Learn How to Extract the First Item from Split Text Using Excel’s TEXTSPLIT Function

Authored by

Mohammed looti

November 10, 2025

Introduction to Efficient Text Splitting in Excel

The ability to parse and separate structured data--whether it involves splitting names, breaking down product codes, or isolating address components--is a fundamental skill for advanced users of Excel. Historically, achieving reliable text separation required cumbersome and error-prone formulas, often involving a complex nesting of functions such as **LEFT**, **FIND**, and **LEN**. These legacy methods were difficult to read, challenging to maintain, and often failed when encountering minor variations in data structure. Fortunately, the introduction of Dynamic Array Formulas revolutionized text manipulation within the spreadsheet environment.

At the heart of this modernization is the powerful **TEXTSPLIT function**, which allows users to instantly dissect a text string based on a specified character, known as a delimiter. This function outputs a temporary, dynamic array containing all the resulting split components. While **TEXTSPLIT** is highly effective for generating an entire array of split data, often the practical requirement is simpler: to extract just one specific component, such as the first item in the sequence (e.g., a first name or a prefix). To achieve this targeted extraction with maximum efficiency and minimal formula complexity, we must strategically pair **TEXTSPLIT** with another essential modern array function: **CHOOSECOLS**.

This tutorial provides a detailed walkthrough on how to leverage this dynamic pairing to create a clean, single-cell formula solution capable of isolating the required element without relying on auxiliary helper columns or outdated string manipulation techniques. We will focus specifically on utilizing **TEXTSPLIT** and **CHOOSECOLS** in tandem to reliably split a text string and instantly retrieve the very first item resulting from that separation. This methodology significantly enhances formula readability, improves maintenance ease, and ensures precision when handling diverse data formats in your spreadsheets.

The Core Formula: Combining TEXTSPLIT and CHOOSECOLS

To efficiently isolate the first segment of text following a split, we rely on a concise and elegant formula structure that leverages the internal data processing capabilities of Excel's dynamic array engine. The process begins with the nested **TEXTSPLIT function**, which analyzes the source text and divides it into multiple parts wherever the specified delimiter is located. This initial action generates a horizontal array in memory, where each split item occupies a distinct column.

Once this temporary array is constructed, the outer function, **CHOOSECOLS function**, immediately intercepts this array. The primary purpose of **CHOOSECOLS** is to select specific columns from an array or range provided to it. Since the first item produced by **TEXTSPLIT** always resides in the first column of the resulting array, we instruct **CHOOSECOLS** to retrieve only that column. This elegant architectural choice allows us to isolate the desired component without any

intermediate steps. The necessary inputs for this combined formula are minimal: the cell containing the source text, the character used as the delimiter, and the column index number (which is 1 for the first item).

For instance, to retrieve the first item from the text string housed in cell **A2**, using a standard space character as the separation point, the following highly efficient syntax is employed. This formula encapsulates the entire splitting operation and the precise selection process within a single formula cell, ensuring maximum clarity and performance:

=CHOOSECOLS(TEXTSPLIT(A2, " "), 1)

In practice, if cell **A2** contains the string **Andy Bernard**, the inner function, `TEXTSPLIT(A2, " ")`, first generates the internal array `{"Andy", "Bernard"}`. The outer function, `CHOOSECOLS(..., 1)`, then acts upon this array by selecting the data corresponding to the index 1, returning only the string **Andy** as the ultimate output. This powerful, nested construction makes the extraction of specific components from any structured text source both instantaneous and supremely reliable.

Practical Application: Extracting First Names

A ubiquitous data cleaning task involves separating full names listed in a single column into their constituent parts, most commonly requiring the isolation of the first name for reporting or analysis purposes. The synergistic combination of **TEXTSPLIT** and **CHOOSECOLS** provides an ideal solution for automating this scenario, replacing traditional complex multi-function formulas with a single, intuitive expression.

Imagine you have a dataset starting in column A, listing full names, and your objective is to populate column B exclusively with the first name of each individual. The structure of the data is shown below:

	A	B	C	D	E
1	**Name**				
2	Andy Bernard				
3	Chad Douglas				
4	Eric Ferdinand				
5	Josh Mann				
6	Arnold Smith				
7	Jake Johnson				
8	Tyson Reed				
9	Brett Handzel				
10	Mike Scott				
11					
12					
13					
14					

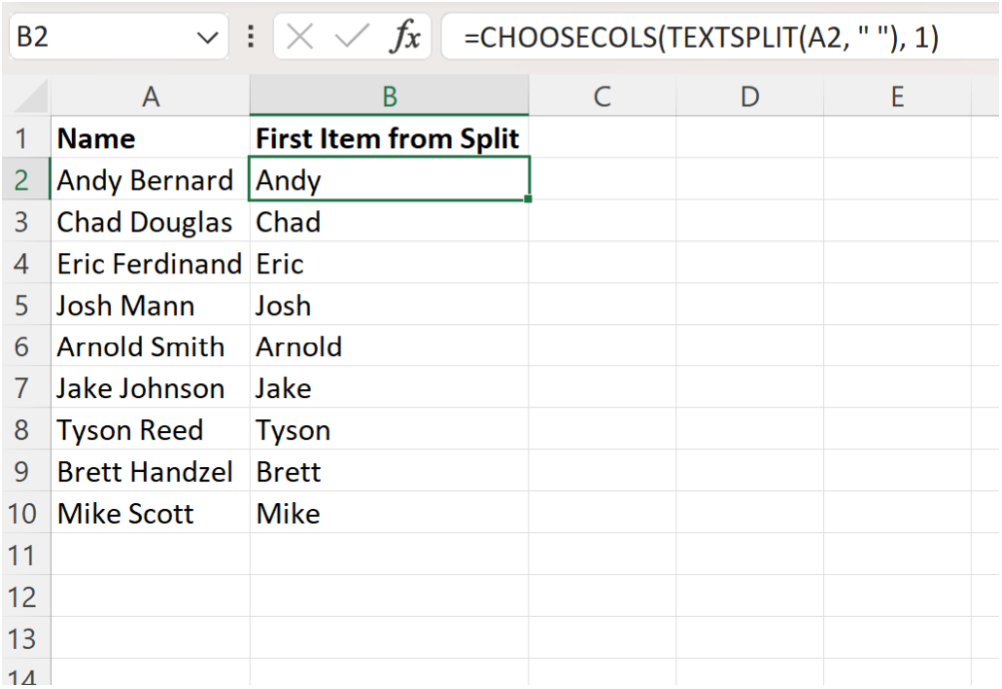
Our goal is for the formula to systematically analyze each full name string, split it wherever a space occurs, and then return only the first resulting item. Given that names are conventionally separated by a single space, the space character (`" "`) is designated as the perfect delimiter. We initiate this process by inputting the necessary formula into cell **B2**, ensuring it references the name contained within cell **A2**.

The exact formula to perform the split and extraction into cell **B2** is as follows:

=CHOOSECOLS(TEXTSPLIT(A2, " "), 1)

Once this formula is entered into **B2**, we can seamlessly apply this logic to the remainder of the dataset by utilizing Excel's auto-fill feature, dragging the formula down the column. Because this approach is rooted in **Dynamic Array Formulas**, the entire calculation is executed with remarkable speed and efficiency, providing the desired output across the entire column instantly, regardless of dataset size.

The final result clearly illustrates the power of this method: the formula successfully splits the names in column A based on the space delimiter and subsequently returns the first item from the split, accurately retrieving the first name for every person listed in the original source data:



Advanced Extraction: Retrieving Subsequent Items

A major functional benefit of adopting the **TEXTSPLIT** and **CHOOSECOLS** pairing is the inherent flexibility it offers when you need to retrieve items other than the first. Whether your requirement is the second, third, or any subsequent item in the split string, the core logic of the formula remains perfectly sound. Only one single numerical argument needs to be modified to change the target extraction.

For instance, if your objective shifted from extracting the first name to isolating the last name--which typically corresponds to the second item in a standard two-part name structure--you would simply adjust the column index argument within the **CHOOSECOLS** function. By changing this parameter from **1** to **2**, you explicitly instruct the function to select the second column from the dynamic array generated by **TEXTSPLIT**.

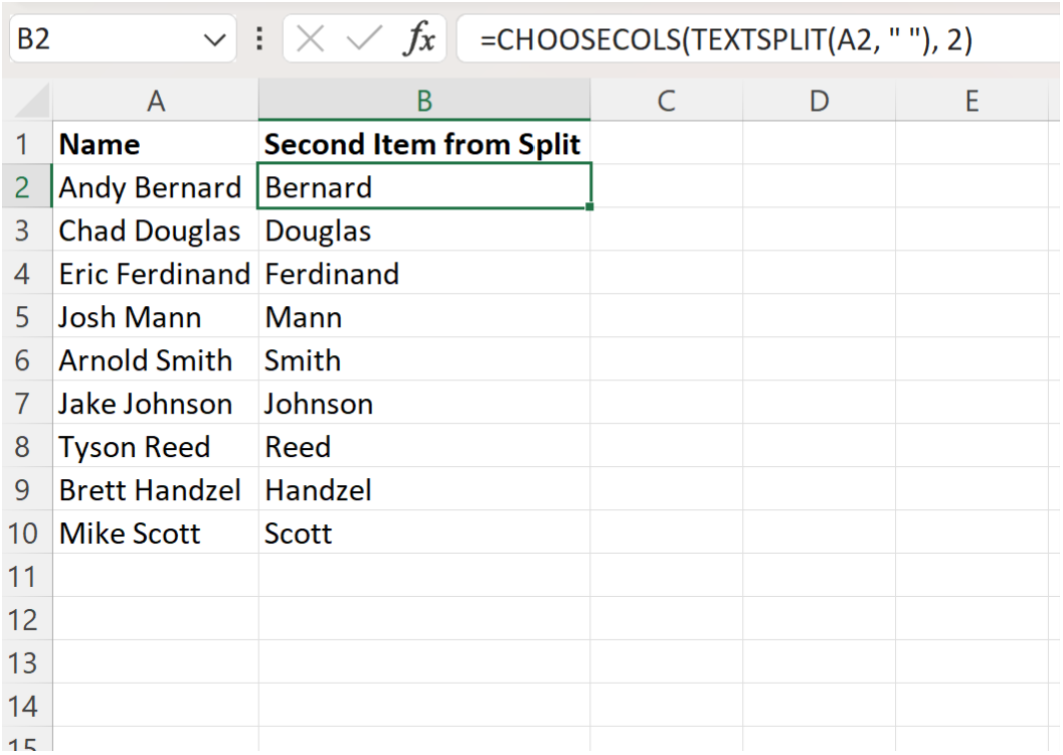
To return the second item from the text located in cell **A2**, the revised and equally concise formula syntax is as follows:

=CHOOSECOLS(TEXTSPLIT(A2, " "), 2)

Applying this minimally modified formula to the same full-name dataset in column A immediately yields the last names in column B. This scalability is invaluable when working with diverse data structures, such as extracting a middle initial (column index 2 if the data contains three parts) or isolating specific numerical components from structured product codes separated by various

punctuation marks like hyphens or commas.

The following visual demonstrates the outcome of implementing this slight but powerful modification, where the formula now successfully returns the second element resulting from the text split:



	A	B	C	D	E
1	**Name**	**Second Item from Split**			
2	Andy Bernard	Bernard			
3	Chad Douglas	Douglas			
4	Eric Ferdinand	Ferdinand			
5	Josh Mann	Mann			
6	Arnold Smith	Smith			
7	Jake Johnson	Johnson			
8	Tyson Reed	Reed			
9	Brett Handzel	Handzel			
10	Mike Scott	Scott			
11					
12					
13					
14					

As clearly illustrated, the formula expertly isolates the second part of each text string--the surname--after segmenting the original content using the space character. This effortless capability to index the split results is precisely what establishes the combination of **TEXTSPLIT** and **CHOOSECOLS** as a foundational tool for modern Excel text handling and data preparation.

Understanding the Dynamic Array Mechanics

A deeper comprehension of the internal mechanics of this process is beneficial for fully appreciating its efficiency and reliability. The operation relies heavily on how the functions interact within the context of Excel's Dynamic Array Formulas capability. Let us revisit the fundamental structure used for retrieving the first item:

=CHOOSECOLS(TEXTSPLIT(A2, " "), 1)

This formula executes a precise sequence of calculation steps internally, making the complex process appear seamless to the end-user. The first step involves the execution of the inner

function, `TEXTSPLIT(A2, " ")`. This function processes the text string in cell **A2** and, based on the space delimiter, instantaneously generates a temporary, multi-column array entirely within Excel's memory. For instance, if **A2** contains the string **Andy Bernard**, the resulting virtual array is `{"Andy", "Bernard"}`. It is crucial to note that this array is ephemeral and is not displayed on the worksheet, preventing unnecessary data spill unless **TEXTSPLIT** were used by itself.

In the subsequent step, the outer function, **CHOOSECOLS**, receives this dynamically generated array as its primary input. The final argument provided to **CHOOSECOLS**--the column index, specified as `1` in this example--determines which specific column from the received array should be returned to the cell. By specifying the index `1`, the function selects and outputs only the data contained in the first column of the split array.

The final result is that **CHOOSECOLS** operates essentially as an advanced array filter, efficiently isolating the required component (e.g., **Andy**) from the list of items produced by the split operation. This integrated, nested structure ensures that the entire text parsing and extraction is performed within a single cell, significantly simplifying tasks that previously demanded multiple steps or complex logic trees.

Conclusion and Further Resources

The combination of the **TEXTSPLIT function** and the **CHOOSECOLS function** offers a superior, modern, and highly readable solution for tackling complex text parsing challenges in Excel. This dual function approach provides unparalleled flexibility and clarity, whether your goal is to extract the first element, the last element, or any specific component located in the middle of a delimited text string. By mastering how these powerful **Dynamic Array Formulas** interact, users can dramatically enhance the speed and robustness of their data management and preparation workflows.

Moving beyond legacy string manipulation techniques and embracing the speed and simplicity offered by Excel's latest features is a crucial step for data professionals. We strongly recommend continuing to explore other functions within the dynamic array suite--such as FILTER, SORT, and UNIQUE--to further streamline and modernize your overall spreadsheet operations and formula design.

For those interested in expanding their knowledge of advanced data manipulation in Excel, the following tutorials offer deeper dives into related topics:

Tutorial on extracting the Nth word using advanced string methods.

Guide to using the **FILTER** function for dynamic data subsets.

Deep dive into multiple delimiter handling with **TEXTSPLIT**.