@push.rocks/smartmanifest

A toolkit for constructing web application manifests with TypeScript support.

- readme.md for @push.rocks/smartmanifest

readme.md for @push.rocks/smartmanifest

a module for creating web app manifests

Install

To start using `@push.rocks/smartmanifest`, first, you need to install it through npm or yarn. Here's how you can do it using npm:

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

Or, if you are using yarn, you can do:

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

This will add `@push.rocks/smartmanifest` to your project dependencies and you're ready to start using it to generate web app manifests with ease.

Usage

The `@push.rocks/smartmanifest` module simplifies the creation of web app manifests, providing a straightforward TypeScript interface to construct and output manifest files, which are essential for modern web applications. Here's how to get started:

Basic Example

Create an instance of `SmartManifest` by importing the module and configuring it with essential manifest properties:

```
// Import the module
import { SmartManifest, ISmartManifestConstructorOptions } from '@push.rocks/smartmanifest';
```

```
// Define manifest options
const manifestOptions: ISmartManifestConstructorOptions = {
  name: 'Your App Name',
  short_name: 'AppName',
  start_url: '/',
  display: 'standalone',
  theme_color: '#000000',
  background_color: '#ffffff',
  icons: [
    {
      src: 'icons/icon-192x192.png',
      sizes: '192x192',
      type: 'image/png'
    },
    {
      src: 'icons/icon-512x512.png',
      sizes: '512x512',
      type: 'image/png'
    }
  ]
};

// Create SmartManifest instance
const yourAppManifest = new SmartManifest(manifestOptions);

// Output the manifest as a JSON string
console.log(yourAppManifest.jsonString());
```

This example shows the creation of a simple web app manifest using `@push.rocks/smartmanifest`. Essential attributes such as `name`, `short_name`, `start_url`, `display`, `theme_color`, `background_color`, and `icons` are specified in the `ISmartManifestConstructorOptions`.

Advanced Configuration

The module also allows for more advanced configuration of the manifest, supporting fields for orientation, related applications, and even custom extensions. For example, setting the manifest to encourage the application to be shown in a fullscreen mode with a specific orientation or setting related applications for app stores:

```
import { SmartManifest, ISmartManifestConstructorOptions } from '@push.rocks/smartmanifest';

const advancedManifestOptions: ISmartManifestConstructorOptions = {
  name: 'Advanced App',
  short_name: 'AdvApp',
  display: 'fullscreen',
  orientation: 'landscape',
  related_applications: [
    {
      platform: 'play',
      id: 'com.example.app'
    }
  ],
  // Further customization here
};

const advancedAppManifest = new SmartManifest(advancedManifestOptions);

console.log(advancedAppManifest.jsonString());
```

Working with Icons

The icons field in the manifest is particularly important for ensuring your application has a compelling presence on the user's home screen and within task switchers across various platforms. `@push.rocks/smartmanifest` allows you to specify multiple icons, different dimensions, and purposes (for example, any, maskable, or monochrome).

Using the Output

Once you have generated the JSON string representation of your web app's manifest, you can write this to a `manifest.json` file at the root of your web project or serve it dynamically via a web server. Ensuring the manifest is correctly linked within your application's `<head>` section with `<link rel="manifest" href="/manifest.json">` is crucial for the web app to be recognized as such by browsers and platforms.

Conclusion

By leveraging `@push.rocks/smartmanifest`, developers can streamline the generation of web app manifests, ensuring their applications meet the criteria for progressive web apps (PWAs) and providing a superior user experience across devices and platforms. The use of TypeScript for configuration encapsulates the complexity of manifest generation, making the process more intuitive and error-free.

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.