https://habr.com/ru/post/309958/

**yarn add @babel/core @babel/plugin-proposal-decorators @babel/polyfill @babel/preset-env @babel/preset-react react react-dom babel-loader && yarn add -D autoprefixer clean-webpack-plugin copy-webpack-plugin css-loader css-mqpacker cssnano html-webpack-plugin mini-css-extract-plugin node-sass path postcss-loader sass sass-loader style-loader webpack webpack-cli webpack-dev-server webpack-merge**

**babel-loader ??? куда в dev или в прод**

Эти зависимости надо устанавливать как зависимости проекта, так как серверной части приложения тоже нужен babel.

## Устаревшие плагины и пресеты

## Babel Legacy Decorator plugin - This plugin is specifically for Babel 6.x. If you're using Babel 7, this plugin is not for you. Babel 7's @babel/plugin-proposal-decorators officially supports the same logic that this plugin has, but integrates better with Babel 7's other plugins.

**@babel/preset-stage-0**

As of Babel v7, all the stage presets have been deprecated. Check [the blog post](https://babeljs.io/blog/2018/07/27/removing-babels-stage-presets) for more information.

For upgrade instructions, see [the README](https://github.com/babel/babel/blob/master/packages/babel-preset-stage-0/README.md).

**@babel/preset-es2015**

As of Babel v6, all the yearly presets have been deprecated. We recommend using [@babel/preset-env](https://babeljs.io/docs/en/babel-preset-env) instead.

**@babel/polyfill**

As of Babel 7.4.0, this package has been deprecated in favor of directly including core-js/stable (to polyfill ECMAScript features) and regenerator-runtime/runtime (needed to use transpiled generator functions):

Вместо **babel-plugin-transform-decorators-legacy** буду использовать **@babel/plugin-proposal-decorators** так как он поддерживает ES7

Изменим .babelrc в соответстивие с текущим версия плагинов:

{

    "presets": [

        "@babel/preset-env",

        "@babel/preset-react"

    ],

    "plugins": [

        "@babel/plugin-proposal-decorators"

    ]

}

Еще одна крутая особенность процесса сборки *webpack* заключается в том, что если вы используете огромную библиотеку, например *lodash*, но явно указываете, что вам нужна только определенная функция, например

import assign from 'lodash/assign';

то в итоговую сборку войдет лишь используемая часть библиотеки, а не вся она целиком, что может существенно уменьшить размер собранного файла.

**Примечание:** это будет работать, только если используемая библиотека поддерживает модульность. По этой причине автор отказался от использования в своих проектах библиотек Moment.js, XRegExp и ряда других.

*resolve* означает, что когда мы пишем в коде нашего приложения

import SomeClass from './SomeClass';

*webpack* будет искать *SomeClass* в файлах *SomeClass.js* или *SomeClass.jsx* прежде, чем сообщит, что не может найти указанный файл.

*The webpack repository contains an*[*example showing the effect of all devtool variants*](https://github.com/webpack/webpack/tree/master/examples/source-map)*. Those examples will likely help you to understand the differences.*

yarn add bluebird && yarn add -D babel-loader

yarn add -D babel-eslint eslint eslint-loader eslint-plugin-react

**ESLint**

ывмыв