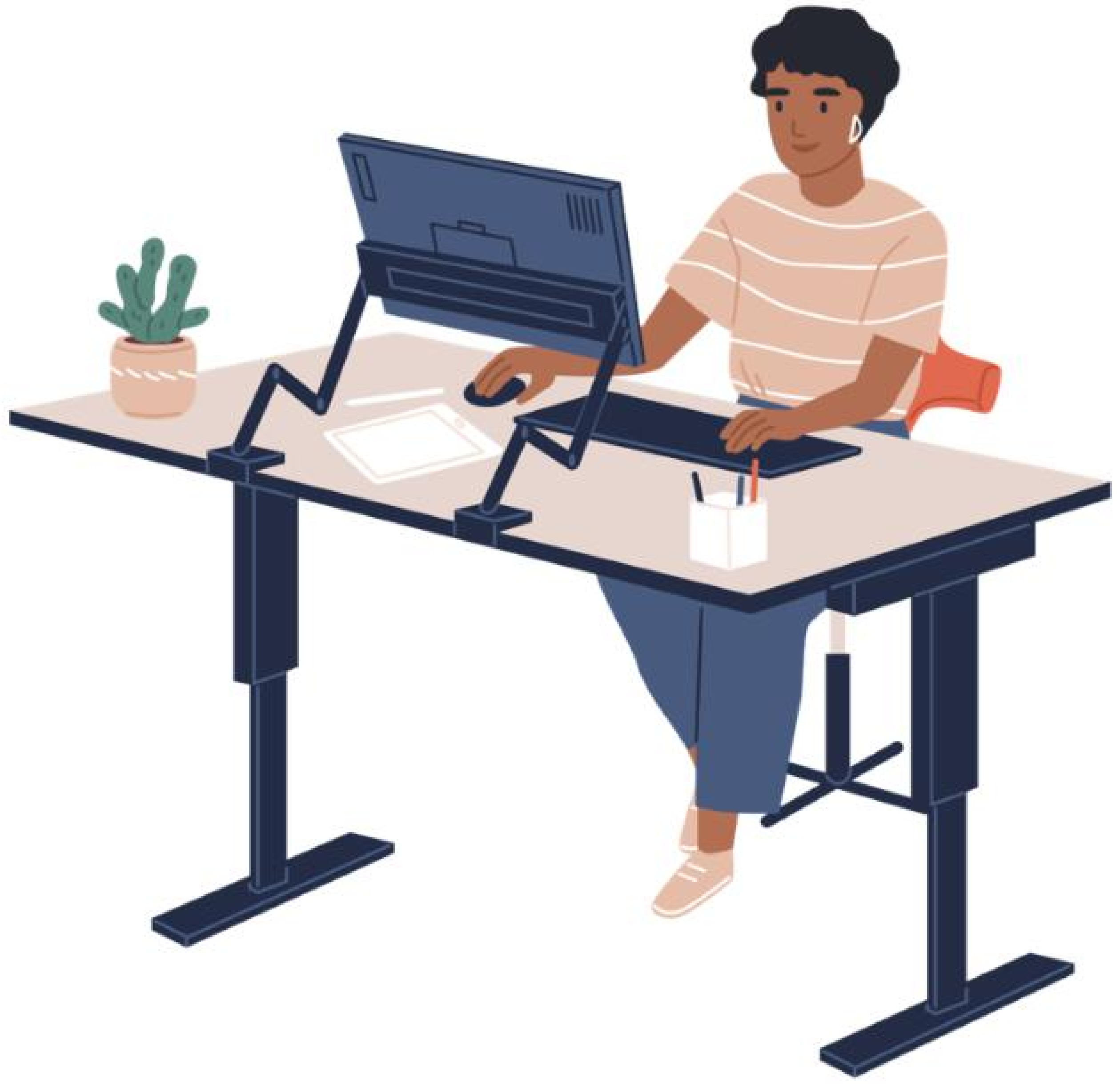
Slide Note

Learning Consolidation

Implement Stateless Communication Using REST APIs and JSON



Menu

00:01:00:00:30 01/ 30

Slide Note

In this sprint, you have learned to:

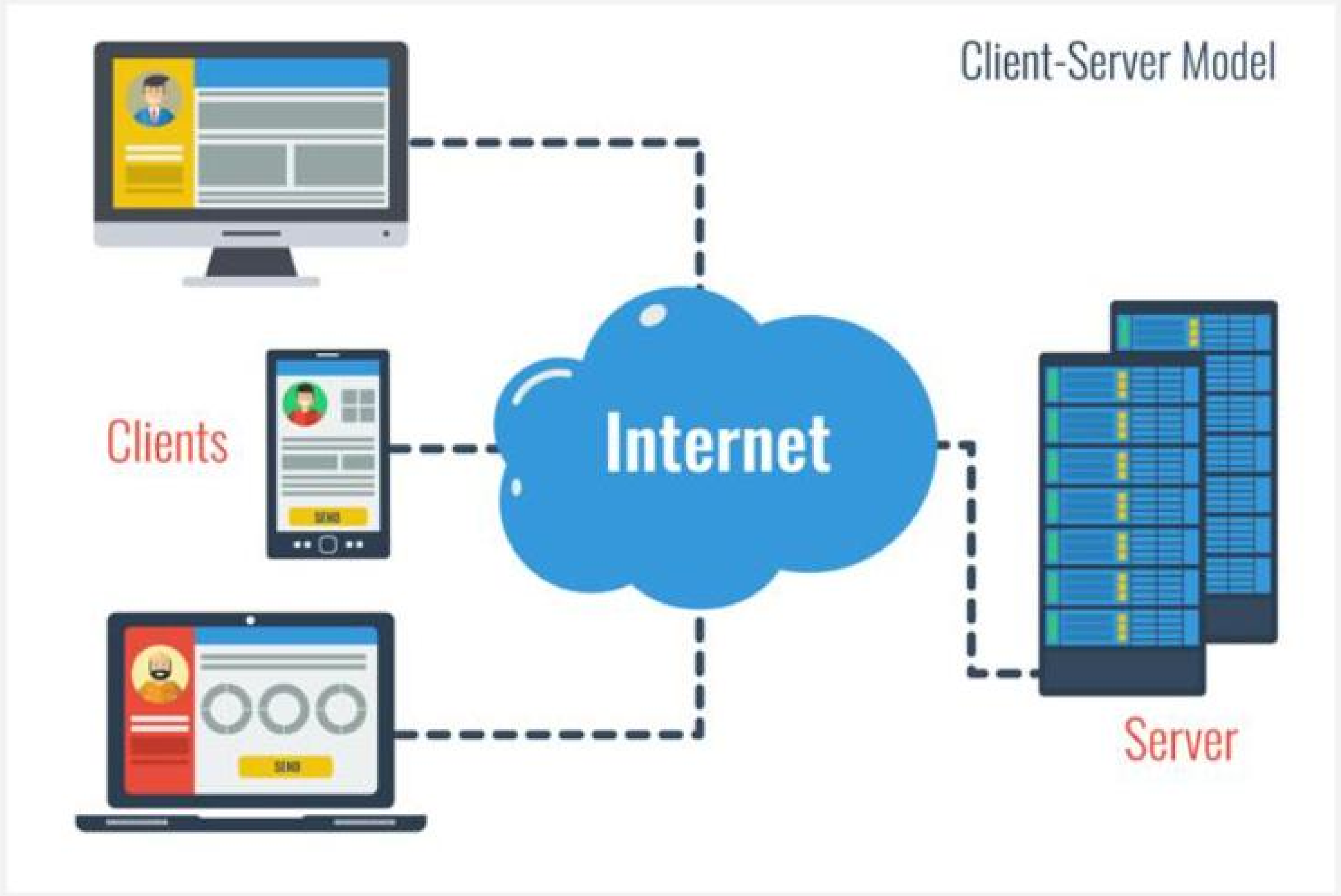
- Explain the role of API in client-server communication
- Explore the rules of client-server communication
- Explain REST
- Make HTTP Request
- List HTTP Methods
- Handle HTTP Response
- Explore HTTP Status Codes
- Create fake REST APIs using json-server
- Test REST APIs using Postman

Menu

00:02 00:00:30 02/ 30

Slide Note

Web Application - Architecture



Frontend **Backend**

- A web application does not run on a single machine.
- The application is distributed across multiple machines.
- The code running at the user end is known as a client application.
- The code running at the remote end is known as a server application.
- Along with HTML and CSS, JavaScript – the scripting language - allows developing client-side application.
- The client-side application is also popularly known as the frontend application.
- The server-side application is popularly known as the backend application.

Menu

Slide Note

Why Distributed Architecture?

- The application can centralize the data and processing on the server as it is distributed in nature.
- Requests can also be received from clients, allowing a large number of users to access the data.

Menu

Slide Note

Can There Be More Server Serving Requests?

- As the popularity of web application grows, number of users increase exponentially.
- At any given time, millions of requests can be sent to the server for data.
- For example, imagine the volume of requests being sent to the Amazon server at 12 midnight on the date it has announced a sale!
- Multiple servers are configured at the backend to handle the large volume of requests efficiently, ensuring a great user experience.

Menu

00:05 00:00:30 05/ 30

Slide Note

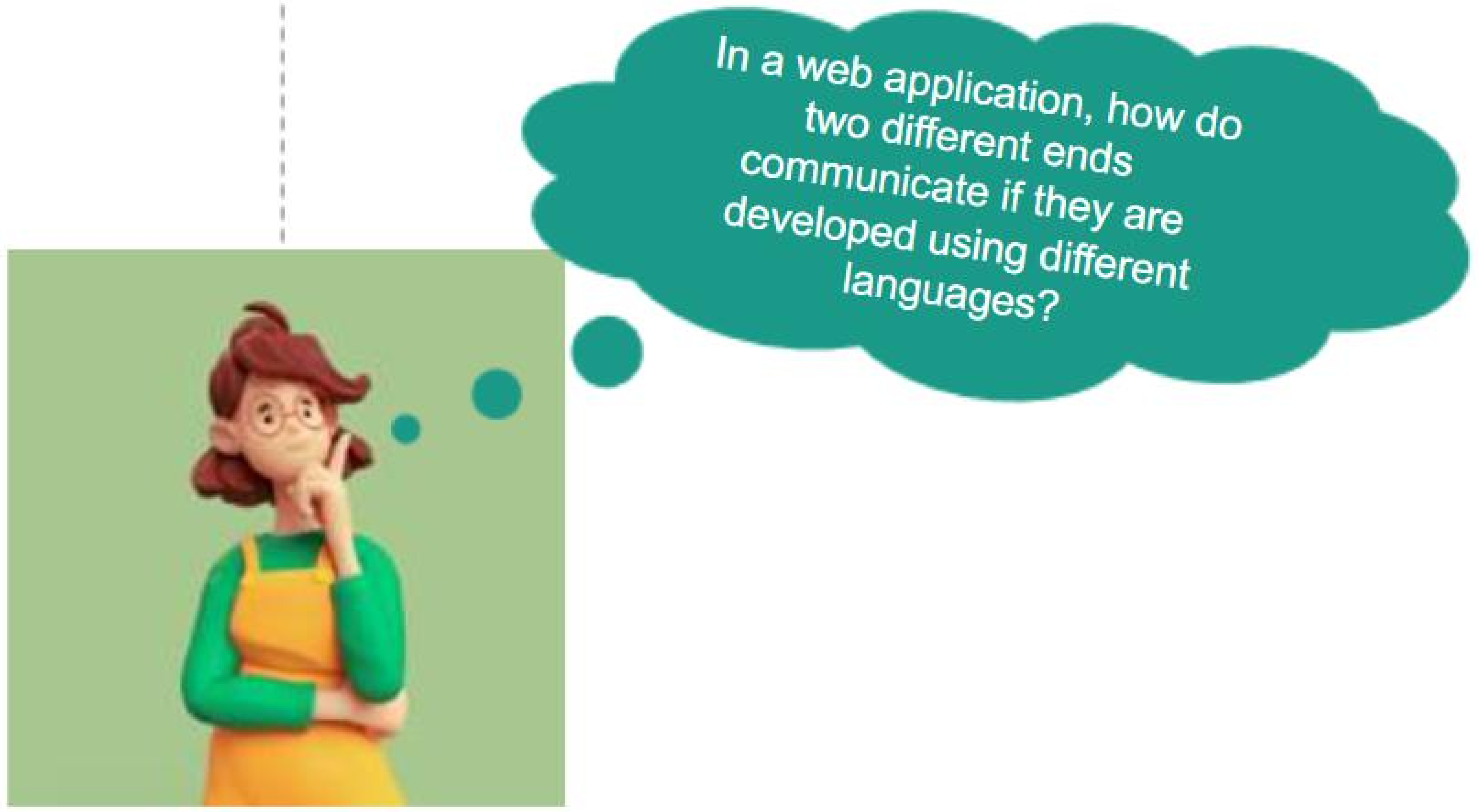
Web Application Development

Frontend Technologies

- HTML → Web Page Structure
- CSS → Style Web Page
- Client-Side JavaScript → Add Interactive
- Browser → Run-time environment

Backend Technologies

- Server-Side JavaScript
- Java
- C# (.NET)


In a web application, how do two different ends communicate if they are developed using different languages?

Menu

Slide Note

Application Programming Interface

- Application Programming Interface, or API, is a program that helps to establish a communication link between client and server applications.
- It is developed at the server end but is programmed in a way that allows the client to make requests.
- The requests from clients are intercepted by API, which makes a call to the server as per the request.
- Once the server has processed the request, the response generated is sent to the client by the API.
- Of course, this requires a set of rules that has to be followed by the client, server and API.
- The term Protocol defines the set of rules that are defined for establishing communication.

Menu

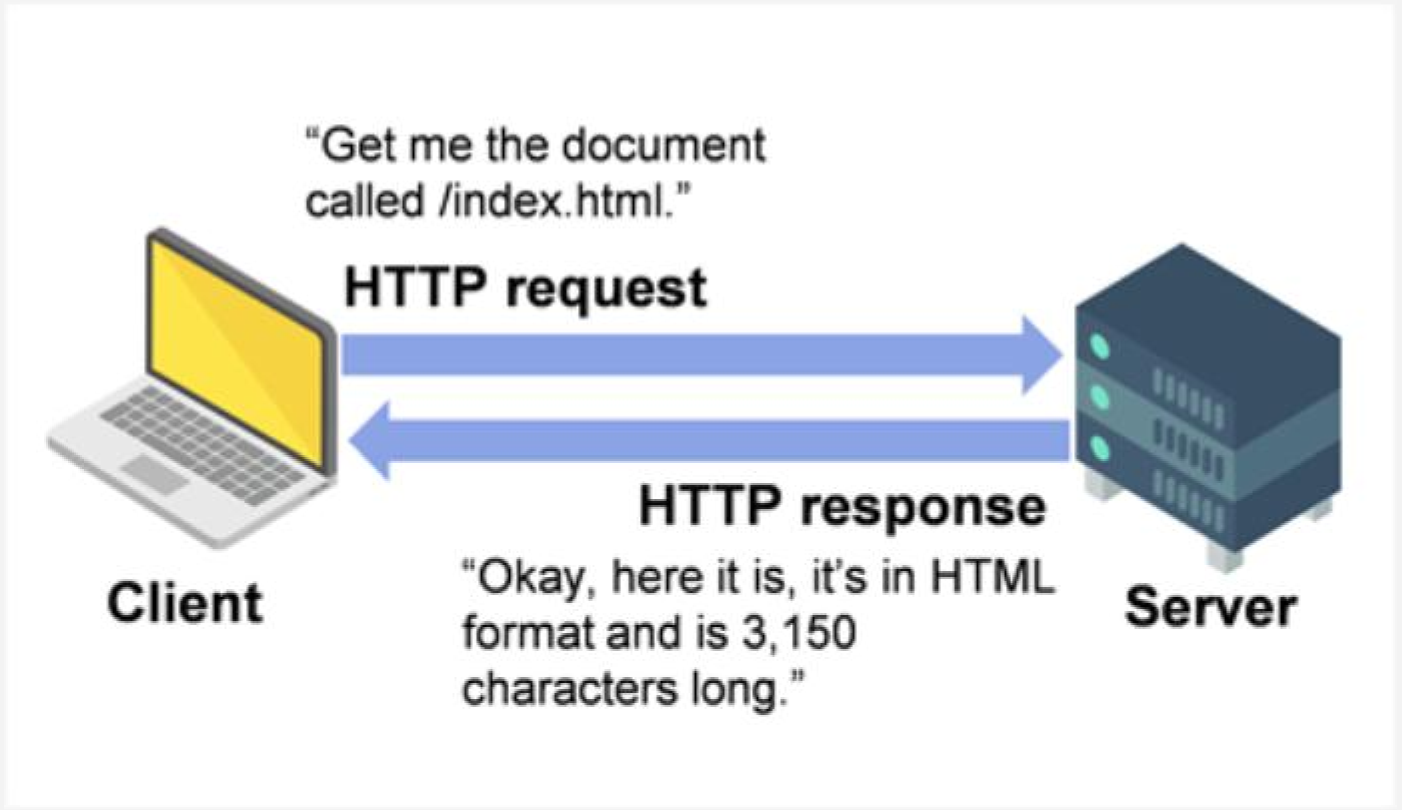
Slide Note

Hyper Text Transfer Protocol - HTTP

- The Hypertext Transfer Protocol (HTTP) is one of the most ubiquitous and widely adopted application protocols on the internet.
- It is the common language between clients and servers, enabling the modern web.
- With HTTP:
  - The client, which is a web browser, raises a request to the server, known as an HTTP Request.
  - The server answers the request and generates response for the client, known as an HTTP Response.

Slide Note

Communication With HTTP


"Get me the document called /index.html."
HTTP request
HTTP response
"Okay, here it is, it's in HTML format and is 3,150 characters long."
Client
Server

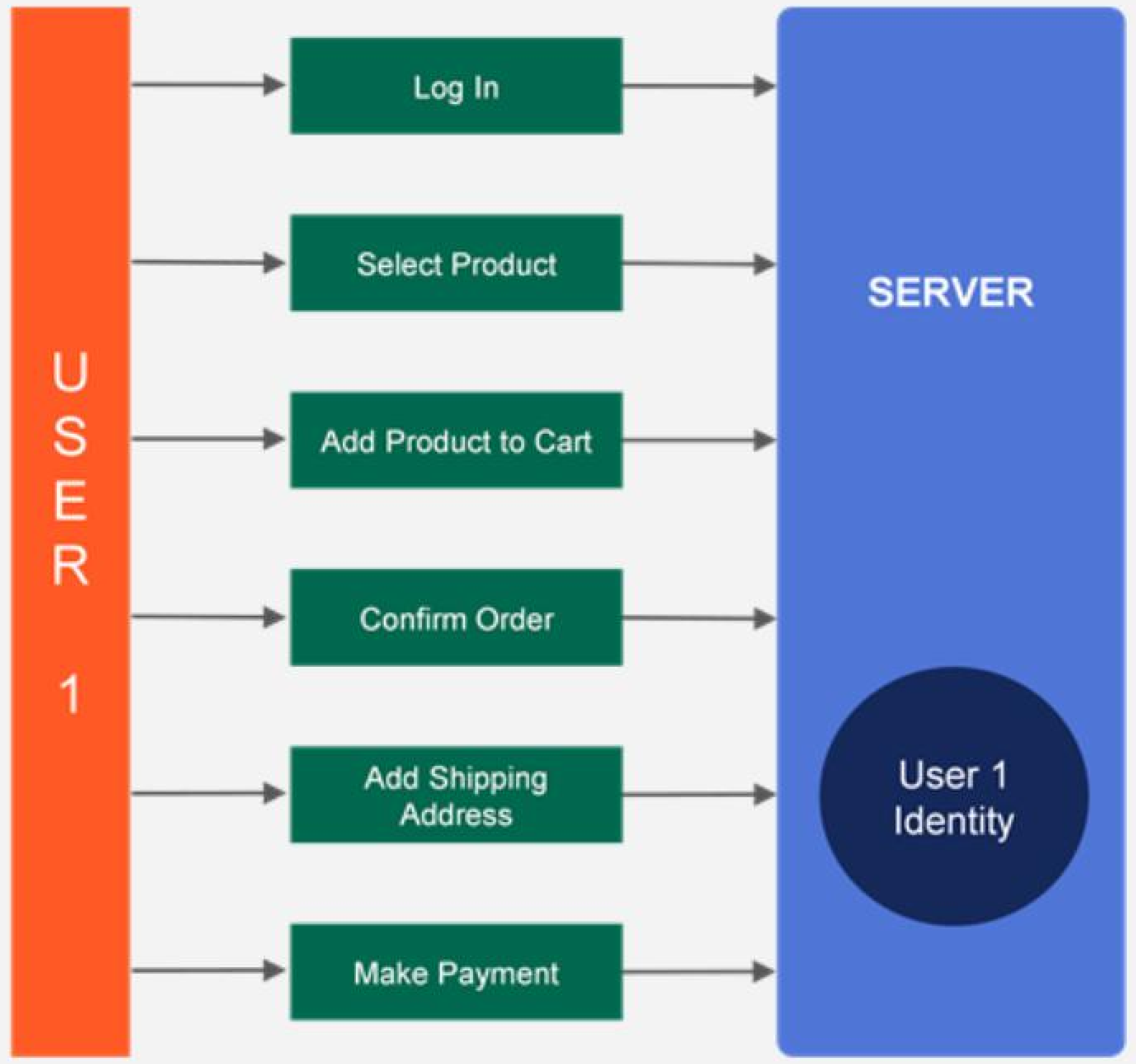
- When you browse to a page, such as http://www.nytimes.com/index.html your browser sends an HTTP request to the server www.nytimes.com.
- The server tries to find the desired object (in this case, "/index.html") and, if successful, sends the object to the client in an HTTP response.
- Along with the object data, it also sends the type of object, the length of the object, and other information.

Slide Note

HTTP Is a Stateless Protocol

- Let's say the user logs in with an online hotel booking site by providing his credentials (login ID and password).
- At the server end, credentials are verified, and users' status is changed to login and the same is communicated to the user via response, say with status 200 (OK).
- HTTP is a stateless protocol.
- Being stateless, the server forgets what request was received and what the response was.
- After the user selects a hotel and proceeds to book a confirmation page, the server does not remember that the request is coming from the same user. Thus, it requests the user to again provide the credentials or data with proof of a previously logged-in state and confirms the identity.
- If the server does not remember the state, is it good to have stateless communication?
- Is it not adding additional overhead on verifying users' identity at every request?

Slide Note



Need for Stateless Communication

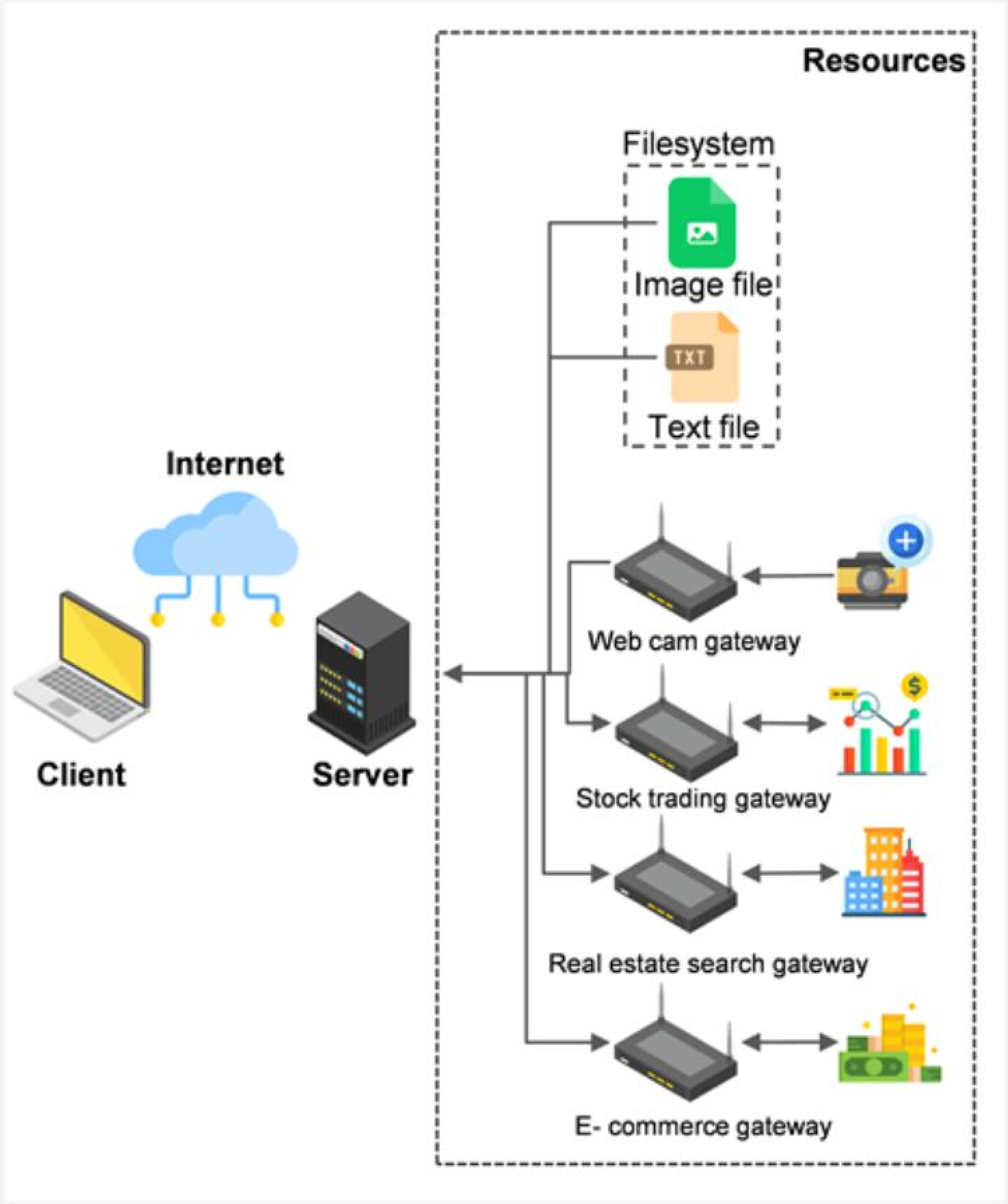
- This diagram is an example of a purchase transaction by a user on an online shopping site.
- To handle requests from many users, multiple servers are required to process them.
- For each task, the server processing the request should know the identity of the user who makes the purchase.
- One of the ways to keep track of the user is to store the user identity once he logs in with the server and keep the communication channel on with the client. What if the server crashes?
- Additionally, keeping the connection channel open between a client and server to track the user is expensive, as it keeps the memory and computing resources occupied.
- Hence, it is more efficient to have Stateless communication as the overheads of performing verification with every request are less expensive than maintaining the state and connection.

Menu

Slide Note

Resources

- Web servers host web resources.
- Web resources are the source of web content.
- The resource can be a file, an image, a movie file, a pdf file or even data in some format.
- The resource could be static, meaning it always provides the same information unless changed.
- The resource can be dynamic which is generated when the request is made.
  - Example: Data in online shopping sites and data from stock trading resources.



Slide Note

HTTP Transaction – Request Methods

HTTP method	Description
GET	Send named resource from the server to the client.
POST	Send client data into a server gateway application.
PUT	Store data from client into a named server resource.
DELETE	Delete the named resource from a server.

- The transaction gets initiated when the client sends a request (command) to the server to perform an operation on a particular resource.
- Example: The browser makes a request to an online shopping site server to fetch product catalog details.
- HTTP supports several different request commands, known as HTTP methods.
- Every HTTP request message has a method.
- The method tells the server what action to perform (fetch products, update shipping details, save order details, cancel order, etc.)
- The slide lists five common HTTP methods.

Slide Note

HTTP Transaction - Response Status Codes

Responses are grouped in five classes:

100 – 199: Informational responses

200 – 299: Successful responses

300 – 399: Redirection responses

400 – 499: Client errors

500 – 599: Server errors

- For every HTTP request sent by the client to the server, the server returns a response as a result.
- The response message contains a status code.
- The status code is a 3-digit numeric value that tells the client if the request is successful, or if some other action is required.
- HTTP also sends an explanatory textual “reason phrase” with each numeric status code.

Menu

00:14/00:00:30 14/ 30

Slide Note

REST – REpresentational State Transfer

- REST is an acronym for **RE**presentational **S**tate **T**ransfer.
- It is a design pattern or architectural style for designing web APIs.
- REST relies on HTTP, and thus is stateless.
- The client makes an HTTP request to RESTful API for requested information referred to as a resource.
    - Example: Products, orders and shipping addresses are examples of resources on an online shopping website.
- At a given instance, the values of the resource determine its state.
    - Example: If a user has successfully signed in his state of login property would be logged in or else it would be logged out.
- When RESTful API is called, the server transfers the representation of the state of the requested resource to the client, usually in JSON (language-independent) text-based format.
    - Example: When a client requests an order delivery status, the server transfers the current state of the delivery with track details.

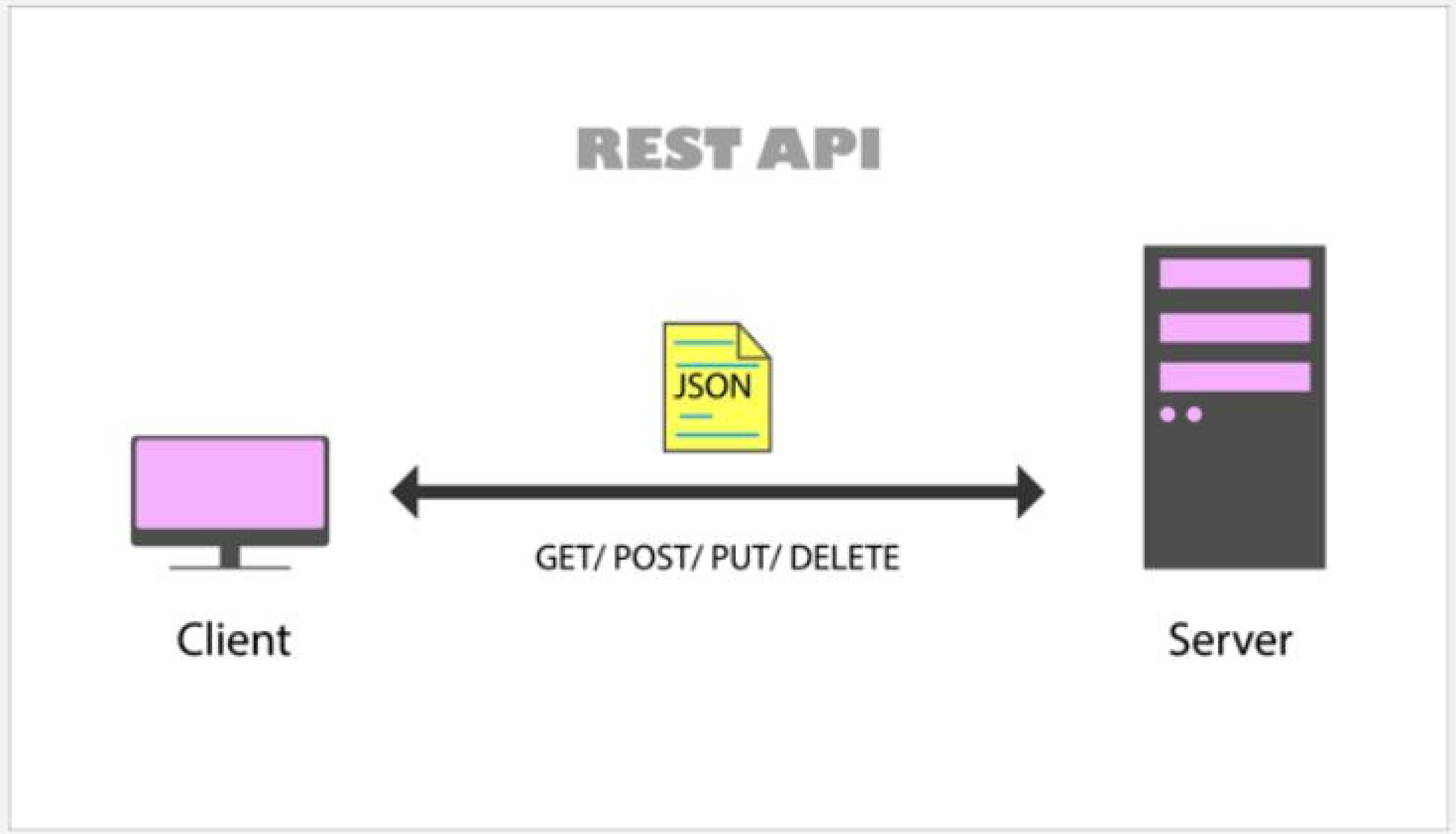
Menu

00:15:00:00:30 15/ 30

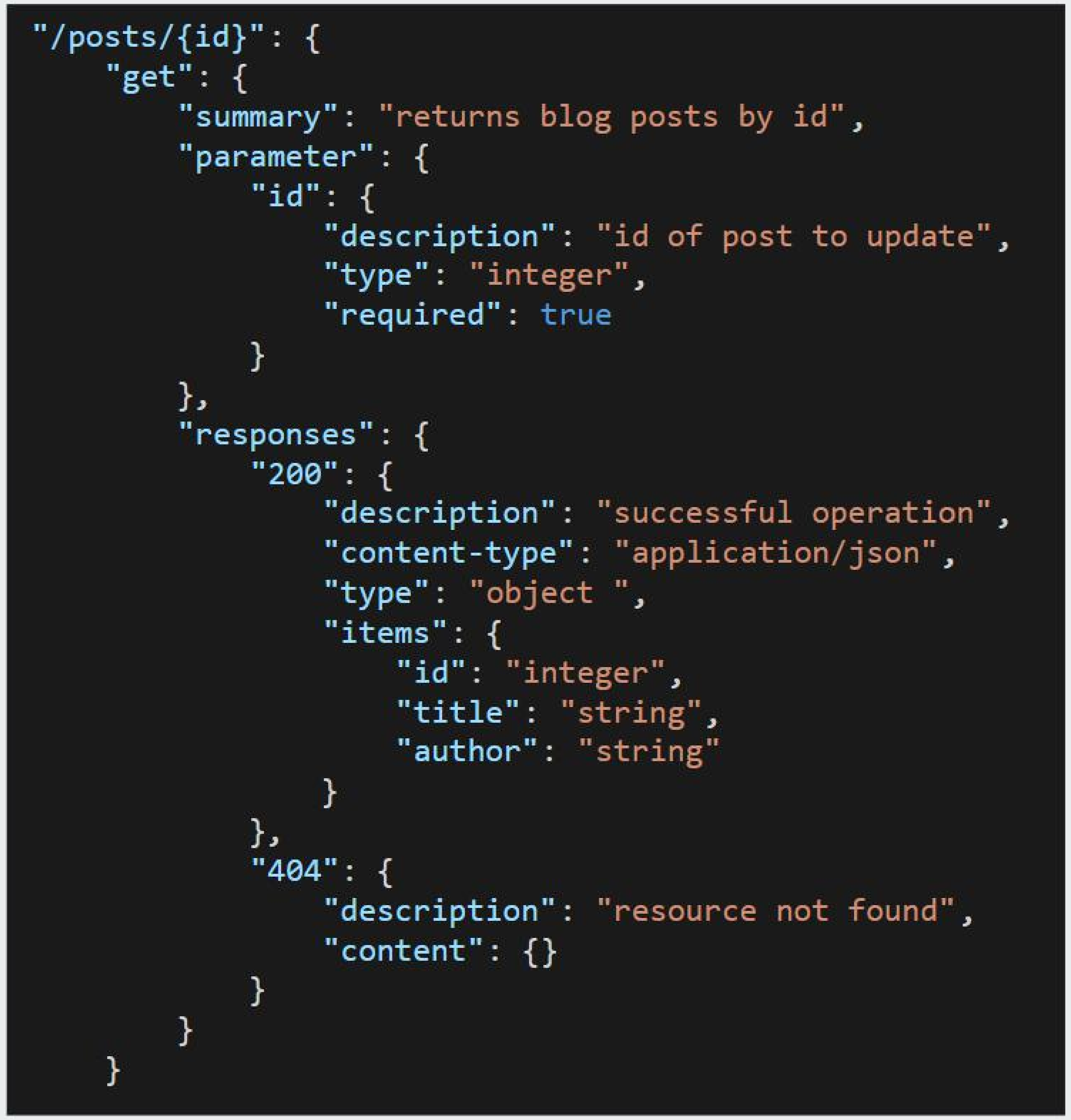
Slide Note

Working With REST APIs

- REST APIs are developed at the server end.
- To make clients aware of the REST APIs, the server should publish them.
  - When the API is published, it means it is now available to receive requests from client applications.
- The published APIs should be documented and shared with the client application developers.
  - The document should contain request and response details in a format and should be language-agnostic like JSON.
- These documents are known as API Specifications.
- At the client end, the APIs are consumed.
  - To consume an API means to make a request to the server using the request URL and handle the response.



Slide Note



```
"/posts/{id}": {
    "get": {
        "summary": "returns blog posts by id",
        "parameter": {
            "id": {
                "description": "id of post to update",
                "type": "integer",
                "required": true
            }
        },
        "responses": {
            "200": {
                "description": "successful operation",
                "content-type": "application/json",
                "type": "object ",
                "items": {
                    "id": "integer",
                    "title": "string",
                    "author": "string"
                }
            },
            "404": {
                "description": "resource not found",
                "content": {}
            }
        }
    }
```

Reading API Specification

- The API spec shown here says:
    - You may send a request for posts to me:
        - With URL path `/posts`
        - For the method `GET`
        - With the parameter `ID`
        - Of type `integer`
- The complete request URL would be of the form **`http://localhost:3000/posts/1`**

Menu

Slide Note

```
"/posts/{id}": {
    "get": {
        "summary": "returns blog posts by id",
        "parameter": {
            "id": {
                "description": "id of post to update",
                "type": "integer",
                "required": true
            }
        },
        "responses": {
            "200": {
                "description": "successful operation",
                "content-type": "application/json",
                "type": "object ",
                "items": {
                    "id": "integer",
                    "title": "string",
                    "author": "string"
                }
            },
            "404": {
                "description": "resource not found",
                "content": {}
            }
        }
    }
```

Reading API Specification (cont'd)

- In return you will receive one of the following responses:
  - Status Code 200
    - Description **successful operation**
    - Content-Type **application/json**
  - Status Code 404
    - Status message **resource not found**
    - Content {}

Menu

Slide Note

Think and Tell

- Let's say JSON data providing details of Hollywood movies is released to date.
- An interactive web application needs to be designed to display and add more movies as per the user's request.
- The backend server with REST APIs is not yet ready.
- Is there a way to simulate the server and get the APIs created?
  - This will help achieve the main objective, which is to write client code.
- Can we create fake REST APIs with zero coding to allow the client application to access movie data?
- Also, is it possible to test these REST APIs before consuming them in the application?

Menu

Slide Note

`json-server` – A Development Server

- A development server is a server that runs locally.
- It is intended for local development and testing.
- `json-server` is one such development server.
- It is developed using JavaScript and is packed with functionalities that help create a fake REST API.
- To install and run it on the local machine, a JavaScript engine is required.
- Node.js is a JavaScript engine.
- Installing Node.js also installs the tool node package manager, or npm.
- This tool helps to install packages like `json-server`.

Menu

Slide Note

Testing APIs Using Postman

- Postman is an API platform for building and using APIs.
- Requests can be sent to API in Postman.
- Postman displays the response received for the request sent from the API allowing us to examine, visualize, and if necessary, troubleshoot the request.

Slide Note

Self-Check

Which of the following is an advantage of RESTful web service being stateless?

1. Web services can treat each method request independently.

2. Web services need not maintain the client's previous interactions. It simplifies application design.

3. As HTTP is itself a stateless protocol, RESTful Web services work seamlessly with HTTP protocol.

4. All of the above.

Menu

Slide Note

Self-Check: Solution

Which of the following is an advantage of RESTful web service being stateless?

1. Web services can treat each method request independently.
2. Web services need not maintain the client's previous interactions. It simplifies application design.
3. As HTTP is itself a stateless protocol, RESTful Web services work seamlessly with HTTP protocol.
4. **All of the above.**

Menu

Slide Note

Self-Check

The type of operation is determined by_______________.

1. HTTP Method
2. HTTP Status Code
3. URL
4. Query string

Menu

00:24 00:00:30 24/ 30

Slide Note

Self-Check: Solution

The type of operation is determined by___________.

1. **HTTP Method**
2. HTTP Status Code
3. URL
4. Query string

Menu

00:25 00:00:30 25/ 30

Slide Note

Self-Check

What does REST stand for?

1. Representational State Tool
2. Resource State Tool
3. Representational State Transfer
4. Resource State Transfer

Menu

Slide Note

Self-Check: Solution

What does REST stand for?

1. Representational State Tool
2. Resource State Tool
3. **Representational State Transfer**
4. Resource State Transfer

Menu

00:27 00:00:30 27/ 30

Slide Note

Self-Check

What is the recommended method and URL pattern for retrieving a specific user?

1. GET /users/id={id}
2. GET /users/{id}
3. GET /users/id:{id}
3. GET /users/?id={id}

Menu

00:28:00:00:30 28/ 30

Slide Note

Self-Check: Solution

What is the recommended method and URL pattern for retrieving a specific user?

1. GET /users/id={id}

2. **GET /users/{id}**

3. GET /users/id:{id}

3. GET /users/?id={id}