**Introduction and Basics of JavaScript | Part 5 -** [Denis listiadi](https://denislistiadi.medium.com/?source=post_page-----b431ddbfabd8--------------------------------) Apr 1, 2022

Before continuing with basic JavaScript, if you haven’t read part 4, please read it first at [Introduction and Basics of JavaScript | Part 4](https://denislistiadi.medium.com/introduction-and-basics-of-javascript-part-4-41629ee1b588). Without further ado, let’s get straight to the basics of JavaScript.

**Default Parameter**

Now that we know that the parameter is optional, it means that we can give no value to the parameter.

We can also give the parameter a default value, meaning that if we do not send data to the parameter or we send undefined data, the parameter will automatically be filled by the default value.

Text

Description automatically generated

Default Parameter

**Rest Parameter**

Rest Parameter is a feature where we can send as much data as possible in one parameter, and it will automatically be converted into an Array. To create a rest parameter, there are conditions.

There can only be one rest parameter in the function, there can be no more than one.

Rest parameters can only be in the last position, not in front or the middle unless there is only 1 parameter. In other programming languages, some say this is a variable argument

Text

Description automatically generated

Rest Parameter

**Spread Syntax**

Sometimes we already have data in the form of an Array. But luckily we can also pass Array to Rest Parameter. We can use … (three dots) followed by the array when calling the function. This feature is called Spread Syntax.

A screenshot of a computer

Description automatically generated with medium confidence

Spread Syntax

**Arguments Object**

Before there was a rest parameter feature, in JavaScript there was already a feature called arguments object. This is a feature where we can retrieve all the parameters in the form of an array by using an object named argumentscan automatically be used in functions.

But keep in mind, that for current JavaScript, it’s better to use the rest parameter over the arguments object

Text

Description automatically generated

Arguments Object

**Functions As Value**

Functions can not only be used as executable program code but can also be used as values. That is, functions can be stored in variables, can also be sent via parameters to other functions.

**Function in Variable**

Text

Description automatically generated

Function in Variable

In this case, we create a function sayHello and put it in the say variable. So to call the sayHello function we simply call the variable name followed by the parameters.

**Function in Parameter**

Text

Description automatically generated

Function in Parameter

Continuing the code above, we use the sayHello function as a parameter to the giveMeName function. to call it we just use the name of the function followed by the name of the function parameter, or you can use the name of the variable that holds the function.

**Anonymous Function**

Previously we always created a function with the name. We can also create a function without a function name, or the term is an anonymous function.

We can create an anonymous function in a variable or we can create it when we fill in parameters

Text

Description automatically generated

Anonymous function in a variable

Text

Description automatically generated

Anonymous function in the parameter

**Functions in Function**

There is no limit to where we can create functions. Including if we want to create a function inside a function, we can do that.

The function contained within, we call the inner function. The inner function can only be accessed where we created the function, it cannot be accessed from outside the function

Text

Description automatically generated

Functions in Function

**Scope**

The scope is a data access area. There are two types of scope, global scope, and local scope. Every time we create a function, we will create a local scope for that function

Data in the global scope can be accessed from the local scope, but the data in the local scope can only be accessed in the local scope or the local scope below it (in the case of functions within functions)

Text

Description automatically generated

global scope

Text

Description automatically generated

local scope

Text

Description automatically generated

nested function scope

**Recursive Function**

A recursive function is the ability of a function to call its own function  
Sometimes there are indeed many problems, which are easier to solve using a recursive function, such as the factorial case

Text

Description automatically generated

Factorial Loop

Text

Description automatically generated

Factorial Recursive

**Function Generator**

A function generator is a function that is used to create a data generator  
A generator is a data that can be iterated like an Array

To create a function generator, we need to use an \* (asterisk) after the word function. And to return data in each iteration, we can use the yield keyword followed by the data

Text

Description automatically generated

simple function generator

A screenshot of a computer

Description automatically generated with medium confidence

complex function generator

**Lazy Evaluation**

Generators are lazy. So it means that if we don’t take the data from the generator, then the next yield will not be executed

Text

Description automatically generated

Lazy Generator

**Arrow Function**

The arrow function is an alternative to creating a simpler function than the usual function. But there are limitations and also cannot be used in all situations. Named arrow function because it uses the => sign (like an arrow)

The following are examples of some of the disadvantages of arrow functions:

* does not have arguments object
* can’t use a function generator
* can’t access this (which will be discussed later in the function in object)
* can’t access super (which will be discussed later in JavaScript Object-Oriented Programming)

Text

Description automatically generated

Arrow Function

**Arrow Function Without Block**

If an arrow function is simple, for example, only one line  
We can create arrow functions without having to use blocks

Graphical user interface, text

Description automatically generated

Arrow Function Without Block

**Arrow Function Return Value**

Arrow functions can return values, just like normal functions  
If using block, then we need to use the return keyword  
If you don’t use block, you don’t need to use the return keyword

Text

Description automatically generated

Arrow Function Return Value

**Arrow Function Without Parameter Brackets**

If the parameter in the arrow function is only one, we can not use brackets in the parameter

A screenshot of a computer

Description automatically generated with medium confidence

Arrow Function Without Parameter Bracket

**Arrow Function As Parameter**

Because the arrow function is the same as the anonymous function  
So we can also use the arrow function as a parameter in other function

Text

Description automatically generated

Arrow Function As Parameter

**Closure**

A closure is a combination of a function and a bundle of references to the surrounding data. Okay, it’s a bit confusing, especially for those who are just learning for the first time. We already know that the local scope cannot be accessed outside the scope

With closure capabilities, we can create a function in the local scope and reference the data around the local scope, leaving the scope

Text

Description automatically generated

closure

Maybe get here first. More basic JavaScript will be posted later in part 6