**How To Use JavaScript Arrow Functions -** [Sebastian Eschweiler](https://medium.com/@s_eschweiler?source=post_page-----7ea724e3e888--------------------------------) Aug 1, 2022

Arrow functions are a way of writing anonymous function expressions in JavaScript which has been introduced in the 2015 version of the ECMAScript language specification (ES6). This article gives you a quick overview of how you can use JavaScript arrow functions which ease.

[**Find online courses to boost your web dev skills on Udemy (\*)**](https://click.linksynergy.com/deeplink?id=a1o1REVAqJg&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourses%2Fdevelopment%2Fweb-development%2F)

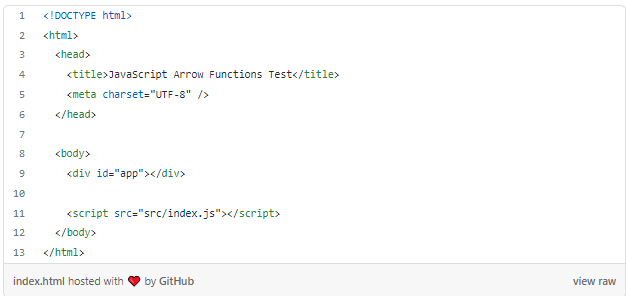
Being able to write anonymous function expressions could streamline and shorten your code a lot. Many modern programming languages have introduced this possibility recently, like Python did when introducing Lambda functions.

In JavaScript the concept of writing anonymous function expression is called Arrow functions. This concept differs from the traditional way of writing functions in JavaScript. Let’s explore this concept …

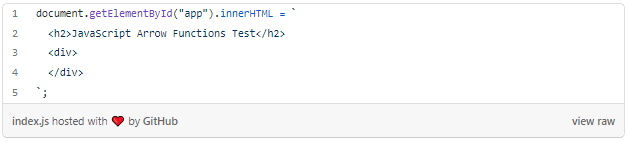
**From Traditional JavaScript Functions To Arrow Functions**

Let’s start with a simple JavaScript project to explore how arrow functions differ from traditional functions.

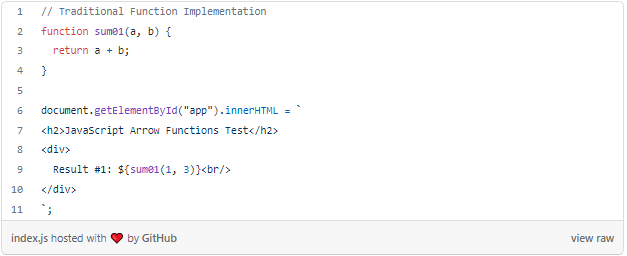
Create an empty project folder and inside this folder create a new file *index.html* and insert the following HTML code:



This points to the JavaScript file src/index.js, so let’s create this file as well in the project folder and insert the following code:



Let’s add a simple function which is just adding to values:



*sum02* is performing the same task as function *sum01* but is defined in an anonymous way. No function name is specified, instead a reference to the function is assigned to const *sum02*. If you’re defining the function in this anonymous way you can only call this function after it has been assigned to the constant.

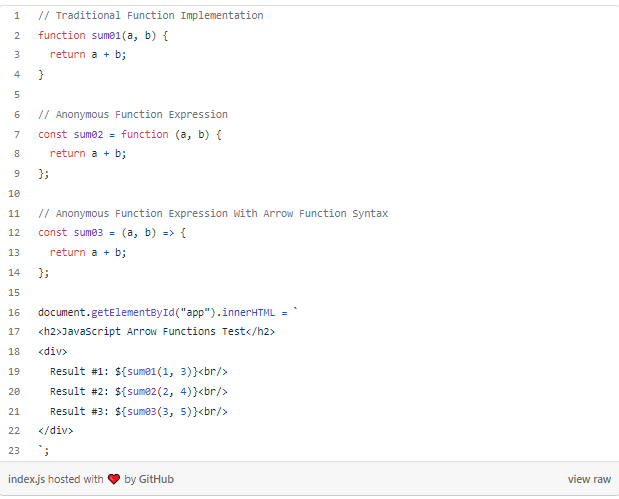
The result should then look like what you can see in the following screenshot:

Graphical user interface, text, application

Description automatically generated

Output of the result of the sum01 and sum02 function call

Now let’s take it one step further and use JavaScript’s arrow function syntax to define a third variant of the sum function:



To define the anonymous array function, we do not use the *function* keyword anymore. Instead the definition is done by just listing the parameters followed by the fat arrow symbol (=>) followed by the function body.

The output should then look like the following:

Graphical user interface, text, application

Description automatically generated

Output of the result of the sum01, sum02, and sum03 function call

**Arrow Function With Just One Parameter**

By implementing sum03 you’ve seen an example of an arrow function with multiple parameter. If the function only requires one parameter the syntax can be further shortened, parenthesis can be omitted.

const square = x => {  
 return x \* x  
}

**Arrow Function With No Parameter**

However if your arrow function is not requiring any parameters, you need to use parenthesis:

const helloWorld = () => {  
 return 'Hello!'  
}

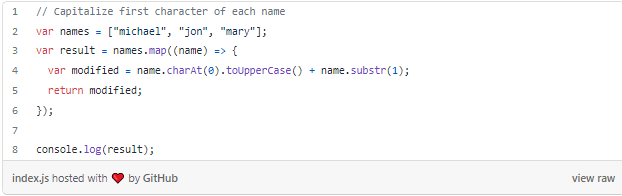
**Arrow Functions With Implicit Return**

If your arrow function body just consists of a return statement you can further reduce the syntax and make use of implicit return like you can see in the following:

const square = x => x \* x

**Use Case: Arrow Functions As Parameters**

By using arrow functions in JavaScript you now have a very compact syntax available for writing function expressions. Oftentimes this syntax is used in cases were you need to pass in a function as parameter. Just take a look at the following example:



Here we’re using the Array map function to iterate over the elements of the names array. An anonymous arrow function is passed into the call of map. This function is executed for each single array element and capitalize the first letter of each name:

Graphical user interface, text, application

Description automatically generated

**Arrow Functions Use Lexical This**

Another important aspect to know about arrow functions: these type of functions do not have their own scope of *this*. The value of *this* in an arrow function is inherited from the enclosing scope. This concept is called lexical this and enables you to access the outside scope of the function easily.

☞ [**Find online courses to boost your web dev skills on Udemy (\*)**](https://click.linksynergy.com/deeplink?id=a1o1REVAqJg&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourses%2Fdevelopment%2Fweb-development%2F)☜

*\* Affiliate Link / Advertisement: This blog post contains affiliate links, which means that if you click on one of the product links and buy a product, we’ll receive a small commission. There are no additional costs for you. Thank you for the support!*