**Настройка ESLint, Prettier, pre-commit hook   
(create-react-app, visual studio code)﻿**

https://maxpfrontend.ru/vebinary/nastroyka-eslint-prettier-pre-commit-hook-create-react-app-visual-studio-code/

1. Установка Node JS

2. Проверка версий:

node –v

npm –v

Проверить переменную PATH в переменных средах:

C:\Program Files\nodejs\

### Установка и настройка ESLint

https://itchief.ru/javascript/vscode-configuration-for-frontend-development

ESLint – это инструмент, который крайне желательно установить в систему, если вы разрабатываете код на JavaScript. Он будет показывать вам ошибки в коде, а также направлять вас при его написании так, чтобы он был выдержан в едином стиле.

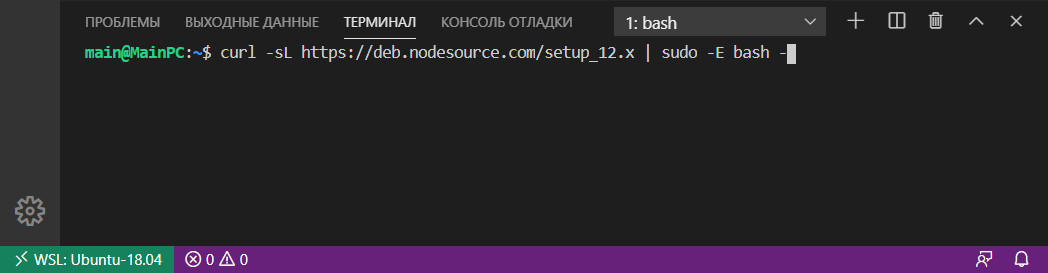
Перед тем как переходить к установке ESLint сначала инсталлируем в ОС «Node.js v12.x».

В Ubuntu это осуществляется следующим образом:

**curl -sL https://deb.nodesource.com/setup\_12.x | sudo -E bash -**

**sudo apt-get install -y nodejs**

Вводить эти команды будем через терминал VS Code. Открыть его можно посредством комбинации клавиш Ctrl+Shift+` или кликнув в главном меню на пункт «Терминал -> Создать терминал».



Проверить номер установленной версии «Node.js» можно так:

**node -v**

После установки «Node.js» создадим папку для проекта в файловой системе ОС, а затем откроем её с помощью VS Code.

Создание проекта обычно начинается с его инициализации посредством npm. Этот процесс можно выполнить посредством следующей команды:

**npm init -y**

В результате выполнения этой команды у вас появится файл «package.json». Этот файл кроме информации о проекте и других вещей, ещё будет содержать набор зависимостей для данного проекта. Имея этот файл, мы сможем при необходимости очень быстро развернуть проект на любом другом компьютере.

Теперь перейдём к установке ESLint и некоторых других npm пакетов в проект:

**npm install --save-dev eslint eslint-config-airbnb-base eslint-plugin-import**

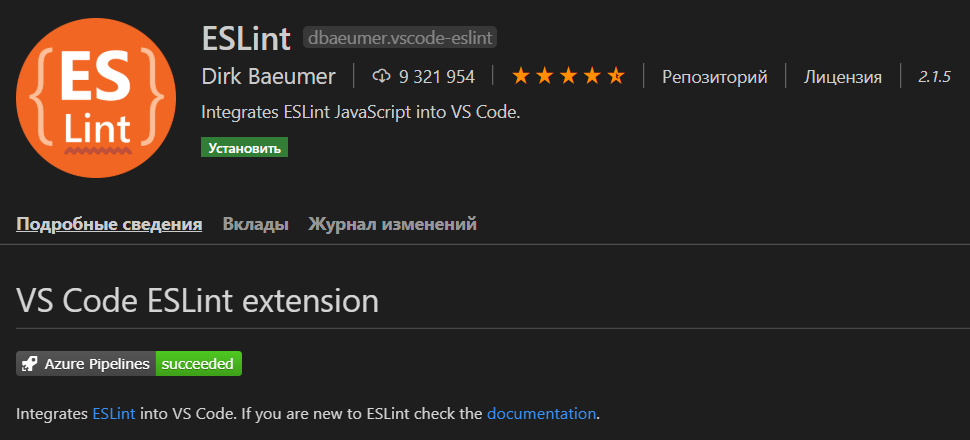
Ключ **--save-dev** используется для того чтобы сделать запись об этих пакетах в «package.json». Данный ключ добавит их в секцию **devDependencies**.

Установка npm пакетов осуществляется в папку «node\_modules» этого проекта.

В качестве стиля кода мы будем использовать Airbnb. Это руководство используется многими известными организациями и имеет очень большое количество звёзд на GitHub.

Для того, чтобы можно было использовать Airbnb для расширения базовой конфигурации ESLint мы установили пакеты **eslint-config-airbnb-base** (без зависимостей от React) и **eslint-plugin-import** (для поддержки синтаксиса импорта/экспорта ES6+ и предотвращения проблем с неправильным написанием путей к файлам и имен импорта).

После окончания загрузки пакетов приступим к интегрированию ESLint в Visual Studio Code. Осуществляется это посредством установки расширения с одноимённым названием.



Для того чтобы ESLint работал необходимо создать конфигурационный файл. Это можно выполнить как посредством команды **./node\_modules/.bin/eslint --init** (настройка осуществляется посредством ответов на вопросы мастера), так и самостоятельно.

Конфигурационный файл необходим для задания настроек, в соответствии с которыми ESLint будет осуществлять проверку JavaScript кода.

Чтобы сделать это самостоятельно нам необходимо в корне проекта создать файл **.eslintrc** и добавить в него, например, следующие данные:

**{**

**"env": { "browser": true, "es6": true },**

**"extends": "airbnb-base",**

**"parserOptions": { "ecmaVersion": 6 }**

**}**

Эти данные будут определять следующие настройки для ESLint:

* **env** - это свойство, которое определяет среды, в которых JavaScript должен работать. Для фронтенда средой выступает браузер, поэтому добавим в **env** свойство **"browser": true**. Свойство **"es6": true** предназначено для автоматического включения синтаксиса ES6.
* **extends** - предназначен для указания конфигурации, с помощью которой мы хотим расширить общую конфигурацию ESLint. В качестве значения **extends** укажем конфигурацию **airbnb-base** (без React). При необходимости можно указать не одну, а несколько конфигурации. В этом случае каждая следующая конфигурация будет расширять предыдущую. Т.е. если мы укажем какую-то конфигурацию после **airbnb-base**, то она будет уже расширять не общую конфигурацию ESLint, а **airbnb-base**;
* **parserOptions** - позволяет задать параметры языку JavaScript, которые мы хотим поддерживать. В настоящее время рекомендуется использовать при разработке проектов синтаксис ECMAScript 6. Указание поддержки этой версии осуществляется посредством задания ключу **ecmaVersion** значения 6. При необходимости вы можете указать вместо этой другую версию.

Если вам необходимо дополнительно линтить ошибки кода, размещенного в теге <script>, то установите плагин **eslint-plugin-html**:

**npm install --save-dev eslint-plugin-html**

Проверяет пути к модулям и используются ли все прикрепленные module)

**npm install --save-dev eslint-plugin-import**

**npm install --save-dev eslint-plugin-jsx-ally**

**Prettier**  
Форматирование кода JavaScript будем выполнять с помощью **Prettier**. Для правильной совместной работы Prettier и ESLint установим следующие npm пакеты:

**npm install --save-dev prettier eslint-plugin-prettier eslint-config-prettier**

Для того чтобы ESLint не просматривал определённые папки и не выдавал по ним ошибки следует создать файл **.eslintignore** и прописать их в нём:

**/.vscode**

**/.git**

**node\_modules**

Если у вас включено стандартное форматирование кода в VS Code при сохранении, то чтобы в качестве плагина для форматирования js файлов применялся ESLint, следует в конфигурационный файл «settings.json» добавить следующее:

**"editor.formatOnSave": true,**

**// добавить это**

**"[javascript]": {**

**"editor.defaultFormatter": "dbaeumer.vscode-eslint"**

**},**

**"eslint.format.enable": true**

Если вы хотите чтобы при сохранении файлов форматировались только js файлы, то **editor.formatOnSave** необходимо установить значение **false**, а **true** этому ключу только в секции **"[javascript]"**:

**"editor.formatOnSave": false,**

**"[javascript]": {**

**"editor.formatOnSave": true,**

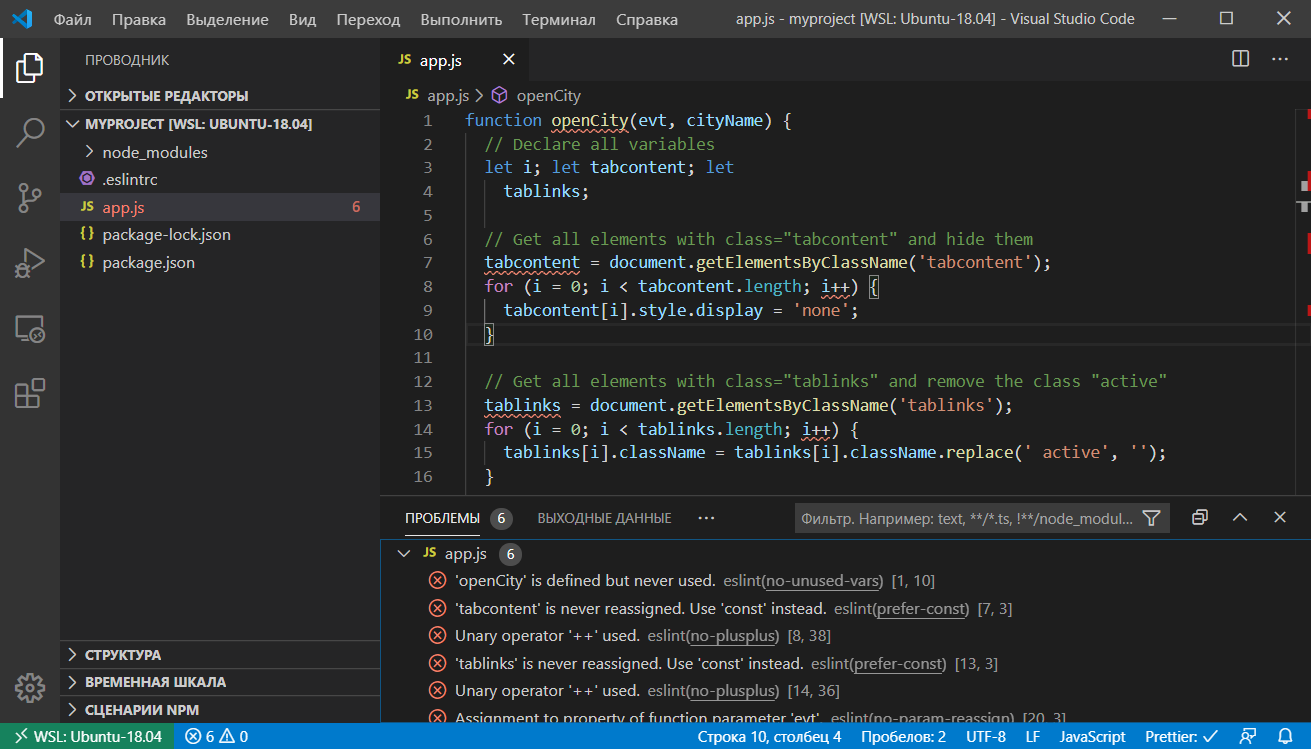
**"editor.defaultFormatter": "dbaeumer.vscode-eslint"**

**},**

**"eslint.format.enable": true**

Теперь, ESlint будет проверять JavaScript код и показывать в нём ошибки и предупреждения. Они будут помечаться с использованием волнистых линий.

Результат проверки JavaScript кода ESLint:



**Prettier -** <https://maxpfrontend.ru/vebinary/nastroyka-eslint-prettier-pre-commit-hook-create-react-app-visual-studio-code/>

Могу ли я настраивать Prettier?

Да. Настроек в парсере Prettier не так много, как в ESLint. С Prettier вы полностью во власти парсера Prettier. Основываясь на небольшом количестве опций, он сам решает, как будет выглядеть ваш код.

Это мои настройки, которые описаны в файле***.prettierrc* .** Полный список опций вы можете найти по [ссылке](https://prettier.io/docs/en/options.html). Давайте создадим ***.prettierrc***-файл с такими опциями.

JSON

{

"useTabs": false, // использовать табы? нет (я за пробелы)

"printWidth": 80, // длина строки - 80

"tabWidth": 2, // длина "таба" - 2 пробела

"singleQuote": true, // использовать одинарные кавычки - да!

"trailingComma": "es5", // запятая в последней строке - да

"jsxBracketSameLine": false, // закрывающийся jsx в этой же строке

"parser": "flow", // парсер - flow (пока не важно)

"semi": false // точка с запятой - нет

}

**Rules for eslint**

Enforce specifying rules to disable in *eslint-disable comments*

This rule makes you specify the rules you want to disable when using *eslint-disable, eslint-disable-line* or*eslint-disable-next-line comments.*

**Принудительное указание правил для отключения в eslint-disable comments**

**Это правило позволяет указать правила, которые вы хотите отключить при использовании комментариев eslint-disable, eslint-disable-line или eslint-disable-next-line.**

const message = 'foo';

console.log(message); // eslint-disable-line no-console

// eslint-disable-next-line no-console

console.log(message);

const message = 'foo';

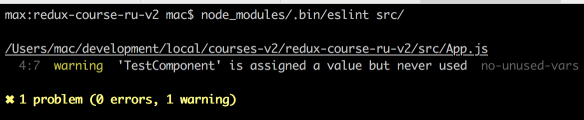
console.log(message); // eslint-disable-line no-console

// eslint-disable-next-line no-console

console.log(message);

**Выполните команду в терминале (находясь в директории с проектом):**

**node\_modules/.bin/eslint js (или js/test.js)**



**Быстрая настройка для запуска eslint:**

**Ошибка: eslint не распознается как внутренняя или внешняя команда» при попытке запустить eslint**

Если конфигурации скопированы и даже папка **node\_modules** и вылазит ошибка как высшее, попробуй**:**

eslintМодуль должен **не** быть установлен в глобальный.  
вместо этого вы должны установить eslint-cliмодуль в глобальный.

Итак, сначала установите **eslint-** cli:

npm -g i eslint-cli

затем в папке проекта: установить **eslint** локально

npm i eslint --save-dev

Затем в папке проекта вы можете запустить что-то вроде: (в Windows)

eslint .\

или в нашем случае есть папка js, в ней test.js (js/test.js)

eslint js // все файлы в этой папке

eslint js/

[**https://tproger.ru/translations/setting-up-eslint-and-prettier/**](https://tproger.ru/translations/setting-up-eslint-and-prettier/)

## Создайте test.js

Давайте создадим простой JavaScript-файл в папке lint-example, куда мы установили ESLint. Не переживайте о плохом форматировании примера. Нам нужно это для старта.

var foo = 1

console.log(foo)

var bar

bar = 1

function test(  
  
 ) {

console.log(baz)

}

var baz = 123

**Решение проблемы с политиками безопасности**

./node\_modules/.bin/eslint : Невозможно загрузить файл   
D:\All\_for\_site\\_JavaScript-Learn\itgid.info\JavaScript\_2.0\node\_modules\.bin\eslint.ps1, так как выполнение сценариев отключено в этой системе. Для получения дополнительных сведений см. about\_Execution\_Policies по адресу https:/go.microsoft.com/fwlink/?LinkID=135170.

строка:1 знак:1

+ ./node\_modules/.bin/eslint --init

+ ~~~~~~~~~~~~~~~~~~~~~~~~~~

+ CategoryInfo : Ошибка безопасности: (:) [], PSSecurityException

+ FullyQualifiedErrorId : UnauthorizedAccess

**Проверить политики безопасности можно в PowerShell командой:**

Get-ExecutionPolicy -List

Для этого нужно зайти в реестр установить ключ в параметре (**ExecutionPolicy) путь 🡪**

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell   
 **ExecutionPolicy : RemoteSigned**

Если ключа нет, то надо создать его.

<https://winitpro.ru/index.php/2020/06/03/powershell-execution-policy-zapusk-scriptov/>

<https://windowsnotes.ru/powershell-2/politika-vypolneniya-skriptov-powershell/>

еще

<https://www.securitylab.ru/analytics/461333.php>

<https://winitpro.ru/index.php/2019/05/13/proverit-prava-administratora-v-skripte-powershell/>