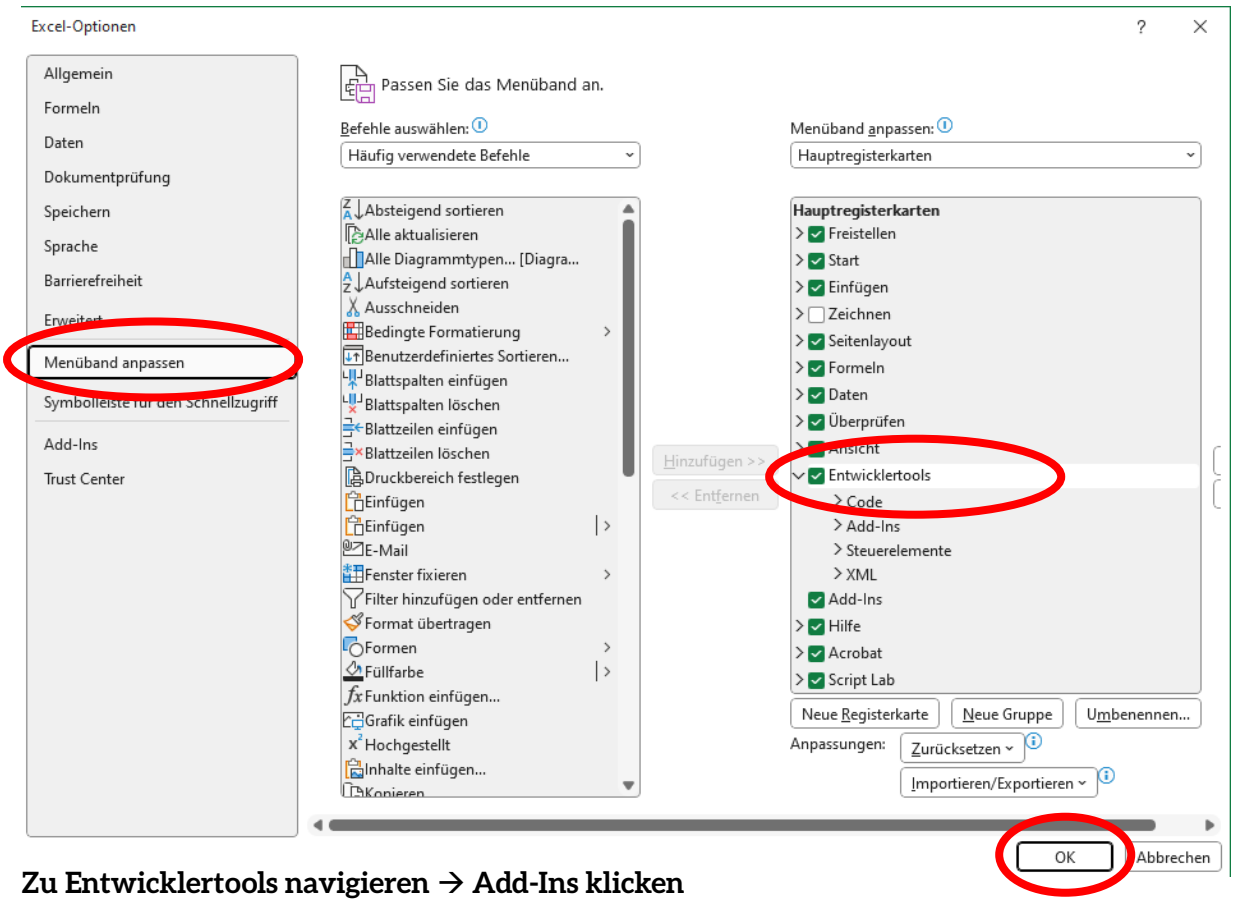
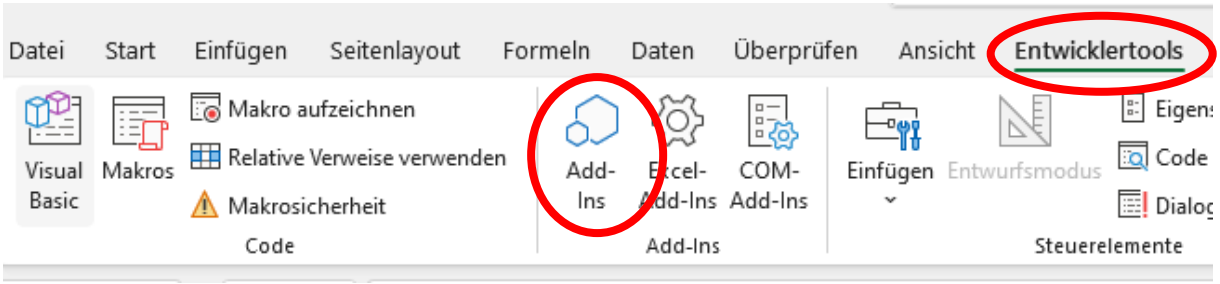
Anleitung

Auslagenerstattung zum Einscannen in Excel einrichten:

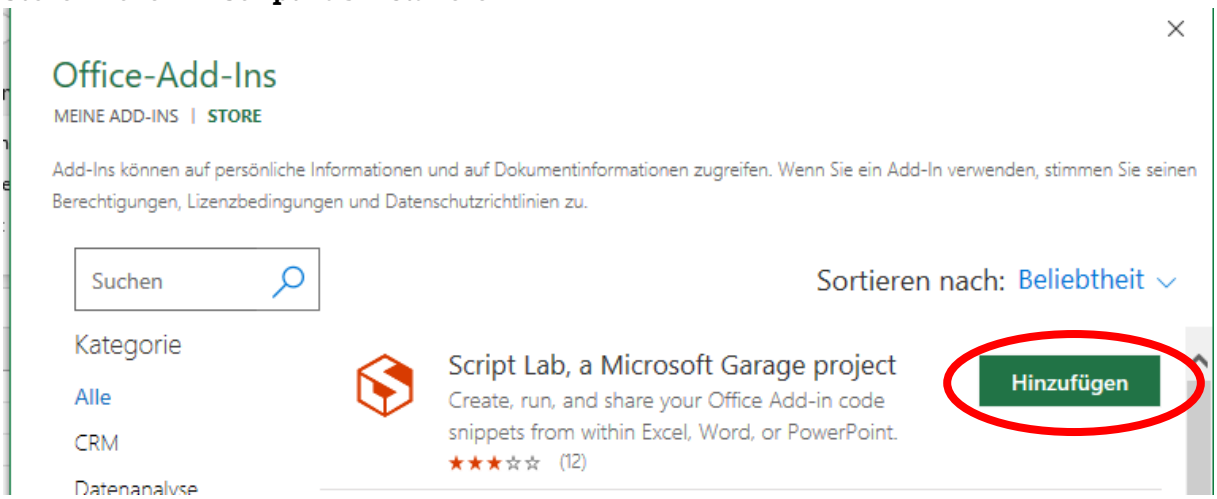
1. **Excel öffnen**
2. **EVN-Datei auswählen**
3. Datei **Klicken**
4. Optionen **Klicken**
5. **Menübandanpassen klicken → Entwicklertools aktivieren → ok klicken**



6. **Zu Entwicklertools navigieren → Add-Ins klicken**

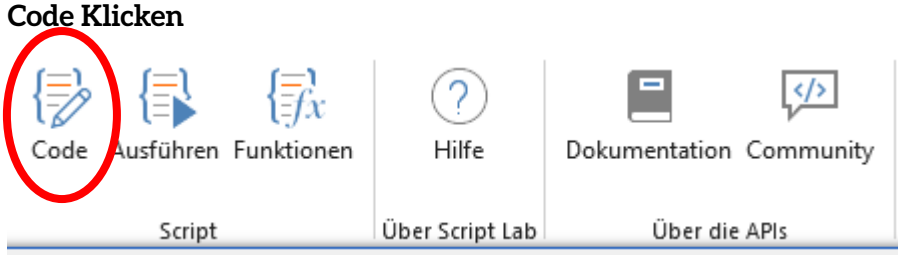


7. **Store klicken → Script Lab Instalieren**



8.  **Klicken**

9. **Code Klicken**



10. **Untenstehenden Code bei Script einfügen:**

```
// Ensure jQuery is loaded and DOM is ready
$(document).ready(function() {
  console.log("jQuery is working, DOM is ready.");

  // Dynamically load the HTML5 QR Code library using jsDelivr CDN
  $.getScript("https://cdn.jsdelivr.net/npm/html5-qrcode/minified/html5-qrcode.min.js")
    .done(function(script, textStatus) {
      console.log("QR Code scanner library loaded successfully.");

      // Check if Html5Qrcode is defined
      if (typeof Html5Qrcode !== "undefined") {
        console.log("Html5Qrcode is defined and ready to use.");

        // Attach event handlers for start and stop buttons
        $("#start-scan").on("click", () => tryCatch(startQRScanner));
        $("#stop-scan").on("click", () => tryCatch(stopQRScanner));
      } else {
        console.error("Html5Qrcode is still undefined after loading the script.");
      }
    })
    .fail(function(jqxhr, settings, exception) {
      console.error("Failed to load QR Code scanner library: ", exception);
```

```
  });

let qrReader = null;
let scanningEnabled = true; // Flag to enable or disable scanning
let currentRow = 2; // Starting row for inserting data into Excel
let timerInterval = null; // Variable to hold the timer interval
let countdownTime = 5; // Countdown time in seconds

// Function to create a delay using Promise
function delay(ms) {
  return new Promise((resolve) => setTimeout(resolve, ms));
}

function startQRScanner() {
  console.log("Starting QR scanner...");

  if (!qrReader) {
    qrReader = new Html5Qrcode("qr-reader");
  }

  // Show stop button, hide start button
  $("#start-scan").hide();
  $("#stop-scan").show();

  qrReader
    .start(
      { facingMode: "environment" }, // Use rear camera (for mobile), or try without constraints
      {
        fps: 10, // frames per second for scanning
        qrbox: 300 // QR code scanning box size
      },
      async (qrCodeMessage) => {
        if (!scanningEnabled) return; // Ignore if scanning is disabled

        console.log(`QR Code detected: ${qrCodeMessage}`);

        // Decode the URI
        try {
          const decodedURI = decodeURIComponent(qrCodeMessage);
          console.log(`Decoded URI: ${decodedURI}`);

          // Insert decoded URI into Excel
          insertQRCodeIntoExcel(decodedURI);

          // Increment the current row for the next scan
          currentRow++;

          // Disable scanning temporarily
          scanningEnabled = false;
```

```
72.
73.          // Show timer and start countdown
74.
75.          $("#timer").show(); // Show the timer display
76.          startCountdown();
77.
78.          // Wait for 5 seconds before re-enabling scanning
79.          await delay(5000); // 5000 milliseconds = 5 seconds
80.
81.          // Re-enable scanning
82.          scanningEnabled = true;
83.          console.log("Scanning re-enabled.");
84.        } catch (error) {
85.          console.error("Error decoding QR code message:", error);
86.        }
87.       },
88.       (errorMessage) => {
89.         console.warn(`QR Code scan error: ${errorMessage}`);
90.       }
91.     )
92.     .then(() => {
93.       console.log("Camera started successfully.");
94.     })
95.     .catch((err) => {
96.       console.error(`Unable to start QR code scanner: ${err}`);
97.     });
98.   }
99.
100.          function stopQRScanner() {
101.            if (qrReader) {
102.              qrReader
103.                .stop()
104.                .then(() => {
105.                  console.log("QR Code scanning stopped.");
106.                  // Show start button, hide stop button
107.                  $("#start-scan").show();
108.                  $("#stop-scan").hide();
109.                  $("#timer").hide(); // Hide the timer display
110.                })
111.                .catch((err) => {
112.                  console.error("Failed to stop QR code scanner: ", err);
113.                });
114.            }
115.          }
116.
117.          // Function to start countdown timer
118.          function startCountdown() {
119.            // Clear any existing timer interval
120.            clearInterval(timerInterval);
121.            countdownTime = 5; // Reset countdown to 5 seconds
```

```
122.             $("#countdown-
    display").text(countdownTime); // Display initial countdown time
123.
124.             // Update timer display every second
125.             timerInterval = setInterval(() => {
126.               countdownTime--; // Decrement countdown time
127.               $("#countdown-
    display").text(countdownTime); // Update countdown display
128.
129.               if (countdownTime <= 0) {
130.                 // When countdown reaches zero
131.                 clearInterval(timerInterval); // Clear the interval
132.                 $("#timer").hide(); // Hide the timer display
133.                 scanningEnabled = true; // Re-enable scanning
134.                 console.log("Scanning re-enabled.");
135.                 $("#start-scan").show(); // Show start button again
136.                 $("#stop-scan").hide(); // Hide stop button
137.               }
138.             }, 1000); // Update every second
139.           }
140.
141.           async function insertQRCodeIntoExcel(decodedURI) {
142.             // Split the decoded URI into parts based on the '|'
143.             const parts = decodedURI.split("|");
144.
145.             // Make sure there are enough parts to avoid errors
146.             if (parts.length < 10) {
147.               console.error("Decoded URI does not have enough parts:", parts);
148.               return; // Early exit if there are not enough parts
149.             }
150.
151.             return Excel.run(function(context) {
152.               const sheet = context.workbook.worksheets.getActiveWorksheet();
153.
154.               // Insert values into the respective columns
155.               sheet.getRange(`B${currentRow}`).values = [[parts[0] || ""]]; // Everyth
    ing till the first |
156.               sheet.getRange(`C${currentRow}`).values = [[parts[1] || ""]]; // Everyth
    ing between the first and second |
157.
158.               // Check the conditions for inserting into columns D, E, F, G, and H
159.               if (parts[9] === "1") {
160.                 // Between the 9th and 10th |
161.                 if (parts[10] === "2") {
162.                   // Between the 10th and 11th |
163.                   sheet.getRange(`D${currentRow}`).values = [[parts[2] || ""]];
164.                 } else if (parts[11] === "3" || parts[10] === "3") {
165.                   // Between the 11th and 12th |
166.                   sheet.getRange(`E${currentRow}`).values = [[parts[2] || ""]];
167.                 } else if (parts[12] === "4" || parts[10] === "4") {
```

```
                // Between the 12th and 13th |
                sheet.getRange(`F${currentRow}`).values = [[parts[2] || ""]];
              } else if (parts[13] === "5" || parts[10] === "5") {
                // Between the 13th and 14th |
                sheet.getRange(`H${currentRow}`).values = [[parts[2] || ""]];
              } else if (parts[14] === "6" || parts[10] === "6") {
                // Between the 14th and 15th |
                sheet.getRange(`G${currentRow}`).values = [[parts[2] || ""]];
              }
            }

            return context.sync().then(function() {
              console.log("Data inserted into the worksheet.");
            });
          });
        }

        /** Default helper for invoking an action and handling errors. */
        function tryCatch(callback) {
          Promise.resolve()
            .then(callback)
            .catch(function(error) {
              console.error(error);
            });
        }
      });

```

11. Untenstehenden Code bei HTML einfügen:

```
<section class="samples ms-font-m">
  <p class="ms-font-m">This sample demonstrates basic QR Code scanning and data insertion into Excel.</p>
  <h3>Try it out</h3>
  <button id="start-scan" class="ms-Button" >
      <span class="ms-Button-label">Start Scanning</span>
   </button>
  <button id="stop-scan" class="ms-Button" style="display: none;">
      <span class="ms-Button-label">Stop Scanning</span>
   </button>
  <div id="qr-reader" style="width: 600px;"></div>
</section>

<section class="ms-font-m">

  <div id="timer" style="display: none;">Next scan available in: <span id="countdown-display">5</span> seconds</div>
</section>
```

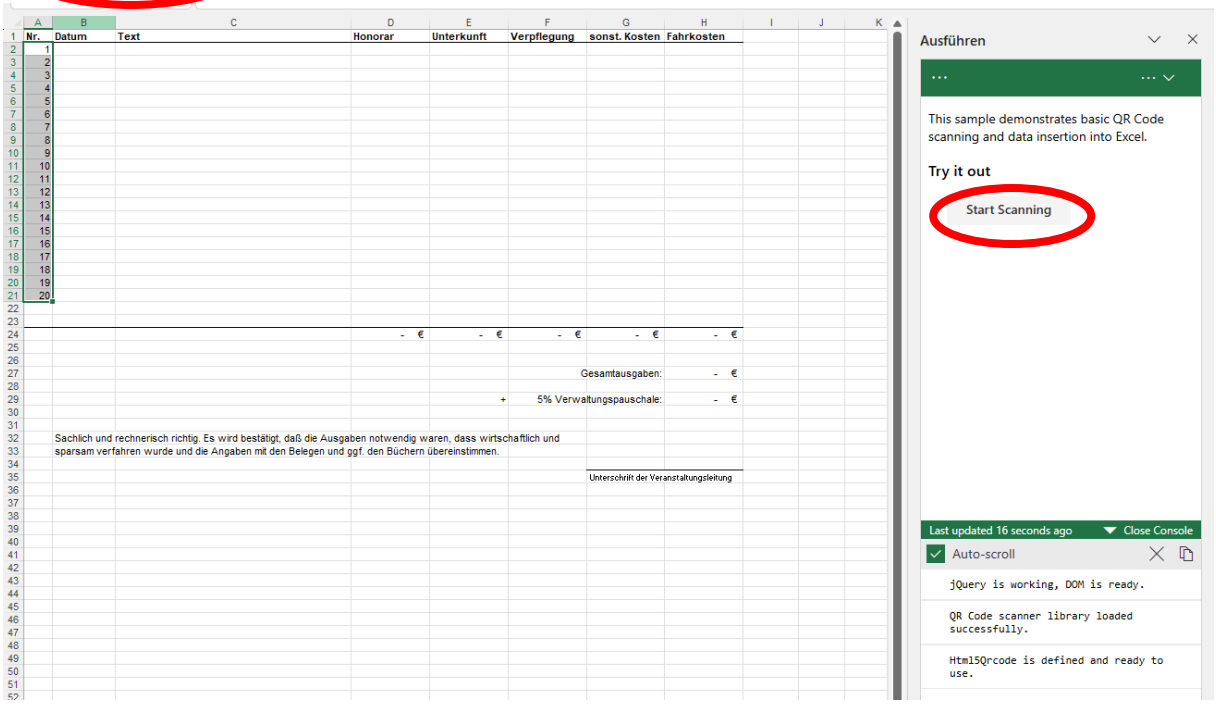
12. **Untenstehenden Code bei CSS einfügen:**

```
section.samples .ms-Button, section.setup .ms-Button {
    display: block;
    margin-bottom: 5px;
    margin-left: 20px;
    min-width: 80px;
}

#qr-reader {
    margin-top: 10px;
}

```

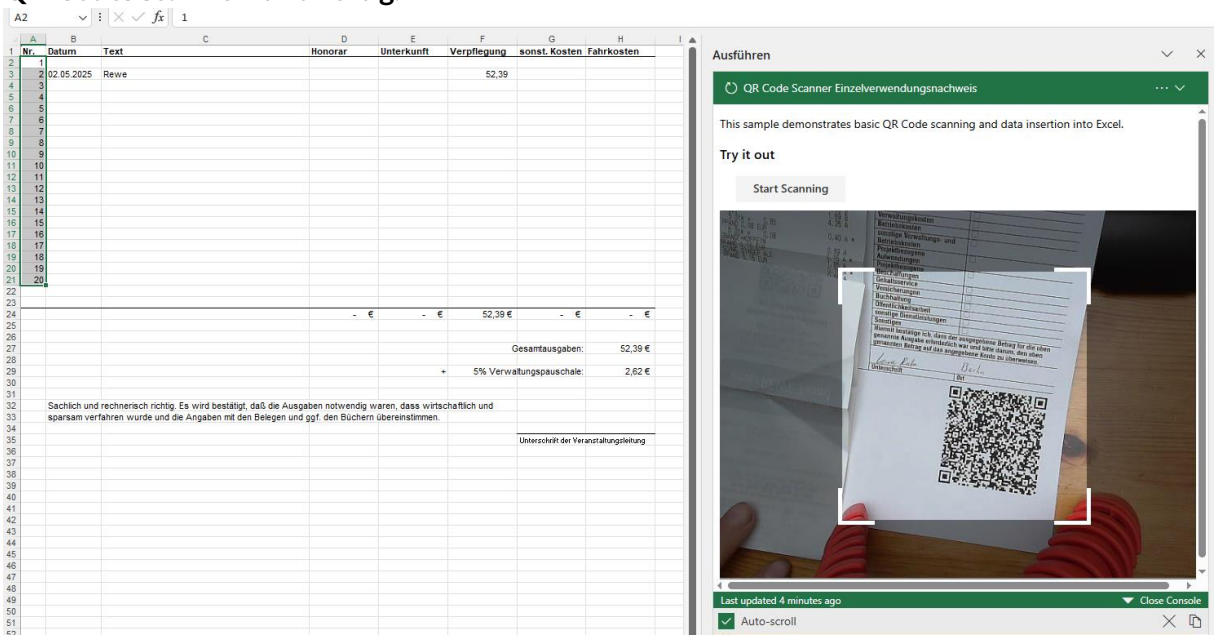
13. **Untenstehenden Code bei Libraries einfügen:**

```
https://appsforoffice.microsoft.com/lib/1/hosted/office.js
@types/office-js

office-ui-fabric-js@1.4.0/dist/css/fabric.min.css
office-ui-fabric-js@1.4.0/dist/css/fabric.components.min.css

core-js@2.4.1/client/core.min.js
@types/core-js

jquery@3.1.1
@types/jquery@3.3.1

```

13. **Speichern und starten klicken → Start Scanning klicken**



14. **QR-Codes scannen und fertig:**



Einmal eingerichtet müsst ihr beim nächsten Mal nur die Schritte 13. Und 14. Ausführen um eure Auslagenerstattung einzuscannen.