

Ace your JavaScript Interview. [Get my ebook](#). 100 solved Javascript, 20 solved React, & 2 frontend system design questions (**1160+ copies sold**). Get a [Free preview](#).



Advertisements

Program to reverse a linked list using a stack

Posted on [July 17, 2019](#) | by [Prashant Yadav](#)

Posted in [Algorithms](#), [Linked-List](#), [Stack](#)

An algorithm to reverse a single [linked list](#) using a [stack](#) in [Javascript](#).

Example

Input:

20 -> 5 -> 30 -> 7 -> 3

Output:

3 -> 7 -> 30 -> 5 -> 20

Copy

Practically
prepare for
your
JavaScript
interview

[JavaScript
Revision](#)

[JavaScript-
Concept Based
Problems](#)

[Data Structures](#)

[Algorithms](#)

[Machine
Coding](#)

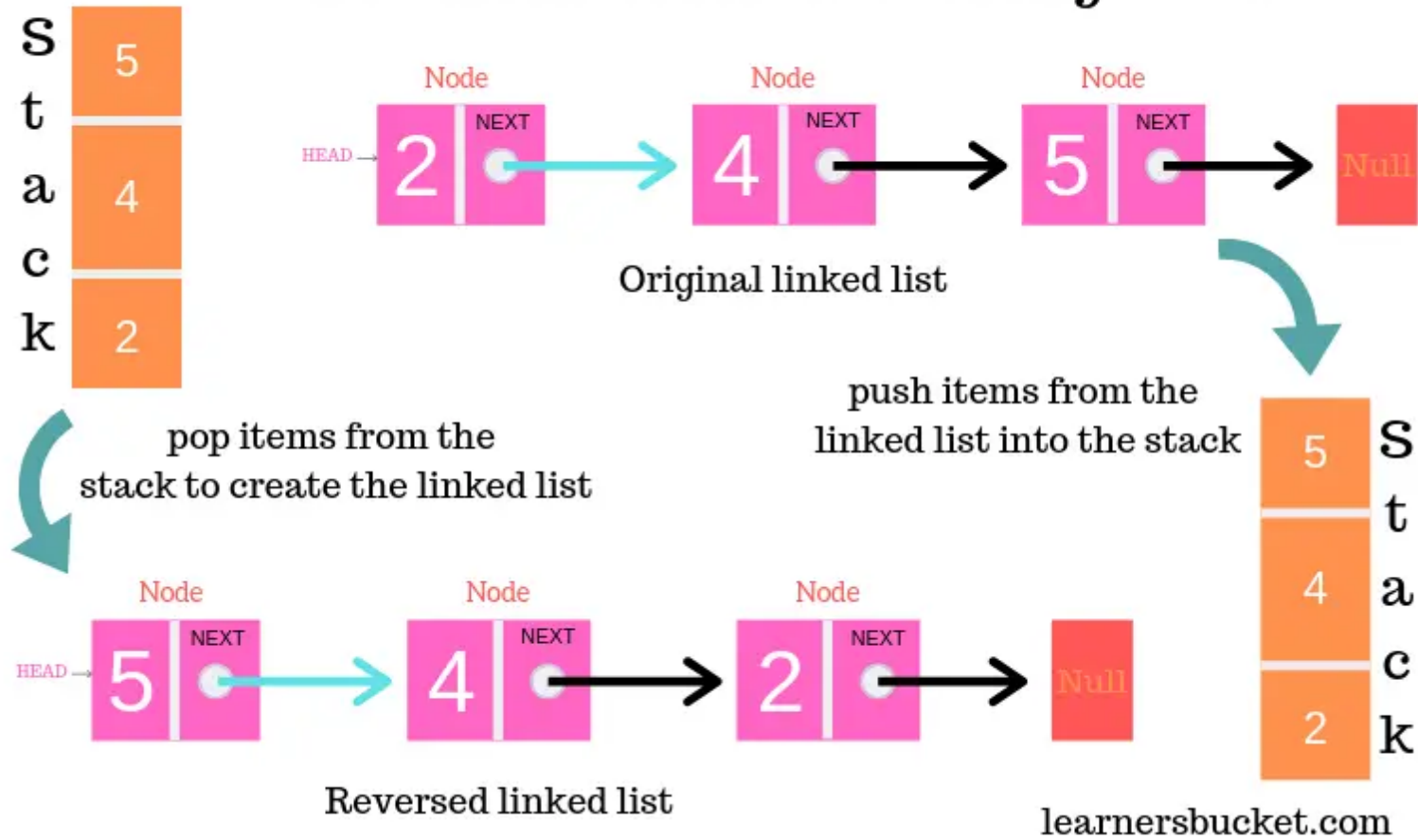
[Web
Fundamentals](#)

Advertisements

Implementation to reverse a single linked list.

- We will use a temp stack, which will store all the elements of the linked list.
- First we will copy all the elements from the linked list to the stack and then create a new linked list from the elements in the stack.
- As stack follows [LIFO](#) (Last In First Out) principle the elements will be stored in reversed order and hence the new linked list will also be created with the elements in reverse order.

Reverse a linked list using stack



```
let reverseLL = (list) => {  
  //Stack to store the List element  
  let stack = new Stack();  
  
  //Get the head of the list  
  let head = list.getHead();  
  
  //Copy all the items of the list to the stack  
  while(head){  
    stack.push(head.element);  
    head = head.next;  
  }  
  
  //Temp list to store the elements in reversed order  
  let reversedList = new LinkedList();  
  
  //Copy all the elements from the stack to the LinkedList  
  while(!stack.isEmpty()){  
    reversedList.append(stack.pop());  
  }  
  
  return reversedList;  
}
```

Copy

```
Input:  
let l1 = new LinkedList();  
l1.append(20);  
l1.append(5);  
l1.append(30);  
l1.append(7);  
l1.append(3);  
console.log(reverseLL(l1).toArray());
```

```
Output:  
[3, 7, 30, 5, 20]
```

Copy

Time complexity: $O(n)$.

Space complexity: $O(n)$.

Time and Space complexity

- As we are copying all the linked list elements into the stack and creating a new list from the stack again, Time complexity is $O(n + n) = O(n)$.
- We are using a stack to store all the elements of the linked list, so Space complexity is $O(n)$.

Prepare for your **JavaScript Interview** practically on each Interview rounds and grab that job.

BEGIN LEARNING

Recommended Posts:

[How to find loop in linked list](#)

[Check if given number is armstrong in javascript](#)

[Javascript program to find leap year](#)

[Find the largest prime factor](#)

[Count all substrings having character k.](#)

[Converting string to jadencase](#)

[Maximum consecutive one's or zero's in a circular array](#)

[Maximum Collatz sequence under 1000000](#)

[Count number of sub string recursively](#)

[Reverse a stack using recursion.](#)

[Prev](#)

[Next](#)

Advertisements





Handcrafted with ♥ somewhere in **Mumbai**

© 2023 [LearnersBucket](#) | [Prashant Yadav](#)

