

ChatGPT: Build custom file picker

Yes! You can build a **custom file picker** for a **static web page** that interacts with an `<iframe>` for displaying content. However, due to **browser security restrictions**, if your webpage is hosted statically (e.g., on GitHub Pages, Netlify, or Vercel), the **file picker will not be able to load local files into an `<iframe>` directly** unless the files are hosted online.

✅ Possible Solutions

❌ Use `<input type="file">` to Read Local Files and Display in an `<iframe>` (Without Uploading)

If your goal is to preview local HTML files, you **cannot directly load them into an `<iframe>`** (due to browser security). However, you can **read the file's contents** and use Blob URLs to preview them.

✅ Working Solution

```
<input type="file" id="filePicker" accept=".html,.htm" />
```

```
<iframe id="previewFrame"></iframe>
```

```
<script>
```

```
document.getElementById('filePicker').addEventListener('change', function(event) {
```

```
    const file = event.target.files[0];
```

```
    if (file) {
```

```
        const blobURL = URL.createObjectURL(file);
```

```
        document.getElementById('previewFrame').src = blobURL;
```

```
    }
```

```
});
```

```
</script>
```

```
<style>
```

```
#previewFrame {
```

```
    width: 100%;
```

```
    height: 500px;
```

```
border: 1px solid black;
}
</style>
```

◆ How It Works

- The file is read using a Blob URL, which allows temporary access to the file.
- The <iframe> **displays the file's contents** inside the page.
- No file is uploaded; everything is done **in the browser**.

🔗 Alternative: Render HTML Content Directly in a <div>

If you **only need to display the HTML content** (not actually load a file inside an <iframe>), you can read the file and inject it into a <div>.

Example:

```
<input type="file" id="filePicker" accept=".html,.htm" />
```

```
<div id="previewContainer"></div>
```

```
<script>
```

```
document.getElementById('filePicker').addEventListener('change', function(event) {
```

```
    const file = event.target.files[0];
```

```
    if (file) {
```

```
        const reader = new FileReader();
```

```
        reader.onload = function(e) {
```

```
            document.getElementById('previewContainer').innerHTML = e.target.result;
```

```
        };
```

```
        reader.readAsText(file);
```

```
    }
```

```
});
```

```
</script>
```

```
<style>
#previewContainer {
  width: 100%;
  height: 500px;
  border: 1px solid black;
  overflow: auto;
  background: #f9f9f9;
  padding: 10px;
}
</style>
```

◆ How It Works

- The file content is **read as text** and injected into a <div>.
 - **This is safer than using <iframe>**, since there are fewer security restrictions.
 - Scripts inside the loaded HTML file **will not execute** (for security reasons).
-

🚫 What You CANNOT Do on a Static Webpage

- **Directly load file:// URLs into an <iframe>** → Browsers block this for security.
 - **Load local files into <iframe> without user interaction** → Requires server-side processing.
-

🎯 Best Option?

- ✅ Use **Blob URLs (URL.createObjectURL(file))** to display the file in an <iframe>.
 - ✅ Use **FileReader.readAsText()** to inject HTML directly into a <div>.
-

🚀 Now you have a fully functional custom file picker for a static webpage! Let me know if you need further improvements. 😊