@m_e_i_s

THIS WHOLE

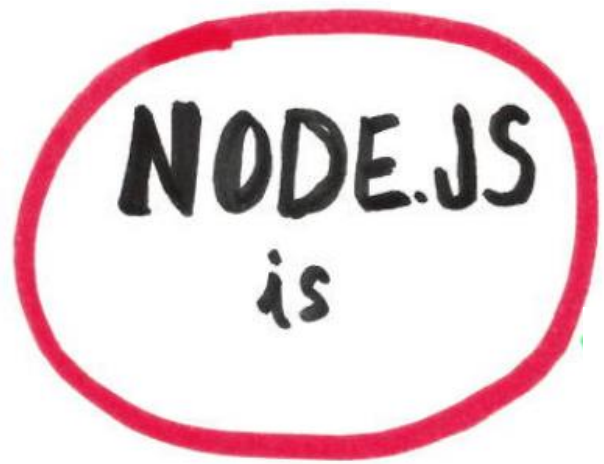
"JAVASCRIPT IN THE SERVER"

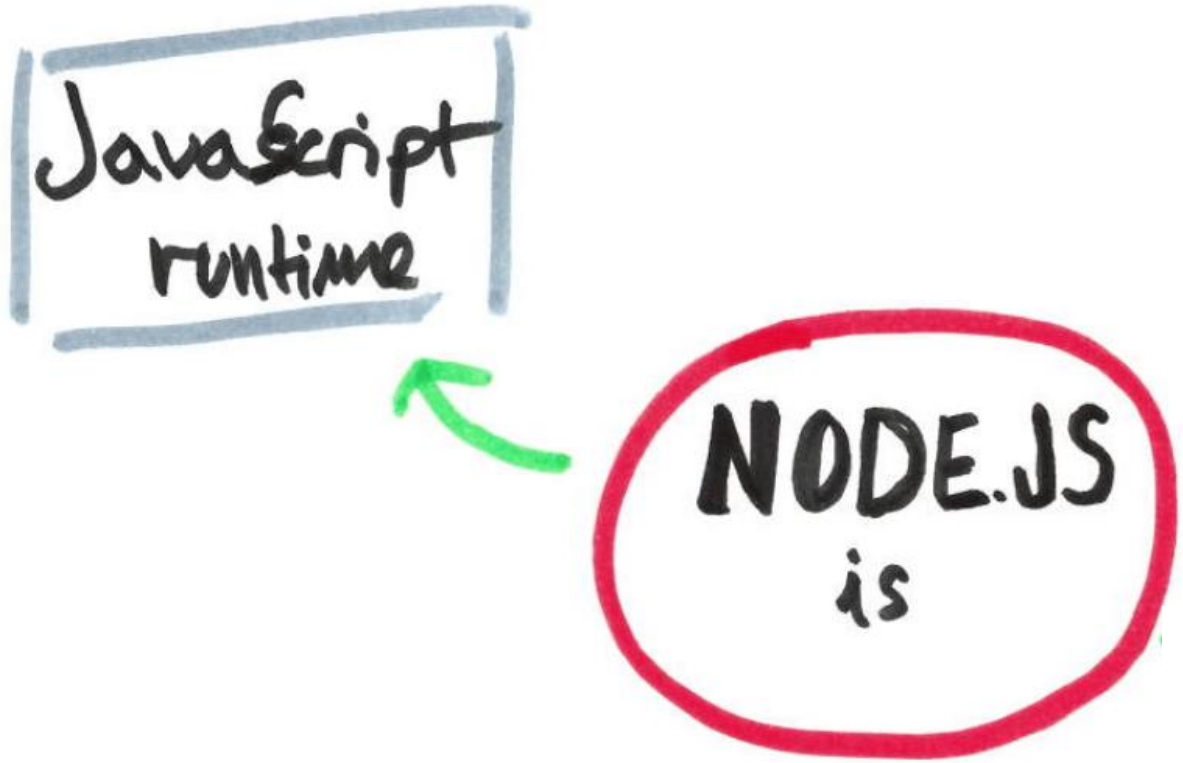
THING

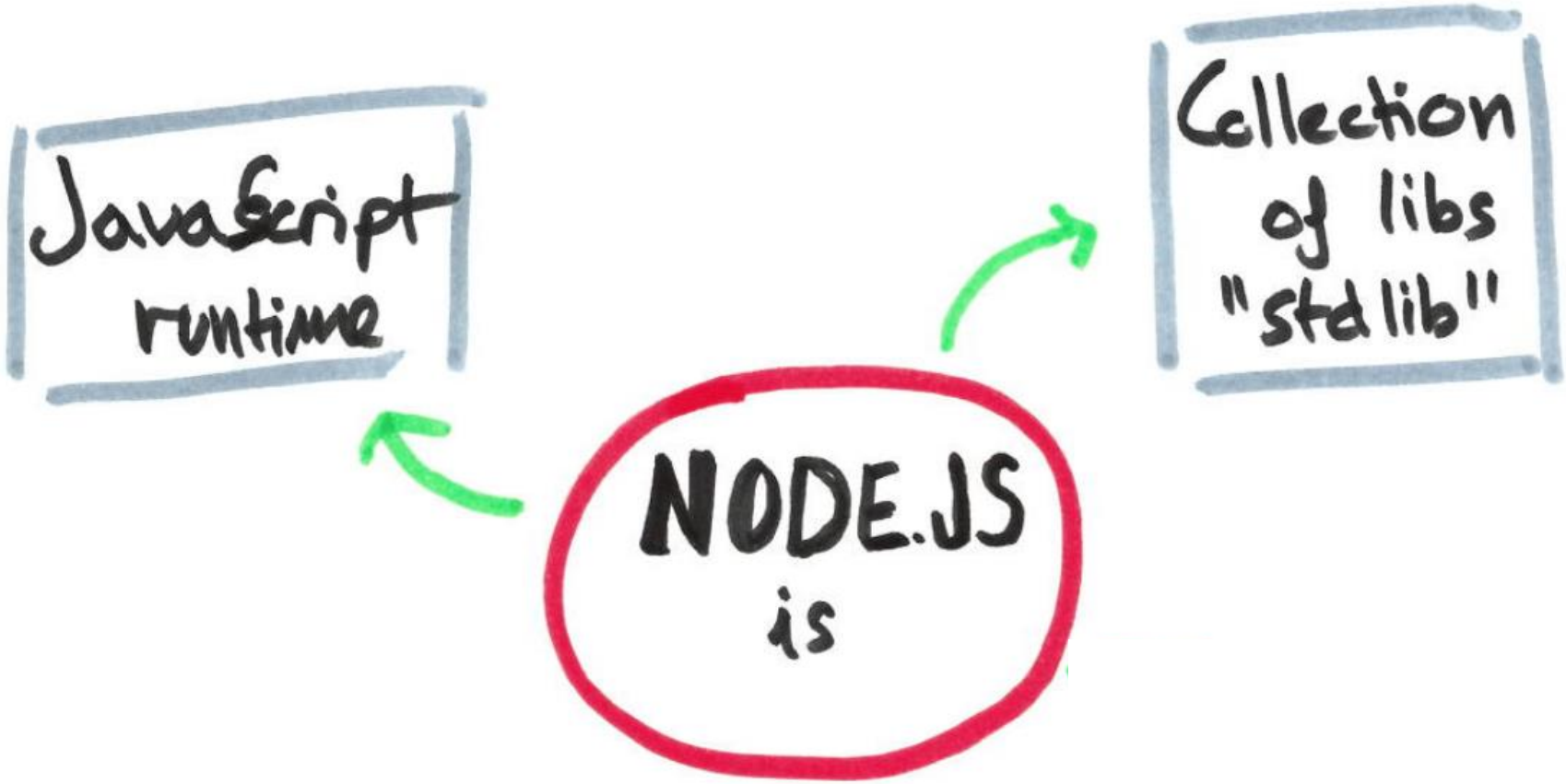
A. K. A.

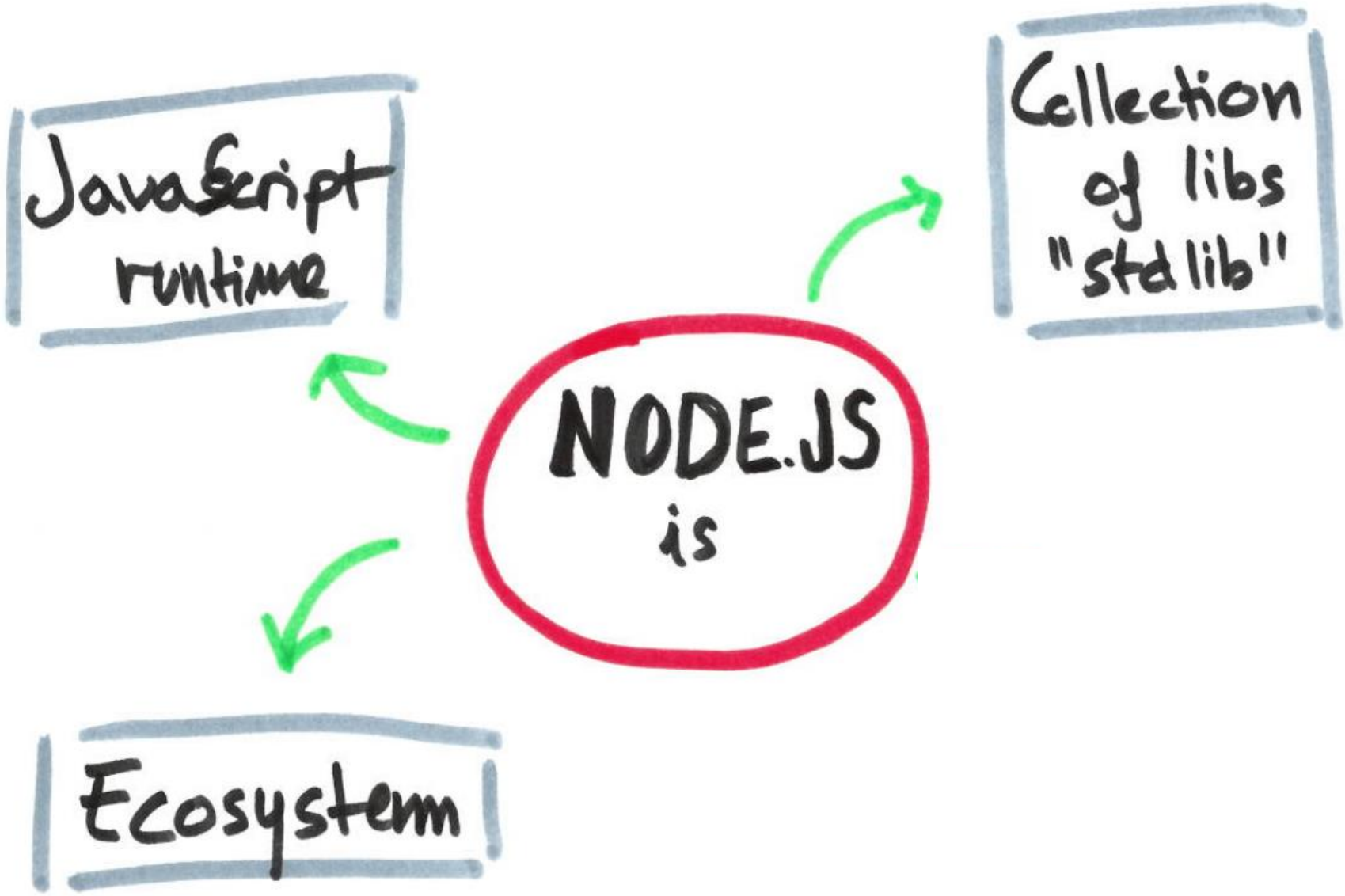
WTF IS NODE

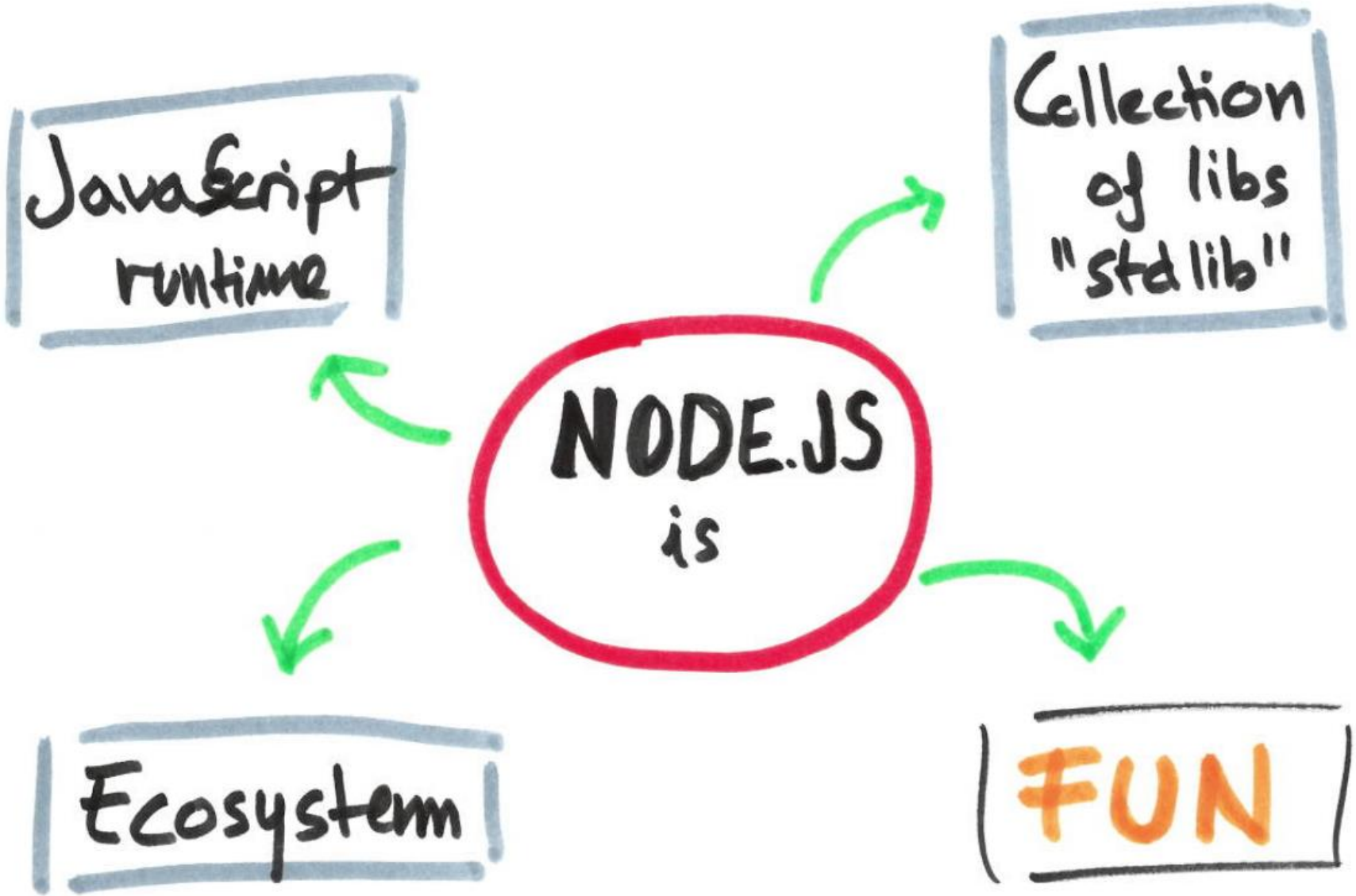

HELLO
Marc Egea i Sala
@m_e_i_s


NODE.JS
is


JavaScript
runtime
NODE.JS
is


JavaScript runtime
Collection of libs "std lib"
NODE.JS is


JavaScript runtime
Collection of libs "std lib"
NODE.JS is
Ecosystem


JavaScript runtime
Collection of libs "std lib"
NODE.JS is
Ecosystem
FUN

JavaScript runtime

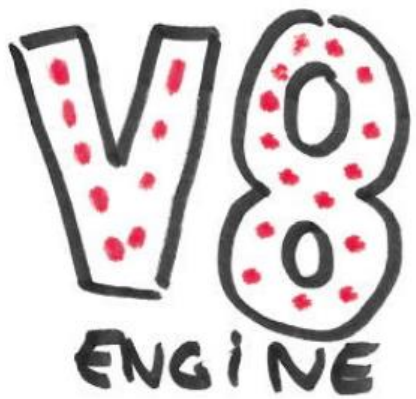
JavaScript runtime

```
> node my_program.js
```

JavaScript runtime

```
> node my_program.js
```

IT RUNS!!


V8
ENGINE

V8

ENGINE

WRITEN IN C++

FAST

JIT

COMPILER

BLACK MAGIC

INFERS TYPES

CACHES

V8 ENGINE

BY GOOGLE

USED IN BROWSERS (CHROME-ISH)

FREQUENTLY UPDATED

JiT COMPILER

WRITEN IN C++

FAST

BLACK MAGIC

INFERS TYPES

CACHES

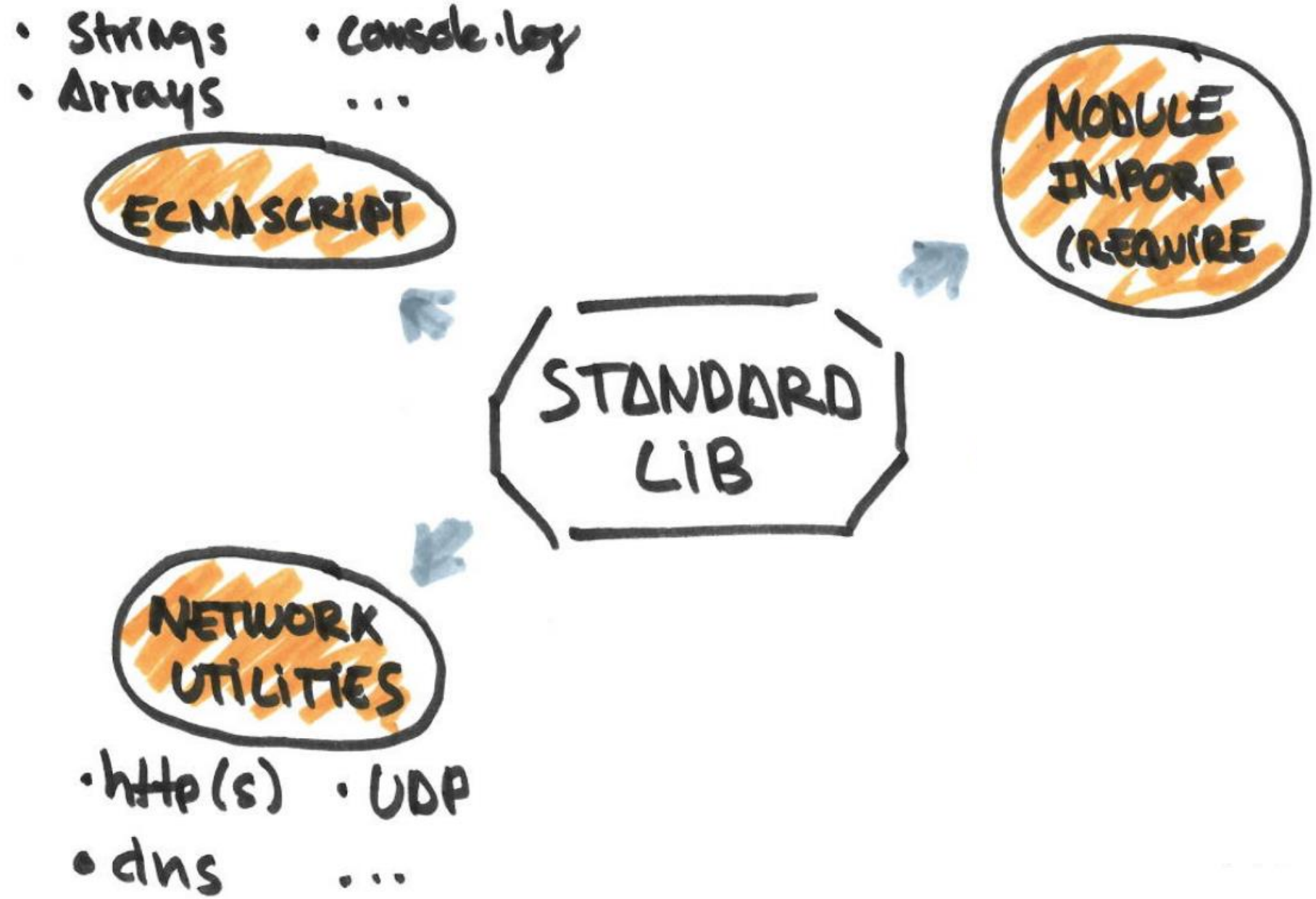

STANDARD
LIB

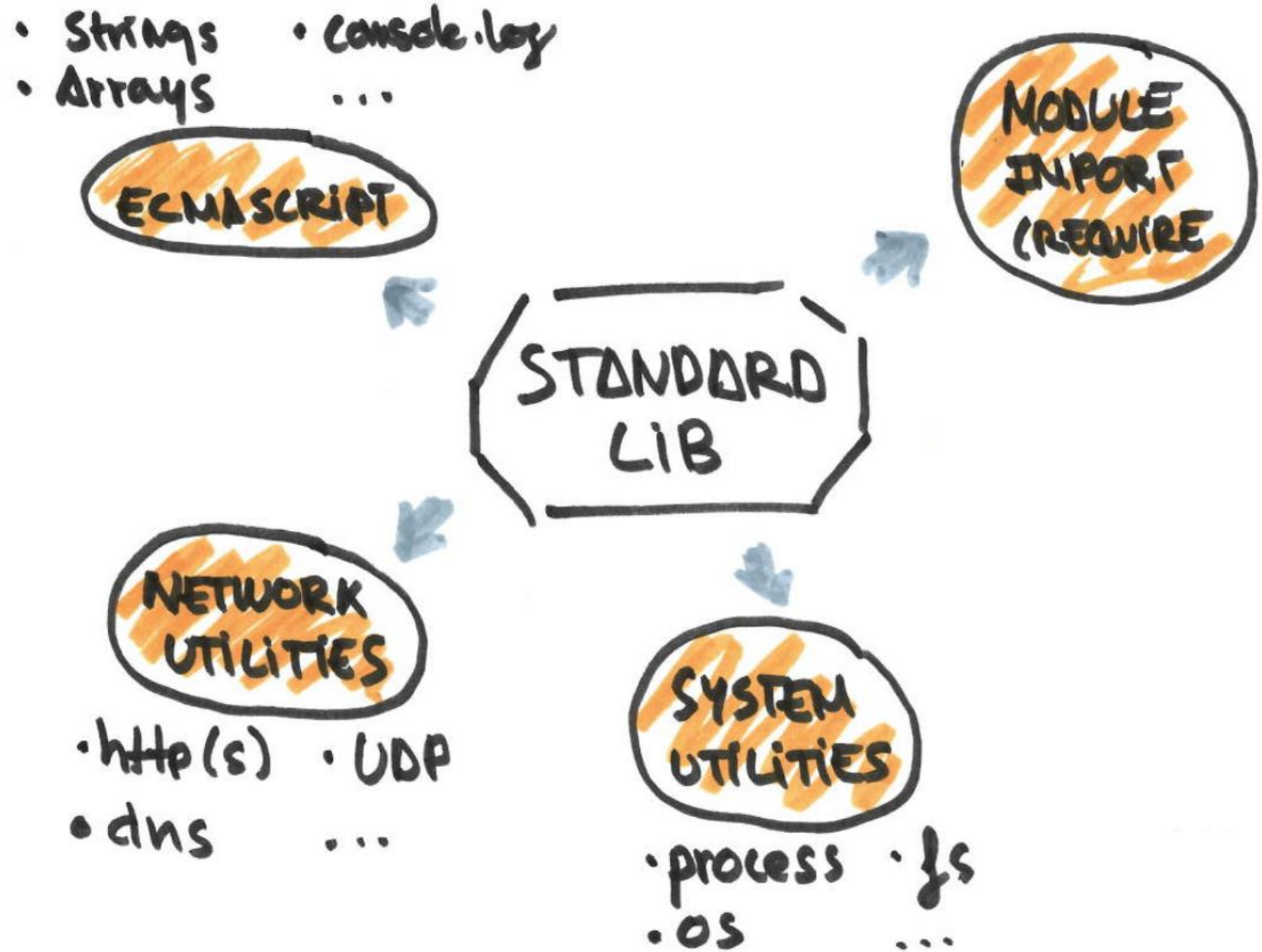
- Strings
- Arrays

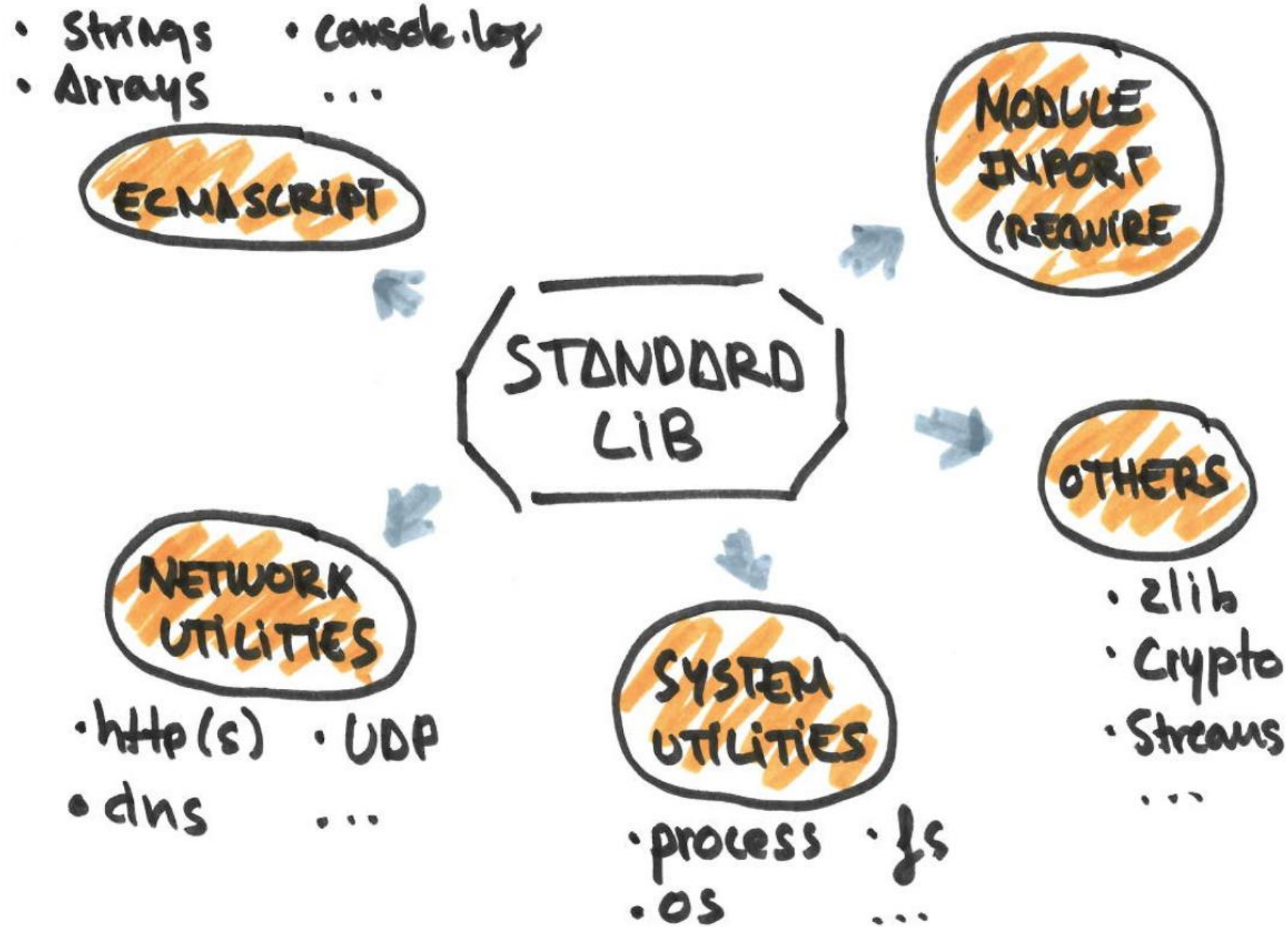
- console.log
- ...

- Strings
- Arrays

- console.log
- ...


• Strings
• Arrays
• console.log
...
ECMASCRIPT
MODULE
IMPORT
(REQUIRE
STANDARD
LIB
NETWORK
UTILITIES
• http(s)
• UDP
• dns
...

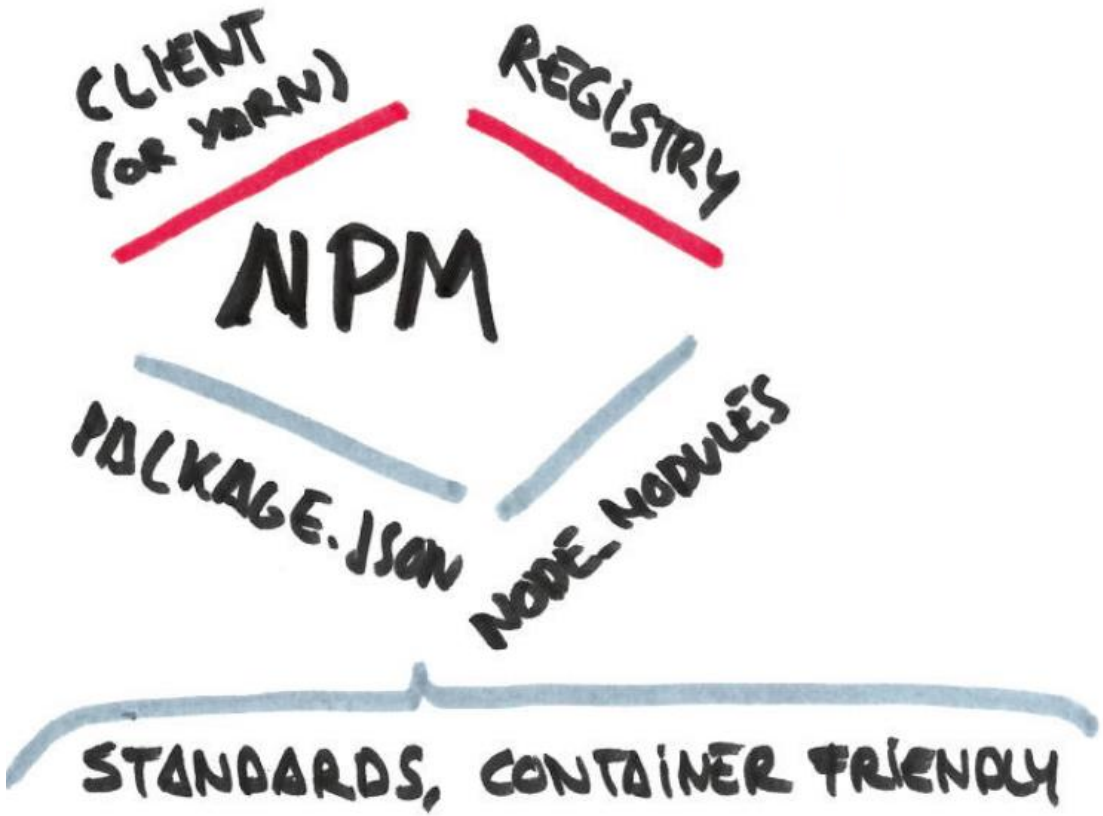

• Strings
• Arrays
• console.log
...
ECMASCRIPT
MODULE
IMPORT
(REQUIRE
STANDARD
LIB
NETWORK
UTILITIES
• http(s)
• dns
• UDP
...
SYSTEM
UTILITIES
• process
• os
• fs
...


• Strings
• Arrays
• console.log
...
ECMASCRIPT
MODULE
IMPORT
(REQUIRE
STANDARD
LIB
NETWORK
UTILITIES
• http(s)
• dns
• UDP
...
SYSTEM
UTILITIES
• process
• os
• fs
...
OTHERS
• zlib
• Crypto
• Streams
...

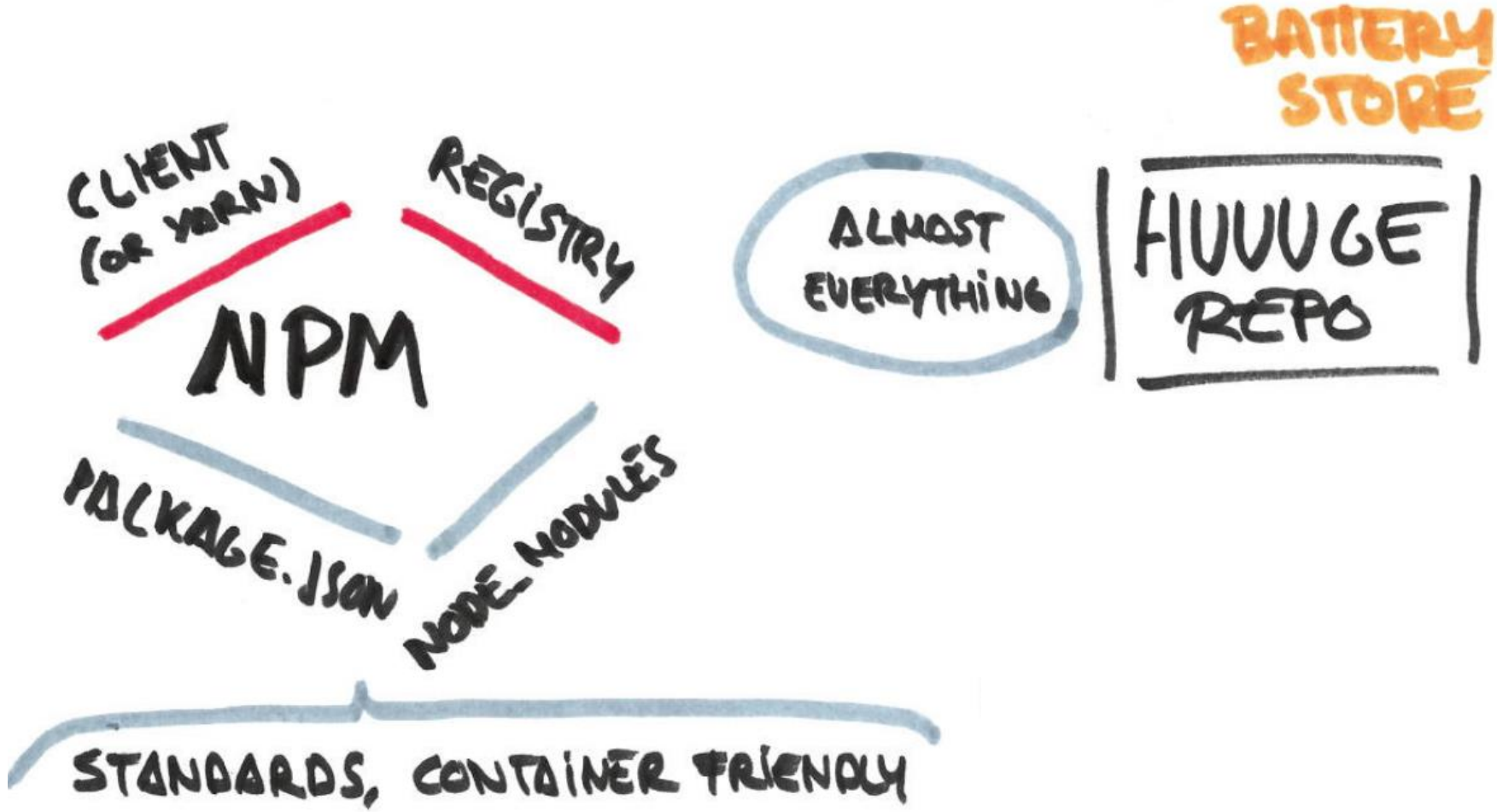
THE ECOSYSTEM

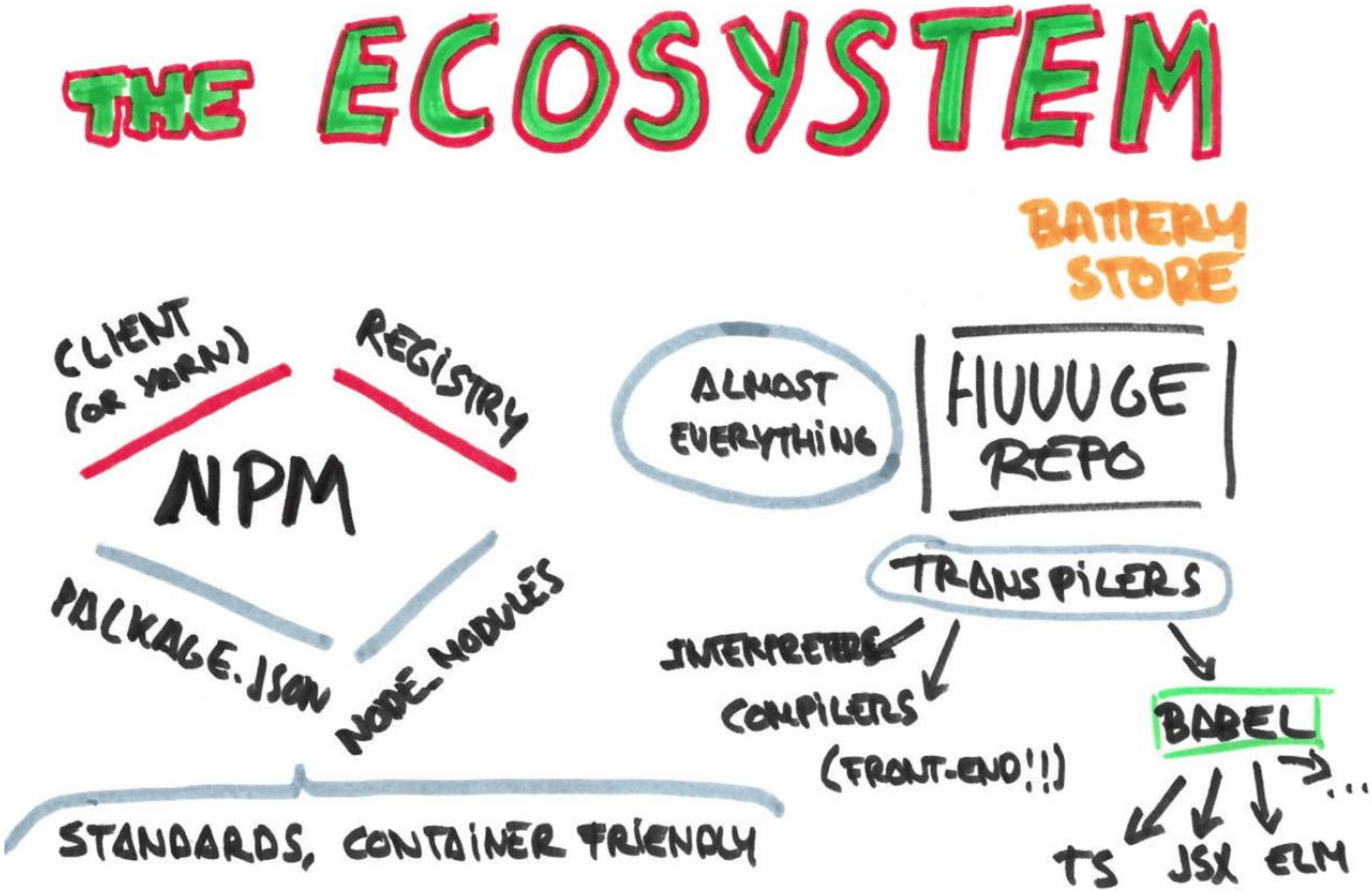
THE ECOSYSTEM

THE ECOSYSTEM



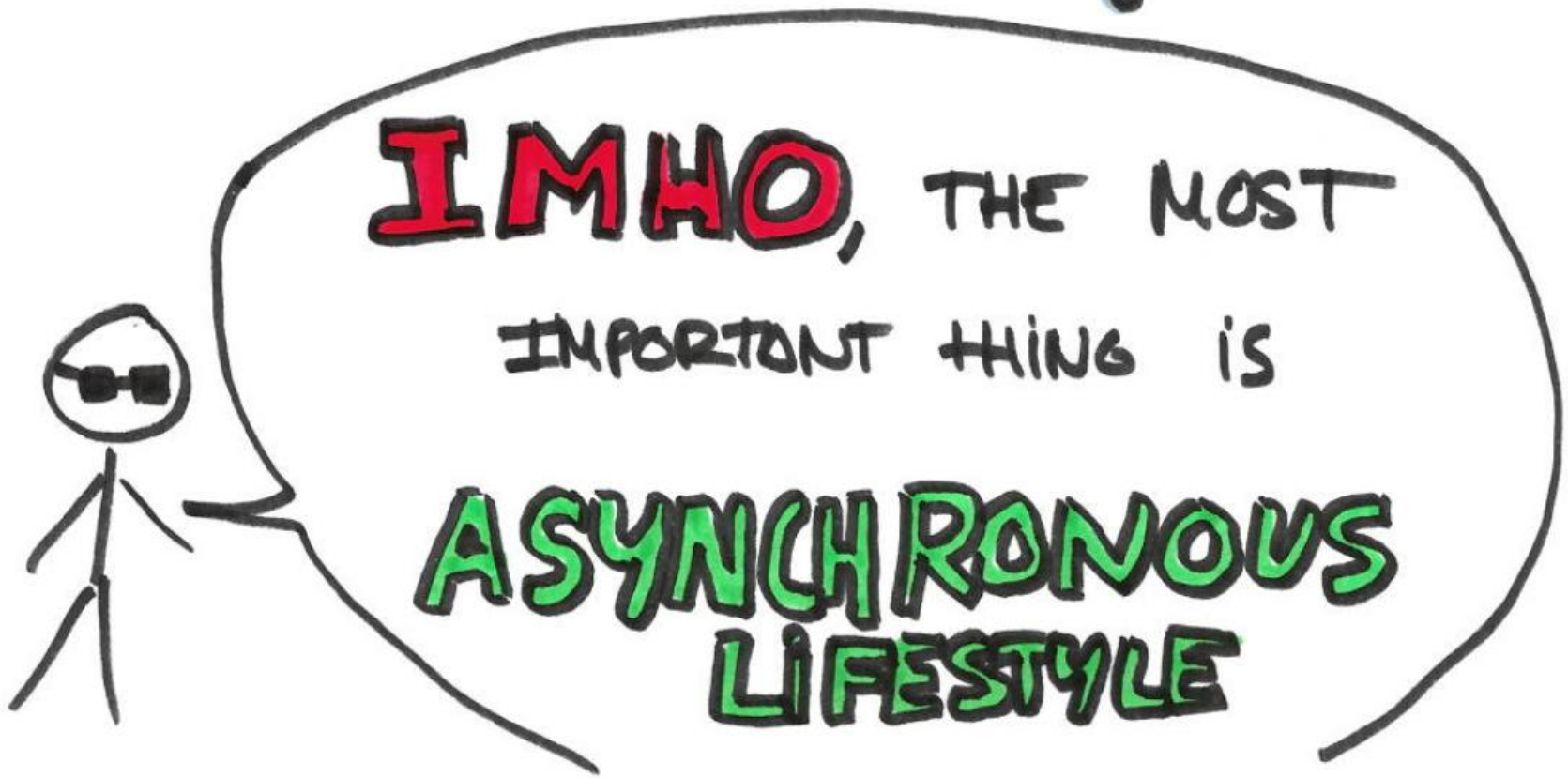
THE ECOSYSTEM




THE ECOSYSTEM
BATTERY STORE
CLIENT (OR YARN)
REGISTRY
NPM
PACKAGE.JSON
NODE_MODULES
STANDARDS, CONTAINER FRIENDLY
ALMOST EVERYTHING
HUUUGE REPO
TRANSPILERS
INTERPRETERS
COMPILERS
(FRONT-END!!)
BABEL
...
TS
JSX
ELM


AND THAT'S ALL?

AND THAT'S ALL?



DO THIS

CALL ME BACK WHEN

IT'S DONE

NODE USES AN EVENT LOOP

SO,

LET'S BUILD

AN

EVENT LOOP !!!

NODE USES AN EVENT LOOP

SO,

LET'S BUILD

A

PSEUDO EVENT LOOP !!!

NODE USES AN EVENT LOOP

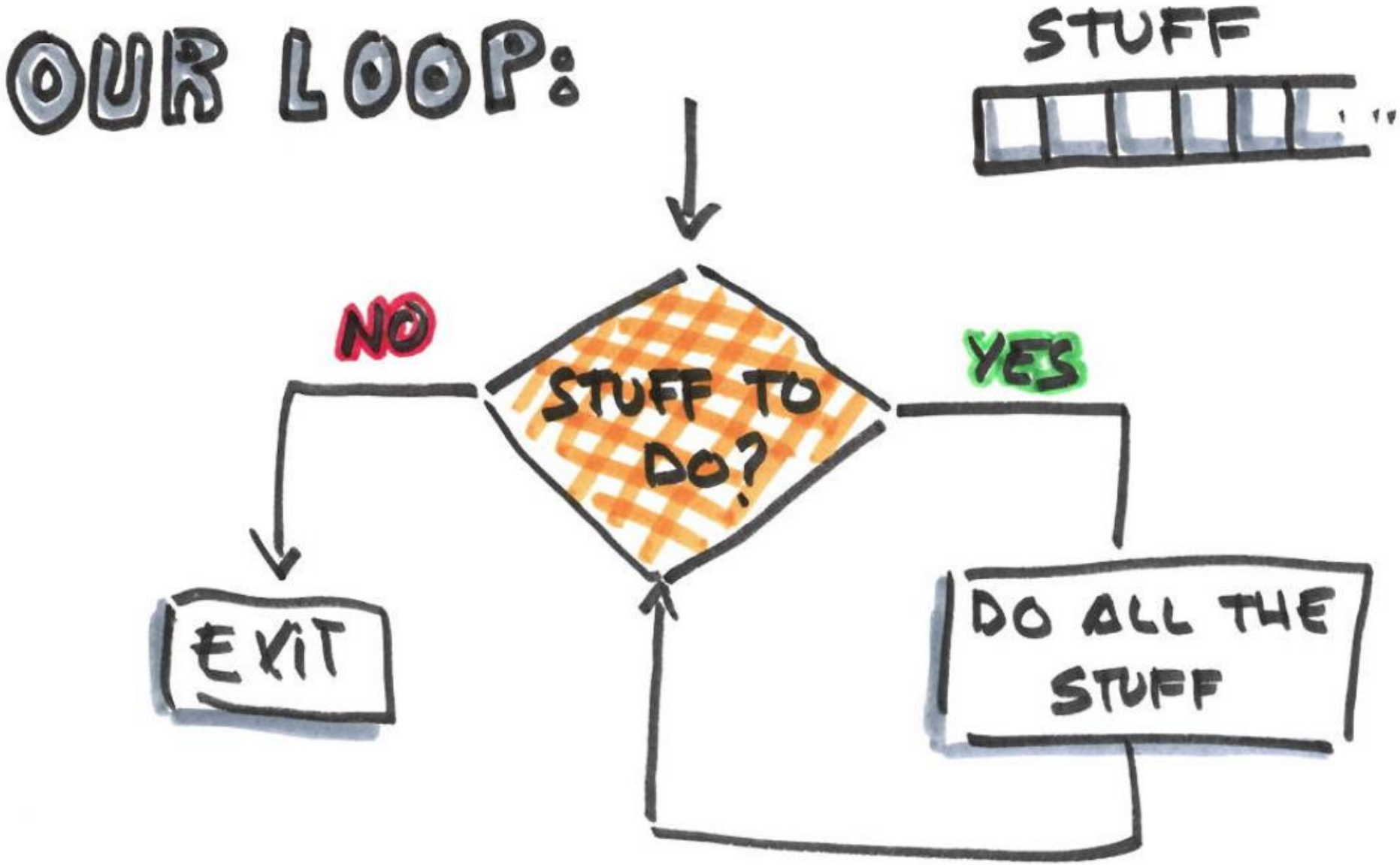
SO,

LET'S BUILD

A

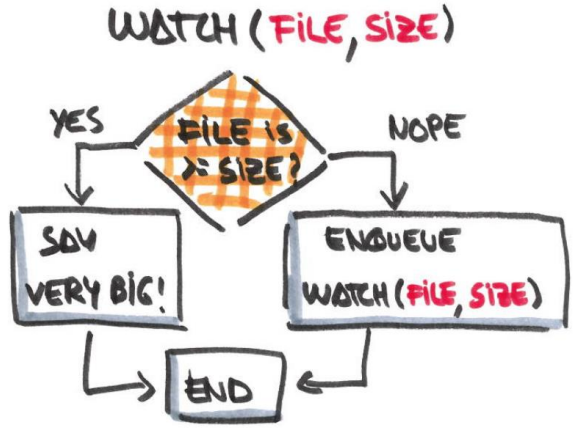
LOOP !!!

OUR LOOP:

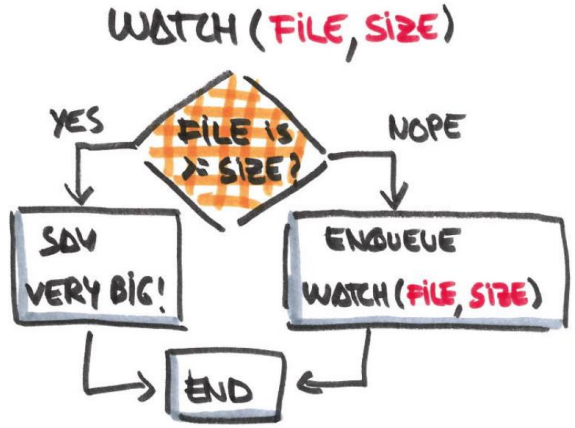

OUR LOOP:
STUFF
NO
STUFF TO DO?
YES
EXIT
DO ALL THE STUFF

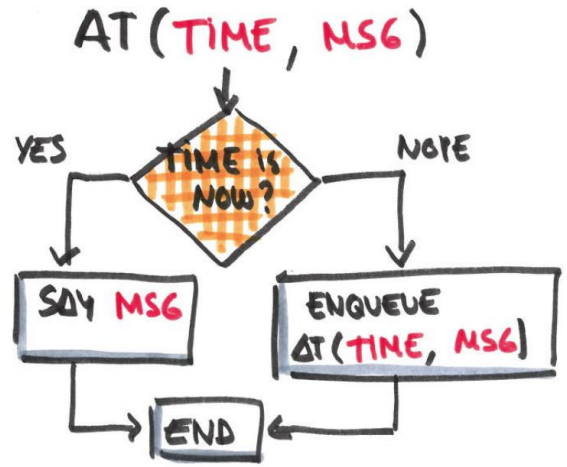
ASYNC FUNCTIONS

ASYNC FUNCTIONS

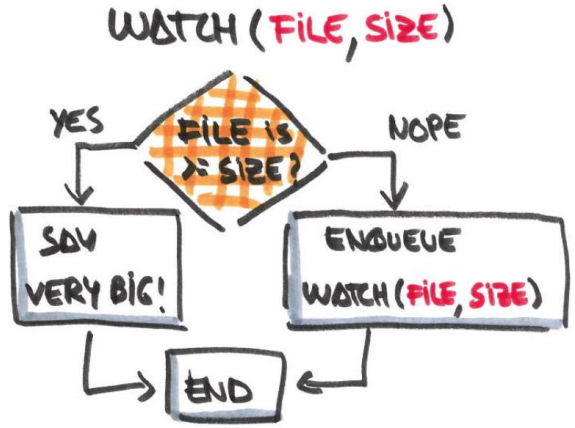


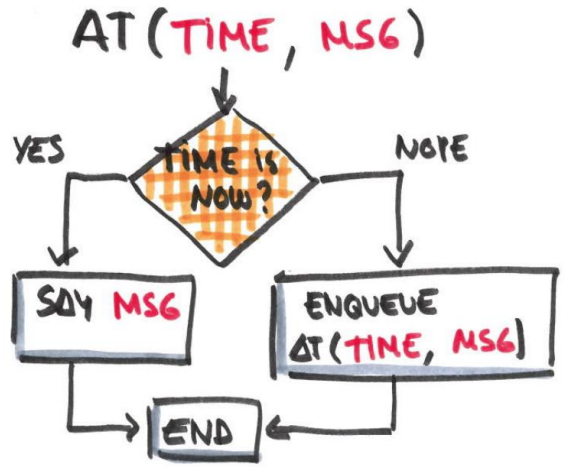
ASYNC FUNCTIONS

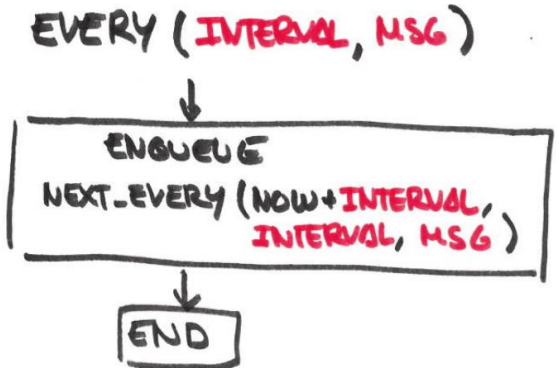


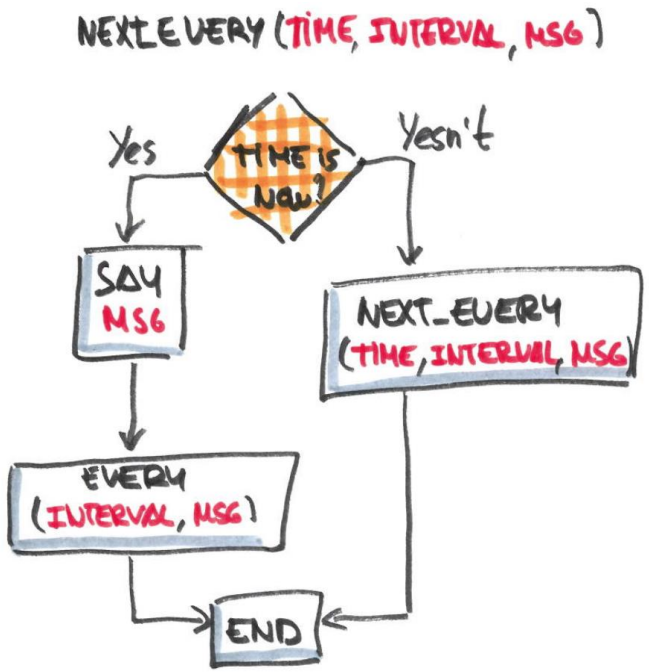


ASYNC FUNCTIONS

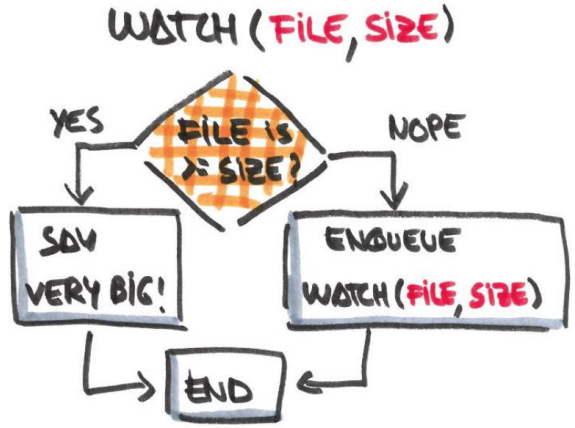


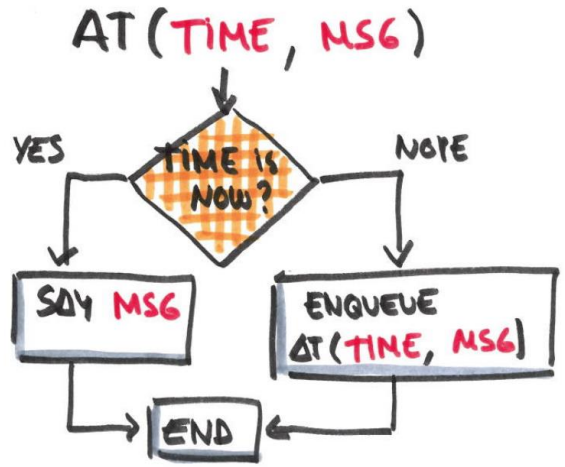


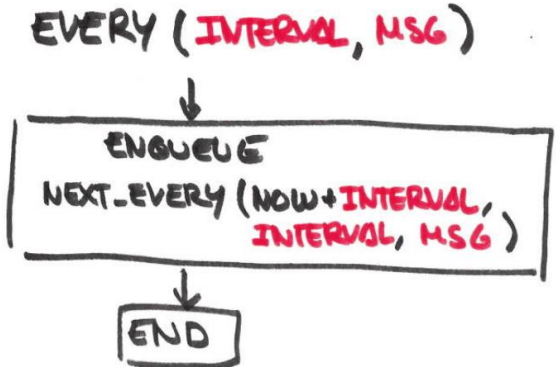


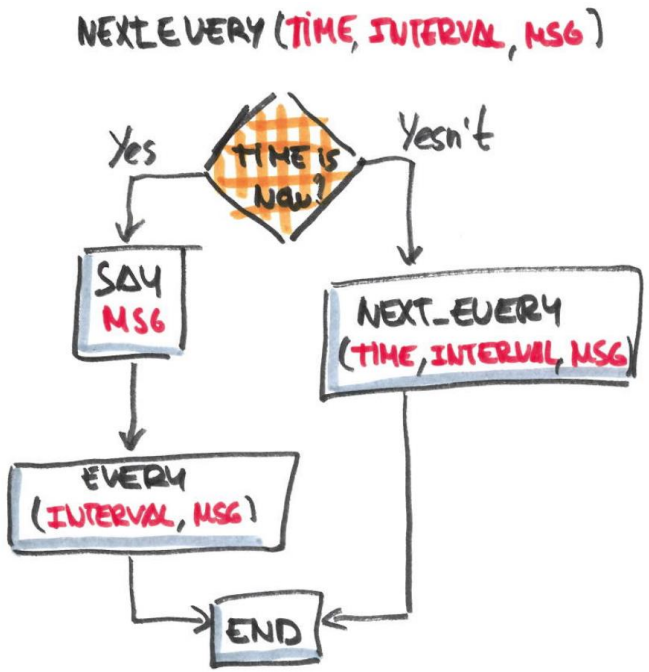


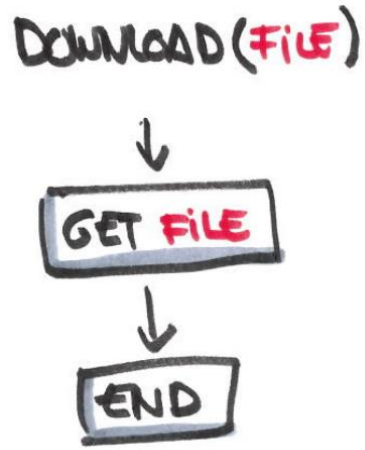
ASYNC FUNCTIONS

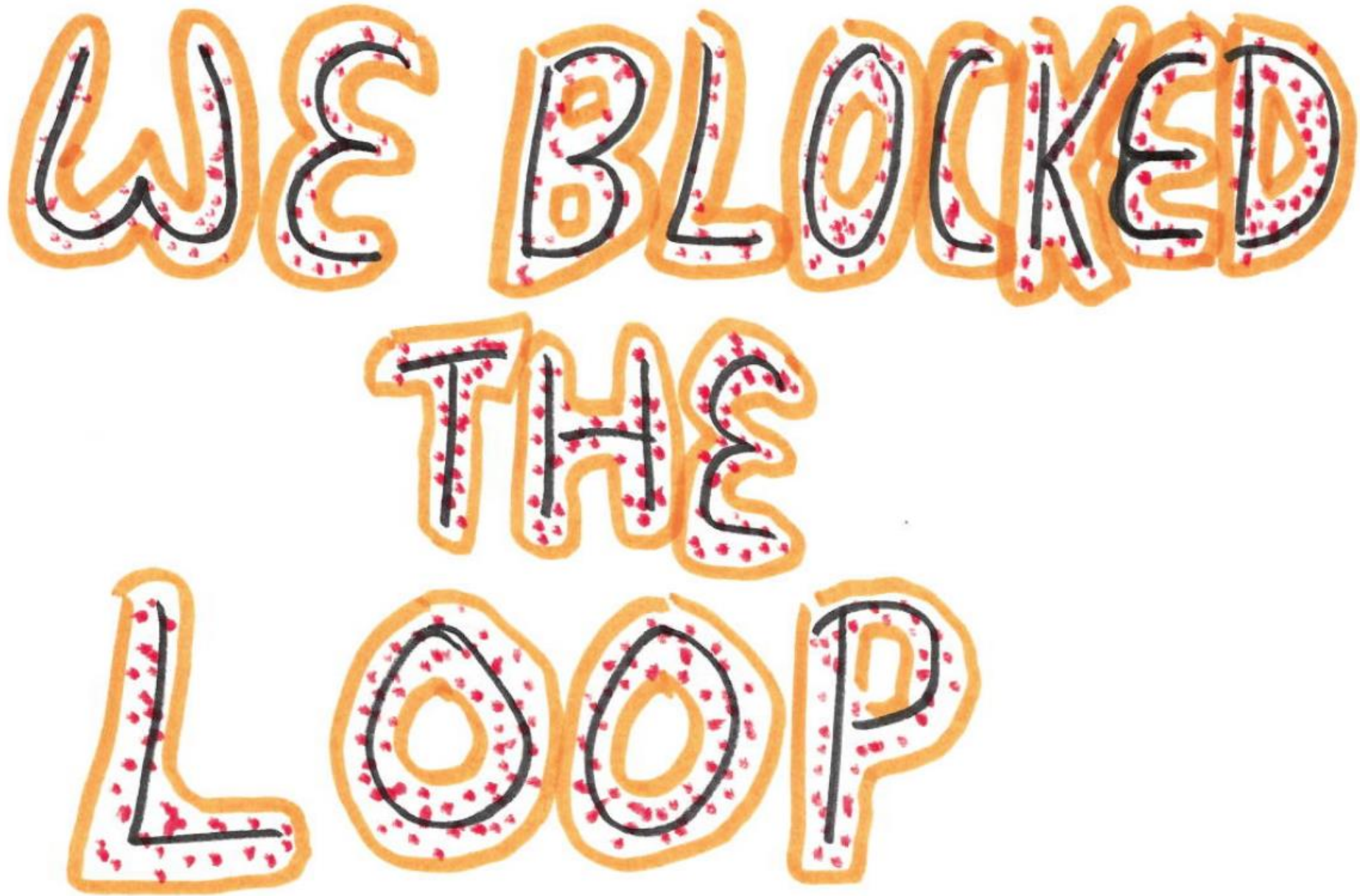











WE BLOCKED
THE
LOOP

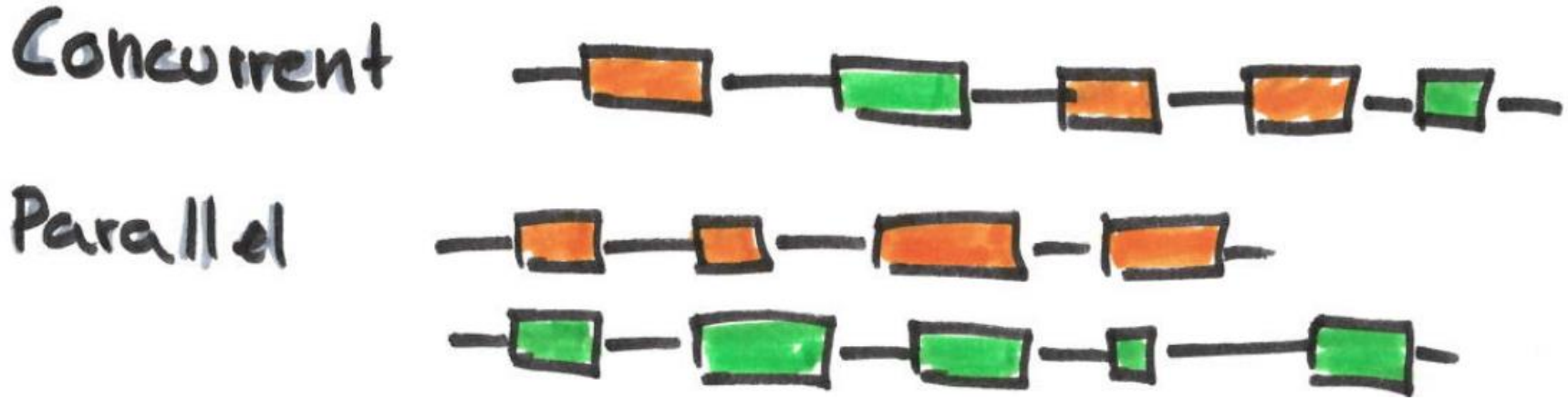
Concurrency vs Parallelism

Concurrency VS Parallelism

Concurrency VS Parallelism

Concurrent

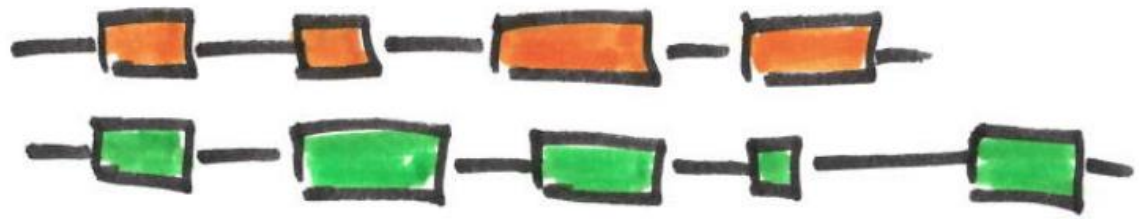
Concurrency VS Parallelism



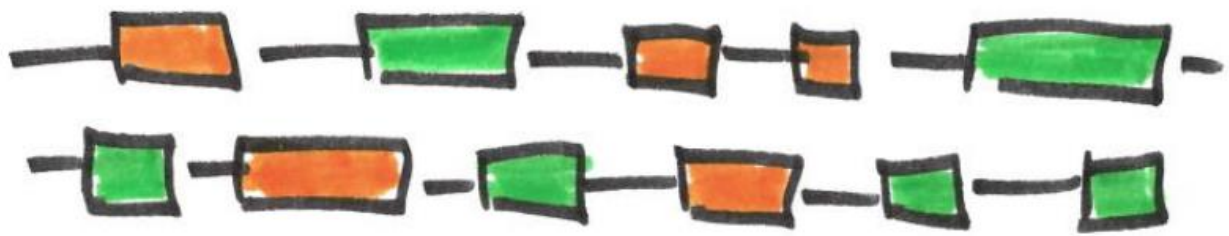
Concurrency VS Parallelism

Concurrent

Parallel



Both



(INVOLVES SOME KIND OF SYNC)

NODE does PARALLELISM with a SINGLE THREAD

NODE does PARALLELISM with a SINGLE THREAD

HOW?

NODE does PARALLELISM with a SINGLE THREAD

HOW? CHEATING

NODE does PARALLELISM with a SINGLE THREAD

HOW? CHEATING

OFFLOADS PARALLELISM TO OTHERS

LIBUV

LIBRARY FOR EVENT-DRIVEN
ASYNCHRONOUS I/O MODEL

LIBUV

LIBRARY FOR EVENT-DRIVEN
ASYNCHRONOUS I/O MODEL

LIBUV

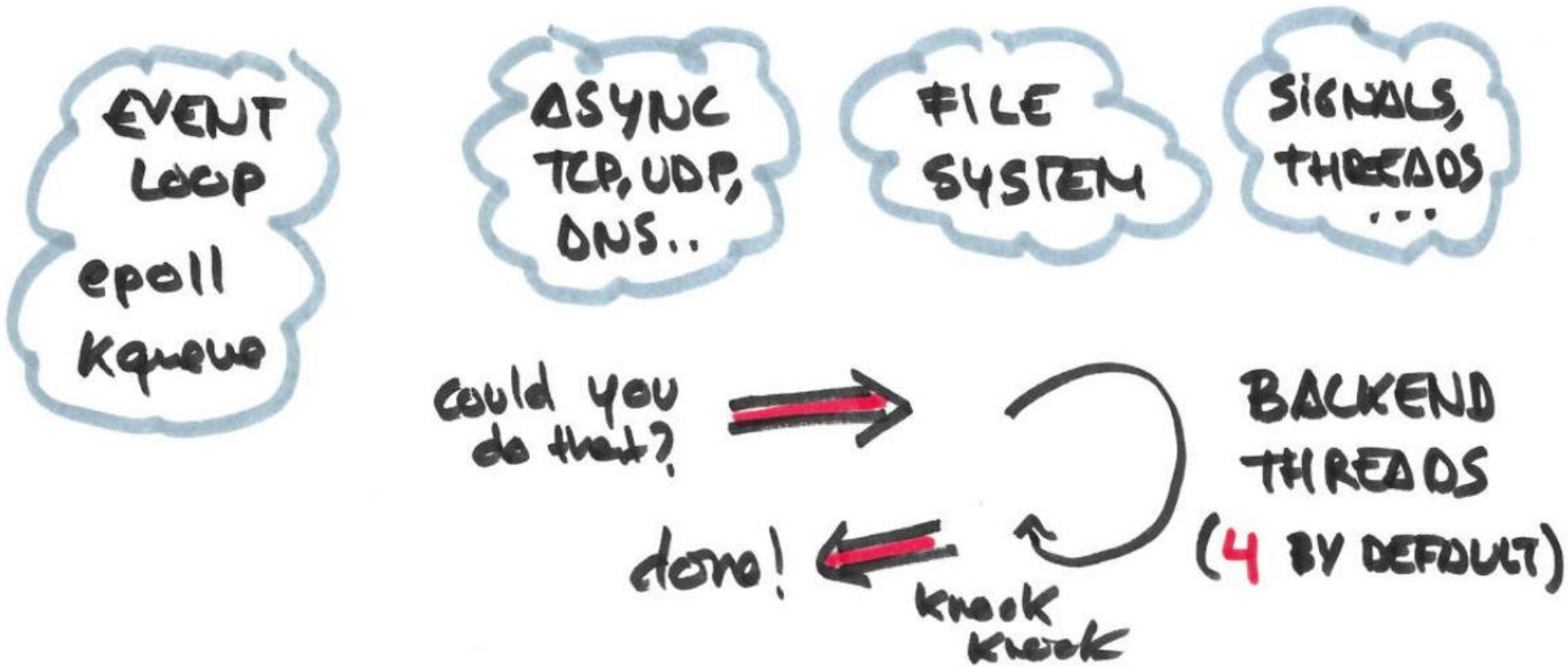
(ALSO USED BY JULIA, PYUV, ...)

LIBRARY FOR EVENT-DRIVEN ASYNCHRONOUS I/O MODEL

LIBUV

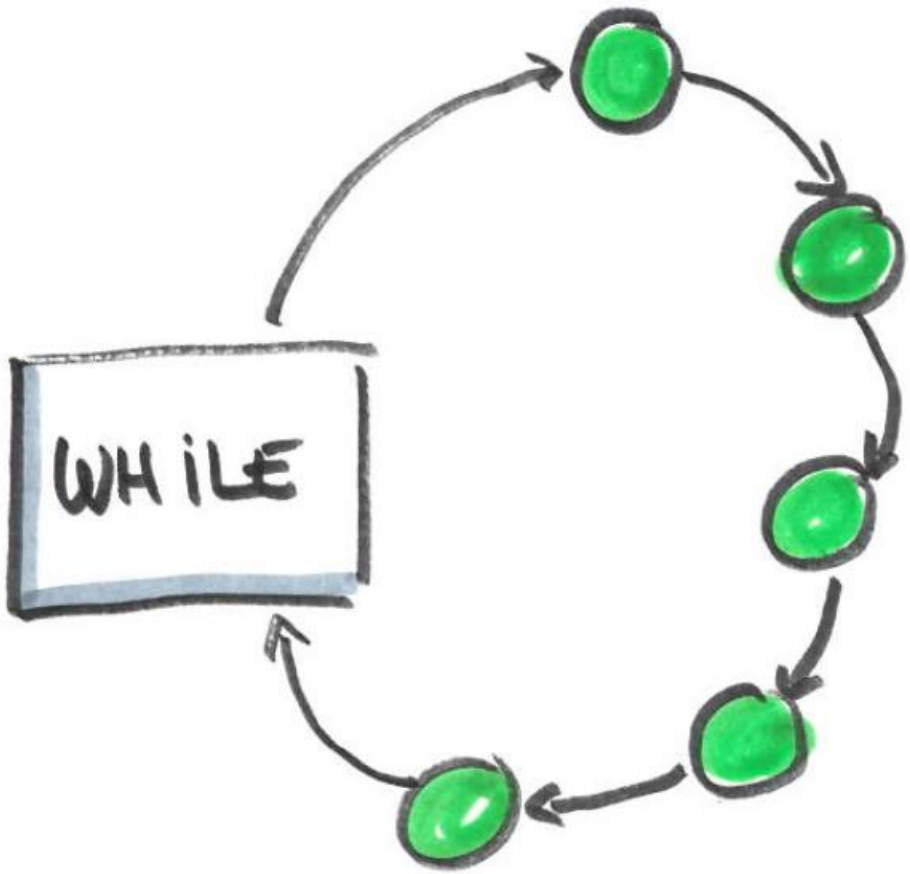
(ALSO USED BY JULIA, PYUV, ...)

LIBRARY FOR EVENT-DRIVEN ASYNCHRONOUS I/O MODEL

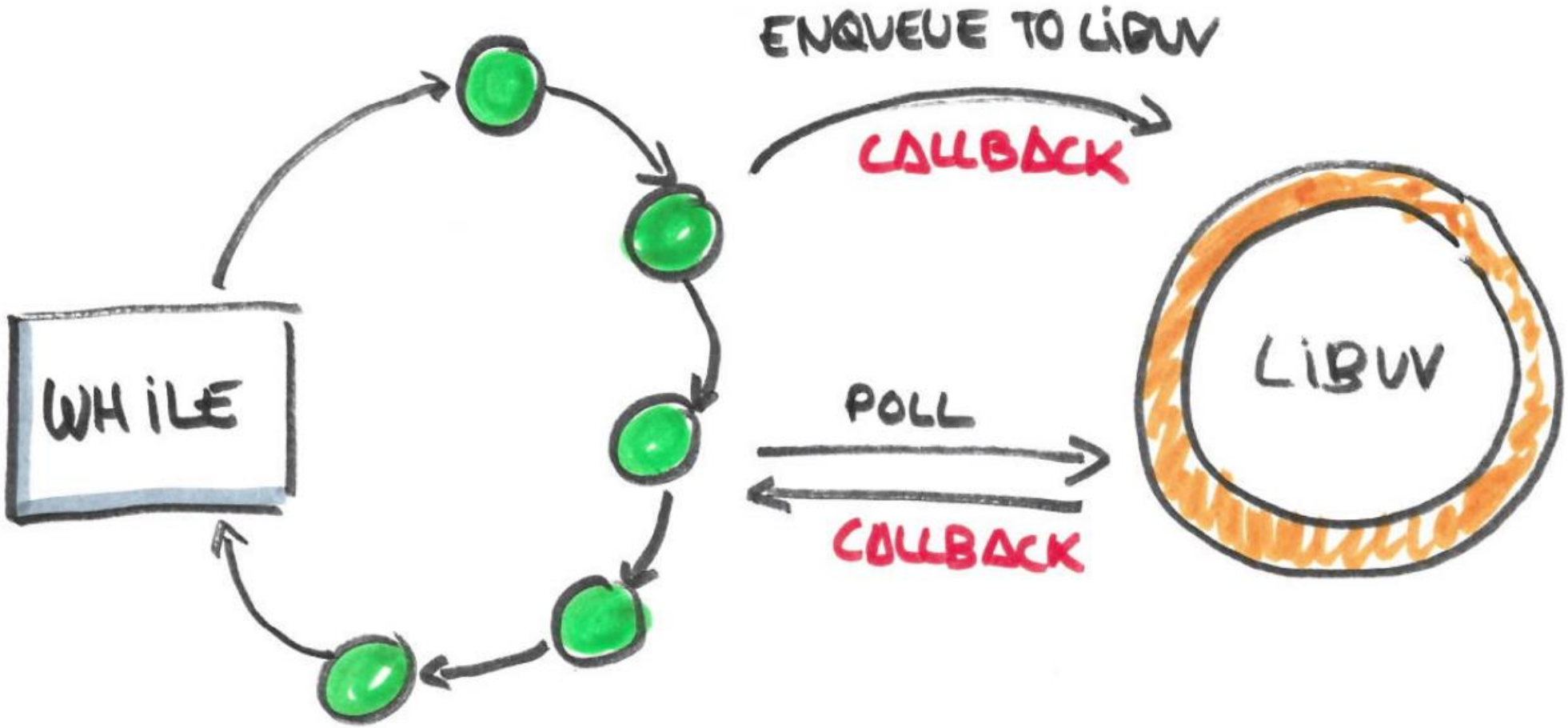


THE NODE EVENT LOOP

THE NODE EVENT LOOP

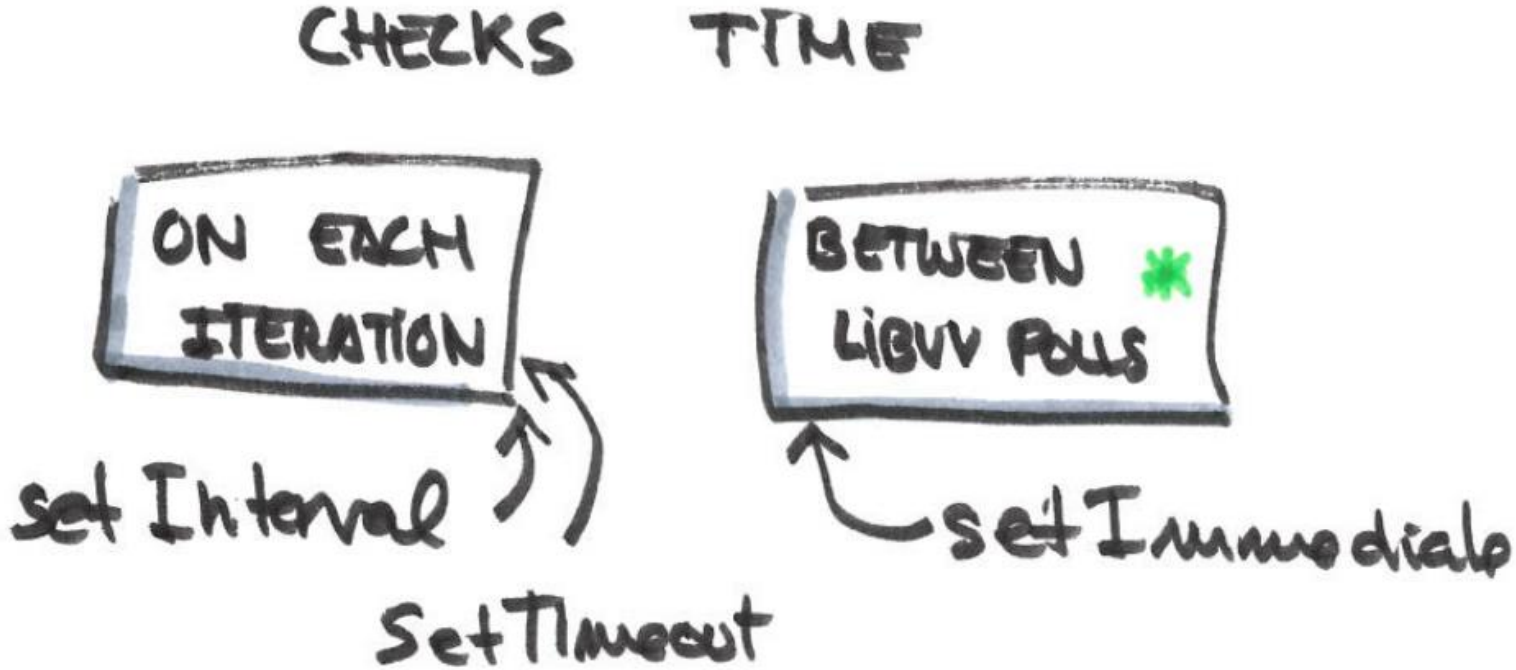


THE NODE EVENT LOOP



HOW TIMERS WORK

HOW TIMERS WORK



HOW TIMERS WORK

CHECKS TIME

ON EACH ITERATION

BETWEEN * LIBUV POLLS

setInterval

setTimeout

setImmediate

ACCURACY GUARANTEED

WEB SERVER EKOMPLE

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSEMPEL

WEB SERVER EKSAMPLE

MADE UP NUMBERS: IN REALITY IS MUCH WORSE

EVENT-LOOP MODEL

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- STILL TRICKY TO REASON ABOUT

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- STILL TRICKY TO REASON ABOUT

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- LESS CONTEXT SWITCH
- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- LESS CONTEXT SWITCH
- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING
- BAD FOR CPU-BOUND TASKS

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- LESS CONTEXT SWITCH
- WORKS WELL ON EVENT-DRIVEN SCENARIOS (WEB, UI)
- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING
- BAD FOR CPU-BOUND TASKS

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- LESS CONTEXT SWITCH
- WORKS WELL ON EVENT-DRIVEN SCENARIOS (WEB, UI)
- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING
- BAD FOR CPU-BOUND TASKS
- DOES NOT WORK FOR REAL TIME SYSTEMS

EVENT-LOOP MODEL

- "EASY" CONCURRENCY MODEL
- NO DATA SHARING BETWEEN THREADS (LESS BUGS)
- LESS CONTEXT SWITCH
- WORKS WELL ON EVENT-DRIVEN SCENARIOS (WEB, UI)

- STILL TRICKY TO REASON ABOUT
- 1 ERROR ⇒ DISCARDS EVERYTHING
- BAD FOR CPU-BOUND TASKS
- DOES NOT WORK FOR REAL TIME SYSTEMS

FITS THE SCALE-HORIZONTALLY MODEL (CLOUD, CONTAINERS)

What does ASYNC PROGRAMMING look like?

CALLBACKS

CALLBACKS

```
getUser(1, (user) => {
  getBlogPosts(user.name, (blogposts) => {
    getComments(blogposts[0], (comments) => {
      console.log(user, blogposts[0], comments);
    })
  })
});

console.log("When will this be printed?");
```

PROMISES

PROMISES

```
getUser(1)
  .then(user => getBlogPosts(user.name))
  .then(blogposts => getComments(blogposts[0]))
  .then(comments => console.log(comments))
  .catch(err => console.log('Error: ', err.message));

console.log("When will this be printed?");
```

ASYNC/AWAIT

ASYNC/AWAIT

SYNC SEMANTICS → ASYNC BEHAVIOR

ASYNC/AWAIT

SYNC SEMANTICS → ASYNC BEHAVIOR

```
async function displayComments() {
  try {
    const user = await getUser(1);
    const blogposts = await getBlogPosts(user.name);
    const comments = await getComments(blogposts[0]);
    console.log(comments);
  } catch (err) {
        console.log('Error', err.message);
  }
}

displayComments();
console.log("When will this be printed?");
```

ASYNC/AWAIT

SYNC SEMANTICS → ASYNC BEHAVIOR

```
await Promise.all(call1, call2, call3)
```


node
JS

- A JS ASYNC RUNTIME

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM
- GREAT FOR WEB APPS

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM
- GREAT FOR WEB APPS
- SCRIPTS AND CLI

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

- GREAT FOR WEB APPS
- SCRIPTS AND CLI
- ELECTRON USES NODE

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

- GREAT FOR WEB APPS
- SCRIPTS AND CLI
- ELECTRON USES NODE
- CONTAINER/CLOUD FRIENDLY

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

IT'S **FUN**

- GREAT FOR WEB APPS
- SCRIPTS AND CLI
- ELECTRON USES NODE
- CONTAINER/CLOUD FRIENDLY

- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

IT'S FUN AND FAST

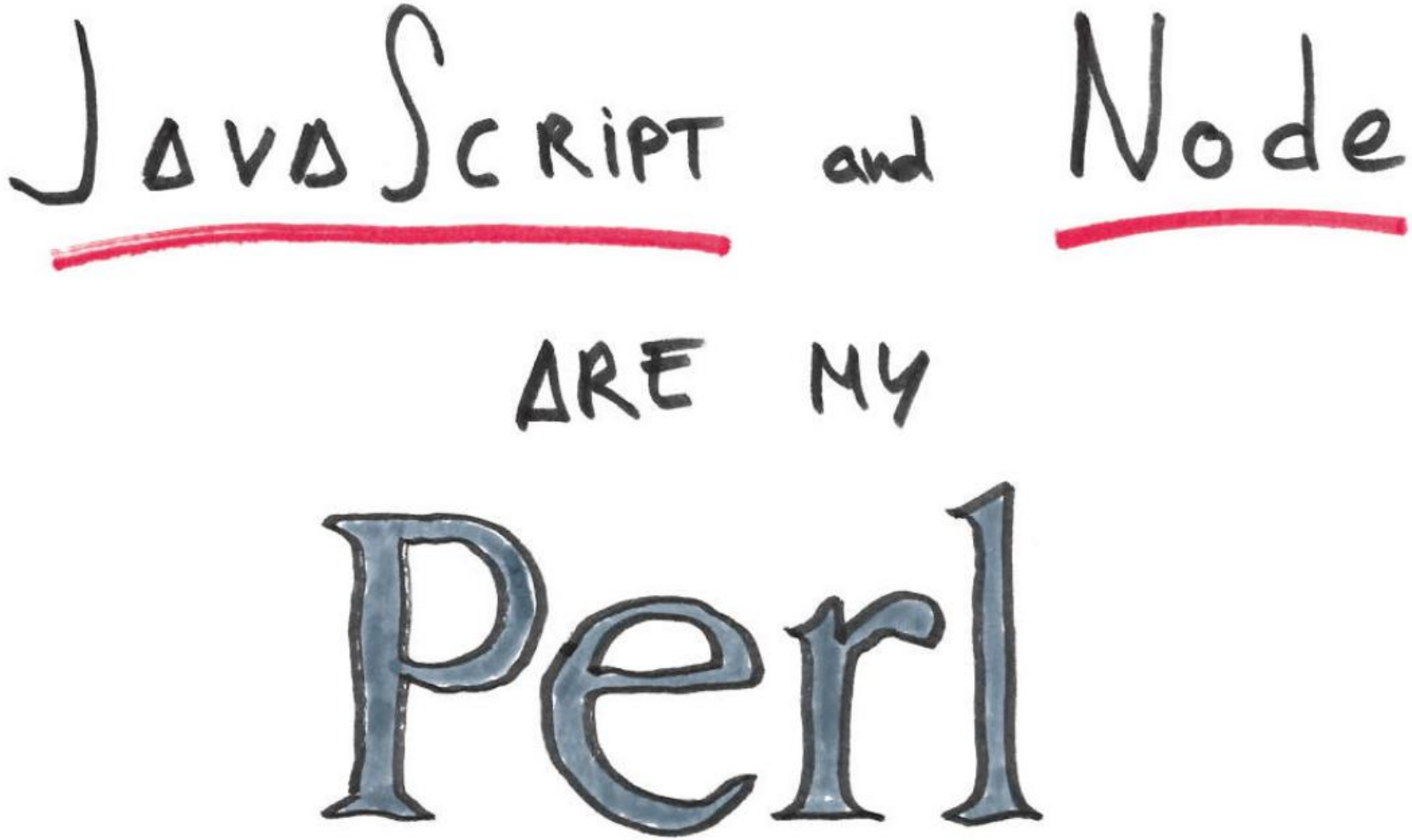
- GREAT FOR WEB APPS
- SCRIPTS AND CLI
- ELECTRON USES NODE
- CONTAINER/CLOUD FRIENDLY

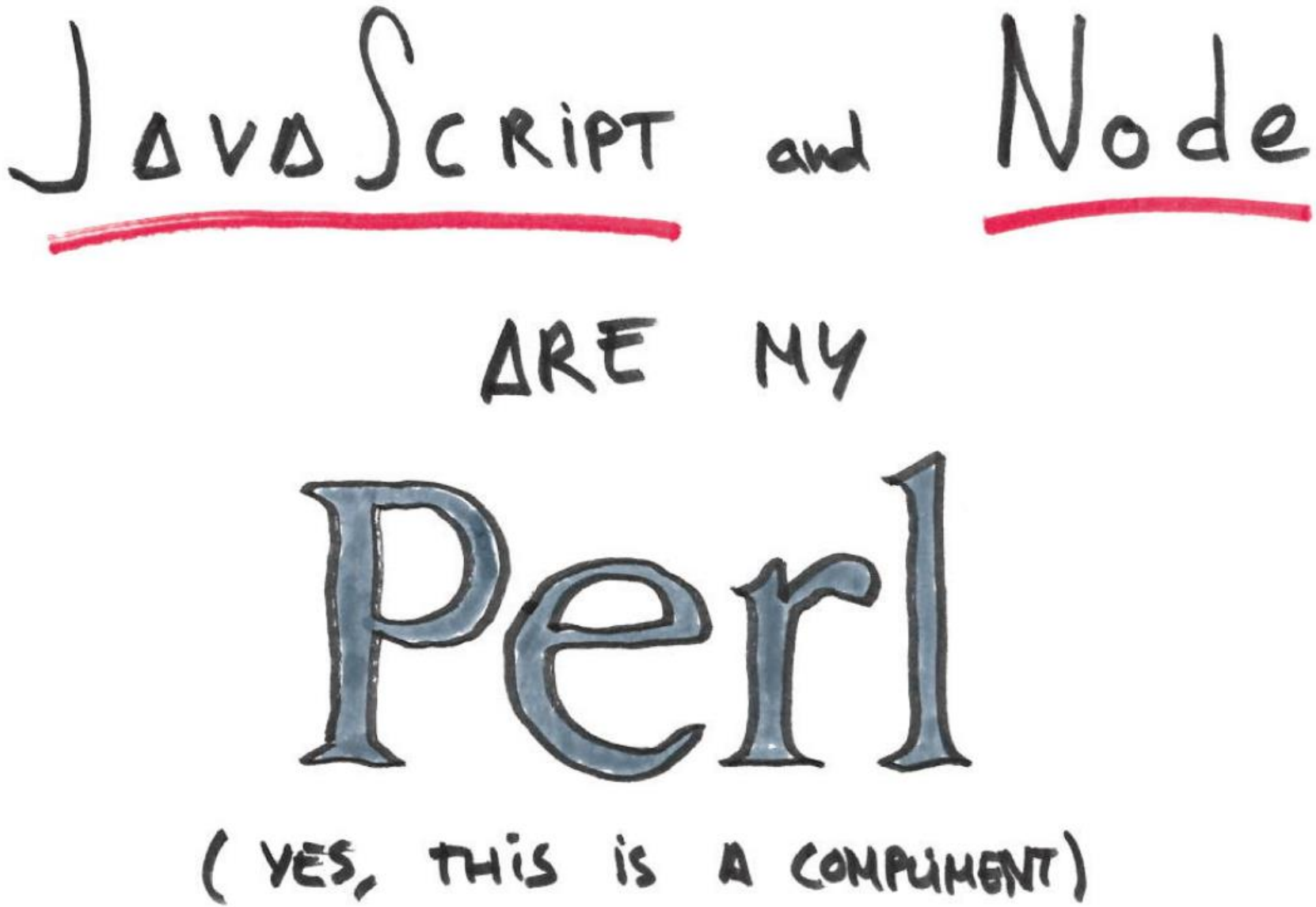
- A JS ASYNC RUNTIME
- A SET OF ASYNC LIBS
- AN ASYNC ECOSYSTEM

IT'S FUN AND FAST AND FAST

- GREAT FOR WEB APPS
- SCRIPTS AND CLI
- ELECTRON USES NODE
- CONTAINER/CLOUD FRIENDLY

JavaScript and Node


JavaScript and Node
ARE MY
Perl


JavaScript and Node
ARE MY
Perl
(YES, THIS IS A COMPLIMENT)


THANK
YOU!

QUESTIONS