JavaScript Asynchronous & Synchronous

Section 1: Theoretical Questions

1. What does “synchronous” code mean in JavaScript?
A) Code that runs in parallel
~~B)~~ Code that runs one line after another, blocking further execution until completion
C) Code that runs in the background
D) Code that depends on callbacks

2. Which of the following is **true** about asynchronous code in JavaScript?
A) It blocks other code until it finishes
B) It runs only on multiple threads
~~C)~~ It allows non-blocking operations
D) It cannot use callbacks

3. What is the main purpose of a **callback function**?
A) To run before another function
B) To delay execution until a task completes
~~C)~~ To handle synchronous code
D) To define global variables

4. Which statement is **true** about Promises?
~~A)~~ Promises replace all callbacks
B) Promises represent a value that might be available now, later, or never
C) Promises always resolve successfully
D) Promises are synchronous by default

5. Which of the following states can a Promise be in?
A) Waiting, Done, Failed
~~B)~~ Pending, Fulfilled, Rejected
C) Open, Closed, Broken
D) Running, Completed, Failed

6. What is the output of asynchronous code handled by the JavaScript runtime?
A) Immediately returned values
B) Handled via the call stack only
~~C)~~ Managed through the event loop and callback queue
D) Run in a separate thread

7. What does `.then()` do in a Promise chain?
A) Blocks code execution until resolved
~~B)~~ Runs when the Promise is resolved

C) Runs when the Promise is rejected
D) Cancels a Promise

8. Which function allows you to pause execution until a Promise resolves when used inside an async function?

A) `.wait()`
B) `setTimeout()`
~~C)~~ `await`
D) `pause()`

Section 2: Practical Questions

9. What will be logged in the console?

```
console.log("A");
setTimeout(() => console.log("B"), 0);
console.log("C");
```

A) A, B, C
~~B)~~ A, C, B
C) B, A, C
D) C, B, A

10. What will be the output?

```
function greet(callback) {
  console.log("Hello");
  callback();
}

function sayBye() {
  console.log("Goodbye");
}

greet(sayBye);
```

~~A)~~ Hello
Goodbye
B) Goodbye
Hello
C) Error
D) Undefined

11.
What will the following log?

```
let promise = new Promise((resolve, reject) => {
  resolve("Success");
});

promise.then(result => console.log(result));
console.log("Done");
```

~~A)~~ Done
Success
B) Success
Done
C) Error
D) Nothing

12.
What is the output of this code?

```
async function fetchData() {
  return "Data received";
}

fetchData().then(result => console.log(result));
```

~~A)~~ Data received
B) Promise { "Data received" }
C) Undefined
D) Error

13.
What happens if a Promise is rejected and no `.catch()` block is provided?
A) It silently fails
~~B)~~ It throws an unhandled rejection error
C) It retries automatically
D) It returns undefined

14.
What will be logged here?

```
async function test() {
  console.log("1");
```

```
  await console.log("2");
  console.log("3");
}
test();
console.log("4");
```

A) 1, 2, 3, 4
B) 1, 2, 4, 3
C) 4, 1, 2, 3
D) 2, 1, 4, 3

15.
What is the main difference between using callbacks and promises?
A) Promises make code synchronous
B) Promises improve readability and allow chaining
C) Callbacks are faster than Promises
D) Promises can only handle rejections

16.
What will be printed?

```
Promise.resolve(5)
  .then(x => x * 2)
  .then(x => console.log(x));
```

A) 5
B) 10
C) undefined
D) Promise

17.
Which function simulates asynchronous behavior in JavaScript?
A) `setTimeout()`
B) `parseInt()`
C) `Math.random()`
D) `alert()`

18.
What is the correct way to handle an error in a Promise chain?
A) `.fail()`
B) `.catch()`

C) `.reject()`
~~D)~~ `try...catch` outside

19.
What will happen here?

```
new Promise((resolve, reject) => {
  reject("Failed");
})
.then(res => console.log(res))
.catch(err => console.log(err));
```

A) Nothing
~~B)~~ Failed
C) Error
D) Undefined

20.
What is the purpose of `async/await`?
~~A)~~ To write asynchronous code that looks synchronous
B) To execute multiple functions in parallel
C) To delay synchronous code
D) To handle DOM updates