



## NODE.JS EVENTS

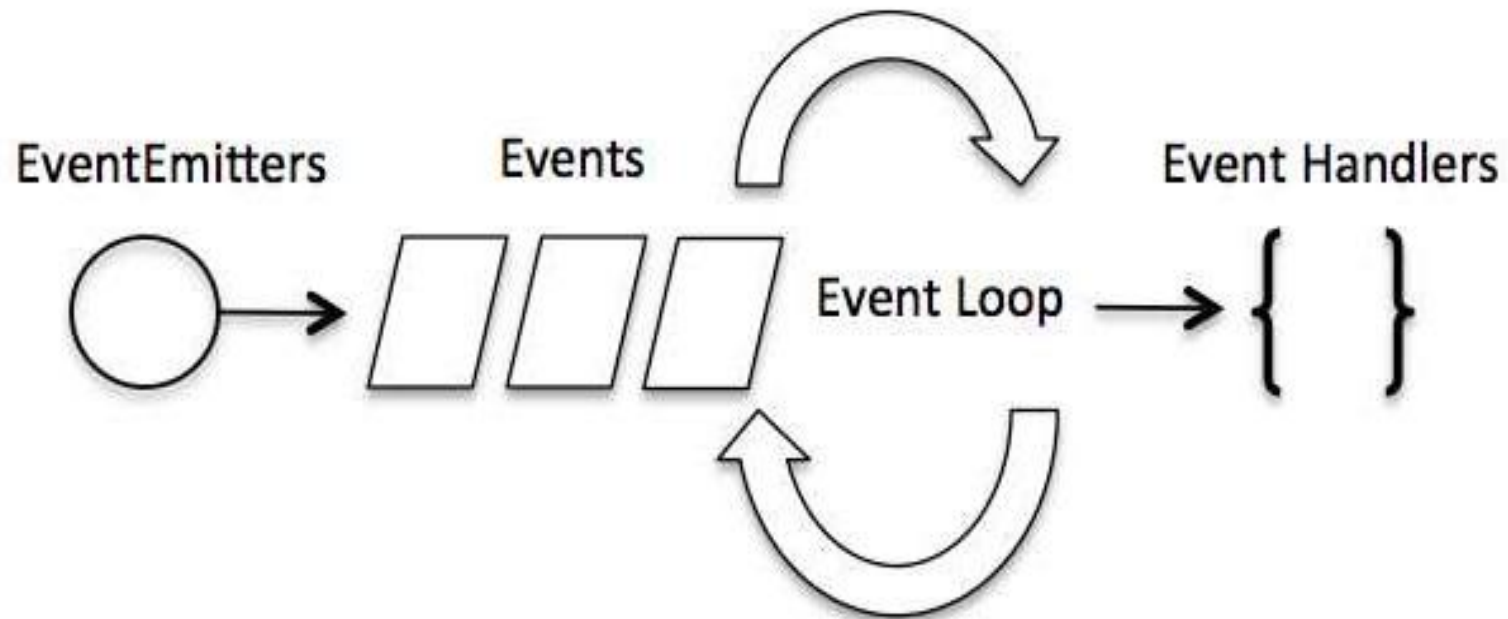
Prof. Rachana V. Modi

# NODE.JS EVENTS

- In Node JS applications, every operation generates an event.
- **Events module:** create, fire and listen own events
- **Syntax:** `var events = require('events');`
- Event components:
  - EventEmitter class
  - Java Script Callback functions



# NODE.JS EVENT DRIVEN PROGRAMMING



# EVENTEMITTER CLASS

## EventEmitter class:

- Available in the events module
- Used to generate events

## Create EventEmitter object:

- **Syntax:**

```
var eventsEmitter = new events.EventEmitter();
```



# EVENTEMITTER CLASS: LISTENING EVENTS

## Listening events:

- Before emits any event, it must register functions (callbacks) to listen the events.
- There are 3 methods for listening events:
  - `addListener(event, listener)`
  - `on(event, listener)`
  - `once(event, listener)`

## EventEmitter “on()” function:

- Used to bind an event with event handler function
- **Syntax:** `eventsEmitter.on(EventName, Listener);`



# EVENTEMITTER CLASS: EMITTING EVENTS

## Emitting events:

- Every event has named event in node.js.
- emit function is used to raise an event.

## EventEmitter “emit()” function:

- Raise the specified events with the supplied arguments
- **Syntax:** eventsEmitter.emit(EventName, [arg1], [arg2], [...]);



## EVENTEMITTER CLASS: EXAMPLE

**Example: Create an event emitter instance and register a couple of callbacks**

```
var events = require('events');  
var eventsEmitter = new events.EventEmitter();  
  
eventsEmitter.on('myevent', (msg) => console.log(msg));  
eventsEmitter.emit('myevent', 'Event-1');  
eventsEmitter.emit('myevent', 'Event-2');
```



# EVENTEMITTER CLASS: REMOVING LISTENER

## Removing Listener:

- It removes that listener from the listeners array that is subscribed to that event.
- There are 2 methods for removing events:
  - `removeListener(event, listener)`
  - `removeAllListeners([event])`
- `removeListener()` method will remove at most one instance of the listener which is in front of the queue.





# EVENTEMITTER CLASS: SETTING AND GETTING MAX LISTENER

## **setMaxListeners(n):**

- Set max listener of particular event.
- Maximum 10 listeners can be registered for any single event.
- if more than 10 listeners are added for a particular event then EventEmitters will print a warning.

## **getMaxListeners():**

- Returns the current maximum listener value for the emitter which is either set by setMaxListeners(n) or defaults to defaultMaxListeners.



# EVENTEMITTER CLASS: SETTING AND GETTING MAX LISTENER

## **defaultMaxListeners:**

- Used to change the default value for all EventEmitter instances.
- **Syntax:**

`EventEmitter.defaultMaxListeners = value;`



# EVENTEMITTER CLASS: METHODS

## **eventNames():**

- Get all the active event names.
- Return type is an array.
- **Syntax:** EventEmitter.eventNames()

## **eventEmitter.listeners():**

- It returns an array of listeners for the specified event.
- **Syntax:** EventEmitter.listeners(event)



# EVENTEMITTER CLASS: METHODS

## **listenerCount():**

- It returns the number of listeners listening to the specified event.
- **Syntax:** EventEmitter.listenerCount(event)

## **prependOnceListener():**

- It will add the one-time listener to the beginning of the array.
- **Syntax:** EventEmitter.prependOnceListener(event, listener)

## **prependListener():**

- It will add the listener to the beginning of the array.
- **Syntax:** EventEmitter.prependListener(event, listener)



# EVENTEMITTER CLASS: SPECIAL EVENTS

## **newListener:**


- The newListener event is emitted before a listener is added to the internal array of listeners.
- Any EventEmitter instance will emit its own 'newListener' event.
- **Syntax:** EventEmitter.on( 'newListener', listener)

## **removeListener:**


- The removeListener event is emitted after a listener is removed.
- It used to stop event listener functions from listening to events.
- **Syntax:** EventEmitter.on( 'removeListener', listener)



# IN-BUILT EVENTS

1. **'data' event:** This event is emitted by readable streams when data is available to be read. It's commonly used with streams to read chunks of data.
  2. **'end' event:** This event is emitted by readable streams when there is no more data to be read. It indicates the end of the data stream.
  3. **'error' event:** This event is emitted whenever an error occurs. It's used to handle errors that might occur during the execution of a program.
  4. **'close' event:** This event is emitted when a stream or a socket is closed. It can be used to perform cleanup tasks when a resource is no longer needed.
  5. **'connect' event:** This event is emitted by client sockets when a connection is successfully established with a server.
- 

# IN-BUILT EVENTS

6. **'request' event:** In the context of the built-in HTTP server, this event is emitted whenever an HTTP request is made to the server. It's used to handle incoming HTTP requests.
  7. **'response' event:** This event is emitted by client HTTP requests when a response is received from the server.
  8. **'listening' event:** This event is emitted by servers when they start listening on a specific port for incoming connections.
  9. **'message' event:** This event is emitted by child processes (created using the `child_process` module) when they send messages to the parent process using the `process.send()` method.
  10. **'exit' event:** This event is emitted when the Node.js process is about to exit, either due to an error or because the `process.exit()` method is called.
- 

Any query??

