# readme.md for @push.rocks/smartparcel

a wrapper for parcel

## Install

To install `@push.rocks/smartparcel`, you'll need to run the following command using npm (or yarn, or your preferred npm client). Make sure you have Node.js installed on your machine before doing so.

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

or if you are using yarn:

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

This will add `@push.rocks/smartparcel` as a dependency to your project and download it to your `node_modules` folder.

## Usage

To make the most out of `@push.rocks/smartparcel`, you need to understand how to integrate it into your project for bundling your assets using Parcel. This guide will walk you through setting up and using `@push.rocks/smartparcel` effectively in a TypeScript environment.

First, ensure you are working in a TypeScript enabled project setup. This walkthrough assumes you have TypeScript configured and ready to go.

### Step 1: Importing `@push.rocks/smartparcel`

Start by importing the module where you intend to use it. Typically, this would be in your build process file or a specific script designated for handling your asset pipeline.

```
import { Parcel } from '@push.rocks/smartparcel';
```

The key class here is `Parcel`, which is a wrapper around the Parcel bundler, providing you with a simplified and potential custom API tailored for your project needs.

# Step 2: Configuring the Parcel Instance

Before you can use the `Parcel` class to bundle your assets, you need to instantiate it with appropriate configuration. The configuration involves specifying the entry files, the output directory, and the output file name.

```
// Define the configuration
const entryFiles = ['./src/index.html']; // Entry points for Parcel can be a single file or an array of files.
const outputDir = './dist'; // The directory where the output should be stored.
const outputFile = 'bundle.js'; // The name of the output file.

// Create a new Parcel instance
const myParcel = new Parcel(entryFiles, outputDir, outputFile);
```

In this setup, we're telling `@push.rocks/smartparcel` to take an `index.html` file located in the `src` directory, bundle it (alongside any assets it references), and output the results in a `dist` directory with `bundle.js` as the bundled file name.

# Step 3: Building or Watching with Server

## Building Assets

To bundle your assets without watching for changes, you can use the `build` method.

```
(async () => {
  await myParcel.build();
})();
```

This approach is ideal for production builds where you only need to bundle your assets once.

## Developing with Watch and Serve

During development, you might want your assets to be rebuilt automatically upon file changes. Additionally, serving these assets through a local server can be highly beneficial. `@push.rocks/smartparcel` allows you to watch assets and serve them using a single call.

```
(async () => {
  await myParcel.watchAndServe();
})();
```

This method starts a process that watches for file changes and serves the bundled assets on a local server, making development workflows more efficient and streamlined.

# Conclusion

`@push.rocks/smartparcel` offers a simplified, yet powerful, interface for working with Parcel in TypeScript projects. By wrapping the complexity of configuring Parcel into an easy-to-use API, it allows developers to focus more on development rather than build tool configuration. As shown in this guide, setting up and using `@push.rocks/smartparcel` is straightforward, making it an excellent choice for projects looking to leverage the power of Parcel with minimal setup.

Remember, the examples provided here are meant to get you started. Depending on your project's needs, you may need to adjust the configurations. Refer to the official Parcel documentation for more detailed information on the options and capabilities available.

# 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.

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 11:11:36 UTC by foss.global Team
Updated 2026-03-28 12:18:27 UTC by foss.global Team