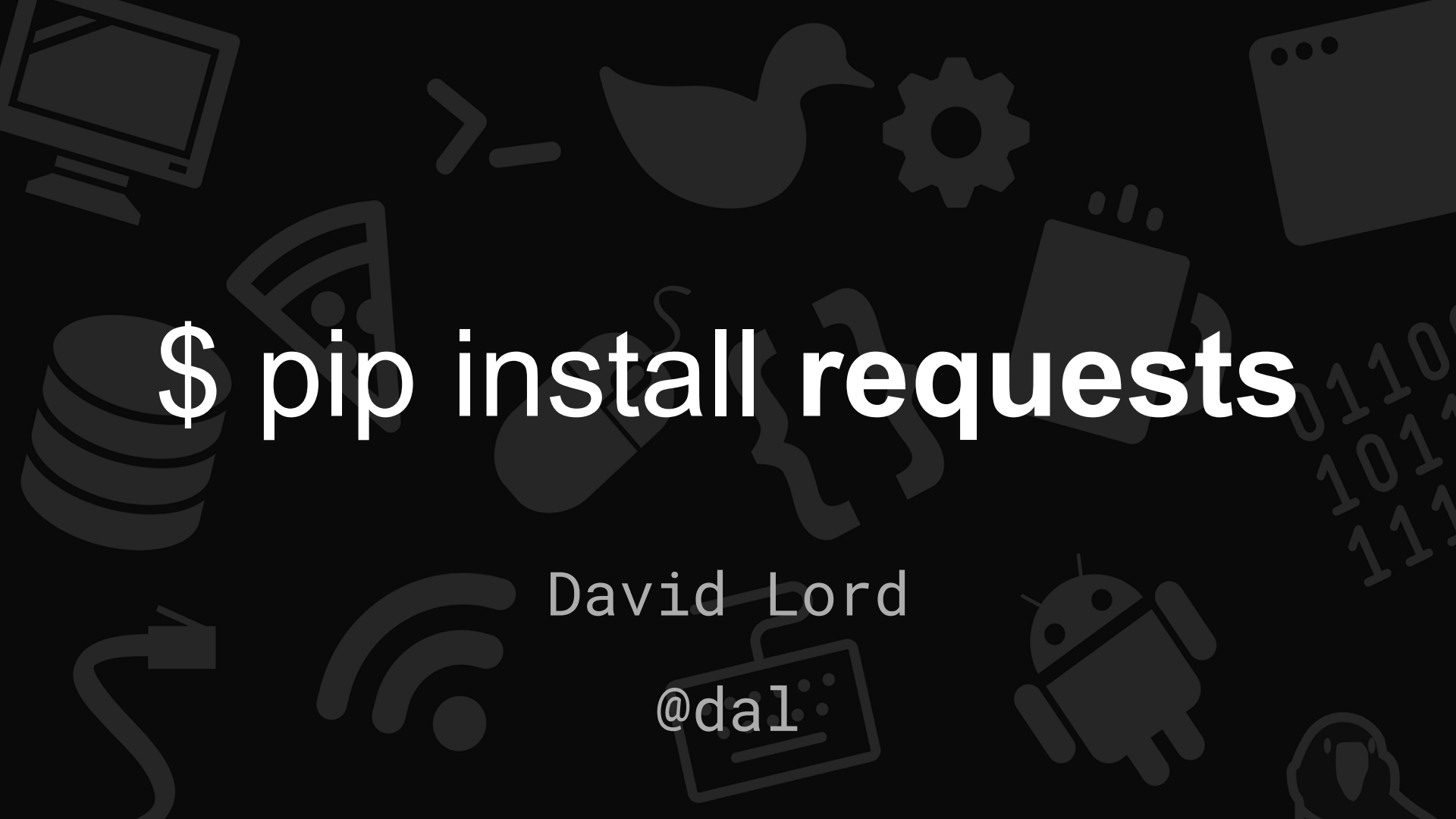


$ pip install **requests**

David Lord

@dal

HTTPS

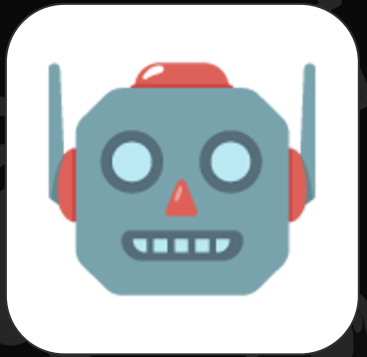
Websites

Application Programming Interfaces

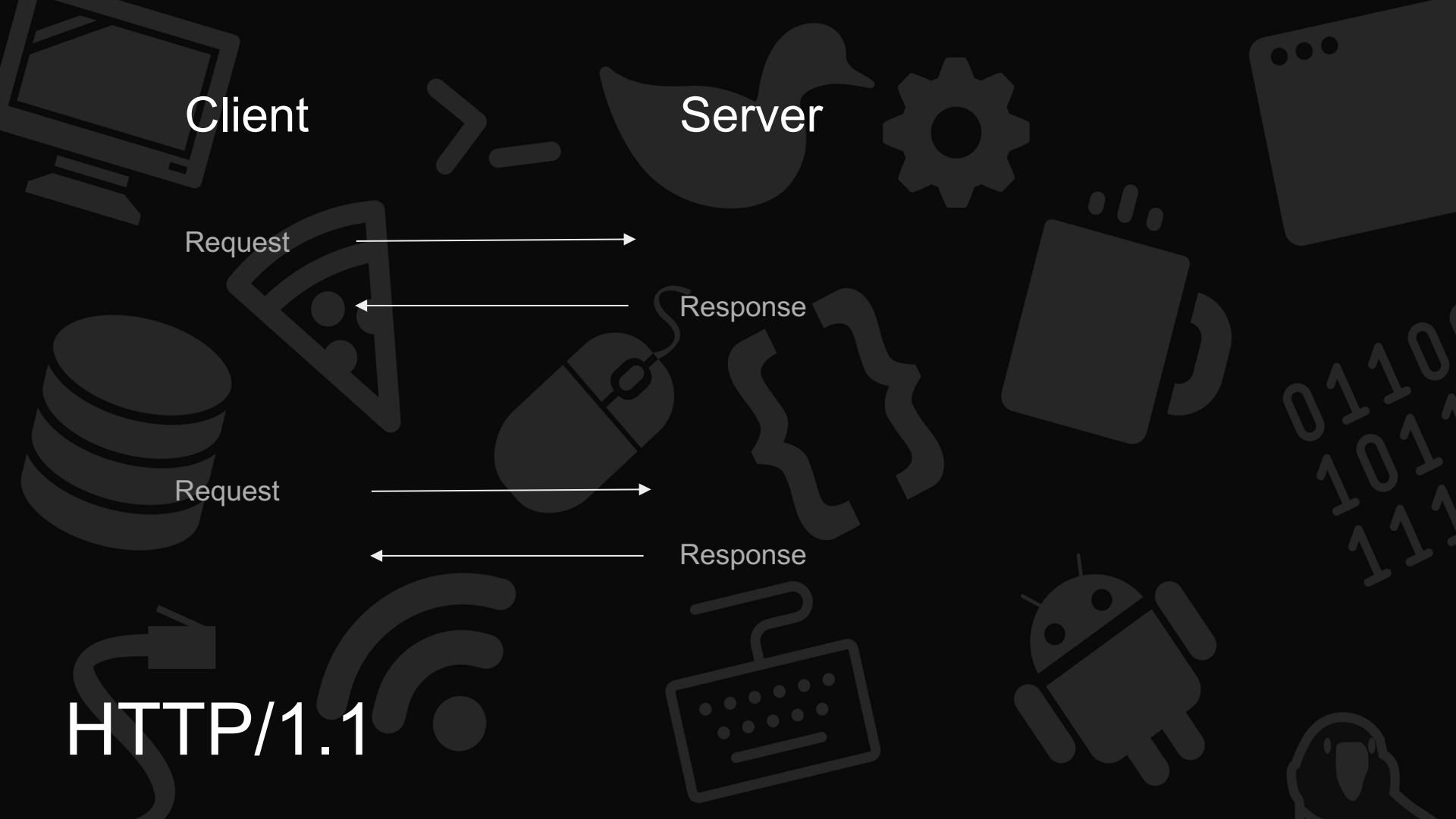
Mobile apps

Internet of Things

Slack bots



HyperText Transfer Protocol



HTTP/1.1

HTTP request: GET UQCS homepage

path

method → `GET /index.html HTTP/1.1` ← protocol

headers
as {key: value}

```
Host: www.uqcs.org.au
Accept: text/html, */*
Accept-Language: en-au
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

request body
(optional)

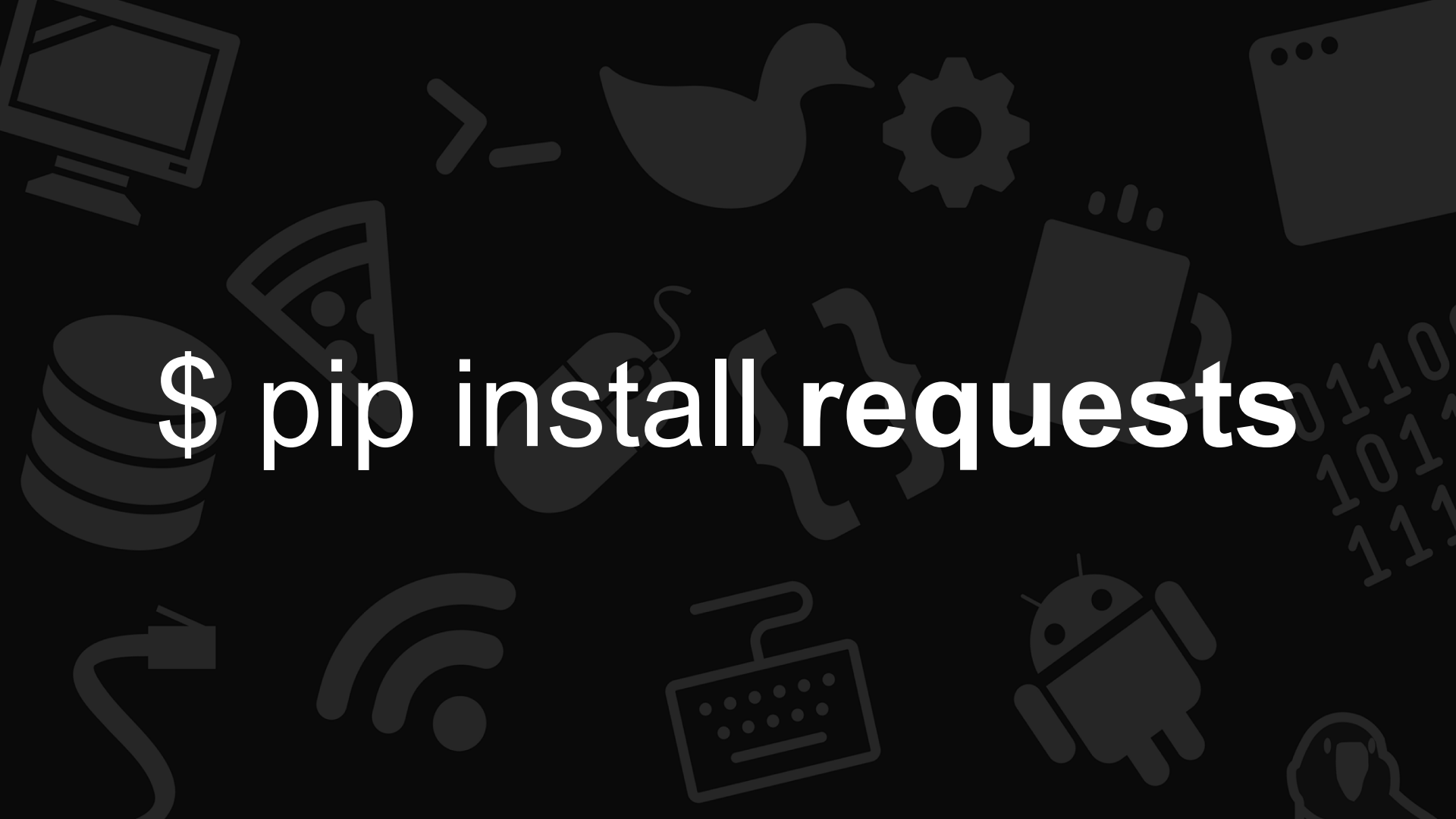
HTTP response

```
HTTP/1.1 200 OK

Content-Length: 48
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html>
<h1>Welcome</h1>
</html>
```

protocol response code

response headers

response body/content
(optional)


$ pip install **requests**

$ pip info **requests**

You are the **client** of the protocol

You send **requests** to the server

52 million downloads last month*

The most downloads of any of today's libraries 💪

* source

Example code: GET UQCS homepage

```
GET /index.html HTTP/1.1

Host: www.uqcs.org.au
Accept: text/html, */*
Accept-Language: en-au
Accept-Encoding: gzip,
deflate
Connection: keep-alive
```

requests.get(‘https://uqcs.org.au/index.html’)

Behind the scenes

- Requests sets up an HTTPS session (DNS, TLS, all that jazz)
- Crafts HTTP request with query parameters, headers, body
  - Including tricky things like content-length and encoding
- Sends request (in multiple packets) to server
- Blocks waiting for response and assembles it
- Provides response information as tidy object with status code, headers, body, etc

POST a form

Content-Type: ‘application/x-www-form-urlencoded’

The format of the web.

POST a form: request

raw

```
POST /usr/profile HTTP/1.1
Host: uqcs.org.au
User-Agent: sesame-1.0
Cookie: b20gbm9tIG5vbQ==
Content-Type: application/x-www-
form-urlencoded
Content-Length: 43

first_name=Sam&last_name=O%27Brien&
gender=X
```

code

```
url = 'https://uqcs.org.au/usr/profile'

headers = {'User-Agent': 'sesame-1.0',
           'Cookie': 'b20gbm9tIG5vbQ=='}

form_data = {'first_name': 'Sam',
             'last_name': "O'Brien",
             'gender': 'X'}

requests.post(url, headers=headers,
data=form_data)
```

POST a form: response

raw

```
HTTP/1.1 204 No Content
X-Powered-By: nginx/WAI/Haskell
```

code

```
response = ... # as before

>>> response.status_code
204

>>> response.headers['x-powered-by']
'nginx/WAI/Haskell'

>>> response.raise_for_status()
None
```

POST some JSON

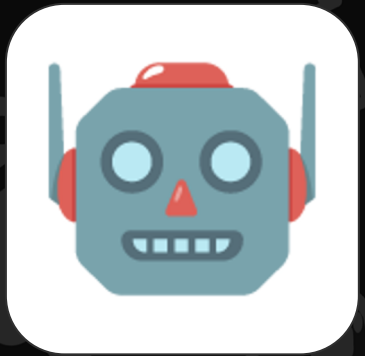
Content-Type: ‘application/json’

The format of the API gods.

JSON

JavaScript Object Notation

```
{
    "channel": "0AQQDE",
    "text": "Hello, world!",
    "blocks": [{
    "type": "section",
    "text": {
            "text": "*Hello, world!*",
            "type": "mrkdwn"}}]
}
```



Make a JSON request

```
response = requests.post('https://api.slack.com/ABCDE/31337',
        headers={
                'Authorization': 'Bearer 1234'},
        json={
                "channel": "0AQQDE",
                "text": "Hello, world!",
                "blocks": [{
                "type": "section",
                "text": {
                        "text": "*Hello, world!*",
                        "type": "mrkdwn"}}]
                })
```

Get a JSON response

```
>>> response.status_code
200

>>> response.json()
{'ok': True, 'channel': '0AQQDE'}

>>> response.json()['ok']
True

>>> response.json()['channel']
'0AQQDE'
```

Other features we don’t have time for

- HTTPS
- Authentication
- Timeouts
- Sessions
  - Cookie jar
  - Connection reuse
- TLS certificates
  - Client certificate
  - Server certificate trust/pinning
- Proxies
- Response streaming
- Parsing web pages