Extracting pages from a PDF document and saving them as separate image files, JavaScript edition with async

devblogs.microsoft.com/oldnewthing/20170628-00

June 28, 2017

Raymond Chen

Last time, we converted the JavaScript version of the PDF Document sample program so that it saved the pages to disk as image files. The asynchonous behavior was expressed via Promises. Today we'll use the `async` and `await` keywords which didn't make ECMAScript 7, but may make it into ECMAScript 8. Support for it arrived in Microsoft Edge as an experimental feature back in 2015.

```
async function viewPage() {
  WinJS.log && WinJS.log("", "sample", "status");

  var pageNumber = parseInt(pageNumberBox.value, 10);
  if (isNaN(pageNumber) || (pageNumber < 1) ||
    (pageNumber > pdfDocument.pageCount)) {
    WinJS.log && WinJS.log("Invalid page number.", "sample", "error");
    return;
  }

  output.src = "";
  progressControl.style.display = "block";

  // Convert from 1-based page number to 0-based page index.
  var pageIndex = pageNumber - 1;

  var page = pdfDocument.getPage(pageIndex);

  var picker = new Windows.Storage.Pickers.FileSavePicker();
  picker.fileTypeChoices["PNG image2"] = [".png"];
  var outfile = await picker.pickSaveFileAsync();
  if (outfile) {
    var transaction = await outfile.openTransactedWriteAsync();
    var options = new PdfPageRenderOptions();
    options.destinationHeight = page.size.height * 2;
    options.destinationWidth = page.size.width * 2;
    await page.renderToStreamAsync(transaction.stream, options);
    transaction.close();
  }
  page.close();
  progressControl.style.display = "none";
}
```

The `async` and `await` keywords are analogous to their C# counterparts. Declaring a function as `async` causes it to return a Promise whose result is the nominal type of the function. Inside an `async` function, you can use the `await` keyword to cause the continuation to be connected to the resolution of the Promise you are awaiting.

There's not much interesting to discuss here; it's a straightforward translation of the C# sample. Note that JavaScript doesn't have a `using` keyword, so we have to `close()` the closable objects manually.

Next time, we'll move on to C++/CX.

Raymond Chen

Follow