

YAML Considered Harmful

Philipp Krenn

@xeraa

Edgar Dijkstra: Go To Statement Considered Harmful

Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing

CR Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such bad effects, and I have concluded that the **go to** statement should be abolished from all "high-level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

My first remark is that, although the programmer's activity ends when he has constructed a correct program, the process

dynamic progress is only characterized when we also give a call of the procedure we refer. With the inclusion of these calls we can characterize the progress of the process via a textual indices, the length of this sequence being equal to the dynamic depth of procedure calling.

Let us now consider repetition clauses (like, **while** or **repeat A until B**). Logically speaking, such clauses are superfluous, because we can express repetition with recursive procedures. For reasons of realism I don't exclude them: on the one hand, repetition clauses came into being quite comfortably with present day finite equipment, and the other hand, the reasoning pattern known as "loop invariants" makes us well equipped to retain our intellectual grasp of the processes generated by repetition clauses. With the inclusion of the repetition clauses textual indices are no longer sufficient to describe the dynamic progress of the process. With each repetition clause, however, we can associate a so-called "dynamic index," inexorably counting the ordinal number of the corresponding current repetition. As repetition clauses (and procedure calls) may be applied nestedly, we find

<https://homepages.cwi.nl/~storm/teaching/reader/>

Dijkstra68.pdf

Why YAML?

**Human readable
Comments**

Whitespace Sensitive

It depends

Issues

<http://www.yamllint.com>

ports:

- 80:80
- 20:20

Issues

<https://docs.docker.com/compose/compose-file/#short-syntax-1>

ports:

- "80:80"
- 73200

Issues

countries:

- DE
- FR
- GB
- IE
- NO
- PT

Issues

surname: Null

windows_drive: c:

version: 1.0

Abuse

Behavior instead of data

Gitlab

Deploying itself in 1,100+ lines

**[https://gitlab.com/gitlab-org/gitlab-ce/blob/
master/.gitlab-ci.yml](https://gitlab.com/gitlab-org/gitlab-ce/blob/master/.gitlab-ci.yml)**

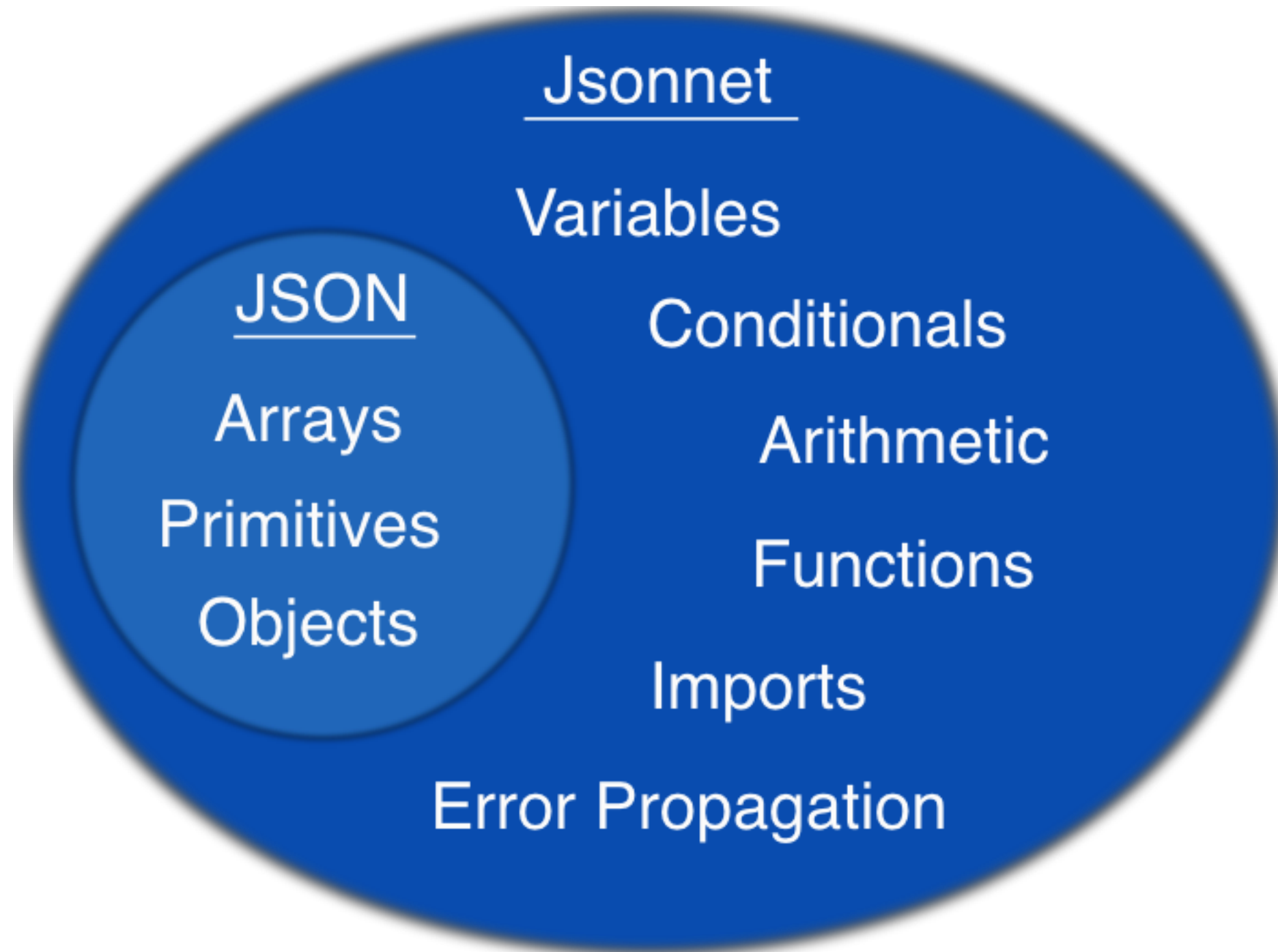
```
package-and-qa:
  image: ruby:2.5-alpine
  stage: test
  before_script: []
  dependencies: []
  cache: {}
  variables:
    GIT_DEPTH: "1"
    API_TOKEN: "${GITLAB_BOT_MULTI_PROJECT_PIPELINE_POLLING_TOKEN}"
  retry: 0
  script:
    - apk add --update openssl curl jq
    - gem install gitlab --no-document
    - source ./scripts/review_apps/review-apps.sh
    - wait_for_job_to_be_done "gitlab:assets:compile"
    - ./scripts/trigger-build omnibus
  when: manual
  only:
    - /.+/@gitlab-org/gitlab-ce
    - /.+/@gitlab-org/gitlab-ee
```

Helm Templates

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
  labels:
    {{- include "mychart.app" . | nindent 4 }}
data:
  myvalue: "Hello World"
  {{- range $key, $val := .Values.favorite }}
  {{ $key }}: {{ $val | quote }}
  {{- end }}
  {{- include "mychart.app" . | nindent 2 }}
```

**If you want a DSL,
use a DSL**

Or Maybe Jsonnet



Alternative Data Format: XML

```
<?xml version="1.0" encoding="UTF-8" ?>
  <!--Last modified 1 April 2001 by John Doe-->
  <title>XML Example</title>
  <owner>
    <name>John Doe</name>
  </owner>
  <database>
    <server>192.168.1.1</server>
    <ports>
      <port>8001</port>
      <port>8002</port>
      <port>8003</port>
    </ports>
  </database>
  <servers>
    <server name="alpha">
      <ip>10.0.0.1</ip>
      <dc>eqdc10</dc>
    </server>
    <server name="beta">
      <ip>10.0.0.2</ip>
      <dc>eqdc10</dc>
    </server>
  </servers>
```

Alternative Data Format: JSON

```
{
  "_comment": "Last modified 1 April 2001 by John Doe",
  "title": "JSON Example",

  "owner": {
    "name": "John Doe"
  },

  "database": {
    "server": "192.168.1.1",
    "ports": [ 8001, 8002, 8003 ]
  },

  "servers": {
    "alpha": {
      "ip": "10.0.0.1",
      "dc": "eqdc10"
    },
    "beta": {
      "ip": "10.0.0.2",
      "dc": "eqdc10"
    }
  }
}
```


Alternative Data Format: INI

```
; Last modified 1 April 2001 by John Doe
```

```
[owner]
```

```
name=John Doe
```

```
organization=Acme Widgets Inc.
```

```
[database]
```

```
; use IP address in case network name resolution is not working
```

```
server=192.168.1.1
```

```
port=8001
```

```
[servers]
```

```
serverAlphaIp=10.0.0.1
```

```
serverAlphaDc=eqdc10
```

```
serverBetaIp=10.0.0.2
```

```
serverBetaDc=eqdc10
```

Alternative Data Format: TOML

```
# Last modified 1 April 2001 by John Doe
title = "TOML Example"

[owner]
name = "John Doe"
dob = 1970-01-01T01:00:00-02:00 # First class dates

[database]
server = "192.168.1.1"
ports = [ 8001, 8002, 8003 ]

[servers]
  [servers.alpha]
  ip = "10.0.0.1"
  dc = "eqdc10"

  [servers.beta]
  ip = "10.0.0.2"
  dc = "eqdc10"
```

Alternative Data Format: JSONx

**JSONx is an IBM® standard format to represent JSON
as XML**

YAML Considered Harmful

Philipp Krenn

@xeraa