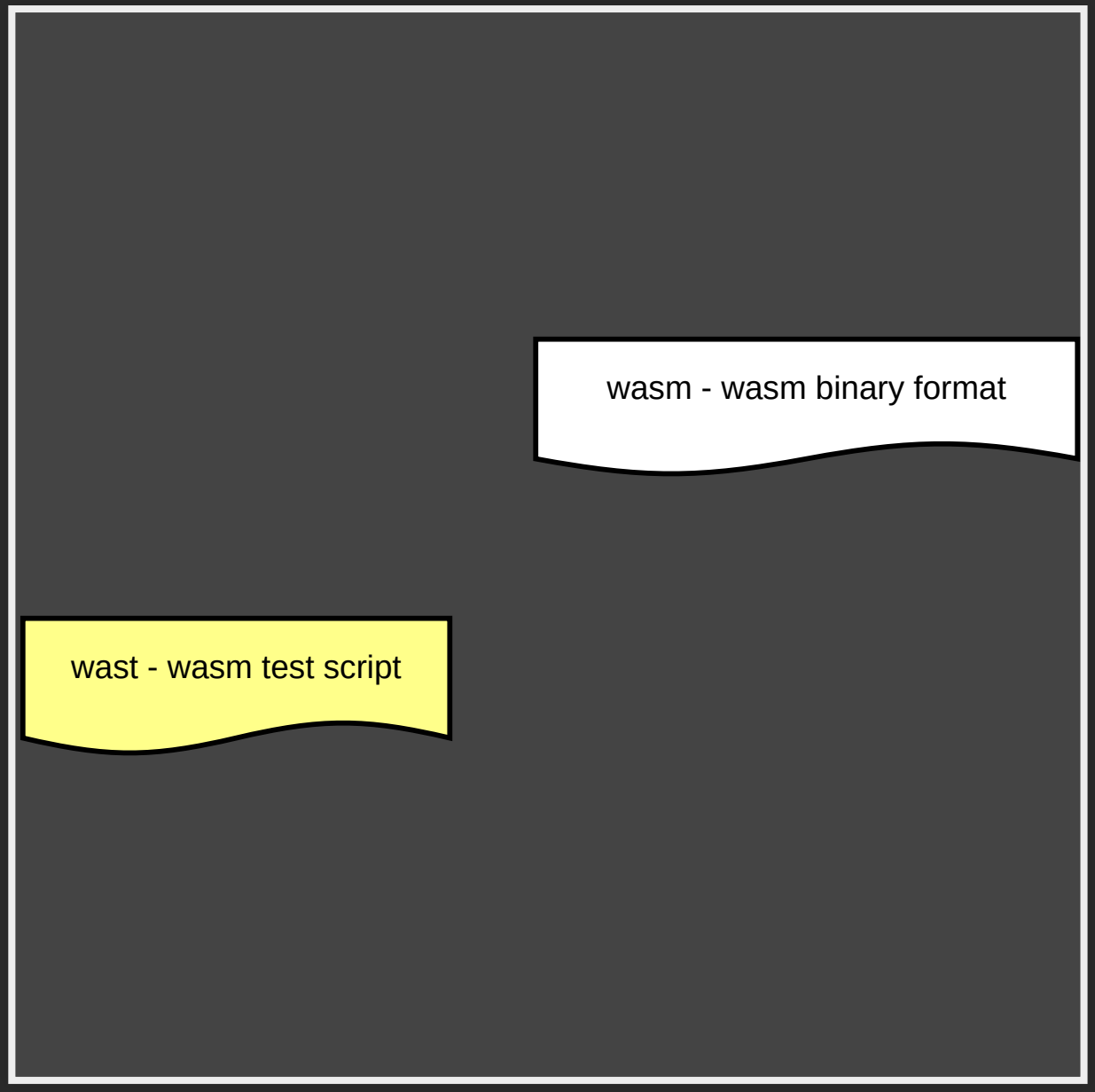
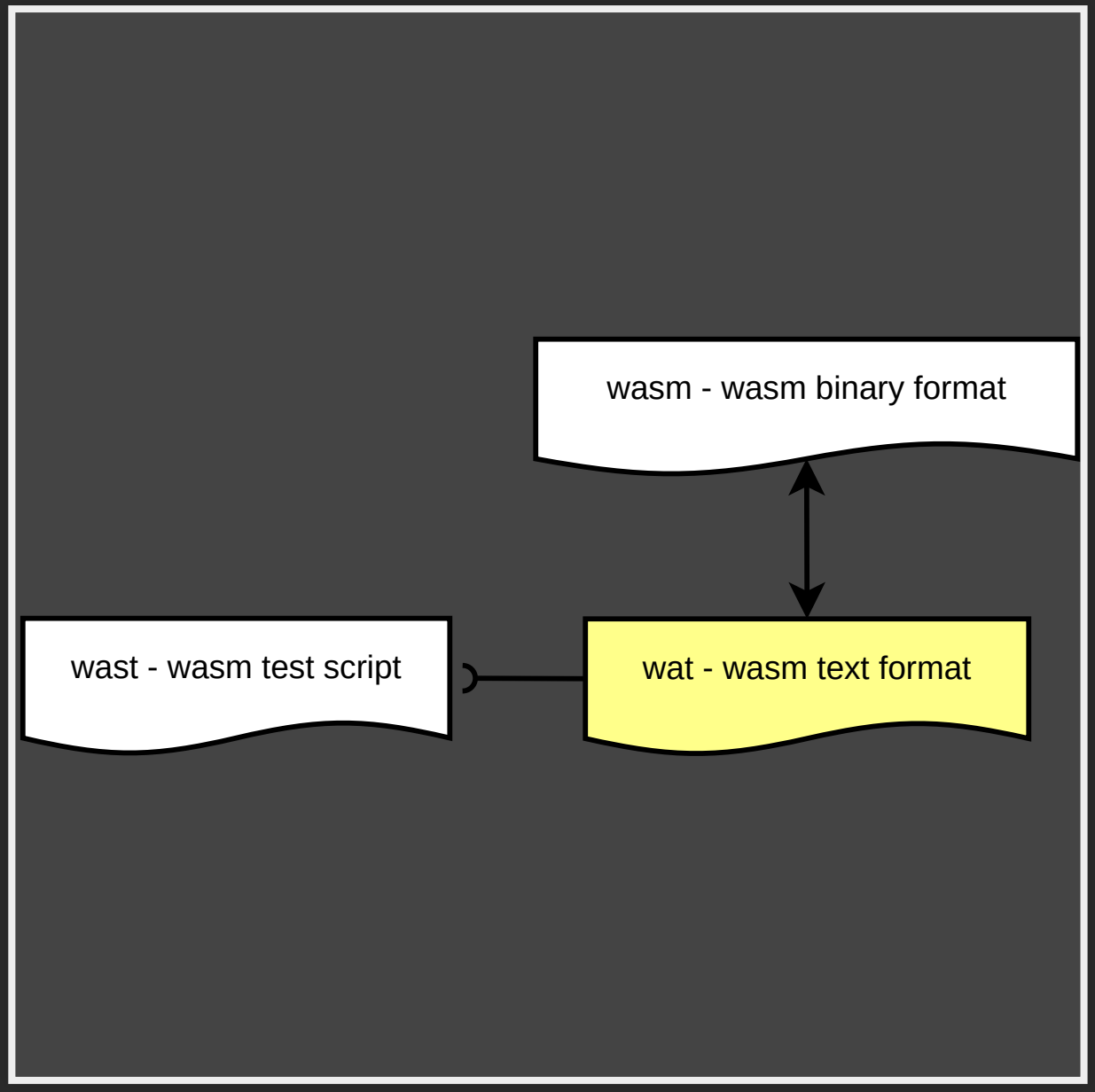
WASI SECURITY, AND CONNECTION TO OF WASM PROPOSALS

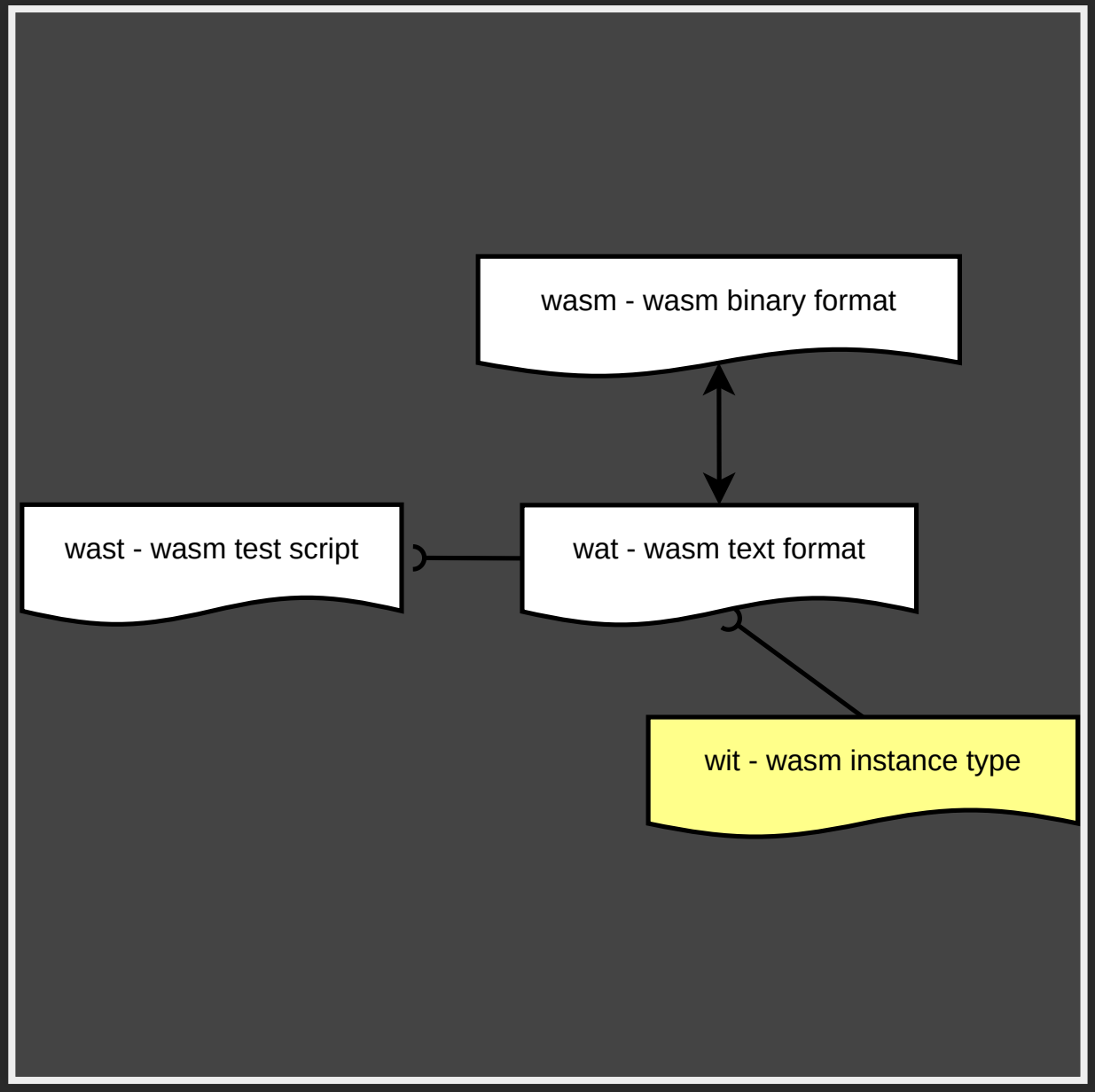
- witx
- Capability-based security
- Relationship to POSIX
- Reference types, interface types, instance types, type imports, and more!

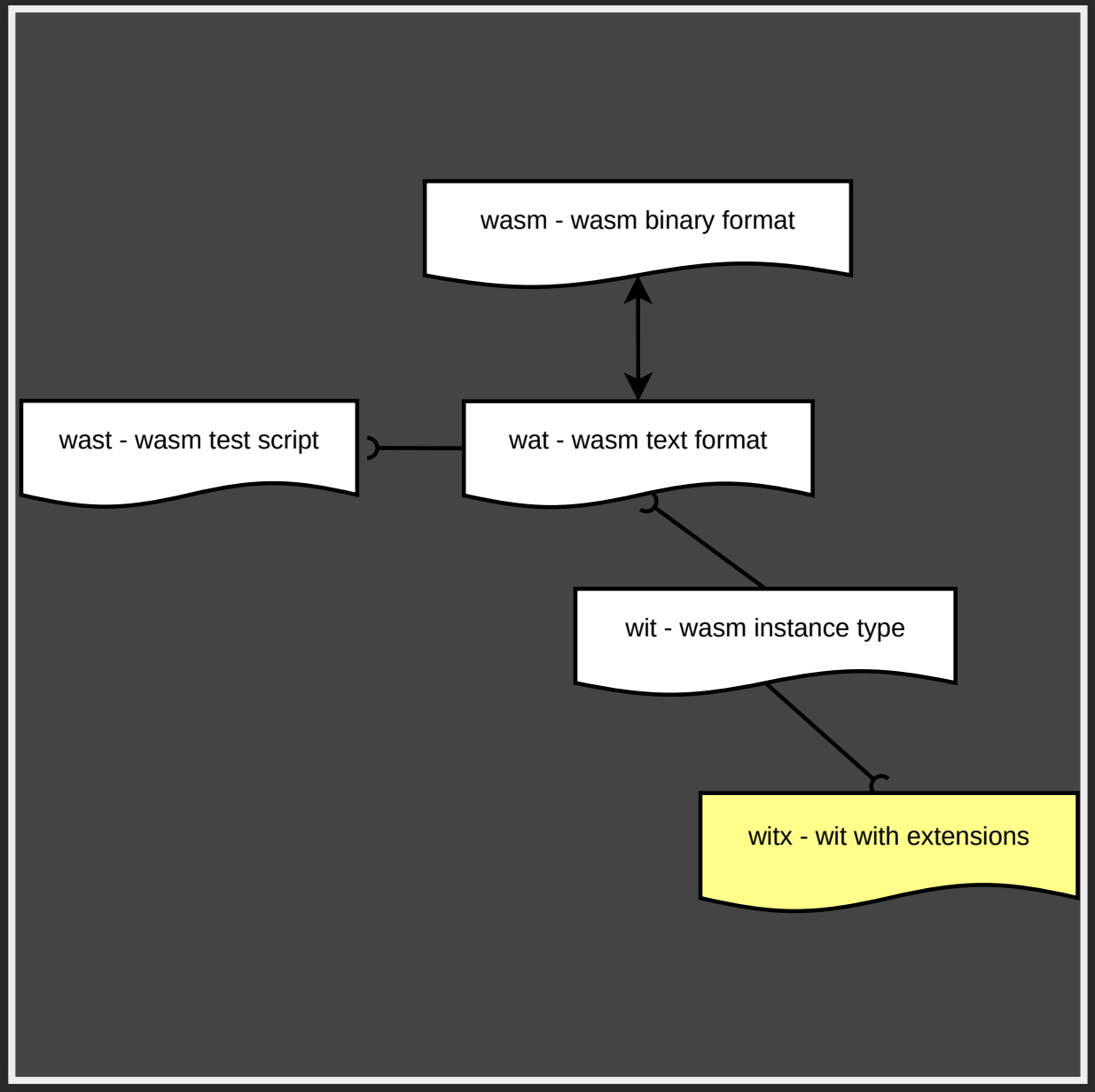
LIVE FROM THE FILE EXTENSION ZOO...

wasm - wasm binary format


wasm - wasm binary format
wast - wasm test script


wasm - wasm binary format
wast - wasm test script
wat - wasm text format


wasm - wasm binary format
wast - wasm test script
wat - wasm text format
wit - wasm instance type


wasm - wasm binary format
wast - wasm test script
wat - wasm text format
wit - wasm instance type
witx - wit with extensions

WITX - WIT WITH EXTENSIONS

- evolving text format
- reference parser and tools in the WASI repo
- anticipate and align with wasm proposals
- add things useful for humans

WITX - WIT WITH EXTENSIONS

Anticipated proposals:

- module-types: `.wit`
- annotations: `(@witx ...)`
- interface-types
  - `(@interface ...)`
  - `(param $path string)`
- gc: `(struct $foo ...)`
- multi-value
  - `(result $error $errno)`
  - `(result $nwritten $size)`

WITX - WIT WITH EXTENSIONS

Looking forward:

- reference-types
- type-imports-proposal
- exception-handling

WITX - WIT WITH EXTENSIONS

Plus a few goodies

```
(use "typenames.witx")
```

```
;;; Return the current offset of a file descriptor.
;;; Note: This is similar to `lseek(fd, 0, SEEK_CUR)` in POSIX.
(@interface func (export "tell")
  (param $fd $fd)
  (result $error $errno)
  ;;; The current offset of the file descriptor, relative to the start of the file.
  (result $offset $filesize)
)
```

```
;;; A file descriptor handle.
(typename $fd (handle))
```

CAPABILITY-BASED SECURITY

Extend the core wasm sandbox into the API space

IMPLICIT VS. EXPLICIT SANDBOXING (CORE)

Host pointers + access control

vs.

Linear Memory

IMPLICIT VS. EXPLICIT SANDBOXING (WASI)

Host filesystem paths + custom namespaces

vs.

All filesystem paths must be relative to a handle

STATIC VS DYNAMIC TYPE CHECKING (CORE)

Dynamically-checked serialized messages

vs.

Statically checked signatures

STATIC VS DYNAMIC TYPE CHECKING (WASI)

Filesystem-based APIs (eg. `/proc`, `/dev`, etc.)

vs.

Function-based APIs.

POSIX

Wasm doesn't have `fork`, signals, full `mmap`, etc.

Users and groups aren't helpful to applications.

Async.

Non-Web Wasm is important, and should have APIs that work for Wasm.

PIPELINES

This paper itself was typeset camera-ready using flo, pic, ditroff, and other ditroff preprocessors with sequence of commands essentially equivalent to the command line,

```
refer paper | flo | pic | tbl | eqn | ditroff -mXP
```

"EVERYTHING IS A FILE"

EVERYTHING IS A NAME IN A GLOBAL NAMESPACE?

EVERYTHING IS A FILE(?) DESCRIPTOR

EVERYTHING IS A HANDLE 

- handles as reference types
- handles as indices into isolated tables

WASI SECURITY, AND USE OF WASM PROPOSALS