@design.estate/dees-comms

a comms module for locally communicating cross DOM, workers, tabs within a domain scope.

- readme.md for @design.estate/dees-comms
- changelog.md for @design.estate/dees-comms

readme.md for @design.estate/dees-comms

a comms module for communicating within the DOM

Install

To integrate `@design.estate/dees-comms` into your project, run the following command in your terminal:

```
npm install @design.estate/dees-comms --save
```

This will add the package to your project dependencies.

Usage

`@design.estate/dees-comms` facilitates communication within the DOM, especially between various workers and tabs in client-side JavaScript. Using TypeScript and the provided ESM syntax will enhance your development experience with better intellisense support and type safety.

Setting Up

First, make sure to import the module in your TypeScript file:

```
import { DeesComms } from '@design.estate/dees-comms';
```

Initializing the Communication Module

Create an instance of the `DeesComms` class to start using the module. This instance will act as the central communication hub in your application.

```
const myDeesComms = new DeesComms();
```

Sending Messages

To send messages, you'll need to create a typed request. Let's assume we are sending a message that requires a response:

```
// Define the message type
interface MyMessageType {
  method: 'myMethod';
  someData: string;
}

// Creating a typed request
const myTypedRequest = myDeesComms.createTypedRequest<MyMessageType>('myMethod');

// Firing the request
myTypedRequest.fire({
  someData: 'Hello, World!',
}).then(response => {
  console.log('Received response:', response);
});
```

Receiving and Handling Messages

To handle incoming messages, you need to define handlers for the types of messages you expect to receive.

```
// Define the handler for the 'myMethod' message
myDeesComms.createTypedHandler<MyMessageType>('myMethod', async (incomingMessage) => {
  console.log('Message received:', incomingMessage);

  // Process the incoming message...

  // Return a response (if needed)
  return {
    response: 'Message received loud and clear!'
  };
```

```
});
```

Communication Between Tabs or Workers

`@design.estate/dees-comms` uses the Broadcast Channel API and a polyfill for environments where it's not available, enabling seamless communication between different contexts like tabs, iframes, or web workers.

Clean Up

When you're done with communication, or before creating a new instance for any reason, ensure you clean up properly to prevent memory leaks or other unexpected behaviors:

```
// Currently, the cleanup should be handled according to the specific needs of your application
// e.g., closing or nullifying references to the DeesComms instance.
```

Complete Example

Putting it all together, let's create a small application that sends a message and listens for a response:

```
import { DeesComms } from '@design.estate/dees-comms';

const appDeesComms = new DeesComms();

// Setting up the listener
appDeesComms.createTypedHandler<{ method: 'greet'; name: string }>('greet', async (message) => {
  console.log(`Greetings received from ${message.name}`);
  return { reply: `Hello ${message.name}, nice to meet you!` };
});

// Sending a greeting message
const greetRequest = appDeesComms.createTypedRequest<{ method: 'greet'; name: string }>('greet');
greetRequest.fire({ name: 'Alice' }).then(response => {
  console.log(response.reply); // Expected output: "Hello Alice, nice to meet you!"
```

```
});
```

This module provides a powerful and flexible way to handle communications within the DOM, leveraging modern JavaScript features and patterns. The usage of TypeScript ensures type safety and enhances code understandability and maintainability.

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.

changelog.md for @design.estate/dees-comms

2025-12-04 - 1.0.30 - fix(DeesComms)

Skip routing hooks for BroadcastChannel-received messages to prevent infinite loops; update typedrequest dependency and bump package version.

- Skip hooks when routing responses for messages received from the BroadcastChannel (prevents infinite loops when global traffic-logging hooks are present).
- Bump dependency @api.global/typedrequest to ^3.2.3.
- Bump package version to 1.0.29.

2025-12-04 - 1.0.28 - fix(build)

Upgrade build/test tooling and runtime deps; enable verbose tests

- Change test script to run tstest with --verbose instead of --web
- Upgrade devDependencies: @git.zone/tsbuild -> ^3.1.2, @git.zone/tsbundle -> ^2.6.3, @git.zone/tsrun -> ^2.0.0, @git.zone/tstest -> ^3.1.3, @push.rocks/tapbundle -> ^6.0.3, @types/node -> ^24.10.1
- Bump runtime dependencies: @api.global/typedrequest -> ^3.2.2, broadcast-channel -> ^7.2.0
- No source code changes outside package.json

2024-05-25 - 1.0.27 - maintenance (1.0.2 → 1.0.27)

Consolidated maintenance releases covering many small patches, core fixes, and version bumps between 2020-10-06 and 2024-05-25.

- Aggregated numerous "fix(core): update" changes applied across multiple patch releases.
- Many releases were version bumps or had only minimal/internal changes with no user-facing features.
- No breaking API changes recorded in these patch releases.

2024-04-20 - 1.0.24 - docs

Update documentation.

- Updated project documentation.