THE Supervisor CHEATSHEET

Version 1.0

Initialization

CLIENT

```
def start_link(opts \\ []) do
  Supervisor.start_link(__MODULE__, :ok, opts)
end
```

Returns {:ok, ***pid***}

CALLBACK

```
def init(:ok) do
  return value
end
```

RETVAL

```
supervise(children, options)

:ignore
```

Define `children` in the Supervisor Specification

`supervise(`**`children`**`, options)`

EXAMPLE

```
children = [
  supervisor(FooSup,[:bar, :baz], []),
  worker(Foo, [:baz, :qux, :quux], []),
]
```

SUPERVISOR

```
supervisor(module, arg, options)
```

DEFAULT OPTIONS

```
[id: module,
 function: :start_link,
 restart:  :permanent,
 shutdown: :infinity,
 modules:  [module]]
```

```
FooSup.start_link(:bar, :baz) # Expected to be implemented
```

WORKER

```
worker(module, arg, options)
```

DEFAULT OPTIONS

```
[id: module,
 function: :start_link,
 restart:  :permanent,
 shutdown: 5000,
 modules:  [module]]
```

This invokes:

```
Foo.start_link(:baz, :qux, :quux) # Expected to be implemented
```

Name Registration

Note: Neither `worker/3` nor `supervise/3` do anything special for name registration. It' up to you to pass the arguments to `GenServer.start_link` or `Process.register/2` via **arg**.

EXAMPLE

```
worker(Foo, [:baz, :qux, :quux, name: Foo.Worker], [])
```

calls

```
Foo.start_link(:baz, :qux, :quux, name: Foo.Worker)
```

Then, pass `[name: Foo.Worker]` *(4th parameter)* into `GenServer.start_link` OR `Process.register/2`.

THE Supervisor CHEATSHEET

Version 1.0

Name Registration (Optional)

Note: I prefer passing in a *singly nested list of arguments* for GenServers because I can hand them to `GenServer.start_link`, which invokes the `init/1` callback – without modifying the argument.

EXAMPLE

```
worker(Foo, [[:baz, :qux, :quux], name: Foo.Worker], [])
```

calls

```
Foo.start_link([:baz, :qux, :quux], name: Foo.Worker)
```

Then, pass `[name: Foo.Worker]` *(2nd parameter)* into `GenServer.start_link` OR `Process.register/2`.

Define `children` in the Supervisor Specification

`supervise(`**`children`**`, options)`

OPTIONS

```
[id: module,
 function: :start_link,
 restart:  :permanent,
 shutdown: 5_000,
 modules:  [module]]
```

1. Determine the **`restart`** values:

PERMANENT

The child process is always restarted

TEMPORARY

The child process is never restarted (not even when the supervisor's strategy is `:rest_for_one` or `:one_for_all`)

TRANSIENT

The child process is restarted only if it terminates abnormally (exit reason other than `:normal`, `:shutdown`, `{:shutdown, term}`

OPTIONS

```
[id: module,
 function: :start_link,
 restart:  :permanent,
 shutdown: 5_000,
 modules:  [module]]
```

2. Determine the **`shutdown`** values:

BRUTAL KILL

Child process is unconditionally terminated with `Process.exit(child, :kill)`

INFINITY

If child is a *supervisor*, this gives the subtree enough time to shutdown.
If child is a *worker*, you can also use this – with care!

5_000

When given an integer, the supervisor terminates the child process using `Process.exit(child, :shutdown)` and waits for an exist signal within the time (in milliseconds). Otherwise, the child process is brutally killed.

THE Supervisor CHEATSHEET

Version 1.0

Define `options` in the Supervisor Specification

`supervise(children, options)`

EXAMPLE

```
options = [
  strategy: :one_for_all, max_restarts: 3, max_seconds: 5
]
```

1. Decide on the **`strategy`**:

ONE FOR ONE

If a child process terminates, only that process is restarted.

ONE FOR ALL

If a child process terminates, all other child processes are terminated and then all child processes (including the terminated one) are restarted.

REST FOR ONE

If a child process terminates, the "rest" of the child processes, i.e., the child processes after the terminated one in start order, are terminated. Then the terminated child process and the rest of the child processes are restarted.

SIMPLE ONE FOR ONE

Similar to `:one_for_one` but suits better when dynamically attaching children. This strategy requires the supervisor specification to contain only one child. Many functions in this module behave slightly differently when this strategy is used.

2. Determine the **`restart`** threshold:

This reads:

Using a *one for all strategy*, allow for a *maximum of 3 restarts within 5 seconds*.