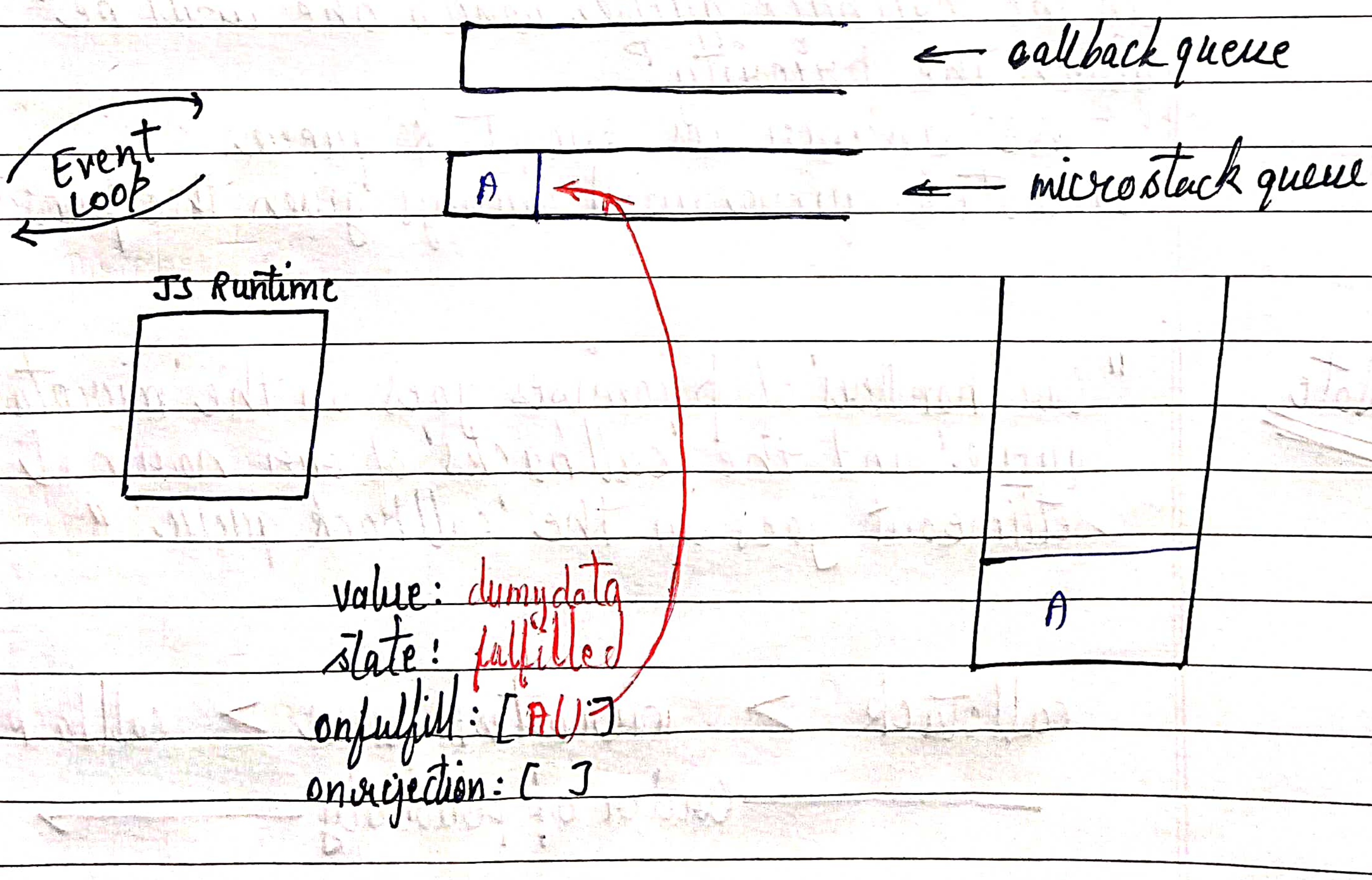
The moment the callstack is empty and global code is done, then the event loop will start taking the callback from the microtask queue and executed.

" At any point of time if event loop has a choice to pick from microtask queue or call back quick (macrotask queue) then it always give preference to microtask queue. "

⇒ When the promise will be fullfilled, the function A() (fulfillment hander) will go in microstack queue.



" The only duty of promise object is to register your handlers (fulfillment or rejection). "

Date / /

??? Does the promise.then registers handlers in the microstack queue.

No, the promise.then registers the handlers in the onfulfillment ~~or~~ reject and the onreject array.

When the 'state' of the promise changes, then based on the state either the callbacks from 'onfulfillment' goes to the microstask queue 'or' 'onreject' goes to the microstack queue.

??? If we have one handeer waiting in the 'microtask queue' and one handler waiting in the 'callback queue' which one will be given the priority?

The answer is, Microtask queue.
Microtask queue will always given the priority.

Note "Our 'handlers' of promises goes in the 'microtask queue' and the 'callbacks' of our normal settimeout goes in the 'callback queue.'"

callstack > microstask queue > callback queue

$\xrightarrow{\text{Order of priority}}$

Date / /

→ Inversion of Control :-

The control of the our 'callback', we are giving to the other function. Now the other function is deciding ~~went~~ + when to call our callback, where to call our callback, whether to call callback or not, how many times to call our callback.

⇒ '.then()' also returns a promise object.

⟶ returns a new promise object