CS 310 : Scalable Software Architectures

Class session on Thursday, October 24th

October 2024

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

www.a-printable-calendar.com

Notes:

- ***Project 02*** *due Friday night (or Sunday night late)*
  - *Part 01: web service for photoapp, rewrite client*
  - *Part 02: deploy to AWS EC2*
  - *Gradescope open*
- ***Class today?***
  - *Review for Tuesday's exam*
  - *Exam covers all material through Tues 10/22*

Northwestern University

JavaScript [20 points]

1) Consider the following JavaScript function (written to execute with Node.js and Express js). Assume no errors occur when the code executes. Answer the questions below.

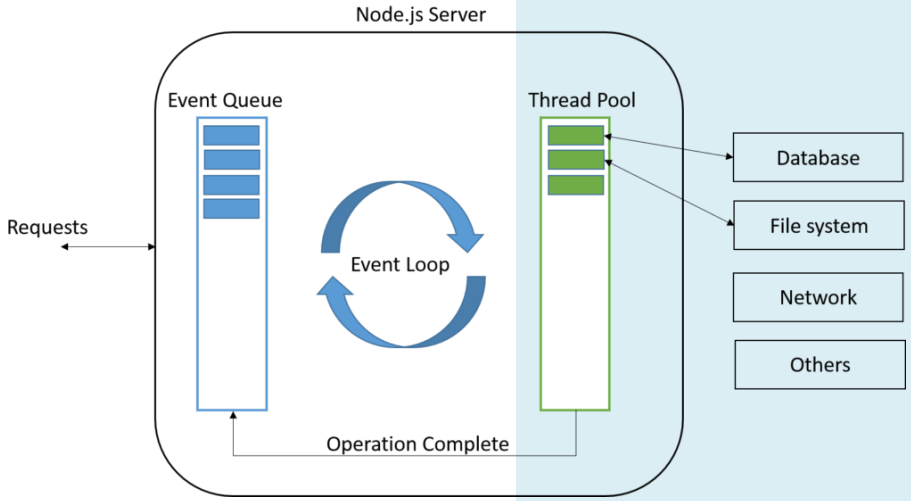
```
app.get('/path', (req, res) => {
  try {
    var count = -1;
    console.log("1");

    dbConnection.query("Select * from …;", (err, rows) => {
       if (err) { console.log("2"); }
       count = count + 5;
       console.log("3");

       dbConnection.query("Select * from …;", (err2, rows2) => {
          if (err) { console.log("4"); }
          count = count + 5;
          console.log("5");
       });

       console.log("6");
    });

    console.log("7");
    res.send(count.toString());
    console.log("8");
  }
  catch(err) {
    console.log("9");
```

a) What value is returned to the client?

b) What output appears on the server console as the function executes?

12) Consider the following JavaScript function (written to execute with Node.js and Express js). Assume this code executes without error:

```
1.  app.get('/somepath', async (req, res) => {
2.      response1 = s3.send("Command1");
3.      response2 = s3.send("Command2");
4.      response3 = s3.send("Command3");
5.
6.      var result1 = await response1;
7.      console.log("1");
8.
9.      var result2 = await response2;
10.     console.log("2");
11.
12.     var result3 = await response3;
13.     console.log("3");
14.
15.     res.send("done");
16. });
```



Answer here: https://pollev.com/hummel

Assume that response1 has resolved when execution reaches line 6, but response2 and response3 have not yet resolved. What happens next? Circle the best answer:

a) The main thread outputs "1" to the server console, then executes the await on line 9 which causes the main thread to block and wait on this line

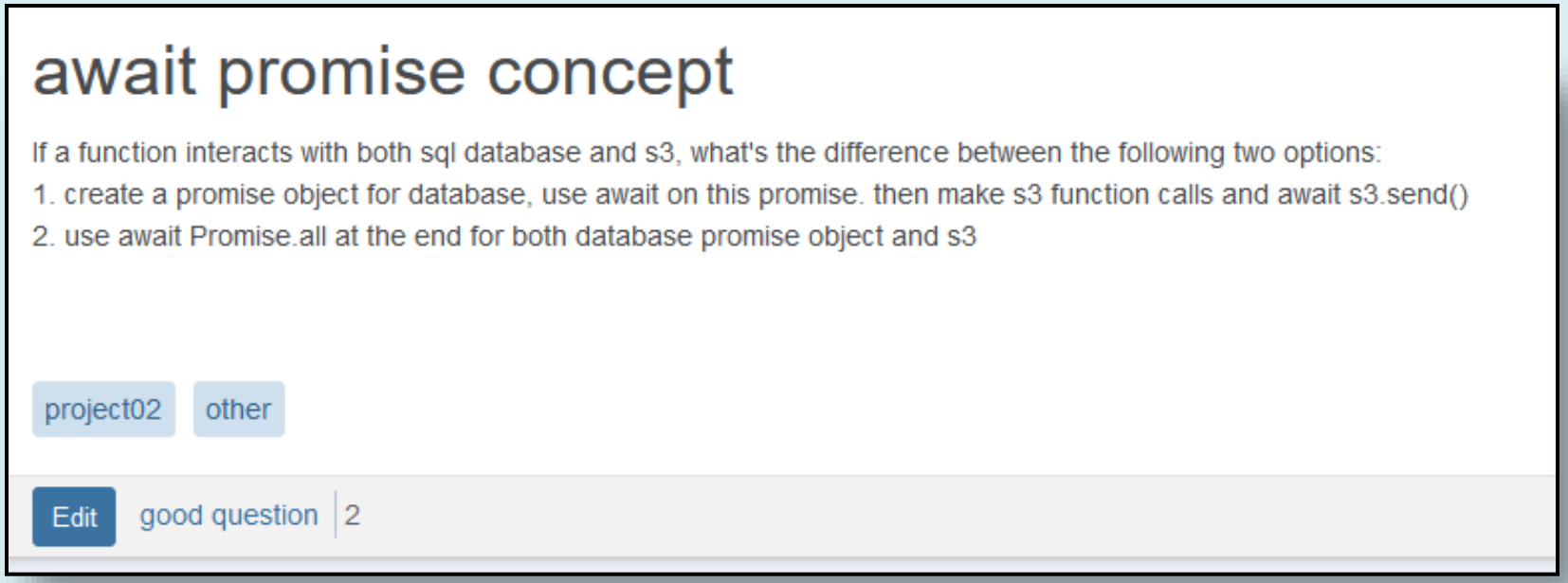
b) The main thread outputs "1" to the server console, executes the await on line 9, and returns from the function in order to process other work

c) The main thread sends result1 to the client, outputs "1" to the server console, then executes the await on line 9 to wait

d) The main thread outputs "1" then "2" then "3" to the server console, then waits before responding to the client

Visualize

- **Explain 1 vs. 2 by drawing fork-join diagrams…**



Design scenario example

- **We have a scenario with 4 async operations:**
  - *Functions A, B, C, and D each return a promise*
  - *Respond with "done" when operations have completed*

Design scenario #1

- **We have a scenario with 4 async operations:**
  - *Functions A, B, C, and D each return a promise*
  - *The results of A and B are input parameters to C*
  - *Respond with "done" when operations have completed*

Design scenario #2

- **We have a scenario with 4 async operations:**
  - *Functions A, B and C each return a promise*
  - *Function D uses a callback, where the result of B is a parameter to D*
  - *Respond with "done" when operations have completed*

latency

3) Clients are calling into the web service to obtain data from a database, and server traffic is light (not many users). Clients are reporting that the application feels slow --- you determine that latencies are too long. A solution is (circle the best answer):

a) Scale up with more cores

b) Scale out with more machines

d) Add caching (also known as a content delivery network, e.g. AWS CloudFront)

e) None of the above

Answer here: https://pollev.com/hummel

scalability of node.js

Node.js is a runtime engine for executing JavaScript. It is often combined with the Express framework, and represents one of the most popular approaches for building web services. Interestingly, Node.js is single-threaded, and thus limited in how many clients it supports. How can Node.js scale to support millions of clients? Circle the best answer:

a) With simple programming changes, it is easily replicated across cores and machines

b) It is easily reconfigured into multi-client mode with a few button clicks / deployment script changes

c) Cloud providers --- e.g. Amazon with Elastic Beanstalk --- provide services that take Node.js code and automatically rewrite the code to execute at scale

d) Even though Node.js is single-threaded, requests and responses are handled by the main thread on separate cores / machines to dramatically increase throughput

Answer here: https://pollev.com/hummel

S3

13) AWS S3 is Amazon's Simple Storage Service. What is the purpose of S3? Circle the best answer:

a) High-performance access to files in support of EC2 compute instances

b) Long-term, lower-cost storage of data

c) Replace your local laptop or phone storage with a high-performance alternative that is automatically backed up

d) All of the above

Answer here: https://pollev.com/hummel

JSON

15) Why is JSON such a popular serialization format? Circle the best answer:

a) JSON data is faster to send over a network vs. data sent as raw 0's and 1's

b) JSON yields shorter messages in comparison to messages built from raw 0's and 1's

c) JSON is human-readable and platform-neutral

d) All of the above

Answer here: https://pollev.com/hummel

web services

- **When a client calls a web service function, which network protocol is the client using? Circle the best answer:**

A) HTTP

B) FTP

C) SMTP

D) HTML

Answer here: https://pollev.com/hummel

Database design

11) You are building a shopping application and need to decide where to store the shopping cart. After much discussion, you decide to store the shopping cart in the database so that (1) the cart is accessible no matter which web server receives the next request, and (2) the cart isn't lost if the client closes their browser or reboots their machine. The question is what's the best way to store the cart in the database. Circle the best answer based on following proper relational database design principles:

a) Store the contents of the shopping cart in one row of the Customers table

b) Store the contents of the shopping cart in one row of the Products table

c) Store the contents of the shopping cart in one row of the Orders table

d) Store the contents of the shopping cart in an OrderItems table, with one row per cart item

Answer here: https://pollev.com/hummel

Database design

14) In a relational database, what is a foreign key? Circle the best answer:

a) A foreign key is a value in one table that uniquely identifies a row in another table

b) A foreign key is a value in one database that uniquely identifies data in another database

c) Foreign keys are used to join tables --- a foreign key in one table must be joined with a foreign key in another table

d) Foreign keys are used to provide alternative indexing structures to improve search performance

Answer here: https://pollev.com/hummel

That's it, thank you!