# Looping Statements

Looping statements in JavaScript allow you to execute a block of code repeatedly. There are several types of loops available in JavaScript:

## 1. for Loop:

The for loop is used when you know the number of iterations beforehand.

### Example:

```
for (let i = 0; i < 5; i++) {
   // Code to be executed in each iteration
   console.log(i);
}
```

## 2. while Loop:

The while loop is used when the number of iterations is not known in advance.

### Example:

```
let i = 0;

while (i < 5) {
   // Code to be executed in each iteration
   console.log(i);
```

```
    i++;
}
```

## 3. do-while Loop:

Similar to the while loop, but the code block is executed at least once, even if the condition is false.

## Example:

```
let i = 0;


do {
    // Code to be executed in each iteration
    console.log(i);
    i++;
} while (i < 5);
```

## 4. for...in Loop:

Used to iterate over the properties of an object.

## Example:

```
const obj = { a: 1, b: 2, c: 3 };


for (let key in obj) {
    // Code to be executed in each iteration
    console.log(key, obj[key]);
}
```

## 5. for...of Loop:

Introduced in ES6, used to iterate over iterable objects (arrays, strings, etc.).

### Example:

```
const array = [1, 2, 3];


for (let element of array) {
    // Code to be executed in each iteration
    console.log(element);
}
```

## 6. forEach Method:

Available for arrays, it executes a provided function once for each array element.

### Example:

```
const array = [1, 2, 3];
array.forEach(function(element) {
    // Code to be executed in each iteration
    console.log(element);
});
```

Choose the loop that best fits your use case based on whether you know the number of iterations beforehand and the type of data you are working with. Each loop has its own strengths and use cases.