**NODE JS**

**What Is Node JS?**

* Node.js is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser.
* It is used for creating server-side web applications.
* Node.js is perfect for data-intensive applications as it uses an asynchronous, event-driven model.
* Node.js was developed by **Ryan Dahl in 2009** and its latest version is v21

**Why Is Node JS?**

* Node.js is really fast.
* Node Package Manager has over 50,000 bundles for at the developers' disposal.
* Perfect for data intensive, real-time web applications as Node.js never waits for an API to return data.
* Better synchronization of code between server and client due to same code base.
* Easy for web developers to start using Node.js in their projects as it is a JavaScript library.

**Parts Of Node JS:**

1. **MODULES**:

* Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions.
* To include a module in a Node.js application.
* Use require() function with the parenthesis containing the name of the module.

// CREATING A WEB SERVER

// Include modules

var http = requireChttp');

var server =

http.createServer(function(req. res)(

//write your code here

server.listen(2000);

Node.js has many modules to provide basic functionality needed for a web application

Some of them are mentioned in this table :

|  |  |
| --- | --- |
| Core Modules | Description |
| HTTP | Includes classes. methods and events to create Node.js http server |
| UTIL | Includes utility functions useful for developers |
| FS | Includes events, classes. and methods to deal with file 1/0 operations |
| URL | Includes methods for URL parsing |
| Querystring | Includes methods to work with query string |
| Stream | Includes methods to handle streaming data |
| Zlib | Includes methods to compress or decompress files |

1. **CONSOLE**:

* Console is a module that provides a way to debug similar to that of the JavaScript console provided by the internet browsers.
* It prints messages to stdout and stderr.

1. **CLUSTER**:

* Node.js is built upon the concept of single-threaded programming.
* Cluster is a module that allows multi-threading by creating child processes that share the same server port and run simultaneously.
* A cluster can be added to an application in this way:

var cluster = require('cluster'):

if (cluster.isWorker) {

console.log('Child thread');

} else {

console.logCParent thread');

cluster.fork():

cluster.fork();

1. **GLOBAL**:

* Global objects in Node.js are available in all modules.
* These objects are functions, modules, strings. etc.
* Some Node.js global objects are mentioned in the table:

|  |  |
| --- | --- |
| Global Objects | Description |
| Dirname | Specifies the name of the directory that contains the code of application |
| Filename | Specifies the filename of the code |
| Exports | A reference to the module.exports, shorter to type |
| Module | A reference to the current module |
| Require | Used to import modules, local files, and also JSON |

1. **ERROR HANDLING:**

* Errors in Node.js are handled through exceptions.

try {

var m =

var n = 1/0;

catch (err) {

// Handling the error here.

}

1. **STREAMING**:
   * Streams are objects that let you read data or write data continuously.
   * There are four types of streams:

* READABLE: Streams from which data can be read.
* WRITABLE: Streams to which data can be written.
* DUPLEX: Both readable and writeable streams.
* TRANSFORM: Streams that can manipulate the data while it is being read or written.

1. **DOMAIN**:

* Domain module intercepts errors that remain unhandled.
* Two methods for intercepting:
* INTERNAL BINDING: Error emitter executes its code inside the run method.
* EXTERNAL BINDING: Error emitter is explicitly added to a domain via its add method.

1. **DNS**:

* DNS module is used to connect to a DNS server and perform name resolution.
* DNS module can also be used for performing name resolution without a network communication.

dns.lookup()

1. **DEBUGGER**:

* Node.js includes a debugging utility that can be accessed by a built-in debugging client.
* Node.js debugger is not feature-packed, but supports simple inspection of code.
* Debugger can be used in the terminal by using the 'inspect' keyword before the name of the JavaScript file.

$ node inspect myscript.js

**NODE.JS EXPRESS FRAMEWORK:**

* Express is a flexible Node.js web application framework which provides a wide set of features to develop both web and mobile applications.
* Let's see some of the core features of Express framework:
* It can be used to design single-page, multi-page and hybrid web applications.
* It allows to setup middlewares to respond to HTTP requests.
* It defines a routing table which is used to perform different actions based on HTTP method and URL.
* It allows to dynamically render HTML Pages based on passing arguments to templates.
* Since Node.js is a JavaScript framework, the development files have a ".js" extension.

var express = require('express'); ---> //Importing Express framework into our Node.js application.

var app = express();

app.get('/'. function (req. res) {

res.send('Hello World'); ---> //Callback function with parameters 'request' and 'response'.

})

var server = app.Iisten(8081. function () {

var host = server.address().address ---> //Application will listen on the defined port which in this case is "8081", and variables 'host' and 'port' will contain the address and the port respectively.

var port = server.address().port

console.IogCExample app listening at http://%s:%s", host. port) ---> // This snippet is to show the address and port in the command prompt or terminal.

})

* + The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.
  + The response object represents the HTTP response that an Express app sends when it gets an HTTP request.

OUTPUT :

Command Prompt - node Hello\_world.js

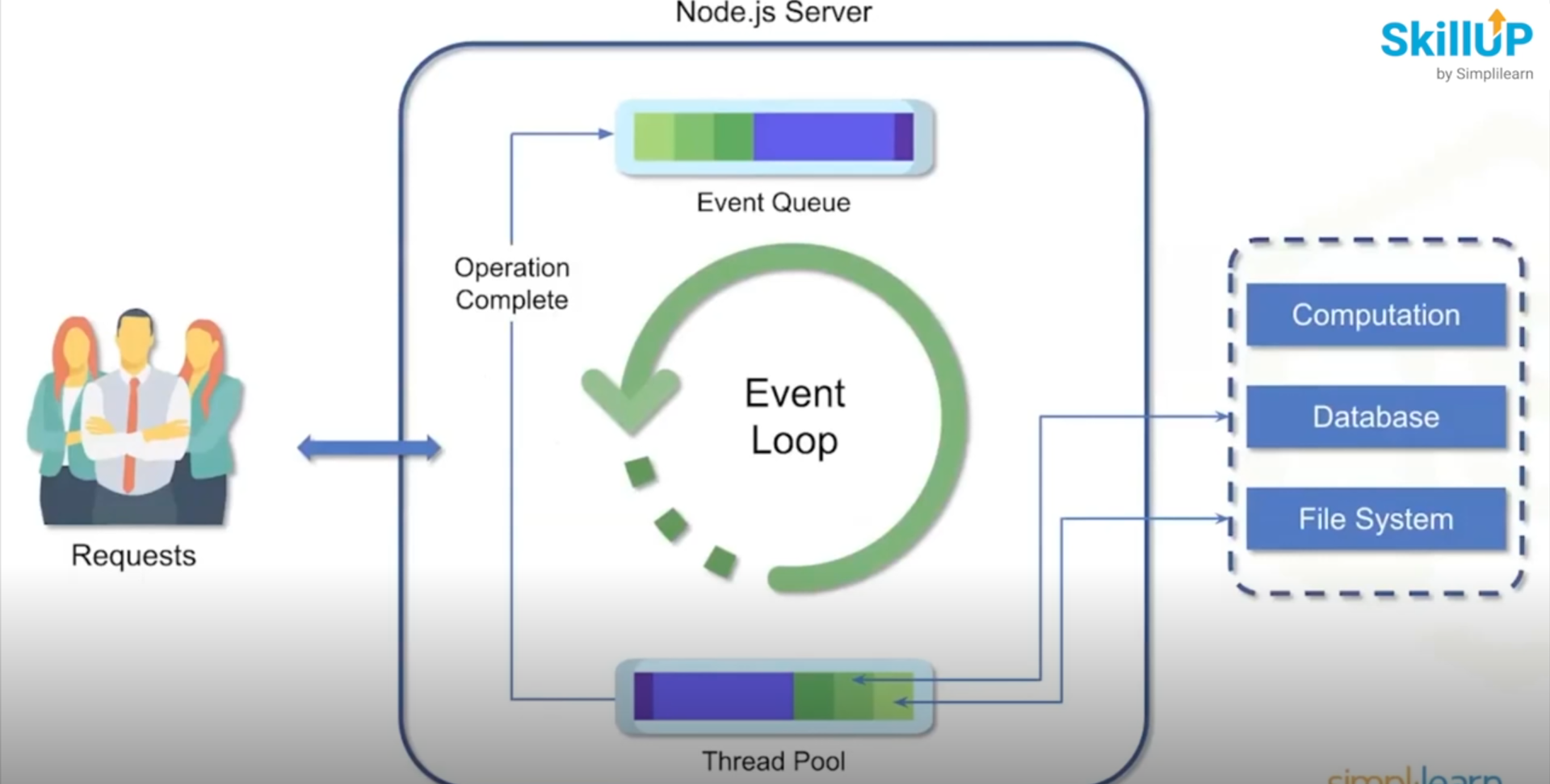
C: Desktop

C:\Users\Taha\Desktop>node Hello\_world.js Example app listening at http://:::8081 ---> //Command prompt shows the address and port.

And finally, the output is displayed on web browser.

**NODE.JS ARCHITECTURE:**

* Node.js uses "Single Threaded Event Loop" architecture to handle multiple concurrent clients.
* Node.js Processing model is based on the JavaScript Event based model along with the JavaScript callback mechanism.



Node.js is a server-side platform that takes requests from users, processes those requests and returns responses to the corresponding users.

**REQUEST**:

* Clients send request to Web Server
* Requests can be :
* querying for data
* deleting data
* updating the data, etc.

**EVENT QUEUE:**

* Node.js receives request places them into the Event Queue.

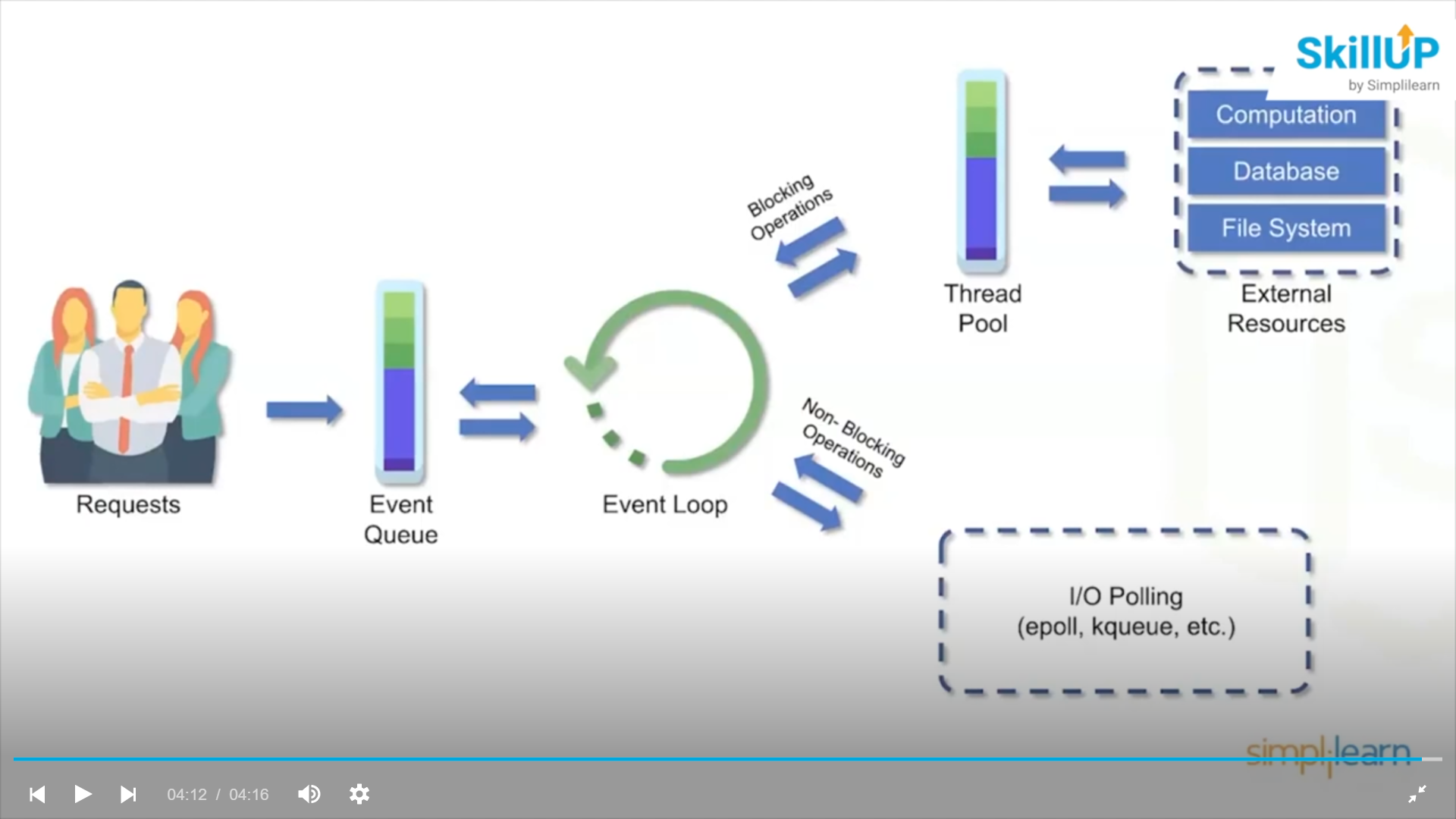
**EVENT LOOP:**

* Event Loop indefinitely receives requests and processes them.

**THREAD POOL:**

* Node.js internally maintains a Thread Pool.

External components such as database, file system, and computation.



**WHAT IS NPM?**

* NPM stands for Node Package Manager.
* NPM is responsible for managing all the packages and modules for Node.js.

**Node Package Manager provides two main functionalities:**

* It provides online repositories for node.js packages/modules which are searchable on search.nodejs.org.
* It also provides command line utility to install Node.js packages, does version management and dependency management of Node.js packages.

**Why NPM?**

* Helps in adding the required packages into a project.
* Manages and maintains various versions of codes and their dependencies.
* NPM automatically updates the application when the project code undergoes any change.

**NPM INSTALLATION:**

* Node Package Manager is included with the installation Of Node.js.
* Download Node.js from <https://nodeis.orq/en/download/>
* Run the downloaded Node.js installer and accept the default settings.
* Verify if Node.js is properly installed using command prompt: node –version.
* NPM gets installed alongside, verify if it is properly installed using command prompt: npm –version.

**PACKAGE.JSON FILE:**



* The package.json file is the heart of a Node.js system.
* This file holds the metadata about a particular project.
* Package.json is present in the root directory of any Node application or module.
* package.json provides the Node package manager with the necessary information to understand how the project should be handled along with its dependencies.



{



"name": "node-npm",

"version": "1.0.0",🡪 Version Of Our Projects

"description": "A demo application",

"main": "index.js", 🡪 package.json File

"scripts": {



"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "Taha",

"license": "ISC"

}



* Package.json file after creating a Node.js project using the command: npm init.
* You can edit these parameters at the time of creation of a Node.js project.

**PACKAGE INSTALLATION:**

In Command Prompt with our file working directory:

* Step 1: npm init -y 🡪 To Install Package.JSON File .
* Step 2: npm install express 🡪 To install Express JS

**EXPRESS JS:**

**What is Express JS?**

* Express is a flexible Node.js web application framework which provides a wide set of features to develop both web and mobile applications.
* Express JS makes it much easier and simpler to build a web server with the use of middleware that can handle requests and responses.

**Express JS Features?**

Let's see some of the core features of Express framework:

* It can be used to design single-page, multi-page and hybrid web applications.
* It allows to setup middlewares to respond to HTTP Requests.
* It defines a routing table which is used to perform different actions based on HTTP method and URL.
* It allows to dynamically render HTML Pages based on passing arguments to templates.

**Example:**



var express = require('express'); 🡪 importing Express framework into our Node.js application

var app = express();

app.get('/', function (req, res) { 🡪 Callback function with parameters 'request' and 'response'

res.send('He110 World');

})



var server = app.listen(8081, function ( ) {

var host = server.address().address 🡪 Application will listen on the defied port which in this case is "8081", and variables 'host' and 'port' will contain the address and the port respectively

var port = server.address().port

Console.log(“Example app listening at http://

%s:%s", host, port) 🡪This snippet is to show the address and port in the command

}) prompt or terminal



* The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on.
* The response object represents the HTTP response that an Express app sends when it gets an HTTP request.

**OUTPUT IN CMD:**

