

# **Function VS Arrow Function**

**@coding\_comics**

# Syntax

- **Function:** Uses the function keyword.
- **Arrow Function:** Uses the => arrow syntax.



```
1 function greet(name) {  
2   return `Hello, ${name}!`;  
3 }
```



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

# Hoisting

- **Function:** Functions are hoisted, meaning they can be called before they are defined.
- **Arrow Function:** Not hoisted; must be defined before being called.



```
1 console.log(greet("Ajith")); // Works even before definition  
2  
3 function greet(name) {  
4   return `Hello, ${name}!`;  
5 }
```



```
1 console.log(greet("John")); // Error: greet is not defined  
2  
3 const greet = (name) => `Hello, ${name}!`;
```

# this Binding

- **Function:** Has its own **this**, determined by how the function is called.

```
1 function Person() {
2   this.name = "John";
3
4   setTimeout(function () {
5     console.log(this.name); // Undefined (depends on how it's called)
6   }, 1000);
7 }
8 new Person();
```

- **Arrow Function:** Inherits **this** from the surrounding context (lexical **this**).

```
1 function Person() {
2   this.name = "John";
3
4   setTimeout(() => {
5     console.log(this.name); // John (inherits from the enclosing scope)
6   }, 1000);
7 }
8 new Person();
```


# Arguments Object

- **Function:** Has access to the **arguments** object, an array-like object holding all passed arguments.



```
1 function showArgs() {  
2   console.log(arguments); // Logs all arguments  
3 }  
4 showArgs(1, 2, 3); // Output: [1, 2, 3]
```

- **Arrow Function:** Does not have its own **arguments** object. Use rest parameters instead.



```
1 const showArgs = (...args) => {  
2   console.log(args); // Use rest parameters  
3 };  
4 showArgs(1, 2, 3); // Output: [1, 2, 3]
```

# Return Statement

- **Function:** Requires **return** for returning values (unless using shorthand).



```
1 function add(a, b) {  
2   return a + b; // Explicit return  
3 }  
4 console.log(add(5, 3)); // Output: 8
```

- **Arrow Function:** Allows implicit return for single-line expressions.



```
1 const add = (a, b) => a + b; // Implicit return  
2 console.log(add(5, 3)); // Output: 8
```



## Use Case

- **Function:** Use for defining methods, event handlers, or complex functions.

```
1 // Function for defining methods
2 const person = {
3   name: "John",
4   greet: function () {
5     console.log(`Hello, ${this.name}!`); // Works well
6   },
7 };
```

- **Arrow Function:** Use for callbacks, array methods (map, filter, reduce), or inline logic.

```
1 // Arrow function for callbacks
2 const numbers = [1, 2, 3, 4];
3 const squares = numbers.map((n) => n * n);
4 console.log(squares); // Output: [1, 4, 9, 16]
```