# Web Application Development

Node.js - React

1

1

## Node Project 1: React

```
express --no-view Node1-React
cd Node1-React
npm install
npm start
```

Open http://localhost:3000 in your browser.
You should see **Welcome to Express**.

```
jung@SCICPSC2222-MAC Node1-React % npm start

> node1-react@0.0.0 start
> node ./bin/www
```

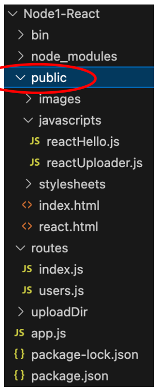
**If you don't know why we are doing this, refer to:**
**https://seikyung.com/README.html**

2

2

- Download `Node1-React-files.zip` from BB
- Did you put all files appropriate folder?

**public/react.html**
**public/javascripts/reactHello.js**
**public/javascripts/reactUploader.js**

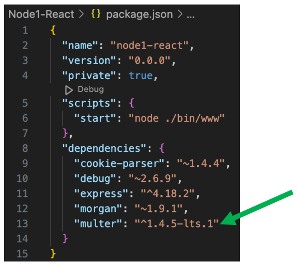


```
Node1-React
  bin
  node_modules
  public
    images
    javascripts
      reactHello.js
      reactUploader.js
    stylesheets
    index.html
    react.html
  routes
    index.js
    users.js
  uploadDir
  app.js
  package-lock.json
  package.json
```

3

```
Node1-React % npm install multer --save
```

Installing **“`multer`”** package for file upload
**`-- save`** will update **`package.json`**



```
Node1-React > {} package.json > ...
{
  "name": "node1-react",
  "version": "0.0.0",
  "private": true,
  ▷ Debug
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "^4.18.2",
    "morgan": "~1.9.1",
    "multer": "^1.4.5-lts.1"
  }
}
```

4

4

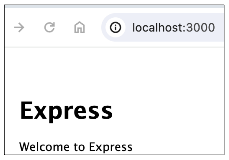
## routes/index.js

```
// Do NOT replace, add this code right before module.exports = router;

// require means include package
var multer  = require('multer');

// will create a folder name "uploadDir"
var upload = multer({ dest: 'uploadDir' });

// HTTP POST request from http://localhost:3000/upload is received
// "file_up" is the input field name from the HTML form
// selected file will be uploaded into the destination folder
// upload object is from multer package

router.post('/upload', upload.single('file_up'), function(req, res) {

  var message =  "This will show up in your browser. "

  res.send(message);

});
```

5

5

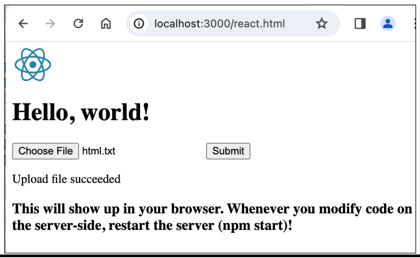
```
Node1-React % npm start

> node1-react@0.0.0 start
> node ./bin/www
```



localhost:3000

**Express**

Welcome to Express



localhost:3000/react.html

**Hello, world!**

Choose File html.txt Submit

Upload file succeeded

**This will show up in your browser. Whenever you modify code on the server-side, restart the server (npm start)!**

6

https://react.dev/learn

public/react.html
public/javascripts/reactHello.js
public/javascripts/reactUploader.js

## React

- It comes in two parts:
  - the first is the React Core module, the one that is responsible for dealing with React components, their state manipulation, etc.
  - The second is the ReactDOM module, which deals with converting React components to a DOM that a browser can understand.
- These two libraries can be found in a CDN that makes all open-source JavaScript libraries available online.

```
React
"https://unpkg.com/react@18/umd/react.development.js"

ReactDOM
"https://unpkg.com/react-dom@18/umd/react-dom.development.js"
```

7

7

## JavaScript XML or JavaScript Syntax Extension

- React uses a markup language called `JSX`.
  - `JSX` looks very much like HTML.
- Browsers do not understand `JSX.` It should be transformed into regular `javaScript`.
  - To do this, a compiler is needed. The compiler that does this is `Babel`.
  - We should ideally pre-compile the code and inject it into the browser, but for simple example like "Hello World", `Babel` provides a standalone compiler that can be used in the browser.
  - The compiler also needs to be told which scripts should be transformed, so must include `type="text/babel"`

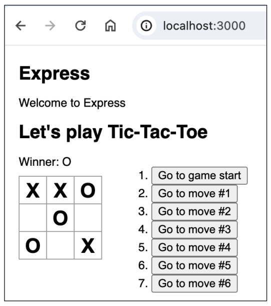
https://react.dev/learn/writing-markup-with-jsx
https://en.wikipedia.org/wiki/JSX_(JavaScript)
https://www.w3schools.com/react/react_jsx.asp

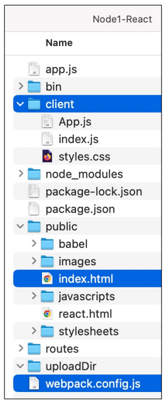
```
"https://unpkg.com/@babel/standalone/babel.min.js"
```

8

## React: Tic-Tac-Toe

https://react.dev/learn/tutorial-tic-tac-toe





9

9

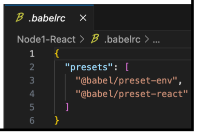
1. **Install React and React DOM**: React and React DOM are the main packages you need for building React applications.

```
Node1-React % npm install  react react-dom --save
```

2. **Install Babel**: Install Babel and its presets that allow you to use modern JavaScript features and JSX syntax.

```
npm install  @babel/core @babel/preset-env @babel/preset-react babel-loader --s
```

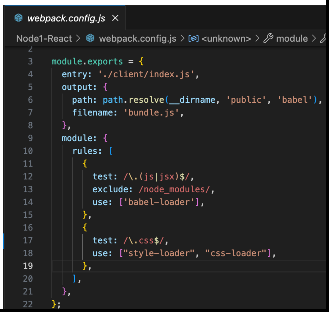
3. **Create a Babel Configuration File**: Create a `.babelrc` file in the root of your project directory. (You can find it from download ZIP file from BlackBoard. Open gitbash or terminal (Mac users) to see the file. It is hidden file.



```
{
  "presets": [
    "@babel/preset-env",
    "@babel/preset-react"
  ]
}
```

10

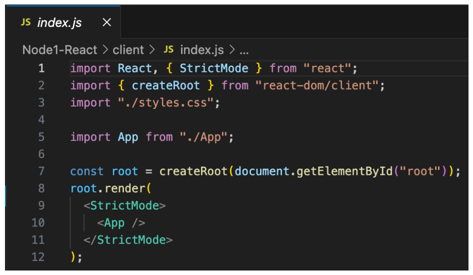
**4. Set Up Webpack**: Webpack is a module bundler that helps bundle your JavaScript files for usage in a browser. Create a `webpack.config.js` file in the root of your project directory. (You can find it from download ZIP file from BlackBoard)



```
module.exports = {
  entry: './client/index.js',
  output: {
    path: path.resolve(__dirname, 'public', 'babel'),
    filename: 'bundle.js',
  },
  module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        exclude: /node_modules/,
        use: ['babel-loader'],
      },
      {
        test: /\.css$/,
        use: ["style-loader", "css-loader"],
      },
    ],
  },
};
```

11

**5. Create Your React Component**: Create a directory named `client` in the root of your project directory. Inside this directory, create a file named `index.js` and write your React component. (You can find it from download ZIP file from BlackBoard)

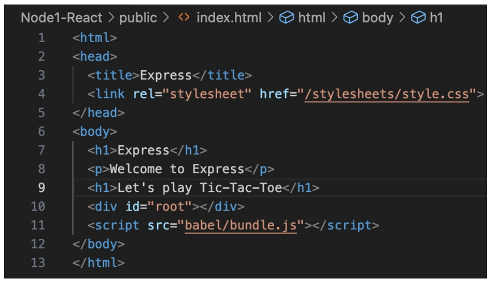


```
import React, { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./styles.css";

import App from "./App";

const root = createRoot(document.getElementById("root"));
root.render(
  <StrictMode>
    <App />
  </StrictMode>
);
```

12

12

**6. Modify an HTML File**: Modify an `index.html` file in the `public` directory. (You can find it from download ZIP file from BlackBoard)



```html
<html>
<head>
  <title>Express</title>
  <link rel="stylesheet" href="/stylesheets/style.css">
</head>
<body>
  <h1>Express</h1>
  <p>Welcome to Express</p>
  <h1>Let's play Tic-Tac-Toe</h1>
  <div id="root"></div>
  <script src="babel/bundle.js"></script>
</body>
</html>
```

13

13

**7. Install necessary loaders**: You need to install `style-loader` and `css-loader` to handle CSS files. You can do this using `npm` or `yarn`:

```
Node1-React % npm install style-loader css-loader --save
```

**8. Install one more package**

```
Node1-React % npm install path --save
```

**9. Add files for Tic-Tac-Toe:** `client/App.js` and `client/styles.css` (You can find it from download ZIP file from BlackBoard)

**10. Build Your Project**: Run webpack to bundle your React files using Babel.

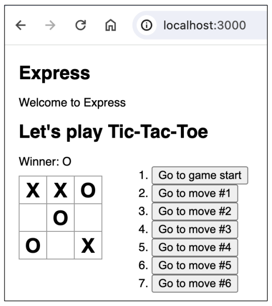
```
Node1-React % npx webpack
```

14

14

## React: Tic-Tac-Toe

```
Node1-React % npm start

> node1-react@0.0.0 start
> node ./bin/www
```



**Express**

Welcome to Express

**Let's play Tic-Tac-Toe**

Winner: O

| | | |
|---|---|---|
| X | X | O |
| | O | |
| O | | X |

1. Go to game start
2. Go to move #1
3. Go to move #2
4. Go to move #3
5. Go to move #4
6. Go to move #5
7. Go to move #6

15

15