IN THIS CHAPTER

- Recognizing the differences between Office Scripts and VBA
- Understanding the requirements and limitations of Office Scripts
- Seeing what Office Scripts can do for you

Chapter 1
Introducing Office Scripts for Excel

Whether you're new to Excel programming or a seasoned veteran, you've likely heard about Excel's native programming language, *VBA (Visual Basic for Applications)*. Every Office application has its own version of the VBA programming language. For decades, developers and business analysts have used VBA to automate the Microsoft suite of applications.

Then along came Office 365, a set of online business applications allowing for seamless sharing and collaboration of Excel, Word, and PowerPoint documents. With Office 365, you can create an online Excel document that can be used by multiple people simultaneously, can be used on tablets as well as PCs, and can be integrated with the broader suite of Microsoft cloud-based platforms such as Power Automate.

Ah, but there's a catch. Because online Excel documents are not stored on your local PC, they don't have access to your local operating system. More importantly, they don't have access to the fundamental requirements needed to run VBA. Therefore, it's impossible to run VBA macros and programs with Excel online documents.

Enter Office Scripts, a language based on TypeScript, an open-source superset of JavaScript. (JavaScript is a lightweight programming language that runs directly in the browser, allowing developers to manipulate web page content, respond to

user events, and communicate with servers without reloading the page.) Because Office Scripts has TypeScript at its core, it compiles to JavaScript, which means it can be used in online cloud environments. In short, with Office Scripts, unlike VBA, we can automate Excel online workbooks.

Before you start your journey into using Office Scripts, it's important to understand some of its requirements and the fundamental programming differences you'll encounter when working in this new realm of Excel automation. This chapter paves the way for everything else that follows and gives you a feel for how Office Scripts fit into the big picture of Excel development. Be patient and resist the urge to jump to Chapter 2 instead.

Seeing How VBA and Office Scripts Differ

In this section, you explore the fundamental differences between Office Scripts and VBA.

Offline versus online

Office Scripts will not replace VBA anytime soon. Office Scripts is designed for the automation of online cloud-based Office documents, while VBA still holds a fundamental role in the development of offline desktop applications.

If your application needs to write to your file system, respond to keyboard or mouse clicks, or interact with your operating system, VBA is the language you would use. VBA would also need to be used if your solution is installed as an Excel add-in (a supplemental program that integrates directly into the Excel ribbon interface).

If your organization runs on a modern cloud environment — meaning your Excel workbooks are saved online, available from any device, and connected to other Microsoft services — you need Office Scripts for online development. Online development with Office Scripts allows your automated processes to be managed in a controlled cloud environment, to be accessed by multiple users, and to be used across various platforms such as tablets and handheld devices.

Table 1-1 is a quick reference to help you choose between VBA and Office Scripts when tackling a new project.

TABLE 1-1 Using VBA versus Office Scripts

Project Requirement	VBA	Office Scripts
Write to the local PC file system	Yes	
Respond to keyboard or mouse events	Yes	
Provide a custom ribbon interface	Yes	
Respond to workbook, worksheet, or other events	Yes	
Install as an add-in	Yes	
Automate shared Excel online workbooks		Yes
Function in a controlled cloud environment		Yes
Integrate with SharePoint lists and files		Yes
Integrate with Power Automate cloud workflows		Yes

Security

Because VBA runs on your local PC, VBA macros, especially from untrusted sources, can pose significant security risks. The unfettered access to the operating system that VBA provides can open the possibility for malicious code to manipulate files, access system resources, and install malware. To mitigate this threat, Excel requires users to explicitly enable macros for each file. If macros are disabled, VBA code won't run.

In contrast, Office Scripts run in the cloud, so they don't access your local machine or file system. Because they don't access local resources, the risk of file-based malware is reduced, making scripts inherently more secure.

A FEW WORDS ABOUT TERMINOLOGY

Excel online programming terminology can be confusing. Search for information about this topic online, and you may see the term *Office Scripts* or *Excel Scripts*. What's the difference? Just like Microsoft Office has an overarching programming language called VBA, you can think of *Office Scripts* as the overarching programming language for online Office development. *Excel Scripts* is often used as a shorthand for referring to the ExcelScript Package, which is the core API (application programming interface) in Office Scripts and provides all the objects and methods necessary to interact with Excel. For our purposes, however, Office Scripts for Excel and Excel Scripts mean the same thing.

Events

VBA supports a wide range of event triggers. For example, VBA can trigger a macro to run when a workbook is opened, a worksheet is selected, or a specific cell is changed. Event triggers give VBA developers great flexibility when designing the user experience for their Excel application.

As of this writing, Office Scripts has no native support for events. You can't trigger a script to run when a workbook opens or a cell changes. A script can run only when a user explicitly starts it. In Part 4, however, you discover how to simulate events by using Power Automate flows to trigger scripts to run.

Language

VBA uses the proprietary Microsoft scripting language with application-specific objects and methods and rich support for Windows API calls. VBA provides a wide array of reference libraries that let you interact with practically every aspect of the local environment in which it runs.

Office Scripts uses TypeScript and application-specific API packages, which compile down to cloud-compatible JavaScript. Because the objects and methods come from managed cloud resources and API packages, you'll always have the latest versions of Excel and Office Scripts without needing to install anything.

Types of procedures

In VBA, you can write code in subprocedures, functions, or events. A subprocedure executes one line of code at a time until the procedure finishes. Functions are like subprocedures in that they execute each line until complete, except they can return a value back to the user. For example, you can create a function that adds two numbers and returns the answer to a specific cell. Finally, VBA can be written as events that trigger when an action is taken (a workbook is opened, a worksheet is selected, a specific cell is changed, and so on).

With Office Scripts, you're limited to using only functions. Every Office Script for Excel requires a minimum of one function called `main` with a parameter that represents the current active Excel workbook. If you record a script, the first line of the script will always be the following:

```
Function main(workbook: ExcelScript.Workbook)
```

When you run a script, it looks for this function and passes the active workbook into the function. You can add functions as needed, but the `main` function is required.

REMEMBER

A script has access to only the workbook in which it is running. Even something as simple as copying a value from one workbook to another can't be done with a single script.

Code storage

Code written in VBA is embedded in the `.xlsm` workbook in which it's saved. This embedded code travels with the workbook when it's emailed or saved to another location. Users have immediate access to VBA code in an .xlsm file when they enable macros for that file.

In contrast, Office Scripts offers centralized, cloud-based script management. Scripts are not embedded in the workbook. Instead, they're stored as separate `.osts` (Office Scripts TypeScript) files on your OneDrive account under Documents/Office Scripts/. You can run these scripts from any workbook as long as you're logged in to the same OneDrive account and have permission to access and edit the workbook.

Scripts can also be saved on SharePoint, where you can access them if you have access to that site. You can view, manage, and share scripts through the Excel Office Scripts interface.

Script triggers

You can trigger a VBA script in a variety of ways. You can start the code manually from the Macros dialog box, assign your macro to a shape or button, use an event to trigger your code, or use a worksheet function in a cell.

Office Scripts can be started manually via the Automate tab or from the Code Editor task pane. Additionally, your script can be assigned to a button. In Part 4, you discover how to trigger a script by using Power Automate flows.

Understanding the Requirements and Limitations of Office Scripts

Now that you understand where Office Scripts fits into the world of Excel development, let's take a moment to review some of the requirements and limitations of Office Scripts.

TIP

The list of requirements and limitations may change as Microsoft continues to develop Office Scripts. For updates, visit `https://learn.microsoft.com/en-us/office/dev/scripts/testing/platform-limits`.

Subscription requirement

Office Scripts is available only with a Microsoft 365 subscription. To get started, you need one of the following licenses:

- Microsoft 365 Personal or Family
- Office 365 Business or Business Premium
- Office 365 ProPlus or ProPlus for Devices
- Office 365 A3 or A5
- Office 365 Enterprise E1, Enterprise E3, or Enterprise E5
- Office 365 F3

REMEMBER

If you're using a Personal or Family plan, the features that allow Office Scripts to integrate Power Automate are not available. You can check your current subscription by signing into `https://account.microsoft.com`.

Technical and organizational limitations

Some organizations turn off automation features by default for security reasons. If you find that you can't run Office Scripts, check with your Microsoft 365 administrator to ensure that Office Scripts for Excel is enabled and that no group policy settings are limiting its use.

To see the Automate tab when using Excel online, third-party cookies must be enabled on your browser. If you don't see the Automate tab, check your browser settings or speak with your IT department to ensure that third-party cookies have not been disabled.

Data limits

If you work with a range of cells in an Excel worksheet and need to store that range object in memory, it can't have more than 5 million cells. In addition, data transferred between the script and the Excel workbook can't exceed 5MB. If you exceed one of these Excel data limits, you'll receive an error message that reads

"The response payload size has exceeded the limit." In these rare cases, you'll need to split your ranges or data into smaller, more manageable chunks.

You are limited to 1,600 calls to the Run script action per day. This limit is usually an issue only with scheduled Power Automate flows that run Office Scripts.

Finally, all scripts have a 120-second timeout. A long-running script will fail if all the code is not resolved within this time limit. In this case, you would need to optimize or split your script into separate scripts.

Meager documentation

Documentation for Office Scripts is sparse. Unlike VBA, which has been around for decades, Office Scripts code samples and object model diagrams will be scarce until more people adopt the program. Fortunately, Microsoft's built-in Action Recorder can assist you in writing Office Scripts based on your keyboard entries and mouse clicks. You explore Action Recorder in the next chapter.

Benefits of Office Scripts

With it robust features, Office Scripts enable countless ways to improve data processing and gain efficiencies. I end this chapter with five key benefits of leveraging Office Scripts:

- **Automate a recurring task.** Are you producing the same report week after week using the same tedious steps? You can turn those steps into a script that can be run on demand. And because the steps are codified, you'll achieve the same results every time with no chance for human error.
- **Automate a repetitive task.** Excel work is often repetitive. When you need to do the same set of tasks for every worksheet in a workbook, use Office Scripts instead.
- **Optimize work with functions.** Use Office Scripts to create functions that perform custom calculations, run custom transformations on text, or apply custom formatting based on certain conditions. You can extend the capabilities of your workbooks beyond Excel's built-in features.
- **Simplify the user experience.** Oftentimes, your workbooks will be used by people who know little to nothing about Excel. You can employ Office Scripts to build an interface that provides users with friendly buttons that perform actions for them. You'll be surprised at how much you can cut down on

human error by relying on your scripts to guide users through a planned workflow.

- **Interact with other applications.** One of the more exciting aspects of Office Scripts is how it can be integrated in your cloud workflows. With the help of Power Automate flows, you can integrate your Excel workbooks with SharePoint, automatically send emails through Outlook, and save parts of your Excel workbook to a PDF stored in the cloud. The best part is that these Power Automate flows can be configured to run on a schedule, freeing you up to work on other things.

Are you excited yet? Good. *Now* you can jump to Chapter 2.