

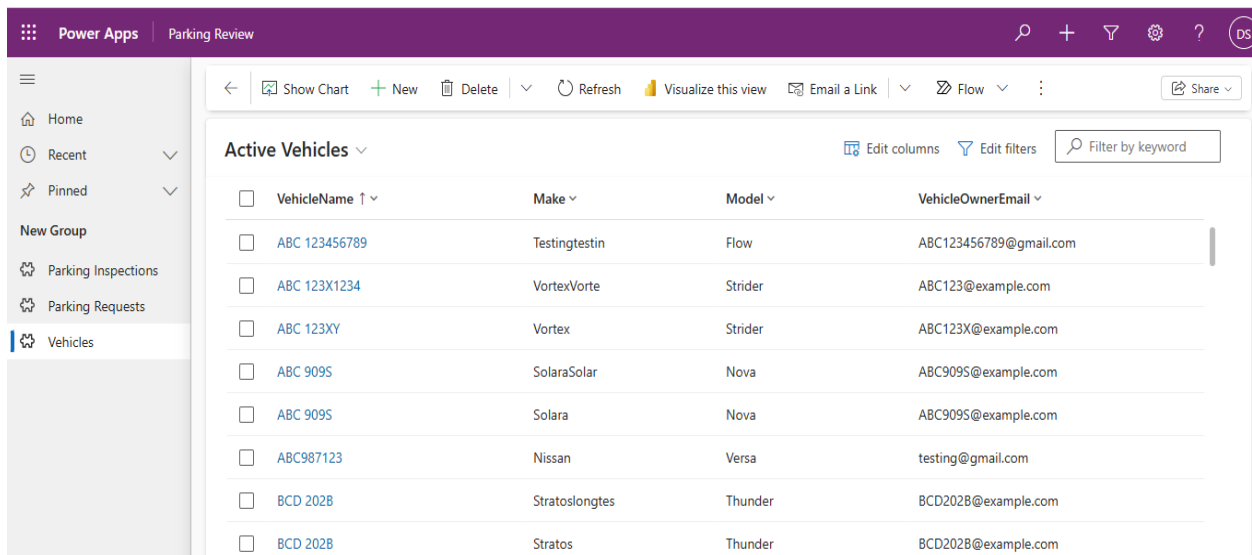
JavaScript Web Resource in PowerApps

1.Introduction:

In this guide, I've documented a simple yet powerful use case where I added a custom JavaScript Web Resource in a Power Apps Model-Driven App to extend form logic. This use case involves showing a popup alert and automatically updating a field when a user selects a vehicle in the Parking Request form.

This document walks through the exact steps I followed — from uploading the script, configuring it, and testing it in a real-world app scenario.

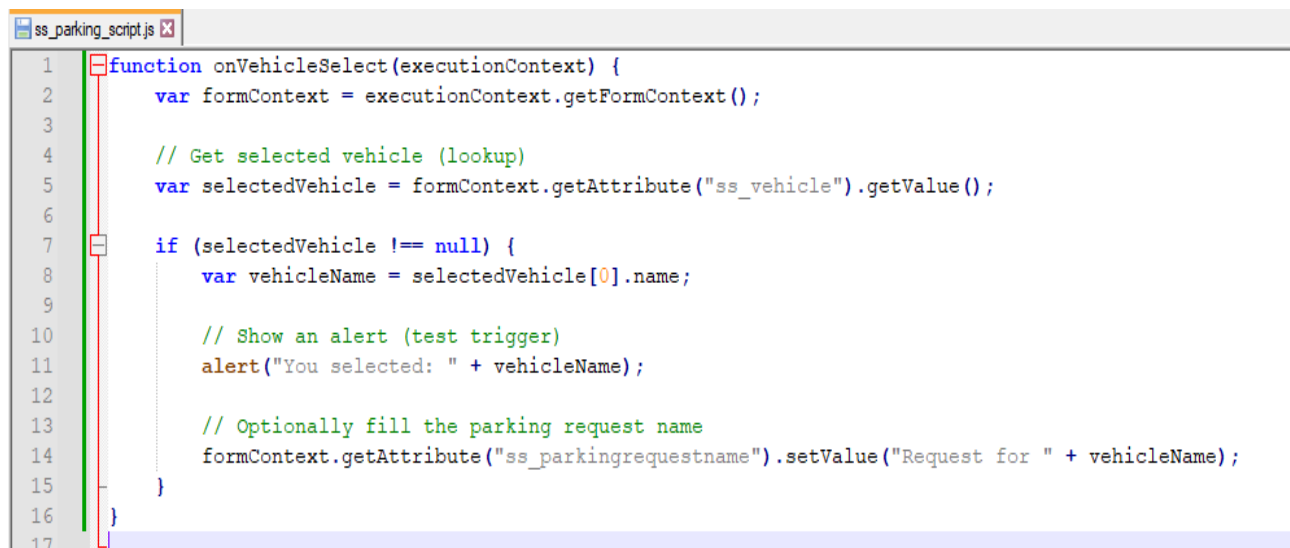
Model Driven App:



The screenshot shows the Power Apps interface for a 'Parking Review' app. On the left is a navigation pane with options: Home, Recent, Pinned, New Group, Parking Inspections, Parking Requests, and Vehicles (selected). The main area displays a table titled 'Active Vehicles' with columns: VehicleName, Make, Model, and VehicleOwnerEmail. The table contains 10 rows of vehicle data.

VehicleName	Make	Model	VehicleOwnerEmail
ABC 123456789	Testingtestin	Flow	ABC123456789@gmail.com
ABC 123X1234	VortexVorte	Strider	ABC123@example.com
ABC 123XY	Vortex	Strider	ABC123X@example.com
ABC 909S	SolaraSolar	Nova	ABC909S@example.com
ABC 909S	Solara	Nova	ABC909S@example.com
ABC987123	Nissan	Versa	testing@gmail.com
BCD 202B	Stratoslongtes	Thunder	BCD202B@example.com
BCD 202B	Stratos	Thunder	BCD202B@example.com

2.JavaScript Code Overview: The script I wrote handles the “On Change” event of a lookup field (Vehicle) and triggers an alert message while also pre-filling the Parking Request Name. This kind of dynamic interaction improves user experience and reduces manual input errors.



```
1 function onVehicleSelect(executionContext) {
2     var formContext = executionContext.getFormContext();
3
4     // Get selected vehicle (lookup)
5     var selectedVehicle = formContext.getAttribute("ss_vehicle").getValue();
6
7     if (selectedVehicle !== null) {
8         var vehicleName = selectedVehicle[0].name;
9
10        // Show an alert (test trigger)
11        alert("You selected: " + vehicleName);
12
13        // Optionally fill the parking request name
14        formContext.getAttribute("ss_parkingrequestname").setValue("Request for " + vehicleName);
15    }
16 }
17
```

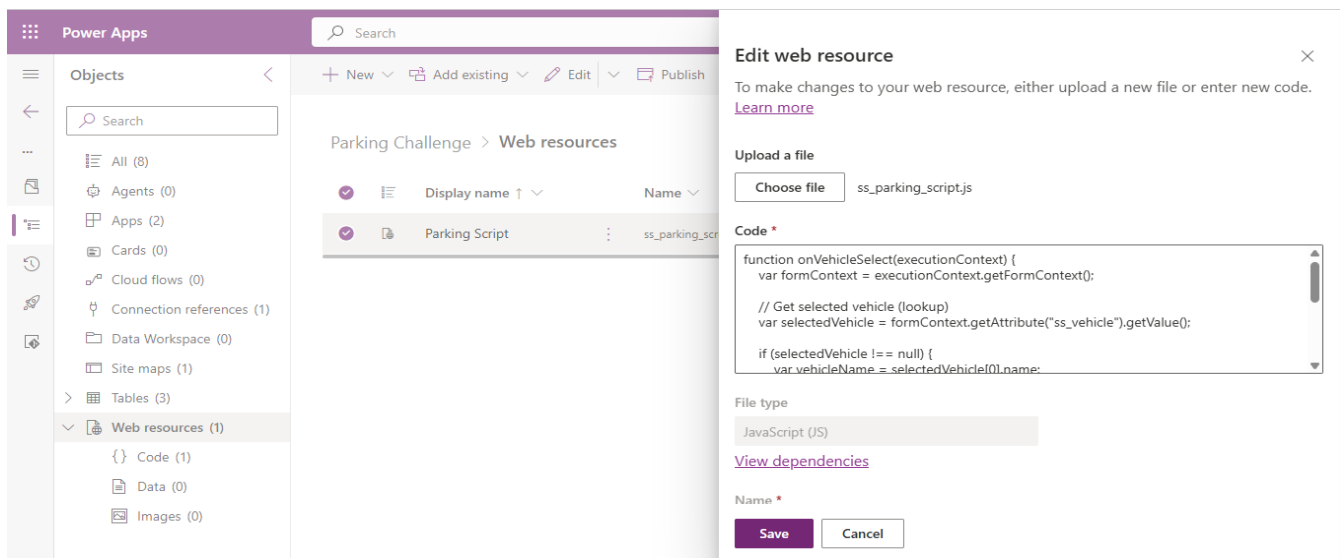
3.How to Upload the Web Resource:

Once the code was ready, I uploaded it into my solution in Power Apps as a Web Resource. This makes it reusable and available across forms.

Here's how I did it:

1. Open the Power Apps maker portal: <https://make.powerapps.com>
2. Go to **Solutions** → open your solution (e.g., *Parking Challenge*)
3. Click **+ New > Web Resource**
4. Fill the details:
 - Name: `ss_parking_script.js`
 - Type: JavaScript (JS)
 - Upload the saved .js file
5. Save and **Publish**

Web Resource upload screen



4. How to Attach to Form:

After uploading the script, I added it to the relevant form (Parking Request). This allows the script to run during field events like OnChange.

Steps I followed:

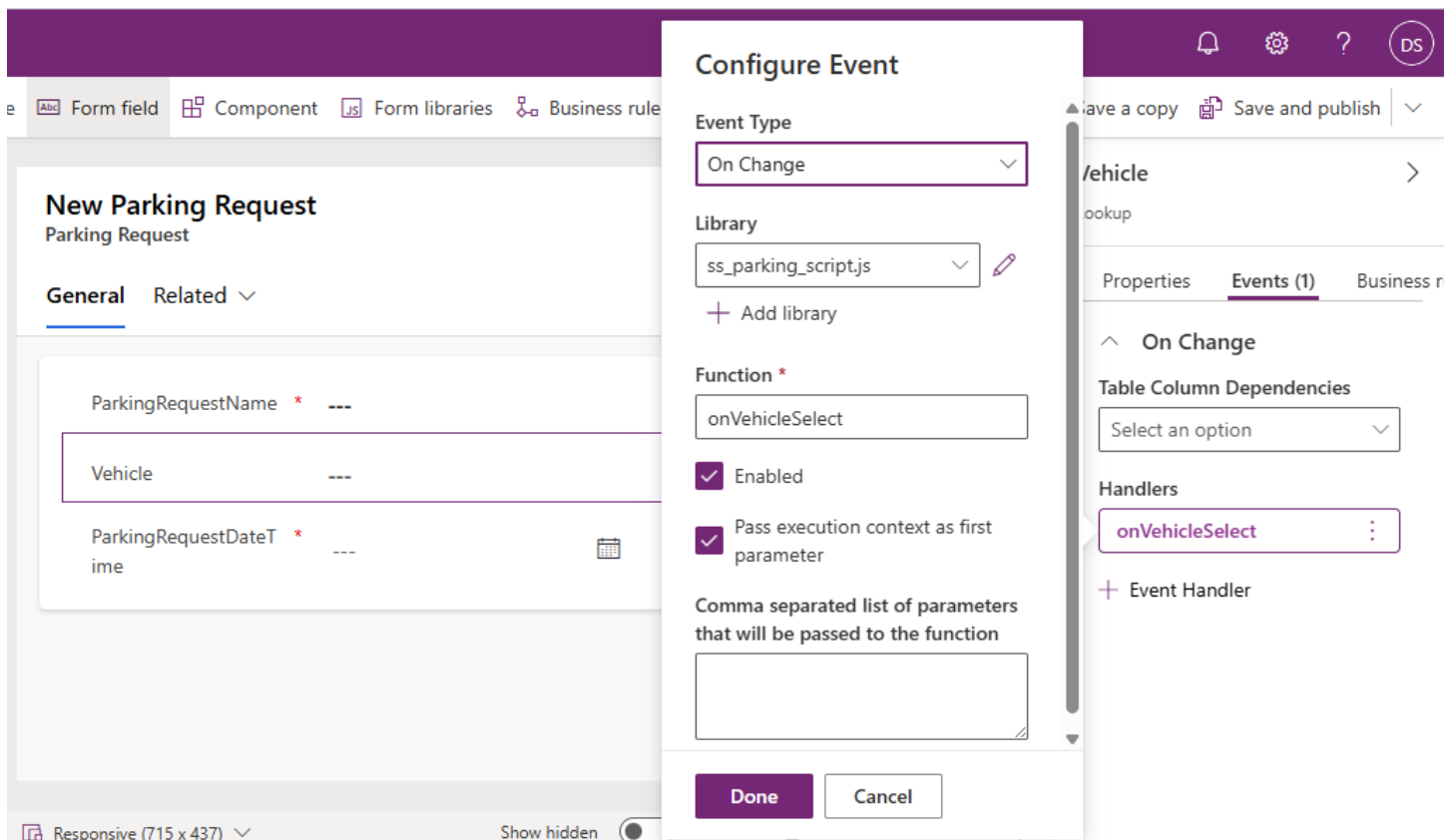
1. In the same solution, go to **Tables > Parking Request > Forms**
2. Open the **Main Form**
3. Click **Form Settings > Form Libraries**
4. Click **Add Library** → select `ss_parking_script.js`
5. Click **Done** and **Save**

5.Event Configuration:

To make the script run when the Vehicle field is changed, I added an event handler to that control.

Here's how I configured it:

1. Select the **Vehicle** field
2. Switch to the **Events** tab
3. Under OnChange:
 - Click **+ Add Event Handler**
 - Choose Library: `ss_parking_script.js`
 - Function Name: `onVehicleSelect`
 - ☒ Check "Pass execution context as first parameter"
4. Click **Done**, then **Save and Publish**



6.Testing the Setup:

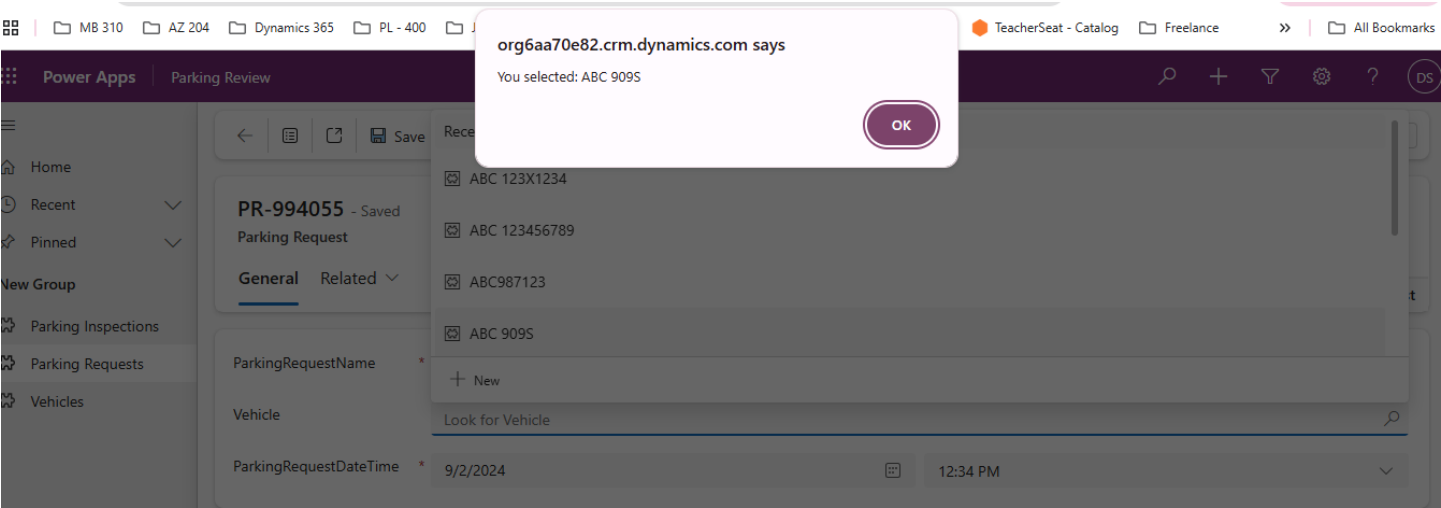
Once published, I opened my Model-Driven App and tested the Parking Request form.

When I selected a vehicle:

- An alert popped up showing the vehicle name
- The Parking Request Name field was automatically filled

This confirmed that the JavaScript was successfully integrated into the app.

An alert popped up showing the vehicle name



The Parking Request Name field was automatically filled

