Promises and EFL Data Model

Felipe Magno de Almeida

EXPERTISE. SOLUTIONS

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

EXPERTISE. SOLUTIONS

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

What are promises

- Promises are a placeholder for a value that is going to be complete later in time
- Also work as a synchronization point
- Eolian convenience
- Examples

EXPERTISE. SOLUTIONS

What are promises

- Promises are a placeholder for a value that is going to be complete later in time
- Also work as a synchronization point
- Eolian convenience
- Examples

What are promises

- Promises are a placeholder for a value that is going to be complete later in time
- Also work as a synchronization point
- Eolian convenience
- Examples

Eolian convenience

```
methods {
  foo {
    params {
      promise: promise<int>;
    }
  }
}
```

Eolian convenience

```
void foo(Eina_Promise_Owner* promise)
{
   int i = 5;
   eina_promisee_owner_value_set(promise,
        &i, NULL);
}
```

Eolian convenience

```
void then_cb(void* data, void* value)
{
  int i = *(int*)value;
  printf("value %d", i);
}
...
Eina_Promise* promise;
foo(&promise);

eina_promise_then(promise, &then_cb,
     &fail_cb);
```

Can we do this?

```
void then_cb(void* data, int value)
{
  printf("value %d", value);
}
...
Eina_Promise* promise;
foo(&promise);

eina_promise_then(promise, &then_cb,
     &fail_cb);
```

What are promises

- Promises are a placeholder for a value that is going to be complete later in time
- Also work as a synchronization point
- Eolian convenience
- Examples

What are promises

- Promises are a placeholder for a value that is going to be complete later in time
- Also work as a synchronization point
- Eolian convenience
- Examples

Examples

```
void then_cb(void* data, void* value)
{
  int i = *(int*)value;
  printf("value %d", i);
}
...
Eina_Promise* promise;
foo(&promise);

eina_promise_then(promise, &then_cb,
     &fail_cb);
```

Examples

```
void fail_cb(void* data,
   Eina_Error const* error)
{
  ...
}
...
Eina_Promise* promise;
foo(&promise);

eina_promise_then(promise, &then_cb,
     &fail_cb);
```

EXPERTISE. SOLUTIONS

Examples

```
Eina_Promise* promise;
foo(&promise);

void* v = eina_promise_value_get(promise);
```

Examples with composition

```
Eina_Promise* promises[] = {NULL, NULL, NULL};
foo(&promise[0]);
foo(&promise[1]);

Eina_Promise result = eina_promise_all
 (eina_carray_iterator_new(promises));
```

Examples with composition

```
Eina_Promise* promises[] = {NULL, NULL, NULL};
foo(&promise[0]);
foo(&promise[1]);

Eina_Promise result = eina_promise_race
 (eina_carray_iterator_new(promises));
```

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

How does that relate to Mainloop

It does not

How does that relate to Mainloop

```
// When the asynchronous function calls:
eina_promise_value_set(owner, &value,
  &free_cb);

// Then eina_promise_value_set calls all
// then_cb's directly
```

How does that relate to Mainloop

- What about threading?
  - What threading?
  - Ecore_Threading?

How does that relate to Mainloop

- What about threading?
  - What threading?
  - Ecore_Threading?

How does that relate to Mainloop

- What about threading?
  - What threading?
  - Ecore_Threading?

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

EXPERTISE. SOLUTIONS

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

Eina Promises inheritance

```
struct _Eina_Promise
{
   int version;
   Eina_Promise_Then_Cb then;
   Eina_Promise_Value_Get_Cb value_get;
   Eina_Promise_Error_Get_Cb error_get;
   Eina_Promise_Pending_Is_Cb pending_is;
   ...
```

Ecore Thread Promise

```
/*
 * @brief Function that instantiates a
 * Ecore_Promise and automatically
 * executes func_blocking callback
 * function in another thread
 */
EAPI Ecore_Thread* ecore_thread_promise_run
  (Ecore_Thread_Promise_Cb func_heavy,
   Ecore_Thread_Promise_Cb func_cancel,
   const void* data, size_t value_size,
   Eina_Promise** promise);
```

Ecore Thread Promise

```
void thread_cb(const void* data,
   Eina_Promise_Owner* promise,
   Ecore_Thread* thread)
{
   eina_promise_owner_error_set
    (promise, EINA_ERROR_OUT_OF_MEMORY);
}

Eina_Promise* promise;
ecore_thread_promise_run
 (&promise_error_thread, NULL, NULL,
   0, &promise);
eina_promise_then(promise, NULL,
   &promise_error_callback, NULL);
```

EXPERTISE. SOLUTIONS

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

Overview

- What are promises
- How does that relate to Mainloop
- How does that relate to Ecore Thread
- EFL Data Model

EXPERTISE. SOLUTIONS

EFL Data Model

- EFL Data Model is completely asynchronous
- Updated with promises
- Both implementation and use have become **way simpler**

EFL Data Model

- EFL Data Model is completely asynchronous
- Updated with promises
- Both implementation and use have become **way simpler**

EXPERTISE. SOLUTIONS

EFL Data Model

- EFL Data Model is completely asynchronous
- Updated with promises
- Both implementation and use have become **way simpler**

EFL Data Model with promises

```
Eina_Promise *prms[4];
prms[3] = NULL;
efl_model_property_get(m, "filename", &prms[0]);
efl_model_property_get(m, "size", &prms[1]);
efl_model_property_get(m, "mtime", &prms[2]);

eina_promise_then
 (eina_promise_all
  (eina_carray_iterator_new(prms)),
   &then_cb, &fail_cb, NULL);
```

EFL Data Model with promises

```
void then_cb(void* data, void* value)
{
   Eina_Iterator** iterator = (void**)value;
   ...
}
```

EFL Data Model with promises

- For Eio implementation model implementation we just queue requisitions and do a stat asynchronous operation
- When the stat operation is finished, we walk the queue and eina_promise_owner_value_set each one.
- We don't have to cache any result from the stat operation
- Instead we cache requisitions that are pending
- If they are all made before the stat operation, then just one stat operation is needed

EFL Data Model with promises

- For Eio implementation model implementation we just queue requisitions and do a stat asynchronous operation
- When the stat operation is finished, we walk the queue and eina_promise_owner_value_set each one.
- We don't have to cache any result from the stat operation
- Instead we cache requisitions that are pending
- If they are all made before the stat operation, then just one stat operation is needed

EFL Data Model with promises

- For Eio implementation model implementation we just queue requisitions and do a stat asynchronous operation
- When the stat operation is finished, we walk the queue and eina_promise_owner_value_set each one.
- We don't have to cache any result from the stat operation
- Instead we cache requisitions that are pending
- If they are all made before the stat operation, then just one stat operation is needed

EFL Data Model with promises

- For Eio implementation model implementation we just queue requisitions and do a stat asynchronous operation
- When the stat operation is finished, we walk the queue and eina_promise_owner_value_set each one.
- We don't have to cache any result from the stat operation
- Instead we cache requisitions that are pending
- If they are all made before the stat operation, then just one stat operation is needed

EFL Data Model with promises

- For Eio implementation model implementation we just queue requisitions and do a stat asynchronous operation
- When the stat operation is finished, we walk the queue and eina_promise_owner_value_set each one.
- We don't have to cache any result from the stat operation
- Instead we cache requisitions that are pending
- If they are all made before the stat operation, then just one stat operation is needed

Thank You

Q&A