

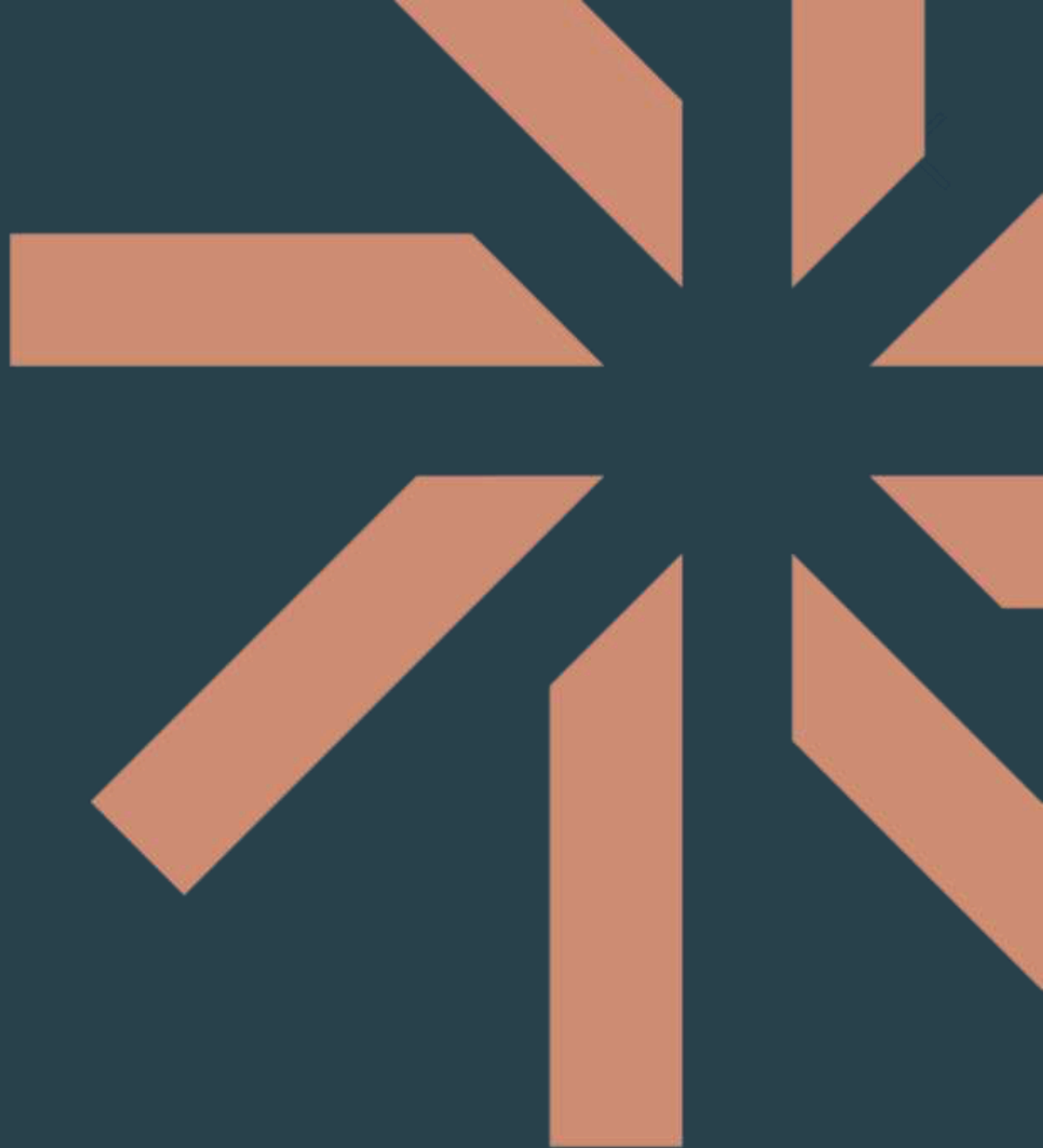
Events



Qinshift 
Academy

WHAT ARE EVENTS?

- Events are things that happen in the system you are programming, which the system tells you about so your code can react to them. For example, if the user clicks a button on a webpage, you might want to react to that action by displaying an information box.
- An event is a change in state, or an update, like an item being placed in a shopping cart on an e-commerce website.



THE EVENT EMITTER



- All objects that emit events are instances of the EventEmitter class. The event can be emitted or listen to an event with the help of EventEmitter.
- The EventEmitter is a Node module that allows objects to communicate with one another. The core of Node's asynchronous event-driven architecture is EventEmitter. Many of Node's built-in modules inherit from EventEmitter.

THE EVENT EMITTER



The idea is simple – emitter objects send out named events, which trigger listeners that have already been registered. Hence, an emitter object has two key characteristics:

- Emitting name events: The signal that something has happened is called emitting an event. A status change in the emitting object is often the cause of this condition.
- Registering and unregistering listener functions: It refers to the binding and unbinding of the callback functions with their corresponding events.

EVENT LISTENERS & TRIGGERS



All the objects from the EventEmitter class expose an `eventEmitter.on()` function that allows one or more functions to be attached to named events emitted by the object. Typically, event names are camel-cased strings but any valid JavaScript property key can be used.

When the EventEmitter object emits an event, all of the functions attached to that specific event are called synchronously. Any values returned by the called listeners are ignored and discarded.

EVENT LISTENERS & TRIGGERS



The following example shows a simple EventEmitter instance with a single listener. The `eventEmitter.on()` method is used to register listeners, while the `eventEmitter.emit()` method is used to trigger the event.

```
import { EventEmitter } from 'node:events';

class MyEmitter extends EventEmitter {}


const myEmitter = new MyEmitter();
myEmitter.on('event', () => {
  console.log('an event occurred!');
});
myEmitter.emit('event');
```

Passing arguments and 'this' to listeners



The `eventEmitter.emit()` method allows an arbitrary set of arguments to be passed to the listener functions. Keep in mind that when an ordinary listener function is called, the standard `this` keyword is intentionally set to reference the `EventEmitter` instance to which the listener is attached.

```
import { EventEmitter } from 'node:events';
class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();
myEmitter.on('event', function(a, b) {
  console.log(a, b, this, this === myEmitter);
  // Prints:
  //   a b MyEmitter {
  //     _events: [Object: null prototype] { event: [Function (anonymous)] },
  //     _eventsCount: 1,
  //     _maxListeners: undefined,
  //     [Symbol(kCapture)]: false
  //   } true
});
myEmitter.emit('event', 'a', 'b');
```

It is possible to use ES6 Arrow Functions as listeners, however, when doing so  the `this` keyword will no longer reference the EventEmitter instance.

```
import { EventEmitter } from 'node:events';  
class MyEmitter extends EventEmitter {}  
const myEmitter = new MyEmitter();  
myEmitter.on('event', (a, b) => {  
  console.log(a, b, this);  
  // Prints: a b {}  
});  
myEmitter.emit('event', 'a', 'b');
```


EVENT-DRIVEN PROGRAMMING



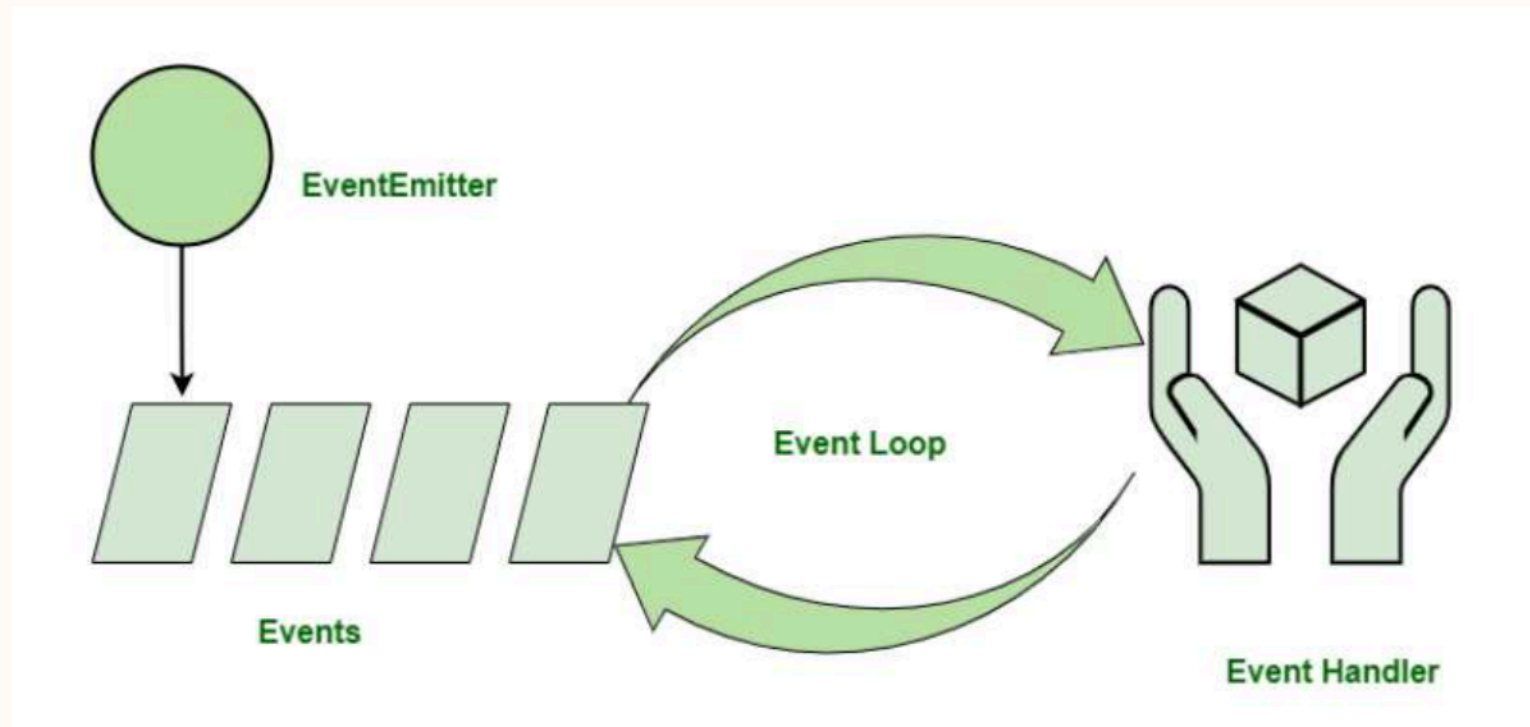
Event-driven programming is a programming paradigm in which the flow of program execution is determined by events - for example a user action such as a mouse click, key press, or a message from the operating system or another program. An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate event-handling procedure. Event-driven programs can be written in any programming language which makes this a universal concept.

Event-Driven Programming in Node.js



Event-driven programming is used to synchronize the occurrence of multiple events and to make the program as simple as possible. The basic components of an Event-Driven Program are:

- A callback function (called an event handler) is called when an event is triggered.
- An event loop that listens for event triggers and calls the corresponding event handler for that event.



Event-Driven Programming in Node.js



A function that listens for the triggering of an event is said to be an 'Observer'. It gets triggered when an event occurs. Node.js provides a range of events that are already in-built. These 'events' can be accessed via the 'events' module and the EventEmitter class. Most of the in-built modules of Node.js inherit from the EventEmitter class.

Event-Driven Programming Principles



- A suite of functions for handling the events. These can be either blocking or non-blocking, depending on the implementation.
- Binding registered functions to events.
- When a registered event is received, an event loop polls for new events and calls the matching event handler(s).

EXERCISE



- Instantiate an EventEmitter object
- Register one event handler
- Trigger the event

QUESTIONS?

You can find us at:

anetastankovskaane@gmail.com

igorveic7@gmail.com

