4.9.4.10 Client-Server & Thin vs Thick Clients

Client Server Model

A client-server network is a centralised network where a server provides services and resources to client devices.

The client requests services or resources from the server, which processes the request and returns a response.

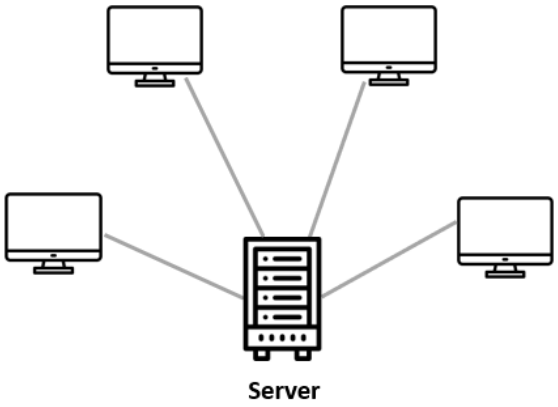
API (Application Programming Interface) is a programmable interface that a server can provide to allow client to use its web resources.

It is a set of functions and protocols that can be used by programmers to build web applications to use resources on the server.

For example, programmers might use the APIs of Google Maps to provide location information on a website or app.

A simple line of code can allow any web page to interact with the server and request data.

<script src= "http://maps.googleapis.com/maps/api/js"></script>



The server is responsible for the bulk of the processing

WebSocket Protocol

WebSocket is a protocol that allows a web browser and server to establish a persistent connection over TCP.

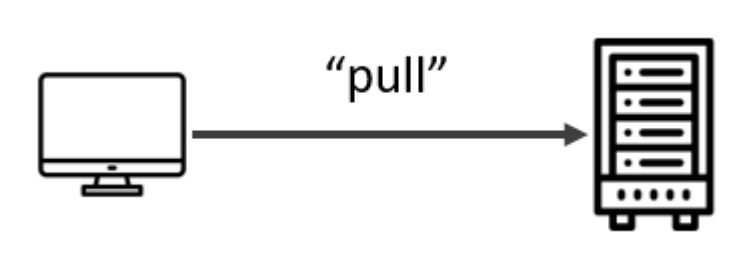
Once the connection is established, both the client and server can send data without repeatedly establishing new connections.

With traditional HTTP request/response communication, the client sends a request and the server sends a response.

The server cannot simply send new application data to the client whenever it wants without a request/connection mechanism.

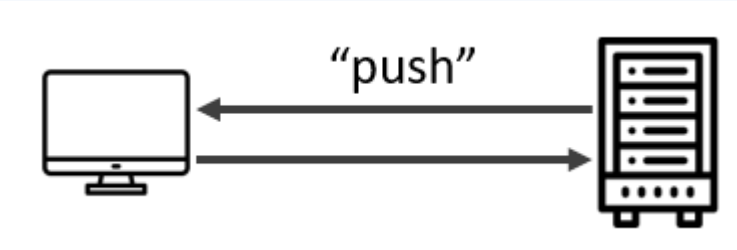
The WebSocket protocol allows the client and server to send data at any time and the server can "push" data to the client.

The connection is full-duplex, meaning both sides can send data simultaneously.



Using HTTP:

1. Client requests "pull" data from the server
2. Too much time passes establishing a connection
3. The server drops the connection
4. The webpage/app displays an error



Using WebSocket:

- Creates a two way connection
- Both sides can send data anytime without re-establishing a connection
- Server can "push" data to client

CRUD

Client server models often make use of databases to store and retrieve information

There are four basic data manipulation operations for databases (often referred to as **CRUD**)

- **Create** – write a record to a database
- **Retrieve** – read/retrieve data from a database
- **Update** – amend a record
- **Delete** – remove a record from the database

HTTP request methods and SQL commands can be used to implement these CRUD operations.

CRUD	HTTP Request Method	SQL Database Command
Create	POST	INSERT
Retrieve	GET	SELECT
Update	PUT	UPDATE
Delete	DELETE	DELETE

REST

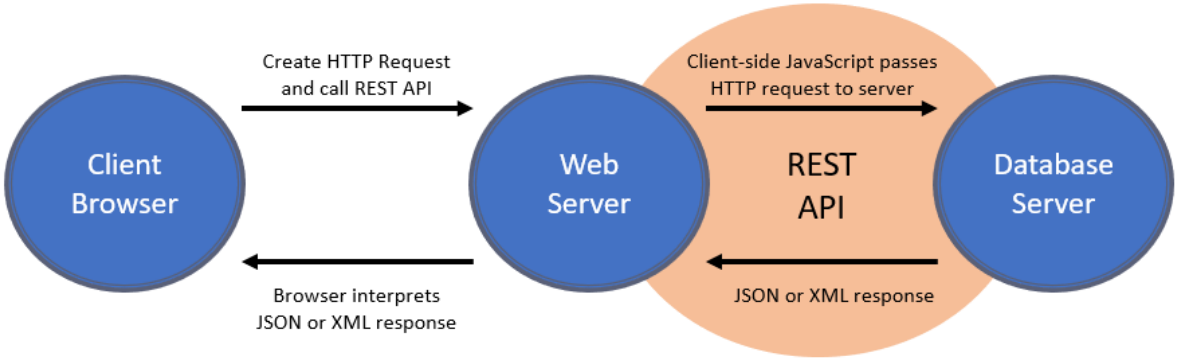
REST (Representational State Transfer) is an architectural style for designing web services and APIs.

It defines how clients and servers should communicate over HTTP in a simple, consistent and stateless way.

RESTful systems commonly use HTTP methods such as GET, POST, PUT and DELETE to perform CRUD operations.

Each request contains all the information needed to process it. The server does not need to store the client's state between requests.

RESTful systems and APIs separate the client from the server database allowing either to be updated independently without impacting on the other.



JSON and XML

JSON and XML are two widely used formats for the standardised data objects that server and client can both process.

- **JSON** – JavaScript Object Notation is written in a standard programming form, similar to JavaScript, and can be easily parsed and manipulated by JavaScript.
- **XML** – Extensible Markup Language has a format similar to HTML, using tags to wrap content

JSON format	XML format
{“dinosaurs” : [{“name”: “Brachiosaurus”, “length”: 30, “period”: “Jurassic” }, {“name”: “Triceratops”, “length”: 30, “period”: “Cretaceous” }] }	<dinosaurs> <dinosaur> <name>Brachiosaurus</name> <length>30</length> <period>Jurassic</period> </dinosaur> <dinosaur> <name>Triceratops</name> <length>30</length> <period>Cretaceous</period> </dinosaur> </dinosaurs>

	Advantages	Disadvantages
JSON	• Easier to for a human to read, write and maintain. • More compact. • Can be directly manipulated by Javascript.	• Supports fewer built-in data types than XML.
• **XML**	• Can represent complex and structured data using custom tags.	• Use of tags makes it more difficult to follow.

Thick vs Thin Clients

Thick and thin-client processing refers to where the bulk of the processing happens when implementing a client-server network

Thick Client

- Most processing is carried out by the client device, while the server mainly provides data, resources or services.
- Client devices need to be reasonably powerful to perform the processing.
- This is relatively expensive and each node needs to be administered individually.

Thin Client

- The majority of the processing takes place on a powerful central server with the clients only displaying the results returned by the server.
- It requires a powerful server and a reliable, responsive network connection but saves the cost of many smaller powerful devices.

	Advantages	Disadvantages
Thin	• Easy to set up, maintain and add terminals. • Server software and updates can be automatically distributed to each client terminal. • Security can be easier to manage centrally.	• Reliant on the server. • Requires a very powerful server. • Server demand and bandwidth increased.
Thick	• Robust and reliable. • Can operate without a continuous connection to the server. • Generally better for running more powerful applications.	• Integrity issues with distributed data. • More expensive client devices are required. • Each client may need to be maintained and updated individually