import { Version } from "@microsoft/sp-core-library";

import {

IPropertyPaneConfiguration,

PropertyPaneTextField,

} from "@microsoft/sp-property-pane";

import { BaseClientSideWebPart } from "@microsoft/sp-webpart-base";

import { IReadonlyTheme } from "@microsoft/sp-component-base";

import { escape } from "@microsoft/sp-lodash-subset";

import styles from "./MsGraphApiWebPartWebPart.module.scss";

import \* as strings from "MsGraphApiWebPartWebPartStrings";

import { MSGraphClient } from "@microsoft/sp-http";

import \* as MicrosoftGraph from "@microsoft/microsoft-graph-types";

export interface IMsGraphApiWebPartWebPartProps {

description: string;

}

export default class MsGraphApiWebPartWebPart extends BaseClientSideWebPart<IMsGraphApiWebPartWebPartProps> {

private \_isDarkTheme: boolean = false;

private \_environmentMessage: string = "";

protected onInit(): Promise<void> {

this.\_environmentMessage = this.\_getEnvironmentMessage();

return super.onInit();

}

//="${styles.msGraphApiWebPart} ${!!this.context.sdks.microsoftTeams ? styles.teams : ''}"

public render(): void {

this.context.msGraphClientFactory

.getClient()

.then((msgraphclient: MSGraphClient): void => {

// get information about the current user from the Microsoft Graph

msgraphclient

.api("/me")

.get((error, user: MicrosoftGraph.User, rawResponse?: any) => {

// handle the response

console.log(JSON.stringify(user));

this.domElement.innerHTML = `

<section>

<p>Display Name:${user.displayName} </p>

<p>Given Name:${user.givenName} </p>

<p>Surname Name:${user.surname} </p>

<p>Email ID:${user.mail} </p>

<p>Mobile Phone:${user.mobilePhone} </p>

</section>`;

});

});

}

private \_getEnvironmentMessage(): string {

if (!!this.context.sdks.microsoftTeams) {

// running in Teams

return this.context.isServedFromLocalhost

? strings.AppLocalEnvironmentTeams

: strings.AppTeamsTabEnvironment;

}

return this.context.isServedFromLocalhost

? strings.AppLocalEnvironmentSharePoint

: strings.AppSharePointEnvironment;

}

protected onThemeChanged(currentTheme: IReadonlyTheme | undefined): void {

if (!currentTheme) {

return;

}

this.\_isDarkTheme = !!currentTheme.isInverted;

const { semanticColors } = currentTheme;

this.domElement.style.setProperty("--bodyText", semanticColors.bodyText);

this.domElement.style.setProperty("--link", semanticColors.link);

this.domElement.style.setProperty(

"--linkHovered",

semanticColors.linkHovered

);

}

protected get dataVersion(): Version {

return Version.parse("1.0");

}

protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {

return {

pages: [

{

header: {

description: strings.PropertyPaneDescription,

},

groups: [

{

groupName: strings.BasicGroupName,

groupFields: [

PropertyPaneTextField("description", {

label: strings.DescriptionFieldLabel,

}),

],

},

],

},

],

};

}

}