Introduction

The Lightweight Component Model

- https://docs.google.com/presentation/d/1PSC3Q5oFsJEaYyV5lNJvVgh-SNxhySWUqZ6puyojMi8
- Module Linking, Interface types, Typed Main
- What's great about wasm, plus:
  - black-box reuse
  - external composition

Examples:

- Don't export all of linear memory
- Don't pass around slices with arbitrary lifetimes

The Canonical ABI proposal

https://github.com/WebAssembly/interface-types/pull/132

A specific ABI for Interface Types in terms of Core Wasm types and features

Lower `list`, `variant`, etc. into Core Wasm types and concepts.

Canonical ABI implications

The Canonical ABI gives us a path to:

- Adopt Interface Types without waiting for full adapter functions
  - Preserves many of the important properties
- A way to adopt interface types for Core Wasm modules as well
  - So we can share witx tooling

Application to Core-Wasm WASI

Handles and Resources

Presentation to WASI:

- https://docs.google.com/presentation/d/1ikwS2Ps-KLXFofuS5VAs6Bn14q4LBEaxMjPfLj61UZE

Handles in the Canonical ABI are i32 indices.

- Think "file descriptors"

The Canonical ABI handles the management of tables.

```
(resource $input_byte_stream_resource)
(typename $input_byte_stream (handle $input_byte_stream_resource))

(export "read" (func
  (param $source $input_byte_stream)
  (param $buffer (pull-buffer u8))
  (result $result (expected (tuple u64 $read_status) (error)))
))
```

Buffers

- `push-buffer`, `pull-buffer`
- Temporary views for memory buffers
- In the Canonical ABI, this is memory + offset + len

```
(resource $input_byte_stream_resource)
(typename $input_byte_stream (handle $input_byte_stream_resource))

(export "read" (func
  (param $source $input_byte_stream)
  (param $buffer (pull-buffer u8))
  (result $result (expected (tuple u64 $read_status) (error)))
))
```

Vectored I/O can be considered when we have full adapter functions.

Working with witx

Witx is evolving into an IDL.

One interface description covers:

- Canonical ABI for core Wasm
- Canonical ABI for components
- Full adapter-function ABI for components

And cf. last week's presentation:

- Blocking
- Non-blocking