Control flow in JavaScript refers to the order in which statements are executed in a program. JavaScript provides various constructs for controlling the flow of execution, such as conditionals (if statements, switch statements), loops (for, while, do-while), and jumps (break, continue, return). I'll provide examples for each of these constructs.

1. **If Statement:**

The `if` statement is used for conditional execution. If a given condition evaluates to `true`, the block of code inside the `if` statement will be executed.

```javascript
let number = 10;
Let ages=[10,18,20,40]

if (number > 0) {
    console.log("The number is positive.");
} else {
    console.log("The number is non-positive.");
}
```

2. **Switch Statement:**

The `switch` statement is used to perform different actions based on different conditions. It's an alternative to a series of `if-else` statements.

```javascript
let day = "Monday";

switch (day) {
    case "Monday":
        console.log("It's the start of the week.");
        break;
    case "Friday":
        console.log("It's almost the weekend!");
        break;
    default:
        console.log("It's a regular day.");
}
```

3. **For Loop:**

The `for` loop is used to repeatedly execute a block of code a specific number of times.

```javascript
for (let i = 0; i < 5; i++) {
  console.log("Iteration:", i);
}
```

4. **While Loop:**

The `while` loop continues to execute a block of code as long as a specified condition is true.

```javascript
let count = 0;

while (count < 3) {
  console.log("Count:", count);
  count++;
}
```

5. **Do-While Loop:**

Similar to the `while` loop, the `do-while` loop executes a block of code at least once before checking the condition.

```javascript
let x = 5;

do {
  console.log("Value of x:", x);
  x--;
} while (x > 0);
```

6. **Break Statement:**

The `break` statement is used to terminate the execution of a loop or switch statement.

```javascript
for (let i = 0; i < 10; i++) {
  if (i === 5) {
    break;
  }
  console.log("Value of i:", i);
```

}

7. **Continue Statement:**

The `continue` statement is used to skip the rest of the code inside a loop for the current iteration and proceed to the next iteration.

```javascript
for (let i = 0; i < 5; i++) {
    if (i === 2) {
        continue;
    }
    console.log("Value of i:", i);
}
```

8. **Return Statement:**

The `return` statement is used to end the execution of a function and specifies a value to be returned to the caller.

```javascript
function addNumbers(a, b) {
    return a + b;
}

let result = addNumbers(3, 4);
console.log("Sum:", result);
```

These examples cover the basic control flow constructs in JavaScript.