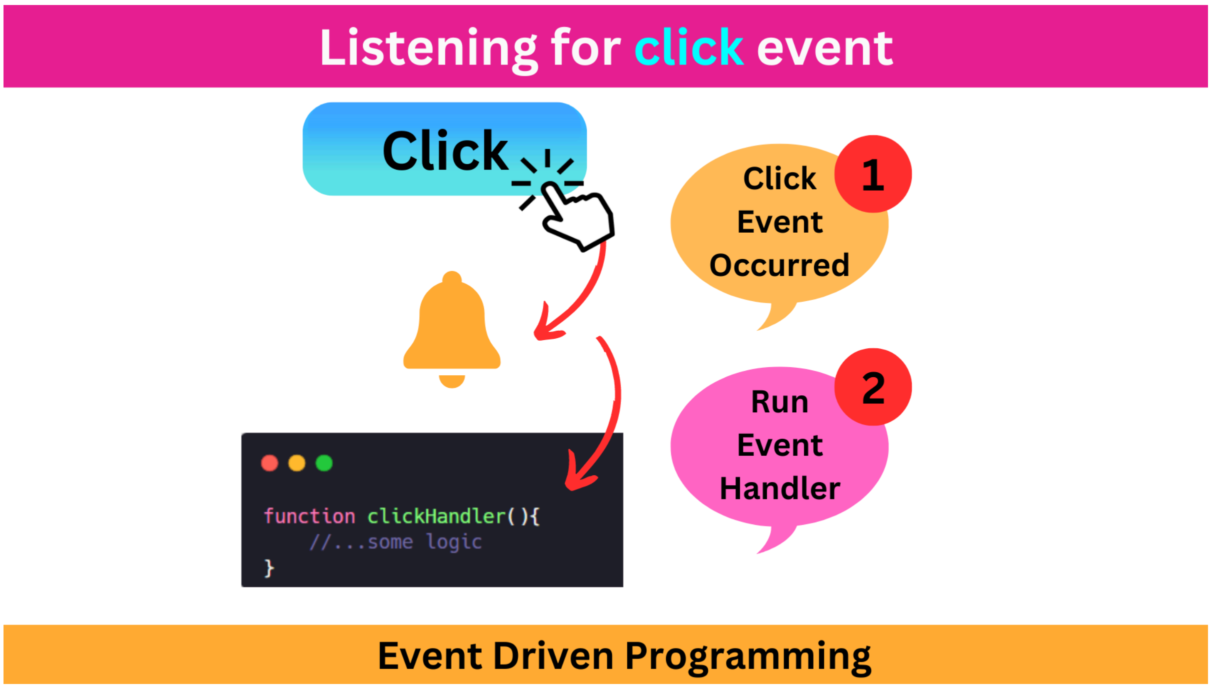
Event-Driven Programming

Event-driven programming is a programming paradigm where program flow is largely determined by events or user actions, rather than by the program's logic.

In this type of programming, the program listens for events, and when they occur, it executes some code/function that should run in response to that event.

An event could be anything from a mouse click or a button press to the arrival of new data in the system.

The below image demonstrates how event driven programming works. In this form of programming, we write some code which constantly listens for a particular event and once that event occurs, we run some code in response to it.



```
function clickHandler(){
    //...some logic
}
```

In this section of the tutorial we will be learning about events in NodeJS. Although we may not be using events directly for our day-to-day coding tasks, but a lot of NodeJS modules use the concept of events under the hood. This is why it becomes important to be aware about it.

To implement Event Driven Programming in NodeJS, You need to remember 2 things:

- There is a function called `emit()` which causes an event to occur.
  For example, `emit('myEvent')` emits/causes an event called `myEvent`.

- There is another function called `on()` which is used to listen for a particular event and when this event occurs, the `on()` method executes a listener function in response to it. For example, Consider this code: `on('myEvent', myFunction)`: Here we are listening for an event called `myEvent` and when this event takes place, we run the `myFunction` listener function in response to it.

We can access the `on()` and `emit()` functions by creating an instance of the `EventEmitter` class. The `EventEmitter` class can be imported from a built-in package called `events`.

In the below code, we are listening for the `userJoined` event and once this event takes place, we run the `welcomeUser()` function using the `on()` method and we emit the `userJoined` event using the `emit()` method:

```
// Importing 'events' module and creating an instance of the EventEmitter Class
const EventEmitter = require('events');
const myEmitter = new EventEmitter();

// Listener Function - welcomeUser()
const welcomeUser = () => {
    console.log('Hi There, Welcome to the server!');
}

// Listening for the userJoined event using the on() method
myEmitter.on('userJoined', welcomeUser);

// Emitting the userJoined event using the emit() method
myEmitter.emit('userJoined');
```

Points to Note:

There are 3 points you should note while working with events in Node.

Each point in shown in action in the corresponding code snippets:

- There can be multiple `on()`'s for a single `emit()`:

Check out the following code, where multiple `on()` functions are listening for a single event to happen (`userJoined` event) and when this event is emitted in the last line of the code using the `emit()` function, you will see that the all the listener functions which were attached to the `on()` function get's executed:

```javascript
// Importing `events` module and creating an instance of EventEmitter class
const EventEmitter = require('events');
const myEmitter = new EventEmitter();

// Listener Function 1: sayHello
const sayHello = () => {
    console.log('Hello User');
}

// Listener Function 2: sayHi
const sayHi = () => {
    console.log('Hi User');
}

// Listener Function 3: greetNewYear
const greetNewYear = () => {
    console.log('Happy New Year!');
}

// Subscribing to `userJoined` event
myEmitter.on('userJoined', sayHello);
myEmitter.on('userJoined', sayHi);
myEmitter.on('userJoined', greetNewYear);

// Emiting the `userJoined` Event
myEmitter.emit('userJoined');
```

You can think of it this way: Each time the `userJoined` event is emitted, a notification is sent to all the `on()` functions listening for the event and then all of them will run their corresponding listener functions: `sayHello`, `sayHi`, `greetNewYear`.

Thus, when you run the code, you will see the following output printed in the console:

```
Hello User
Hi User
Happy New Year!
```

- The `emit()` can also contain arguments which will be passed to the listener functions:

In the following code, We are using the `on()` method to subscribe to an event called `birthdayEvent` and when this event is emitted, we run the `greetBirthday()` function in response to it.

The extra parameters mentioned in the `emit()` function, gets passed as parameters to all the listener functions which will run in response to the `birthdayEvent`. Therefore `John` and `24` gets passed as parameters to the `greetBirthday()` function.

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

// Listener function
const greetBirthday = (name, newAge) => {
    // name = John
    // newAge = 24
    console.log(`Happy Birthday ${name}. You are now {newAge}!`);
}

// Listening for the birthdayEvent
myEmitter.on('birthdayEvent', greetBirthday);

// Emitting the birthdayEvent with some extra parameters
myEmitter.emit('birthdayEvent', 'John', '24');
```

The following output will be seen printed in the console: `Happy Birthday John, You are now 24!`.

- The `emit()` function should always be defined after the `on()` function(s):

The entire process of communication between `on()` and `emit()` works like this: Before emitting any event, you need to make sure that all the listener functions have subscribed/registered to that event. Any function which is registered as a listener after the event has been emitted, will not be executed.

Check out the following code where this process is studied in detail:

```javascript
const EventEmitter = require('events');
const myEmitter = new EventEmitter();

// Listener Function 1 - sayHi
const sayHi = () => {
    console.log('Hi User');
}

// Listener Function 2 - sayHello
const sayHello = () => {
    console.log('Hello User');
}

// Registering sayHi function as listener
myEmitter.on('userJoined', sayHi);

// Emitting the event
myEmitter.emit('userJoined');

// Registering sayHello function as listener
myEmitter.on('userJoined', sayHello);
```

When the above code is executed, we see `Hi User` printed in the console but `Hello User` is not printed.

The reason behind this is: First we had registered the `sayHi` function as the listener using `myEmitter.on('userJoined', sayHi)`. Next we emit the `userJoined` event which results in execution of the `sayHi` function. In the next line we are registering the `sayHello` function as the listener, but it's already too late to do this because `userJoined` event has been emitted. This signifies the importance of defining all the `on()` functions before we emit the event using `emit()`.

You can also think of it this way: When you use `emit()` to trigger an event, NodeJS looks for any corresponding `on()` methods that have been defined in your code above the `emit()` method. If it finds any, it will execute them in order to handle the event.