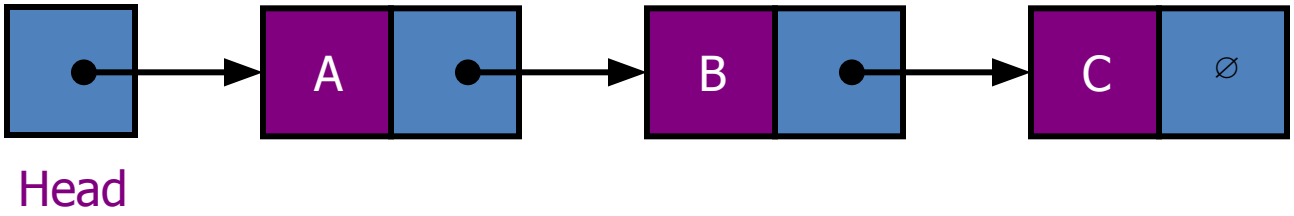
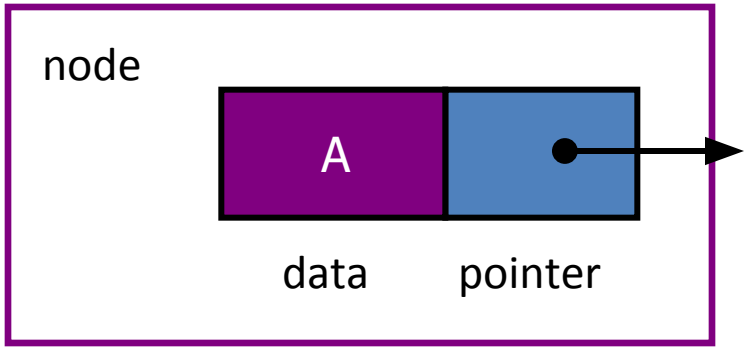
Chp 1- Linked List

Linked Lists



- A *linked list* is a series of connected *nodes*
- Each node contains at least
  - A piece of data (any type)
  - Pointer to the next node in the list
- *Head*: pointer to the first node
- The last node points to `NULL`



Singly Linked Lists

- The linked list data structure is one where each *node (element) has reference to the next* node (see Figure 1).

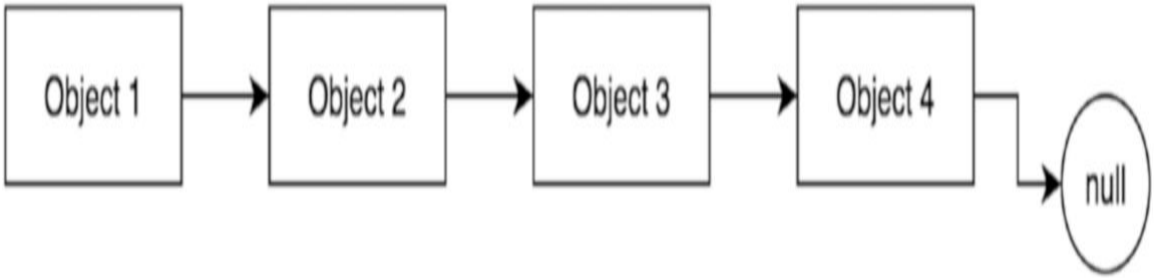
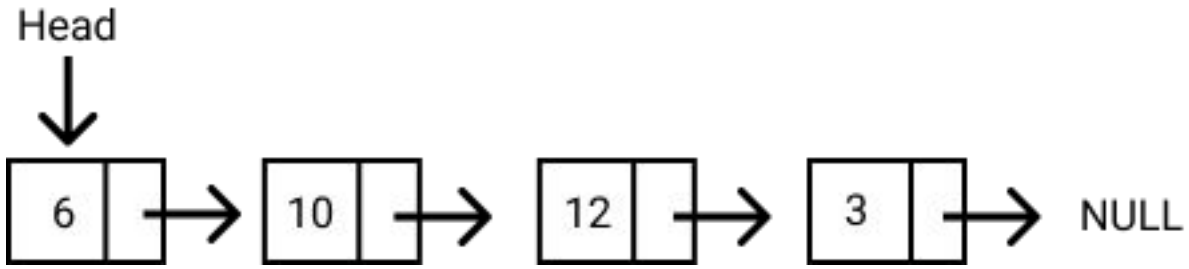


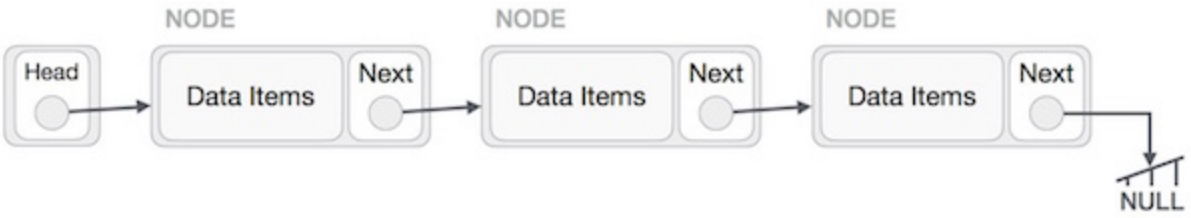
Figure 1. Singly linked list

- A node in a singly linked list has the following properties: data and next.
- Data is the value for the linked list node, and next is a pointer to another instance of SinglyLinkedListNode.

- The entry point to a linked list is called the head.
- The head is a reference to the first node in the linked list.
- The last node on the list points to null.
- If a list is empty, the head is a null reference.


Head
6
10
12
3
NULL

- Linked List Representation
- Linked list can be visualized as a chain of nodes, where every node points to the next node.
-



- Types of Linked List
- Following are the various types of linked list.
- **Single Linked List** − Item navigation is forward only.
- **Doubly Linked List** − Items can be navigated forward and backward.
- **Circular Linked List** − Last item contains link of the first element as next and the first element has a link to the last element as previous.

- Basic Operations
- Following are the basic operations supported by a list.
- **Create a list**
- **Insertion** − Adds an element in the list.
- **Deletion** − Deletes an element from the list.
- **Display** − Displays the complete list.
- **Search** − Searches an element using the given key.

(i) Creation of Linked List:

- **Algorithm:**Following are the steps for the creation of a linked list:
- (a) [Initialize the list empty]
- Set head=NULL
- (b) [Allocate space to the newly created node]
- Set p = new node
- (c) [Copy the item to the newly created node]
- Set p.data = x
- (d) [Attach the newly created node to the linked list]
- if(head ==NULL) then
- Set head = p
- else
- let current = head
- while (current.next != null) {
- current = current.next;
- }
- assign the new node to the `next` pointer
- current.next = p;
-
- (e) End.

(ii) Traversal of Linked List:

- **Algorithm:**Following are the steps for the traversal of a linked list:
- (a) [Traverse each element of the list]
- p =head
- Repeat step (b) and (c) while (p!=NULL)
- (b) [Visit element and perform the operation]
- Print the value of element of the list using- p.data
- (c) [Increment counter]
- Set p = p.next
- (d) End.

- **Implementing a LinkList Node in JavaScript**
- As stated earlier, a list node contains two items: the data and the pointer to the next node. We can implement a list node in JavaScript as follows:
- Class LLNode {
- constructor(data)
- { this.data = data
- this.next = null
- }
- }

- **Implementing a Linked List in JavaScript**
- The code below shows the implementation of a linked list class with a constructor. Here the head is initialised to null.
- class LinkedList {
- constructor() {
- this.head = null
- } }

- **Putting it all together**
- Let's create a linked list with the class we just created.