Τεχνολογία Διαδικτύου
9. Nodejs

Γρηγόρης Τζιάλλας

Καθηγητής

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Σχολή Θετικών Επιστημών

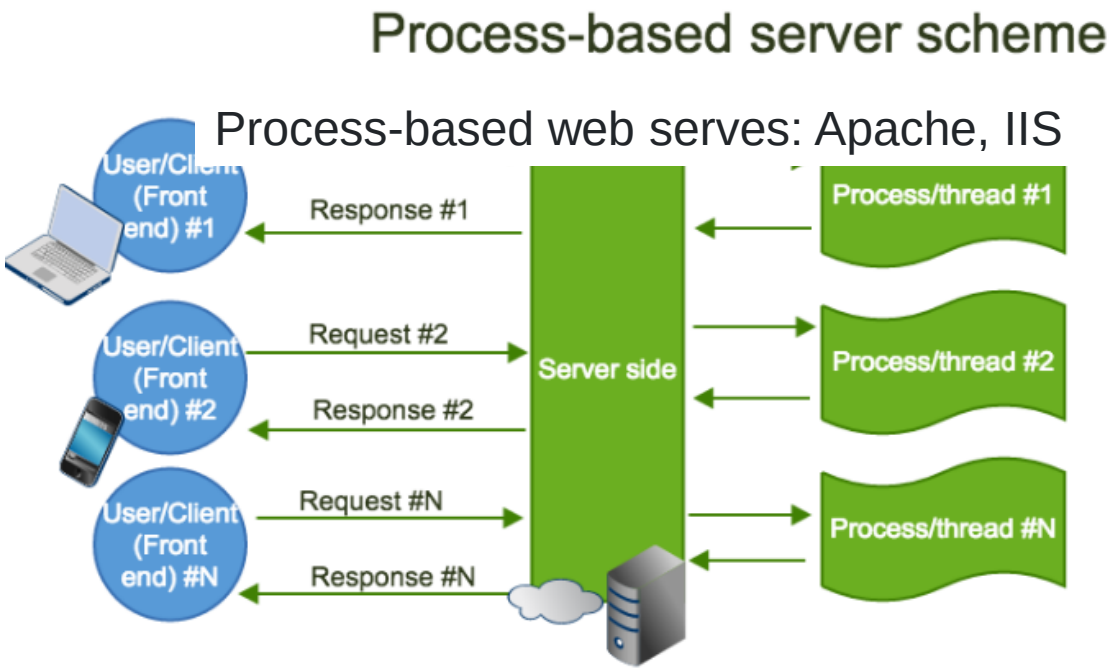
Πανεπιστήμιο Θεσσαλίας

Node.js

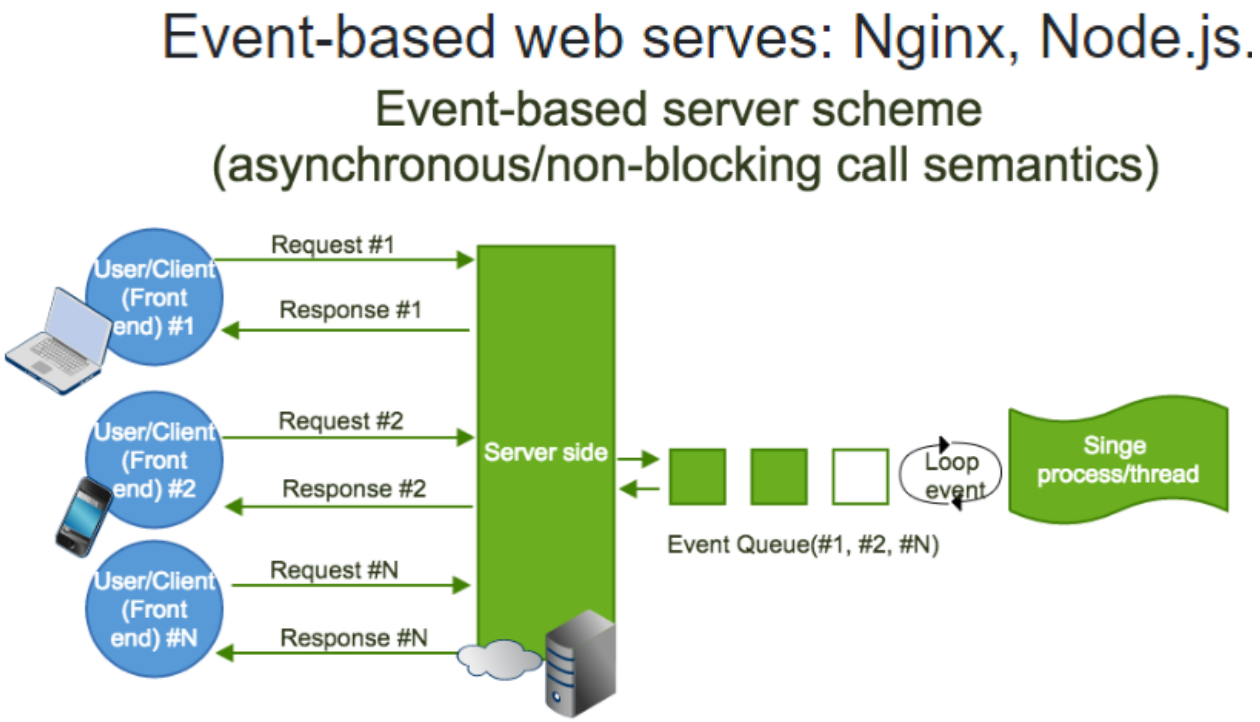
Node.js

- Node.js is open source, high performance JavaScript server
  - executing on top of Google's V8 engine
- Node.js supports high throughput, real-time, scalable use
  - via asynchronous and event driven API calls
  - running as a single non-blocking thread
  - without buffering data - it is output in chunks
- Node is free and open source
  - Node.js official web site: https://nodejs.org
  - Node.js on github: https://github.com/nodejs/node
  - Node.js community conference http://nodeconf.com

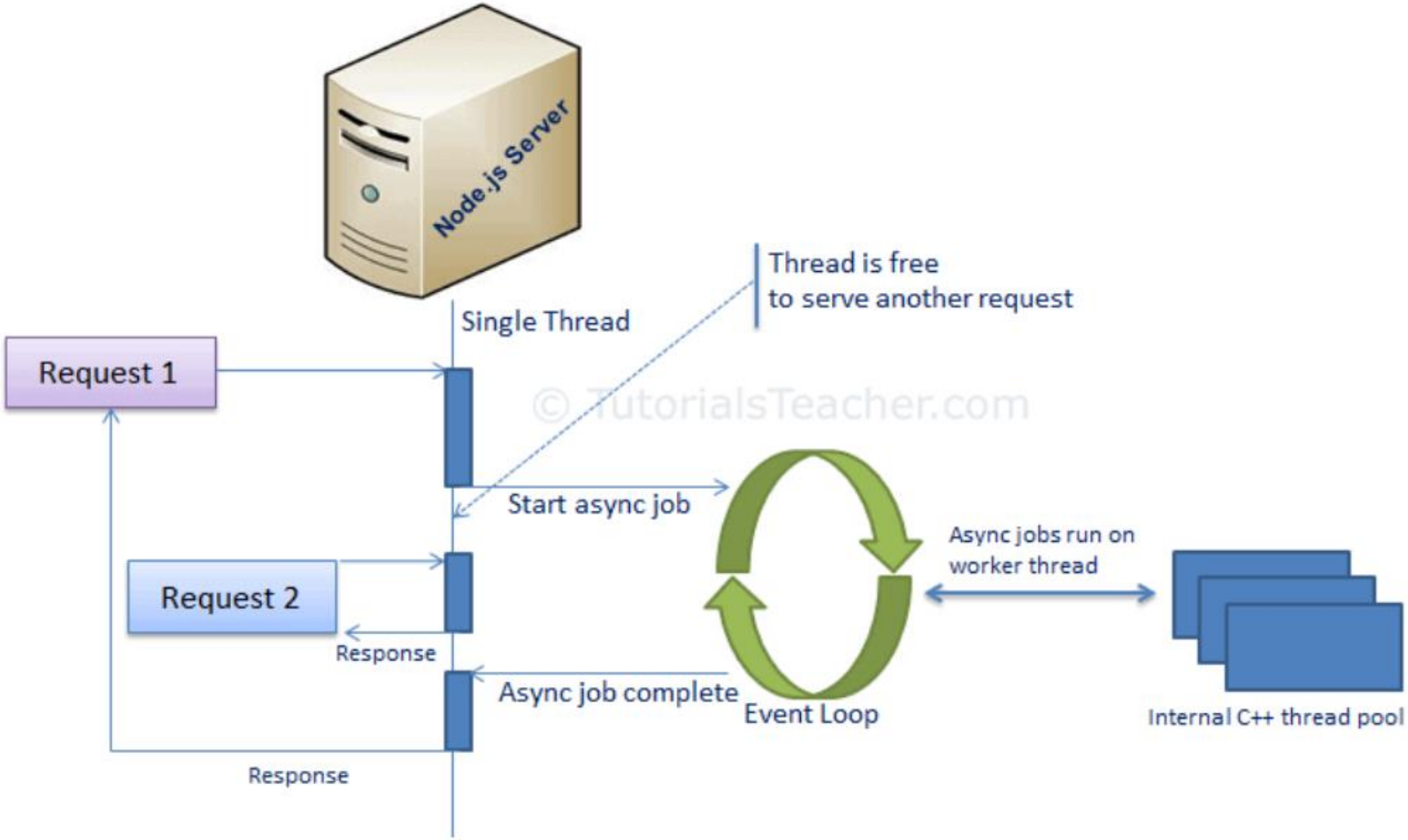
Node.js is an event-based web server

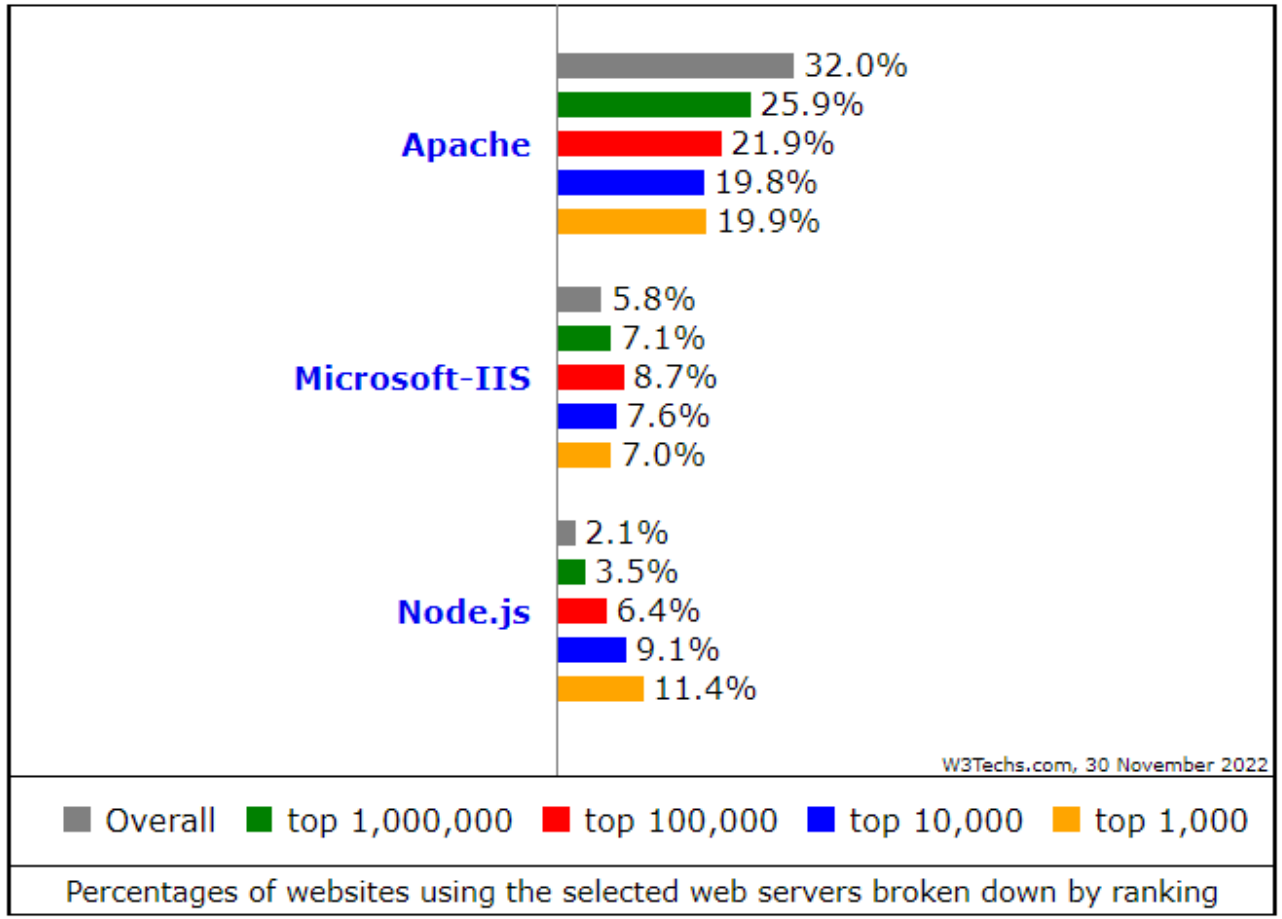


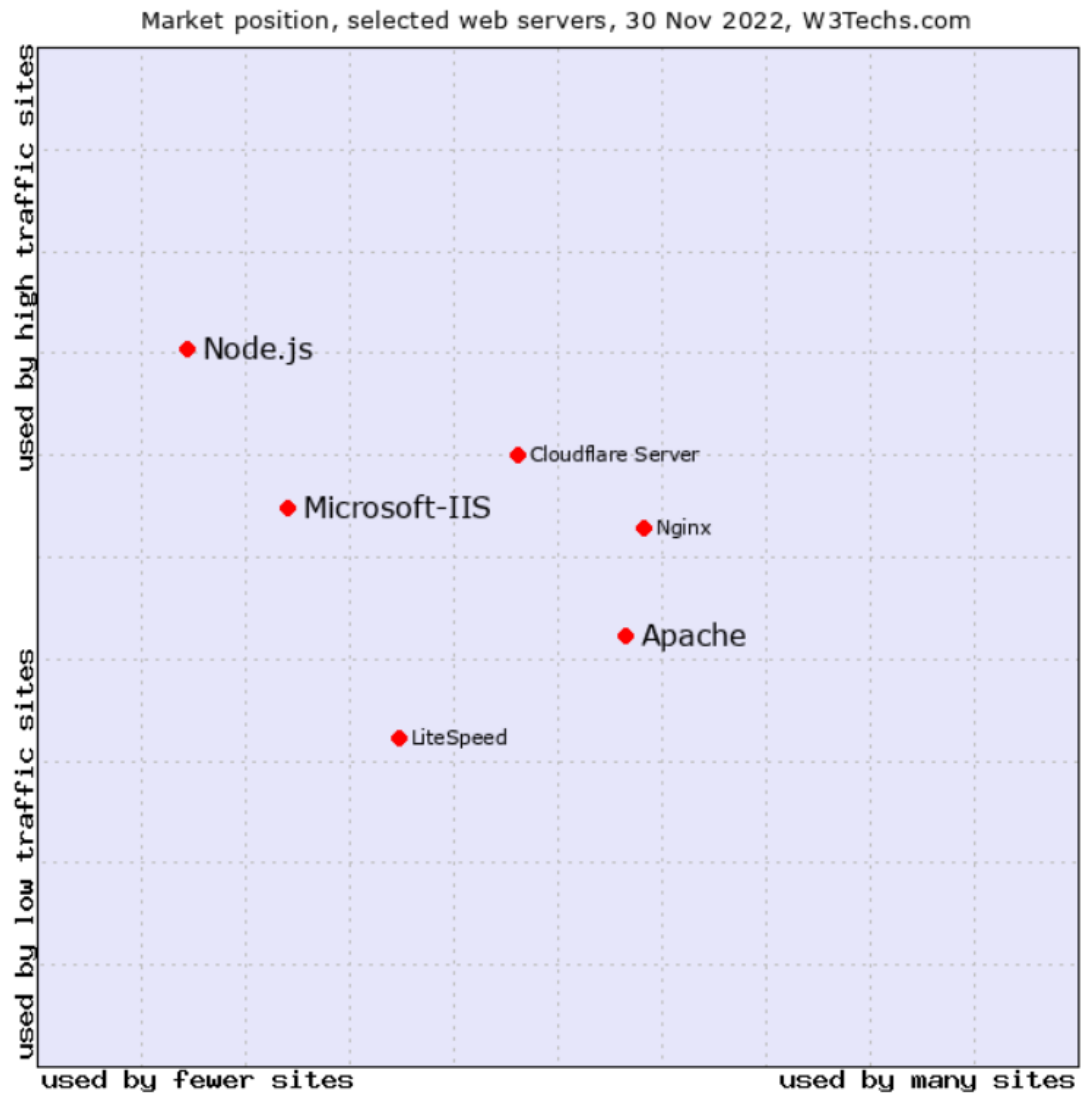
"Process-based" web servers.



Node.js event loop







https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nodejs

Advantages of Node.js

- Node.js is an open-source framework under MIT license. (MIT license is a free software license originating at the Massachusetts Institute of Technology (MIT).)
- Uses JavaScript to build entire server side application.
- Lightweight framework that includes bare minimum modules. Other modules can be included as per the need of an application.
- Asynchronous by default. So it performs faster than other frameworks.
- Cross-platform framework that runs on Windows, MAC or Linux

Node.js installation

- Install Node.js
  - https://nodejs.org/en/download/current/

- VS-Code and Node.js tutorial
  - https://code.visualstudio.com/docs/nodejs/nodejs-tutorial

- On line (requires a github account or sign-in)
  - https://codesandbox.io/
  - https://stackblitz.com
  - https://replit.com/

Node js

- Execution
  - You can execute an external JavaScript file by executing the node fileName command e.g. node server.js
- Buffer
  - Node.js includes an additional data type called Buffer (not available in browser's JavaScript). Buffer is mainly used to store binary data, while reading from a file or receiving packets over the network.
- process object
  - Each Node.js script runs in a process. It includes process object to get all the information about the current process of Node.js application.
- Defaults to local
  - Node's JavaScript is different from browser's JavaScript when it comes to global scope. In the browser's JavaScript, variables declared without var keyword become global. In Node.js, everything becomes local by default.
- Access Global Scope
  - In a browser, global scope is the window object. In Node.js, global object represents the global scope.

Node.js Code Modules

- Node.js implements CommonJS modules standard.
- 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.

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.

Loading Core Modules – Hello Example

- In order to use Node.js core or NPM modules, you first need to import it using require() function as shown below.

  var module = require('module_name');

The following example demonstrates how to use Node.js http module to create a web server.

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Use the createServer() method to create an HTTP server:

```
const http = require('http');
const hostname = 'localhost';
const port = 4000;

const server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello from Node JS');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`)
})
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\app\uthcst\tutorial\nodejs_hello> node .\app.js
Server running at http://localhost:4000/
```

localhost:4000

Hello for Node JS

You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_hello

The HTTP module

- The createServer() method of http creates a new HTTP server and returns it.
- The server is set to listen on the specified port and hostname. When the server is ready, the callback function is called, in this case informing us that the server is running.
- Whenever a new request is received, the request event is called, providing two objects: a request (an http.IncomingMessage object) and a response (an http.ServerResponse object). The first provides the request details. The second is used to return data to the caller.
  - We set the statusCode property to 200, to indicate a successful response. We also set the Content-Type header:

    ```
    res.setHeader('Content-Type', 'text/plain’)
    ```

    and we end close the response, adding the content as an argument to end():

Using your computer to host a Node.js app

- If you have a dynamic IP, or you’re under a NAT, you can deploy your app and serve the requests right from your computer using a local tunnel.
- A tool for this, available on all platforms, is ngrok. Using it, you can just type ngrok PORT and the PORT you want is exposed to the internet.
- Another service you can use is https://github.com/localtunnel/localtunnel

Creating new modules

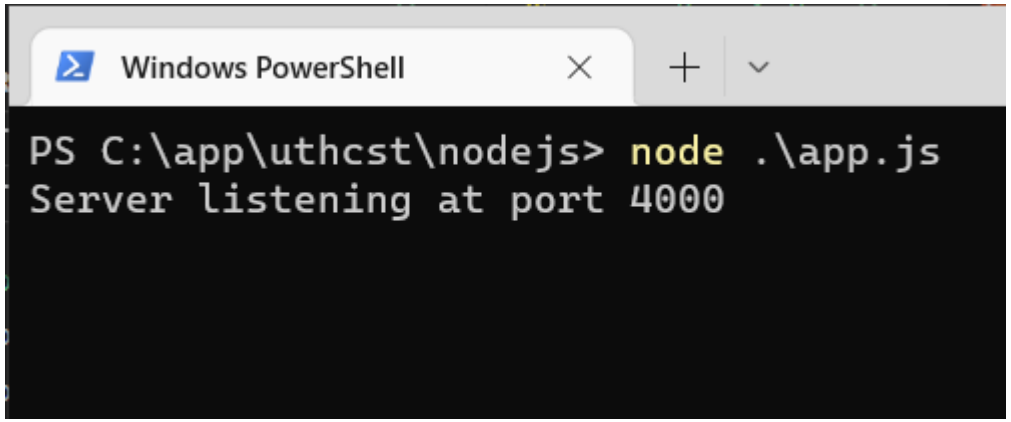
File: **app.js**

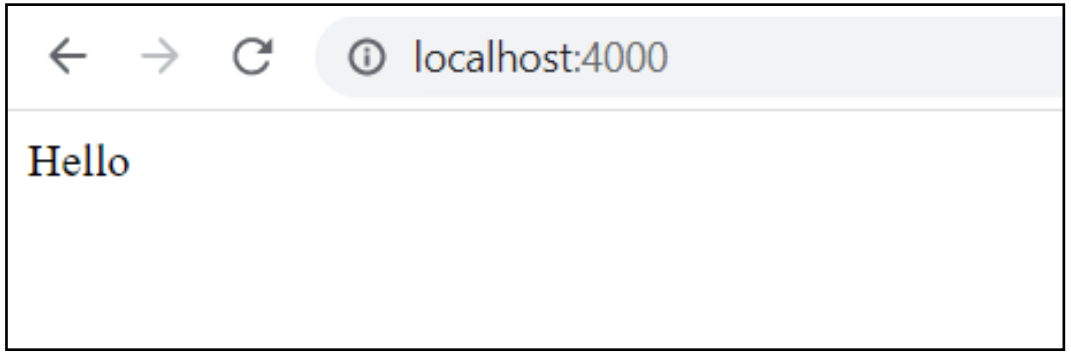
```
const http = require('http');
const messages = require('./modules/messages');
const port = process.env.port || 4000;

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(messages.title());
  res.write(messages.subtitle);
  res.end();
}).listen(port);

console.log("Running at port " + port);
```

File **./modules/messages.js** module:

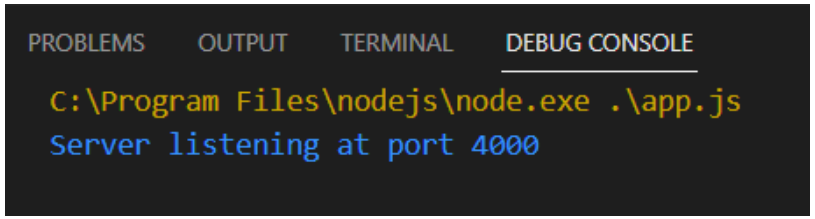
```
exports.title = function () {
  return "<h1>My website title</h1>";
};
exports.subtitle = "<h2>Welcome</h2>";
```

Execution from terminal





or Debugging with VS Code



You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_modules

Using ES6 modules

- Node js can use also the ES6 module syntax to import a module.

```
import { createServer } from 'http';
import { myGreeting } from './myModule';
const port = 4000;

createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(myGreeting());
  res.end();
}).listen(port);

console.log("Running at port " + port);
```

Read the Query String

- The request from the client (http.IncomingMessage object) has a property called "url" which holds the part of the url that comes after the domain name.
- The url built-in modules can be used to split the query string into readable parts.

```
const http = require('http');
const url = require('url');
const PORT = 4000;
http.createServer(function (req, res) {
  res.writeHead(200, { 'Content-Type': 'text/html' });
  let q = url.parse(req.url, true).query;
  let txt = q.year + " " + q.month;
  res.end(txt);
}).listen(port);

console.log("Running at port " + port);
console.log("test with url
http://localhost:4000/?year=2022&month=October")
```

http://localhost:4000/?year=2022&month=October
url with parameters month and year

Node.js File System Module

- The Node.js fs (file system) module allows you to work with the file system on your computer.
- Common use for the fs module:
  - Read files
    - The fs.readFile() method is used to read files on your computer.
  - Create files
    - fs.appendFile()
    - fs.open()
    - fs.writeFile()
  - Update files
    - fs.appendFile()
    - fs.writeFile()
  - Delete files
    - The fs.unlink() method deletes a specified file:
  - Rename files
    - he fs.rename() method renames a specified file

Node.js as a Web Server returning an html file

The Node.js file system module allows you to work with the file system on your computer.

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:

Requires the fs (file system) module

Reads the home.html file

The first argument of the res.writeHead() method is the status code, 200 means that all is OK, the second argument is an object containing the response headers.

```
const http = require('http');
const fs = require('fs');
const port = 4000;

http.createServer(function (req, res) {
  fs.readFile("./www/home.html", function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    res.end();
  });
}).listen(port);

console.log("Running at port " + port);
```

The example waits for incoming requests and responds returning the web page home.html

Node.js as a web server for static html pages

The example application waits for requests, reads the corresponding html file and returns the webpage to the client.

Requires the url module

Parses the url

If root / add index for default page

Add .html at the end if not specified

__dirname is an environment variable that tells you the absolute path of the currently executing file

Returns page not found

Returns the html file

```
const http = require('http');
const fs = require('fs');
const url = require('url');
const port = process.env.port || 4000;
http.createServer(function (req, res) {
  let pathname = url.parse(req.url, true).pathname;
  //handle root
  if (pathname=="/") pathname="/index";
  //add .html if not specified
  if (!pathname.toLowerCase().endsWith(".html"))
          pathname+=".html";
  let filename = __dirname + "/www" + pathname;
  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': 'text/html' });
    res.write(data);
    res.end();
  });
}).listen(port);
console.log("Running at port " + port);
```

You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_fileserver

Storing and retrieving JSON data using the file system module

```
const http = require('http');
const fs = require('fs’);

let nick = {
    name: "nick",
    age: 20
}
let mary = {
    name: "mary",
    age: 21
}
let aFileName='./www/data/persons.json';
let aJsonString = JSON.stringify([nick, mary]);
//write to file
fs.writeFile(aFileName, aJsonString, function (err) {
    if (err) throw err;
    console.log('Saved!');
});
//read from file
fs.readFile(aFileName,function (err, data) {
    if (err) throw err;
    let [p1, p2] = JSON.parse(data);
    console.log(p1, p2);
});
```

Installing new packages

- NPM
  - NPM is a package manager for Node.js packages, or modules
  - 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
- Installing a new package

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\app\uthcst\nodejs> npm install upper-case
```

Using the new package

```
const http = require('http');
const uc = require('upper-case');
const port = 4000;

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(uc.upperCase("Hello to upper case!"));
  res.end();
  }).listen(port);

console.log("Running at port " + port);
```

Combined example with files and upper-case module

```
const http = require('http');
//require module installed with npm
const uc = require('upper-case');
const fs = require('fs');
const url = require('url');
const port = process.env.port || 4000;


http.createServer(function (req, res) {
  let result = uc.upperCase("testing upper-case module ");
  let aFileName = __dirname + '/www/data/persons.json';
  let nick = { name: "nick", age: 20 };
  let mary = { name: "mary", age: 20 };
  let aJsonString = JSON.stringify([nick, mary]);
```

```
  //write to file
  fs.writeFile(aFileName, aJsonString, function (err) {
    if (err) {
      result += err.name + " " + err.massage;
      res.end(result);
  }
    else {
      result += "<div> File: <b>" + aFileName + "</b> created </div>";
      //read from file
      fs.readFile(aFileName, function (err, data) {
        if (err) {
          result += err.name + " " + err.massage;
          res.end(result);
        }
        else {
          let jsonData = JSON.parse(data);
          result += "<div> File contents: " +
                        JSON.stringify(jsonData) +"</div>";
          res.writeHead(200, {'Content-Type': 'text/html'});
          res.end(result);
        }
      });
    }
  });

}).listen(port);
console.log("Running at port " + port);
```

You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_files

Node.js Events

- Objects in Node.js can fire events, like the readStream object fires events when opening and closing a file.
- Node.js has a built-in module, called "Events", where you can create, fire, and listen for your own events.
- The EventEmitter Object
  - You can assign event handlers to your own events with the EventEmitter object.
  - To fire an event, use the emit() method.

Node.js events example

Define Event Emitter

Handler for high temp

Handler for low temp

Assign event handlers to events

Emit event

```
const events = require('events');
const tempAlarm = new events.EventEmitter();

//Create an event handler:
const highTemp = function () {
  console.log('High Temp Alarm');
}
const lowTemp = function () {
  console.log('Low Temp Alarm');
}
//Assign the event handler to an event:
tempAlarm.on('highTemp', highTemp);
tempAlarm.on('lowTemp', lowTemp);

//Fire the 'scream' event:
tempAlarm.emit('lowTemp');
```

Using WebSockets in Node.js

- WebSockets are an alternative to HTTP communication in web applications.
- They offer a long lived, bidirectional communication channel between client and server.
- Once established, the channel is kept open, offering a very fast connection with low latency and overhead.

Secured WebSockets

- Always use the secure, encrypted protocol for WebSockets, wss://
- ws:// refers to the unsafe WebSockets version

How WebSockets differ from HTTP

- HTTP is a very different protocol, and has a different way of communicating.
- HTTP is a request/response protocol: the server returns some data when the client requests it.
- With WebSockets:
  - the server can send a message to the client without the client explicitly requesting something
  - the client and the server can talk to each other simultaneously
  - very little data overhead needs to be exchanged to send messages. This means a low latency communication.
- WebSockets are great for real-time and long-lived communications.
- HTTP is great for occasional data exchange and interactions initiated by the client.
- HTTP is much simpler to implement, while WebSockets require a bit more overhead.

Web Sockets Example for the back-end side

app.js for node.js

Requires the webSockets module

Create a web socket listening on port 4001

Handles client connection

Every 1 second send local time to connected client

Handles client disconnection

Handles erors

```
const WebSocket = require('ws');

const wss = new WebSocket.Server({ port: 4001 });

wss.on('connection', function connection(ws) {
  console.log('Client connected');
  const interval = setInterval(() => {
    ws.send((new Date).toLocaleTimeString());
  }, 1000)
  ws.on("close", () => {
    console.log("Client disconnected");
  });
  ws.onerror = function () {
    console.log("Some Error occurred");
  }
});
```

You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_webSockets

Web Sockets Example for the front-end side

script running at browser

Connects to a server web Socket

Handles connection event

Displays received message

Handles disconnection

Closes connection

```
let ws;
function openWebSocket() {
    ws = new WebSocket('ws://localhost:4001');
    ws.onopen = function open() {
        document.getElementById("output")
            .innerHTML = "Connection is opened...";
    };
    ws.onmessage = function (evt) {
          var received_msg = evt.data;
          document.getElementById("output")
             .innerHTML = received_msg;
       };

       ws.onclose = function() {
          document.getElementById("output")
              .innerHTML = "Connection is closed...";
       };
}
function closeWebSocket() {
    ws.close();
}
```

You can find the example at https://codesandbox.io/p/github/uthcst/nodejs_webSockets

Αναφορές

Javascript

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript
- https://www.w3schools.com/js/default.asp
- https://www.w3schools.com/js/js_examples.asp
- https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/#basic-javascript

Node.js

- https://www.tutorialsteacher.com/nodejs/
- https://matfuvit.github.io/UVIT/predavanja/literatura/TutorialsPoint%20node.js.pdf
- https://www.w3schools.com/nodejs/nodejs_http.asp
- https://www.freecodecamp.org/news/the-definitive-node-js-handbook-6912378afc6e/