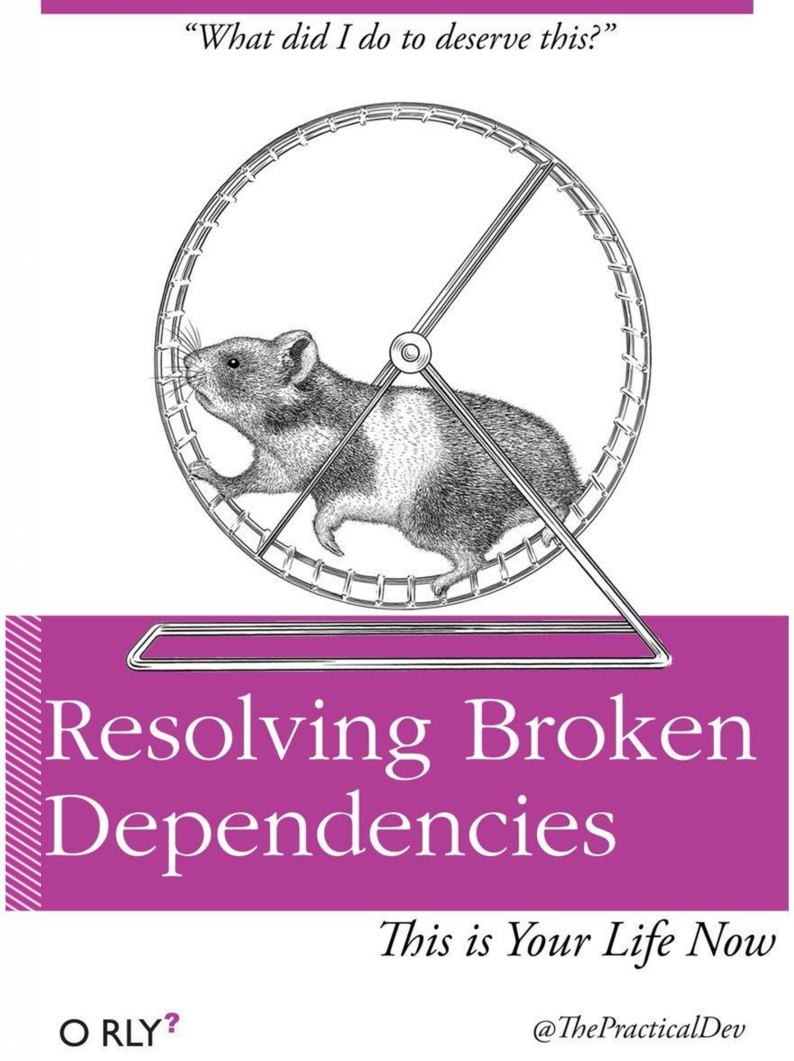
Tools

(google or not?)

Frontend Junior Program - 2022


"What did I do to deserve this?"
Resolving Broken Dependencies
This is Your Life Now
O RLY?
@ThePracticalDev

Introduction

Before we start doing anything (new), we need clear goals

Especially when we want to learn something, or we'd like to prepare a lecture about tooling in FE development. Maybe it could sound surprising, but these are not trivial tasks, either.

* * *

You may ask, why?

First, what you learn now, should be useful you (and for us) in future. And – generally speaking – knowing that how to build a proper tooling setup in FE is definitely not useful.

Shocked? Let me explain…

*Feltett kedvem megcselekszem, ezernyi elem egybe legyen!**

** Let's create a building system, should be bundled every item!*

Tooling – cost / benefit

*Egybetennem kellemetlen, de elemekre szedegethetem!**

But tooling is a must, right?

Sure, it is. But how many times you'd need to do that? Once per 2 years? A little math, presuming that it could take 1 week to complete:

a) you would put a significant effort (lecture + obligatory readings + practicing) which you'd utilize only in 1% of your time in future

b) by the time you would really need it, you should learn the whole thing from start again

c) honestly, it is very uninspiring to understand the awkward philosophy of these toolkits – after a while, you would want to unlearn all of that.

** Before you build, it should be teared!*

Problem solving – the approach

Instead of how to do, let's see how to approach

…any problems. To setup a build system is complex problem – so solving it is not that different from solving other problems.

We need to do it step-by-step, focusing on only one thing, and one thing only.

By doing that, we could mitigate 2 critical issues at once – and these obstacles are real blockers. The chance that you are fighting with at least one of these, is very high.

*E terembe nem mehetsz be, fel – helyette – emeletre!**

** You don't need a door, just an approach for!*

Blocker 1 – underestimation

*Egyszerre meglesz ez, de ereszem fent legyen, egyenesen!**

“It should not be a big deal”

“Everyone else can do that easily.” “I am a professional, so I have to do it in a timely manner.” “If I am good enough it should be done soon.”

* * *

The fact is: we just don’t know. Usually, things are much more complex than it looks like at first. While we could have an educated guess (called estimation), it is not a prediction, and especially not a promise.

Underestimation, however, will lead to rush and unnecessary stress, therefore it could block the progress completely.

** It should be ready any minute, make sure you hold it as it needed.*

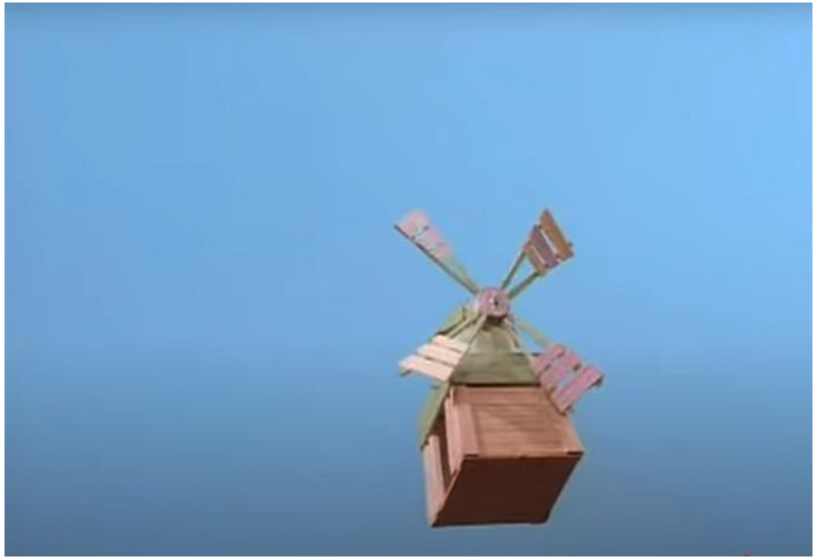
Blocker 2 – overestimation

Usually, there are some problems with *problems*

Let's list just the common ones:

a) I don't understand the problem
b) do I have a blocker? If so, who can resolve that?
c) can I ask for help?
d) can I ask for help 10 times a day?
e) what if I forget something?
f) am I good enough for this? (my knowledge / experience / domain background)

Don't make mistake: you will face with these. Apparently, with all of these, at once. It's no wonder that someone will feel: it is just too much. I cannot do it.



*Egekben e remek kellem, de megszereznem lehetetlen.**

** When the goal is the sky, there is no way to fly that high!*

Problem solving – step by step

So how the step-by-step method can solve these issues?

It is pretty simple: these questions won’t come up at all. Is it too easy? Is it impossible to do?

We just don’t care. We just start doing it without overthinking it.

It does not mean that it should be done ignorantly, or in a careless way. We have to do it precisely, according to our best knowledge and expertise. But we need to focus on only one thing at a time. With providing maximal focus, though.

A well-known application of this method is the Test-driven development (TDD).

Just as an example: how to configure a build system

First, let’s describe our task: *Create a build system!*

Now, we follow the step-by-step approach. The first step of the problem solving is always to clarify and understand the problem:

Q	A
a build system for what?	for a *FE application*
what kind of FE application?	we don’t know yet, presumably a generalized solution will fit
what does it consist of?	*source code*
what kind of source code?	*TypeScript, HTML* (for sure), *CSS* (most likely)
why we need a build system?	to *transpile* TypeScript, to *do something* with HTML, to *compile* CSS (maybe),

“we don’t know yet” – is it a blocker?

yes:
need an answer!

no:
move forward!

Acronyms

TBC - To Be Clarified

$\nearrow$ We can presume that there is an answer, we just don't know. Probably it won't be blocker (we will see) and we can move ahead at this point.

TBD - To Be **Decided**

$\searrow$ We do know that there is no answer, *we have to initiate the process* of having one. It could be a blocker.

This will happen if you miss to clarify / resolve the blockers *before the development*... It basically requires to read and process the entire story description (Acceptance Criteria, Tech Notes, etc.), *before writing any code* lines.

It does not exclude the situations, when blockers come up during the development. It may happen, but that is a completely different story.

Let’s rewrite our story

New information: *FE application, source code, transpile TypeScript, do something with HTML, compile CSS*

Create a build system! → *Create a build system to support a FE application by transpiling TypeScript, doing something HTML, compiling CSS (maybe).*

We have a refined story now! Is that perfect?
Nope, but *good enough* to be able to start.

Let’s do it step by step, again!

Create the repository

we need a *place* for the code:

```
> cd ~
> mkdir mekk
> cd mekk
```

we need *Git* -> install it ->

```
> brew install git
```

```
> git init
```

we need an *editor* (any editor will fit, if that is *WebStorm*)

create a folder src and create *index.html*

```
> mkdir src
> touch src/index.html
```

that is not knife...

...this is a knife

Edit index.html

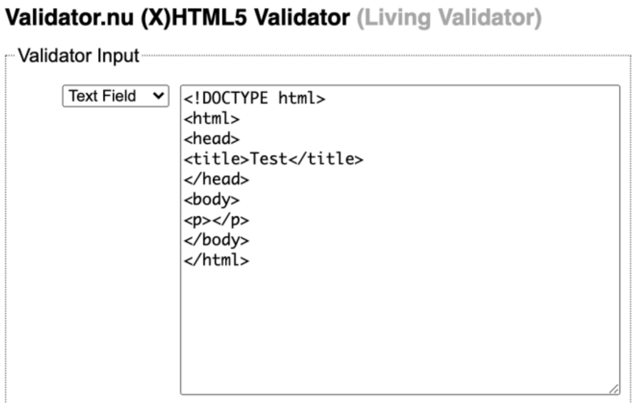
we need a *minimal but valid html file* ->
google html *validator* ->
Validator.nu (X)HTML5 Validator ->
choose “text input” from the dropdown

edit index.html

we could have many html pages ->
we want to have common elements (menu) -> dynamic html ->
server side? -> nope, client side -> SPA ->
Angular? -> hell, no -> *React* -> install React ->
need a package manager -> yarn? -> nope, *npm* ->
install npm

install React -> need package.json ->
> npm init -y
> npm i react

create react file -> jsx -> need a compiler -> TypeScript ->
> npm i typescript

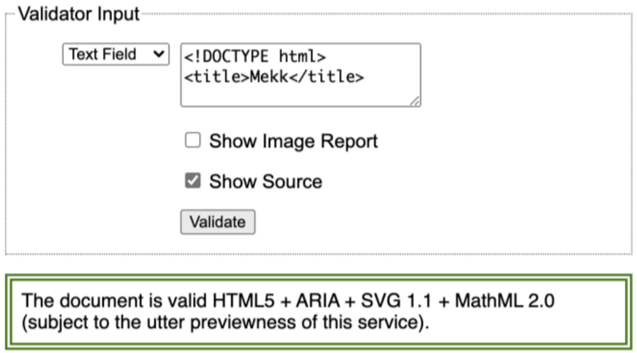

Validator.nu (X)HTML5 Validator (Living Validator)

Validator Input

Text Field

```
<!DOCTYPE html>
<html>
<head>
<title>Test</title>
</head>
<body>
<p></p>
</body>
</html>
```

that is not ~~knife~~ minimal html…


Validator Input

Text Field

```
<!DOCTYPE html>
<title>Mekk</title>
```

☐ Show Image Report

☑ Show Source

Validate

The document is valid HTML5 + ARIA + SVG 1.1 + MathML 2.0 (subject to the utter previewness of this service).

…this is a minimal html

Create a component - prerequisites

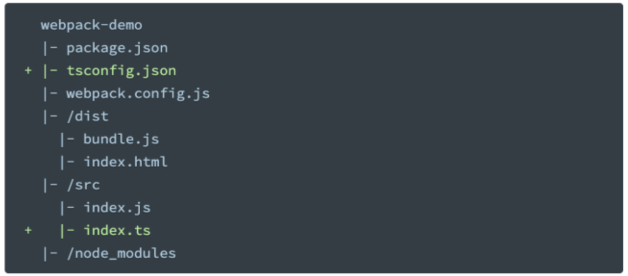
we need a minimal, but valid React component ->
React page -> ahh, we have a Tutorial -> too complex -> check Docs ->
Docs is garbage (forces CDN version) -> read more -> create-react-app? -> skip ->
read more -> found a link how to create a toolchain -> uses *Webpack* ->
we have an html, nice! -> read more -> uses Babel, grrr, useless ->
google for “typescript webpack” -> TypeScript | webpack ->
> npm install --save-dev typescript ts-loader

create tsconfig.json
create webpack.config.js

run webpack -> w8 a minute, we don’t have Webpack yet ->
docs, Getting started -> install ->
> npm install webpack webpack-cli --save-dev

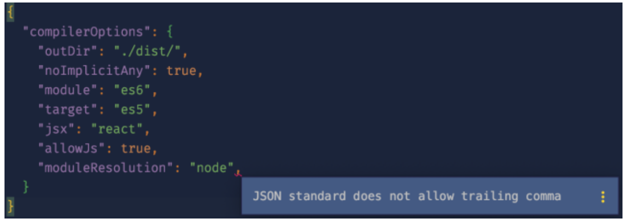
edit package.json according to the doc

stop there and evaluate what we have so far



```
  webpack-demo
  |- package.json
+ |- tsconfig.json
  |- webpack.config.js
  |- /dist
    |- bundle.js
    |- index.html
  |- /src
    |- index.js
+   |- index.ts
  |- /node_modules
```

we have something now…

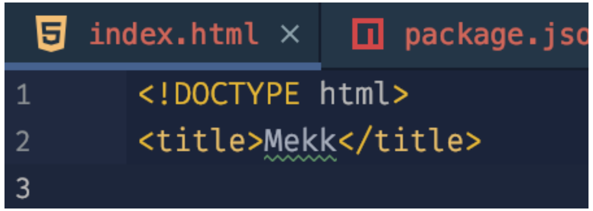


```
{
  "compilerOptions": {
    "outDir": "./dist/",
    "noImplicitAny": true,
    "module": "es6",
    "target": "es5",
    "jsx": "react",
    "allowJs": true,
    "moduleResolution": "node",
  }
}
```

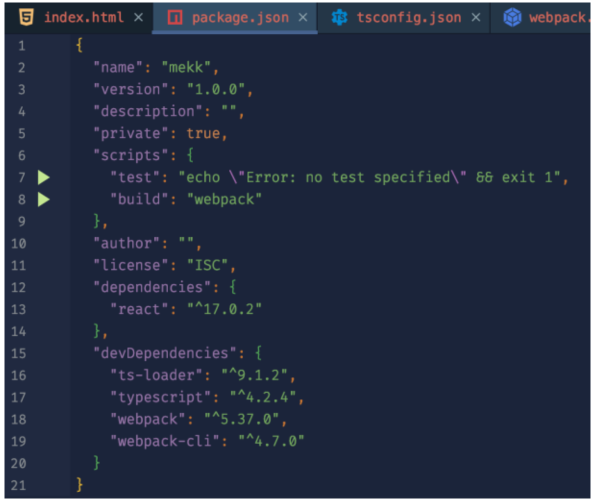
JSON standard does not allow trailing comma

out of the box smarter than the Webpack docs

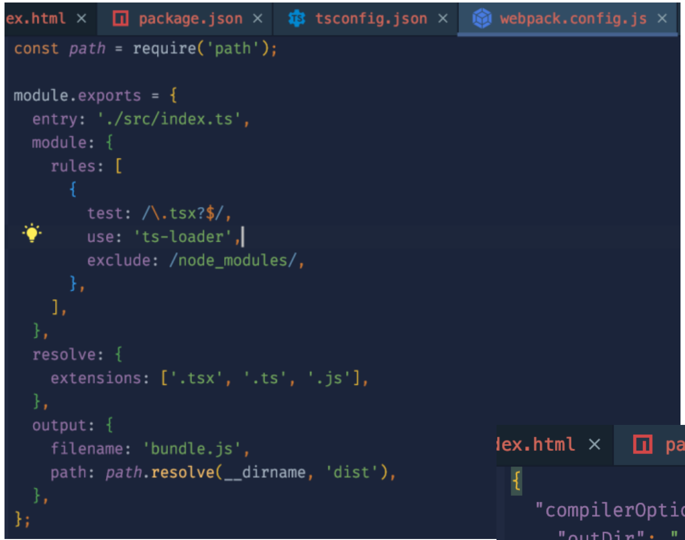
Check what we have so far



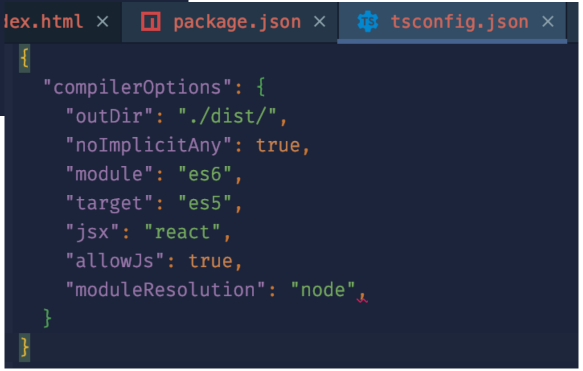
```html
<!DOCTYPE html>
<title>Mekk</title>
```



```json
{
  "name": "mekk",
  "version": "1.0.0",
  "description": "",
  "private": true,
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "react": "^17.0.2"
  },
  "devDependencies": {
    "ts-loader": "^9.1.2",
    "typescript": "^4.2.4",
    "webpack": "^5.37.0",
    "webpack-cli": "^4.7.0"
  }
}
```



```js
const path = require('path');

module.exports = {
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
    extensions: ['.tsx', '.ts', '.js'],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```



```json
{
  "compilerOptions": {
    "outDir": "./dist/",
    "noImplicitAny": true,
    "module": "es6",
    "target": "es5",
    "jsx": "react",
    "allowJs": true,
    "moduleResolution": "node",
  }
}
```

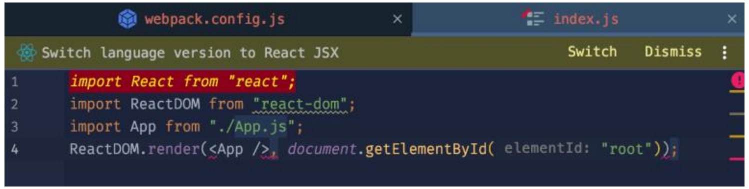
Create a component – really

go back to toolchain docs, that was fine, except the Babel -> we have and index.js there -> create index.js in src and copy its content

rename index.js -> index.tsx

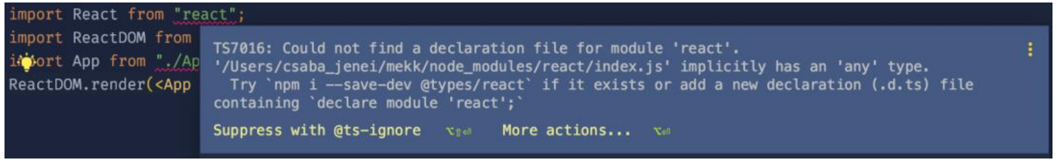
> npm i --save-dev @types/react

add ”allowWhateverIsThis” to tsconfig



```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App.js";
ReactDOM.render(<App />, document.getElementById( elementId: "root"));
```

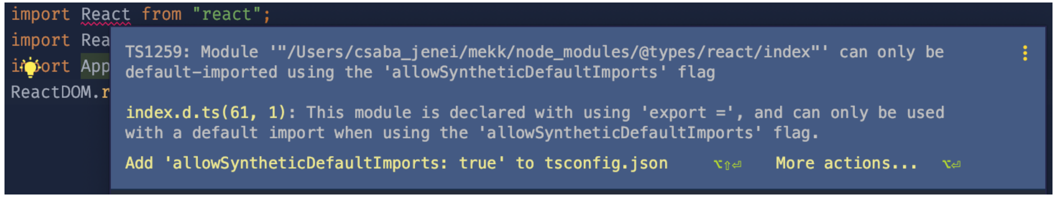
ahh, the tutorial is stupid, it is NOT a js file…



```
import React from "react";
import ReactDOM from
import App from "./Ap
ReactDOM.render(<App
```

```
TS7016: Could not find a declaration file for module 'react'.
'/Users/csaba_jenei/mekk/node_modules/react/index.js' implicitly has an 'any' type.
  Try `npm i --save-dev @types/react` if it exists or add a new declaration (.d.ts) file
containing `declare module 'react';`
```

let’s have typings!



```
import React from "react";
import Rea
import App
ReactDOM.r
```

```
TS1259: Module '"/Users/csaba_jenei/mekk/node_modules/@types/react/index"' can only be
default-imported using the 'allowSyntheticDefaultImports' flag

index.d.ts(61, 1): This module is declared with using 'export =', and can only be used
with a default import when using the 'allowSyntheticDefaultImports' flag.
```

Houston, we have a problem…

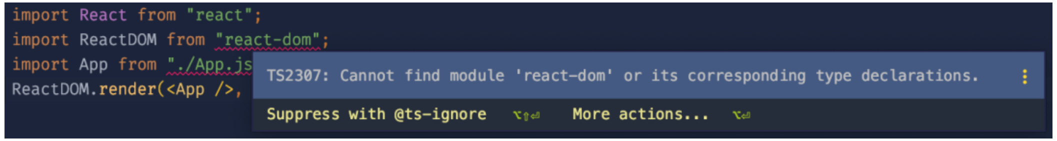
Create a component – I mean really-really

try this adding react-dom
> npm i react-dom

add @types/react-dom
> npm i --save-dev @types/react-dom

add App.ts, not App.js
edit the App.js reference in index.tsx
add App.css, but comment out from App.ts at this point

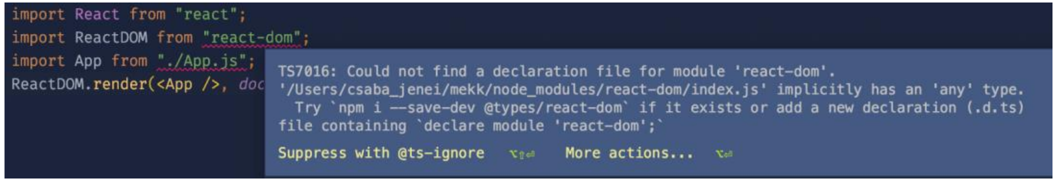
check our status



```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App.js
ReactDOM.render(<App />,
```

```
TS2307: Cannot find module 'react-dom' or its corresponding type declarations.
Suppress with @ts-ignore   More actions...
```

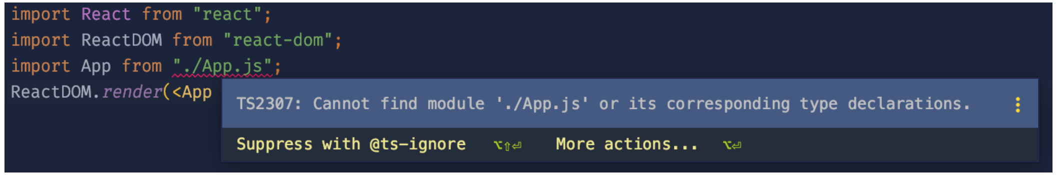
still complaining, nothing can be good enough…



```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App.js";
ReactDOM.render(<App />, doc
```

```
TS7016: Could not find a declaration file for module 'react-dom'.
'/Users/csaba_jenei/mekk/node_modules/react-dom/index.js' implicitly has an 'any' type.
  Try `npm i --save-dev @types/react-dom` if it exists or add a new declaration (.d.ts)
file containing `declare module 'react-dom';`
Suppress with @ts-ignore   More actions...
```

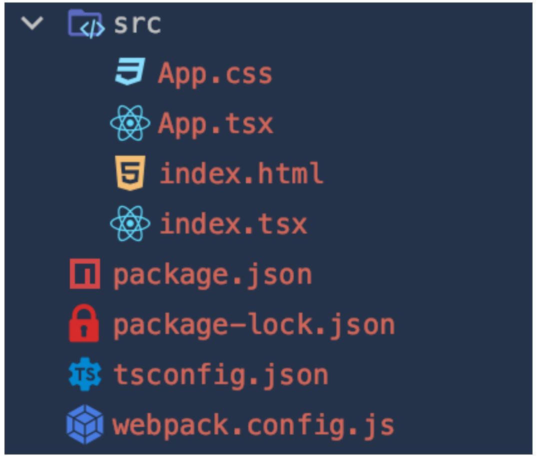
why am I not surprised?



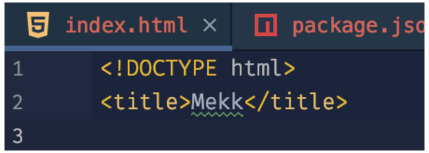
```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App.js";
ReactDOM.render(<App
```

```
TS2307: Cannot find module './App.js' or its corresponding type declarations.
Suppress with @ts-ignore   More actions...
```

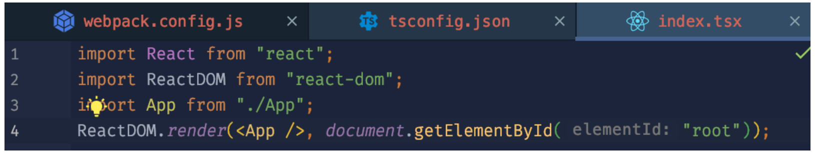
good, that is expected



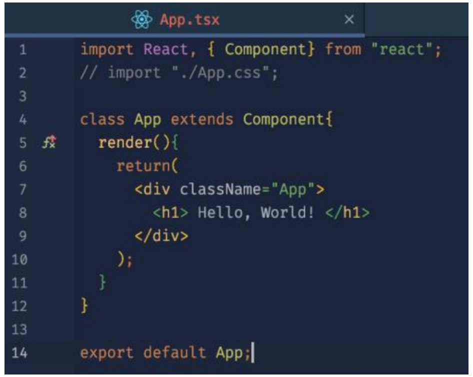
```
src
    App.css
    App.tsx
    index.html
    index.tsx
  package.json
  package-lock.json
  tsconfig.json
  webpack.config.js
```



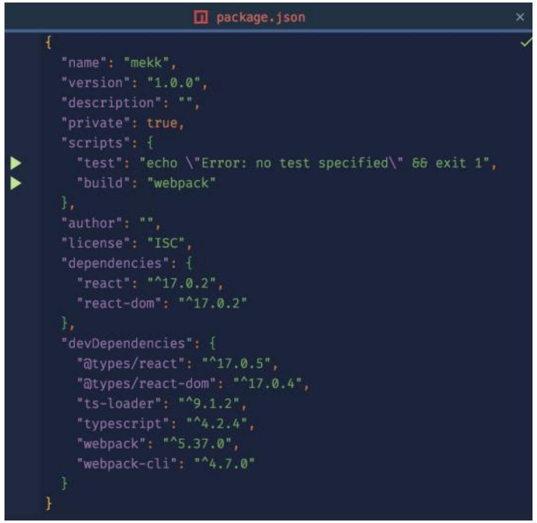
```html
<!DOCTYPE html>
<title>Mekk</title>

```



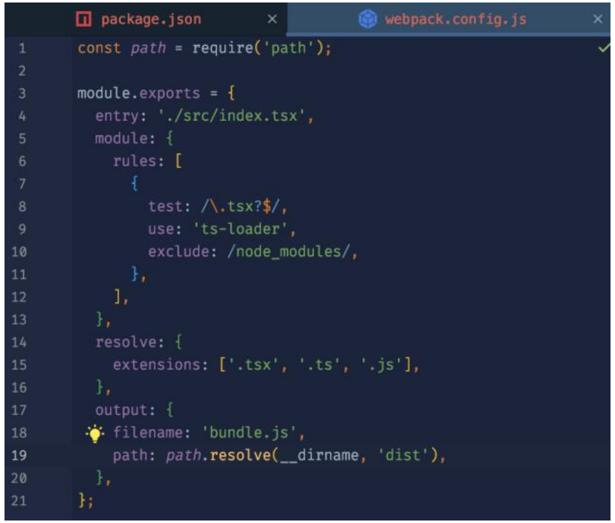
```tsx
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
ReactDOM.render(<App />, document.getElementById( elementId: "root"));
```



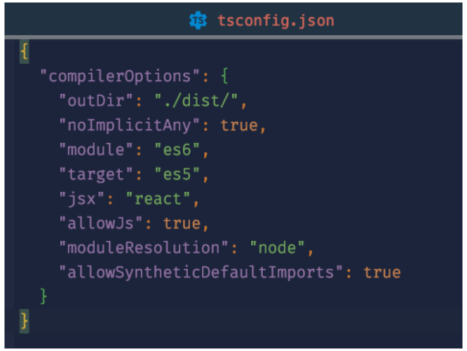
```tsx
import React, { Component} from "react";
// import "./App.css";

class App extends Component{
  render(){
    return(
      <div className="App">
        <h1> Hello, World! </h1>
      </div>
    );
  }
}

export default App;
```



```json
{
  "name": "mekk",
  "version": "1.0.0",
  "description": "",
  "private": true,
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2"
  },
  "devDependencies": {
    "@types/react": "^17.0.5",
    "@types/react-dom": "^17.0.4",
    "ts-loader": "^9.1.2",
    "typescript": "^4.2.4",
    "webpack": "^5.37.0",
    "webpack-cli": "^4.7.0"
  }
}
```



```js
const path = require('path');

module.exports = {
  entry: './src/index.tsx',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  resolve: {
    extensions: ['.tsx', '.ts', '.js'],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```



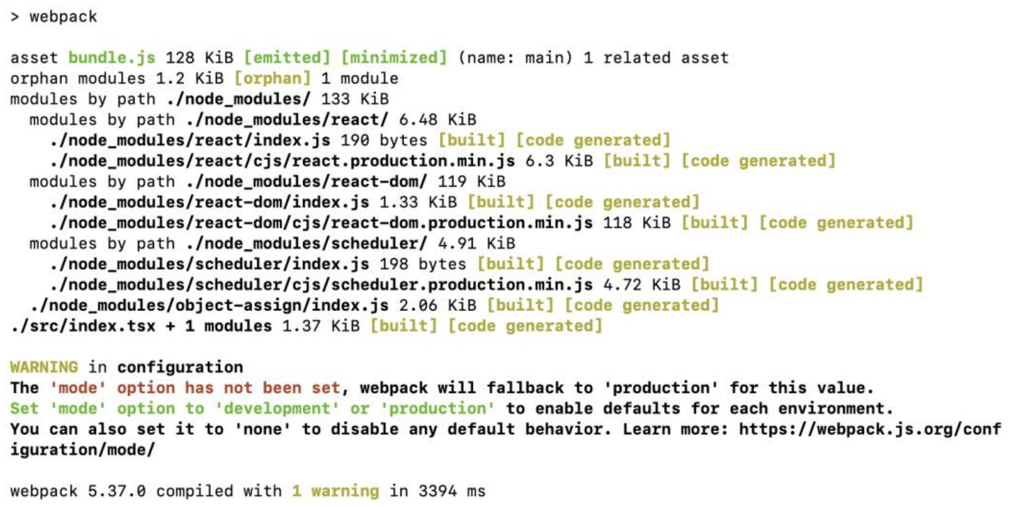
```json
{
  "compilerOptions": {
    "outDir": "./dist/",
    "noImplicitAny": true,
    "module": "es6",
    "target": "es5",
    "jsx": "react",
    "allowJs": true,
    "moduleResolution": "node",
    "allowSyntheticDefaultImports": true
  }
}
```

Build it!

build the app
> npm run build

add mode to webpack.config.js
build again to check
> npm run build

no warnings -> check the bundle



```
> webpack

asset bundle.js 128 KiB [emitted] [minimized] (name: main) 1 related asset
orphan modules 1.2 KiB [orphan] 1 module
modules by path ./node_modules/ 133 KiB
  modules by path ./node_modules/react/ 6.48 KiB
    ./node_modules/react/index.js 190 bytes [built] [code generated]
    ./node_modules/react/cjs/react.production.min.js 6.3 KiB [built] [code generated]
  modules by path ./node_modules/react-dom/ 119 KiB
    ./node_modules/react-dom/index.js 1.33 KiB [built] [code generated]
    ./node_modules/react-dom/cjs/react-dom.production.min.js 118 KiB [built] [code generated]
  modules by path ./node_modules/scheduler/ 4.91 KiB
    ./node_modules/scheduler/index.js 198 bytes [built] [code generated]
    ./node_modules/scheduler/cjs/scheduler.production.min.js 4.72 KiB [built] [code generated]
  ./node_modules/object-assign/index.js 2.06 KiB [built] [code generated]
./src/index.tsx + 1 modules 1.37 KiB [built] [code generated]

WARNING in configuration
The 'mode' option has not been set, webpack will fallback to 'production' for this value.
Set 'mode' option to 'development' or 'production' to enable defaults for each environment.
You can also set it to 'none' to disable any default behavior. Learn more: https://webpack.js.org/conf
iguration/mode/

webpack 5.37.0 compiled with 1 warning in 3394 ms
```

it's green, hurray!, still we have a warning…

Add CSS

add the bundle and the anchor into index.html

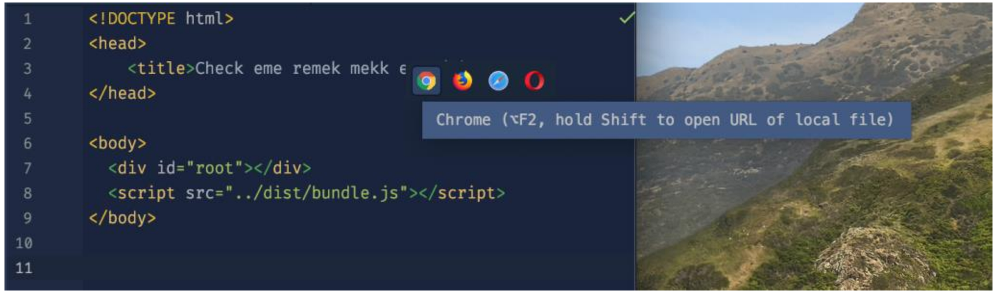
run the app (we are fine with the built-in web-server in Webpack as for now)

it's working! our first React + TypeScript app in a Webpack toolchain -> add CSS back

build the app
> npm run build

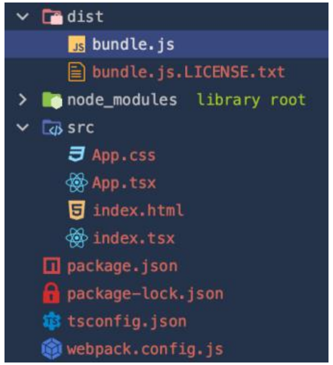
error! -> we need CSS loader -> go back to toolchain docs ->
add CSS loader into webpack config

install CSS loader
> npm i css-loader style-loader
> npm run build

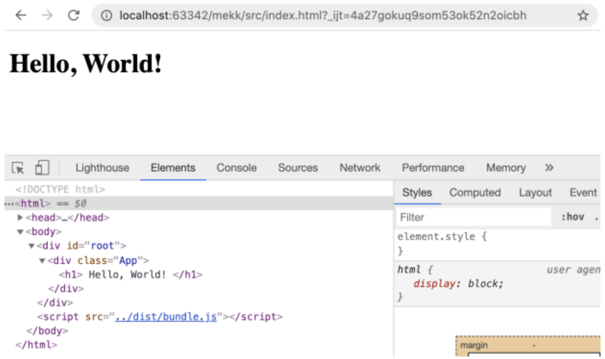


```
<!DOCTYPE html>
<head>
    <title>Check eme remek mekk e
</head>

<body>
  <div id="root"></div>
  <script src="../dist/bundle.js"></script>
</body>
```

Chrome (⌥F2, hold Shift to open URL of local file)



```
v dist
    bundle.js
    bundle.js.LICENSE.txt
> node_modules  library root
v src
    App.css
    App.tsx
    index.html
    index.tsx
  package.json
  package-lock.json
  tsconfig.json
  webpack.config.js
```

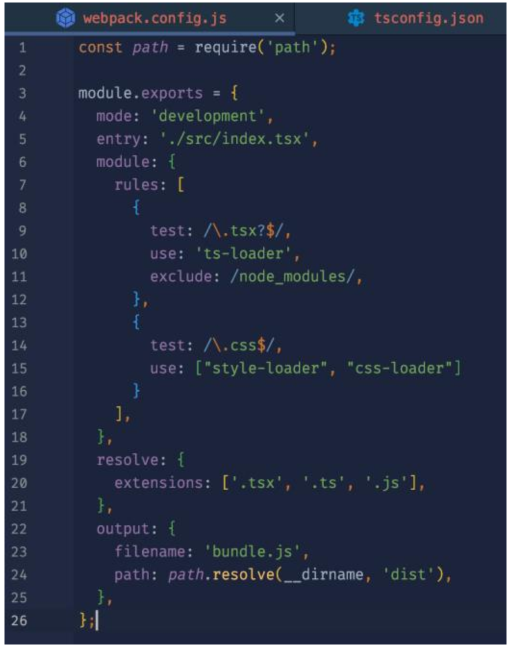


localhost:63342/mekk/src/index.html?_ijt=4a27gokuq9som53ok52n2oicbh

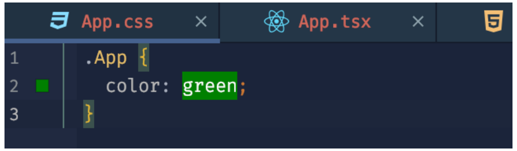
Hello, World!

```
<!DOCTYPE html>
<html> == $0
  <head>…</head>
  <body>
    <div id="root">
      <div class="App">
        <h1> Hello, World! </h1>
      </div>
    </div>
    <script src="../dist/bundle.js"></script>
  </body>
</html>
```

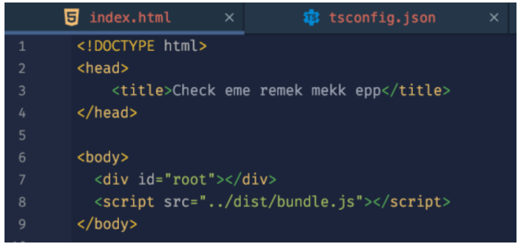
```
element.style {
}
html {                user agen
  display: block;
}
```



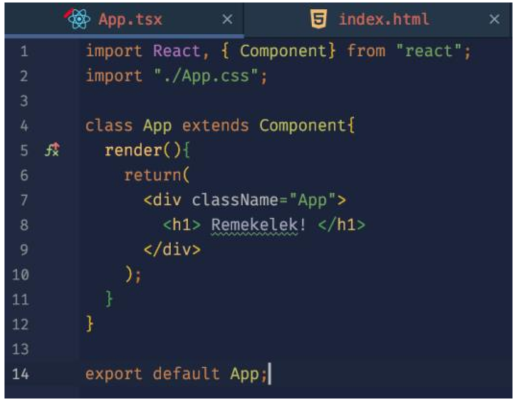
```js
const path = require('path');

module.exports = {
  mode: 'development',
  entry: './src/index.tsx',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"]
      }
    ],
  },
  resolve: {
    extensions: ['.tsx', '.ts', '.js'],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```



```css
.App {
  color: green;
}
```

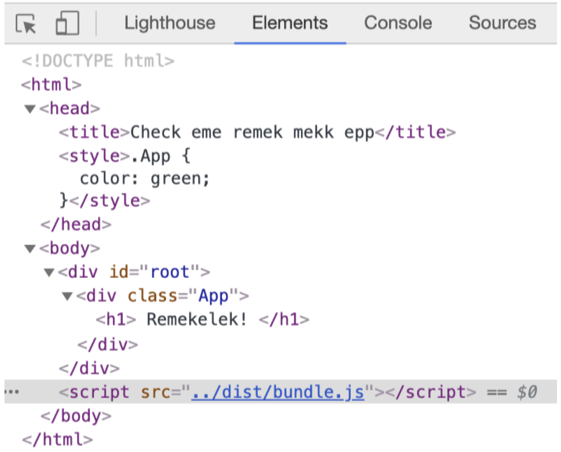


```html
<!DOCTYPE html>
<head>
    <title>Check eme remek mekk epp</title>
</head>

<body>
  <div id="root"></div>
  <script src="../dist/bundle.js"></script>
</body>
```



```tsx
import React, { Component} from "react";
import "./App.css";

class App extends Component{
  render(){
    return(
      <div className="App">
        <h1> Remekelek! </h1>
      </div>
    );
  }
}

export default App;
```

Remekelek!



```html
<!DOCTYPE html>
<html>
▼<head>
    <title>Check eme remek mekk epp</title>
    <style>.App {
      color: green;
    }</style>
  </head>
▼<body>
  ▼<div id="root">
    ▼<div class="App">
        <h1> Remekelek! </h1>
      </div>
    </div>
...   <script src="../dist/bundle.js"></script> == $0
  </body>
</html>
```

Always check the story description!

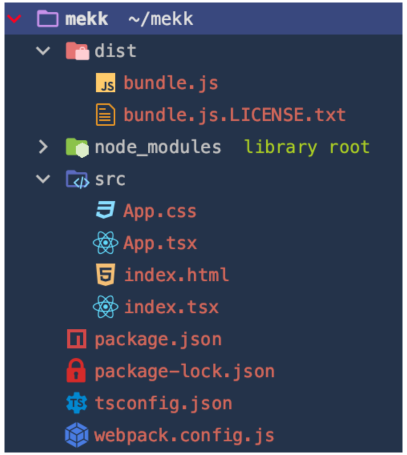
we need bundle the index.html as well ->
search for "index.html" in webpack docs ->
Getting Started | webpack ->
Output Management | webpack ->
GitHub - jantimon/html-webpack-plugin ->

```
> npm i --save-dev html-webpack-plugin
```

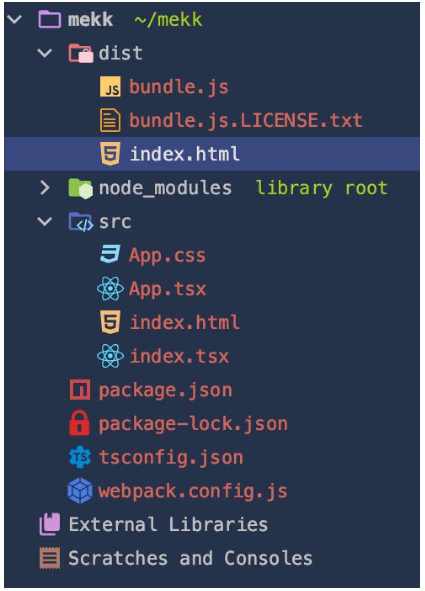
add html-webpack-plugin config into webpack config

```
> npm run build
```

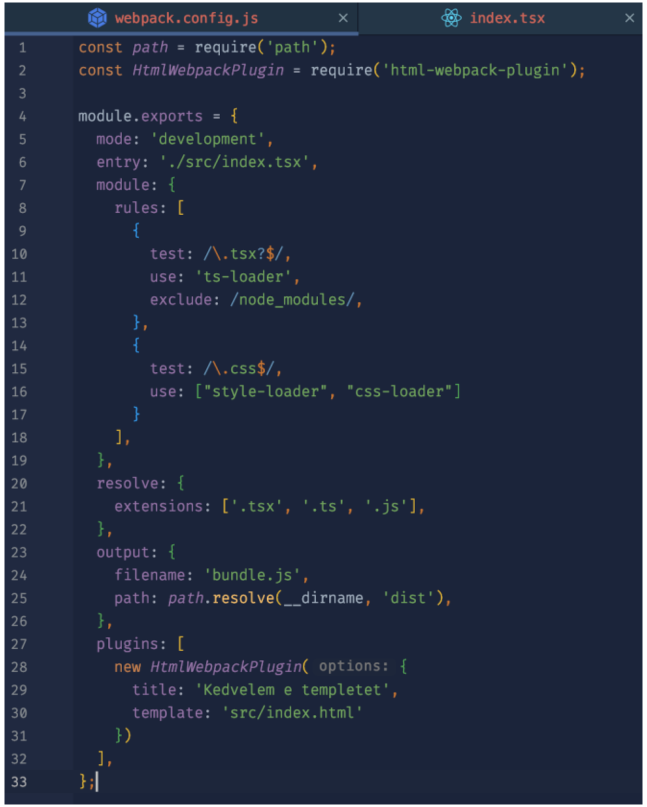
Create a build system to support a FE application by transpiling TypeScript, doing something HTML, compiling CSS (maybe).



```
mekk ~/mekk
  dist
    bundle.js
    bundle.js.LICENSE.txt
  node_modules  library root
  src
    App.css
    App.tsx
    index.html
    index.tsx
  package.json
  package-lock.json
  tsconfig.json
  webpack.config.js
```

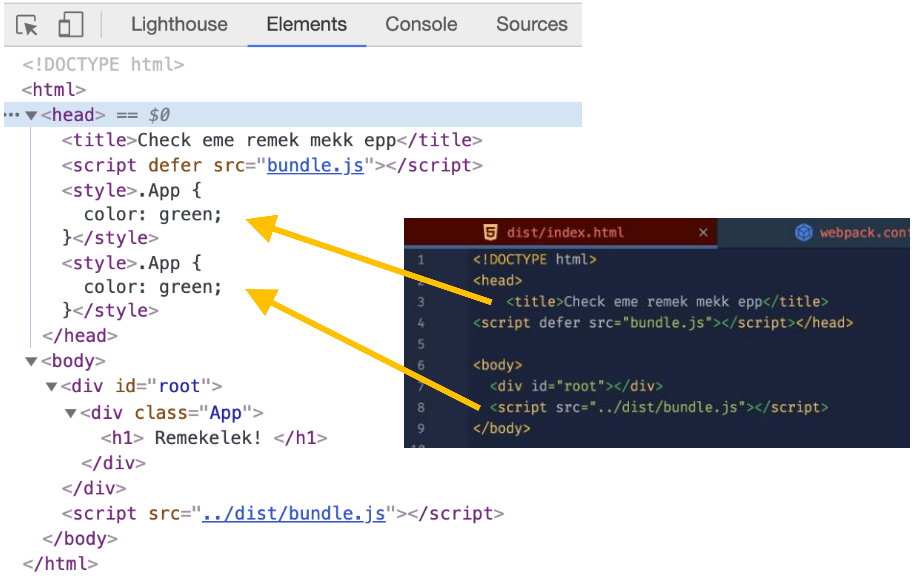


```
mekk ~/mekk
  dist
    bundle.js
    bundle.js.LICENSE.txt
    index.html
  node_modules  library root
  src
    App.css
    App.tsx
    index.html
    index.tsx
  package.json
  package-lock.json
  tsconfig.json
  webpack.config.js
External Libraries
Scratches and Consoles
```



```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  mode: 'development',
  entry: './src/index.tsx',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"]
      }
    ],
  },
  resolve: {
    extensions: ['.tsx', '.ts', '.js'],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  plugins: [
    new HtmlWebpackPlugin( options: {
      title: 'Kedvelem e templetet',
      template: 'src/index.html'
    })
  ],
};
```

Remekelek!



```
<!DOCTYPE html>
<html>
··· ▼<head> == $0
    <title>Check eme remek mekk epp</title>
    <script defer src="bundle.js"></script>
    <style>.App {
      color: green;
    }</style>
    <style>.App {
      color: green;
    }</style>
  </head>
  ▼<body>
    ▼<div id="root">
      ▼<div class="App">
          <h1> Remekelek! </h1>
        </div>
      </div>
    <script src="../dist/bundle.js"></script>
  </body>
</html>
```

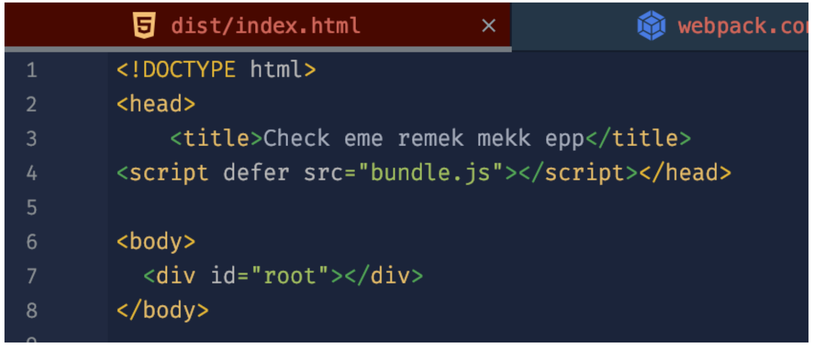
```
<!DOCTYPE html>
<head>
  <title>Check eme remek mekk epp</title>
<script defer src="bundle.js"></script></head>

<body>
  <div id="root"></div>
  <script src="../dist/bundle.js"></script>
</body>
```

Fix the index.html template + check the AC again

fix index.html ->
remove the original reference for the bundle

> npm run build

WE
ARE
DONE!



```html
<!DOCTYPE html>
<head>
    <title>Check eme remek mekk epp</title>
<script defer src="bundle.js"></script></head>

<body>
  <div id="root"></div>
</body>
```

Create a build system to support a FE application by transpiling TypeScript, doing something HTML, compiling CSS (maybe).

Bonus - eslint

need quality gates -> add eslint ->
GitHub - typescript-eslint/typescript-eslint ->
typescript-eslint/packages/eslint-plugin

> npm i --save-dev typescript @typescript-eslint/parser @typescript-eslint/eslint-plugin

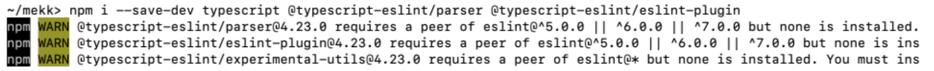
> npm i --save-dev eslint

add .eslintrc with default content -> how to run? -> typescript-eslint/README.md at master · typescript-eslint/typescript-eslint · GitHub

found a new config: .eslintignore ->
add it with default content

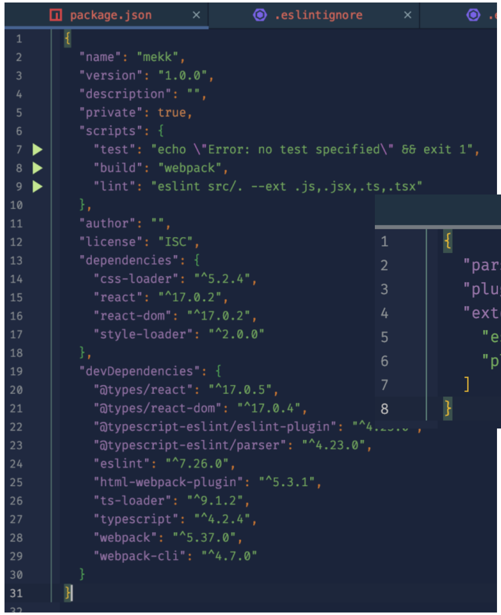
how to run? ->
add a command to package.json

run ->
> npm run lint

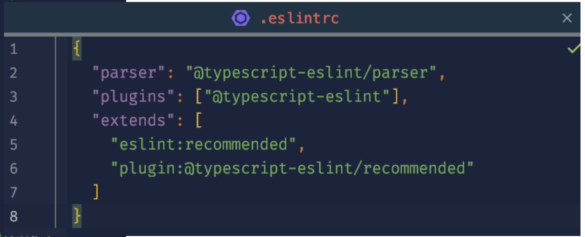


```
~/mekk> npm i --save-dev typescript @typescript-eslint/parser @typescript-eslint/eslint-plugin
npm WARN @typescript-eslint/parser@4.23.0 requires a peer of eslint@^5.0.0 || ^6.0.0 || ^7.0.0 but none is installed.
npm WARN @typescript-eslint/eslint-plugin@4.23.0 requires a peer of eslint@^5.0.0 || ^6.0.0 || ^7.0.0 but none is ins
npm WARN @typescript-eslint/experimental-utils@4.23.0 requires a peer of eslint@* but none is installed. You must ins
```

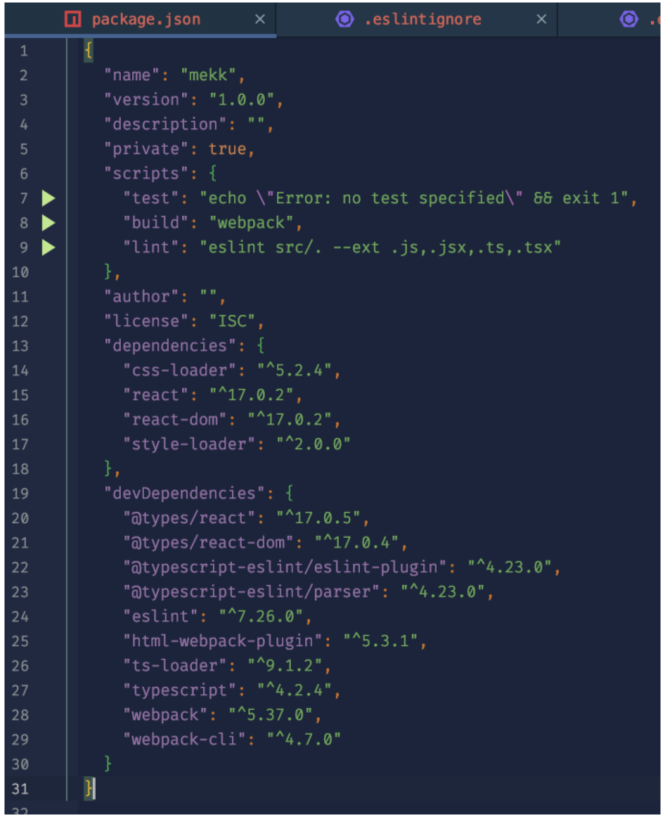
opps!



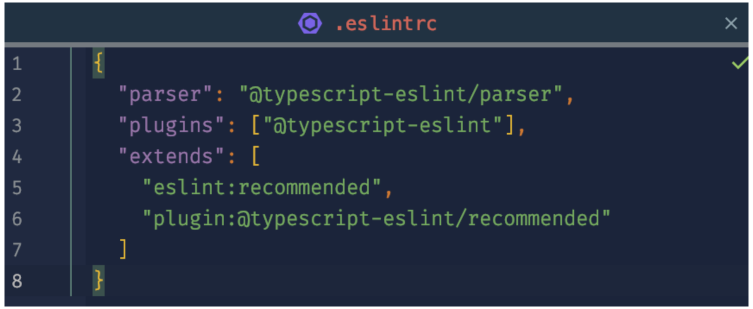
```
{
  "name": "mekk",
  "version": "1.0.0",
  "description": "",
  "private": true,
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack",
    "lint": "eslint src/. --ext .js,.jsx,.ts,.tsx"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "css-loader": "^5.2.4",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "style-loader": "^2.0.0"
  },
  "devDependencies": {
    "@types/react": "^17.0.5",
    "@types/react-dom": "^17.0.4",
    "@typescript-eslint/eslint-plugin": "^4.
    "@typescript-eslint/parser": "^4.23.0",
    "eslint": "^7.26.0",
    "html-webpack-plugin": "^5.3.1",
    "ts-loader": "^9.1.2",
    "typescript": "^4.2.4",
    "webpack": "^5.37.0",
    "webpack-cli": "^4.7.0"
  }
}
```



```
{
  "parser": "@typescript-eslint/parser",
  "plugins": ["@typescript-eslint"],
  "extends": [
    "eslint:recommended",
    "plugin:@typescript-eslint/recommended"
  ]
}
```

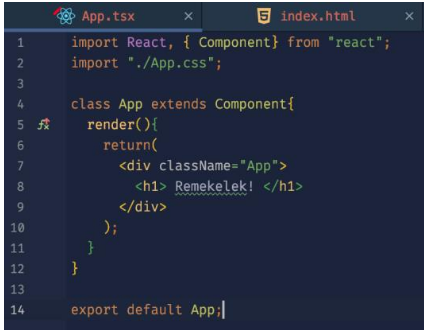


```json
{
  "name": "mekk",
  "version": "1.0.0",
  "description": "",
  "private": true,
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack",
    "lint": "eslint src/. --ext .js,.jsx,.ts,.tsx"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "css-loader": "^5.2.4",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "style-loader": "^2.0.0"
  },
  "devDependencies": {
    "@types/react": "^17.0.5",
    "@types/react-dom": "^17.0.4",
    "@typescript-eslint/eslint-plugin": "^4.23.0",
    "@typescript-eslint/parser": "^4.23.0",
    "eslint": "^7.26.0",
    "html-webpack-plugin": "^5.3.1",
    "ts-loader": "^9.1.2",
    "typescript": "^4.2.4",
    "webpack": "^5.37.0",
    "webpack-cli": "^4.7.0"
  }
}
```



```json
{
  "parser": "@typescript-eslint/parser",
  "plugins": ["@typescript-eslint"],
  "extends": [
    "eslint:recommended",
    "plugin:@typescript-eslint/recommended"
  ]
}
```

```
  5:3  warning  Missing return type on function  @typescri
pt-eslint/explicit-module-boundary-types

✖ 1 problem (0 errors, 1 warning)
```



```tsx
import React, { Component} from "react";
import "./App.css";

class App extends Component{
  render(){
    return(
      <div className="App">
        <h1> Remekelek! </h1>
      </div>
    );
  }
}

export default App;
```

Q&A

edu_hu@epam.com