

# Anonymous Functions

This lesson briefly explains what are Anonymous functions and why are they used.

Declaration is not the only way to create functions in JavaScript. Check out this example.

```
1 const hello = function(name) {  
2   const message = `Hello, ${name}!`;  
3   return message;  
4 };  
5  
6 console.log(hello("Richard")); // "Hello, Richard!"
```

In this example, the function is assigned to the `hello` variable. The value of this variable is a function. We call the function using that variable. This is an example of a *function expression*. A function expression defines a function as part of a larger expression, typically a variable assignment.

The function created in this example has no name: it is *anonymous*. As you'll soon discover, anonymous functions are heavily used in JavaScript. Here's how to create an anonymous function and assign it to a variable.

```
1 // Assignment of an anonymous function to the myFunc variable  
2 const myFunc = function(param1, param2, ...) {  
3   // Statements using param1, param2, ...  
4 };  
5  
6 // Anonymous function call  
7 // param1 value is set to arg1, param2 to arg2, ...  
8 myFunc(arg1, arg2, ...);
```

Recent language evolutions have introduced a more concise way to create anonymous functions:

```
1 const hello = (name) => {  
2   const message = `Hello, ${name}!`;   
3   return message;  
4 };  
5  
6 console.log(hello("William")); // "Hello, William!"
```



Functions created this way are called *fat arrow functions*.

```
1 // Assignment of an anonymous function to the myFunc variable  
2 const myFunc = (param1, param2, ...) => {  
3   // Statements using param1, param2, ...  
4 };  
5  
6 // Anonymous function call  
7 // param1 value is set to arg1, param2 to arg2, ...  
8 myFunc(arg1, arg2, ...);
```



Fat arrow function syntax can be further simplified in some particular cases:

- When there's only one statement in the function body, everything can be written on the same line without curly braces. The `return` statement is omitted and implicit.
- When the function accepts only one parameter, parentheses around it can be omitted.

```
// Minimalist to the max  
const hello = name => `Hello, ${name}!`;  
  
console.log(hello("Kate")); // "Hello, Kate!"
```



Functions are a core part of the JavaScript toolset. You'll use them constantly in your programs.