# How does Async work?

# ~ Matthias Endler ~

# Matthias Endler

BACKEND ENGINEER AT TRIVAGO

# ~ Matthias Endler ~

BACKEND ENGINEER AT TRIVAGO

MOSTLY GO, RUST, PYTHON

# ~ Matthias Endler ~

BACKEND ENGINEER AT TRIVAGO

MOSTLY GO, RUST, PYTHON

INTERESTED IN LOW LEVEL STUFF

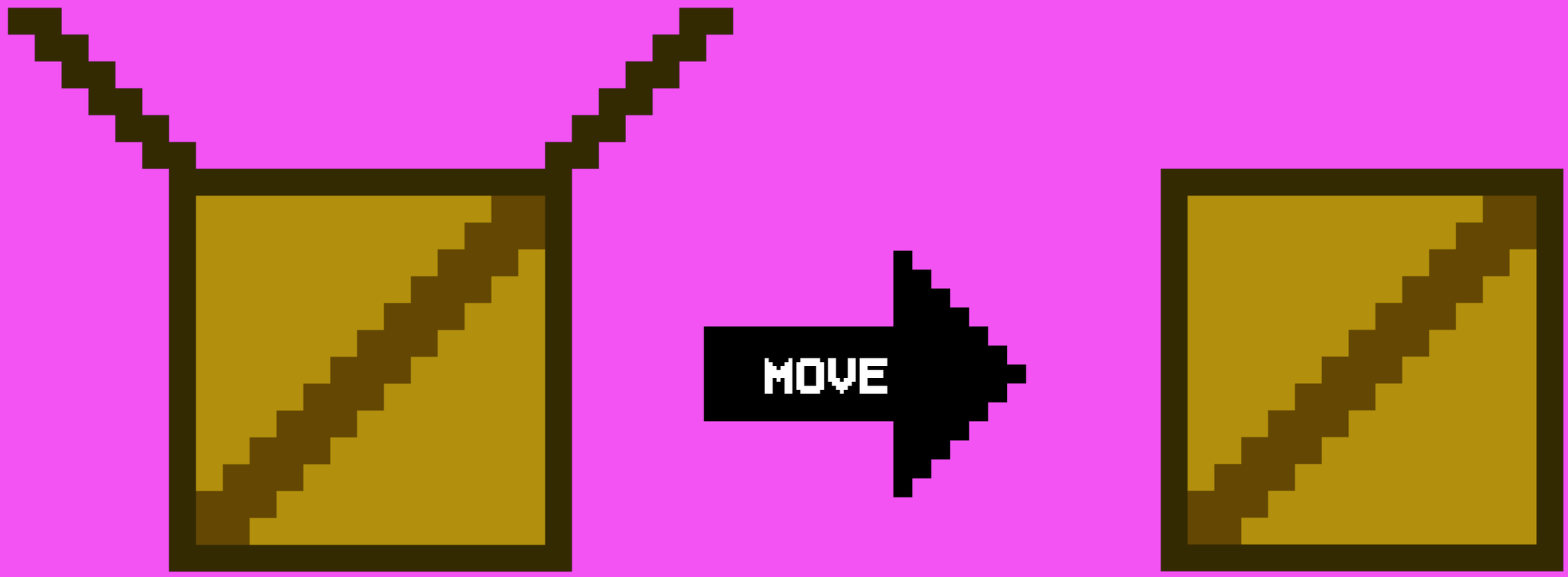
# Matthias Endler

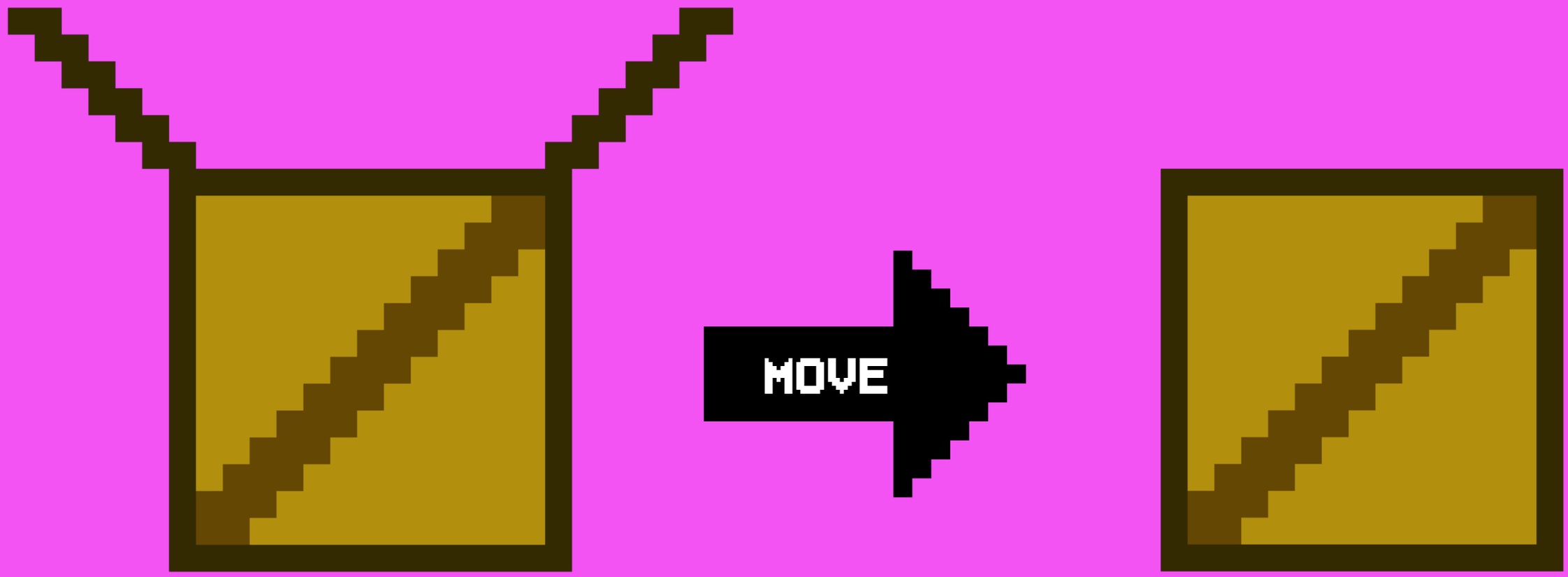
BACKEND ENGINEER AT TRIVAGO

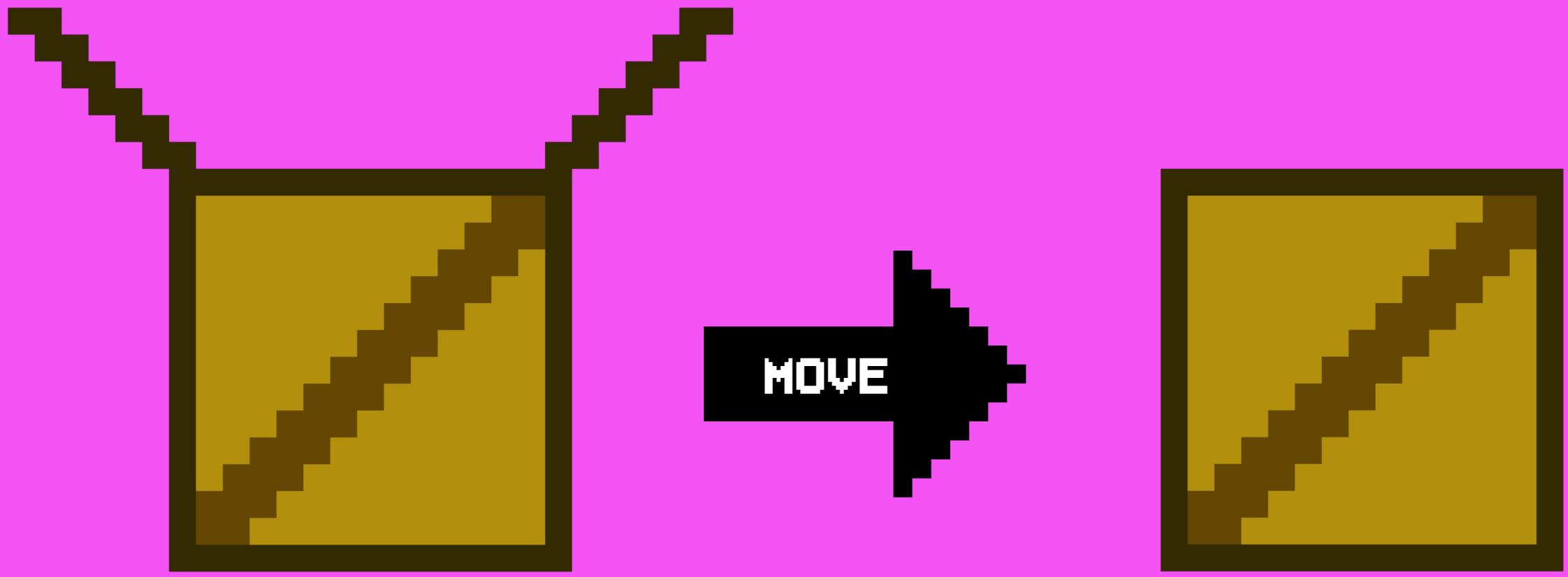
MOSTLY GO, RUST, PYTHON

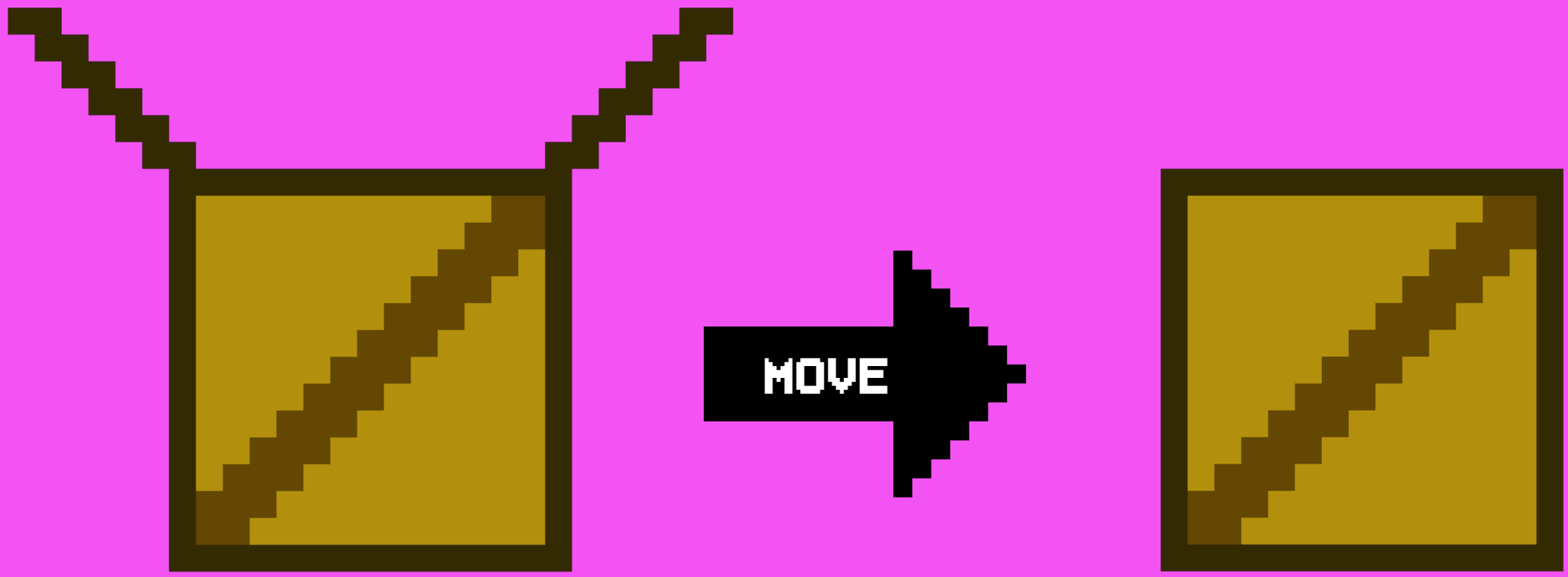
INTERESTED IN LOW LEVEL STUFF

ENDLER.DEV AND HELLO-RUST.SHOW


MOVE

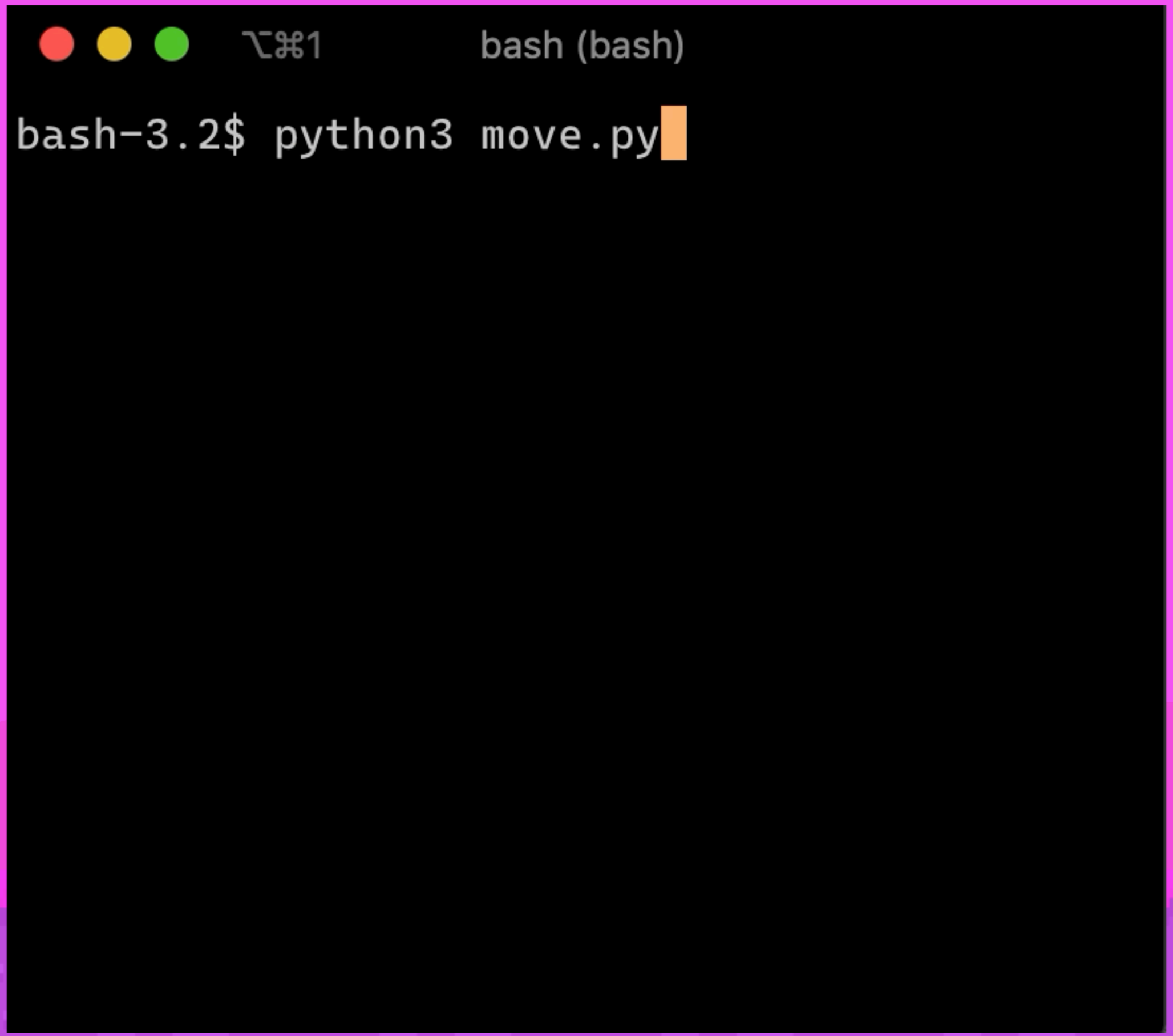

MOVE

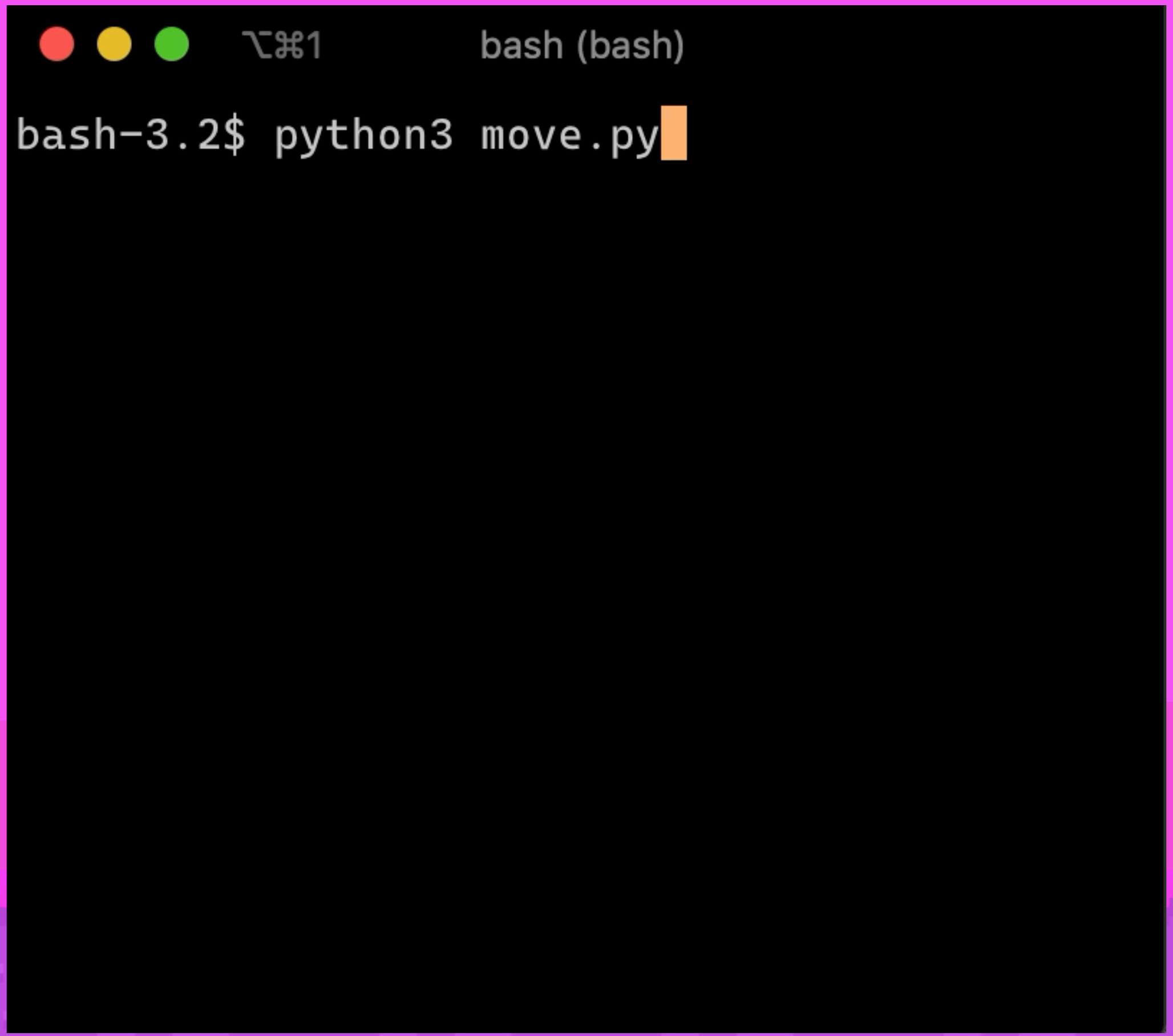

MOVE


MOVE

# ~ Python ~

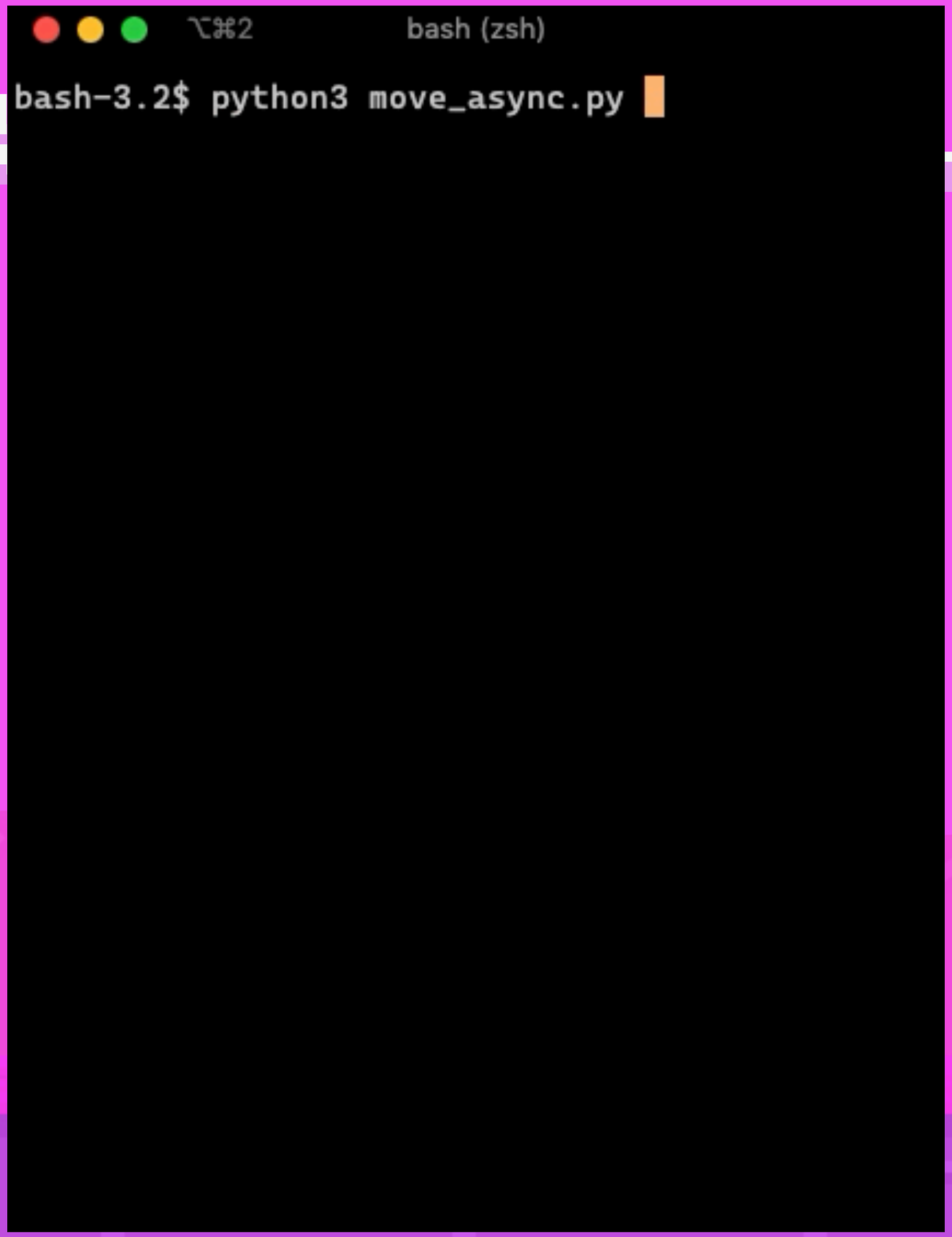
```python
def move(batch):
    pack(batch)
    transport(batch)
    unpack(batch)


def main():
    for batch in batches(boxes, 10):
        move(batch)
```

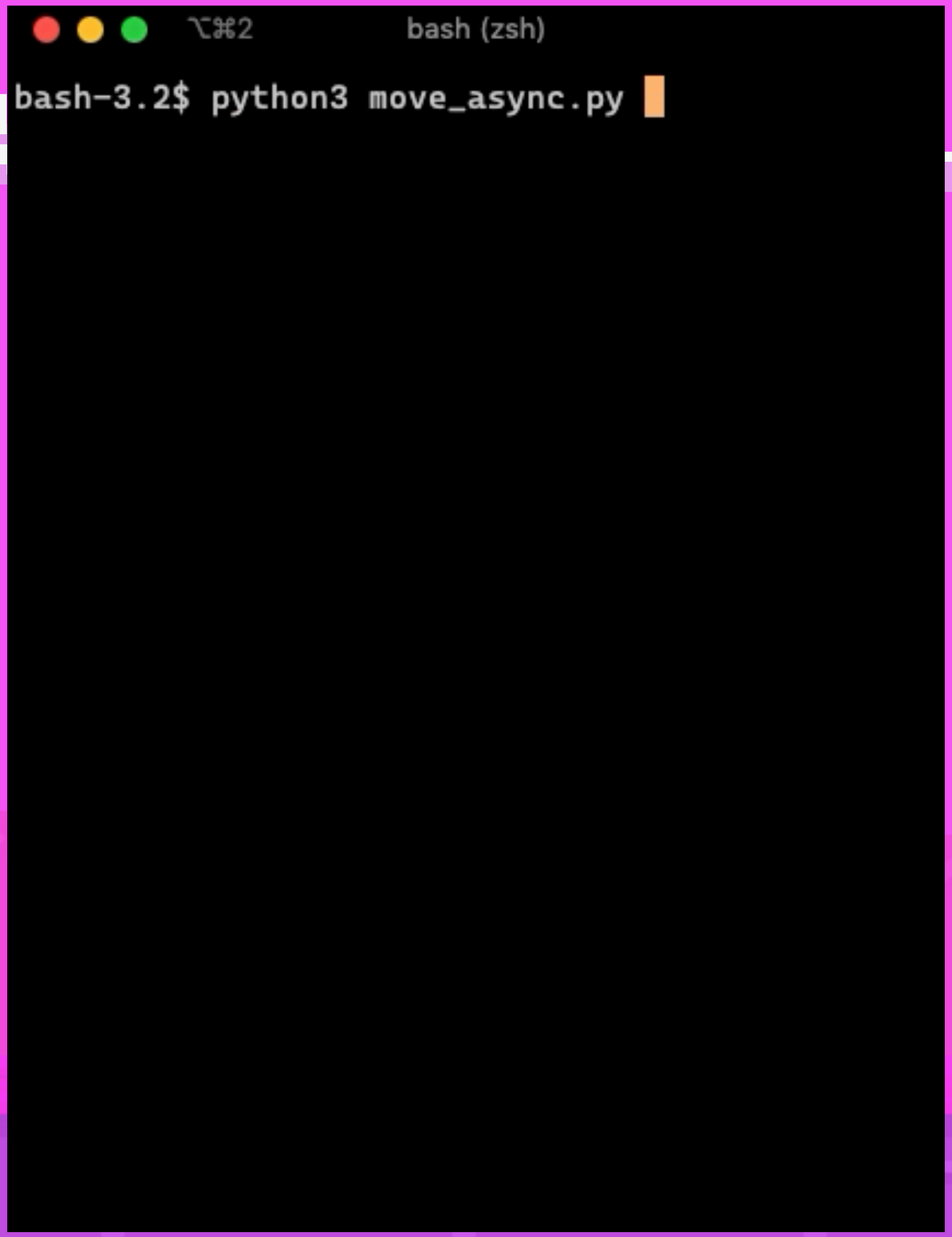

⌥⌘1
bash (bash)
bash-3.2$ python3 move.py


⌥⌘1
bash (bash)
bash-3.2$ python3 move.py

```python
def move(batch):
    pack(batch)
    transport(batch)
    unpack(batch)


def main():
    for batch in batches(boxes, 10):
        move(batch)
```

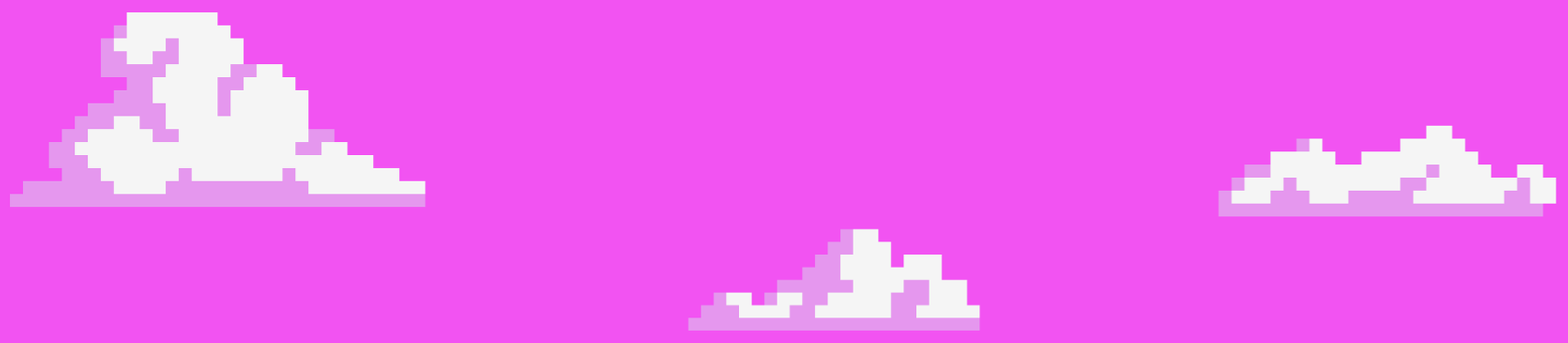
```python
async def move(batch):
    await pack(batch)
    await transport(batch)
    await unpack(batch)


async def main():
    all_batches = [move(batch) for batch in batches(boxes, 10)]
    await asyncio.gather(*all_batches)
```


⌥⌘2
bash (zsh)
bash-3.2$ python3 move_async.py

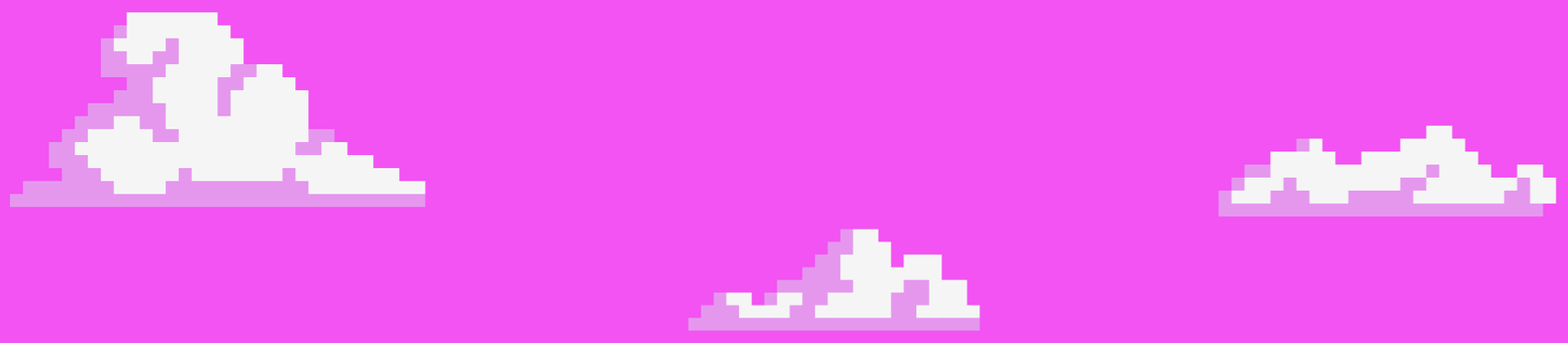

⌥⌘2
bash (zsh)
bash-3.2$ python3 move_async.py

# Python's asyncio

* COROUTINES
* TASKS
* EVENT LOOP
* SELECT



```python
async def pack(boxes):
    print("Packing...")
    await asyncio.sleep(PACK_TIME
```



```python
async def pack(boxes):
    print("Packing...")
    await asyncio.sleep(PACK_TIME
```

# COROUTINE

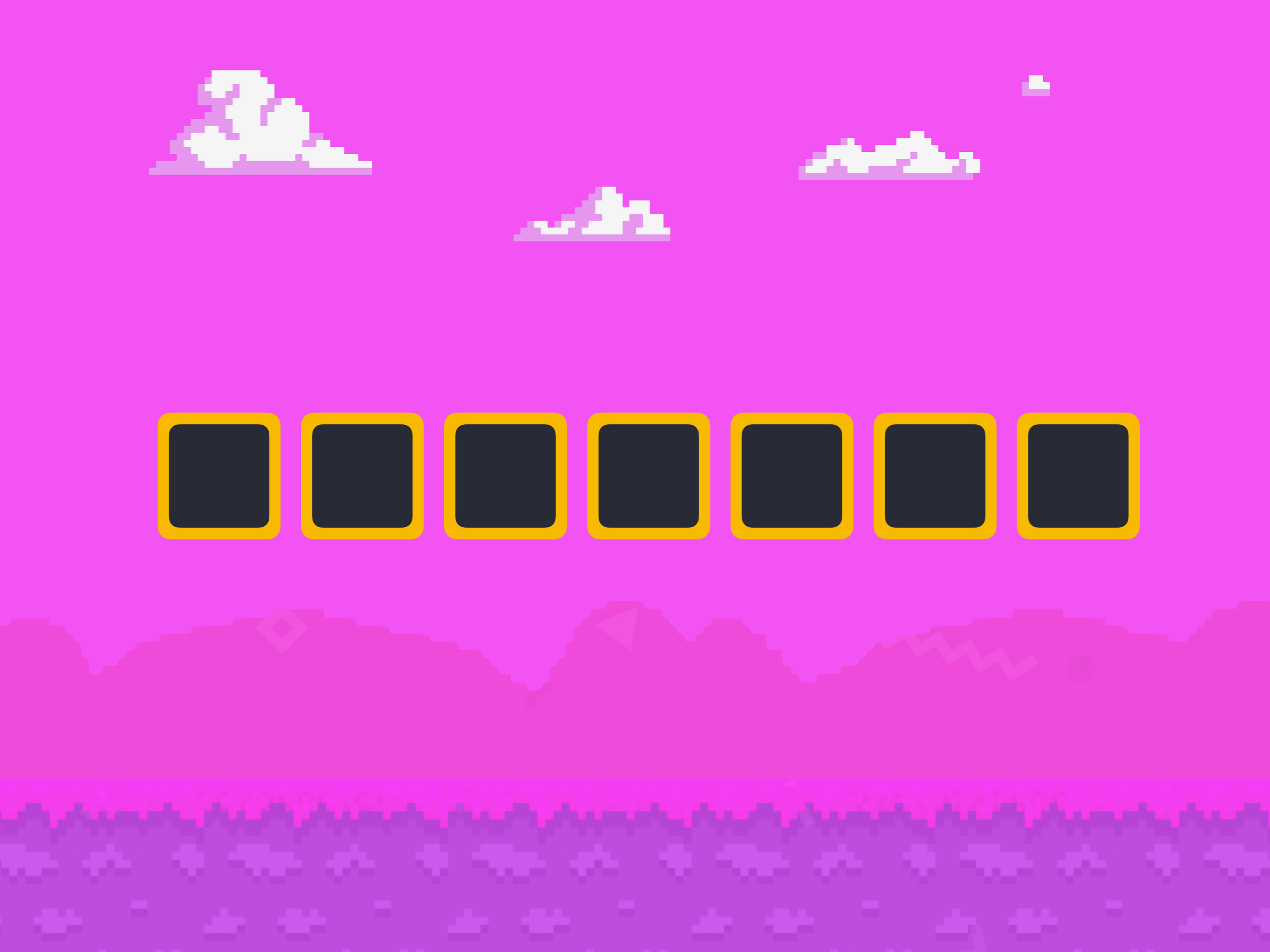
ASYNCIO.GATHER( COROUTINE )

TASK

COROUTINE

ASYNCIO.RUN( )

TASK



USERSPACE

SELECT

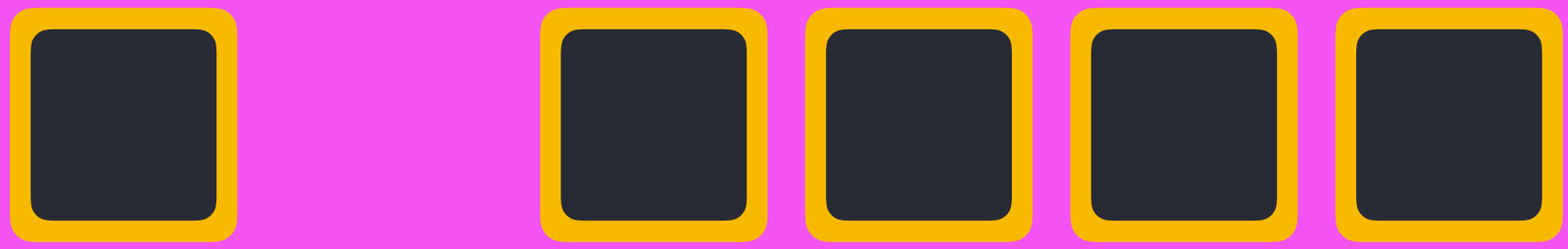
KERNEL

USERSPACE

SELECT

KERNEL

USERSPACE



SELECT

KERNEL

USERSPACE

SELECT

KERNEL

USERSPACE

$ELECT

KERNEL

USERSPACE

$ELECT

KERNEL

USERSPACE

$ELECT

KERNEL

USERSPACE

$ELECT

KERNEL

## DESCRIPTION top

**select**() and **pselect**() allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation (e.g., input possible). A file descriptor is considered ready if it is possible to perform a corresponding I/O operation (e.g., read(2), or a sufficiently small write(2)) without blocking.

Previous topic
Networking and Interprocess Communication

Next topic
Coroutines and Tasks

This Page
Report a Bug
Show Source

# asyncio — Asynchronous I/O

asyncio is a library to write **concurrent** code using the **async/await** syntax.

asyncio is used as a foundation for multiple Python asynchronous frameworks that provide high-performance network and web-servers, database connection libraries, distributed task queues, etc.

asyncio is often a perfect fit for IO-bound and high-level **structured** network code.

asyncio provides a set of **high-level** APIs to:

- run Python coroutines concurrently and have full control over their execution;
- perform network IO and IPC;
- control subprocesses;
- distribute tasks via queues;
- synchronize concurrent code;

Additionally, there are **low-level** APIs for *library and framework developers* to:

- create and manage event loops, which provide asynchronous APIs for `networking`, running `subprocesses`, handling `OS signals`, etc;
- implement efficient protocols using transports;
- bridge callback-based libraries and code with async/await syntax.

**Hello World!**

```
import asyncio

async def main():
    print('Hello ...')
    await asyncio.sleep(1)
    print('... World!')

# Python 3.7+
asyncio.run(main())
```

**Reference**

High-level APIs

- Coroutines and Tasks
- Streams
- Synchronization Primitives

python / cpython

Sponsor | Watch 1,090 | Star 26,748 | Fork 11,659



Branch: master ▾ cpython / Lib / asyncio / | Create new file | Upload files | Find file | History

aeros167 and asvetlov bpo-34037: Fix test_asyncio failure and add loop.shutdown_default_exe... ··· ✓ Latest commit 9fdc64c 3 days ago

| | | |
|---|---|---|
| .. | | |
| __init__.py | bpo-36889: Merge asyncio streams (GH-13251) | 4 months ago |
| __main__.py | bpo-37028: asyncio REPL; activated via 'python -m asyncio'. (GH-13472) | 4 months ago |
| base_events.py | bpo-34037: Fix test_asyncio failure and add loop.shutdown_default_exe... | 3 days ago |
| base_futures.py | bpo-34622: Extract asyncio exceptions into a separate module (GH-9141) | last year |
| base_subprocess.py | bpo-32528: Make asyncio.CancelledError a BaseException. (GH-13528) | 4 months ago |
| base_tasks.py | bpo-34270: Make it possible to name asyncio tasks (GH-8547) | last year |
| constants.py | bpo-33734: asyncio/ssl: a bunch of bugfixes (#7321) | last year |
| coroutines.py | bpo-36921: Deprecate **@coroutine** for sake of async def (GH-13346) | 4 months ago |
| events.py | bpo-34037: Fix test_asyncio failure and add loop.shutdown_default_exe... | 3 days ago |
| exceptions.py | bpo-32528: Make asyncio.CancelledError a BaseException. (GH-13528) | 4 months ago |
| format_helpers.py | bpo-33672: Fix Task.__repr__ crash with Cython's bogus coroutines (GH... | last year |
| futures.py | bpo-34622: Extract asyncio exceptions into a separate module (GH-9141) | last year |
| locks.py | bpo-36373: Fix deprecation warnings (GH-15889) | 12 days ago |
| log.py | Rename the logger to plain "logger". | 6 years ago |
| proactor_events.py | bpo-34679: Restore instantiation Windows IOCP event loop from non-mai... | 28 days ago |
| protocols.py | bpo-35394: Add empty slots to abstract asyncio protocols (#10889) | 10 months ago |
| queues.py | bpo-36373: Fix deprecation warnings (GH-15889) | 12 days ago |

| | | |
|---|---|---|
| events.py | bpo-34037: Fix test_asyncio failure and add loop.shutdown_default_exe... | 3 days ago |
| exceptions.py | bpo-32528: Make asyncio.CancelledError a BaseException. (GH-13528) | 4 months ago |
| format_helpers.py | bpo-33672: Fix Task.__repr__ crash with Cython's bogus coroutines (GH... | last year |
| futures.py | bpo-34622: Extract asyncio exceptions into a separate module (GH-9141) | last year |
| locks.py | bpo-36373: Fix deprecation warnings (GH-15889) | 12 days ago |
| log.py | Rename the logger to plain "logger". | 6 years ago |
| proactor_events.py | bpo-34679: Restore instantiation Windows IOCP event loop from non-mai... | 28 days ago |
| protocols.py | bpo-35394: Add empty slots to abstract asyncio protocols (#10889) | 10 months ago |
| queues.py | bpo-36373: Fix deprecation warnings (GH-15889) | 12 days ago |
| runners.py | bpo-34037: Fix test_asyncio failure and add loop.shutdown_default_exe... | 3 days ago |
| selector_events.py | bpo-37027: Return a proxy socket object from transp.get_extra_info('s... | 4 months ago |
| sslproto.py | bpo-37035: Don't log OSError (GH-13548) | 4 months ago |
| staggered.py | bpo-32528: Make asyncio.CancelledError a BaseException. (GH-13528) | 4 months ago |
| streams.py | bpo-36889: Document Stream class and add docstrings (GH-14488) | 10 days ago |
| subprocess.py | bpo-36373: Deprecate explicit loop in task and subprocess API (GH-16033) | 10 days ago |
| tasks.py | bpo-36373: Deprecate explicit loop in task and subprocess API (GH-16033) | 10 days ago |
| transports.py | bpo-38148: Add slots to asyncio transports (GH-16077) | 9 days ago |
| trsock.py | bpo-37027: Return a proxy socket object from transp.get_extra_info('s... | 4 months ago |
| unix_events.py | Fix typos in docs, comments and test assert messages (#14872) | 2 months ago |
| windows_events.py | bpo-29883: Asyncio proactor udp (GH-13440) | 4 months ago |
| windows_utils.py | asyncio: __del__() keep reference to warnings.warn (GH-11491) | 9 months ago |

# JavaScript

# Async JavaScript

* CALLBACKS
* PROMISES
* ASYNC/AWAIT

# JavaScript Callbacks

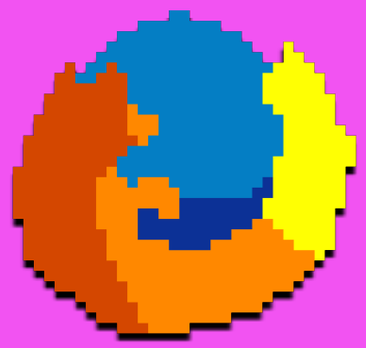
* THE PROBLEM
* THE SOLUTION

```
var button = document.getElementById('button');

button.onclick = function() {
    while (true);
};
```

```
document.querySelector('button').addEventListener('click', () => {
  function loop() {
    setTimeout(loop, 0);
  }
  loop();
});
```

V8

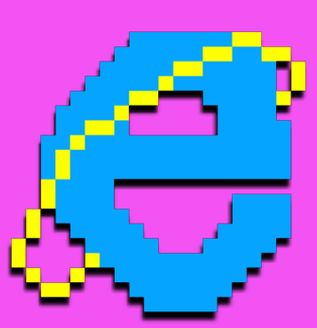
V8



SPIDERMONKEY

V8

SPIDERMONKEY



CHAKRA CORE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

JS

WEB APIS

SETTIMEOUT

DOM

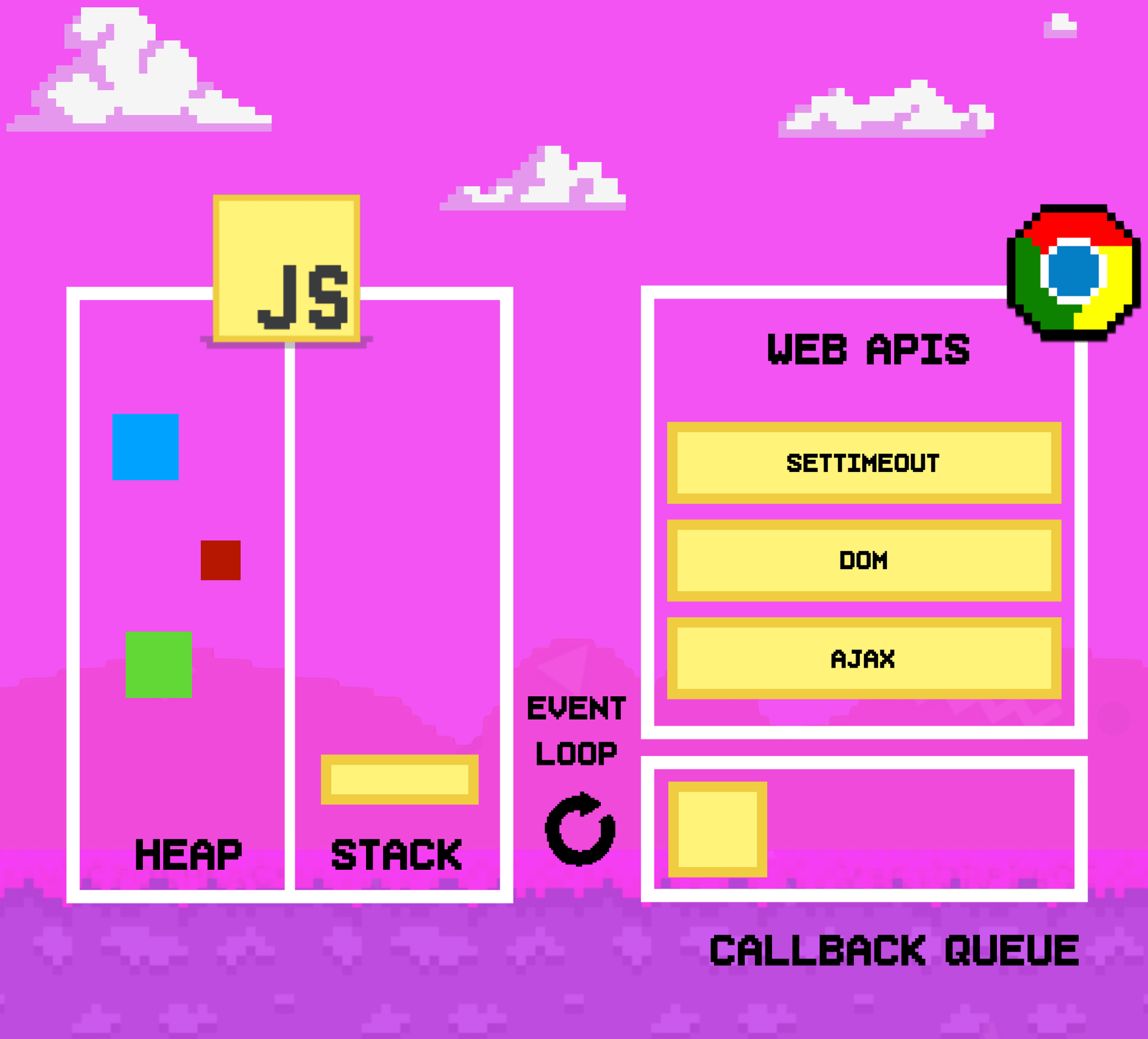
AJAX

EVENT LOOP

HEAP

STACK

CALLBACK QUEUE


JS
WEB APIS
SETTIMEOUT
DOM
AJAX
EVENT
LOOP
HEAP
STACK
CALLBACK QUEUE

JS

WEB APIS

SETTIMEOUT

DOM

AJAX

EVENT LOOP

HEAP

STACK

CALLBACK QUEUE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

JS

HEAP

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CALLBACK QUEUE

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

JS

WEB APIS

SETTIMEOUT

DOM

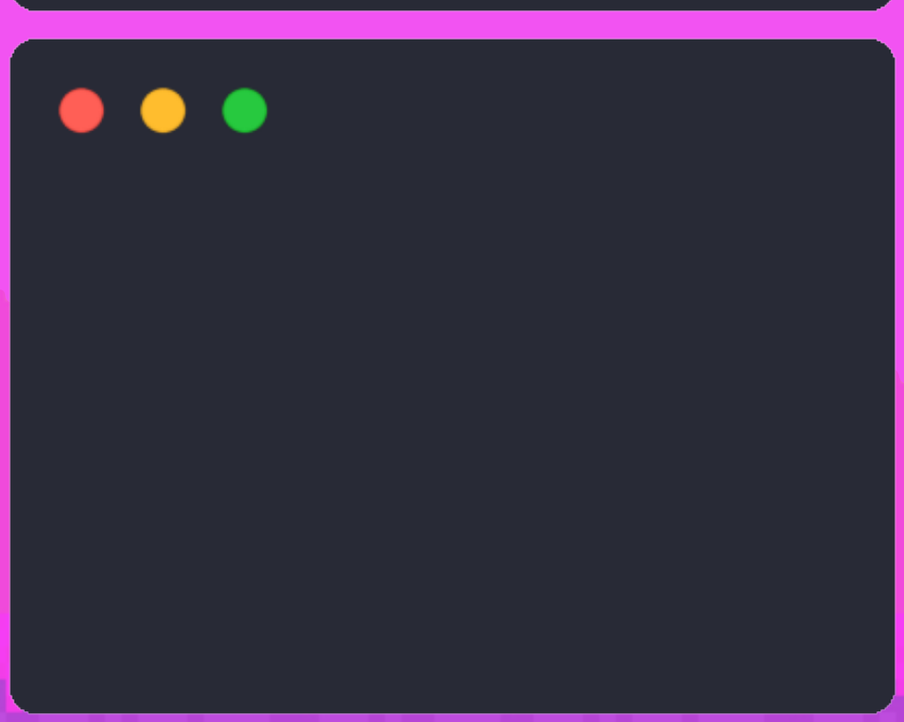
AJAX

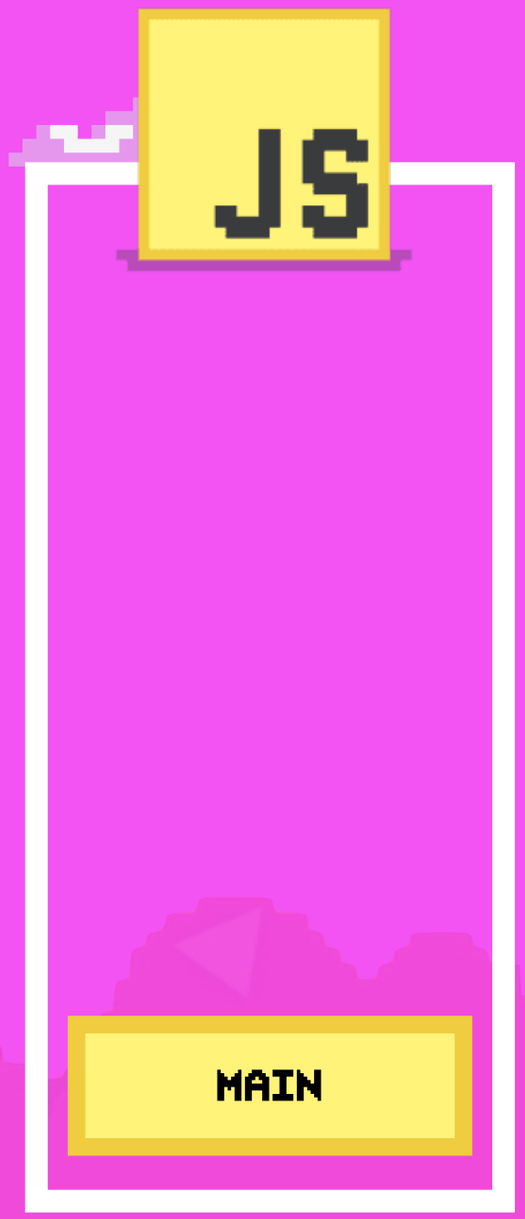
EVENT LOOP

STACK

CALLBACK QUEUE

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```






EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

JS

CONSOLE.LOG

MAIN

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

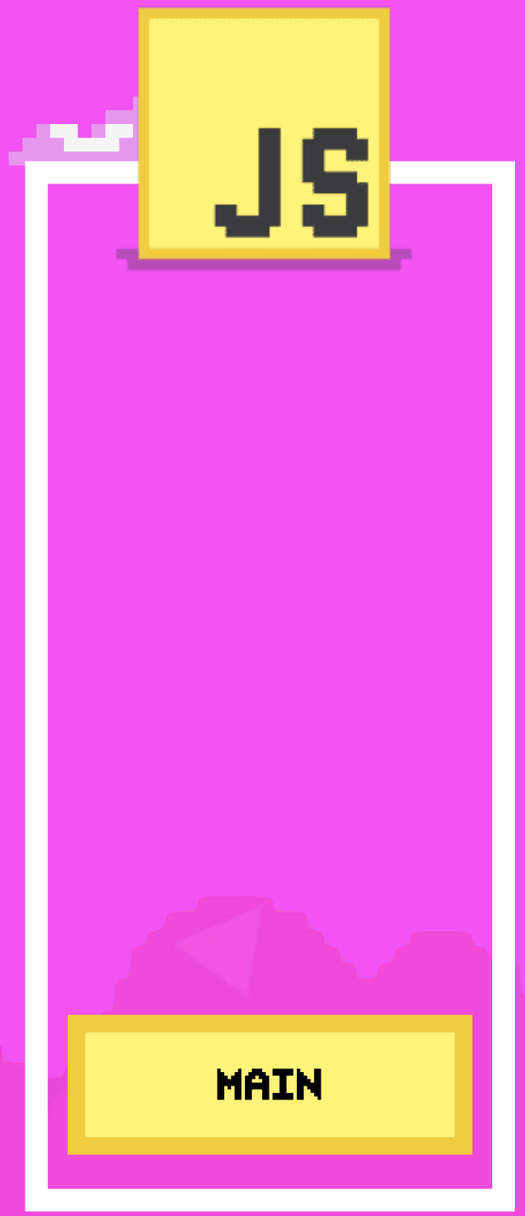
DOM

AJAX

CALLBACK QUEUE

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

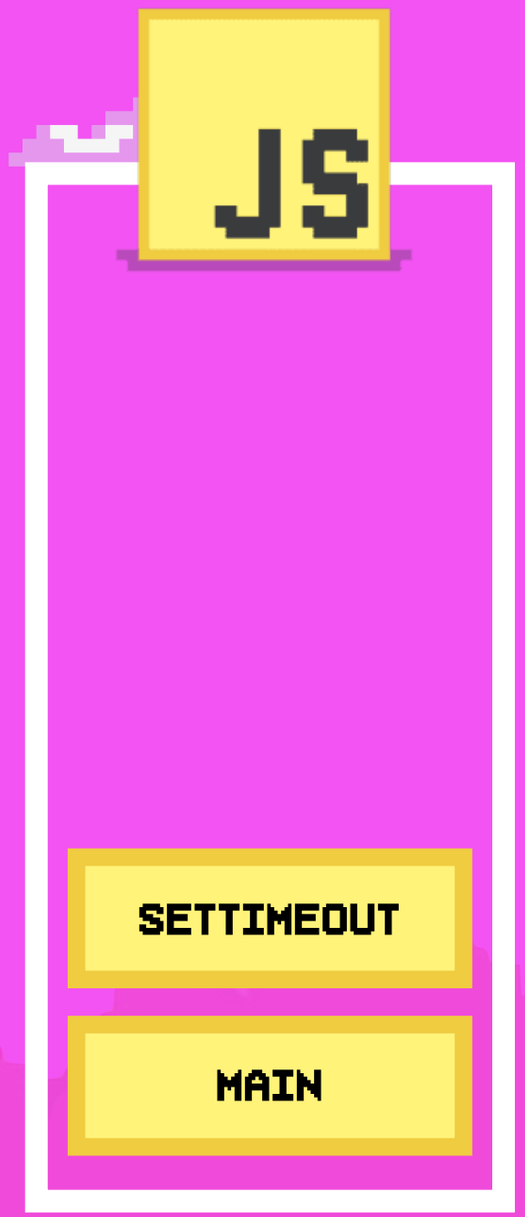
```
ONE
```




EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

```
ONE
```



```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

ONE

JS

MAIN

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

AJAX

CB

CALLBACK QUEUE

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

ONE

JS

MAIN

STACK

EVENT LOOP

WEB APIS

SETTIMEOUT

DOM

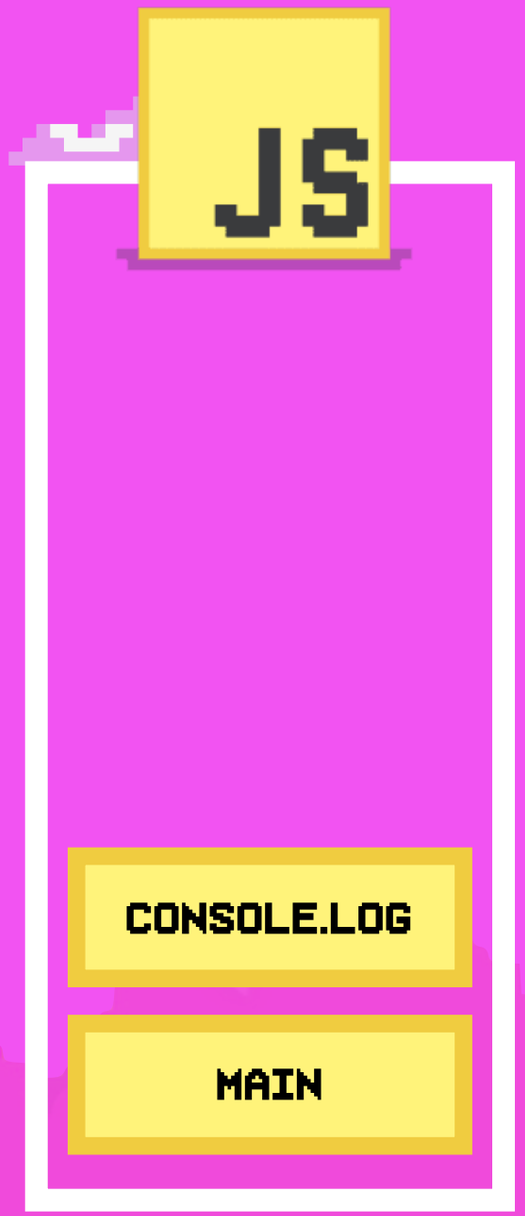
AJAX

CB

CALLBACK QUEUE

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

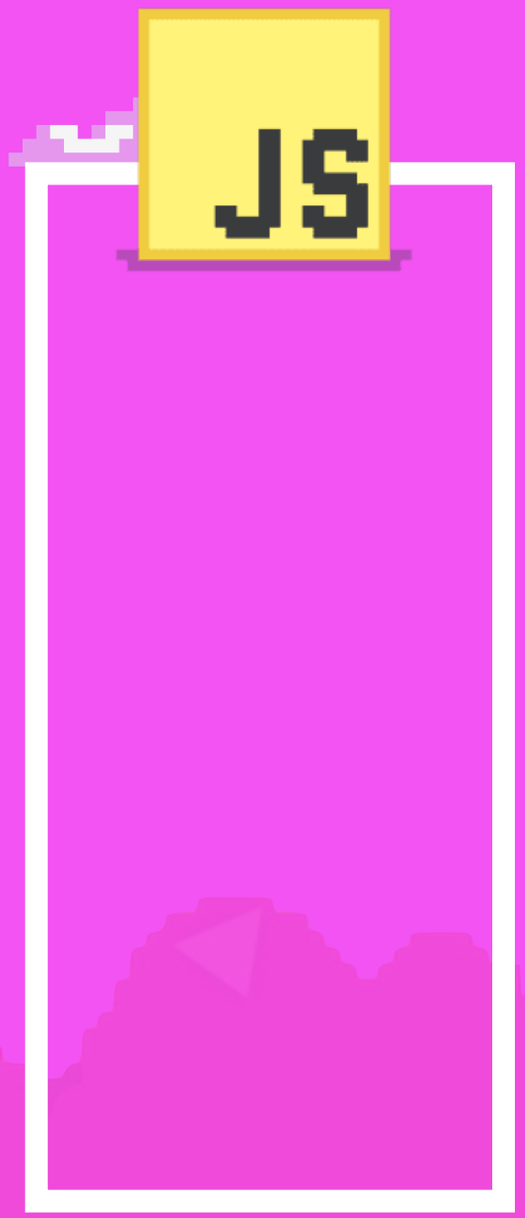
```
ONE
```




EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

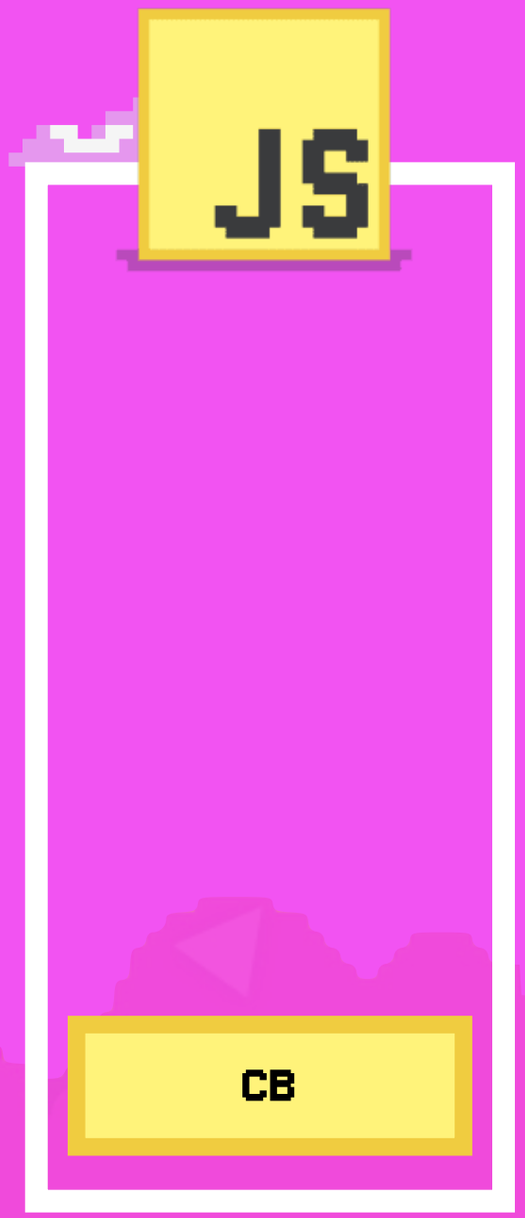
```
ONE
THREE
```




EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

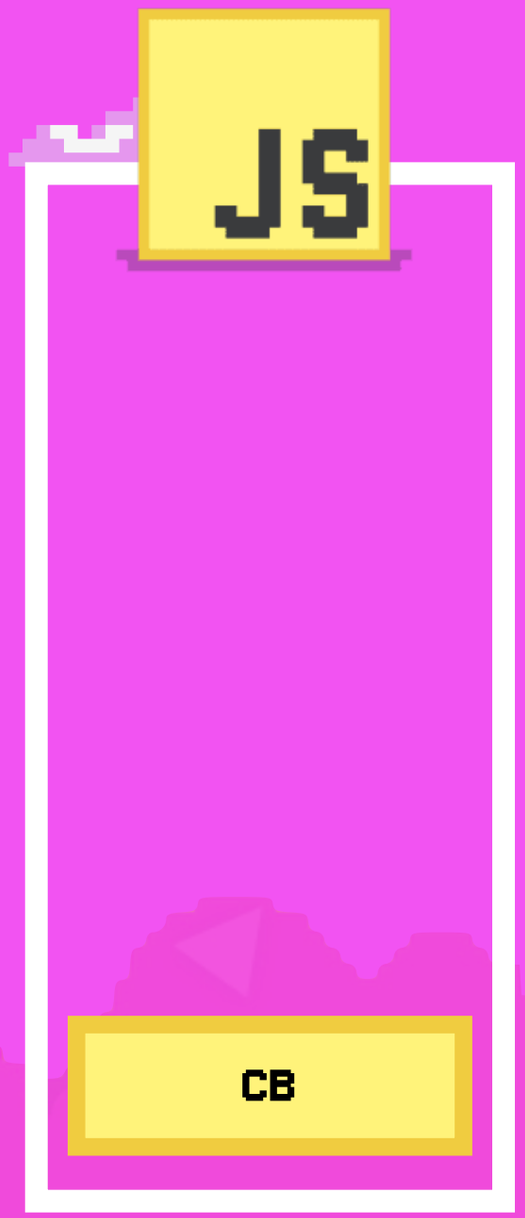
```
ONE
THREE
```




EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

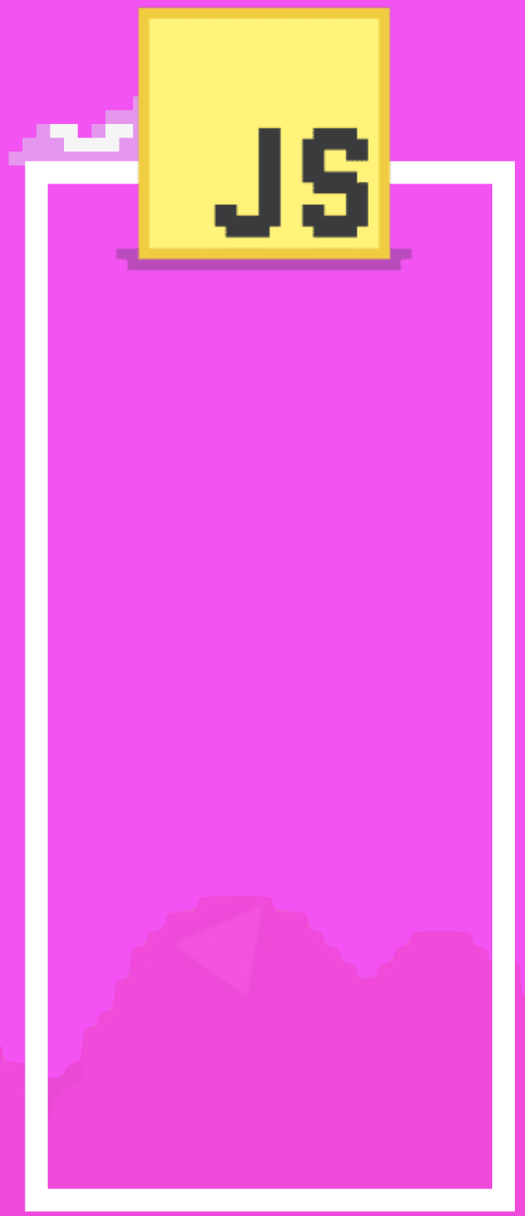
```
ONE
THREE
```



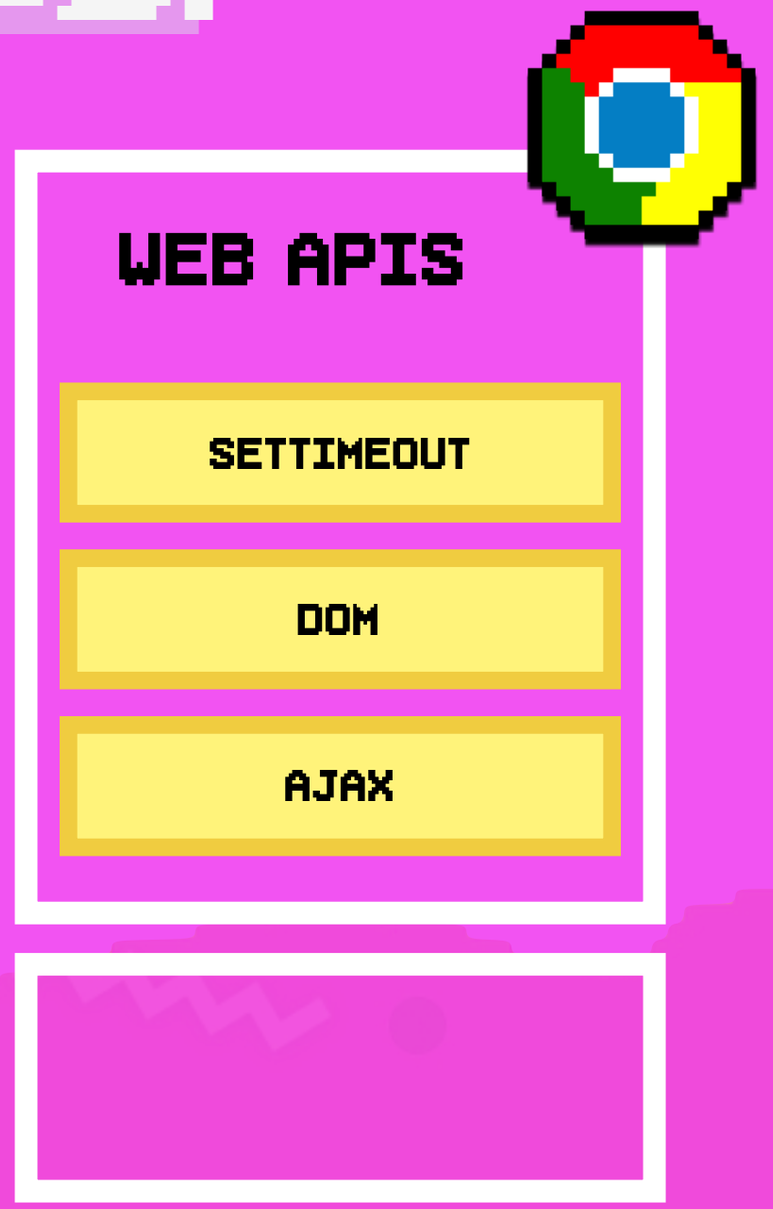

EVENT LOOP

```
console.log("One");

setTimeout(function () {
    console.log("Two")
}, 0);

console.log("Three");
```

```
ONE
THREE
TWO
```

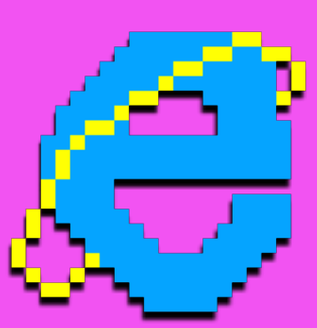



EVENT LOOP

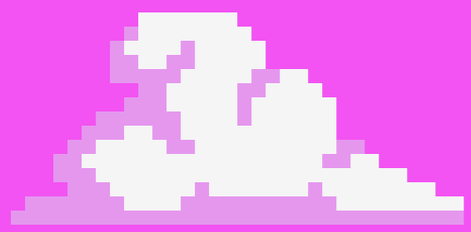


V8

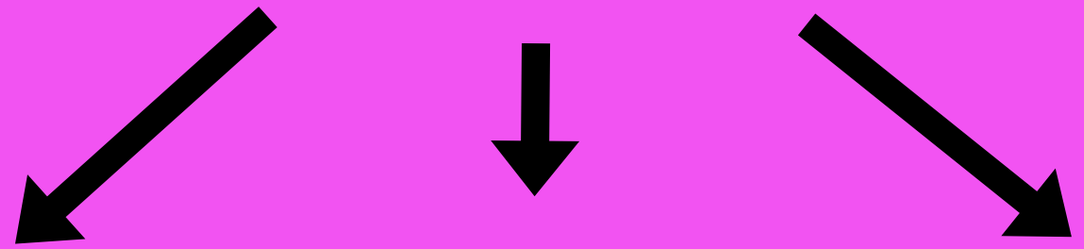
SPIDERMONKEY



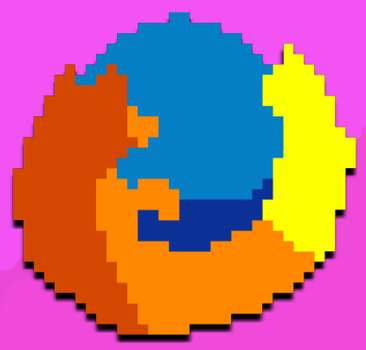
CHAKRA CORE



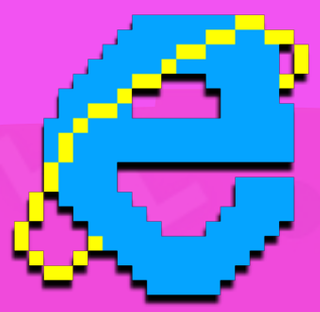
NODE.JS



V8

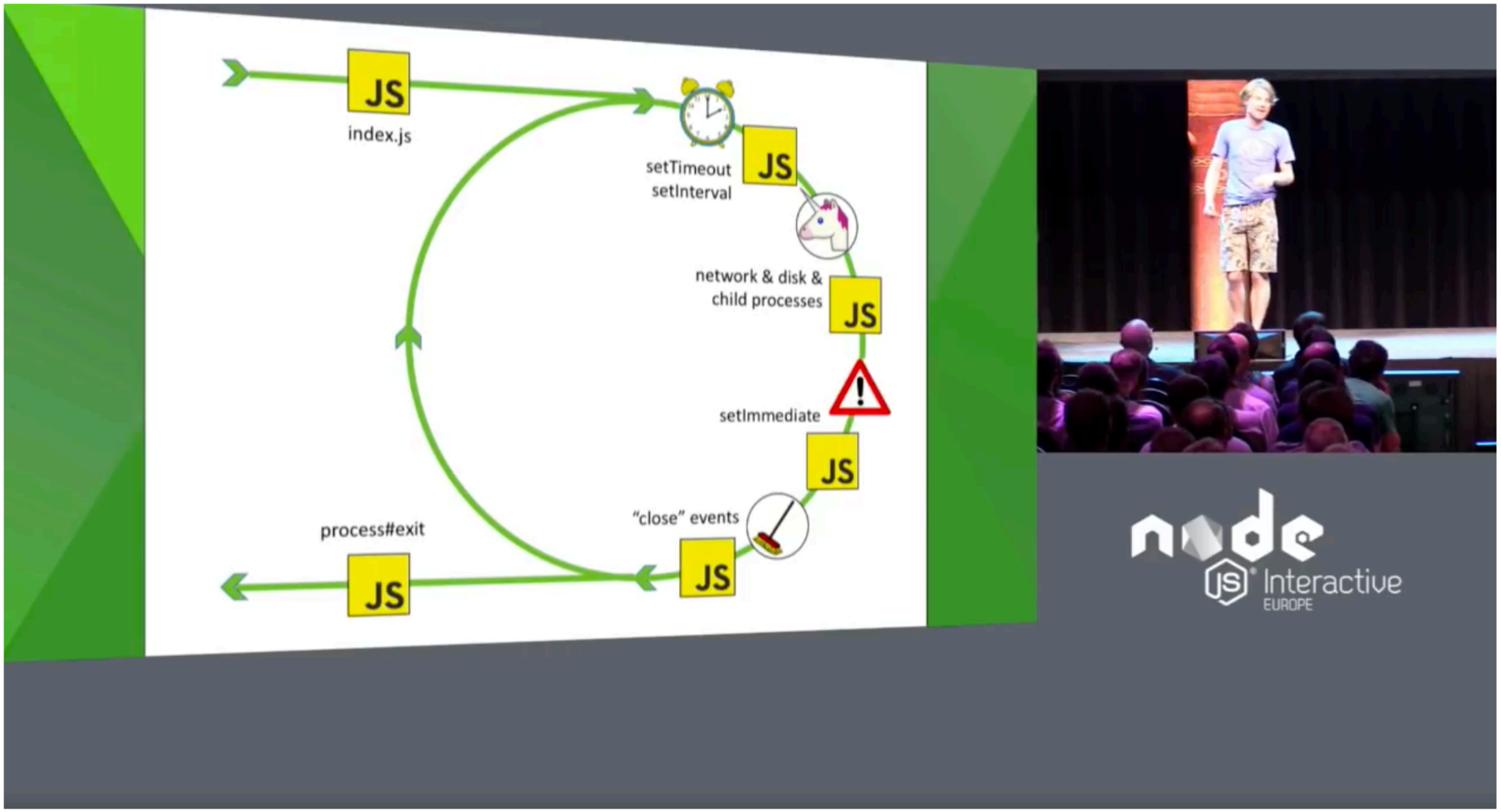


SPIDERMONKEY

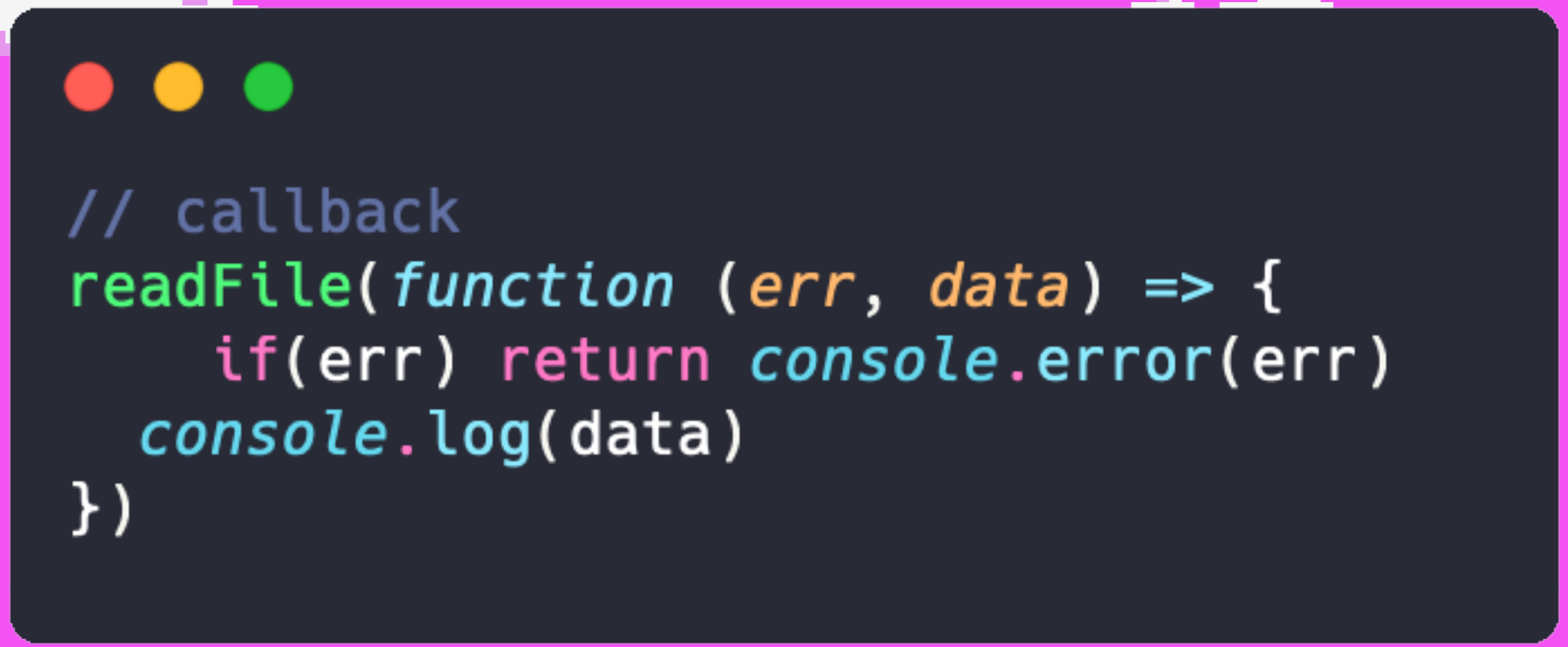


CHAKRA CORE

# libuv




# EVERYTHING YOU NEED TO KNOW ABOUT NODE.JS EVENT LOOP


// callback
readFile(function (err, data) => {
if(err) return console.error(err)
console.log(data)
})

```
// callback
readFile(function (err, data) => {
    if(err) return console.error(err)
  console.log(data)
})

// promise
let promise = readFile()
promise.then(console.log, console.error)
```

```
// callback
readFile(function (err, data) => {
    if(err) return console.error(err)
  console.log(data)
})

// promise
let promise = readFile()
promise.then(console.log, console.error)

// await
let data = await readFile()
console.log(contents)
```

# Rust

# What is Rust?

EMPOWERING EVERYONE TO BUILD
RELIABLE AND EFFICIENT SOFTWARE.

# What is Rust?

* SYSTEMS PROGRAMMING LANGUAGE
* STARTED AT MOZILLA (FIREFOX)
* USED BY AMAZON/MICROSOFT/DROPBOX...
* MAIN GOALS: SAFETY, PERFORMANCE
* FEARLESS CONCURRENCY

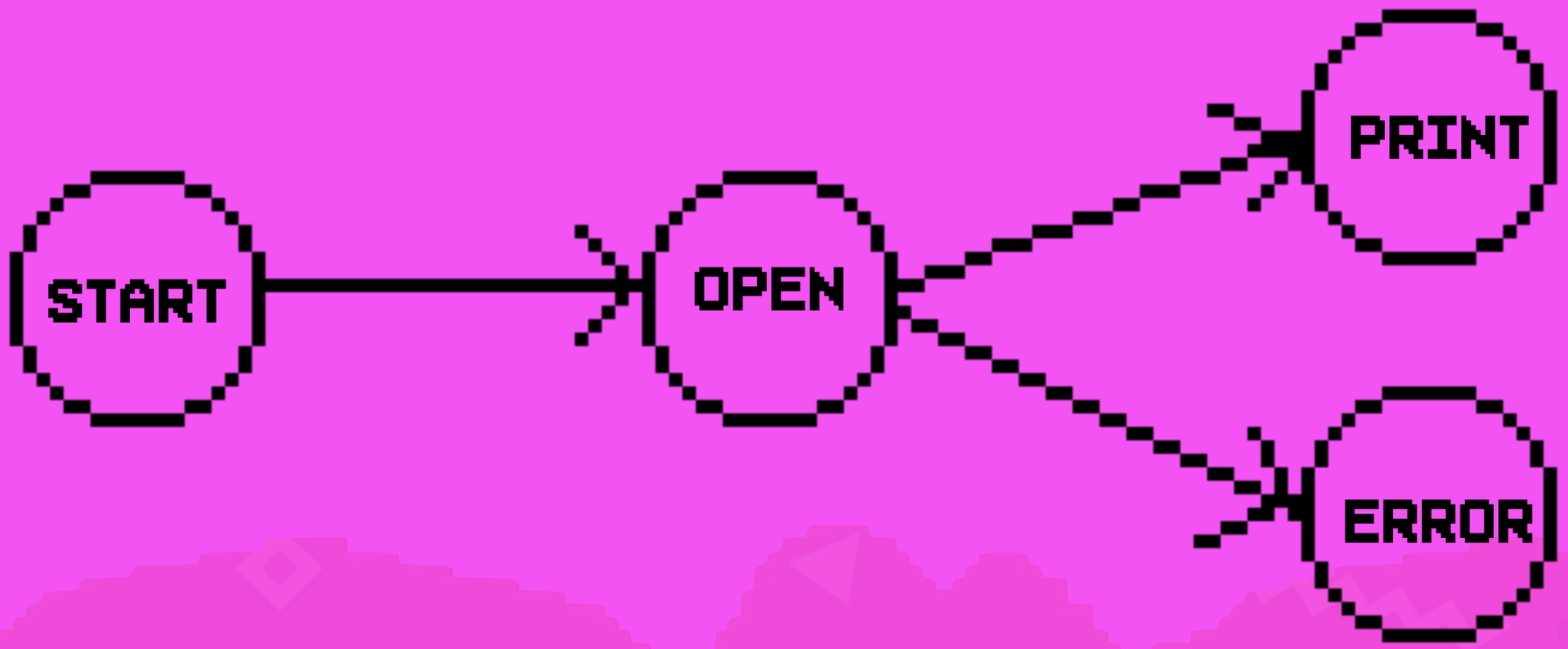
```
extern crate tokio;

use tokio::prelude::{AsyncRead, Future};

fn main() {
    let task = tokio::fs::File::open("foo.txt")
        .and_then(|mut file| {
            let mut contents = vec![];
            file.read_buf(&mut contents).map(|res| {
                println!("{:?}", res);
            })
        })
        .map_err(|err| eprintln!("IO error: {:?}", err));
    tokio::run(task);
}
```

```
extern crate tokio;

use tokio::prelude::{AsyncRead, Future};

                                                FUTURE
fn main() {
    let task = tokio::fs::File::open("foo.txt")
        .and_then(|mut file| {
            let mut contents = vec![];
            file.read_buf(&mut contents).map(|res| {
                println!("{:?}", res);
            })
        })
        .map_err(|err| eprintln!("IO error: {:?}", err));
    tokio::run(task);
}
```

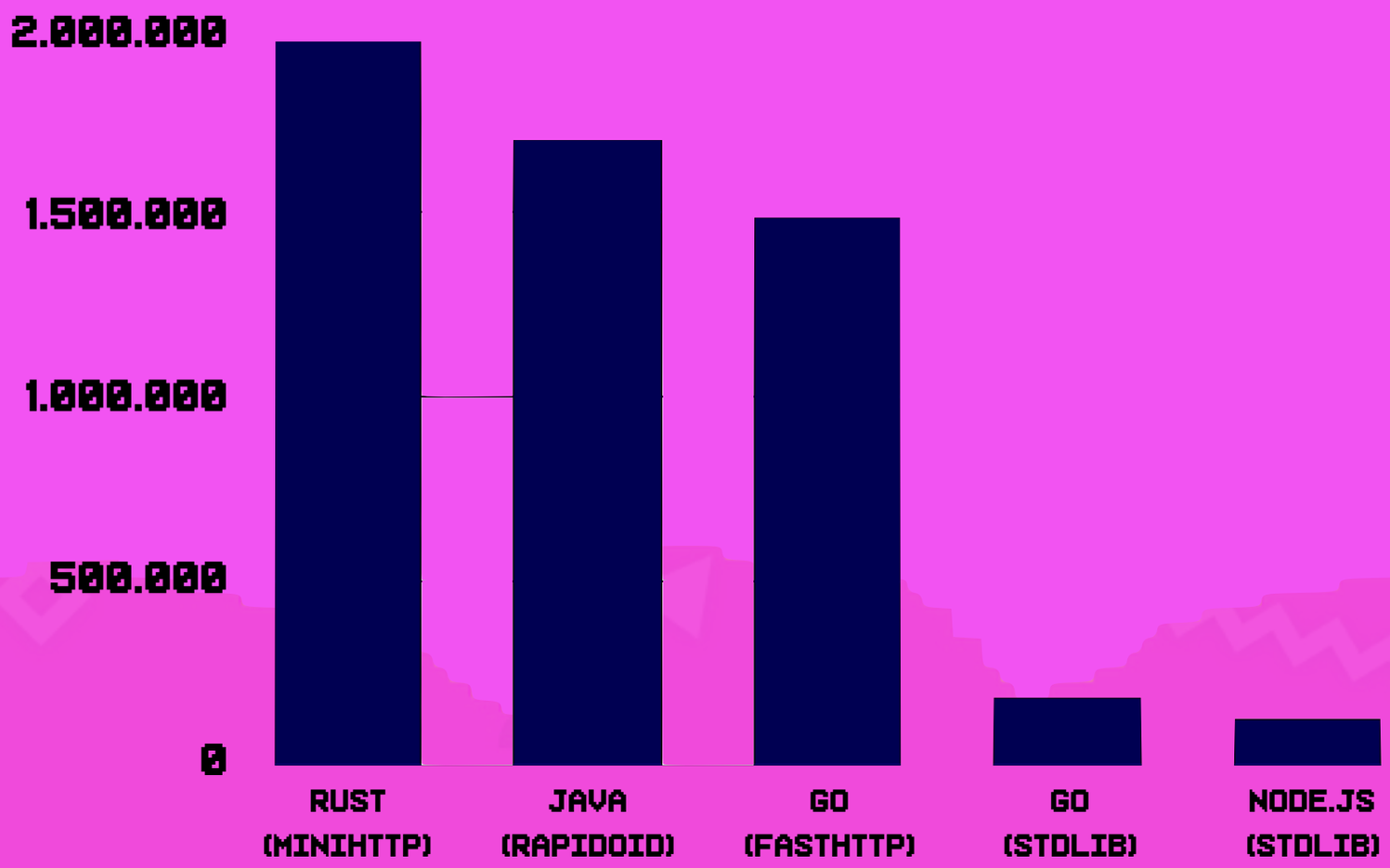
```
extern crate tokio;

use tokio::prelude::{AsyncRead, Future};

fn main() {
    let task = tokio::fs::File::open("foo.txt")
        .and_then(|mut file| {
            let mut contents = vec![];
            file.read_buf(&mut contents).map(|res| {
                println!("{:?}", res);
            })
        })
        .map_err(|err| eprintln!("IO error: {:?}", err));
    tokio::run(task);
}
```

```rust
extern crate tokio;

use tokio::prelude::{AsyncRead, Future};

fn main() {
    let task = tokio::fs::File::open("foo.txt")
        .and_then(|mut file| {
            let mut contents = vec![];
            file.read_buf(&mut contents).map(|res| {
                println!("{:?}", res);
            })
        })
        .map_err(|err| eprintln!("IO error: {:?}", err));
    tokio::run(task);
}
```

```rust
extern crate tokio;

use tokio::prelude::{AsyncRead, Future};

fn main() {
    let task = tokio::fs::File::open("foo.txt")
        .and_then(|mut file| {
            let mut contents = vec![];
            file.read_buf(&mut contents).map(|res| {
                println!("{:?}", res);
            })
        })
        .map_err(|err| eprintln!("IO error: {:?}", err));
    tokio::run(task);
}
```


START
OPEN
PRINT
ERROR

# requests/s


2.000.000
1.500.000
1.000.000
500.000
0
RUST (MINIHTTP)
JAVA (RAPIDOID)
GO (FASTHTTP)
GO (STDLIB)
NODE.JS (STDLIB)

```rust
let f = file::open("foo.txt").await?;
let mut buffer = Vec::new();
let contents = f.read_to_end(&mut buffer).await?;
println!("{:?}", contents);
```

# Futures in Rust

# Summary

* ASYNCIO IN PYTHON IS BASED ON COROUTINES
* ASYNC IN JAVASCRIPT IS BASED ON CALLBACKS
* RUST'S FUTURES ARE ZERO-COST

# Credits

* PIXEL BACKGROUNDS BY SLANCE.COM
* TRUCK AND ENGINE LOGOS BY PIXELART MAKER
* JAKE ARCHIBALD - IN THE LOOP