subpage
webSetup.revertToBaseLevel();
```

This approach allows developers to manage site-wide and subpage-specific properties efficiently, enhancing user experience and SEO.

# Handling Ready for Server-Side Rendering (SSR)

For projects using server-side rendering (SSR), it is essential to signal when the page is ready to be captured. This is especially useful when using SSR solutions like smartSSR that wait for a signal from the client.

```
// Inform that the page is ready for SSR capture
webSetup.informReadyForSmartssr();
```

Note: This feature should be used if you have `smartssrWaitForReadySignal` option enabled during the `WebSetup` initialization.

# Conclusion

The `@push.rocks/websetup` package provides a streamlined way to manage basic page properties, structured data using JSON-LD for SEO, and dynamic settings for different page sections or states. By following the examples above and integrating @push.rocks/websetup into your projects, you can improve your web application's SEO, maintainability, and developer experience.

# License and Legal Information

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 11:13:32 UTC by foss.global Team
Updated 2026-03-28 12:20:17 UTC by foss.global Team