Async/Await Review Sheet & API Activities

QUICK REFERENCE GUIDE

Basic Async/Await Syntax

```
// Making a function asynchronous
async function myFunction() {
    // Code here
}

// Waiting for a promise to resolve
async function fetchData() {
    const result = await someAsyncOperation();
    return result;
}

// Always use try/catch for error handling
async function safeFunction() {
    try {
        const data = await riskyOperation();
        console.log(data);
    } catch (error) {
        console.log('Error:', error.message);
    }
}
```

Fetch API Pattern

```
async function getData(url) {
    try {
        const response = await fetch(url);
        // Check if request was successful
        if (!response.ok) {
            throw new Error(`HTTP error! status:
${response.status}`);
        }

        const data = await response.json();
        return data;
    } catch (error) {
        console.log('Fetch error:', error.message);
        return null;
    }
}
```

Common Mistakes to Avoid

1. **Forgetting 'await'** - Promise won't wait
2. **Not using 'async'** - Can't use await without async function
3. **Missing try/catch** - Errors will crash your program
4. **Not checking response.ok** - Failed requests still return response objects

SETUP INSTRUCTIONS

How to Run the Code

Option 1: Browser Console

1. Open any website in your browser
2. Press F12 to open Developer Tools
3. Go to the Console tab
4. Copy and paste code directly
5. Press Enter to run

Option 2: HTML File

1. Create a new file called async-practice.html
2. Use this template:

```
<!DOCTYPE html>
<html>
<head>
    <title>Async Practice</title>
</head>
<body>
    <h1>Check the Console for Output</h1>
    <script>
        // Your JavaScript code goes here
        console.log('Starting async practice...');
    </script>
</body>
</html>
```

3. Open the HTML file in your browser
4. Press F12 to see console output

Option 3: Code Editor with Live Server

1. Use VS Code with Live Server extension
2. Create HTML file as above
3. Right-click and select "Open with Live Server"

ACTIVITY 1: BASIC API CALLS (15 minutes)

Task A: Random Quote Fetcher

API: https://api.quotable.io/random

Your Mission: Fetch a random quote and display the content and author.

```
async function getRandomQuote() {
    try {
        const response = await
fetch('https://api.quotable.io/random');
        const quote = await response.json();

        console.log('Quote:', quote.content);
        console.log('Author:', quote.author);
        console.log('Length:', quote.length, 'characters');

        return quote;
    } catch (error) {
        console.log('Error getting quote:', error.message);
    }
}

// Test it
getRandomQuote();
```

Challenge Questions:

1. What other properties are in the response object?
2. How would you get a quote from a specific author?
3. Add error handling for network failures.

Task B: Dog Image Fetcher

API: https://dog.ceo/api/breeds/image/random

Your Mission: Fetch a random dog image URL.

```
async function getRandomDog() {
    // YOUR CODE HERE
    // Hint: The image URL is in the 'message' property
}

getRandomDog();
```

Expected Output:

Dog image: https://images.dog.ceo/breeds/hound-english/n02089973_612.jpg
Status: success

ACTIVITY 2: HANDLING DIFFERENT DATA TYPES (15 minutes)

Task C: Number Facts

API: http://numbersapi.com/42

Your Mission: Get an interesting fact about the number 42.

```
async function getNumberFact(number) {
    try {
        const response = await
fetch(`http://numbersapi.com/${number}`);

        // Note: This API returns plain text, not JSON
        const fact = await response.text();

        console.log(`Fact about ${number}:`, fact);
        return fact;
    } catch (error) {
        console.log('Error:', error.message);
    }
}

// Test with different numbers
getNumberFact(42);
getNumberFact(7);
getNumberFact(365);
```

Student Challenges:

1. Try your birthday (month and day as one number)
2. Try your age

3. What happens with negative numbers?
4. Modify the function to handle dates (add /date to the URL)

Task D: Weather Data

API: https://api.open-meteo.com/v1/forecast?latitude=40.7128&longitude=-74.0060&current_weather=true

Your Mission: Get current weather for New York City.

```
async function getCurrentWeather() {
    const url = 'https://api.open-
meteo.com/v1/forecast?latitude=40.7128&longitude=-
74.0060&current_weather=true';

    try {
        const response = await fetch(url);
        const data = await response.json();

        const weather = data.current_weather;
        console.log('Temperature:', weather.temperature, '°C');
        console.log('Wind Speed:', weather.windspeed, 'km/h');
        console.log('Weather Code:', weather.weathercode);

        return weather;
    } catch (error) {
        console.log('Weather error:', error.message);
    }
}

getCurrentWeather();
```

Challenges:

1. Convert temperature from Celsius to Fahrenheit
2. Look up what the weather codes mean
3. Try different coordinates (your city)

ACTIVITY 3: ERROR HANDLING PRACTICE (15 minutes)

Task E: Handling Bad Requests

Test these scenarios and observe what happens:

```
async function testErrorHandling() {
    const badUrls = [
        'https://httpstat.us/404',  // Will return 404 error
        'https://httpstat.us/500',  // Will return 500 error
        'https://fake-url-that-does-not-exist.com', // Network
error
    ];

    for (const url of badUrls) {
        console.log(`Testing: ${url}`);
        try {
            const response = await fetch(url);

            if (!response.ok) {
                throw new Error(`HTTP ${response.status}:
${response.statusText}`);
            }

            const data = await response.text();
            console.log('Success:', data);
        } catch (error) {
            console.log('Caught error:', error.message);
        }
        console.log('---');
    }
}

testErrorHandling();
```

Questions to Consider:

1. What's the difference between network errors and HTTP errors?
2. Why do we need to check response.ok?

3. When should you use .json() vs .text()?

ACTIVITY 4: MULTIPLE REQUESTS (15 minutes)

Task F: Sequential vs Parallel Requests

Compare these two approaches:

```
// Sequential (slow - one after another)
async function getQuotesSequential() {
    console.log('Getting quotes sequentially...');
    const start = Date.now();

    try {
        const quote1 = await
fetch('https://api.quotable.io/random');
        const data1 = await quote1.json();

        const quote2 = await
fetch('https://api.quotable.io/random');
        const data2 = await quote2.json();

        const quote3 = await
fetch('https://api.quotable.io/random');
        const data3 = await quote3.json();

        const end = Date.now();
        console.log(`Sequential took ${end - start}ms`);
        console.log('Quote 1:', data1.content);
        console.log('Quote 2:', data2.content);
        console.log('Quote 3:', data3.content);
    } catch (error) {
        console.log('Error:', error.message);
    }
}

// Parallel (fast - all at once)
async function getQuotesParallel() {
    console.log('Getting quotes in parallel...');
    const start = Date.now();

    try {
        const promises = [
            fetch('https://api.quotable.io/random'),
            fetch('https://api.quotable.io/random'),
            fetch('https://api.quotable.io/random')
        ];

        const responses = await Promise.all(promises);
        const quotes = await Promise.all(responses.map(r =>
r.json()));

        const end = Date.now();
        console.log(`Parallel took ${end - start}ms`);
        quotes.forEach((quote, index) => {
            console.log(`Quote ${index + 1}:`, quote.content);
        });
    } catch (error) {
        console.log('Error:', error.message);
    }
}

// Test both
getQuotesSequential();
setTimeout(() => getQuotesParallel(), 2000);
```

Student Experiment:

1. Run both functions and compare the timing
2. Which is faster and why?
3. When might you prefer sequential over parallel?

FINAL CHALLENGE: BUILD YOUR OWN API MASHUP

Task G: Multi-API Application

Your Mission: Combine 2-3 different APIs to create something interesting.

Example Ideas:

- Get a random quote + random dog image + current weather
- Fetch number fact + dog image for that breed number
- Get quote + weather + display both with styling

Template:

```
async function createMashup() {
    try {
        // Step 1: Get data from first API
        const response1 = await fetch('API_URL_1');
        const data1 = await response1.json();

        // Step 2: Get data from second API
        const response2 = await fetch('API_URL_2');
        const data2 = await response2.json();

        // Step 3: Combine and display the data
        console.log('Mashup Result:');
        // Your creative combination here

    } catch (error) {
        console.log('Mashup error:', error.message);
    }
}

createMashup();
```

DEBUGGING CHECKLIST

When your async code isn't working:

Check These Common Issues:

- [ ] Did you use async before the function?
- [ ] Did you use await before the fetch call?
- [ ] Did you check response.ok before parsing JSON?
- [ ] Are you using .json() for JSON APIs and .text() for text APIs?
- [ ] Is your try/catch block around the right code?
- [ ] Are you calling the function after defining it?
- [ ] Check the browser console for error messages
- [ ] Verify the API URL is correct (copy/paste from examples)

Testing Tips:

- Test with simple console.log statements first
- Add console.log before and after each await
- Check network tab in browser dev tools
- Try the API URL directly in your browser

EXTENSION ACTIVITIES

For Students Who Finish Early:

1. **API Explorer:** Find a new public API and write a function to use it
2. **Error Recovery:** Build retry logic for failed requests
3. **Rate Limiting:** Add delays between requests
4. **Local Storage:** Save API responses to avoid repeated calls
5. **User Interface:** Create HTML elements to display your API data

Useful Free APIs to Explore:

- https://jsonplaceholder.typicode.com/posts/1 - Fake blog posts
- https://api.github.com/users/octocat - GitHub user info
- https://catfact.ninja/fact - Cat facts
- https://official-joke-api.appspot.com/random_joke - Random jokes