

# Web Dev

Foundation



HTML



CSS



Selectors, Flex, Grid

Bootstrap, Break points



CSS challenges

Tailwind



Challenges



JS



Foundation



OOPs



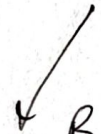
DOM  
BOM



Advanced  
Topics



JS  
Projects



Backend

(Node.js + Projects)



DB



SQL + Postgres



Mega Project

node + express + mongo

Frontend

React + projects

Learn Anytime

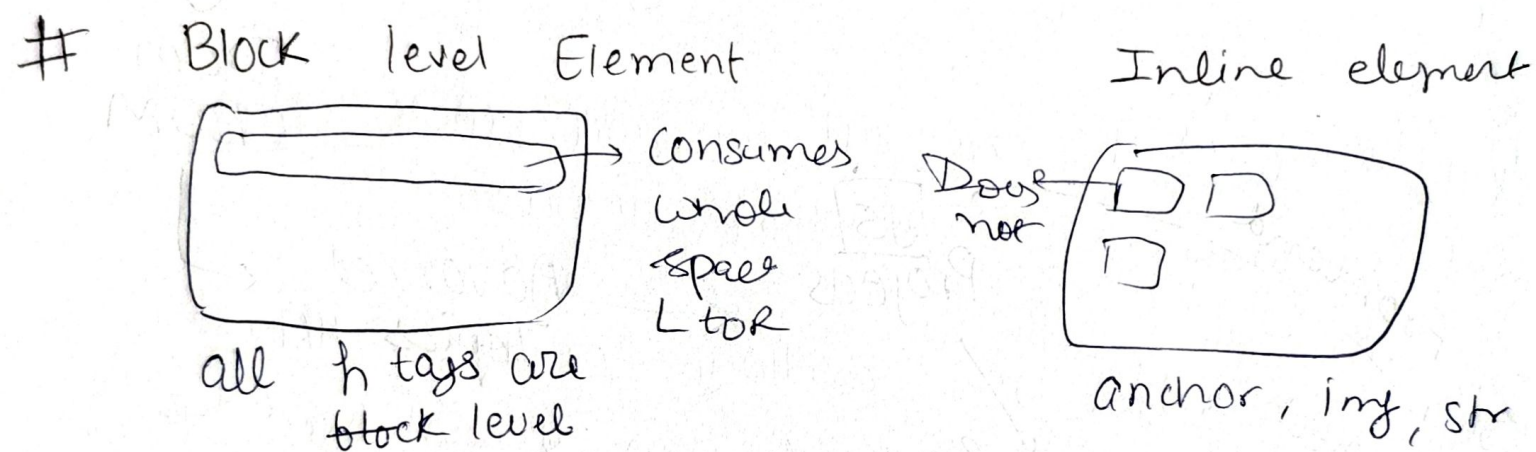
# Deploy (everything)

Full stack Auth Next.js

QNA System Full stack

- 1) Microsystems - Queue Systems } Proj.
- 2) Sockets - WebRTC
- 3) Docker

- # ① Server is not a big machine, it's just a soft which serves
- ② Server interacts with DB.
- ③ API is just function calls from frontend to backend. And UI.
- ④ Anything in ~~the~~ tags = attributes like ref, width = ... etc.
- Tag + Content = Element



- # SELECTORS to get rid of browser applied styling
- ① Universal (applies to all)
- ```
* { margin: 0; }
```
- ② Type selector
- ```
h1 { applies — }
```
- ③ Class selector
- ```
use . classname { }
```
- ④ ID selector
- ```
# idname { }
```
- ⑤ Attribute selector
- ```
input [type = "password"] { border = rounded. }
```
- NOTE
- ① Dom: Data rep. of obj
- ↓
- It is not the backbone of HTML, Rather a "repr of HTML" structure that browser creates to allow for dynamic interaction.



# Border Box → when u define  $H=300$   $W=300$ , it is absolutely fixed, if u add padding, it will be included in that.

Content Box - padding, mar. etc are not included, so real size of box is larger

# :: after and :: before

Syntax: 

```
• classname::after {  
  content: " ";  
}
```

→ dynamically generates HTML after the class (inline)

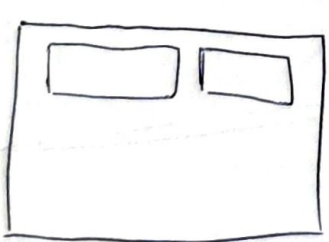
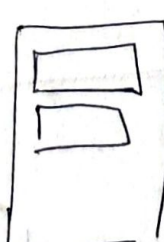
# Position: absolute is positioned absolutely but according to the nearest ancestor.


if no ancestor has the property pos: relative then it scales acc. to the entire page

Intv: Ques • When you use absolute positioning it rips apart content from DOM <sup>normal flow</sup> and puts it somewhere else in DOM. where it won't affect other elements.

• ~~Everything is 3D not 2D~~

# Tips for tailwind- ✱ min-h-screen instead of h-lvh

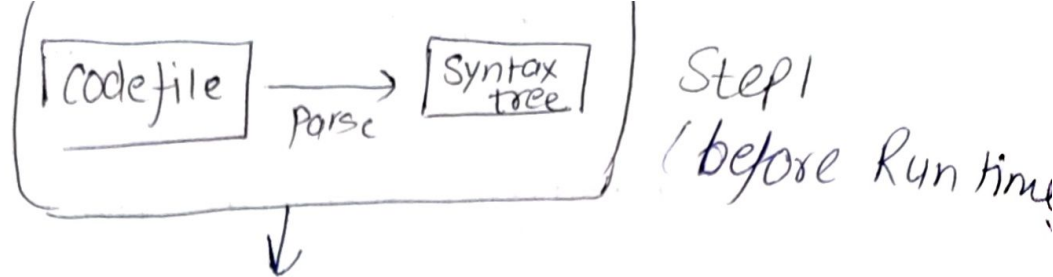
✱  to  grid <sup>sm</sup> ~~sm~~: grid-cols-2

if  wont only then grid <sup>sm</sup>: grid <sup>Both w and h of sm</sup> - (cols-2)

to distribute unequally  
use grid-col-12

and  $\left. \begin{array}{l} \text{grid-span-2} \\ \text{" - " - 10} \end{array} \right\} \text{ } \square \square \text{ etc.}$

JS working -



VIT compiler (Just in time)

Byte code → Machine code

///

# OOPs

Prototype Programming -

- JS is prototype based language
- OOPs here is Fake, it literally compile to prototype

Prototype mechanism through which JS objects inherit features from one another

This, in sense is a Prototype chain

All "built in" methods are called prototype and these are also "prototypes" thus having their own methods.

- First it looks for "methods" in ~~the~~ inside then looks methods of ... eg:-

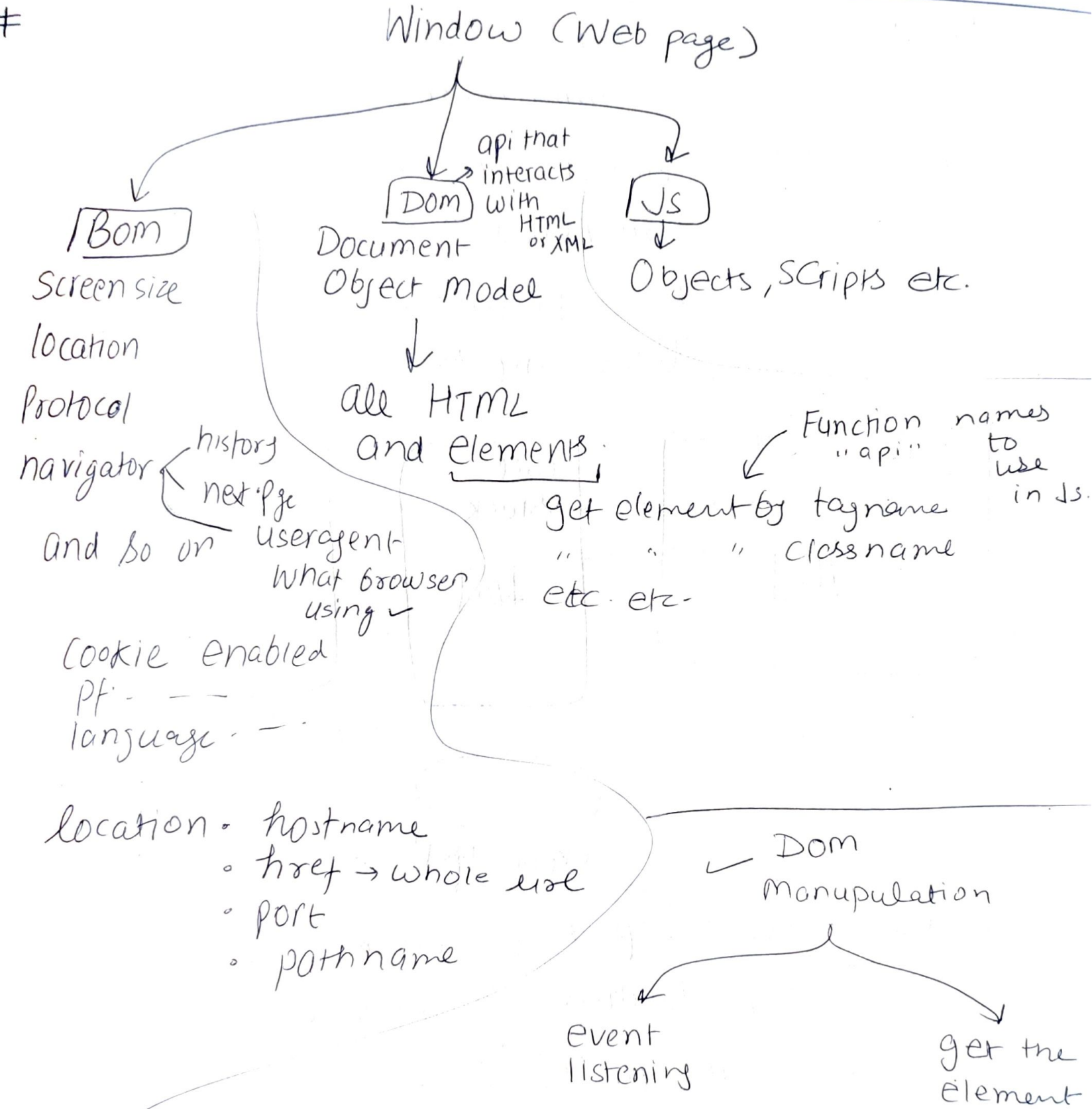
hi.

# Constructor functions

- First name word capital
- Use this. - - - everywhere



Encapsulation - use # behind something that'll make sure it can't be accessed outside the main ~~function~~ class.



### Important Note

document. getElement by ID ("..."). add event listener

("click", ① () ⇒ { console.log(this) } will point to global head  
② function () { "this" } will point

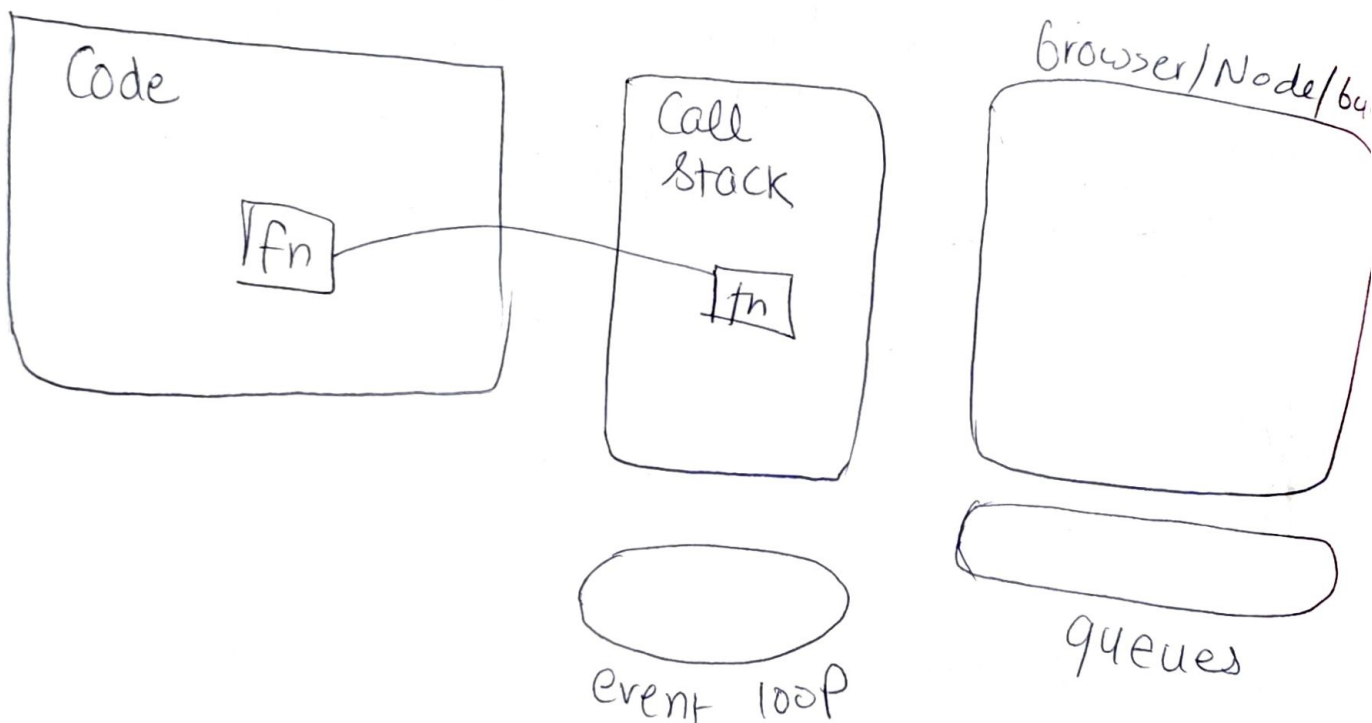
Processed HTML is called Node  
Modelist is not an array it has to be  
Converted into one.

# Async is system Running event loop on  
single thread

Asynchronous

↓  
having pause

Why need? : network calls.  
Write/Read from files  
time functions  
User input.



Normally : Event loop keeps checking if there's  
any function to execute  
(2) if it finds, it executes and  
removes it from the call  
stack.

But for time related fns, they → Stack

Queue ← Browser

But when they are waiting, event loop  
executes stuff written after them!  
Suppose we're fetching some data, that fn is written

## # Closure functions

function inside function can access stuff from this function.

---

## # Promises in JS

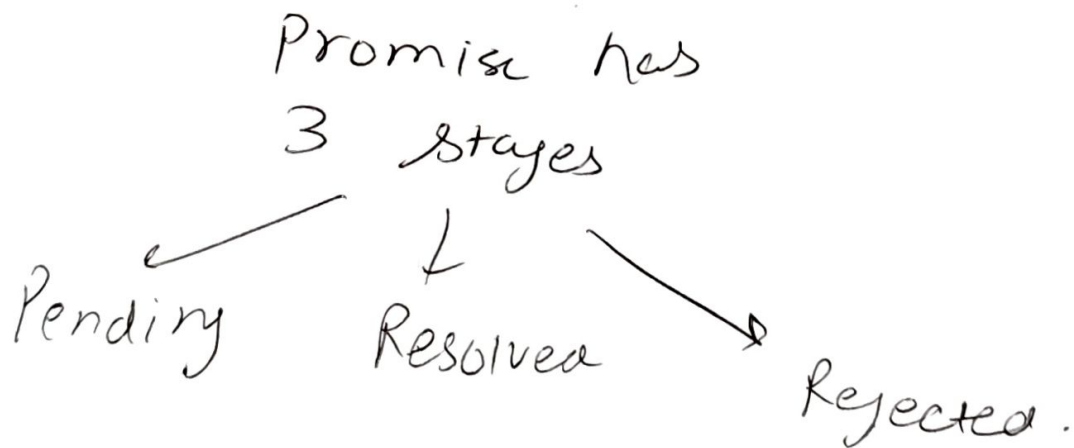
```
func hi() {  
  return new Promise((resolved, reject) => {  
    setTimeout(() => {  
      resolved('hi')  
    }, 1000)  
  })  
}
```

Basically, There are only two outcomes for any promise

- Pass → resolved
- Fail → reject

if function inside promise is fetching data from somewhere, we can return that.

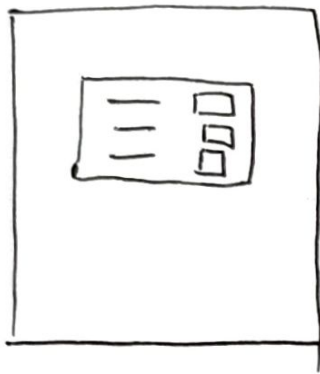
In case of reject we can return error msg



- Fetch all HTML els

- Have each product render on the website

S1



make  
product  
add to cart  
option  
display

S2

Make add to  
cart actually  
add to cart



S3

Kart list Total

Calculate and display at the end.