



Web



From SharePoint on-premises



On this page



WEB > OPEN A DOCUMENT > IMAGE FROM SHAREPOINT ONLINE

Open and display image files in SharePoint Online



COPY PAGE



Image



Nutrient SharePoint SDK supports two methods for directly opening and rendering documents inside your SharePoint Online application.

- 1 From a web part. For more information, see our guide on [how to get started with the web part](#).
- 2 From a file handler. For more information, see our guide on [how to get started with the file handler](#).

Web part file picker

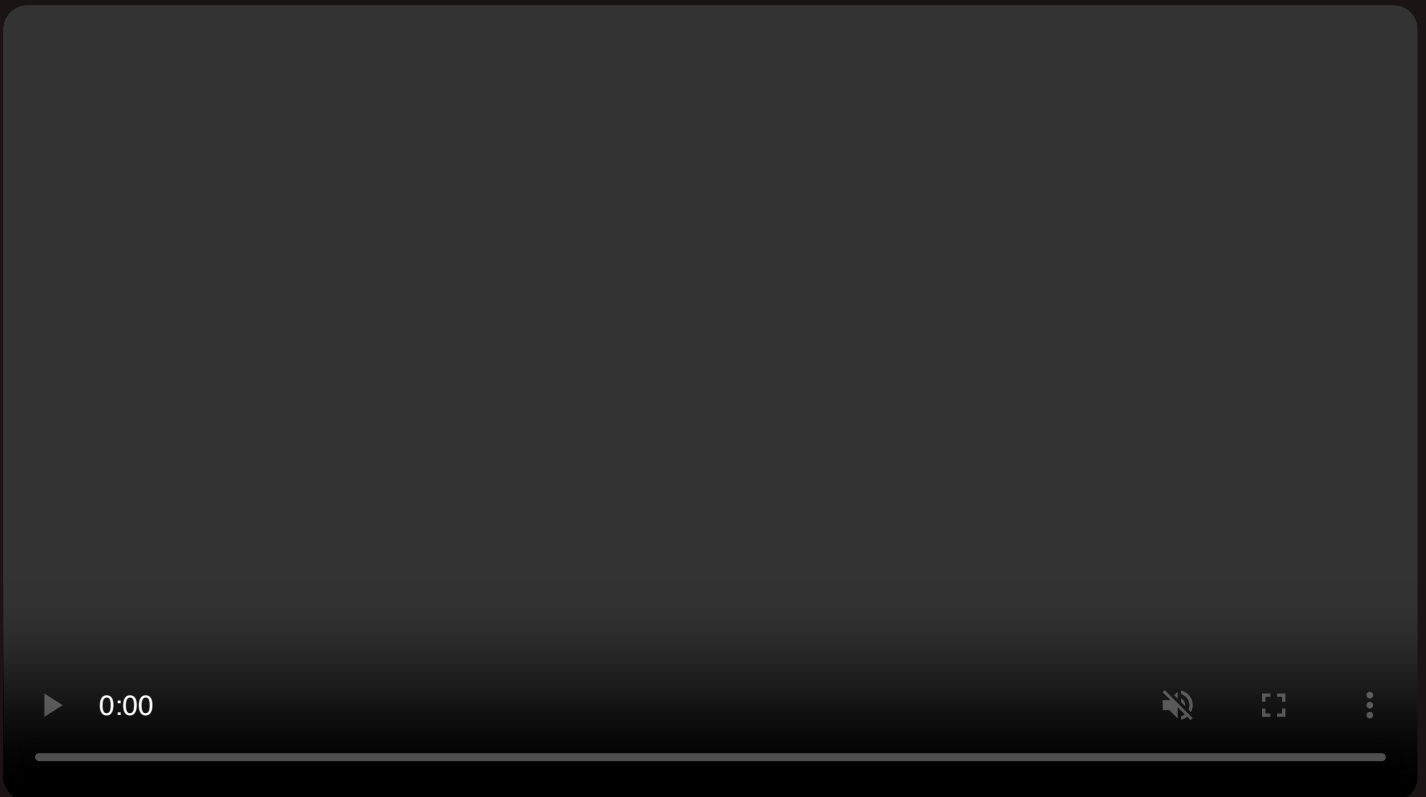
After the Nutrient web part is placed on a SharePoint page, a document is selected from the file picker that's accessible from the property pane. This document will be loaded into the web part whenever the SharePoint page is loaded.



ASK AI

You can find an example implementation of the file picker and web part in our [GitHub project](#).

We used the [pnp/sp-dev-fx-property-controls](#) package as the native file picker, and it can be customized as needed.



Here's the code sample for a SharePoint Framework web part:

```
import { PropertyFieldFilePicker } from "@pnp/spfx-property-controls/lib/PropertyFieldFilePicker";

export default class PspdfSampleWebPart extends BaseClientSideWebPart {
  protected getPropertyPaneConfiguration() {
    return {
      pages: [
        {
          header: {
            description: strings.PropertyPaneDescription,
          },
          groups: [
            {
              groupName: strings.PDFViewingGroupName,
              groupFields: [
                PropertyFieldFilePicker("filePicker", {
                  context: this.context,
                  filePickerResult: this.properties.filePickerResult,
                  onPropertyChange: this.onPropertyPaneFieldChanged.bind(this),
                })
              ]
            }
          ]
        }
      ]
    };
  }
}
```

```

        properties: this.properties,
        accepts: [".pdf", ".png", ".jpg", ".jpeg", ".tif", ".tiff"],
        hideLinkUploadTab: true,
        hideStockImages: true,
        hideWebSearchTab: true,
        hideRecentTab: true,
        hideOrganisationalAssetTab: true,
        hideOneDriveTab: true,
        hideLocalUploadTab: true,
        onSave: (e) => {
            this.properties.filePickerResult = e;
        },
        onChange: (e) => {
            this.properties.filePickerResult = e;
        },
        key: "documentPicker",
        buttonLabel: "Select Document",
        label: "Document",
    )),
    ],
},
],
},
],
};
}
}

```

The file URL will be available on methods of the web part — such as `render()` — as `this.properties.filePickerResult.fileAbsoluteUrl`. This value can be provided to a Nutrient Web SDK instance to display the document:

```

NutrientViewer.load({
  document: this.properties.filePickerResult.fileAbsoluteUrl
  // ...
});

```



SharePoint Online file handler

File handlers instruct SharePoint to redirect to your URL of choice whenever a file with a particular extension is clicked. By using a file handler, you override SharePoint's default behavior so that PDF files stored in SharePoint are opened directly inside your application. Any edits to the opened files using your application can then be directly saved back to SharePoint.

You can find an example implementation of a file handler in our [GitHub project](#).

File handlers can be implemented using any server-side technology. The file handler will be configured to handle specific HTTP requests sent by SharePoint Online and OneDrive.

HTTP requests are sent with these activation parameters. The most relevant part is the `items` property, which is an array of Microsoft Graph URLs to the selected item(s).

You can extract the URL specified in the `items` property and fetch it from the server side, specifying a proper Microsoft Graph access token, to get its

`@microsoft.graph.downloadUrl` property as described here to be able to load the document from the client.

Here's a Node.js example that parses the activation parameters and returns the URL that should be opened:

```
// `token` is a reference to the token needed to interact with
// the Microsoft Graph API.
const response = await fetch(activationParams.items[0], {
  headers: {
    authorization: `Bearer ${token}`
  }
});

const json = await response.json();
const { "@microsoft.graph.downloadUrl": fileUrl } = json;
```



In the previous example, `fileUrl` will contain the URL needed to pass client-side to Nutrient to load:

```
NutrientViewer.load({
  document: fileUrl
```



```
// ...  
});
```

Was this helpful?

✓ YES

✗ NO

Copyright 2023 Nutrient. All rights reserved.

[About](#) [Contact](#) [Legal](#)