16/5/21

Javascript & Browsers

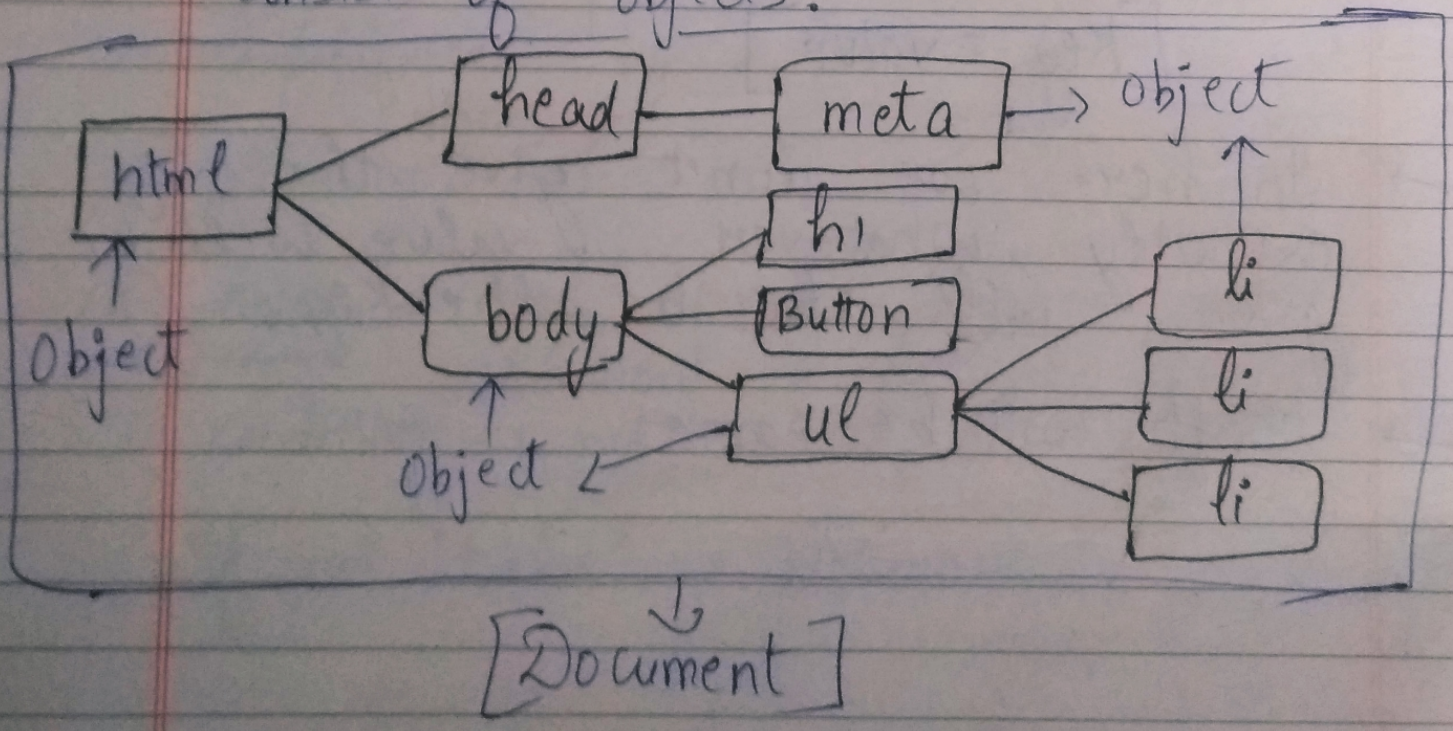
Page No.
Date:

* Introduction to DOM →

→ "Document Object Model" is the data representation of the objects that comprise the structure and content of a document on the web.

→ It represents the page so that programs can change the document structure, style, and content. The DOM represents the documents as objects. That way, languages like javascript can connect to the page.

→ DOM is a tree like structure which consists of objects.



* Everything that exists around us is an object.

Ex → Person, Table, Car etc...

→ Every object has properties & methods.

Properties → name
subject
occupation

methods → bio() → return bio
→ teach

→ Teacher

→ Properties are like adjectives like if their is a car, what is color and name of the car

whereas,

→ methods are like verbs like if she is a teacher what methods can be applied. Ex→ bio() (more like function) returns the bio-data of the teacher.

→ {So, in here we try to create objects just like real world objects.}

So,

method → hover button

h1

Properties → color;
size;

Ex → car = {
name: 'Honda city',
color: 'red',
drive: function() {
console.log();
}
}

overview of how to create an object.

o → So, in order to access the properties of an object we use →

car.name
car.color
car.drive

o → If we have property inside a property.

name: {
model:
brand:
}

then we can do → car.name.model.

———— x ————

→ So, if we need to access objects inside a DOM we use "document". (super object)

* Accessing DOM Elements →

→ document.getElementByTagName("h1")

then we get,

O/P ▸HTMLCollection [h1]
0: h1
length: 1

So, now if we want to style it what we need to do is, →

[document.getElementByTagName("h1")[0].style.color = "red".]

→ Since we are accessing the elements and getting them in the form of array, so to style an element we must use index number of that element.

→ We can also access the elements using class name or id.

Ex → document.getElementByClassName(" ");

document.getElementById(" ");

⇒ ~~Console.log(document.getElement~~

⇒ We use →

```
Consde.log(document.querySelector("li"));
```

→ It returns the first element that matches a specified CSS selector(s) in the document.

→ To returnall we use →

"{ querySelectorAll() }"

* To access a class →

[querySelector(".classname");]

* To access all elements of a class →

[querySelectorAll(".classname");]

* To access an 'id' →

[querySelector("#idname");]

Ex → Suppose we have two list and we want to access one of it along with the elements present inside it, we do →

```
<ul class="list">
    <li class="list-item"> </li>
    ——— " ———————
    ——— " ———————
</ul>

<ul class="list-1">
    <li class="list-item"> </li>
    ——— " ———————
    ——— " ———————
</ul>
```

[To access 'list' class → we use space

document.querySelectorAll(".list .list-item");

Parent Class — child class.]

* Suppose we have class name as "text" for 'li' tag and 'p' tag.
If we want access the p tag then we can simply do →

[querySelectorAll("p.text");]

(we don't use space b/w the element & class name)

* Manipulating DOM Elements →

→ In CSS, we had →

	JavaScript
border-color	borderColor
background-color	backgroundColor

→ We can do →

Ex - (1)
```
var listItem = document.querySelectorAll(".list-item");
listItem.forEach(item => {
    item.style.color = "blue";
})
```

* To add a class through JS without writing inside html element we do ⇒

→
```
var heading = document.querySelector("#");
heading.classList.add("page-heading");
Consok.list(heading.classList);
```

↓ It will add class and properties to the element

(1) To remove a class →

→ heading.classList.remove("classname");

(2) heading.classList.toggle("xyz");

If the class "xyz" is present inside the tag it will remove it

else, if it is not their, it will add it.