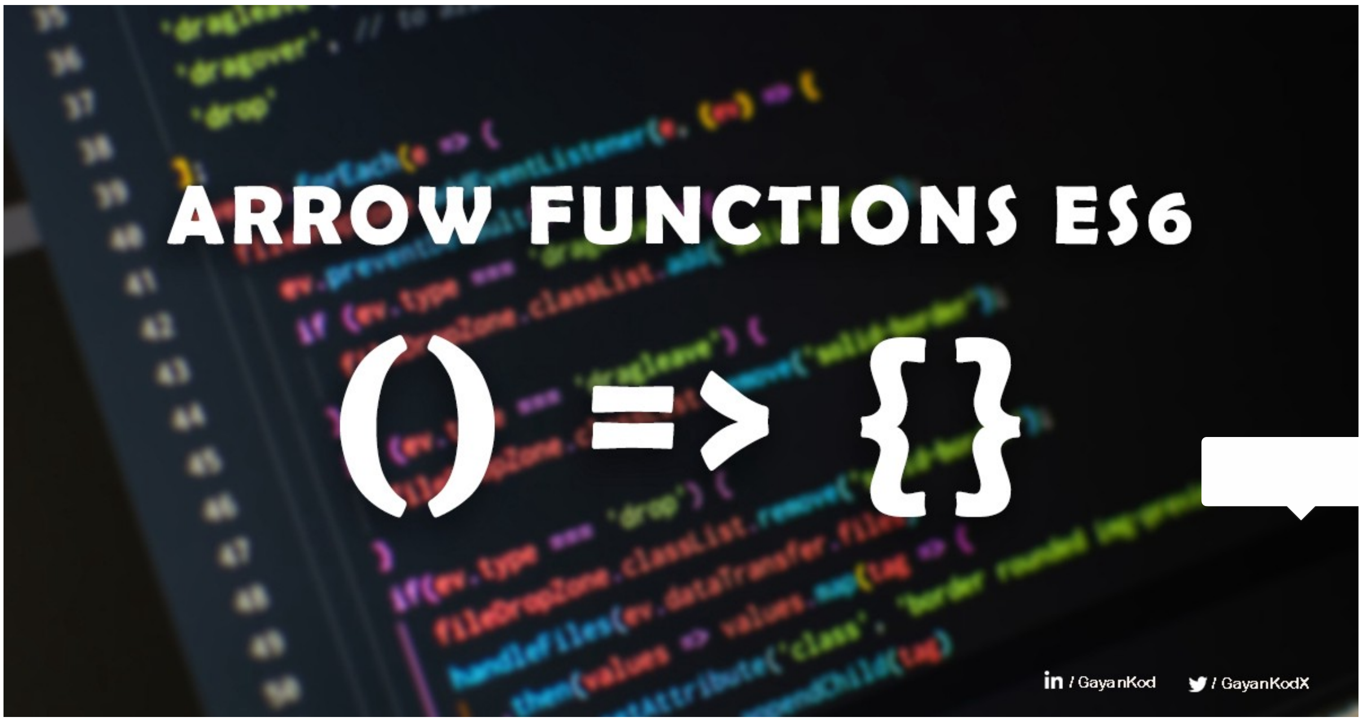
Published in Nerd For Tech

Gayan Kodithuwakku Follow
Mar 23 · 6 min read · Listen

Understanding Arrow Functions in JavaScript ES6 — Full Guide



JavaScript is the language that powers the web. The web is certainly dominated by it. You may be wondering what is ES6 means? You maybe have seen ES4, ES5, ES6… terms.
Javascript wasn't even always called Javascript. In fact, when Brendan first created it, it was actually called LiveScript, and then the people at Microsoft decided to try and reverse engineer the program and they ended up with something called Jscript. And so there were all the slightly different versions of Javascript that were being run on the web and it was starting to get quite confusing.
So the Europeans did what Europeans do best and they decided to standardize the language. And that's where you get the ECMAscript from. And that stands for the European Computer Manufacturers Association Script. And the only reason why this is interesting is that often you'll see different versions of Javascript not referred to as JS5 or JS6 but as ES6 or ES7, and the ES comes of course from ECMAscript.
So Arrow functions were introduced in the ES6 version of JS.

This is quite easy when you understand what ... here. But if you don't know what is the concept behind that Arrow functions. I'm sure you will already be confused what's the meaning of that two braces, the arrow, and the two curly brackets.

Simply Arrow functions is also a normal function. But it was simplified to reduce lines of code.

Steps of Simple Function → Arrow Function

```
function myFunction(a, b){
  return a * b;
}
```

So, this is a normal javascript function that returns the product of the two parameters a and b.

but we can write this function without the function name. That calls anonymous functions in JS.

```
function (a, b){
  return a * b;
}
```

Now, you may be wondering without a function name, how we call that function, without calling the function what is the use of that function. Okay, I agree but these anonymous functions don't write for calling purposes.

For example, assume that we have a button to get the product of two numbers. To do that we have to write a function for onClick. So directly we can write like this with an anonymous function.

```
<button onClick={function(a,b){
     return a*b;
}}>
   Get Product
</button>
```

with the clarification of that, Now let's see the next step.

We can remove the 'function' keyword as well. let's remove the function keyword.

```
(a, b){          //This will give an error
  return a * b;
}
```

```
(a, b) => {
  return a * b;
}

//let's write it in one line
(a, b) => {return a * b;}
```

As you can see, this is the basically Arrow function in JS. Arrow function is also a function that is simplified.
Just remove the function name and function keyword. Instead of the function keyword, we need to put an arrow.
Hope you now got the idea about Arrow functions in JavaScript!

So, now look at the previous example. We can simply write that as the following.

```
<button onClick={(a,b)=>{return a*b;}}>
   Get Product
</button>
```

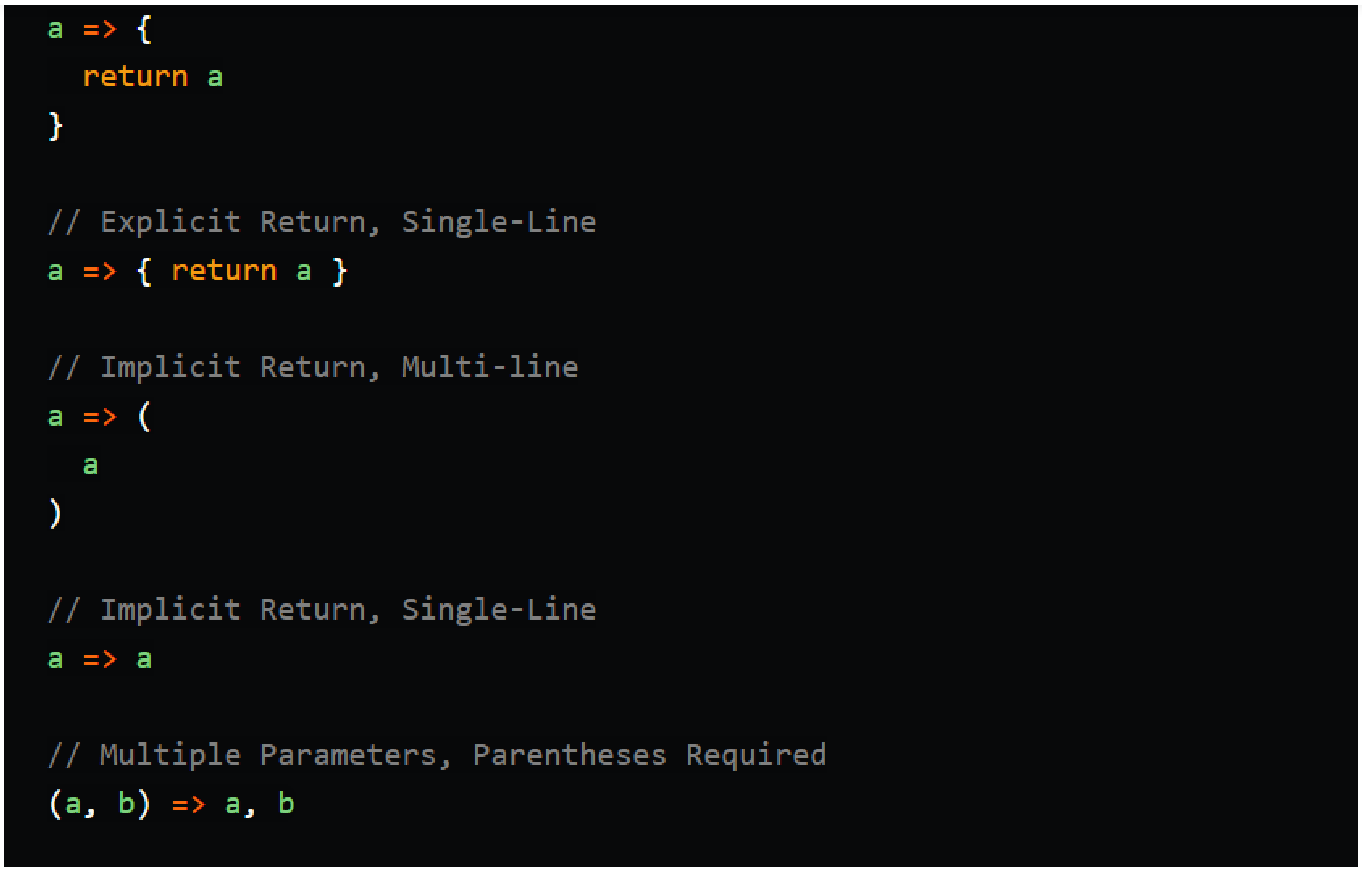
Wait, another important fact!
If your function only has one line, you do not need to wrap that line with curly braces. In this case, no need to wrap return a*b with curly braces and you do not need to write return keywords as well. The compiler knows that it should be returned.

```
<button onClick={(a,b)=> a*b}>
   Get Product
</button>
```

So simply we can write our Arrow function like this. Now you can get some idea of why arrow functions are used in JS. It reduces a lot of lines of code when developing a complex system.

Likewise, there are a lot of ways to write arrow functions in JavaScript. Now let’s quickly get informed on the ways you can write an Arrow Function.

Cheat Sheet of Arrow Functions in JS.



```
a => {
  return a
}

// Explicit Return, Single-Line
a => { return a }

// Implicit Return, Multi-line
a => (
  a
)

// Implicit Return, Single-Line
a => a

// Multiple Parameters, Parentheses Required
(a, b) => a, b
```

Cheat Sheet — Arrow Functions ES6

This is the cheat sheet of the Arrow Functions ES6.
Maybe when you see this, you feel like what the heck is this, like that… :D Don't worry I'll explain everything in this cheat sheet. let's gooo…

Implicit vs Explicit Return

First, you need to understand what is Implicit Return and Explicit Return.

With normal functions, if you want to return something, you have to use the `return` keyword. Arrow functions also have that. When you use the `return` keyword, it's called an **explicit return**.
However, the arrow functions allow something called **implicit return** where the `return` keyword can be skipped. Let's look at some examples.

Example A: Normal Function

```
const sayHi = function(name) {
  return name
}
```

Example B: Arrow Function with Explicit Return

```
    return name
  }

  // Single-line
  const sayHi = (name) => { return name }
```

Example C: Arrow Function with Implicit Return

```
  // Single-line
  const sayHi = (name) => name

  // Multi-line
  const sayHi = (name) => (
    name
  )
```

Notice the difference? When you use curly braces `{}` , you need to explicitly state the return. However, when you don't use curly braces, the `return` is implied and you don't need it.

There's actually a name for this.

When you use curly braces like in *Example B*, it's called a **Block Body.**
And the syntax in *Example C* is called a **Concise Body.**

Simply,

- Block body is where you use curly braces and have an explicit `return` .
- The concise body is where you don't use curly braces, and you skip the `return` keyword.

Alright, another thing about Parentheses () of Arrow functions.
First of all, parentheses mean two braces that we put in functions to wrap parameters. Sometimes it will be empty.

So, In normal functions we always need parentheses. But **in Arrow Functions we do not need parentheses if there is only one parameter.**

— Parentheses are optional for a SINGLE parameter —

```
const numbers = function(one) {}

// Arrow Function, with parentheses
const numbers = (one) => {}

// Arrow Function, without parentheses
const numbers = one => {}
```

— Parentheses are required for MULTIPLE parameters —

```
// Normal Function
const numbers = function(one, two) {}

// Arrow Function, with parentheses
const numbers = (one, two) => {}
```

IMPORTANT — Returning Objects

However, when you return an object in Arrow functions, you should put parentheses even you have only one line in the function.

```
const me = () => { name: "Gayan" };

me(); //Output --> undefined
```

The above one is wrong ❌

```
const me = () => ({ name: "Gayan" });

me(); //Output --> { name: "Gayan" }
```

This is correct ✅

That's all about it. I hope you got the idea about Arrow Functions ES6 and find this post useful, and I would love to see your feedback about the article. Or if you have any questions please put them all in the comment section, I will reply to you all.