Sri Lanka Institute of Information Technology

APPLICATION FRAMEWORKS

JAVASCRIPT, AND NOSQL

LECTURE 03

Faculty of Computing
Department of Software Engineering
Module Code: SE3040

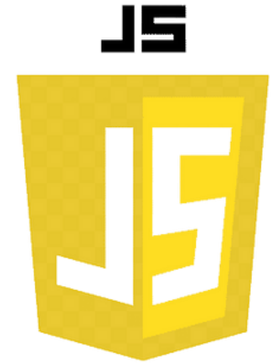
Agenda

1. JavaScript
2. NoSQL

JAVASCRIPT

- Introduction
- Classes, objects
- How ‘this’ acts
- Function closure
- Callbacks and promises

01. JAVASCRIPT



- JavaScript is an interpreted programming language.
- It programs run using a single thread. Though there are ways to create new threads, it is considered as Single threaded language.
- It is considered asynchronous because it follows a Non-Blocking I/O (NIO) model.
- JavaScript is dynamically typed.
- JavaScript support OOP as well as functional programming (Multi-paradigm).
- JavaScript has an eventing system which manages it’s asynchronous operations.

I. CLASSES AND OBJECTS

- A constructor function is used with the ‘new’ key keyword when creating an object.
- When function is used with ‘new’ keyword that function acts as a Class.
- Recently JavaScript introduced ‘class’ keyword, but it is not yet adopted by all JavaScript engines (ES6 Classes)
- Another way of creating an object is using object literals (‘{}’).
- JavaScript supports static methods and variables.

II. “THIS” IN JAVASCRIPT

Unlikely other languages in JavaScript ‘this’ keyword acts differently.

- Inside an object ‘this’ refers to object itself.
- In global context ‘this’ refers to global object (in browser it is the window object). This behavior will get changed in strict mode.
- If a function which is using ‘this’ keyword is being passed to another object then ‘this’ will refers to that object, but not to the original object where function was declared at first place.
- This behavior is very noticeable in callback and closures.

III. CLOSURE

- JavaScript closure is a function which returns another function. In other words, a closure gives you access to an outer function's scope from an inner function.
- It is used to encapsulate variables into a function and restrict access to it from the outside.
- Closure help create private variables that can only be accessed through specific methods.

IV. CALLBACK AND PROMISES

- JavaScript uses asynchronous programming, and it is single threaded.
- Callbacks and Promises are two ways to handle asynchronous execution.

CALLBACK:

- A callback is a function passed as an argument to another function and is executed later, usually after an asynchronous operation completes.
- Nested callbacks passed into sequence of async tasks is referred to a ‘callback hell’.

IV. CALLBACK AND PROMISES CONT...

PROMISES:

- Promise is an object that represents the eventual completion (or failure) of an asynchronous operation. Promise have properties to deal with async operations synchronously.
- Promise object was introduced to solve callback hell problem.
- Promise object has a set of properties, methods and mechanism of chaining to handle complex async tasks nicely.

ACTIVITY

Objective:

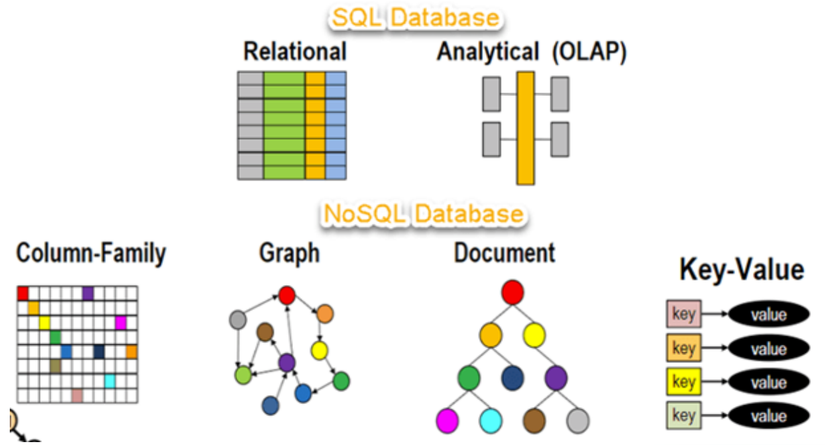
Learn to work with JavaScript Promises to handle asynchronous operations.

Tasks:

1. Write a function that returns a Promise which resolves if a condition (e.g., a number comparison) is true, and rejects if false.

NOSQL

- What?
- Why not SQL?
- Strengths and weaknesses
- MongoDB



WHAT?

- Database system
- Non relational (not using tables), Schema free.
- **Distributed**: Data spread across multiple machines
- **Horizontally scalable:** adding more nodes to a database cluster
- **Easy replication**: copying data across multiple servers
- **Eventually consistent**: prefer availability over strict consistency
- Open source (mostly).
- Lack support to full ACID (mostly)

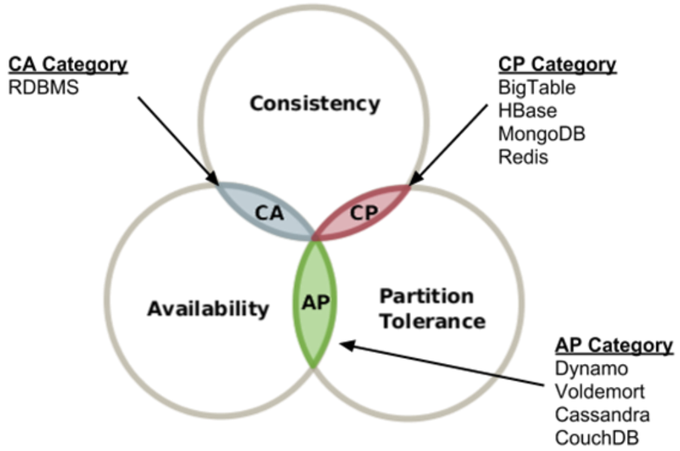
WHY?

- Remove the burden of data structures mismatch between application in-memory (objects) and relational databases (table).
- Integrate databases using services. (ElasticSearch etc.)
- Relational databases not designed to run efficiently on clusters.
- Aggregate oriented databases (store related data together), are easier to manage inside clusters.
- But inter aggregate relationships are harder to manage.

CAP THEOREM

It explains the trade-offs in distributed databases. It states that a distributed system can only guarantee two out of the three properties

- **Consistency** - All nodes see the same data at the same time.
- **Availability** - very request gets a response (even if some nodes fail)
- **Partition tolerance** - The system continues working even if network failures occur



TYPES of NoSQL

- Key-Value - Stores data as key value pairs.
  - Ex: Redis, Riak, Memcached.
- Document - Stores data as documents (JSON, BSON, XML) in maps or collections.
  - Ex: MongoDB
- Column Family - Store data in column families as rows that have many columns associated with.
  - Ex: Cassandra
- Graph - Store entities(nodes) and relationships(edges) between them and represent it in a graph.
  - Ex: Neo4j

TYPES of NoSQL - Summary

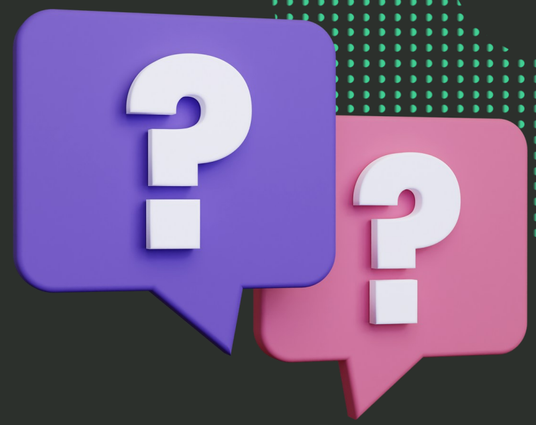
Feature	Document-Oriented	Key-Value Store	Column-Family Store	Graph Database
Data Structure	JSON, BSON Documents	Key-Value Pairs	Column-Family	Nodes & Relationships
Schema Flexibility	✅ Schema-free	✅ Schema-free	✅ Schema-free	✅ Schema-free
Best For	Complex, semi-structured data	Fast lookups, caching	Big data, analytics	Relationship-heavy data
Scalability	✅ High (Sharding & Replication)	✅ Very High	✅ Very High	❌ Limited
Querying	Queryable (like SQL)	Key-based (GET/SET)	Optimized for column queries	Graph queries
Joins Supported?	❌ No joins	❌ No joins	❌ No joins	✅ Supports joins
Example Databases	MongoDB, CouchDB, Firebase	Redis, DynamoDB, Riak	Cassandra, HBase, ScyllaDB	Neo4j, ArangoDB, JanusGraph

MongoDB

- NoSQL document database.
- Strong query capabilities with aggregations using JavaScript.
- Use SpiderMonkey JavaScript engine.
- High availability with replica sets.
- Reads and writes on primary by default.
- Eventually consistent on secondary instances.
- In built file storage called Grid File System.

MongoDB queries

- Insert
- Find
- Update
- Remove



THAT'S ALL FOLKS !

ANY QUESTIONS ?