# An ASGI Server from scratch

P G Jones - 2020-07-23

pgjones.dev

https://pgjones.dev/talks/ | https://github.com/pgjones/asgi_server_from_scratch

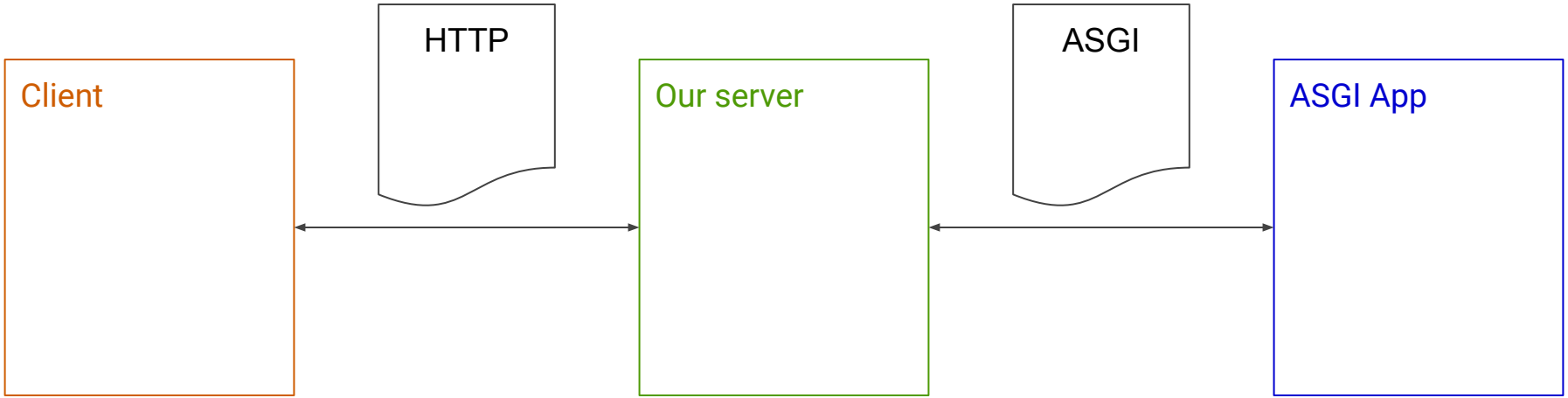
# Me

pgjones.dev
@pgjones github/gitlab
@pdgjones twitter

moneyed

moneyed.co.uk

# Aim



# async/await and asyncio

---

```python
async def coroutine_function():
    await ...


asyncio.run(coroutine_function())


asyncio.start_server(...)
```

# WSGI Intro (Web Server Gateway Interface)?

---

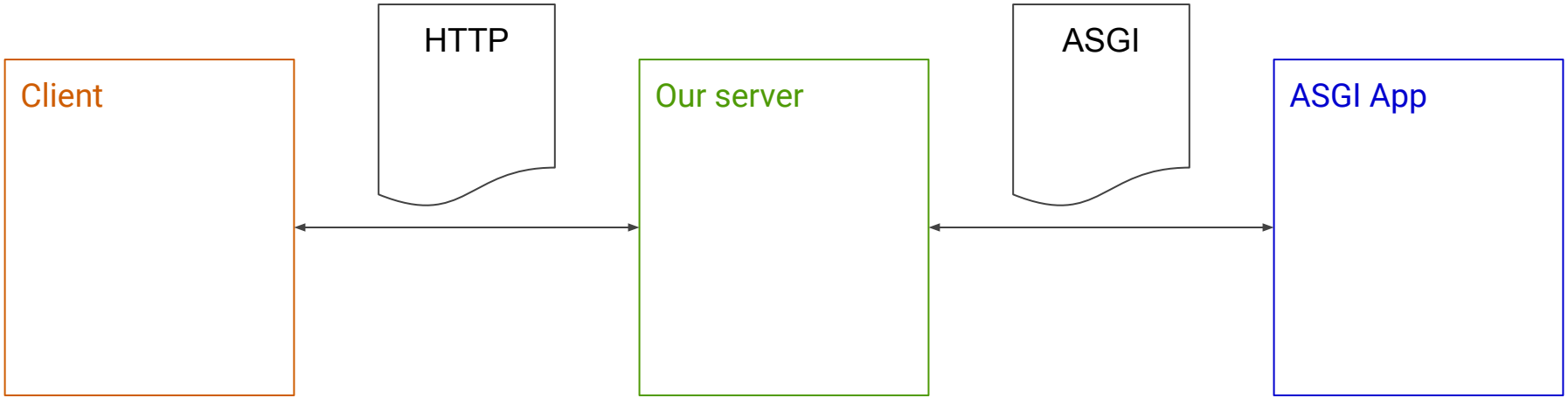
```
def application(environ, start_response):
    start_response(
        '200 OK',
        [('Content-Type', 'text/plain')],
    )
    yield b'Hello, World\n'
```

# What is ASGI (Asynchronous Server Gateway Interface)?

---

```python
async def application(scope, receive, send):
    await send({
        "type": "http.response.start",
        "status": 200,
        "headers": [(b'Content-Type', b'text/plain')],
    })
    await send({
        "type": "http.response.body",
        "body": b"Hello, World\n",
    })
```

# Aim



# Echo server

```
import asyncio
import sys

async def echo_server(reader, writer):
    while not reader.at_eof():
        data = await reader.read(100)
        writer.write(data)
        await writer.drain()
    writer.close()

async def main(host, port):
    server = await asyncio.start_server(echo_server, host, port)
    await server.serve_forever()
```

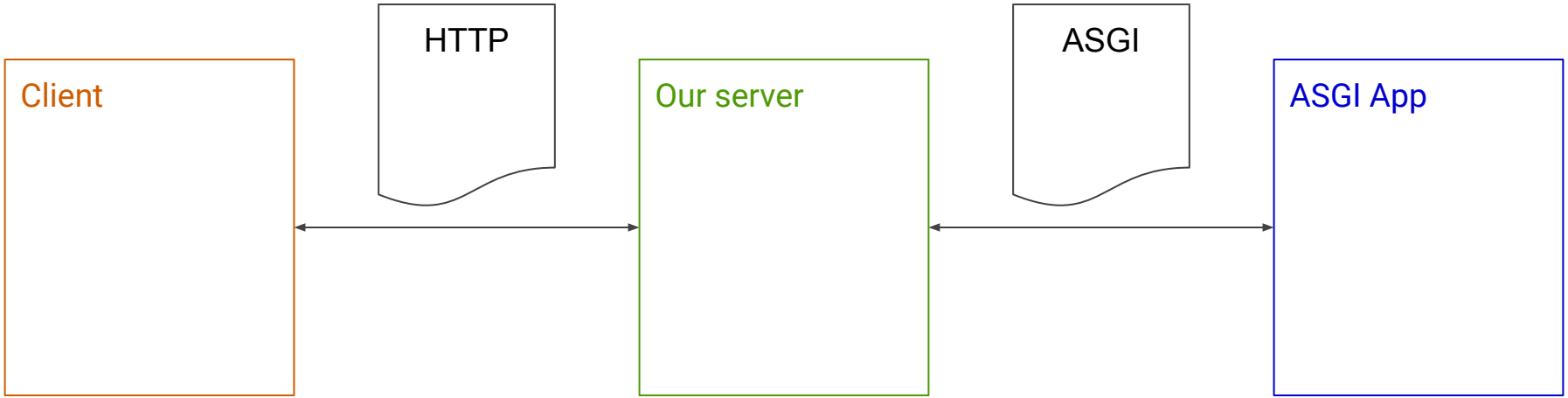
https://docs.python.org/3/library/asyncio-stream.html#tcp-echo-server-using-streams

# Echo server test

```
$ python server.py localhost 5005
```

```
$ telnet localhost 5005
Trying ::1...
Connected to localhost.
Escape character is '^]'.
hello
hello
goodbye
goodbye
^]
telnet> Connection closed.
```

# Aim



# HTTP Parsing

---

```
class HTTPParser:
    def __init__(self):
        self.part = "REQUEST"
        self.headers = []
        self.body_length = 0

    def feed_line(self, line: bytes):
        if self.part == "REQUEST":
            self.method, self.path, self.version = line.split(b" ", 2)
            self.part = "HEADERS"
        elif self.part == "HEADERS" and line.strip() == b"":
            self.part = "BODY"
        elif self.part == "HEADERS":
            name, value = line.split(b":", 1)
            self.headers.append((name.strip(), value.strip()))
            if name.lower() == b"content-length":
                self.body_length = int(value)
```

**HTTP**

```
POST / HTTP/1.1
Host: localhost:5005
Content-Length: 5

Hello
-------------------------
Method path version
Header-name: Header-value

Body
```

# HTTP Parsing server

---

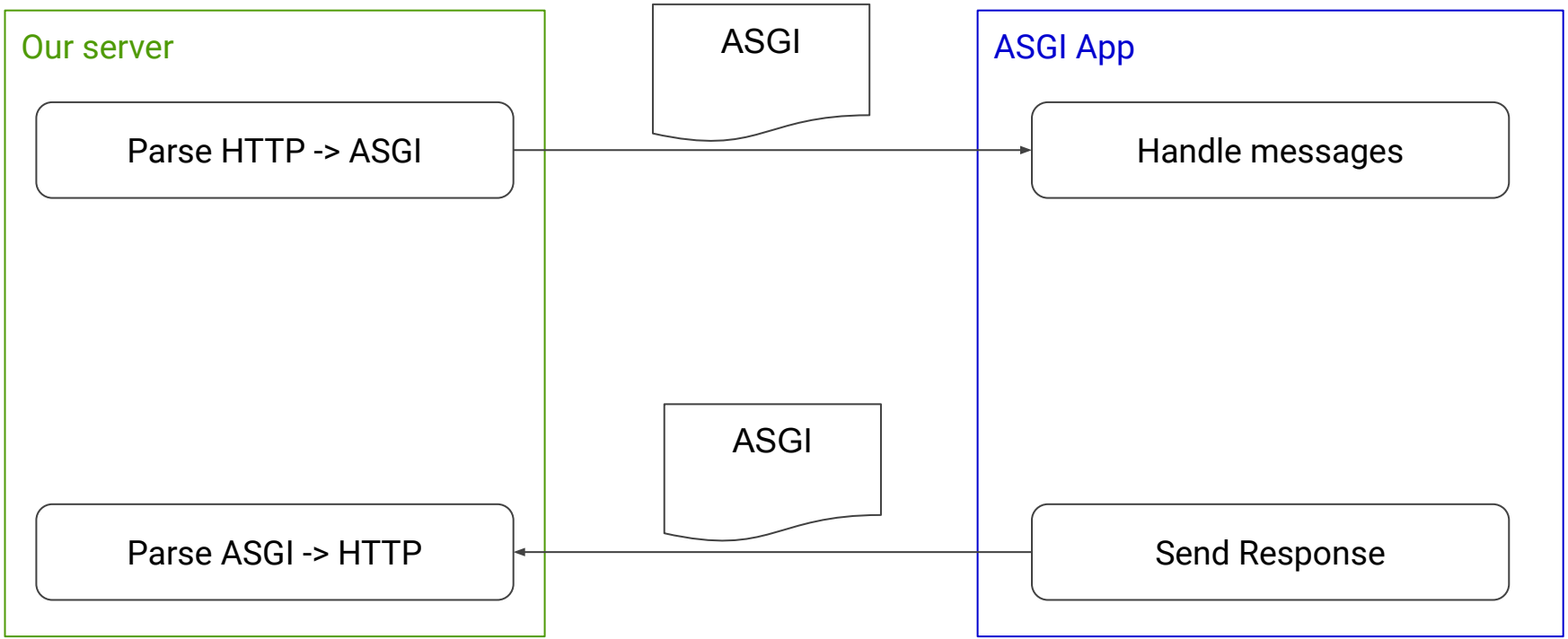
```
async def http_parser_server(reader, writer):
    parser = HTTPParser()
    body = bytearray()
    while not reader.at_eof():
        if parser.part != "BODY":
            parser.feed_line(await reader.readline())
        else:
            if len(body) >= parser.body_length:
                break
            body.extend(await reader.read(100))
    print(parser.method, parser.path, parser.headers)
    print(body)
    writer.write(b"HTTP/1.1 200\r\nContent-Length: 0\r\n\r\n")
    await writer.drain()
    writer.close()
```

# HTTP Parsing server test

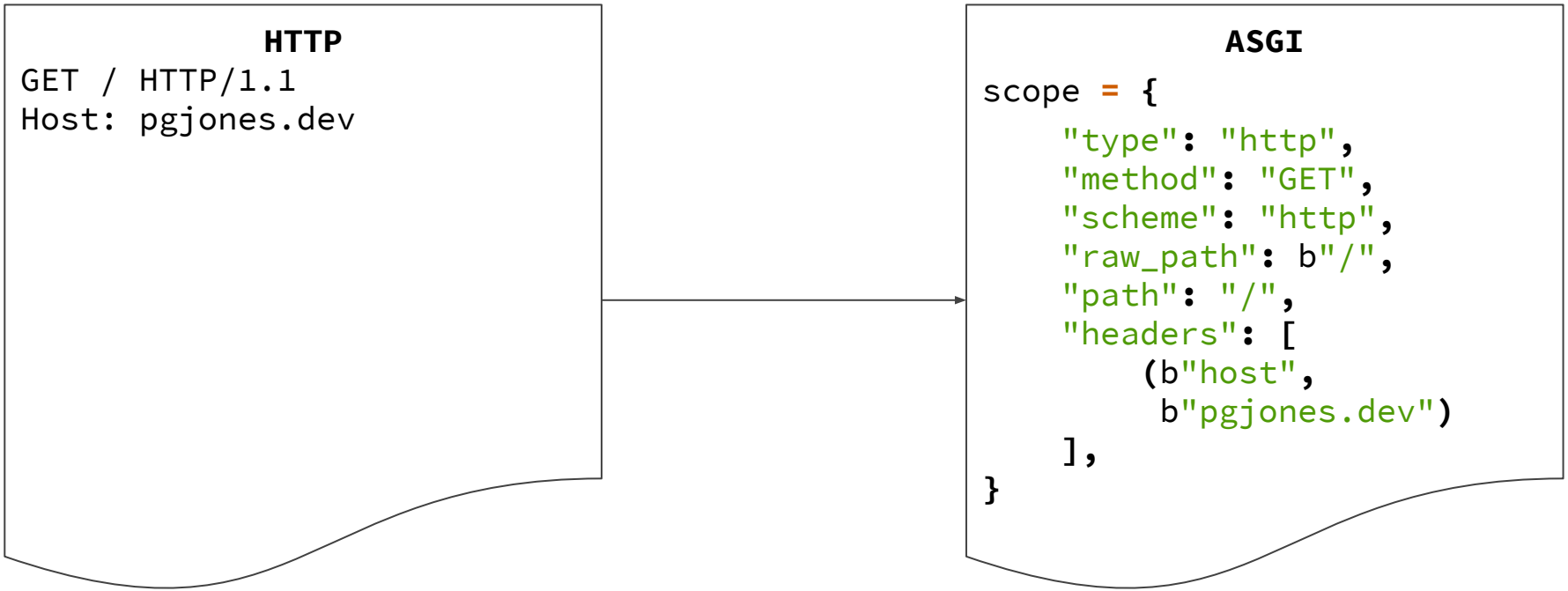
```
$ python http_server.py localhost 5006

b'POST' b'/' [(b'Host',
b'localhost:5006'), (b'User-Agent',
b'curl/7.64.1'), (b'Accept', b'*/*'),
(b'Content-Length', b'5'),
(b'Content-Type',
b'application/x-www-form-urlencoded')]
bytearray(b'Hello')
```

```
$ curl -v -d "Hello" localhost:5006/
* Connected to localhost (::1) port 5006
> POST / HTTP/1.1
> Host: localhost:5006
> User-Agent: curl/7.64.1
> Accept: */*
> Content-Length: 5
> Content-Type:
application/x-www-form-urlencoded
>
* upload completely sent off: 5 out of 5
bytes
< HTTP/1.1 200
< Content-Length: 0
<
* Closing connection 0
```

# Server Process



# HTTP -> ASGI -> App



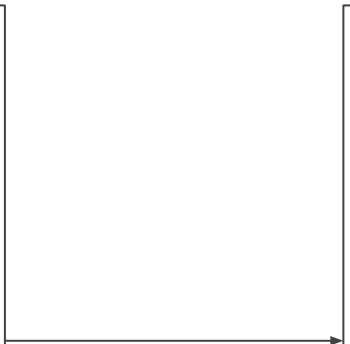
# ASGI Scope

---

```
def create_scope(parser):
    return {
        "type": "http",
        "method": parser.method,
        "scheme": "http",
        "raw_path": parser.path,
        "path": parser.path.decode(),
        "headers": parser.headers,
    }
```

# HTTP (Body) -> ASGI (Body) -> App

```
POST / HTTP/1.1
Host: pgjones.dev
Content-Length: 5

Hello
```



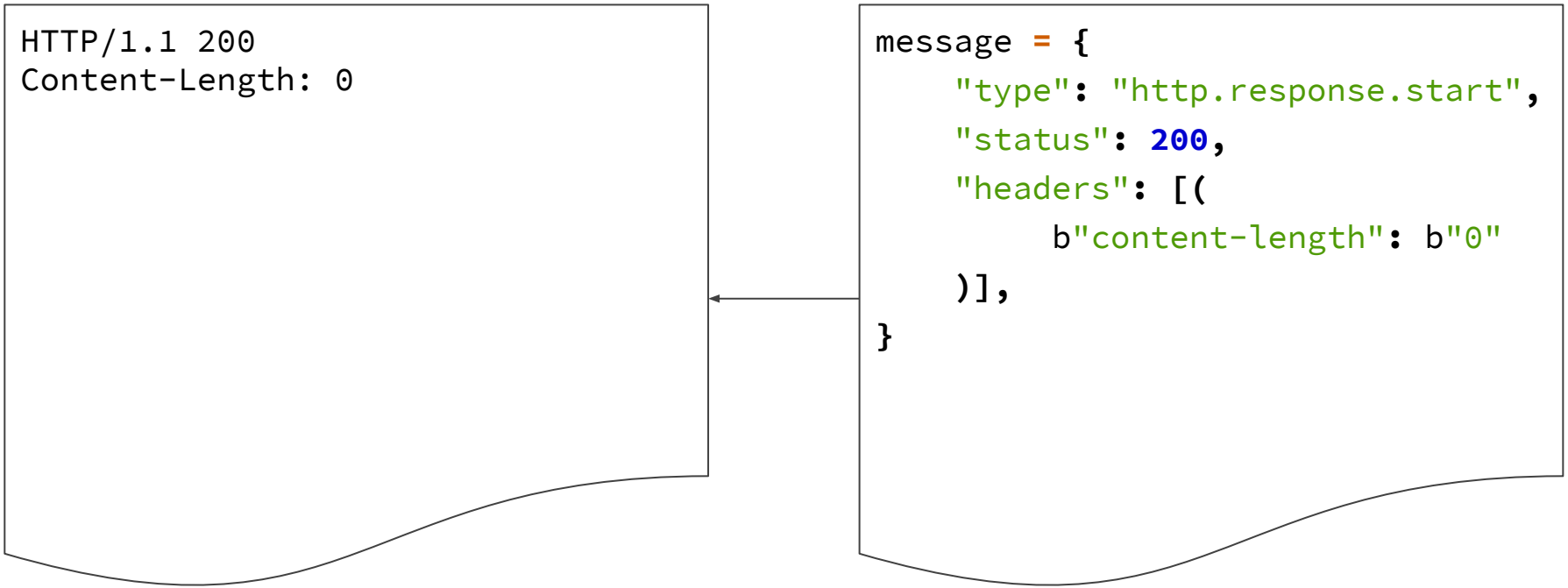
```
scope = {...}
message = {
    "type": "http.request",
    "body": "Hello",
    "more_body": False,
}
```

# ASGI messages

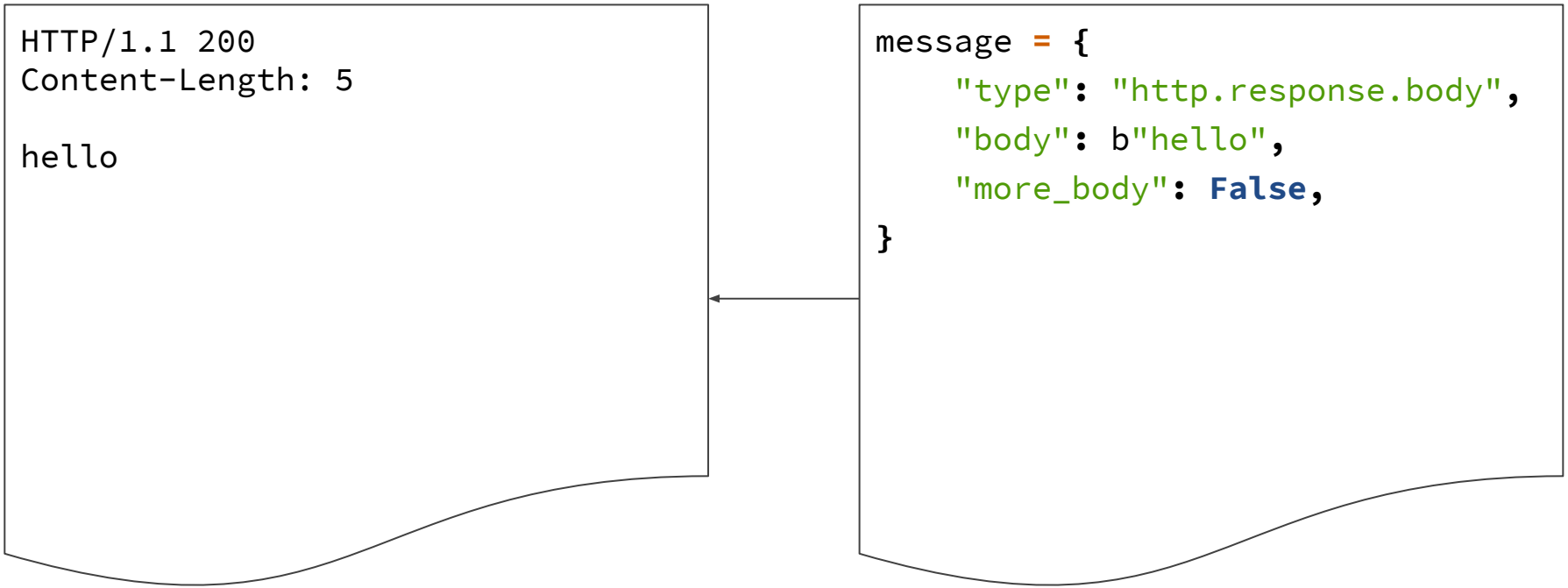
---

```
def create_message(body, more_body):
    return {
        "type": "http.request",
        "body": body,
        "more_body": more_body,
    }
```

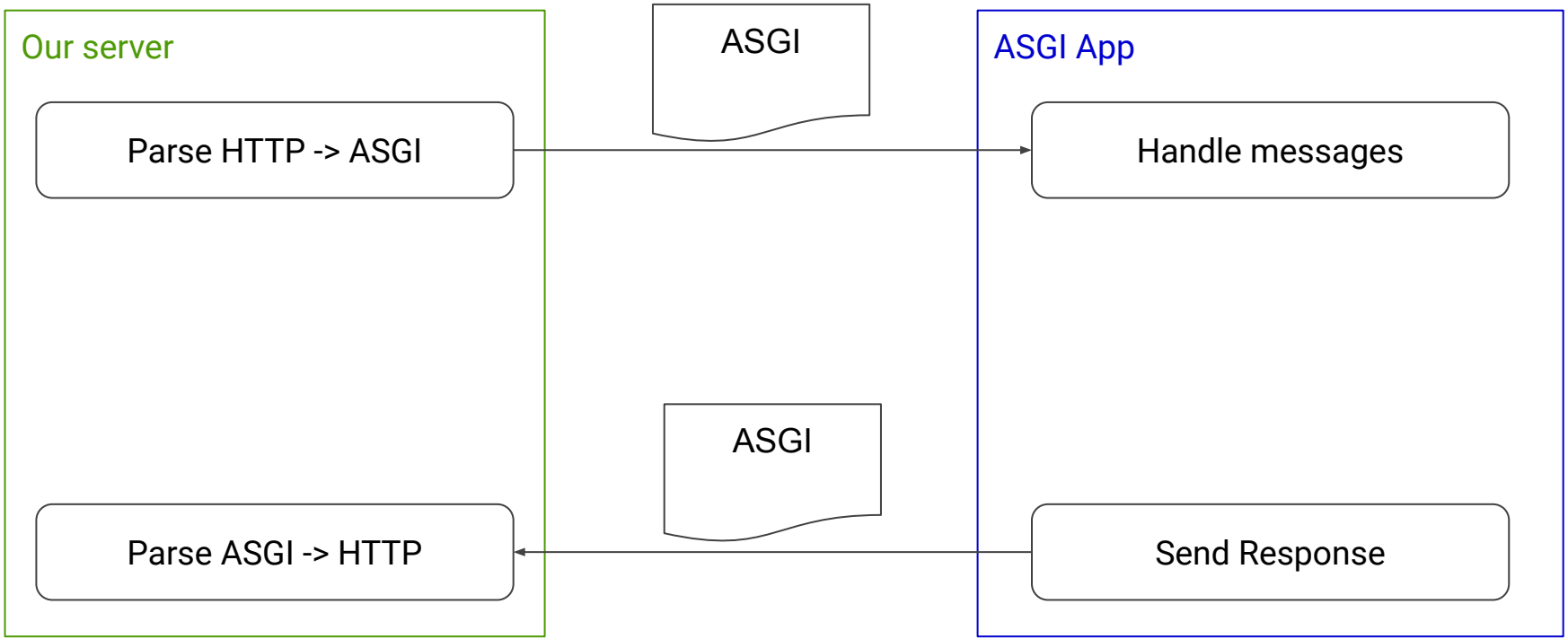
# App -> ASGI (Response) -> HTTP (Response)



# App -> ASGI (Response body) -> HTTP (Response body)



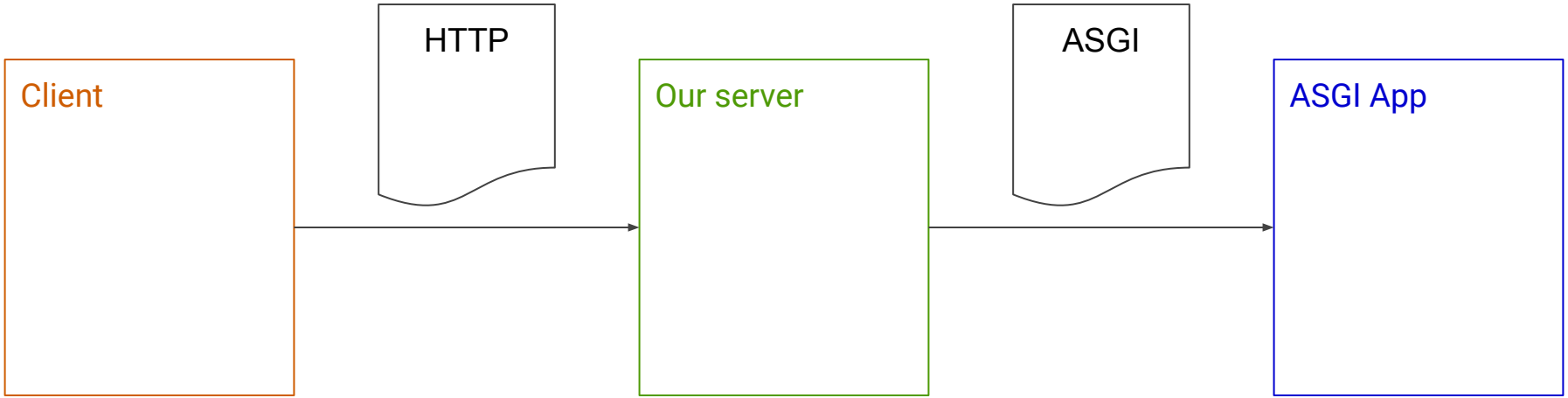
# Server Process



# ASGI App

```python
async def echo_app(scope, receive, send):
    body = bytearray()
    while True:
        event = await receive()
        if event["type"] == "http.request":
            body.extend(event.get("body", b""))
            if not event.get("more_body", False):
                break
    ...
    await send({
        "type": "http.response.start",
        "status": 200,
        "headers": [
            (b"Content-Length", b"%d" % len(body)),
        ],
    })
    await send({
        "type": "http.response.body",
        "body": body,
    })
```

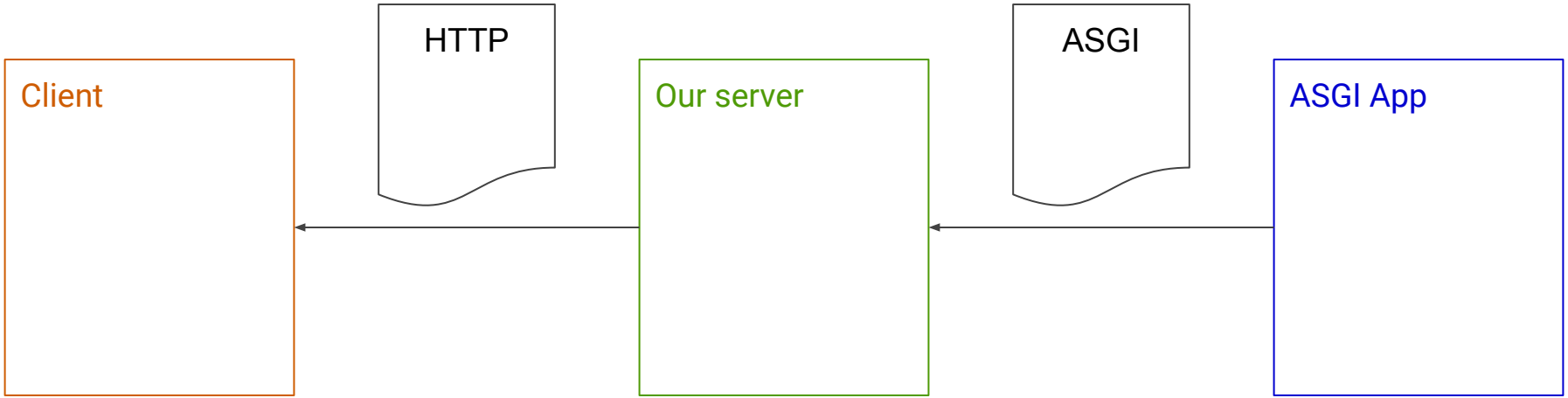
# Aim



# HTTP -> ASGI

```
async def asgi_http_parser_server(reader, writer):
    parser = HTTPParser()
    to_app = asyncio.Queue()
    read = 0
    while not reader.at_eof():
        if parser.part != "BODY":
            parser.feed_line(await reader.readline())
        elif parser.body_length == 0:
            await to_app.put(create_message(b"", False))
            break
        else:
            body = await reader.read(100)
            read += len(body)
            await to_app.put(
                create_message(body, read < parser.body_length)
            )
            if len(body) >= parser.body_length:
                break
    scope = create_scope(parser)
    ...
```

# Aim



# ASGI -> HTTP

```
...
from_app = asyncio.Queue()
await app(scope, to_app.get, from_app.put)
while True:
    message = await from_app.get()
    if message["type"] == "http.response.start":
        writer.write(b"HTTP/1.1 %d\r\n" % message["status"])
        for header in message["headers"]:
            writer.write(b"%s: %s\r\n" % (header))
        writer.write(b"\r\n")
    elif message["type"] == "http.response.body":
        if message.get("body") is not None:
            writer.write(message["body"])
        if not message.get("more_body", False):
            break

await writer.drain()
writer.close()
```

# ASGI Parsing server test

```
$ python asgi_http_parser_server.py\
  localhost 5008
```

```
$ curl -v -d "Hello" localhost:5008/
* Connected to localhost (::1) port 5008
> GET / HTTP/1.1
> Host: localhost:5008
> User-Agent: curl/7.64.1
> Accept: */*
>
< HTTP/1.1 200
< Content-Length: 5
< Content-Type: text/plain
<
* Connection #0 to host localhost left
intact
hello* Closing connection 0
```

# What next?


Asyncio, Trio

HTTP/1 [h11]
HTTP/2 [h2]
HTTP/3 [aioquic]

WebSockets [wsproto]

https://gitlab.com/pgjones/hypercorn