Programming Paradigms

Type Systems (Part 1)

Prof. Dr. Michael Pradel
Software Lab, University of Stuttgart
Summer 2020
Quiz

What values do these JavaScript expressions evaluate to?

```
'' == '0'
0 == ''
0 == '0'
false == 'false'
```

Please vote via Ilias.
Quiz

What values do these JavaScript expressions evaluate to?

```
''    ==  '0'    // false
0     ==  ''     // true
0     ==  '0'    // true
false ==  'false' // false
```
Quiz

What values do these JavaScript expressions evaluate to?

```
''    ==    '0'                  // false
0     ==    ''                   // true
0     ==    '0'                  // true
false ==    'false'              // false
```

Two strings that are not the same

Please vote via Ilias.
Quiz

What values do these JavaScript expressions evaluate to?

```
''   ==  '0'       // false
0    ==  ''        // true
0    ==  '0'       // true
false ==  'false'  // false
```

Number and string:
String is coerced into a number (here: 0)

Please vote via Ilias.
Quiz

What values do these JavaScript expressions evaluate to?

```
''  ==  '0'   // false
0   ==  ''    // true
0   ==  '0'   // true
false ==  'false' // false
```

Boolean and another type:

- Boolean gets coerced to a number (here: 0)
- String also get coerced to a number (here: NaN)
- The two numbers differ

Please vote via Ilias.
Overview

- Introduction
- Types in Programming Languages
- Polymorphism
- Type Equivalence
- Type Compatibility
- Formally Defined Type Systems
Types

- Most PLs: Expressions and memory objects have types

- Examples
  - Assignment \( x=4 \) (implicitly) says \( x \) has a number type
  - Declaration \( \text{int } n; \) says \( n \) has integer type
  - Expression \( a+b \) has a type, which depends on the type of \( a \) and \( b \)
  - \( \text{new } X() \) has a type
Why Do We Need Types?

Reason 1: Provide context for operations

- Meaning of $a+b$ depends on types of $a$ and $b$
  - E.g., addition vs. string concatenation
- Meaning of `new x` depends in the type of $x$
  - E.g., which initialization code to call?

**PL implementation uses this context information**
Why Do We Need Types?

**Reason 2: Limit valid operations**

- Many syntactically valid operations don’t make any sense
  - Adding a character and a record
  - Computing the logarithm of a set

Helps **developers** find bugs early
Why Do We Need Types?

Reason 3: Code readability and understandability

- Types = stylized documentation
- Makes maintaining and extending code easier

But: Sometimes, types make code harder to write
Why Do We Need Types?

Reason 4: **Compile-time optimizations**

- Compiler knows that some behavior is impossible
  - E.g., assignment of type T1 may not influence values of type T2

Works both for **explicitly specified** and **implicitly inferred types**
Bits Are Untyped

- (Most) **hardware** stores and computes on **raw bits**
  - Bits may be code, integer data, addresses, etc.

- (Most) **assembly languages are untyped**
  - Operation of any kind can be applied to values at arbitrary locations
Type Systems

- Definition of types and their association with PL constructs
  - Every PL construct that has/refers to a value has a type (e.g., named constants, variables, record fields, functions)

- Rules for
  - Type equivalence
  - Type compatibility
  - Type inference
Ensure that program obeys the type compatibility rules

Example (Java):

```java
int a = 3;
String b = a - 2;
```
Type Checking

Ensure that program obeys the type compatibility rules

Example (Java):

```java
int a = 3;
String b = a - 2;
```

Type error: Can’t assign int value to String variable