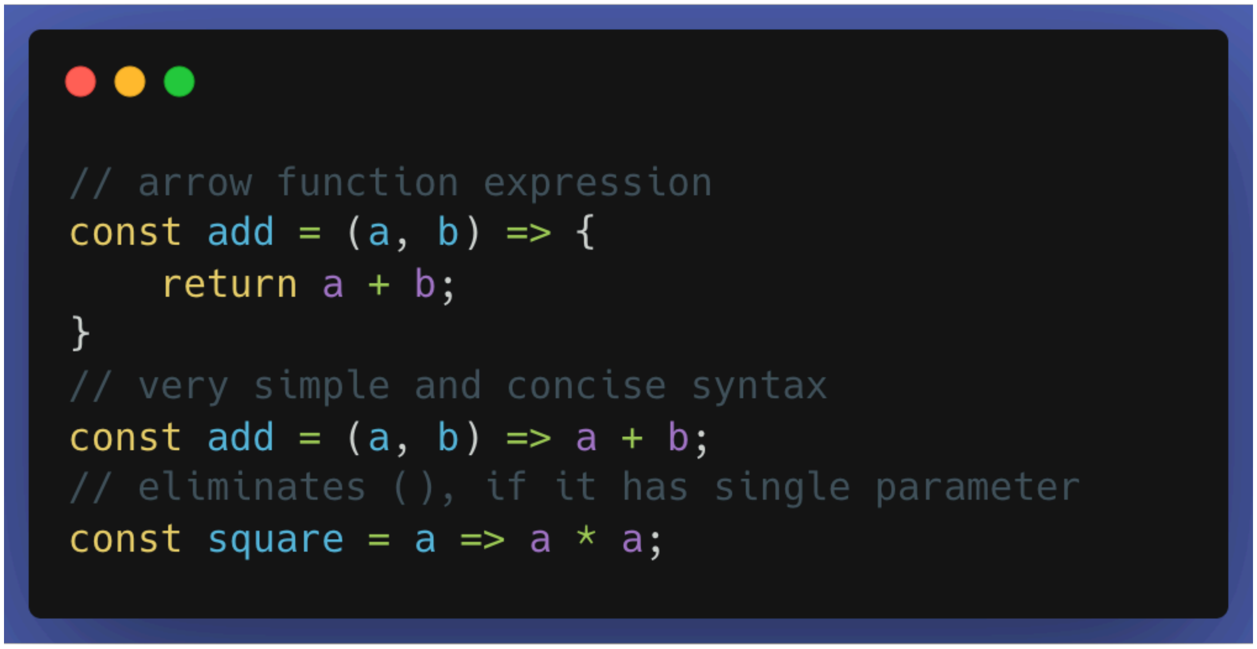
1- What is difference between Arrow Functions, Function Expressions, Function Declarations?

Arrow function



```
// arrow function expression
const add = (a, b) => {
    return a + b;
}
// very simple and concise syntax
const add = (a, b) => a + b;
// eliminates (), if it has single parameter
const square = a => a * a;
```

What is Arrow Function?

An Arrow Function in JavaScript, introduced in ES6, offers a concise syntax for defining function expressions using =>. It maintains lexical `this` binding and provides shorter, more readable code compared to traditional functions. Arrow functions enhance code structure by simplifying function syntax without sacrificing functionality or clarity.

Arrow functions are anonymous functions i.e. functions without a name but they are often assigned to any variable. They are also called **Lambda Functions**.

The below examples show the working of the Arrow functions in JavaScript.

1) Arrow Function without Parameters
An arrow function without parameters is defined as **() => { },** useful for concise anonymous function expressions without arguments in JavaScript.

2) Arrow Function with Single Parameters
An arrow function with a single parameter is written as **(param) => { }**, allowing concise handling of functions with one argument in JavaScript.

3) Arrow Function with Multiple Parameters
Arrow functions with multiple parameters, like **(param1, param2) => { }**, simplify writing concise function expressions in JavaScript, useful for functions requiring more than one argument.

4) Arrow Function with Default Parameters
Arrow functions support default parameters, allowing predefined values if no argument is passed, making JavaScript function definitions more flexible and concise.

5) Return Object Literals
In JavaScript, returning object literals within functions is concise: **() => ({ key: value })** returns an object { key: value }, useful for immediate object creation and returning.

Advantages of Arrow Functions

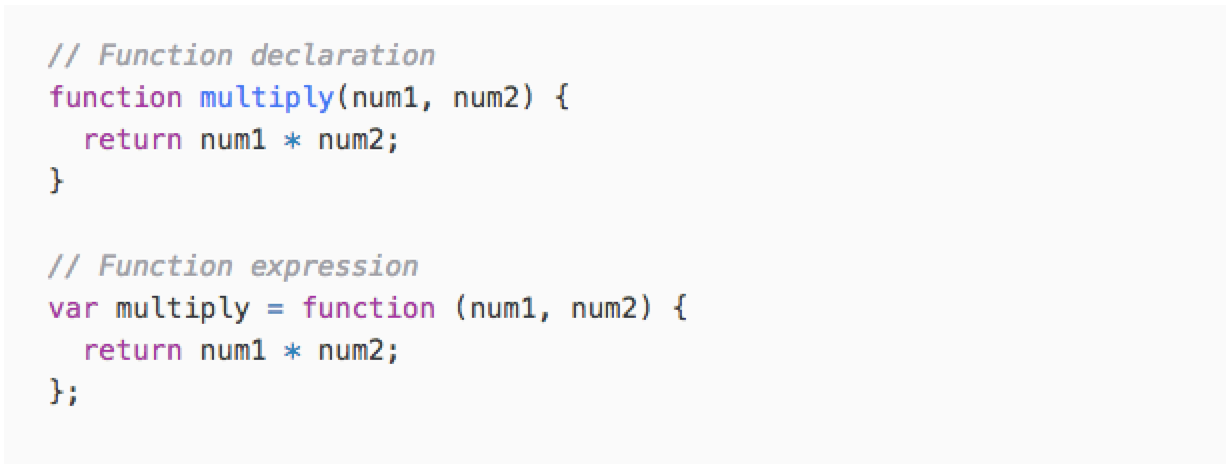
- Arrow functions reduce the size of the code.
- The return statement and function brackets are optional for single-line functions.

- It increases the readability of the code.
- Arrow functions provide a lexical this binding. It means, they inherit the value of “this” from the enclosing scope. This feature can be advantageous when dealing with event listeners or callback functions where the value of “this” can be uncertain.

Limitations of Arrow Functions

- Arrow functions do not have the prototype property.
- Arrow functions cannot be used with the new keyword.
- Arrow functions cannot be used as constructors.
- These functions are anonymous and it is hard to debug the code.
- Arrow functions cannot be used as generator functions that use the yield keyword to return multiple values over time.

Expression Function



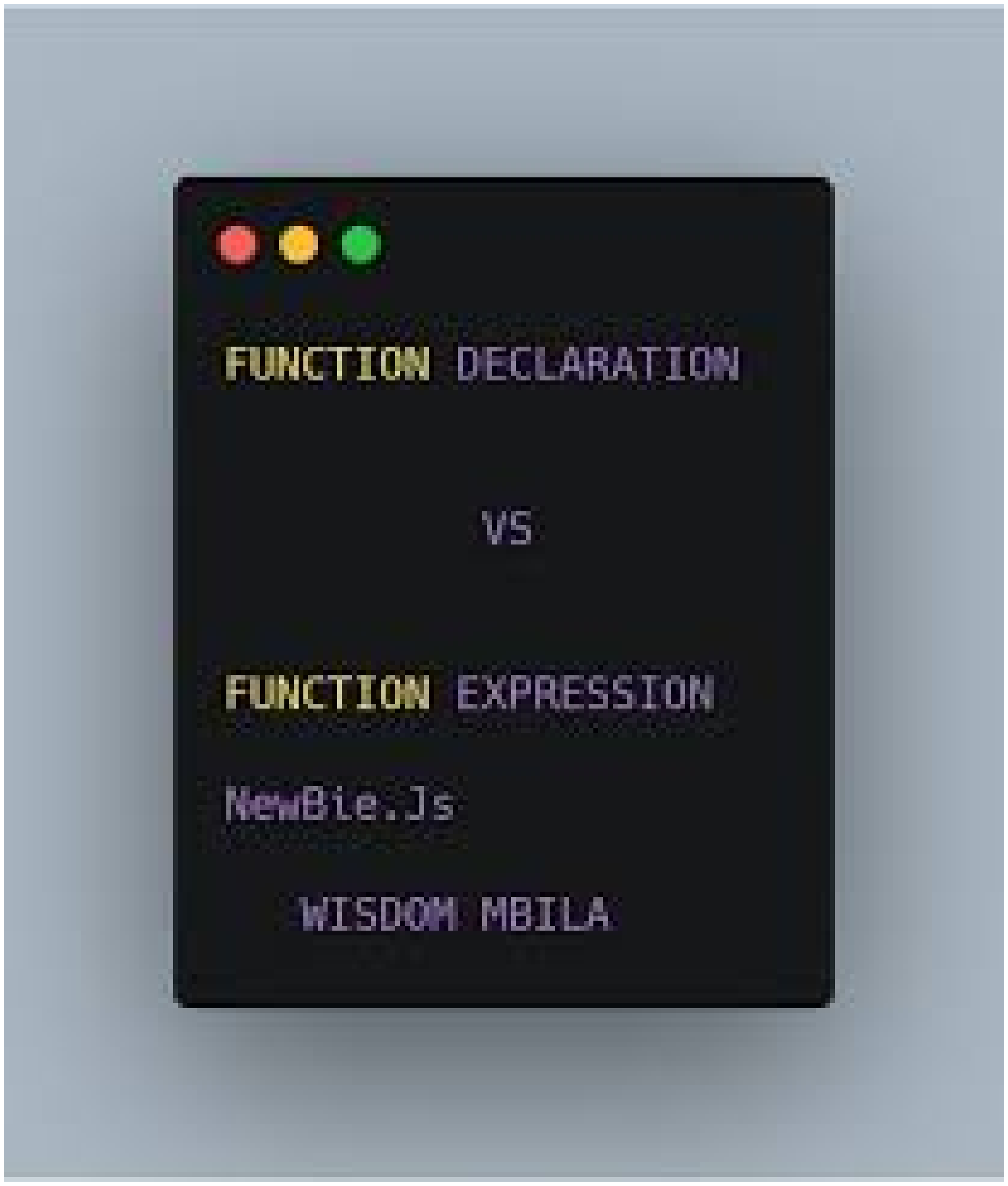
```
// Function declaration
function multiply(num1, num2) {
  return num1 * num2;
}

// Function expression
var multiply = function (num1, num2) {
  return num1 * num2;
};
```

Function expression in JavaScript is a statement that represents a constant value and it has limited use in code development and reduction of program robustness.

A `function` expression is very similar to, and has almost the same syntax as, a `function` declaration. The main difference between a `function` expression and a `function` declaration is the *function name*, which can be omitted in `function` expressions to create *anonymous* functions. A `function` expression can be used as an IIFE (Immediately Invoked Function Expression) which runs as soon as it is defined, allowing you to create an ad-hoc iterable iterator object.

Declaration Function

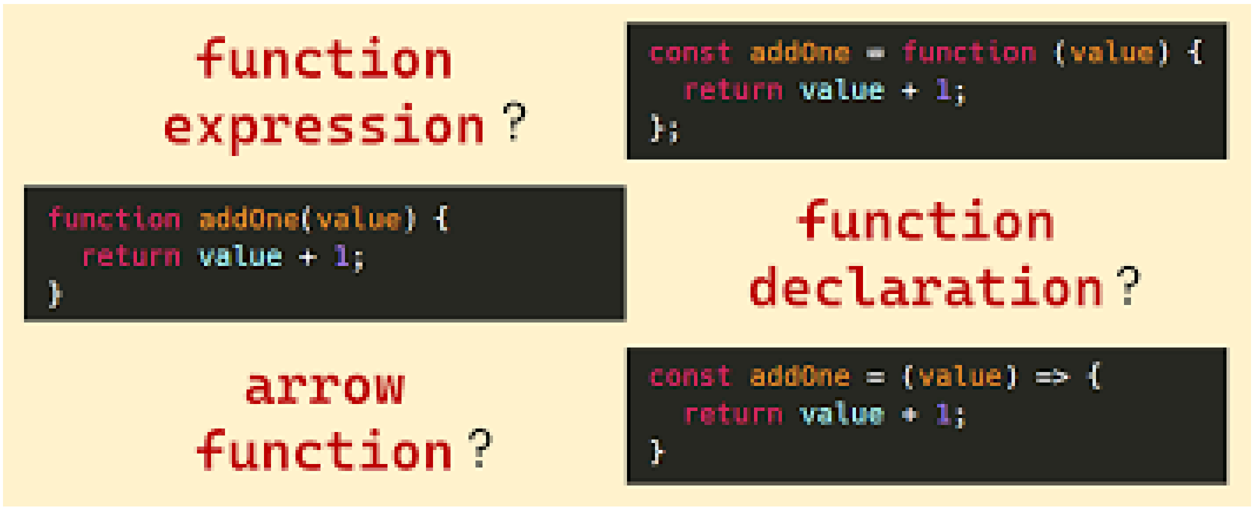


A function definition (also called a function declaration, or function statement) consists of the `function` keyword, followed by:

- The name of the function.
- A list of parameters to the function, enclosed in parentheses and separated by commas.

- The JavaScript statements that define the function, enclosed in curly braces, `{ /* … */ }`.

what is the difference between arrow function and declaration function and expression function?



```
const addOne = function (value) {
  return value + 1;
};
```

```
function addOne(value) {
  return value + 1;
}
```

```
const addOne = (value) => {
  return value + 1;
}
```

Function Declarations vs Function Expressions

1. Expressed functions cannot be used before initialization

2. Expressed functions need to be assigned to be used later

3. Anonymous functions are useful for anonymous operations

Arrow Functions:

- **Concise and Readable: Arrow functions are known for their concise syntax, which can enhance code**

readability. However, this syntax may come at a slight cost in terms of performance due to the need for lexical this binding.

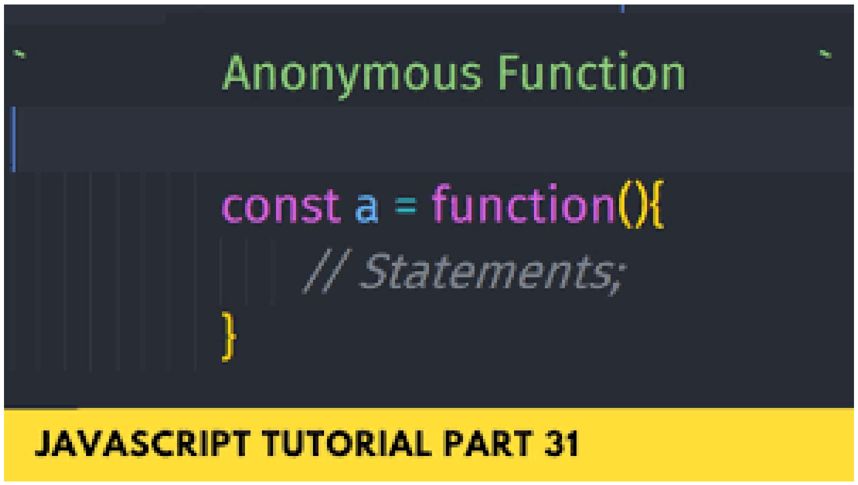
- **Predictable this Context: Arrow functions have a predictable this context, which eliminates the need for complex binding solutions. This can lead to more straightforward and less error-prone code.**

Expression Functions (Regular Functions):

- **Dynamic this Binding: Expression functions have a dynamic this binding, which allows them to adapt to different this contexts. While this flexibility is powerful, it can also lead to more complex code and potential performance overhead when handling complex binding scenarios.**
- **Optimizable: Expression functions may be more optimizable in certain cases, especially when dealing with highly performance-critical operations. JavaScript**

engines can apply optimization techniques to regular functions due to their dynamic nature.

2- What is difference between anonymous functions and arrow functions



```
const a = function(){
    // Statements;
}
```

Anonymous Functions

What are Anonymous Functions in JavaScript?

Anonymous functions in JavaScript are the functions that do not have any name or identity. Just like, you have a name by which everyone calls you or identifies you. But, the anonymous functions, do not have any name, so we cannot call them like any other function in JavaScript.

Use of Anonymous Functions in JavaScript

There are multiple uses of anonymous functions in javascript. Let us look at some very important and popular usage of the same.

Using Anonymous Functions as Arguments of Other Functions

Functions are called first-class citizens because, they can be passed as an argument to any function, or also returned by any function. The best use of anonymous functions is that they can be passed as arguments to other functions

Immediately Invoked Function Execution

Sometimes, we need to execute a function right after we declare it so that we don't need to go through the hassle of calling it. In these situations, anonymous functions are a great option since they execute right when declared.

Conclusion

Anonymous functions are used in many situations, including as callback functions and immediately invoked function expressions. Arrow functions are particularly useful when working with arrays and objects, and are commonly used in array methods such as map, filter, and reduce.