File Handling Access API

In The File System Access API: simplifying access to local files, the authorsdescribe how to use the API to create file management functionality for web applications

The idea is that we'll be able to open a file from, display it in the browser and then, if desired, save it to a new location in our local file system.

To open a file we'll use the following code.

We define the following constants and variables:

- Capture the button to open a file
- Capture the button to save a file
- Capture the text area where we'll insert the text
- Capture the content that we'll save
- Sets up an empty variable for the file handle we want to work with

```
const openButton = document.querySelector('.openButton');
const saveButton = document.querySelector('.saveButton');
const display = document.querySelector('.content-area');
const dataToSave = display.innerText;
let fileHandle;
```

The openFile function is called when the open file button is clicked. It opens the file, reads the content and puts the content in the text area.

The function runs the following steps:

1. We use feature detection to make sure the feature is supported, if not then we bail
2. We create a new file handle and show the file picker dialogue
3. Open the file
4. Read and convert the content of the file to text
5. Insert the content of the file into the display text area

```
async function openFile() {
  if ('showOpenFilePicker' in window) { // 1
```

```
    [fileHandle] = await window.showOpenFilePicker(); // 2
    const file = await fileHandle.getFile(); // 3
    const contents = await file.text(); // 4
    display.innerText = contents; // 5
  } else {
    console.error('filehandle API not supported');
  }
}
'filehandle API not supported'
```

Saving a file

Saving a file is a little more complicated. We need to create a new file handle, capture the content to save, write the content to the file and then close the file and save it.

> The steps are as follows:

1. Use a feature query to detect if showSaveFilePicker is supported. If it's not then bail
2. Show the save file picker
3. Create a `FileSystemWritableFileStream` to write to
4. Write the file
5. close the file and write the contents to disk.

```
async function saveFile() {
  if ('showSaveFilePicker' in window) { // 1
    const newHandle = await window.showSaveFilePicker(); // 2
      const writableStream = await newHandle.createWritable(); // 3
        await writableStream.write(dataToSave); // 4
      await writableStream.close(); // 5
  }
}
```

Finally we attack the functions to the click event of the corresponding button.

```
openButton.addEventListener('click', openFile);
saveButton.addEventListener('click', saveFile);
```

Links and resources

- Public explainer
- File System Access and File specifications at WICG
- ChromeStatus.com entry
- File System Access API — MDN
- File System Access API — Chromium Security Model