Vietnam National University of HCMC

International University

School of Computer Science and Engineering

Web Application Development (IT093IU)

Assoc. Prof. Nguyen Van Sinh

Email: nvsinh@hcmiu.edu.vn

(Semester 2, 2023-2024)

Introduction to Node.js

- Introduction and installation
- What will we learn
- MySQL Database connection
- APIs
- Building a web-app based on Node.js
- Conclusion
- References:

https://www.w3schools.com/nodejs/nodejs_intro.asp
https://nodejs.org/en/about/
Book: Mithun Satheesh, Bruno Joseph, D'mello Jason Krol, Web Development with MongoDB and NodeJS, Second edition, 2015.

Introduction to Node.js

- Node.js is an asynchronous event-driven JavaScript runtime, it is designed to build scalable network applications
- A JavaScript runtime environment built on Chrome's V8 JavaScript engine. It was developed by Google
- The V8 engine compiles JavaScript into native machine-level code. The execution happens on the compiled code instead of interpreting the JavaScript, making it very efficient
- Developers can easily add/modify the functionality of the web applications with many open-source libraries

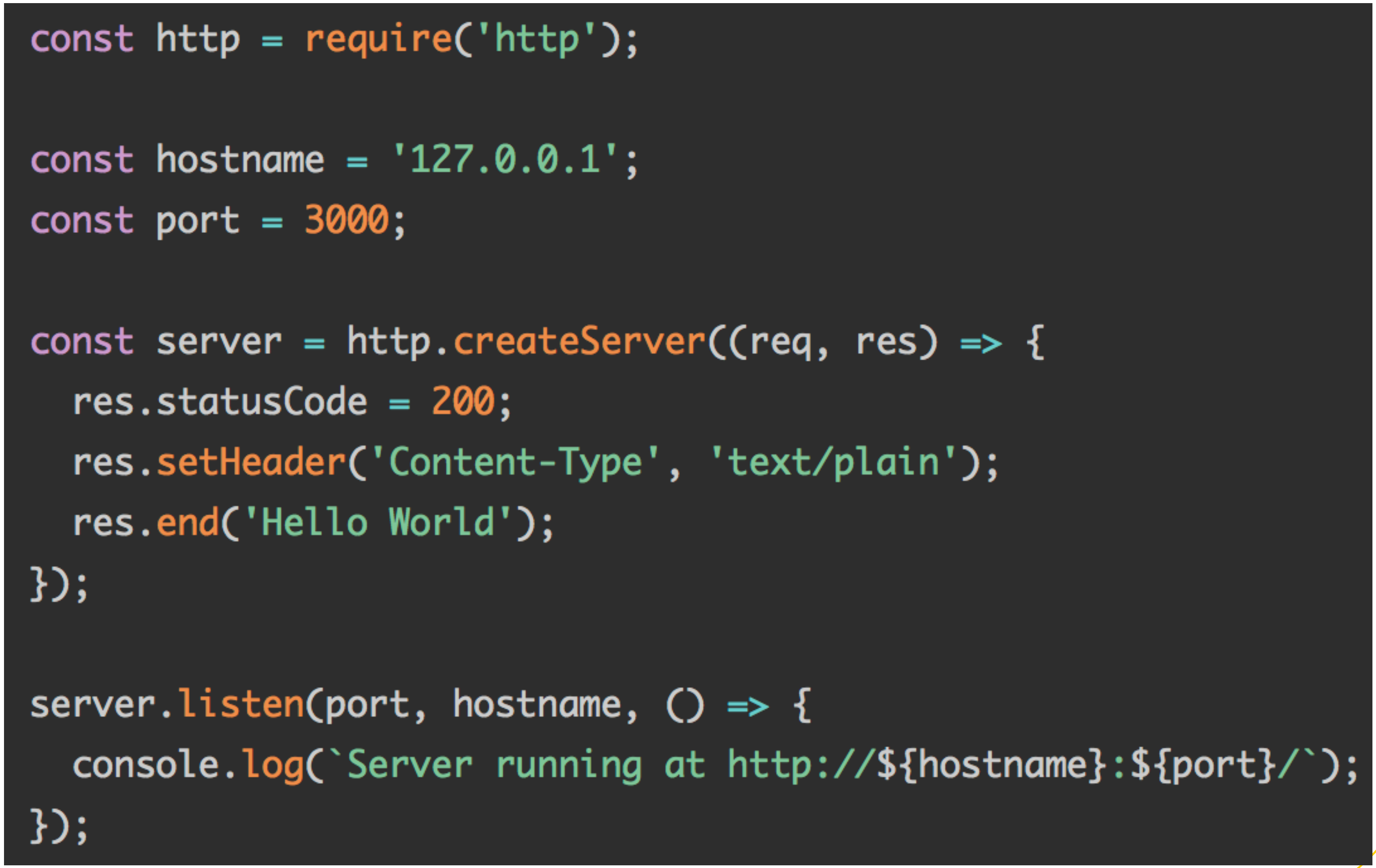
Introduction to Node.js

- Node.js uses a basic event-driven architecture so everything executed on it; and it is called as a series of non-blocking asynchronous callback
- Node.js do asynchronous processing on a single thread to provide more performance and scalability for applications that are supposed to handle too much web traffic

Introduction to Node.js

- Imagine web applications that handle millions of concurrent requests, if the server makes a new thread for handling each request that comes in, it will consume a lot of resources and we would end up trying to add more and more servers to increase the scalability of the application.
- In this case, the single threaded asynchronous processing model allows us to process much more concurrent requests with less number of server-side resources
- Node.js (by default) will not utilize the number of CPU cores available on the server without using extra modules

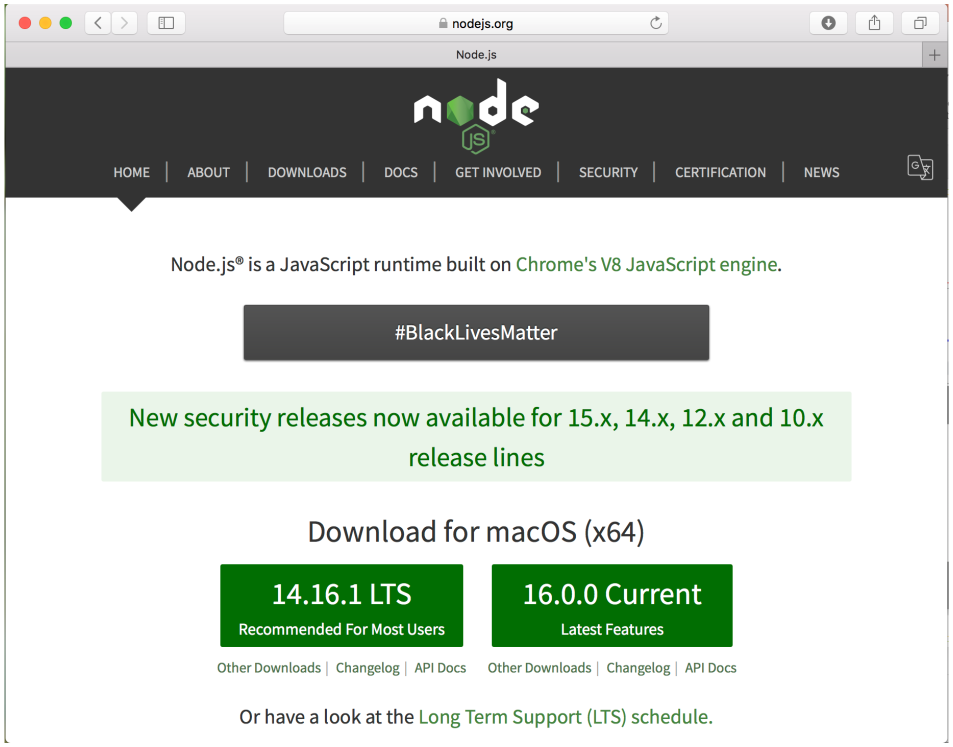
Take a look on this page: https://nodejs.org/en/about/



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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

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

Install Node.js

- Go to https://nodejs.org/en/
- Download and install the latest version 14.16.1 LTS?


nodejs.org
Node.js
node
HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

#BlackLivesMatter

New security releases now available for 15.x, 14.x, 12.x and 10.x release lines

Download for macOS (x64)

14.16.1 LTS
Recommended For Most Users

16.0.0 Current
Latest Features

Other Downloads | Changelog | API Docs

Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

- In detail: go to the lab section

Introduction to IDE & NPM

- In order to implement a web-app based on Node.js the following IDE can be considered:
  - GoormIDE: a powerful cloud IDE service to maximize productivity for developers and teams.
  - Support many programming languages: NodeJS, C++, C#, Java, Python, Swift, Kotlin, https://ide.goorm.io/
  - Visual Studio Code (it is used in this course): A freeware source-code editor made by Microsoft for Windows. It support JavaScript, TypeScript, JSON, CSS, HTML and debug for Node.js
  - Refer: https://code.visualstudio.com/docs/nodejs/nodejs-tutorial
- Using NPM (Node Package Manager), a power tool to create and manage the JavaScript libraries for Node.js https://www.npmjs.com

When Node.js is used?

- Node.js is well suited for applications that are expected to handle many concurrent connections and each incoming request requires very few CPU cycles
- If you intend to do computationally intensive tasks on requests, it will end up blocking the event loop. Thus, it impacts on other requests concurrently processed by the web server
- Node.js is also well suited for real-time web applications, such as chat rooms, collaborative tools, etc.
- Some big companies that are now using Node.js: Netflix, Uber, eBay, LinkedIn, PayPal, Walmart, Citibank, Firefox, Medium, Groupon, etc.

What will we learn?

- Node.js with MySQL
- Node.js with MongoDB (an option to extend)
- Hence, we will work with both SQL and NoSQL databases. Depending on the use case, you may prefer one to another. Here are some references for you to decide which database to use:
  - https://www.guru99.com/sql-vs-nosql.html
  - https://www.mongodb.com/nosql-explained/nosql-vs-sql
- Some packages that we use: express, ejs, body-parser, passport, axios, mongoose, express-session, mysql, jsonwebtoken, bcryptjs, cookie-parser

Building a sample web application

- Go to the lab section, you will build a web application with the following requirements:
  - User authentication and management
  - Create, Read, Update, Delete objects in the MySQL database
  - Use API to get real-time data

- When finished, you will:
  - Evaluate and compare it to a web-app based on J2EE technique
  - Apply in practice for developing a web-app

MySQL Database connection:

Introduction to MySQL

- MySQL is one of the most popular databases
- Refer: https://www.mysql.com
- It is freeware
- It has a lot of functionalities
- It is updated frequently with new features and security improvements
- Batch commands allows us to process enormous amount of data
- We can handle on both command line and UI (install MAMP and work on phpMyAdmin)

MySQL Database connection:

- Refer: https://www.w3schools.com/nodejs/nodejs_mysql.asp
- Install MySQL on MacOS: npm install mysql
- Source code to connect MySQL:

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

var con = mysql.createConnection({
 host: "localhost",
 user: "yourusername",
 password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

- In detail: in the lab section.

APIs - What is it?

- API (Application Programming Interface) is a set of functions that allows applications to access data and interact with each other
- API lets a develop make a specific “request” to send or receive information. It is shown in the JSON (JavaScript Object Notation) format
- For example: Facebook and Twitter have a robust API that allows developers to build plugins and work with data directly, without them being fully granted full access as a general security precaution
- There are several components of an API Request for it to function.

Components of an API Request

- Endpoint
  - It often consists of 2 parts: REST Endpoint and API Endpoint
  - The REST Endpoint may look like a regular URL but if you put this into a web browser, you will receive a 404-error message
  - The API Endpoint is the path that will vary depending on the user’s specific purpose when making the API request
  - When we put these two parts together, we have a complete Endpoint

- Header:
  - Headers will give the server the variables in the request.
  - Common examples of a header would be authentication credentials such as a “Auth Token” or “Client ID”. These credentials are provided to you automatically when you create an API Account.

Components of an API Request

- Method
  - It can be one of four methods: Get, Post, Put, Delete

- Data
  - Data is often given when we are creating, updating, or deleting something. In other words, data is generally required when the method is Post, Put or Delete. For the GET request, data is not required

- Params
  - These are the specifications (parameters) given by the user. They will be appended to the Endpoint to make the URL serve the user's purpose

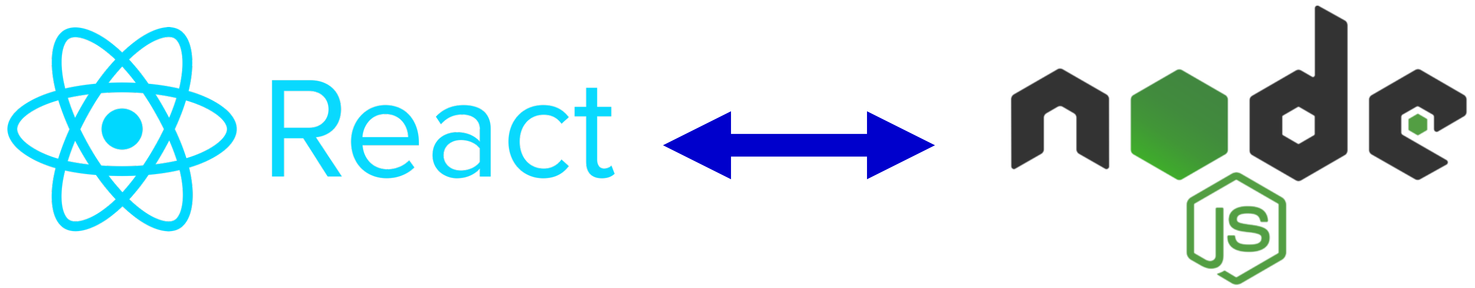
Note: Please refer to the documentations of the API that you want to use in order to understand what components and values you should use to suit your purpose

Building a web-app based on Node.js

Objective:

Create a simple web-app that apply React for frontend and NodeJS for backend.

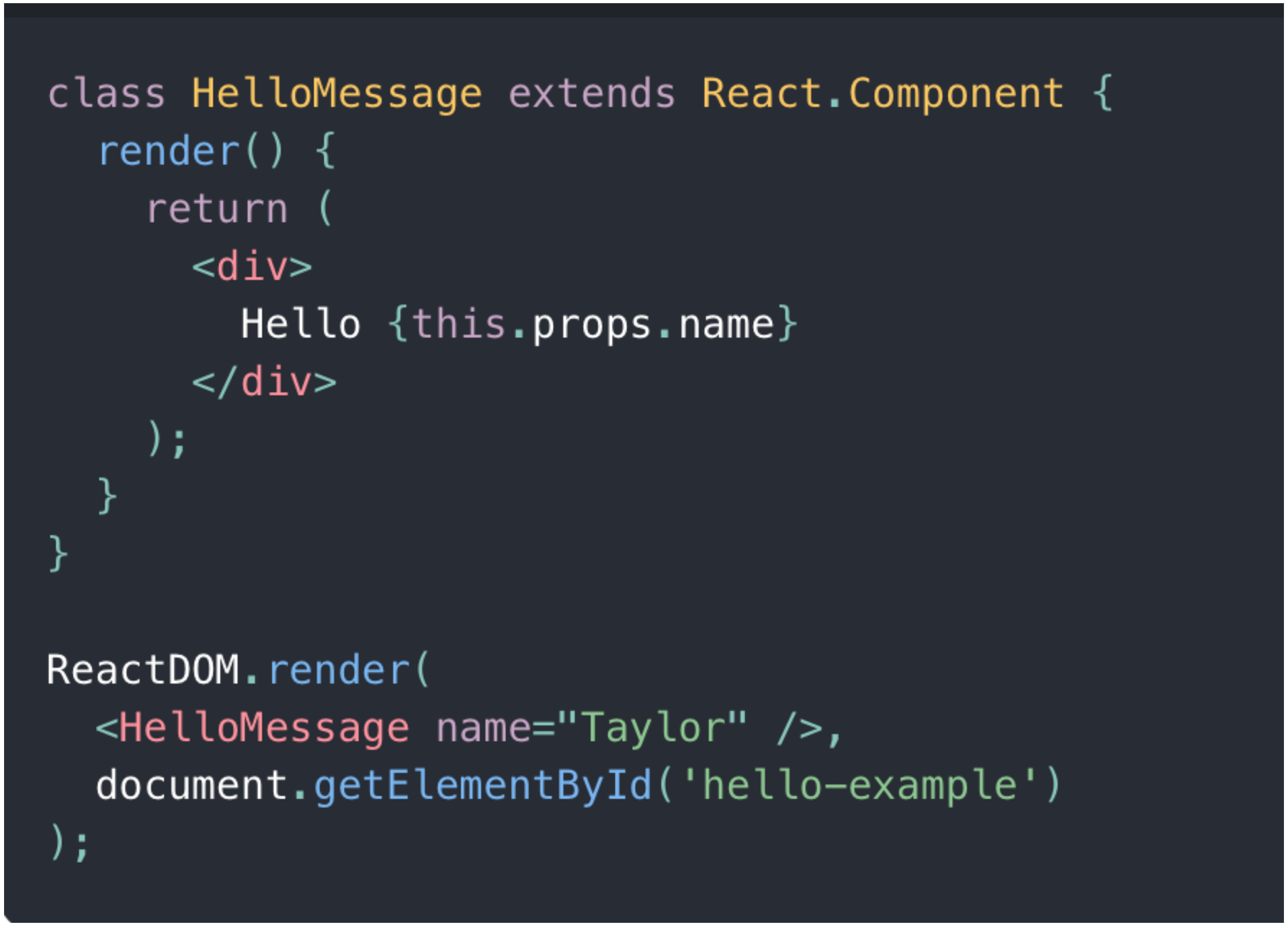
(Advanced project will be guided in the lab section)



Introduction to ReactJS

- A declarative, efficient and flexible javascript library for creating UI
- Refer at https://reactjs.org for detail
- Support block of code as a collection of HTML, CSS, JS
- The components can be written in pure JS or JSX (JavaScript XML)
- Using Visual Studio Code as an IDE for code editor.
- Refer at https://reactjs.org/tutorial/tutorial.html

A Simple Component



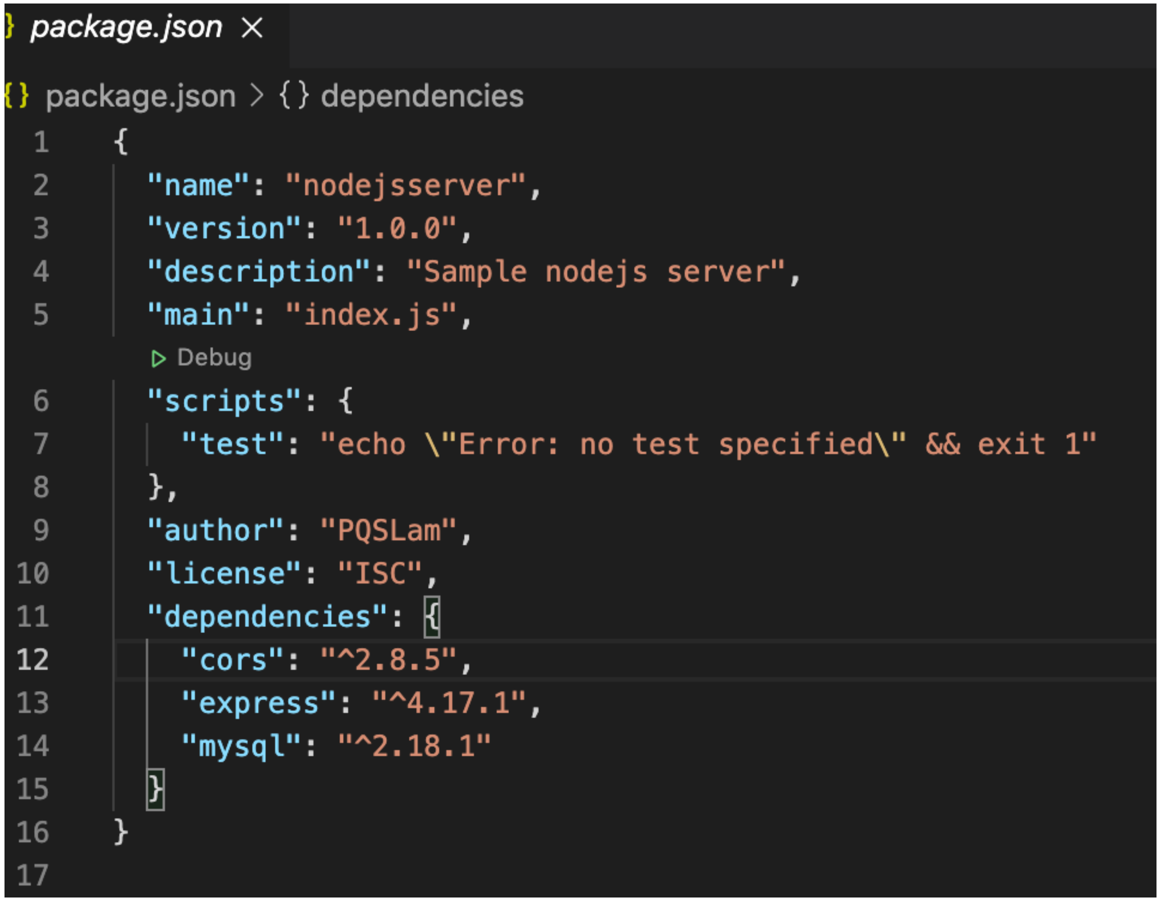
```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

NodeJS - Setting up

- Two common ways to install NodeJS:
  - Download installer: *https://nodejs.org/en/*
  - *Using Package manager (Linux/Macos):*
    sudo apt install nodejs
    sudo apt install npm
- Creating new NodeJS project:
  - Start Visual Studio Code.
  - Open your workspace (the folder contained your projects).
  - Open integrated terminal (select *Terminal -> New Terminal).*
  - *Using the command npm init to initialize new project.*

NodeJS - Setting up

npm init will ask you a series of questions and generate a ***package.json*** file to get you started



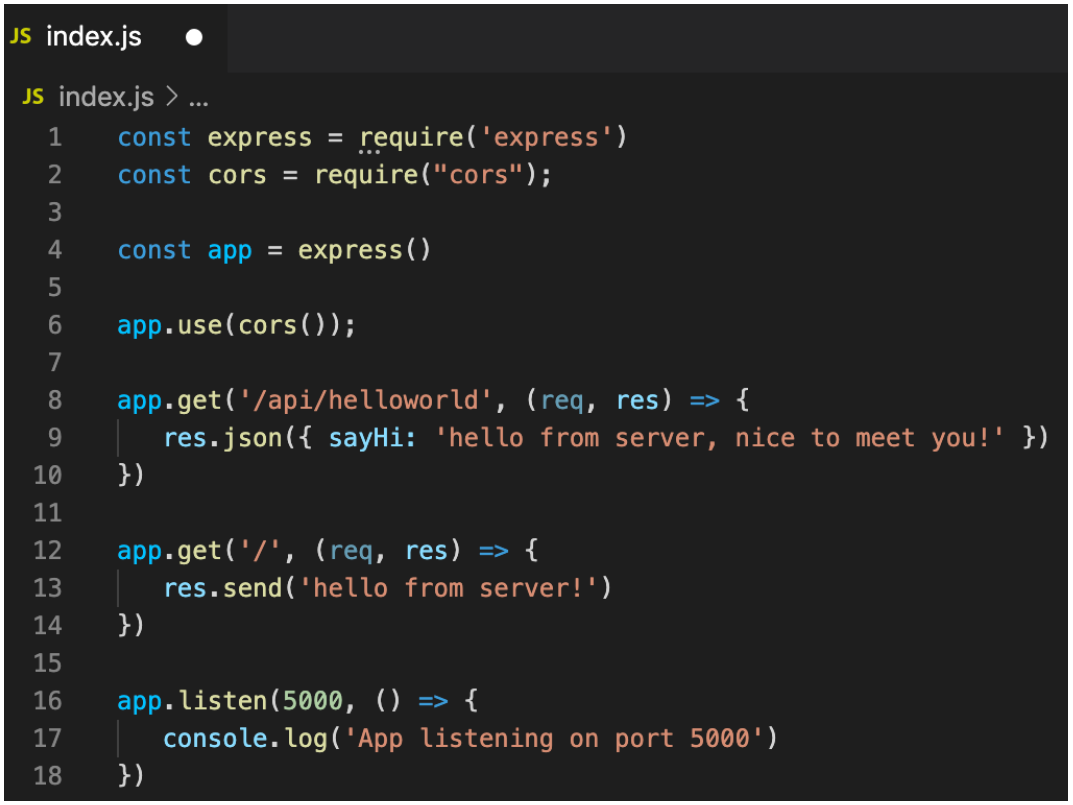
```
{
  "name": "nodejsserver",
  "version": "1.0.0",
  "description": "Sample nodejs server",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "PQSLam",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "mysql": "^2.18.1"
  }
}
```

NodeJS - Setting up

- Install require packages/framework for server:
  - ExpressJS: *npm install --save express*
  - Express cors: *npm install --save cors*

NodeJS - Setting up

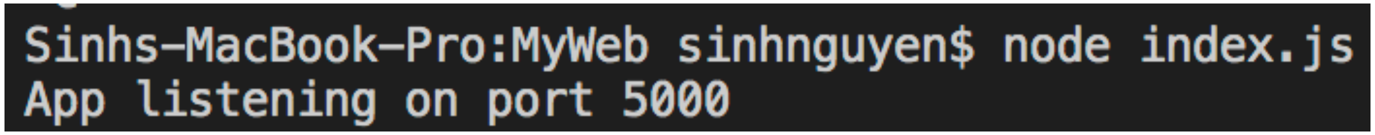
- Create *index.js* in your project with following content:



```javascript
const express = require('express')
const cors = require("cors");

const app = express()

app.use(cors());

app.get('/api/helloworld', (req, res) => {
    res.json({ sayHi: 'hello from server, nice to meet you!' })
})

app.get('/', (req, res) => {
    res.send('hello from server!')
})

app.listen(5000, () => {
    console.log('App listening on port 5000')
})
```

NodeJS - Setting up

- Try to run your NodeJS server using integrated terminal: *node index.js*
- If your settings are correct, the server will be started and listen on port 5000
  (you may edit index.js to choose another port if you want)



```
Sinhs-MacBook-Pro:MyWeb sinhnguyen$ node index.js
App listening on port 5000
```

React - Setting up

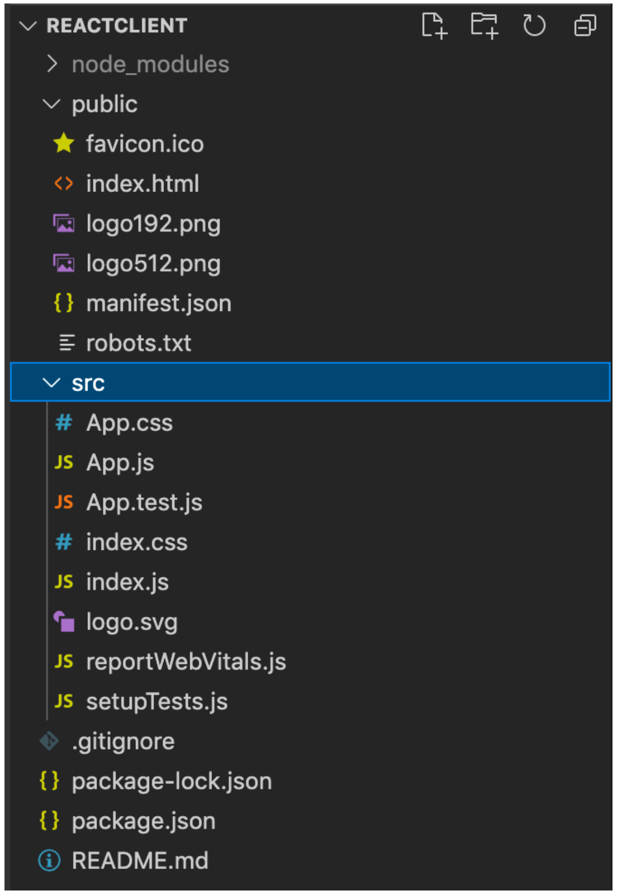
- Install create-react-app (officially supported way to create React applications):
  - MacOs, Linux: *sudo npm install -g create-react-app*
  - Windows: *npm install -g create-react-app*
- Install axios (promise based HTTP client for NodeJS and React):
  - *npm install --save axios*

React - Setting up

- Creating new React project:
  - Start Visual Studio Code.
  - Open your workspace (the folder contained your projects).
  - Open integrated terminal (select *Terminal* ➔ *New Terminal*).
  - *Using the command create-react-app <your app name> to initialize new project. For example: create-react-app reactclient.*

React - Setting up

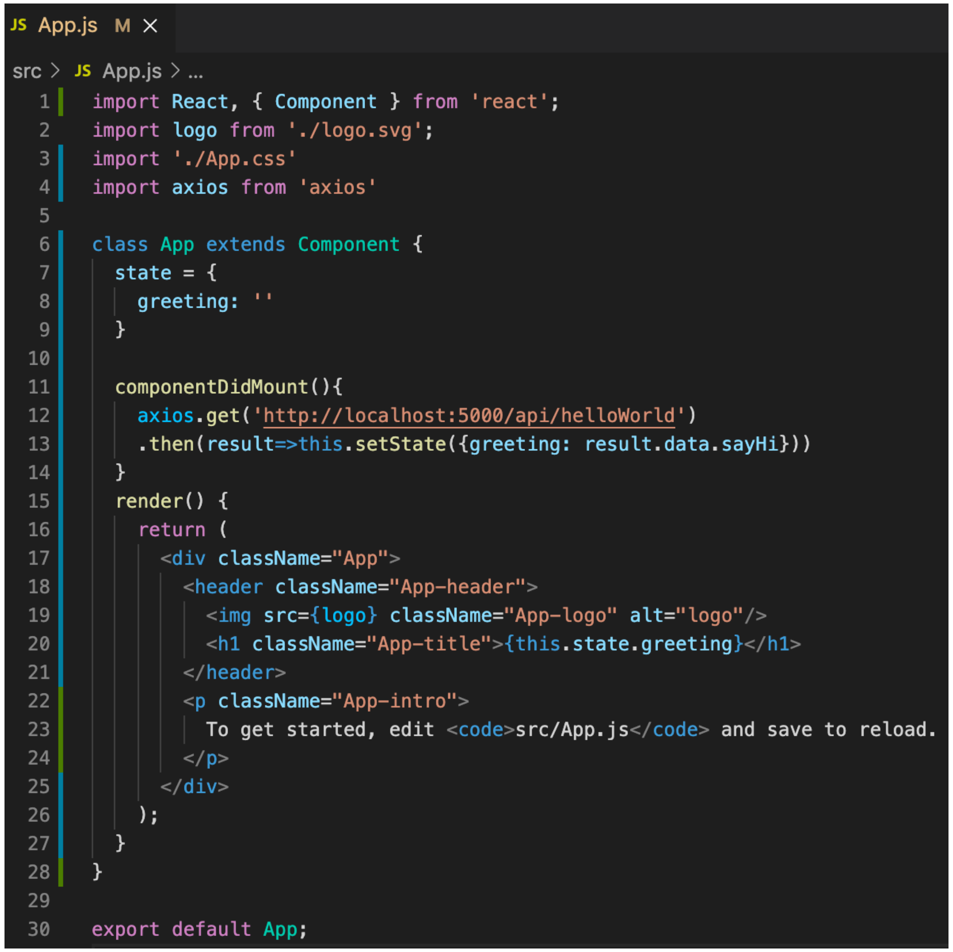
- *create-react-app* will automatically create base structure for your project



```
REACTCLIENT
  > node_modules
  v public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  v src
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
```

React - Setting up

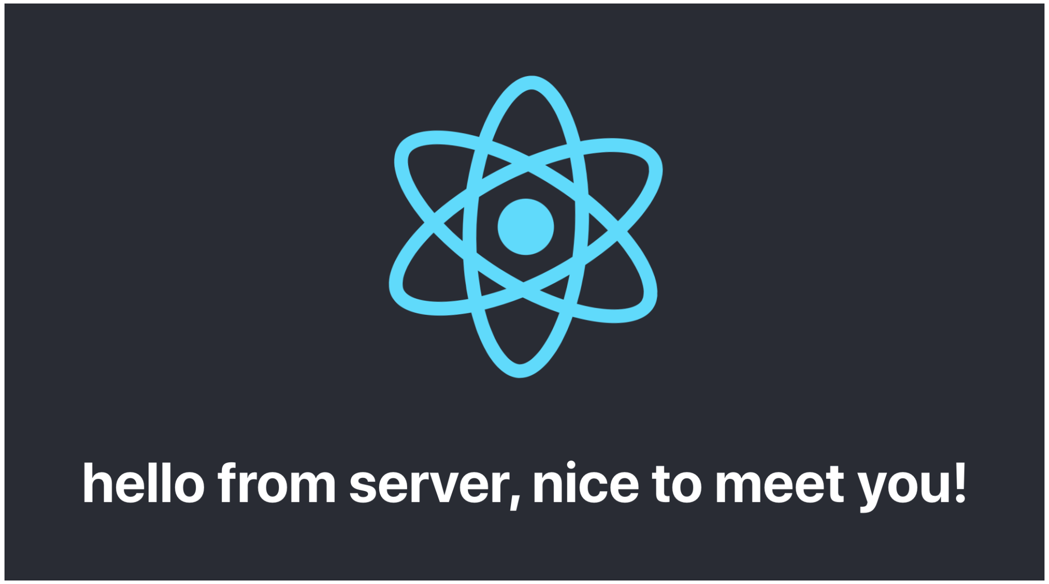
- Open and modify App.js with following content:



```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css'
import axios from 'axios'

class App extends Component {
  state = {
    greeting: ''
  }

  componentDidMount(){
    axios.get('http://localhost:5000/api/helloWorld')
    .then(result=>this.setState({greeting: result.data.sayHi}))
  }
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo"/>
          <h1 className="App-title">{this.state.greeting}</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
```

React - Setting up

- Try to run your React server using integrated terminal: go to folder -> *npm run start (npm start)*
- If your settings are correct, your default web browser will be opened and display as follow:



hello from server, nice to meet you!

React - NodeJS

- Congratulation, you have just created your first NodeJS application and React application. But what did we actually do?
- NodeJS:
  - Install **express** framework for NodeJS
  - Install **cors** package which enabled Cross-origin resource sharing
  - Create a new API: **GET - *'/api/helloworld'***
  - Make the server started and listened on port 5000
- React:
  - Install **create-react-app** package which helped creating react applications
  - Install **axios** package which helped sending and receiving data via HTTP methods.
  - Send GET request to NodeJS server **('/api/helloworld'**) and receive response
- All of them are run based on Visual Studio Code
  - Right-click at web_folder_name ➔ Open in integrated terminal ➔ npm start

Conclusion

- This extended lecture help you:
  - Know one more tool and technique for web application development
  - Understand about the Node.js
  - How to build a web-app based on the Node.js and MySQL database
  - Have ability to self-study existing tools, frameworks, techniques, APIs, Libraries, etc., for web application development in practice.
- Can be applied to do in the Internship
- Can have a job in such field!

Homework

- Install Node.js in your computer
- Create a web-app based on the Node.js
- Test DB connection
- Create a small web-app to test