# JavaScript

В предыдущих разделах мы часто использовали C#, чтобы продемонстрировать интеграцию с Unity, отладку и некоторые возможности интерфейса. Но, множество разработчиков используют Visual Studio Code для построения web приложений. Ну, а если мы говорим о web, то тут следует упомянуть и JavaScript и сопутствующие технологии, такие как node.js, TypeScript и другие. В этом разделе мы посмотрим какие интерфейсные возможности предоставляет Code для JavaScript, а в последующих разделах перейдем к Typescript и Node.js.

Поскольку JavaScript является одним из основных языков для Visual Studio Code, то для того, чтобы иметь возможность использовать IntelliSense систему и возможности редактора Вам даже нет необходимости создавать какие-то файлы проектов. Так Вы можете создать два простых файла с несколькими функциями внутри и Code автоматически вычитает всю информацию из этих файлов и добавит все поля и функции в IntelliSense.

Но, учитывая различные спецификации JavaScript, необходимость хранить несколько проектов в одном рабочем каталоге, требования исключать некоторые рабочие JavaScript файлы из проекта – лучше начинать любой JavaScript проект с проекта. Visual Studio Code работает с такими файлами проектов как **jsconfig.json**. В поиске этих файлов Code просматривает основной каталог и все вложенные каталоги. Все каталоги с **jsconfig.json** внутри считаются как отдельные проекты, куда включаются все файлы из основного каталога из всех внутренних каталогов (если в них нет своего jsconfig.json). Таким образом, если Вы хотите исключить целый каталог из проекта, то просто разместите там свой jsconfig.json. Соответственно, на основе наличия jsconfig.json будет работать и intelliSense система.

Чтобы создать простой jsconfig.json достаточно ввести следующий код:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs",

}

}

Тут присутствует элемент **compilerOptions**, который позволяет задать используемую спецификацию ECMAScript и платформу для работы с модульной архитектурой в JavaScript (commonjs или amd).

Дополнительно, Вы можете включить массив файлов, которые взодят в Ваш проект, если Вы не хотите, чтобы все файлы были включены автоматически:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs"

},

"files": [

"index.js"

]

}

Чтобы лучше понять, как работает jsconfig.json, просто создайте два файла JavaScript и в каждый поместите по одной функции:

function f(message){

alert(message);

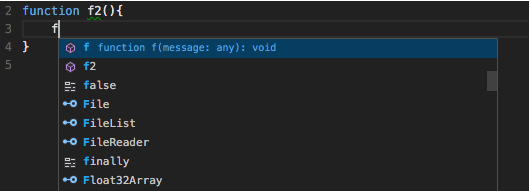
}

function f2(){

f("Hello");

}

Обратите внимание, что даже без проектного файла у Вас работает IntelliSense система:



Теперь добавим jsconfig.json:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs"

},

"files": [

"file2.js"

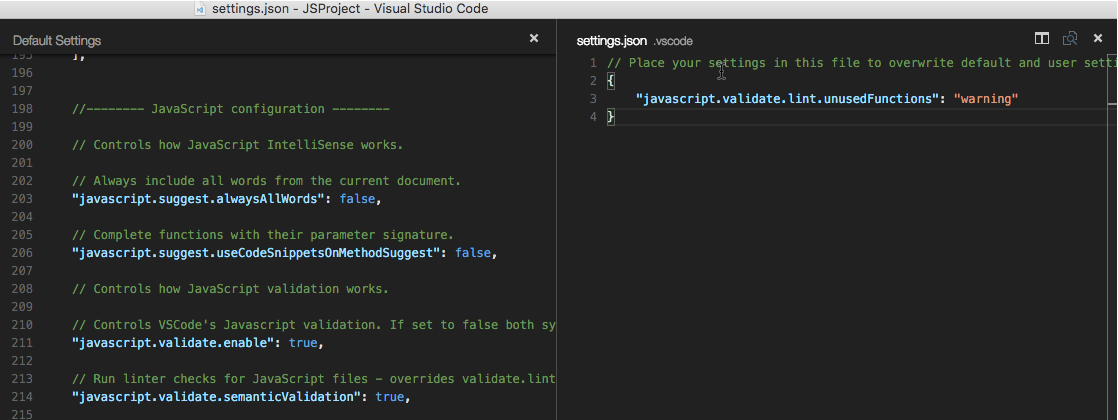
]

}

Тут я подразумеваю, что функция f находится в файле file1.js, а f2 в файле file2.js.

Как Вы можете видеть, прототип функции f был удален из IntelliSense системы.

Также, Visual Studio Code поддерживает ряд настроек, часть из которых выключена по умолчанию. Например, Code может отображать предупреждение, если у Вас в коде есть неиспользуемые функции, но эта настройка выключена. Чтобы ее включить используйте Workspace Settings или User Settings для задания настроек редактора в рамках рабочего каталога или всех проектов и найдите раздел, связанный с JavaScript:



Переопределите настройку для неиспользуемых функций и Visual Studio Code тут же отобразит два предупреждения в строке состояния:



Щелкнув на это сообщение Вы можете увидеть список предупреждений и перейти в код к любой из неиспользуемых функций:

