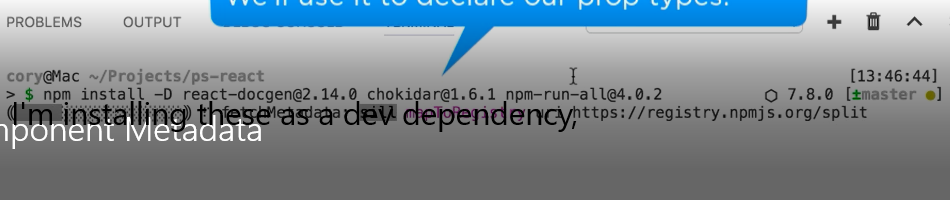
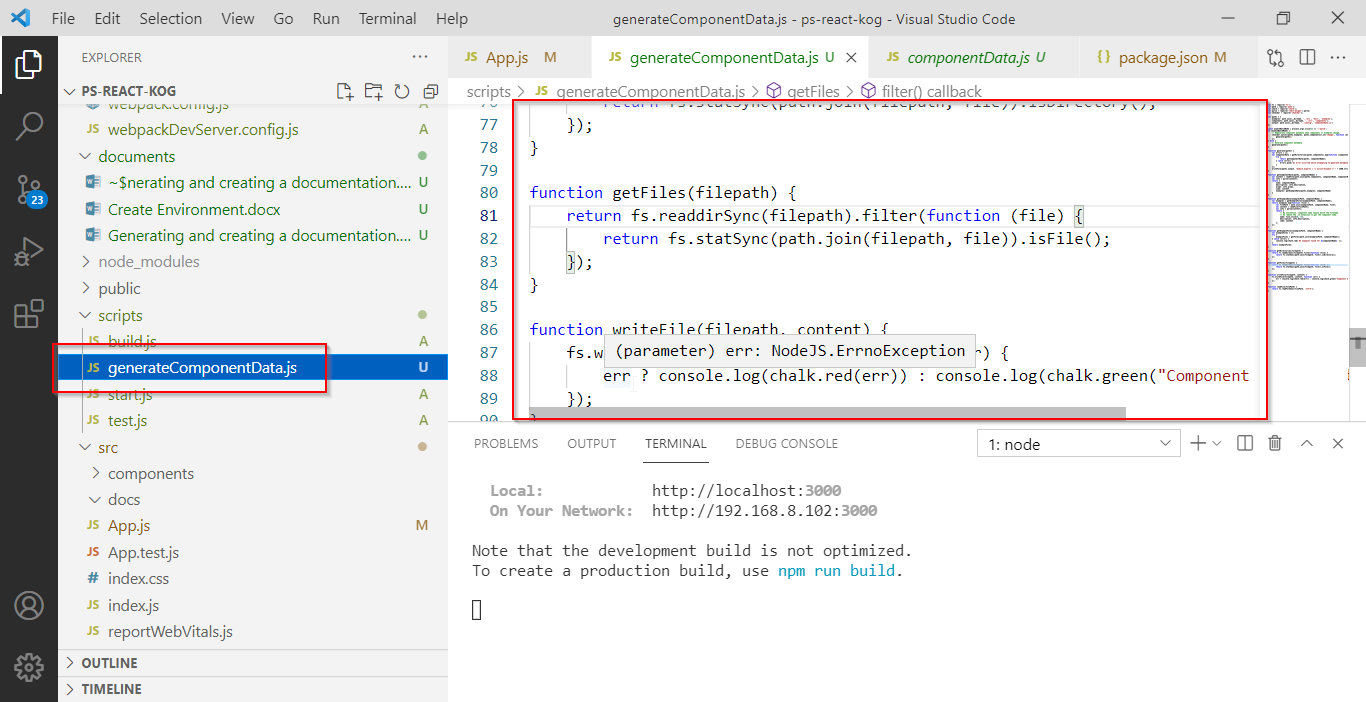
Generating and creating a documentation



npm install -d react-docgen@2.14.0 chokidar@1.6.1 [npm-run-all@4.0.2](mailto:npm-run-all@4.0.2)

npm i -s higlight.js@ 9.11.0

create a generateComponentData.js files and add below code



var fs = require('fs');

var path = require('path');

var chalk = require('chalk');

var parse = require('react-docgen').parse;

var chokidar = require('chokidar');

var paths = {

    examples: path.join(\_\_dirname, '../src', 'docs', 'examples'),

    components: path.join(\_\_dirname, '../src', 'components'),

    output: path.join(\_\_dirname, '../config', 'componentData.js')

};

const enableWatchMode = process.argv.slice(2) == '--watch';

if (enableWatchMode) {

    // Regenerate component metadata when components or examples change.

    chokidar.watch([paths.examples, paths.components]).on('change', function (event, path) {

        generate(paths);

    });

} else {

    // Generate component metadata

    generate(paths);

}

function generate(paths) {

    var errors = [];

    var componentData = getDirectories(paths.components).map(function (componentName) {

        try {

            return getComponentData(paths, componentName);

        } catch (error) {

            errors.push('An error occurred while attempting to generate metadata for ' + componentName + '. ' + error);

        }

    });

    writeFile(paths.output, "module.exports = /\* eslint-disable \*/ " + JSON.stringify(errors.length ? errors : componentData));

}

function getComponentData(paths, componentName) {

    var content = readFile(path.join(paths.components, componentName, componentName + '.js'));

    var info = parse(content);

    return {

        name: componentName,

        description: info.description,

        props: info.props,

        code: content,

        examples: getExampleData(paths.examples, componentName)

    }

}

function getExampleData(examplesPath, componentName) {

    var examples = getExampleFiles(examplesPath, componentName);

    return examples.map(function (file) {

        var filePath = path.join(examplesPath, componentName, file);

        var content = readFile(filePath);

        var info = parse(content);

        return {

            // By convention, component name should match the filename.

            // So remove the .js extension to get the component name.

            name: file.slice(0, -3),

            description: info.description,

            code: content

        };

    });

}

function getExampleFiles(examplesPath, componentName) {

    var exampleFiles = [];

    try {

        exampleFiles = getFiles(path.join(examplesPath, componentName));

    } catch (error) {

        console.log(chalk.red(`No examples found for ${componentName}.`));

    }

    return exampleFiles;

}

function getDirectories(filepath) {

    return fs.readdirSync(filepath).filter(function (file) {

        return fs.statSync(path.join(filepath, file)).isDirectory();

    });

}

function getFiles(filepath) {

    return fs.readdirSync(filepath).filter(function (file) {

        return fs.statSync(path.join(filepath, file)).isFile();

    });

}

function writeFile(filepath, content) {

    fs.writeFile(filepath, content, function (err) {

        err ? console.log(chalk.red(err)) : console.log(chalk.green("Component data saved."));

    });

}

function readFile(filePath) {

    return fs.readFileSync(filePath, 'utf-8');

}

