🧠 Complex Output-based Questions

setTimeout and Event Loop

```
setTimeout(() => console.log('1'), 100);
setTimeout(() => console.log('2'), 0);
console.log('3');
```

1.

```
console.log('A');
setTimeout(() => console.log('B'), 0);
setTimeout(() => console.log('C'), 100);
console.log('D');
```

2.

```
for (let i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), i * 100);
}
console.log('Done');
```

3.

```
for (var i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 0);
}
```

4.

```
setTimeout(() => console.log('X'), 0);
Promise.resolve().then(() => console.log('Y'));
console.log('Z');
```

5.

```
console.log('Start');
setTimeout(() => console.log('Middle'), 0);
Promise.resolve().then(() => console.log('End'));
```

6.

```
setTimeout(() => console.log('First'), 0);
setTimeout(() => {
  console.log('Second');
  setTimeout(() => console.log('Third'), 0);
}, 0);
```

7.

```
function delayLog(msg) {
  setTimeout(() => console.log(msg), 0);
}
delayLog('Hello');
console.log('World');
```

8.

```
setTimeout(() => console.log('A'), 0);
for (let i = 0; i < 1e6; i++) {}
console.log('B');
```

9.

```
setTimeout(() => console.log('A'), 10);
setTimeout(() => console.log('B'), 5);
setTimeout(() => console.log('C'), 0);
```

10.

Closures with setTimeout

```
for (let i = 0; i < 3; i++) {
  setTimeout(() => console.log(i), 100);
}
```

11.

```
for (var i = 0; i < 3; i++) {
  setTimeout((function(i) {
    return () => console.log(i);
  })(i), 100);
}
```

12.

```
function outer() {
  let count = 0;
  setTimeout(() => {
    count++;
    console.log(count);
  }, 0);
}
outer();
```

13.

```
function makeCounter() {
  let count = 0;
  return () => {
    count++;
```

```
    console.log(count);
  };
}
let counter = makeCounter();
setTimeout(counter, 100);
```

14.

```
function delayMessage(msg, delay) {
  setTimeout(() => console.log(msg), delay);
}
delayMessage('Hello', 0);
delayMessage('World', 10);
```

15.

Promises and setTimeout

```
console.log('Start');
setTimeout(() => console.log('Timeout'), 0);
Promise.resolve().then(() => console.log('Promise'));
console.log('End');
```

16.

```
setTimeout(() => console.log('A'), 0);
Promise.resolve().then(() => console.log('B'));
```

17.

```
new Promise((resolve) => {
  console.log('1');
  resolve();
}).then(() => console.log('2'));
console.log('3');
```

18.

```
setTimeout(() => console.log('A'), 0);
Promise.resolve().then(() => {
  setTimeout(() => console.log('B'), 0);
});
console.log('C');
```

19.

```
console.log('Start');
setTimeout(() => console.log('Timeout 1'), 0);
Promise.resolve().then(() => {
  console.log('Promise');
  setTimeout(() => console.log('Timeout 2'), 0);
```

```
});
console.log('End');
```

20.

Async/Await with setTimeout

```
async function foo() {
  console.log('Start');
  await new Promise((resolve) => setTimeout(resolve, 0));
  console.log('End');
}
foo();
console.log('Middle');
```

21.

```
async function fetchData() {
  console.log('Fetching');
  await new Promise((resolve) => setTimeout(resolve, 100));
  console.log('Done');
}
fetchData();
console.log('Loading');
```

22.

```
async function test() {
  console.log('1');
  await Promise.resolve();
  console.log('2');
}
test();
console.log('3');
```

23.

```
async function asyncTask() {
  console.log('Task 1');
  await new Promise((resolve) => setTimeout(resolve, 0));
  console.log('Task 2');
}
asyncTask();
console.log('Task 3');
```

24.

Callback Hell

```
setTimeout(() => {
  console.log('First');
```

```
  setTimeout(() => {
    console.log('Second');
    setTimeout(() => {
      console.log('Third');
    }, 100);
  }, 50);
}, 0);
```

25.

```
function step1(callback) {
  setTimeout(() => {
    console.log('Step 1');
    callback();
  }, 100);
}
function step2(callback) {
  setTimeout(() => {
    console.log('Step 2');
    callback();
  }, 50);
}
step1(() => step2(() => console.log('Done')));
```

26.

Mixed Scenarios

```
console.log('Start');
setTimeout(() => console.log('Timeout 1'), 0);
Promise.resolve().then(() => console.log('Promise 1'));
setTimeout(() => console.log('Timeout 2'), 0);
console.log('End');
```

27.

```
function delay() {
  return new Promise((resolve) => setTimeout(resolve, 100));
}
async function asyncFunc() {
  console.log('Start');
  await delay();
  console.log('After Delay');
}
asyncFunc();
console.log('End');
```

28.

```
console.log('A');
```

```
setTimeout(() => {
  console.log('B');
  Promise.resolve().then(() => console.log('C'));
}, 0);
console.log('D');
```

29.

```
function delayedPromise() {
  return new Promise((resolve) => setTimeout(() => resolve('Done'), 100));
}
delayedPromise().then((msg) => console.log(msg));
console.log('Loading');
```

30.

Here are **20 output-based questions** that combine **JavaScript objects** with **asynchronous concepts** like `setTimeout`, `setInterval`, promises, `async/await`, and closures. These questions are designed to challenge your understanding of asynchronous behavior when dealing with objects and their methods.

🧠 Mixed Questions: Objects & Asynchronous JS

```
const obj = {
  message: 'Hello',
  greet() {
    setTimeout(() => console.log(this.message), 0);
  },
};
obj.greet();
```

1.

```
const obj = {
  count: 0,
  increment() {
    setInterval(() => {
      this.count++;
      console.log(this.count);
    }, 100);
  },
};
obj.increment();
```

2.

```
const obj = {
  value: 42,
```

```
  delayedLog: function() {
    setTimeout(function() {
      console.log(this.value);
    }, 100);
  },
};
obj.delayedLog();
```

3.

```
const user = {
  name: 'Alice',
  sayHi() {
    setTimeout(() => console.log(`Hi, ${this.name}`), 0);
  },
};
user.sayHi();
```

4.

```
const car = {
  brand: 'Toyota',
  getBrand: function() {
    setTimeout(() => console.log(this.brand), 50);
  },
};
car.getBrand();
```

5.

```
const obj = {
  num: 1,
  increment() {
    setTimeout(() => {
      this.num++;
      console.log(this.num);
    }, 0);
  },
};
obj.increment();
console.log(obj.num);
```

6.

```
const person = {
  name: 'Bob',
  greet: function() {
    setTimeout(() => console.log(this.name), 0);
  },
```

```
};
const greet = person.greet;
greet();
```

7.

```
const obj = {
  count: 0,
  start() {
    setInterval(function() {
      this.count++;
      console.log(this.count);
    }, 100);
  },
};
obj.start();
```

8.

```
const obj = {
  data: 'Test',
  logData: function() {
    setTimeout(() => console.log(this.data), 0);
  },
};
obj.logData();
```

9.

```
const user = {
  name: 'John',
  sayName: function() {
    setTimeout(() => console.log(this.name), 100);
  },
};
user.sayName();
```

10.

```
const obj = {
  a: 10,
  b: 20,
  add() {
    return new Promise((resolve) => {
      setTimeout(() => resolve(this.a + this.b), 100);
    });
  },
};
obj.add().then((sum) => console.log(sum));
```

11.

```
const obj = {
  message: 'Hello',
  delayedPrint: function() {
    setTimeout(() => console.log(this.message), 50);
  },
};
const newObj = { message: 'Hi' };
obj.delayedPrint.call(newObj);
```

12.

```
const obj = {
  x: 5,
  y: 10,
  sum() {
    setTimeout(() => console.log(this.x + this.y), 0);
  },
};
obj.sum();
```

13.

```
const obj = {
  name: 'Alice',
  greet: async function() {
    await new Promise((resolve) => setTimeout(resolve, 100));
    console.log(this.name);
  },
};
obj.greet();
```

14.

```
const person = {
  firstName: 'John',
  lastName: 'Doe',
  getFullName: function() {
    setTimeout(() => console.log(`${this.firstName} ${this.lastName}`), 0);
  },
};
person.getFullName();
```

15.

```
const obj = {
  value: 0,
  increase() {
```

```
    setTimeout(() => {
      this.value++;
      console.log(this.value);
    }, 100);
  },
};
obj.increase();
```

16.

```
const obj = {
  number: 1,
  increment: function() {
    setTimeout(function() {
      console.log(this.number);
    }, 0);
  },
};
obj.increment();
```

17.

```
const counter = {
  count: 0,
  start() {
    setInterval(() => {
      this.count += 1;
      console.log(this.count);
    }, 200);
  },
};
counter.start();
```

18.

```
const obj = {
  data: 10,
  showData: function() {
    setTimeout(() => console.log(this.data), 0);
  },
};
obj.showData();
```

19.

```
const obj = {
  value: 5,
  delayedDouble() {
    setTimeout(() => {
```

```
      console.log(this.value * 2);
    }, 100);
  },
};
obj.delayedDouble();
```

20.