GenServer - a cheat sheet

— last version: https://elixir-lang.org/cheatsheets/gen-server.pdf
— reference: https://hexdocs.pm/elixir/GenServer.html

**Initialization:** `.start` → `.init/1`

```elixir
def start_link(opts \ []) do
  GenServer.start_link(__MODULE__, match_this, opts)
end
```

**Returns**
```
{:ok, pid}
:ignore
{:error, term}
```

**Callback**
```
def init(match_this) do
  # process input and compute result
  result
end
```

**Result**
```
{:ok, state}
{:ok, state, then_what}
{:stop, reason}
:ignore
```

**Termination:** `.stop` → `.terminate/2`

```elixir
def stop(pid, reason \ :normal, timeout \ :infinity) do
  GenServer.stop(pid, reason, timeout)
end
```

**Returns**
```
:ok
```

**Callback**
```
def terminate(reason, state) do
  # perform cleanup
  # result will not be used
  end
```

**Asynchronous Operation:** `.cast` → `.handle_cast/2`

```elixir
def your_api_async_op(pid, args) do
  GenServer.cast(pid, match_this)
end
```

**Returns**
```
:ok
```

**Callback**
```
def handle_cast(match_this, state) do
  # process input and compute result
  result
end
```

**Result**
```
{:noreply, state}
{:noreply, state, then_what}
{:stop, reason, state}
```

**Then What**
```
:continue, match_this
```

**Applies Globally**
```
timeout_milliseconds
:hibernate
```

**terminate/2 is also called when**: `.stop` is returned and in case of errors, when `Process.flag(:trap_exit)` is true.
synchronous operation: `call` → `handle_call/3`

```elixir
def your_api_sync_op(pid, args) do
  GenServer.call(pid, match_this)
end
```

```
def handle_call(match_this, from, state) do
  # process input and compute result
  result
end
```

```
{:reply, reply, reply, ...} or
{:stop, _, reply, _}.
```

```elixir
def handle_info(match_this, state) do
  # process input and compute result
  result
end
```

```
{:noreply, state}
{:noreply, state, then_what}
{:stop, reason, state}
```

```elixir
{:^result =

def handle_continue(match_this, state) do
  # process input and compute result
  result
end
```

```
{:noreply, state}
{:noreply, state, then_what}
{:stop, reason, state}
```

---

**footnotes**

— More on exit reasons: https://hexdocs.pm/elixir/Supervisor.html#module-exit-reasons-and-restarts
— use `@impl true` before each definition to guarantee it matches the equivalent GenServer callback.
— callbacks not listed here are: `code_change/3` and `format_status/2`.
— source: https://github.com/elixir-lang/elixir-lang.github.com
— copyright: by its authors, listed in the source — license: CC:BY-SA