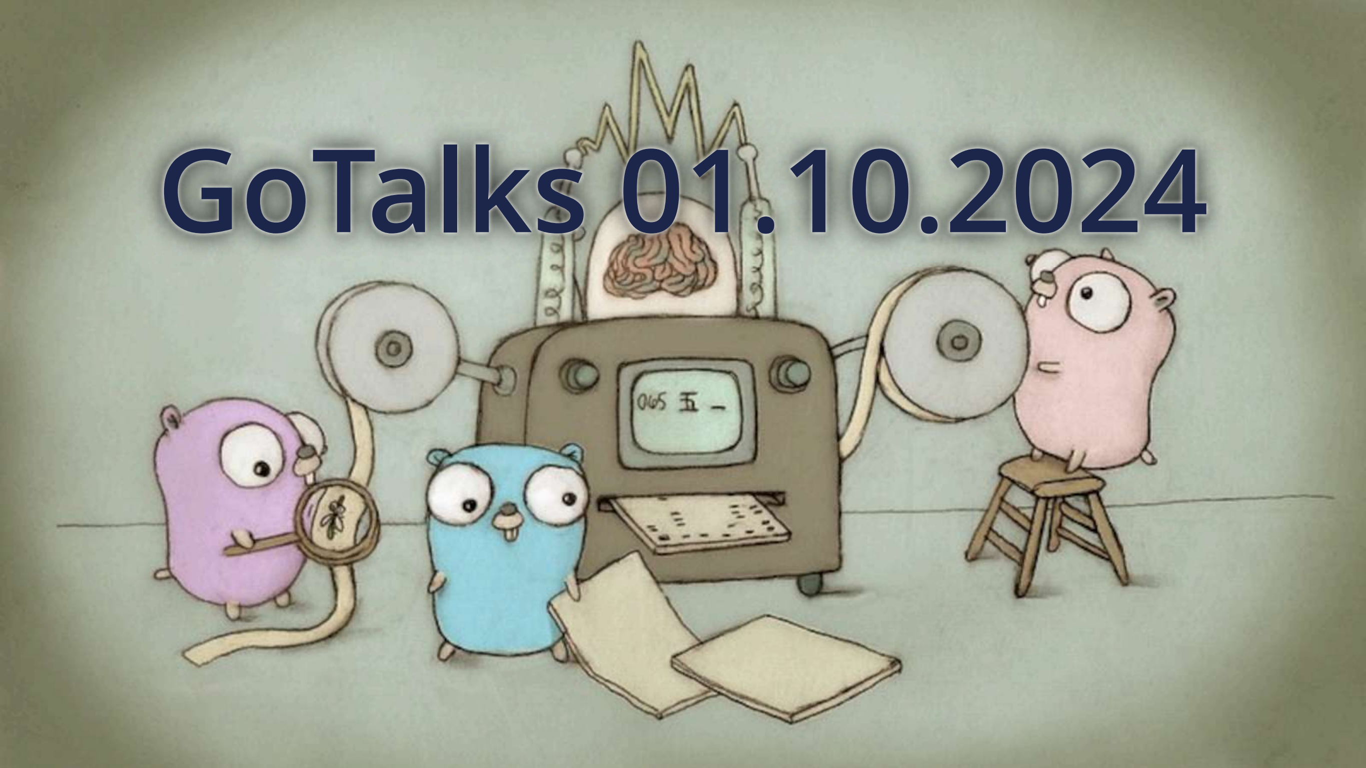
GoTalks 01.10.2024



Go 1.23

- https://go.dev/doc/go1.23

```
$ go install golang.org/dl/go1.23.0@latest
$ go1.23.0 download
```

Meetup GolangZG

Changes - range-over-func

The `range` clause in a `for-range` loop now accepts iterator functions of the following types

```
func(func() bool)
func(func(K) bool)
func(func(K, V) bool)
```

Changes - Iterators - slices

https://tip.golang.org/doc/go1.23#iterators

- All returns an iterator over slice indexes and values.
- Values returns an iterator over slice elements.
- Backward returns an iterator that loops over a slice backward.
- Collect collects values from an iterator into a new slice.
- AppendSeq appends values from an iterator to an existing slice.
- Sorted collects values from an iterator into a new slice, and then sorts the slice.
- SortedFunc is like Sorted but with a comparison function.
- SortedStableFunc is like SortFunc but uses a stable sort algorithm.
- Chunk returns an iterator over consecutive sub-slices of up to n elements of a slice.

Changes - Iterators - maps

https://tip.golang.org/doc/go1.23#iterators

- All returns an iterator over key-value pairs from a map.
- Keys returns an iterator over keys in a map.
- Values returns an iterator over values in a map.
- Insert adds the key-value pairs from an iterator to an existing map.
- Collect collects key-value pairs from an iterator into a new map and returns it.

Changes - range-over-func - slices

```go
slices.All(s)
slices.Values(s)
slices.Backward(s)

slices.Collect(slices.Values(s)) // new slice

slices.AppendSeq(s, slices.Values(source))
slices.Sorted(slices.Values(s))
slices.SortedFunc(slices.Values(s), compare)
```

Meetup GolangZG

Changes - range-over-func - maps

```go
maps.All(m)
maps.Keys(m) // Iterator over keys
maps.Values(m)
maps.Insert(m, maps.All(m1))
maps.Collect(maps.All(m1)) // new map
```

Meetup GolangZG

Changes - range-over-func

go.mod main.go

```go
import (
	"fmt"
	"slices"
)

func main() {
  s := []string{"a", "b", "c"}
  for i, v := range slices.All(s) {
    fmt.Printf("%d:%v ", i, v)
  }
}
```

Changes - range-over-func

go.mod | backward.go | backward-str.go | main.go

```go
func Backward[E any](s []E) func(func(int, E) bool) {
    return func(yield func(int, E) bool) {
        for i := len(s)-1; i >= 0; i-- {
            if !yield(i, s[i]) {
                return
            }
        }
    }
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | backward.go | backward-str.go | main.go

```go
func BackwardStr(s []string) func(func(int, string) bool) {
    return func(yield func(int, string) bool) {
        for i := len(s)-1; i >= 0; i-- {
            if !yield(i, s[i]) {
                return
            }
        }
    }
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | backward.go | backward-str.go | main.go

```go
func main() {
  s := []string{"a", "b", "c"}
  for i, v := range Backward(s) {
    fmt.Printf("%d:%v ", i, v)
  }
  fmt.Println("")
  for i, v := range BackwardStr(s) {
    fmt.Printf("%d:%v ", i, v)
  }
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | sorting | sort.go | main.go

```go
type Grocery struct {
    Name string
    Type string
}

type Warehouse struct {
    Groceries map[string]Grocery
    // ...
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | sorting | sort.go | main.go

```go
var shopping = Warehouse{
    Groceries: map[string]Grocery{
        "SN_1": {
            Name: "banana",
            Type: "fruit",
        },
        "SN_2": {
            Name: "apple",
            Type: "fruit",
        },
        "SN_3": {
            Name: "carrot",
            Type: "vegetable",
        },
    },
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | type.go | data.go | **filtering** | filter.go | sorting | sort.go | main.go

```go
for _, v := range v.Groceries {
    if v.Type != foodType {
        continue
    }
    if !continueLoop(v) {
        return
    }
}
```

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | sorting | sort.go | main.go

```go
func (v Warehouse) Filter(foodType string) func(func(Grocery) bool) {
	return func(continueLoop func(Grocery) bool) {
		for _, v := range v.Groceries {
			if v.Type != foodType {
				continue
			}
			if !continueLoop(v) {
				return
			}
		}
	}
}
```

Meetup GolangZG

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | **sorting** | sort.go | main.go

```
names := []string{}
for _, v := range v.Groceries {
    names = append(names, v.Name)
}
sort.Strings(names)
for _, k := range names {
    for serial, v := range v.Groceries { // example only
        if v.Name != k {
            continue
        }
        if !continueLoop(serial, v) {
            return
        }
    }
}
```

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | sorting | sort.go | main.go

```go
func (v Warehouse) SortedByName() func(func(string, Grocery) bool) {
    return func(continueLoop func(string, Grocery) bool) {
        names := []string{}
        for _, v := range v.Groceries {
            names = append(names, v.Name)
        }
        sort.Strings(names)
        for _, k := range names {
            for serial, v := range v.Groceries { // example only
                if v.Name != k {
                    continue
                }
                if !continueLoop(serial, v) {
                    return
                }
            }
        }
    }
}
```

Changes - range-over-func

go.mod | type.go | data.go | filtering | filter.go | sorting | sort.go | main.go

```go
    for k, v := range shopping.Groceries {
        fmt.Println(k, v)
    }
    fmt.Println("====================")

    for i, g := range shopping.SortedByName() {
        fmt.Println(i, g)
    }
    fmt.Println("====================")

    for g := range shopping.Filter("fruit") {
        fmt.Println(g)
    }
```

Telemetry

```
module hello

go 1.23
```

```
go telemetry on
```

https://telemetry.go.dev/

Other Changes

Go

```
go mod tidy -diff
```

- causes the command not to modify the files but instead print the necessary changes as a unified diff.
- It exits with a non-zero code if updates are needed.

CGo

- `cmd/cgo` supports the new `-ldflags` flag for passing flags to the C linker

Other Changes

Compiler

- The build time overhead to building with Profile Guided Optimization has been reduced significantly.
  Previously, large builds could see 100%+ build time increase from enabling PGO. In Go 1.23, overhead should be in the `single digit` percentages.
- The compiler in Go 1.23 can now overlap the stack frame slots of local variables accessed in disjoint regions of a function, which `reduces stack usage` for Go applications.

Other Changes

Compiler

- For 386 and amd64, the compiler will use information from PGO to align certain hot blocks in loops.
  This `improves performance` an additional 1-1.5% at a cost of an additional 0.1% text and binary size.
  This is currently only implemented on `386` and `amd64` because it has not shown an improvement on other platforms.

Timer

- Significant changes to the implementation of `time.Timer` and `time.Ticker`.
  - Timers and Tickers that are no longer referred to by the program become eligible for garbage collection immediately
  - the timer channel associated with a Timer or Ticker is now unbuffered, with capacity 0
  - `time.After()`

```
// As of Go 1.23, the garbage collector can recover
// unreferenced, unstopped timers. There is no reason
// to prefer NewTimer when After will do.
```

Community

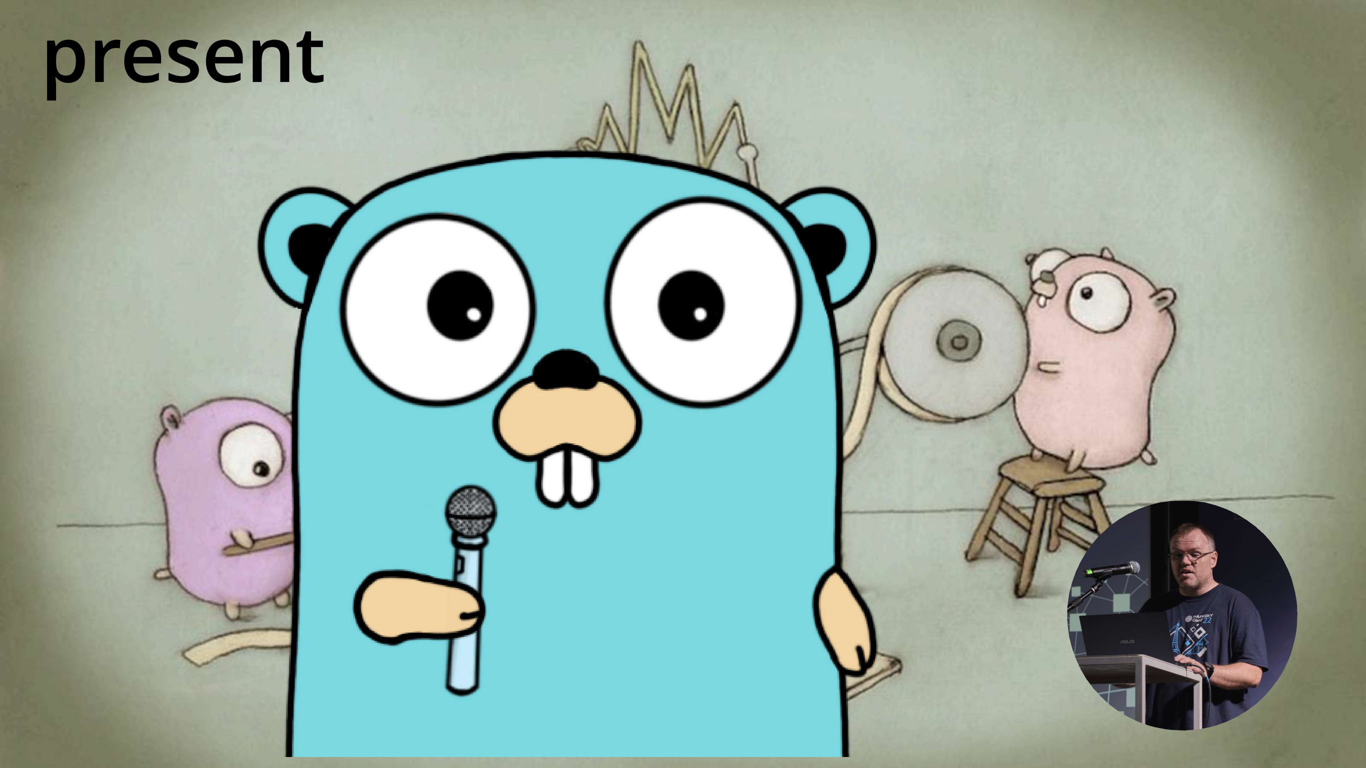
- Go version manager, written in Go
  - https://github.com/kevincobain2000/gobrew

```
go install github.com/kevincobain2000/gobrew/cmd/gobrew@lat
```

```
gobrew use latest
```

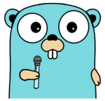
- Go features by version
  - https://antonz.org/which-go/
- https://antonz.org/go-1-23/

present



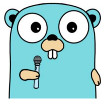
present - `overview`

- tool for viewing presentations written in markdown like format
  - markdown - easy to read
- slides are written in text friendly format and follow all standard markdown rules (with some additions)
- easy to run
- primarily for presentations + code
  - inlined of linked code
- tutorials
  - *run/experiment on your machine*



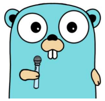
present - `goals`

- view presentation in browser
- text format
    - git friendly
- run the code (**any** language) directly from presentation
    - run complex examples
- standard header/footer options
- fully customizable (settings and css/js customizations)
- live share (as a help in large rooms) / remote watching*
- easy share presentations
    - share on github, run with link
- print friendly (*chromium* browsers)



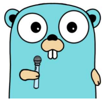
present - `live`

- *****************
- follow online (hopefully :))



present - `alternatives`

- existing tools
  - `golang.org/x/tools/present`
  - https://jupyter.org/
  - google slides
  - ...



present - `installation`

Installation

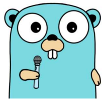
Use the following command to download and install this tool:

```
go install github.com/oktalz/present@latest
```

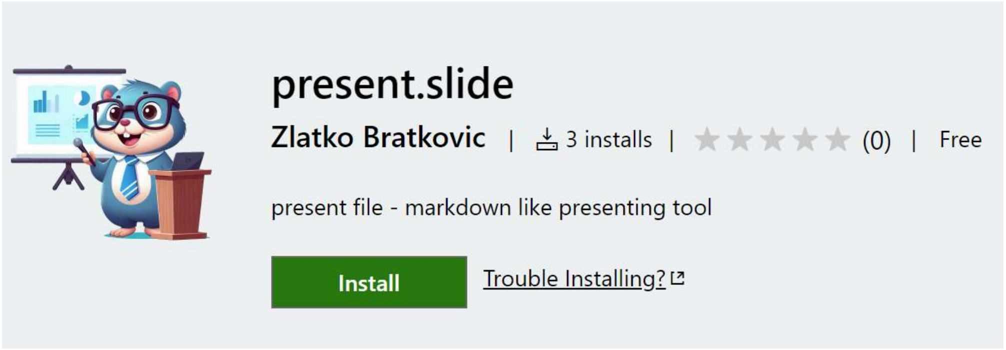
```
go install github.com/oktalz/present@v1.0.0
```

Binaries

prebuilt binaries can be found on [releases]() page



present - `VS Code`



present.slide

Zlatko Bratkovic | 3 installs | ☆☆☆☆☆ (0) | Free

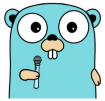
present file - markdown like presenting tool

[Visual Studio Code Marketplace]

[Open VSX Marketplace]

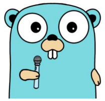
present - `running`

```
present --help
```



present - `running`

- enter folder, type `present`
  - program should read all files and start web server on port 8080 (default)
    - port can be customized (see `present.env` file)
- run `present -d /path/to/files`
- run `present -g github.com/oktalz/present -d examples/showcase`
  - for `gitlab.com` and `github.com` project url is detected, for others use full path `-g https://github.com/oktalz/present.git`

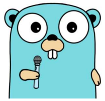


present - `overview - markdown`

```
### title
- text that is **Bold**, *Italics*, `highlighted`, ~strikethrough~
  - random point
- :speech_balloon: :thought_balloon: :warning: :construction:
.cut
- 😂 ⭐ ❗ 🔥 👍
```

title

- text that is **Bold**, *Italics*, `highlighted`, ~~strikethrough~~
  - random point
- 💬 💭 ⚠️ 🚧
- 😂 ⭐ ❗ 🔥 👍



present - overview - transitions, tables

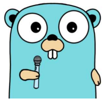
simple transition

- topic one
- topic two

projects written in Go

Docker	CockroachDB
Kubernetes	Etherium blockchain

- can be a bit complex to setup (at first)



images, style, header, footer

- any html style formatting is possible
  - custom styling of span, div, blocks (div)
  - existing styling can be overridden
- use `svh` and `svw` for font size (and in general)

even rotated, positional

HTML styles background

present - code

- editable
- inline or imported from file
- run in `tmp` or in specific folder
- partially shown (with edit option)
- response seen in presentation in terminal

present - code

```
.cast.edit.save(main.go).run(go run .).before(go mod init x)
```

```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println("hello world")
}
```

present - code

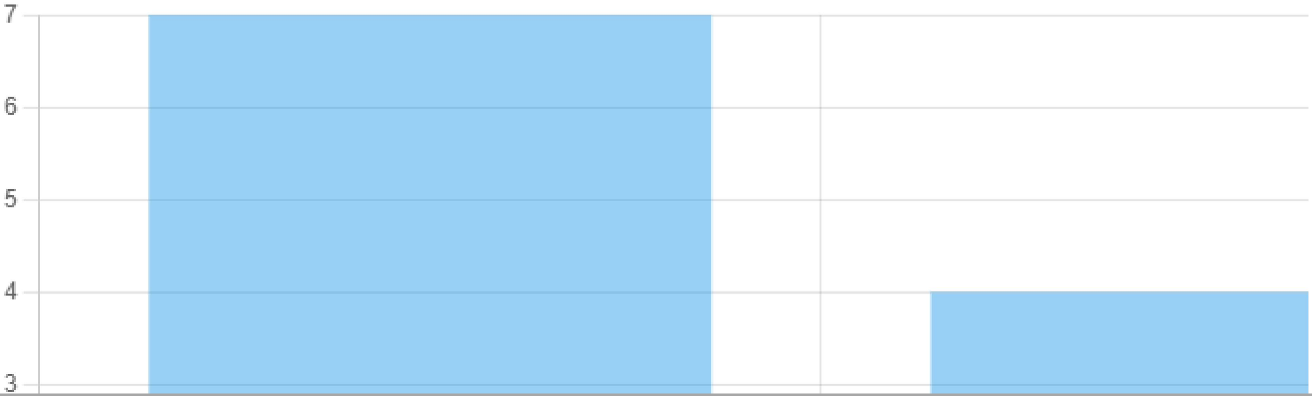
```
.cast.stream.edit.save(main.go).run(go run .).show(9:11)
```

```
  fmt.Println("hello")
  time.Sleep(3 * time.Second)
  fmt.Println("world")
```

Links

- pick 😺 or 🐶

⚠️ - this is experimental feature



Links

- pick 😺 or 🐶 (click with mouse)

Links 😺

Links 🐶

Links 😟

no pick ?

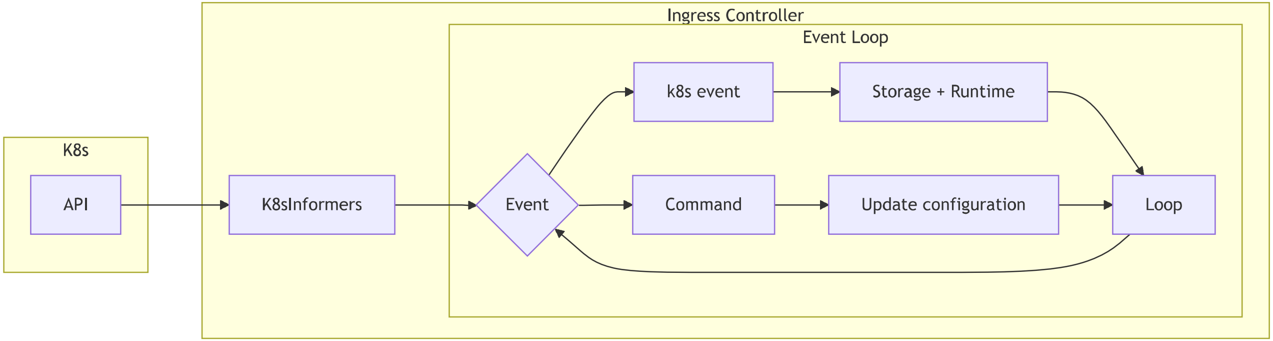
Links 🐱 & 🐶

happy path

Links

next slide from all paths

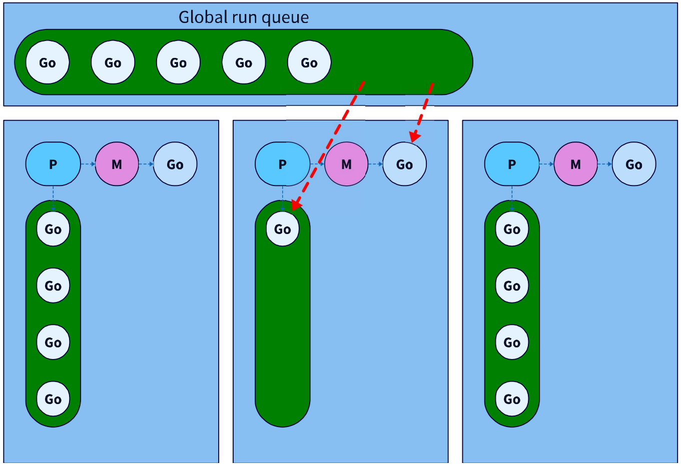
Graphs - mermaid



```
.block.path(graphs).source(k8sIC.mermaid).lang(mermaid)
```

- ℹ `.lang(mermaid)` is not needed if extension matches block type
- ℹ can be embedded directly in file

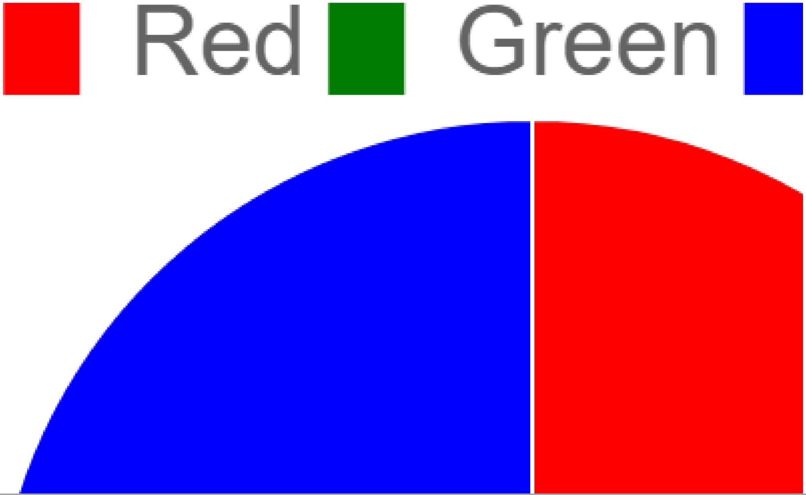
Graphs - d2



```
.css{width: 100svw; height: 75svh; overflow: hidden; font-size: 4svh!important;}
.block.path(graphs).source(scheduler.d2)
.css.end
```

Graphs - chart.js (`experimental`) - pie

- integrated js library, need to use raw html
- **only** pixel options for sizes :(



Roadmap

- experimental
  - pools
  - graph.js
- tests :)
- documentation
- animations (native)
- output compression
- mobile friendly 22
- performance
- security

Examples - `live`