Diagnostics Channel

Stability: 1 - Experimental

The `diagnostics_channel` module provides an API to create named channels to report arbitrary message data for diagnostics purposes.

It can be accessed using:

```
import diagnostics_channel from 'diagnostics_channel';
```

```
const diagnostics_channel = require('diagnostics_channel');
```

It is intended that a module writer wanting to report diagnostics messages will create one or many top-level channels to report messages through. Channels may also be acquired at runtime but it is not encouraged due to the additional overhead of doing so. Channels may be exported for convenience, but as long as the name is known it can be acquired anywhere.

If you intend for your module to produce diagnostics data for others to consume it is recommended that you include documentation of what named channels are used along with the shape of the message data. Channel names should generally include the module name to avoid collisions with data from other modules.

Public API

Overview

Following is a simple overview of the public API.

```
import diagnostics_channel from 'diagnostics_channel';

// Get a reusable channel object
const channel = diagnostics_channel.channel('my-channel');

// Subscribe to the channel
channel.subscribe((message, name) => {
  // Received data
});

// Check if the channel has an active subscriber
if (channel.hasSubscribers) {
  // Publish data to the channel
  channel.publish({
    some: 'data'
  });
}
```

```
const diagnostics_channel = require('diagnostics_channel');
```

```
// Get a reusable channel object
const channel = diagnostics_channel.channel('my-channel');

// Subscribe to the channel
channel.subscribe((message, name) => {
  // Received data
});

// Check if the channel has an active subscriber
if (channel.hasSubscribers) {
  // Publish data to the channel
  channel.publish({
    some: 'data'
  });
}
```

`diagnostics_channel.hasSubscribers(name)`

- `name` {string|symbol} The channel name
- Returns: {boolean} If there are active subscribers

Check if there are active subscribers to the named channel. This is helpful if the message you want to send might be expensive to prepare.

This API is optional but helpful when trying to publish messages from very performance-sensitive code.

```
import diagnostics_channel from 'diagnostics_channel';

if (diagnostics_channel.hasSubscribers('my-channel')) {
  // There are subscribers, prepare and publish message
}
```

```
const diagnostics_channel = require('diagnostics_channel');

if (diagnostics_channel.hasSubscribers('my-channel')) {
  // There are subscribers, prepare and publish message
}
```

`diagnostics_channel.channel(name)`

- `name` {string|symbol} The channel name
- Returns: {Channel} The named channel object

This is the primary entry-point for anyone wanting to interact with a named channel. It produces a channel object which is optimized to reduce overhead at publish time as much as possible.

```
import diagnostics_channel from 'diagnostics_channel';

const channel = diagnostics_channel.channel('my-channel');
```

```
const diagnostics_channel = require('diagnostics_channel');

const channel = diagnostics_channel.channel('my-channel');
```

Class: `Channel`

The class `Channel` represents an individual named channel within the data pipeline. It is use to track subscribers and to publish messages when there are subscribers present. It exists as a separate object to avoid channel lookups at publish time, enabling very fast publish speeds and allowing for heavy use while incurring very minimal cost. Channels are created with `diagnostics_channel.channel(name)`, constructing a channel directly with `new Channel(name)` is not supported.

`channel.hasSubscribers`

- Returns: {boolean} If there are active subscribers

Check if there are active subscribers to this channel. This is helpful if the message you want to send might be expensive to prepare.

This API is optional but helpful when trying to publish messages from very performance-sensitive code.

```
import diagnostics_channel from 'diagnostics_channel';

const channel = diagnostics_channel.channel('my-channel');

if (channel.hasSubscribers) {
  // There are subscribers, prepare and publish message
}
```

```
const diagnostics_channel = require('diagnostics_channel');

const channel = diagnostics_channel.channel('my-channel');

if (channel.hasSubscribers) {
  // There are subscribers, prepare and publish message
}
```

`channel.publish(message)`

- `message` {any} The message to send to the channel subscribers

Publish a message to any subscribers to the channel. This will trigger message handlers synchronously so they will execute within the same context.

```
import diagnostics_channel from 'diagnostics_channel';

const channel = diagnostics_channel.channel('my-channel');

channel.publish({
  some: 'message'
});
```

```
const diagnostics_channel = require('diagnostics_channel');

const channel = diagnostics_channel.channel('my-channel');

channel.publish({
  some: 'message'
});
```

`channel.subscribe(onMessage)`

- `onMessage` {Function} The handler to receive channel messages
  - `message` {any} The message data
  - `name` {string|symbol} The name of the channel

Register a message handler to subscribe to this channel. This message handler will be run synchronously whenever a message is published to the channel. Any errors thrown in the message handler will trigger an `'uncaughtException'`.

```
import diagnostics_channel from 'diagnostics_channel';

const channel = diagnostics_channel.channel('my-channel');

channel.subscribe((message, name) => {
  // Received data
});
```

```
const diagnostics_channel = require('diagnostics_channel');

const channel = diagnostics_channel.channel('my-channel');

channel.subscribe((message, name) => {
  // Received data
});
```

`channel.unsubscribe(onMessage)`

- `onMessage` {Function} The previous subscribed handler to remove
- Returns: {boolean} `true` if the handler was found, `false` otherwise.

Remove a message handler previously registered to this channel with `channel.subscribe(onMessage)`.

```
import diagnostics_channel from 'diagnostics_channel';

const channel = diagnostics_channel.channel('my-channel');

function onMessage(message, name) {
  // Received data
}

channel.subscribe(onMessage);

channel.unsubscribe(onMessage);
```

```
const diagnostics_channel = require('diagnostics_channel');

const channel = diagnostics_channel.channel('my-channel');

function onMessage(message, name) {
  // Received data
}

channel.subscribe(onMessage);

channel.unsubscribe(onMessage);
```