DOM in JS

What is DOM?

- DOM stands for (Document Object MOdel).
- The DOM is a programming interface for web documents that represents the HTML or XML documents as a tre structure, where each node represents an element, attribute, or piece of text in the document.
- When a web page loaded, the browser creates a DOM tree that represents the document's structure and content.
- Each node is represent as a js object, which we can access and manipulate using DOM API.
- All html elements, comments, text, content etc are referred as nodes od DOM tree.

DOM API

- DOM API is a set of programming interfaces and methods that allow developer to interact with the DOM tree and manipulate the content and structure as their requirements.
- It helps in create, modify and delete elements and attributes, change styles and classes, handle events, and more.

HTML structure.

> Referrences HTML Structure

```
<body>
  <h1>Falling In Love With Javascript</h1>
  <div class="container">
    <div class="item item1" id="itemone">1</div>
    <div class="item item2">2</div>
    <div class="item item3">3</div>
    <div id="itemfour" class="item item4">4</div>
    <div class="item item5">5</div>
  </div>
  <p>hello i'm paragraph</p>
</body>
```

Target Elements

1. getElementById('id_name')

- It returns references of single object where id_name matches.

```
let divone = document.getElementById("itemone");
console.log(divone);
```

---

2. getElementByClassName('class_name')

- It returns html collection all element matches with class name.

```
> Apply backgroundColor, Margin, fontsize and padding on each div

let div_child = document.getElementsByClassName("item");
console.log(div_child);
```

---

3. getElementByTagName('tag_name')

- It returns html collection all element matches with tag name.

```
> Display parent div as flexbox

 let divs = document.getElementsByTagName("div");
 console.log(divs)
```

```
 divs[0].style.backgroundColor = "yellow";
 divs[0].style.padding = "10px";
 divs[0].style.display = "flex";
 divs[0].style.gap = "10px";
 divs[0].style.justifyContent = "space-between";

 for (let i = 1; i < divs.length; i++) {
 divs[i].style.backgroundColor = "blue";

  divs[i].style.fontSize = "32px";
  divs[i].style.padding = "10px";
  divs[i].style.color = "white";
}
```

---

4. querySelector('css_selector')

- It returns references of the first element that matches a specified CSS selector.

```
> Change fontsize of first div with 'item' class.

  let ele = document.querySelector(".item");
  ele.style.fontSize = "52px";
  console.log(ele);
```

---

5. querySelectorAll('css_selector')

- It returns Nodelist of all elements that matches a specified CSS selector.

```
> Change fontWeight of all div with 'item'.

  let eles = document.querySelectorAll(".item");
  for(rel of eles){
    ele.style.fontWeight = "bold";
   }
   console.log(eles);
```

Create an insert Element.

1. CreateElement('tag_name')

- it is used to create a new html element of the specified type and returns a reference to it as a javascript object.

```
> Create Section tag.

  let sec = document.createElement("section");
  console.log(sec)
```

---

2. apeendChild(element)

- it is used to to insert the element as last child.

```
> Insert the section tag inside div tag having class 'container'.

  let sec = document.createElement("section");
  let pdiv = document.getElementsByClassName("container")[0];
  pdiv.appendChild(sec);
```

---

3. insertAdjacentElement('position',element)

- it is used to to insert the element as a child or sibling.
- Positions: beforebegin, afterbegin, beforeend, afterend.

```
> show how to display element as a child and sibling of div having class
'continer'.

  pdiv.insertAdjacentElement("beforebegin", sec);
  pdiv.insertAdjacentElement("afterend", sec);
  pdiv.insertAdjacentElement("afterbegin", sec);
  pdiv.insertAdjacentElement("beforeend", sec);
```

Insert text elements.

1. textContent

- It is used to insert text inside element.

```
> Insert "Hello" text inside p tag.

let p = document.getElementsByTagName("p")[0];
p.textContent="Hello";
```

```
> Insert "Hello" text inside p tag and preserve previous text also.

let p = document.getElementsByTagName("p")[0];
p.textContent +="Hello";
```

---

innerHTML

- it is used to insert text and html tag inside element.

> insert "**Hello**" inside P tag.

```
let p = document.getElementsByTagName("p")[0];
p.innerHTML="<strong>Hello</strong>";
```

> insert `"<strong>Hello</strong>"` inside P tag and preserve previous text and element.

```
let p = document.getElementsByTagName("p")[0];
p.innerHTML+="<strong>Hello</strong>";
```

Insert and remove attribute.

1. setAttribute('attribute_name', 'value')

- It is used to insert the attribute to an element.

> insert id = "Avinash" to third div in the container.

```
let divs = document.getElementbyClassName("item");
divs[2].setAttribute("id", "Avinash");
```

---

1. removeAttribute('attribute_name')

- It is used to remove the attribute from an element.

> Remove attribute is attribute from third div of the container.

```
let divs = document.getElementbyClassName("item");
divs[2].removeAttribute("id");
```

Traverse HTMl nodes.

1. parentElement

- it returns the reference of parent html element.

> Print parent element of div whose class is "item"

```
let div_child = document.getElementsByClassName("item");
console.log(div_child[1].parentElement);
```

---

2. nextElementSibling

- it returns the reference of next html sibling.

> Print next sibling element of third div whose class is "item"

```
let div_child = document.getElementsByClassName("item");
console.log(div_child[1].parentElement);
```

---

3. previousElementSibling

- it returns the reference of previous html sibling.

> Print previous sibling element of third div whose class is "item"

```
let div_child = document.getElementsByClassName("item");
console.log(div_child[2].previousElementSibling);
```

---

4. children

- it returns html collection of html all childs element.

> Print children element of div whose class is "container"

```
let pdiv = document.getElementsByClassName("container")[0];
console.log(pdiv.children);
```

---

5. childnodes

- it returns Nodelist of all types o nodes like string, text, comment, ets.

> Print all child nodes of div whose class is "container".

```
let pdiv = document.getElementsByClassName("container")[0];
console.log(pdiv.childNodes);
```

Remove HTML element.

1. remove

- It is used to delete html element.

> Remove p element from DOM.

```
let p = document.getElementByName("p")[0];
p.remove();
```