

How to Create a Custom Webpack build

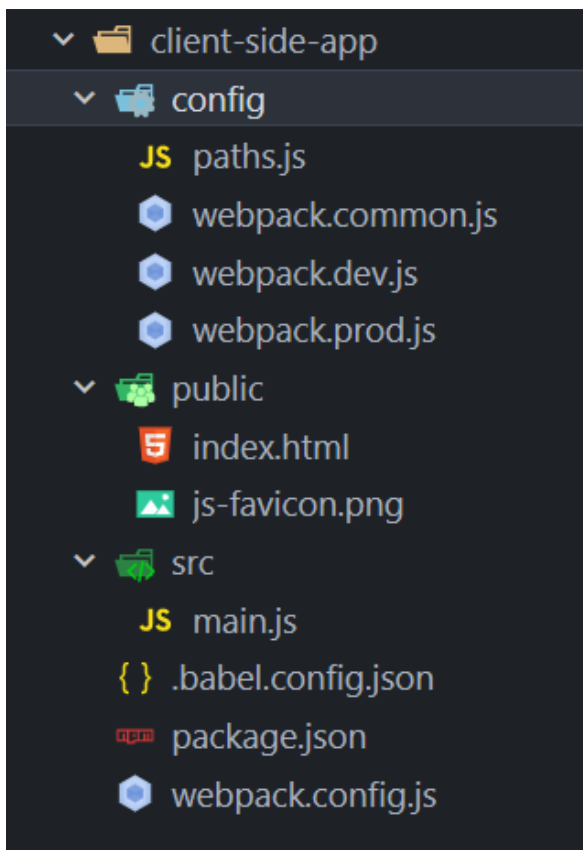
1. Make a new directory (client-side-app) and cd into it
2. Create package.json file using the following command from terminal window:

```
npm init -y
```

3. Open package.json file and copy paste the below contents in it:

```
{
  "name": "client-side-app",
  "version": "1.0.0",
  "description": "Client-Side Application Build",
  "private": true,
  "scripts": {
    "start": "webpack serve",
    "build": "webpack",
    "build-serve": "webpack && serve ./dist"
  },
  "author": "Manish Sharma",
  "license": "ISC"
}
```

4. Open the folder in Visual Studio Code and create the files and folder as per the below screen:



5. After the folders are created, we must do package installations using the following commands, using the terminal window

Production Dependencies

```
npm i core-js
```

Development Dependencies

npm i -D webpack webpack-cli webpack-dev-server webpack-merge babel-loader @babel/core @babel/preset-env html-loader file-loader clean-webpack-plugin html-webpack-plugin progress-bar-webpack-plugin terser-webpack-plugin chalk@4 serve

6. After the package installations are completed, open and copy paste the following code in each of the files, created earlier:

config/paths.js

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

const appDirectory = fs.realpathSync(process.cwd());

const resolvePath = function (relativePath) {
  return path.resolve(appDirectory, relativePath);
}

module.exports = {
  appRootPath: appDirectory,
  appBuildPath: resolvePath('dist'),
  outputJSPath: 'static/js/',
  outputCSSPath: 'static/css/'
};
```

config/webpack.common.js

```
const HtmlWebpackPlugin = require("html-webpack-plugin");
var ProgressBarPlugin = require('progress-bar-webpack-plugin');
const chalk = require('chalk');

module.exports = function (env) {
  return {
    entry: {
      app: './src/main'
    },
    resolve: {
      extensions: [".js"]
    },
    module: {
      rules: [
        {
          test: /\.js$/,
          exclude: /node_modules/,
          use: {
            loader: "babel-loader"
          }
        },
        {
          test: /\.html$/,
```

```

        use: [
          {
            loader: "html-loader"
          }
        ]
      },
    ],
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: "./public/index.html", // Input FileName
      filename: "./index.html", // Output FileName
      scriptLoading: 'blocking',
      favicon: "./public/js-favicon.png"
    }),
    new ProgressBarPlugin({
      format: '  build [:bar] ' + chalk.green.bold(':percent') +
'\t' + chalk.blue.bold(':elapsed seconds'),
      clear: false
    })
  ],
  optimization: {
    splitChunks: {
      chunks: 'all'
    }
  }
};
};

```

config/webpack.dev.js

```

const { merge } = require('webpack-merge');
const commonConfig = require('./webpack.common.js');

module.exports = function (env) {
  return merge(commonConfig(env), {
    mode: 'development',

    devtool: 'eval-cheap-source-map',

    output: {
      publicPath: 'http://localhost:3000/',
      filename: '[name].js',
      chunkFilename: '[id].chunk.js'
    },

    devServer: {
      port: 3000,
      historyApiFallback: true,
      client: {
        logging: "none"
      }
    }
  });
};

```

```

    },
    devMiddleware: {
      stats: 'minimal'
    },
    open: {
      app: {
        name: 'Chrome'
      }
    },
    hot: true
  }
});
}

```

config/webpack.prod.js

```

const { merge } = require('webpack-merge');
const { CleanWebpackPlugin } = require('clean-webpack-plugin');
const TerserPlugin = require('terser-webpack-plugin');
const commonConfig = require('./webpack.common.js');
const paths = require('./paths');

module.exports = function (env) {
  return merge(commonConfig(env), {
    mode: 'production',

    output: {
      path: paths.appBuildPath,
      publicPath: './',
      filename: `${paths.outputJSPath}[name].[hash].js`,
      chunkFilename: `${paths.outputJSPath}[id].[hash].chunk.js`
    },

    plugins: [
      new CleanWebpackPlugin()
    ],

    optimization: {
      minimize: true,
      minimizer: [new TerserPlugin()]
    }
  });
}

```

public/index.html

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.cs
s">
    <title>ECMAScript Demos</title>
</head>

<body class="container">
    <div class="text-center">
        <h1 class="text-primary">Please Check the Console for Output</h1>
    </div>
</body>

</html>

```

src/main.js

```
console.log("Hello from the Main file");
```

.babel.config.json

```

{
  "presets": [
    "@babel/env",
    {
      "targets": {
        "edge": "17",
        "firefox": "60",
        "chrome": "67",
        "safari": "11.1"
      },
      "core-js": "3.20.2",
      "useBuiltIns": "usage"
    }
  ],
  "plugins": []
}

```

webpack.config.js

```

module.exports = function (env) {
  // console.log("webpack.config.js is loaded - ", env);
  var env_file = env.WEBPACK_SERVE ? 'dev' : 'prod';
  return require(`./config/webpack.${env_file}.js`)(env);
}

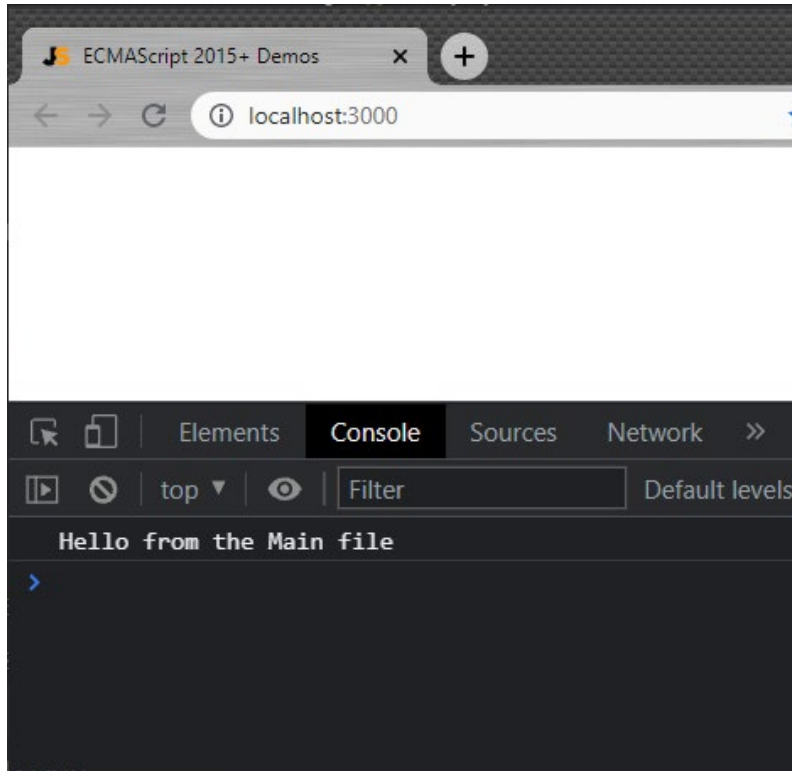
```

Let's run the application

After the files are modified and saved, use the following command from the terminal window to run the application.

```
npm start
```

Once you start the server, webpack-dev-server will start and chrome browser will open automatically on localhost:3000, open the browser console to verify the following output:



If you want to create a production build to create dist folder, use following command

Stop the server if it is running using Ctrl + c

```
npm run build
```

If you want to create a production build to create dist folder and load it on local http server, use following command

Stop the server if it is running using Ctrl + c

```
npm run build-serve
```