Wirtual

How does it work ?

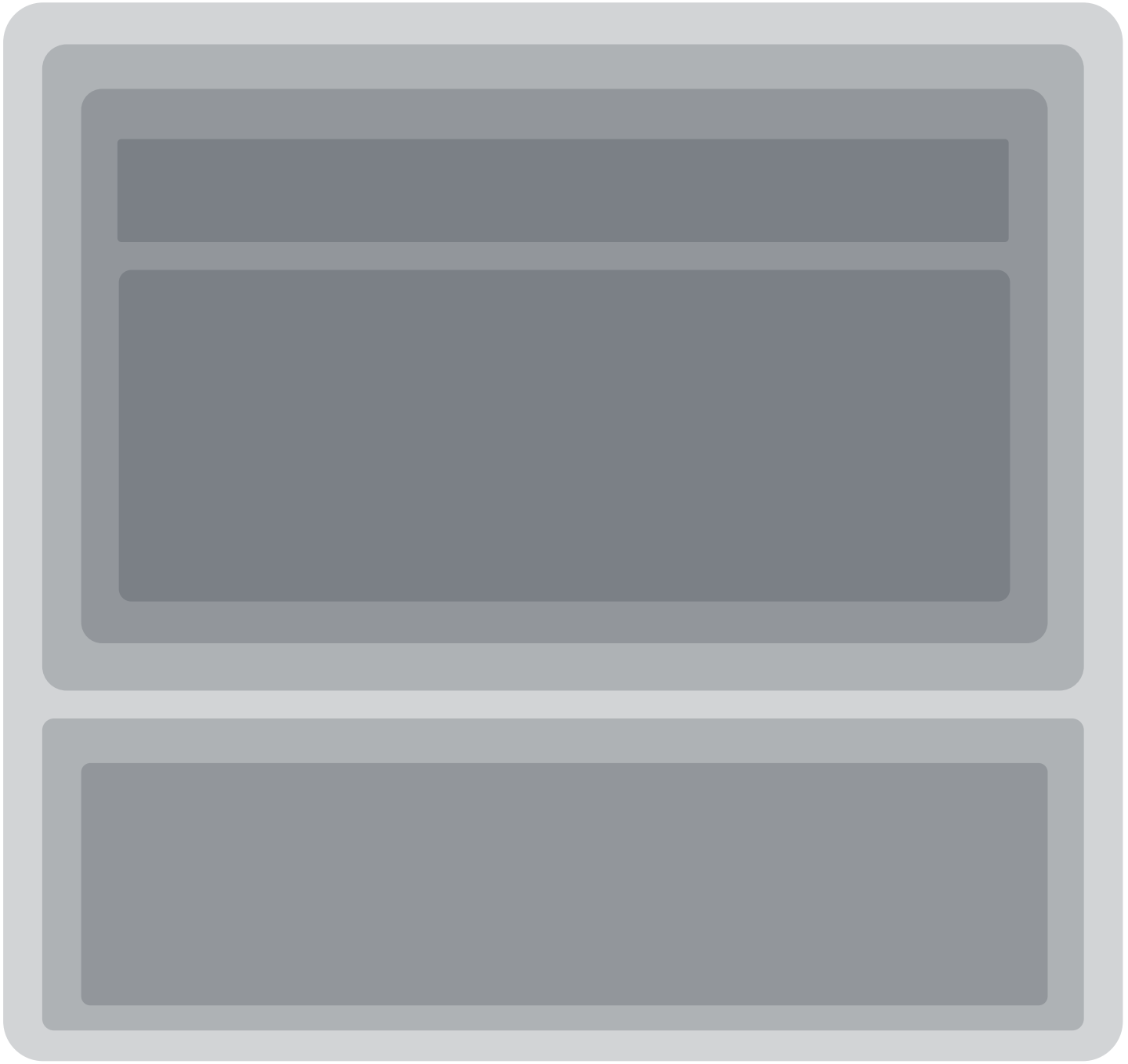
- Developer:
  - Includes wirtual library.
  - Writes simple HTML code.
- The rest is pretty much handled by the compiler.
- Compiler (very briefly):
  - Parses the HTML
  - Creates the canvas.
  - Compiles components and adds them into the canvas.
  - Watches for changes.

Compiler

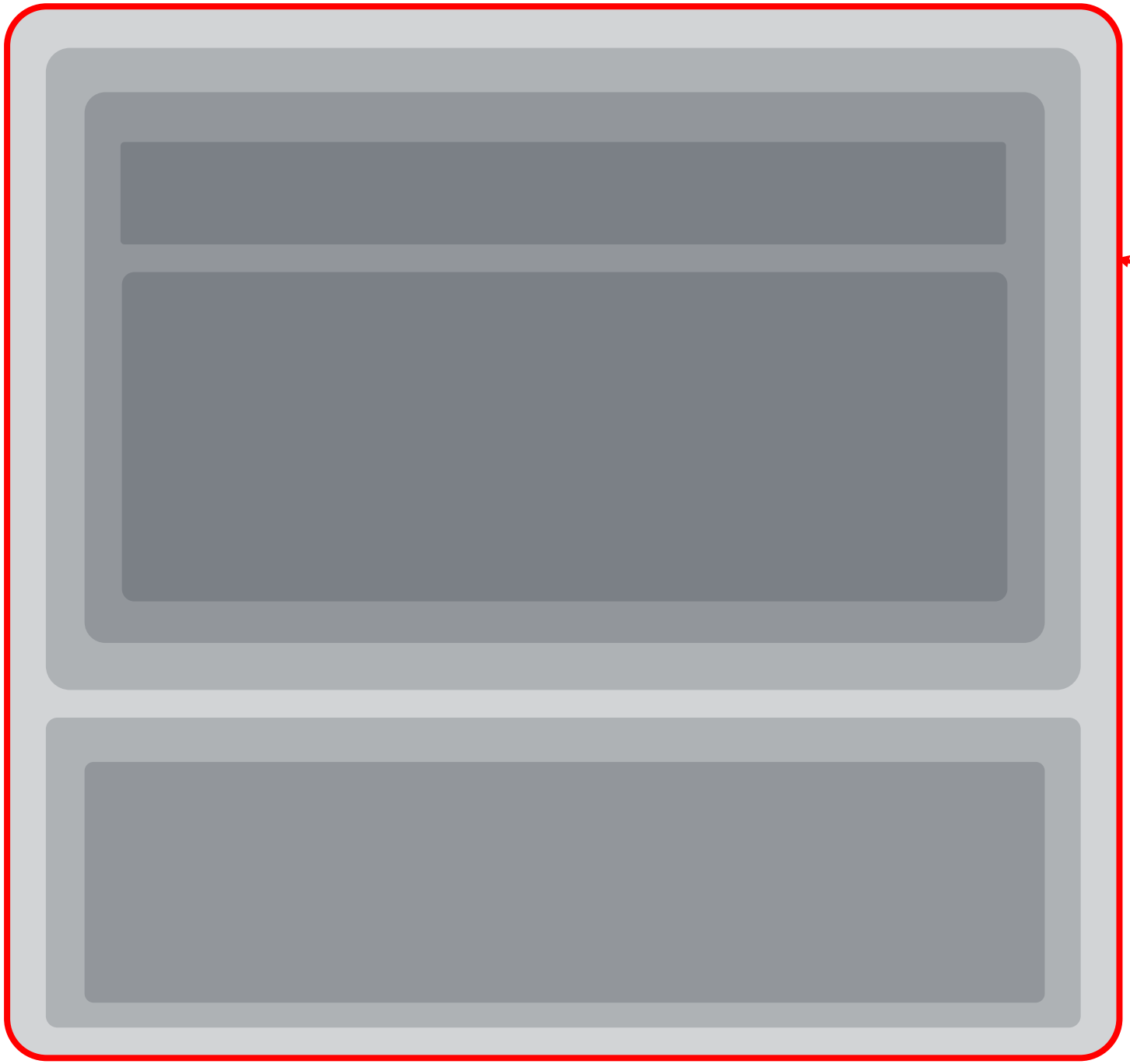
How it works – step by step

Step 1 – Stamp & Store

Start from the root, scan all the elements, stamp them with a unique ID, create the linked tree structure, store it.

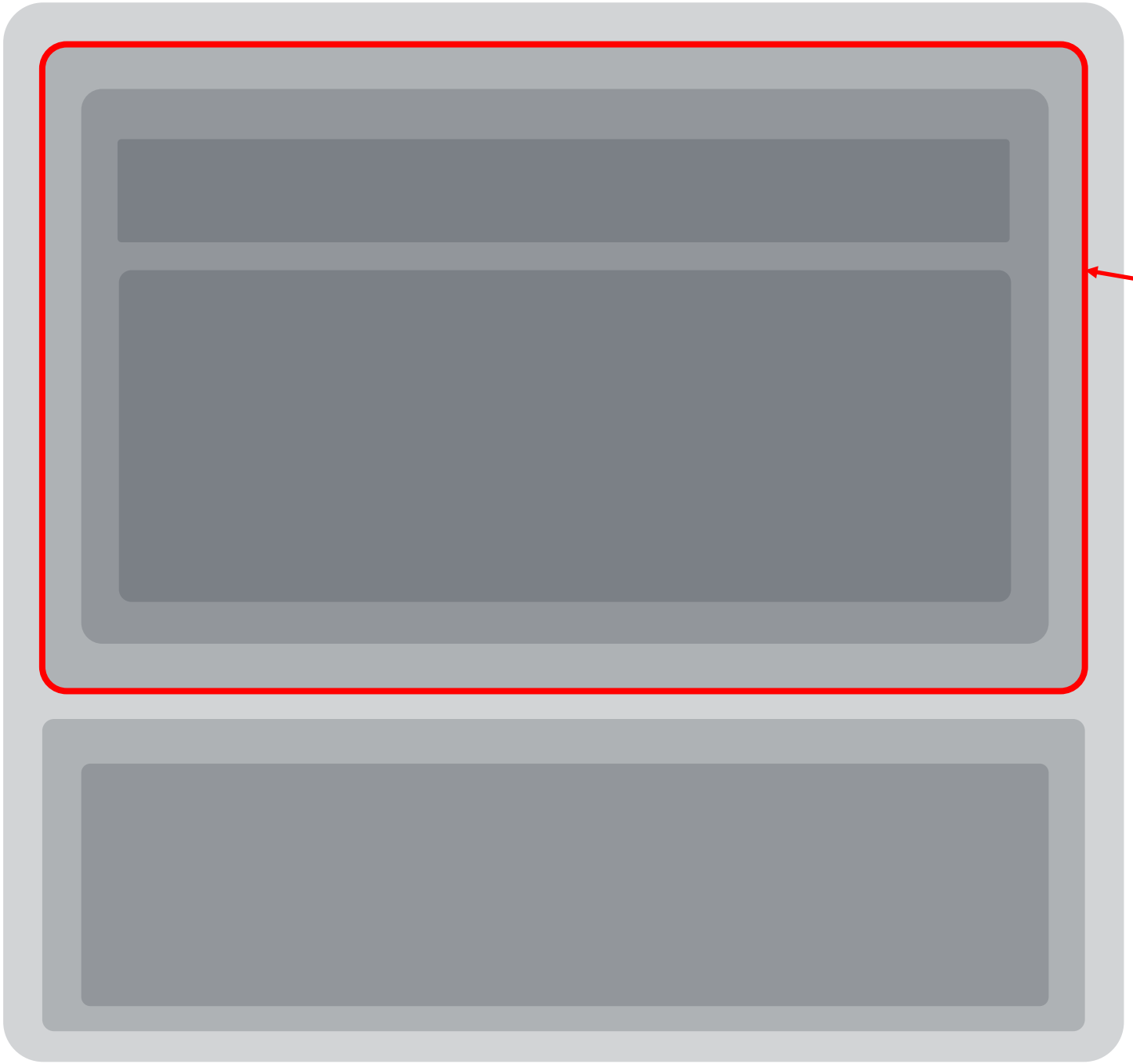


This represents typical nested DOM elements.



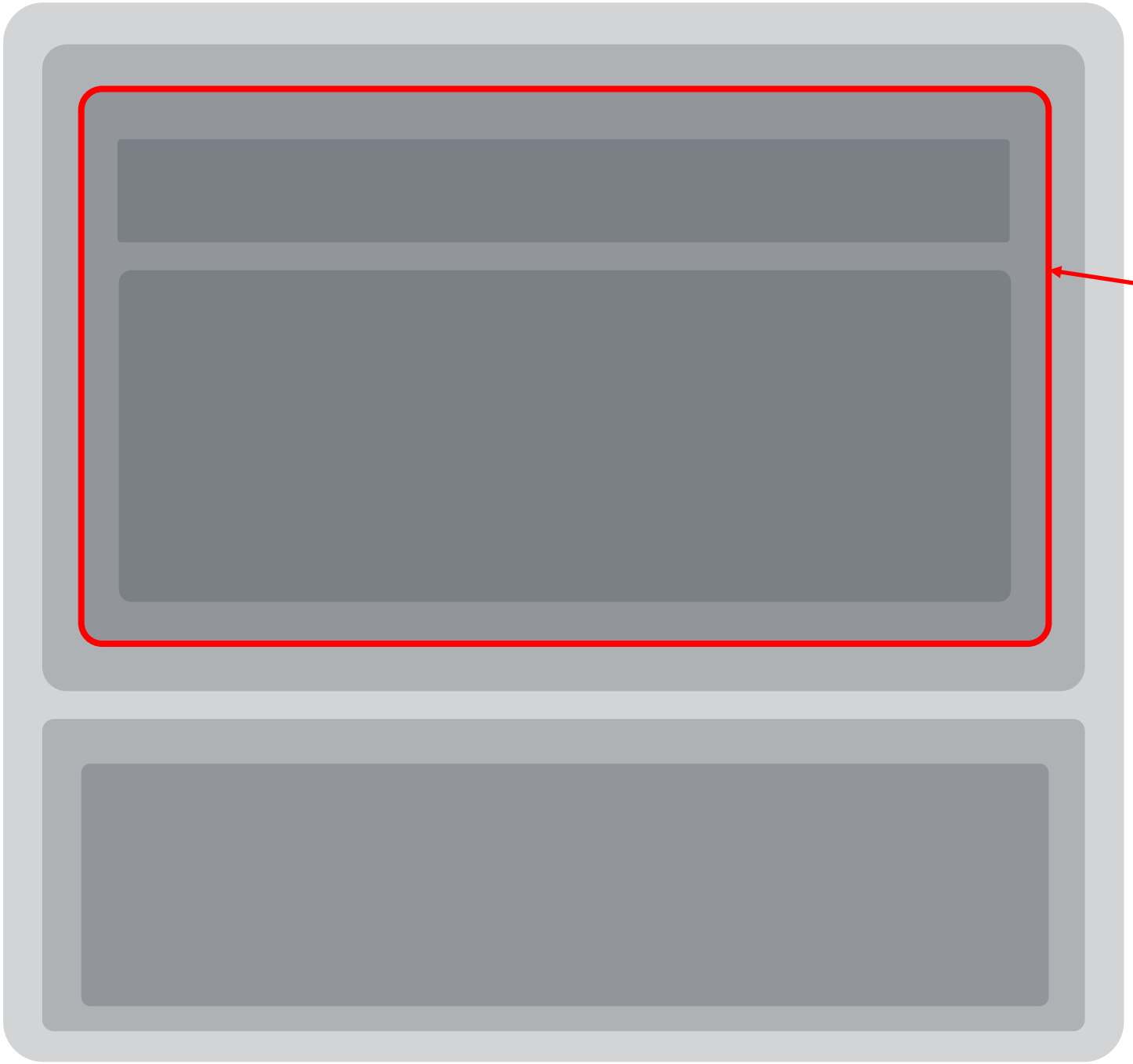
Start from the root

id	_we5gege5g
hash	<divclass="wr-container…
dTarget	(pointer)
vTarget	null
parent	null
children	[‘_w…’, ‘_w…’]



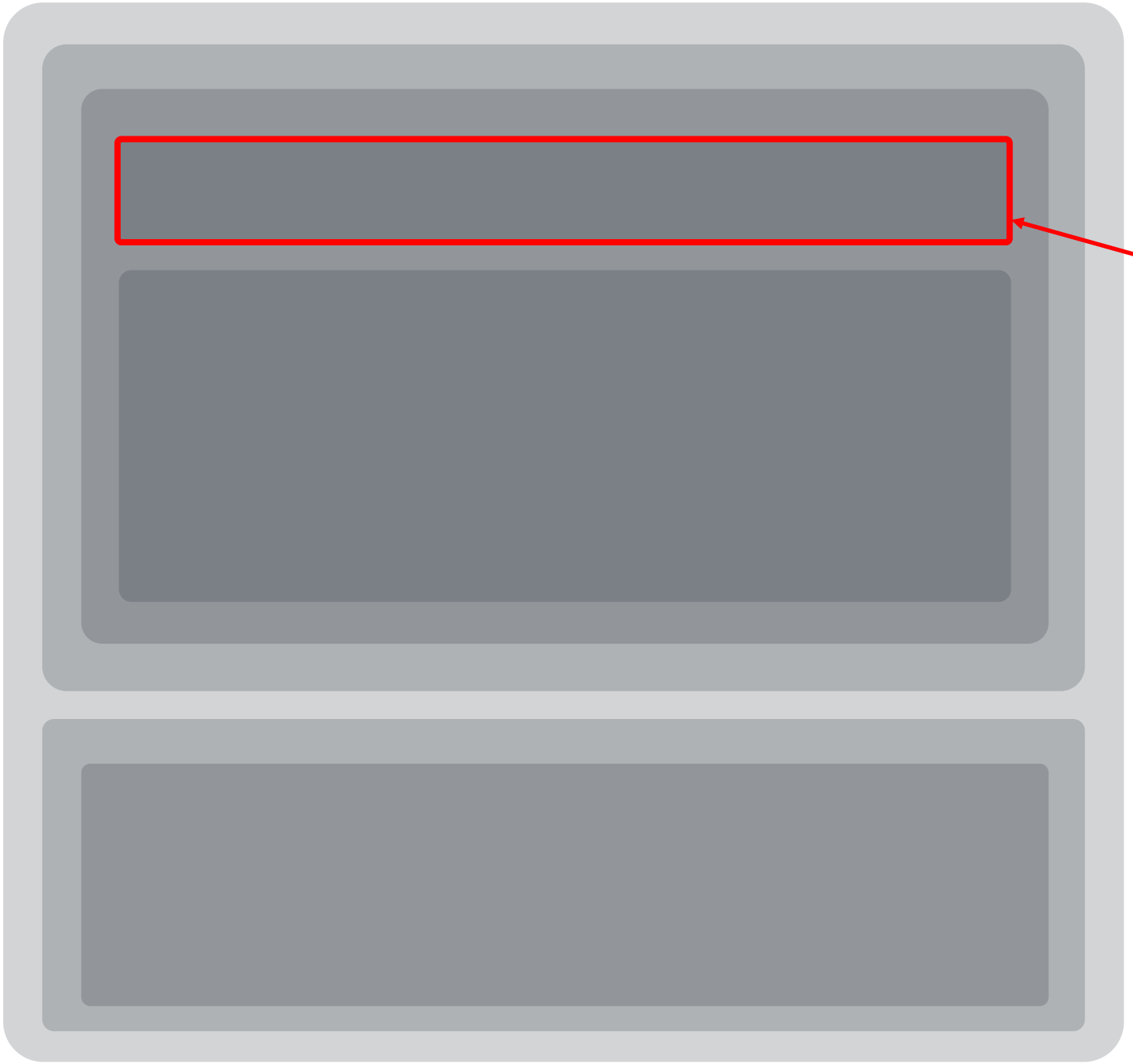
Recursively visit all children

id	_w4tsrgsrg
hash	<divclass="wr-depth…
dTarget	(pointer)
vTarget	null
parent	'_w…'
children	['_w…']



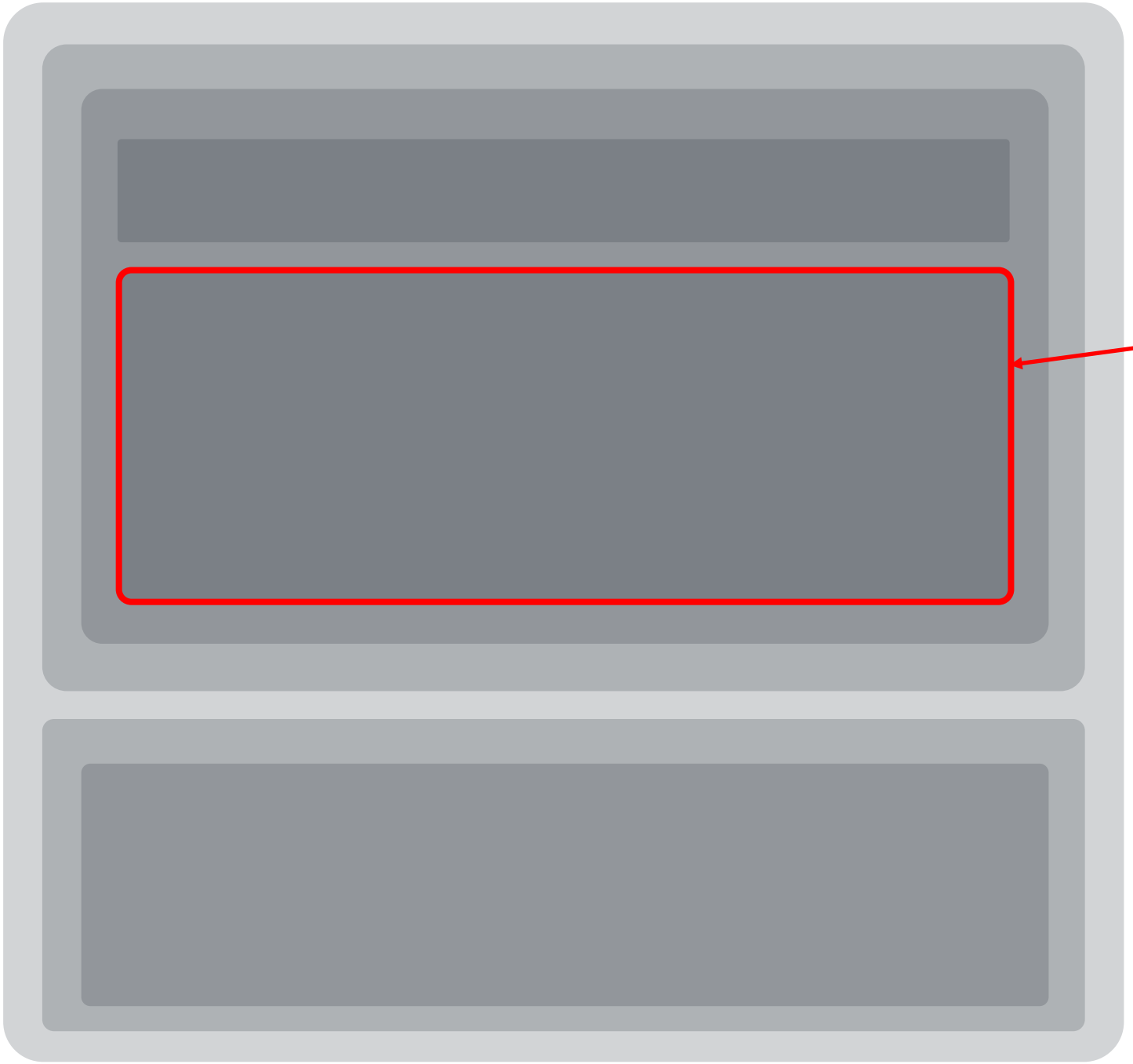
In a depth-first manner…

id	_w4g4g4srg
hash	<divclass="wr-axis…
dTarget	(pointer)
vTarget	null
parent	'_w…'
children	['_w…', '_w…']



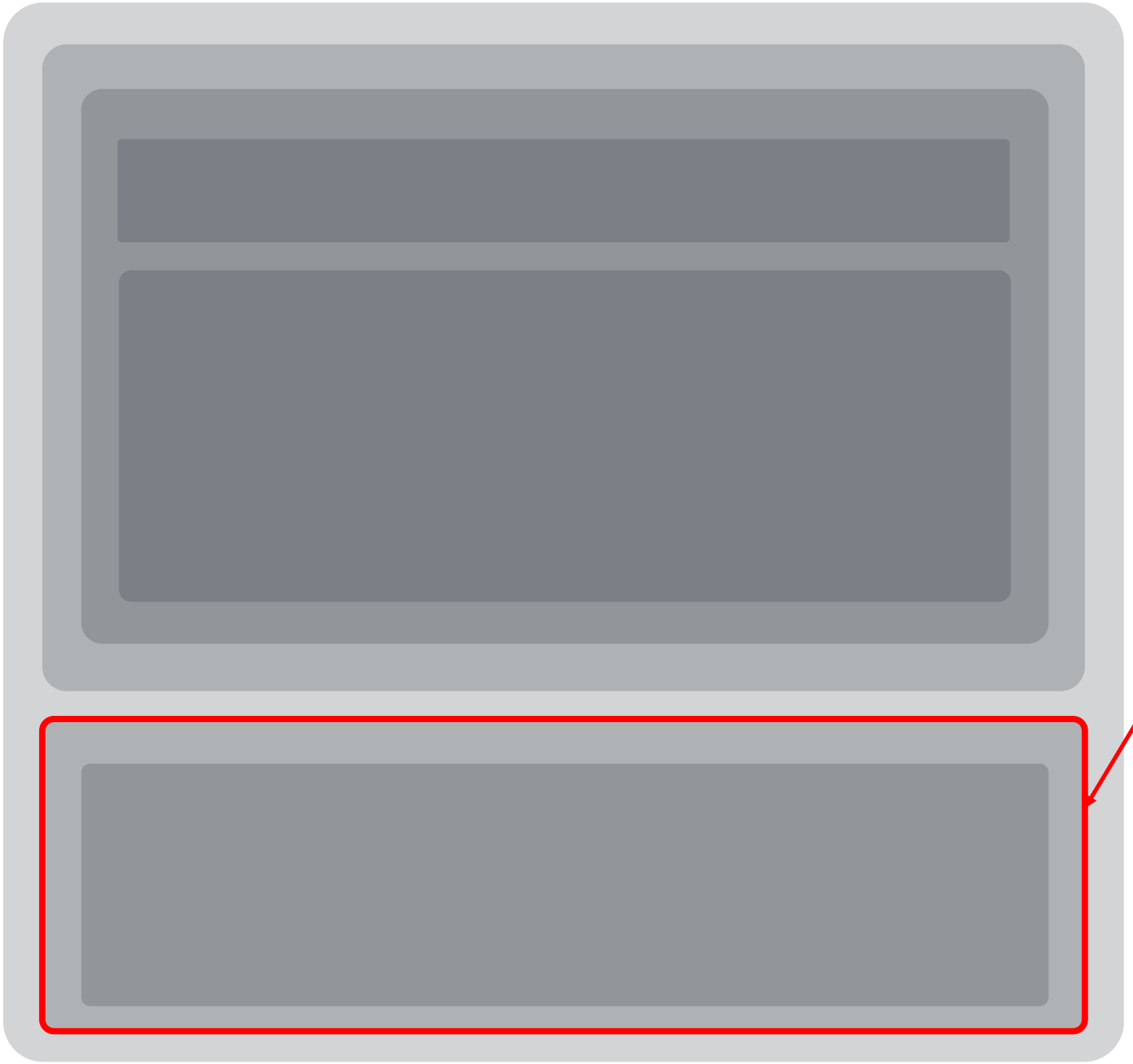
In a depth-first manner…

id	_wh5drgg
hash	<divclass=“wr-level…
dTarget	(pointer)
vTarget	null
parent	’_w…’
children	[]



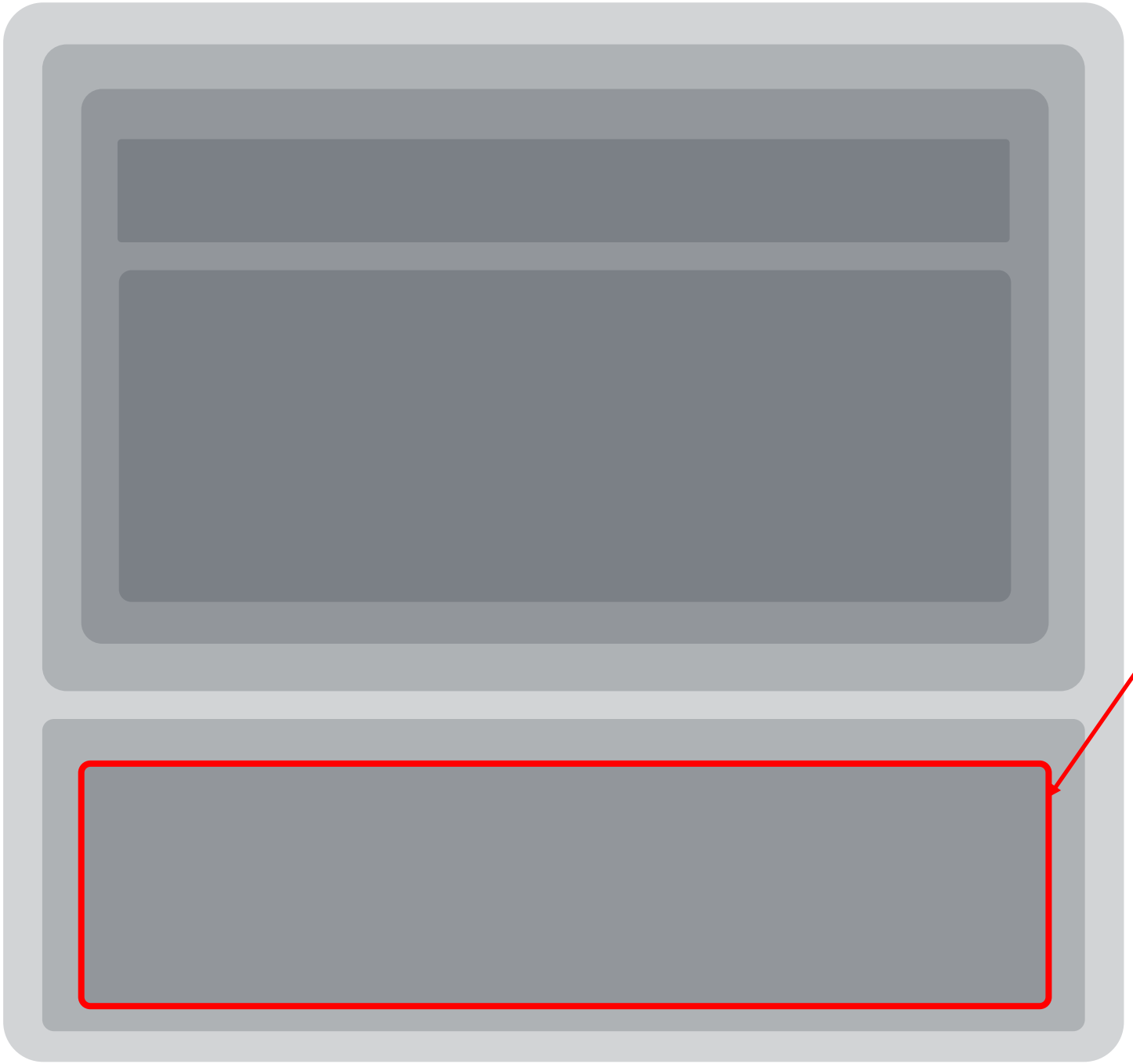
In a depth-first manner…

id	_wh5d4wg4wf
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	null
parent	'_w…'
children	[]



In a depth-first manner…

id	_wh5d4wg4wf
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	null
parent	'_w…'
children	['_w…']

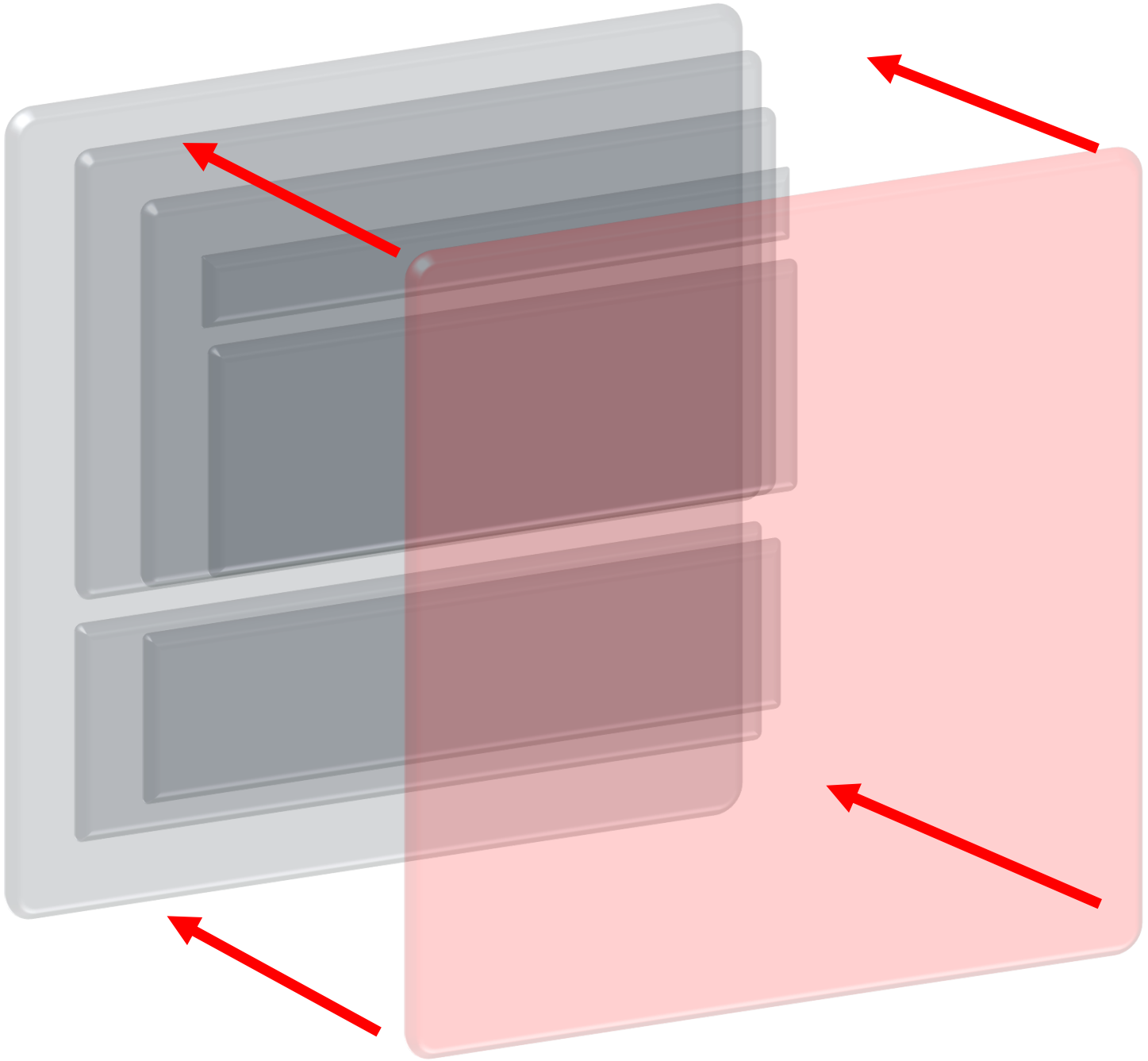


In a depth-first manner…

id	_wh5d4wg4wf
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	null
parent	'_w…'
children	[]

Step 2 – Add the canvas

Appends the canvas on top of the DOM

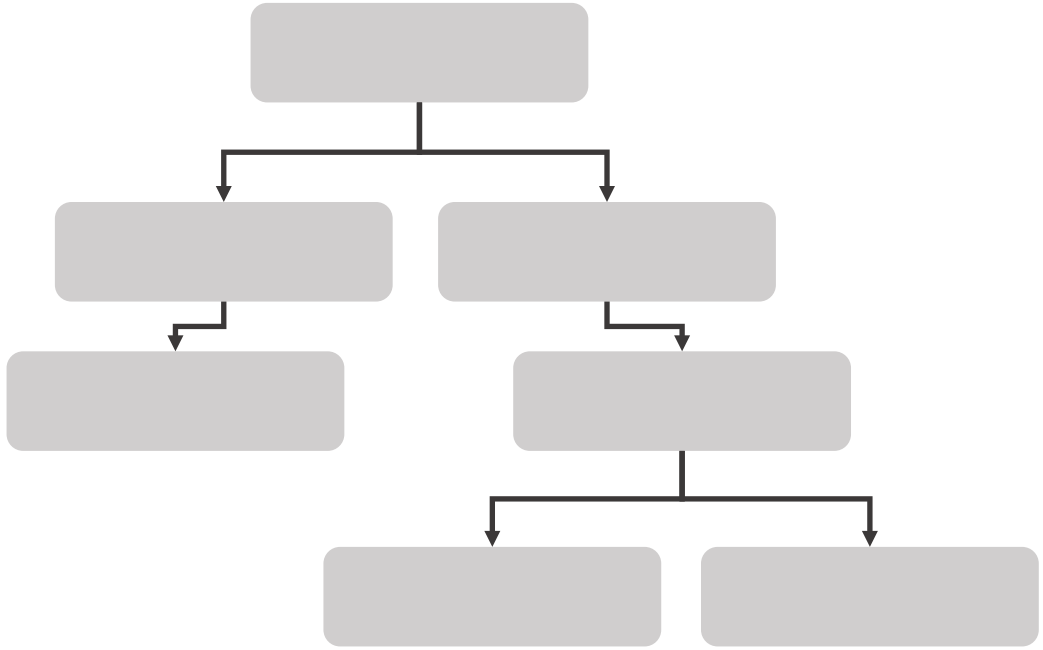


Append the canvas on top of the DOM elements.

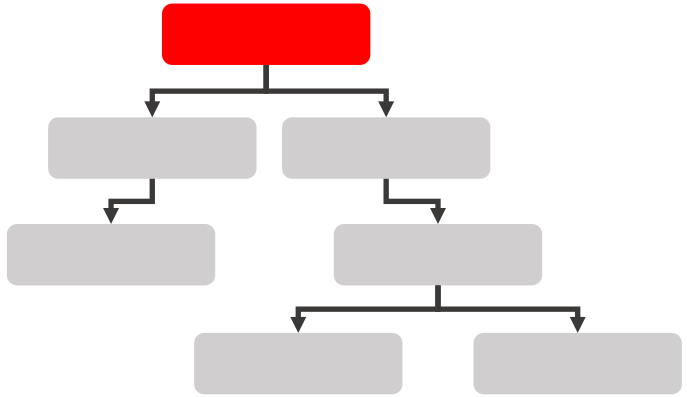
Step 3 – Compile

Compile elements from the register, create 3D components, add them to the canvas

- The tree below represents stored elements in the register.
- Scans the linked tree recursively, in a depth-first manner.
- Creates 3D components and/or passes a payload to the children.
- Populates the vTarget and points to the created elementz

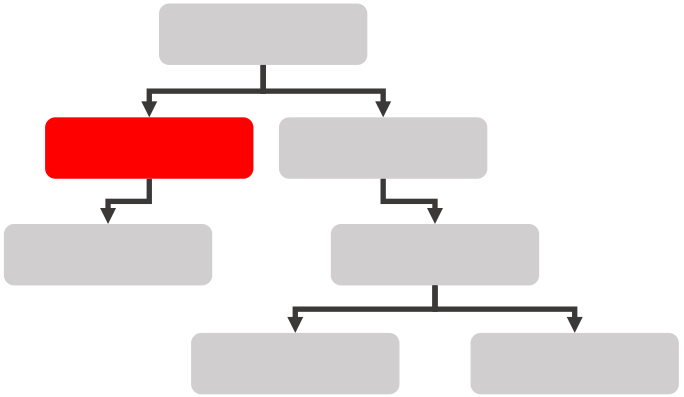


Start from the root



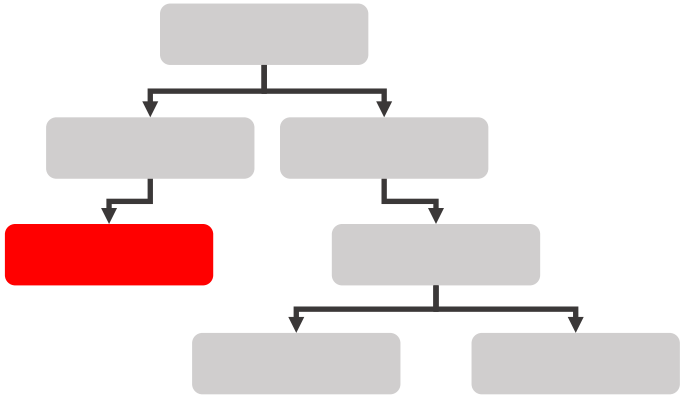
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	null
children	[‘_w…’, ‘_w…’]

Recursively compile all children

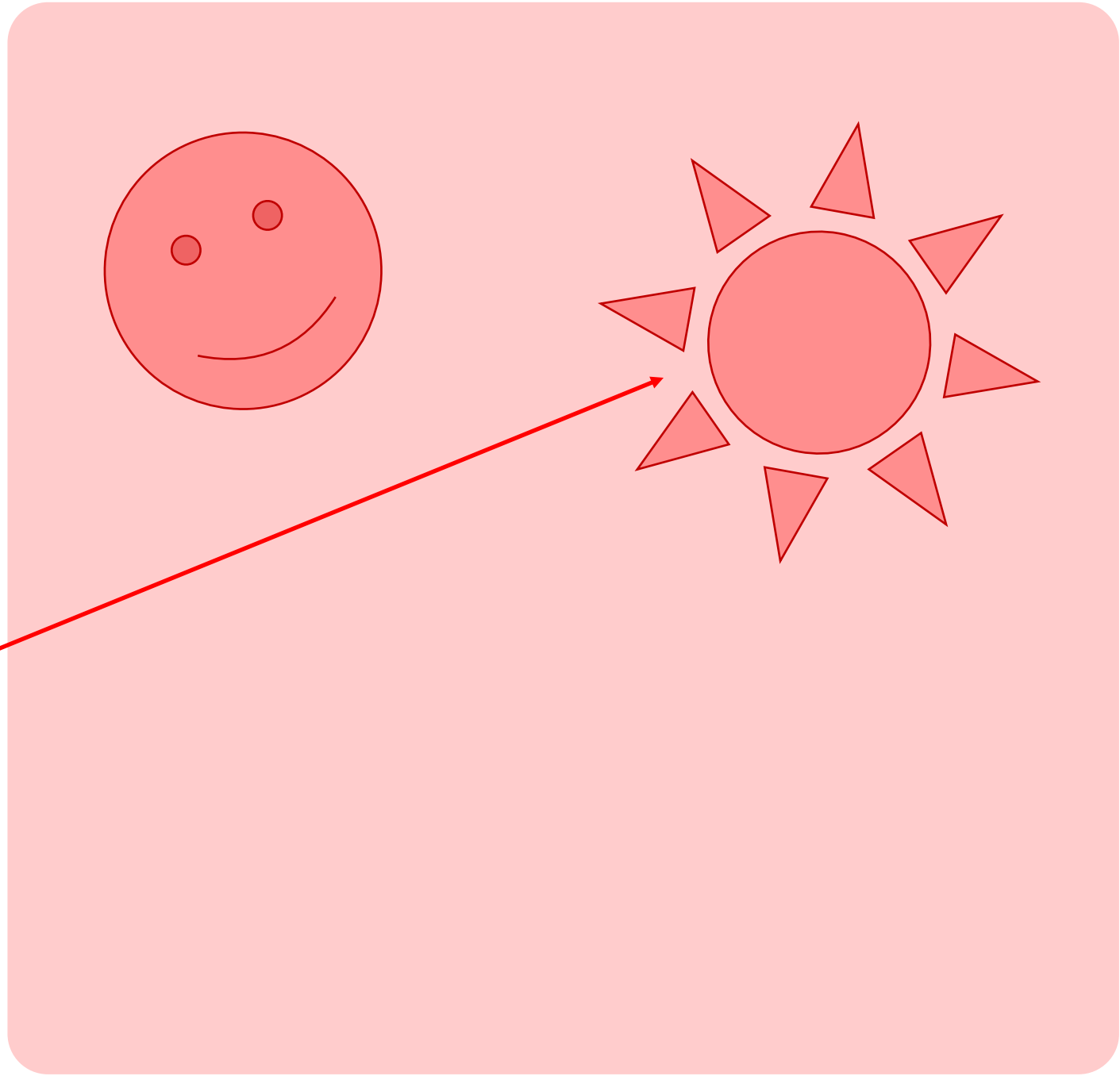


id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	['_w…']

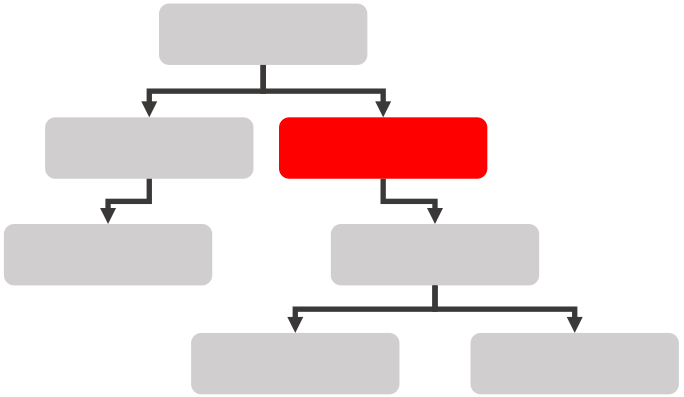
In a depth-first manner



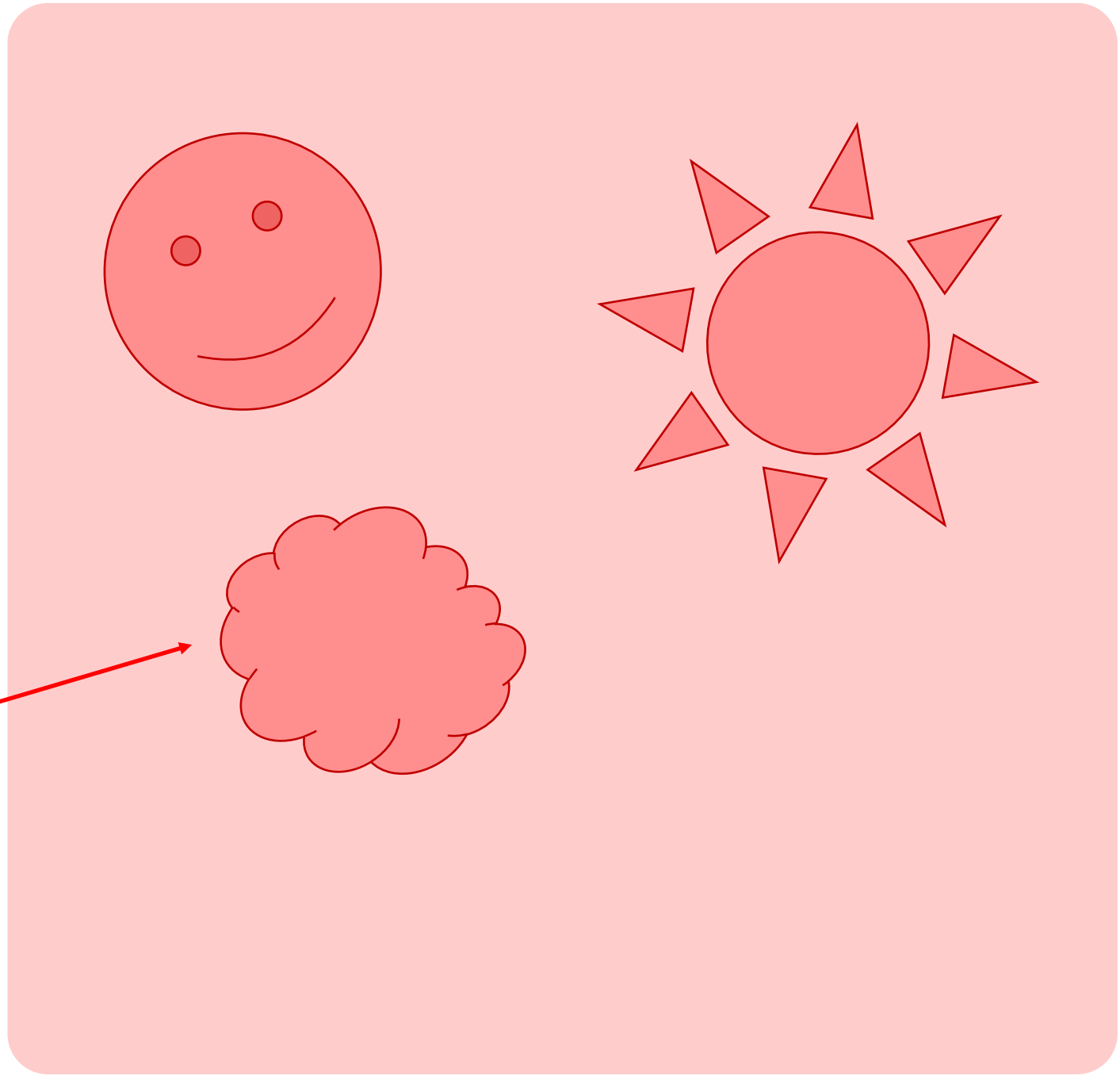
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	[]



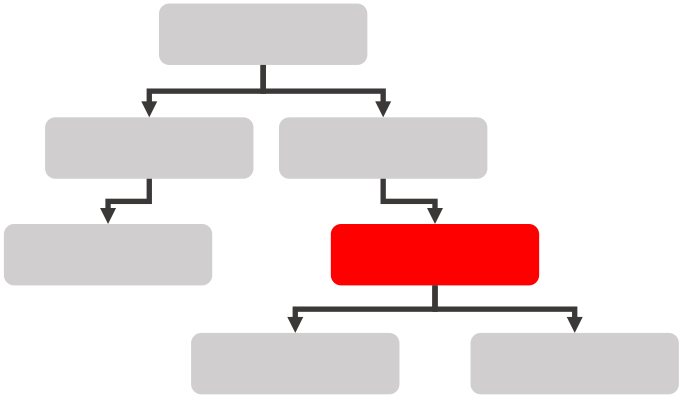
In a depth-first manner



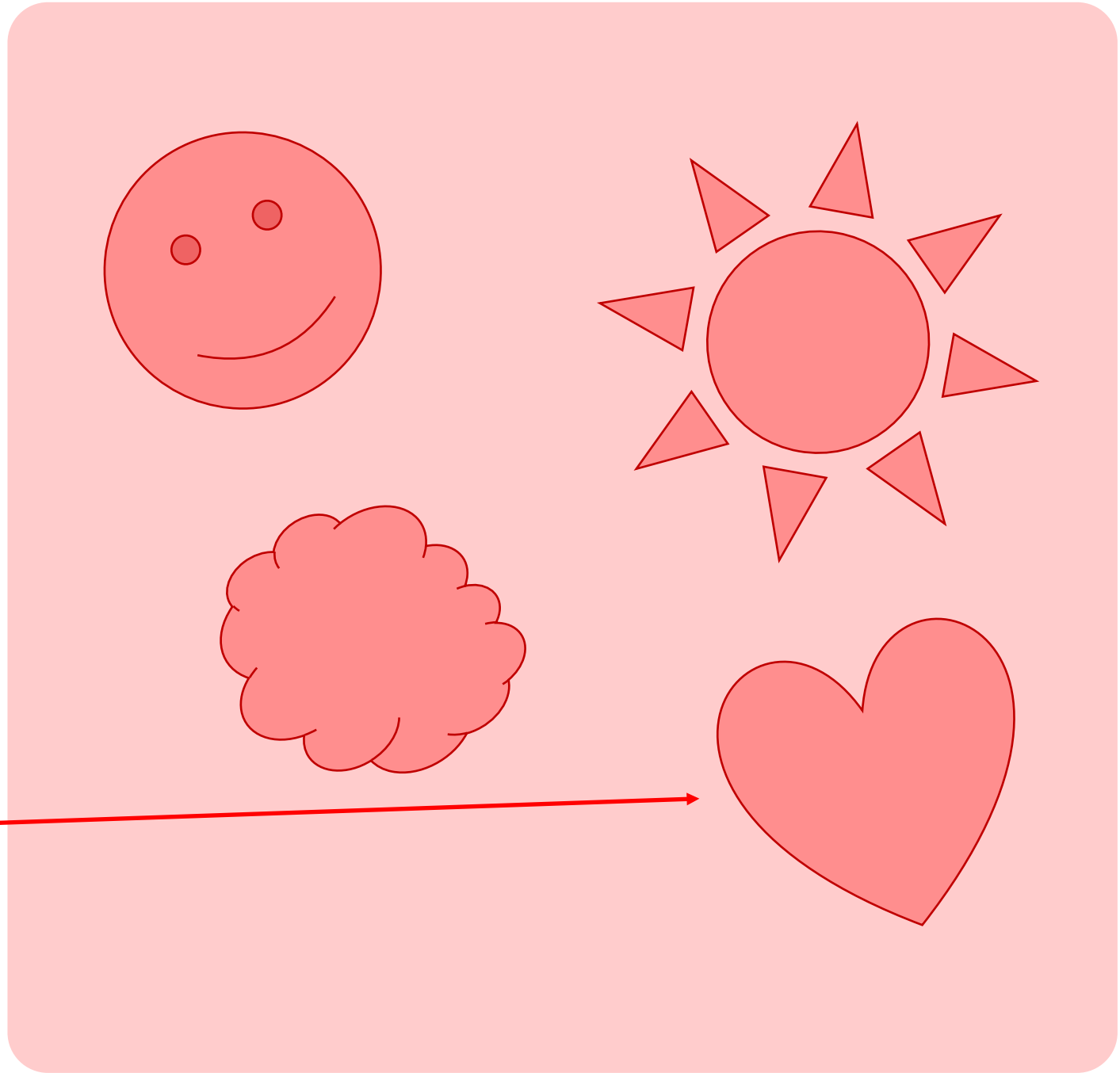
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	['_w…']



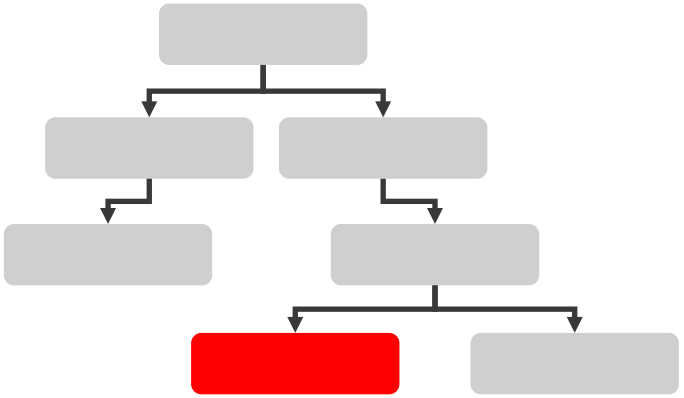
In a depth-first manner



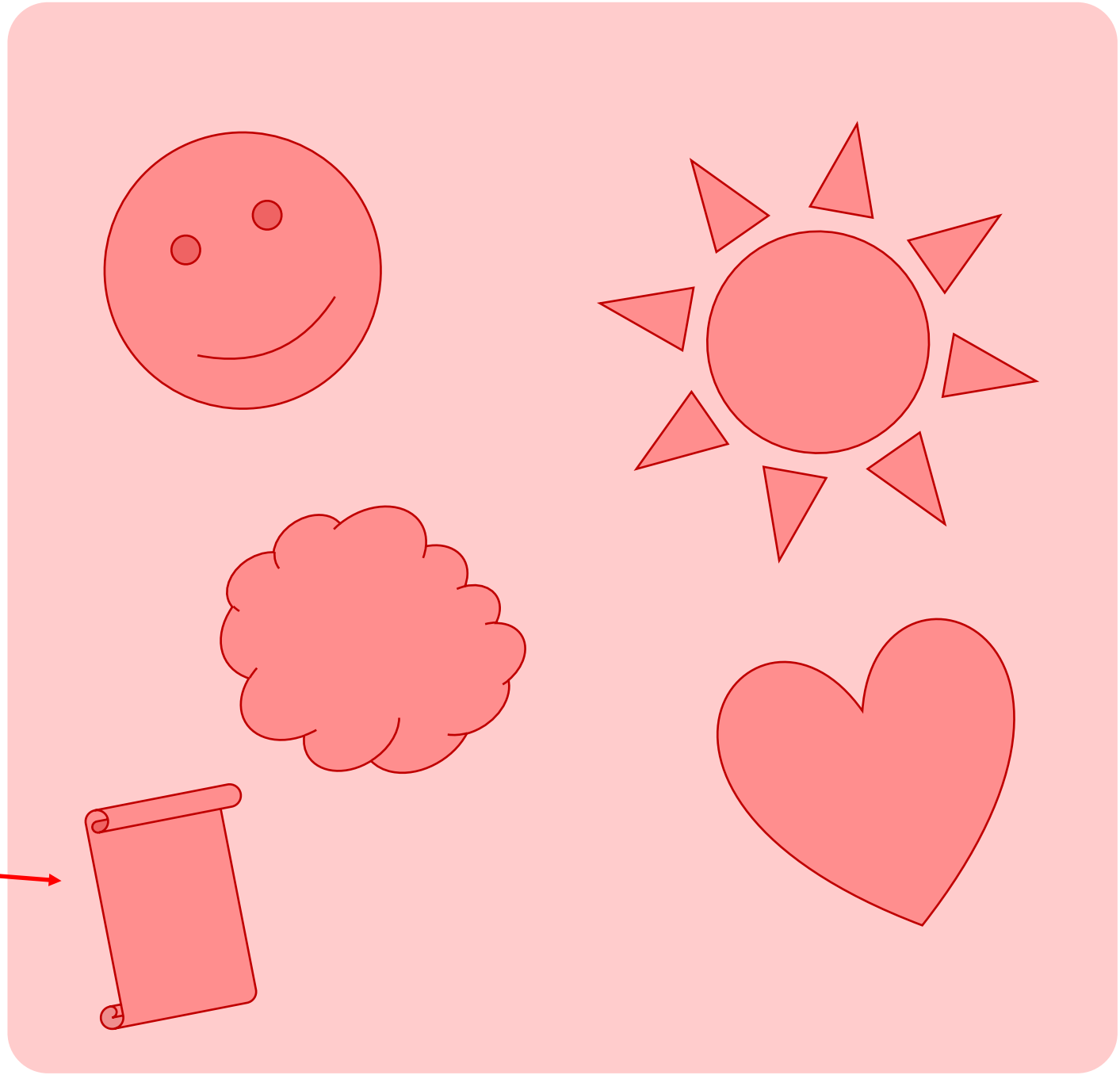
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	['_w…', '_w…']



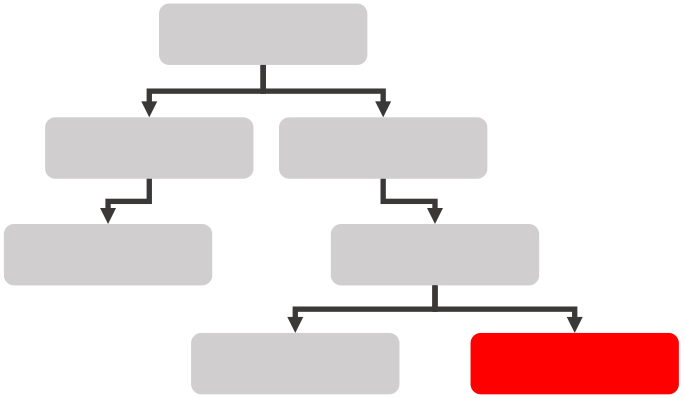
In a depth-first manner



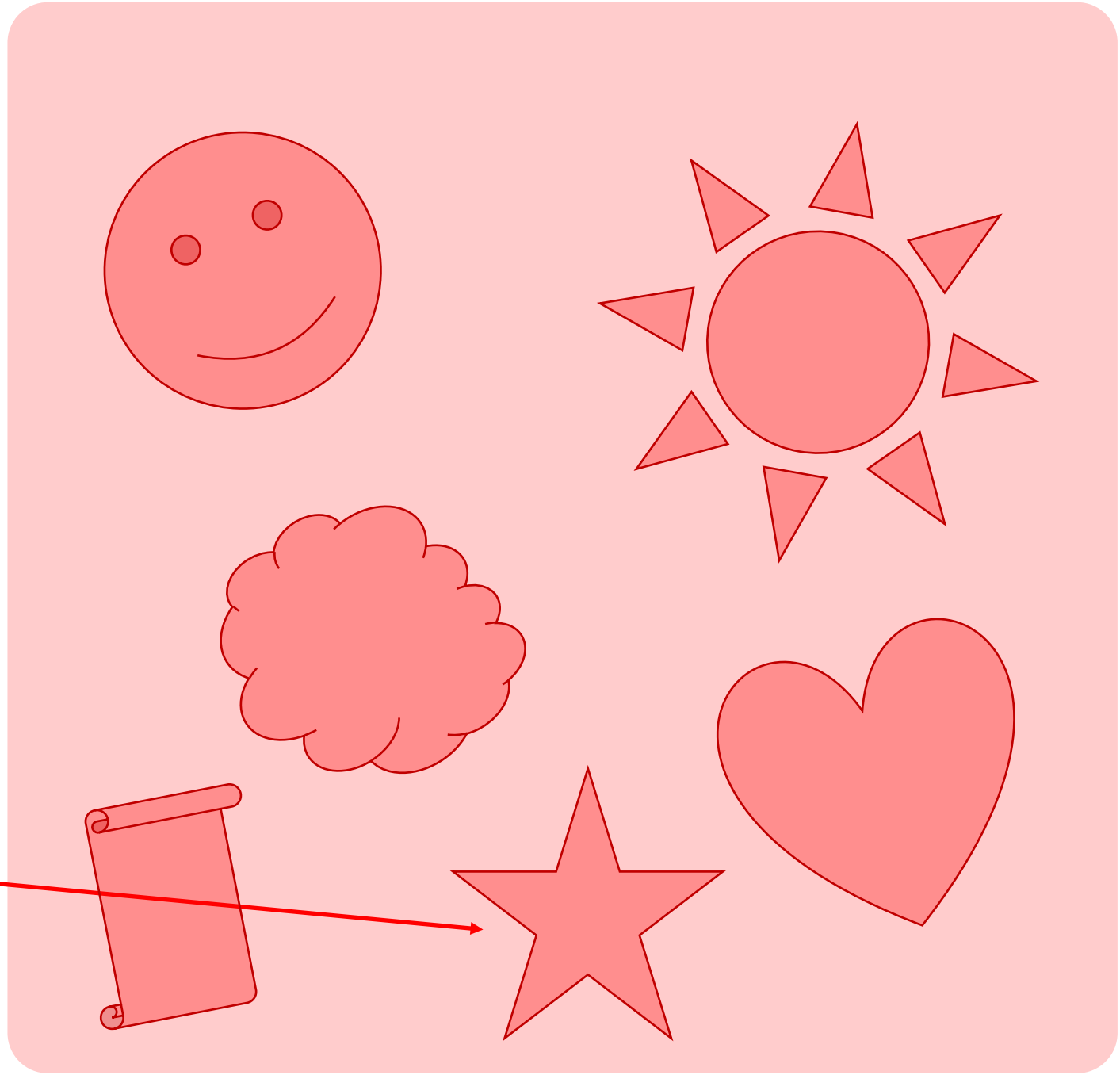
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	[]



In a depth-first manner



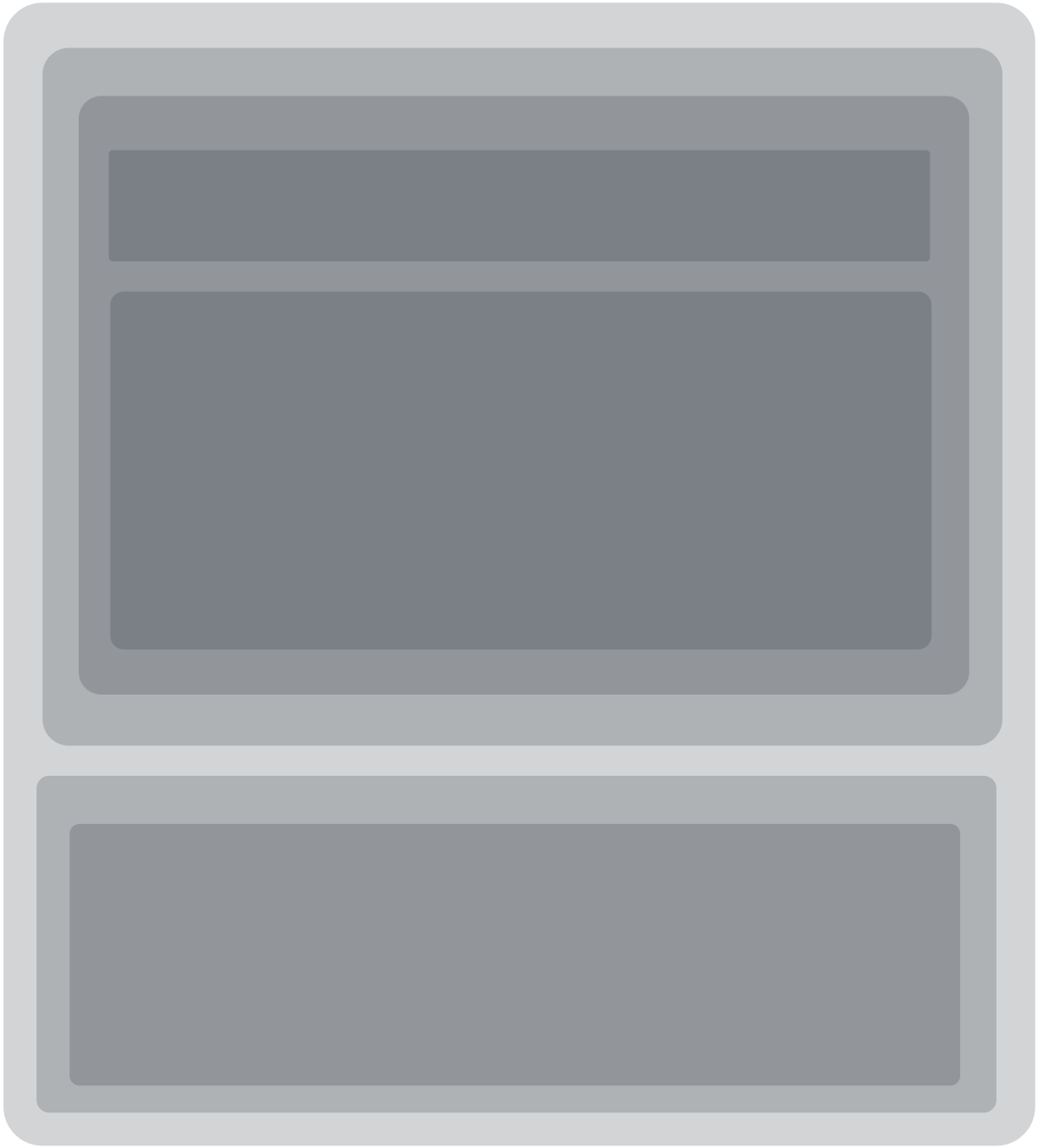
id	_w…
hash	<divclass="wr-…
dTarget	(pointer)
vTarget	(pointer)
parent	'_w…'
children	[]



Step 3 – Change Detection

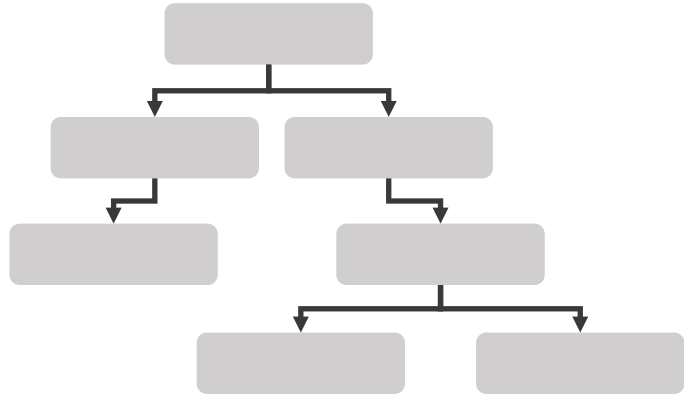
On every heartbeat (50ms), scan for changes in DOM elements

Case - No change

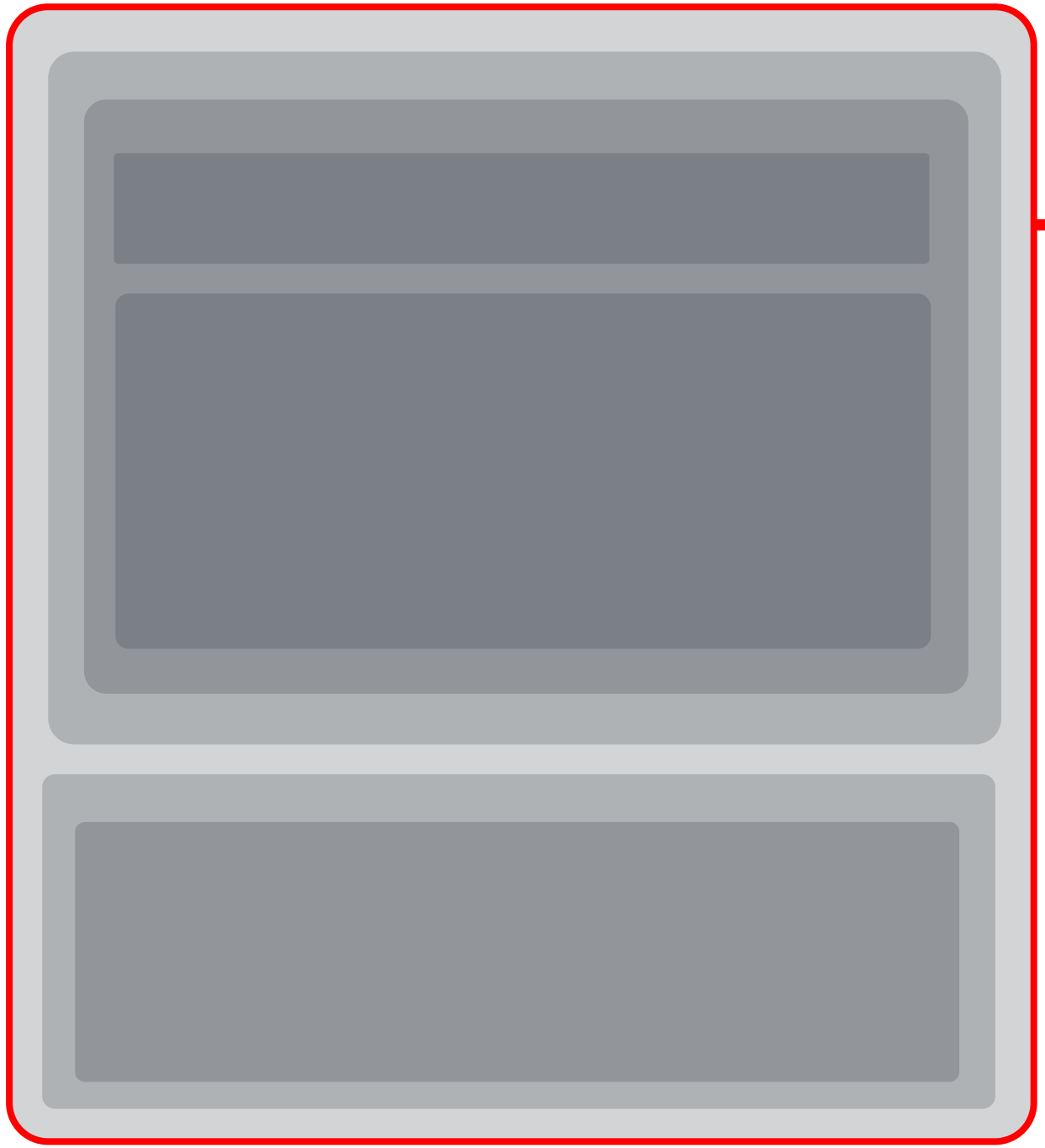


DOM elements

Representation of DOM elements stored in the register



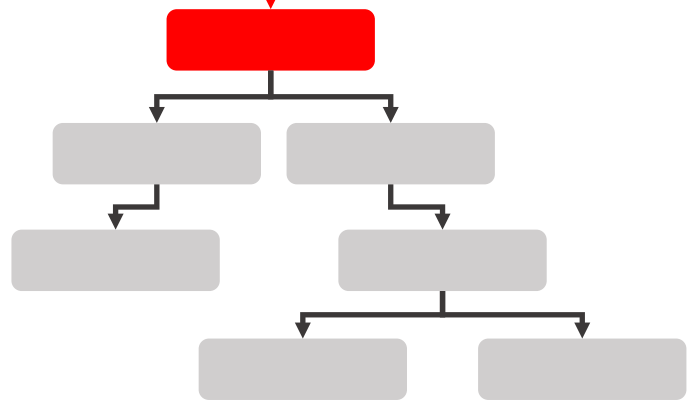
Case - No change



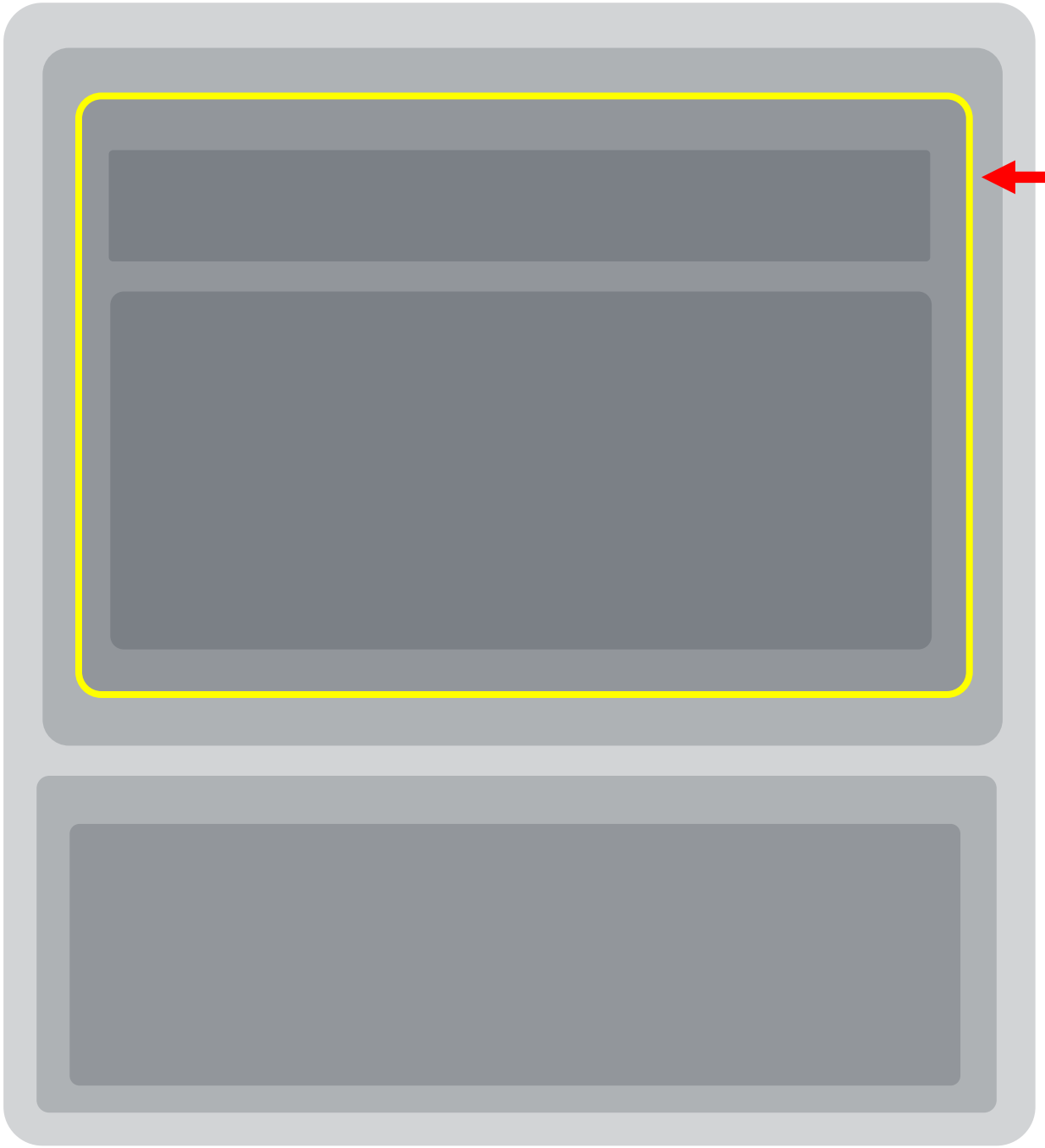
Hash → Compare Hash

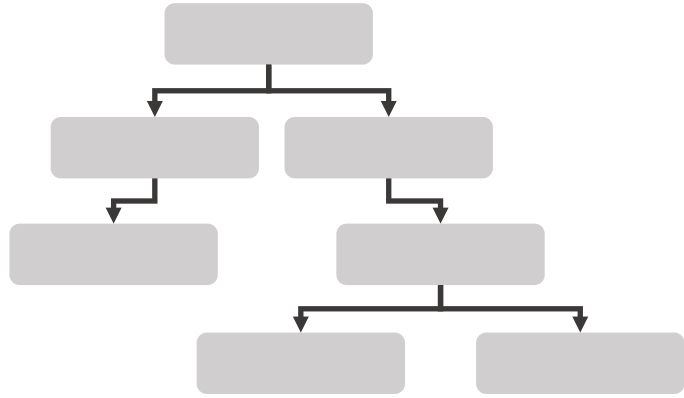
Same - No change in this element or its children.
DON'T STEP IN

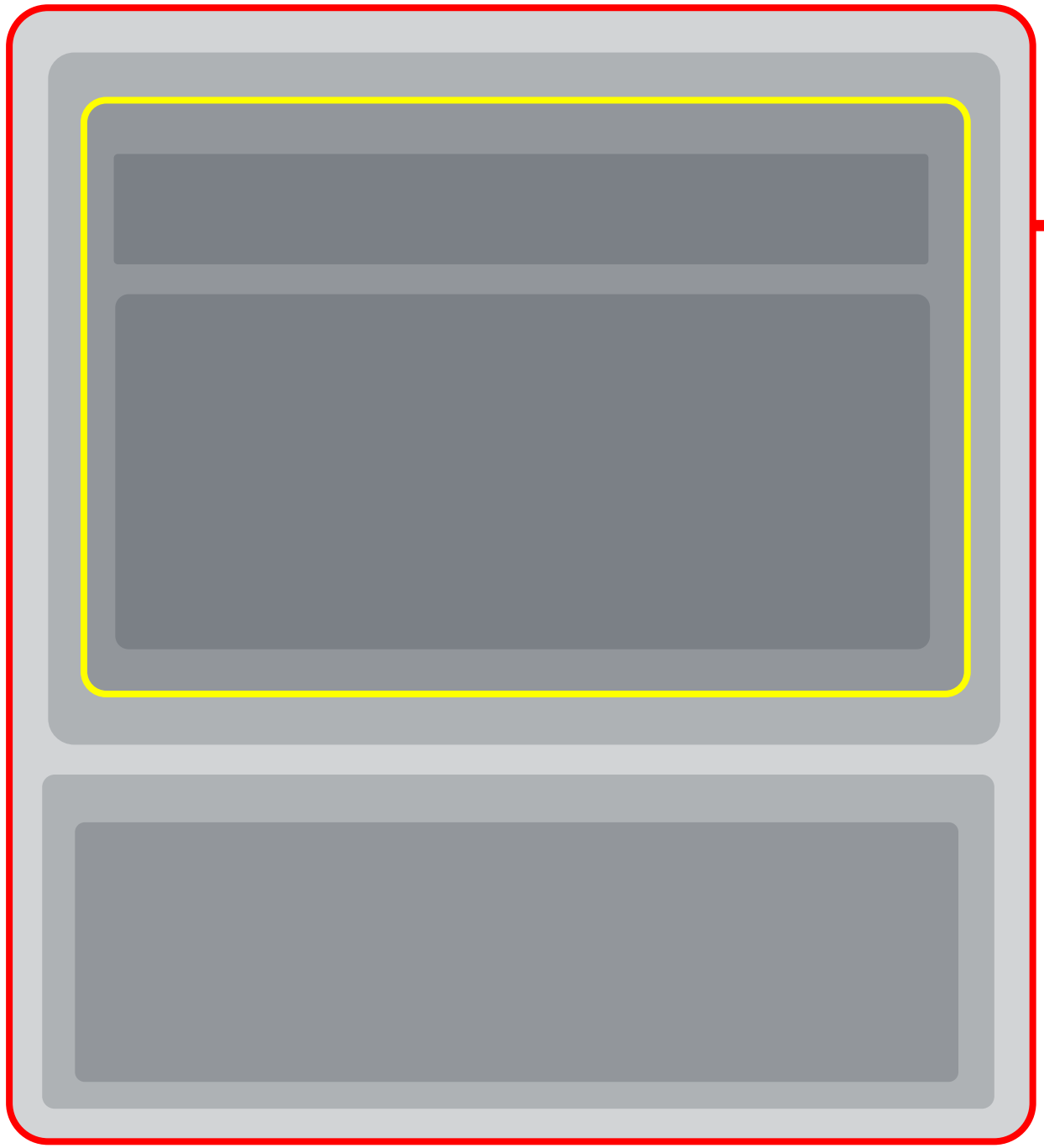
Done



Case - Change



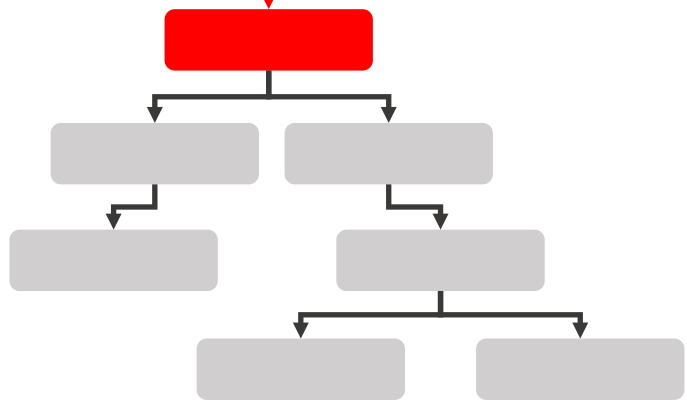




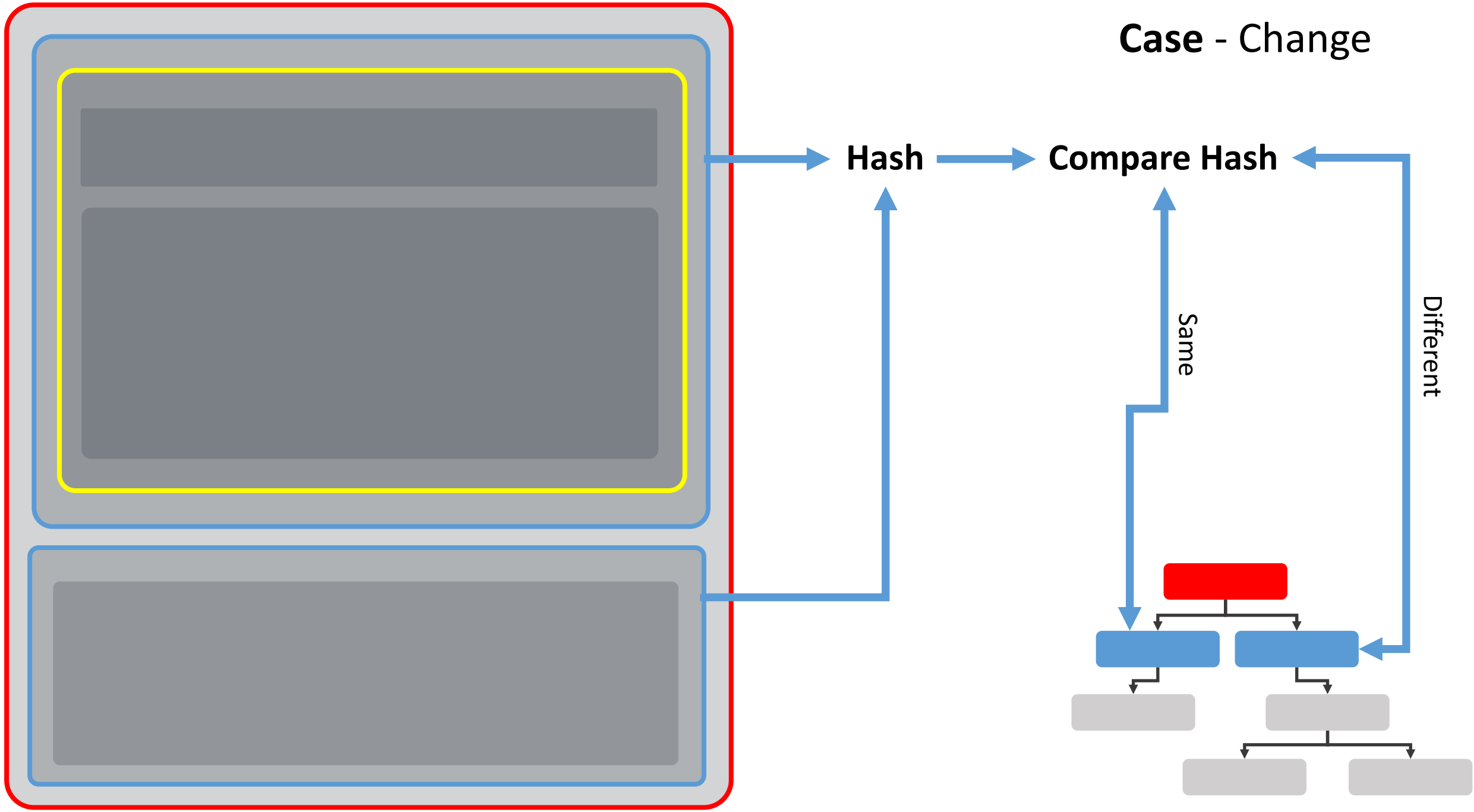
Case - Change

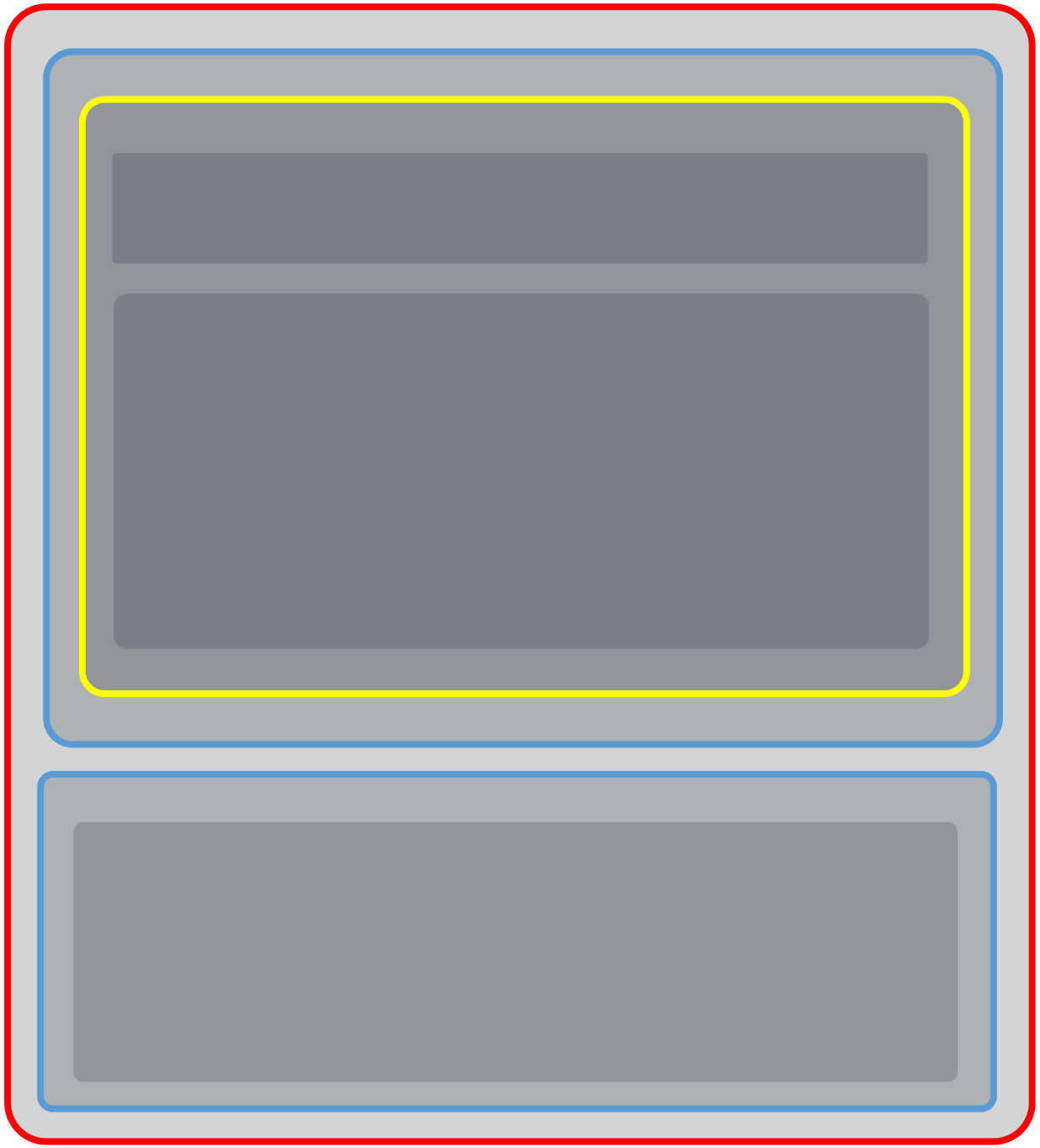
Hash → **Compare Hash**

Different – Either this, or one of its children changed



Case - Change

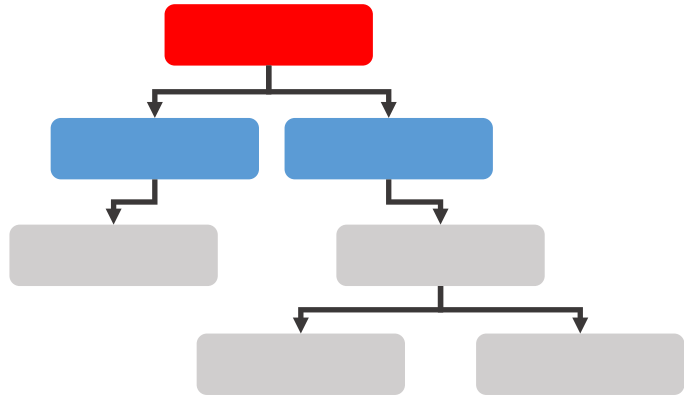




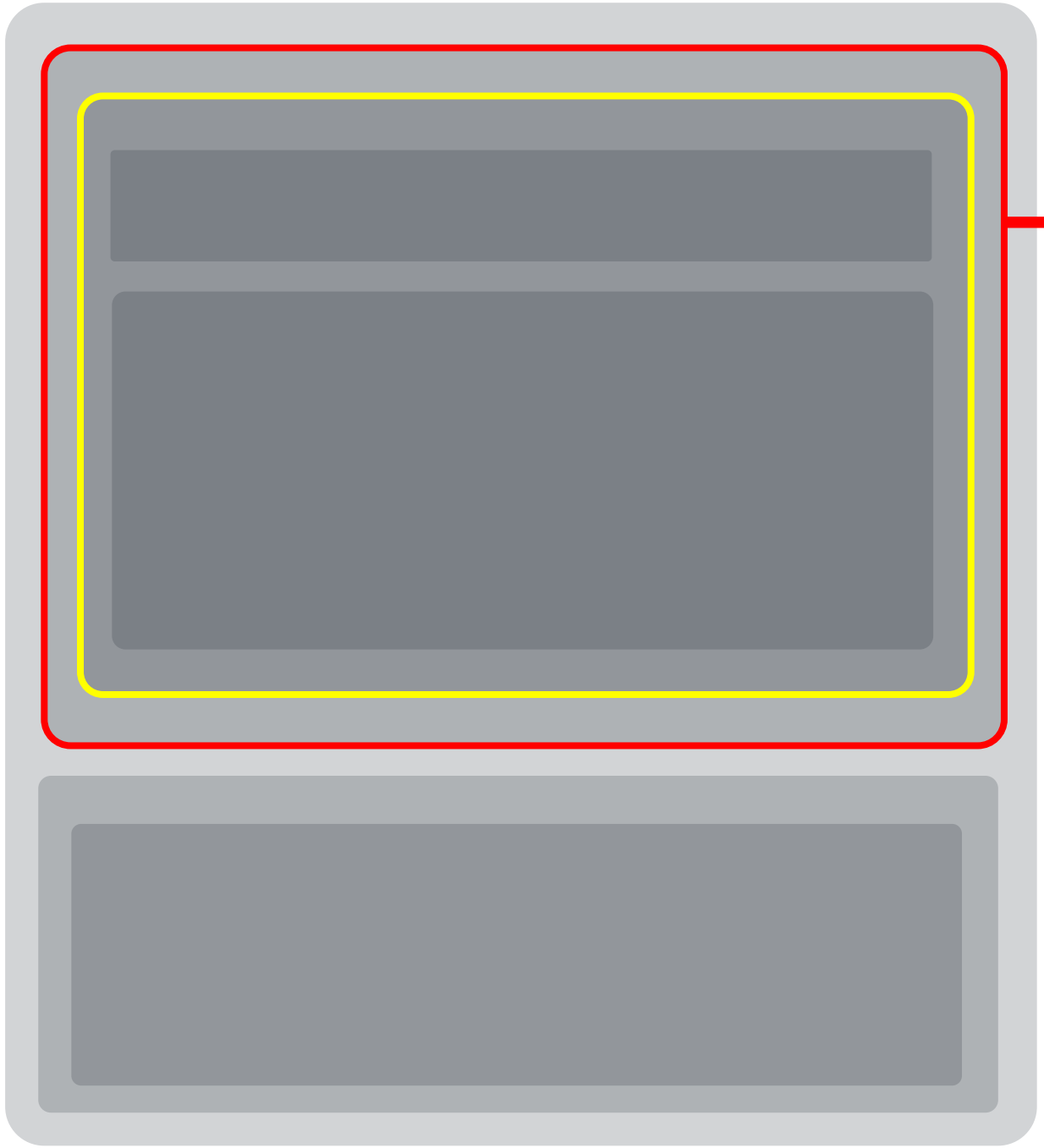
Case - Change

Compare Hash

Only one children has change
STEP IN (to the changed children)

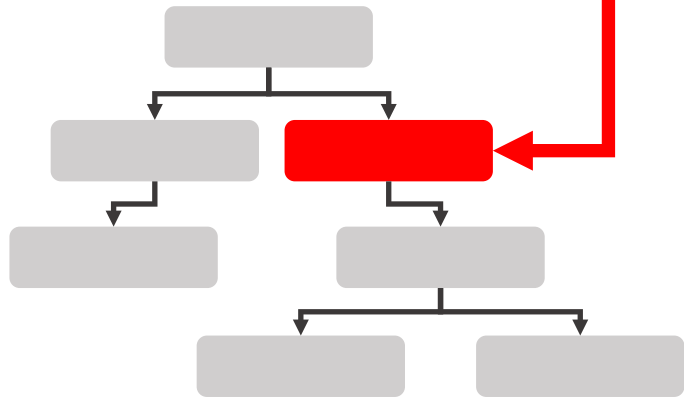


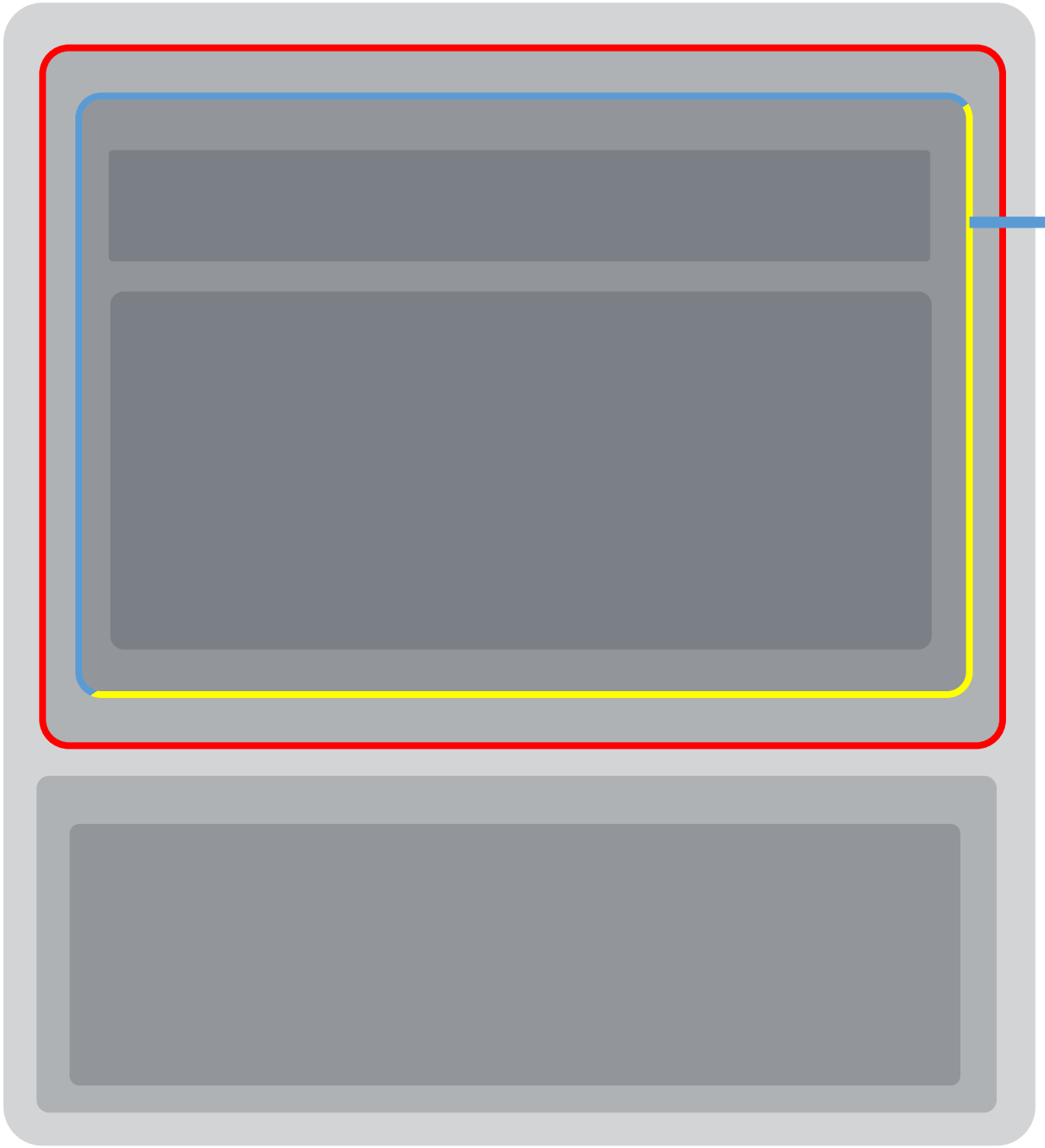
Case - Change



Hash → **Compare Hash**

Different – Either this, or one of its children changed



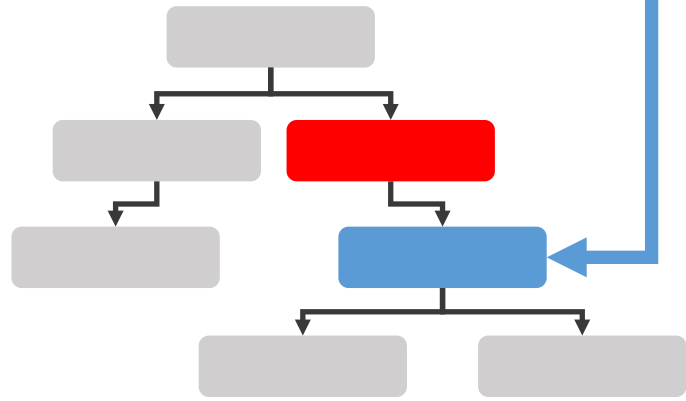


Case - Change

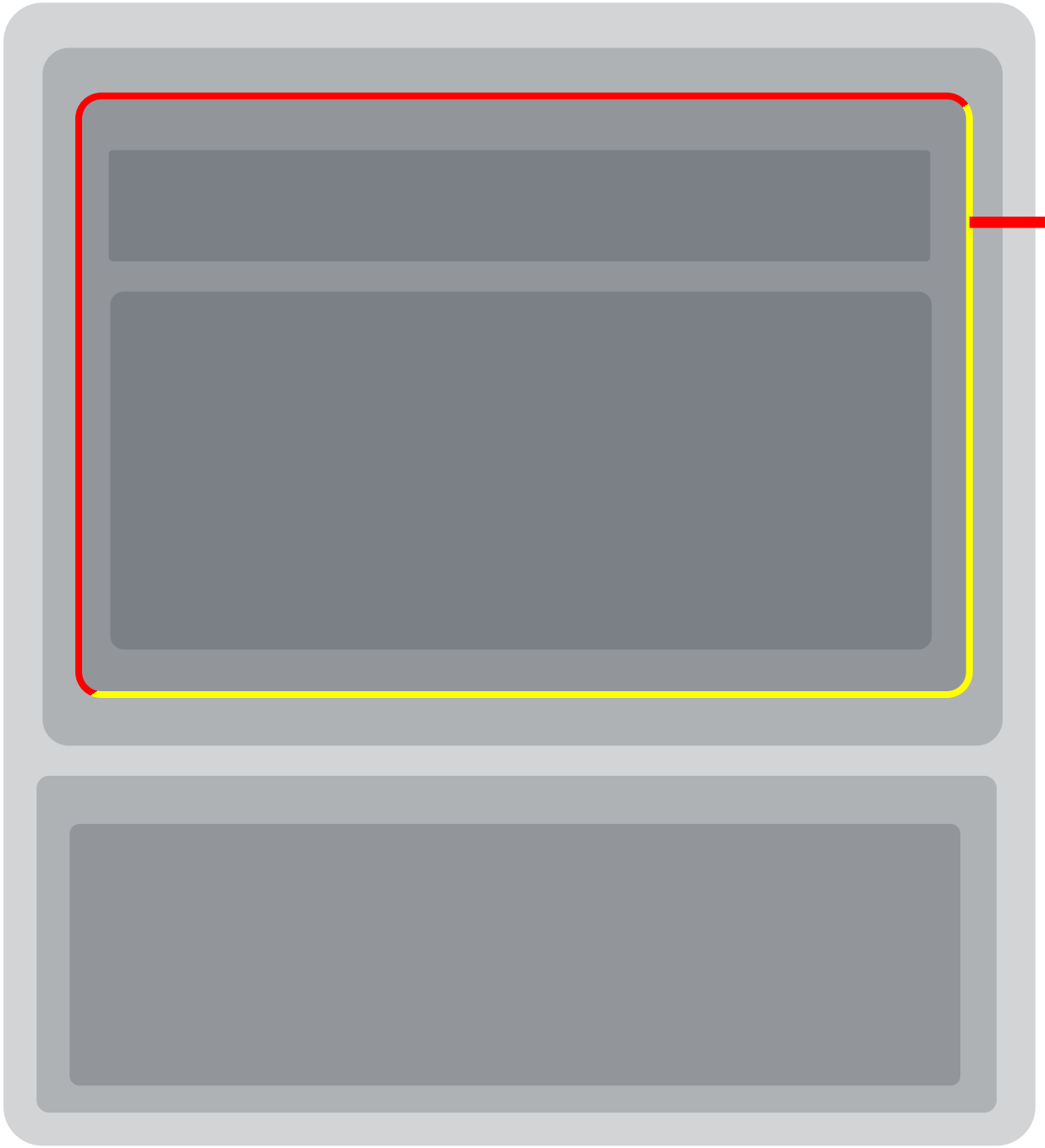
Hash → **Compare Hash**

Children has change.

STEP IN

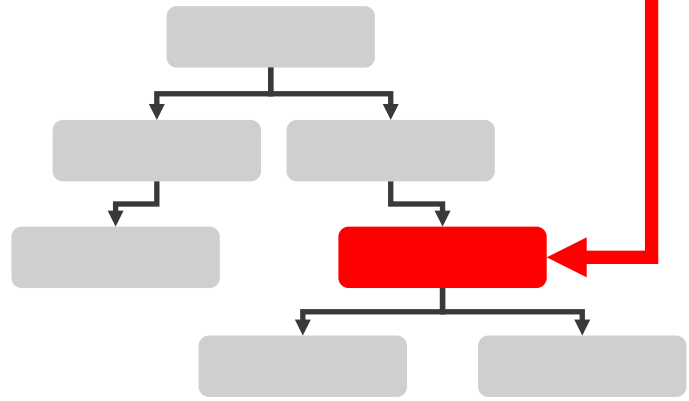


Case - Change

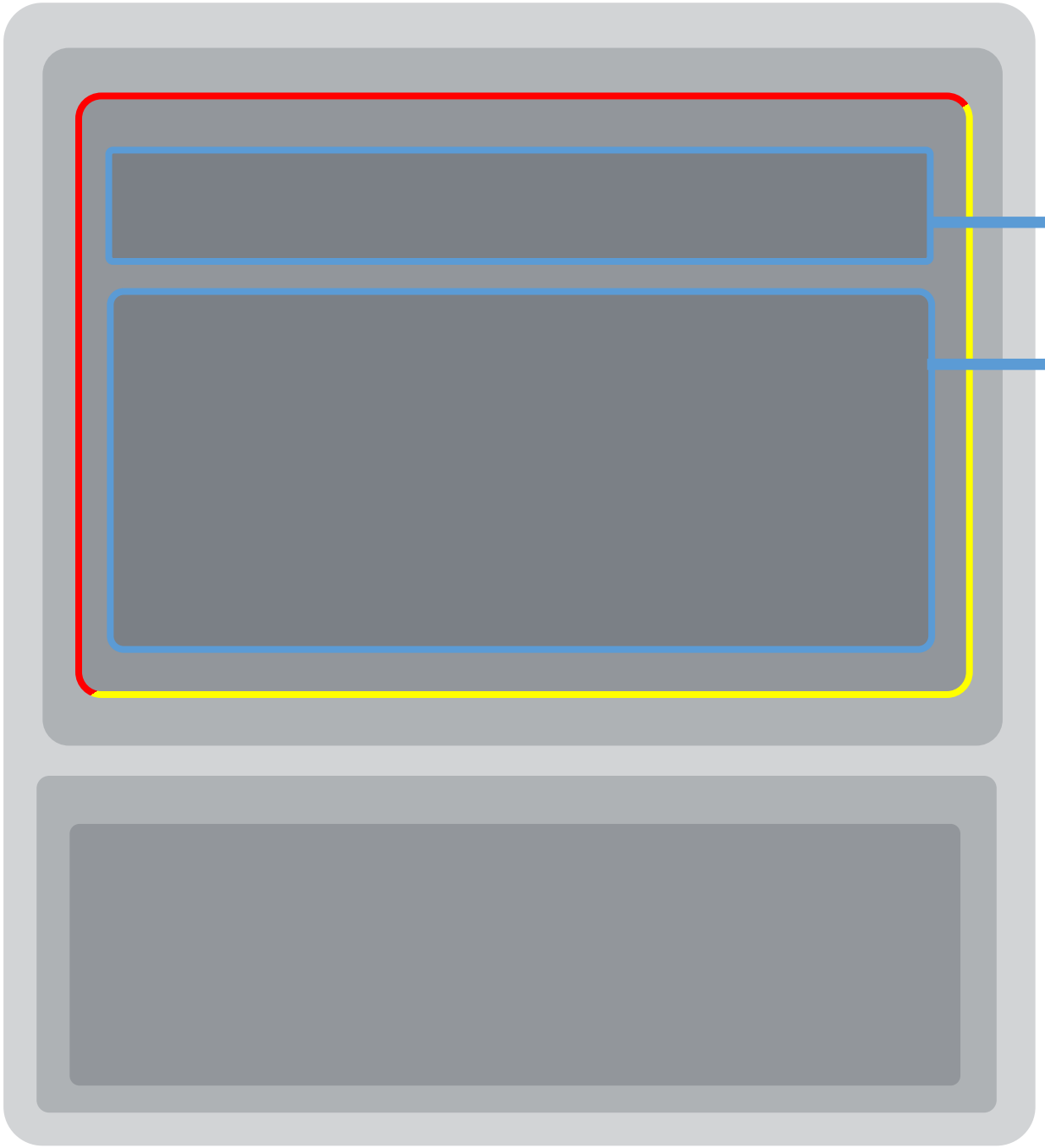


Hash → **Compare Hash**

Different – Either this, or one of its children changed



Case - Change

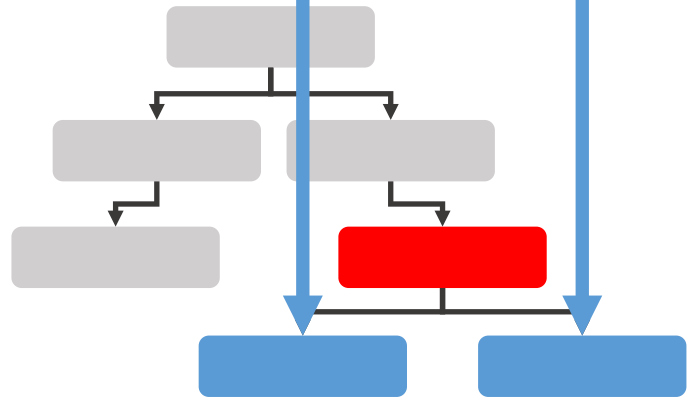


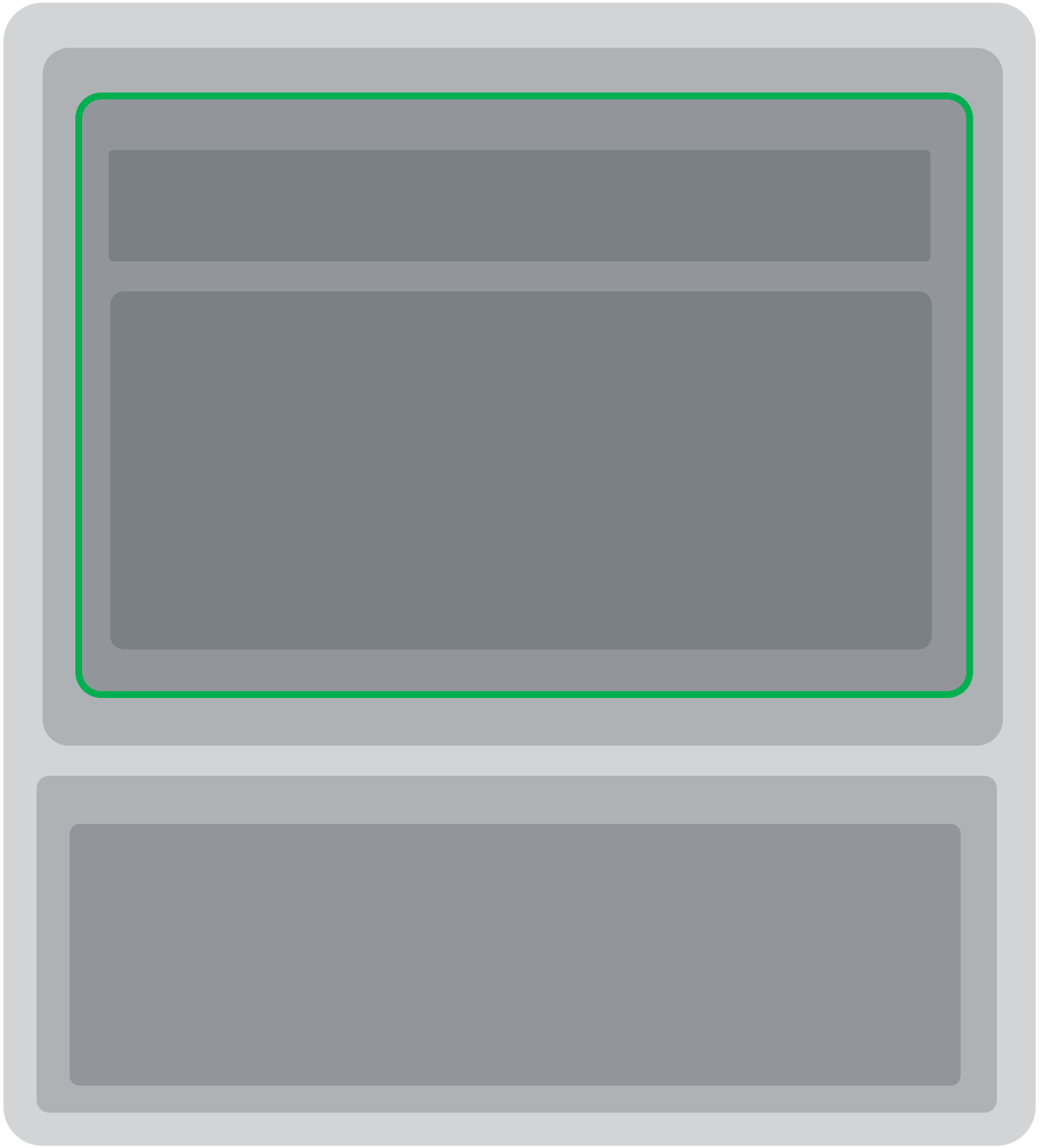
Hash

Hash

Compare Hash

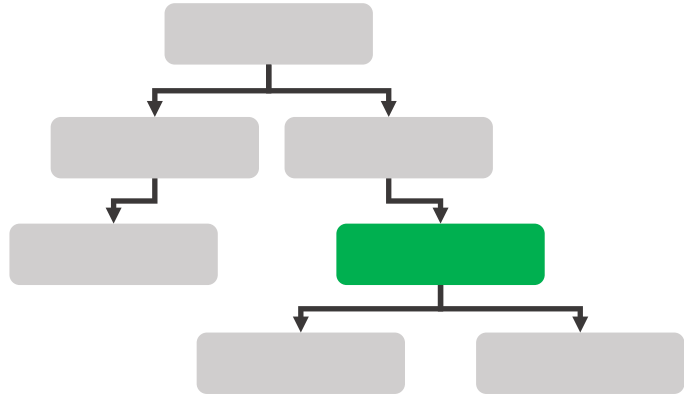
Same – Parent element has a change, but its children don’t.

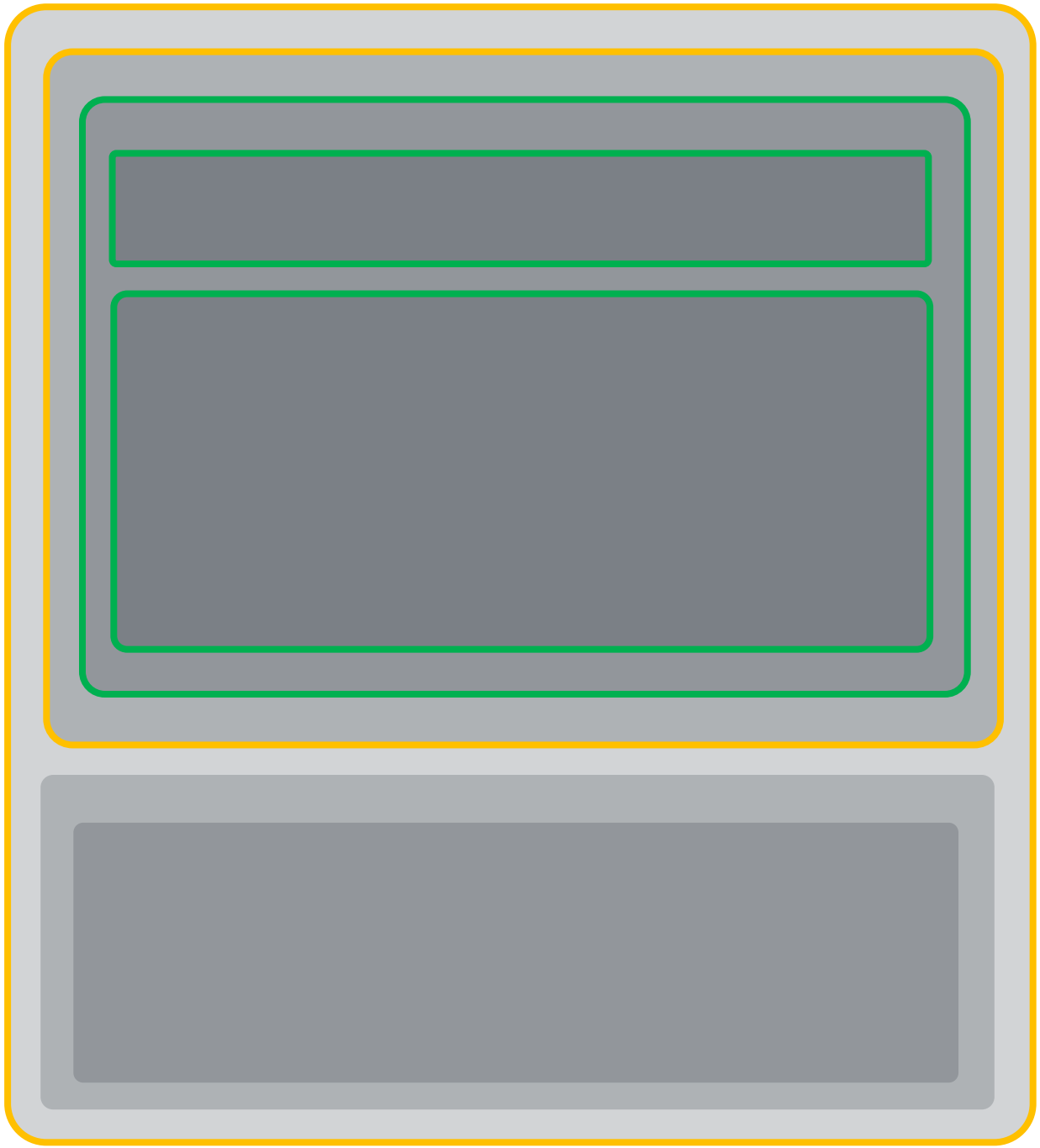




Case - Change

We found the DOM element that has changed





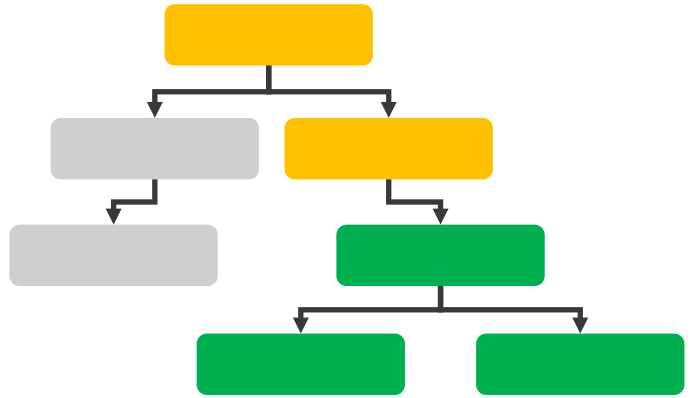
Case - Change

RE-COMPILE THE WAY DOWN

compile / recompile the changed element and all its children

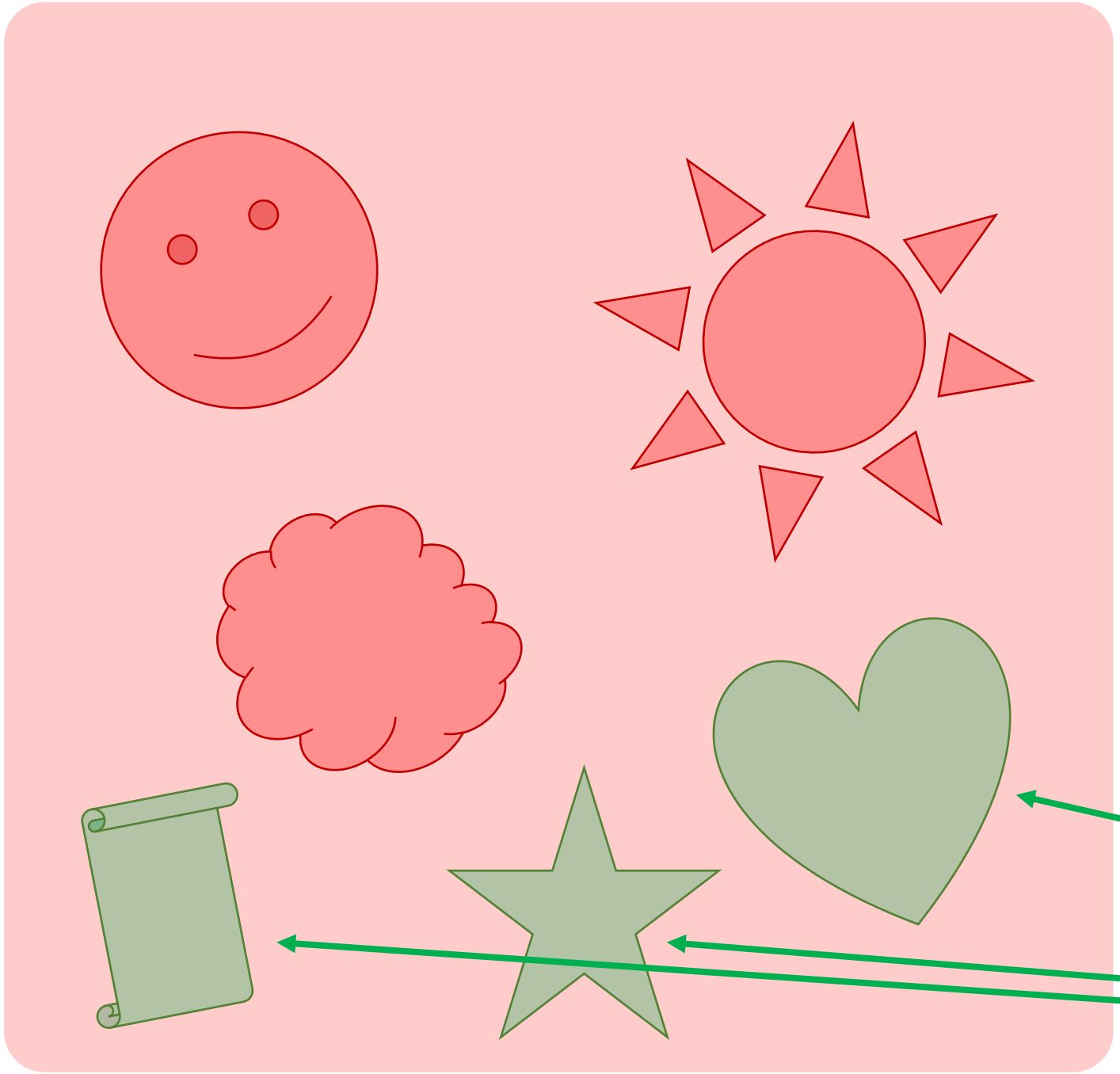
RE-HASH THE WAY UP

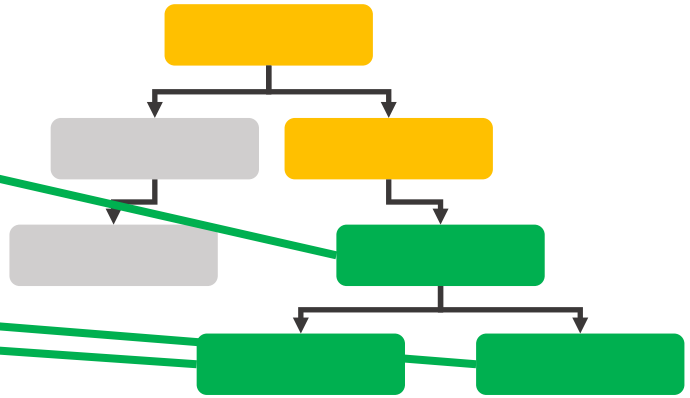
rehash the elements parents all the way up (parent hash values are also changed)



Case - Change

All the elements that are affected by the change gets redrawn.





Wirtual

DEMO

wirtual.io