VBA Tutorial: Mastering the Weekday Function for Date Calculations

Authored by

Mohammed looti

November 9, 2025

The mastery of date and time manipulation is a foundational requirement for developing robust and advanced procedures in **VBA** (Visual Basic for Applications). Among the essential tools for this task is the **Weekday** function. This powerful function provides developers with a streamlined method to retrieve the day of the week, returning the result as an Integer value derived from any given date input.

In business logic, scheduling routines, and conditional processing within spreadsheets, knowing the specific day of the week a date falls on is frequently critical. The **Weekday** function addresses this need by simplifying complex date calculations. By outputting a clear numerical index (ranging from 1 to 7), the function allows your code to execute specific actions easily--for instance, differentiating between weekdays and weekends, or ensuring a routine only runs on a Monday.

Understanding the VBA Weekday Function

The core objective of the **Weekday** function is to efficiently translate a date--which internally is stored as a date serial number within Excel and VBA--into its corresponding numerical day-of-week representation. This numerical index is indispensable for automating tasks where the precise day determines subsequent processing steps, such as automating financial reports that must close on the last Friday of the month or adjusting workflows based on the current day.

While Excel offers a similar function directly usable on the worksheet, the VBA **Weekday** function is specifically optimized for integration within procedural code blocks. Its output simplifies logical comparisons dramatically. Instead of writing code that attempts to compare a date to a text string like "Saturday," the developer can rely on a simple numerical test, such as checking if the function returns the integer 7 (assuming the standard indexing system is in use).

For scenarios that require implementing this functionality across a range of cells, such as iterating through a column of dates, developers typically access the function through the WorksheetFunction object model. This approach ensures your macro seamlessly combines native Excel worksheet capabilities with powerful VBA procedural logic, greatly enhancing the flexibility and compatibility of your solution within the Microsoft application ecosystem.

Syntax and Arguments of the Weekday Function

The syntax required to use the **Weekday** function is concise and straightforward. It requires one mandatory argument--the date being evaluated--and includes a single optional argument that is vital for controlling the indexing standard used to assign the day numbers. Understanding and correctly utilizing this optional argument is essential, as it dictates whether Sunday, Monday, or another day receives the starting index of 1.

The fundamental structure is defined as: `Weekday(Date, )`. The initial `Date` parameter must be a

valid date expression, a variable containing a date type, or a recognized date serial number. If the provided date parameter is null or cannot be evaluated as a valid date, the function will logically return `Null`. Developers must ensure the input is correctly formatted to prevent unexpected runtime errors in their procedures.

The optional `FirstDayOfWeek` argument is what determines the start of the week and, consequently, controls which integer maps to which day. By default, if this argument is omitted, VBA uses the constant `vbSunday`, meaning Sunday is invariably represented by the number 1. However, developers can leverage several built-in constants to alter this behavior, ensuring compliance with international standards or specific project mandates. For example, explicitly using `vbMonday` sets Monday as 1 and Sunday as 7, which is the common standard in many European and ISO contexts. Other specialized constants are available, allowing the week to start on Tuesday (`vbTuesday`), Wednesday (`vbWednesday`), and so forth.

Decoding the Return Values (The Day Index System)

The **Weekday** function guarantees the return of an Integer value bounded between 1 and 7. Crucially, the practical interpretation of these numerical results is entirely dependent on the specific indexing standard selected via the optional `FirstDayOfWeek` argument. If the developer chooses to omit the argument, or if the constant `vbSunday` is explicitly used, the following universally recognized default mapping applies:

1: Sunday
2: Monday
3: Tuesday
4: Wednesday
5: Thursday
6: Friday
7: Saturday

For the development of robust and reliable code, it is absolutely paramount that the developer maintains full awareness of the indexing system currently being used. If the resulting code is intended for deployment in a global environment, or if it must interface with external systems or databases that employ differing definitions of the start of the week, explicitly specifying the `FirstDayOfWeek` argument is strongly advised. This practice eliminates potential ambiguities and guarantees consistent results, irrespective of the user's local system settings or regional defaults.

While there is a constant available, `vbUseSystemDayOfWeek`, which allows the week start to be dictated by the user's default Excel or operating system settings, utilizing hardcoded constants (e.g., `vbMonday` or `vbSunday`) offers the greatest degree of control and predictability within mission-critical macros. Predictability is always favored over reliance on variable user environments when

writing automation procedures.

Practical Application: Retrieving Weekday Indices via VBA Macro

To clearly demonstrate the function's utility, let us consider a typical scenario: an Excel worksheet contains a list of dates in Column A, and the requirement is to automatically populate Column B with the corresponding day index for every date listed. This task necessitates iterating through the designated range of dates and systematically applying the **Weekday** function to each cell.

The following VBA procedure, implemented as a standard `For...Next` loop, illustrates an efficient and clean method to achieve this data transformation. This specific procedure is designed to read date values from the range **A2:A9** and subsequently write the calculated integer indices into the parallel range **B2:B9**. Note the standard practice of accessing the function via `WorksheetFunction.Weekday` when interacting directly with cell values within a sheet-level procedure.

The complete implementation of the procedure is provided below:

Sub FindWeekday()

Dim i As Integer

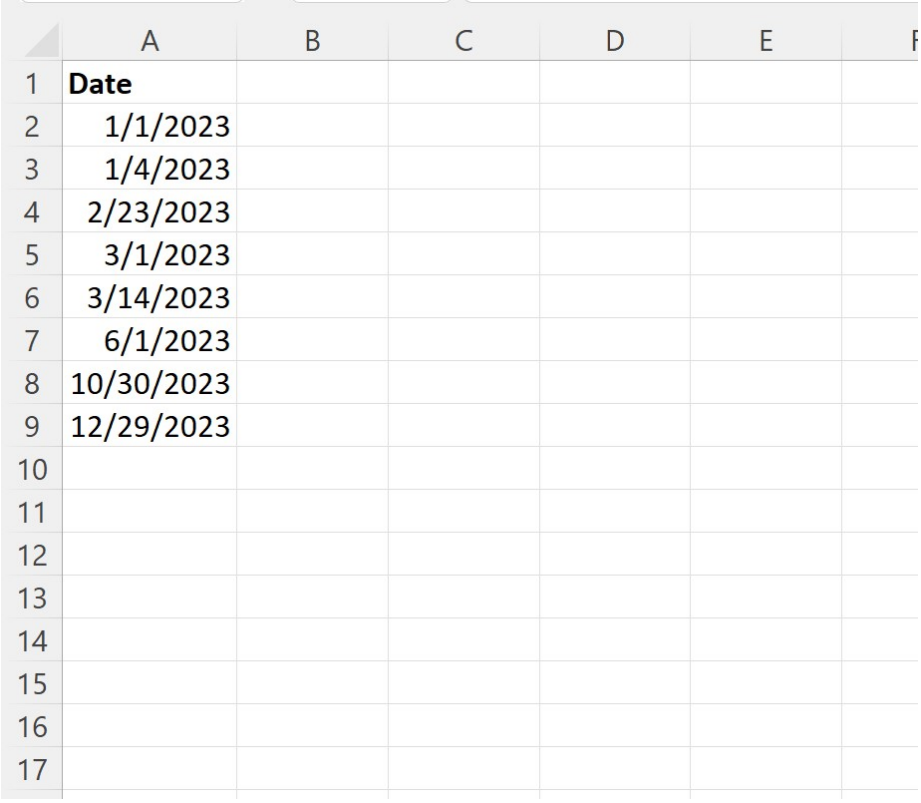
For i = 2 To 9
Range("B" & i) = WorksheetFunction.Weekday(Range("A" & i))
Next i

End Sub

Within this code block, the loop counter `i` is first declared as an Integer. The loop initiates its process at row 2 and continues execution until it reaches row 9. During each iteration, the code retrieves the date value from column A, applies the **Weekday** function (which utilizes the default `vbSunday` indexing since no second argument is provided), and finally assigns the resulting numerical index to the corresponding cell in column B, effectively linking the date to its day index.

Visual Example: Mapping Dates to Day Indices

Let us use a concrete example to solidify this concept. Suppose we begin with the following dataset, which contains a collection of sample dates located in Column A:



	A	B	C	D	E
1	**Date**				
2	1/1/2023				
3	1/4/2023				
4	2/23/2023				
5	3/1/2023				
6	3/14/2023				
7	6/1/2023				
8	10/30/2023				
9	12/29/2023				
10					
11					
12					
13					
14					
15					
16					
17					

Our goal is to accurately calculate and display the day of the week index for every entry, placing these results immediately into column B. We will execute the previously defined macro, `FindWeekday()`, which, by design, uses the default indexing convention where Sunday corresponds to the number 1.

Upon successful execution of the macro, the procedure processes each date sequentially from A2 through A9. For instance, the macro reads the date in A2, determines its weekday index (1 for Sunday), and writes that index to B2. This automated workflow continues until all dates in the specified range have been processed. The resulting output clearly demonstrates how the numerical indices align perfectly with the actual days of the week:

	A	B	C	D	E
1	**Date**	**Weekday**			
2	1/1/2023	1			
3	1/4/2023	4			
4	2/23/2023	5			
5	3/1/2023	4			
6	3/14/2023	3			
7	6/1/2023	5			
8	10/30/2023	2			
9	12/29/2023	6			
10					
11					
12					
13					
14					
15					
16					
17					

As the output confirms, Column B now holds the correct day of the week index for every date in Column A. We can quickly verify a few key results against the standard indexing system (Sunday=1):

The date **1/1/2023** falls precisely on a Sunday, leading the **Weekday** function to return the index **1**.
The date **1/4/2023** is a Wednesday, which results in the **Weekday** function returning the index **4**.
The date **2/23/2023** is a Thursday, corresponding to the **Weekday** function returning the index **5**.

This systematic, numerical output provides clean, structured data that is immediately ready for subsequent calculations or complex conditional logic within the VBA environment.

Beyond the Index: Alternative Methods and Resources

While the **Weekday** function excels at providing a swift numerical index, developers frequently require the full, descriptive name of the day (e.g., "Monday" or "Friday") displayed as human-readable text. For this specific requirement, relying solely on the numerical output of the **Weekday** function is not the most efficient approach.

Instead, if your application requires the day of the week to be returned as a string name rather than an Integer index, you should utilize the related WeekdayName function. This utility takes the

numerical index produced by **Weekday** (or any raw number from 1 to 7) and converts it directly into the corresponding day name, offering a much easier path to generating user-friendly output.

Mastering the intricacies of date and time functions, especially the core mechanics of the **Weekday** function and the critical importance of the optional `FirstDayOfWeek` argument, is paramount for writing robust VBA code. By specifying the indexing system, you guarantee that your code handles date logic accurately and predictably, regardless of the regional settings configured on the user's machine.

For a full technical reference, including a comprehensive list of all available constants that can be used for the `FirstDayOfWeek` argument, developers should always consult the official Microsoft documentation for the Weekday function.

Additional Resources for VBA Development

The following tutorials provide guidance on how to perform other common and necessary tasks within the VBA programming environment: