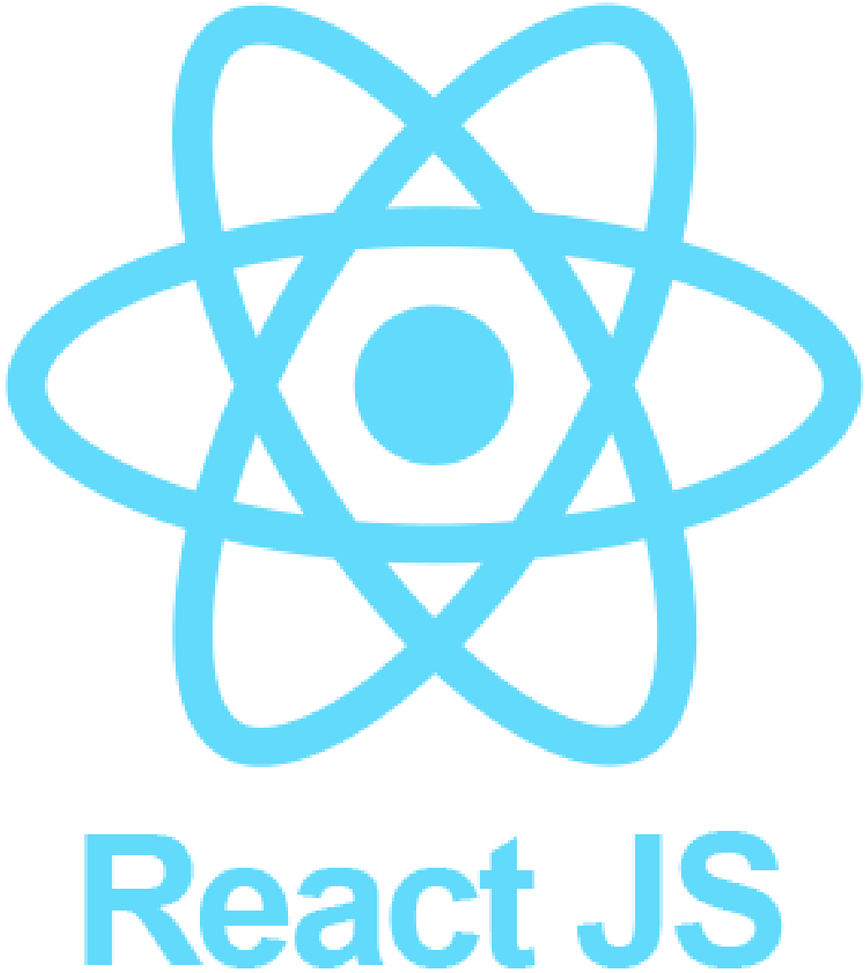
Estructura de Datos | |

Jonathan López Londoño
jlopezl@uao.edu.co
315 926 5443



Linked Lists

They are like Arrays in JavaScript, but not the same.

They are a structure of linked elements.

Linked Lists

✓ Node: Every element in the list. Each node will have 2 parts:

- Value (node data).
- Reference (pointer) to the next node in the list.

✓ Head: Is the first node in the list.

✓ Pointer: Link to next object.

✓ Tail: Last node in the list, this node will point to null.

Linked Lists Methods

A list must have methods that allow adding new elements, removing them and reviewing them (one or more).

We can make our own implementation and call these methods in any way.

There is an agreed convention of using the following definitions:

- ✓ **append**: Adds a new node to the list, placing it at any place you like, even at bottom of the list.
- ✓ **peek**: Returns some node entered on the list.
- ✓ **size**: Returns the number of elements in the list.
- ✓ **remove**: Removes a node from the list and joins previous and next node from this removed.
- ✓ **print**: Prints the contents of the stack.

How to implement Linked Lists?

First, you need to use a **Node Class**, with a value and a next in the constructor.

```
class Node {
  constructor(value) {
    this.value = value;
    this.next = null;
  }
}
```

How to implement Linked Lists?

Second, you need to use a **LinkedList class** with a head, a tail and a length in the constructor method.

```
class LinkedList {
  constructor() {
    this.head = null;
    this.tail = null;
    this.length = 0;
  }

  append(value) {···
  }

  peek(value, current = this.head) {···
  }

  size() {···
  }

  remove(value, current = this.head) {}

  print() {···
  }
}
```

How to implement Linked Lists?

Third, you need to implement basic methods:

✓ **append:** Adds a new node to the list, placing it at any place you like, even at bottom of the list.

```
class LinkedList {
  constructor() {···
  }

  append(value) {
    const newNode = new Node(value);

    if (!this.head) {
      this.head = newNode;
    } else {
      this.tail.next = newNode
    }

    this.tail = newNode
    this.length ++;
  }

  peek(value, current = this.head) {···
  }

  size() {···
  }

  remove(value, current = this.head) {}

  print() {···
  }
}
```

How to implement Linked Lists?

Third, you need to implement basic methods:

✓ **peek:** Returns some node entered on the list.

```
class LinkedList {
  constructor() {···
  }

  append(value) {···
  }

  peek(value, current = this.head) {
    while (current) {
      if (current.value === value) {
        return current;
      }

      current = current.next;
    }

    return null;
  }

  size() {···
  }

  remove(value, current = this.head) {}

  print() {···
  }
}
```

How to implement Linked Lists?

Third, you need to implement basic methods:

✓ **size:** Returns the number of elements in the list.

```
class LinkedList {
  constructor() {···
  }

  append(value) {···
  }

  peek(value, current = this.head) {···
  }

  size() {
    return this.length;
  }

  remove(value, current = this.head) {}

  print() {···
  }
}
```

How to implement Linked Lists?

Third, you need to implement basic methods:

✓ **remove:** Removes a node from the list and joins previous and next node from this removed.

```
class LinkedList {
  constructor() {···
  }

  append(value) {···
  }

  peek(value, current = this.head) {···
  }

  size() {···
  }

  remove(value, current = this.head) {
    if (!this.head) return null;

    if (this.head.value === value) {
      this.head = this.head.next;

      if (!this.head) {
        this.tail = null;
      }

      this.length--;
      return;
    }

    let current = this.head;
    while (current.next && current.next.value !== value) {
      current = current.next;
    }

    if (current.next) {
      current.next = current.next.next;
      if (!current.next) this.tail = current;
      this.length--;
    }
  }

  print() {···
  }
}
```

How to implement Linked Lists?

Third, you need to implement basic methods:

✓ **print:** Prints the contents of the stack.

```
class LinkedList {
  constructor() {···
  }

  append(value) {···
  }

  peek(value, current = this.head) {···
  }

  size() {···
  }

  remove(value, current = this.head) {···
  }

  print() {
    let current = this.head;
    let result = "";
    while (current) {
      result += current.value + " -> ";
      current = current.next;
    }
    console.log(result + "null");
  }
}
```

Double Linked Lists

They are a structure of linked elements by its previous and its next node.

They are used to go through the list from head to tail and vice versa.

Linked Lists

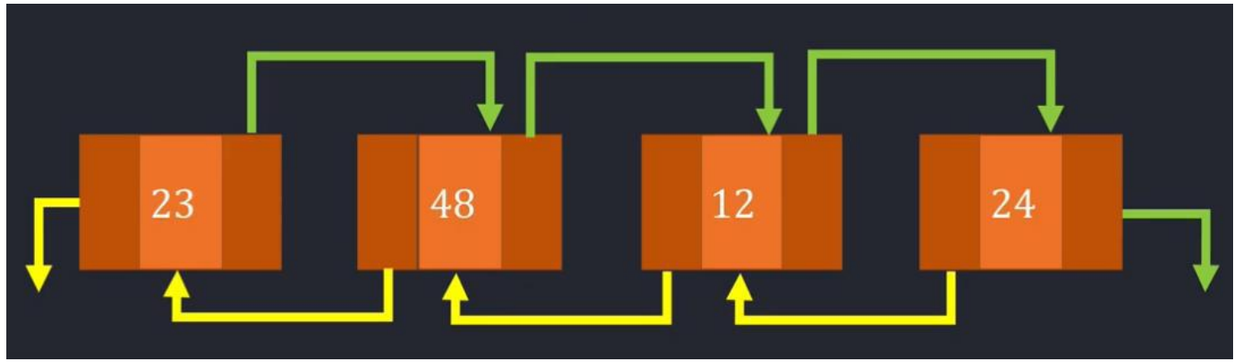
✓ Node: Every element in the list. Each node will have 2 parts:

- Value (node data).
- Reference (pointer) to the next node in the list.

✓ Head: Is the first node in the list.

✓ Pointer: Link to **next/previous** object..

✓ Tail: Last node in the list, this node will point to null.



How to implement Linked Lists?

First, just add a previous constructor of **Node Class**.

```
class Node {
  constructor(value) {
    this.value = value;
    this.next = null;
    this.prev = null;
  }
}
```

How to implement Linked Lists?

Second, add a previous in the **append** method of LinkedList Class.

```
class LinkedList {
  constructor() {···
  }

  append(value) {
    const newNode = new Node(value);

    if (!this.head) {
      this.head = newNode;
      this.tail = newNode;
      return;
    }

    this.tail.next = newNode;
    newNode.prev = this.tail;
    this.tail = newNode;

    this.length ++;
  }
```

How to implement Linked Lists?

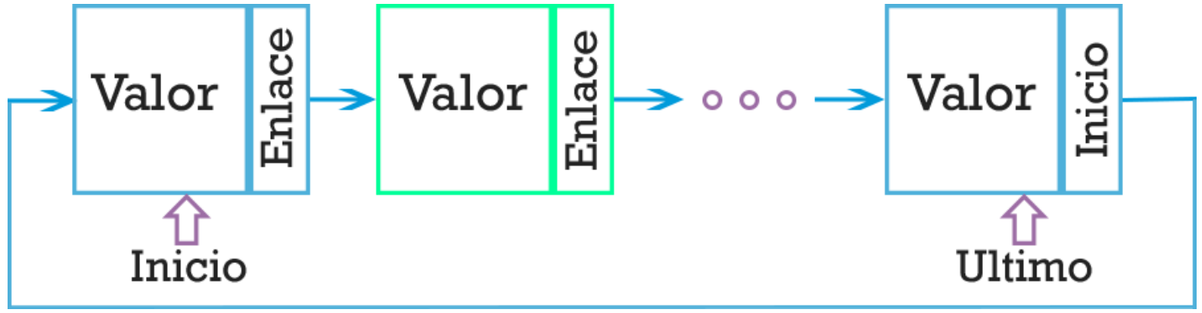
Third, add a previous in the **remove** method of LinkedList Class.

```
size() {···
}

remove(value) {
  if (!this.head) return null;

  let current = this.head;

  while (current) {
    if (current.value === value) {

      if (current === this.head) {
        this.head = current.next;
        if (this.head) this.head.prev = null;
      }

      if (current === this.tail) {
        this.tail = current.prev;
        if (this.tail) this.tail.next = null;
      }

      if (current.prev) current.prev.next = current.next;
      if (current.next) current.next.prev = current.prev;

      this.size--;
      return current;
    }
    current = current.next;
  }

  return null;
}
```

Circular Linked Lists

They are a structure of linked elements from their last node to their first node.

They are used to go through the list from tail to head.



CHALLENGE 07

1. Implement a linked list to play songs in order. Fill the list with mocked data
2. Implement a doubly linked list to navigate back and forward between visited pages in the browser . Fill the list with mocked data
3. Create a project in react with 2 new pages, linked and doubly linked lists.
4. Use the implemented lists in each page. Navigate through the lists by using buttons inside the pages