Introduction to NodeJS

Objective

- Have a basic understanding of NodeJS.
- Understand event loop & callback concepts in NodeJS.
- Know how to create your first application in NodeJS

Agenda

- What is NodeJS?
- Who uses NodeJS?
- Event loop
- Event-driven programming

What is NodeJS

- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009
- Feature of NodeJS
  - Asynchronous and Event Driven
  - Single Threaded but Highly Scalable
  - No Buffering

Who uses Node.js

Who uses Node.js

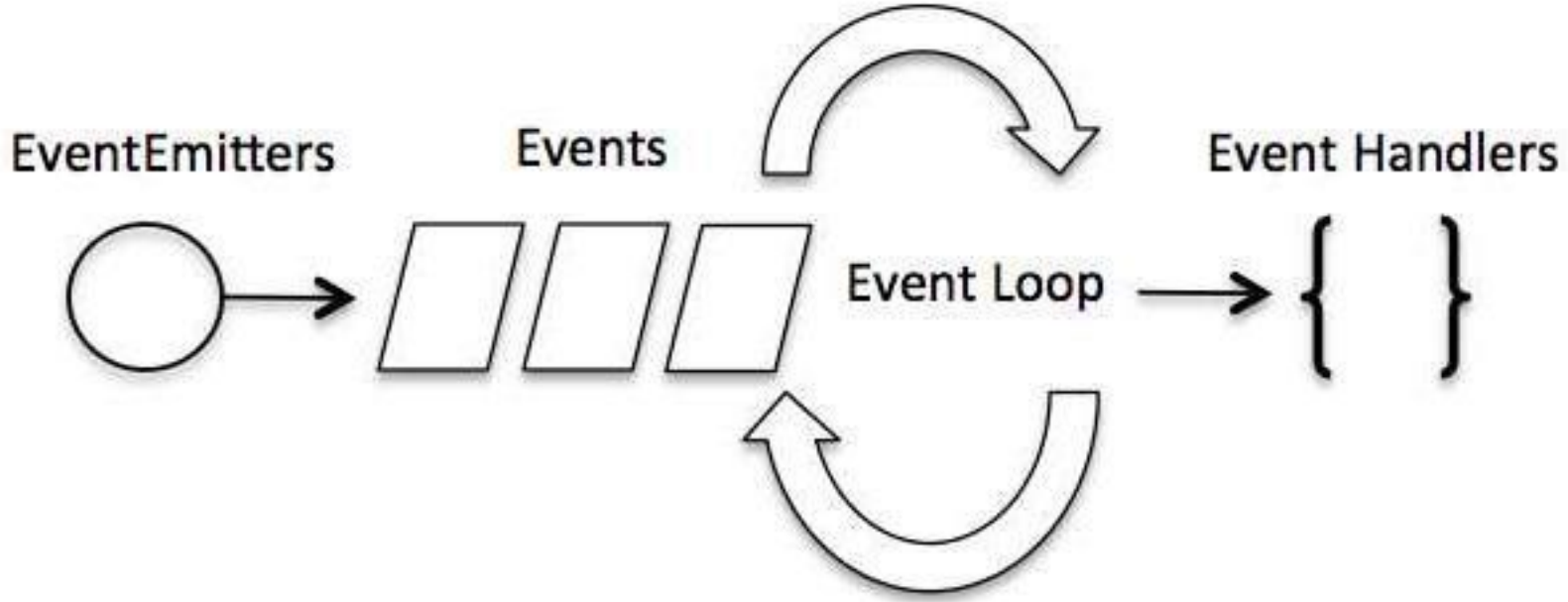
Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Event loop

- Node.js is a single-threaded application, but it can support concurrency via the concept of **event** and **callbacks**.
- Node uses observer pattern
- Node thread keeps an event loop and whenever a task gets completed, it fires the corresponding event which signals the event-listener function to execute.

Event-Driven Programming



Code demo

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();

// Create an event handler as follows
var connectHandler = function connected() {
   console.log('connection succesful.');

   // Fire the data_received event
   eventEmitter.emit('data_received');
}

// Bind the connection event with the handler
eventEmitter.on('connection', connectHandler);

// Bind the data_received event with the anonymous function
eventEmitter.on('data_received', function(){
   console.log('data received succesfully.');
});

// Fire the connection event
eventEmitter.emit('connection');

console.log("Program Ended.");
```

Code demo

Now let's try to run the above program and check its output –

- *$ node main.js*

The following result –

connection successful.

data received successfully.

Program Ended.

Event Emitter

- Create a js file named main.js with the following Node.js code –

- Now run the main.js to see the result
  - `$ node main.js`

Callbacks Concept

- Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

Blocking Code Example

```
var fs = require("fs");
var data = fs.readFileSync('input.txt');
console.log(data.toString());
console.log("Program Ended");
```

- Now run the main.js to see the result
  - *$ node main.js*

Non-Blocking Code Example

```
var fs = require("fs");
fs.readFile('input.txt', function (err, data) {
    if (err) return console.error(err);   console.log(data.toString());
});
console.log("Program Ended");
```

- Now run the main.js to see the result –

  $ node main.js

First Application

- **Import required modules** – We use the **require** directive to load Node.js modules.
- **Create server** – A server which will listen to client's requests similar to Apache HTTP Server.
- **Read request and return response** – The server created in an earlier step will read the HTTP request made by the client which can be a browser or a console and return the response.

Code sample

```
var http = require("http");

http.createServer(function (request, response) {

   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World\n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Thank you