**Nama: Muhammad Rizki Fonna**

Level 1 Basic Knowledge

1. Explain about Event Loop & how Event Loop works

2. Give an explanation about Callback Hell & how to avoid it?

3. Give an explanation about the Middleware concept

4. Give an explanation about Buffer

5. Give a list of differentiation of JavaScript iterator functions & For Loop

6. What is GraphQL?

7. What is Resolver in GraphQL?

8. Give a list of pros & cons using Embedded Document & Reference ID in MongoDB

9. What is Aggregation in MongoDB?

10. Give an explanation about Pipeline in Aggregation MongoDB

Answer:

1. Event loop is one of a programming design pattern. It waits for and dispatches events or messages. The event loop makes a request to event provider, and then calls the corresponding event handler. Javascript uses message queue to know which events that hasn’t been done to not lose track of the events and messages.
2. Callback hell is defined as too many callbacks inside of another callback. If it seen, it is just like pyramid, since there are many indentations (every callback function need to be indented and so on). To avoid it we can use promise instead of callback.
3. Middleware are functions that is usually executes between the requester and the receiver (node js app) It have access to the request object, the response object, and the next function in the application’s request-response cycle. The next function is a function in the router which, when invoked, executes the middleware succeeding the current middleware.
4. Buffer objects are objects that are used to represent sequence of bytes that is fixed in length. Buffer also can be converted to strings and vise versa.
5. Generally, javascript iterator function is faster than for loop. Iterator is an interface provided to traverse a collection and for a sequential access of items in the collection. For loop only loops through certain conditions doesn’t always iterate through collections.
6. GraphQL is a query language for API. Usually with REST API, we need for fetch one resource from one endpoint and join it with resource from another endpoint. This usually makes it several steps to fetch something, but in graphql we just need to fetch data from one endpoint and use the query to define what we want to fetch.
7. Resolver in graphQL is a function that return or resolves a value for a type or field in a graphql schema. It can return object or primitive (strings, numbers, etc). If an object is returned, the execution runs through to the next child field. Else, the execution stops.
8. - Using embedded document would eliminates join (like in relational database), but wouldn’t really fit if the logic needs many joins.

* Complex data structures: if there too many nested objects (using embedded document), it would be harder to find something in the data.
* Data consistency: If using embedded document, it would be always atomic operations, whereas if it is using reference is, the operation might be not atomic and can become inconsistent.

1. Aggregation in mongodb is quite similar to aggragation in relational database, that we group values from multiple documents together. In the processes of grouping, we can filter, sort, count, sum several documents.
2. Pipeline in agregation in mongodb is similar to pipelines in general, consists of several processes. We can make a pipilene, let’s say we have pipeline sort and after that filter. It means that after we sort the document, we filter it. Then, we get grouping of the several documents.