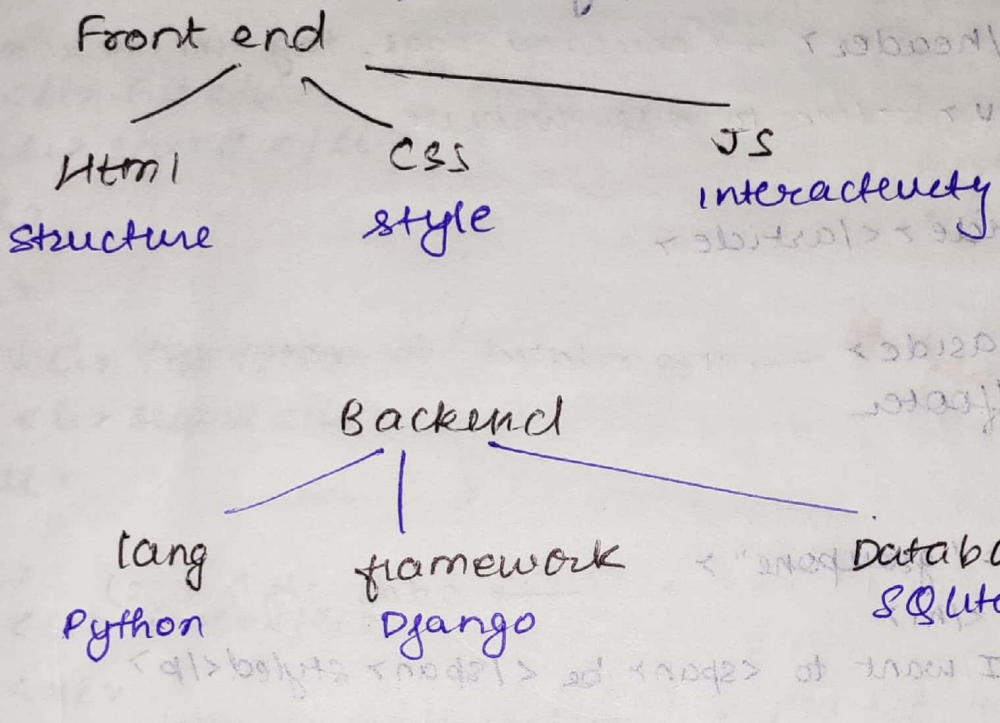
HTML



FORMAT →

<!DOCTYPE html> → signal to use standard html

<html>

<head> → describe main item on page

<meta charset = "utf-8">

<title> This is title! </title> → required!! shows at tab

</head>

<body>

<!-- Anything in is comment --> → or write text ctrl + /

<h1> This is </h1> → Big heading on page

  → para tags in body

h1
h2
h3 ↓ size decreases

</body>

</html>

Tags in body →

① <h1></h1> ② ③ ④ → new

Heading, para, bold, italics

↓ new

⑧

⑤ <header></header> → company logos, tag line, header info

⑥ <nav></nav> → lines to diff website

⑦ <section>
<article></article>
</section>

⑧ <aside></aside> → info related to main topic

⑨ <footer></footer>

⑩ ~~ ~~

```
<div class="groupone">
    <h1></h1>
    <p> I want to <span> be </span> styled</p>
</div>
```

→ adds style (CSS)

apply effects for only 'be'

→ List < ordered / unordered

```
<body>
  <h1> Lists </h1>
  <ol>
    <li> fist </li>
    <li> second </li>
  </ol>
  <ul>
    <li> fist </li>
    <li> second </li>
  </ul>
  <ol>
    <li> First </li>
    <ul>
      <li> First subpoint <li>
    </ul>
  <ol>
```

ordered

unordered

nested list

Lists
1. First
2. second
- First
- second
1. First
   - First subpoint

* include ul or only li tag allowed**

→ Attributes → allows to add more info to html tags
- adding link, referencing image

① Images →
self ending tag

```
<image src = "url.jpg" width = "400" height = "235"
  alt = "This is the picture" >
```

if image in same directory src = "imagename.jpg"

shows only if image is missing

② linking page →

```
<a href = "newpage.html" > Click me </a>
<a href = " url " </a>
```

if in same directory

* to open in new tab the link

```
<a href = "url" target = "_blank" > Click </a>
```

If picture show

if doesn't

This is the picture

Click me → will direct to page

③ Linking on same page

```
<h1> Link on same page </h1>
<section>
   <a href = "#section 1"> section 1 </a>
   <a href = "#top"> back to top </a>
```

```
<h1 id="top"> Links to same page </h1>
<section>
   <ul>
     <li><a href = "#section 1"> section 1 </a></li></ul>
</section>
<section id = "section 1">
 <h3> section 1 </h3>
  <p> - - - </p>
 </section>
```

→ TABLES →

```
<body>
  <table border = "1">
    <thead>
      <th> Number </th>
      <th> colour </th>
    <thead>
    <tr>
      <td> 150 </td>
      <td> yellow </td>
    </tr>
    <tr>
      ---
    </tr>
  </table>
```

will make border

space b/w inner & outer margin

creating row of headings for each column

table Heads

row 1

row 2

Number	Colour
150	yellow
200	Red

FORMS →

```
<body>
  <form>
    <h1>Log in</h1>
    <h2>Please input email & pass</h2>
    <input type="text" name="" value="">
  </form>
</body>
```

Enter email → looks like this if unfilled

type of input — name of input (assigns value to this name) — value of input (to be prefilled)

Submit → [submit] value="submit" text → [enter text]

password → [●●●●●●] → Enters like this

email → [@gmail.com] submits only if detected @gm

Action →

```
<form action="url" method="get">
```

destination page → on pressing submit leads to url

label →

① (inside <form>)

Individual

```
<label>
  Enter text:
  <input type="text" name="" value="">
</label>
```

→ Enter text: [ ]

② for all

```
<label for="user">Enter Input:</label>
<input id="user" type= - - - - >
```

label for → id

→ Enter text: [ ]

id → uniquely identifies the type.

★ Instead of value, use placeholder.

need to be erased

```
<input type="text" name="" value="Help">
```

→ [Help]

```
<input type="text" name="" placeholder="Help">
```

[Help] ↓ type over

★ If you want to make a field mandatory to be filled before submit, use required.

```
<input type="password" name="" value="password" required>
```

if submit without fill → popup → [submit] → please fill this

Scanned with CamScanner

→ Input Methods.

① Radio button ◉

★ If multiple radio types are named same, only one of those ◉ can be selected →

```
<input type="radio" name="loc" value="inside">
<h2> How service ? </h2>
```

shows in site as name = value

```
<select name="stars">
    <option value="Great"> 3 </option>
    <option value="Bad"> 2 </option>
</select>
```

→ 3 ▽ / 2 Dropbox

```
<h2> any other feedback (rect box to write) </h2>
<textarea name="mytext" rows="8" cols="80"></textarea>
```

CSS

→ linking to HTML →

(In Head)

```
<Head>
    <link rel = "stylesheet" href = "Path_master.css">
</Head>
```

(necessary; url/Path loc)

→ comment → /* */ → ctrl + /

→ General form →

```
selected tag {
    property : value;
}
```

eg →

```
h1 {
    color: blue;
}
```

→ changes colour of

↓ other way

```
colour : rgb(8,84,1);
colour: #0895a8;
color : rgba(
```

↓

α = 0 < < 1 → opaque

↓

→ transparent

→ div, span, background →

```
body {
    background : gsay;
}
```

→ can also pass a pic url by url(...)

→ changes background of body to gray

```
div {
    background : red;
    border-colour : blue;
    border-width : 10px;
    border-style : dotted;
}
```

→ background of only div to gsay

→ border type

div → also instead of px, can chose thin, medium, thick

```
span {
    background : white;
    color : black;
}
```

→ for inside span

★ to repeat background of body (image repeats on big page)

```
body { background : url ( --- ) ;
       background - repeat : repeat ;
     }
```

→ not repeat if don't want to repeat

text-decoration
→ line-through
~~text~~

→ Selectors →

id → targets single element
classes → targets groups of elements.

① HTML: `<div class = "name" > </div>` ⟷ CSS: `.name { colour : blue; }` (dot)

② `<p id = "name" > </p>` ⟷ `#name { colour : green; }`

③
```
* { colour : black ;
  }
```
→ will colour black all text except id

④
```
h3 + ul { border : 4px dotted purple ;
  }
```
→ any ul immediately after h3 (size, style, colour) } can write all border together.

⑤
```
li a { colour : red
  }

li a[href = "www.ppr.com"] { colour : blue ;
                             border : 5px solid orange ;
                           }
```
→ colour anchor tag → specifically site

→ **Specificity** →

<li class = "n" ① → If

<li class = "n" id = "t" ②

.n {
color: blue { → ①,② blue

#t { color: red { ② red.

★ id has ↑ priority.

Fonts → cssfontstack.com → check availability of font on OS

fonts.google.com → download fonts

① font-family: "Arial"; → font you know
font-family: cursive; → available inside

② font-size: 20px; → size of letter

Now, If we want size of 2nd para to be twice size of 1st para fonts -

```
body {
    font-size: 20px;
}
#two {
    fontsize: 2.0em;
}
```

← work acc to first defined px

1em = 20px

```
<body>
<p> ------ </p>
<p id="two"> -- </p>
</body>
```

③ Font-style: italic; → style.

④ font-weight: bold; → structure

⑤ text-align: centre; → aligns centre, left, right.

Scanned with CamScanner

BOX MODEL

can style left right bottom top

top {

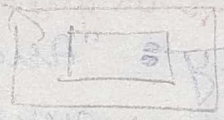
border : 4px solid blue; → width of border

width : 25%; → border length with rpct to page size

text-align : centre; → text is aligned at centre of border box

margin : auto; → sets border in centre of page

} margin : 10px 40px 200px 7px;
top right bot left



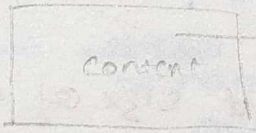
bottom {

border : 4px solid red;

width : 50%;

padding : 200px; → 200px is the space b/w content & border

}



Bootstrap

getbootstrap.com (site)

★ to get bootstrap classes → getbootstrap.com / download / get started / copy CDN link.

- container class → centres everything in → shifts right (creates a 10px margin)

∴ various classes for various requirement

① button class (check various on getbootstrap.com)

```
<button class="  " type="button" name="  ">
```

↳ type of look needed (eg → btn btn-success = green button)

can add more class as per requirement

(i) like class = "btn btn-success btn-lg"
green large

(ii) class " " disabled = "disabled"
can't click the button

② jumbotron class → design a page with heading para button

```
<div class="container">
  <div class="jumbotron">
    <h1></h1>
    <p></p>
    <p><a class="  ">  </p>
  </div></div>
```

margin to centres

from page link

③ form class →

① div <class = "form-group"

provides spacing b/w different inputs

② <input class = "form-control" >

puts email to email

provides design
- font highlighted
- box for email is stretched

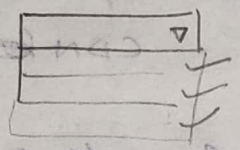
③ <small id = " " class = "form-text form-muted"> ---- </small>

email

we never share your info → [para like muted text]

④ for multiple selection drop box →

<select multiple class = "form control"---->



can select multiple options

⑤ text area →

★ form-control - also gives ability to click and stretch the area box

⑥ File upload input →

<input type = "file" class = "form-control-file" id = "">

CHOOSE FILE | NO FILE CHOSEN

↳ click gets to open a folder (selection - backend)

⑦ Radio button →

<legend> --- </legend> → underline

<fo class = "form-check> → used for radio button + check box

★ to disable any input just write <input --- disabled>

Navbars → navigation bars

① class = "navbar navbar-default navbar-fixed-top navbar-inverse"

navbar-default → in a grey tile

navbar-inverse → tile = black

navbar-fixed-top → even if scroll down, navbar is visible at top

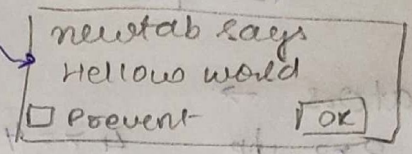
② class = "navbar-header" → main

Right click - inspect - console

JAVASCRIPT

use incognito tab.

① alert("Hello World") → string



② comment → // text

④ Strings → "text"

③ All no. are treated same ⑤ Booleans → false, true

⑥ undefined, null → variable created but not defined
→ setting nothing on purpose

⑦ to clear console → clear()

⑧ can do calculation → 2+2, 9/2 etc → will output result
15%14, 15*2
• Power → 2 ** 4 (2^4)

⑨ concatenate string → "Django" + "here" → o/p Djangohere

⑩ length of string → "Django".length → o/p = 6
* whitespace also counts

⑪ newline → ~~\n~~ \n

⑫ \t → tab space

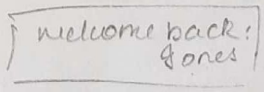
⑬ to add qoute → \" text \"

⑬ "Hello"[0] → o/p H

⑭ Variables → var varName = value;

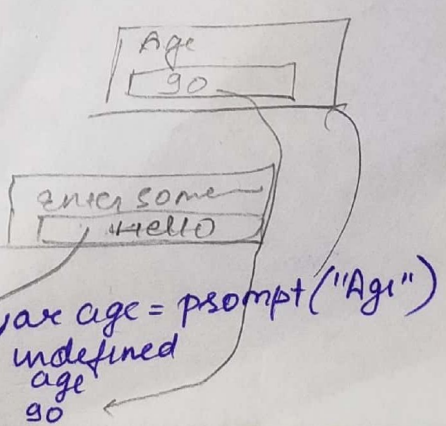
eg var account = 100; i/p
undefined o/p
account i/p
100 o/p

Eg - var greeting = "welcome back: ";
var name = "jose";
alert(greeting + name) →



⑮ to see output in console →
console.log("text") → text

⑯ prompt("Enter something") → o/p
"Hello"



dp

link js and html → file name
<script src = " " ></script> </head>

→ Operators →

3 > 2 i/p → true o/p

4 >= 4 → true

2 == 2 → true

"text" == "text" → true

"2" == 2 → true (type check "2" == 2 → false)

== → js changes into simple datatype to check equality

=== → check on datatype too

5 != "5" → true

5 !== "5" → false

- AND → 1 === 1 && 2 === 2 && 3 === 3 → false
- OR → 1 === 2 || 1 === 1 → true
- NOT → !(1 === 1) → false (opposite of cond)

→ if else

```
if ( )
{ }
else if ( )
{ }
else
{ }
```

→ while

```
while (condition)
{ }
```

→ break

```
while ( )
{ if ( )
  { break; }
}
```

leads out of while loop

→ for (initialization; cond; incrum)

```
{ }
```

→ Functions →

```
func_name (parameter1, para2)
{ }
```

→ In console, can call fn o/p - fn def.

→ console
function name → fn full

- ~~hello Cnam~~ • function hello (name = "Frank") → default parameter
  { console.log ("Hello" + name) } in case no para is passed

→ if on console →
① hello ("jose") o/p → Hello jose
② Hello () o/p → Hello Frank

- function formal (name = "Sam", title = "Sir") {
  return title + " " + name;
  }
  formal () o/p Sir Sam
- global variable assigned value doesn't change if assigned in a fn.
- function_variable inside fn remains to the scope of fn.

Arrays →

var var_name = ["1", "2", "3"].

★ string is immutable (can't be changed by changing index)

- array_name.pop() → pops last element

Eg. var last_element = arr.pop();

- arr.push ("item") → pushes "item" in the array arr
- arr.length → gives length

→ 2D array → var matrix = [[1,2,3], [4,5,6], [7,8,9]]

→ for printing array elements →
① for (i=0, i < arr.length, i++) { } ② console.log(arr)

② for (letter of arr)
{ console.log (letter); }

→ ★ arr.splice (index, till)
index of element
how many no. to delete starting from index

③ arr.forEach (alert) → fn name
arr.forEach(console.log) → arr elements are passed as parameter to the function

→ ★ arr.indexOf (name) → Finds index of name

Objects

- Syn → { key1: "value one", key2: "value two"; - }

values are accessed through keys

eg → var carInfo = { make: "Toyota", year: 1990 }

* when you make key don't have to address as string But

- to access any element →

carInfo ["make"] ← when calling, have to address as string

o/p → "Toyota"

- can nest objects

- var my = { a: "Hello", b: [1,2,3], c: { inside: ['a','b'] } };

→ my ['b'] ⟶ o/p ⟶ [1,2,3]

→ my ['b'][2] ⟶ o/p ⟶ 2

→ my ⟶ o/p → object { a: "hello", b: Array[3], c: Object }

→ my ['c']['inside'][1] ⟶ o/p → 'b'

- can change values →

my ['a'] = "Bye";

carInfo ['year'] += 1;

- console.dir(carInfo) → displays objects (object)

- for (K in carInfo) { → object

name console.log(K) → keys of K { make, year }

console.log (carInfo [K])

} → values in each key.

```
var simple {
    prop: "Hello",
    myMethod: function() {
        console.log("Hey" + this.prop)
    }
}
```

method

keyword

this → call object's prop

→ simple o/p ▽ object { prop: Hello }

Dbl this → console.dir(simple) o/p → ▽ object

→ simple.myMethod() ⟶ o/p → Hey Hello