Oops! I Did It Again (I Broke Eo)

stosb.com/talks

Tom Hacohen
Samsung Open Source Group

tom@osg.samsung.com
@TomHacohen

Born To Make You Happy

(Why Change?)

- We thought we could do better
- Complaints about `eo_do()`
- Even more complaints about `eo_do_ret()`
- It was now or never

I'm Not a Girl, Not Yet a Woman

(Previous Incarnations)

```
/* Eo 1 */
#define EFL_TEXT_ID(sub_id) (EFL_TEXT_BASE_ID + (sub_id))
#define efl_text_set(text) EFL_TEXT_ID(EFL_TEXT_SET_OP_ID), \
    EO_TYPECHECK(const char *, text)

const char *tmp = NULL;
eo_do(obj, efl_text_set("text"), efl_text_get(&tmp));
if (!tmp) return;
```

```
/* Eo 2 */
void efl_text_set(const char *text);

const char *tmp;
eo_do(obj, efl_text_set("text"), tmp = efl_text_get());
if (!tmp) return;
```

Gimme More

(Improvements Over Eo 2)

- Faster
- Easier to debug
- Easier to use (No more `eo_do()`/`ret()`)
- More familiar syntax
- No more TLS and a custom call stack
- Upgrade path to Eo can be easier in some cases

Gimme More

(How Is It Faster?)

- No more custom stack to update
- No more TLS (slow)
- Because of the above - a lot less code
- Faster when less than four consecutive calls

Gimme More

(Faster When Less Than Four Calls?)

```
/* CONSECUTIVE */
eo_do(obj, efl_text_set("bla"),
           evas_object_move(100, 100),
           evas_object_resize(200, 200),
           evas_object_show());

/* Not CONSECUTIVE */
eo_do(obj, efl_text_set("bla"),
           evas_object_move(100, 100));

obj2 = eo_add(...);
eo_do(obj, elm_box_append(obj2),
           evas_object_show());
```

Gimme More

(How Is It Easier To Debug?)

- Better compiler warnings
- Macros confuse debuggers
- Easier to step without `eo_do()`
- Easier to step with only one function per line

Gimme More

(How Is It Easier To Use?)

```
/* Before */
const char *tmp, *tmp_name;
int tmp_foo;
eo_do(obj, tmp = efl_text_get());
if (tmp)
    eo_do(obj, tmp_name = efl_name_get());
    if (tmp_name) {
        do_do_super(obj, MY_CLASS, tmp_foo = foo_get(1));
        return tmp_foo;
    }
}

/* Now */
if (efl_text_get(obj) && efl_name_get(obj))
    return foo_get(eo_super(obj, MY_CLASS), 1);
```

Hold It Against Me

(Not Everything Is Better)

- The `eo_add()` syntax is slightly worse
- Slightly ugly `eo_add()` fallback code

Hold It Against Me

How Is eo_add() Worse?

```
/* Before */
obj = eo_add(CLASS, parent, efl_text_set("bla"),
                            efl_size_set(200, 200));

/* Now */
obj = eo_add(CLASS, parent, efl_text_set(eo_self, "bla"),
                            efl_size(eo_self, 200, 200));
```

Hold It Against Me

What eo_add() Fallback Code?

```
#if defined(__GNUC__) && !defined(_EO_ADD_FALLBACK_FORCE)
# define eo_self __eo_self
# define _eo_add_common(klass, parent, is_ref, ...) ({ \
     Eo * const __eo_self = _eo_add_internal_start(...); \
     (void) ((void)0, ##__VA_ARGS__); \
     (Eo *) _eo_add_end(eo_self, EINA_FALSE); })
#else
# define eo_self _eo_self_get()
# define _eo_add_common(klass, parent, is_ref, ...) ( \
     _eo_add_internal_start(...), \
     ##__VA_ARGS__, \
     (Eo *) _eo_add_end(eo_self, EINA_TRUE) )
#endif
```

…Baby One More Time

(Yet Another Guide to Using Eo)

- Basic usage
- Using `eo_super()`
- Using Eolian

...Baby One More Time

(Basic Usage)

```
/* Win has constructing functions. */
Elm_Win *win = eo_add(ELM_WIN_CLASS, NULL,
    elm_obj_win_name_set(eo_self, "My win"),
    elm_obj_win_type_set(eo_self, ELM_WIN_TYPE_BASIC));

Elm_Button *obj = eo_add(ELM_BUTTON_CLASS, win,
    elm_obj_theme_set(eo_self, "flat"),
    efl_text_set(eo_self, "OK!"),
    eo_event_callback_add(eo_self, EFL_EVENT_CLICKED, cb, data));
```

...Baby One More Time

(Basic Usage - Lifecycle)

- `eo_add()` for C
- `eo_add_ref()` for bindings
- `eo_add()` ↔ `eo_del()`
- `eo_ref()` ↔ `eo_unref()`

SAMSUNG
Open Source Group

...Baby One More Time

(Using `eo_super()`)

```
EOLIAN static Eina_Bool
_elm_slider_elm_widget_theme_apply(Eo *obj, Elm_Slider_Data *sd)
{
    if (!elm_obj_widget_theme_apply(eo_super(obj, MY_CLASS)))
        return EINA_FALSE;

    /* Theme changed, reapply settings */
    if (sd->range_enable)
        elm_layout_signal_emit(obj, "range,enable", "elm");

    if (_is_inverted(sd->orientation))
        elm_layout_signal_emit(obj, "inverted,on", "elm");
}
```

...Baby One More Time

Using Eolian - Methods

```
class Evas.Object (Eo.Base, Evas.Common_Interface)
{
   methods {
      @property visibility {
         [[The visibility status of the object]]
         set {
         }
         get {
         }
         values {
            visibility: bool; [[True if visible]]
         }
      }
   }
   // ...
}
```

...Baby One More Time

Using Eolian - Implements and Events

```
class Evas.Object (Eo.Base, Evas.Common_Interface)
{
   // ...
   implements {
      Eo.Base.constructor;
      Eo.Base.destructor;
      Eo.Base.dbg_info_get;
      Evas.Common_Interface.evas.get;
   }
   events {
       show; [[Show Event]]
       hide; [[Hide Event]]
       move: Move_Event_Info *; [[Move Event]]
       resize: Resize_Event_Info *; [[Resize Event]]
   }
}
```

Overprotected

(Best Practices)

- Use Eolian
- Mark Eolian implementations with `EOLIAN`
- Document properties not functions
- Prefer properties over methods
- Use the strictest relevant class

Overprotected

(Use the EOLIAN tag)

```
/* BAD */
static void
_bar_eo_base_constructor(Eo *obj, Bar_Private_Data *pd)
{
    /* ... */
}

/* GOOD */
EOLIAN static void
_bar_eo_base_constructor(Eo *obj, Bar_Private_Data *pd)
{
    /* ... */
}
```

Overprotected

(Document Properties Not Functions)

```
class Evas.Object (...)
{
   methods {
      @property visibility {
         [[Document the property here]]
         set {
            [[Only put extra "set" only docs here]]
         }
         get {
            [[Only put extra "get" only docs here]]
         }
      }
   }
}
```

Overprotected

(Prefer Properties Over Methods)

```
class Evas.Object (...)
{
   methods {
      /* BAD */
      visibility_get {
      }

      /* GOOD */
      @property visibility {
         get {
         }
      }
   }
}
```

Overprotected

(Use the Strictest Relevant Class)

```
class Elm.Object (...)
{
   methods {
      /* BAD */
      top_win_find {
          return: Eo.Base *;
      }

      /* GOOD */
      top_win_find {
          return: Elm.Win *;
      }
   }
}
```

Overprotected

(Use the Strictest Relevant Class - Composite)

```
// We allow composite objects that implement Efl.Text
class Elm.Selector (..., Efl.Text)
{
    // ...
}
```

```
parent = eo_add(ELM_SELECTOR_CLASS, win);
comp = eo_add(ELM_BUTTON_CLASS, parent);
eo_composite_attach(parent, comp);
```

(You Drive Me) Crazy

(Anti-Patterns, Pitfalls and Limitations)

- No "super" skips!
- No need for `.Base` anymore
- Referencing private classes from public ones
- Don't use `legacy_prefix: null;`
- Naming specifications
- No function pointers or `va_args`

(You Drive Me) Crazy

(No "super" Skips!)

```
/* bar inherits from foo inherits from Eo.Base */
/* This function is an implementation in Bar. */
EOLIAN static void
_bar_eo_base_constructor(Eo *obj, Bar_Private_Data *pd)
{
    /* BAD - skipping a class in the MRO */
    eo_constructor(eo_super(obj, FOO_CLASS));

    /* GOOD */
    eo_constructor(eo_super(obj, BAR_CLASS));

    /* BEST */
    /* Define MY_CLASS: #define MY_CLASS BAR_CLASS */
    eo_constructor(eo_super(obj, MY_CLASS));
}
```

(You Drive Me) Crazy

(No More `.Base`)

```
/* Before */
class Eo.Base { ... }
class Eo.Class { ... }
class Eo.Class_Enum { ... }

/* Now */
class Eo { ... }
class Eo.Class { ... }
class Eo.Class.Enum { ... }
```

(You Drive Me) Crazy

(Referencing private classes from public ones)

```
class Efl.Canvas.Image (Evas.Image, ...)
{
    // ...
}
```

- Efl.Canvas.Image is the new "image"
- Evas.Image is the old "image"
- Evas.Image should not be installed

(You Drive Me) Crazy

(Don't use `legacy_prefix: null;`)

```
class Efl.Canvas.Image (...)
{
    legacy_prefix: null;
    // ...
}
```

The `legacy_prefix: null;` behaviour is now the default, and `null` is no longer a special value.

(You Drive Me) Crazy

(Naming Specifications)

- Legal characters are: `[A-Za-z0-9_]`
- Legal initial characters: `[A-Za-z_]`
- All keywords are allowed (e.g. class and object)
- Naming clashes for languages is handled by generators

Till The World Ends

(Future Ideas and Open Issues)

- "Visual parent" - fixing parenting
- Iron out the details of proxy objects
- The `@own` annotation is still untested
- Decide about `eo_halt()` and `eo_unref()` changes
- Should we rename all of `eo` to `efl`?

Till The World Ends

("Visual Parent" - Fixing Parenting)

```
obj = eo_add(win);
// eo_parent_get(obj) == win

/* Currently */
elm_swallow(layout, obj); // Uses an edje object internally
// eo_parent_get(obj) == layout

/* What I want */
elm_swallow(layout, obj) // Uses an edje object internally
// eo_parent_get(obj) == edje - No warranties (internal)
```

I'm a Slave 4 U

(Useful Links)

- My Website (Shameless Plug)
- Eo Wiki (Needs Work)
- Eolian Wiki
- EFL Mailing List

Questions?

stosb.com/talks

Tom Hacohen
Samsung Open Source Group

tom@osg.samsung.com
@TomHacohen