# What is a session?



www.yzokul.com.tr

# What is a session?



You have likely seen the word session on a website and wondered what it means. A session ties a user's actions to an application in one context. In plain terms, it is how the app knows that several clicks belong to the same person during the same visit. The short answer is that a session groups your actions together so the app can treat them as one continuous use.



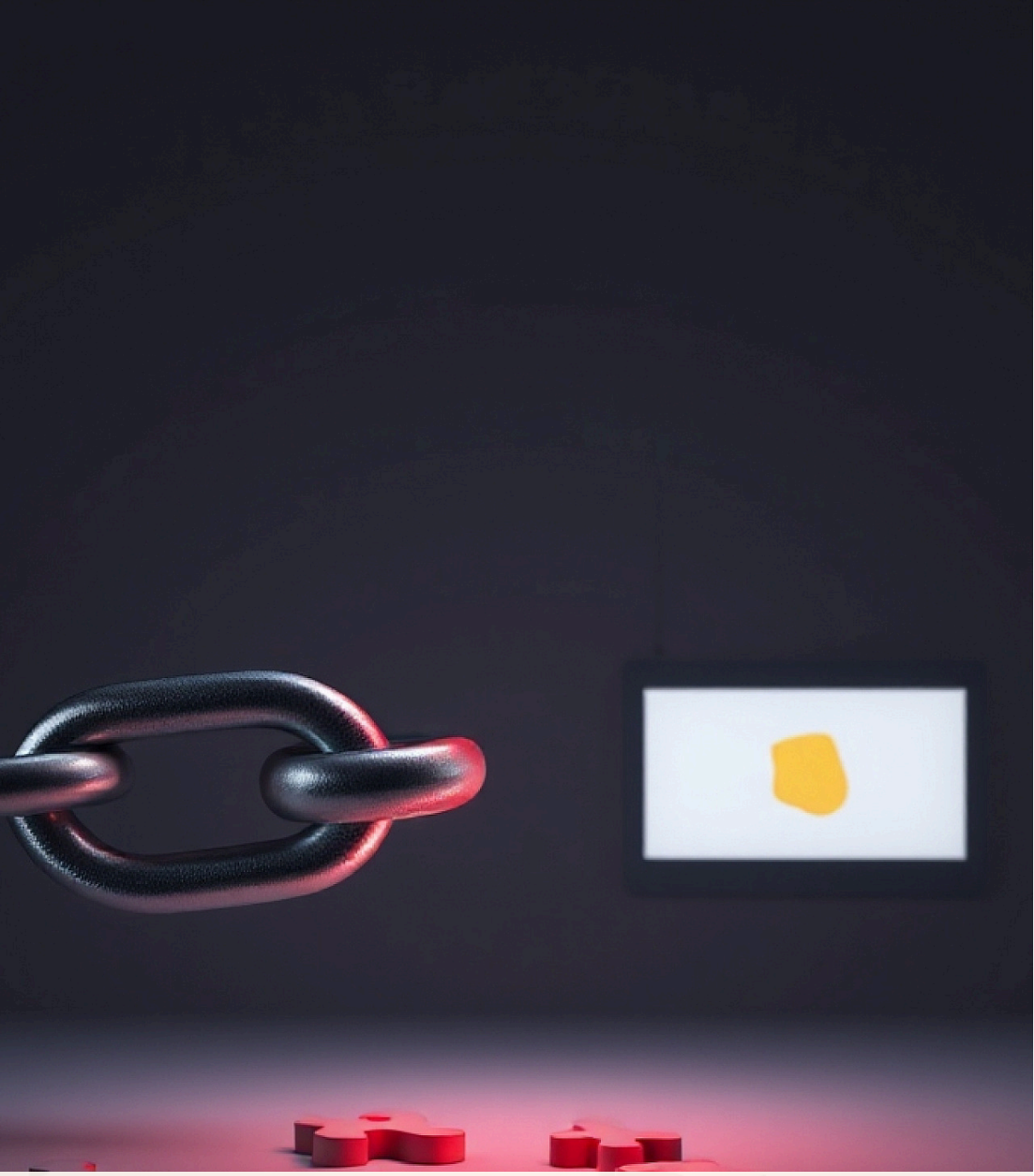
Picture a user clicking through several pages of an application, such as viewing a profile, opening a settings page, and saving a change. Each click is a separate action, but they all happen in one visit. A session groups those actions as belonging to the same person in the same context. Without that grouping, the app would see each click as a stranger.



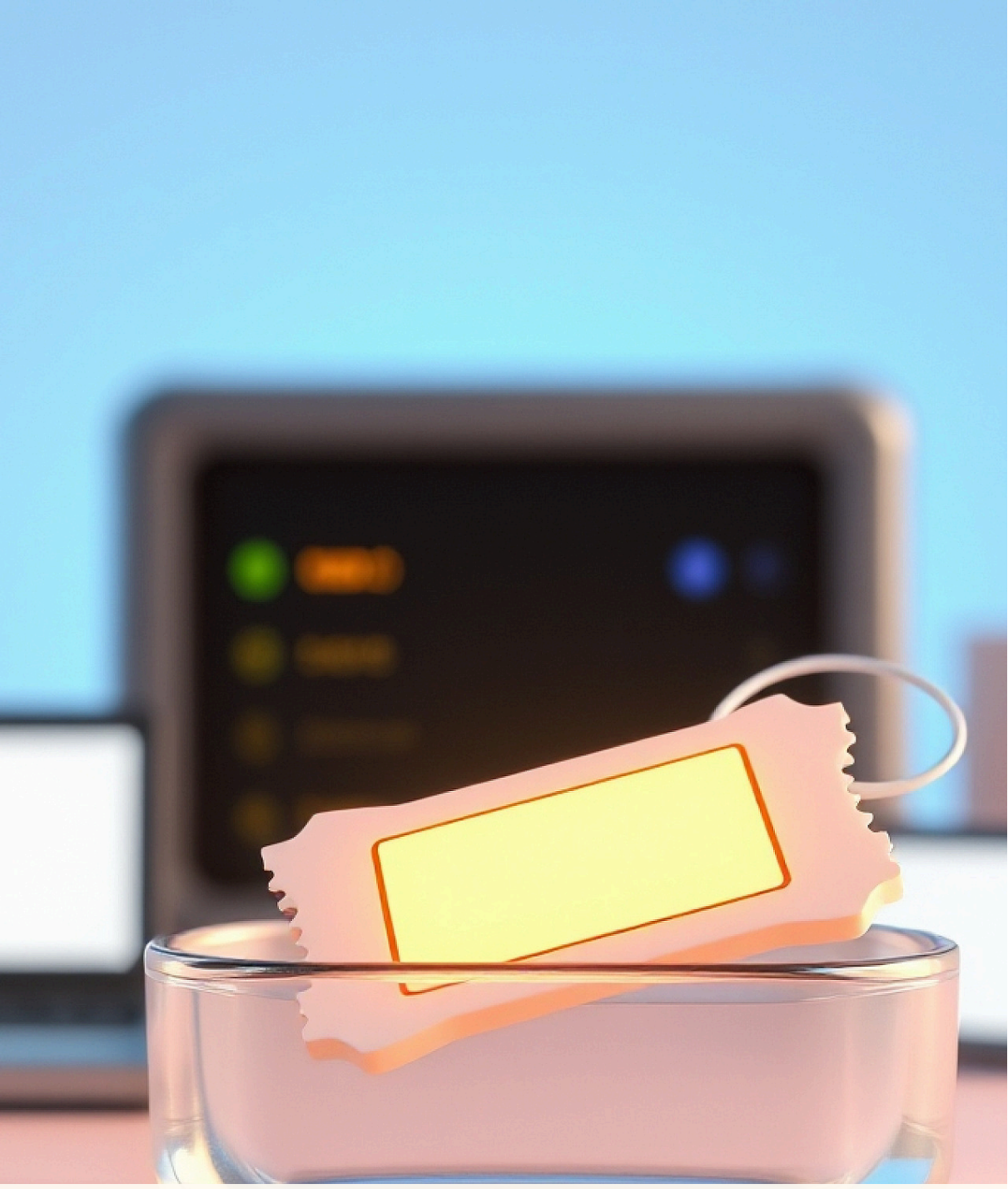
A session often begins when a user types a password on a login page and submits it. The application checks the password and, if it is correct, starts a session for that user. From that moment, the app remembers that this is the same user. Logging in is the moment the association starts, not the whole session itself.



After logging in, the user moves to other pages without retyping the password. The session is what makes this possible. The app links each new request to the same user because the session is still active. You experience this as staying logged in while you browse, and it saves you from repeated typing.



Imagine a website that forgets you after every click. You would log in, open a page, and be asked for your password again. Then you would open another page and be asked again. This is annoying and slow. Sessions solve this problem by keeping the link between your actions and your account alive across many pages.



To keep that link, the browser can hold a small identifier for the user. Think of it like a ticket number the app can look up later. The identifier is usually a short string of characters. It does not contain your password or your personal details. It is just a reference the app can use on each request.



The details linked to that identifier stay on the server. The server is the computer that runs the application and stores its data. The browser only holds the identifier, not the whole story. This design keeps sensitive information away from the browser. When the app needs to know who you are, it looks up the identifier on the server.



On each request, the server takes the identifier from the browser. It finds the matching information stored on the server and knows who the user is. For example, the identifier might map to a user account and a start time. The server then treats the request as coming from that user. This matching happens quickly and quietly in the background.



Keeping an identifier in the browser and data on the server is one common design. Other ways to build sessions also exist. For example, some systems store signed information in the browser itself, and some use different tokens. The core idea stays the same: the app needs a way to recognize the same user across actions. The details can vary between applications.



A session has a clear start. It begins at a specific moment, for example when the user logs in successfully. From then on, the actions are linked together under that session. The start is like opening a new notebook: everything written after that belongs to the same notebook until it is closed. The app records the beginning so it can manage the session later.



A session can end on its own after some time. This is called expiring. Expiring happens without the user doing anything. For example, if you leave a site idle for a long period, the session may expire for safety. After it expires, the app no longer links new actions to that session, and you may need to log in again.



A user can also choose to log out. Logging out ends the session right away. The app stops linking new actions to it, and the identifier is no longer useful. For example, on a shared computer, logging out prevents the next person from continuing your session. Logout is the deliberate way to close the association.



A common mix-up is thinking the browser stores all the session details. In the design described here, the browser keeps only the identifier. The user information and session data stay on the server. So if someone looks at the browser storage, they see a reference, not your password or full profile. This separation is an important safety point.



Another common mistake is believing a session lasts forever. Sessions can expire or be ended by logout. They are not permanent. For example, a session might expire after a set period of inactivity. This limits how long an identifier can be misused if it is stolen. Sessions are temporary by design, not endless.



Logging in is only one moment. The session is the whole association of the user's actions in one context. Do not confuse the act of entering a password with the session itself. The login starts the session, but the session continues across many pages and actions. It is the ongoing link, not just the entry step.



To sum up, a session links a user's actions in one context. It often works with an identifier in the browser and data on the server. A session starts at a moment like login and ends by expiring or by logout. Remember that the browser holds only the identifier in this design, and that sessions are temporary, not permanent.