Javascript Core

2. Functions

There are three ways of defining functions in Javascript -:

i.Function Definitions
ii.Function Expressions
iii.Arrow Functions

Syntax for function definitions

function myFunc() {}

Syntax for function expressions

const myFunc = function() {}

The major difference between function definitions and function expressions is that while function definitions are hoisted, function expressions are not hoisted. This is an important point to remember between the two.

Also, the this keyword in the case of function definitions and function expressions behave in the same way.

Hoisting is a phenomenon by the virtue which moves the function definitions and variable declarations at the top of their respective scope. In reality, the actual code does not move to the top of the scope, rather its just the Javascript running in 2 phases : the memory allocation phase and the execution phase. During this memory allocation phase, the JS engine gets aware of all the function definitions and variable declarations.

Arrow functions

Arrow functions were the newest addition to the ES6 Javascript feature, which re-defined the way how functions were defined in the JS language.

Syntax for arrow functions/fat arrow function

const myFunc = () => {}

Arrow functions provide a much more concise way of defining functions if there is only one line of code inside the function body. It also reduces the number of words you have to type to define a function.

For example, the following code can be re-written in arrow function as follows -:

```
function sum(a,b) {
    return a + b;
}
```

AS

```
const sum = () => a+b;
```

PS : If there are multiple lines of code in your arrow function, you will have to use curly brackets to write your function body.

The major difference, however, between arrow functions and normal functions stems from how the this keyword in Javascript behaves in both of these functions.

While ***normal functions have their own this context defined based on how that function is executing***, ***arrow functions don't define their own this keyword. Rather they borrow the meaning of this from their lexical scope (immediate environment)***. *This eliminates a lot of this problems we used to have with normal functions.*

This is an important point here because now, we don't require explicit this binding of functions using call, bind and apply, which creates complexities in the code.

Anonymous functions

Another type of function declaration that you might encounter is an anonymous function. An anonymous function is a function which does not have a name associated to it. Usually, you see anonymous functions when you define callback functions.

Although anonymous functions are defined without a formal name, one major disadvantage is that it cannot be traced in the call stack.

Callback functions

All functions in Javascript are first class functions. This simply means that a function has the ability to take another function as an argument and can return back function from inside of it's body.

This lays the bedrock foundation of callback functions.

Callback functions are those functions which are passed inside of a method/function and is executed as a result of something. For example, the .map() method of Javascript takes in a callback function as an argument.

==========End of notes==========

ssss