WEB TECHNOLOGIES

Node JS – Event Loop and Event Emitter

Prof. Vinay Joshi and Dr. Sarasvathi V
Department of Computer Science and Engineering

Acknowledgement
The slides are created from various internet resources with valuable contributions from multiple professors

NODE JS

Events

- Node.js is a single-threaded application, but it can support concurrency via the concept of **event** and **callbacks**.
- Every API of Node.js is asynchronous and being single-threaded, they use **async function calls** to maintain concurrency.
- Node uses observer pattern. Node thread keeps an event loop and whenever a task gets completed, it fires the corresponding event which signals the event-listener function to execute.

Events Loop

- The Event Loop is a core concept in Node.js that enables asynchronous, non-blocking I/O operations.
- It allows Node.js to handle multiple operations concurrently without blocking the execution of other code.

How the Event Loop Works

- The Event Loop continuously checks the event queue for pending events or tasks.
- When an event or task is found, it's processed, and its associated callback function is executed.
- This ensures that Node.js can efficiently manage I/O operations without waiting.

Events Loop

Example

```
const fs = require('fs');

fs.readFile('file.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log('File data:', data);
});

console.log('Reading file...');
```

NODE JS

Event Emitter

- The Event Emitter is a built-in Node.js module that allows you to create custom event-driven APIs.

How Event Emitter Works

- Objects in Node.js can emit named events.
- Event Emitters can register listeners (functions) that are invoked when the emitter emits the corresponding event.
- This enables the creation of custom events and handling of asynchronous operations.

NODE JS

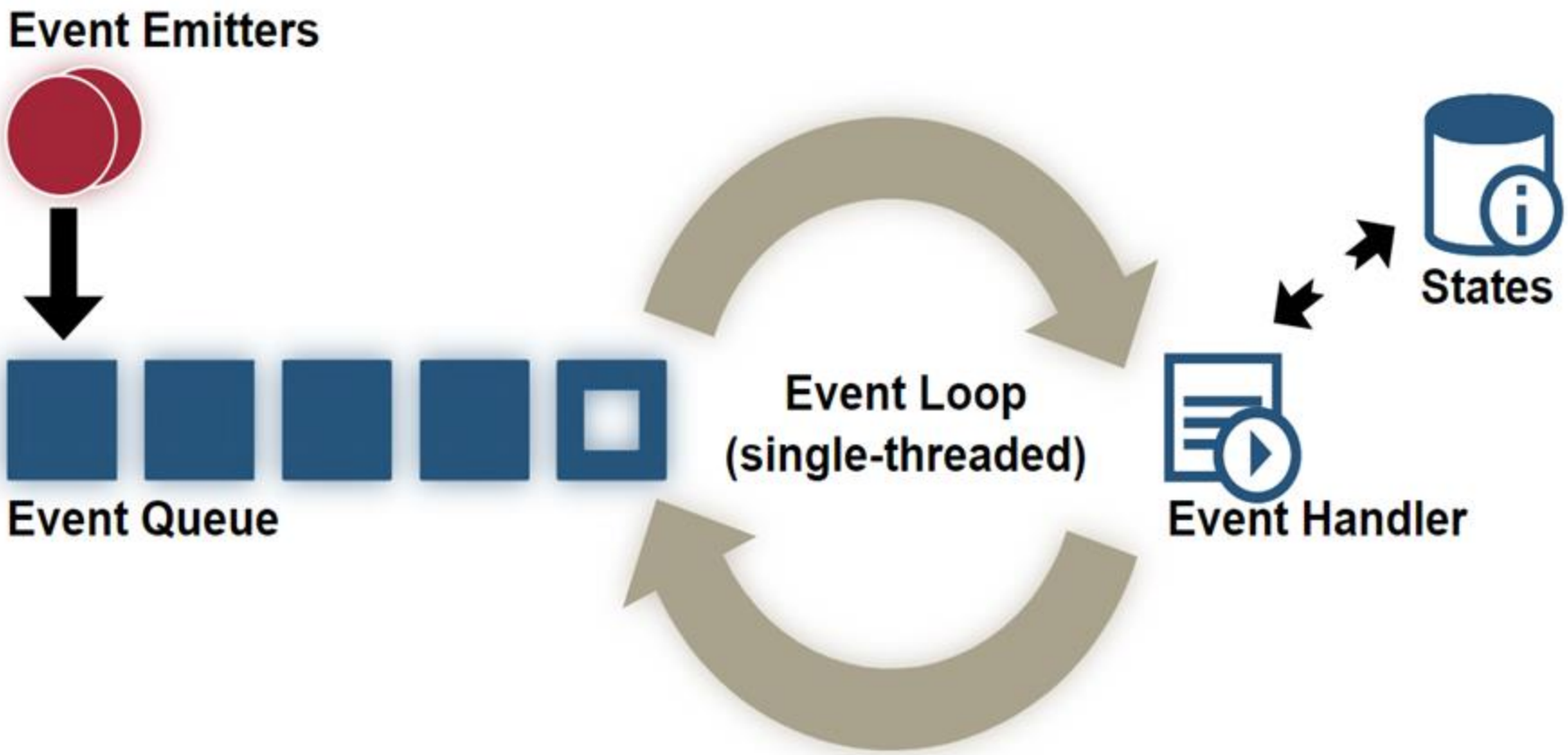
Event Emitter

Example

```
const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();

myEmitter.on('customEvent', () => {
  console.log('Custom event emitted!');
});

myEmitter.emit('customEvent');
```

NODE JS

Events Architecture flow



NODE JS

Event Driven Programming

- Node.js uses events heavily and it is also one of the reasons why Node.js is pretty fast compared to other similar technologies.
- As soon as Node starts its server, it simply initiates its variables, declares functions and then simply waits for the event to occur.
- In an event-driven application, there is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected.

NODE JS

EventEmitter Class

- The functions that listen to events act as **Observers**.
- Whenever an event gets fired, its listener function starts executing. Node.js has multiple in-built events available through events module and EventEmitter class which are used to bind events and event-listeners

```
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

NODE JS

EventEmitter Class

EventEmitter provides multiple properties like **on** and **emit**. **on** property is used to bind a function with the event and **emit** is used to fire an event.

Methods:

- addListener(event, listener)
- on(event, listener) –used to register listeners, adds a listener at the end of the listeners array for specified event
- once(event, listener)
- removeListener(event, listener)
- removeAllListeners([event])
- setMaxListeners(n)
- listeners(event) -Returns an array of listeners for the specified event.
- emit(event, [arg1], [arg2], [...])

EventEmitter Class

Class Methods:

- listenerCount(eventName) -Returns the number of listeners for a given event.

Events:

- newListener

event – String: the event name

listener – Function: the event handler function

This event is emitted any time a listener is added. When this event is triggered, the listener may not yet have been added to the array of listeners for the event.

- removeListener

This event is emitted any time someone removes a listener. When this event is triggered, the listener may not yet have been removed from the array of listeners for the event.

THANK YOU

Vinay Joshi and Dr.Sarasvathi V

Department of Computer Science and Engineering

vinayj@pes.edu

sarsvathiv@pes.edu

Acknowledgement

The slides are created from various internet resources with valuable contributions from multiple professors