TW-09 TEAM LEAD VERSION

TEAMWORK

CLARUSWAY
WAY TO REINVENT YOURSELF

Meeting Agenda

- Icebreaking
- Workshop Activities - Tuesday
- Teamwork Activities - Friday
  - Questions
  - Interview Questions
- Video of the week
- Retro meeting
- Case study / project

Teamwork Schedule

Ice-breaking 10m

- Personal Questions (Stay at home & Corona, Study Environment, Kids etc.)
- Any challenges (Classes, Coding, studying, etc.)
- Ask how they're studying, give personal advice.
- Remind that practice makes perfect.

Workshop Activities (Tuesday) 90m

> No extra task is given for the workshop. Before Session 2, a recap of Coin App session 1 is expected.

Team Work Activities (Friday) 90m

Ask Questions 20m

1. What is the function to stop an interval timer?

A. stopTimer
B. clearInterval
C. shutdownTimer
D. clearTimer

Answer: B

2. What are 2 native functions to run code asynchronously in JavaScript ?

A. timeout - setTimeout
B. startInternal - setInterval
C. setTimeout - setInterval
D. interval - setInterval

Answer: C

3. What is the output of the code below?

```
let x = 0;

async function test() {
  x += await 2;
  console.log(x);
}

test();
x += 1;
console.log(x);
```

A. 2 3
B. 0 1
C. 1 2
D. 2 2

Answer: C

Explanation : When test() function called x was 0 and an asyncronous task started as x = x + 2. Once this promise fulfilled result of x will be 2 as 0 + 2 equals to 2.

4. Which method converts JSON data to a JavaScript object?

A. `JSON.fromString();`
B. `JSON.toObject()`
C. `JSON.stringify()`
D. `JSON.parse()`

Answer: D

5. Which statement is true about the "async" attribute for the HTML script tag?

A.It can be used for both internal and external JavaScript code.
B.It can be used only for internal JavaScript code.
C.It can be used only for internal or external JavaScript code that exports a promise.
D. It can be used only for external JavaScript code.

Answer: D

6. Which Queue Is Executed First?

```
// Callback queue
setTimeout(() => console.log("timeout"), 0);

// Microtask queue
Promise.resolve().then(() => console.log("promise"));
```

A. Callback queue
B. Microtask queue
C. No priority
D. Depends on which one called first

Answer: B

7. Consider the following async function and its output. What will be displayed to the console when calling the f() function?

```
async function f() {
  let result = "first!";
  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("done!"), 1000);
  });

  result = await promise;

  console.log(result);
}

f();
```

A. first!
B. done!
C. JavaScript error
D. Something else

Answer: B

8. What will the output be?

```
Promise.resolve("Success!")
  .then((data) => {
    return data.toUpperCase();
  })
  .then((data) => {
    console.log(data);
  });
```

A. print "Success!" and "SUCCESS!"
B. print "Success!"
C. print "SUCCESS!"
D. nothing prints

Answer: C

9. What is the role of the event loop in asynchronous JavaScript?

A. It ensures synchronous execution.
B. It manages asynchronous tasks and callbacks.
C. It handles UI events exclusively.
D. It prevents the use of async/await.

Answer: B

10. What will be logged to the console when the following code is executed?

```
async function example() {
  throw new Error("Something went wrong");
}

example().catch((error) => console.error(error.message));
```

A. No output, the error is ignored.
B. The code will throw an unhandled promise rejection error.
C. `undefined`
D. `Something went wrong`

Answer: D

Interview Questions 15m

1. How can you convert a callback-based function to a Promise-based function?

Answer: To convert a callback-based function to a Promise-based function, you can wrap the callback-based function with a Promise and pass the resolve and reject functions to it. You call the resolve function when the asynchronous operation is successful and the reject function when the operation fails. You return the Promise in the function and resolve or reject the Promise based on the result of the asynchronous operation. It is possible for callers of the Promise-based function to use the then method to access the resolved value and the catch method to handle errors.

2. Can you explain the difference between an async function and a regular function in JavaScript?

Answer : Async functions are functions that allow you to use the await keyword to wait for a promise to resolve before continuing execution of the function. Regular functions do not have this ability, and will instead execute the code inside of them immediately.

3. How do you handle errors in an asynchronous JavaScript code?

Answer : You can use **try/catch** blocks to handle errors in asynchronous code. When using async/await, the **try** block wraps the asynchronous code, and the **catch** block catches any errors that occur during the asynchronous execution.

4. Can you use async/await with regular synchronous functions in JavaScript?

Answer: Yes, you can use **async/await** with regular synchronous functions. If an **async** function calls a regular synchronous function, the synchronous function's result is implicitly wrapped in a resolved promise, allowing you to use await with it.

☕

Coffee Break 10m

☕

Retro Meeting on a personal and team level 5m

Ask the questions below:

- What went well?
- What went wrong?
- What is the improvement areas?

Case study/Project 15m

- StopWatch
- **Optional project will be shared on Tuesday for those who wish.**

Closing 5m

- Next week's plan
- QA Session