JavaScript Memory Management

- JavaScript stores data in two places, those are the call stack and the heap.
- Primitives like numbers are allocated in the stack.
- Objects, funtions are stored in the heap.
- Strings are special, due to implementation details the act like primitives.

Variable Assignment

- Primitives type treat assignments as pass by value, creating a new memory address.

Pass By Copy Of a Reference

- JavaScript uses pass by copy of a Reference.
- When copying an object, it passes the memory address as the value.

Functions as Objects

- Functions are stored on the heap same as objects.

Copying an object (Shallow Copy)

- Shallow copy with spread operator copies values, not the references.

Scopes

- JavaScript has 4 different types of scopes. Those are Global, Module, Block and Function scopes.
- In Node.js varibles that are not defined using let, var or const those are global.
- Inside the function, If a variable declared with let, have block scope, meaning they are scoped to block in which they are defined.

Closer

- A closer allows inner functions to access the variables of their outer functions even after the outer functions have finised their execution.

setTimeOut(func, timeout)

- Its a JS built in funciton, that takes two argument: func and timeout
- func is typically a function or code block that you want to run asynchronously after a specified delay.
- timeout is the amount of time to wait before executing func, measured in milliseconds.

Arrays in JS

Arrays in JS are used to store collection of values. They are created using square brackets and can hold various data type.

```
let listOfNumbers = [2, 3, 5, 7, 11];
```

Accessing Array Element

We can access elements in an array using square bracket.

```
console.log(listOfNumbers[2]); // Outputs 5
```

Array vs Object

Arrays are used when you have a collection of values with numeric indices, while objects are used when you want to associate values with named properties.

- In JavaScript, arrays are actually a type of object. When you use typeof to check the type of an array, it returns "object" because arrays are a specialized form of objects with some additional behavior.

  ```
  let num = [1, 2, 3];
  console.log(typeof num); // output: object
  ```

Methods

- Methods are functions that are associated with objects or data types
- Objects and arrays in JS have their built in methods that can be used to perform actions or operations on them.

Arrays And Methods:

Arrays have methods like push and pop that allows to manipulate their contents.

```
let num = [1, 2, 3];
num.push(4);
num.push(5);
console.log(num); // output: [1, 2, 3, 4, 5]
console.log(num.pop()); // output: 5
```