# Typed (Function) References Proposal status update

Andreas Rossberg

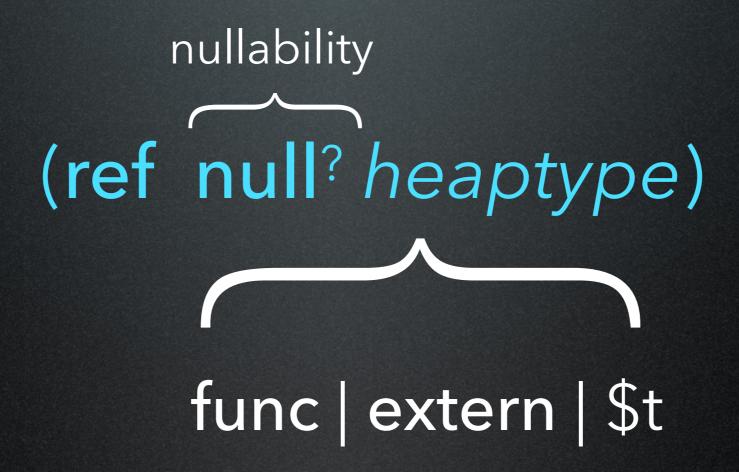


### Reference Types Refactored

(ref heaptype)

func extern \$t

### Reference Types Refactored



```
funcref = (ref null func)
externref = (ref null extern)
```

## Subtyping

(ref ht) <: (ref null ht)

\$t <: func

#### Null References

```
ref.null ht:
                             [] \rightarrow (ref null ht)
ref.is_null:
                             (ref null ht) \rightarrow i32
ref.as_non_null:
                             (ref null ht) \rightarrow (ref ht)
br_on_null $/:
                             (ref null \$t) \rightarrow (ref \$t)
br on non null $I: (ref null $t) \rightarrow []
```

#### Function References

```
ref.func f: [] \rightarrow (ref t)
```

where \$f:\$t

```
call_ref: t_1^* (ref null $t) \rightarrow t_2^*
where $t = func t_1^* \rightarrow t_2^*
```

### Recent Resolutions

### Defaultability

Locals and tables rely on default initialisation

Only nullable references have default value

### Initialisation Typing for Locals

locals with non-defaultable type start as unset

local.set marks variable as set

end of block resets to status quo ante

conservative, possible future extension to extend past end (e.g., extended block types)

#### Default Value for Tables

(table \$tab 10 (ref \$t) (ref.func \$f))

...may be omitted if type is nullable, shorthand for (ref.null ht)

#### Status

- Specification
- Reference interpreter
- Test suite
- ✓ Implemented in V8, SM, Wasmtime

Phase 2 (2021/10/26), no open issues or  $\overline{PRs}$ 

#### Status

JS API spec has no owner meets all requirements for phase 3 prerequisite for GC, stack switching call\_ref independently useful

#### Poll

Move to phase 3?