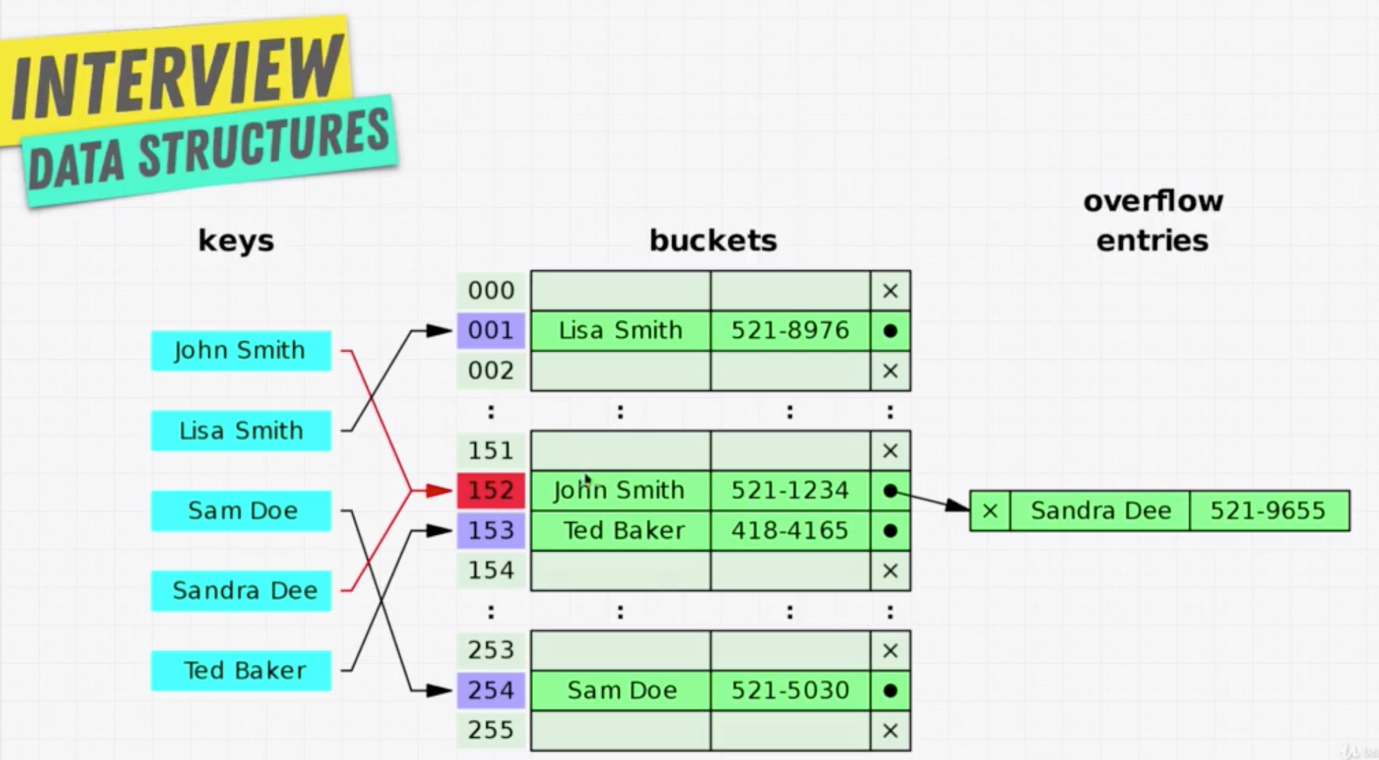
**Linked List**

* **Introduction:**

**A picture containing text, toy

Description automatically generated**

****

* **What is a Linked List?**

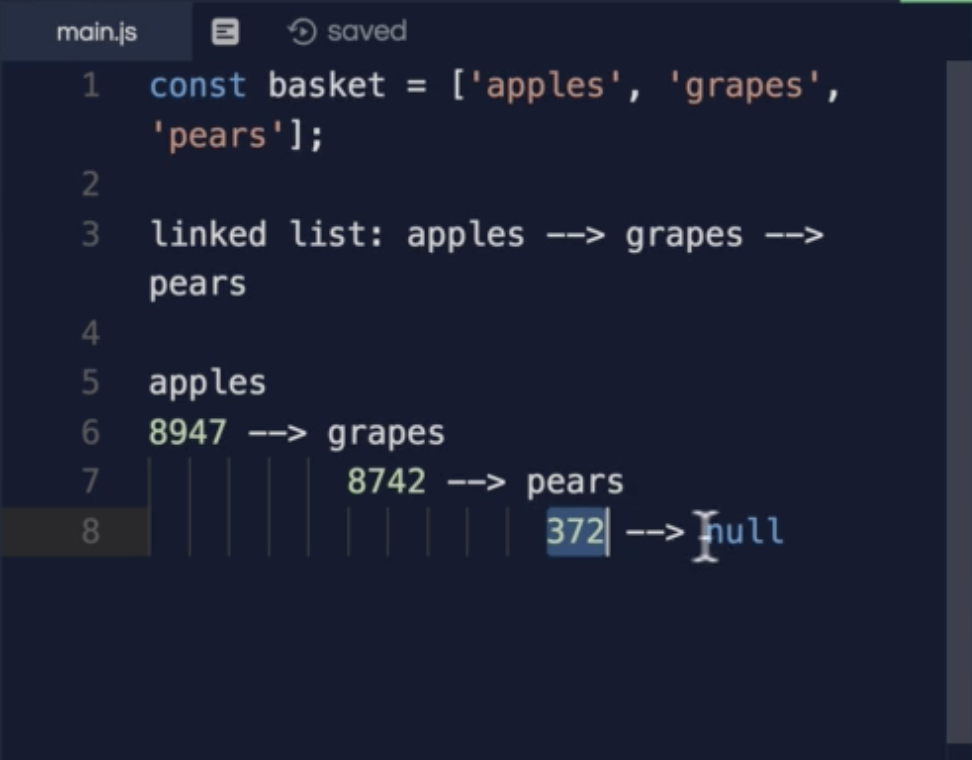
Every linked list have a Node (in the above diagram, the set of boxes green and blue).

Node will contain, two information, one is the data and another is the pointer to the next node in the list.

**A picture containing text, toy

Description automatically generated**

If the node doesn’t have a pointer to the next node, then that is considered as the end of the node.



* **Why Linked Lists?**

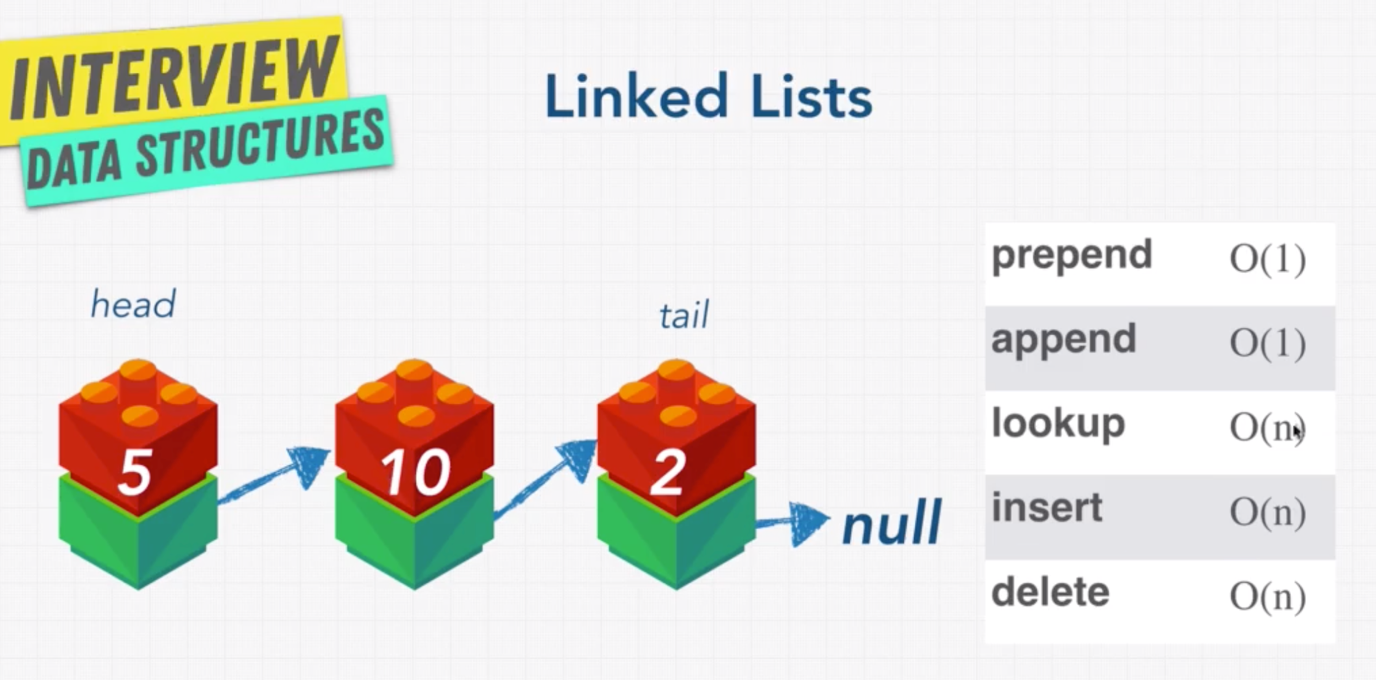
[**https://visualgo.net/en/list**](https://visualgo.net/en/list)

* **Solution – Why Linked List?**

Arrays are faster in search as they are sequential (due to indexes).

Linked List is slower in search as it is scattered, search take O(n).

To update or delete on Linked List is faster as it doesn’t need to shift like array.



* **What is a Pointer?**

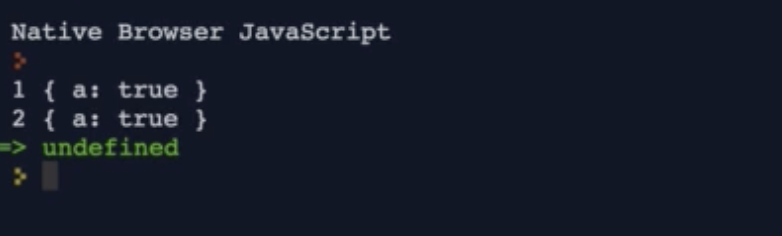
Pointers are a reference to the another node or object in the memory.

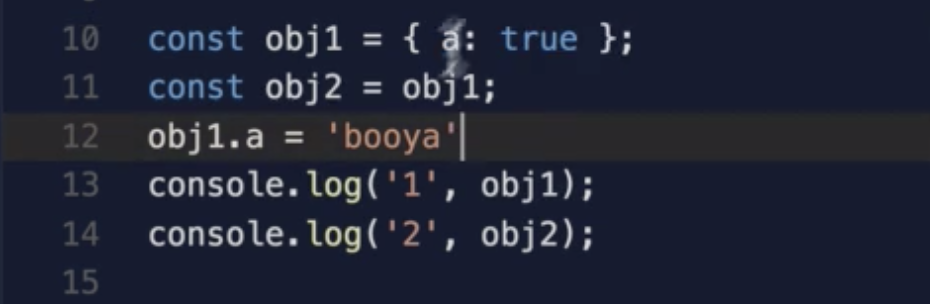
Graphical user interface

Description automatically generated

Text

Description automatically generated





A picture containing text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

In above example, even though we deleted the object1, still we are able to get the value for obj2 as booya. This is depends on the programming language we use, in general they will delete the object if the object is not referenced anywhere. In this case it is referenced by obj2.

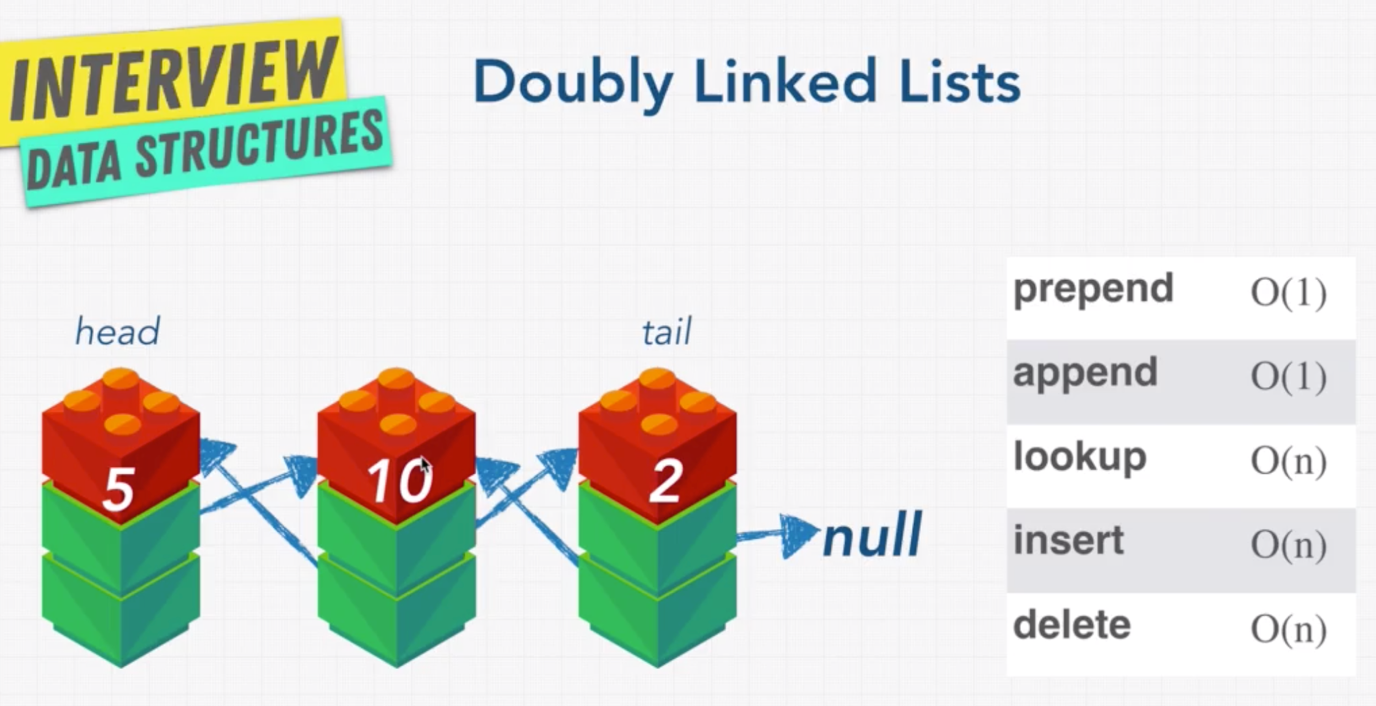
Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

* **Doubly Linked List:**

****

Doubly linked list allow us to traverse the element in backward (reverse).

* **Singly vs Doubly Linked List:**

**Singly Liked List:**

It is a simple implementation.

It requires small memory when compared to doubly linked list.

As it has only one pointer, the operations are easy.

Downside:

We can’t traverse back in the list.

**Doubly Linked List:**

Can be traversed from front and back.

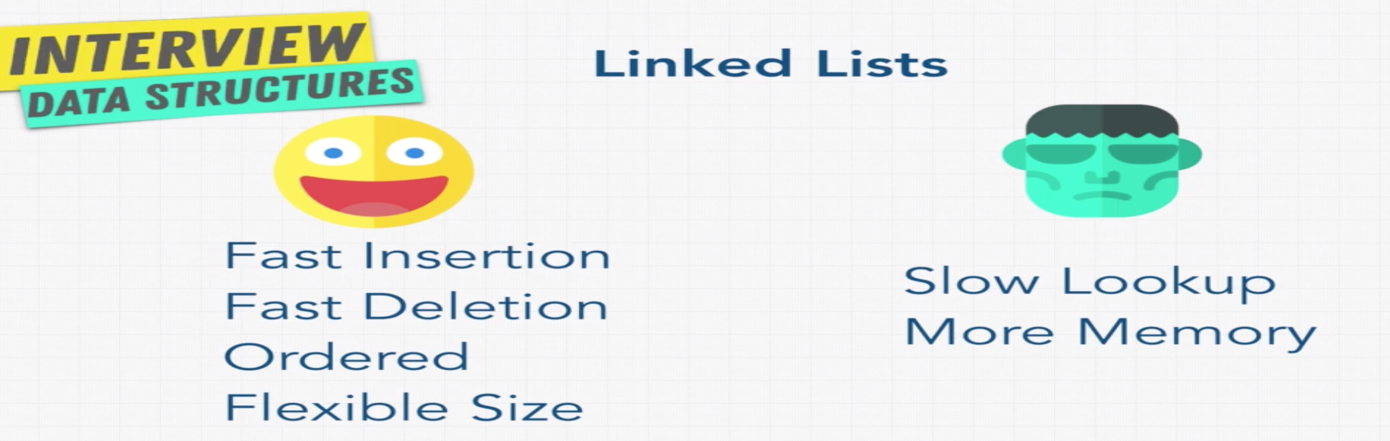
If you want to delete a previous node, you don’t need to traverse from the head node to find what is the previous node is as it has both the pointers.

Downside:

It is fairly complex.

Requires more memory in storage because is has an additional pointer.

* **Linked List Review:**

****