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 `yield` 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 it allows omitting name from the function.	*Named*: `function* function_name(Arg1,Arg2..) {}` *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 into a function body. This may cause security and engine optimization problems which is why it's always never recommended to use.	`var var_name = new Function(Arg1, Arg2..,'FunctionBodyString');`