Is PowerShell a strongly-typed language?

Asked 16 years, 3 months ago Modified 5 years, 10 months ago

Viewed 12k times

17

PowerShell is definitely in the category of dynamic languages, but would it be considered strongly typed?

powershell strong-typing

Share

Improve this question

Follow

edited Jan 22, 2016 at 12:53

Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Sep 22, 2008 at 15:43

Chris Sutton

2,781 ● 5 ● 26 ● 32

3 The term "strongly typed" is not defined. Could you please define it, so that your question can be answered objectively? Otherwise everybody will just substitute their *own* definition and we will end up in a massive flamefest. – Jörg W Mittag Sep 22, 2008 at 19:53

You mean: is PowerShell *statically* typed? – Kellen Stuart Nov 1, 2016 at 16:21

7 Answers

Sorted by: Highest score (default)

29

There is a certain amount of confusion around the terminlogy. This article explains a useful taxonomy of type systems.

PowerShell is dynamically, implicit typed:

```
> $x=100
> $x=dir
```

No type errors - a variable can change its type at runtime. This is like Python, Perl, JavaScript but different from C++, Java, C#, etc.

However:

```
> [int]$x = 100
> $x = dir
Cannot convert "scripts-2.5" to "System.Int32".
```

So it also supports *explicit* typing of variables if you want. However, the type checking is done at runtime rather than compile time, so it's not *statically* typed.

I have seen some say that PowerShell uses *type inference* (because you don't have to declare the type of a variable), but I think that is the wrong words. Type inference is a feature of systems that does type-checking at compile time (like "`var`" in C#). PowerShell only checks types at runtime, so it can check the actual value rather than do inference.

However, there is some amount of automatic type-conversion going on:

```
> [int]$a = 1
> [string]$b = $a
> $b
1
> $b.GetType()

IsPublic IsSerial Name
-------- -------- ----
True     True     String
```

So *some* types are converted on the fly. This will by most definitions make PowerShell a *weakly typed* language. It is certainly more weak than e.g. Python which (almost?) never convert types on the fly. But probably not at weak as Perl which will convert almost anything as needed.

Share Improve this answer

Follow

edited Jun 30, 2017 at 19:24

mklement0

434k ● 68 ● 698 ● 903

answered Sep 22, 2008 at 16:14

JacquesB

42.6k ● 13 ● 75 ● 88

3 It is worth mentioning that *dynamic typed* languages like PowerShell generally get worse error messages and more run-time death versus something *statically typed* like Java. However, even with *static typing* you can have run-time death with null pointer exceptions and such, you just get better error

messages and debugging with *static typed* languages.
– Kellen Stuart Nov 1, 2016 at 16:24

1 @KolobCanyon That's because it's a **scripting** language and typing `$obj.SetFlag("None")` in the console with an implicit convert is way easier than `$obj.SetFlag([Company.Example.Product.Data.Transfer.Enums.Flags]::None)` – marsze Dec 18, 2018 at 10:32

6

It can be if you need it to be.

Like so:

```
[1] » [int]$x = 5
[2] » $x
5
[3] » $x = 'haha'
Cannot convert value "haha" to type "System.Int32". Er
not in a correct format."
At line:1 char:3
+ $x  <<<< = 'haha'
[4] »
```

Use the [type] notation to indicate if you care about variables being strongly typed.

EDIT:

As edg pointed out, this doesn't prevent PowerShell from interpreting "5" as an integer, when executing (5 + "5"). I dug a little more, and according to Bruce Payette in Windows PowerShell in Action, PowerShell is actually a "type-promiscuous language." So, I guess, my answer is "sort of."

Share Improve this answer Follow

edited May 23, 2017 at 12:32

Community Bot
1 ● 1

answered Sep 22, 2008 at 15:48

David Mohundro
12.4k ● 5 ● 42 ● 45

Nope. You can add a string to an integer and get a numeric result. If you reverse the order of addition you get concatenation. – Ed Guiness Sep 22, 2008 at 16:04

I think this shows that it *can* be statically typed which does not mean that it *must* be strongly typed. – EBGreen Sep 22, 2008 at 16:17

So, it's weakly and statically typed? – John Millikin Sep 24, 2008 at 22:44

1 Sort of both, thus the "promiscuous" part. :)
– David Mohundro Oct 2, 2008 at 13:23

1

Technically it is a strongly typed language.

You can decline to declare types in the shell, allowing it to behave like a dynamic typed scripting language, but it will wrap weakly-typed objects in a wrapper of type "PsObject". By declaring objects using the "New-Object" syntax, objects are strongly typed and not wrappered.

```
$compilerParameters = New-Object System.CodeDom.Compil
```

Share Improve this answer Follow

answered Sep 22, 2008 at 15:48

Guy Starbuck
21.8k ● 7 ● 54 ● 64

1

I think you will need to define what you mean by "Strongly Typed":

> In computer science and computer programming, the term strong typing is used to describe those situations where programming languages specify one or more restrictions on how operations involving values having different datatypes can be intermixed. The antonym is weak typing. However, these terms have been given such a wide variety of meanings over the short history of computing that it is often difficult to know, out of context, what an individual writer means when using them.

--Wikipedia

Share Improve this answer Follow

answered Sep 22, 2008 at 16:06

EBGreen
37.6k ● 12 ● 68 ● 86

1

PowerShell is dynamically typed, plain and simple. It is described as such by its creator, Bruce Payette.

Additionally, if anyone has taken a basic programming language theory class they would know this. Just because there is a type annotation system doesn't mean it is strongly typed. Even the type annotated variables behave dynamically during a cast. Any language that allows you to assign a string to a variable and print it out and then assign a number to the same variable and do calculations with it is dynamically typed.

Additionally, PowerShell is dynamically scoped (if anyone here knows what that means).

Share Improve this answer Follow

edited Jan 13, 2017 at 15:39
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Apr 14, 2010 at 10:52
The Old Hag
11 ● 1

1

I think looking at the adding a String to an Int example further would provide more grist for the discussion mill. What is considered to be dynamic type casting? Someone in one of the comments said that in this case:

```
4 + "4"
```

The `"4"` **becomes an Int32**. I don't believe that is the case at all. I believe instead that an intermediate step happens where the command is changed to:

```
4 + [System.Convert]::ToInt32("4")
```

Note that this means that `"4"` stays a String through the entire process. To demonstrate this, consider this example:

```
19# $foo = "4"
20# $foo.GetType()

IsPublic IsSerial Name
-------- -------- ----
True     True     String


21# 4 + $foo
8
22# $foo.GetType()

IsPublic IsSerial Name
-------- -------- ----
True     True     String
```

Share Improve this answer

Follow

edited Feb 14, 2019 at 12:30

AdminOfThings
24.9k ● 4 ● 20 ● 29

answered Sep 22, 2008 at 16:27

EBGreen
37.6k ● 12 ● 68 ● 86

You can however change the type of a variable through assignment. If you do $foo="4" and then $foo = 8 + $foo, then the type of $foo will change from string to int. Values are immutable, though. – JacquesB Sep 25, 2008 at 16:11

Obviously 4 + $foo isn't going to modify the variable $foo. It's going to read the variable, get a String, and convert that to an Int32. This happens dynamically based on the runtime type (not the static type) of $foo. – Jesse Dec 1, 2019 at 17:43

-2

I retract my previous answer -- quoted below. I should have said something more nuanced like:

PowerShell has a strong type system with robust type inference and is dynamically typed.

It seems to me that there are several issues at work here, so the answers asking for a better definition of what was meant by a "strongly-typed language" were probably more wise in their approach to the question.

Since PowerShell crosses many boundaries, the answer to where PowerShell lies probably exists in a Venn diagram consisting of the following areas:

- Static vs. dynamic type checking
- Strong vs. weak typing
- Safe vs. unsafe typing
- Explicit vs. implicit declaration and inference
- Structural vs. nominative type systems

> > "PowerShell is a strongly typed language.
> >
> > However, it only requires you to declare the type where there is ambiguity.
> >
> > If it is able to infer a type, it does not require you to specify it."

Share Improve this answer Follow

edited Jan 22, 2016 at 12:58

Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Sep 22, 2008 at 16:03

Timothy Lee Russell
3,748 ● 1 ● 37 ● 45