



Clipboard API

- [_](#)
- [Document.execCommand\(\)](#) [_](#)
- [_ Clipboard API](#)
- [Clipboard](#) [_](#)
 - [Clipboard.readText\(\)](#)
 - [Clipboard.read\(\)](#)
 - [Clipboard.writeText\(\)](#)
 - [Clipboard.write\(\)](#)
- [copy](#) [_](#) [cut](#) [_](#)
- [paste](#) [_](#)
- [_](#)



JavaScript

“ ”

- `Document.execCommand()` [_](#)
- [_](#) Clipboard API
- `copy` [_](#) `paste` [_](#)

Document.execCommand()

Document.execCommand() []

[]

- document.execCommand('copy') []
- document.execCommand('cut') []
- document.execCommand('paste') []

1 []

[]

document.execCommand('copy') []

```
const inputElement = document.querySelector('#input');
inputElement.select();
document.execCommand('copy');
```

[]

inputElement []

inputElement.select() []

document.execCommand('copy') []

[]

[]

2 []

[]

document.execCommand('paste') []

```
const pasteText = document.querySelector('#output');
pasteText.focus();
document.execCommand('paste');
```

3 []

Document.execCommand() []

[]

[]

/

[]

[]

[]

Clipboard API []



Clipboard API

Clipboard API

document.execCommand()

Promise

navigator.clipboard

Clipboard

```
const clipboardObj = navigator.clipboard;
```

navigator.clipboard

undefined

API

API

Chrome

HTTPS

API

localhost

Permissions API

clipboard-write

clipboard-read

" "



```
(async () => {
  const text = await navigator.clipboard.readText();
  console.log(text);
})();
```

Clipboard API

DOM

setTimeout()

```
setTimeout(async () => {
  const text = await navigator.clipboard.readText();
  console.log(text);
}, 2000);
```



Promise

Clipboard.readText()

```
document.body.addEventListener(
  'click',
  async (e) => {
    const text = await navigator.clipboard.readText();
    console.log(text);
  }
)
```



```
async function getClipboardContents() {
  try {
    const text = await navigator.clipboard.readText();
    console.log('Pasted content: ', text);
  } catch (err) {
    console.error('Failed to read clipboard contents: ', err);
  }
}
```

Clipboard.read()

--	--	--	--	--

Promise resolved

ClipboardItem

```
async function getClipboardContents() {
  try {
    const clipboardItems = await navigator.clipboard.read();
    for (const clipboardItem of clipboardItems) {
      for (const type of clipboardItem.types) {
        const blob = await clipboardItem.getType(type);
        console.log(URL.createObjectURL(blob));
      }
    }
  } catch (err) {
    console.error(err.name, err.message);
  }
}
```

ClipboardItem

ClipboardItem.getType()

ClipboardItem.types

HTML

MIME

MIME

text/html

text/plain

ClipboardItem.getType(type)

MIME

Promise

Clipboard.writeText()

Clipboard.writeText()

```
document.body.addEventListener(
  'click',
  async (e) => {
    await navigator.clipboard.writeText('Yo')
  }
)
```

try...catch

```

async function copyPageUrl() {
  try {
    await navigator.clipboard.writeText(location.href);
    console.log('Page URL copied to clipboard');
  } catch (err) {
    console.error('Failed to copy: ', err);
  }
}

```

Clipboard.write()

Clipboard.write()

ClipboardItem

```

try {
  const imgURL = 'https://dummyimage.com/300.png';
  const data = await fetch(imgURL);
  const blob = await data.blob();
  await navigator.clipboard.write([
    new ClipboardItem({
      [blob.type]: blob
    })
  ]);
  console.log('Image copied.');
```

```

} catch (err) {
  console.error(err.name, err.message);
}

```

Chrome

PNG

ClipboardItem()

ClipboardItem

MIME

```

function copy() {
  const image = await fetch('kitten.png');
  const text = new Blob(['Cute sleeping kitten'], {type: 'text/plain'});

```

```
const item = new ClipboardItem({
  'text/plain': text,
  'image/png': image
});
await navigator.clipboard.write([item]);
}
```

copy cut



copy 



```
const source = document.querySelector('.source');

source.addEventListener('copy', (event) => {
  const selection = document.getSelection();
  event.clipboardData.setData('text/plain', selection.toString().toUpperCase());
  event.preventDefault();
});
```



clipboardData 

- Event.clipboardData.setData(type, data) 
- Event.clipboardData.getData(type) 
- Event.clipboardData.clearData([type])

- Event.clipboardData.items




```
const clipboardItems = [];

document.addEventListener('copy', async (e) => {
  e.preventDefault();
  try {
    let clipboardItems = [];
    for (const item of e.clipboardData.items) {
      if (!item.type.startsWith('image/')) {
```

```

        continue;
    }
    clipboardItems.push(
        new ClipboardItem({
            [item.type]: item,
        })
    );
    await navigator.clipboard.write(clipboardItems);
    console.log('Image copied.');
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

paste

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

```

document.addEventListener('paste', async (e) => {
  e.preventDefault();
  const text = await navigator.clipboard.readText();
  console.log('Pasted text: ', text);
});
```

```

    }
  } catch (err) {
    console.error(err.name, err.message);
  }
});
```

- [Unblocking clipboard access](#)
- [Interact with the clipboard](#)
- [Multi-MIME Type Copying with the Async Clipboard API](#)

□□ #2

□ □□ □□ 2021-01-12 23:46:06 UTC

□ □□ □□ 2021-01-25 15:35:18 UTC