**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа №5**

**Разработка плагина**

Выполнила студент группы №М32111

Чу Тхи Фыонг Тхао

Проверил

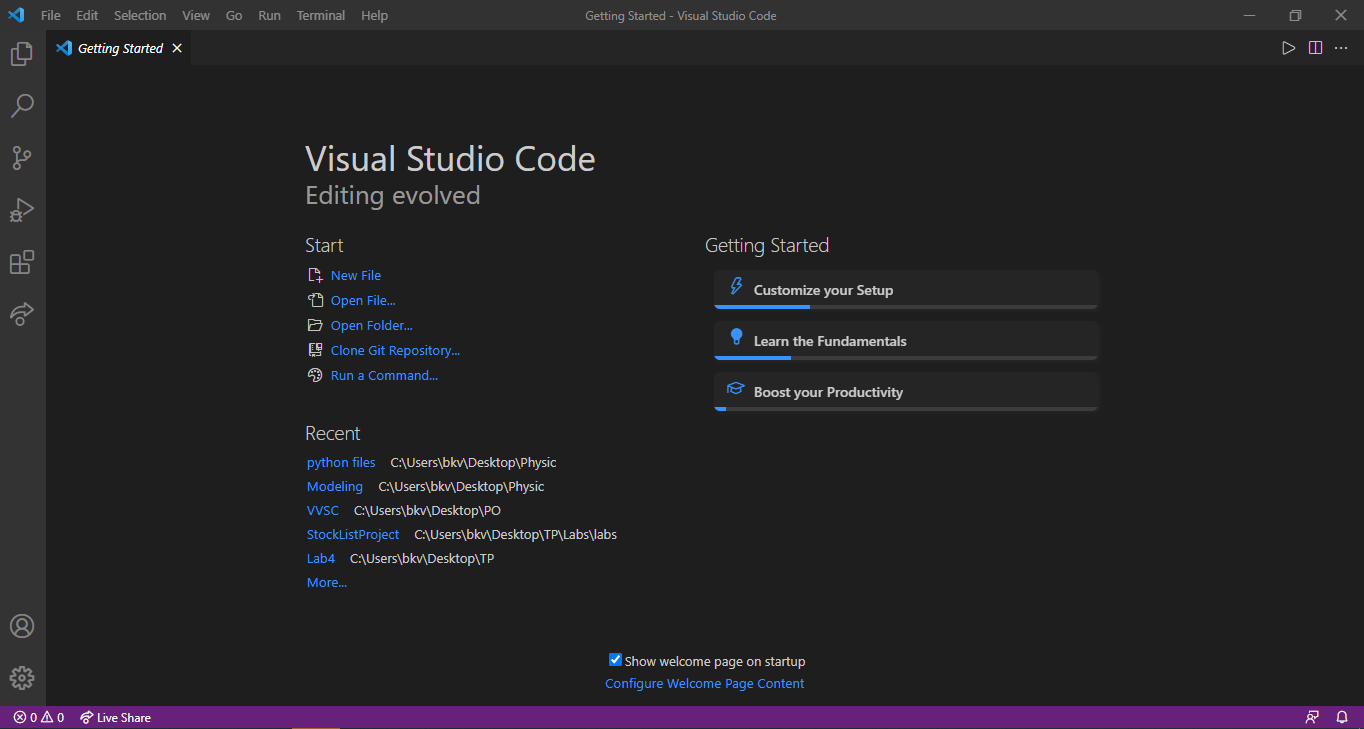
Дядюшкин Александр Александрович

САНКТ-ПЕТЕРБУРГ

2021

**Интегрированная среда разработки (IDE):** Visual Studio Code

**Visual Studio Code** — это легкий, но мощный редактор исходного кода, обладающий функциями IDE, который работает на вашем рабочем столе и доступен для Windows, macOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C ++, C #, Java, Python, PHP, Go) и сред выполнения (таких как .NET и Unity).



**Файлы, папки, и проекты**

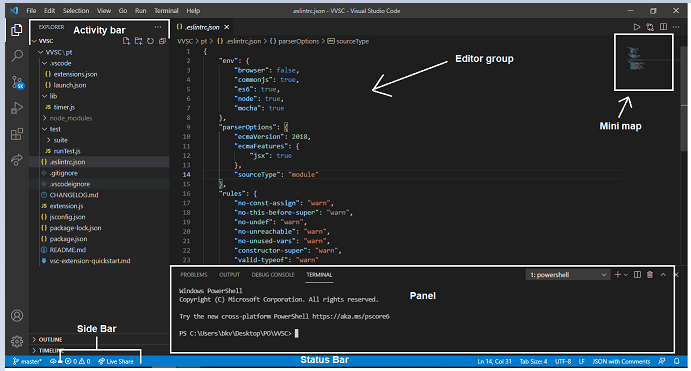
В простейшем варианте разработки вы можете открыть файл на редактирование, щелкнув по файлу и выбрав из контекстного меню «Open with Code» или просто выполнив команду code filename.ext. При этом редактор автоматически (определив расширение файла) включит множество полезных функций, обеспечивающих IntelliSence, подсказки, навигацию по коду, выполнение команд и др. В более продвинутом варианте вы можете открывать папки (проекты), щелкнув по папке и выбрав из контекстного меню «Open with Code» или просто выполнив команду code dirname. VSCode также определит тип проекта и подключит множество полезных функций.

**Расположение основных элементов:**

Интерфейс разделен на пять основных блоков: Editor, Side Bar, Status Bar, Activity Bar, Panels.

* **Editor** - основная область для редактирования ваших файлов. Вы можете открыть столько редакторов, сколько захотите, вертикально и горизонтально
* **Side Bar** - содержит различные виды, такие как Проводник, чтобы помочь вам при работе над проектом.
* **Status Bar** - информация об открытом проекте и файлах, которые вы редактируете.
* **Activity Bar** - расположена в крайнем левом углу, позволяет переключаться между представлениями и дает дополнительные индикаторы, зависящие от контекста, например количество исходящих изменений при включении Git.
* **Panels** - Вы можете отображать различные панели под областью редактора для вывода или отладки информации, ошибок и предупреждений или встроенного терминала. Панель также может быть перемещена вправо для большего вертикального пространства.

Каждый раз, когда вы запускаете VS Code, он открывается в том же состоянии, в каком он был в последний раз. Папка, макет и открытые файлы сохраняются

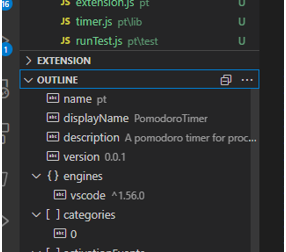


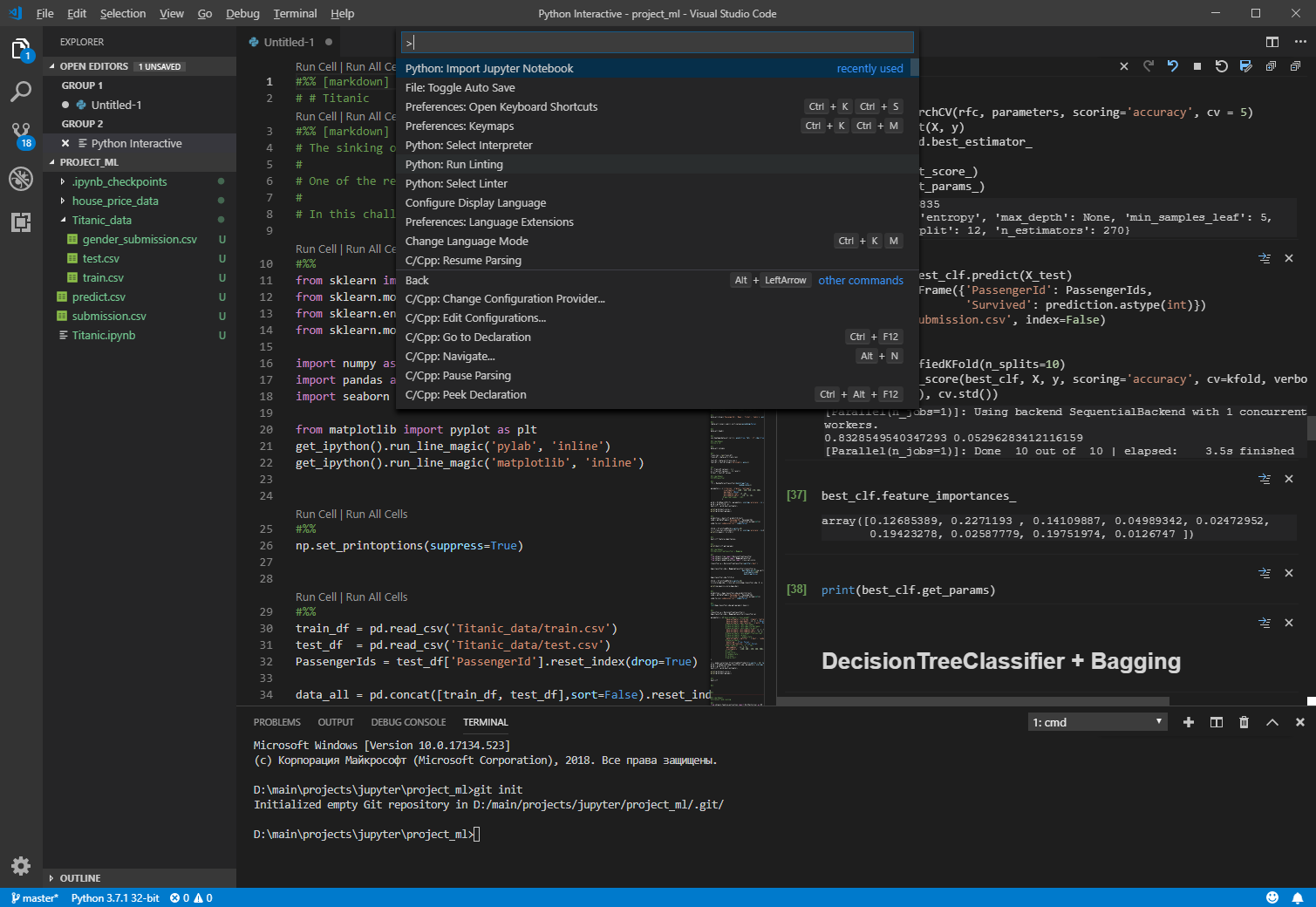
**Мини-карта (схема кода)** дает вам общий обзор исходного кода, который очень полезен для быстрой навигации и понимания кода. Мини-карта файла показана в правой части редактора. Вы можете нажать или перетащить затененную область, чтобы быстро перейти к различным разделам вашего файла. Вы можете переместить мини-карту на левую сторону или отключить ее полностью, соответственно , установив "editor.minimap.side": "left"или "editor.minimap.enabled": false в ваших настройках рабочей области.

Если у вас уже открыт один редактор, существует несколько способов открыть другой редактор рядом с существующим:

* Alt + клик по файлу в проводнике
* Ctrl + \ чтобы разделить редактор на две части
* Ctrl + Enter из контекстного меню проводника файла
* Ctrl + Enter в списке файлов Quick Open
* Нажмите Split Editor в правом верхнем углу редактора
* Перетащите файл в любую область редактора

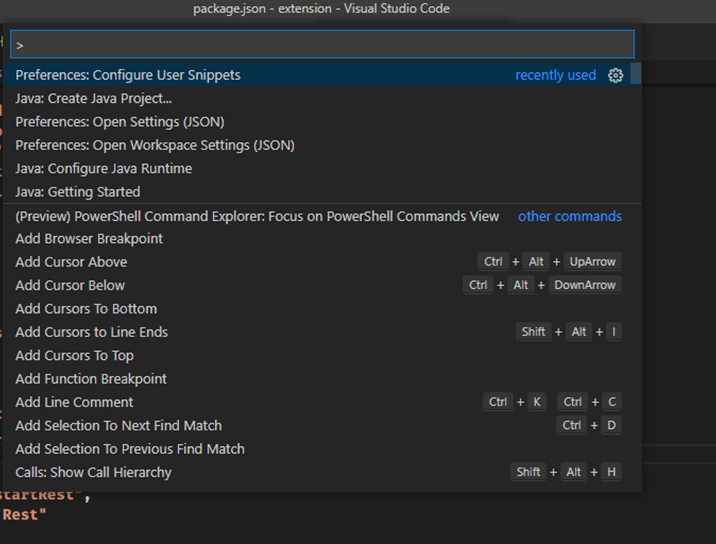
**Outline —** это отдельный раздел внизу Проводника. Когда он развернут, он показывает дерево символов текущего активного редактора. Outline имеет различные режимы сортировки, дополнительное отслеживание курсора и поддерживает обычные открытые вкладки. Он также включает в себя поле ввода, которое находит или фильтрует символы при вводе. Ошибки и предупреждения также отображаются на панели Outline, что позволяет сразу



**Проводник** — это только одно из представлений, доступных в VS Code. Есть также представления для:

* **Search** - обеспечивает глобальный поиск и замену в открытой папке.
* **Source** **Control** - VS Code по умолчанию включает в себя git
* **Debug** - в Debug View VS Code отображаются переменные, стеки вызовов и точки останова.
* **Extensions** - Установка и управление своими расширениями в VS Code.
* **Custom views** - представления, представленные расширениями в представлениях Explorer и Debug

**Палитра команд** - самый главный инструмент взаимодействия с редактором в VS Code. Вызвать ее можно через клавиатуру, нажав на комбинацию Ctrl+Shift+P. Множество команд перечисленных в палитре тоже привязаны к клавишам



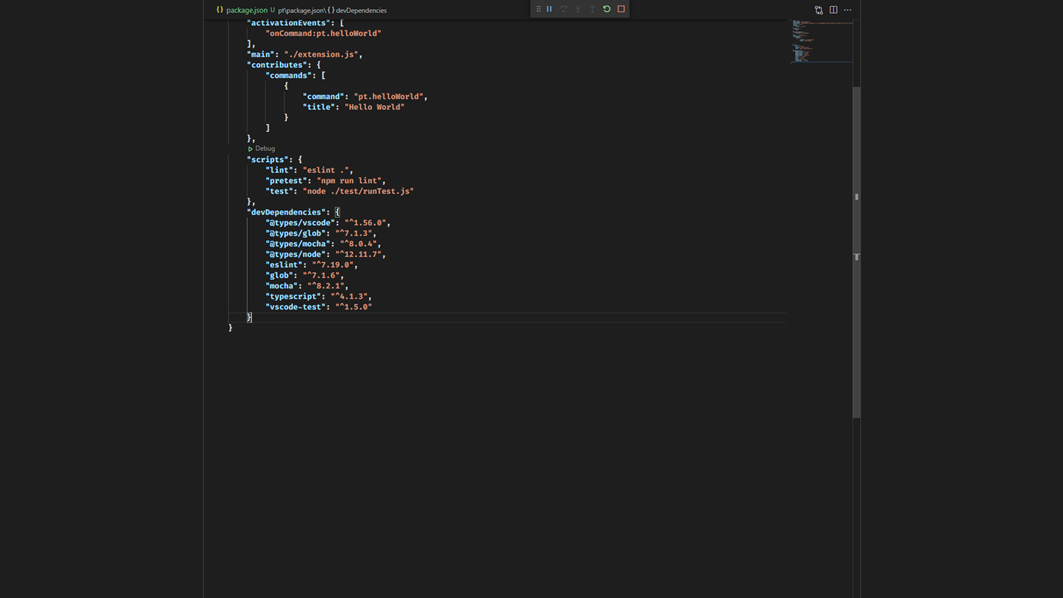
**Обозреватель**

Обозреватель предназначен для открытия и управления файлами находящимися в папке вашего проекта. Вы можете осуществлять множество привычных операций по созданию, изменению имени, удалению файлов и папок, и их перемещению.

Файлы, которые открыты в редакторе, помещаются в верхнюю часть обозревателя и получают статус Open Editors. Если открыто несколько групп редакторов, то еще и указана принадлежность файлов к группам.

|  |  |
| --- | --- |
|  |  |

**Zen Mode:** Режим Zen позволяет вам сосредоточиться на своем коде, скрывая весь пользовательский интерфейс, кроме редактора (без панели активности, строки состояния, боковой панели и панели), переходя на полный экран и центрируя макет редактора. Режим Zen можно переключать с помощью меню « Вид» , палитры команд или сочетания клавиш Ctrl + K Z . Двойной Esc выходит из режима дзен. Переход в полноэкранный режим можно отключить с помощью zenMode.fullScreen. Режим Zen может быть дополнительно настроен следующими параметрами: zenMode.hideStatusBar, zenMode.hideTabs, zenMode.fullScreen, zenMode.restoreи zenMode.centerLayout.

****

**Коротко о архитектуре и плагинах:**

**Плагины** в VS Code изолированы от самого редактора и запускаются в отдельном хост-процессе (**extension host process**), который представляет собой процесс Node.js с возможностью использования VS Code API. Такой подход не позволяет плагинам влиять на производительность редактора при его запуске или в процессе его работы. Для пользователя это означает, что редактор не зависнет на время выполнения задач каким-либо плагином или, если плагин выдаст фатальную ошибку. Для экономии расхода памяти разработчики также добавили **ленивую загрузку** плагинов. Это означает, что плагины активируются лишь в тот момент, когда они нужны. Например, если пользователь открывает Markdown-файл, то плагины, работающие с Markdown, будут загружены только в момент открытия файла. Разумеется, мы должны сообщить редактору о том, когда именно он должен активировать какой-либо плагин.

Обычно **плагины** vscode делят на три вида в зависимости от функционала:

* стандартные плагины, которые запускаются в хост-процессе, активируются в нужный момент и не требуют серьёзных вычислений.
* «**языковые серверы**». Клиентская часть плагина запускается в хост-процессе, а серверная часть создаёт дополнительный процесс, в котором производятся все сложные вычисления. К такому виду плагинов относятся линтеры.
* «**службы отладки**», которые пишутся в виде отдельной программы и взаимодействуют с VS Code по специальному протоколу CDP (VS Code Debug Protocol).

Написание файла начинается, внезапно, не с кода, а с файла манифеста которым в мире Node.js является файл package.json. Расмотрим основные поля в нем:

* **publisher [string]:** Имя пользователя
* **icon [string]:** Путь до иконки
* **displayName [string]:** Название плагина
* **categories [array]:** Имена категорий плагина   
  [Languages, Snippets, Linters, Themes, Debuggers, Other]
* **galleryBanner [object]:** Оформление
* **preview [boolean]:** Предварительного просмотра
* **activationEvents [array]:** события активации (\*(load redactor) - не очень):
  + onLanguage — открыт файл указанного языка (не расширения).
  + onCommand — вызвана указанная команда.
  + onDebug — запущен сеанс отладки указанного типа.
  + workspaceContains — найден указанный файл в корневой папке проекта.
* **contributes [object]:** описание расширения:   
  (configuration, commands, keybindings, languages, debuggers, grammars, themes, Snippets, jsonValidation)
* **extensionDependencies [array]:** зависимости расширений

**VS Code API**

VS Code имеет довольно обширный API, доступный разработчикам плагинов. Рассмотрим лишь основные пространства имён:

• **commands** — это пространство имён для работы с командами. С помощью доступных методов разработчик может регистрировать, получать и выполнять команды.

• **env** — пространство имён, содержащее описание переменных окружения редактора при запуске. С помощью соответствующих методов можно получить имя окна редактора, его язык, идентификатор редактора в системе и идентификатора сессии редактора, которые устанавливается при запуске.

• **extensions** — пространство имён для работы с установленными расширениями. С помощью этого API можно получить все или конкретные расширения, известные редактору.

• **languages** — пространство имён, позволяющее получить доступ к языковым возможностям редактора, например, к IntelliSense, подсказкам, а также функциям диагностики кода для линтеров.

• **window** — пространство имён для работы с текущим окном редактора. Доступно API для работы с видимыми и активными окнами редактора, а также элементами пользовательского интерфейса. Последнее подразумевает под собой возможность отображения различных сообщений, ввода текста или выбора каких-либо вариантов.

• **workspace** — пространство имён для работы с текущей рабочей областью, включая открытую директорию и файлы. С помощью этого API осуществляется вся работа с содержимым открытого файла.

**Создание плагина**

**Описание плагина:**

В качестве плагина я решила разработать "Таймер Pomodoro" - инструмент, который обеспечит простой инструмент/процесс для повышения продуктивности (вашей собственной и членов вашей команды), который может сделать следующее: Уменьшить беспокойство, связанное с началом работы. Усилить фокус и концентрацию, сократив количество прерываний. Повысить осознанность своих решений.

**Процесс создания:**

Генерация пустого плагина

Чтобы не писать кучу файлов необходимых для запуска базового (пустого) плагина, воспользуемя генератором плагинов:

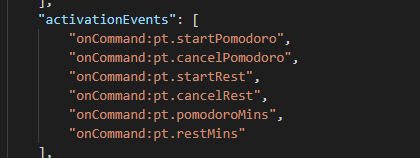
npm install -g yo generator-code

yo code

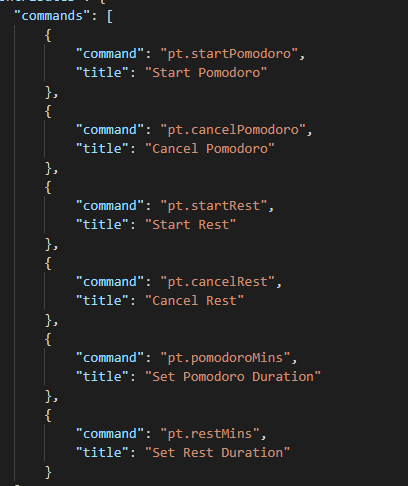
**Работа с командами в package.json и extension.js**

Добавим в **package.json** события активации и описание проекта со всеми командами, хоткеями и параметрами.

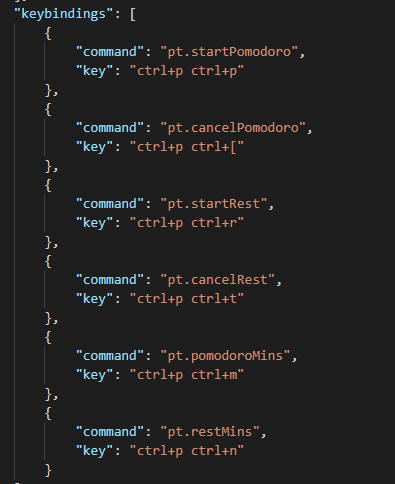
Нам понадобятся три события активации: команды старт, стоп и пауза

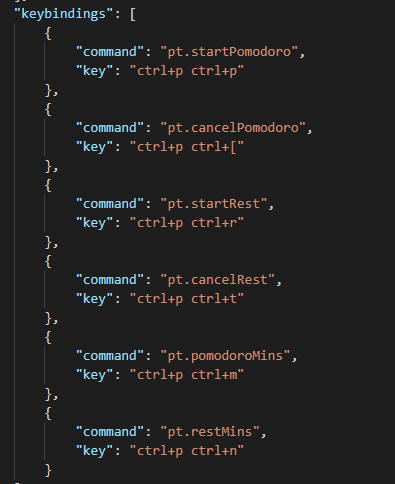


Создадим их в **commands:** (код идентичен для всех команд, поэтому здесь и далее буду приводить пример кода лишь для одной команды):



Назначим каждой хоткей:



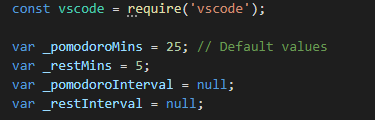


Теперь свяжем эти команды с функциями в **extension.js:**



**Получение глобальной конфигурации и установка значений:**

Чтобы использовать определенные пользователем значения наших переменных, нужно получить нашу конфигурацию:

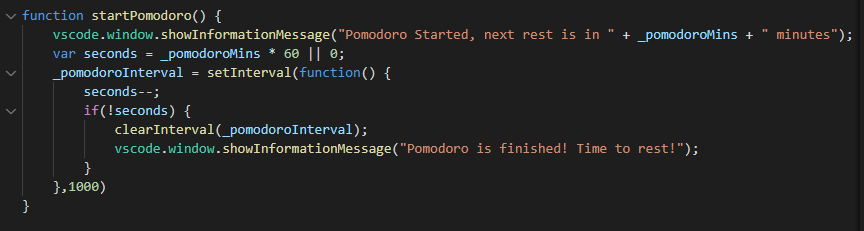


Теперь мы можем использовать их в коде

Для передачи их в экземпляры классов, нам нужно будет создать соответствующие конструкторы

**Базовый таймер:**

В расширении у нас есть функция для запуска Pomodoro, а также для отмены pomodoro, у нас есть функции для установки Rest и отмены Rest. Установка минут помодоро и установка минут отдыха. Я также предоставила значение по умолчанию для этого. вызывая различные методы в разное время, мы можем использовать его функциональность



**Результат**

В палитре команд доступны три новые функции, отвечающие за старт, паузу и остановку времени.

