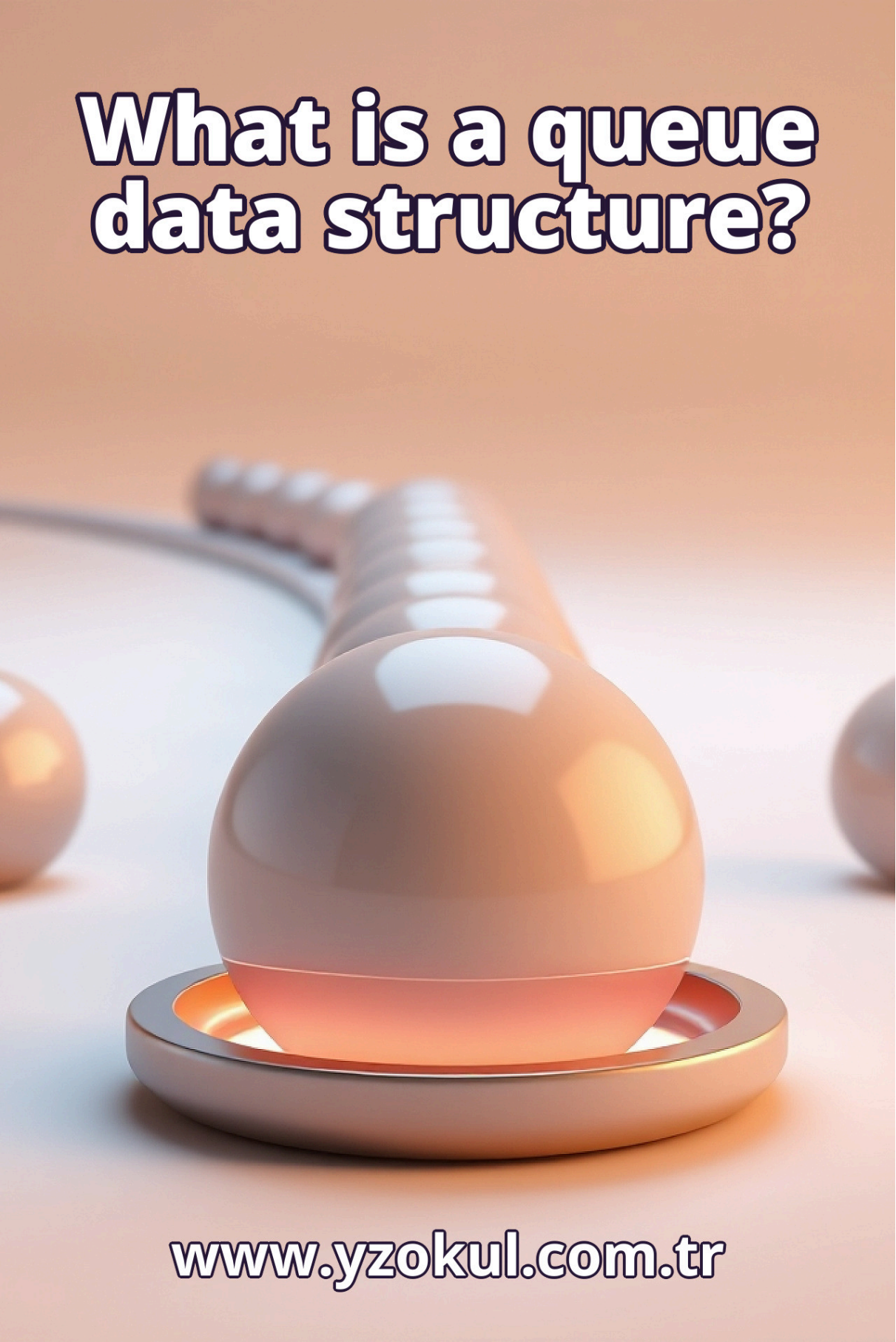
What is a queue data structure?



www.yzokul.com.tr

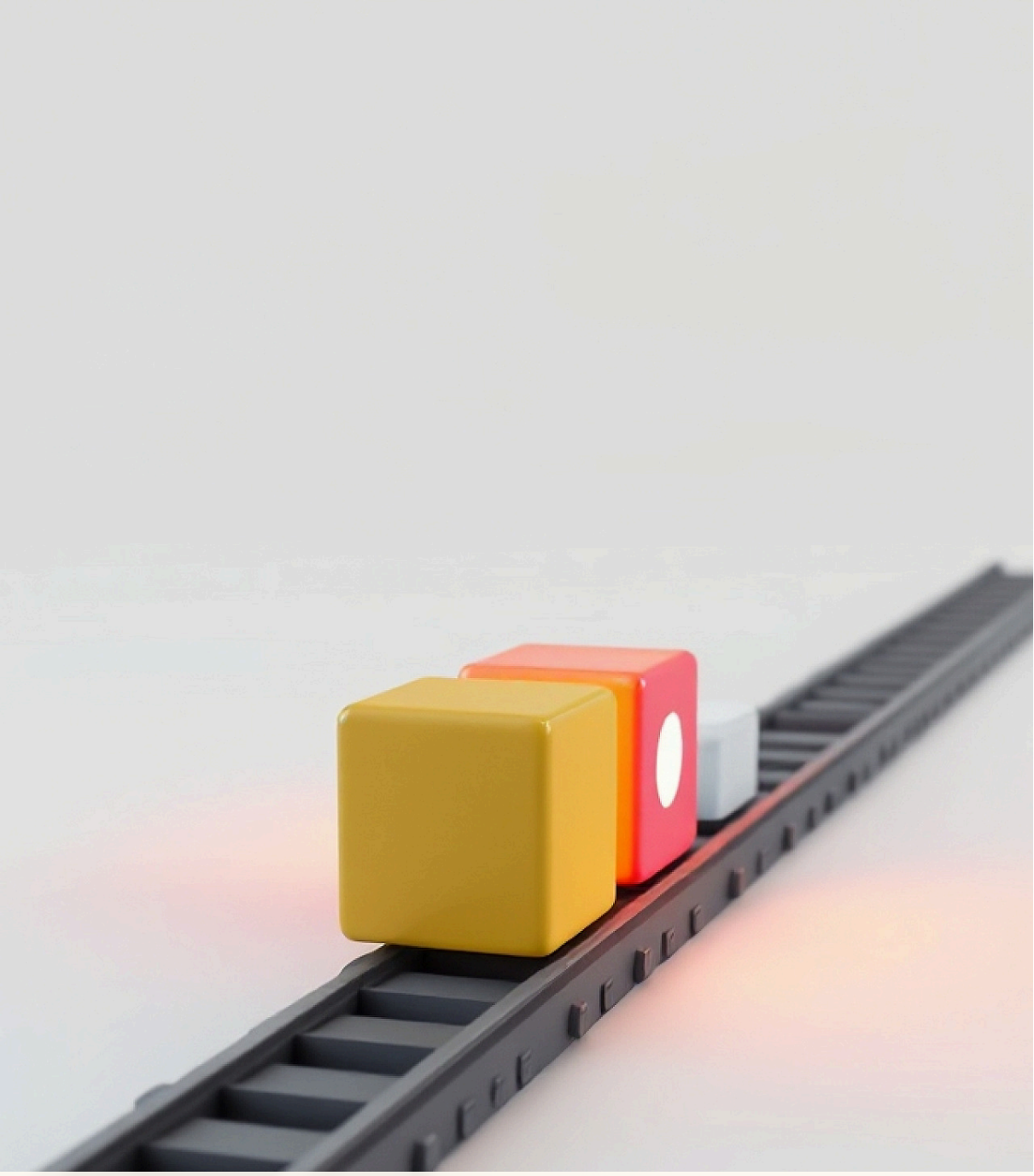
What is a queue data structure?



A queue data structure is a way to hold items in order. The short answer is simple: a queue is a structure where the first item added is the first item removed. You do not need any other idea to begin. This single rule defines the whole structure, and everything else follows from it. Keep that sentence in mind as you read on.



First in, first out is the one rule at the heart of a queue. The item that arrives first is the item that leaves first. No item can leave before the ones that arrived earlier. This rule is often shortened to FIFO, which stands for first in, first out. Every basic queue follows this rule without exception.



New items always join at the back of the queue. Adding an item never changes the order of the items already waiting. The newcomer goes behind everyone else, no matter how important it seems. For example, a new document sent to a printer joins the end of the waiting list. The items ahead keep their places.



Items always leave from the front of the queue. The item removed is the one that has been waiting the longest. Nothing leaves from the middle or the back. For example, the first document sent to a printer is the first one printed. Removing from the front keeps the arrival order intact at all times.



Picture people standing in line at a counter. The person at the front of the line is served first, and everyone else waits behind. A queue works the same way: the item at the front is handled first. This everyday picture makes the first in, first out rule easy to remember. Lines and queues share the same simple logic.



Picture a new person joining the end of the line. That person stands behind everyone already waiting and does not jump ahead. A new item joins the back of a queue in exactly the same way. The people already in line keep their positions. Joining at the back keeps the order fair and predictable for everyone.



Picture the person at the front of the line being served and then leaving. The next person moves up to the front, and the line shortens by one. A queue removes its front item the same way, and the next item becomes the new front. The order of arrival is kept through every removal. This is how a queue serves items one by one.



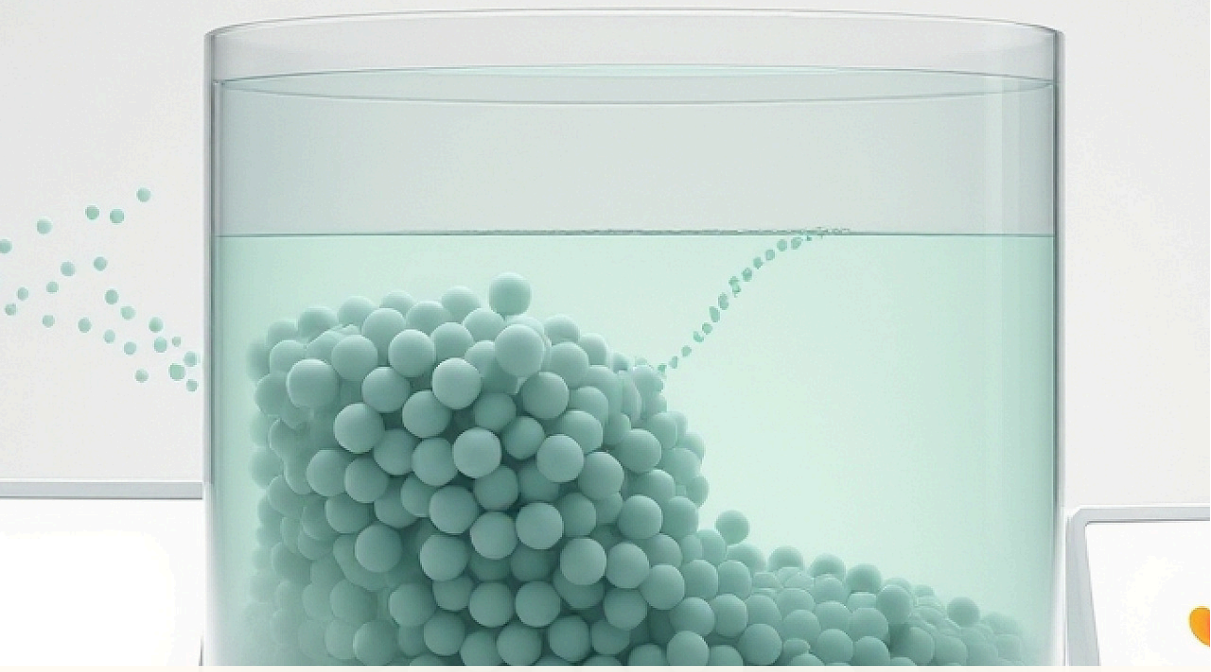
Documents sent to a printer are a practical example of a queue. When several documents are sent, the printer holds them in a queue and prints them in the order they were sent. The first document sent is the first one printed. This stops documents from printing in a jumbled or random order.



A queue makes it easier to organize jobs by the order in which they arrive. Each job waits its turn instead of being handled at random. For example, three documents sent one after another print in that same order. Jobs are handled fairly, and no job is forgotten while it waits. Arrival order becomes the processing order.



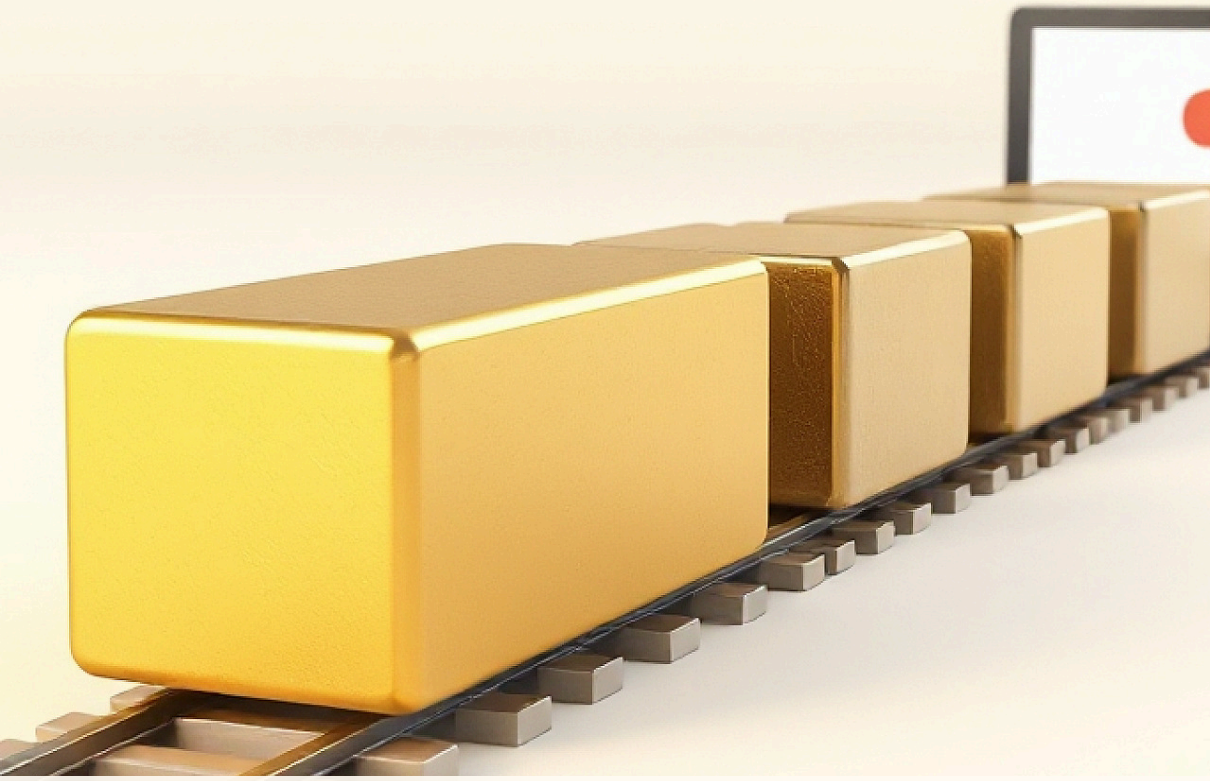
A queue helps when jobs are produced faster than they can be processed. For example, a computer can send documents to a printer much faster than the printer can print them. The extra documents wait in the queue instead of being lost. The waiting jobs stay in order until the printer is ready for each one.



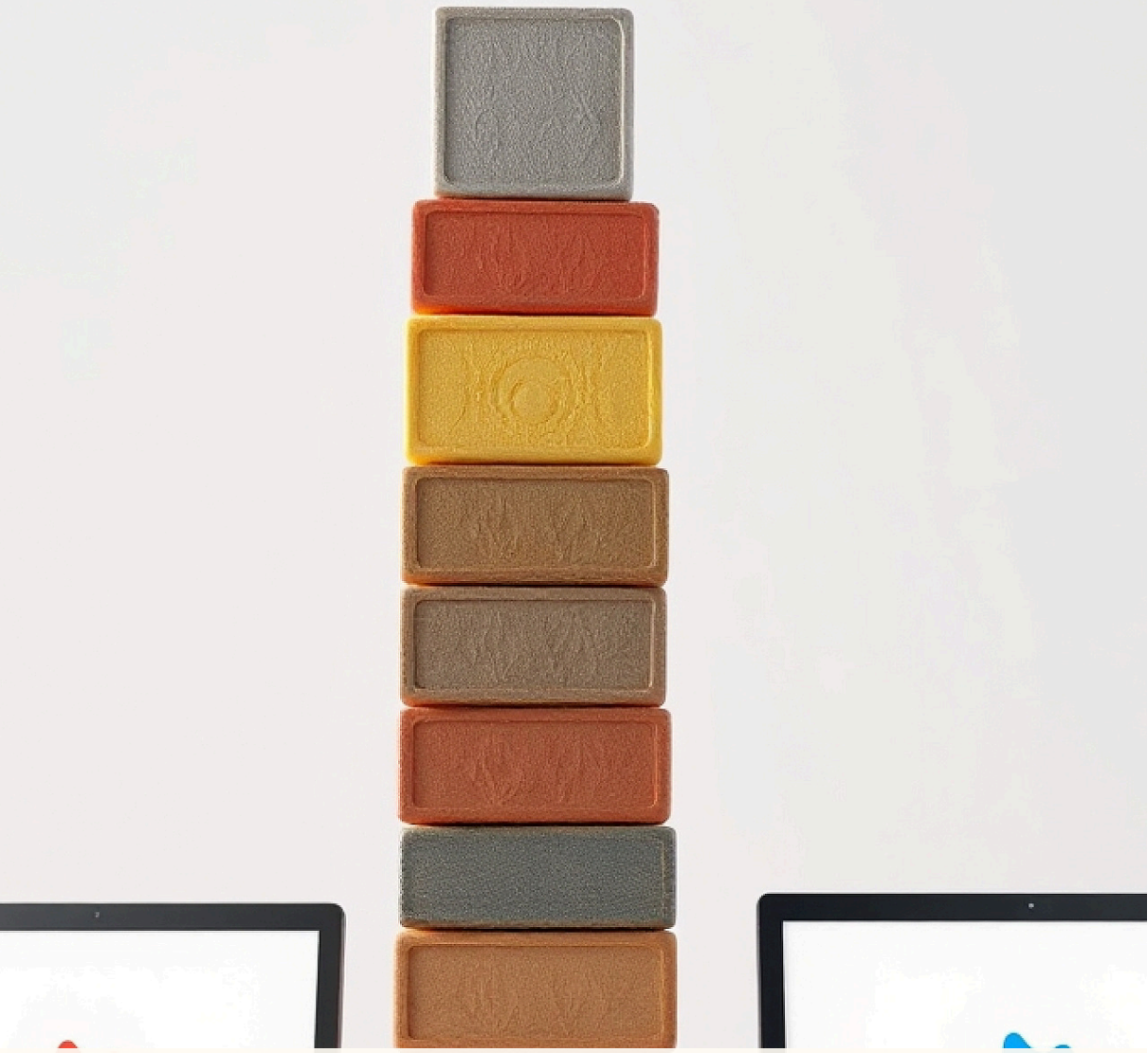
The queue acts as a holding place between a fast producer and a slower processor. The producer adds items at the back, and the processor removes them from the front at its own speed. For example, the computer produces documents quickly while the printer processes them slowly. Nothing is lost, and nothing is handled out of order. The queue absorbs the difference in speed.



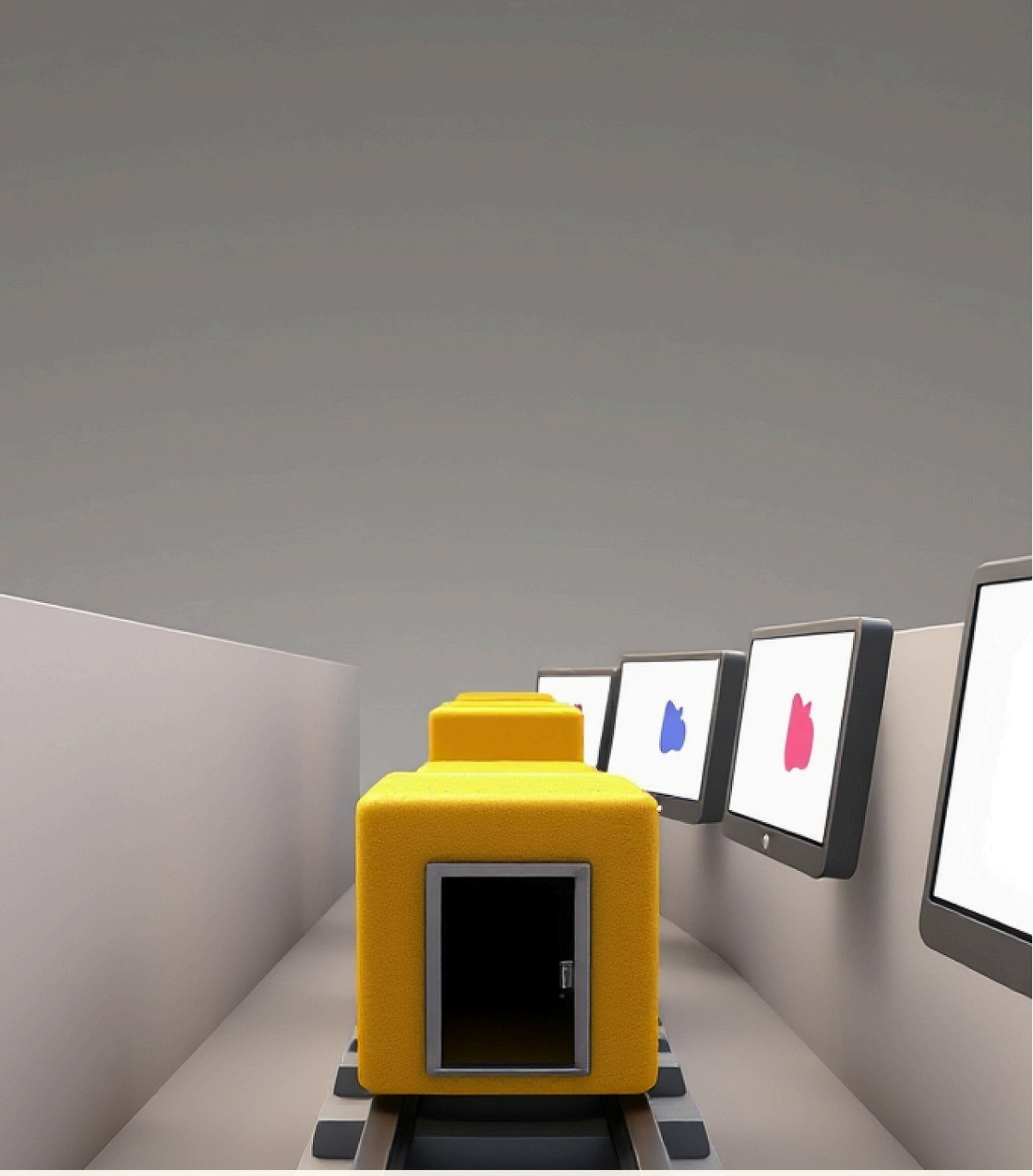
Real systems can apply additional rules, such as priority. For example, a system might let one urgent document print before older ones. A basic queue still follows first in, first out, but real use may add more. These extra rules are common in practice and do not change what a queue is at its core.



Priority rules are an extra layer, not a replacement for the basic queue idea. The simple first in, first out queue remains the starting point for understanding all queue behavior. For example, a priority document still waits in a queue; it is simply chosen earlier by an added rule. Learn the basic queue first, then add such rules.



A common mistake is thinking the newest item leaves first. That describes a different structure, called last in, first out. In a queue, the oldest item leaves first, not the newest. For example, the first document sent to a printer prints first, not the last one sent. Check yourself: the oldest item always leaves first.



Another common mistake is thinking items can leave in any order. In a basic queue, items leave only from the front, in arrival order. No item can skip ahead of the ones already waiting. For example, a document sent second cannot print before the document sent first. The line moves only from front to back.



A queue removes the first item added first, like a line of people or documents waiting for a printer. It keeps jobs in arrival order and handles differences between production and processing speeds. Real systems may add priority rules on top, but the basic queue stays first in, first out. Remember the line, and you remember the queue.