**NODE + EXPRESS WITH TYPESCRIPT**

The initial setup for node JS application is similar to what we do to create a JavaScript nodejs application.

To enable the typescript support we have to install some dev dependencies that is **@types/node , @types/express** .

They enable the typescript support to your application.

**Points to Remember:**

* When we utilize node JS traditional import syntax we will not going to get typescript type support in our application.
* In order to take benefits of wide range of types and IDE auto completion we should utilize the **ES6 import/exports**  even in our node application.

*const express = require(“express”) ❌*

*import express from ‘express’ ✅*

* We should set the **moduleResolution** property of the tsconfig.json file to **node**.
* Since we have set the module resolution to node we no longer need to keep .js extension in import statements.

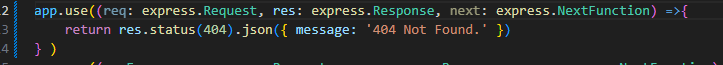
*import sampleModule from ‘./sampleModule.js’ (without moduleResolution)*

*import sampleModule from ‘./sampleModule’ (with moduleResolution)*

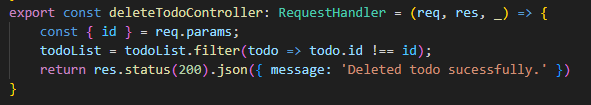
* Webpack is only utilized in the browser applications to bundle the code.

# **Middleware and Types:**

* In case of only node server the type of **req, res** will be node Specific i.e. **Request** and **Response**.
* But when Node + Express server application the types of **req**, **res** and **next**  will be express specific i.e. **express.Request, express.Response** and **express.NextFunction**



* Just like in react all the events are React specific similar in express all the middleware arguments are express specific.
* The **RequestHandler** type is used to specify that the function is a middleware. With it we no longer have to explicitily define the type of arguments.



* Similar to react where the type of the functional Component is **React.FC<{}>,**  in node+express the type of middlewares is **express.RequestHandler.**

# **Middleware’s Type (RequestHandler<P, RS, RE, RQ>):**

* The type **express.RequestHandler<{},{},{},{}>**  is the type of express that tells typescript that this function will acts as a middleware.
* The **RequestHandler** is the type of middleware functions.
* Defining the types of params, response-body, request-body and request query will give us e better type support and IDE Autocompletion.
* Along with it **RequestHandler**  is also an generic type:

*RequestHandler<P = type of params,RS = type of response body, RE = type of request body, RQ = type of request query>*

