


Day 8, Today's Topic

Arrow Functions

DAILY
JS



```
/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */
```

Day 8: Arrow Functions

Introduced in ES6, arrow function allows us to write functions in a much shorter syntax.

If you started learning JavaScript after ES6, you must have come across arrow functions quite a lot, and they might have confused you some times. Fear not, today, let's clear all doubts!

If you are coming from any other language to JS, the syntax of arrow function might look a little new, but once you understand it, I am sure, you will never go back to the plain old `functions` (unless necessary).

Syntax wise it's easier to understand, remove the `function` keyword, declare the function like a variable, and after arguments, put a fat arrow.

Syntax won't scare you, use cases will. For example, one liner functions, no need for parenthesis in case of single argument, and the use of `this`. Let's discuss them in next sections.

```
/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */
```

```
/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */
```

Syntax

As discussed above, remove the function keyword, declare the function as a variable (const/let/var), and put a fat arrow after the arguments.

****Normal Function****

```
```js
function myFunc (arg1, arg2, arg3) {
 ...
}
```
```

****Arrow Function****

```
```js
let myFunc = (arg1, arg2, arg3) => {
 ...
}
```
```

```
/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */
```

```

/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */

/**
 * Write a function which takes in 2 numbers as arguments
 * and returns their sum
 */

const a = 10;
const b = 40;

// Normal Function
function sumTwoNums1 (num1, num2) {
    return num1+num2;
}
console.log (`Normal: ${a} + ${b} = ${sumTwoNums1(a, b)}`);

// Arrow Function
const sumTwoNums2 = (num1, num2) => {
    return num1+num2;
}
console.log (`Arrow: ${a} + ${b} = ${sumTwoNums2(a, b)}`);

// One Liner
const sumTwoNums3 = (num1, num2) => num1+num2;
console.log (`One Liner: ${a} + ${b} = ${sumTwoNums3(a, b)}`);

/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */

```

```

/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */

/**
 * Sum of elements in an array
 */

const numbers = [1, 2, 3, 4, 5];


// Using Normal Function
function addAllElements1 (arr) {
  let sum = 0;
  arr.forEach (function (number) {
    sum+=number;
  });
  return sum;
}
console.log (`Normal: Sum of elements of ${numbers} = ${addAllElements1(numbers)}`);

// Using Arrow Function
const addAllElements2 = (arr) => {
  let sum = 0;
  arr.forEach ((num) => {
    sum += num;
  })
  return sum;
}
console.log (`Normal: Sum of elements of ${numbers} = ${addAllElements2(numbers)}`);

// One Liner
const addAllElements3 = arr => arr.reduce ((sum, num) => sum+num);
console.log (`Normal: Sum of elements of ${numbers} = ${addAllElements3(numbers)}`);

/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */

```

```
/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */
```

The one liners

Even though arrow functions are more concise than normal functions, they can still be reduced.

If the arrow function has only one statement inside it, it can be reduced further into a one liner.

****Example****

```
```js
const add = (a, b) ⇒ a+b;
```
```

One liners follow the concept of implicit return, the single statement that is written after the fat arrow (without curly braces), is performed and the result is returned.

```
/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */
```



```
/* ===== */
/* ===== Daily JS - Day 8 ===== */
/* ===== */
```


Single Argument Benefit

We can reduce the arrow function even further. If there is only one argument, there is no need for parenthesis.

****Example****

```
```js
 const square = a => a*a;
```
```

```
/* ===== madhavbahl.tech/dailyjs/day8 ===== */
/* ===== Join Discord: madhavbahl.tech/discord-c2e ===== */
```



```
/* ===== */  
/* ===== Daily JS - Day 8 ===== */  
/* ===== */
```

What about `this`?

Handling of `this` is different in arrow functions as compared to normal functions.

In very simple words, in arrow functions there is no binding of `this`.

In regular functions, `this` represents the object that calls the function.

But, in arrow functions, `this` keyword always represents the object that defines the arrow function.

```
/* ===== madhavbahl.tech/dailyjs/day8 ===== */  
/* ==== Join Discord: madhavbahl.tech/discord-c2e ==== */
```




To read the full content visit,

<https://github.com/MadhavBahlMD/dailyjs>

**DAILY
JS**