Questions

Javascript basics

1. Basics

Data Types:

1. **What are the different primitive data types in JavaScript?**
    - `string`, `number`, `bigint`, `boolean`, `undefined`, `null`, `symbol`
2. **What is the difference between `null` and `undefined`?**
    - `null` is an intentional absence of a value, while `undefined` means a variable has been declared but not assigned a value.
3. **How do JavaScript objects differ from arrays?**
    - Objects store key-value pairs, while arrays store ordered collections of elements with numerical indices.

Variables:

1. **What is the difference between `var`, `let`, and `const`?**
    - `var` is function-scoped and allows redeclaration.
    - `let` is block-scoped and cannot be redeclared.
    - `const` is also block-scoped but cannot be reassigned after initialization.
2. **What happens if you declare a variable without `var`, `let`, or `const`?**
    - It becomes a global variable, which can lead to unexpected behavior.

Operators:

1. **What is the difference between `==` and `===`?**
    - `==` checks for equality with type coercion, while `===` checks for strict equality (no type conversion).

2. **How does the `??` (nullish coalescing) operator work?**
    - It returns the right-hand value only if the left-hand value is `null` or `undefined`.

    ```
    let value = null ?? "default";
    console.log(value); // "default"
    ```

3. **How does the `||` (logical OR) operator work? How is it different from `??`?**
    - `||` returns the right-hand value if the left-hand value is **falsy** (e.g., `0`, `""`, `null`, `undefined`, `false`).
    - `??` only checks for `null` or `undefined`.

    ```
    console.log(0 || "fallback");  // "fallback"
    console.log(0 ?? "fallback");  // 0
    ```

---

2. Conditional Logic

if/else statements:

1. **How do `if` and `else` statements work? Can you provide an example?**
    - They execute different blocks of code based on conditions.

    ```
    let age = 18;
    if (age >= 18) {
        console.log("Adult");
    } else {
        console.log("Minor");
    }
    ```

2. **What happens if you don’t provide an `else` block in an `if` statement?**

- Nothing happens if the `if` condition is false. The program simply moves on.

switch statements:

1. **How does a `switch` statement work in JavaScript?**

- It evaluates an expression and matches it against `case` values.

```
let fruit = "apple";
switch (fruit) {
    case "banana":
        console.log("Yellow");
        break;
    case "apple":
        console.log("Red");
        break;
    default:
        console.log("Unknown");
}
```

1. **Why is the `break` statement important in a `switch` case? What happens if we omit it?**

- Without `break`, execution continues to the next case, causing unintended behavior.

Ternary operators:

1. **Rewrite this `if-else` statement using a ternary operator:**

```
let num = 5;
console.log(num > 0 ? "Positive" : "Negative or Zero");
```

1. **Can you use multiple ternary operators in a single expression?**

- Yes, but it can make the code harder to read.

```
let age = 20;
let category = age < 13 ? "Child" : age < 18 ? "Teenager" : "Adult";
console.log(category);
```

Using && and || in conditions:

1. **What will be the output of this code snippet? Why?**

```
console.log(true || false && false);
```

- `true`, because `&&` has higher precedence than `||`, so `false && false` evaluates to `false`, and then `true || false` results in `true`.

1. **How can `&&` be used as a shorthand for `if` statements?**

```
let isLoggedIn = true;
isLoggedIn && console.log("Welcome!");  // Shorthand for if (isLoggedIn) { co
nsole.log("Welcome!"); }
```

---

3. Loops & Control Flow

for loops:

1. **Write a `for` loop that prints numbers from 1 to 10.**

```
for (let i = 1; i <= 10; i++) {
    console.log(i);
}
```

1. **What happens if you don't provide an increment condition in a `for` loop?**

- It results in an infinite loop unless broken manually.

while loops:

1. **What is the difference between a `for` loop and a `while` loop?**

- `for` is used when the number of iterations is known, whereas `while` is used when the loop should run until a condition changes.

1. **Write a `while` loop that prints numbers from 10 to 1.**

```
let i = 10;
while (i >= 1) {
    console.log(i);
    i--;
}
```

for...of loops:

1. **What is the `for...of` loop used for? How is it different from `forEach`?**

- It iterates over iterable objects (arrays, strings, etc.). Unlike `forEach`, it can be `break`ed.

1. **Can `for...of` be used with objects? Why or why not?**

- No, because objects are not iterable directly. Use `Object.keys()` or `Object.entries()`.

for...in loops:

1. **What does a `for...in` loop iterate over?**

- It iterates over an object's enumerable properties.

1. **How would you use a `for...in` loop to iterate over an object's properties?**

```
let person = { name: "John", age: 30 };
for (let key in person) {
    console.log(key, person[key]);
}
```

break statements:

1. **How does the `break` statement work inside a loop? Provide an example.**

- It exits the loop immediately.

```
for (let i = 0; i < 5; i++) {
    if (i === 3) break;
    console.log(i);
}
```

1. **What will be the output of this code?**

```
0
1
2
```

continue statements:

1. **What is the purpose of the `continue` statement in loops?**

- It skips the current iteration and moves to the next.

1. **What will be the output of this code?**

```
for (let i = 1; i <= 5; i++) {
   if (i % 2 === 0) continue;
   console.log(i);
}
```

- Output:

  ```
  1
  3
  5
  ```

---

1. Conditional Logic (Advanced Questions)

1. Nested and Complex Conditions

Q1. What will be the output?

```
let a = 5, b = "5";
if (a == b && a === b || typeof a !== "number") {
   console.log("Condition met");
} else {
```

```
    console.log("Condition not met");
}
```

- **Output:** `"Condition not met"`
- **Explanation:**
  - `a == b` is `true` because `==` performs type coercion.
  - `a === b` is `false` because `===` checks strict equality.
  - `typeof a !== "number"` is `false` because `a` is a number.
  - The condition becomes `true && false || false`, which evaluates to `false`.

Q2. Convert nested `if-else` to a cleaner approach

```
let score = 85;
console.log(score >= 90 ? "Grade: A" : score >= 80 ? "Grade: B" : score >= 70 ? "Grade: C" : "Grade: F");
```

- **Explanation:** This replaces the nested `if-else` with a **ternary operator**, making the code more concise.

Q3. What will be the output?

```
let x = 10;
let y = 5;
let result = x > y ? x < 15 ? "A" : "B" : "C";
console.log(result);
```

- **Output:** `"A"`
- **Explanation:**
  - `x > y` is `true`, so we check `x < 15`, which is also `true`.
  - Therefore, `"A"` is assigned to `result`.

Q4. Find the largest of three numbers using a single-line ternary

```
let a = 10, b = 25, c = 15;
let largest = a > b ? (a > c ? a : c) : (b > c ? b : c);
console.log(largest);
```

Q5. Output of condition checking divisibility

```
let num = 7;
if (num % 2 === 0 && num % 3 === 0) {
  console.log("Divisible by both 2 and 3");
} else if (num % 2 === 0 || num % 3 === 0) {
  console.log("Divisible by 2 or 3, but not both");
} else {
  console.log("Not divisible by 2 or 3");
}
```

- `num = 9` → `"Divisible by 2 or 3, but not both"`
- `num = 12` → `"Divisible by both 2 and 3"`
- `num = 11` → `"Not divisible by 2 or 3"`

2. Loops (Advanced Questions)

6. Pattern Printing

Print triangle pattern

```
for (let i = 1; i <= 5; i++) {
  console.log("*".repeat(i));
}
```

Print inverted triangle

```
for (let i = 5; i >= 1; i--) {
    console.log("*".repeat(i));
}
```

8. Find the largest odd number in an array

```
let numbers = [12, 45, 23, 18, 91, 78];
let maxOdd = -Infinity;
for (let num of numbers) {
    if (num % 2 !== 0 && num > maxOdd) {
        maxOdd = num;
    }
}
console.log(maxOdd);
```

Output: `91`

9. Output of for-loop

```
for (let i = 1; i < 10; i *= 2) {
    console.log(i);
}
```

Output:

```
1
2
4
8
```

Explanation: `i` starts at `1` and multiplies by `2` in each iteration.

10. Output of while-loop

```
let i = 0;
while (i < 5) {
    console.log(i);
    i += 2;
}
```

Output:

```
0
2
4
```

11. Modify while-loop to print odd numbers

```
let i = 1;
while (i <= 10) {
    console.log(i);
    i += 2;
}
```

Output:

```
1
3
5
7
9
```

12. Remove duplicates from an array without using Set

```
let arr = [1, 2, 3, 2, 4, 1, 5];
let uniqueArr = [];
for (let num of arr) {
    if (!uniqueArr.includes(num)) {
        uniqueArr.push(num);
    }
}
console.log(uniqueArr);
```

Output: `[1, 2, 3, 4, 5]`

---

13. Count vowels in a string using `for...in` loop

```
let str = "hello world";
let vowels = "aeiou";
let count = 0;
for (let i in str) {
    if (vowels.includes(str[i])) count++;
}
console.log(count);
```

Output: `3`

---

14. Output of nested loops

```
for (let i = 1; i <= 3; i++) {
    for (let j = 1; j <= 3; j++) {
        if (i === j) break;
        console.log(i, j);
    }
}
```

Output:

```
2 1
3 1
3 2
```

Explanation: The `break` stops the inner loop when `i === j`.

15. Fibonacci sequence up to `n` terms

```
let n = 10, a = 0, b = 1;
console.log(a);
console.log(b);
for (let i = 2; i < n; i++) {
  let next = a + b;
  console.log(next);
  a = b;
  b = next;
}
```

Output: `0, 1, 1, 2, 3, 5, 8, 13, 21, 34`

Bonus Challenge Questions

16. Modify loop to skip multiples of 4

```
for (let i = 1; i <= 20; i++) {
  if (i % 4 === 0) continue;
  console.log(i);
}
```

Output: Skips `4, 8, 12, 16, 20`

17. FizzBuzz but skip numbers with digit '7'

```
for (let i = 1; i <= 50; i++) {
  if (i.toString().includes('7')) continue;
  if (i % 3 === 0 && i % 5 === 0) {
    console.log("FizzBuzz");
  } else if (i % 3 === 0) {
    console.log("Fizz");
    } else if (i % 5 === 0) {
        console.log("Buzz");
    } else {
        console.log(i);
    }
}
```

Explanation:

- **Skips** numbers like `7, 17, 27, 37, etc.`
- Prints `"Fizz"` for multiples of 3.
- Prints `"Buzz"` for multiples of 5.
- Prints `"FizzBuzz"` for multiples of both.
