Types

Before we begin, all sources were accessed on 29 April 2016.

Type - to “write (something) on a typewriter or computer by pressing the keys.” — Google “define type”. Nah, just kidding, this is not the definition we are talking about.

“a category of people or things having common characteristics.” — Google “define type”

“a label given to a group sharing common characteristics” — my refined definition, building from above.

More specifically, for our purposes, “[a] set of values from which a variable, constant, function, or other expression may take its value.” — Free On-Line Dictionary Of Computing

“Assigning a data type—typing—gives meaning to a sequences of bits such as a value in memory or some object such as a variable. The hardware of a general purpose computer is unable to discriminate between for example a memory address and an instruction code, or between a character, an integer, or a floating-point number, because it makes no intrinsic distinction between any of the possible values that a sequence of bits might mean.” — Wikipedia, Type system#Fundamentals

Pop quiz

Static vs. Dynamic
Strong vs. Weak
Where does Ruby fall? Java? JavaScript? Python? Haskell? C?

	Strong	Weak
Static	Haskell Java	C
Dynamic	Ruby Python	JavaScript

Static typing – “[e]nforcement of type rules at compile time” — Free On-Line Dictionary Of Computing
Dynamic typing – “[e]nforcement of type rules at run time” — Free On-Line Dictionary Of Computing
Latent typing – “types are associated with values and not variables. […] This typically requires run-time type checking and so is commonly used synonymously with dynamic typing.” — Wikipedia, Latent typing

Strong typing – no commonly agreed upon definition — C2 Wiki, Strongly Typed

Some attempted definitions I find useful:

> “[a] language is strongly typed if there are checks for type constraint violations. If no checking is done, it is weakly typed.” — C2 Wiki, Strongly Typed
>
> “A language is strongly typed if there is no language-level way to disable or evade the type system. If there are casts or other type-evasive mechanisms, it is weakly typed.” — C2 Wiki, Strongly Typed
>
> “Strict enforcement of type rules with no exceptions. Incorrect type usage can be detected either at run time or at compile time.” — Free On-Line Dictionary Of Computing

Many people consider strong/weak typing to be a continuum on opposite ends of a spectrum, and I would agree (see, https://en.wikipedia.org/wiki/Type_system#Variable_levels_of_type_checking, and https://en.wikipedia.org/wiki/Type_system#Optional_type_systems). For example, in Haskell, there is no truthy, only True or False. That doesn't make languages like Ruby weak, just relatively weaker than Haskell (though they may still be stronger in other ways). I would add that dynamic/static typing are two different types of type systems, and are not mutually exclusive, and each fall on a separate continuum. There is also some debate to be had on whether implicit type conversion is a form of weak typing?

More types of type systems

- Gradual typing
- Soft typing
- Dependent types
- Flow typing
- Uniqueness type

- Gradual typing, e.g. Perl 6, Objective-C (for object pointers), uses special `dynamic` type.
- Soft typing – “Where the type checker can prove that the program is type safe, all is fine and dandy. Where the type checker can't prove correctness it informs the programmer and inserts appropriate type checks, but doesn't reject the program.” — C2 Wiki, Soft Typing
- Another type of soft typing meaning to combine static and dynamic type checking.
- Dependent types, “In computer science and logic, a dependent type is a type whose definition depends on a value. A "pair of integers" is a type. A "pair of integers where the second is greater than the first" is a dependent type because of the dependence on the value.” — Wikipedia, Dependant type
- And more, such as flow typing, uniqueness types, etc.

Polymorphism

Duck typing (convention-based) is generally used for dynamically typed languages. In statically typed languages, generics/templates/typeclasses, whatever you want to call them, are generally used.

Maybe demonstrate this with GHC?

Algebraic Data Types

Types that are defined in an algebraic fashion. Haskell uses an algebraic type system, types are composed by either adding together possible values that the type can take on or combining other types to form a distinct type (do *not* use the phrase "new type" here).

Maybe demonstrate this with GHC?

Recommended reading:

- The Algebra of Algebraic Data Types, Part 1
- What the Heck are Algebraic Data Types? (for Programmers)
- Haskell wiki's “Algebraic data type”
- The algebra (and calculus!) of algebraic data types
- Selected answer to the Programmers SE question, “What Are The Uses of Algebraic Data Types?”

Empirical evidence

“While one controlled experiment showed an increase in developer productivity for duck typing in single developer projects,[20] other controlled experiments on API usability show the opposite.[21][22]” — Wikipedia, Type system#Static and dynamic type checking in practice

The first study referred to was by Stefan Hanenberg. “An experiment about static and dynamic type systems: doubts about the positive impact of static type systems on development time” (OOPSLA 2010).

The second two are by “Kleinschmager, Hanenberg, Robbes, Tanter, Stefik: Do static type systems improve the maintainability of software systems? An empirical study” (ICPC 2012) and by the same experimenters, “An empirical study on the impact of static typing on software maintainability” (ESE 2014).

Recommended reading

Recommended reading

- Classifying Programming Languages
  - http://cs.lmu.edu/~ray/notes/pltypes
- An Introduction To Programming Type Systems
  - https://www.smashingmagazine.com/2013/04/introduction-to-programming-type-systems
- Wikipedia articles on type system and type safety
  - https://en.wikipedia.org/wiki/Type_safety
  - https://en.wikipedia.org/wiki/Type_system
- Null References: The Billion Dollar Mistake
  - http://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare
- What to know before debating type systems
  - https://cdsmith.wordpress.com/2011/01/09/an-old-article-i-wrote
- More on my blog if I've missed any here.
  - https://alaminium.me