PUSHING WASM FROM THE BROWSER TO THE EDGE WITH SLEDGEHAMMER

BY THIJS FERYN

ConFoo.CA
WEB TECHNO CONFERENCE

PUSHING WASM FROM THE BROWSER TO THE EDGE WITH SLEDGEHAMMER

PUSHING WASM FROM THE BROWSER TO THE EDGE WITH SLEDGEHAMMER

PUSHING WASM FROM THE BROWSER TO THE EDGE WITH SLEDGEHAMMER

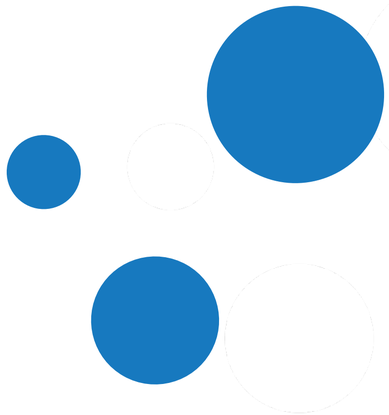
PUSHING WASM FROM THE BROWSER TO THE EDGE WITH SLEDGEHAMMER

HI,

I'M THIJS

I'M THE TECH EVANGELIST AT VARNISH SOFTWARE


VARNISH
SOFTWARE



VARNISH CACHE

VARNISH CACHE
SOFTWARE

About Varnish Software

We deliver enterprise solutions based on open source **Varnish Cache**

10M

22%

10M

10M active websites worldwide powered by **Varnish**

22% of the world's top 10,000 sites use **Varnish**

10M pulls on Docker hub of the **Varnish** image

2005 - Open source Varnish Cache project begins

2010 - Varnish Software spun out of Redpill Linpro

2014 - *Varnish Enterprise* solution launched to support content-heavy, high-traffic businesses

2017 - Varnish start delivering custom, private CDNs to enterprises

2020 - *Varnish Edge Cloud* launched to support 5G content delivery within telco networks

VARNISH SOFTWARE

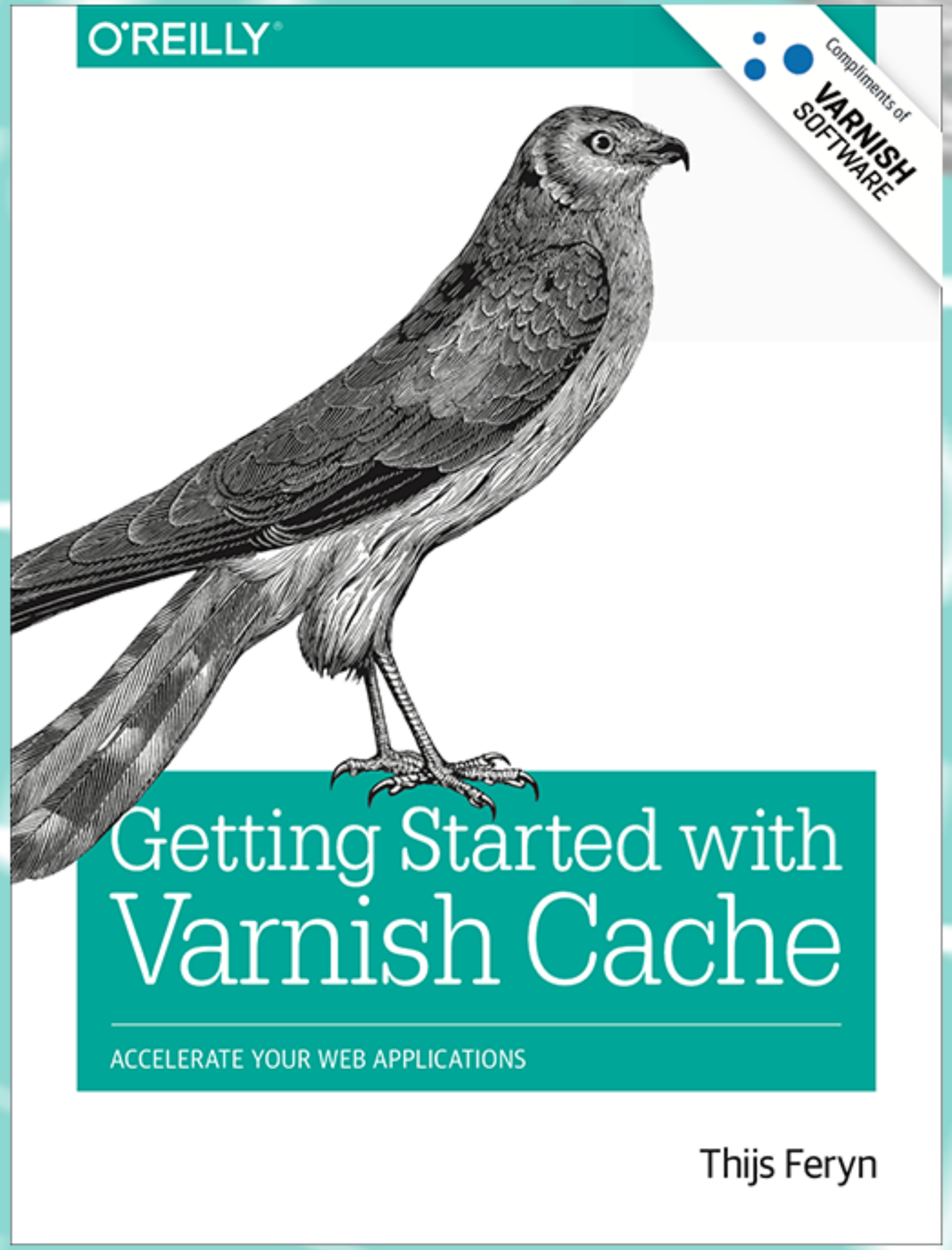
WE BUILD SOFTWARE-DEFINED WEB ACCELERATION & CONTENT DELIVERY SOLUTIONS

ACHIEVE GROWTH, PERFORMANCE & SUSTAINABILITY GOALS

1.3 Tbps
per server

1.17 Gbps
per watt

WORLD'S FASTEST EDGE CONTENT DELIVERY SOFTWARE


O'REILLY
Compliments of
VARNISH SOFTWARE
Getting Started with
Varnish Cache
ACCELERATE YOUR WEB APPLICATIONS
Thijs Feryn

Thijs Feryn

VARNISH 6

BY EXAMPLE

A practical guide to web acceleration and content delivery with Varnish 6 technology

WEBASSEMBLY

WEBASSEMBLY (ABBREVIATED WASM) IS A BINARY INSTRUCTION FORMAT FOR A STACK-BASED VIRTUAL MACHINE.

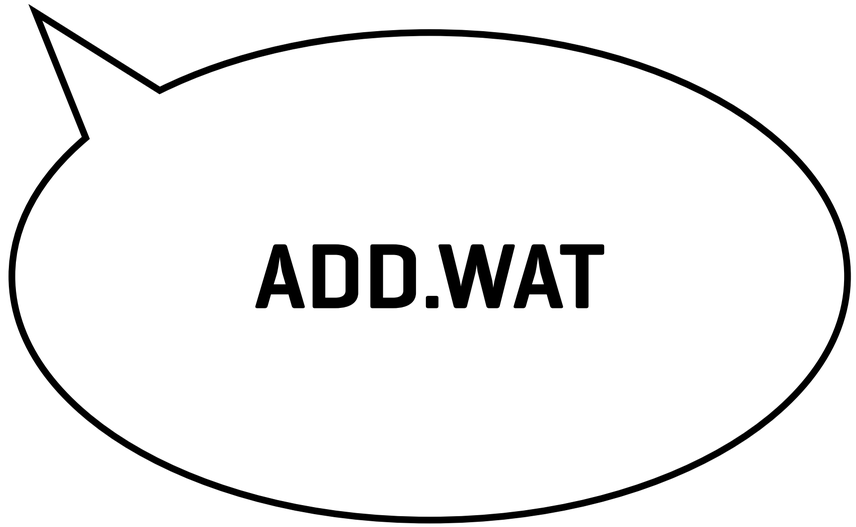
WASM IS DESIGNED AS A PORTABLE COMPILATION TARGET FOR PROGRAMMING LANGUAGES, ENABLING DEPLOYMENT ON THE WEB FOR CLIENT AND SERVER APPLICATIONS.

WEBASSEMBLY (ABBREVIATED WASM) IS A BINARY INSTRUCTION FORMAT FOR A STACK-BASED VIRTUAL MACHINE.

WASM IS DESIGNED AS A PORTABLE COMPILATION TARGET FOR PROGRAMMING LANGUAGES, ENABLING DEPLOYMENT ON THE WEB FOR CLIENT AND SERVER APPLICATIONS.

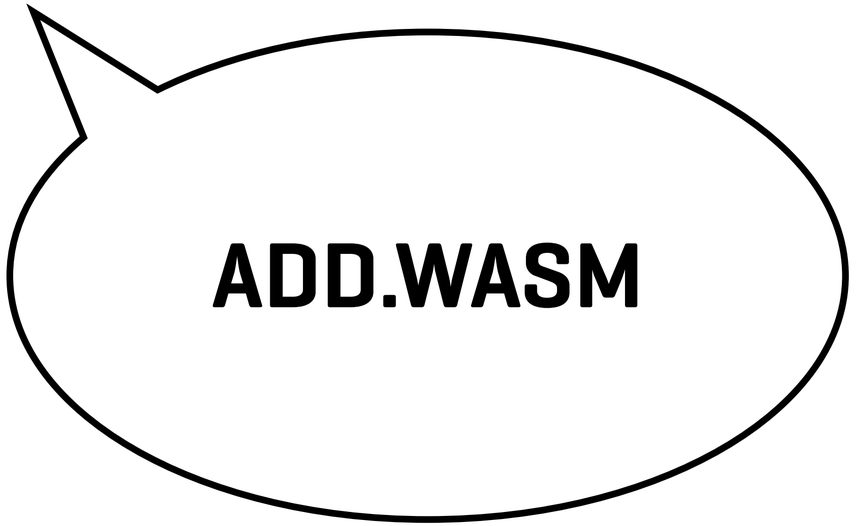
```
(module
  (func $add (param i32) (param i32) (result i32)
    local.get 0
    local.get 1
    i32.add)
  (export "add" (func $add))
)
```

```
(module
  (func $add (param i32) (param i32) (result i32)
    local.get 0
    local.get 1
    i32.add)
  (export "add" (func $add))
)
```



```
0000000 6100 6d73 0001 0000 0701 6001 7f02 017f
0000010 037f 0102 0700 0107 6103 6464 0000 090a
0000020 0701 2000 2000 6a01 000b
0000029
```

```
0000000 6100 6d73 0001 0000 0701 6001 7f02 017f
0000010 037f 0102 0700 0107 6103 6464 0000 090a
0000020 0701 2000 2000 6a01 000b
0000029
```



```
Code Disassembly:

000022 func[0] <add>:
 000023: 20 00                                  | local.get 0
 000025: 20 01                                  | local.get 1
 000027: 6a                                     | i32.add
 000028: 0b                                     | end
```

COMPILATION TARGET FOR OTHER LANGUAGES

WASM ADVANTAGES

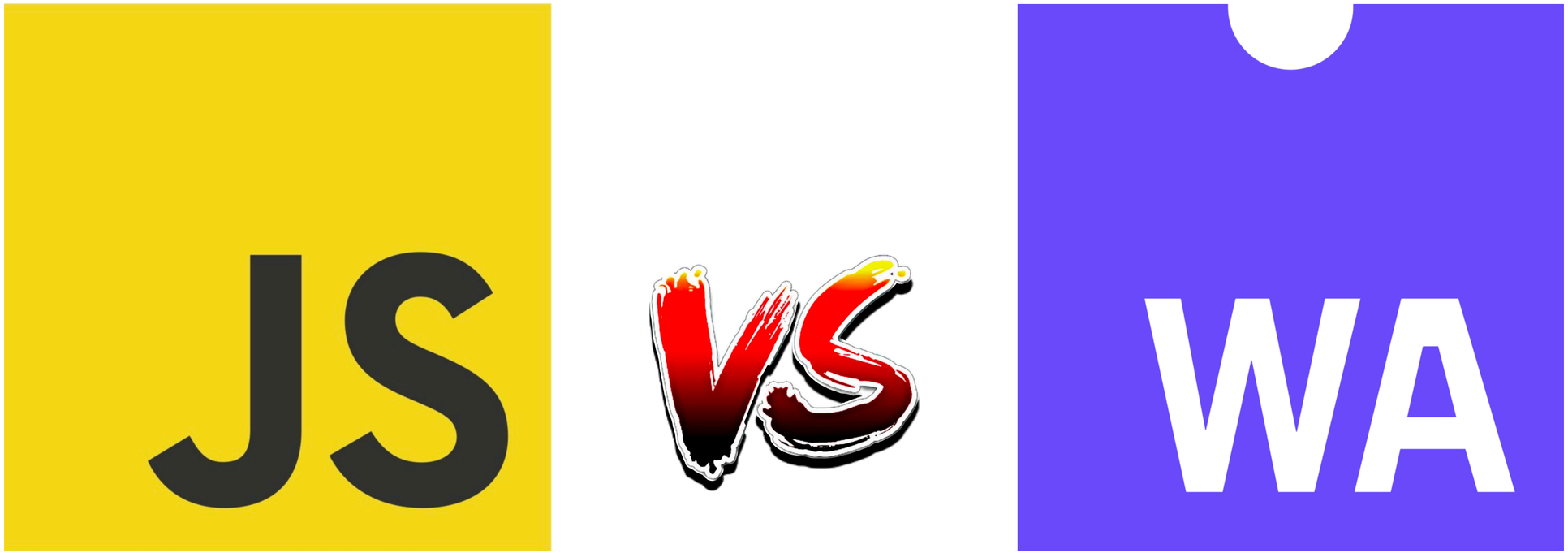
- ✓ LIGHTWEIGHT, EFFICIENT & FAST
- ✓ OPEN & DEBUGGABLE
- ✓ SANDBOXED & SAFE
- ✓ WELL-SUPPORTED

INITIALLY DESIGNED TO RUN IN BROWSERS




SUPPORTED BY THE MAIN BROWSERS

	Your browser	Chrome	Firefox	Safari	Wasmtime	Wasmer	Node.js	Deno	wasm2c
Standardized features									
JS BigInt to Wasm i64 integration	✓	85	78	14.1 [e]	N/A	N/A	15.0	1.1.2	N/A
Bulk memory operations	✓	75	79	15	0.20	1.0	12.5	0.4	1.0.30
Extended constant expressions	✓	114	112	✗	✗	✗	⚑ [h]	⚑ [o]	⚑ [v]
Garbage collection	✓	119	120	✗	✗	✗	⚑ [i]	✗	✗
Multiple memories	✓	120	⚑ [d]	✗	15	✗	✗	✗	⚑ [x]
Multi-value	✓	85	78	✓	0.17	1.0	15.0	1.3.2	1.0.24
Mutable globals	✓	74	61	12	✓	0.7	12.0	0.1	1.0.1
Reference types	✓	96	79	15	0.20	2.0	17.2	1.16	1.0.31
Relaxed SIMD	✓	114	⚑ [c]	✗	15	✗	⚑ [l]	⚑ [r]	✗
Non-trapping float-to-int conversions	✓	75	64	15	✓	✓	12.5	0.4	1.0.24
Sign-extension operations	✓	74	62	14.1 [e]	✓	✓	12.0	0.1	1.0.24
Fixed-width SIMD	✓	91	89	16.4	0.33	2.0	16.4	1.9	1.0.33
Tail calls	✓	112	121	✗	⚑ [g]	✗	⚑ [m]	⚑ [s]	⚑ [y]
Threads and atomics	✗	74	79	14.1 [e]	15	✗	16.4	1.9	✗
In-progress proposals									
Exception handling	✓	95	100	15.2	✗	✗	17.0	1.16	⚑ [u]
JS Promise Integration	✗	⚑ [a]	✗	✗	N/A	N/A	⚑ [j]	⚑ [p]	N/A
Memory64	✗	⚑ [b]	⚑ [c]	✗	⚑ [f]	✗	⚑ [k]	⚑ [q]	⚑ [w]
Type reflection	✗	⚑ [b]	⚑ [c]	✗	✗	2.0	⚑ [n]	⚑ [t]	✗


JS
VS
WA

```
<script>
WebAssembly.instantiateStreaming(fetch("add.wasm")).then(
  (obj) => {
    console.log(obj.instance.exports.add(3,5));
  },
);
</script>
```

```
(module
  (func $add (param i32) (param i32) (result i32)
    local.get 0
    local.get 1
    i32.add)
  (export "add" (func $add))
)
```

```
(module
  (func $add (param i32) (param i32) (result i32)
    local.get 0
    local.get 1
    i32.add)
  (export "add" (func $add))
)
```

```
(module
  (func $i (import "imports" "i") (param i32))
  (func $add (param i32) (param i32) (result i32)
(local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```

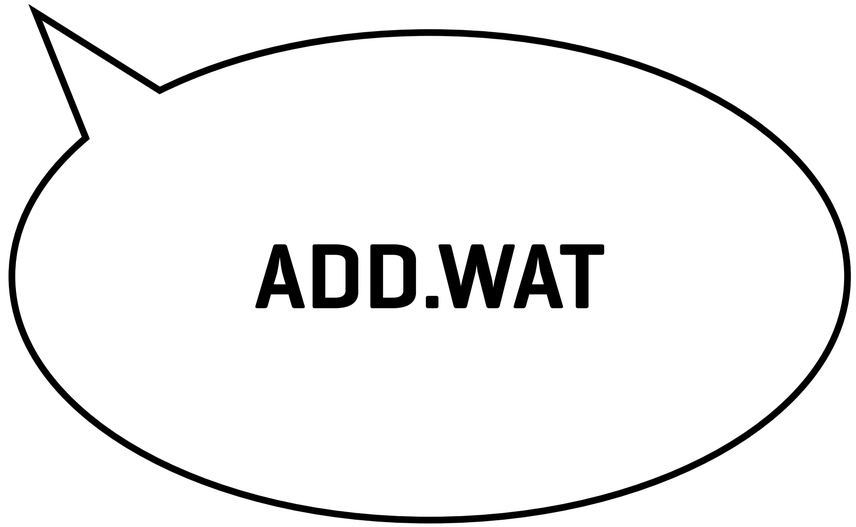
```
(module
  (func $i (import "imports" "i") (param i32))
  (func $add (param i32) (param i32) (result i32)
(local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```

```
<script>
var importObject = { imports: { i: arg => alert(arg) } };

WebAssembly.instantiateStreaming(fetch("add.wasm"),
importObject).then(
  (obj) => {
    console.log(obj.instance.exports.add(3,5));
  },
);
</script>
```

```
<script>
var importObject = { imports: { i: arg => alert(arg) } };

WebAssembly.instantiateStreaming(fetch("add.wasm"),
importObject).then(
  (obj) => {
    console.log(obj.instance.exports.add(3,5));
  },
);
</script>
```

```
(module
  (func $i (import "imports" "i") (param i32))
  (func $add (param i32) (param i32) (result i32)
(local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```

```
(module
  (func $i (import "imports" "i") (param i32))
  (func $add (param i32) (param i32) (result i32)
(local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```



```
wat2wasm add.wat -o add.wasm
```

WABT: The WebAssembly Binary Toolkit

WABT (we pronounce it "wabbit") is a suite of tools for WebAssembly, including:

- **wat2wasm**: translate from WebAssembly text format to the WebAssembly binary format
- **wasm2wat**: the inverse of wat2wasm, translate from the binary format back to the text format (also known as a .wat)
- **wasm-objdump**: print information about a wasm binary. Similiar to objdump.
- **wasm-interp**: decode and run a WebAssembly binary file using a stack-based interpreter
- **wasm-decompile**: decompile a wasm binary into readable C-like syntax.
- **wat-desugar**: parse .wat text form as supported by the spec interpreter (s-expressions, flat syntax, or mixed) and print "canonical" flat format
- **wasm2c**: convert a WebAssembly binary file to a C source and header
- **wasm-strip**: remove sections of a WebAssembly binary file
- **wasm-validate**: validate a file in the WebAssembly binary format
- **wast2json**: convert a file in the wasm spec test format to a JSON file and associated wasm binary files
- **wasm-stats**: output stats for a module
- **spectest-interp**: read a Spectest JSON file, and run its tests in the interpreter

These tools are intended for use in (or for development of) toolchains or other systems that want to manipulate WebAssembly files. Unlike the WebAssembly spec interpreter (which is written to be as simple, declarative and "speccy" as possible), they are written in C/C++ and designed for easier integration into other systems. Unlike Binaryen these tools do not aim to provide an optimization platform or a higher-level compiler target; instead they aim for full fidelity and compliance with the spec (e.g. 1:1 round-trips with no changes to instructions).

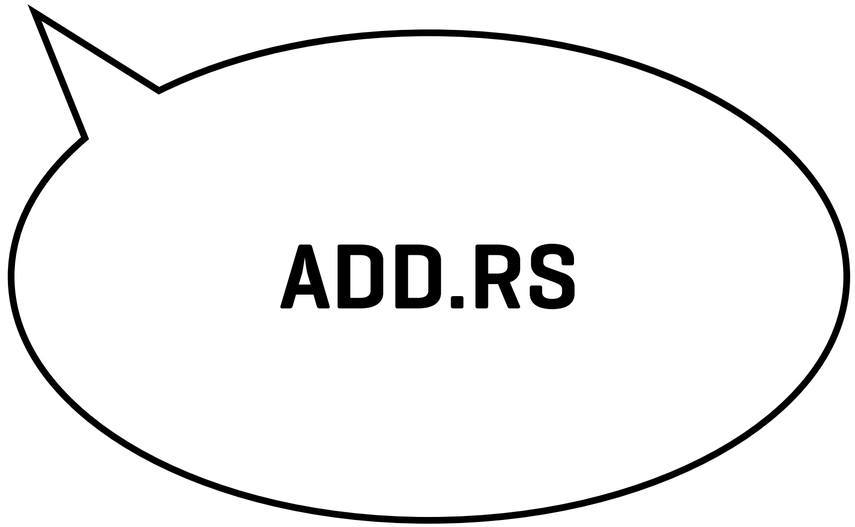
```
$ wasm-decompile add.wasm

import function imports_i(a:int);

export function add(a:int, b:int):int {
  var c:int = a + b;
  imports_i(c);
  return c;
}
```

```
(module
  (func $i (import "imports" "i")
(param i32))
  (func $add (param i32)
  (param i32) (result i32)
  (local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```

```
import function imports_i(a:int);

export function add(a:int, b:int):int
{
  var c:int = a + b;
  imports_i(c);
  return c;
}
```

COMPILATION TARGET FOR OTHER LANGUAGES

```rust
fn main() {}

#[export_name = "add"]
pub extern "C" fn add(left: i32, right: i32) -> i32 {
    left + right
}
```

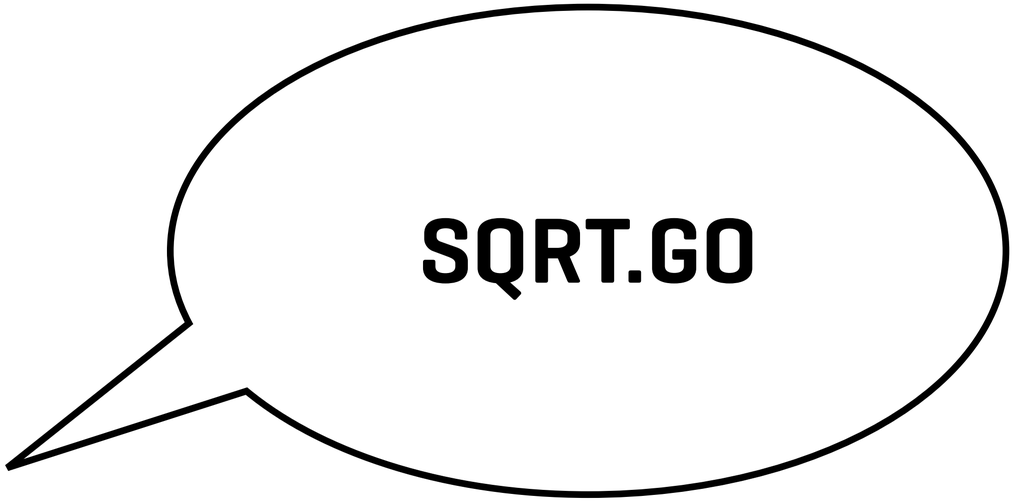


Rust

```
rustup target add wasm32-unknown-unknown
rustc add.rs --target wasm32-unknown-unknown
```

```
<script>
WebAssembly.instantiateStreaming(fetch("add.wasm")).then(
  (obj) => {
    console.log(obj.instance.exports.add(3,5));
  },
);
</script>
```

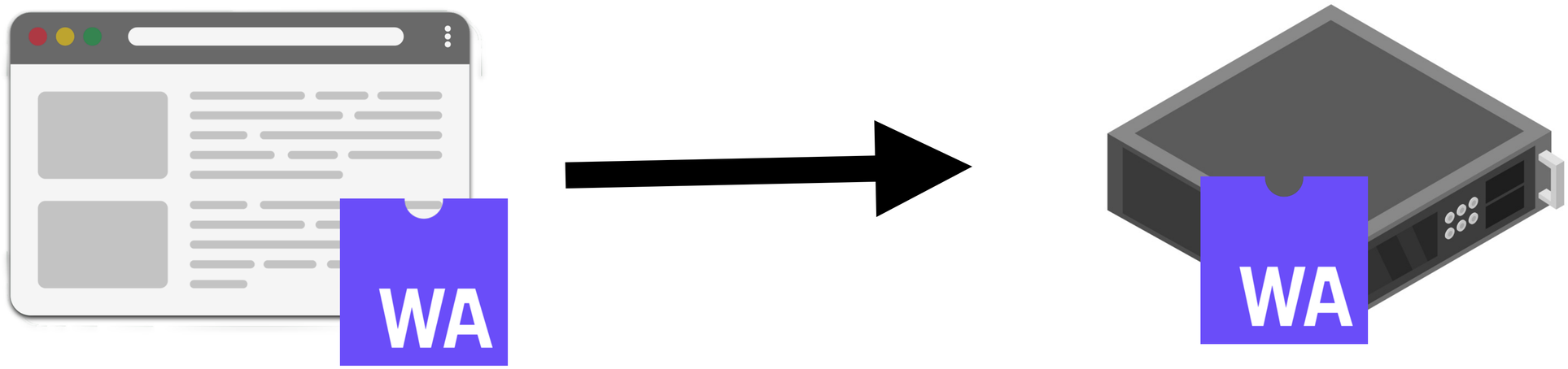
```go
package main

import (
    "math"
)

func main() {}

//export sqrt
func sqrt(number float64) float64 {
    return math.Sqrt(number)
}
```



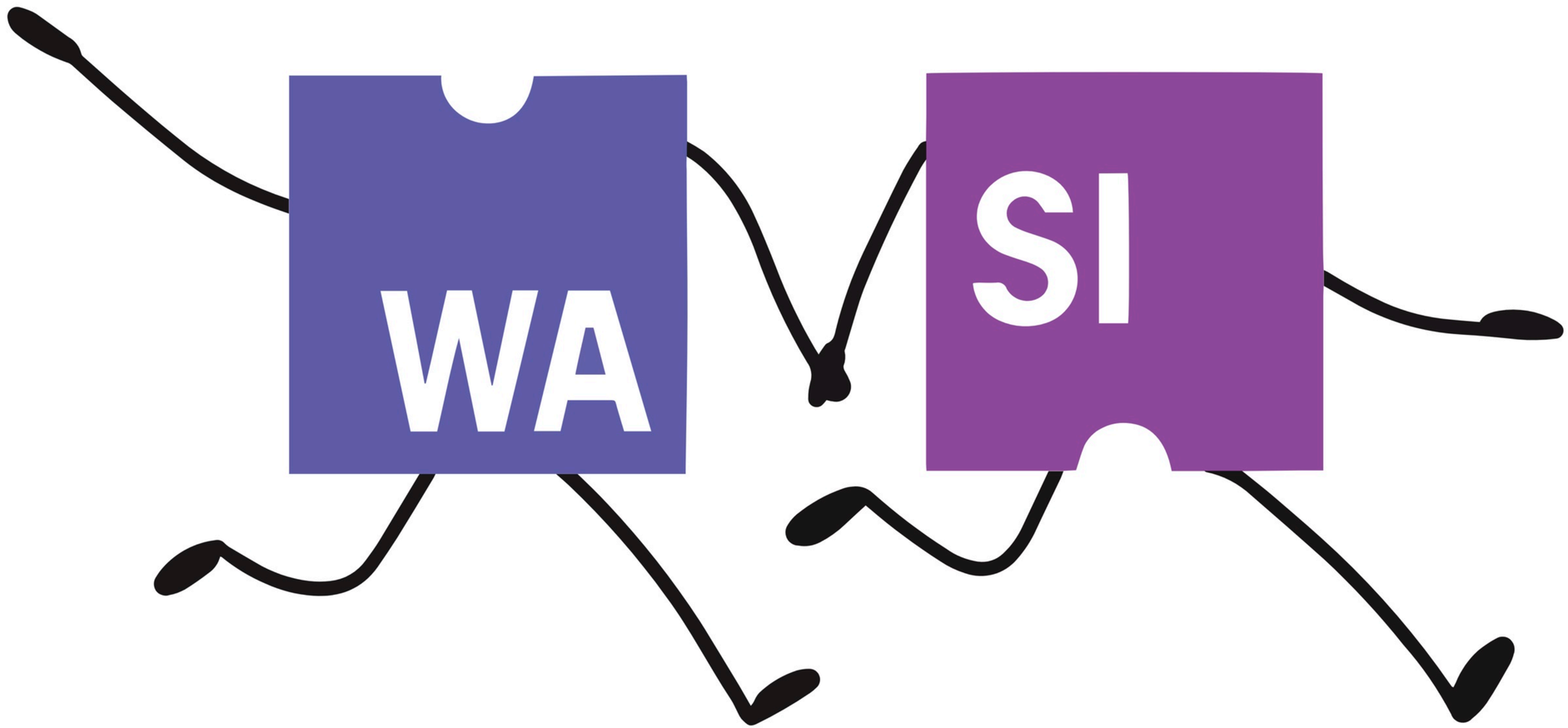
```
tinygo build -target=wasm sqrt.go
```

```
<script src="wasm_exec.js"></script>
<script>
  const go = new Go();
  WebAssembly.instantiateStreaming(fetch("sqrt.wasm"),
  go.importObject).then(
  (obj) => {
    wasm = obj.instance;
    go.run(wasm);
    console.log(obj.instance.exports.sqrt(25));
  },
);
</script>
```

RUNNING WASM ON THE SERVER



THE WEBASSEMBLY SYSTEM INTERFACE

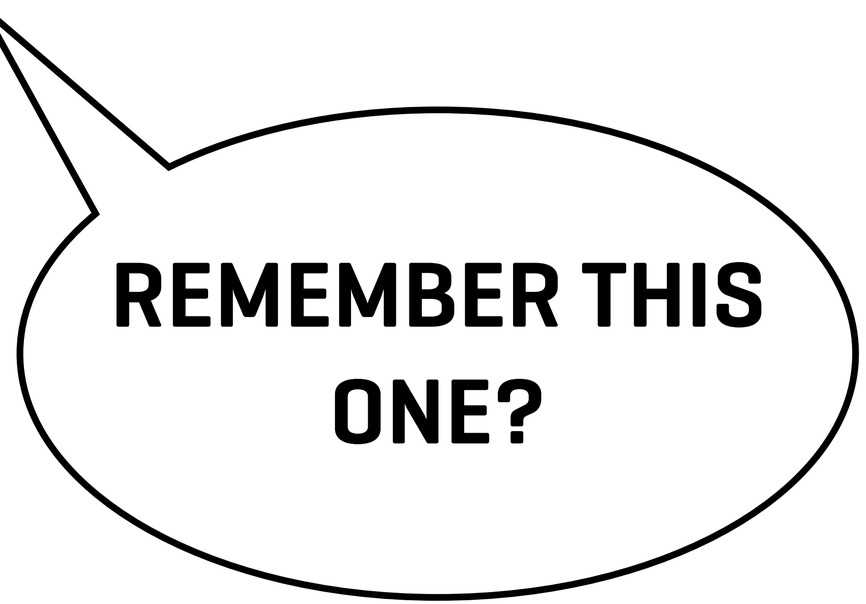


WASMTIME

```
fn main() {
    println!("Hello, world!");
}
```

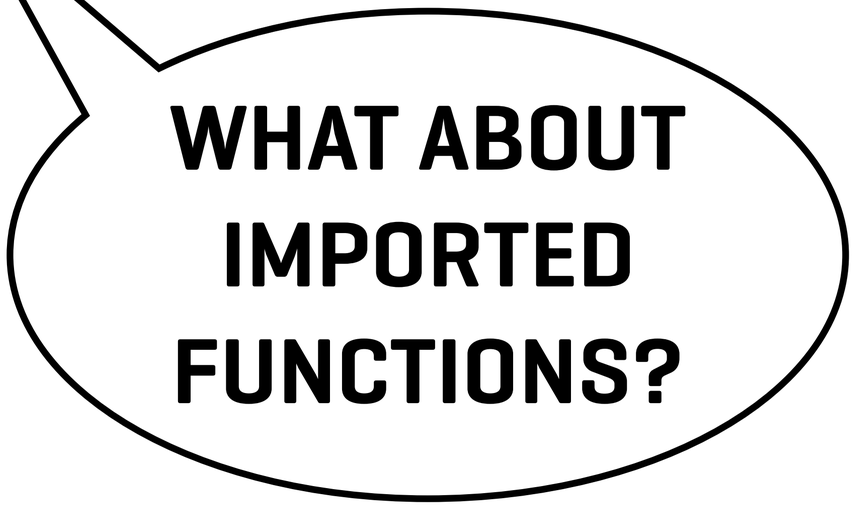
```
rustup target add wasm32-wasi
rustc hello.rs --target wasm32-wasi
wasmtime hello.wasm

Hello, world!
```

```
(module
  (func $add (param i32) (param i32) (result i32)
    local.get 0
    local.get 1
    i32.add)
  (export "add" (func $add))
)
```



```
wasmtime --invoke=add add.wasm 12 3
15
```

```
(module
  (func $i (import "imports" "i") (param i32))
  (func $add (param i32) (param i32) (result i32)
(local i32)
    local.get 0
    local.get 1
    i32.add
    local.tee 2
    call $i
    local.get 2)
  (export "add" (func $add))
)
```



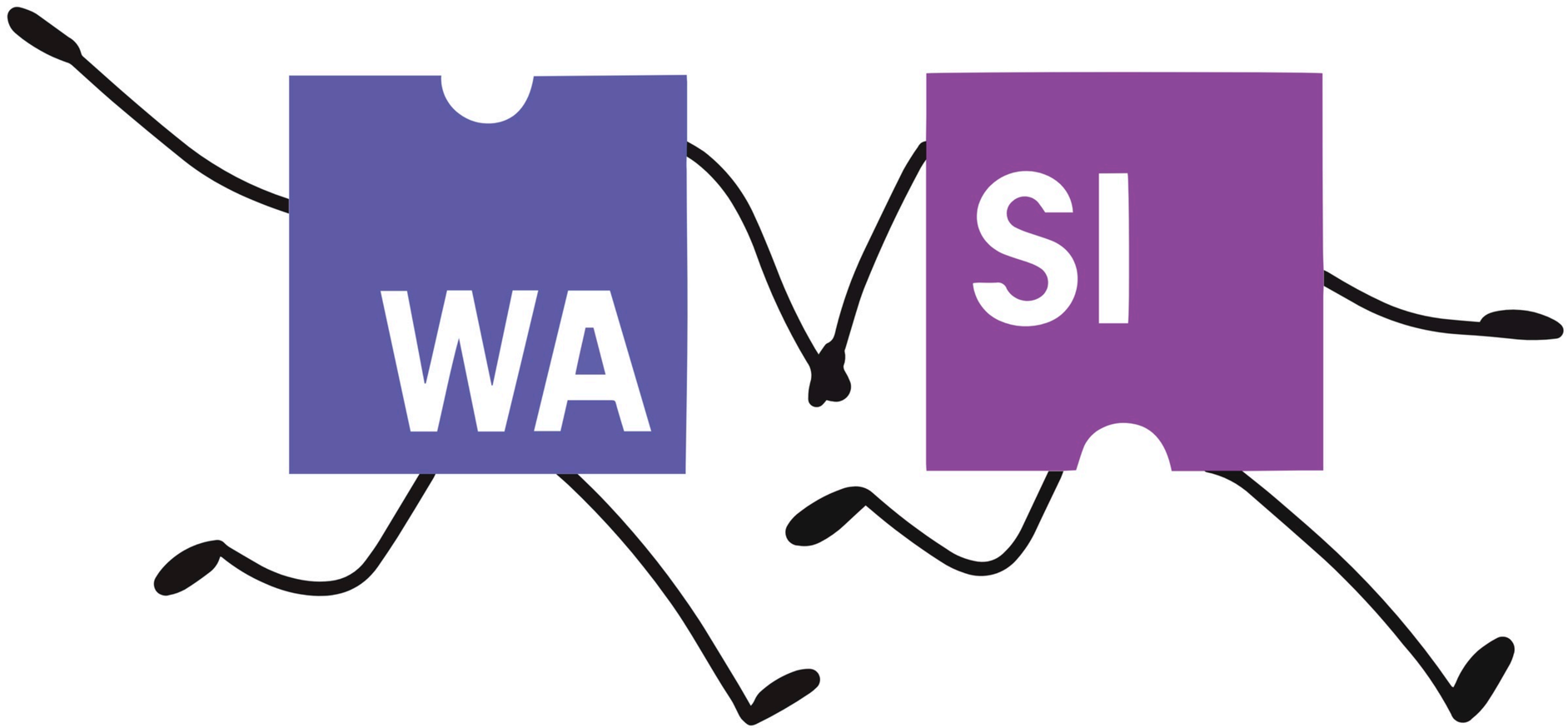
```
wasmtime --invoke=add add.wasm 12 3

Error: failed to run main module `add.wasm`

Caused by:
    0: failed to instantiate "add.wasm"
    1: unknown import: `imports::i` has not
been defined
```

```
wasmtime -W unknown-imports-default=y \
--invoke=add add-import.wasm 12 3
```

WASMTIME API

```rust
use anyhow::Result;
use wasmtime::*;

fn main() -> Result<()> {
    let engine = Engine::default();
    let wat = r#"
        (module
            (import "host" "host_func" (func $host_hello (param i32)))

            (func (export "hello")
                i32.const 3
                call $host_hello)
        )
    "#;
    let module = Module::new(&engine, wat)?;

    let mut linker = Linker::new(&engine);
    linker.func_wrap("host", "host_func", |caller: Caller<'_, u32>, param: i32| {
        println!("Got {} from WebAssembly", param);
        println!("my host state is: {}", caller.data());
    })?;

    let mut store = Store::new(&engine, 4);
    let instance = linker.instantiate(&mut store, &module)?;
    let hello = instance.get_typed_func::<(), ()>(&mut store, "hello")?;

    hello.call(&mut store, ())?;

    Ok(())
}
```

THE WASI API



```rust
use std::env;
use std::fs;
use std::io::{Read, Write};

fn process(input_fname: &str, output_fname: &str) -> Result<(), String> {
    let mut input_file =
        fs::File::open(input_fname).map_err(|err| format!("error opening input {}: {}",
input_fname, err))?;
    let mut contents = Vec::new();
    input_file
        .read_to_end(&mut contents)
        .map_err(|err| format!("read error: {}", err))?;

    let mut output_file = fs::File::create(output_fname)
        .map_err(|err| format!("error opening output {}: {}", output_fname, err))?;
    output_file
        .write_all(&contents)
        .map_err(|err| format!("write error: {}", err))
}

fn main() {
    let args: Vec<String> = env::args().collect();
    let program = args[0].clone();
    if args.len() < 3 {
        eprintln!("usage: {} <from> <to>", program);
        return;
    }
    if let Err(err) = process(&args[1], &args[2]) {
        eprintln!("{}", err)
    }
}
```

```rust
use std::env;
use std::fs;
use std::io::{Read, Write};

fn process(input_fname: &str, output_fname: &str) -> Result<(), String> {
    let mut input_file =
        fs::File::open(input_fname).map_err(|err| format!("error opening input {}: {}",
input_fname, err))?;
    let mut contents = Vec::new();
    input_file
        .read_to_end(&mut contents)
        .map_err(|err| format!("read error: {}", err))?;

    let mut output_file = fs::File::create(output_fname)
        .map_err(|err| format!("error opening output {}: {}", output_fname, err))?;
    output_file
        .write_all(&contents)
        .map_err(|err| format!("write error: {}", err))
}

fn main() {
    let args: Vec<String> = env::args().collect();
    let program = args[0].clone();
    if args.len() < 3 {
        eprintln!("usage: {} <from> <to>", program);
        return;
    }
    if let Err(err) = process(&args[1], &args[2]) {
        eprintln!("{}", err)
    }
}
```

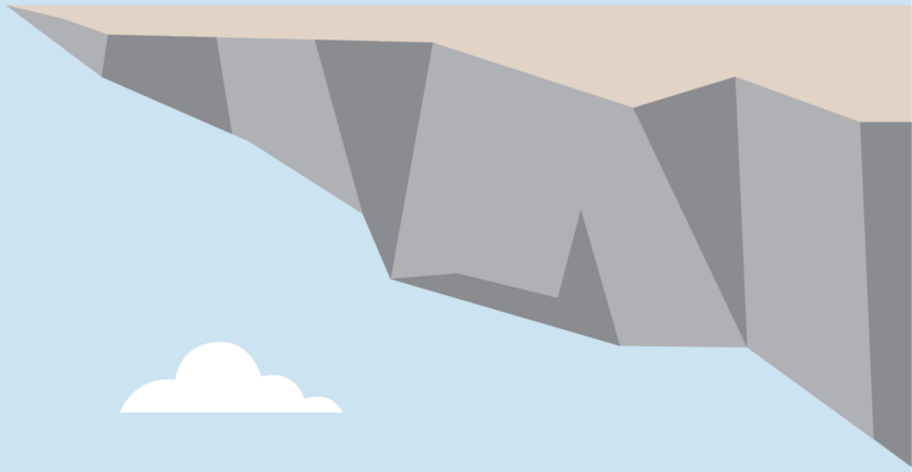
```
rustc copy.rs --target wasm32-wasi

echo "yo" > one.txt

wasmtime copy.wasm one.txt two.txt
error opening input one.txt: failed to find
a pre-opened file descriptor through which
"one.txt" could be opened

wasmtime --dir=. copy.wasm one.txt two.txt
```

WEB ASSEMBLY IS AN INSTRUCTION SET.

EXTRA CAPABILITIES COME FROM THE BROWSER OR FROM WASI.

RUNNING WASM ON TH EDGE

EDGE?



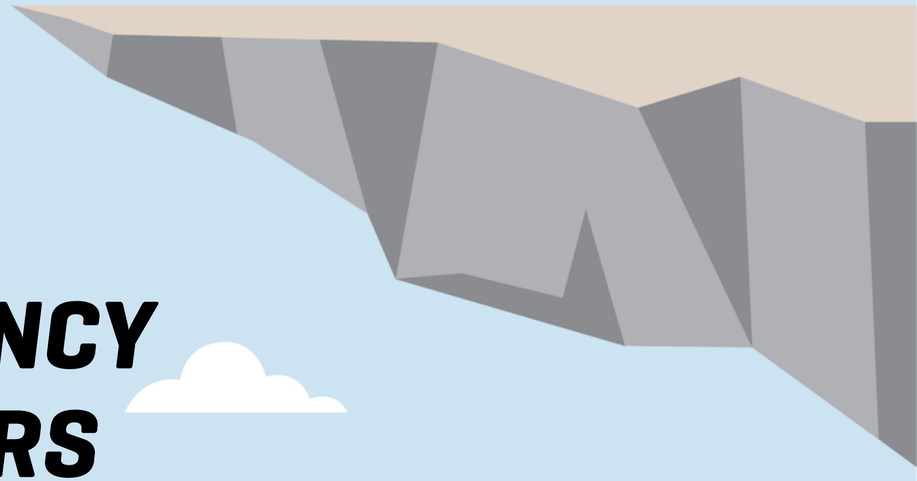
EDGE?

THE OUTER TIER OF YOUR WEB PLATFORM THAT RECEIVES ALL THE USER REQUESTS

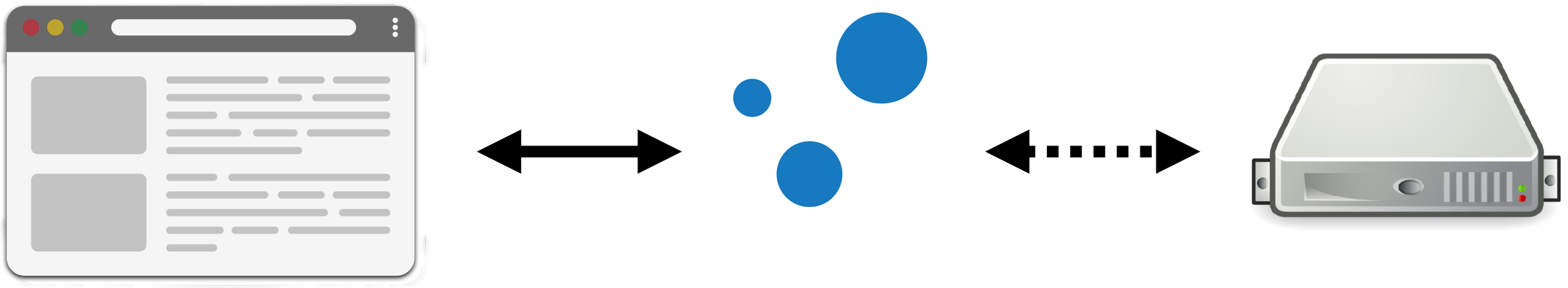
TYPICALLY A CACHE OR CDN

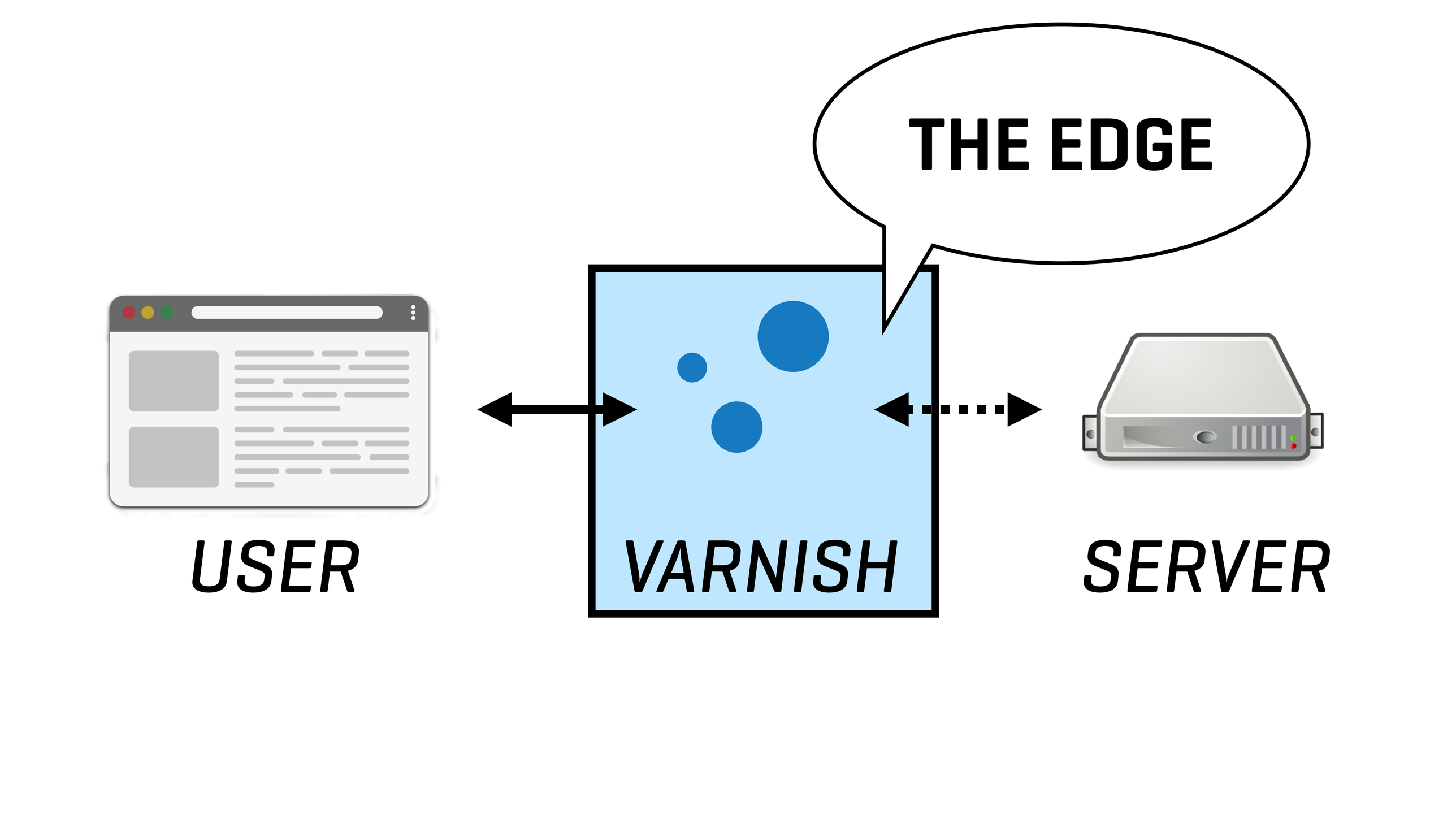
NOT NECESSARILY IN THE SAME PLACE AS THE WEB SERVER.

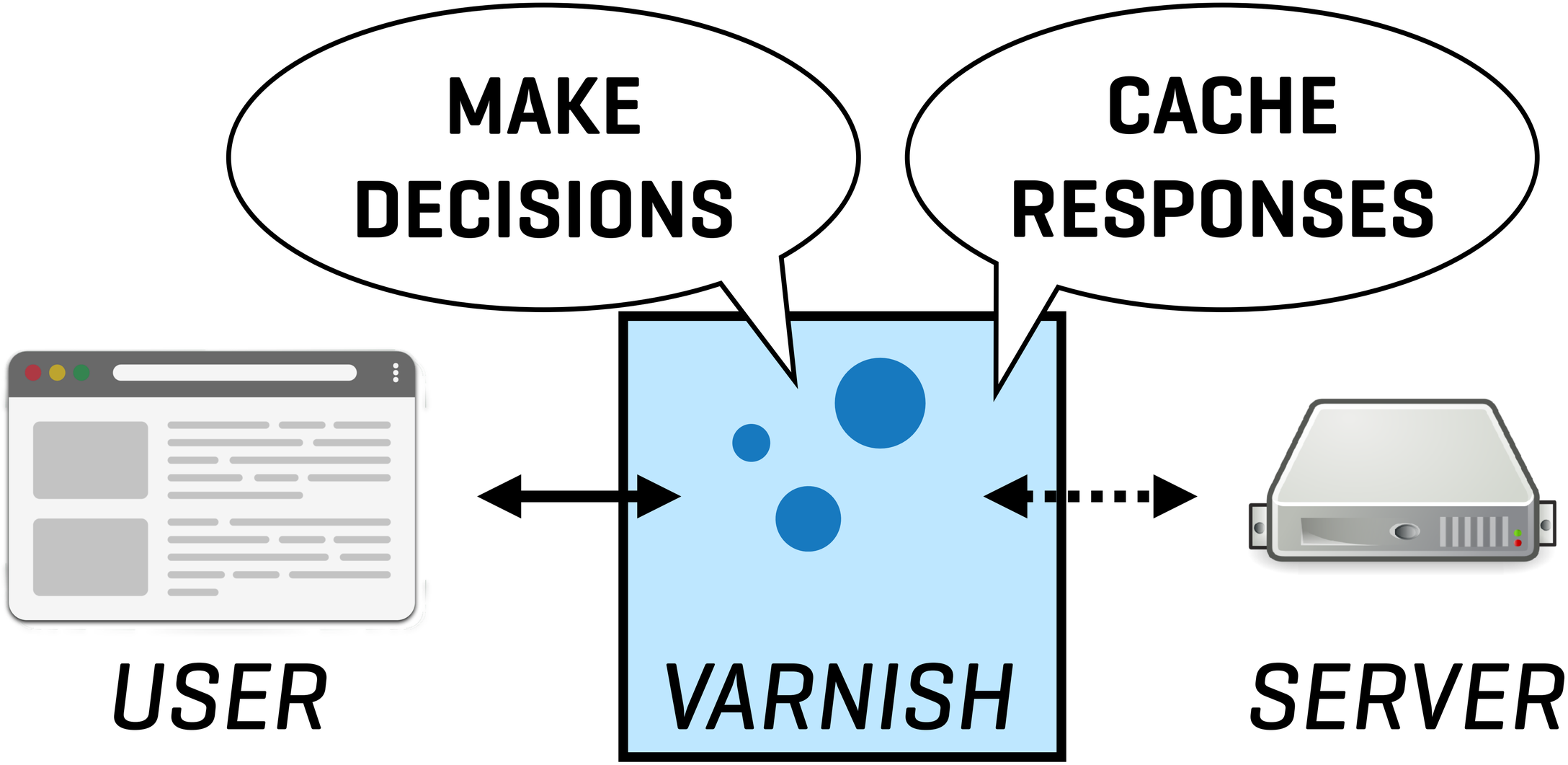
CLOSER TO THE USER TO REDUCE LATENCY AND OFFLOAD THE NETWORK & SERVERS

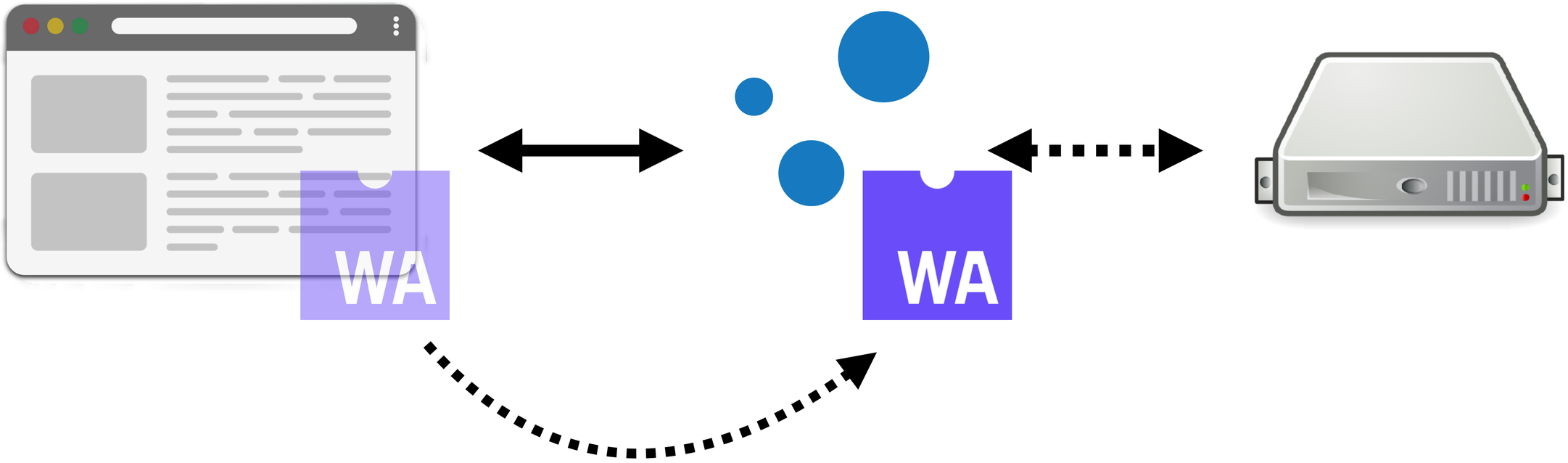


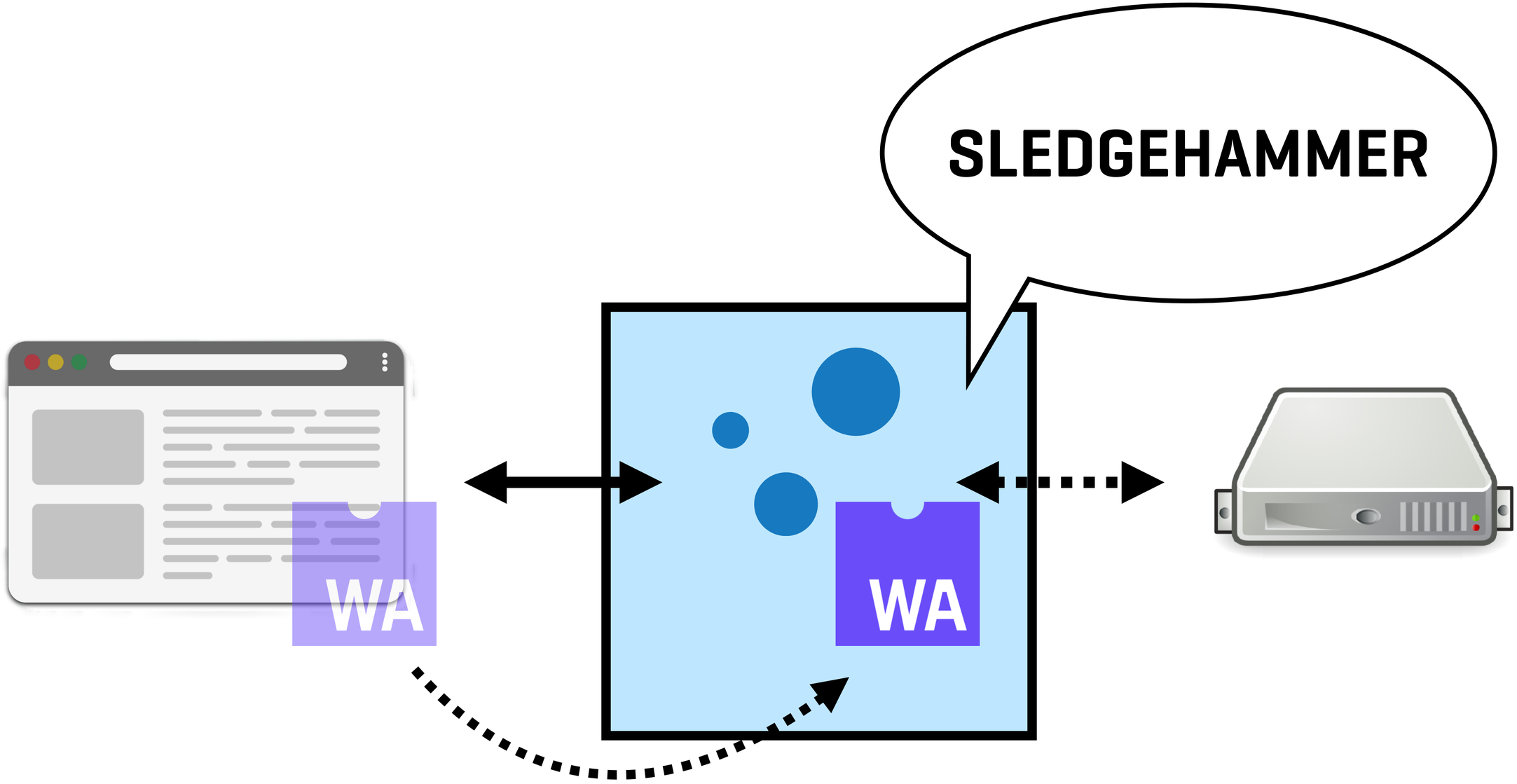

VARNISH
SOFTWARE


USER
VARNISH
SERVER

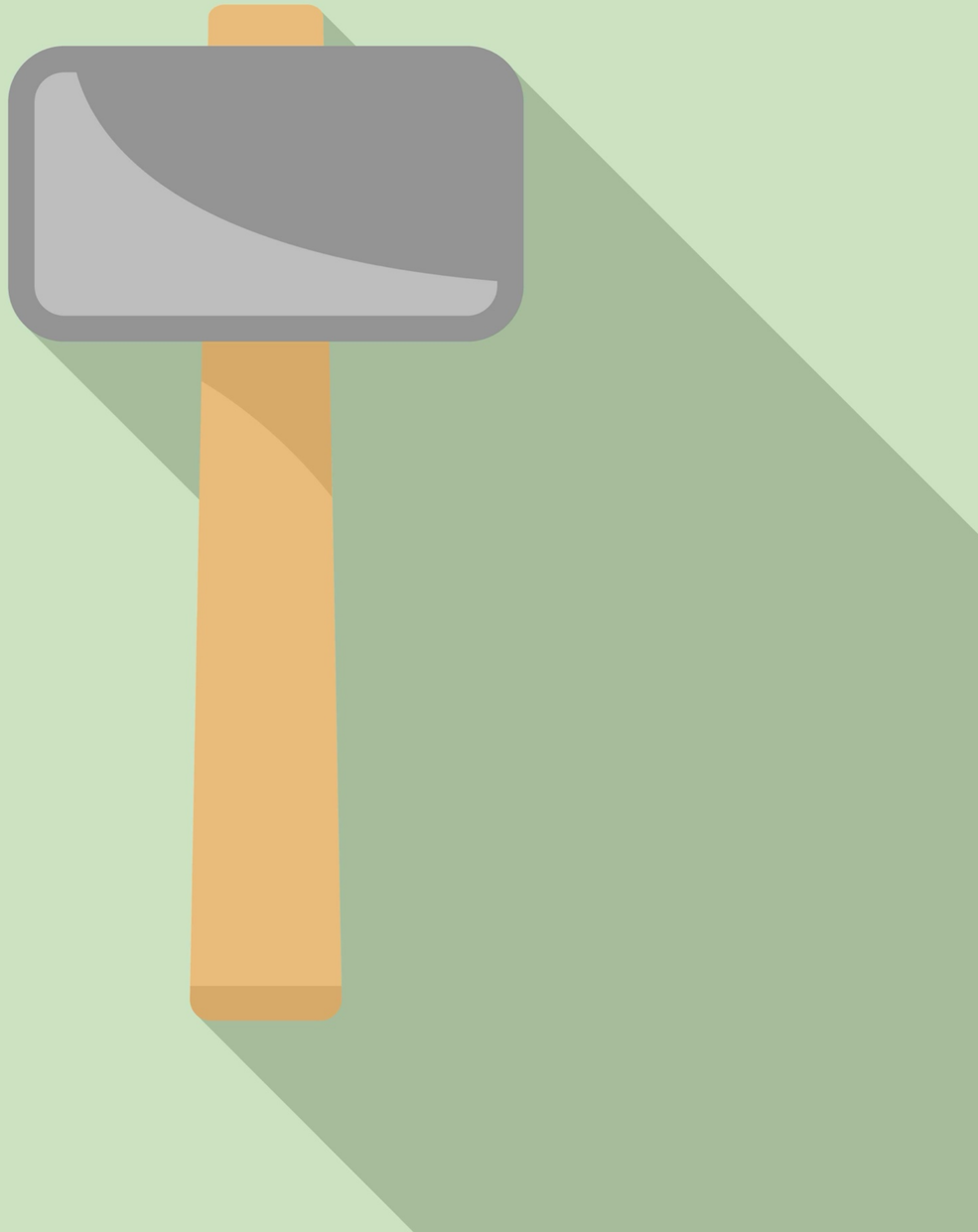

THE EDGE
USER
VARNISH
SERVER


MAKE DECISIONS
CACHE RESPONSES
USER
VARNISH
SERVER

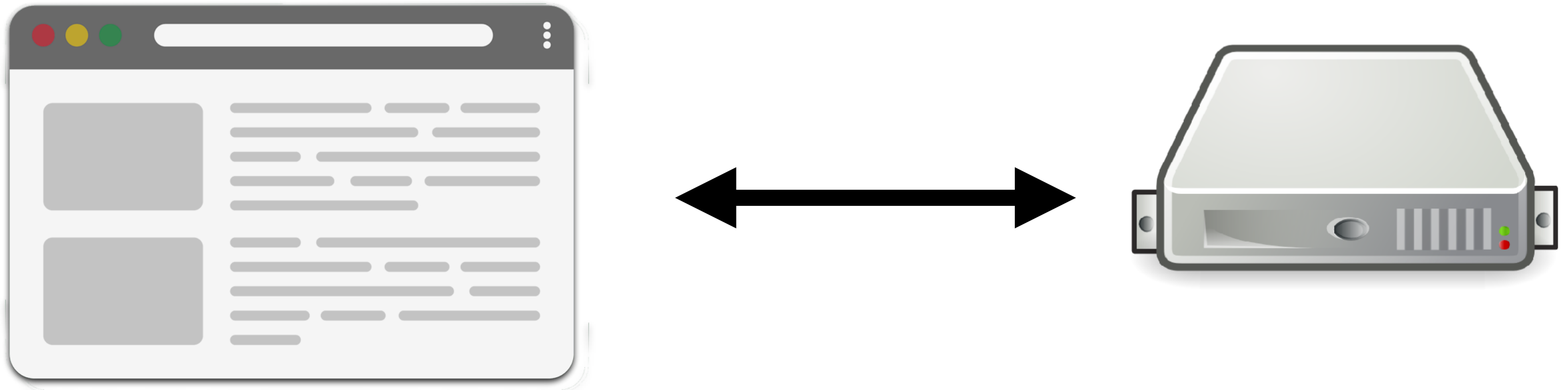

WA
WA


SLEDGEHAMMER
WA
WA

SLEDGEHAMMER IS A RESEARCH PROJECT AT VARNISH SOFTWARE THAT FOCUSES ON RUNNING WEBASSEMBLY WORKLOADS IN A VARNISH MODULE

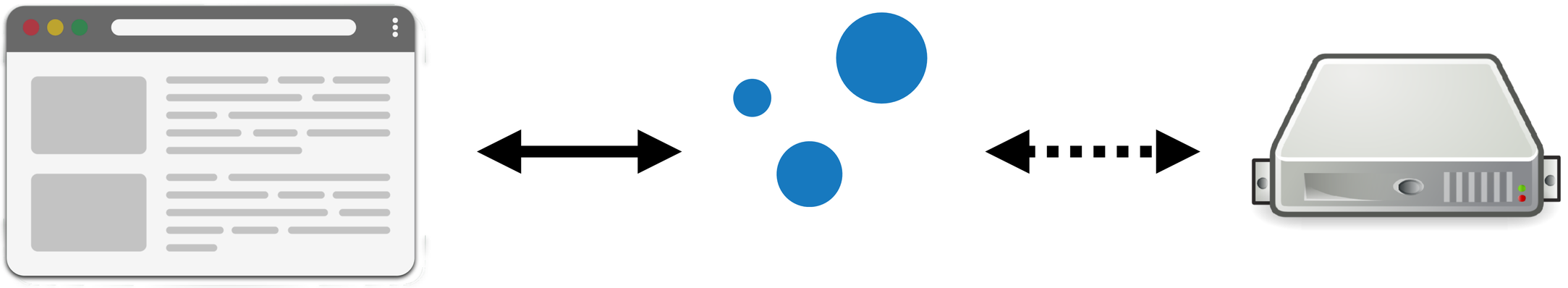


CACHING

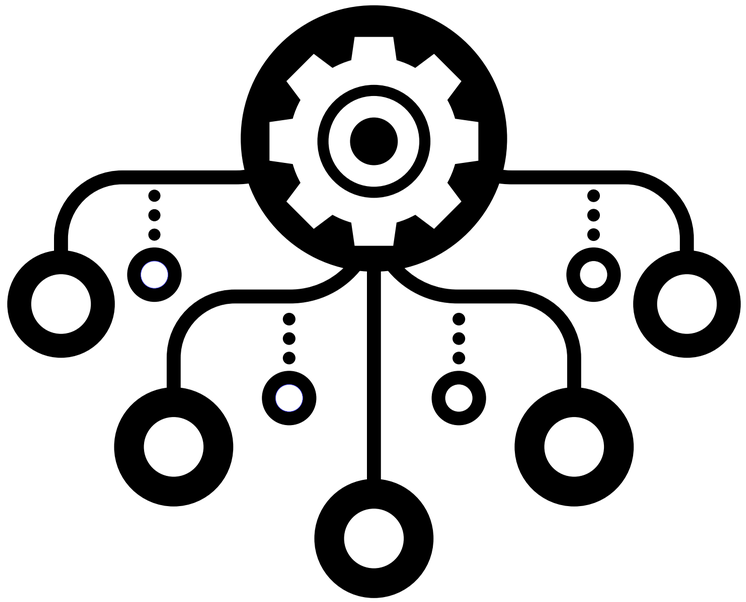

USER
SERVER

UNDER PRESSURE

SERVER


USER
VARNISH
SERVER

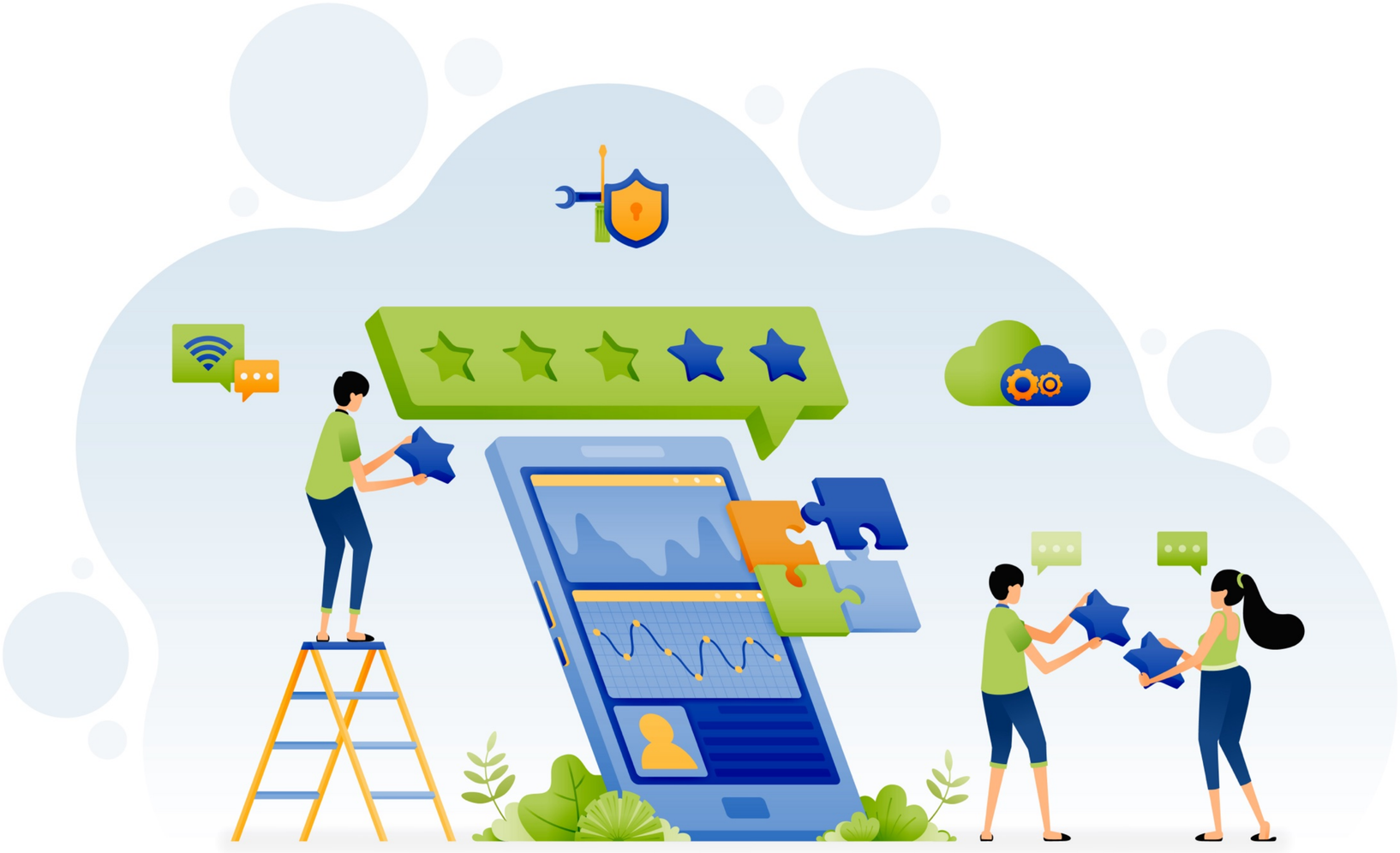
HIGHER CONCURRENCY

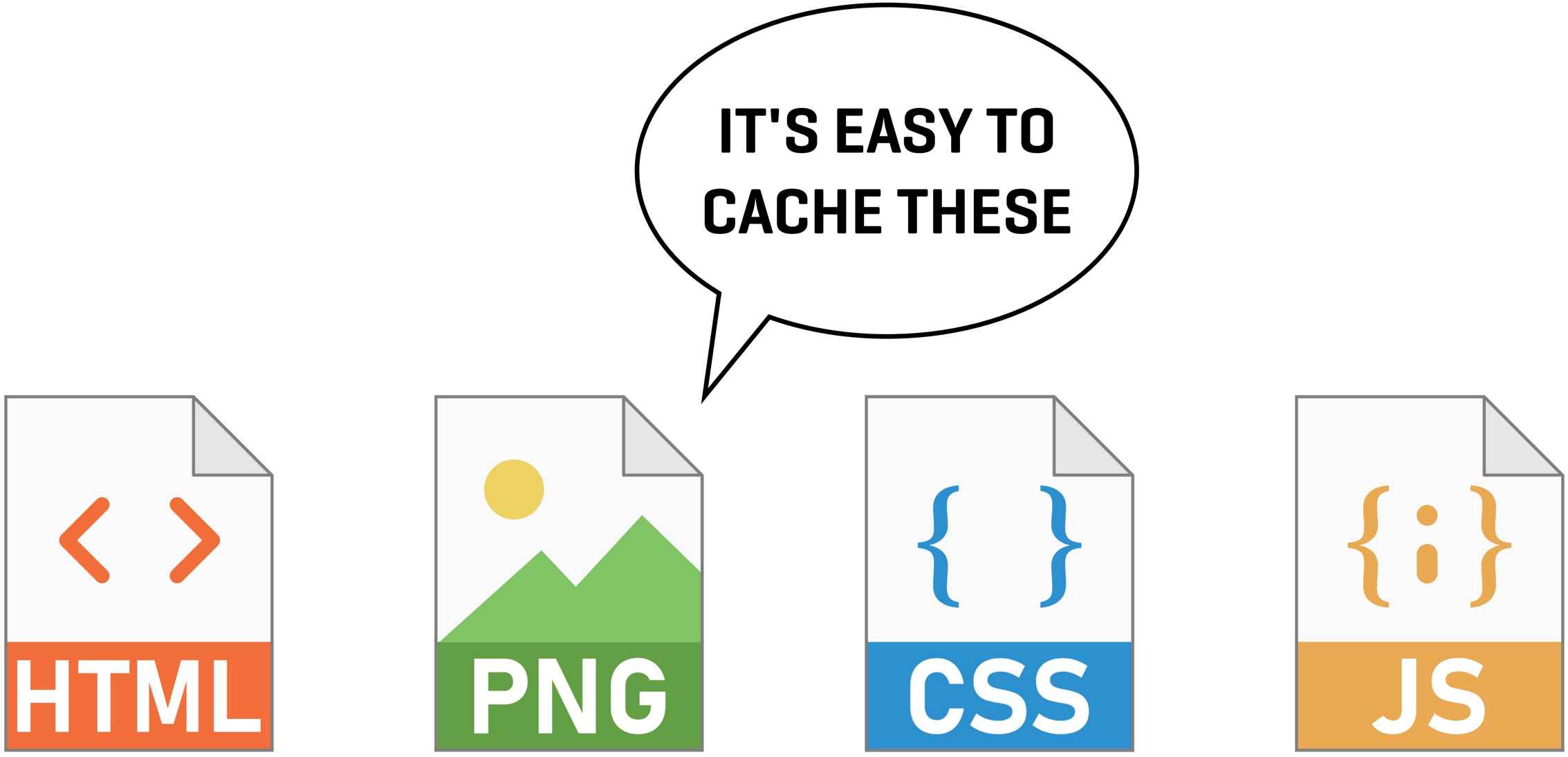


HIGHER THROUGHPUT

LOWER LATENCY

IMPROVE QUALITY OF EXPERIENCE




IT'S EASY TO
CACHE THESE
HTML
PNG
CSS
JS

VARNISH DOESN'T CACHE WHEN COOKIES ARE USED

We use cookies

Lorem ipsum dolor sit amet, cu agam integre his. Zril aperiam ut vim. Cum ne tempor nostro .

Reject Accept

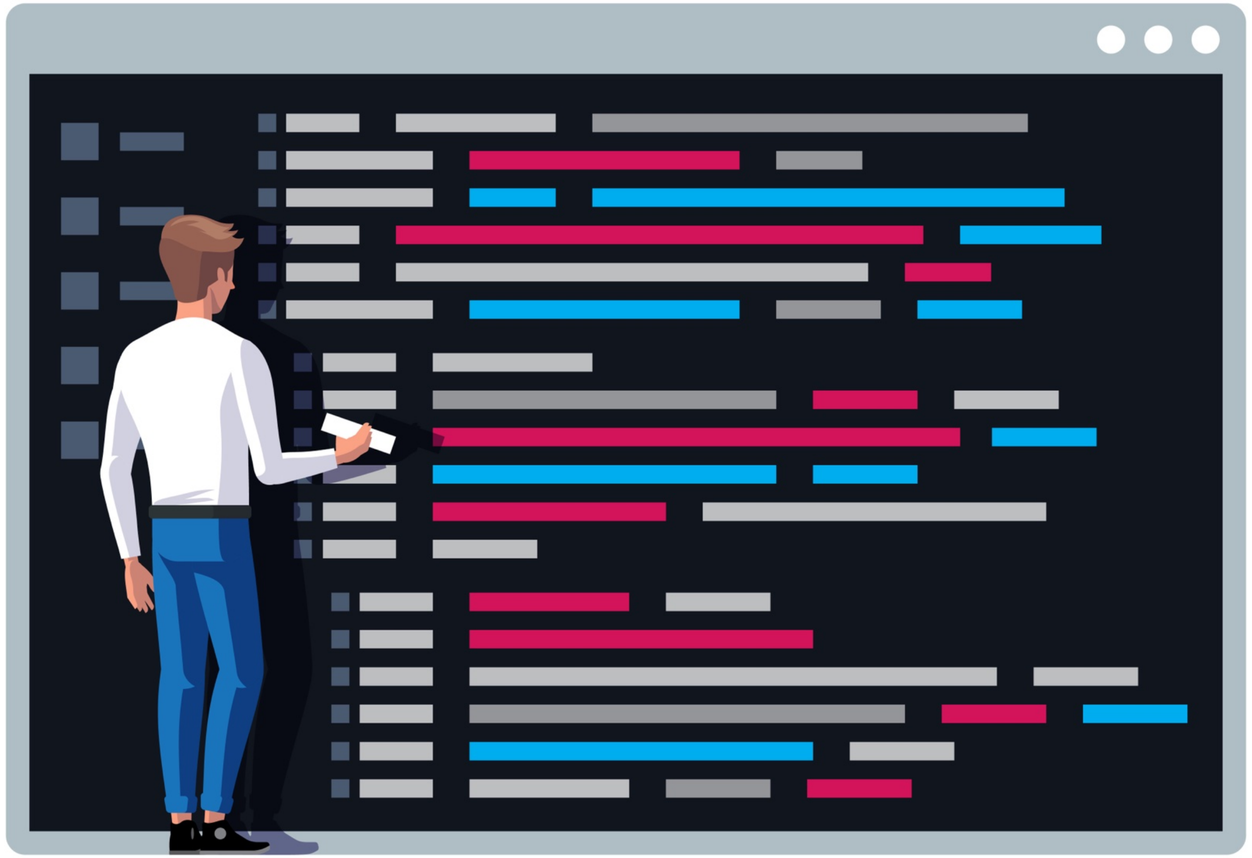

VARNISH DOESN'T CACHE WHEN THERE'S AN AUTHORIZATION HEADER
***

FOR YOUR EYES ONLY

1. STATEMENT
2. QUESTION
3. CONCLUSION

BUILT-IN BEHAVIOR CAN BE EXTENDED THROUGH THE VARNISH CONFIGURATION LANGUAGE

VARNISH CONFIGURATION LANGUAGE

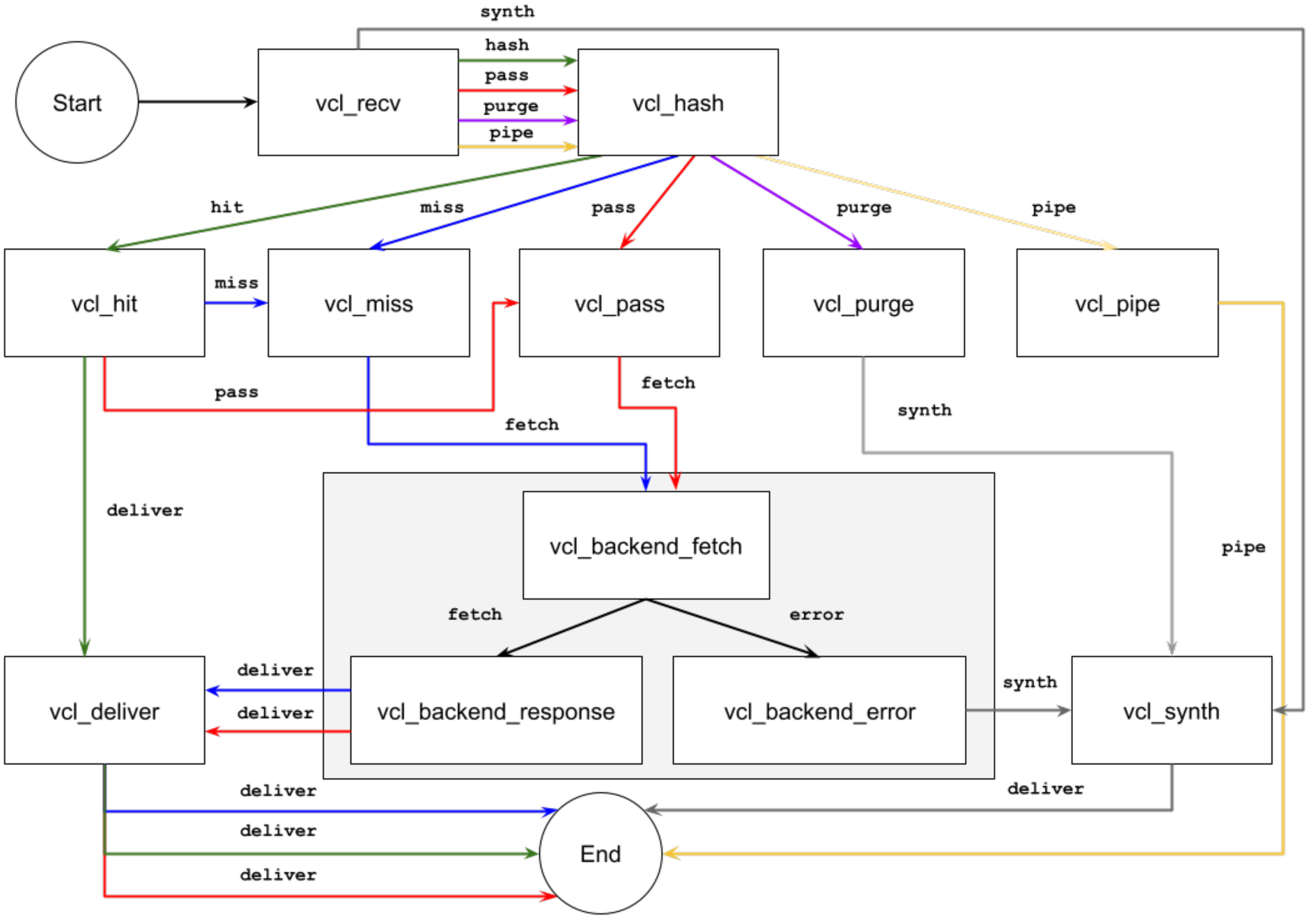


VCL CAPABILITIES

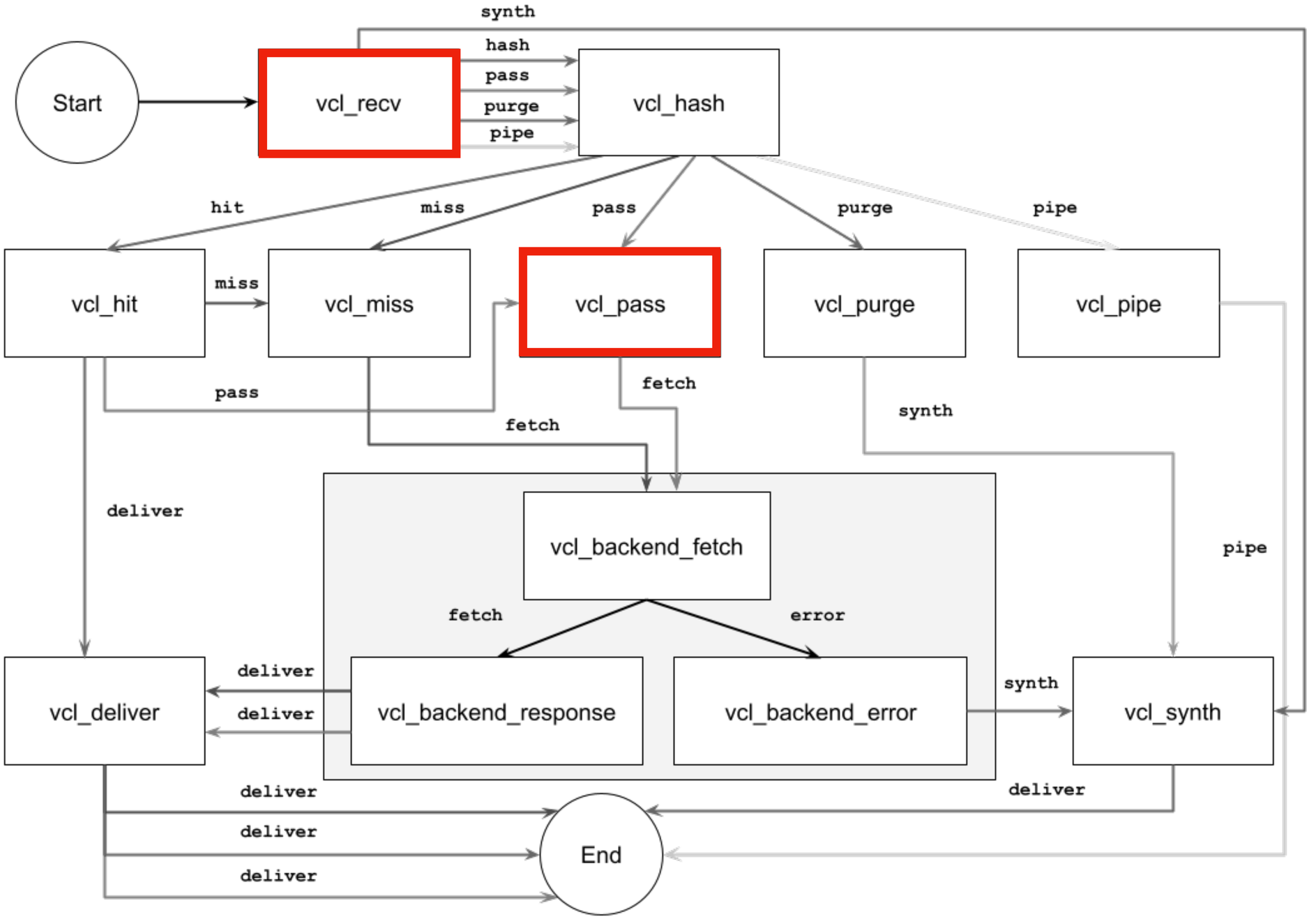
- ✓ REQUEST HANDLING
- ✓ REQUEST ROUTING
- ✓ RESPONSE MANIPULATION
- ✓ BACKEND SELECTION
- ✓ CONTROLLING THE CACHE
- ✓ DECISION-MAKING "ON THE EDGE"

VCL

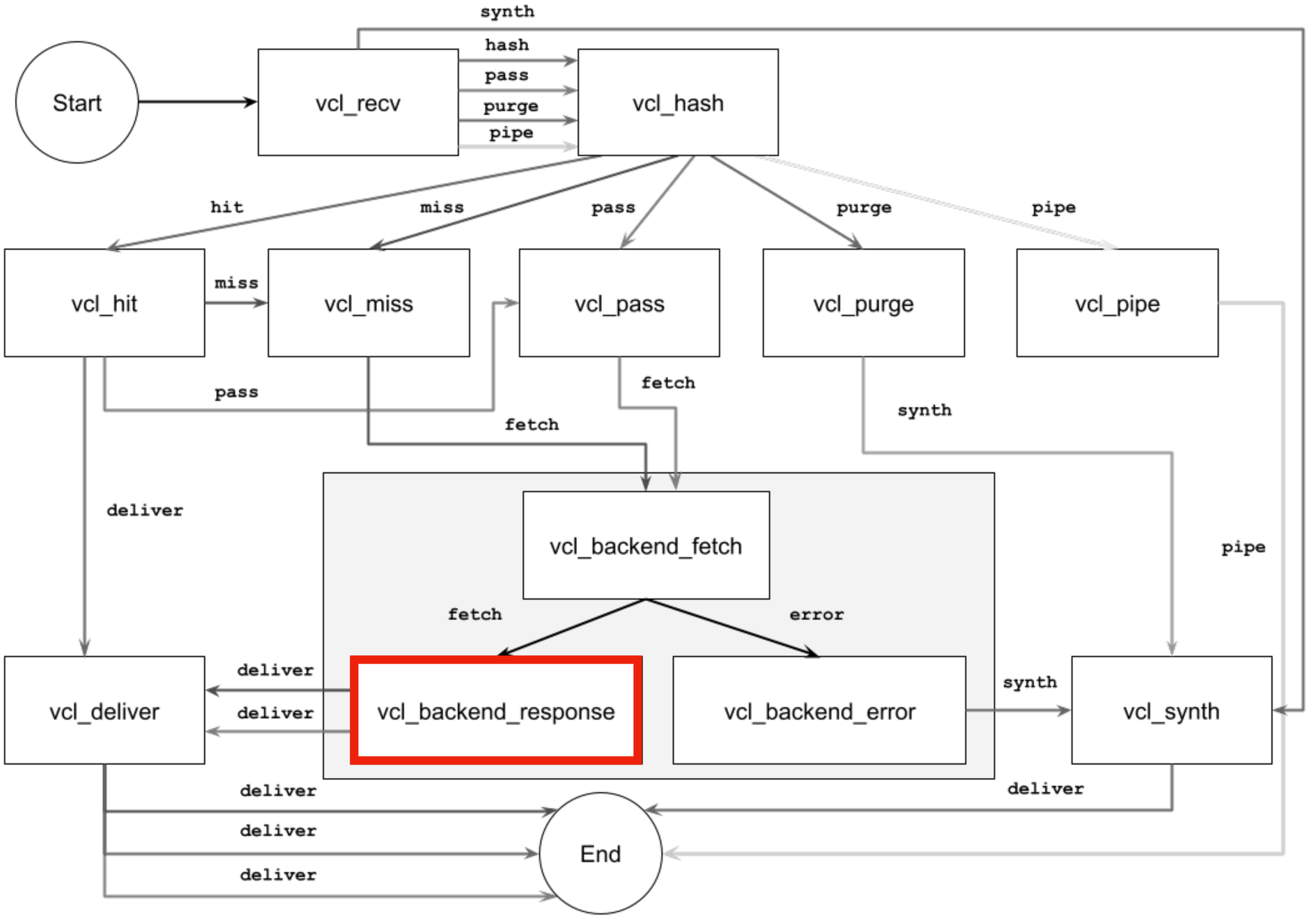
- ✓ DOMAIN-SPECIFIC LANGUAGE
- ✓ CURLY BRACES
- ✓ TRANSPILED INTO C-CODE AND COMPILED INTO MACHINE CODE
- ✓ NOT A TOP-DOWN PROGRAMMING LANGUAGE
- ✓ HOOKS INTO FINITE STATE MACHINE


synth
Start
vcl_recv
hash
pass
purge
pipe
vcl_hash
hit
miss
pass
purge
pipe
vcl_hit
miss
vcl_miss
vcl_pass
vcl_purge
vcl_pipe
pass
fetch
synth
fetch
deliver
vcl_backend_fetch
pipe
fetch
error
deliver
deliver
vcl_deliver
vcl_backend_response
vcl_backend_error
synth
vcl_synth
deliver
deliver
deliver
deliver
End

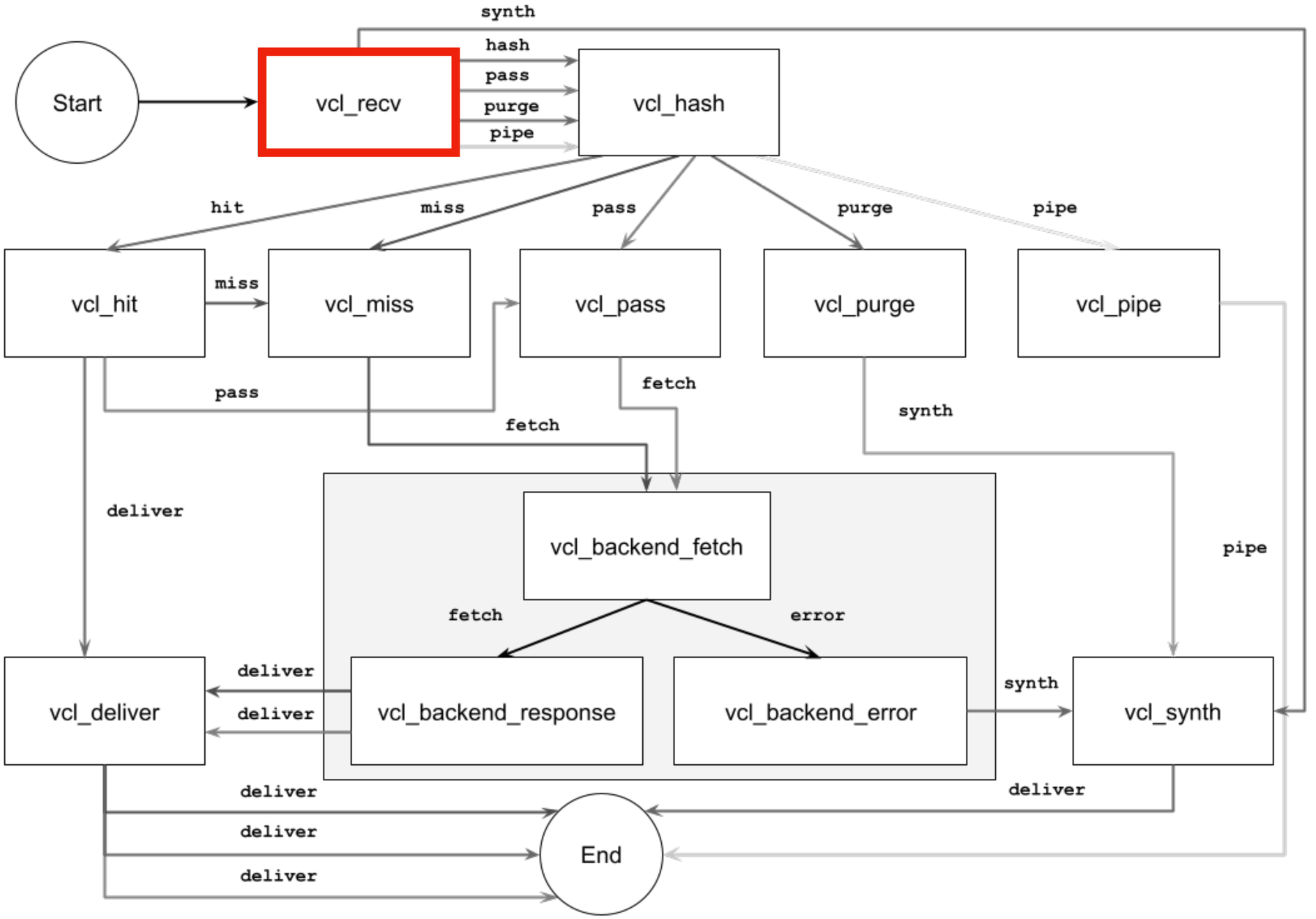
```
vcl 4.1;

backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_recv {
    if(req.url ~ "^/admin(/.*|$)") {
        return(pass);
    }
    unset req.http.Cookie;
}
```


synth
Start
vcl_recv
hash
pass
purge
pipe
vcl_hash
hit
miss
pass
purge
pipe
vcl_hit
miss
vcl_miss
vcl_pass
vcl_purge
vcl_pipe
pass
fetch
synth
fetch
deliver
vcl_backend_fetch
pipe
fetch
error
deliver
deliver
vcl_deliver
vcl_backend_response
vcl_backend_error
synth
vcl_synth
deliver
deliver
deliver
deliver
End

```
vcl 4.1;

backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_backend_response {
    if (beresp.http.Content-Type ~ "^(image|video)/") {
        set beresp.ttl = 1y;
        set beresp.http.Cache-Control = "public, max-age=31536000";
        return(deliver);
    } elseif(beresp.http.Content-Type ~ "^text/html" ||
      beresp.http.Content-Type ~ "^application/json") {
        set beresp.ttl = 1h;
    }
}
```


Start
vcl_recv
vcl_hash
synth
hash
pass
purge
pipe
hit
miss
pass
purge
pipe
vcl_hit
miss
vcl_miss
vcl_pass
vcl_purge
vcl_pipe
pass
fetch
synth
fetch
deliver
vcl_backend_fetch
pipe
fetch
error
deliver
deliver
vcl_deliver
vcl_backend_response
vcl_backend_error
synth
vcl_synth
deliver
deliver
deliver
deliver
End

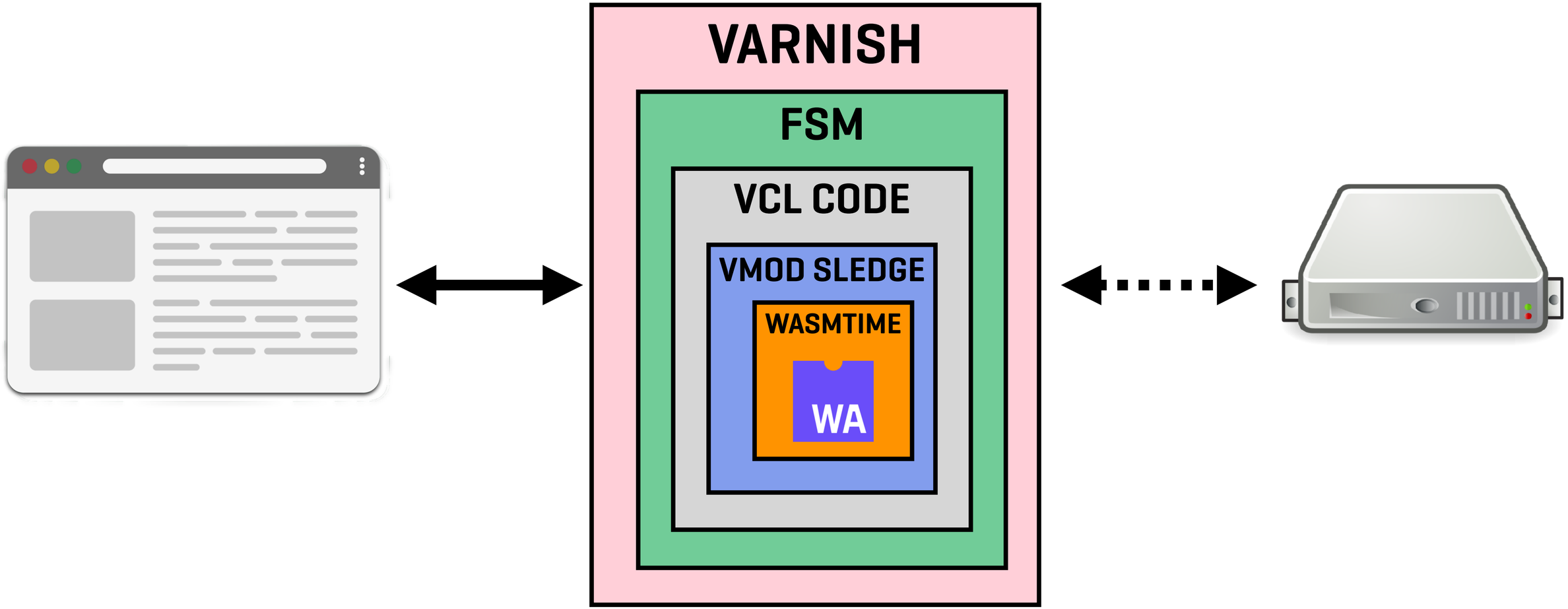
```
vcl 4.1;

import std;

sub vcl_recv {
    # Sort the query string parameters alphabetically
    set req.url = std.querysort(req.url);

    # Remove third-party tracking parameters
    if (req.url ~ "(\?|&)(utm_source|utm_medium|utm_campaign|utm_content)=") {
        set req.url = regsuball(req.url, "&(utm_source|utm_medium|
utm_campaign|utm_content)=([A-z0-9_\-\.%25]+)", "");
        set req.url = regsuball(req.url, "\?(utm_source|utm_medium|
utm_campaign|utm_content)=([A-z0-9_\-\.%25]+)", "?");
        set req.url = regsub(req.url, "\?&", "?");
        set req.url = regsub(req.url, "\?$", "");
    }

    # Remove hashes from the URL
    if (req.url ~ "\#") {
        set req.url = regsub(req.url, "\#.*$", "");
    }

    # Strip off trailing question marks
    if (req.url ~ "\?$") {
        set req.url = regsub(req.url, "\?$", "");
    }
}
```


synth
Start
vcl_recv
hash
pass
purge
pipe
vcl_hash
hit
miss
pass
purge
pipe
vcl_hit
miss
vcl_miss
vcl_pass
vcl_purge
vcl_pipe
pass
fetch
synth
fetch
deliver
vcl_backend_fetch
pipe
fetch
error
deliver
deliver
vcl_deliver
vcl_backend_response
vcl_backend_error
synth
vcl_synth
deliver
deliver
deliver
deliver
End

```
vcl 4.1;

import cookie;

sub vcl_recv {
    if (req.http.cookie) {
        cookie.parse(req.http.cookie);
        cookie.keep("language");

        set req.http.cookie = cookie.get_string();
        if (req.http.cookie ~ "^\s*$") {
            unset req.http.cookie;
        }

        return(hash);
    }
}

sub vcl_hash {
    if(cookie.get("language") ~ "^(nl|fr|de)$") {
        hash_data(cookie.get("language"));
    } else {
        hash_data("en");
    }
}
```

SLEDGEHAMMER

- ✓ EXTEND THE CAPABILITIES OF VARNISH
- ✓ COMPLEMENT VCL
- ✓ LOWER BARRIER OF ENTRY
- ✓ INCREASE ADOPTION


VARNISH
FSM
VCL CODE
VMOD SLEDGE
WASMTIME
WA


DISCLAIMER

IMPORTED FUNCTIONS

EXPERIMENTAL

```
Event.get_request()
Event.get_response()
Event.set_request()
Event.set_response()

Request.new()
Request.get_method()
Request.get_url()
Request.get_protocol()
Request.get_header()
Request.get_body()
Request.add_header()
Request.remove_header()
Request.backend()
Response.new()
```

```
Response.set_body_transformer()
Response.get_body()
Response.add_header()
Response.remove_header()
Response.get_header()

Header.new()
Header.get_name()
Header.get_value()

request_recv()
response_deliver()
backend_request()
backend_response()
synth()
```


Rust

```rust
use std::sync::Arc;

use sledge_rs::{Response, ReturnValue, StatusHeader, StringBody};

#[export_name = "request_recv"]
pub extern "C" fn request_recv() {
    sledge_rs::request_recv(move |event| {
        // Continue with the same request
        return Ok(None);
    });
}

#[export_name = "backend_request"]
pub extern "C" fn backend_request() {
    sledge_rs::backend_request(move |_event| {
        // Continue with the same request
        return Ok(None);
    });
}
```

```rust
#[export_name = "backend_response"]
pub extern "C" fn backend_response() {
    sledge_rs::backend_response(move |event| {
        // Continue with the same response
        return Ok(None);
    });
}

#[export_name = "response_deliver"]
pub extern "C" fn deliver() {
    sledge_rs::response_deliver(move |event| {
        // Continue with the same response
        return Ok(None);
    });
}
```

```
sledge new -t rust-empty my-rust-app
cd my-rust-app
sledge build .
sledge generate-vcl -O default.vcl target/
```

EXPERIMENTAL


.
├── CACHEDIR.TAG
├── release
│   ├── build
│   ├── deps
│   ├── examples
│   └── incremental
├── sledge.manifest
└── wasm32-wasi
    ├── CACHEDIR.TAG
    └── release
        ├── my-rust-app.d
        ├── my-rust-app.wasm
        ├── build
        ├── deps
        │   ├── my-rust-app.my-rust-app.d5df1d2654ca7cd2-cgu.0.rcgu.o
        │   ├── my-rust-app.my-rust-app.d5df1d2654ca7cd2-cgu.1.rcgu.o
        │   ├── my-rust-app.my-rust-app.d5df1d2654ca7cd2-cgu.3.rcgu.o
        │   ├── my-rust-app.d
        │   ├── my-rust-app.wasm
        │   ├── libonce_cell-8a9b2f74a8a66c35.rlib
        │   ├── libonce_cell-8a9b2f74a8a66c35.rmeta
        │   ├── libsledge_rs-d9d434d0db7c9783.rlib
        │   ├── libsledge_rs-d9d434d0db7c9783.rmeta
        │   ├── once_cell-8a9b2f74a8a66c35.d
        │   └── sledge_rs-d9d434d0db7c9783.d
        ├── examples
        └── incremental
EXPERIMENTAL

```
vcl 4.1;

import sledge;

backend default {
    .host = "server.example.com";
}

sub vcl_init {
    sledge.load_application("/path/to/my-rust-app/target");
}

sub vcl_recv {
     sledge.recv();
     if (sledge.next_state() == "synth") {
          return ( synth(999, "Unknown") );
     }
     if (sledge.next_state() == "fail") {
          return ( fail );
     }
}

sub vcl_synth {
     sledge.synth();
     return( deliver );
}
```

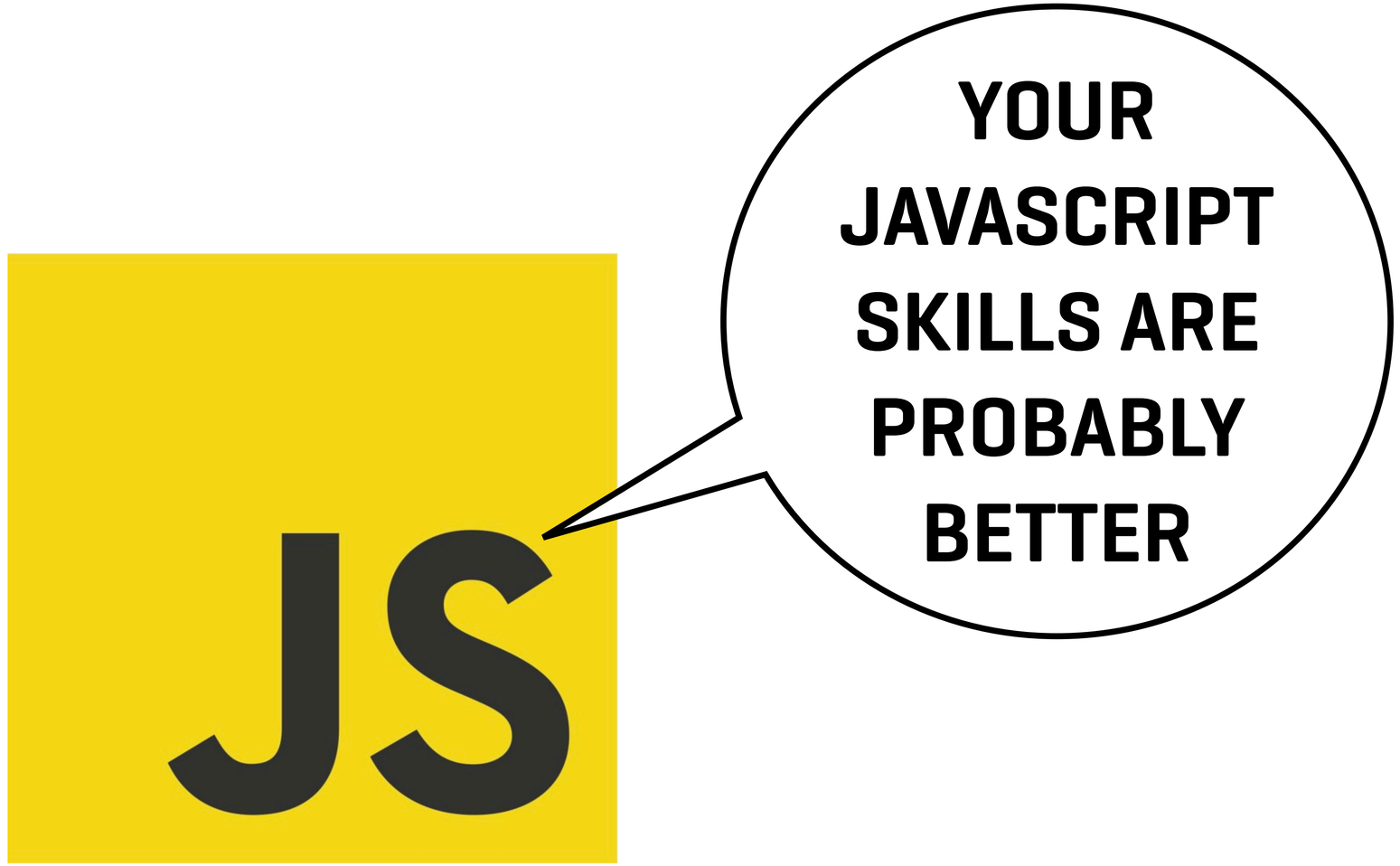
```
sub vcl_backend_fetch {
     sledge.backend_request();
}

sub vcl_backend_response {
     sledge.backend_response();
}

sub vcl_deliver {
     sledge.deliver();
     if (sledge.next_state() == "synth") {
          return ( synth(999, "Unkown") );
     }
     if (sledge.next_state() == "fail") {
          return ( fail );
     }
}
```

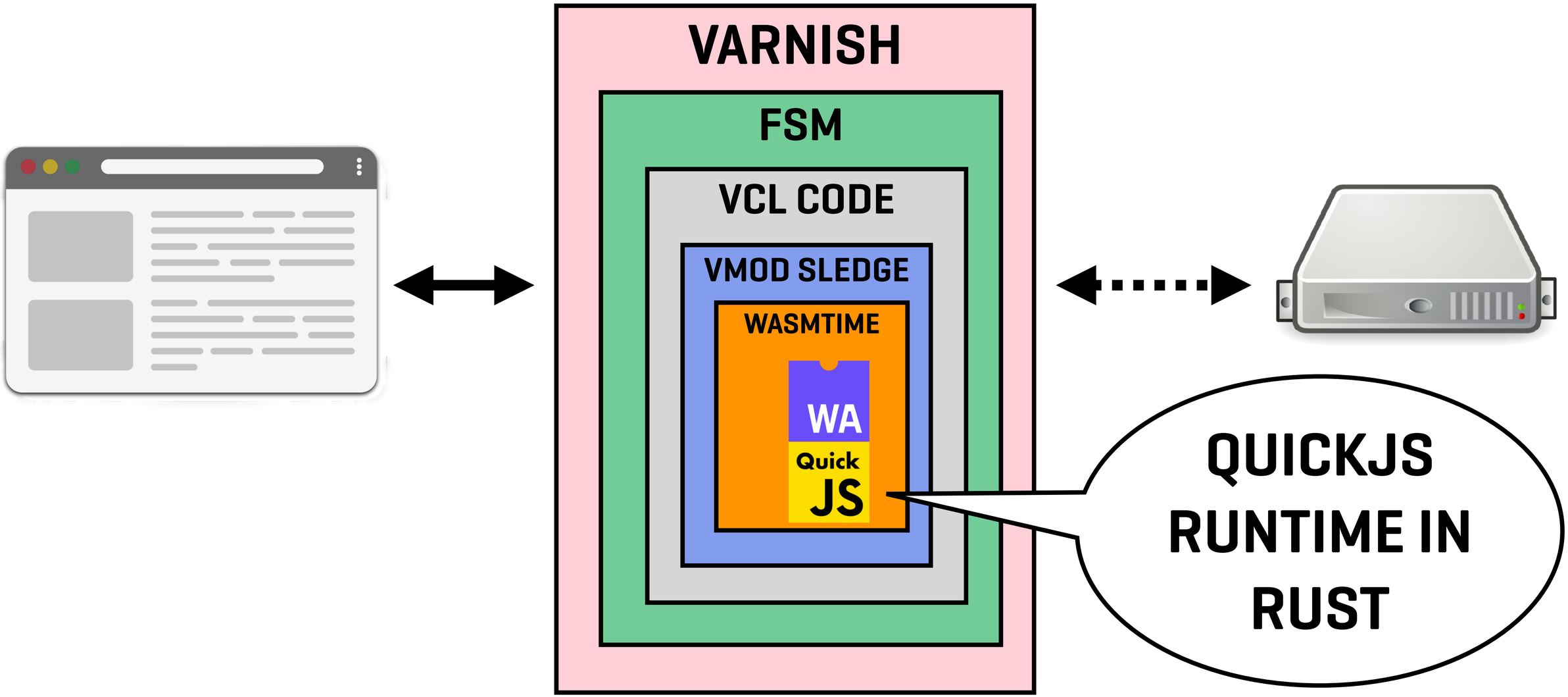
```rust
use chrono::{DateTime, NaiveDateTime, TimeZone, Utc};
use sledge_rs::{Response, ReturnValue, StatusHeader, StringBody};
use std::sync::Arc;
use std::time::SystemTime;

#[export_name = "request_recv"]
pub extern "C" fn internal_request_recv() {
    sledge_rs::request_recv(move |event| {
        let mut req = event.get_request().unwrap();
        req.cache_req_body(512000);

        let path = req.get_url();

        if path.as_str() == "/login" {
            let maybeAuthHeader = req.get_header("Authorization".to_string());

            if maybeAuthHeader.is_none() || maybeAuthHeader.unwrap().ne("Basic YWRtaW46c2VjcmV0") {
                let statusHeader = StatusHeader::new(401, Some("Restricted".to_owned()));
                let mut resp = Response::new(Some(statusHeader));
                resp.add_header(
                    "WWW-Authenticate".to_string(),
                    "Basic realm=\"Restricted area\"".to_string(),
                );
                resp.set_body_transformer(Arc::new(StringBody::new("Restricted".to_string())));
                return Ok(Some(ReturnValue::Response(resp)));
            }
        }
        return Ok(Some(ReturnValue::Request(req)));
    });
}
```

EXPERIMENTAL

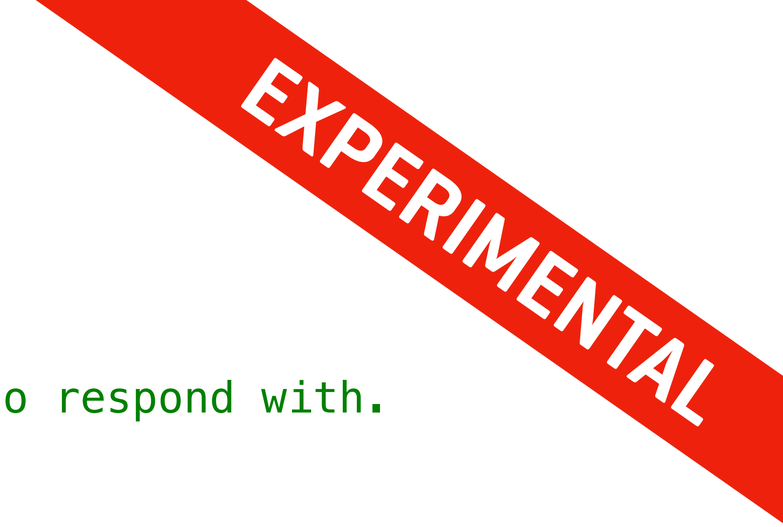
```rust
#[export_name = "backend_request"]
pub extern "C" fn backend_request() {
    sledge_rs::backend_request(move |_event| {
        // Continue with the same request
        return Ok(None);
    });
}

#[export_name = "backend_response"]
pub extern "C" fn backend_response() {
    sledge_rs::backend_response(move |event| {
        let mut resp = event.get_response().unwrap();
        resp.add_header("Cache-Control".to_string(), "private, no-cache, no-store, must-
revalidate".to_string());
        return Ok(Some(ReturnValue::Response(resp)));
    });
}

#[export_name = "response_deliver"]
pub extern "C" fn deliver() {
    sledge_rs::response_deliver(move |event| {
        let mut resp = event.get_response().unwrap();
        let datetime = chrono::offset::Local::now();
        let formatted_time = datetime.format("%Y-%m-%dT%H:%M:%S%.fZ");
        resp.add_header("X-Current-Time".to_string(), formatted_time.to_string());
        return Ok(Some(ReturnValue::Response(resp)));
    });
}
```


HOW GOOD ARE YOUR RUST CODING SKILLS?


JS
YOUR JAVASCRIPT SKILLS ARE PROBABLY BETTER


VARNISH
FSM
VCL CODE
VMOD SLEDGE
WASMTIME
WA
Quick
JS
QUICKJS
RUNTIME IN
RUST

```
import { Event, Request, Response } from 'sledge-js';

/**
 * Callback for the "recv" event.
 *
 * @param {Event} event
 * @returns {Response | Request} The response or request object to respond with.
 */
function handleRecv(event) {
    return undefined;
}

/**
 * Callback for the "backend_request" event (aka backend_fetch).
 *
 * @param {Event} event
 * @returns
 */
function handleBeRequest(event) {
    return undefined;
}
```



```
/**
 * Callback for the "backend_response" event.
 *
 * @param {Event} event
 * @returns
 */
function handleBeResponse(event) {
    return undefined;
}

/**
 * Callback for the "deliver" event.
 *
 * @param {Event} event
 * @returns
 */
function handleDeliverResponse(event) {
    return undefined;
}


// On startup we need to register the event listeners
addEventListener("recv", event => event.respondWith(handleRecv(event)))
addEventListener("backend_request", event => event.respondWith(handleBeRequest(event)))
addEventListener("backend_response", event => event.respondWith(handleBeResponse(event)))
addEventListener("deliver", event => event.respondWith(handleDeliverResponse(event)))
```

EXPERIMENTAL

```
sledge new -t js-empty my-js-app
cd my-js-app
sledge build .
sledge generate-vcl -O default.vcl target/
```

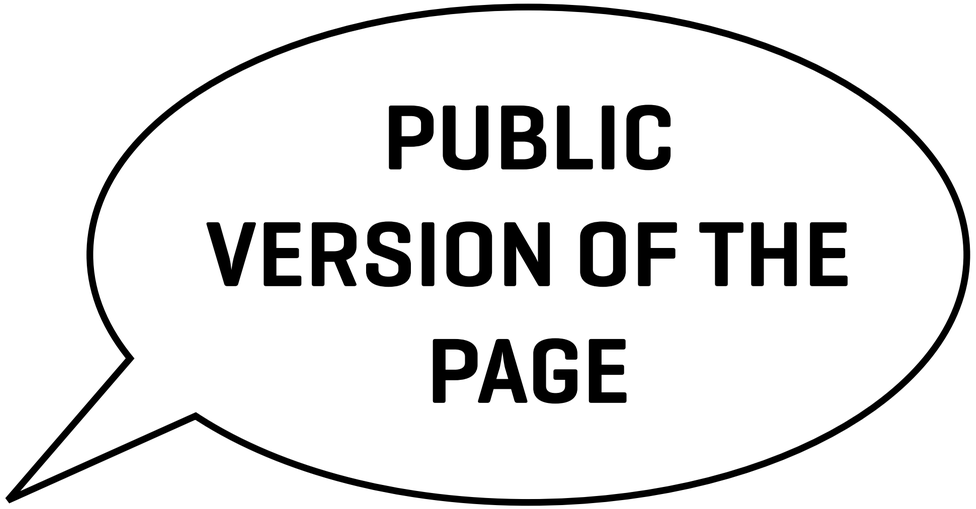
```
<!doctype html>
<html lang="en" data-bs-theme="auto">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/
bootstrap.min.css" rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">
    </head>
    <body>
      <main>
        <section class="py-5 text-center container">
          <div class="row py-lg-5">
            <div class="col-lg-6 col-md-8 mx-auto">
              <h1 class="fw-light">Name goes here</h1>
              <p class="lead text-body-secondary">Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Sed mattis placerat facilisis. Vivamus placerat porttitor
pharetra.</p>
            </div>
          </div>
        </section>
      </main>
    </body>
</html>
```

Name goes here

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mattis placerat facilisis. Vivamus placerat porttitor pharetra.

Name goes here

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mattis placerat facilisis. Vivamus placerat porttitor pharetra.



Guest

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mattis placerat facilisis. Vivamus placerat porttitor pharetra.


/LOGIN
Sign in
http://localhost
Username
Password
Cancel
Sign In

Admin

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mattis placerat facilisis. Vivamus placerat porttitor pharetra.

```
Cookie:
token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkFkb
WluIiwiaWF0IjoxNzA1OTk1NzkxODkzfQ.A6uMiKS6
UdK5EgRHqZE5fL8Qy7tY00b3Ji9KiFIIaYA
```

eyJhbGciOiJIUzI1NiIsInR5cCI6
IkpXVCJ9

eyJzdWIiOiIxMjM0NTY3ODkwIiwi
bmFtZSI6IkFkbWluIiwiaWF0Ijox
NzA1OTk1NzkxODkzfQ

A6uMiKS6UdK5EgRHqZE5fL8Qy7tY
00b3Ji9KiFIIaYA

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "sub": "1234567890",
  "name": "Admin",
  "iat": 1705995791893
}
```

```
HMACSHA256(

base64UrlEncode(header)
+ "." +
base64UrlEncode(payload,
  your-256-bit-secret
)
```

EXPERIMENTAL

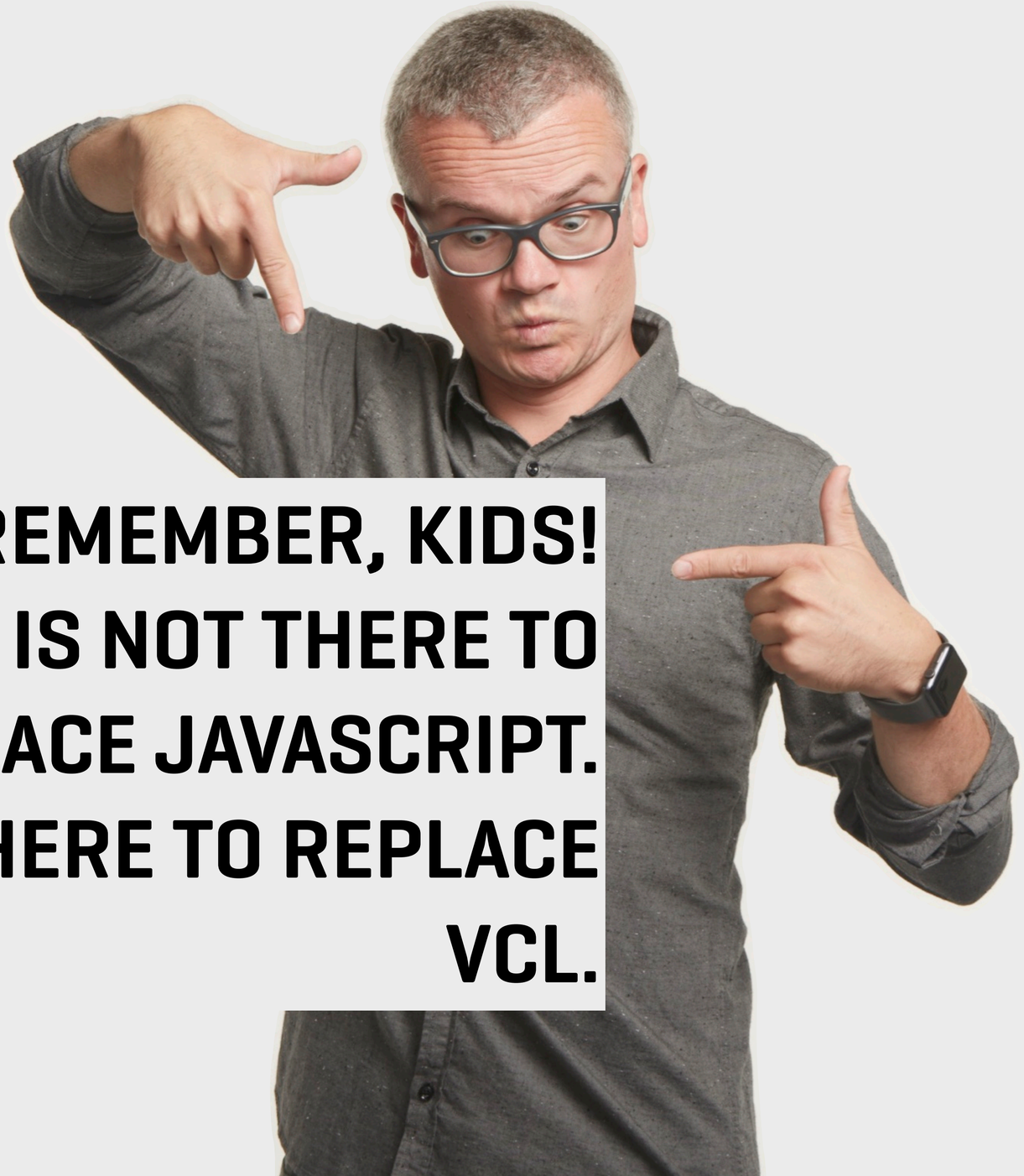
```
import { Event, Request, Response } from 'sledge-js';
import {DOMParser,XMLSerializer} from '@xmldom/xmldom';
import xpath from 'xpath';
import "core-js/stable/atob.js";
import sign from 'jwt-encode';
import { jwtDecode } from 'jwt-decode';
import { parse } from 'cookie';

function handleRecv(event) {
    let request = event.request();
    let path = request.getUrl();
    let authHeader = request.getHeader("Authorization");
    if(path == "/login") {
        if(authHeader != "Basic YWRtaW46c2VjcmV0") {
            let response = new Response(401, "Restricted");
            response.addHeader("WWW-Authenticate","Basic realm=\"Restricted area\"");
            return response;
        }
        const data = {
            sub: '1234567890',
            name: 'Admin',
            iat: Date.now()
        }
        let response = new Response(301, "Moved");
        response.addHeader("Set-Cookie","token=" + sign(data,"secret"));
        response.addHeader("Location","/");
        return response;
    }
```

```
    } else if(path == "/") {
        if(request.getHeader("Cookie") !== undefined) {
            try {
                const cookies = parse(request.getHeader("Cookie"));
                if(cookies.token !== undefined) {
                    request.addHeader("token",cookies.token);
                }
            } catch(err) {
                console.error(err);
            }
        }
    }

    return request;
}
```

EXPERIMENTAL

```
function handleBeResponse(event) {
    let request = event.request();
    let response = event.response();
    let path = request.getUrl();
    if(path == "/login") {
        response.addHeader("Cache-Control","private, no-cache, no-store, must-
revalidate");
    } else if(path == "/") {
        let token = request.getHeader("token");
        let name = "Guest";
        if(token !== undefined) {
            try {
                const decoded = jwtDecode(token);
                if(decoded.name !== undefined) {
                    name = decoded.name;
                }
            } catch(err) {
                console.error(err);
            }
        }

        response.addHeader("Cache-Control","private, no-cache, no-store, must-
revalidate");
```


```
        response.setBody(event => {
            let body = response.getBodyAsString();
            const parser = new DOMParser({
                errorHandler: { warning: function (w) { },
                error: function (e) { },
                fatalError: function (e) { console.error(e) } }
            });

            var doc = parser.parseFromString(body);
            var node = xpath.select("//h1", doc, true);
            node.firstChild.data = name;
            const serialized = new XMLSerializer().serializeToString(doc)

            return serialized;
        });
    }
    return response;
}

addEventListener("recv", event => event.respondWith(handleRecv(event)))
addEventListener("backend_response", event => event.respondWith(handleBeResponse(event)))
```



REMEMBER, KIDS!
WASM IS NOT THERE TO
REPLACE JAVASCRIPT.
NOR IS IT THERE TO REPLACE
VCL.

Thijs Feryn

VARNISH 6

BY EXAMPLE

A practical guide to web acceleration and content delivery with Varnish 6 technology


THE END