ipcMain

> *Communicate asynchronously from the main process to renderer processes.*

Process: Main

The `ipcMain` module is an Event Emitter. When used in the main process, it handles asynchronous and synchronous messages sent from a renderer process (web page). Messages sent from a renderer will be emitted to this module.

For usage examples, check out the IPC tutorial.

Sending messages

It is also possible to send messages from the main process to the renderer process, see webContents.send for more information.

- When sending a message, the event name is the `channel`.
- To reply to a synchronous message, you need to set `event.returnValue`.
- To send an asynchronous message back to the sender, you can use `event.reply(...)`. This helper method will automatically handle messages coming from frames that aren't the main frame (e.g. iframes) whereas `event.sender.send(...)` will always send to the main frame.

Methods

The `ipcMain` module has the following method to listen for events:

`ipcMain.on(channel, listener)`

- `channel` string
- `listener` Function
  - `event` IpcMainEvent
  - `...args` any[]

Listens to `channel`, when a new message arrives `listener` would be called with `listener(event, args...)`.

`ipcMain.once(channel, listener)`

- `channel` string
- `listener` Function
  - `event` IpcMainEvent
  - `...args` any[]

Adds a one time `listener` function for the event. This `listener` is invoked only the next time a message is sent to `channel`, after which it is removed.

`ipcMain.removeListener(channel, listener)`

- `channel` string
- `listener` Function
  - `...args` any[]

Removes the specified `listener` from the listener array for the specified `channel`.

`ipcMain.removeAllListeners([channel])`

- `channel` string (optional)

Removes listeners of the specified `channel`.

`ipcMain.handle(channel, listener)`

- `channel` string
- `listener` Function<Promise<void> | any>
  - `event` IpcMainInvokeEvent
  - `...args` any[]

Adds a handler for an `invoke` able IPC. This handler will be called whenever a renderer calls `ipcRenderer.invoke(channel, ...args)`.

If `listener` returns a Promise, the eventual result of the promise will be returned as a reply to the remote caller. Otherwise, the return value of the listener will be used as the value of the reply.

```
ipcMain.handle('my-invokable-ipc', async (event, ...args) => {
  const result = await somePromise(...args)
  return result
})
```

```
async () => {
  const result = await ipcRenderer.invoke('my-invokable-ipc', arg1, arg2)
  // ...
}
```

The `event` that is passed as the first argument to the handler is the same as that passed to a regular event listener. It includes information about which WebContents is the source of the invoke request.

Errors thrown through `handle` in the main process are not transparent as they are serialized and only the `message` property from the original error is provided to the renderer process. Please refer to #24427 for details.

`ipcMain.handleOnce(channel, listener)`

- `channel` string
- `listener` Function<Promise<void> | any>
  - `event` IpcMainInvokeEvent
  - `...args` any[]

Handles a single `invoke` able IPC message, then removes the listener. See `ipcMain.handle(channel, listener)`.

`ipcMain.removeHandler(channel)`

- `channel` string

Removes any handler for `channel`, if present.

IpcMainEvent object

The documentation for the `event` object passed to the `callback` can be found in the `ipc-main-event` structure docs.

IpcMainInvokeEvent object

The documentation for the `event` object passed to `handle` callbacks can be found in the `ipc-main-invoke-event` structure docs.