Raúl Chouza

Computer Engineer and Elixir Programmer

18th, September, 2024

Raúl Chouza
ESL Jaguars
raul.chouza@erlang-solutions.com

Erlang Solutions
AMERICAS OFFICE


COLORADO
KANSAS
MISSOURI
San Francisco
CALIFORNIA
Las Vegas
OKLAHOMA
ARKANSAS
Los Angeles
ARIZONA
NEW MEXICO
MISSISSIPPI
Dallas
TEXAS
537 hr
1,478 miles
LOUISIANA
Houston
Gulf of California
Monterrey
Gulf of Mexico
Mexico

Erlang Solutions

AMERICAS OFFICE

Raúl Chouza

Sr. Elixir Developer

18th, September, 2024

Raúl Chouza
ESL Jaguars
raul.chouza@erlang-solutions.com

www.erlang-solutions.com
©2021 Erlang Solutions

gleam

intro to the language
and platform

Gleam is fun! 

```
import gleam/list
import gleam/io

const persons = ["Luis", "José", "Roberto", "Miguel"]

pub fn main() -> Nil {
 let [person, ..] = list.shuffle(persons)
 io.println("Hey! " <> person)
}
```

Gleam is fun!

```gleam
import gleam/list
import gleam/io

const persons = ["Luis", "José", "Roberto", "Miguel"]

pub fn main() -> Nil {
  let [person, ..] = list.shuffle(persons)
  io.println("Hey! " <> pers
}
```

```
Inexhaustive pattern

This assignment uses a pattern that does not match all possible
values. If one of the other values is used then the assignment
will crash.

The missing patterns are:

    []

In a future version of Gleam this will become a compile error.
```

Gleam is fun! 

```gleam
import gleam/list
import gleam/io

const persons = ["Luis", "José", "Roberto", "Miguel"]

pub fn main() -> Nil {
 let assert [person, ..] = list.shuffle(persons)
 io.println("Hey! " <> person)
}
```

Gleam is fun! 

```
pub type Coor {
 Coor(x: Int, y: Int)
}

type Piece {
  Pawn
  Rook
  Bishop
  Knight
  Queen
  King
}
```

Gleam is fun!

```
pub fn add(m: Coor, n: Coor) -> Coor {
 let Coor(x: xa, y: ya) = m
 let Coor(x: xa, y: ya) = n
 Coor(x: xa + xb, y: ya + yb)
}
```

Gleam is fun!

```gleam
pub fn is_absolute(m: Coor) -> Boolean {
  case m {
    Coor(x, y) if x >= 0 and y >= 0 -> True
    Coor(_, _) -> False
  }
}
```

Gleam is fun! 

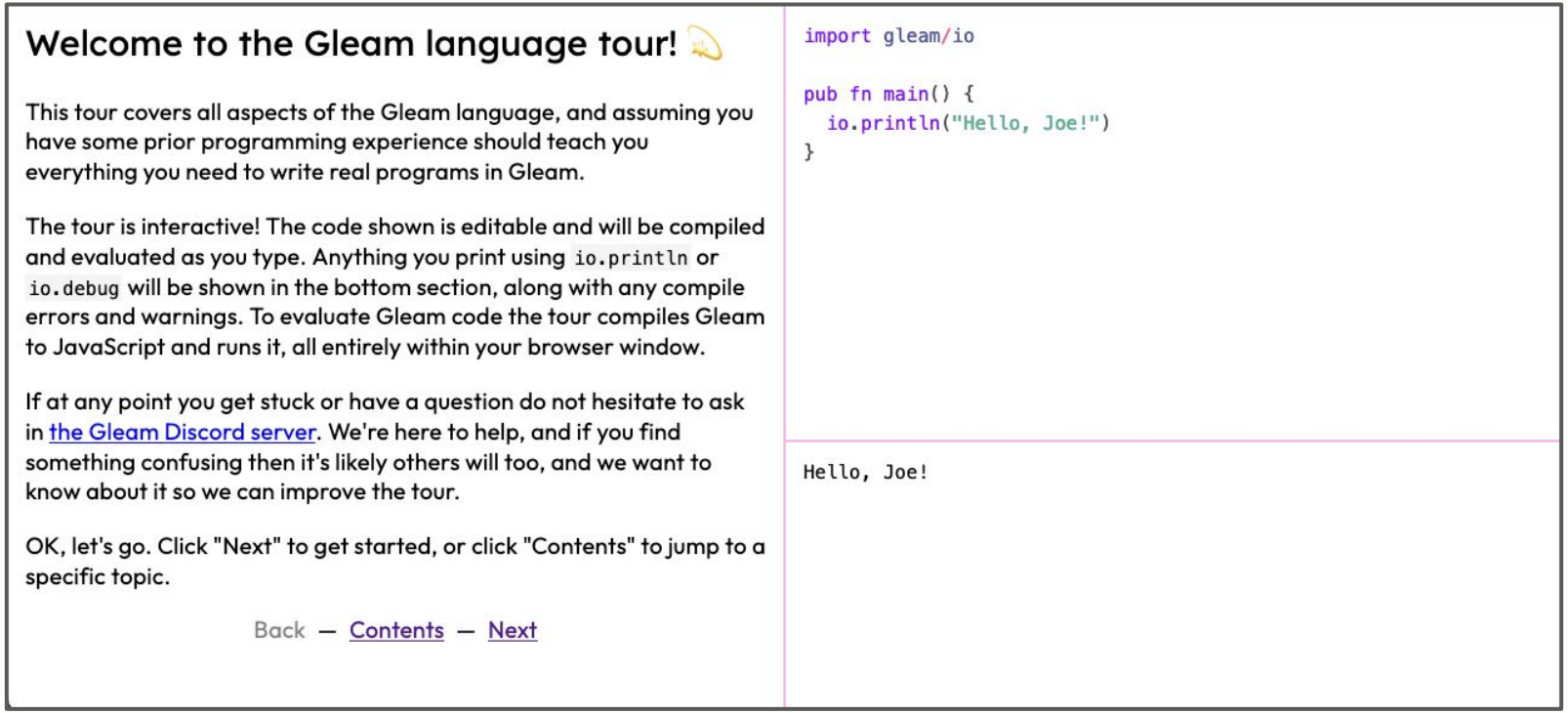
```
fn move(piece: Piece) -> Coor {
  case piece {
    Pawn -> coor_pawn()
    Rook -> coor_rook()
    Bishop -> coor_bishop()
    Knight -> coor_knight()
    Queen -> coor_knight()
    King -> coor_knight()
  }
}
```

Gleam is simple! 

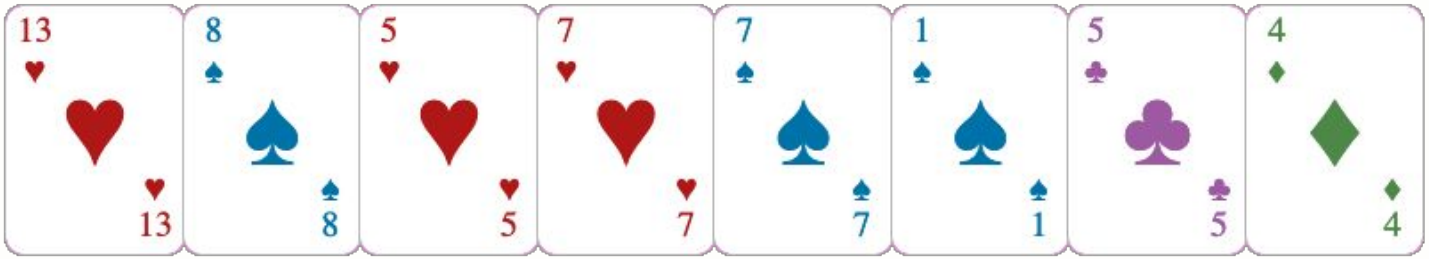
```
import gleam/list
import gleam/io

const persons = ["Luis", "José", "Roberto", "Miguel"]

pub fn main() -> Nil {
 let [person, ..] = list.shuffle(persons)
 io.println("Hey! " <> person)
}
```

- https://mckayla.blog/posts/all-you-need-is-data-and-functions.html
- https://guide.elm-lang.org/webapps/modules.html#growing-modules
- The Erlang Rationale by Robert Virding - Modules, code and code loading

https://tour.gleam.run/



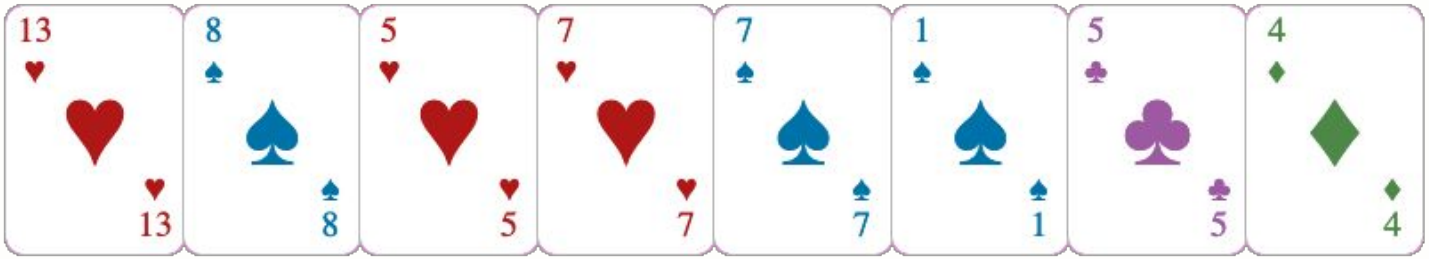
Demo time!

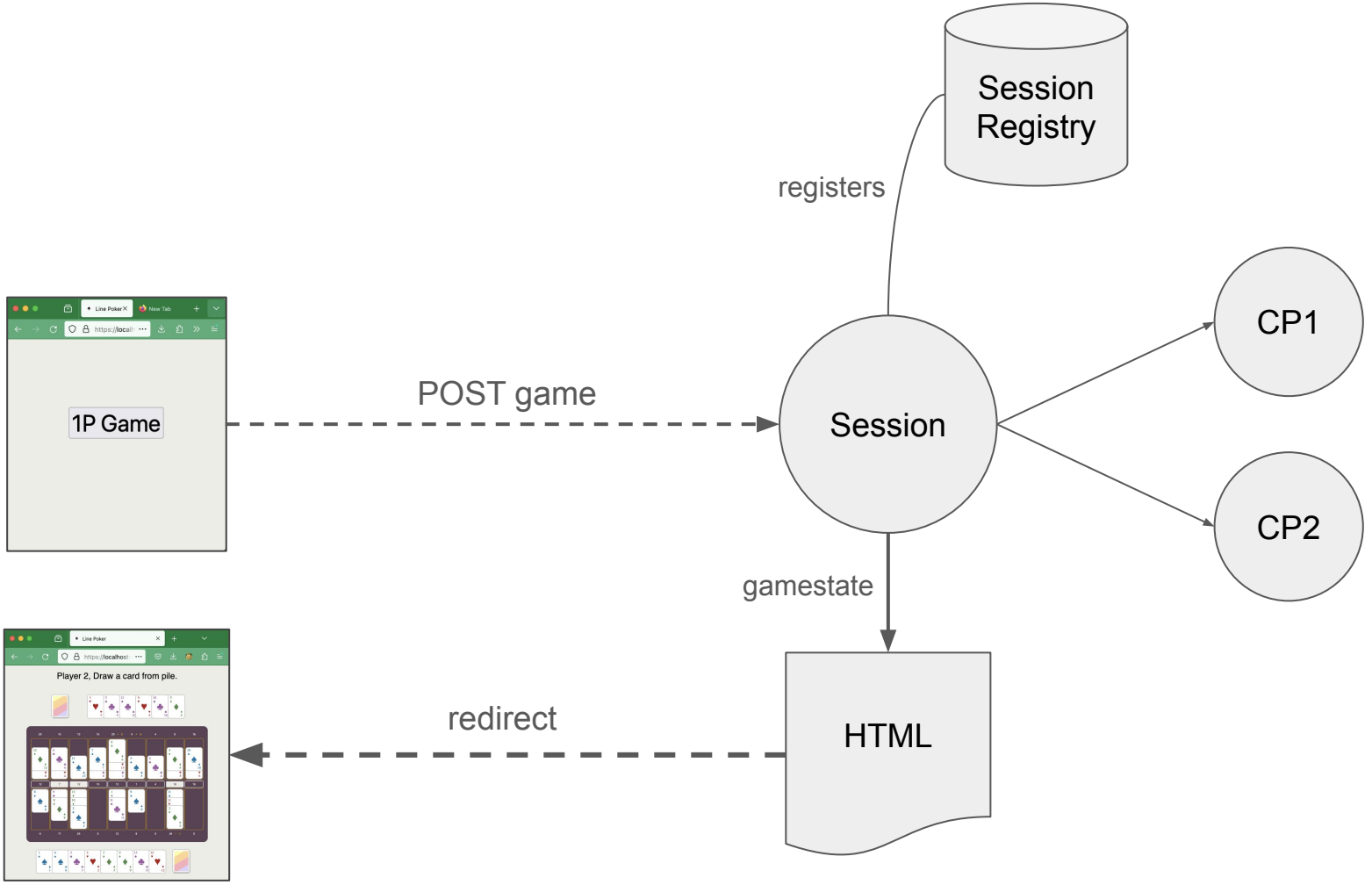


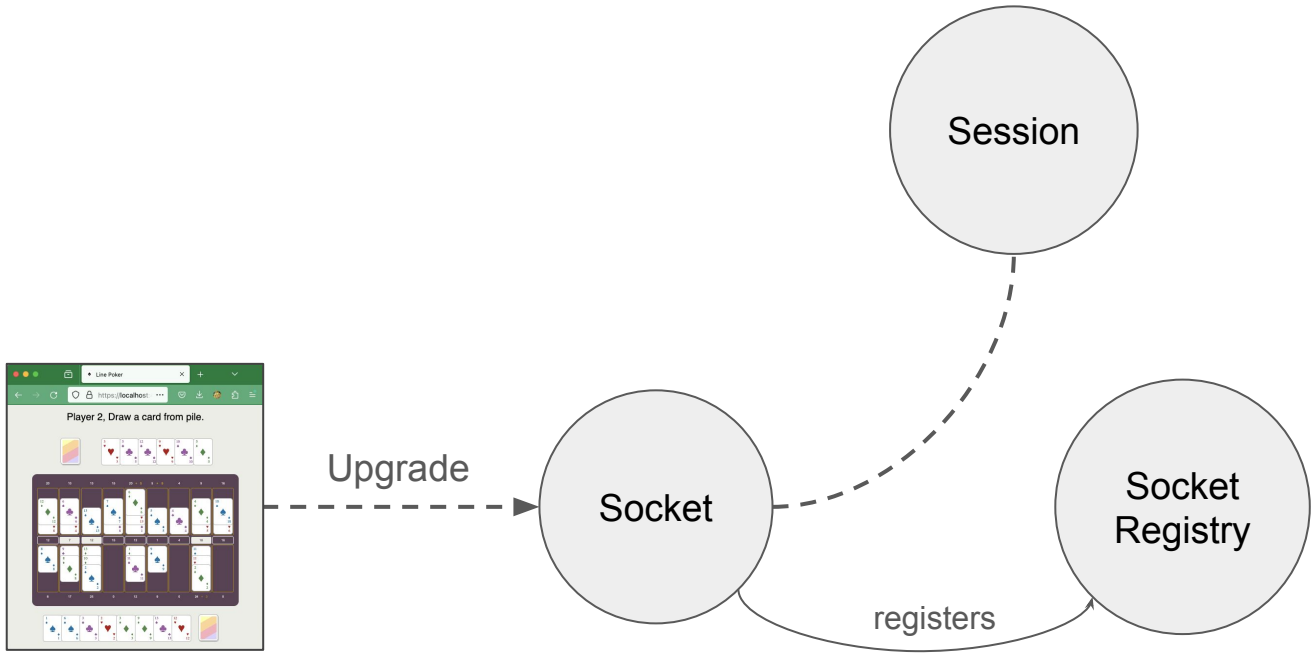
gleam  erlang 

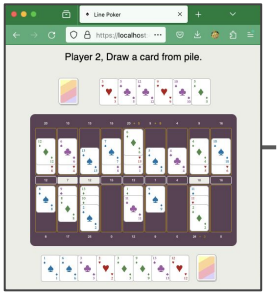
Talk - Saša Jurić the soul of erlang and elixir

Demo time!

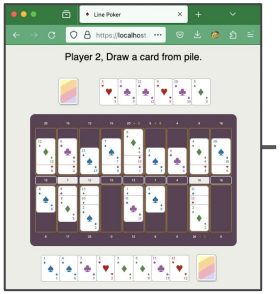



Session Registry
registers
1P Game
POST game
Session
CP1
CP2
gamestate
HTML
redirect
Player 2, Draw a card from pile.


Session
Player 2, Draw a card from pile.
Upgrade
Socket
Socket Registry
registers

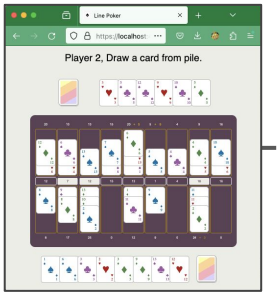


Socket



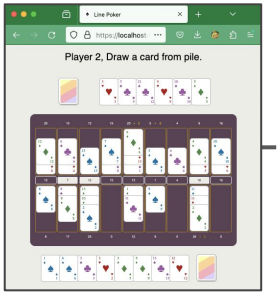
Socket

Session

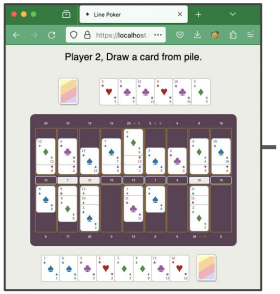


Socket

Socket Registry



Socket



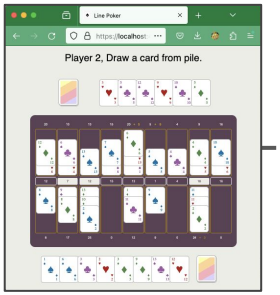
Play Card

Socket

Session

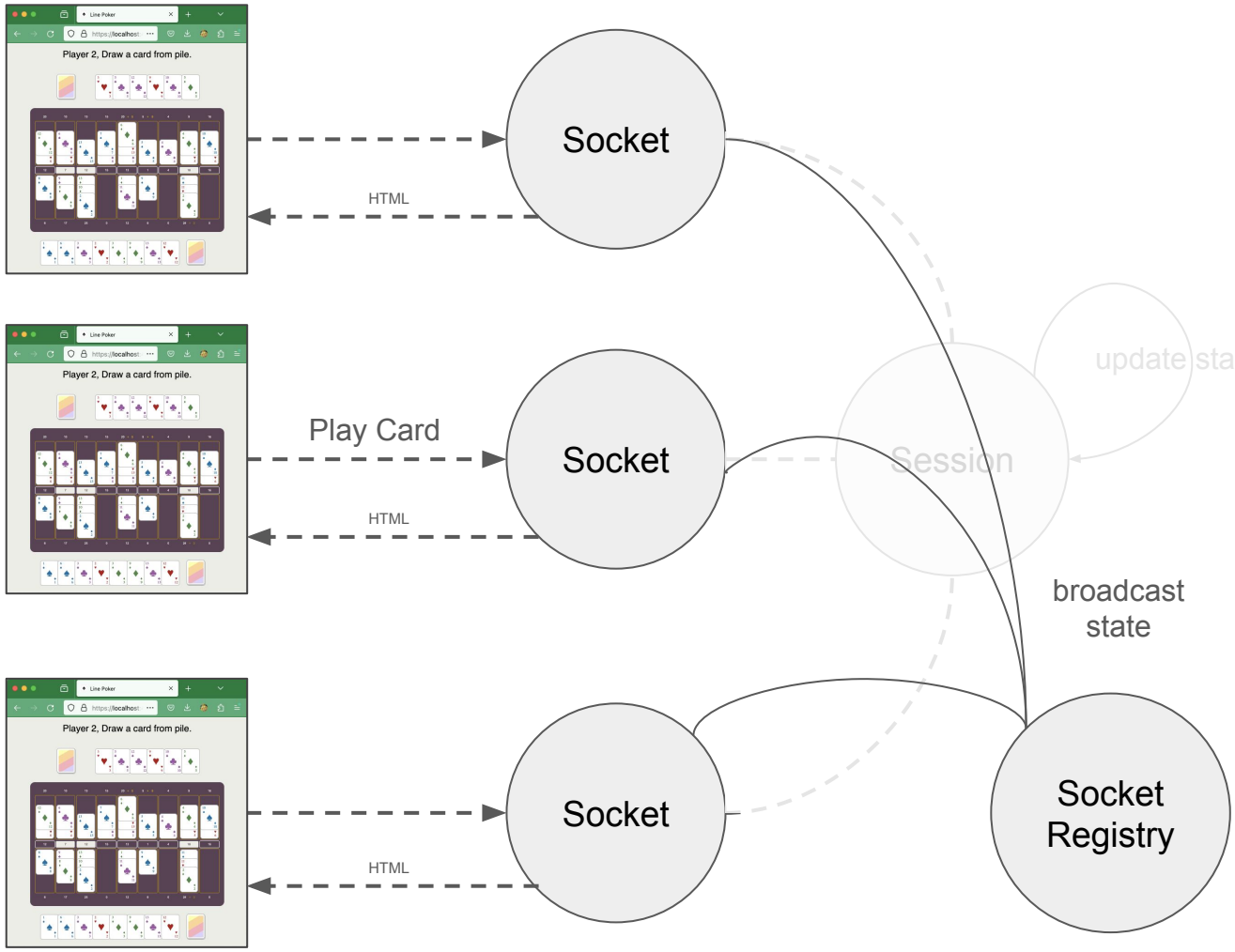
update state

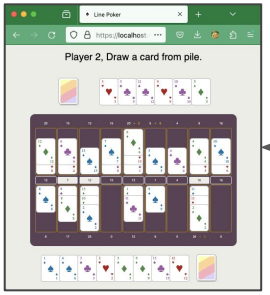
state



Socket

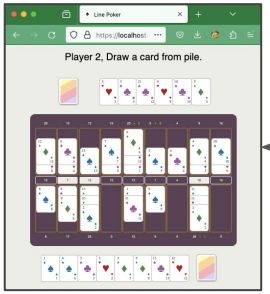
Socket Registry


Player 2, Draw a card from pile.
Socket
HTML
Player 2, Draw a card from pile.
Play Card
Socket
HTML
update state
Session
broadcast state
Player 2, Draw a card from pile.
Socket
HTML
Socket Registry



Socket

Session Registry

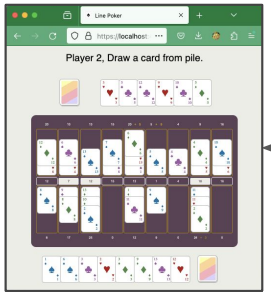


Socket

Session

CP1

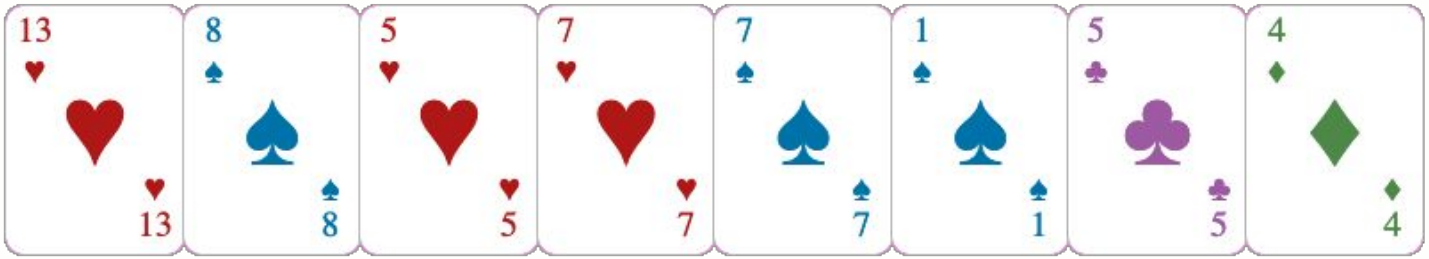
CP2



Socket

Socket Registry

Demo time!



gleam ❤️ JS 

gleam  JS 

"If you have programs and data, then circumstances will happen when they move and you don't know."

"When the code and data are separated disasters will happen."

Joe Armstrong

"Intertwingling the Tiddlywiki"
"The Mess We're In"

gleam ❤️ JS

https://100r.co/site/computing_and_sustainability.html

gleam ❤️ JS 

https://lustre.build

gleam

intro to the language
and platform

the platforms

ERLANG

- erlang VM
- atom VM
- Nerves (linux)
- Grisp µC

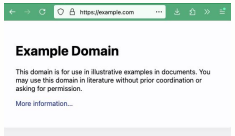
NERVES

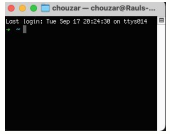
AVM

GRiSP

JS

- Web Browser
- Node
- Deno
- Bun





Erlang Solutions
AMERICAS OFFICE

Credits

Thanks for the amazing slides and graphics Carlo Gilmar.

https://substack.com/@visualpartner

©2021 Erlang Solutions

November
NY City

CODE BEAM LITE NEW YORK CITY

https://codebeamnyc.com

March
San Francisco

CODE BEAM AMERICA

https://codebeamamerica.com

Erlang Solutions

Fin

Erlang Solutions