The error TypeError: Cannot read properties of undefined (reading 'location') typically occurs when the window object is not accessible in the current context. This can happen if the JavaScript code is running in an environment where window is not defined (e.g., in a Node.js environment or a non-browser context). However, in your case, since this is an ASP.NET Core MVC application, the issue is likely caused by a **naming conflict** between the window object and the window variable you are using for the Kendo Window.

**Root Cause**

In your onEmailEstimateClick function, you are using the variable name window to refer to the Kendo Window:

javascript

Copy

var window = $("#emailEstimateWindow").data("kendoWindow");

This **shadows** the global window object, causing the error when you try to access window.location.href.

**Solution**

Rename the window variable to something else (e.g., kendoWindow) to avoid the naming conflict. Here's the corrected code:

javascript

Copy

function onEmailEstimateClick(e) {

e.preventDefault();

var dataItem = this.dataItem($(e.currentTarget).closest("tr"));

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Generate the Excel file by calling SaveToExcel

window.location.href = '@Url.Action("SaveToExcel", "Export")' + '?rebuiltPartNum=' + rebuiltPartNum;

// Set the subject with the rebuilt number and today's date

var today = new Date();

var subject = `Rebuilt Part Estimates for ${rebuiltPartNum} - Date ${today.toLocaleDateString()} ${today.toLocaleTimeString()}`;

// Set the form values

$("#to").val("");

$("#subject").val(subject);

$("#body").val(`Dear Sir/Madam,\nPlease find attached here with Rebuilt Part Estimate for ${rebuiltPartNum}.\n\nThank you`);

// Open the Kendo Window

var kendoWindow = $("#emailEstimateWindow").data("kendoWindow"); // Renamed variable

if (kendoWindow) {

kendoWindow.center().open();

} else {

console.error("Kendo Window not found.");

}

}

**Why This Works**

* The global window object is no longer shadowed by the local window variable.
* The Kendo Window is now referenced using the kendoWindow variable, avoiding any conflicts.

**Additional Notes**

1. **Order of Operations**:
   * The window.location.href line will trigger a navigation to the SaveToExcel action, which will download the Excel file. After the download, the page will reload unless the SaveToExcel action returns a file for download without redirecting.
2. **Kendo Window Opening**:
   * The Kendo Window will open immediately after the window.location.href line is executed. If the page reloads due to the navigation, the Kendo Window may not stay open. To avoid this, you can use an **AJAX call** to download the file instead of navigating away from the page.

**Alternative: Use AJAX to Download the File**

If you want to avoid navigating away from the page, you can use an AJAX call to download the file. Here's how:

**Updated onEmailEstimateClick Function**

javascript

Copy

function onEmailEstimateClick(e) {

e.preventDefault();

var dataItem = this.dataItem($(e.currentTarget).closest("tr"));

var rebuiltPartNum = dataItem.RebuiltStockNum;

// Generate the Excel file by calling SaveToExcel via AJAX

$.ajax({

url: '@Url.Action("SaveToExcel", "Export")',

type: 'GET',

data: { rebuiltPartNum: rebuiltPartNum },

xhrFields: {

responseType: 'blob' // Ensure the response is treated as a binary file

},

success: function (data) {

// Create a link element to trigger the file download

var link = document.createElement('a');

link.href = window.URL.createObjectURL(data);

link.download = `${rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx`;

link.click();

// Set the subject with the rebuilt number and today's date

var today = new Date();

var subject = `Rebuilt Part Estimates for ${rebuiltPartNum} - Date ${today.toLocaleDateString()} ${today.toLocaleTimeString()}`;

// Set the form values

$("#to").val("");

$("#subject").val(subject);

$("#body").val(`Dear Sir/Madam,\nPlease find attached here with Rebuilt Part Estimate for ${rebuiltPartNum}.\n\nThank you`);

// Open the Kendo Window

var kendoWindow = $("#emailEstimateWindow").data("kendoWindow");

if (kendoWindow) {

kendoWindow.center().open();

} else {

console.error("Kendo Window not found.");

}

},

error: function (xhr, status, error) {

console.error("Error downloading Excel file:", error);

}

});

}

**Why Use AJAX?**

* **No Page Reload**: The page does not reload, so the Kendo Window stays open.
* **Better User Experience**: The file is downloaded in the background, and the user can continue interacting with the page.

**Updated ExportController.cs for AJAX**

Ensure the SaveToExcel action returns a FileContentResult for the AJAX call to work correctly:

csharp

Copy

[HttpGet]

public async Task<IActionResult> SaveToExcel([FromQuery] string rebuiltPartNum)

{

try

{

// Fetch data and generate Excel file

var memoryStream = new MemoryStream();

// Your logic to generate the Excel file

// ...

// Return the file for download

return File(memoryStream.ToArray(), "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", $"{rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx");

}

catch (Exception ex)

{

\_logger.LogError(ex, "An error occurred while generating the Excel file.");

return StatusCode(500, "An error occurred while generating the Excel file.");

}

}

**Summary of Changes**

| **Change** | **Description** |
| --- | --- |
| **Renamed window Variable** | Renamed the Kendo Window variable to kendoWindow to avoid conflicts. |
| **AJAX File Download** | Used AJAX to download the file without navigating away from the page. |

This approach ensures that the Excel file is downloaded, the Kendo Window opens correctly, and the page does not reload. Let me know if you need further assistance