What is DOM Manipulation ?

Document Object Model

Theory

JavaScript appeared in 1995 in Netscape Navigator.

It makes web pages more dynamic and interactive. Before JavaScript, user's interaction with the web page was limited to visiting links and submitting forms.

JavaScript makes it possible to:

- ➔ Manipulate elements of a web page: either their content or their attributes, among which classes, style, etc.
- ➔ Take the user’s actions into account, such as clicking on an element, dragging and dropping from one element to another, etc.

Theory

From HTML to DOM

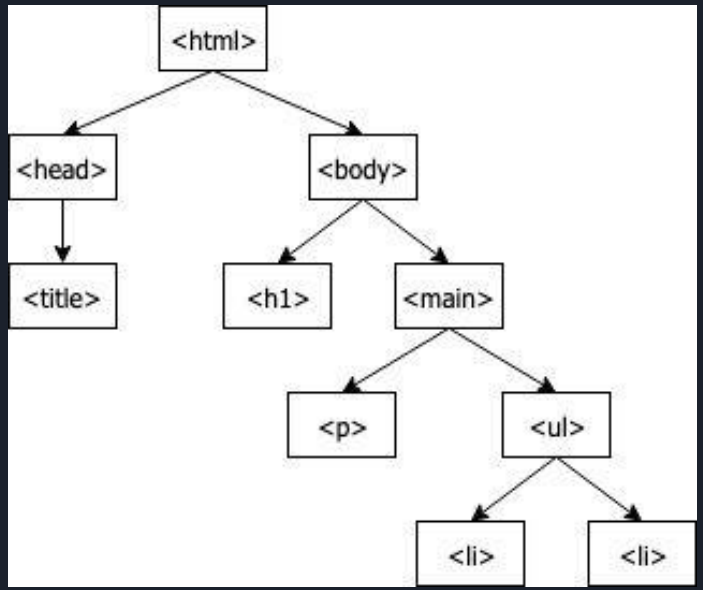
When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects: called nodes

Theory

From HTML to DOM

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>DOM example</title>
  </head>
  <body>
    <h1>Hello DOM</h1>
    <main>
        <p>There goes a <em>paragraph</em></p>
        <ul>
            <li>first item</li>
            <li>second item</li>
        </ul>
    </main>
  </body>
</html>
```



Theory

What can you do with DOM manipulation?

- Change HTML elements
- Change HTML attributes
- Change CSS styles
- Remove existing HTML elements & attributes
- Add HTML elements & attributes
- React to existing HTML elements

Selection

Retrieve an element by i’ts ID

`document.getElementById('myId')` allows you to retrieve an element via its id

`document.querySelector('#myId')` allows you to retrieve an element via its id thanks to a CSS selector

Wait! `document`?

The document object represents your web page. You have to access the document object first if you want to access an HTML element on your page.

Selection

`document.getElementById()` allows to retrieve an element *via* his id.

```
<body>
  <p id="first">First paragraph.</p>
  <p id="second">Second paragraph.</p>
  <p id="third">Third paragraph.</p>
</body>
```

```
// Get the paragraph with id second
const secondElem = document.getElementById('second');

// Alter both its content and style
secondElem.innerHTML = '<em>Something</em> happened here!';
secondElem.style.fontWeight = 'bold';
```

Selection

```
<!DOCTYPE html>
<html>
  <head>
    <title>I love DOM Manipulation</title>
    <meta charset="UTF-8 " />
  </head>
  <body>
    <div id="app"></div>
    <h1 id="main-heading ">Hello Students!</h1>
    <script>
      document.getElementById("main-heading").style.fontFamily = "Tahoma";
    </script>
  </body>
</html>
```

element.style.property the property will always be in camelCase

Manipulation

Change the content of the body element

```
// Try this in your browser's console, on any webpage

// Replace the body's content with our own
document.body.innerHTML = '<h1>Hello world</h1>'

// Apply styles to the body (much like in CSS)
document.body.style.color = 'blue'
document.body.style.fontSize = '60px'
```

Manipulation

```
<button>First button</button>
<button class="red">Second button (with a class)</button>
<p class="red">I have the same class as the above button!</p>
```

```
// Get all the button elements at once
const allButtons = document.getElementsByTagName("button");
for (let i = 0; i < allButtons.length; i++) {
  allButtons[i].style.backgroundColor = "yellow";
}
// Get all the elements having the class "red"
const allWithRed = document.getElementsByClassName("red");
for (let i = 0; i < allWithRed.length; i++) {
  allWithRed[i].style.color = "red";
}
```

Events

React to the click of an event

```
<body>
  <button id="button1">Change my text!</button>
</body>

// Get the 1st button
const firstBtn = document.getElementById("button1");
// Initialize a counter
let count = 0;
// Attach an event listener for the click event
firstBtn.addEventListener("click", () => {
  // Increment the counter, build a string, update the button's content
  count += 1;
  const text = "I have been clicked " + count + " times";
  firstBtn.innerHTML = text;
});
```

`element.addEventListener()` can detect an event of a certain type (click, mouseover, etc.), and call a function when it occurs. From where these two parameters (type of event in 1st, function in 2nd).

Events

React to an event change on multiple elements

The function passed as 2nd parameter to `element.addEventListener()` can use an `event` parameter whose `target` property is the targeted element.

```
<select>
  <option value="kiwis">Kiwis</option>
  <option value="bananas">Bananas</option>
</select>

<select>
  <option value="carrots">Carrots</option>
  <option value="broccolis">Brocolis</option>
</select>

<div>I want to eat <span id="show-choice">something</span></div>
```

```
const choiceSpan = document.getElementById("show-choice");
const selectors = document.getElementsByTagName("select");
// Attach an event listener to each of the select elements,
// responding to the "change" event
for (let i = 0; i < selectors.length; i++) {
  selectors[i].addEventListener("change", (event) => {
    // event.target is the selector that has been changed,
    // its value is that of the selected option
    choiceSpan.innerHTML = event.target.value;
  });
}
```

Overview

FIND elements	**CHANGE** elements	**ADD** & **DELETE** elements
document.getElementById(id)	element.innerHTML / element.innerText	document.createElement(element)
document.getElementsByTagName(name)	element.attribute (property) element.setAttribute(attribute, value) (method)	document.removeChild(element)
document.getElementsByClassName(name)	element.style.property	document.appendChild(element)