What is a flow?
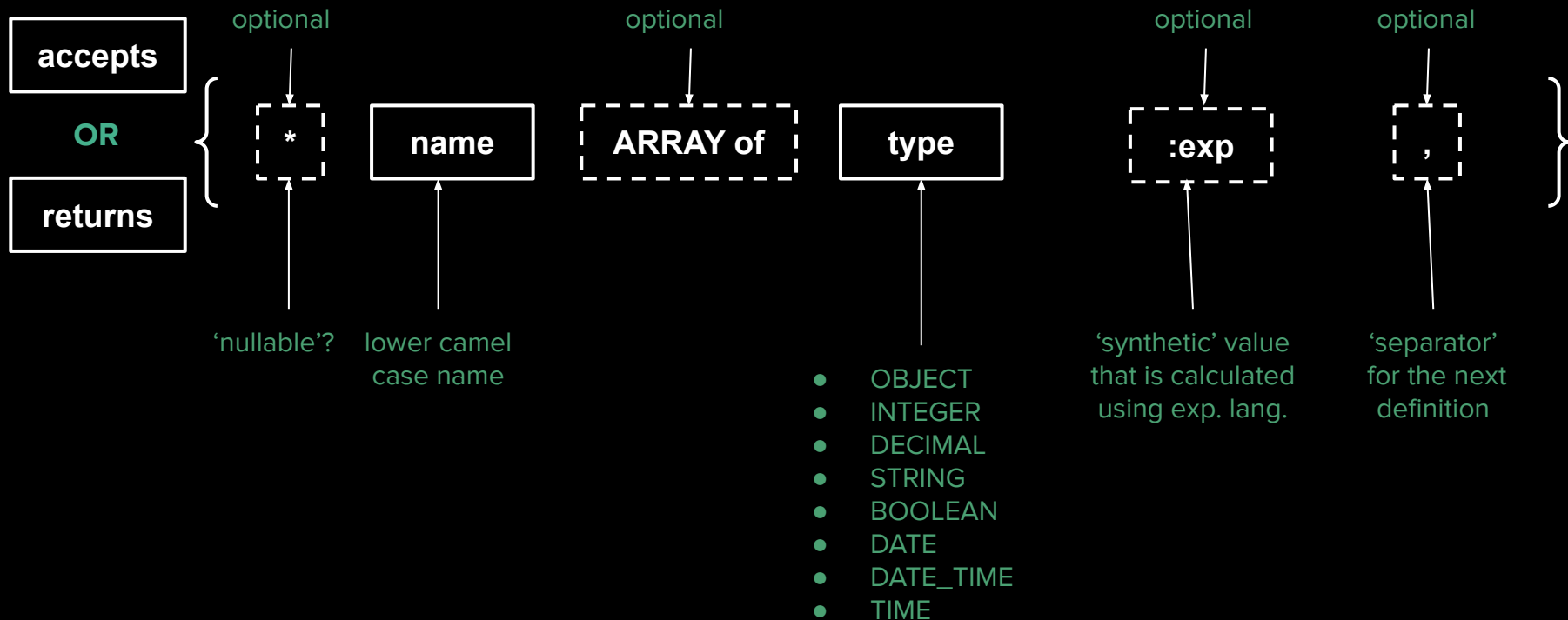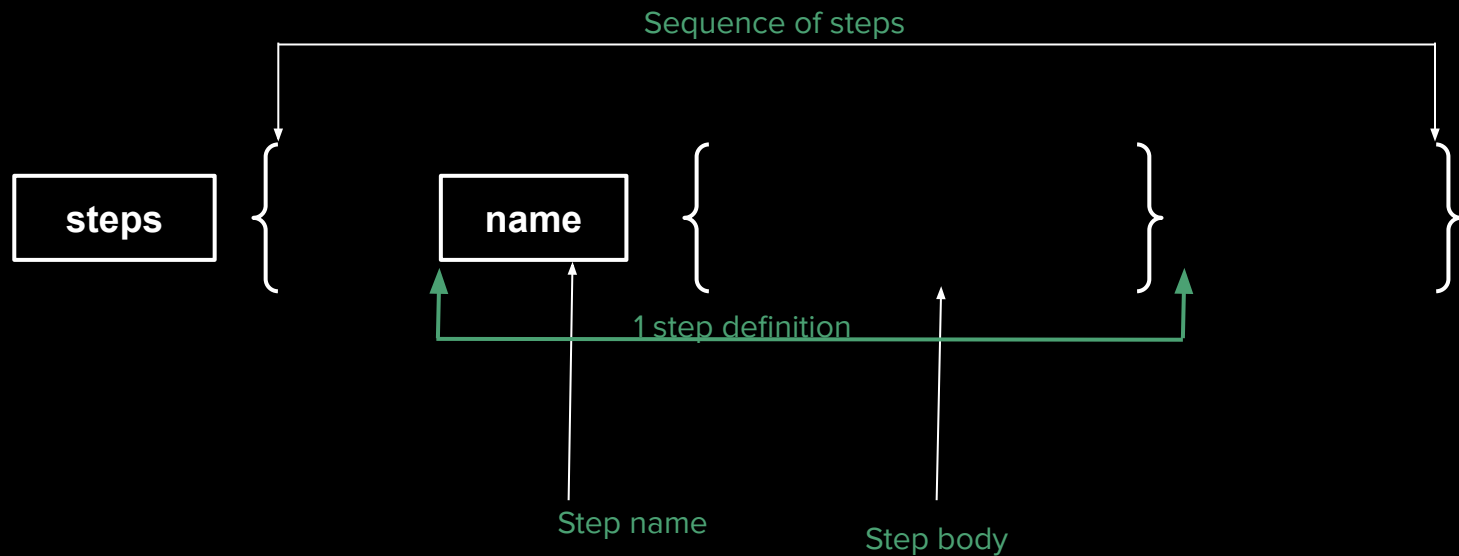
Flow is a 'function' with 'input/output' parameters and a 'body'

- input parameters are called 'accepts' definitions
- output parameters are called 'returns' definitions
- body is a sequence of 'steps' definitions

Defining 'accepts' and 'returns' = change to typescript

```
accepts
   OR
returns
```

{

optional
  *
'nullable'?

name
lower camel
case name

optional
ARRAY of

type
- OBJECT
- INTEGER
- DECIMAL
- STRING
- BOOLEAN
- DATE
- DATE_TIME
- TIME

optional
  :exp
'synthetic' value
that is calculated
using exp. lang.

optional
  ,
'separator'
for the next
definition

}

Defining flow 'body'

Sequence of steps

steps { name { } }

1 step definition

Step name

Step body

Defining flow **'step body'**

define one of
the 'actions'

sequence of
'when-then' pairs

define one of
the 'conclusions'

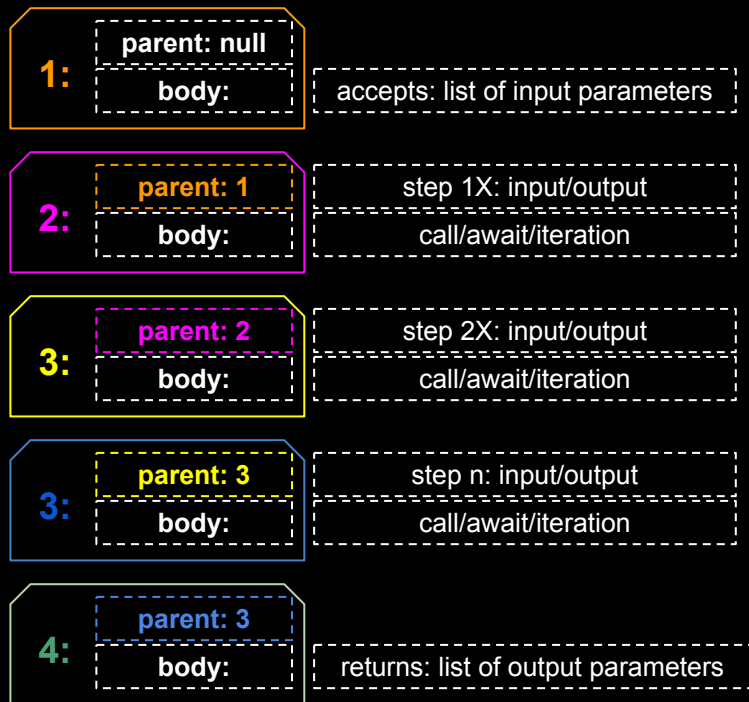|		
maps <exp> to <steps>		**end-as <exp>**
calls <exp>	**when <exp>** **then**	**continue**
await <exp>		**<step name>**
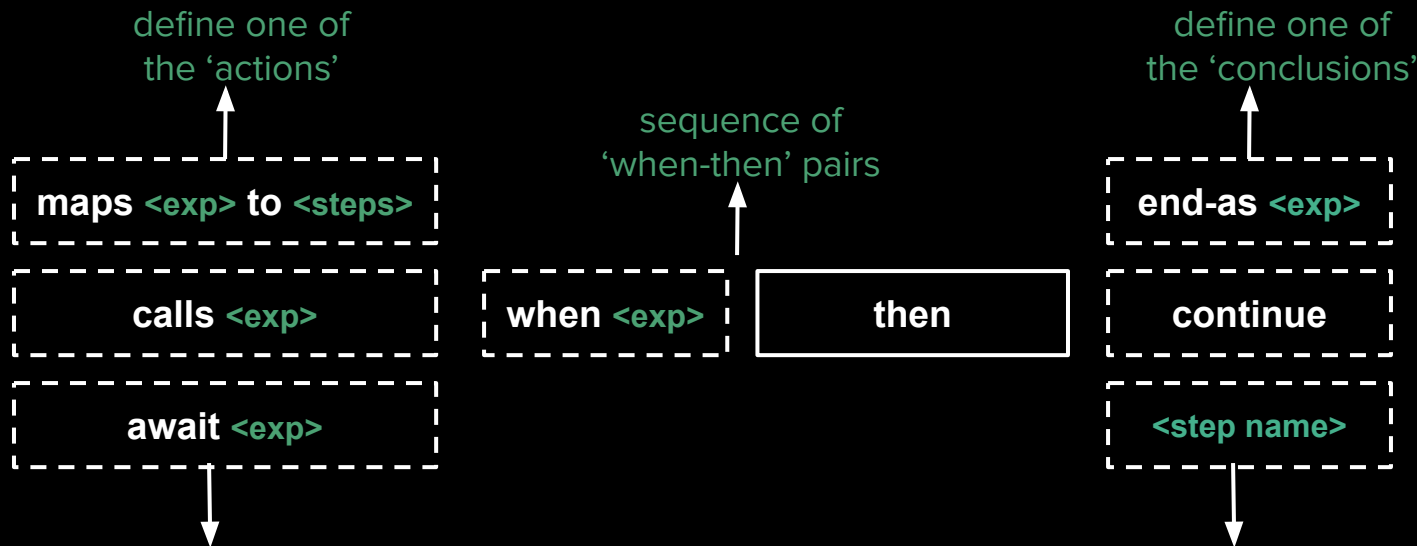
- 'maps-to' is a an iteration of sub steps
- 'calls' is sync. invocation of one or more assets
- 'await' is async invocation of one or more assets

- 'end-as' exits the flow with a concrete 'result'
- 'continue' proceeds to the next flow in the sequence
- 'step-name' is the name of the step where to jump

What happens when flow is executed?

- **a flow is executed by running 'accepts', 'returns' and 'steps' in their respective order**
- **a flow execution produces a 'tree' as a result, which contains a 'parent' and a 'body'**
- **each 'node' in the 'tree' can be viewed as a 'git commit'**

1:
- parent: null
- body: accepts: list of input parameters

2:
- parent: 1
- body: step 1X: input/output — call/await/iteration

3:
- parent: 2
- body: step 2X: input/output — call/await/iteration

3:
- parent: 3
- body: step n: input/output — call/await/iteration

4:
- parent: 3
- body: returns: list of output parameters

Flow definition example

- accepts
- step 1X
- step 2X
- step n
- returns