**Node.js-Event Loop**

Node.js is a single-threaded application, however events and callbacks also make Node.js support concurrency.

**Event-driven Programming**

When a user does something like for example, click a button in a webpage, an event is triggered. These events have functions that when triggered, executes something.

<img src="Node-Event-Loop.png" alt="Node-Event-Loop">

There is a main loop that listens for events and trigger a callback function for the event that was triggered. When an asynchronous function returns its results, a callback function is called.

There are events used to bind events and event-listener. They are available through events module and EventEmiiter class.

<img src="Node1.png" alt="code1">

Syntax in binding an event handler with an event:

<img src="Node2.png" alt="code2">

 Firing an event programmatically:

<img src="Node3.png" alt="code3">

EXAMPLE

//main.js

<img src="Node4.png" alt="sample code">

**Node.js-EventEmitter**

A module that helps in incorporating Event-Driven Programming in projects. The events module contains the EventEmitter class.

<img src="Node5.png" alt="sample code">

There is an ‘error’ event that is triggered when an EventEmitter faces an error. A ‘newListener’ event is triggered when a new listener is added. Lastly, a removeListener event is fired when a listener is removed.

Methods

1. **addListener(event, listener)**

2. **on(event, listener)**

3. **once(event, listener)**

4. **removeListener(event, listener)**

5. **removeAllListeners([event])**

6. **setMaxListeners(n)**

7. **listeners(event)**

8. **emit(event, [arg1], [arg2], [...])**

Class methods

1. **listenerCount(emitter, event)**

Events

1. **newListener**

* **event –** the name of the event
* **listener -**  the event handler function

2. **removeListener**

* **event -** the name of the event
* **listener -**  the event handler function

**EXAMPLE**

<img src="Node6.png" alt="sample code">

Node.js – File System

This allows the user to work with the computer’s file system. To implement a file system, use the require() method.

Example:

var filesystem = require(‘fs’);

It is commonly use for Reading files, Creating files, Updating files, Deleting files and Renaming files.

Methods: \*\*\*\*\*\*\*\*Itable nalangs these

Read Files

-fs.readFile()

Create Files

-fs.appendFile()

-fs.open()

-fs.writeFile()

Update Files

-fs.appendFile()

-fs.writeFile()

Delete Files

-fs.unlink()

Rename Files

-fs.rename