REST API Cheatsheet

ByteByteGo

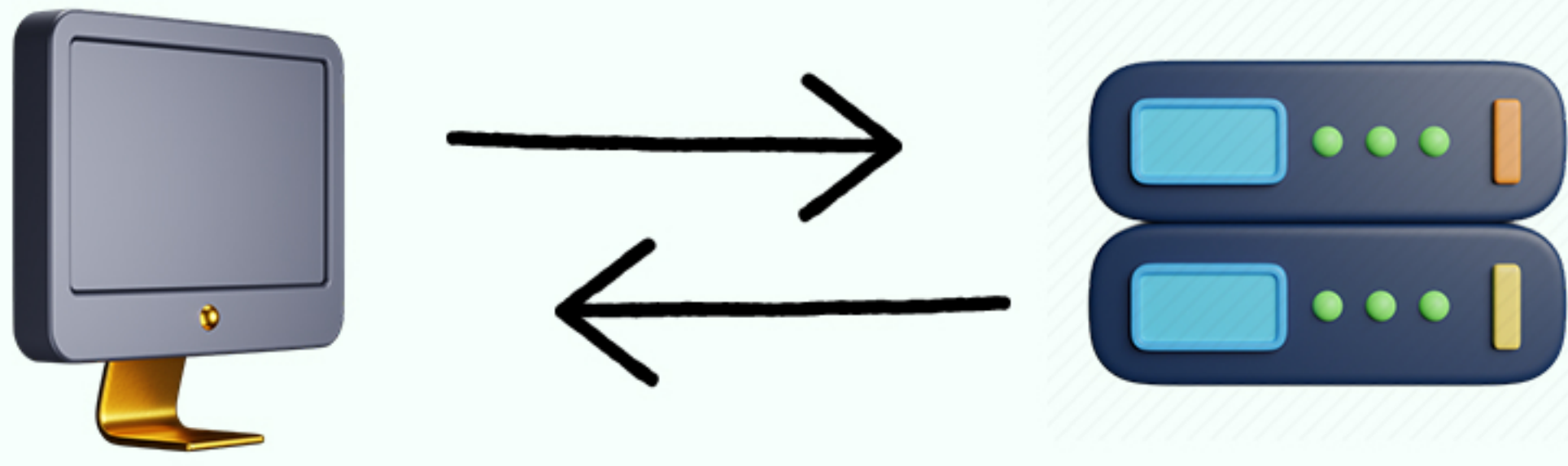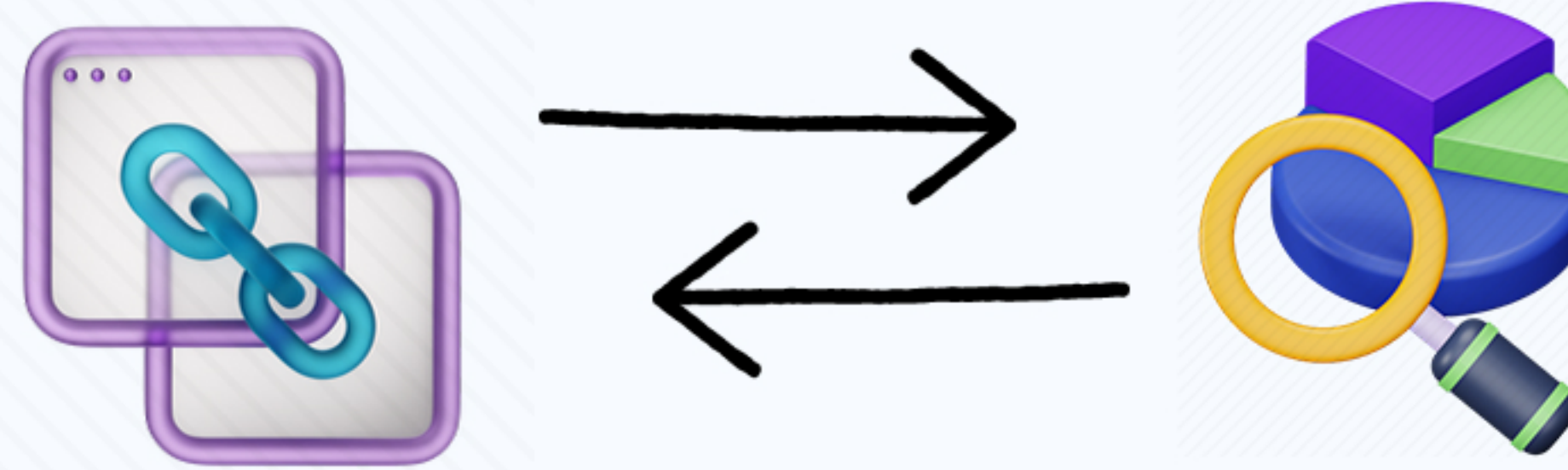## CLIENT-SERVER 1

Emphasizes the separation of user interface concerns(client) from data storage and processing concerns (server) to enhance portability and scalability.

STATELESS 2

Dictates that each request from a client to a server must contain all the necessary information for processing, ensuring no session state is stored on the server between requests.

CACHEABLE 3

Ensures that responses can be explicitly marked as cacheable or non-cacheable,allowing clients to reuse previously fetched responses to enhance efficiency and reduce load.

LAYERED SYSTEM 4

Architecture may have multiple layers, and a client interacts without knowing if it communicates with the end server or an intermediary layer,promoting scalability and security
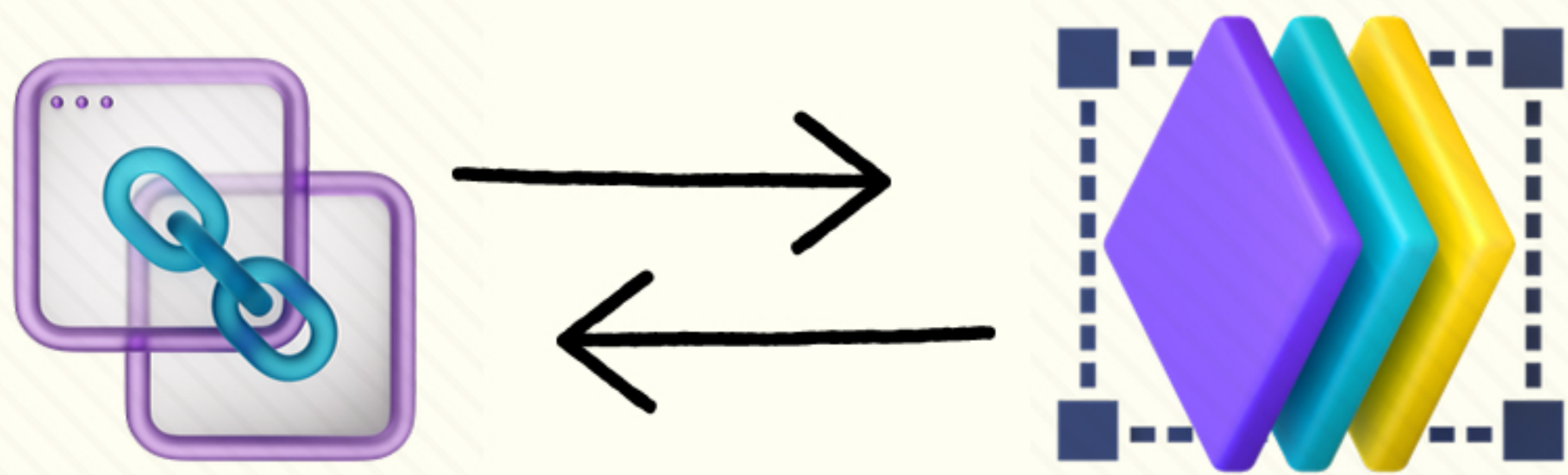
CODE-ON-DEMAND 5

Servers can temporarily extend or customize a client's functionality by transferring executable code,enhancing flexibility and adaptability.

{.js} JavaScript

UNIFORM INTERFACE 6

A consistent and standardized interface simplifies and decouples interactions, making systems independent and easier to evolve.

GET https:// api. example.com /v1 /users
?age=25&gender=male
&page=2&limit=10

PROTOCOLS

Always use HTTPS:// for API calls to ensure security and data privacy

VERSIONING

Version your API to manage changes and maintain backward compatibility.

SUB-DOMAIN

Use clear and consistent domain naming conventions.Subdomains(like **api**) are often used for **API** endpoints.

ENDPOINT

Use **nouns** to represent resources. Paths should be intuitive and follow RESTful principles.

GET https:// api. example.com /v1 /users

?age=25&gender=male & page=2&limit=10

HTTP Methods

GET	for retrieving	POST	for creating
PUT	for updating	PATCH	for patching
DELETE	for deleting		

FILTERING

Divide large data sets into pages for manageable chunk retrieval.

PAGINATION

Specify criteria to retrieve relevant data subset; refine search results