# Functions in JavaScript

Functions are reusable blocks of code designed to perform specific tasks. They improve code organization, reusability, and readability.

---

## Definition

A function is defined using the `function` keyword or as an arrow function ( `=>` in ES6).

**Syntax (Regular Function):**

```
function functionName(parameters) {
    // Function body
    return value; // Optional
}
```

**Example:**

```
function greet(name) {
    return "Hello, " + name + "!";
}
console.log(greet("Alice")); // "Hello, Alice!"
```

**Arrow Function:**

```
const greet = (name) => "Hello, " + name + "!";
console.log(greet("Bob")); // "Hello, Bob!"
```

---

## Parameters

Parameters are variables listed as part of the function definition.

**Default Parameters:** You can set default values for parameters.

```
function greet(name = "Guest") {
    return "Hello, " + name + "!";
}
console.log(greet()); // "Hello, Guest!"
```

**Rest Parameters:** The `...` syntax collects multiple arguments into an array.

```
function sum(...numbers) {
    return numbers.reduce((total, num) => total + num, 0);
}
console.log(sum(1, 2, 3, 4)); // 10
```

---

### Return

The `return` statement specifies the value a function should output.

**Example:**

```
function square(num) {
    return num * num;
}
console.log(square(5)); // 25
```

---

### Closure

A **closure** is a function that retains access to its outer scope even after the outer function has executed. Closures are useful for creating private variables or functions.

**Example:**

```
function counter() {
    let count = 0;
    return function () {
        count++;
        return count;
    };
}

const increment = counter();
console.log(increment()); // 1
console.log(increment()); // 2
```

## Strings

Strings are sequences of characters. JavaScript provides many methods for string manipulation.

### Common String Methods

| Method | Description | Example | Output |
|---|---|---|---|
| `charAt(index)` | Returns the character at the specified index | `"hello".charAt(1)` | `"e"` |
| `concat()` | Concatenates strings | `"Hello".concat(" ", "World")` | `"Hello World"` |
| `includes()` | Checks if a string contains a substring | `"hello".includes("he")` | `true` |
| `indexOf()` | Returns the index of the first occurrence | `"hello".indexOf("l")` | `2` |
| `toLowerCase()` | Converts to lowercase | `"HELLO".toLowerCase()` | `"hello"` |
| `toUpperCase()` | Converts to uppercase | `"hello".toUpperCase()` | `"HELLO"` |
| `trim()` | Removes whitespace | `" hello ".trim()` | `"hello"` |
| `split()` | Splits a string into an array | `"a,b,c".split(",")` | `["a", "b", "c"]` |
| `slice(start, end)` | Extracts part of a string | `"hello".slice(1, 4)` | `"ell"` |
| `replace()` | Replaces part of the string | `"hello".replace("l", "r")` | `"herlo"` |

## Arrays

An array is a list-like object used to store multiple values.

---

**Common Array Methods**

| Method | Description | Example | Output |
|---|---|---|---|
| `push()` | Adds an element to the end | `[1, 2].push(3)` | `[1, 2, 3]` |
| `pop()` | Removes the last element | `[1, 2, 3].pop()` | `[1, 2]` |
| `shift()` | Removes the first element | `[1, 2, 3].shift()` | `[2, 3]` |
| `unshift()` | Adds an element to the beginning | `[1, 2].unshift(0)` | `[0, 1, 2]` |
| `concat()` | Combines arrays | `[1, 2].concat([3, 4])` | `[1, 2, 3, 4]` |
| `slice()` | Returns a portion of the array | `[1, 2, 3].slice(0, 2)` | `[1, 2]` |
| `splice()` | Removes or replaces elements | `[1, 2, 3].splice(1, 1)` | `[1, 3]` |
| `indexOf()` | Finds the first index of an element | `[1, 2, 3].indexOf(2)` | `1` |
| `includes()` | Checks if an element exists | `[1, 2, 3].includes(2)` | `true` |
| `join()` | Combines elements into a string | `[1, 2, 3].join("-")` | `"1-2-3"` |
| `reverse()` | Reverses the array | `[1, 2, 3].reverse()` | `[3, 2, 1]` |
| `sort()` | Sorts the array | `[3, 1, 2].sort()` | `[1, 2, 3]` |

---

**Search in Arrays**

1. **`indexOf`**

```javascript
const arr = [10, 20, 30];
console.log(arr.indexOf(20)); // 1
```

2. **`find`** Returns the first element that matches a condition.

```javascript
const arr = [10, 20, 30];
console.log(arr.find((x) => x > 15)); // 20
```

3. **`findIndex`**

```javascript
console.log(arr.findIndex((x) => x > 15)); // 1
```

---

### Sorting in Arrays

```javascript
let numbers = [40, 10, 30, 20];
numbers.sort((a, b) => a - b); // Ascending
console.log(numbers); // [10, 20, 30, 40]
```

---

### Iteration in Arrays

1. **`forEach()`**

```javascript
[1, 2, 3].forEach((x) => console.log(x));
```

2. **`map()`**

```javascript
let doubled = [1, 2, 3].map((x) => x * 2);
console.log(doubled); // [2, 4, 6]
```

3. **`filter()`**

```javascript
let evens = [1, 2, 3, 4].filter((x) => x % 2 === 0);
console.log(evens); // [2, 4]
```

---

## Objects

Objects are key-value pairs and are used to store structured data.

---

### Object Properties

Properties are key-value pairs where the key is a string (or symbol) and the value can be any type.

**Example:**

```
let person = {
    name: "John",
    age: 30,
    isEmployed: true,
};
console.log(person.name); // "John"
```

---

### Object Methods

Methods are functions defined as object properties.

**Example:**

```
let person = {
    name: "John",
    greet: function() {
        return "Hello, " + this.name;
    },
};
console.log(person.greet()); // "Hello, John"
```

---

### Object Iteration

1. **`for...in`**

   ```
   let person = { name: "John", age: 30 };
   for (let key in person) {
   ```

```
    console.log(key, person[key]);
}
```

2. **`Object.keys()`**

   ```
   console.log(Object.keys(person)); // ["name", "age"]
   ```

3. **`Object.values()`**

   ```
   console.log(Object.values(person)); // ["John", 30]
   ```