Concepts of Programming Languages, CSCI 305, Fall 2021
Type Systems, Chapter 7, Nov. 1
Section 7.1 Overview
Section 7.2 Type Checking
Section 7.5 Summary and Concluding Remarks

Purpose of type systems:

- Implicit context for many operations.
- Limit the set of operations that are semantically valid so compiler can catch errors.
- If declared explicitly, can make the program easier to read and understand. Types are a kind of stylized documentation. (Easier to read and understand, harder to write.)
- Can help optimization.

JavaScript, Python and Ruby are strongly, but dynamically typed.
Java, C#, Rust are strongly and statically typed (C# gives choices)
C is weakly but statically typed

Languages are not purely one type or the other.

Strong verses Weak Typing

Strong – language defines each type of data. Variables and expressions must be described with one of these data types

Weak – many operations allowed without regard to types of the operands (typically dynamically typed)

Terms not well defined. Wikipedia says "No universally accepted definition for strong and weak typing

Static versus Dynamic Typing

Static – variables are typed at compile time
Dynamic – variables are typed at run-time

Gradual typing – variables may be typed at compile or run-time. Allow developers to choose static or dynamic within a single language

Duck typing (typically considered style of dynamic typing) – determine typing from how it is used, rather than the object's type

> "If it walks like a duck and it quacks like a duck, then it must be a duck."

Inference typing – automatic detection of the data type

Static versus Dynamic

Pros of Static Typing	**Pros of Dynamic Typing**
• Earlier detection of programming mistakes • Better documentation in the form of type signatures • More opportunities for compiler optimizations • Increased runtime efficiency	• Ideally suited for prototyping systems with changing or unknown requirements • Good when dealing with truly dynamic program behavior • Code is more reusable • Code is more concise • Code is not any less safe (some claim) • Code is not any less expressive (some claim)