Function Declaration in JavaScript

A short introduction to JavaScript functions, the number of ways it can be declared in JavaScript and a few concepts related to it.

A *function* is a piece of code which is defined only once but can be called a countless number of times. A JavaScript function comprises several components which affect its behavior. A typical JavaScript function has the following components:

- the `function` keyword
- the name
- the parameter(s)
- the returned value
- the return type
- the context `this`

> **Did you know?**
>
> *In JavaScript, functions are actually objects. Just like any typical object, they have attributes and methods too. The only thing that differentiates them from objects is that they can be called*

What's the difference between Named and Anonymous function?

The only difference is that *Anonymous Functions* are declared at runtime, means they are defined and called at the same time. The reason why they are called Anonymous is that they are not given a proper name before compilation. A typical way to declare a function in JavaScript is:

```
function myFunction()
{
```

```
{
  console.log("Hello! I'm a named function!");
}


myFunction();
```

Now here’s how an anonymous function is dynamically declared:

```
var myFunction = function()
{
  console.log("Hello! I'm an Anonymous function");
}

myFunction();
```

Ways to Declare JavaScript Function

There are many ways to declare a function in JavaScript. The two most common ways are by using function declaration or by function operator. In the function declaration, the `function` keywords appear before the function name. Whereas if the `function` keyword appears anywhere else, that means we are declaring the function by the function operator method. Given below are the six different methods which are used in JavaScript to declare a function.

- *Function Declaration*
- *Function Expression*
- *Generator Function*
- *Generator Function Expression*
- *Arrow Function*
- *Function Constructor*

These declaration types control how the function interacts with its external components like the outer scope, the context, and the object that owns it. Given below is a table which covers all the types, their syntax and when is suitable to use them:

Name	Explanation	Syntax
Function Declaration	This is the most typical method to declare a function in JavaScript. All functions declared using this method allow hoisting; means they can be used before declaration.	• `function function_name(Arg1, Arg2..){}`
Function Expression	This is the most commonly used type. It is most suitable to use when you want to assign your function as an object to a variable. It's often used when you want to use your function as callback function.	• *Named*: `var var_name = function function_name(Arg1,Arg2..){};` • *Anonymous*: `var var_name = function(Arg1, Arg2..){};`
Generator Function Declaration	It is used to declare a Generator Function, a function that uses `yeild` keyword to return a Generator-Iterator object on which `next` method can be called later.	• `function* name(Arg1, Arg2..) {}`
Generator Function Expression	This is much similar to the type we just discussed above. The only difference is that	• *Named*: `function* function_name(Arg1,Arg2..){}`

	it allows omitting name from the function.	• *Anonymous*: `function*(Arg1,Arg2..){}`
Arrow Function	The two reasons why this type of functions were introduced in ES6 are: writer shorter syntax for function expressions and get rid of `this` value. You can exclude function parentheses if it only takes one parameter. You can also erase the curly brackets if there's only one statement inside function body.	• `var var_name = (Arg1, Arg2..) => {};`
Function Constructor	This is the least recommended way of declaring a function. Here, the `Function` keyword is actually a constructor which creates a new function. The arguments passed to the constructor become arguments of the newly created function and the last parameter is a string which is converted	• `var var_name = new Function(Arg1, Arg2..,'FunctionBodyString');`

	into a function body. This may cause security and engine optimization problems which is why it’s always never recommended to use.	

Coding Challenge: Write a JavaScript function expression

Rewrite the `cube()` function given using a ‘function expression’. It can be named or anonymous but remember to name the object (i.e., `var` etc) `cube` or your code won’t compile.

```
// Add your function expression here
function cube(n) {
}
```

That’s pretty much about functions! In the next lesson, we will discuss the Arrow Functions in detail and see what we can achieve through them in React.