

Use of var keyword in C#

Asked 16 years, 3 months ago Modified 9 years, 10 months ago Viewed 150k times

405

votes



Locked. This question and its answers are [locked](#) because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

After discussion with colleagues regarding the use of the 'var' keyword in C# 3 I wondered what people's opinions were on the appropriate uses of type inference via var?

For example I rather lazily used var in questionable circumstances, e.g.:-

```
foreach(var item in someList) { // ... } // Type of 'item' not clear.
var something = someObject.SomeProperty; // Type of 'something' not clear.
var something = someMethod(); // Type of 'something' not clear.
```

More legitimate uses of var are as follows:-

```
var l = new List<string>(); // Obvious what l will be.
var s = new SomeClass(); // Obvious what s will be.
```

Interestingly LINQ seems to be a bit of a grey area, e.g.:-

```
var results = from r in dataContext.SomeTable
               select r; // Not *entirely clear* what results will be here.
```

It's clear what results will be in that it will be a type which implements IEnumerable, however it isn't entirely obvious in the same way a var declaring a new object is.

It's even worse when it comes to LINQ to objects, e.g.:-

```
var results = from item in someList
               where item != 3
               select item;
```

This is no better than the equivalent foreach(var item in someList) { // ... } equivalent.

There is a real concern about type safety here - for example if we were to place the results of that query into an overloaded method that accepted IEnumerable<int> and IEnumerable<double> the caller might inadvertently pass in the wrong type.

`var` does maintain strong typing but the question is really whether it's dangerous for the type to not be immediately apparent on definition, something which is magnified when overloads mean compiler errors might not be issued when you unintentionally pass the wrong type to a method.

`c#` `type-inference` `var`

Share

edited Feb 3, 2015 at 3:25

community wiki

10 revs, 7 users 57%
ljs

68 Using `var` is fine, but "*I don't have to figure out the type*" seems like a **very bad** reason to use it... you're supposed to **know** what the type is, `var` is just a shortcut to avoid typing it
– [Thomas Levesque](#) May 19, 2010 at 14:03

53 `var i = 0; == fail!` `var c = new CrazyFrigginLongClassName(); == win!` – [dotjoe](#)
May 19, 2010 at 14:08

6 "Var also reminds me of the VB/VBA variant type. It also had its place. I recall (from many years ago) its usage being less-than-desirable type and it was rather resource hungry." <- shows this is a baiting question as the 'var' type of C# has nothing to do with the VB/VBA variant type. – [user7116](#) May 19, 2010 at 15:02

20 `var readabilityBeDamned = true;` – [spoulson](#) May 19, 2010 at 15:14

9 The problem with all the examples stated here is that we're looking at ONE line of code. When I see line after line of "var" declarations, one after the other, it gets out of hand. Readability is subjective, but I find `var` abused far more than used respectably. – [jro](#) Jun 13, 2011 at 4:48 ✎

Comments disabled on deleted / locked posts / reviews |

86 Answers

Sorted by: Highest score (default)



1 2 3 Next

292

votes



```
var orders = cust.Orders;
```



I don't care if `Customer.Orders` is `IEnumerable<Order>`, `ObservableCollection<Order>` or `BindingList<Order>` - all I want is to keep that list in memory to iterate over it or get its count or something later on.

Contrast the above declaration with:

```
ObservableCollection<Order> orders = cust.Orders;
```

To me, the type name is just noise. And if I go back and decide to change the type of the Customer.Orders down the track (say from `ObservableCollection<Order>` to `ICollection<Order>`) then I need to change that declaration too - something I wouldn't have to do if I'd used `var` in the first place.

Share

edited Jun 19, 2012 at 12:35

community wiki

2 revs, 2 users 95%

Matt Hamilton

-
- 48 I like having the explicit type in front of me when I'm reading code. How do I know what "cust.Orders" is here without the type? Yes, I could hover my mouse over to find out, but why should I have to? :) – [Jon Tackabury](#) Oct 21, 2008 at 18:30
-
- 77 But the point is that in general it doesn't matter. Provided cust.Orders is something you can enumerate (eg foreach over) then it doesn't matter what type it is. The extra code just gets in the way of reading the intent. – [Matt Hamilton](#) Oct 21, 2008 at 19:53
-
- 67 But if you only require that cust.Orders is "something you can enumerate" then doesn't declaring it as an `IEnumerable<Order>` make that requirement explicit and clear? Declaring it as `var` means you effectively lose that requirement. – [GrahamS](#) Apr 15, 2009 at 10:03
-
- 5 @jon and how would `IEnumerator orders = cust.Orders` make a difference? the only thing `IEnumerator` says is that you can put it in a foreach but you already knew that from the line below – [Rune FS](#) Feb 4, 2010 at 22:08
-
- 18 "the only thing `IEnumerator` says is that you can put it in a foreach" - it also expresses the intention that the *only* thing you can do is enumerate. Using `var` gives access to and intellisense for public members of the concrete collection type. – [to StackOverflow](#) Apr 29, 2010 at 5:45
-

167

votes



I use `var` extensively. There has been criticism that this diminishes the readability of the code, but no argument to support that claim.

Admittedly, it may mean that it's not clear what type we are dealing with. So what?

This is actually the point of a decoupled design. When dealing with interfaces, you are emphatically *not* interested in the type a variable has. `var` takes this much further, true, but I think that the argument remains the same from a readability point of view: The programmer shouldn't actually be interested in the type of the variable but rather in what a variable *does*. This is why Microsoft also calls type inference "duck typing."

So, what does a variable do when I declare it using `var`? Easy, it does whatever IntelliSense tells me it does. Any reasoning about C# that ignores the IDE falls short

of reality. In practice, every C# code is programmed in an IDE that supports IntelliSense.

If I am using a `var` declared variable and get confused what the variable is there for, there's something fundamentally wrong with my code. `var` is not the cause, it only makes the symptoms visible. Don't blame the messenger.

Now, the C# team has released a coding guideline stating that `var` should *only* be used to capture the result of a LINQ statement that creates an anonymous type (because here, we have no real alternative to `var`). Well, screw that. As long as the C# team doesn't give me a sound argument for this guideline, I am going to ignore it because in my professional and personal opinion, it's pure baloney. (Sorry; I've got no link to the guideline in question.)

Actually, there are some (superficially) [good explanations](#) on why you shouldn't use `var` but I still believe they are largely wrong. Take the example of “searchability”: the author claims that `var` makes it hard to search for places where `MyType` is used. Right. So do interfaces. Actually, why would I want to know where the class is used? I might be more interested in where it is instantiated and this will still be searchable because somewhere its constructor has to be invoked (even if this is done indirectly, the type name has to be mentioned somewhere).

Share

answered Sep 3, 2008 at 12:01

community wiki
[Konrad Rudolph](#)

-
- 14 In any decent IDE, you don't use a text search to obtain class usage, you get the IDE to do it based on it's parse tree or however else it identifies the types of objects. Since we're talking about static typing, this will find everything but the anonymous types. – [Dustman](#) Sep 18, 2008 at 20:33
-
- 6 I'd hate to inherit 100k lines of source with no documentation and liberal use of var. Especially if you combine var with less-than-helpful variable names. I could see it being helpful when illustrating a point (or dealing with anonymous types) but in production code? – [Shea](#) Jul 16, 2009 at 20:53
-
- 7 Arnshea: well you've already pointed to the real problem there: the useless *variable names*, and the missing *documentation*. I really fail to see what `var` contributes to the confusion. Certainly, there may be cases where it is important to emphasize the type/size of a variable (e.g. low-level bit operations). That's fine: nobody ever claimed that `var` should be used exclusively. The rule is simple: if it really causes confusion, don't use it. – [Konrad Rudolph](#) Jul 17, 2009 at 6:56
-
- 17 Every example opposed to var I've seen assumes that the programmer will be using meaningless variable names. But maybe this is why var is *better* than explicitly specifying a type: It forces the programmer to come up with good variable names. – [Ryan Lundy](#) Jul 26, 2009 at 16:46
-

16 Microsoft also calls type inference “duck typing.” — do they really? I'd be shocked...
– [Anton Tykhyy](#) Aug 19, 2009 at 7:56

118

votes



Var, in my opinion, in C# is a *good thing*tm. Any variable so typed is still strongly typed, but it gets its type from the right-hand side of the assignment where it is defined. Because the type information is available on the right-hand side, in most cases, it's unnecessary and overly verbose to also have to enter it on the left-hand side. I think this significantly increases readability without decreasing type safety.

From my perspective, using good naming conventions for variables and methods is more important from a readability perspective than explicit type information. If I need the type information, I can always hover over the variable (in VS) and get it. Generally, though, explicit type information shouldn't be necessary to the reader. For the developer, in VS you still get Intellisense, regardless of how the variable is declared. Having said all of that, there may still be cases where it does make sense to explicitly declare the type -- perhaps you have a method that returns a `List<T>`, but you want to treat it as an `IEnumerable<T>` in your method. To ensure that you are using the interface, declaring the variable of the interface type can make this explicit. Or, perhaps, you want to declare a variable without an initial value -- because it immediately gets a value based on some condition. In that case you need the type. If the type information is useful or necessary, go ahead and use it. I feel, though, that typically it isn't necessary and the code is easier to read without it in most cases.

Share

edited Feb 3, 2015 at 3:24

answered May 19, 2010 at 13:50



[Robert Harvey](#)

181k ● 48 ● 346 ● 512



[tvanfosson](#)

532k ● 102 ● 699 ● 798

-
- 3 I agree overall. The key point here in my opinion is to be certain that intent is clear. If it isn't clear to the majority developers on the team, then it is detrimental, not helpful. That said, I personally am a HUGE fan of this, but have had to rein myself in a bit when other devs on my team have had trouble determining my intent. Go figure. – [Steve Brouillard](#) May 19, 2010 at 14:15
-
- 3 Eh, I find it easier to read when the type is on the left. Using ReSharper, I don't have to retype the type on the right anyways, so it doesn't bother me. – [BlueRaja - Danny Pflughoeft](#) May 19, 2010 at 16:06
-
- 1 @BlueRaja - I find that using good variable names usually eliminates the need to bother with the actual type anyway for understanding. Intellisense is still available on "var" defined variables, so I don't need to know the type to choose a method/property on it when coding. – [tvanfosson](#) May 19, 2010 at 16:11
-
- 2 It is not so much about when you are coding as it is when you have to read the code. – [BlueRaja - Danny Pflughoeft](#) May 19, 2010 at 16:34
-

- 1 I must just be on thick side then, I need to good Method and variable names **and** an understanding of the types being operated on to be able to properly understand the code I'm reading. I think you are right though it is symptomatic of bigger problems. One of trust. To be sure that the code is doing what it appears to be doing you need to be **sure** of the types being used. – [AnthonyWJones](#) May 21, 2010 at 11:11

91

votes



Neither of those is absolutely true; `var` can have both positive and negative effects on readability. In my opinion, `var` should be used when either of the following is true:

1. The type is anonymous (well, you don't have any choice here, as it *must* be `var` in this case)
2. The type is obvious based upon the assigned expression (i.e. `var foo = new TypeWithAReallyLongNameTheresNoSenseRepeating()`)

`var` has no performance impacts, as it's syntactic sugar; the compiler infers the type and defines it once it's compiled into IL; there's nothing *actually* dynamic about it.

Share

edited Jan 30, 2013 at 15:37



[Andrew Whitaker](#)

126k ● 32 ● 295 ● 308

answered May 19, 2010 at 13:49



[Adam Robinson](#)

186k ● 36 ● 292 ● 350

- 1 agree on both, though I have long restrictions over the second case, if a type is that long the it's usually a generic type with lots of nested generic arguments, in that case a strongly typed "ShortType equivalent to TypeWithAReallyLongNameTheresNoSenseRepeating" makes more sense – [user90843](#) May 19, 2010 at 13:59 ✎
- 12 I have no desire to start a religious war, but I personnaly tend to disagree with Ion. I would much prefer a very long, but very precise type name (ala Uncle Bob Martin) than an abbreviated and possibly ambiguous shorter type name. I will add the caveat that I DON'T agree with artificially inflating the length of the name either. If 5 characters creates a concise name that clearly shows intent, then use 5 and not 25. – [Steve Brouillard](#) May 19, 2010 at 14:06
- 2 @Steve: I have to agree with you; creating a "short type" (which I assume you mean as inheriting from the specific generic type solely for the purpose of shortening the name) is not a good practice; it will require you to duplicate and pass through any constructors from the generic type, as well as preventing you from passing a *different* type that inherits from the generic type to the function. You're using inheritance like a `typedef` when it isn't. – [Adam Robinson](#) May 19, 2010 at 14:19
- 7 Use `var` when the type is obvious? Maybe, but the true gain is when the type *doesn't matter*. If you're doing things like `var customers = whatever; var query = from c in customers select stuff` it doesn't matter what the exact type of 'customers' is. It's obvious how you can use it, and that's enough. And if the actual type is cumbersome, it's worthwhile to suppress it. – [Joren](#) May 19, 2010 at 15:02
- 2 @Kristoffer: I can understand wanting to eliminate the mess of tags, but wrapping them into a class is not (IMO) an acceptable alternative, as you then cut off the possibility of any other

(legitimate) child class of that generic type from being a valid parameter. I would rather use an `using ShortType = LongGenericType<A, B, C>` directive at the top of a file, since that gives the same readability, doesn't require that you recreate the constructors, and doesn't eliminate child classes from being candidates. – [Adam Robinson](#) May 19, 2010 at 15:42

68 From Eric Lippert, a Senior Software Design Engineer on the C# team:

votes



Why was the `var` keyword introduced?

There are two reasons, one which exists today, one which will crop up in 3.0.

The first reason is that this code is incredibly ugly because of all the redundancy:

```
Dictionary<string, List<int>> mylists = new Dictionary<string,
List<int>>();
```

And that's a simple example – I've written worse. Any time you're forced to type exactly the same thing twice, that's a redundancy that we can remove. Much nicer to write

```
var mylists = new Dictionary<string, List<int>>();
```

and let the compiler figure out what the type is based on the assignment.

Second, C# 3.0 introduces anonymous types. Since anonymous types by definition have no names, you **need** to be able to infer the type of the variable from the initializing expression if its type is anonymous.

Emphasis mine. The whole article, [C# 3.0 is still statically typed, honest!](#), and the ensuing [series](#) are pretty good.

This is what `var` is for. Other uses probably will not work so well. Any comparison to JScript, VBScript, or dynamic typing is total bunk. Note again, `var` is **required** in order to have certain other features work in .NET.

Share

[edited May 6, 2011 at 17:31](#)

community wiki


[2 revs, 2 users 99%](#)

[Dustman](#)

I find Lippert's argument odd. No one types the second class name, they let Intellisense write it for them, but then he turns around and claims that var is better because the compiler/Intellisense works it out. You can't have it both ways! – [Dave](#) Oct 17, 2008 at 20:28

- 5 The C# team doesn't control intellisense, they control the compiler. In any event that's not the main issue. I don't think var would have made the 100 points on saving typing alone.

– [Dustman](#) Mar 9, 2009 at 5:19

@Dustman: var doesn't save typing at all. It makes you write more! – [Piotr Perak](#) Oct 20, 2011 at 20:45 

53 I think the use of var should be coupled with wisely-chosen variable names.

votes



I have no problem using var in a foreach statement, provided it's not like this:

```
foreach (var c in list) { ... }
```

If it were more like this:

```
foreach (var customer in list) { ... }
```

... then someone reading the code would be much more likely to understand what "list" is. If you have control over the name of the list variable itself, that's even better.

The same can apply to other situations. This is pretty useless:

```
var x = SaveFoo(foo);
```

... but this makes sense:

```
var saveSucceeded = SaveFoo(foo);
```

Each to his own, I guess. I've found myself doing this, which is just insane:

```
var f = (float)3;
```

I need some sort of 12-step var program. My name is Matt, and I (ab)use var.

Share

[edited Sep 3, 2008 at 11:42](#)

community wiki

[2 revs](#)

[Matt Hamilton](#)

-
- 6 Well the only thing wrong with "var f = (float)3;" is that it should be "var f = 3f" or "var f = 3.0 (cause single precision sucks)". – [MichaelGG](#) Oct 16, 2008 at 1:20
-

3 Heh yeah 3f or 3.0 is the way to go! Us var maniacs have to stick together! – [Matt Hamilton](#)
Oct 16, 2008 at 2:05

27 The real problem in that first example is "list", not "c". "list" of *what*? "list" should be renamed to "customers", or "customersWhoOweMoney", or "currentCustomers", or something far more descriptive. And once you have that, the "c" can stay as-is, because you already know what it'll contain. – [Ryan Lundy](#) Sep 16, 2009 at 17:12

4 Hi Matt! My name is Kenny and I'm a varaddict. – [kenny](#) Mar 21, 2010 at 12:37

var MattHamil = "Luke Skywalker"; //stripped a ton out of it – [Binoj Antony](#) Jun 2, 2010 at 5:51



40

votes



We've adopted the ethos "Code for people, not machines", based on the assumption that you spend multiple times longer in maintenance mode than on new development.

For me, that rules out the argument that the compiler "knows" what type the variable is - sure, you can't write invalid code the first time because the compiler stops your code from compiling, but when the next developer is reading the code in 6 months time they need to be able to deduce what the variable is doing correctly or incorrectly and quickly identify the cause of issues.

Thus,

```
var something = SomeMethod();
```

is outlawed by our coding standards, but the following is encouraged in our team because it increases readability:

```
var list = new List<KeyValuePair<string, double>>();  
FillList( list );  
foreach( var item in list ) {  
    DoWork( item );  
}
```


Share

[edited Apr 29, 2012 at 5:26](#)

community wiki

[3 revs, 3 users 93%](#)

[Dexter](#)

-
- 4 I've found that ("Code for people, not machines") to be an excellent guideline - following it can result in better code and helps avoid premature optimization. – [Thanatos](#) May 28, 2010 at 0:21 
-
- 3 I don't get `var list = new KeyValuePair<string, double>?` For me a list can have more than one thing. – [TTT](#) Oct 16, 2010 at 3:51
-
- "Code for people, not machines" - amen. – [user1068352](#) Dec 12, 2012 at 11:42
-

39

votes



It's not bad, it's more a stylistic thing, which tends to be subjective. It can add inconsistencies, when you do use `var` and when you don't.

Another case of concern, in the following call you can't tell just by looking at the code the type returned by `CallMe`:

```
var variable = CallMe();
```

That's my main complaint against `var`.

I use `var` when I declare anonymous delegates in methods, somehow `var` looks cleaner than if I'd use `Func`. Consider this code:

```
var callback = new Func<IntPtr, bool>(delegate(IntPtr hwnd) {  
    ...  
});
```

EDIT: Updated the last code sample based on Julian's input



Share

edited May 19, 2010 at 18:40

answered May 19, 2010 at 13:48



user90843

-
- 3 But do you really need to know the type right there at that line? You know that `CallMe` returns something; isn't it enough to know that a local variable named `variable` is created? Unless you expand your example, this isn't a very solid complaint, IMO. – [Randolpho](#) May 19, 2010 at 13:53 
-
- 2 It's not about not being verbose, it's about not letting the compiler do the syntax sugar for you. Consider this: `var getter...`, now this `Func<object> getter...`, with the second you know you don't have to provide any parameters and what it returns. You know from the start "what to do" and can make decisions faster, when designing something or refactoring. Having all the information at hand is more important than a few more characters. This only can be appreciated when you're working with lots of code. – [user90843](#) May 19, 2010 at 14:08 
-
- 2 Since we are all talking about Visual Studio anyway, what's the big deal about hovering your mouse over the variable for a second and just seeing what type it is? Much better than running

to the declaration anyway. – [Rubys](#) May 19, 2010 at 14:15

3 Generic variable and function names (`variable` , `CallMe`) make bad examples. However, if `CallMe()` was a function in some kind of a "phone application", then `var call = CallMe(); call.HangUp();` would make much more sense. – [Danko Durbić](#) May 19, 2010 at 14:32

3 @Randolpho: "what's the big deal about hovering your mouse over the variable for a second and just seeing what type is it?" It adds time to the maintenance overhead... plus one second is only the hover time rather than counting the keyboard to mouse context switch. I don't know about you, but I work with a deadline. Every time I have to hover over a variable to find out what type it is, is time that I could have better spent fixing a problem. – [Powerlord](#) May 19, 2010 at 14:49

36

votes



Var is not like variant at all. The variable is still strongly typed, it's just that you don't press keys to get it that way. You can hover over it in Visual Studio to see the type. If you're reading printed code, it's possible you might have to think a little to work out what the type is. But there is only one line that declares it and many lines that use it, so giving things decent names is still the best way to make your code easier to follow.

Is using Intellisense lazy? It's less typing than the whole name. Or are there things that are less work but don't deserve criticism? I think there are, and var is one of them.

Share

answered May 19, 2010 at 13:47



[Kate Gregory](#)

18.9k ● 8 ● 59 ● 86

6 +1, only person to note that `var` has nothing to do with `Variant`. – [user7116](#) May 19, 2010 at 15:03

27

votes



The most likely time you'll need this is for anonymous types (where it is 100% required); but it also avoids repetition for the trivial cases, and IMO makes the line clearer. I don't need to see the type twice for a simple initialization.

For example:

```
Dictionary<string, List<SomeComplexType<int>>>> data = new Dictionary<string, List<SomeComplexType<int>>>>();
```

(please don't edit the hscroll in the above - it kinda proves the point!!!)

vs:

```
var data = new Dictionary<string, List<SomeComplexType<int>>>>();
```

There are, however, occasions when this is misleading, and can potentially cause bugs. Be careful using `var` if the original variable and initialized type weren't identical. For example:

```
static void DoSomething(IFoo foo) {Console.WriteLine("working happily") }
static void DoSomething(Foo foo) {Console.WriteLine("formatting hard
disk...");}

// this working code...
IFoo oldCode = new Foo();
DoSomething(oldCode);
// ...is **very** different to this code
var newCode = new Foo();
DoSomething(newCode);
```

Share

edited Aug 19, 2009 at 8:52

community wiki

3 revs, 2 users 96%

Marc Gravell

1 (for info, you can get similar issues on anything where this is an implicit type conversion)
– Marc Gravell Aug 19, 2009 at 7:37

1 Disagree with the part that you don't want to see it twice in the same line. It is possible that in the future you are not directly creating the variable after declaring (for example in an if below the definition) than it might not directly be clear what the type is. Most developers are used to see the type directly at the beginning of the line especially in complex code you don't want to make reading the code more difficult. – Gertjan Aug 19, 2009 at 8:35

@TheVillageldiot - I rolled back your edit, because **on this occasion** the hscroll is related to the point ;-p – Marc Gravell Aug 19, 2009 at 8:52

2 @Gertjan - my brain is limited; if it is complex, I **don't** want to see it twice and have to start comparing (interface/concrete/etc). I'm happy to see it once and think "typed as one of them".
– Marc Gravell Aug 19, 2009 at 8:53

17

votes



You can't rely on mouse-overs for that.



Share

answered Sep 3, 2008 at 11:51

community wiki

Jon Limjap

17 Why the heck are you code reviewing on paper? Think of the trees! ;) – Dustman Sep 18, 2008 at 20:35

- 8 You can have the same problem without using var. The problem isn't var; the problem is bad variable names. If variable names are well-chosen, the use of var doesn't matter.

– [Ryan Lundy](#) Jul 26, 2009 at 16:49

I have a guy on my team that reviews his own code, and looks for bugs, by printing his code. His walls are COVERED with code. I once walked in and he had an entire large whiteboard covered with a single of his debugging exercises. It was probably ~10000 LOC – [Beep beep](#) Aug 14, 2009 at 3:55

Variable names alone don't solve the problem unless you go back to hungarian notation where the type is part of the name. – [Kevin Gale](#) Apr 30, 2010 at 18:00

17 I don't see what the big deal is..

votes



```
var something = someMethod(); // Type of 'something' not clear <-- not to the compiler!
```

You still have full intellisense on 'something', and for any ambiguous case you have your unit tests, right? (do you?)

It's not varchar, it's not dim, and it's certainly not dynamic or weak typing. It is stopping maddnes like this:

```
List<somethinglongtypename> v = new List<somethinglongtypename>();
```

and reducing that total mindclutter to:

```
var v = new List<somethinglongtypename>();
```

Nice, not quite as nice as:

```
v = List<somethinglongtypename>();
```


But then that's what [Boo](#) is for.

Share

answered [Sep 3, 2008 at 12:04](#)

community wiki
[Frep D-Oronge](#)

the type of 'something' is very clear to the compiler, internally, the "var" gets set to the return type of 'someMethod', it is clear to the compiler, but may not be clear to the developers working on the project. and Boo is growing on me quickly. – [stephenbayer](#) Sep 25, 2008 at 14:43

No, `v = List<somethinglongtypename>()`; is not even nicer. It is important to distinguish between introducing a new variable and assigning to an existing variable. – [HighCommander4](#) May 8, 2012 at 4:35 

Well if you have assigned to 'v' before in the current scope, then it's assignment to an existing. Otherwise it's assignment to new variable 'v'. Easy. – [Frep D-Orange](#) May 14, 2012 at 21:12

17

votes



If someone is using the `var` keyword because they don't want to "figure out the type", that is definitely the wrong reason. The `var` keyword doesn't create a variable with a dynamic type, the compiler still has to know the type. As the variable always has a specific type, the type should also be evident in the code if possible.

Good reasons to use the `var` keyword are for example:

- Where it's needed, i.e. to declare a reference for an anonymous type.
- Where it makes the code more readable, i.e. removing repetitive declarations.

Writing out the data type often makes the code easier to follow. It shows what data types you are using, so that you don't have to figure out the data type by first figuring out what the code does.

Share

edited May 19, 2010 at 14:04

answered May 19, 2010 at 13:51



[Guffa](#)

700k ● 110 ● 750 ● 1k

11

votes



Given how powerful Intellisense is now, I am not sure `var` is any harder to read than having member variables in a class, or local variables in a method which are defined off the visible screen area.

If you have a line of code such as

```
IDictionary<BigClassName, SomeOtherBigClassName> nameDictionary = new  
Dictionary<BigClassName, SomeOtherBigClassName>();
```

Is is much easier or harder to read than:

```
var nameDictionary = new Dictionary<BigClassName, SomeOtherBigClassName>();
```

Share

edited Aug 16, 2011 at 13:43

community wiki

2 revs

[Colin Desmond](#)

I didn't actually consider this, at it seems from the other question that is a fairly strong point for one other time to use it. – [Finglas](#) Jun 9, 2009 at 19:12

That was basically the only reason they implemented it (besides being able to declare anonymous types, but they could have made "var" a special keyword that was only for anonymous types.) – [mqp](#) Jun 9, 2009 at 19:13

10

votes



I think the key thing with VAR is to only use it where appropriate i.e. when doing things in Linq that it facilitates (and probably in other cases).

If you've *got* a type for something in the then you should use it - not to do so is simple laziness (as opposed to creative laziness which is generally to be encouraged - good programmers oft work very hard to be lazy and could be considered the source of the thing in the first place).

A blanket ban is as bad as abusing the construct in the first place but there does need to be a sensible coding standard.

The other thing to remember is that its not a VB type var in that it can't change types - it **is** a strongly typed variable its just that the type is inferred (which is why there are people that will argue that its not unreasonable to use it in, say, a foreach but I'd disagree for reasons of both readability and maintainability).

I suspect this one is going to run and run (-:

Murph

Share

answered [Sep 3, 2008 at 11:47](#)

community wiki
[Murph](#)

10

votes



Sure, `int` is easy, but when the variable's type is

`IEnumerable<MyStupidLongNamedGenericClass<int, string>>`, var makes things much easier.

Share

answered May 19, 2010 at 13:54



[Blorgbeard](#)

103k ● 50 ● 235 ● 276


2 +1 for this. `int` versus `var` is something to bicker about, but multiple nested generic types make `var` a godsend. This is where using the type name over and over really destroys the readability of code. – [Chris Farmer](#) May 19, 2010 at 14:09


This is absolutely the wrong reason for using it. If your code uses a nested generic type you should derive a specific named class for it. For example: `class IntStringEnumerator : IEnumerable<MyStupidLongNamedGenericClass<int, string>>` and then use the new name. That cleans up the code and describes the type better. – [xxbbcc](#) Feb 21, 2012 at 14:14

Alright, but my example was just hyperbole. `IntStringEnumerator i = new IntStringEnumerator()` is still too much typing. – [Blorgbeard](#) Feb 25, 2012 at 6:58

9 Stolen from the [post on this issue at CodingHorror](#):

votes

 *Unfortunately, you and everyone else pretty much got it wrong. While I agree with you that redundancy is not a good thing, the better way to solve this issue would have been to do something like the following:*



MyObject m = new();

Or if you are passing parameters:

Person p = new("FirstName", "LastName");

Where in the creation of a new object, the compiler infers the type from the left-hand side, and not the right. This has other advantages over "var", in that it could be used in field declarations as well (there are also some other areas that it could be useful as well, but I won't get into it here).

In the end, it just wasn't intended to reduce redundancy. Don't get me wrong, "var" is VERY important in C# for anonymous types/projections, but the use here is just WAY off (and I've been saying this for a long, long time) as you obfuscate the type that is being used. Having to type it twice is too often, but declaring it zero times is too few. Nicholas Paldino .NET/C# MVP on June 20, 2008 08:00 AM

I guess if your main concern is to have to type less -- then there isn't any argument that's going to sway you from using it.

If you are only going to **ever** be the person who looks at your code, then who cares? Otherwise, in a case like this:

```
var people = Managers.People
```

it's fine, but in a case like this:

```
var fc = Factory.Run();
```

it short circuits any immediate type deductions my brain could begin forming from the 'English' of the code.

Otherwise, just use your best judgment and programming 'courtesy' towards others who might have to work on your project.

Share

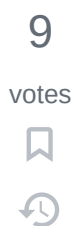
edited Dec 24, 2008 at 15:00

community wiki

2 revs

Rostov

-
- 4 Your examples above aren't an argument for not using var; they're an argument for using good descriptive variable names. If, instead of [var fc = Factory.Run();] you had [bool fc = Factory.Run();], the code wouldn't become any clearer. – [Ryan Lundy](#) Jul 13, 2009 at 15:20
-



Using `var` instead of explicit type makes refactorings much easier (therefore I must contradict the previous posters who meant it made no difference or it was purely "syntactic sugar").

You can change the return type of your methods without changing every file where this method is called. Imagine

```
...  
List<MyClass> SomeMethod() { ... }  
...
```

which is used like

```
...  
IList<MyClass> list = obj.SomeMethod();  
foreach (MyClass c in list)  
    System.Console.WriteLine(c.ToString());  
...
```

If you wanted to refactor `SomeMethod()` to return an `IEnumerable<MySecondClass>`, you would have to change the variable declaration (also inside the `foreach`) in every place you used the method.

If you write

```
...  
var list = obj.SomeMethod();  
foreach (var element in list)  
    System.Console.WriteLine(element.ToString());  
...
```

instead, you don't have to change it.

Share

answered [Aug 19, 2009 at 7:21](#)

community wiki
[MartinStettner](#)

agreed - wondering why this nice side effect isn't being touted as much as some of the other, more subjective pros in the more popular responses. – [fieldingmellish](#) Nov 23, 2009 at 18:44

It hapen to me today. I have a factory class that return instance of a MainMenu class. Today i created a MainMenu2 with the same interface of MainMenu. I have my all app code untouched after the change! – [Marco Staffoli](#) Nov 9, 2010 at 17:02

8 @aku: One example is code reviews. Another example is refactoring scenarios.

votes



Basically I don't want to go type-hunting with my mouse. It might not be available.



Share

answered [Sep 3, 2008 at 12:02](#)

community wiki
[erlando](#)

2 It's interesting you say that because var can make refactoring simpler. If you've used var you don't have to. Now you can always rely on the IDE's refactoring tools, but you know, you can always rely on the IDE for the type as well :) – [Fred](#) Nov 27, 2009 at 9:43

8 It's a matter of taste. All this fussing about the *type* of a variable disappears when you get used to dynamically typed languages. That is, *if* you ever start to like them (I'm not sure if everybody can, but I do).

votes



C#'s `var` is pretty cool in that it *looks* like dynamic typing, but actually is **static** typing - the compiler enforces correct usage.

The type of your variable is not really that important (this has been said before). It should be relatively clear from the context (its interactions with other variables and methods) and its name - don't expect *customerList* to contain an `int` ...

I am still waiting to see what my boss thinks of this matter - I got a blanket "go ahead" to use any new constructs in 3.5, but what will we do about maintenance?

Share

answered [Sep 3, 2008 at 12:10](#)

community wiki
[Daren Thomas](#)

7

votes



In your comparison between `IEnumerable<int>` and `IEnumerable<double>` you don't need to worry - if you pass the wrong type your code won't compile anyway.

There's no concern about type-safety, as `var` is **not** dynamic. It's just compiler magic and any type unsafe calls you make will get caught.

`var` is absolutely needed for Linq:

```
var anonEnumeration =
    from post in AllPosts()
    where post.Date > oldDate
    let author = GetAuthor( post.AuthorId )
    select new {
        PostName = post.Name,
        post.Date,
        AuthorName = author.Name
    };
```

Now look at *anonEnumeration* in intellisense and it will appear something like

`IEnumerable<'a>`

```
foreach( var item in anonEnumeration )
{
    //VS knows the type
    item.PostName; //you'll get intellisense here

    //you still have type safety
    item.ItemId;   //will throw a compiler exception
}
```

The C# compiler is pretty clever - anon types generated separately will have the same generated type if their properties match.

Outside of that, as long as you have intellisense it makes good sense to use `var` anywhere the context is clear.

```
//less typing, this is good
var myList = new List<UnreasonablyLongClassName>();

//also good - I can't be mistaken on type
var anotherList = GetAllOfSomeItem();

//but not here - probably best to leave single value types declared
var decimalNum = 123.456m;
```

Share

edited Sep 7, 2008 at 17:54

community wiki

3 revs

Keith

Please rid off `IQueryable<T>` before iterating: `anonEnumeration.ToList();`

– [David Diez](#) Jan 14, 2013 at 11:23

@DavidDiez could you rephrase? Your statement makes no sense. None of my code snippets reference `IQueryable` or `.ToList()` – [Keith](#) Jan 14, 2013 at 12:51

```
var anonEnumeration = from post in AllPosts() where post.Date >
oldDate let author = GetAuthor( post.AuthorId ) select new {
PostName = post.Name, post.Date, AuthorName = author.Name
};
```

 does not return an `IQueryable<T>` ? – [David Diez](#) Jan 14, 2013 at 15:06

-
- 1 @DavidDiez it depends on what `AllPosts()` returns - the asker refers to `List<T>` so I assumed that. In that case the result is that `anonEnumeration` will be of type `IEnumerable<'a>`. Now if `AllPosts()` returns `IQueryable<T>` instead then `anonEnumeration` will become `IQueryable<'a>` (note no `i` in `Queryable`) - however in that case my code *still* works because `IQueryable<T>` implements `IEnumerable<T>`. There are loads of better Q&As on here about the distinctions between them - here my case is that `'a` is anonymous and `var` allows you to assign it to a statically typed variable. – [Keith](#) Jan 15, 2013 at 12:18

Ohh I see, thanks for explaining it :) My comment was because iterating an `IQueryable<T>` is not a good practice because in each iteration you are making a read statement in DB. Be sure to `*.ToList()` `IQueryable<T>` 's before iterating them – [David Diez](#) Jan 16, 2013 at 11:10

7

votes



I guess it depends on your perspective. I personally have never had any difficulty understanding a piece of code because of `var` "misuse", and my coworkers and I use it quite a lot all over. (I agree that Intellisense is a huge aid in this regard.) I welcome it as a way to remove repetitive cruft.

After all, if statements like

```
var index = 5; // this is supposed to be bad

var firstEligibleObject = FetchSomething(); // oh no what type is it
                                              // i am going to die if i don't
know
```

were really that impossible to deal with, nobody would use dynamically typed languages.

Share

answered [Jun 9, 2009 at 19:08](#)

community wiki

[mqp](#)

Presumably in .Net 4 where dynamic types are common place, this will become more important? – [Colin Desmond](#) Jun 9, 2009 at 19:14

- 2 On the contrary, if you're confused by "var" now, I would expect that you will be additionally confused by "dynamic." God forbid anyone ever declares a dynamic and then makes a reference to it using "var" :) – [mqp](#) Jun 9, 2009 at 19:22

I meant something like `dynamic d = 52; var x = d;` which ought to be fine. – [mqp](#) Aug 7, 2009 at 13:53

7 I only use var when it's clear to see what type is used.

votes



For example, I would use var in this case, because you can see immediately that x will be of the type "MyClass":

```
var x = new MyClass();
```

I would NOT use var in cases like this, because you have to drag the mouse over the code and look at the tooltip to see what type MyFunction returns:

```
var x = MyClass.MyFunction();
```

Especially, I **never** use var in cases where the right side is not even a method, but only a value:

```
var x = 5;
```

(because the compiler can't know if I want a byte, short, int or whatever)

Share

answered May 19, 2010 at 14:09



[Christian Specht](#)

36.4k ● 15 ● 131 ● 188

If the right hand side isn't clear enough to justify the use of `var`, then `var` isn't the problem: the right hand side is the problem. It's not descriptive *enough*. `Customer c = GetContext()` is *still* unclear and no better than using `var`. – [JulianR](#) May 19, 2010 at 18:52

6 To me, the antipathy towards `var` illustrates why bilingualism in .NET is important. To those C# programmers who have also done VB .NET, the advantages of `var` are intuitively obvious. The standard C# declaration of:

votes



```
List<string> whatever = new List<string>();
```

is the equivalent, in VB .NET, of typing this:

```
Dim whatever As List(Of String) = New List(Of String)
```

Nobody does that in VB .NET, though. It would be silly to, because since the first version of .NET you've been able to do this...

```
Dim whatever As New List(Of String)
```

...which creates the variable and initializes it all in one reasonably compact line. Ah, but what if you want an `IList<string>`, not a `List<string>`? Well, in VB .NET that means you have to do this:

```
Dim whatever As IList(Of String) = New List(Of String)
```

Just like you'd have to do in C#, and obviously couldn't use `var` for:

```
IList<string> whatever = new List<string>();
```

If you *need* the type to be something different, it can be. But one of the basic principles of good programming is reducing redundancy, and that's exactly what `var` does.

Share

answered Sep 19, 2008 at 20:45

community wiki
Ryan Lundy

-
- 1 Funny you mention bilingualism as something that promotes the use of `var`. My antagonism towards the `var` keyword stems directly from my fluency in javascript! :) – [urig](#) Jan 17, 2010 at 20:16

`var` in C# has no connection whatsoever with `var` in JavaScript. A `var`-declared variable in C# is strongly-typed. – [Ryan Lundy](#) Dec 23, 2010 at 0:23

6

votes



Use it for anonymous types - that's what it's there for. Anything else is a use too far. Like many people who grew up on C, I'm used to looking at the left of the declaration for the type. I don't look at the right side unless I have to. Using `var` for any old declaration makes me do that all the time, which I personally find uncomfortable.

Those saying 'it doesn't matter, use what you're happy with' are not seeing the whole picture. Everyone will pick up other people's code at one point or another and have to deal with whatever decisions they made at the time they wrote it. It's bad enough having to deal with radically different naming conventions, or - the classic gripe - bracing styles, without adding the whole '`var` or not' thing into the mix. The worst case

will be where one programmer didn't use `var` and then along comes a maintainer who loves it, and extends the code using it. So now you have an unholy mess.

Standards are a good thing precisely because they mean you're that much more likely to be able to pick up random code and be able to grok it quickly. The more things that are different, the harder that gets. And moving to the 'var everywhere' style makes a **big** difference.

I don't mind dynamic typing, and I don't mind implicit typing - in languages that are designed for them. I quite like Python. But C# was designed as a statically explicitly-typed language and that's how it should stay. Breaking the rules for anonymous types was bad enough; letting people take that still further and break the idioms of the language even more is something I'm not happy with. Now that the genie is out of the bottle, it'll never go back in. C# will become balkanised into camps. Not good.

Share

answered [Sep 25, 2008 at 14:34](#)

community wiki
[Neil Hewitt](#)

7 Wow. Ignoring all the arguments brought forth in this thread so far and re-setting the whole discussion is quite an achievement. – [Konrad Rudolph](#) [Sep 25, 2008 at 14:45](#)

This 100% describes how I feel about 'var', particularly from the standpoint of picking up someone else's code. Use of var radically changes the task at hand if it's littered throughout. +1
– [Simon](#) [Oct 16, 2012 at 11:22](#)

6 Many time during testing, I find myself having code like this:

votes



```
var something = myObject.SomeProperty.SomeOtherThing.CallMethod();  
Console.WriteLine(something);
```

Now, sometimes, I'll want to see what the `SomeOtherThing` itself contains, `SomeOtherThing` is not the same type that `CallMethod()` returns. Since I'm using `var` though, I just change this:

```
var something = myObject.SomeProperty.SomeOtherThing.CallMethod();
```

to this:

```
var something = myObject.SomeProperty.SomeOtherThing;
```

Without `var`, I'd have to keep changing the declared type on the left hand side as well. I know it's minor, but it's extremely convenient.

6 For the aficionados that think `var` saves time, it takes less keystrokes to type:

votes



```
StringBuilder sb = new StringBuilder();
```

than

```
var sb = new StringBuilder();
```

Count em if you don't believe me...

19 versus 21

I'll explain if I have to, but just try it... (depending on the current state of your intellisense you may have to type a couple more for each one)

And it's true for every type you can think of!!

My personal feeling is that var should never be used except where the type is not known because it reduces recognition readability in code. It takes the brain longer to recognize the type than a full line. Old timers who understand machine code and bits know exactly what I am talking about. The brain processes in parallel and when you use var you force it to serialize its input. Why would anyone want to make their brain work harder? That's what computers are for.

Share

edited Dec 29, 2011 at 12:46

community wiki
2 revs, 2 users 91%
Wray Smallwood

I find the repetition of `StringBuilder sb = new StringBuilder()` messy and longer to clearly recognise. It's extra noise. The problem is, determining what does and doesn't constitute extra intellectual effort to understand some code is pretty subjective! – [ljs](#) Mar 12, 2010 at 12:59

"var should never be used except where the type is not known..." - You can ALWAYS know the type, as does the compiler. This is not dynamic typing, it just lets the compiler determine the type for you. But the type is never an unknown. – [Gabriel Magana](#) May 19, 2010 at 20:23

5 I split var all over the places, the only questionable places for me are internal short types, e.g. I prefer `int i = 3;` over `var i = 3;`

votes



Share

answered Sep 3, 2008 at 12:13

community wiki
robi-y



5

It can certainly make things simpler, from code I wrote yesterday:

votes



```
var content = new Queue<Pair<Regex, Func<string, bool>>>();  
...  
foreach (var entry in content) { ... }
```

This would have be extremely verbose without `var`.

Addendum: A little time spent with a language with *real* type inference (e.g. F#) will show just how good compilers are at getting the type of expressions right. It certainly has meant I tend to use `var` as much as I can, and using an explicit type now indicates that the variable is not of the initialising expression's type.

Share

edited Jun 9, 2009 at 19:18

community wiki
2 revs
Richard

Yup, compilers are smarter than us, get over it! – Benjol Dec 10, 2009 at 9:57

5

None, except that you don't have to write the type name twice.

<http://msdn.microsoft.com/en-us/library/bb383973.aspx>

votes



Share

answered Aug 19, 2009 at 7:09

community wiki
Ignas R

1

2

3

Next