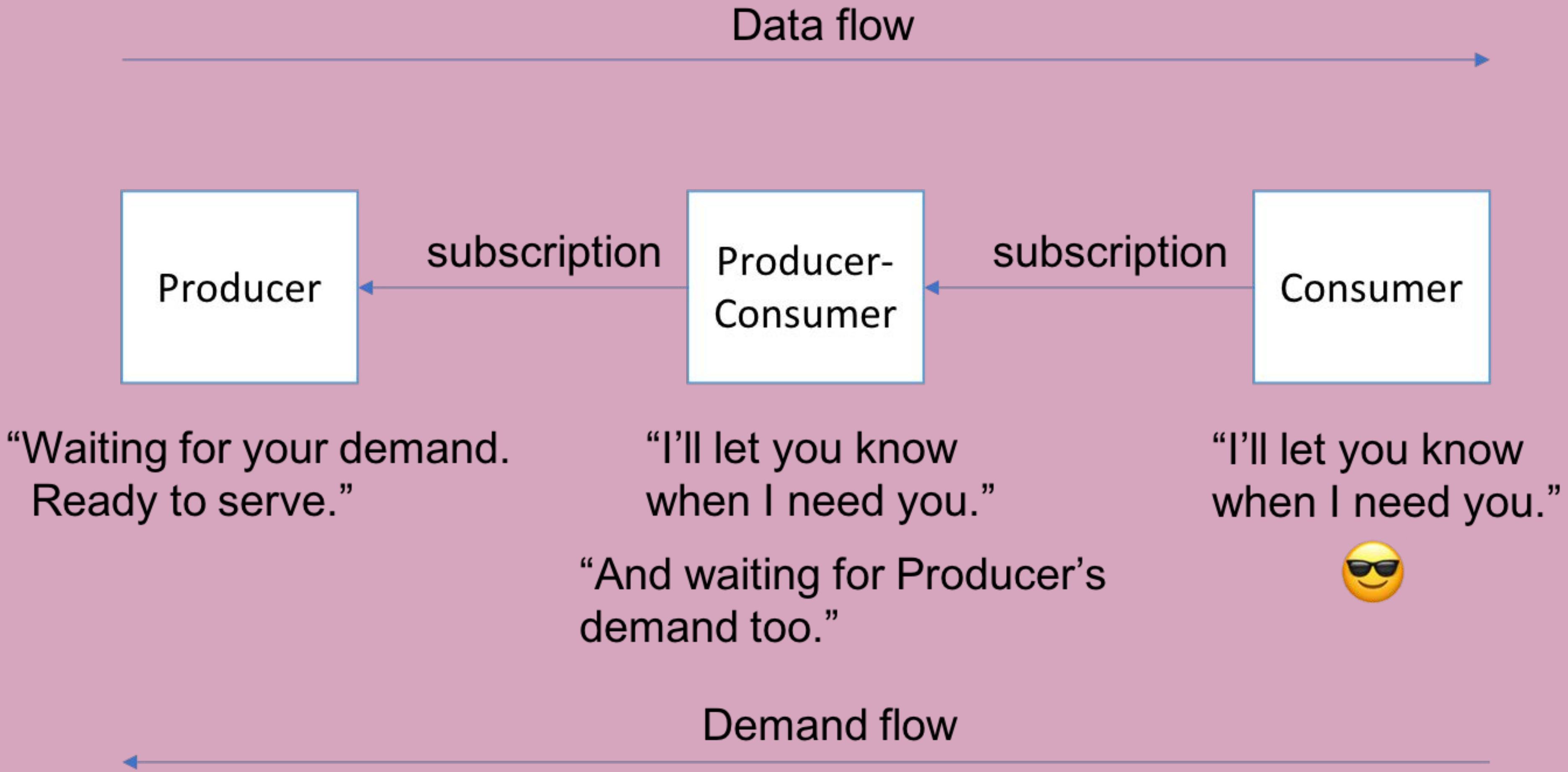
Elixir GenStage

Lena Obertynska

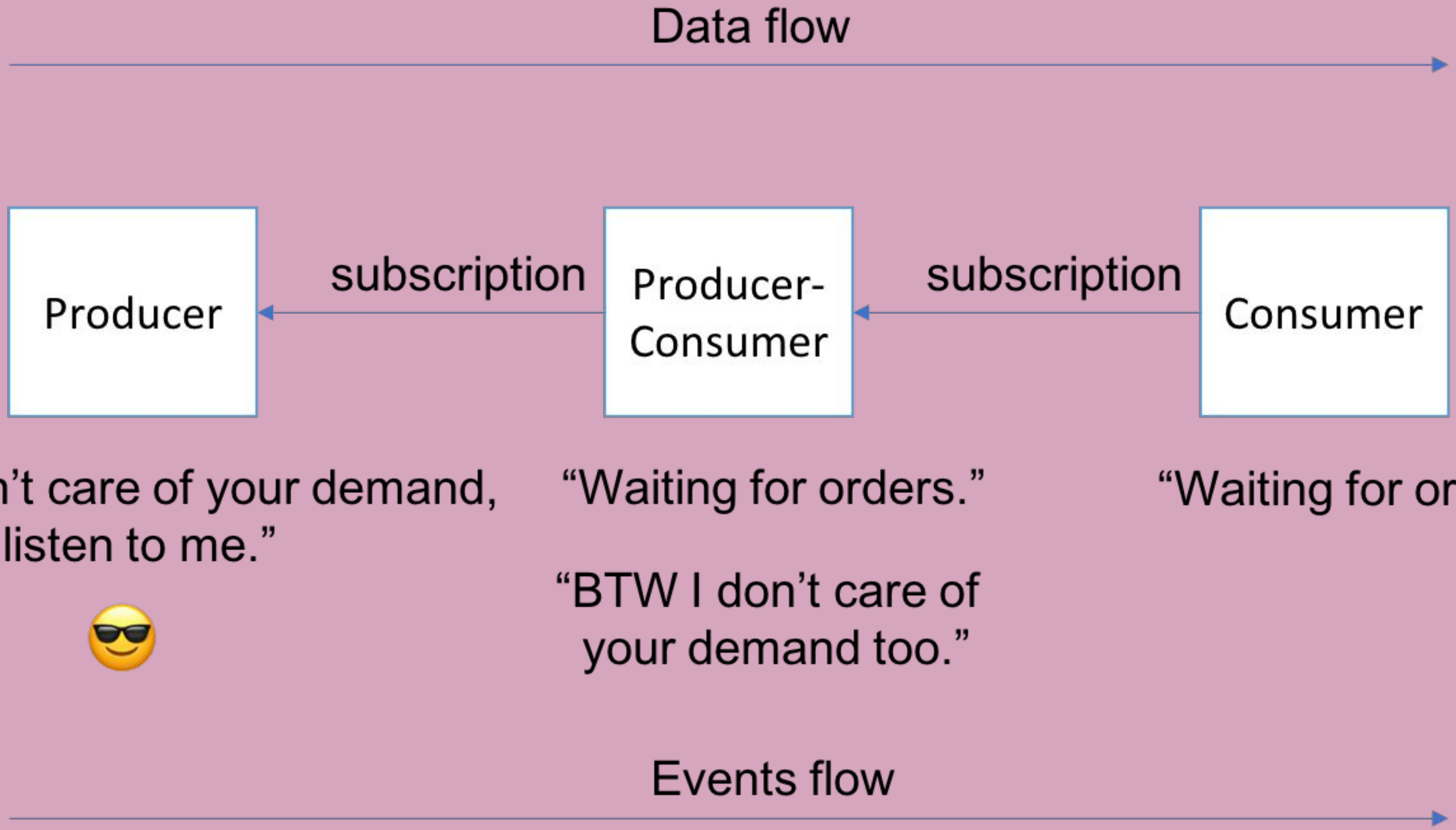


"Waiting for your demand. Ready to serve."

"I'll let you know when I need you."

"And waiting for Producer's demand too."

"I'll let you know when I need you." 😎



“I don’t care of your demand, just listen to me.” 😎

“Waiting for orders.”

“BTW I don’t care of your demand too.”

“Waiting for orders.”

```
$ mix new genstage_example --sup
$ cd genstage_example
```

```
defp deps do
  [
    {:gen_stage, "~> 0.11"}
  ]
end
```

```
$ mix do deps.get, compile
```

Producent

```
defmodule GenstageExample.Producer do
  use GenStage

  def start_link(initial \\ 0) do
    GenStage.start_link(__MODULE__, initial, name: __MODULE__)
  end

  def init(counter), do: {:producer, counter}

  def handle_demand(demand, state) do
    events = Enum.to_list(state..(state + demand - 1))
    {:noreply, events, state + demand}
  end
end
```

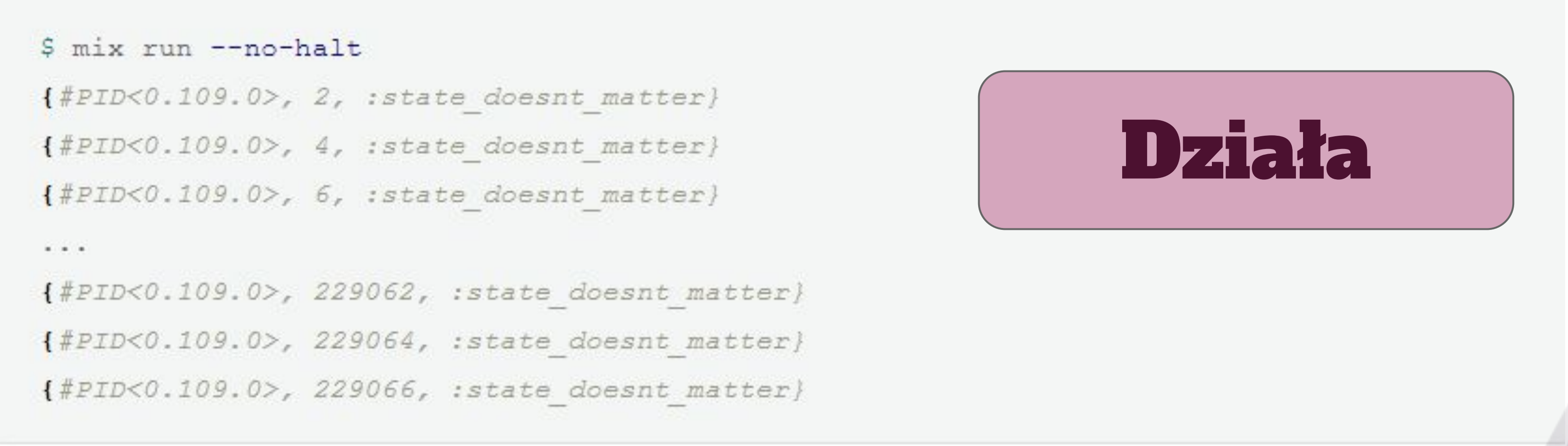
Producent-konsument

```elixir
defmodule GenstageExample.ProducerConsumer do
  use GenStage

  require Integer

  def start_link do
    GenStage.start_link(__MODULE__, :state_doesnt_matter, name: __MODULE__)
  end

  def init(state) do
    {:producer_consumer, state, subscribe_to: [GenstageExample.Producer]}
  end

  def handle_events(events, _from, state) do
    numbers =
      events
      |> Enum.filter(&Integer.is_even/1)

    {:noreply, numbers, state}
  end
end
```

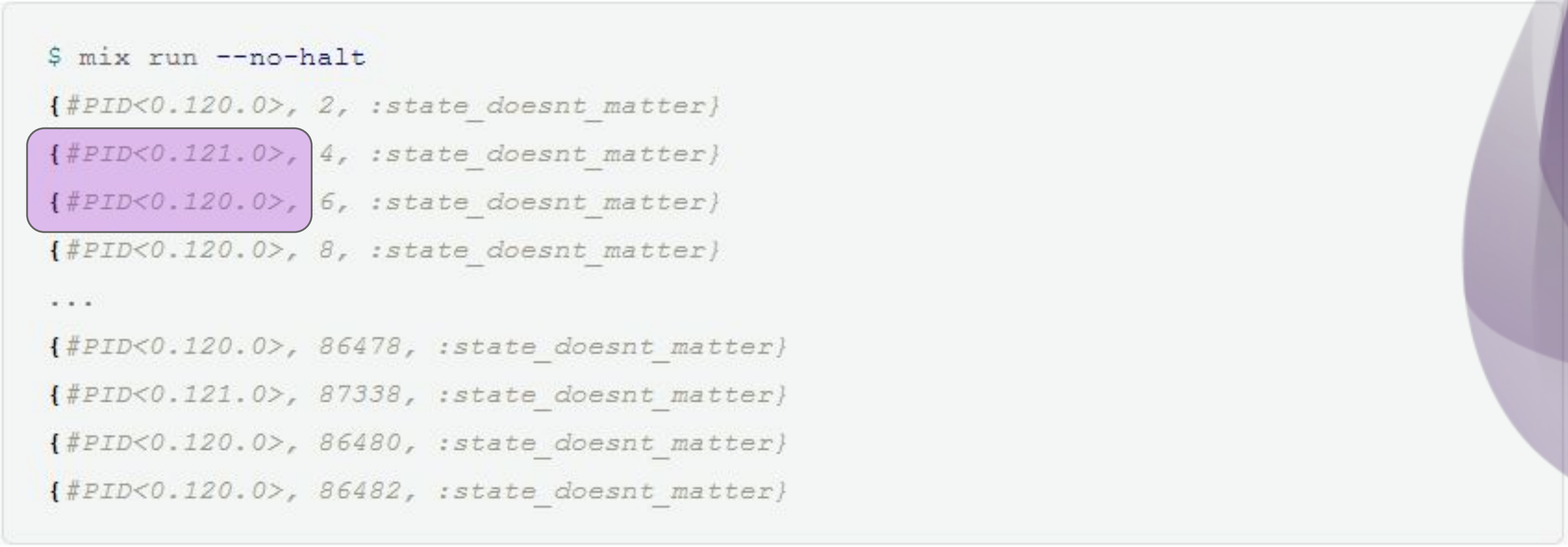
Konsument

```elixir
defmodule GenstageExample.Consumer do
  use GenStage

  def start_link do
    GenStage.start_link(__MODULE__, :state_doesnt_matter)
  end

  def init(state) do
    {:consumer, state, subscribe_to: [GenstageExample.ProducerConsumer]}
  end

  def handle_events(events, _from, state) do
    for event <- events do
      IO.inspect({self(), event, state})
    end

    # As a consumer we never emit events
    {:noreply, [], state}
  end
end
```

Działa



```
$ mix run --no-halt
{#PID<0.109.0>, 2, :state_doesnt_matter}
{#PID<0.109.0>, 4, :state_doesnt_matter}
{#PID<0.109.0>, 6, :state_doesnt_matter}
...
{#PID<0.109.0>, 229062, :state_doesnt_matter}
{#PID<0.109.0>, 229064, :state_doesnt_matter}
{#PID<0.109.0>, 229066, :state_doesnt_matter}
```



```
$ mix run --no-halt
{#PID<0.120.0>, 2, :state_doesnt_matter}
{#PID<0.121.0>, 4, :state_doesnt_matter}
{#PID<0.120.0>, 6, :state_doesnt_matter}
{#PID<0.120.0>, 8, :state_doesnt_matter}
...
{#PID<0.120.0>, 86478, :state_doesnt_matter}
{#PID<0.121.0>, 87338, :state_doesnt_matter}
{#PID<0.120.0>, 86480, :state_doesnt_matter}
{#PID<0.120.0>, 86482, :state_doesnt_matter}
```

Elixir GenStage

★ https://github.com/fsiody/Erlang-Elixir/tree/master/lab5

- https://medium.com/@andreichernykh/elixir-a-few-things-about-genstage-id-wish-to-knew-some-time-ago-b826ca7d48ba
- https://elixirschool.com/ru/lessons/advanced/gen-stage/
- https://github.com/elixir-lang/gen_stage
- https://github.com/elixirschool/elixirschool/blob/master/pl/lessons/advanced/gen-stage.md