# JavaScript

In the previous articles we used C# in order to show integration with Unity, debugging and some interface features. But lots of developers like to use Visual Studio Code to build web applications and, of course, if we are talking about web applications – we are talking about JavaScript and related technologies (node.js, typescript and so on). So, it’s time to make a short introduction to how to use JavaScript and Code together.

Because JavaScript is one of the primary languages for Code you even don’t have to create any project files to start use all editor features. Clicking any JavaScript file, you can use IntelliSense, snippets, compiling features, syntax coloring, formatting and some editor features like peek and go to definition. All these features are working in the box.

But it’s better to start working with a project file. As we mentioned in the previous articles, Visual Studio Code recognizes **jsconfig.json** project files for JavaScript. Visual Studio Code parses the root folder of the project and all subfolders there and is looking for **jsconfig.json**. All folders with **jsconfig.json** are counted as projects’ root folders for all files there including subfolders. Therefore, you can have several projects in your folder or you can even use **jsconfig.json** to exclude some JavaScript files – just place them to the subfolder with not needed files. It’s very important if you want to see a proper work of IntelliSense.

To create a simple jsconfig.json you can use the following code:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs",

}

}

There is an element **compilerOptions** that allows to specify what specification you are going to use (ES6 – EcmaScript 6 is supported as well) and allows to select module framework between commonjs and amd.

Additionally, you can include **files** array to specify list of files in your project:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs"

},

"files": [

"index.js"

]

}

But if you don’t use **files**, VS Code will include all files in the root folder and in all subfolders (if subfolders don’t have own **jsconfig.json**).

To better understand how to work with **jsconfig.json** I would recommend to create couple JavaScript files and place there two simple functions:

function f(message){

alert(message);

}

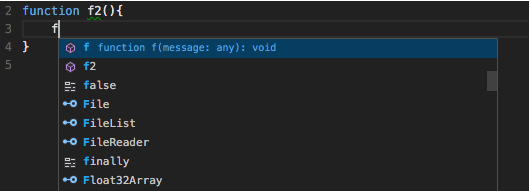
and

function f2(){

f("Hello");

}

Pay attention that when you type f(“Hello”), Visual Studio Code already knows about f() despite it is a separate file and provides IntelliSense features:



Add the following **jsconfig.json** file:

{

"compilerOptions": {

"target": "ES5",

"module": "commonjs"

},

"files": [

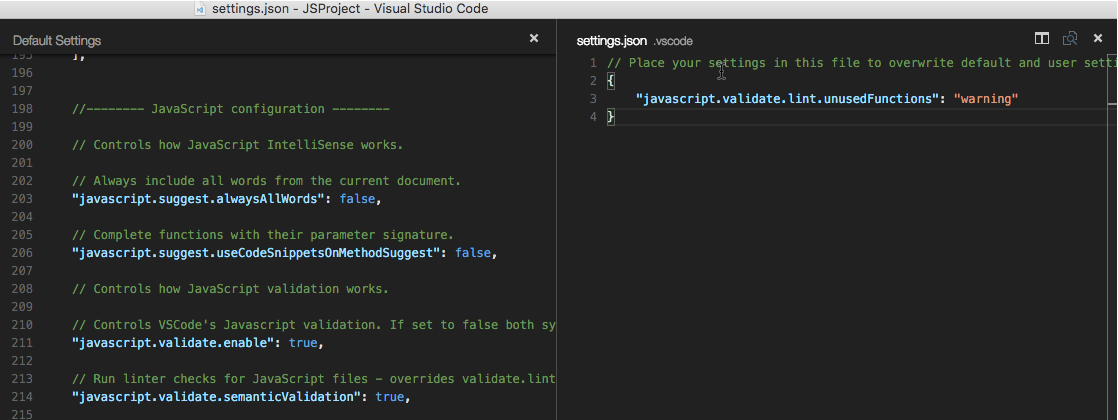
"file2.js"

]

}

And you will see that **f** function was removed from IntelliSense.

You will be surprised but some useful features for JavaScript are disabled by default. For example, if you have unused functions or variables, VS Code doesn’t even notify you. In order to change this behavior, you need to open Workspace Settings or User Settings and find JavaScript editor parameters there:



For example, if you enable warnings for unused functions, VS Code will notify you at once using the status bar:



if you click the message in Status bar, Visual Studio code will display detailed information and provide an opportunity to navigate to sources of the problems:

