Log Better

Why it’s important and how to do it well

Laura Santamaria
Developer Advocate

@nimbinatus
#logbetter

Interactive!

Obligatory “Why?” slide

Legal/regulatory requirements

Monitoring/tracing

Action!

Actionable logs!

Enough info

Enough details

Enough history

Text Logs

Past best practices

Plain language strings

Tokenizable strings

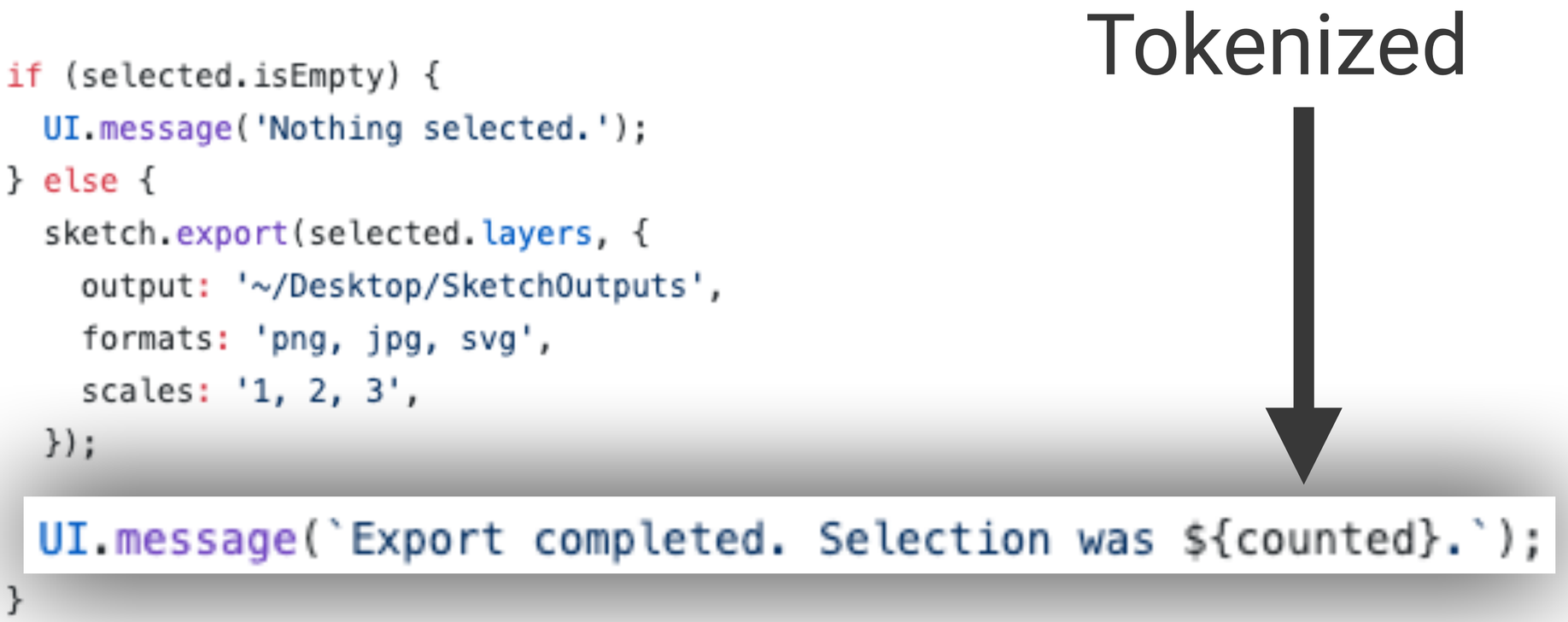
```
if (selected.isEmpty) {
  UI.message('Nothing selected.');
} else {
  sketch.export(selected.layers, {
    output: '~/Desktop/SketchOutputs',
    formats: 'png, jpg, svg',
    scales: '1, 2, 3',
  });

  UI.message(`Export completed. Selection was ${counted}.`);
}
```

```
if (selected.isEmpty) {
  UI.message('Nothing selected.');
} else {
  sketch.export(selected.layers, {
    output: '~/Desktop/SketchOutputs',
    formats: 'png, jpg, svg',
    scales: '1, 2, 3',
  });

  UI.message(`Export completed. Selection was ${counted}.`);
}
```

Plain language strings

```
if (selected.isEmpty) {
  UI.message('Nothing selected.');
} else {
  sketch.export(selected.layers, {
    output: '~/Desktop/SketchOutputs',
    formats: 'png, jpg, svg',
    scales: '1, 2, 3',
  });

  UI.message(`Export completed. Selection was ${counted}.`);
}
```

Tokenized



```
if (selected.isEmpty) {
  UI.message('Nothing selected.');
} else {
  sketch.export(selected.layers, {
    output: '~/Desktop/SketchOutputs',
    formats: 'png, jpg, svg',
    scales: '1, 2, 3',
  });

  UI.message(`Export completed. Selection was ${counted}.`);
}
```

Try It!

https://logbetter.nimbinatus.com

Structured Logs

Best practices now

Timestamp plus object

Structured data

Single human-readable field

```python
@cherrypy.expose
class JsonLogEndpoint(object):
    def OPTIONS(self):
        return

    @cherrypy.tools.accept(media='application/json')
    @cherrypy.tools.json_in()
    @cherrypy.tools.json_out()
    def POST(self):
        log = logger.bind(
            user_agent=cherrypy.request.headers.get("HTTP_USER_AGENT", "UNKNOWN"),
            peer_ip=cherrypy.request.headers.get("REMOTE_ADDR", "0.0.0.0"),
            forwarded_ip=cherrypy.request.headers.get("HTTP_X_FORWARDED_FOR", "0.0.0.0")
        )
        returnstring = []
        try:
            dataSet = cherrypy.request.json
            for elem in dataSet:
                # log.msg("LOG: Data {} as {}".format(elem, dataSet[elem]))
                returnstring.append("Log: {} as {}".format(elem, dataSet[elem]))
            cherrypy.response.body = '\n'.join(returnstring).encode('utf-8')
            cherrypy.response.status = 200
        except Exception as err:
            cherrypy.response.status = 400
            cherrypy.response.body = "{}".format(err)
        finally:
            log.msg("LOG: Data incoming", data=dataSet)
            log.msg("RES: {} with {}".format(cherrypy.response.status, cherrypy.response.body))
            return cherrypy.response.body
```

```python
@cherrypy.expose
class JsonLogEndpoint(object):
    def OPTIONS(self):
        return

    @cherrypy.tools.accept(media='application/json')
    @cherrypy.tools.json_in()
    @cherrypy.tools.json_out()
    def POST(self):
        log = logger.bind(
            user_agent=cherrypy.request.headers.get("HTTP_USER_AGENT", "UNKNOWN"),
            peer_ip=cherrypy.request.headers.get("REMOTE_ADDR", "0.0.0.0"),
            forwarded_ip=cherrypy.request.headers.get("HTTP_X_FORWARDED_FOR", "0.0.0.0")
        )
        returnstring = []
        try:
            dataSet = cherrypy.request.json
            for elem in dataSet:
                # log.msg("LOG: Data {} as {}".format(elem, dataSet[elem]))
                returnstring.append("Log: {} as {}".format(elem, dataSet[elem]))
            cherrypy.response.body = '\n'.join(returnstring).encode('utf-8')
            cherrypy.response.status = 200
        except Exception as err:
            cherrypy.response.status = 400
            cherrypy.response.body = "{}".format(err)
        finally:
            log.msg("LOG: Data incoming", data=dataSet)
            log.msg("RES: {} with {}".format(cherrypy.response.status, cherrypy.response.body))
            return cherrypy.response.body
```

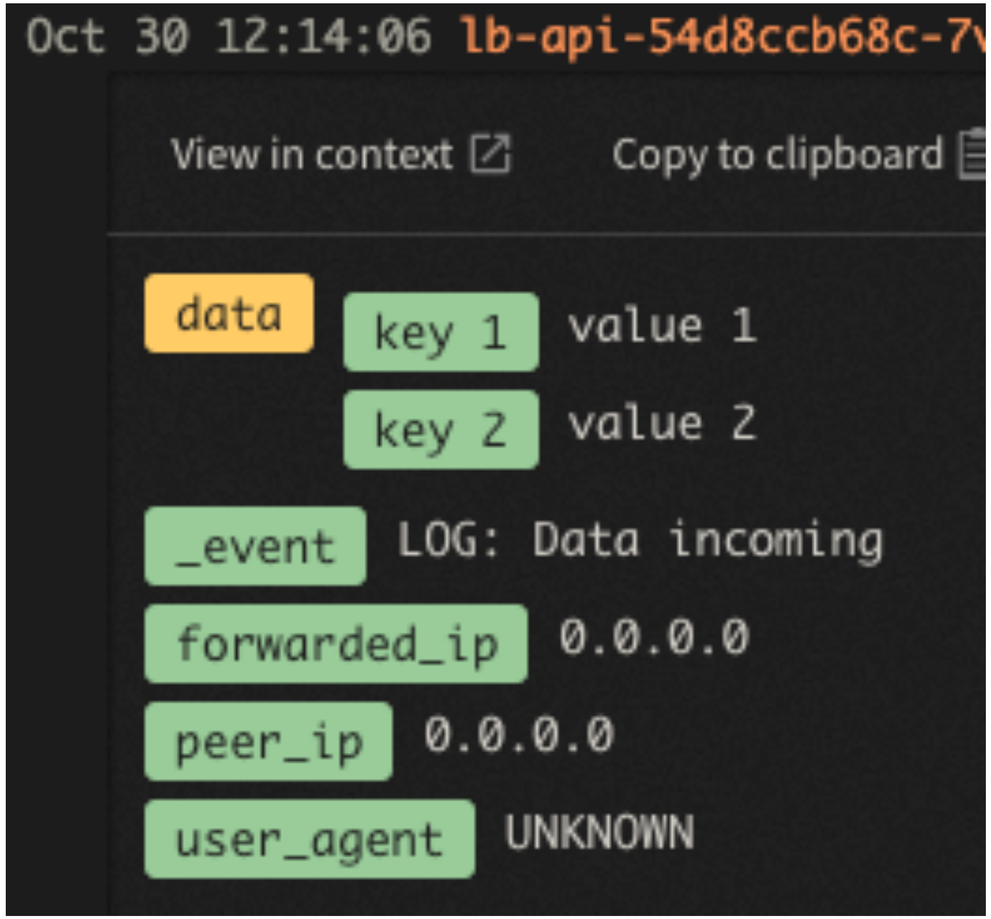
Structured objects

```python
@cherrypy.expose
class JsonLogEndpoint(object):
    def OPTIONS(self):
        return

    @cherrypy.tools.accept(media='application/json')
    @cherrypy.tools.json_in()
    @cherrypy.tools.json_out()
    def POST(self):
        log = logger.bind(
            user_agent=cherrypy.request.headers.get("HTTP_USER_AGENT", "UNKNOWN"),
            peer_ip=cherrypy.request.headers.get("REMOTE_ADDR", "0.0.0.0"),
            forwarded_ip=cherrypy.request.headers.get("HTTP_X_FORWARDED_FOR", "0.0.0.0")
        )
        returnstring = []
        try:
            dataSet = cherrypy.request.json
            for elem in dataSet:
                # log.msg("LOG: Data {} as {}".format(elem, dataSet[elem]))
                returnstring.append("Log: {} as {}".format(elem, dataSet[elem]))
            cherrypy.response.body = '\n'.join(returnstring).encode('utf-8')
            cherrypy.response.status = 200
        except Exception as err:
            cherrypy.response.status = 400
            cherrypy.response.body = "{}".format(err)
        finally:
            log.msg("LOG: Data incoming", data=dataSet)
            log.msg("RES: {} with {}".format(cherrypy.response.status, cherrypy.response.body))
            return cherrypy.response.body
```

Human readable field

logdna

```python
@cherrypy.expose
class JsonLogEndpoint(object):
    def OPTIONS(self):
        return

    @cherrypy.tools.accept(media='application/json')
    @cherrypy.tools.json_in()
    @cherrypy.tools.json_out()
    def POST(self):
        log = logger.bind(
            user_agent=cherrypy.request.headers.get("HTTP_USER_AGENT", "UNKNOWN"),
            peer_ip=cherrypy.request.headers.get("REMOTE_ADDR", "0.0.0.0"),
            forwarded_ip=cherrypy.request.headers.get("HTTP_X_FORWARDED_FOR", "0.0.0.0")
        )
        returnstring = []
        try:
            dataSet = cherrypy.request.json
            for elem in dataSet:
                # log.msg("LOG: Data {} as {}".format(elem, dataSet[elem]))
                returnstring.append("Log: {} as {}".format(elem, dataSet[elem]))
            cherrypy.response.body = '\n'.join(returnstring).encode('utf-8')
            cherrypy.response.status = 200
        except Exception as err:
            cherrypy.response.status = 400
            cherrypy.response.body = "{}".format(err)
        finally:
            log.msg("LOG: Data incoming", data=dataSet)
            log.msg("RES: {} with {}".format(cherrypy.response.status, cherrypy.response.body))
            return cherrypy.response.body
```



Try It!

https://logbetter.nimbinatus.com

Practical Notes

Levels

Debug

Trace

Info

Warn

Error

Critical

Fatal

Log types

System

Application

Audit

Text vs Structured

"But they look the same"

"But they look the same"

Only if your system flattens it

"But I can parse text"

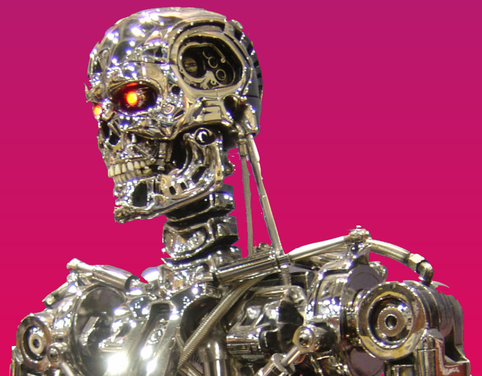
"But I can parse text"

That will take time when your system is generating the data anyway.

It's a way of thinking.

Remember: You are no longer the primary user

Remember: You are no longer the primary user



https://commons.wikimedia.org/wiki/File:Comic-Con_2004_-_Terminator_statue.jpg

logdna

Why

Time

Milliseconds instead of minutes

Cleanliness

Clear sections in the output instead of one longer line

Consistency

Multi-platform. Enough said.

Misconceptions

Logging is just printing to stdout.

No. Just no.

Log everything!

Probably will cause a performance hit

Logs are automatic.

You need to make logs, and you need to generate good ones.

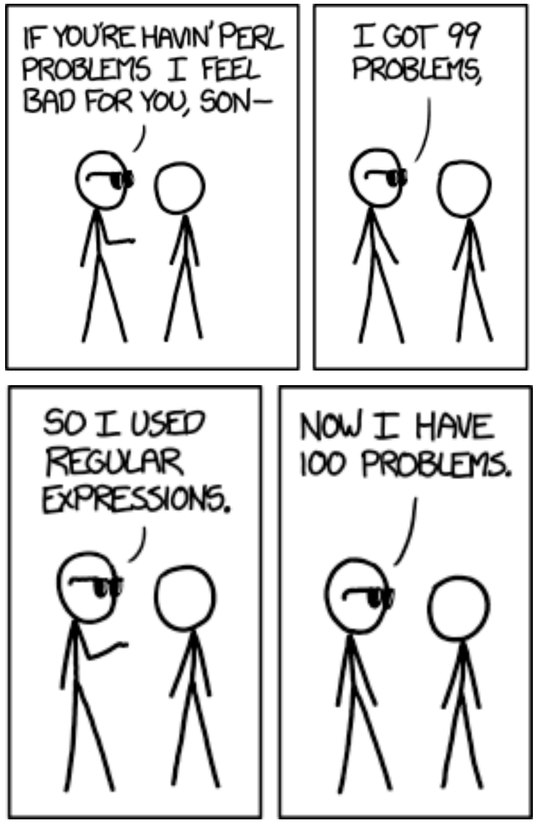
All those log levels are unnecessary.

Parse all the things!

Just use regular expressions.

Just use regular expressions.

... regex.



https://xkcd.com/1171/

Converting Your Logs

Step 1: Identify

What

Why

Step 2: Discuss

All teams

How

Levels

Step 3: Update API

Libraries

Levels

Step 4: Update UI

Optional

Minimal surface

Step 5: Monitor

Observations

Adjustments

Process

Try It!

https://logbetter.nimbinatus.com/scenario

What did you decide?

Thank you!

@nimbinatus || @logdna

#logbetter

https://nimbinatus.com