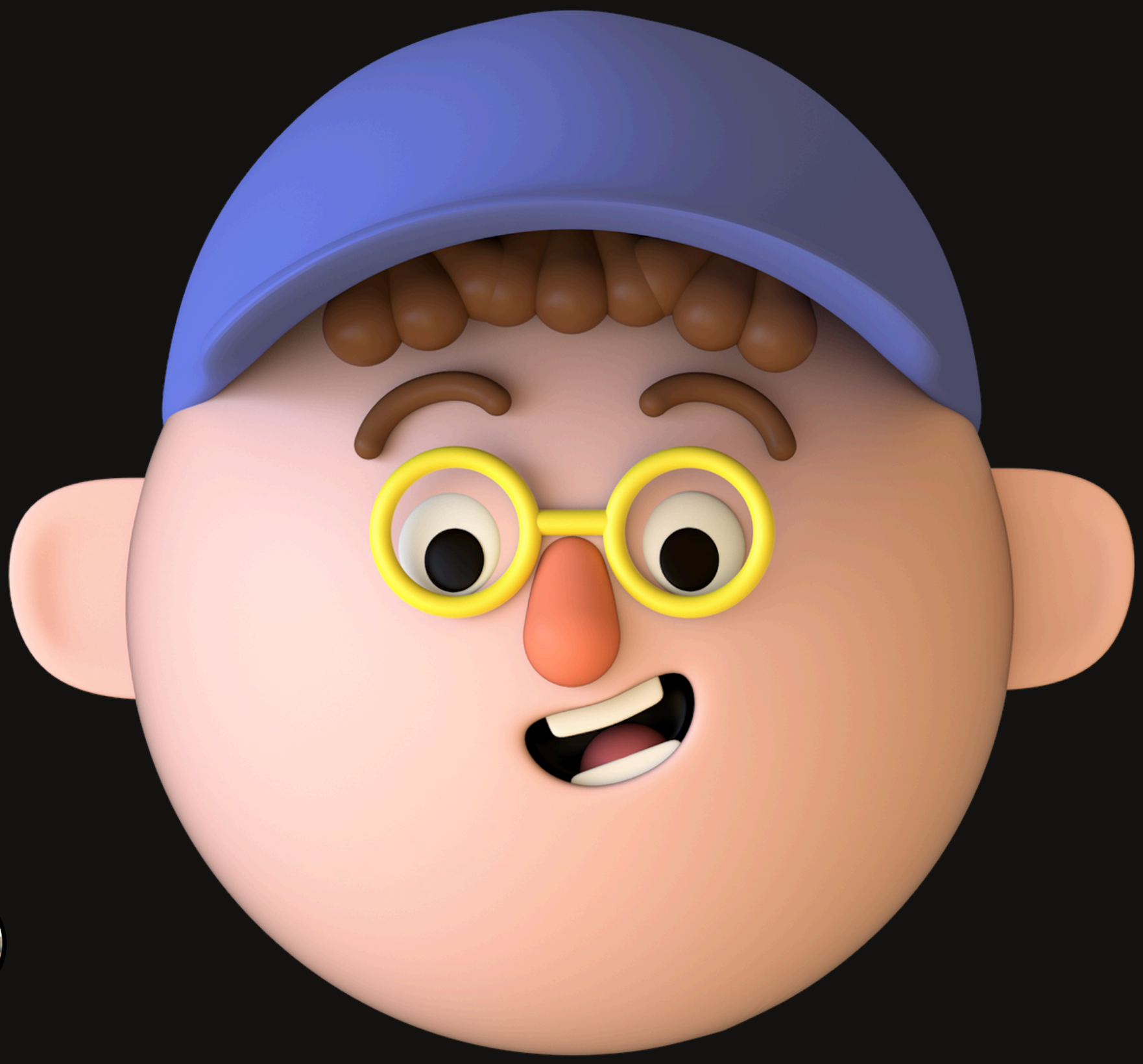
# JS ADVANCED CONCEPTS



# Closures

In JavaScript, a closure is a feature that allows a function to retain access to variables from its outer (enclosing) scope even after the outer function has finished executing. This means that the inner function “closes over” the variables it references, and they remain accessible even when the outer function has completed its execution.

# Currying:

Currying is a technique in functional programming where a function with multiple arguments is transformed into a sequence of functions, each taking a single argument. This allows you to partially apply the function and create new functions with fewer arguments. Currying is often used to create more specialized and reusable functions.

## coalescing

The nullish coalescing (??) operator is a logical operator that returns its right-hand side operand when its left-hand side operand is null or undefined, and otherwise returns its left-hand side operand. It’s commonly used to provide default values for variables.

## Proxies and Reflection:

Proxies allow you to intercept and customize operations performed on objects, providing fine-grained control over their behavior. Reflection allows you to inspect and manipulate objects at runtime. We’ll delve into Proxies and Reflection, showcasing how they can be used for metaprogramming and creating powerful abstractions.

## Generators:

Generators introduce a new way of writing iterable functions. They can be paused and resumed, allowing for more flexible control flow. Generators are particularly useful for implementing lazy evaluation and working with infinite sequences. We'll explore how to create and utilize generators effectively.

## Modules and Bundlers:

JavaScript modules enable code organization, reusability, and encapsulation. We'll explore the different module systems supported by modern JavaScript, including CommonJS and ES modules. Additionally, we'll touch upon bundlers like webpack and rollup, which bundle modules into a single file for deployment.

# Inheritances

Javascript inheritance is the method through which the objects inherit the properties and the methods from the other objects. It enables code reuse and structuring of relationships between objects, creating a hierarchy where a child object can access features of its parent object.

# JavaScript Event Loops

JavaScript has a runtime model based on an event loop, which is responsible for executing the code, collecting and processing events, and executing queued sub-tasks. This model is quite different from models in other languages like C and Java.

# JavaScript Callback Functions

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

# Async/Await And Promises

Software developers use async to define an asynchronous function. These are perfect for involving many iterations, such as fetching data from an API or reading a file from a disk. Asynchronous functions will automatically return a promise, but you can pause their execution using the await keyword. This way, the function will wait for some other promises to resolve. That can improve readability and error handling.

# Functional Programming

Functional programming (FP) has gained significant traction in the world of software development, and JavaScript developers are increasingly turning to this paradigm to solve problems more efficiently and with fewer bugs. At its core, functional programming emphasizes the use of pure functions, immutability, and advanced techniques like currying, memoization, and monads to create cleaner, more predictable code.

# Memoization

Memoization is a technique for speeding up applications by caching the results of expensive function calls and returning them when the same inputs are used again.

# this Keyword:

In JavaScript, the this keyword refers to the current execution context or the object that a function is a method of. The value of this is determined by how a function is called. Understanding this is crucial for writing object-oriented and functional JavaScript code.

# Composition

Composition means to Compose. Everything in JavaScript is treated as an object even functions in JavaScript are treated as a high-class object. Such objects are quite complex in nature to make large complex objects simple, many small objects are composed together.

# FOLLOW