1. Array:

In JavaScript, an array is a type of object that stores an ordered list of items. These items can be of any data type, including numbers, strings, and other objects. Arrays are a very useful tool for working with large amounts of data in JavaScript.

We can create an array in JavaScript by using the [] notation, like this:

code

let myArray = [1, 2, 3, 4, 5];

You can also create an array using the Array() constructor, like this:

code

let myArray = new Array(1, 2, 3, 4, 5);

You can access the elements of an array using their index, which is a zero-based number that represents the position of the element in the array. For example, to access the first element of the array, you would use the code:

code

console.log(myArray[0]);

This would output 1.

JavaScript arrays have several built-in methods that can be used to manipulate the data stored in an array, such as push(), pop(), shift(), unshift(), sort(), reverse(), slice(), splice(), map(), filter() and reduce() among others.

Arrays are widely used in JavaScript for storing and manipulating data, and can be used in combination with loops and other control flow statements to perform complex operations on large sets of data.

Moreover, In JavaScript, an array is a collection of elements that are stored in a single variable. The syntax for creating an array is as follows:

Syntax:

var arrayName = [element1, element2, element3, ...];

For example:

code

var fruits = ["apple", "banana", "orange"];

We can also create an empty array using the following syntax:

Syntax:

var arrayName = [];

We can also create an array using a constructor:

code

var arrayName = new Array(element1, element2, element3, ...);

We can access the elements of an array using their index, which is a number that represents the position of the element in the array. The first element has an index of 0, the second element has an index of 1, and so on.

Copy code

console.log(fruits[0]); // Output: "apple"

console.log(fruits[1]); // Output: "banana"

console.log(fruits[2]); // Output: "orange"

we can also use loops to access all the elements of an array.

Additionally, we can add/remove elements to an array using several array methods like push, pop, unshift, shift, etc.

1. Function:

In JavaScript, a function is a block of code that can be reused multiple times. Functions are defined using the "function" keyword, followed by a function name, a set of parentheses, and a pair of curly braces. The code inside the curly braces is the body of the function. Functions can take input in the form of parameters, and can also return output.

The syntax for defining a function in JavaScript is as follows:

code

function functionName(parameter1, parameter2, ...) {

// function body

// statements to be executed when the function is called

}

functionName is the name of the function. It can be any valid JavaScript identifier.

parameter1, parameter2, ... are the parameters that are passed to the function when it is called. They are optional, and a function can have zero or more parameters.

The code inside the curly braces {} is the body of the function. This is the code that is executed when the function is called.

Here is an example of a simple function in JavaScript that takes in two parameters and returns their sum:

Code:-

function add(a, b) {

return a + b;

}

console.log(add(1, 2)); // Output: 3

console.log(add(3, 4)); // Output: 7

JavaScript also provides Arrow Function which is a shorthand for creating functions, it is also known as a lambda function.

Code:

let add = (a, b) => a + b;

console.log(add(3,4)) // Output: 7

In the above example, (a, b) is the parameter list and a + b is the function body.

And in the case of a single-line function body, we can also write it in the following way

Code:

let add = (a, b) => a + b

Functions can be invoked or called by using the function name followed by a set of parentheses that contain any necessary arguments. For example, the add function above could be called like this:

code

let result = add(3, 4);

console.log(result); // Output: 7

1. Arrow Function:-

An arrow function in JavaScript is a shorthand notation for defining a function. It has a similar syntax to the regular function but uses the => (fat arrow) notation instead of the function keyword. Arrow functions are used to create anonymous functions, which can be assigned to a variable or passed as an argument to another function.

Arrow functions have the following characteristics:

* + They have a shorter syntax compared to regular functions
  + They do not have their own this and arguments bindings, they inherit this and arguments from the surrounding scope.
  + They do not have the new keyword, and cannot be used as constructors.
  + They are not hoisted, so they cannot be called before they are defined.

Syntax:-

The syntax of an arrow function in JavaScript is as follows:

const functionName = (param1, param2, ...) => { function body }

or

const functionName = (param1, param2, ...) => expression;

functionName is the name of the arrow function, which is optional.

param1, param2, ... are the parameters of the function, which are optional as well.

=> is the fat arrow notation that separates the function's parameters from its body or expression.

function body is the code that gets executed when the function is called, enclosed in curly braces {}.

expression is a single expression that gets immediately executed and returned when the function is called, without the need for curly braces {}.

Here are some examples of arrow functions with different syntaxes:

* // with multiple parameters and a function body

const add = (a, b) => {

return a + b;

}

* // with a single parameter and a function body

const square = x => {

return x \* x;

}

* // with multiple parameters and an expression

const multiply = (a, b) => a \* b;

* // with no parameters and a function body

const sayHello = () => {

console.log("Hello");

}

* // with no parameters and an expression

const returnTrue = () => true;

Please note that if the function body only contains one expression then you can put it on the same line as the arrow and return it.

It's worth noting that the const keyword is used to create a constant variable, which means that it cannot be reassigned to a different value. If you want to create a variable that can be reassigned, you can use the let keyword instead.

1. Normal function vs Arrow function example and syntax:-

Both normal functions and arrow functions in JavaScript are used to define blocks of code that can be reused multiple times, but they have some differences in terms of syntax and behavior.

Here is an example of a normal function in JavaScript that takes two parameters and returns their sum:

Code:

function add(a, b) {

return a + b;

}

console.log(add(1, 2)); // Output: 3

In this example, the function keyword is used to define the function named add. It takes two parameters a and b, and returns the result of adding them together using the return statement. The function is called with the arguments (1,2) and returns the sum of the arguments passed to the function.

On the other hand, here is an example of an arrow function that does the same:

Code:

const add = (a, b) => a + b;

console.log(add(1, 2)); // Output: 3

In this example, the arrow function is defined with the fat arrow notation => instead of the function keyword. It also takes two parameters a and b, but it's more concise, and it returns the result of adding them together using an expression. The function is called with the arguments (1,2) and returns the sum of the arguments passed to the function.

Here is a summary of the main differences between normal functions and arrow functions:

* Normal functions are defined using the function keyword, while arrow functions are defined using the => notation.
* Normal functions have their own this and arguments bindings, while arrow functions inherit this and arguments from the surrounding scope.
* Normal functions can be used as constructors with the new keyword, while arrow functions cannot.
* Normal functions are hoisted, while arrow functions are not.
* In terms of usage, both normal functions and arrow functions can be used to achieve the same results and it's mostly a matter of preference, but arrow functions are more concise and recommended for functional programming.