

JavaScript Loops & Functions

Loops and functions are **fundamental building blocks** in JavaScript that help make code **repetitive** and **reusable** with minimal effort.

Types of Loops in JavaScript

Loop Type	Purpose
for	Repeat a block of code a fixed number of times
for...in	Loop through the keys of an object
for...of	Loop through the values of an iterable (like arrays or strings)
while	Loop while a given condition is true
do...while	Same as while but runs at least once before checking the condition


1 For Loop

◇ Syntax:


```
for (initialization; condition; update) {  
  // block of code  
}
```

 Explanation:

- **initialization**: Runs once before the loop starts.
- **condition**: Checked before each loop iteration.
- **update**: Runs after each loop iteration.

 Example:

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


 Output:

```
0  
1  
2
```

3
4

2 For...in Loop

Used to **iterate over object properties** (keys).

 Example:


```
const person = {  
  name: 'John',  
  age: 30,  
  job: 'Developer'  
};  
  
for (let key in person) {  
  console.log(`${key}: ${person[key]}`);  
}
```

 Output:


```
name: John  
age: 30  
job: Developer
```

3 For...of Loop

Used to **iterate over values** of an iterable like arrays or strings.

 Example:

```
const fruits = ['🍌', '🍌', '🍌'];  
  
for (let fruit of fruits) {  
  console.log(fruit);  
}
```

 Output:

🍌
🍌
🍌

 Works with strings too:


```
for (let char of "Hi") {  
  console.log(char);  
}
```

 Output:


```
H  
i
```

4 While Loop

Runs a block of code **as long as** the condition is **true**.

 Example:


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

 Output:

```
0  
1  
2  
3  
4
```

5 Do...While Loop

Runs the block of code **at least once**, then repeats **while the condition is true**.

 Example:

```
let i = 6;  
do {
```

```
    console.log(i);  
    i++;  
  } while (i < 5);
```

 Output:


6

☒ *Runs once even though the condition is initially false.*


Functions in JavaScript

Functions are **reusable blocks of code** designed to perform a particular task.

Function Declaration


 Example:

```
function greet(name) {  
  console.log(`Hello, ${name}!`);  
}  
  
greet("John");
```

 Output:

Hello, John!

Function Expression

 Example:

```
const greet = function(name) {  
  console.log(`Hello, ${name}!`);  
};  
  
greet("Sarah");
```


Arrow Function

More concise, does **not** have its own **this**.

 Example:

```
const greet = (name) => {  
  console.log(`Hello, ${name}!`);  
};  
  
greet("Alex");
```

Returning a Value BACK


 Example:

```
function add(a, b) {  
  return a + b;  
}  
  
const result = add(3, 4);  
console.log(result);
```

 Output:

7

Default Parameters

 Example:

```
function multiply(a, b = 1) {  
  return a * b;  
}  
  
console.log(multiply(5));    // 5  
console.log(multiply(5, 2)); // 10
```

Switch Statement in JavaScript

The **switch** statement is used to perform different actions based on different conditions (typically equality comparisons).

```
const month = "march";

switch (month) {
  case "jan":
    console.log("January");
    break; // stops further execution if matched
  case "feb":
    console.log("February");
    break;
  case "march":
    console.log("March"); // ☒ This will run
    break;
  case "april":
    console.log("April");
    break;
  default:
    console.log("No match found in cases.");
    break;
}
```

Key Points:

- **switch** checks each **case** strictly (**===**) against the expression (**month** here).
- **break** prevents fall-through to other cases.
- If **no case matches**, the **default** block runs.
- Useful alternative to multiple **if-else if** statements for equality checks.

Practice Questions

☒ Q1. Print student marks using **for...in** loop

```
const marks = {
  dp: 100,
  ap: 99,
  hp: 98
};

for (let student in marks) {
  console.log(`${student}: ${marks[student]}`);
}
```

Output:

```
dp: 100  
ap: 99  
hp: 98
```


☒ Q2. Repeat "Try Again" until correct number entered

```
let correctNumber = 7;  
let userNumber;  
  
do {  
  userNumber = parseInt(prompt('Enter a number:'));  
  if (userNumber !== correctNumber) {  
    console.log('Try again');  
  }  
} while (userNumber !== correctNumber);
```

 Uses *prompt* and *do...while* for input loop.

☒ Q3. Find the mean of 5 numbers

```
function findMean(a, b, c, d, e) {  
  return (a + b + c + d + e) / 5;  
}  
  
console.log("Mean:", findMean(10, 20, 30, 40, 50));
```

 Output:

```
Mean: 30
```

☒ Q4. Multiply a number by 2

```
function multiplyByTwo(num) {  
  return num * 2;  
}
```

☒ Q5. Reverse a string

```
function reverseString(str) {  
  return str.split('').reverse().join('');  
}  
  
console.log(reverseString("hello")); // olleh
```

☒ Q6. Print even numbers from 0 to **n**

```
function printEvenNumbers(n) {  
  for (let i = 0; i <= n; i++) {  
    if (i % 2 === 0) {  
      console.log(i);  
    }  
  }  
}
```

☒ Q7. Calculate factorial of a number

```
function calculateFactorial(num) {  
  if (num === 0 || num === 1) return 1;  
  let fact = 1;  
  for (let i = 2; i <= num; i++) {  
    fact *= i;  
  }  
  return fact;  
}
```

☒ Q8. Count occurrences of a number in an array

```
function countOccurrences(arr, target) {  
  let count = 0;  
  for (let num of arr) {  
    if (num === target) count++;  
  }  
  return count;  
}
```

? Q9. What is an arrow function?

☒ **Arrow functions** are a **short syntax** to write function expressions in JavaScript.


```
const add = (a, b) => a + b;
```

🧠 They:

- Are anonymous (no name)
 - Don't bind their own `this`
 - Great for short, inline logic
-