Web Application Programming

Node.js Lecture 1

These slides were drawn from the following references:

https://www.w3schools.com/nodejs/

https://www.tutorialsteacher.com/nodejs/nodejs-tutorials

What is Node.js

- 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

Opening a File

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:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

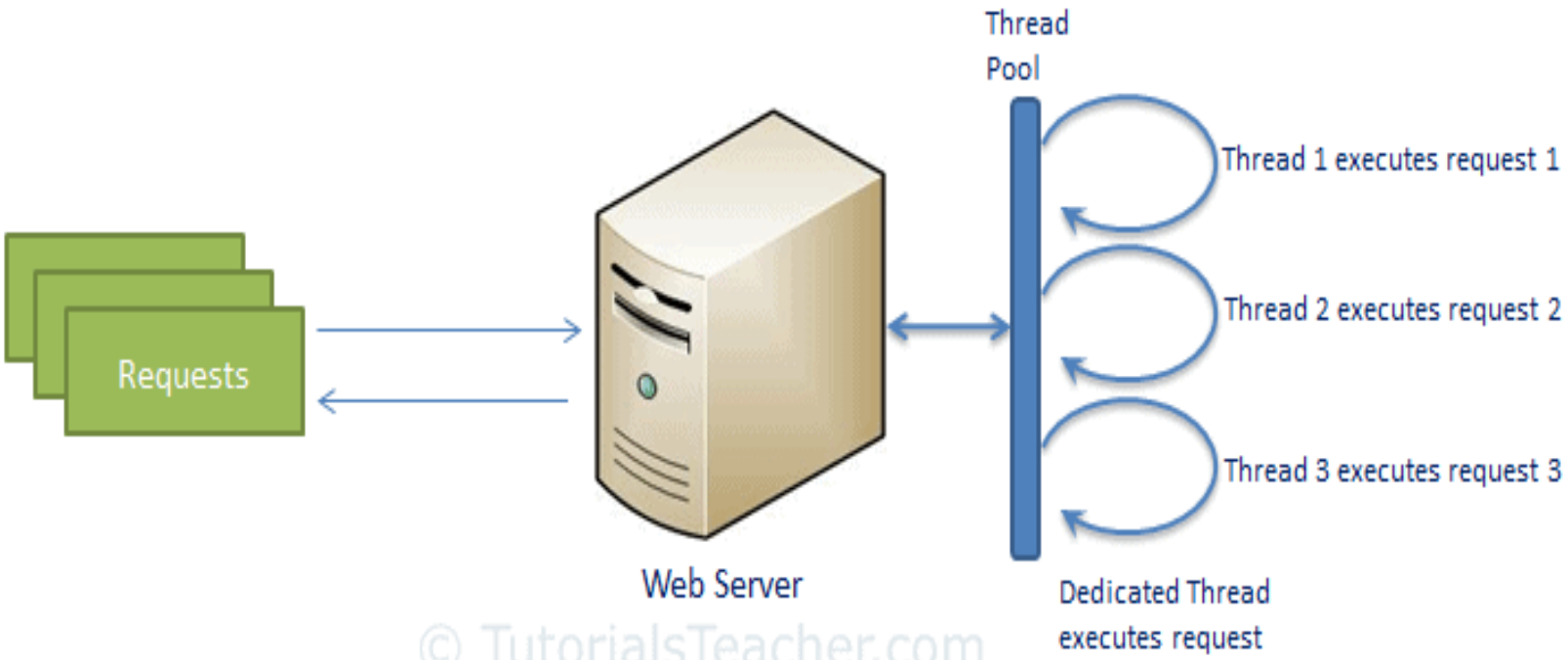
Node.js is Asynchronous

Here is how Node.js handles a file request:

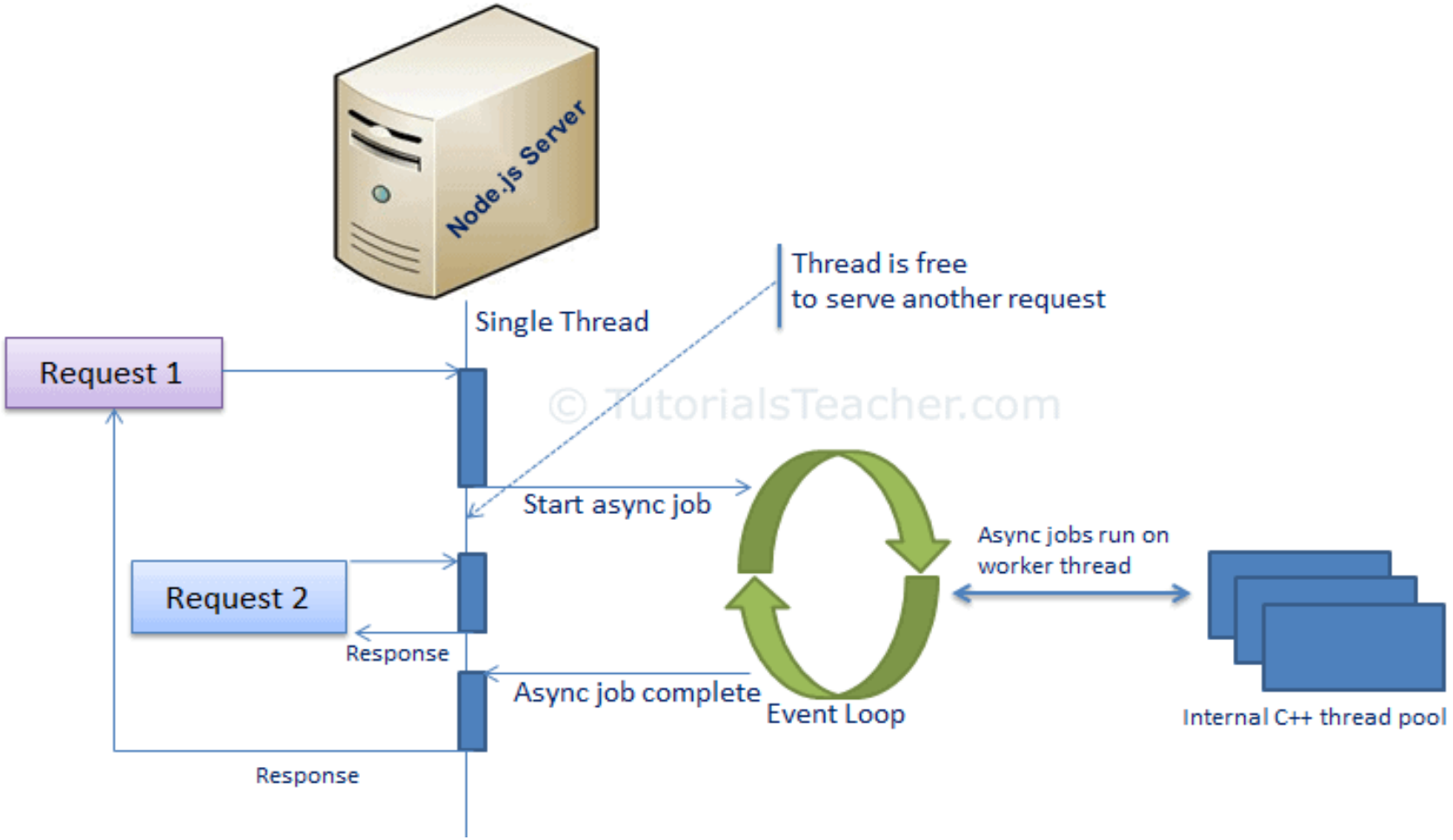
1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

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.

Traditional Web Server



Node.js Process Model



Exercise N1

- Download Node.js from the official Node.js website and install it: https://nodejs.org.
- Create a folder for your Node.js files.
- Save the Hello World program in a file using VS Code:

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

http.createServer(function (req, res) {
  res.writeHead(200,
          {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

Exercise N1 (continued)

- Open a Command Window and run the Hello World program:

```
C:NodeFolder> node HelloWorld.js
```

- In your Browser, enter the URL:

```
http://localhost:8080
```

Modules

- Modules are like JavaScript libraries.
- A *Module* is a set of functions you want to include in your application.
- Node.js has a set of built-in modules which you can use without any further installation.
- To include a module, use the **`require()`** function with the name of the module:

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

Creating New Modules (Exercise N2)

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

- Use the *exports* keyword to make properties and methods available outside the module file.
- Save this in a file called **`myModule.js`**

Exercise N2 (continued)

- Use the module *myModule* in a Node.js file:

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

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

Creating New Modules

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

- A node.js file may contain function definitions, variable declarations, and object declarations.
- By default, all of these declarations are **private**.
- The exports object is used to make any of these public.
- In the above example, function myDate( ) will become public.

Using a New Module

```
var dt = require('./myModule');
```

- The *require* operation needs the file name of the module (including the path from the currently executing node.js file).
- To use the public methods of the module, the variable name *dt* is used:

```
dt.myDate();
```

Another Module Example

```
const getName = () => {
  return 'Jim';
};

const getLocation = () => {
  return 'Munich';
};

const dateOfBirth = '12.01.1982';

exports.getName = getName;
exports.getLocation = getLocation;
exports.dob = dateOfBirth;
```

Using the New Module

```
const user = require('./userfile');
console.log(user.getName() + ' lives in ' + user.getLocation() + ' and was born on ' + user.dob);
```

- The *require* operation needs the file name of the module.
- The constant *user* accesses the public methods.

HTTP Module

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

//create a server object:
http.createServer(function (req, res) {
  //write a response to the client
  res.write('Hello World!');
  res.end(); //end the response
}).listen(8080);
//the server object listens on port 8080
```

HTTP Header

If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

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

http.createServer(function (req, res) {
  res.writeHead(200,
      {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

Client Request

- The *req* argument represents the request from the client.
- The property *url* holds the part of the url that comes after the domain name.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(8080);
```

Exercise N3

Run the code shown below and then enter the following in your Browser:

http://localhost:8080/apples
http://localhost:8080/pears
http://localhost:8080/strawberries

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(8080);
```

Query String

The following code reads the values from the query string corresponding to the names *year* and *month*:

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var q = url.parse(req.url, true).query;
  var txt = q.year + " " + q.month;
  res.end(txt);
}).listen(8080);
```

HTTP Header

If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

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

http.createServer(function (req, res) {
  res.writeHead(200,
      {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

Node.js File System

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

Common use for the File System module:

- Read files
- Create files
- Update files
- Delete files
- Rename files

Exercise N4: Simple Adder

Opening HTML page

```
<form action =
          "http://localhost:8080/add.js">
  <div>Enter two numbers
    <input type = "text" name="first"/><br/>
    <input type = "text" name="second"/><br/>
    <input type="submit" value="Click"/>
  </div>
</form>
```

Module to Perform Addition

```
exports.add = function (req,res,vals) {
    var sum = parseInt(vals.first) + parseInt(vals.second);
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write("<!DOCTYPE html>");
    res.write("<html>");
    res.write("<head><meta charset=\"utf-8\"/>");
    res.write("<title>Calculator Web Site</title>");
    res.write("</head>");
    res.write("<body>");
    res.write("<p>The sum is: ");
    res.write(String(sum));
    res.write("</p>");
    res.write("</body>");
    res.write("</html>");
    return res.end();
};
```

Web Server for the Adder

```
var http = require('http');
var url = require('url');
var fs = require('fs');
var addmod = require('./addmod.js');

http.createServer(function (req, res) {
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  if (q.pathname=="/add.js")
    addmod.add(req,res,q.query)
  else
  fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200);  // Content-Type not included
    res.write(data);
    return res.end();
  });
}).listen(8080);};
```

Reading a File

Use fs.readFile() method to read the physical file asynchronously.

Signature:

fs.readFile(fileName [,options], callback)

Parameter Description:

- filename: Full path and name of the file as a string.
- options: The options parameter can be an object or string which can include encoding and flag. The default encoding is utf8 and default flag is "r".
- callback: A function with two parameters *err* and *fd*. This will get called when *readFile* operation completes.

Read File Example

```
var fs = require('fs');
fs.readFile('TestFile.txt',
     function (err, data) {
        if (err) throw err;
        console.log(data);
      }
);
```