

Index

A

Access Control Lists (ACLs), 248
AccountFileChecker class, 244
AccountURLChecker, 242
aiohttp, 312
Allocator state machine, 268–270
asDeferred helper function, 313
assertNoResult, 278
async / await constructs, 288
Asynchronous exception handling
 callback method, 66–67
 deferred implementation, 70–71
 encounter exceptions, 68–69
 Errbacks, 68
Asynchronous Message Protocol
 (AMP), 128, 214
 commands, 218
 custom plugin, 216–218
Asynchronous programming, 3, 63
Asynchronous Server Gateway Interface
 (ASGI), 366
Asynchronous testing, deferreds, 277
asyncio package, 305
asyncio and Twisted
 event loops, 306
 asyncio.get_event_loop, 307
 asyncio.set_event_loop, 307
 epoll, Linux, 307
 event loop policies, 307
 kqueue, macOS, 307

 reactor, 307
 reactor selection, 307
 system event triggers, 308
 twisted.internet.reactor, 307
 PEP 3156, 305
 promises, 306
asyncio.Future class, 306
asyncio.get_event_loop_policy, 307
asyncio.set_event_loop_policy, 307
Async services, Buildbot's code
 addService, 325
 AsyncMultiService, 325
 ClientService class, 326
 disownServiceParent, 325
 IService, 324
 IServiceCollection, 324
 MessageQueueConnector, 326
 orthogonal problem, 324
 serialization, 324
 setServiceParent, 325
 startService, 324
Autobahn, 259, 286–287
Automat, 268

B

Biased coin toss, 215
Buildbot
 defined, 317
 history
 async build steps, 322–323

INDEX

Buildbot (*cont.*)

- Buildbot's Async Python, evolution
(*see* Buildbot's Async Python)

- HTTP API, 318

- Mozilla, 318

- synchronous APIs, 321–322

- Tinderbox, 317

- request/response cycle, 323

- synchronous approach, 323

Buildbot 0.9.0, 318

Buildbot's Async Python

- additional features, 320

- async/await, 321

- asyncio, 321

- Builder.startBuild method, 318

- callbacks, 319

- Deferred, 319

- deferredGenerator, 320

- errors, 319

- inlineCallbacks, 320

- NodeJS, 321

- synchronous model, 321

- Twisted Python, 318

Buildbot's code

- asynchronous tools, 323

- automated testing, 336

- concurrency barriers, 333–334

- db module, 328

- DB API, 338

- debouncing, 323–324

- Deferred, testing, 337

- Docker protocol, 332–333

- fake components, 338

- flushEventualQueue

 - method, 327

- HTTP API, 329

- HttpClientService, 330

- HTTP sessions, 330–332

- LRU caches, 327

- reactor thread, 333

- re-entrancy, 333

- SQLAlchemy, 328–329

- synchronous libraries, 327

- thread-pool functions, 334–335

- Trial, 337

- buildbot.test.fake.fakemaster.FakeMaster

 - class, 338

C

- callLater, 353

 - logical parts, 354–355

- Canonical multi-tier architecture, 370

- Case study, HTTP proxy with

 - aiohhttp and treq

 - aiohhttp server, 312

 - aiohhttp.web.StreamResponse, 315

 - asFuture helper, 315

 - asyncio reactor, 313

 - coroutine, 313

 - Deferred.fromFuture, 313

 - ensureDeferred, 313

 - handler, 313

 - Twisted's Headers, 315

 - HTTP headers, 315

 - inlineCallbacks, 313

 - proxy, implementation, 313–314

 - URL echoing service, 313

 - web browser, 316

- CGI standard

 - environment parameter, 182

 - environment variables, 182

- Clock() instance, 283

- Common gateway interface (CGI), 179

- Connection hints, 257

- Content-Disposition header, 238

- Content Distribution Network (CDN), 197
- Content-hash key (CHK), 225
- Continuation-passing style technique, 62
- `_convert_error`, errback handler, 247
- Coroutines
 - async method, 90
 - await methods, 91, 95, 97
 - ensureDeferred, 97–98
 - future-like object, 94–95
 - JIT, 98
 - in python, yields from
 - generator, 88–89
 - send, throw method, 90
 - stack of generators, 92
- CORS policy, 294
- Crossbar.io, 296–297, 299
- Cryptographic hash function, 238

- D**
- Daemonization tools
 - Tahoe command-line tool, 231
 - .tap files, 231
- Daphne, 367
- `defer.cancel()`, 263
- `Deferred.addCallback` function, 306
- `Deferred.fromFuture` class method, 311
- `DeferredLock`, 336
- `DeferredQueue`, 336
- Deferreds, 272–274, 306
 - cancellation, 263
 - vs.* state machines, 268
- `DeferredSemaphore`, 336
- Delegate API, 268
- `DelegatingResource`, 201
- Demultiplexing, 8
- Dependency load, 230

- Diffie-Hellman key-exchange protocol, 255
- Dircap, 225
- Distributed hash table (DHT), 260
- Distributed multi-layer systems, 370
- Django Channels, 365–366
 - components of, 367
 - current status, 371
 - future directions, 371
- Docker, 157
 - build images, 160–161
 - client, 159
 - container images, 158–159
 - containers, 157–158
 - defining, 157
 - multi-stage build, 161–163
 - Python on
 - automation with `dockerpy`, 173
 - built options, 172
 - copying environment, stages, 173
 - copying Pex executable file, stages, 173
 - copying wheels, stages, 172–173
 - full environment
 - deployment, 163, 165–168
 - Pex file, 170–171
 - virtualenv, 169–170
 - registry, 160
 - runc and containerd, 159
 - Twisted on
 - custom plugin, 174
 - ENTRYPOINT and PID 1, 174
 - NColony, 175–178
- Docker API library, 332
- Dynamic configuration
 - A/B testable pyramid app, 214–215
 - AMP, custom plugin, 216–219
 - control program, 219–220

E

- Event-driven programming, 3, 63
 - defining, 4
 - event handlers, 7
 - event loop, 7
 - events, 7
 - flow control, 49–50
 - value of, 33–34, 36
- Event handlers, 6–7
 - requestField, 60
- Event loop, 6–7
- Events
 - interfaces with `zope.interface`, 46, 48–49
 - multiple, 5–7
 - in time
 - complexity, 42
 - DelayedCall instance, 41–43
 - with LoopingCall, 44–46
 - reactor.callLater, 41

F

- failureResultOf(), 278, 281
- Feed aggregation, 110–111
 - first draft
 - deferreds, 125
 - handlers, 123
 - implementation, 126–127
 - klein.Plating object, 121
 - Plating.CONTENT
 - slot, 122
 - render method, 125
 - retrieveFeed method, 125
 - SimpleFeedAggregation
 - class, 123–125
 - Slow Increment page, 122
 - project directory structure, 129

- File resource, 199
- Filecaps, 225
- File-transfer protocol (FTP), 223
- Flow control
 - event-driven programming, 49–50
 - Twisted
 - consumers, 54–57
 - push producers, 51–54, 57
- flushEventualQueue() function, 282
- Front-end protocols, 233
- FTPAvatarID object, 244
- FTP front end
 - avatar, 243
 - checker, 242
 - endpoint, 242
 - portal, 243
 - protocol factory, 242
 - realm, 243
 - root dircap, 242
- Function composition tool, 60
- Future.add_done_callback, 306
- Futures, 272

G

- Generators
 - generatorFunction, 81
 - send method, 81, 83
 - throw method, 84–85
 - yield expression, 80–81
- get_size() method, 236
- get_tor() function, 279–280
- Guidelines, asyncio and Twisted
 - asyncio.Futures, 309
 - asyncio's coroutines, 308
 - code, interoperability, 310
 - Deferreds, 308, 311
 - IReactorCore's API, 308

Task, 309
 task.react, 311
 Twisted's asyncio reactor, 310
 Twisted's deferreds, 309

H

HAProxy, 208
 H2Stream, 347
 HTTP/2, 339–340

- backpressure, 359–362
- multiplexing, 348–350
 - problem, 350
 - priority, 351–352

 HTTP AJAX request, 285
 HTTP headers, 183
 HTTP long polling, 259
 HTTP processing chain, 240
 HTTP protocol, 237
 HTTP proxy, 305
 HTTP PUT and POST actions, 228
 HTTP response code, 183
 HyperText Transfer Protocol (HTTP), 339

I

IANA standard, 185
 inlineCallbacks

- deferred execution, 85
- requestFieldGenerator, 86–88

 @inlineCallbacks function, 278
 Integration test, 278
 Interface, 342

- polymorphism, 342

 Internet Engineering Task Force (IETF), 339
 I/O Completion Ports (IOCP), 63
 IPushProducer, 359–360, 362

IReactor, 353
 IReactorCore.addSystemEventTrigger, 308
 IReadable's read() method, 236
 ITransport, 345, 347

J

Jenkins, 336
 Just In Time (JIT), 98

K

Klein, 109

- amount argument, 116
- decodes message, 116
- decorator, 115
- Deferreds, 117–118
- templates with plating, 118–119, 121
- test-driven development (*see* Test-driven development)
- URL pattern, 116
- virtual environment, 115
- Werkzeug's routing documentation, 115

L

libp2p protocol, 260
 list() method, 246
 Load balancer, 208–210

M

Magic wormhole tool

- code, 253
- Python-based command-line tool, 254
- receiver screenshot, 254
- sender screenshot, 254
- workflow diagram, 255
- working, 255, 257

INDEX

makePromiseResolverPair() function, 272
Merkle hash trees, 238
Message broker/queueing system, 368–369
mkdir command, 226
Multiplexing

- busy wait, 8
- defining, 8
- event loop, 8
- mainloop, 7
- sockets, 9

Mutable slots

- readcaps, 225
- writecaps, 225

MySQL, 328

N

NColony, 175–178
Network drive indicator, 226
Network protocols and client compatibility, 258
Non-blocking IO

- complex programs, 23
- tracking state, 19–23
- when to stop, 18–19

O

One-shot observers, 271–272
originalCoroutine object, 310
originalDeferred object, 310
originalFuture object, 310

P, Q

Packaging/distribution, 230
Pattern match routing model, 197
PEP 3153, 305
PEP 3156, 305

Pipeline, 272
pip tool, 110
_populate_row() method, 246
PostgreSQL, 328
Promises, 272

- arrow function, 273
- Deferred.addCallback, 306
- Deferreds and Futures, behavior, 306
- event loop, 307–308
- TC39 standards organization, 273

Publish/Subscribe (PUB/SUB), 287, 300
Pyramid framework, 186–187
Python, 365

- Autobahn library, 287
- on Docker
 - automation with dockerpy, 173
 - built options, 172
 - copying environment, stages, 173
 - copying Pex executable file, stages, 173
 - copying wheels, stages, 172–173
 - full environment
 - deployment, 163, 165–168
 - Pex file, 170–171
 - virtualenv, 169–170
 - versions, 4
- Python 3, 230
- Python and JavaScript, raw Websocket
 - HTML code, 293–294
 - Python server, 292–294
- Python Enhancement Proposal (PEP), 180–181
- PYTHONPATH environment variable, 192
- python-requests library, 329
- Python to Python, raw Websocket
 - autobahn, 290–291
 - client console, 291
 - echo server, 288–289

self.sendMessage(), 292
 server console, 292
 Python 3 virtualenv, 287

R

Raspberry Pi, 286
 reactor.seconds(), 205
 read() method, 238
 Rendezvous server, 255
 architecture, 260–261
 database, 262
 Request.write, 345

S

sans-io HTTP/2, 344
 Secure Socket Layer (SSL), 194
 select multiplexer
 event-driven clients and servers, 15–18
 event loop with, 13–15
 handling events, 12–13
 history, siblings, and purpose, 9
 socket events, 11
 client and server, 12
 readable event, 11
 timeout, 11
 writable event, 11
 and sockets, 9–10
 self.assertFailure() clause, 277
 Server name indication (SNI), 195, 197
 Server sent events, 259
 SFTP front end, 248
 Single Page Applications (SPA), 123
 SPAKE2 protocol
 diagram, 256
 protocol, 256
 SQLite, 328

SSH secure shell encryption layer, 248
 start_response parameter, 182
 Static files
 CDN, 197–198
 leaf resource, 199
 manifest, 199
 postpath, 202
 setup.py, 199
 superclass constructor, 202
 WSGI, 200–202
 successResultOf(), 278, 281
 sum() function, 298
 SynchronousTestCase, 337
 Synchronous testing
 deferreds tools, 278
 eventual send, 275–276, 281, 283

T, U, V

Tahoe-LAFS
 architecture, 227–229
 clients, 224
 distributed storage system, 223
 FTP, 223
 grid diagram, 224
 introducers, 224
 servers, 224
 twisted usage, 229–230
 .tap files, 231
 Test-driven development
 installable project, test on, 128–131
 interface, 128
 Klein with StubTreq
 Channel and Item classes, 138
 FeedAggregation instance, 139
 FeedAggregationTests, 133
 HTML and JSON feed
 renderings, 133, 135

INDEX

Test-driven development (*cont.*)

- HTTP services, 131
 - implementation, 135, 136
 - law of Demeter, 133
 - resource() method, 132
 - root method, 133, 136
 - _service.py, 137, 138
 - setUp method, 132
 - solution, 137
 - XPaths, 135
 - logging with twisted.logger,
 - 143–144, 146, 148
 - running twisted applications, 149–154
 - testing treq with Klein, 140–143
- then() method, 273
- Transit client, 262–265
- Transit connection, 257
- Transit relay server, 265–266
- Transport layer security (TLS), 194
- encryption, 194
 - endpoint authentication, 194
- Transports and protocols
- clients and servers, 31–32
 - dataReceived and connectionLost
 - methods, 26
 - interface, 25
 - managing complexity, 23–24
- PingPongProtocol, 26
- behaviors, 27
 - control flow, 31
 - dataReceived records, 27
 - doRead and doWrite
 - mirror, 30
 - identity string, 27
 - loseConnection, 30
 - _onCompletion, 30
 - read function, 28, 30
 - Transport's interface, 27–28

- write method, 28
 - Protocol interface, 25
 - reactors, 24–25
 - read function, 25
 - Twisted and reactors, 33
- transport.write, 345, 353
- Travis-CI, 336
- treq, 109, 312
- client API, 112
 - decorator, 113
 - Deferred, 112–113
 - dependency injection, 112
 - download function, 112
 - encapsulation, 110
 - feed aggregator, 111
 - feedparser, 113–114
 - library, 330
 - test-driven development (*see* Test-driven development)
 - virtual environment
 - (virtualenv), 111, 112
- Trial, 337
- Twisted, 305
- APIs
 - filepath permission, 250
 - IFTShell interface, 249
 - pipsi tool, 250
 - on Docker
 - custom plugin, 174
 - ENTRYPOINT and PID 1, 174
 - NColony, 175–178
 - flow control system
 - consumers, 54–57
 - push producers, 51–54, 57
 - tap plugin, 189
 - and Real World, 36–38, 40
- Twisted HTTP/2, 353
- backpressure, 357, 359

- current status and future
 - expansion, 362–363
- design goals
 - optimized behavior, 343
 - reusing code, 343–344
 - seamless integration, 341
- objects of, 345
- parameters affecting, 343
- twisted.internet.task.react, 307
- Twisted’s deferred
 - addCallback method, 71, 73
 - callbacks, 76, 78, 79
 - data flow, 78
 - errback method, 73, 76
 - event handlers
 - continuation-passing style, 62
 - function composition, 60–61
 - onCompletion callback, 62
 - requestField, 59–60
 - failures, 74
 - multiplexing, 99–101
 - placeholder, 63–64, 66
 - testing, 102, 104–106
- txrequests library, 330
- txtorcon variable, 279

W, X, Y, Z

- Web Application Programming Interface (WAPI), 234

- Web front end

- directories, 237
- disk saving, 238
- FileNodeHandler, 235
- HTTP error codes, 240–241
- Nevow, 236
- range header, 238–239
- root resource, 234

- URIHandler resource, 234
- WAPI call, 234
- WebOb package, 185–186
- Web Server Gateway Interface (WSGI), 365
- WebSocket, 259
 - authentication platform, 286
 - caching processes, 286
 - defined, 286
 - propagating events, 285
 - pushing notifications, 285
 - signaling changes, 285
- Twisted
 - Autobahn ecosystem, 286
 - autobahn, installing, 287
 - micro-service architecture, 286
 - remote code, 287
 - WAMP, 287
 - website notifications, 286
- WebSocket Application Messaging Protocol (WAMP), 287
 - authentication, 295
 - broadcast messages, 295
 - client call, 296
 - client code, 298
 - compatible router, 296
 - errors, 295
 - events, 295
 - html extension, 303
 - JS client, 299–301
 - manage errors, 295
 - pair up messages, 294
 - py extension, 303
 - Python client, 300–301
 - realm, 298
 - routed RPC, 300
 - RPC part, 295
 - serialization, 295
 - win32api, 296

INDEX

Web stack, divisions, 366

Web Standard Gateway Interface
(WSGI), 179

See also WSGI server

Web user interface (WUI), 234

Werkzeug, 109

Wormhole client

architecture, 267

mailbox, 267

Nameplate, 267

wsgiref module, 183–185

WSGI server, 297

application, 181–182

built-in scheduled tasks, 203, 205–206

callable Python object, 185

custom plugin, 189

default path, 191

demo web application, 188

--port option, 188

production *vs.* development, 192–193

python-m twisted, 188

setup.py, 192