Event Loop in Node.js

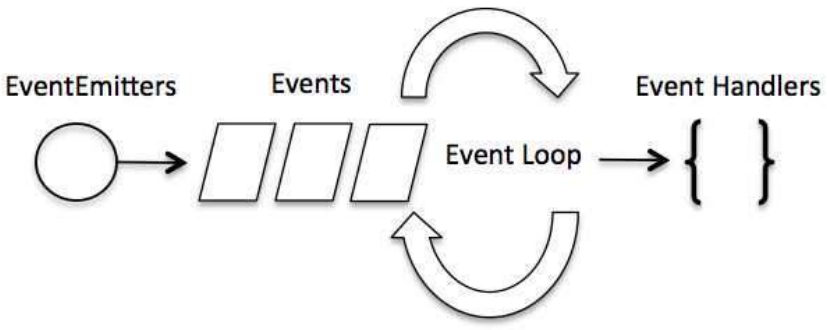
Node.js is a single thread application, but it supports concurrent processing through event definitions and callbacks. As all Node.js APIs are asynchronous and single threaded, it uses the async function to maintain concurrency. Node.js uses the Pattern Observer.

Node.js is a single thread application, but it supports concurrent processing through event definitions and callbacks. As all Node.js APIs are asynchronous and single threaded, it uses the async function to maintain concurrency. Node.js uses the Pattern Observer. The threads in Node.js hold an Event Loop and whenever any task completes, it will trigger the corresponding event to alert the Event Listener to be ready .

Event Driven programming in Node.js

Node.js uses a lot of events, which is why Node.js is quite fast compared to other technology products. As soon as the node starts its server, it will quickly initialize variables, declare functions, and then simply wait for events to occur.

In the event handler application, the main loop generally listens to events, and then triggers the callback function when one of the events is detected.



While events are quite similar to callback functions. The difference is that the callback function calls when a function is out of sync and returns its result while the event handler works on the Pattern Observer. This function will listen to events, acting as an Observers. Whenever an event arises, its Listener functions will start executing. Node.js has many events available through **events** Module and **EventEmitter** class can rely on bind events and listen to events.

Before using the event module, you use the require () method to declare the following:

```
// Khai bao events module var  events  = require ( 'events' ); // Tao mot doi tu
```

Then, to bind the Event Handler to an event, use the following syntax:

```
// Gan ket event voi Event Handler nhu sau:  eventEmitter . on ( 'eventName' ,
```

You can activate an event using the emit () method of EventEmitter:

```
// Kich hoat mot event   eventEmitter . emit ( 'eventName' );
```

Example of Event Loop in Node.js

Create a js file named main.js with the following code:

```
// Declare events module var events = require ( 'events' );  // Create a doi ev
```

Run the above program as follows:

```
$ node main . js
```

The result is:

```
I don't know what to do!
Travel to find the best.
Let's get married.
```

How Node.js application works

In the Node.js application, an asynchronous function accepts a callback as the last parameter and the callback function accepts the error as the first parameter. Let's review the previous example. Create a text file named input.txt with the following content:

```
QTM  la trang  Web  huong dan cac bai lap trinh hoan toan mien phi cho tat ca mo
```

In this example, I use fs Module to handle File I / O operations (I'll cover in the next chapter). First, create a js file named main.js as follows:

```
var fs = require ( "fs" );  fs .  readFile ( 'input.txt' , function ( err , da
```

Here, fs.readFile () is an asynchronous function for the purpose of reading files. If there is an error while reading the file, the err object will contain the error, otherwise the data will contain the contents of the file. The readFile function transmits err and data to the callback function after the file reading process has completed, and finally prints the content.

```
Let's get married
QTM is a Web page that contains all the scripts
Welcome to the world !!!!!!
```

According to Tutorialspoint

Previous article: Concept of Callbacks in Node.js

The following article: Event Emitter in Node.js