24. DEBOUNCING & THROTTLING

Agenda:

(event optimization techniques)

- Debouncing ⟶ What ?
- Throttling ⟶ Custom implementation
- async & defer   use in script

(SEARCH)

FAST

(Delay)

D

mobile
↑22458   m   2/6
$API   mobile

E
B
O

laptop
↑2468.   Slow

                1   lopt   5/6
                la   laptop
                lap

U
N
C
H

iphon   FAST→ Slow
↑2815
                i   4/6
                ipho   iphon
                        iphone

I
N
G

\# or Scroll, Social media → new content.

'Scroll' → event attached.

→ ↓↓ ⟹ fetch data   Sluggish

                                    Perf. issues

throttling

↳ Calling events in a particular
   time interval.

DEBOUNCE :

Q&A

Debouncing

Q1, Q2 Q3, Q4 , rapidly

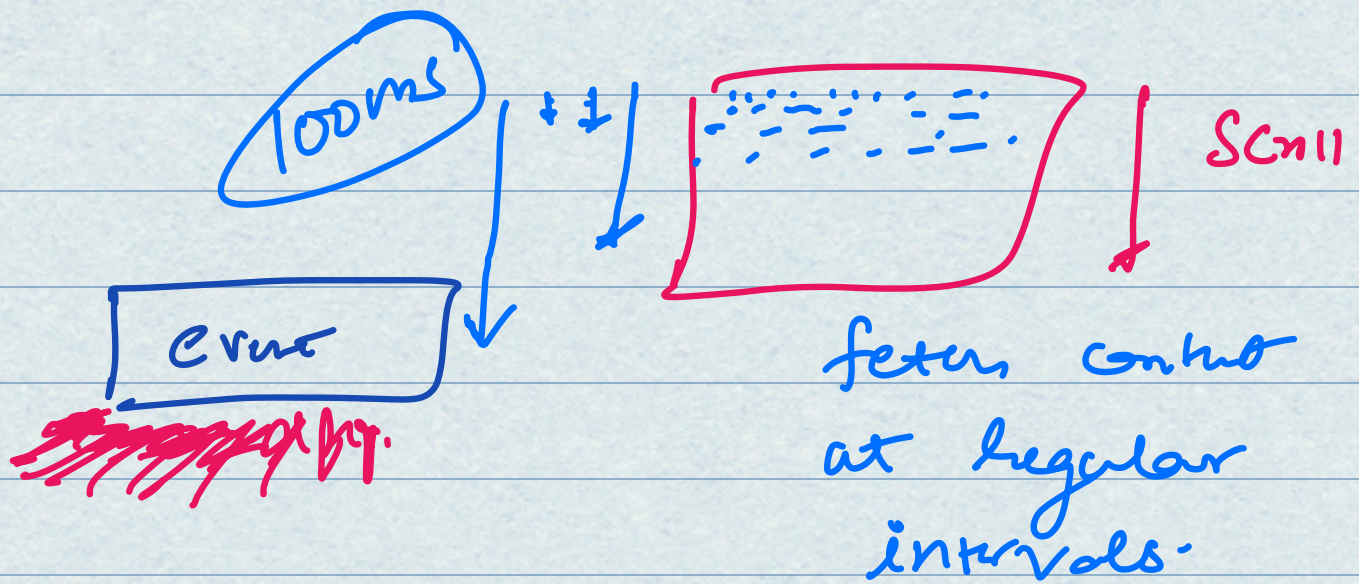(w/o pause)

1000ms

P → —— 200ms ✓

P → → 800ms

→ 500n
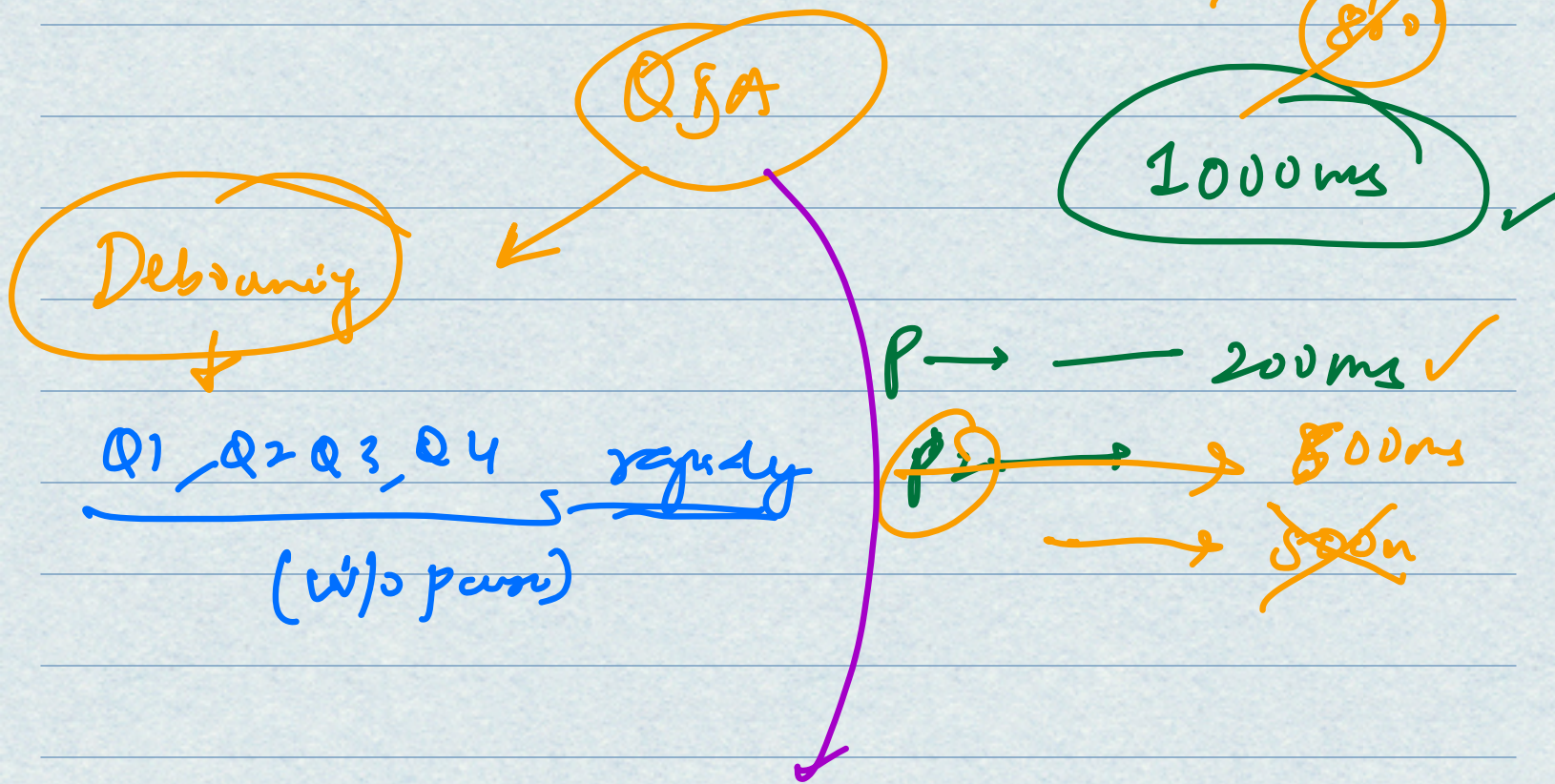
Throttling

↳ Time Constrained Speaker

1 q / every minute.

(no matter how many
question asked in middle).

1mn → 10q ⟹ Answer fast,
ignore rest

100ms

event

~~bouncing~~

Scroll

fetch control
at regular
intervals.

(input scroll)
(Delay)

(Scroll).

Debouncing

Throttle

- Waits for a pause
in activity to
act ONCE.
focusing on final
event

- Limits the frequency
of action, ensuring
it only happens
in regular intervals,
no matter how many
times the event gets
called.

o You wait for a
bit after the last
action before you

• Like setting the timer
and only taking action

take your turn

When timer goesoff,
no matter whatever
happens in between.

Uses of debounce

1. Improves UX.
2. Reduces workload on server.

Async & defer

```
<Script src = `main.js' > </script>
```

(Browser) ⊂ HTML / JS.

✱ Whenever browser encounters a script tag, stops parsing HTML,
- requests the script file   ⎤ User
- Wait for it to download   ⎥ will be
- Execut the script file   ⎦ ideal

aftr that   Continue parsing the HTML.

Async & defer

↓
✱   `<Script src =`main.js' async />`

Async

     `<Script src =`main.js' defer />`

Async

HTML → Continue parsing HTML → Continue parsing.

DOWNLOADING → Execution of JS

< Script async >

Defer

HTML → Continue parsing HTML → HTML finish → execution of JS.

DOWNLOAD

< Script defer >

Normal

HTML → Parse HTML

< Script/ > → DOWNLOAD JS → EXECUTE JS

#
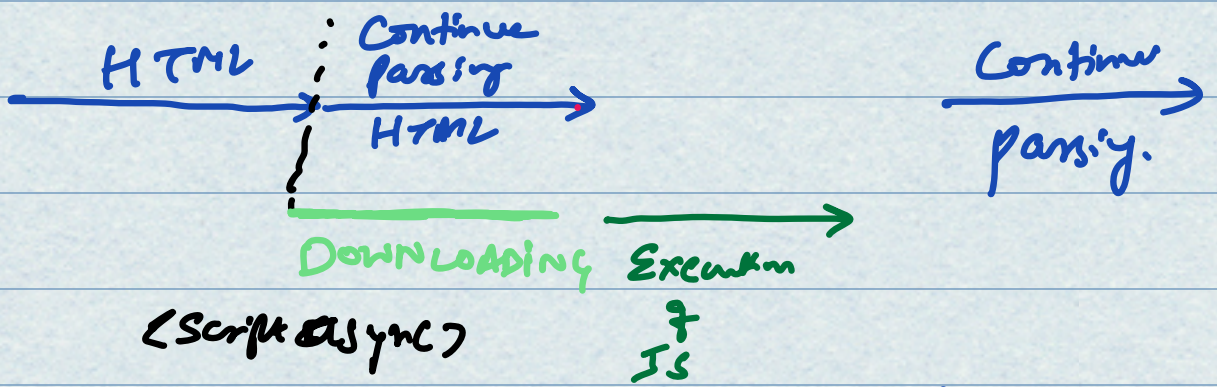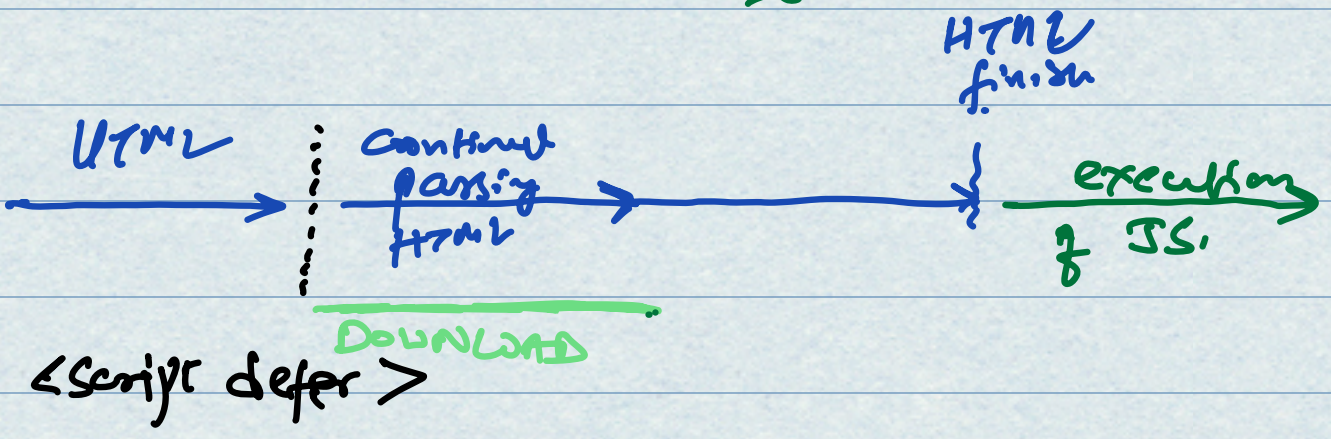
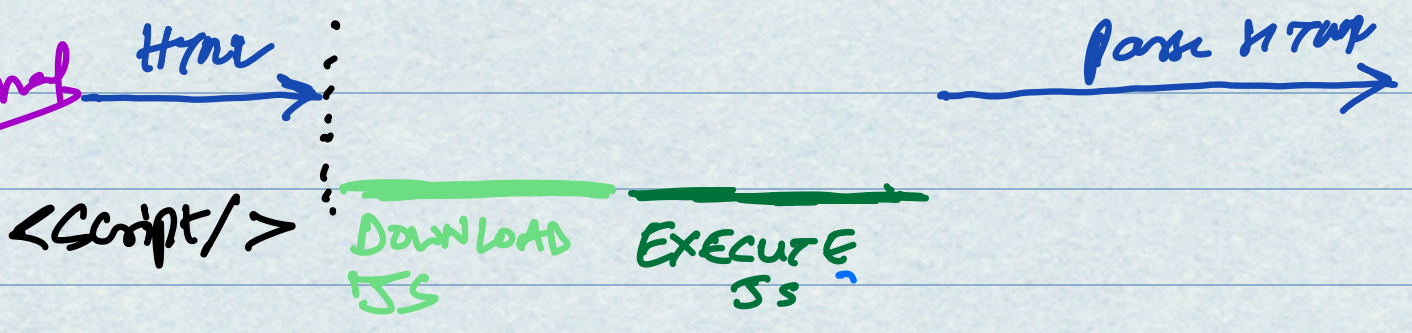**Async ⟹**

- For those scripts that do not depend on other scripts.
- No guarantee of the Order

lib.js   (Needed by main.js to run)
main.js → Depends on lib.js
analytics.js →   Independent.

A: 1000ml

< Script src = 'lib.js' defer / >
< Script src = 'main.js' defer / >

order is granted

< Script src = 'analytics.js' async / >

{ < Script src = "s1.js" async defr / >

✓   defer gets priority