Arrow functions in JS

Arrow Functions (() => {}) in javascript

Sure! Let's learn **Arrow Functions (() => {}) in JavaScript** in a simple and detailed way.

Arrow Functions in JavaScript

Arrow functions were introduced in **ES6 (ECMAScript 2015)** as a shorter and cleaner way to write functions.

Basic Syntax

```javascript
const functionName = () => {
  // code
};
```

---

1. Traditional Function vs Arrow Function

Traditional Function

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

Arrow Function

```javascript
const add = (a, b) => {
  return a + b;
};
```

Short form (Implicit Return)

If there is only one expression, you can skip `{}` and `return`:

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

2. Arrow Functions with Single Parameter

When passing **one argument**, parentheses are optional:

```javascript
const square = x => x * x;
```

With multiple parameters, parentheses are required:

```javascript
const multiply = (a, b) => a * b;
```

---

3. Arrow Functions with No Parameters

Use empty parentheses:

```javascript
const greet = () => console.log("Hello!");
```

---

4. Arrow Functions Returning Object

Use parentheses around the object:

```javascript
const user = () => ({ name: "Omkar", role: "Developer" });
```

Without parentheses, JS thinks **`{}`** is a function block.

---

5. Arrow Functions and `this` Keyword

Arrow functions **do not have their own `this`**. They use the **`this`** value from the surrounding (lexical) scope.

Traditional function

```javascript
function person() {
  this.name = "Omkar";
  setTimeout(function () {
    console.log(this.name);  // undefined (because 'this' refers to window/global)
  }, 1000);
}
person();
```

Arrow function

```
javascript

function person() {
  this.name = "Omkar";
  setTimeout(() => {
    console.log(this.name);  // Omkar (inherits from parent scope)
  }, 1000);
}
person();
```

When to Use Arrow Functions

Use Case	Recommended
Short functions	✔️ Yes
Callbacks (setTimeout, forEach, event listeners)	✔️ Yes
Need lexical **`this`** binding	✔️ Yes
Object methods needing **`this`**	❌ Avoid
Constructor function	❌ Cannot use

Arrow Function Limitations

❌ Cannot be used as constructors (no **`new`**)
❌ No own **`this`**, **`arguments`**, **`super`**, or **`new.target`**
❌ Not suitable for object methods

Example to avoid:

```
javascript

const user = {
  name: "Omkar",
  getName: () => this.name  // returns undefined
};
```

Quick Examples

```
javascript

// map example
const numbers = [1, 2, 3];
const doubled = numbers.map(num => num * 2);
console.log(doubled); // [2, 4, 6]
```

Summary

Feature	Arrow Functions
Short & cleaner syntax	✔️
Implicit return	✔️
Lexical `this`	✔️
No own `this` or arguments	✔️
Can be used as constructor	❌

If you want, I can now teach:

📌 Arrow Functions vs Normal Functions deep comparison

📌 Practice questions & mini exercises

📌 Scope & `this` keyword advanced examples

Would you like **examples and interview questions** next? 😊