Node.js Introduction

- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

Why Node.js?

A common task for a web server can be to open a file on the server and return the content to the client.

- Here is how PHP or ASP handles a file request:
- Sends the task to the computer's file system.
- Waits while the file system opens and reads the file.
- Returns the content to the client.
- Ready to handle the next request.

Why Node.js?

Here is how Node.js handles a file request:

- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.
- Node.js eliminates the waiting, and simply continues with the next request.
- Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient.

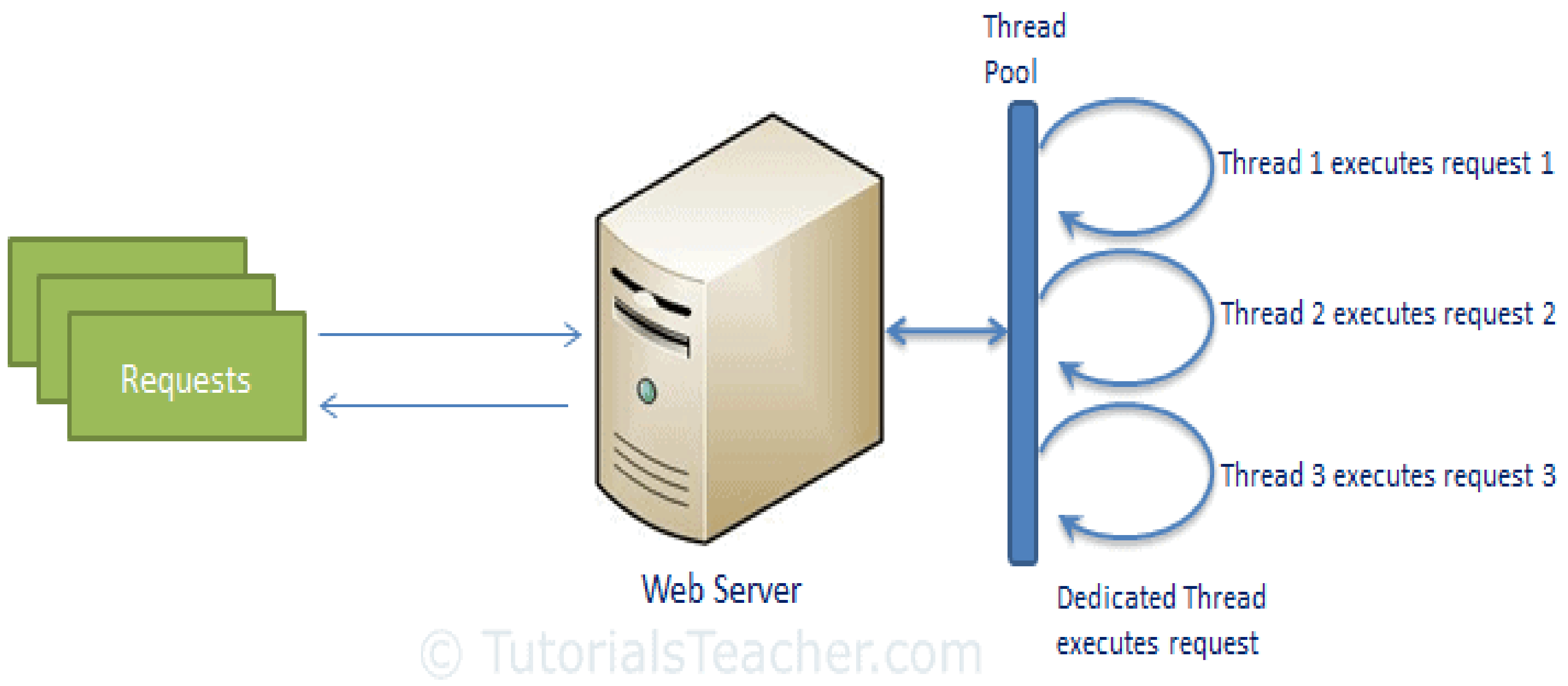
What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

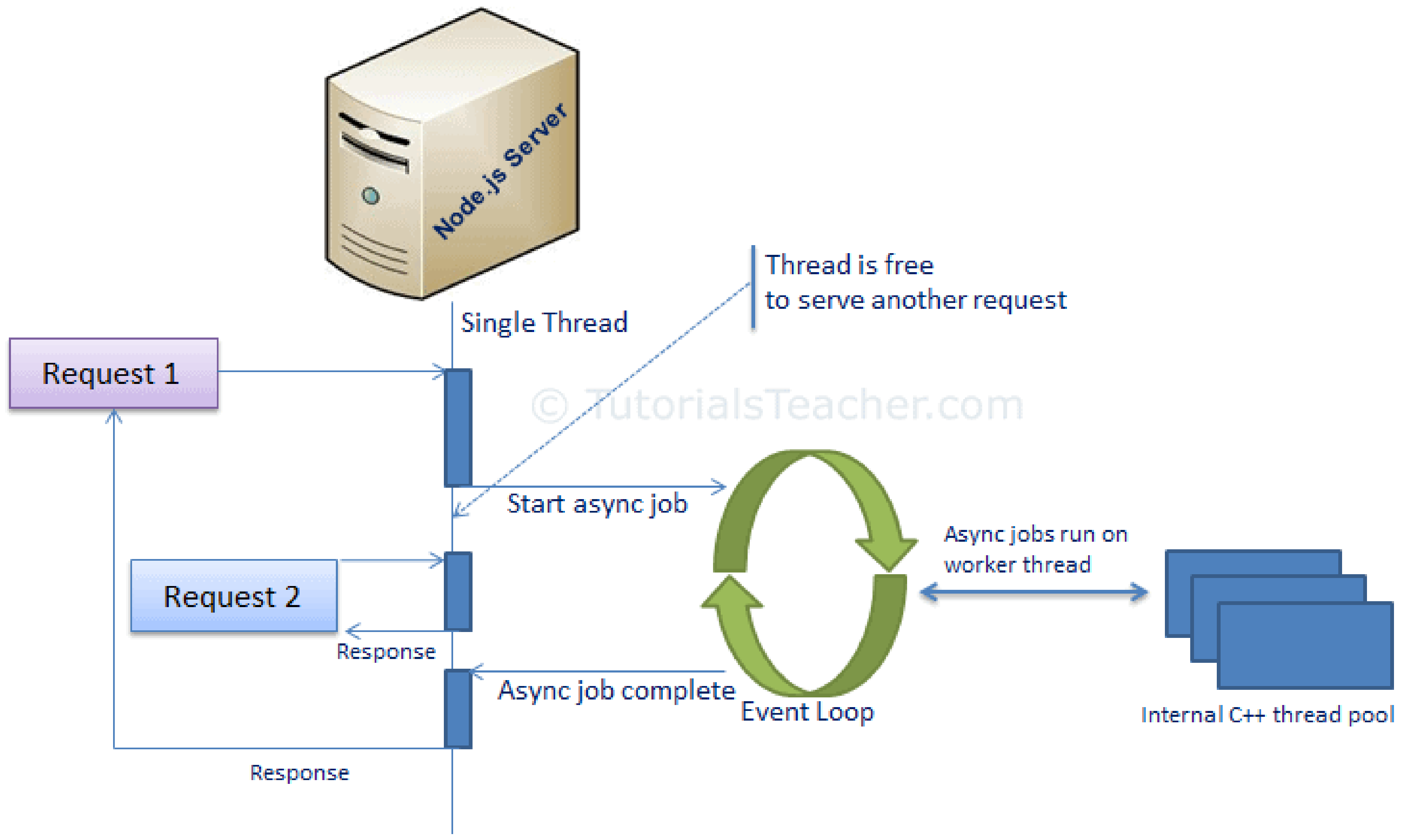
Traditional Web server


Requests
Web Server
Thread
Pool
Thread 1 executes request 1
Thread 2 executes request 2
Thread 3 executes request 3
Dedicated Thread
executes request

Traditional Web server

- In the traditional web server model, each request is handled by a dedicated thread from the thread pool.
- If no thread is available in the thread pool at any point of time then the request waits till the next available thread.
- Dedicated thread executes a particular request and does not return to thread pool until it completes the execution and returns a response.

Node.js Process Model



Node.js Process Model

- Node.js processes user requests differently when compared to a traditional web server model.
- Node.js runs in a single process and the application code runs in a single thread and thereby needs less resources than other platforms.
- All the user requests to your web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request.

Node.js Process Model

- So, this single thread doesn't have to wait for the request to complete and is free to handle the next request.
- When asynchronous I/O work completes then it processes the request further and sends the response.

Node.js Process Model

- Node.js process model increases the performance and scalability with a few caveats.
- Node.js is not fit for an application which performs CPU-intensive operations like image processing or other heavy computation work because it takes time to process a request and thereby blocks the single thread..

REPL

- REPL stands for **Read Eval Print Loop** and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode.

REPL

- Node.js or Node comes bundled with a REPL environment. It performs the following tasks –
- Read – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval – Takes and evaluates the data structure.
- Print – Prints the result.
- Loop – Loops the above command until the user presses ctrl-c twice.

REPL

- Node.js or Node comes bundled with a REPL environment. It performs the following tasks –
- Read – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval – Takes and evaluates the data structure.
- Print – Prints the result.
- Loop – Loops the above command until the user presses ctrl-c twice.

REPL

- Node.js or Node comes bundled with a REPL environment. It performs the following tasks –
- Read – Reads user's input, parses the input into JavaScript data-structure, and stores in memory.
- Eval – Takes and evaluates the data structure.
- Print – Prints the result.
- Loop – Loops the above command until the user presses ctrl-c twice.

SIMPLE Expression using REPL

```
$ node
> 1 + 3
4
> 1 + ( 2 * 3 ) - 4
3

> c = 0;
0
> do { c++;
... console.log(c);
... }while(c<6)
```

REPL Commands

- **ctrl + c** – terminate the current command.
- **ctrl + c twice** – terminate the Node REPL.
- **ctrl + d** – terminate the Node REPL.
- **Up/Down Keys** – see command history and modify previous commands.
- **tab Keys** – list of current commands.
- **.help** – list of all commands.
- **.break** – exit from multiline expression.
- **.clear** – exit from multiline expression.
- **.save *filename*** – save the current Node REPL session to a file.
- **.load *filename*** – load file content in current Node REPL session.

Functions in node JS

```
function calRectArea(height, width)
{
   return height * width;
}
console.log("Your output is-- ", calRectArea(10,20))

function display()
{

   console.log("Hellow World");

}
console.log("Value returned :" + display());
```

Addition of two numbers

```
const prompt = require('prompt-sync')();
function addition ()
{
   var n1 = prompt('Enter your first number :');
   var n2 = prompt('Enter your second number :');
   console.log("your addition of number is : " , parseInt(n1)+parseI
nt(n2));



}

console.log(addition());
```

Exercise :

- Write node JS program that accepts principle, rate of interest, time and compute the simple interest.

- Write node JS program to calculate factorial of given number using function.

Node JS : Buffer

- A buffer is a space in memory (typically RAM) that stores binary data.
- In Node.js, we can access these spaces of memory with the built-in Buffer class.
- Buffers store a sequence of integers, similar to an array in JavaScript.
- Unlike arrays, you cannot change the size of a buffer once it is created.

Why do we need a buffer?

- Buffers were introduced to help developers deal with binary data, in an ecosystem that traditionally only deal with strings rather than binaries.
- Buffers are deeply linked with streams.
- When a stream processor receives data faster than it can digest, it puts the data in a buffer.
- A simple visualization of a buffer is when you are watching a YouTube video and the red line goes beyond your visualization point: you are downloading data faster than you're viewing it, and your browser buffers it.

Node JS Module

- Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.
- Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.

Node.js Module Types

Node.js includes three types of modules:

- Core Modules
- Local Modules
- Third Party Modules

Node.js Module Types

Node.js Core Modules

Core Module	Description
http	http module includes classes, methods and events to create Node.js http server.
url	url module includes methods for URL resolution and parsing.
querystring	querystring module includes methods to deal with query string.
path	path module includes methods to deal with file paths.
fs	fs module includes classes, methods, and events to work with file I/O.
util	util module includes utility functions useful for programmers.

Example : Core module : http

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



//create a server object:

http.createServer(function (req, res) {

  res.write('Hello World!'); //write a response to the client

  res.end(); //end the response

}).listen(8080); //the server object listens on port 8080);
```

Example : Core module : http

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

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

  res.write(req.url);

  res.end();

}).listen(8080);
```

Core Module : Path

The path module provides utilities for working with file and directory paths.

```
var path = require('path');
var filename = path.basename('/Users/Refsnes/demo_path.js');
console.log(filename);
console.log(path.delimiter);
```

Core Module : Path

Get the directories from a file path:

```
var path = require('path');

var directories = path.dirname('/Users/Refsnes/demo_path.js');

console.log(directories);
```

Local Module

- Local modules are modules created locally in your Node.js application.
- These modules include different functionalities of your application in separate files and folders.
- You can also package it and distribute it via NPM, so that Node.js community can use it.

Writing Simple Local Module

```
exports.myDateTime = function () {
  return Date();
};
```

Use the **exports** keyword to make properties and methods available outside the module file.

Save the code above in a file called "myfirstmodule.js

Include Your Own Module

```
var http = require('http');
var dt = require('./myfirstmodule');


http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

Exercise

Create a local module which will contains addition, multiplication and division operations.

```
exports.add = function(n1,n2)
{
   return n1+n2;

}
```

Exercise

Create module student which will have two function, one accept name and roll number of student second display the information of student.

NPM – Node Package Manager

- A package in Node.js contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.
- NPM is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js

NPM – Node Package Manager

- A package in Node.js contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.
- NPM is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js

NPM – Node Package Manager

- Download a Package:
- Downloading a package is very easy.
- Open the command line interface and tell NPM to download the package you want.
- I want to download a package called "upper-case":

NPM – Node Package Manager

- Download "upper-case":

```
C:\Users\Your Name>npm install upper-case

var http = require('http');

var uc = require('upper-case');

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

  res.write(uc.upperCase("Hello World!"));

  res.end();

}).listen(8080);
```

Global vs Local Installation

- By default, NPM installs any dependency in the local mode.
- Here local mode refers to the package installation in node_modules directory lying in the folder where Node application is present.
- Locally deployed packages are accessible via require() method. For example, when we installed express module, it created node_modules directory in the current directory where it installed the express module.

Using package.json

- package.json is present in the root directory of any Node application/module and is used to define the properties of a package. Let's open package.json of express package present in node_modules/express/

Using package.json

Attributes of Package.json

- **name** – name of the package
- **version** – version of the package
- **description** – description of the package
- **homepage** – homepage of the package
- **author** – author of the package
- **contributors** – name of the contributors to the package
- **dependencies** – list of dependencies. NPM automatically installs all the dependencies mentioned here in the node_module folder of the package.
- **repository** – repository type and URL of the package
- **main** – entry point of the package
- **keywords** – keywords

Uninstalling a Module

- Uninstalling a Module

```
npm uninstall express
```

MES’s
IMCC

Node.js File System