How to Create a Custom Webpack build

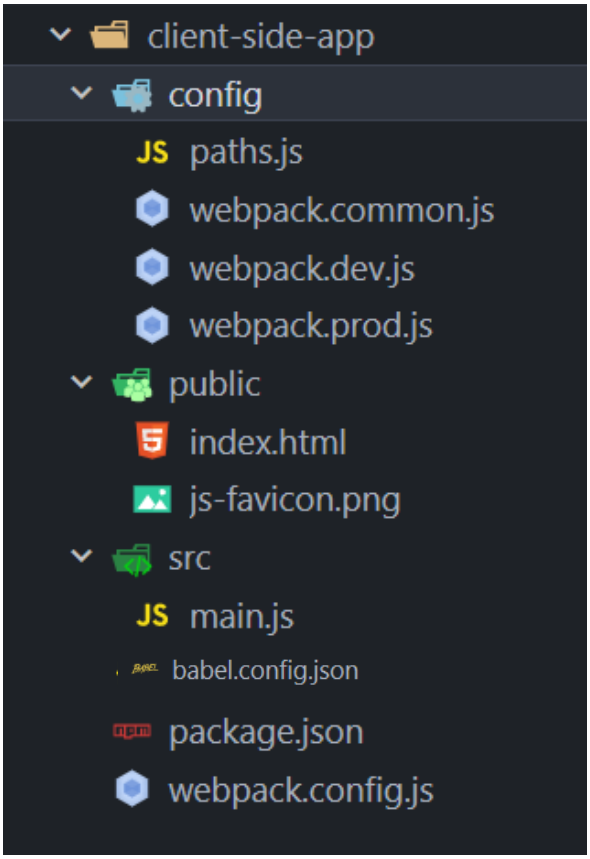
1. Make a new directory (client-side-app) and cd into it
2. Create package.json file using the following command from terminal window:

   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/'
};
```

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",
                overlay: {
```

```
                    errors: true,
                    warnings: false,
                }
            },
            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": "102",
                "firefox": "101",
                "chrome": "102",
                "safari": "11.1"
            },
            "core-js": "3.23.5",
            "useBuiltIns": "usage"
        }
    ]
}
```

webpack.config.js

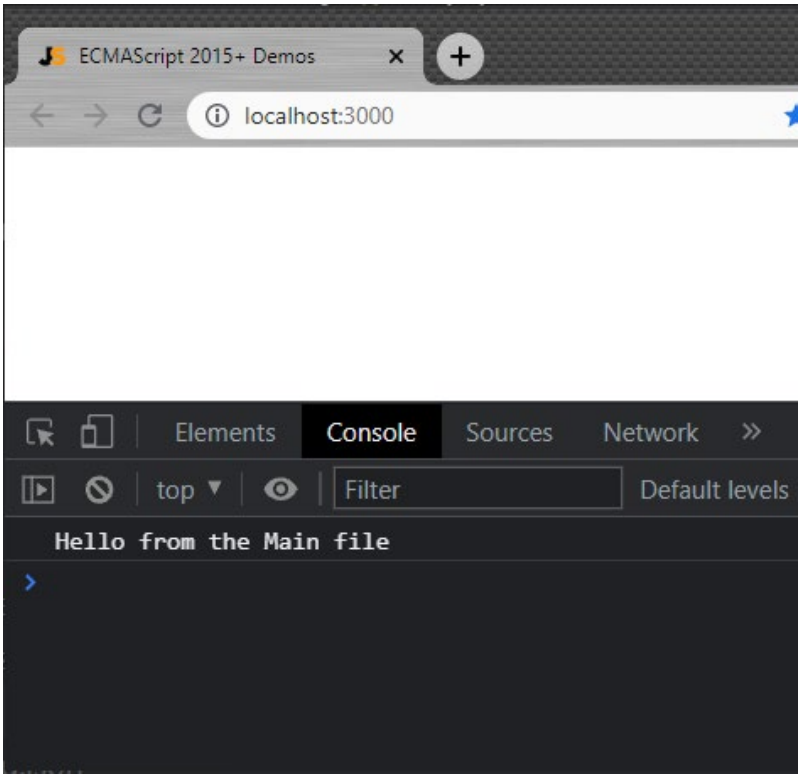
```
module.exports = function (env) {
    // console.log("env - ", env);
    var env_file = env.WEBPACK_SERVE ? 'dev' : 'prod';
    return require(`./config/webpack.${env_file}.js`)(env);
}
```

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

Use of each package installed

core-js	It is a polyfill of the JavaScript standard library, which supports: • The latest ECMAScript standard. • ECMAScript standard library proposals. • Some WHATWG / W3C standards (cross-platform or closely related ECMAScript).
@babel/core	The babel compiler
@babel/preset-env	@babel/preset-env is a smart preset that allows you to use the latest JavaScript without needing to micromanage which syntax transforms (and optionally, browser polyfills) are needed by your target environment(s).
babel-loader	This package allows transpiling JavaScript files using Babel and webpack.
chalk	Terminal string styling done right
clean-webpack-plugin	A webpack plugin to remove/clean your build folder(s).
file-loader	The file-loader resolves import/require() on a file into a url and emits the file into the output directory.
html-loader	Exports HTML as string. HTML is minimized when the compiler demands.
html-webpack-plugin	The HtmlWebpackPlugin simplifies creation of HTML files to serve your webpack bundles. This is especially useful for webpack bundles that include a hash in the filename which changes every compilation.
progress-bar-webpack-plugin	Used for displaying progress percentage of your webpack build
serve	serve helps you serve a static site, single page application or just a static file (no matter if on your device or on the local network). It also provides a neat interface for listing the directory's contents
terser-webpack-plugin	This plugin uses terser to minify/minimize your JavaScript.
webpack	Webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.
webpack-cli	webpack CLI provides a flexible set of commands for developers to increase speed when setting up a custom webpack project.
webpack-dev-server	Use webpack with a development server that provides live reloading. This should be used for development only.
webpack-merge	webpack-merge provides a merge function that concatenates arrays and merges objects creating a new object.