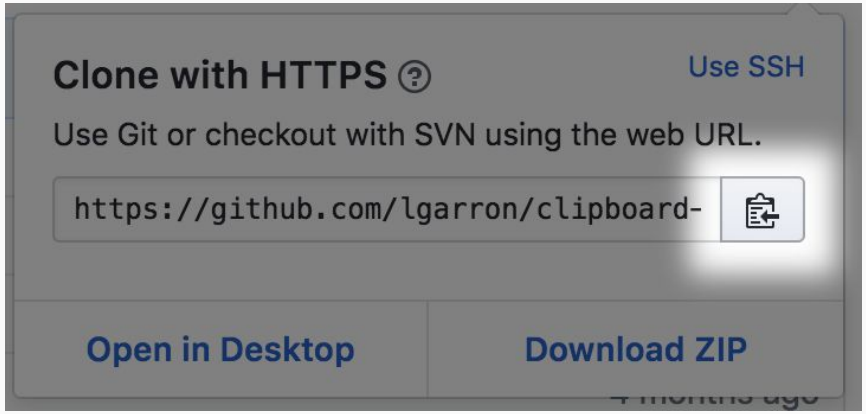
Clipboard APIs on the Web

Lucas Garron
July 10, 2018



Install and run the Node command line tool

To install the Node module:

1. Download Google Chrome for Desktop.
2. Install the current Long-Term Support version of Node.
3. Install Lighthouse. The `-g` flag installs it as a global module.

```
npm install -g lighthouse
```

To run an audit:

```
lighthouse <url>
```

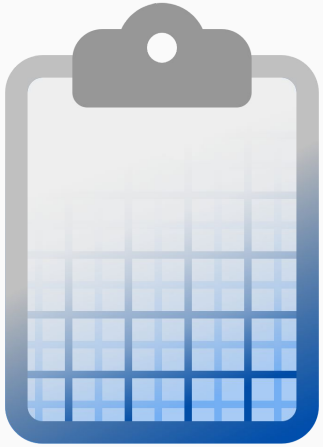
To see audit options:

```
lighthouse --help
```

Agenda

- `clipboard-polyfill`
- Async Clipboard API
- Older APIs
- Future

clipboard-polyfill



github.com/lgarron/clipboard-polyfill

Asynchronous Clipboard API

Asynchronous Clipboard API

```
<button onclick="copy()">Copy text!</button>
<script>
function copy() {

  navigator.clipboard.writeText("Hello world!");

};
</script>
```

Asynchronous Clipboard API

```
<button onclick="copy()">Copy text!</button>
<script>
function copy() {
  navigator.clipboard.writeText("Hello world!").then(
    console.log,
    console.error);
};
</script>
```

Asynchronous Clipboard API

```
var dt = new DataTransfer();
dt.setData("text/plain", "plain text")
dt.setData("text/html", "<b>markup</b> text")
navigator.clipboard.write(dt);
```

Asynchronous Clipboard API

```
navigator.clipboard.readText().then(
  console.log,
  console.error
);
```

navigator.clipboard

```
Promise<void>
  writeText(DOMString)

Promise<void>
  write(DataTransfer)
```

```
Promise<DOMString>
  readText()

Promise<DataTransfer>
  read()
```

https://w3c.github.io/clipboard-apis/#async-clipboard-api

Availability

`writeText()` – with user gesture

 66+

 53+

Availability

Clipboard API and events

Editor's Draft, 12 June 2018

This version:

https://w3c.github.io/clipboard-apis/

Latest published version:

https://www.w3.org/TR/clipboard-apis/

Issue Tracking:

GitHub

Inline In Spec

Editors:

Gary Kacmarcik (Google)

Grisha Lyukshin (Microsoft)

Hallvord R. M. Steen (Mozilla - emeritus)

Explainer:

Async Clipboard API Explainer

https://w3c.github.io/clipboard-apis/

Older APIs

Other APIs

- Flash
- Internet Explorer
- `document.execCommand("copy")`
- Chrome Extension API / Web Extensions
- Async Clipboard API

Most compatible until 2015

- **Flash**
- Internet Explorer
- `document.execCommand("copy")`
- Chrome Extension API / Web Extensions
- Aync Clipboard API

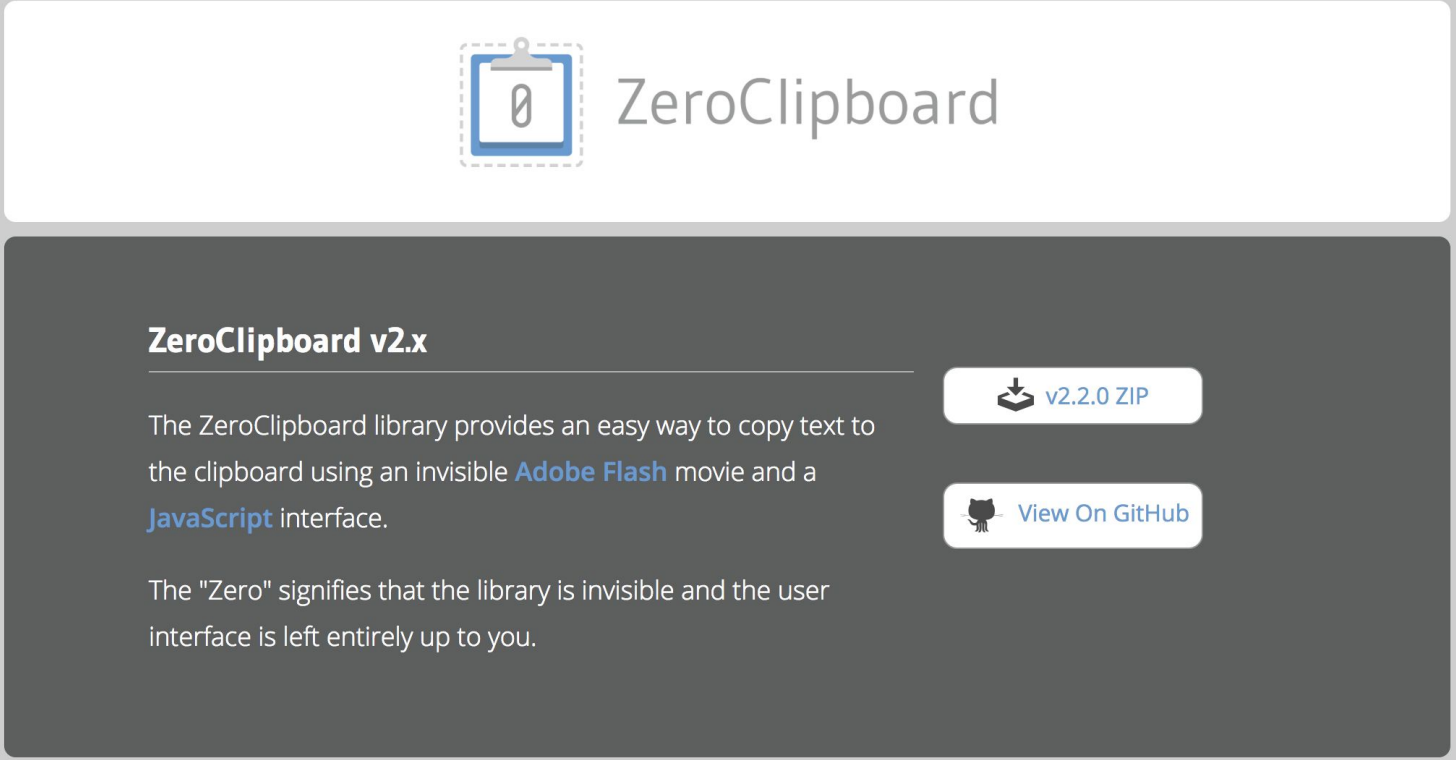
Most compatible in 2018

- Flash
- Internet Explorer
- **`document.execCommand("copy")`**
- Chrome Extension API / Web Extensions
- Aync Clipboard API

Flash

- Formerly the most compatible API
- Requires Flash to be enabled
  - Used to be enabled in most desktop browsers
  - Deprecated in modern browsers
- Also requires a user gesture
- Not super convenient

Flash



https://zeroclipboard.github.io/

Flash

Simple Example

```
<html>
  <body>
    <button id="copy-button" data-clipboard-text="Copy Me!" title="Click to copy me.">Copy to Clipboard</bu
    <script src="ZeroClipboard.js"></script>
    <script src="main.js"></script>
  </body>
</html>
```

```
// main.js
var client = new ZeroClipboard( document.getElementById("copy-button") );

client.on( "ready", function( readyEvent ) {
  // alert( "ZeroClipboard SWF is ready!" );

  client.on( "aftercopy", function( event ) {
    // `this` === `client`
    // `event.target` === the element that was clicked
    event.target.style.display = "none";
    alert("Copied text to clipboard: " + event.data["text/plain"] );
  } );
} );
```

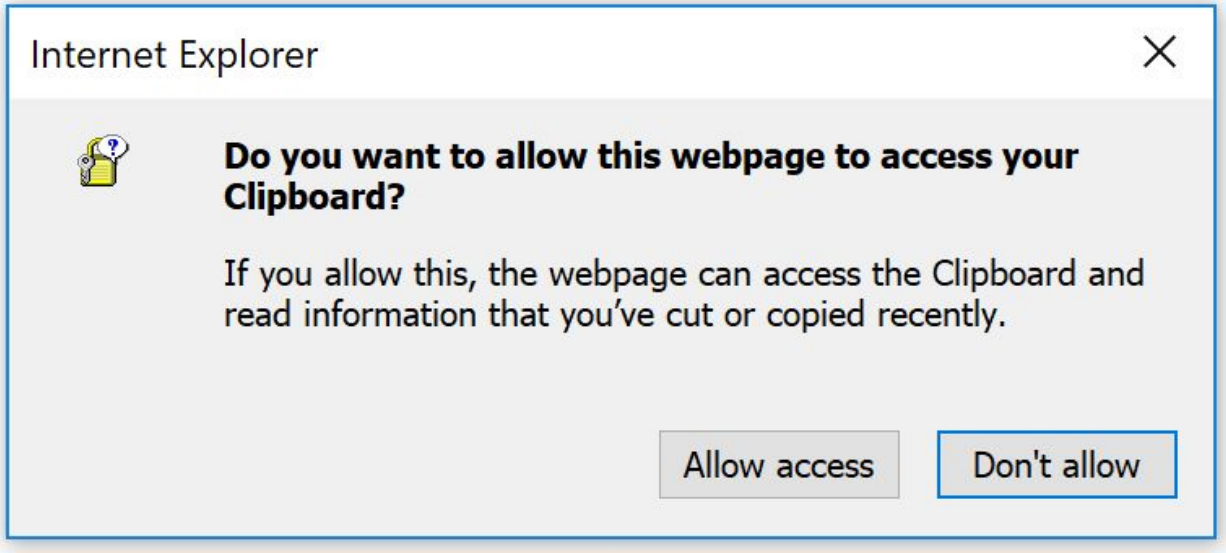
https://github.com/zeroclipboard/zeroclipboard

Internet Explorer 9-11

```
window.clipboardData.setData("Text", text);
window.clipboardData.setData("URL", url);
```

https://msdn.microsoft.com/en-us/windows/ms536744(v=vs.71)

Internet Explorer 9-11



Internet Explorer 9-11

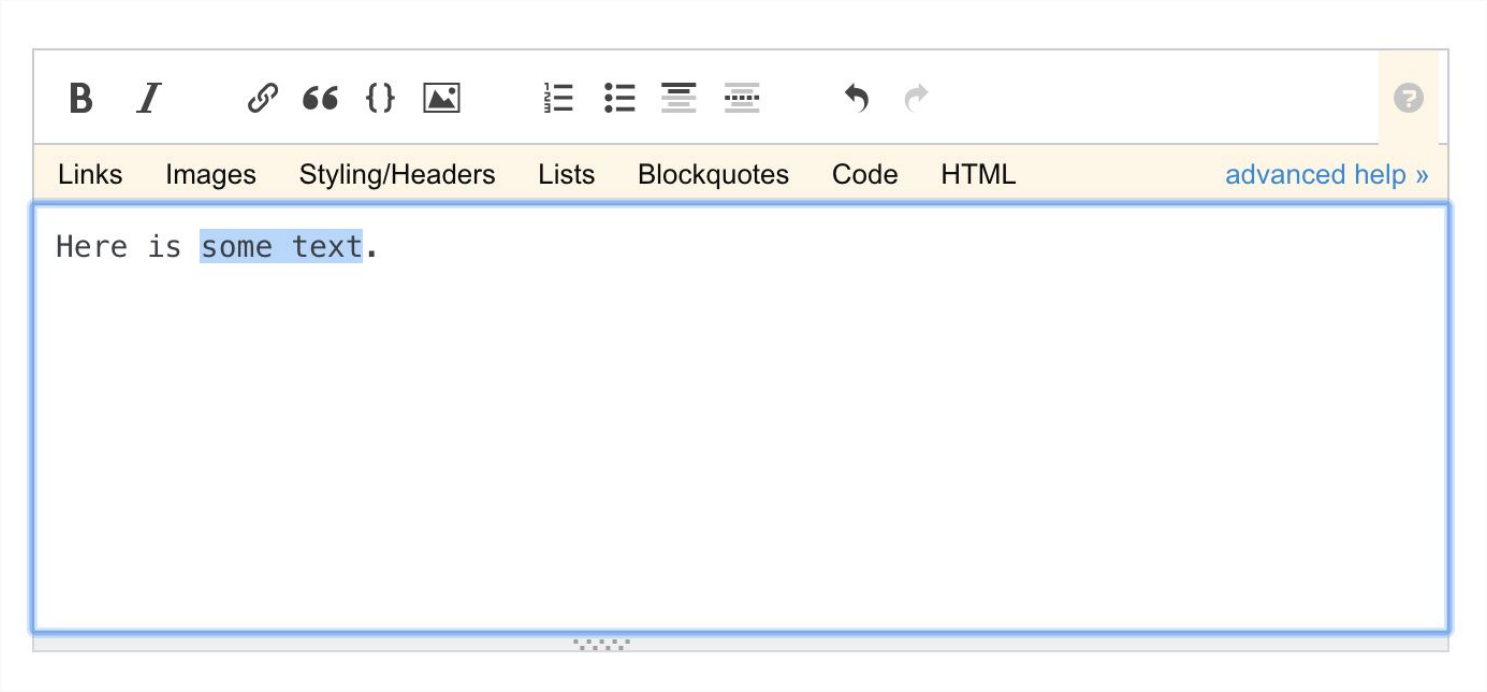
```
window.clipboardData.getData("Text");
window.clipboardData.getData("URL");
```

https://msdn.microsoft.com/en-us/windows/ms536436(v=vs.71)

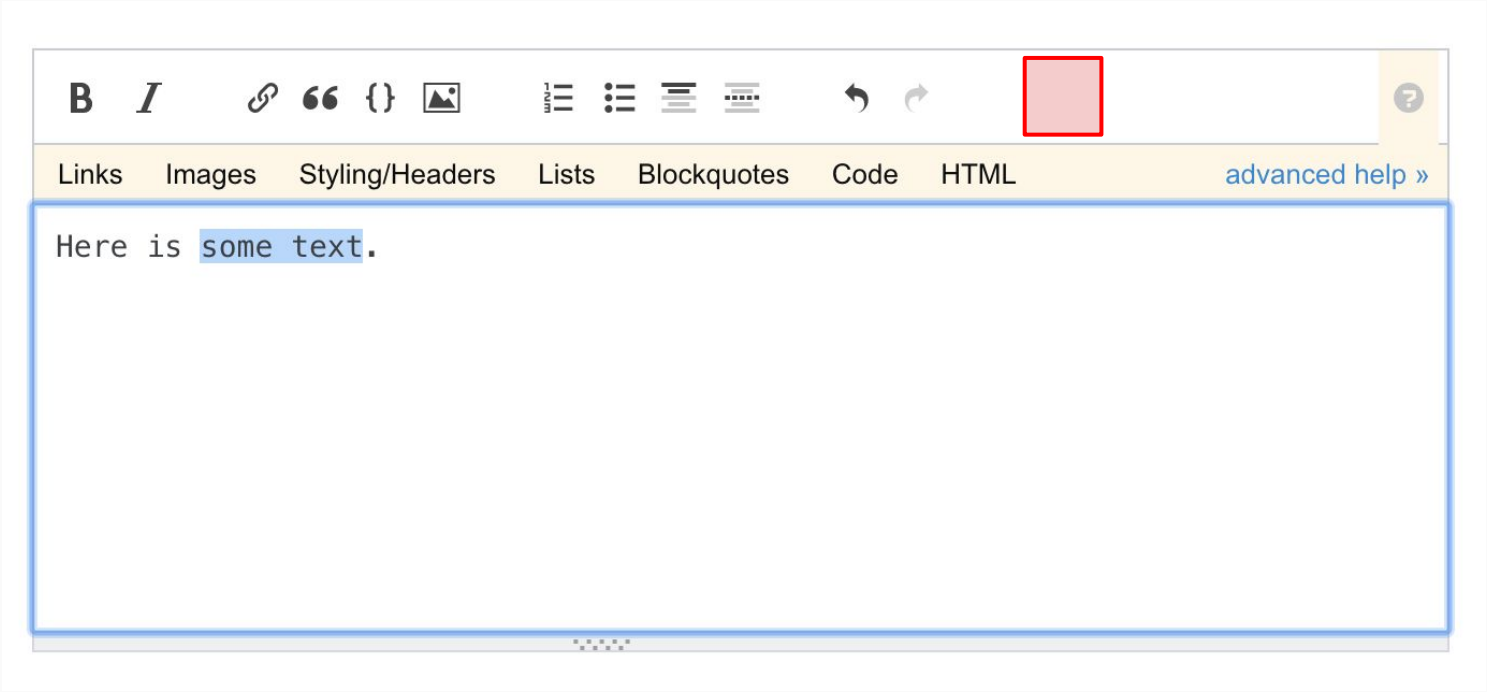
```
document.execCommand("copy")
```

Triggers a copy event
for the current selection
“as if the user initiated it”.

document.execCommand("copy")



document.execCommand("copy")



document.execCommand("copy")

```
function fallbackCopyTextToClipboard(text) {
  var textArea = document.createElement("textarea");
  textArea.value = text;
  document.body.appendChild(textArea);
  textArea.focus();
  textArea.select();

  try {
    var successful = document.execCommand('copy');
    var msg = successful ? 'successful' : 'unsuccessful';
    console.log('Fallback: Copying text command was ' + msg);
  } catch (err) {
    console.error('Fallback: Oops, unable to copy', err);
  }

  document.body.removeChild(textArea);
}
```

https://stackoverflow.com/questions/400212/how-do-i-copy-to-the-clipboard-in-javascript

document.execCommand("copy")

```
<!-- Target -->
<textarea id="bar">Mussum ipsum cacilds...</textarea>

<!-- Trigger -->
<button class="btn" data-clipboard-action="cut"
data-clipboard-target="#bar">
    Cut to clipboard
</button>
```

https://clipboardjs.com/

document.execCommand("copy")

```
function copy(str) {
  function listener(e) {
    e.clipboardData.setData("text/plain", str);
    e.preventDefault();
  }
  document.addEventListener("copy", listener);
  document.execCommand("copy");
  document.removeEventListener("copy", listener);
}
```

Issues with `document.execCommand("copy")`

	Chrome 61	Safari 11 (macOS)	Safari 11 (iOS)	Edge 15	Firefox 54
`supported` always returns true †	✅	✅	✅	✅	✅
`enabled` **without** selection returns true †	❌	❌	❌	❌	✅
`exec` works **without** selection †	✅	⚠️ [1]	⚠️ [1]	✅	✅
`enabled` **with** selection returns true †	✅	✅	✅	✅	✅
`exec` works **with** selection †	✅	✅	✅	✅	✅
`exec` fails outside user gesture	✅	✅	✅	✅	✅
`setData()` in listener works	✅	✅	❌ [2]	✅	✅
`getData()` in listener shows if `setData()` worked	✅	✅	⚠️ [2]	❌ [3]	✅
Copies all types set with `setData()`	✅	✅	✅	❌ [4]	✅
`exec` reports success correctly	✅	✅	⚠️ [2]	❌ [5]	✅
`contenteditable` does not break document selection	❌	❌	❌	✅	✅
`user-select: none` does not break document selection	✅ (Cr 64)	❌	❌	✅ (Edge 16)	✅ (FF 57)
Can construct `new DataTransfer()`	✅	❌	❌	❌	❌
Writes `CF_HTML` on Windows	✅	N/A	N/A	❌ [6]	✅

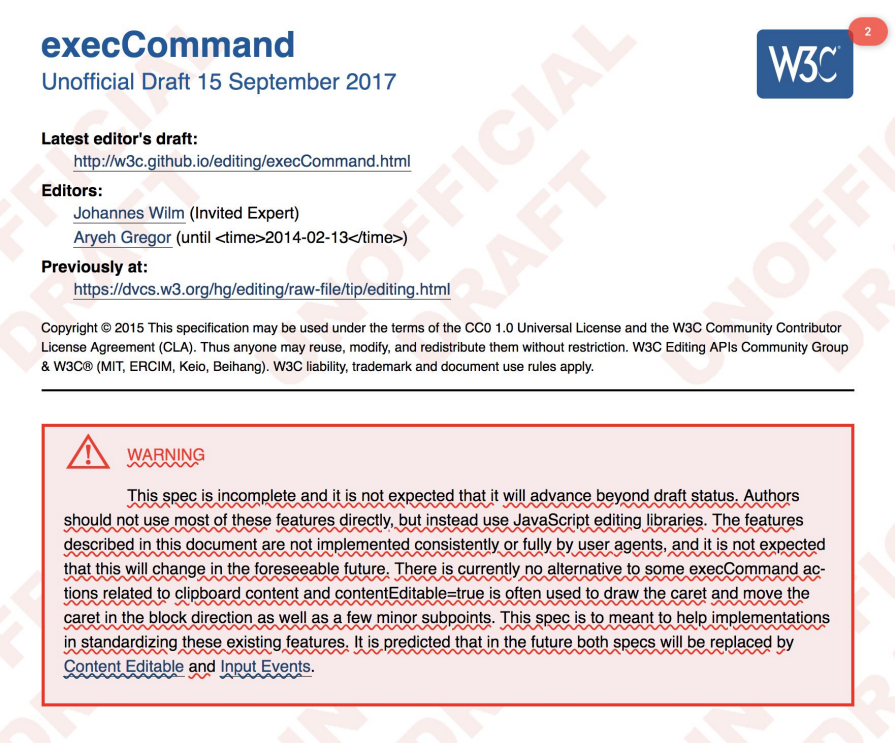
https://github.com/lgarron/clipboard-polyfill/blob/master/experiment/Conclusions.md

Issues with `document.execCommand(...)`

- Synchronous
  - Blocks the main page
  - Adds latency if images are sanitized
  - Permission prompt would block the whole page
- Requires access to `document`
- Tricky to use
- Buggy

https://github.com/w3c/clipboard-apis/blob/master/explainer.adoc

Issues with `document.execCommand(...)`



execCommand

Unofficial Draft 15 September 2017

Latest editor's draft:
http://w3c.github.io/editing/execCommand.html

Editors:
Johannes Wilm (Invited Expert)
Aryeh Gregor (until <time>2014-02-13</time>)

Previously at:
https://dvcs.w3.org/hg/editing/raw-file/tip/editing.html

WARNING

This spec is incomplete and it is not expected that it will advance beyond draft status. Authors should not use most of these features directly, but instead use JavaScript editing libraries. The features described in this document are not implemented consistently or fully by user agents, and it is not expected that this will change in the foreseeable future. There is currently no alternative to some execCommand actions related to clipboard content and contentEditable=true is often used to draw the caret and move the caret in the block direction as well as a few minor subpoints. This spec is to meant to help implementations in standardizing these existing features. It is predicted that in the future both specs will be replaced by Content Editable and Input Events.

https://w3c.github.io/editing/execCommand.html

Chrome

`"clipboardRead"`	Required if the extension or app uses `document.execCommand('paste')`.
`"clipboardWrite"`	Indicates the extension or app uses `document.execCommand('copy')` or `document.execCommand('cut')`. This permission is **required for hosted apps**; it's recommended for extensions and packaged apps.

https://developer.chrome.com/extensions/declare_permissions

Firefox

Extensions built using WebExtension APIs can interact with the system clipboard using `document.execCommand()`:

- `document.execCommand("copy")`
- `document.execCommand("cut")`
- `document.execCommand("paste")`

https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Interact_with_the_clipboard

Chrome Extension API / Web Extensions

This API is experimental. It is only available to Chrome users on the **dev channel**.

chrome.clipboard

Description:	The `chrome.clipboard` API is provided to allow users to access data of the clipboard. This is a temporary solution for chromeos platform apps until open-web alternative is available. It will be deprecated once open-web solution is available, which could be in 2017 Q4.
Availability:	**Dev** channel only. **Learn more**.
Permissions:	`"clipboard"`

Summary

Methods
setImageData - `chrome.clipboard.setImageData(ArrayBuffer imageData, enum of "png", or "jpeg" type, array of object additionalItems, function callback)`
Events
onClipboardDataChanged

https://developer.chrome.com/apps/clipboard

Future

Future

Async Clipboard API in all browsers?

Future

More MIME types?

Future

§ 6.4.2. Writing to the clipboard

These data types must be placed on the clipboard with a corresponding native type description if added to a `DataTransfer` object during *copy* and *cut* events.

- text/plain
- text/uri-list
- text/csv
- text/html
- image/svg+xml
- application/xml, text/xml
- application/json

> **Warning! The data types that untrusted scripts are allowed to write to the clipboard are limited as a security precaution. Untrusted scripts can attempt to exploit security vulnerabilities in local software by placing data known to trigger those vulnerabilities on the clipboard.**

https://w3c.github.io/clipboard-apis/#writing-to-clipboard

Future

Read clipboard?

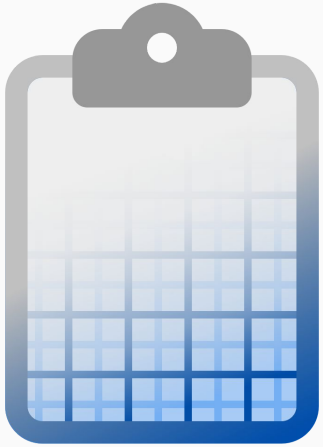
Listen to clipboard changes?

Future

Contribute to the discussion:

https://github.com/w3c/clipboard-apis

Future



I'll be keep `clipboard-polyfill` compatible with what is available.