clipboard

Perform copy and paste operations on the system clipboard.

Process: Main, Renderer

On Linux, there is also a `selection` clipboard. To manipulate it you need to pass `selection` to each method:

```
const { clipboard } = require('electron')

clipboard.writeText('Example string', 'selection')
console.log(clipboard.readText('selection'))
```

Methods

The `clipboard` module has the following methods:

Note: Experimental APIs are marked as such and could be removed in future.

`clipboard.readText([type])`

- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Returns `string` - The content in the clipboard as plain text.

```
const { clipboard } = require('electron')

clipboard.writeText('hello i am a bit of text!')

const text = clipboard.readText()
console.log(text)
// hello i am a bit of text!'
```

`clipboard.writeText(text[, type])`

- `text` string
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Writes the `text` into the clipboard as plain text.

```
const { clipboard } = require('electron')

const text = 'hello i am a bit of text!'
clipboard.writeText(text)
```

clipboard.readHTML([type])

- type string (optional) - Can be selection or clipboard; default is 'clipboard'. selection is only available on Linux.

Returns string - The content in the clipboard as markup.

```
const { clipboard } = require('electron')

clipboard.writeHTML('<b>Hi</b>')
const html = clipboard.readHTML()

console.log(html)
// <meta charset='utf-8'><b>Hi</b>
```

clipboard.writeHTML(markup[, type])

- markup string
- type string (optional) - Can be selection or clipboard; default is 'clipboard'. selection is only available on Linux.

Writes markup to the clipboard.

```
const { clipboard } = require('electron')

clipboard.writeHTML('<b>Hi</b>')
```

clipboard.readImage([type])

- type string (optional) - Can be selection or clipboard; default is 'clipboard'. selection is only available on Linux.

Returns NativeImage - The image content in the clipboard.

clipboard.writeImage(image[, type])

- image NativeImage
- type string (optional) - Can be selection or clipboard; default is 'clipboard'. selection is only available on Linux.

Writes image to the clipboard.

clipboard.readRTF([type])

- type string (optional) - Can be selection or clipboard; default is 'clipboard'. selection is only available on Linux.

Returns string - The content in the clipboard as RTF.

```
const { clipboard } = require('electron')
```

```
clipboard.writeRTF('{\\rtf1\\ansi{\\fonttbl\\f0\\fswiss Helvetica;}\\f0\\pard\nThis is some

const rtf = clipboard.readRTF()
console.log(rtf)
// {\\rtf1\\ansi{\\fonttbl\\f0\\fswiss Helvetica;}\\f0\\pard\nThis is some {\\b bold} text.
```

`clipboard.writeRTF(text[, type])`

- `text` string
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Writes the `text` into the clipboard in RTF.

```
const { clipboard } = require('electron')

const rtf = '{\\rtf1\\ansi{\\fonttbl\\f0\\fswiss Helvetica;}\\f0\\pard\nThis is some {\\b bo
clipboard.writeRTF(rtf)
```

`clipboard.readBookmark()` *macOS Windows*

Returns `Object`:

- `title` string
- `url` string

Returns an Object containing `title` and `url` keys representing the bookmark in the clipboard. The `title` and `url` values will be empty strings when the bookmark is unavailable. The `title` value will always be empty on Windows.

`clipboard.writeBookmark(title, url[, type])` *macOS Windows*

- `title` string - Unused on Windows
- `url` string
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Writes the `title` (macOS only) and `url` into the clipboard as a bookmark.

Note: Most apps on Windows don't support pasting bookmarks into them so you can use `clipboard.write` to write both a bookmark and fallback text to the clipboard.

```
const { clipboard } = require('electron')

clipboard.writeBookmark({
  text: 'https://electronjs.org',
  bookmark: 'Electron Homepage'
})
```

`clipboard.readFindText()` *macOS*

Returns `string` - The text on the find pasteboard, which is the pasteboard that holds information about the current state of the active application's find panel.

This method uses synchronous IPC when called from the renderer process. The cached value is reread from the find pasteboard whenever the application is activated.

`clipboard.writeFindText(text)` *macOS*

- `text` string

Writes the `text` into the find pasteboard (the pasteboard that holds information about the current state of the active application's find panel) as plain text. This method uses synchronous IPC when called from the renderer process.

`clipboard.clear([type])`

- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Clears the clipboard content.

`clipboard.availableFormats([type])`

- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Returns `string[]` - An array of supported formats for the clipboard `type`.

```
const { clipboard } = require('electron')

const formats = clipboard.availableFormats()
console.log(formats)
// [ 'text/plain', 'text/html' ]
```

`clipboard.has(format[, type])` *Experimental*

- `format` string
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Returns `boolean` - Whether the clipboard supports the specified `format`.

```
const { clipboard } = require('electron')

const hasFormat = clipboard.has('public/utf8-plain-text')
console.log(hasFormat)
// 'true' or 'false'
```

`clipboard.read(format)` *Experimental*

- `format` string

Returns `string` - Reads `format` type from the clipboard.

`format` should contain valid ASCII characters and have `/` separator. `a/c`, `a/bc` are valid formats while `/abc`, `abc/`, `a/`, `/a`, `a` are not valid.

`clipboard.readBuffer(format)` *Experimental*

- `format` string

Returns `Buffer` - Reads `format` type from the clipboard.

```
const { clipboard } = require('electron')

const buffer = Buffer.from('this is binary', 'utf8')
clipboard.writeBuffer('public/utf8-plain-text', buffer)

const ret = clipboard.readBuffer('public/utf8-plain-text')

console.log(buffer.equals(out))
// true
```

`clipboard.writeBuffer(format, buffer[, type])` *Experimental*

- `format` string
- `buffer` Buffer
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Writes the `buffer` into the clipboard as `format`.

```
const { clipboard } = require('electron')

const buffer = Buffer.from('writeBuffer', 'utf8')
clipboard.writeBuffer('public/utf8-plain-text', buffer)
```

`clipboard.write(data[, type])`

- `data` Object
  - `text` string (optional)
  - `html` string (optional)
  - `image` NativeImage (optional)
  - `rtf` string (optional)
  - `bookmark` string (optional) - The title of the URL at `text`.
- `type` string (optional) - Can be `selection` or `clipboard`; default is 'clipboard'. `selection` is only available on Linux.

Writes `data` to the clipboard.

```
const { clipboard } = require('electron')

clipboard.write({
  text: 'test',
  html: '<b>Hi</b>',
  rtf: '{\\rtf1\\utf8 text}',
  bookmark: 'a title'
})

console.log(clipboard.readText())
// 'test'

console.log(clipboard.readHTML())
// <meta charset='utf-8'><b>Hi</b>

console.log(clipboard.readRTF())
// '{\\rtf1\\utf8 text}'

console.log(clipboard.readBookmark())
// { title: 'a title', url: 'test' }
```