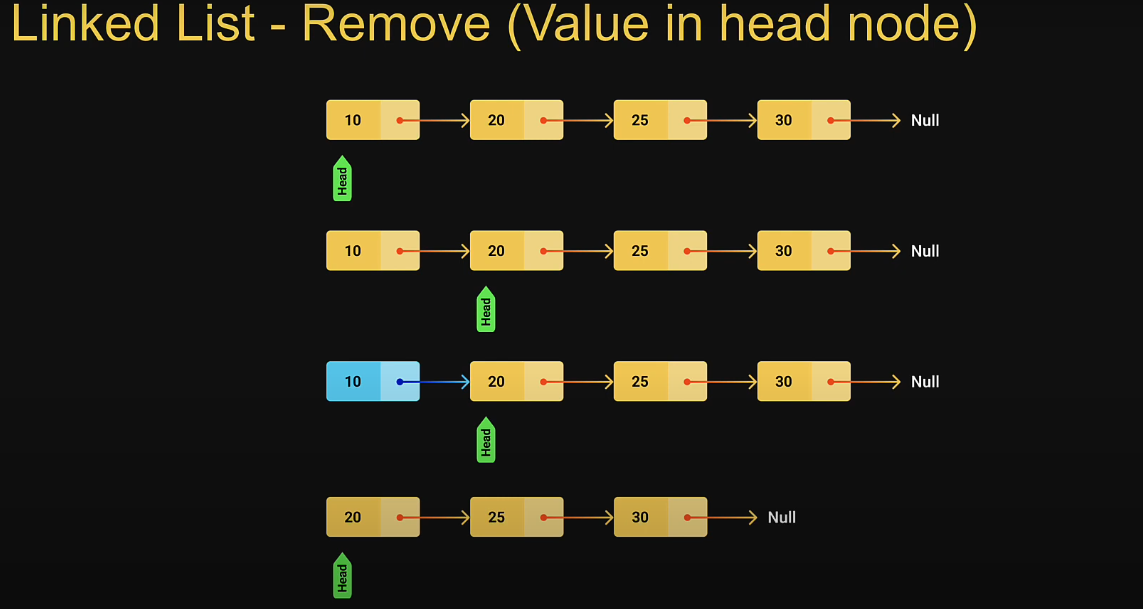
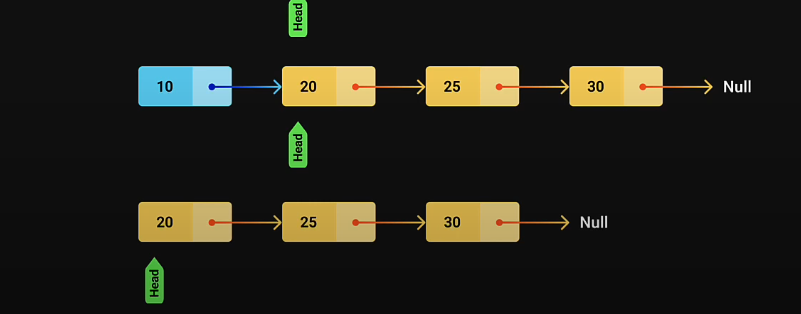
Linked List

Linked list – (Remove the values in head Node)

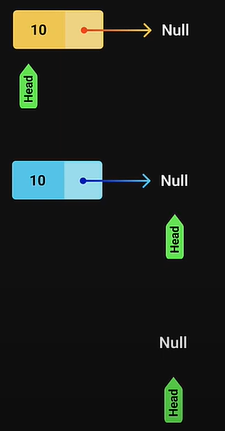


Removing head node involves pointing second node in the list.

In terms of code head points to the next pointer which always points to the second node in the list.



First node not reachable, it Collect to the garbage. Head continue the pointing to the next first Node.

If the list only one Node to begin with Head points at its own pointer it just be Null.

When you remove the Node in the list it would have only list.

  removeNode(val) {

                let remNode

                if (this. head .data == val) {

                    this. head = this.head.next

                    return value

c

                else {

                    let pre = this. head

                    //console.log(pre)

                    console.log(pre)

                    while (pre. next && pre.next.data !== val) {

                        pre = pre. next

                    }

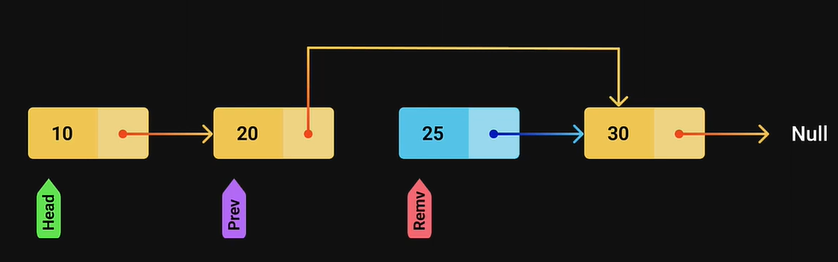
                    console.log(pre)

                    remNode = pre. next

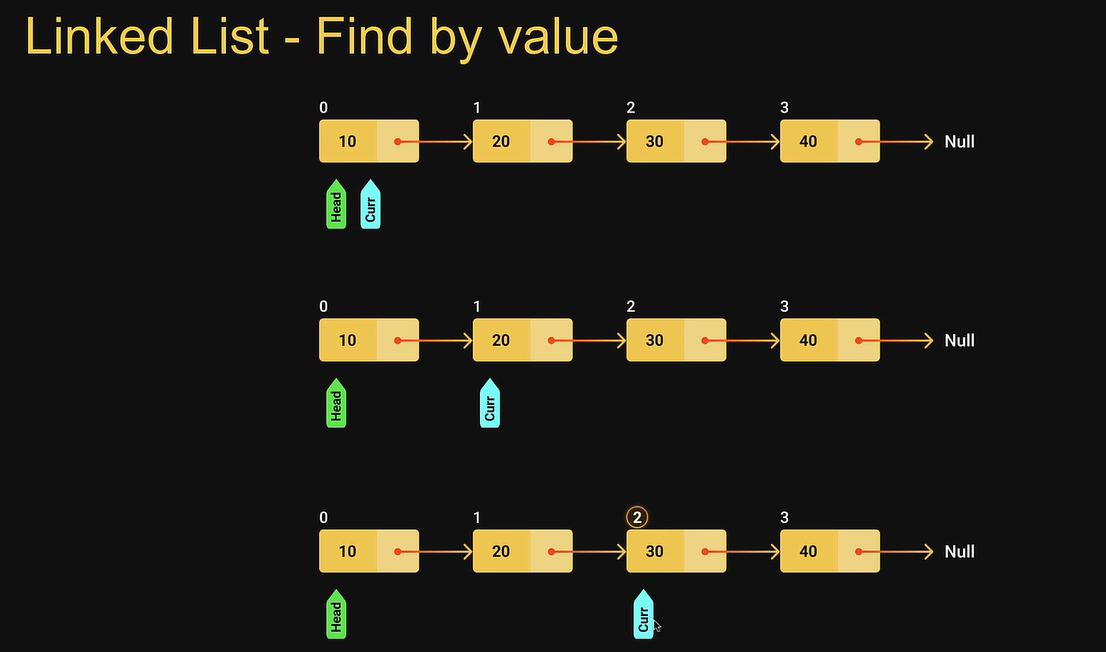
                    pre. next = remNode.next

                }

            }



**Linked List Search**

Find its Node at given value for our implementation we will return its index or Position value present -1 value not present.

Searching Node returning Index value

search(value) {

                let i = 0

                let curr = this. head

                while (curr) {

                    if (curr.data === value) {

                        return i

                    }

                    curr = curr. next

                    i++

                }

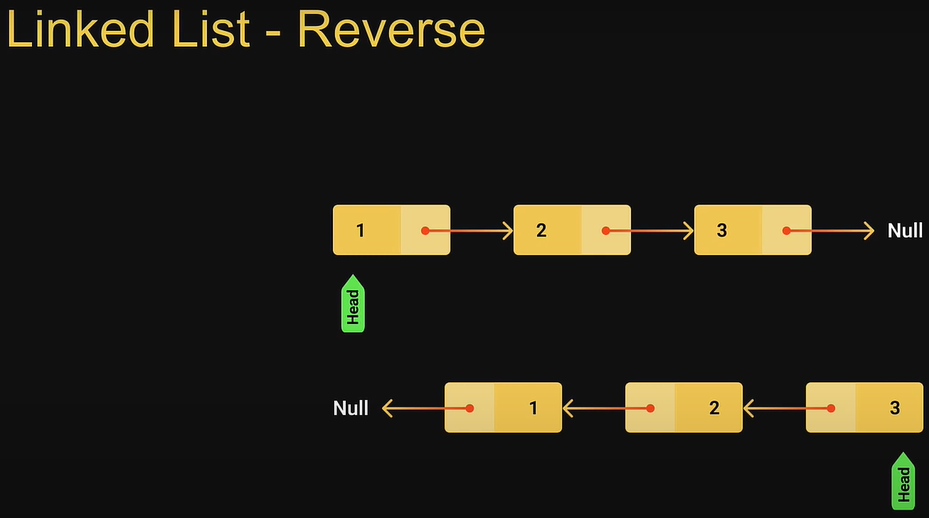
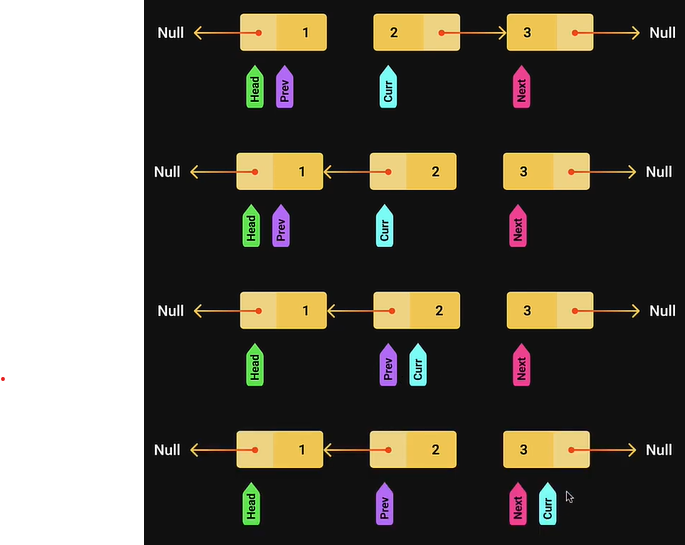
                 // if (curr == null) {

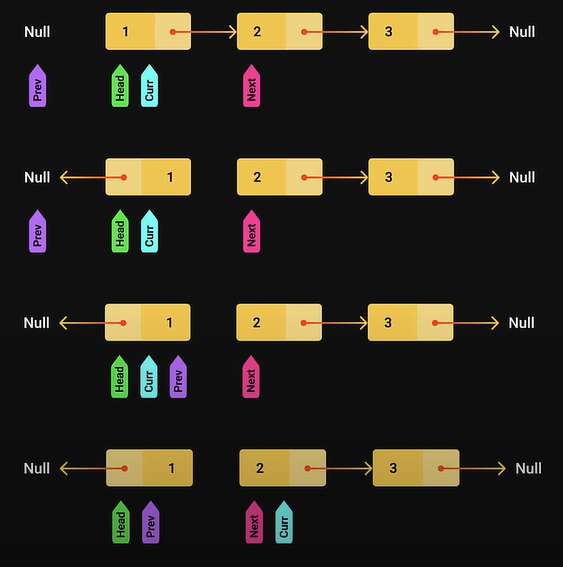
                //     return -1

                // }

                return -1

            }

Reverse Linked List



 //Reverse Linked List

            Reverse () {

                let pre = null

                let curr = this. head

                while (curr) {

                    console.log(curr)

                    let next = curr. next

                    // curr. next=pre

                    // console.log(next)

                    console. Log (curr. next = pre)

                    pre = curr

                    curr = next

                }

                // console.log(pre)

                this. head = pre

            }

**Linked List with Tail overview**

When List is empty We have just head pointer pointing at null

With a tail Both Head and Tail points at null.

**Linked List with Tail implementation**