- **Loops**: `while`, `do...while`, `for`
- **Control Flow**: `break` and `continue`
- **Functions**: declaration, expression, arrow functions, parameters, and return values
- **Arrays and Objects**: creation, access, array methods, and object manipulation

---

Loops in JavaScript

1. `while` Loop

Executes code **as long as a condition is true**.

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

- **Use when the number of iterations is not known beforehand.**

---

2. `do...while` Loop

Executes code **at least once**, and then repeats **while condition is true**.

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

- **Guarantees at least one execution.**

---

3. `for` Loop

Best for situations when the number of iterations is **known**.

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

- Syntax: `for(initialization; condition; increment)`

---

break and continue

`break`

- **Exits** the loop entirely.

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break;
  console.log(i);
}
```

`continue`

- **Skips** the current iteration and moves to the next one.

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

Functions

1. Function Declaration

```
function greet(name) {
  return `Hello, ${name}`;
}
```

.

---

2. Function Expression

```
const greet = function(name) {
  return `Hello, ${name}`;
};
```

- Not hoisted.
- Stored in a variable.

---

3. Arrow Functions

```
const greet = (name) => `Hello, ${name}`;
```

- Shorter syntax.
- Does **not have its own `this`** (inherits from parent scope).

---

4. Parameters and Return Values

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

const result = add(5, 3); // 8
```

- Functions can take any number of parameters.
- `return` gives a value back to the caller.

Arrays and Objects

1. Creating and Accessing Arrays

```
const fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // apple
```

---

2. Creating and Accessing Objects

```
const person = {
  name: "Alice",
  age: 25
};
console.log(person.name); // Alice
```

- Access using **dot** or **bracket** notation: `person["name"]`

Common Array Methods

Method	Description	Example
`push()`	Add to end	`arr.push(4)`
`pop()`	Remove last	`arr.pop()`
`shift()`	Remove first	`arr.shift()`
`unshift()`	Add to start	`arr.unshift(0)`
`slice(start, end)`	Copy part of array	`arr.slice(1, 3)`

Method	Description	Example
`splice(start, deleteCount, ...items)`	Remove/replace/add items	`arr.splice(1, 1, "new")`
`map()`	Transform each element	`arr.map(x => x * 2)`
`filter()`	Return items that match a condition	`arr.filter(x => x > 5)`
`reduce()`	Accumulate values into one	`arr.reduce((sum, x) => sum + x, 0)`

Object Manipulation

Add/Update Property

```
person.job = "developer";
```

Delete Property

```
delete person.age;
```

Loop Through Properties

```
for (let key in person) {
  console.log(key, person[key]);
}
```

Object Methods

```
const user = {
  name: "Bob",
  greet() {
    console.log(`Hi, I’m ${this.name}`);
  }
};
user.greet(); // Hi, I’m Bob
```
