Turwash Notes

16th February 2023

DOM + Modern JS - class 1

→ DOM
→ BOM
→ window

1) Window :-

can access anywhere

↳ global object which represents window created by browser

$$\begin{tikzcd}
 & \text{window} \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\
\text{DOM} & \text{BOM} & \text{JS core}
\end{tikzcd}$$

Topmost hierarchy is window
all methods & properties lie in window.

→ it represents a browser window, can control browser window

eg. window.console.log (—)

2) DOM :- Document object Model.

Convert HTML code to JS object, this is called DOM.

write document in console, for whole HTML code to document

to access body, document.body.

// we will learn How we will change HTML codes or CSS codes using JS. //

3) BOM :- Browser Object Model

↳ It allows JS to talk to browser about matters other than content of page.

matters like location, History, Screen

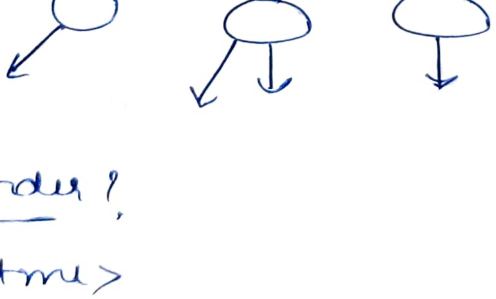
BOM is used to communicate to browser. (Like alert)

→ Indepth, DOM

Document Object Model.

web page converted to JS.

It is a treelike structure.



How it render?

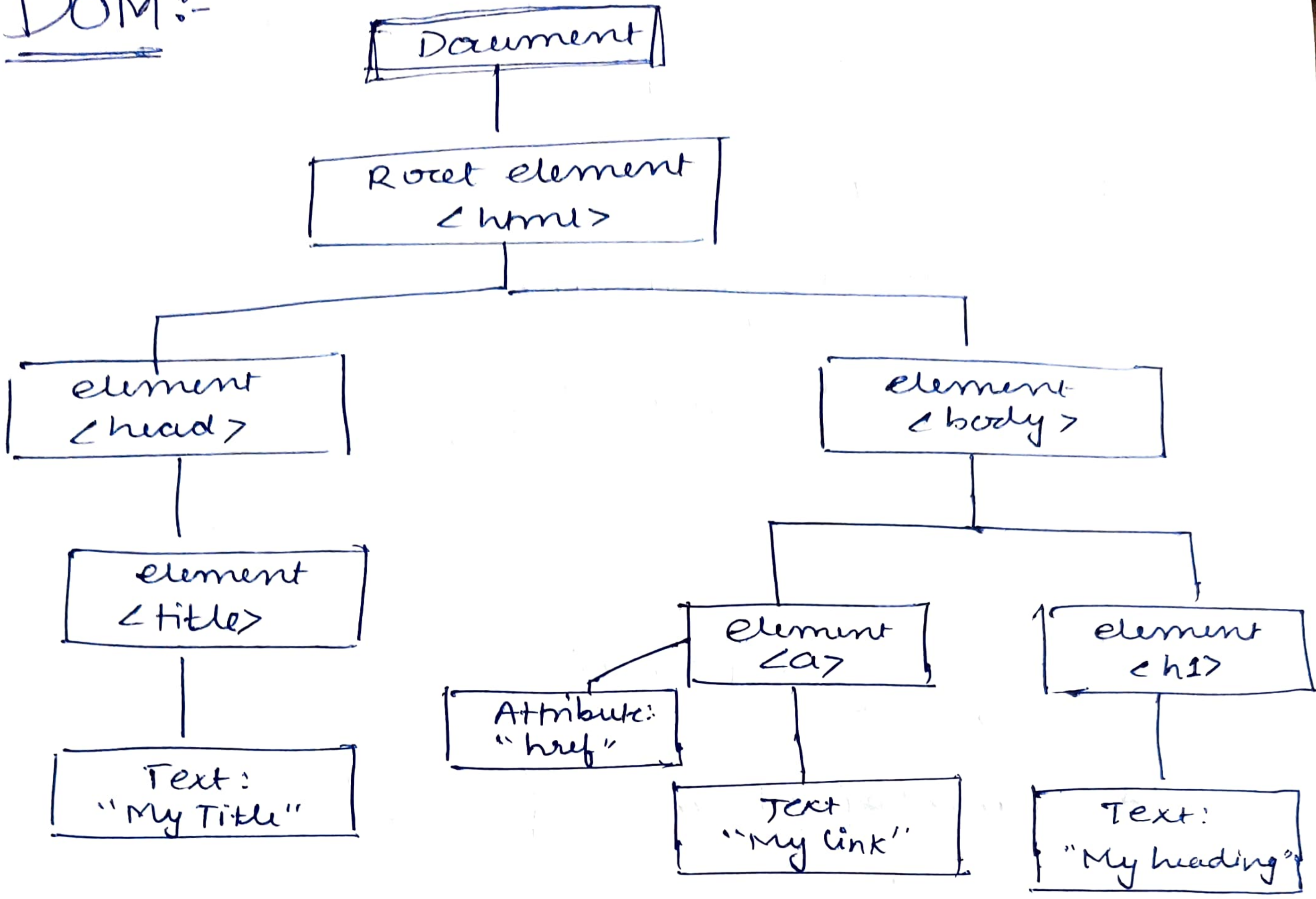
<html>

* First character → <html> → then, Tags

Then

Tags → Token (using tokenizer) → converts to Nodes → DOM is ready

DOM :-



Method to Fetch any particular element

↓

getElementbyId('heading') ← id of html tag.

↓

→ it is called on document object

→ It returns a single object (because id is always unique)

↳ For multiple

↳ getElementsByClassName()

→ returns array-like-object of all child elements.

(HTML collection interface)

✓ to iterate on ~~d~~ Document.getElementsbyClassName we use For Loop.

To Fetch Tag

getElementByTagName.

↳ return multiple tags. of HTML doc.

{ getElementsbyClassName() & getElementByTagName() }

↓

1) Both method use document object
2) Both return multiple items
3) The List returned is Not an Array its HTML Collections.

Trick :-

Select or hover particular element then in Console write $0 to fetch that particular element

---

then we can also put it in variable

let para = $0

✓ We can also fetch class Name

{ para.className
or
$0.className }

more ways :-

querySelector () method.

```
let a = querySelector ('#header'); → Id

let b = querySelector ('.header'); → class (only First)

let c = querySelector ('header'); → tag (only First)
```

↳ only returns single output First one.

For Multiple Selector

✓ querySelectorAll () method.
↑ for all class & tags.

Update Existing Content of web page

properties.
- .innerHTML — get/Set HTML content
- .outerHTML — (H/W)
- .textContent
- .innerText } get/Set textual content

1) .innerHTML

→ get an element / all of it descendants HTML content.

→ Set an element's HTML content

innerHTML
↳ will try to render HTML tags if written in b/w.

but

textContent
↳ tags will also be treated as normal text.

↳ this will also show the Hidden Display

innerText
↳ This will not show the 'Display Hidden'

Adding New Element /Content Using JavaScript :-

→ •createElement()

↓

let newchild =
ex:- document.createElement('span')

↓

(create)

to add

content.appendChild(newchild);

example:-

let content = document.querySelector('.class');

let para = document.createElement('p');

content.appendChild(para);

paragraph tag will be added

↓

(in above of last tag).

→ Creating TextNode:-
(1)

let para = document.createElement('p');

let text = document.createTextNode('I am the text')

~~my~~ para.appendChild(text);

content.appendChild(para);

<p> I am the text </p>

② Easy Way –

~~ta~~ let para = document.createElement('p');

para.textContent = "I am the text";

content.appendChild (~~m~~ para);

↓

last sibling

↳ But,

If we want to do positioning of our added element

→ insertAdjacentHTML()

- has to be called by 2 argument
- Location / position (where) ①
- HTML text / content to be inserted (what ②

{ beforebegin → add previous sibling.
afterbegin
beforeend
afterend }

— before begin —

<p>

— after begin —

<div> —— </div>

— before end —

</p>

— after end —

Example :-

```
let content = $0;

~~let Text = '<h3> Text </h3>~~

let newText = document.createElement ('h3')
newText.textContent = 'ABCD';
content.insertAdjacentElement ("beforeBegin",
                                newText);
```

Remove

↳ .removeChild ()
- ↳ opposite of appendChild ()
- parent element known
- the child element to remove is must be known.

```
parent.removeChild (childElement);
```

give class to element, then.

```
let childElement = document.querySelector
                        ('.tempText');

let parentElement = document.querySelector
                        ('.parentText');

parentElement.removeChild (childElement);
(content)
```

One More Way

↳ without parentElement deletion.

parent = $\underbrace{\text{childElement.parent}}_{\text{To find parent.}}$

child.parent.remove(child); ⟶ H/W

Now, For CSS

Style page content using JS

- .style
- .cssText
- .setAttribute
- .className
- .class list

} properties we have.

$\boxed{\text{Inline CSS} \checkmark \text{ high priority.}}$

① let content = $0

$\underbrace{\text{content.style.color = 'red';}}_{\text{we can only modify one element with this property.}}$

(2) Content.style.cssText = 'color:green;
background-color:yellow;
font-size:4em';

$\underbrace{\hspace{6cm}}$

here we can do
for multiple properties.

(3) content.setAttribute ('style', 'color:Red');

$\underbrace{\text{'style'}}_{\text{name}}$ $\underbrace{\text{'color:Red'}}_{\text{value}}$

(also can add
multiple)

also, we can add id, class etc.

✓ content.setAttribute ("id", "this id");

↳ but we are breaking
Seperation of concern here
to resolve We have
other properties.

(4) content.className
to get all class Names of content.
↳ will return String.

content.className.trim().split(' ');

↓

will return array of classes

↑ Its lengthy use .classList
will return object.
(array of classes) ✓

class List

- return Array of classes.
- add()
- remove()
- toggle() :- if element not present then add, if present then remove.
- contains() :- if element present return True if not present will return False

———————— x ———————— x ————————