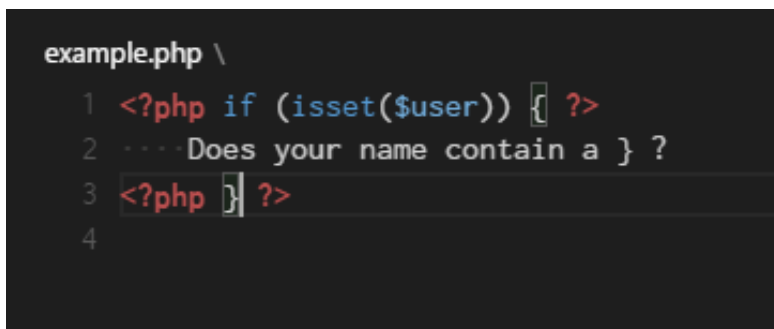


# Editing Evolved

Visual Studio Code features a battle-tested code editor that has most of the industry standard features, but also has some delights. We've been using it to build VS Code and we hope you'll love it too. This topic will walk you through some of the notable features of the code editor.

## Bracket matching

Matching brackets will be highlighted as soon as the cursor is near one of them. The right bracket will always be found, regardless of embedded languages.



```
example.php \
1  <?php if (isset($user)) { ?>
2  ...Does your name contain a } ?
3  <?php } ?>
4
```

**Tip:** You can jump to the matching bracket with Ctrl+Shift+\

## Selection & Multi-cursor

VS Code has support for multiple cursors. You can add secondary cursors (rendered thinner) with Alt+Click. Each cursor operates independently based on the context it sits in. The most common way to add more cursors is with Shift+Alt+Down or Shift+Alt+Up that insert cursors below or above.

**Note:** Your graphics card provider might overwrite these default shortcuts.



```
31 .global-message-list.transition {
32 →  -webkit-transition: top 200ms linear;
33 →  -ms-transition:      top 200ms linear;
34 →  -moz-transition:     top 200ms linear;
35 →  -khtml-transition:   top 200ms linear;
36 →  -o-transition:       top 200ms linear;
37 →  transition:          top 200ms linear;
38 }
```

Ctrl+D selects the word at the cursor, or the next occurrence of the current selection. Ctrl+K Ctrl+D moves the last added cursor to next occurrence of the current selection.

**Tip:** You can add more cursors also with Ctrl+Shift+L, which will add a selection at each occurrence of the current selected text or with Ctrl+F2, which will add a selection at each occurrence of the current word.

## Shrink/expand selection

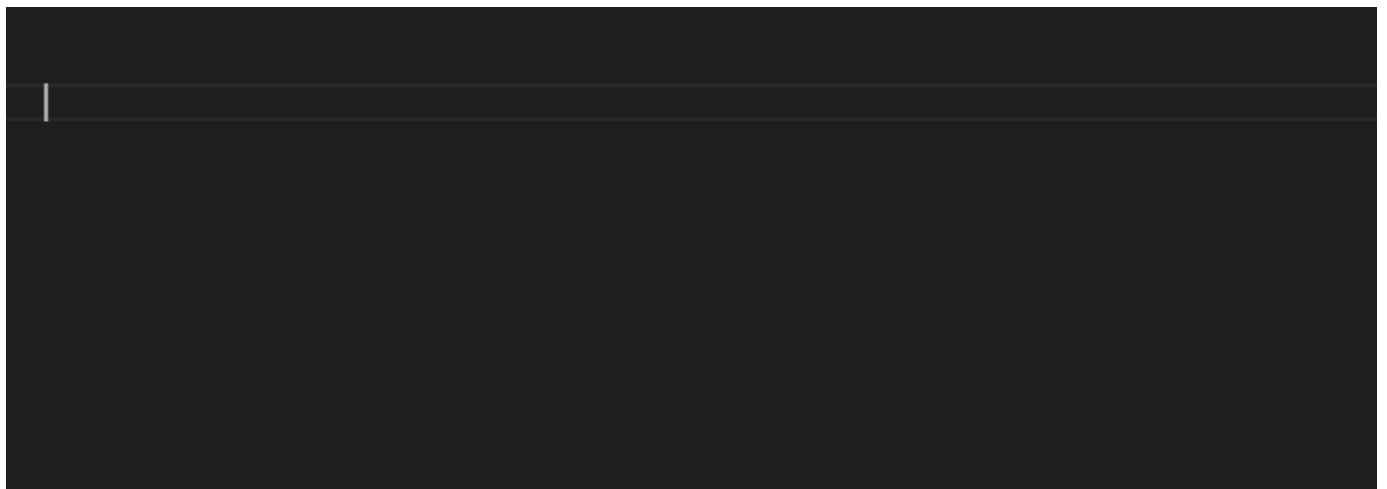
Quickly shrink or expand the current selection (applies to all languages). Trigger it with Shift+Alt+Left and Shift+Alt+Right

Here's an example of expanding the selection with Shift+Alt+Right:

```
7 namespace Hello1
8 {
    0 references
9     class Program
10    {
        0 references
11        static void Main(string[] args)
12        {
13            System.Console.WriteLine("Hello World!");
14        }
15    }
16 }
```

## IntelliSense

We'll always offer word completion, but for the rich [languages](#), such as JavaScript, JSON, HTML, CSS, Less, Sass, C# and TypeScript, we offer a true IntelliSense experience. If a language service knows possible completions, the IntelliSense suggestions will pop up as you type (we call it affectionately 24x7 IntelliSense). You can always manually trigger it with Ctrl+Space. Out of the box, ., Tab or Enter are accept triggers but you can also [customize these key bindings](#).

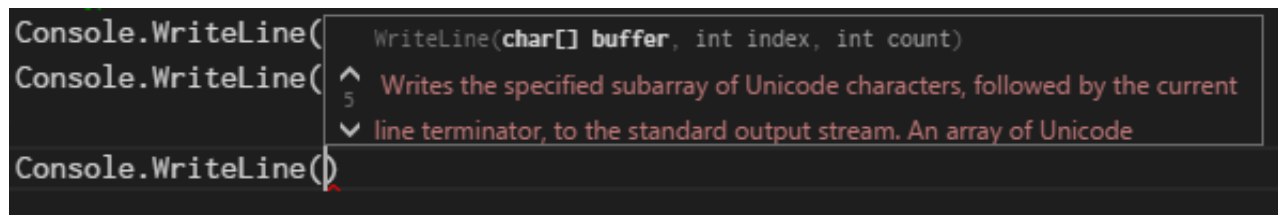


**Tip:** The suggestions filtering supports CamelCase so you can type the upper case letters of a method name to limit the suggestions. For example, "wl" will quickly bring up WriteLine.

**Tip:** The 24x7 IntelliSense can be configured via the `editor.quickSuggestions` and `editor.suggestOnTriggerCharacters` settings.

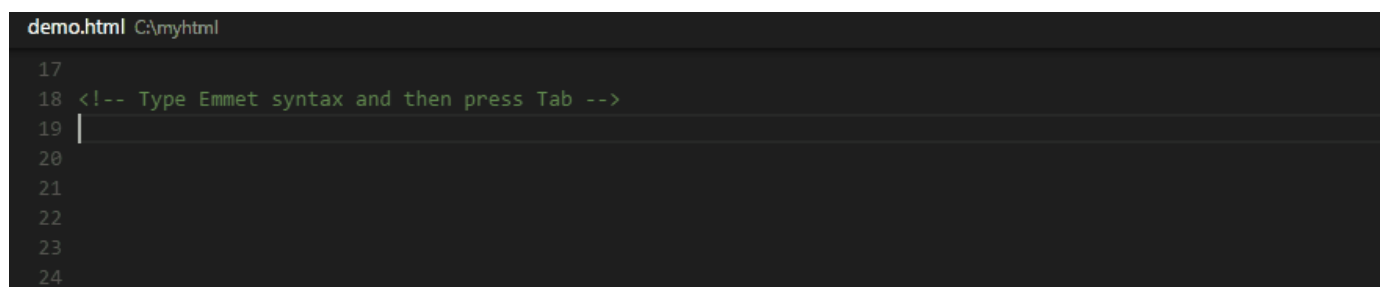
## Parameter Hints

In JavaScript, TypeScript or C#, parameter hints will pop up as you're typing a method invocation. You can navigate between different overloads with Up and Down and the best overload will be presented based on the arguments you pass in.



## Snippets and Emmet Abbreviations

We offer built-in snippets across languages as well as support for [Emmet abbreviations](#). You can expand Emmet abbreviations in HTML, Razor, CSS, Less, Sass, XML or Jade with Tab.



(See the [Emmet cheat sheet](#) for syntax examples.)

You can also define your own snippets: Open **User Snippets** under **File > Preferences** and select the language for which the snippets should appear. Find out more about this in the [customization section](#) of our docs.

## Go to Definition

If a [language](#) supports it, you can go to the definition of a symbol by pressing F12.

If you press Ctrl and hover over a symbol, a preview of the declaration will appear:

```
2 namespace Model
3 {
4
5     class User
6     {
7
8         public void use()
9         {
10
11             var toy = new Toy("train");
12
13         }
14
15     }
```

```
public Toy(string name)
{
    this.name = name;
}
```

```
Toy.Toy(string name)
```

**Tip:** You can jump to the definition with Ctrl+Click or open the definition to the side with Ctrl+Alt+Click. If you opened a new editor window, you can go back to the previous editor with Ctrl+Shift+Alt+Left.

## Goto Symbol

You can navigate symbols inside a file with Ctrl+Shift+O. By typing : the symbols will be grouped by category. Just press Up or Down and navigate to the place you want.

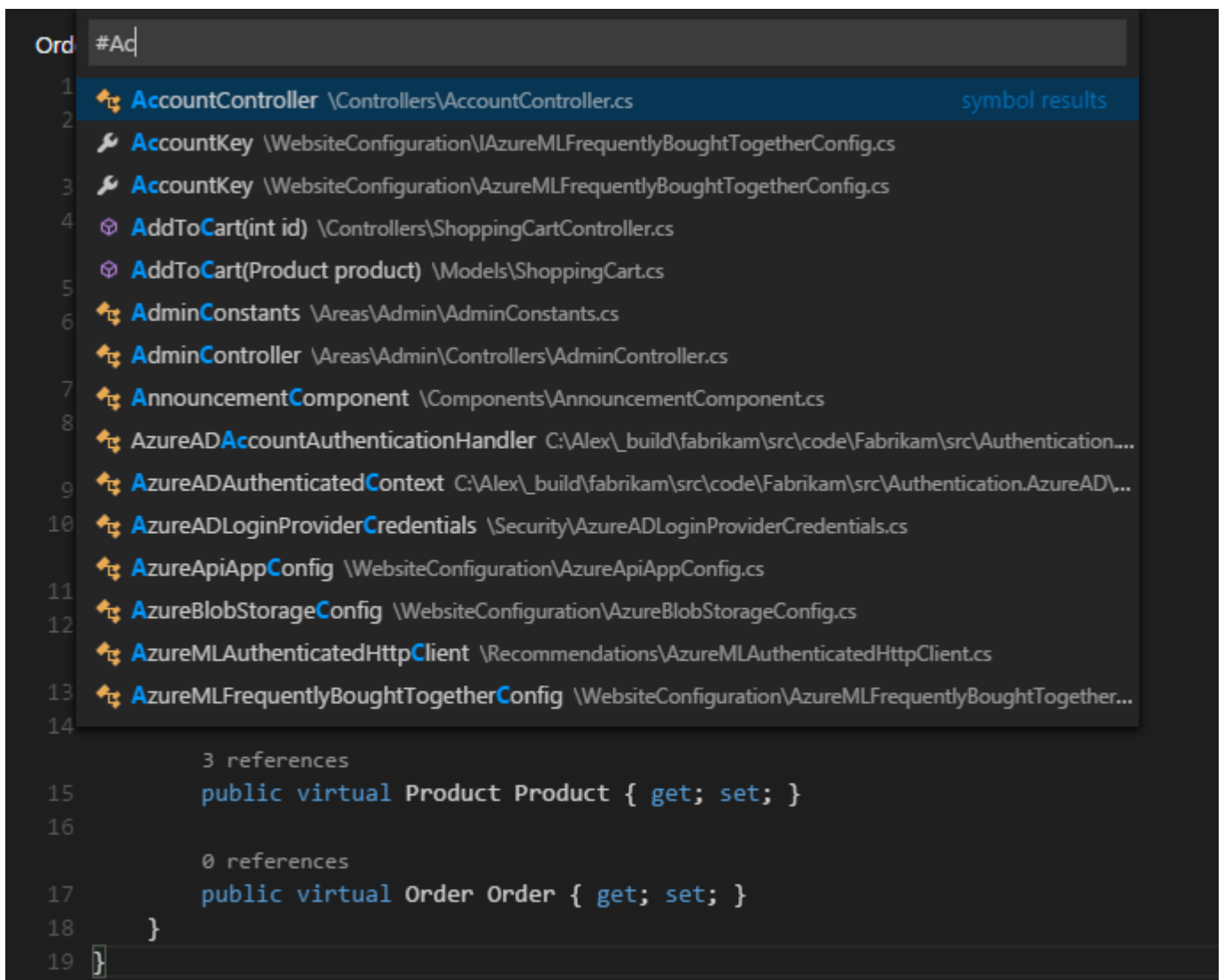
```
editor.ts \client\vs\editor
2242 → /**
2243 →  * Get a unique id
2244 →  */
2245 → getId(): string;
2246 → /**
2247 →  * Get the dom node
2248 →  */
2249 → getDomNode(): HTML
2250 → /**
2251 →  * Get the placemen
2252 →  * If null is retur
2253 →  */
2254 → getPosition(): IOve
2255 → }
2256
2257 /**
2258  * Options for creating the editor.
2259  */
2260 export interface ICodeEditorWidgetCreationOptions extends IEditorOptions {
2261     model?: IModel;
2262 }
2263
```

```
@JEditOp
- ICodeEditorWidgetCreationOptions interfaces (11)
- ICommonEditorOptions
- IDiffEditorOptions
- IEditOperationBuilder
- IEditorOptions
- IEditorPosition
- IEditorScrollbarOptions
- IIdentifiedSingleEditOperation
- IInternalEditorOptions
- ISingleEditOperation
- ISingleEditOperationIdentifier
getInverseEditOperations() ICursorStateComputerData methods (1)
KEYBINDING_CONTEXT_EDITOR_HAS_MULTIPLE_SELECTIONS "editor" variables (2)
```

```
lines)`|Delete Line|`editor.a
lineAfter)`|Insert Line Below
lineBefore)`|Insert Line Above
esDownAction)`|Move Line Dow
esUpAction)`|Move Line Up|`e
esDownAction)`|Copy Line Dow
esUpAction)`|Copy Line Up|`e
ighlights)`|Select all occur
ll)`|Select all occurrences o
ursorBelow)`|Insert Cursor B
ursorAbove)`|Insert Cursor A
racket)`|Jump to matching br
ines)`|Indent Line|`editor.a
lines)`|Outdent Line|`editor
gining of Line|`cursorHome`
66 `kb(cursorEnd)`|Go to End of Line|`cursorEnd`
67 `kb(cursorBottom)`|Go to End of File|`cursorBottom`
68 `kb(cursorTop)`|Go to Beginning of File|`cursorTop`
69 `kb(editor.action.commentLine)`|Toggle Line Comment|`
70 `kb(editor.action.blockComment)`|Toggle Block Comment|`
71 `kb(actions.find)`|Find|`actions.find`
72 `kb(editor.action.startFindReplaceAction)`|Replace|`e
```

## Open symbol by name

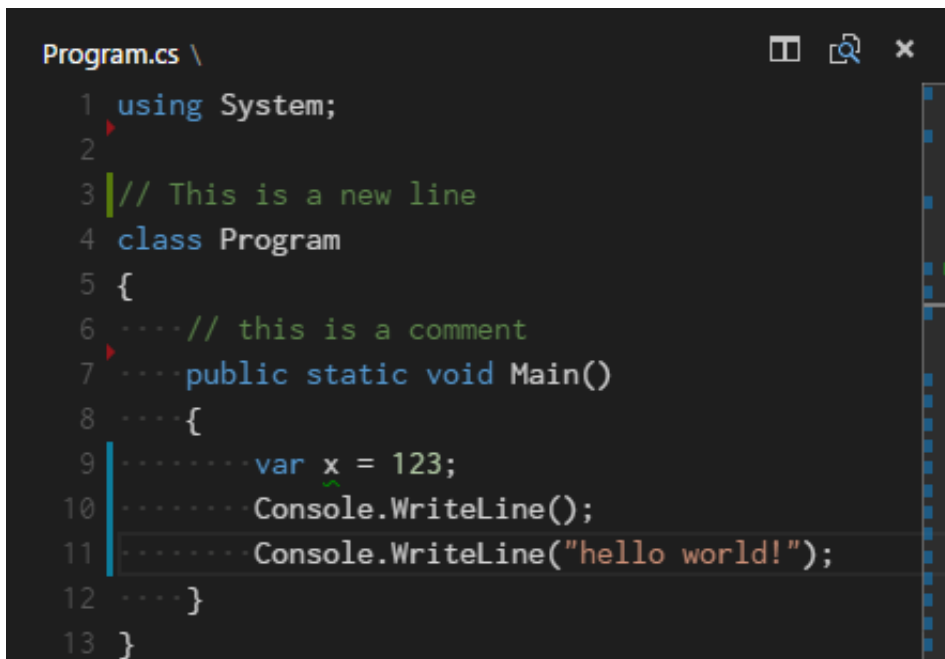
In C# and in TypeScript, you can jump to a symbol across files with Ctrl+T. Just type the first letter of a type you want to navigate to, regardless of which file contains it, and press Enter.



## Gutter indicators

If you open a folder that is a Git repository and begin making changes, VS Code will add useful annotations to the gutter and to the overview ruler.

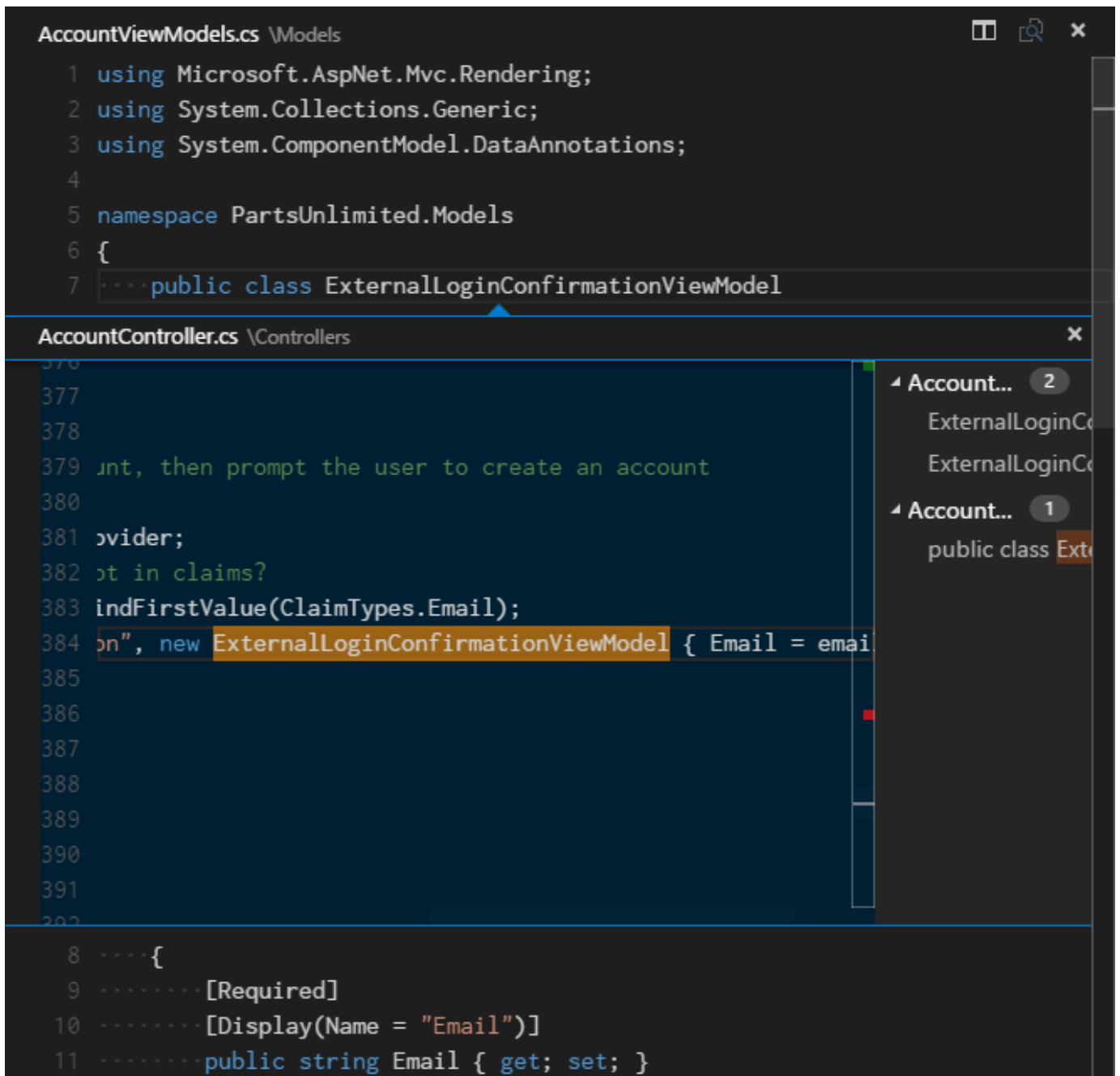
- A red triangle indicates where lines have been deleted
- A green bar indicates new added lines
- A blue bar indicates modified lines



```
Program.cs \  
1 using System;  
2  
3 // This is a new line  
4 class Program  
5 {  
6     // this is a comment  
7     public static void Main()  
8     {  
9         var x = 123;  
10        Console.WriteLine();  
11        Console.WriteLine("hello world!");  
12    }  
13 }
```

## Peek

We think there's nothing worse than a big context switch when all you want is to quickly check something. That's why we support peeked editors. When you execute a Reference Search (via Shift+F12), or a Peek Definition (via Ctrl+Shift+F10), we embed the result inline:



**Tip:** You can navigate between different references in the peeked editor and, if you need to, you can even make quick edits right there!

**Tip:** Clicking on the peeked editor filename or double-clicking in the result list will open the reference in the outer editor.

## Hover

For languages that support it, the hover will show useful information, such as types of symbols, or, in the case of CSS below, the shape of the HTML that would match a certain CSS rule:

```

.FishMenuItem
{
→   font-family: Opificio;
→   color: ■ #00a3ef;
  <element class="header">
    --
    <element class="LoadingCover">
      --
      <p>
.header .LoadingCover p
{
→   margin-top: 300px;
→   text-align: center;
→   color: ■ #4312ff;
→   font-size: 16pt;
}

```

## Reference information

C# supports inline reference information, that is live updated. This allows you to quickly analyze the impact of your edit or the popularity of your specific method or property throughout your project:



OrderDetail.cs \Models

```
1 namespace PartsUnlimited.Models
2 {
3     11 references
4     public class OrderDetail
5     {
6         1 reference
7         public int OrderDetailId { get; set; }
8
9         3 references
10        public int OrderId { get; set; }
11
12        4 references
13        public int ProductId { get; set; }
14
15        7 references
16        public int Quantity { get; set; }
17
18        4 references
19        public decimal UnitPrice { get; set; }
20
21        3 references
22        public virtual Product Product { get; set; }
23
24        0 references
25        public virtual Order Order { get; set; }
26    }
27 }
```

**Tip:** Directly invoke the Find References action by clicking on these annotations.

**Tip:** Reference information can be turned on or off through the `editor.referenceInfos` setting.

## Rename symbol

TypeScript and C# support rename symbol across files. Simply press F2 and then type the new desired name and press Enter. All usages of the symbol will be renamed, across files.

```
3 references
public int OrderId { get; set; }
      OrderIdentifier
4 references
public int ProductId { get; set; }
```

## Code action

JavaScript and CSS support code actions. A lightbulb will appear if there is a code action for the

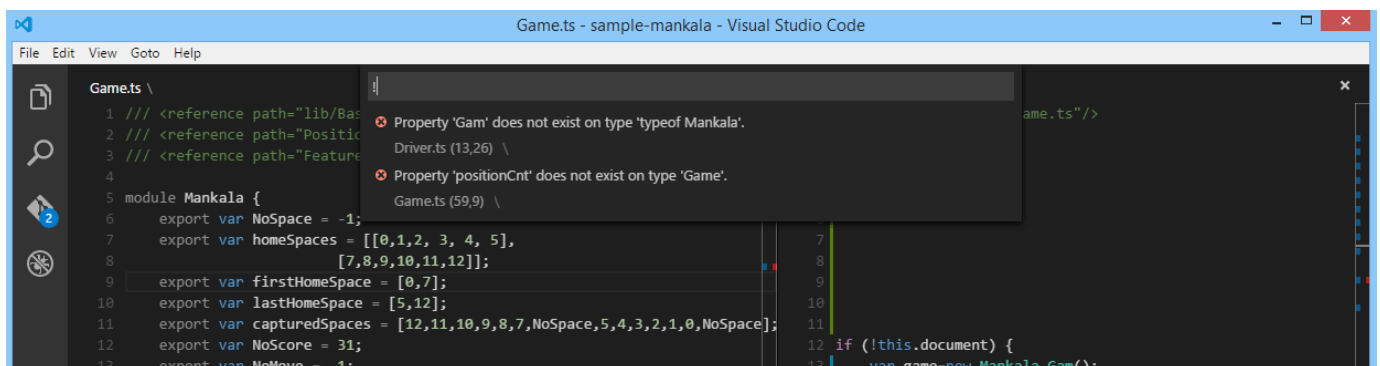
problem under the cursor. In this JavaScript example, due to the usage of `__dirname`, which is a Node.js built-in variable, the code action will propose to download and add a reference to `node.d.ts`, which contains all Node.js definitions.

```
1
2 var fs = require('fs'),
3     path = require('path');
4
5 var resourceFile = path.join(__dirname, './resource.txt');
```

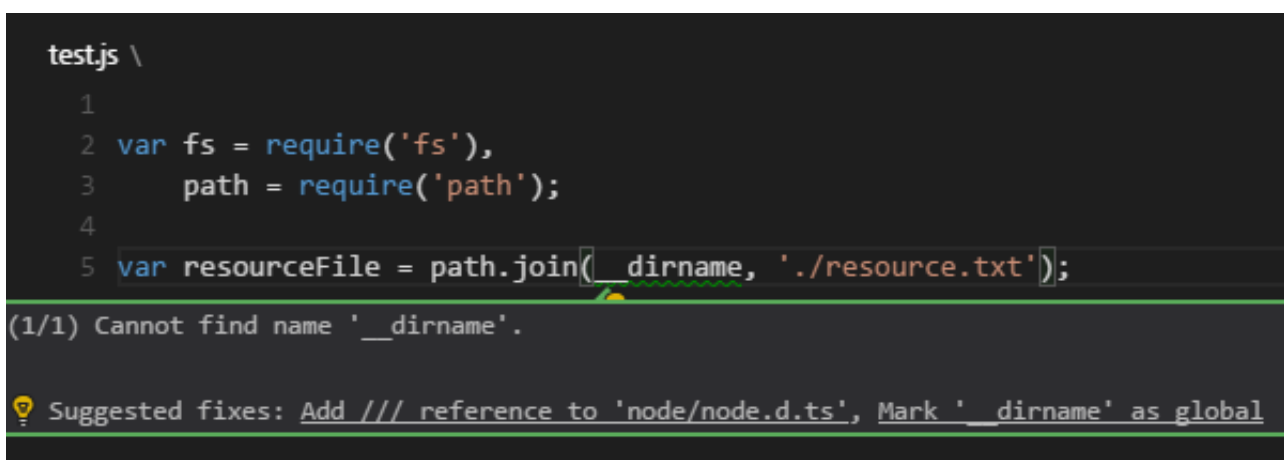
## Errors & Warnings

Warnings or Errors can be generated either via [configured tasks](#) or by the rich language services, that constantly analyze your code in the background. Since we love bug-free code, warnings and errors show up in multiple places:

- In the status line there is a summary of all errors and warnings counts.
- You can click on the summary or press Ctrl+Shift+M to see a list of all current errors.
- If you open a file that has errors or warnings, they will be rendered inline with the text and in the overview ruler.



**Tip:** To loop through errors or warnings in the current file, you can press F8 or Shift+F8 which will show an inline zone detailing the problem and possible code actions (if available):



## Next Steps

Now that you know how the editor works, time to try a few other things...

- [Why VS Code](#) - Why we exist and where we think we can help
- [The Basics](#) - Basic orientation around VS Code
- [Debugging](#) - This is where VS Code really shines
- [Customization](#) - Configure VS Code the way you want - Themes, Settings

### Was this documentation helpful?

Last updated on 3/7/2016