- Task: Search books by genre
- Task: Multiple query parameters
- Path Variable
- Task: Get a book by id
- Quiz #4: Query vs. path parameters

Sending data with POST

- Task: Add a book
- Task: Add an authorization header
- Task: Use Postman Auth instead!
- Quiz #5: Sending Data With Postman

Introduction to variables and scripting

- Variables in Postman (Continued)
- Setting variables programmatically
- Task: Your first script
- Task: Grab the new book id
- Halfway Test
- Quiz #6: Intro to Variables

PATCH and DELETE

- Task: Checkout your book
- Task: Delete your book

Generating code

- Postman's codegen feature

Wrapping Up

- Skillcheck (required)
- Recap

Variables in Postman (Continued)

Previously in the "Request Parameters" section of this course, we saw how using a variable saved us time and helped reduce redundant copy-paste of the request URL using the double curly brace syntax like this: `{{variableName}}`.

Remember, Postman allows you to save values as variables so that you can:

1. Reuse values to keep your work DRY (Don't Repeat Yourself)

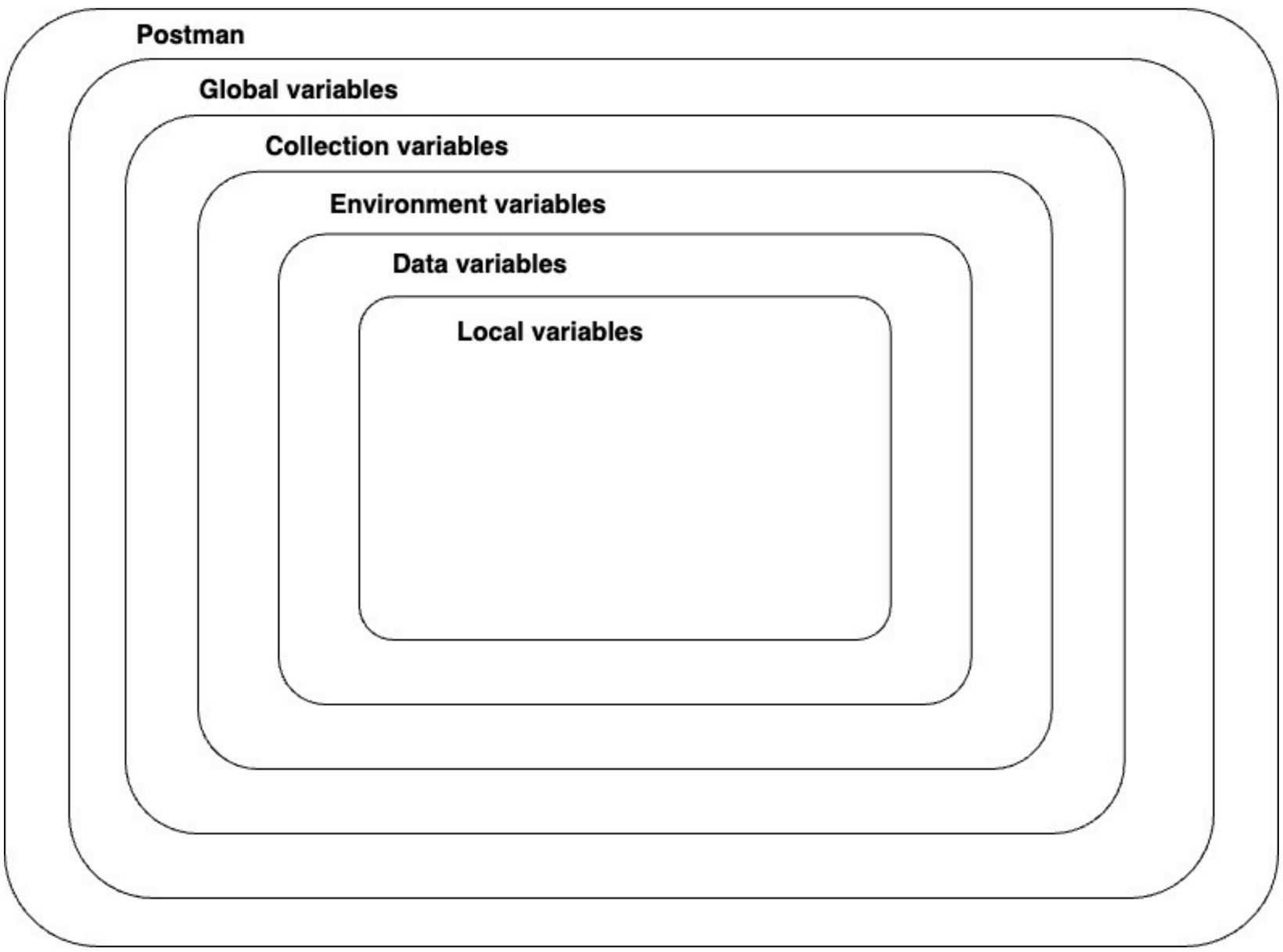
2. Hide sensitive values like API keys from being shared publicly

In this section, we will learn more about variables and introduce better practices that enable us to make dynamic requests.

Variable scopes

You can set variables that live at various scopes. Postman will resolve to the value at the nearest and narrowest scope.

From broadest to narrowest, these scopes are **global**, **collection**, **environment**, **data**, and **local**.



Previous Next

> narrowest scope will be used. For example, if there is a global variable named `username` and a local variable named `username`, the local value will be used when the request runs.

We will work with **collection variables** today, which live at the collection level and can be accessed anywhere inside the collection.

In the next section, you will learn how to set a variable via scripting.