Turwash Notes

19th Feb 2023

DOM + Modern JS - Class 3

→ performance
- measure speed of code
- how to write efficient & performing code
- Event loop.

A standard way to measure how long your code takes to run.

↓

by using

`performance.now()` ← method.

↑ This is very accurate.

```
const time1 = performance.now()

This is your code

const time2 = performance.now()

console.log(time2 - time1);
```

When we add paragraph in DOM,

2 things happened

— Reflow (calculations for element dimension & positioning etc)

— Repaint (to show element pixel by pixel on your Screen)

good practice is → less Reflow & Repaint repetition in your Doc.

{ Reflow takes more time
& Repaint takes time but less than Reflow }

Best practice

↳ Document fragment

↓

Light-weight document object, no reflow & repaint when we add element to it, then we will add Document fragment to Document Then it will do one Reflow & Repaint ✓

→ The Call Stack :-

Single-threading :- One command at a time.
JS is Single-threaded language.

✓ processing of one command at a time.

Single-threaded ~~synchronous~~ language :-
executes line by line.
ignores function but when function is called goes inside function then again line by line.

→ run to complition nature of ~~code~~ language.
→ JS does not execute multiple line or multiple function at a time

Call Stack is a list that tracks or stores the Functions :-

Function invoked then added to list Function ends then removed.

either it is executed it is removed from callstack

```
function a() {
    console.log('Hi')
    b();
}
function b() {
    console.log('Hello');
}
a();
```

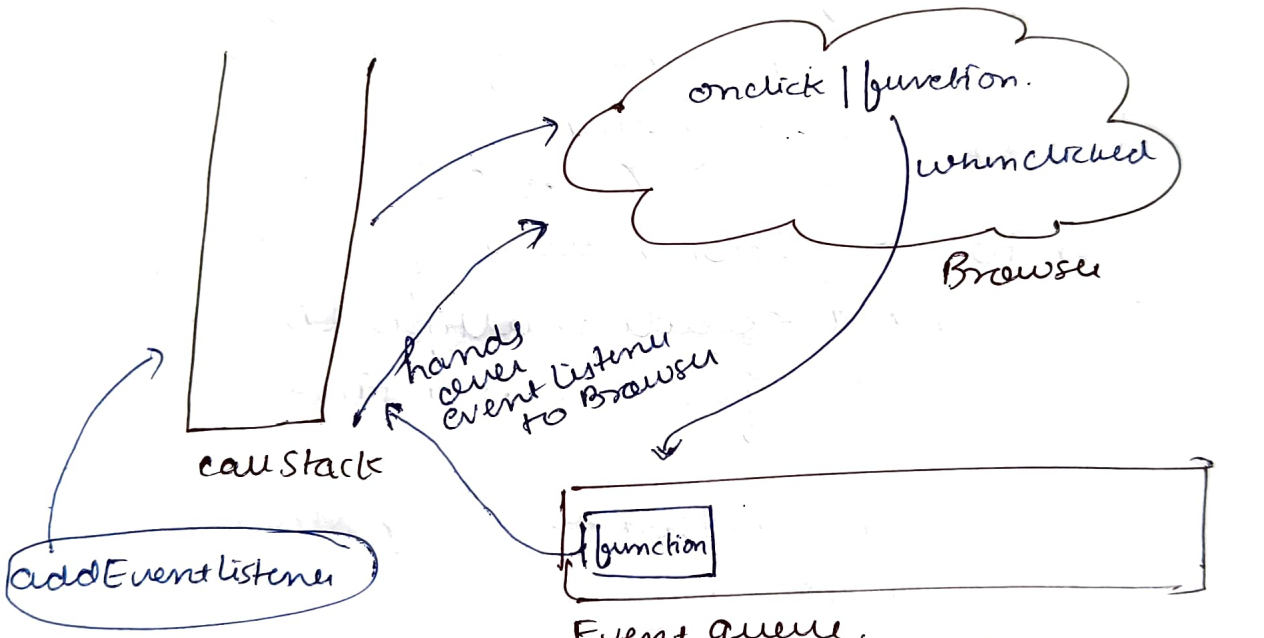
9wp

Event Loop :-

Synchronous → occuring at a same time.

Event-listener is → Asynchronous.
because it works when action is performed.

Event loop :-



Code :-

```
① clg ('Hi');
② element.addEventListener
    ('click' function() {
    clg ('123');
  });
③ clg ('Hello');
```

② only will run when clicked else ③ will be executed

addEventListener loop EXPLAINED

Code :-

```
1) clg ('ABCD')
2) element.addEventListener ('click', function () {
                clg ('1234');
   })
3) clg ('XYZ').
```

Now, in call stack :-

→ entry of (1) and 'ABCD' is printed & (1) is executed

→ Then entry of (2) event listener. but it is when clicked then function, so, callstack hands over event listener to browser & move to (3). clg ('XYZ')

Now, when clicked

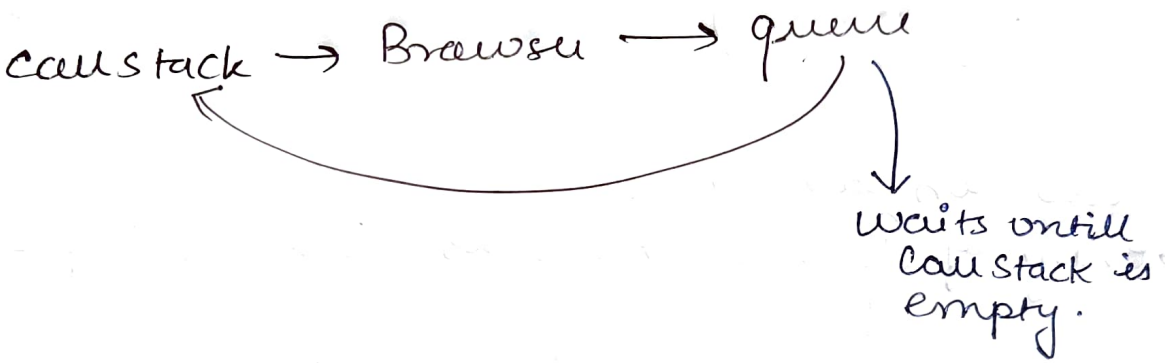
Browser will hand over the function to queue, but the queue will only execute the function, when callstack is empty. if call stack is working on any function, queue will hold the event listener function. when its empty, it is executed Finally.

This loop is called Event Loop

A Bit more :-

1) Asynch code → depends on JS Event loop

2) Any Asynch code is handled by browser.



SetTimeOut()

```
setTimeout( function () {
    clg ('Hi');
}, 4000);
```

↑ waits For 4000ms or 4 sec before execution.

But no guarantee 4sec is minimum time can take more., waits for call Stack to be empty.

↑ because this is also Async Code

SetTimeout & parameter

(function(), Time)

when you want to defer Something, you can ~~you~~ use SetTimeOut.

SetTimeout, 0

↓

does not mean to run immediately.
it will Still do the Event Loop.
