



Options for lambdas

Jens Gustedt

► To cite this version:

| Jens Gustedt. Options for lambdas: (slides). 2022. hal-03553612

HAL Id: hal-03553612

<https://inria.hal.science/hal-03553612>

Preprint submitted on 2 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Options for lambdas

ISO/IEC JTC 1/SC 22/WG14 **N2893**
WG21

Jens Gustedt

INRIA – Camus

ICube – ICPS
Université de Strasbourg



<https://modernc.gforge.inria.fr/>



Table of Contents

1 Current state

2 Proposed options

3 Motivation



Closure

Explicit captures

- *identifier capture*: access an outer variable [&id0,...]
- *value capture*: frozen value of an expression [id0 = expr,...]

where ",..." is empty or a mixed list of both types of explicit captures.



Syntax

Ambiguities

not so bad



Table of Contents

1 Current state

2 Proposed options

3 Motivation



Closure

Implicit captures and explicit captures

- default identifier captures

[&,...]

where "**,...**" is empty or a mixed list of value or shadow captures.

- default shadow captures

[=,...]

where "**,...**" is empty or a mixed list of value or identifier captures.

Explicit shadow captures

- frozen value of an automatic variable

[id0,...]

where "**,...**" is empty or a mixed list of all three types of explicit captures.



Syntax

Ambiguities

Explicit shadow captures introduce new syntax ambiguities with

- designated initializers (bounded lookahead)
- attributes (unbounded lookahead)

Disambiguation

- Force consecutive `[ [` tokens to be the start of an attribute



Table of Contents

1 Current state

2 Proposed options

3 **Motivation**



Default captures

Migration

- users of nested functions and expression statements (`{ ... }`)
=> use `[&]`
- users of the block extension (clang)
=> use `[=]`

Please!

Possible future extensions

- together with wide function pointers
=> standardize nested functions and expression statements

Expressivity

- provides framework for other extensions, such as `defer`



Explicit shadow captures

Coding style

- Better to have an explicit list of shadow captures than [=] improve code after default migration

C++ compatibility



Grammar disambiguation

Explicit shadow capture

- Only really needed when these are introduced

C++ compatibility

