

## Re: What's so great about lisp?

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2005-10/msg00211.html>

---

- *From:* Jon Harrop <[usenet@xxxxxxxxxxxxxxxx](mailto:usenet@xxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 06 Oct 2005 02:05:51 +0100
- 

Joe Marshall wrote:

- > The real issue is not about static vs. dynamic typing. I don't think
- > anyone in their right mind would object to the computer automatically
- > finding errors as soon as possible. There are actually two issues
- > that are of interest:
- >
- > 1. How much does the computer interfere with the task at hand when
- > it discovers errors, and
- >
- > 2. How willing is the programmer to change his style to aid (or
- > conform to) the computer's static checking.

Similarly:

How many errors can be fixed more quickly by exploiting static type checking?

How difficult is it to learn how to exploit static type checking?

How effective is static type checking at forcing the programmer to provide machine-checked interface definitions?

Also:

What functionality is lost in the trade with static type checking (e.g. type safe marshalling, dynamically typed printing, multiple or ad-hoc polymorphic arithmetic operators)?

- > `Strong-arm' type checking appears to be a popular choice in the
- > static typing camp. In this model, it is not possible to compile a
- > program until the type checker is satisfied that the program is free
- > from type errors.

SML and OCaml tend not to reject programs with incomplete pattern matches, IIRC. Are those type errors?

- > While it is true that once the program compiles it
- > will not have type errors (a trivial tautology), it can seriously
- > interfere with interactive development. When developing

Re: What's so great about lisp?

- > interactively, it doesn't matter whether the untaken branch is type
- > safe. Being forced to write type-safe `stubs' that won't be executed
- > is simply useless busywork.

Yes.

- > Static typing with inference and all the modern amenities is a far
- > cry from the static typing found in C or Java. Nonetheless, there are
- > programs that the type checker cannot prove correct, or cannot prove
- > correct without assistance from the programmer. The programmer must
- > either provide hints to the type checker \*or\* modify his programming
- > style to avoid constructs that will stymie the type checker. Some
- > people don't mind doing this because they percieve a real benefit to
- > satisfying the type checker.

Yes.

- > Personally, I want as much static checking as possible provided that
- > it \*never\* rejects a program, even one that is provably wrong (it
- > should compile to a call to the runtime error handler!),

and give a warning?

- > never complains when it cannot determine if the program is type safe,

You don't even want a warning that a program may be type-unsafe?

- > and requires exactly \*zero\* effort on my part to use.

This seems like a silly criterion to me. Static type checkers clearly save some programmer effort. If the overall result is that you have to do less work, why reject it?

- > I may be willing to
- > provide type hints and guarantees at some point of development, but I
- > never want the computer to force me to do so \*before\* I am ready.

Ok.

Like you, I want as much static type checking as possible but I don't care if it rejects code deemed to be incorrect provided it accepts most of my correct code. I don't mind learning about type inference and checking, and coercing my programming style to fit in with static typing provided that the resulting code remains concise and the additional functionality allows me to prove that parts of my production code cannot produce run-time errors (an otherwise infeasible task).

I am particularly keen to avoid deciphering stack traces whenever possible. That takes a long time and requires me to write a lot more code so I am willing to go to some lengths to detect errors at compile time.

Re: What's so great about lisp?

## Re: What's so great about lisp?

Also, I want to be able to choose between explicitly declared types and inferred types (e.g. records vs tuples and variants vs polymorphic variants in OCaml).

During development, I don't find myself fixing type errors in unused functions so that doesn't bother me.

—

Dr Jon D Harrop, Flying Frog Consultancy  
<http://www.ffconsultancy.com>

.

- 
- *Follow-Ups:*
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Alexander Schmolck
  
  - *References:*
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Joe Marshall
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Jon Harrop
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Joe Marshall
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Jon Harrop
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Joe Marshall
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Jon Harrop
    - ◆ ***Re: What's so great about lisp?***
      - ◇ *From:* Joe Marshall
  
  - Prev by Date: ***Re: What's so great about lisp?***
  - Next by Date: ***Re: What's so great about lisp?***
  - Previous by thread: ***Re: What's so great about lisp?***
  - Next by thread: ***Re: What's so great about lisp?***
  - Index(es):
    - ◆ ***Date***
    - ◆ ***Thread***