DOM

CONCEPTS

DOM Introduction

Event Bubbling

Selecting elements :-

Based on element's first/last child

Based on tag, class or id name

Using query selector

Manipulating document :-

Change text or html of a tag

Change styles

Add classes to elements

Manipulate attributes of element

Event handling :-

Event handling – basic syntax

Click & Scroll events

Mouse events

Key events

Form events

DOM - INTRODUCTION

- DOM catalogues the web page into individual objects that we can select & manipulate
- On loading the web page the browser coverts html elements and their relationships into a DOM tree structure
- Objects in DOM have properties & Methods

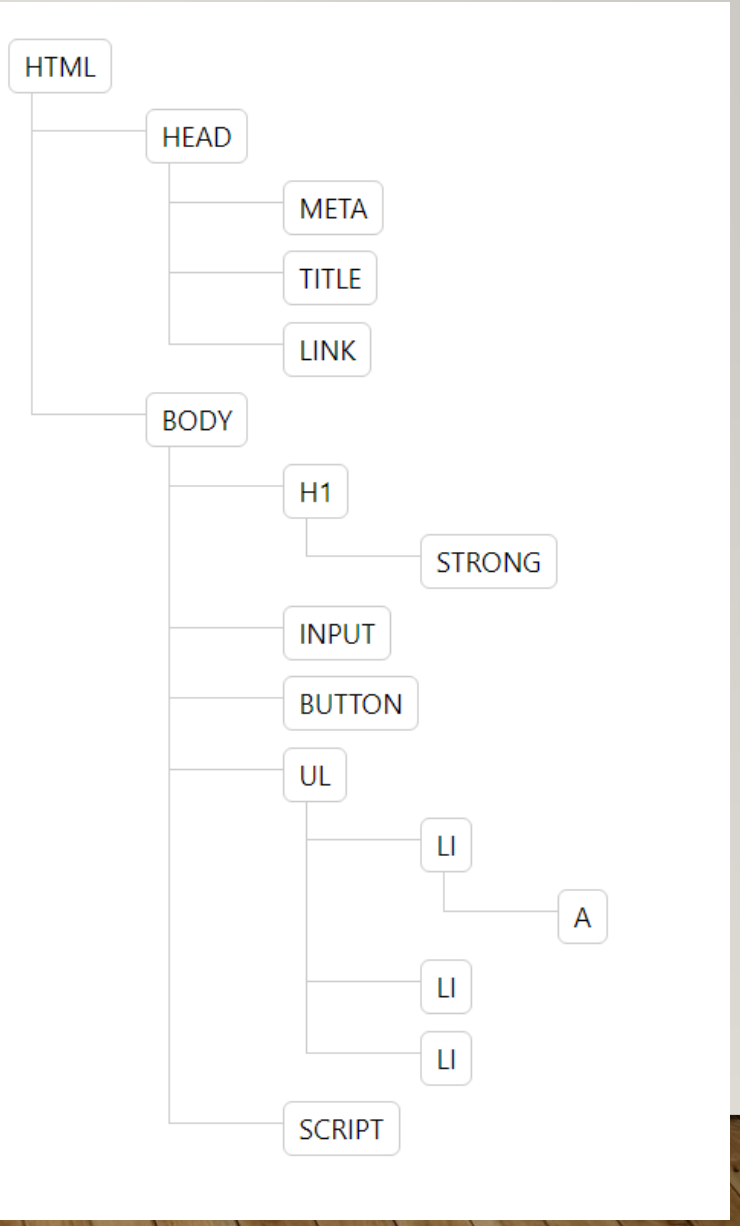
```
<html>
  <head>
    <title>My Website</title>
  </head>
  <body>
    <button>Alert</button>
  </body>
</html>
```

DOM

The following code will be manipulated using DOM as an example

HTML code

```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>My website</title>
    <link rel="stylesheet" href="DOMstyles.css" />
  </head>
  <body>
    <h1 id="title"><strong>Hello</strong></h1>
    <input type="checkbox" />
    <button class="btn">Click Me</button>
    <ul id="list">
      <li class="item">
        <a href="https://www.google.com">Google</a>
      </li>
      <li class="item">Second</li>
      <li class="item">Third</li>
    </ul>
    <script src="index.js"></script>
  </body>
</html>
```



SELECT BASED ON ELEMENT'S FIRST/LAST CHILD

document • The document represents the Entire html document	**// outputs entire HTML file** document;
firstElementChild • It represents the element's first child	**// outputs everthing inside <html>** document.firstElementChild; **// outputs <head> part** document.firstElementChild.firstElementChild;
lastElementChild • It represents the element's last child	**// outputs <body> part** document.firstElementChild.lastElementChild; **// outputs the <h1>** document.firstElementChild.lastElementChild.firstElementChild; **// outputs the last of ** document.firstElementChild.lastElementChild.lastElementChild.lastElementChild;

SELECT BASED ON TAG, CLASS OR ID NAME

getElementsByTagName • It selects elements based on tag name • It returns an array	**// Selects list items & returns array of list items** document.getElementsByTagName("li"); **// returns no. of li elements** document.getElementsByTagName("li").length //3
getElementsByClassName • It selects elements based on class name • It returns an array	**// returns array of items that have btn class** document.getElementsByClassName("btn")
getElementById • It selects elements based on id name • It returns a single item	**// returns item that has title id** document.getElementById("title")

SELECT USING QUERY SELECTOR

querySelector("Selector") • It selects elements based on selector specified. • The selector syntax is same as css selectors syntax. • It always selects the first child that satisfies the selector	**// select object that has input tag** document.querySelector("input") **// select element with an id of "title"** document.querySelector("#title"); **// select the anchor tag inside of li** document.querySelector("li a"); **// Returns first list item that has a class of item** document.querySelector("li.item");
querySelectorAll("Selector") • It selects all the elements with satisfies the selector • It returns an array	**// Returns an array of all list items that has a class of item** document.querySelectorAll("#list .item");

CHANGE TEXT OR HTML OF A TAG

textContent • It changes the text inside of the specified tag	**// Change the text inside of h1 to Good Greetings!!** document.querySelector("h1").textContent = "Good Greetings!!";
innerHTML • it adds/change html code inside of a particular tag • Can also be used to change the text	**// Change the html inside of h1 to Good Bye** document.querySelector("h1").innerHTML = "Good Bye" **// Change text inside of h1 to "Good Bye"** document.getElementById("title").innerHTML = "Good Bye";
Access methods of DOM objects	**// Ticks the checkbox of input** document.querySelector("input").click();

CHANGE STYLES

style.propertyName • It changes the style of the specified property name • More Style Object properties	**// Change color of h1 to red** document.querySelector("h1").style.color = "red"; **// Change font size of h1 to 2rem** document.querySelector("h1").style.fontSize = "2rem"; **// hide the h1 using visibility** document.querySelector("h1").style.visibility = "hidden“ **// Change background of .btn class element to yellow** document.querySelector(".btn").style.backgroundColor = "yellow“
Change style object properties of selectors that return’s an array The following selectors returns an array :- • getElementsByTagName • getElementsByClassName • querySelectorAll	**// changes color of 2nd list item to purple** document.getElementsByTagName("li")[1].style.color='purple’; **// change color of .btn class element to red** document.getElementsByClassName("btn")[0].style.color = 'red’; **// changes color of 3nd list item to blue** document.querySelectorAll("#list .item")[2].style.color = "blue";

ADD CLASSES TO ELEMENTS

classList • It returns all the classes that a particular element has	**// Shows list of classes attached to button element** document.querySelector("button").classList;
add( ) • It adds class to a element's class list • The major advantage of this is that we can use this to specify styles of classes in css rather than changing styles in js	**// Add invisible class to button element's class list** document.querySelector("button").classList.add("invisible");
remove( ) • It removes class from a element's class list	**// Remove invisible class from button element's class list** document.querySelector("button").classList.remove("invisible");
toggle( ) • It will remove class if applied or will apply class if not applied	**// Toggle invisible class from button element's class list** document.querySelector("button").classList.toggle("invisible");

MANIPULATE ATTRIBUTES OF ELEMENT

attributes • It returns a list of attributes that an element has	**// Shows list of attributes of anchor tag** document.querySelector("a").attributes;
getAttribute( ) • It accesses a particular attribute & returns its value	**// Access the href attribute of anchor tag** document.querySelector("a").getAttribute("href");
setAttribute() • It changes the value of a particular attribute	**// change the value of href attribute of anchor tag** document.querySelector("a").setAttribute("href", "bing.com");

EVENT HANDLING – BASIC SYNTAX

- Events are actions that happen in the browser & are triggered by the user.
- Eg. Button is clicked, page is loaded or scrolled, key is pressed etc.
- Event handling thus allows to control events and decide what should happen when an event is triggered

2 ways to handle to events :-

onEvent • We can use the onEvent attribute to handle events • Here onEvent refers to event name like onload, onclick, onscroll etc…	**// Trigger the following function on the element when the specified event takes place** element.onEvent = functionRef;
addEventListener() • We can use the following method to handle events	**// Trigger the following function on the element when the specified event takes place** element.addEventListener("event", function()) or element.addEventListener("event", function) Function() { }
Note :- In event handlers, **this** refers to the HTML element that received the event	**// Show which html element got clicked** document.addEventListener("click", function () { console.log(this); });

CLICK & SCROLL EVENTS

onclick / click

- Is triggered when user clicks on an element.

```
// Tigger the message function when the heading is clicked
const message = () => alert("Event triggered");
document.getElementById("title").onclick = message;

// Tigger the info function when the button is clicked
document.getElementsByClassName("btn")[0].addEventListener("click",
      function info() {
          alert("button pressed");
    });

//  adding click event listener to all buttons
var numbuttons = document.querySelectorAll(".drum").length  ;
for (var i = 0; i < numbuttons;  i++) {
   document.querySelectorAll(".para")[i].addEventListener("click" , function () {
        alert("i got clicked");
} ) }
```

onscroll / scroll

- It triggered when the document or an element has been scrolled

Example

MOUSE EVENTS

onmouseover / mouseover
- Is triggered when pointer is moved in element or it's children

onmouseout / mouseout
- Is triggered when pointer is moved out of element or it's children

onmouseenter / mouseenter
- Is triggered when pointer is moved in element

onmouseleave / mouseleave
- Is triggered when pointer is moved out of element

onmousedown / mousedown
- Is triggered when user presses mouse button over element

onmouseup / mouseup
- Is triggered when user releases mouse button over element

// The following mouse events are triggered on heading

```
document.getElementById("title").onmouseenter = message;
document.getElementById("title").onmouseleave = message;
document.getElementById("title").onmousedown = message;
document.getElementById("title").onmouseup = message;
```

KEY EVENTS

onkeypress / keypress
- Is triggered when key is being pressed.

onkeydown / keydown
- Is triggered when key is pressed down

onkeyup / keyup
- Is triggered when key is released

```
// The page will turn violet when v key is hold
document.addEventListener("keydown", (event) => {
  if (event.key == "v") {
    document.body.style.background = "violet";
  }
});

document.addEventListener("keyup", (event) => {
  if (event.key == "v") {
    document.body.style.background = "";
  }
});


// Show the keycode when key is pressed
document.addEventListener('keypress', function(event){
        alert(event.keyCode);
});
```

FORM EVENTS

submit

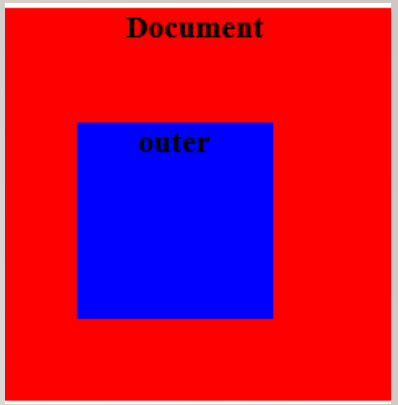
- Is triggered when form is submitted.

```
// html of form
<form id="new-post">
  <input id="post-title" type="text" />
  <textarea id="post-body" cols="30" rows="10"></textarea>
  <button name="submit" type="submit">Publish</button>
</form>


// log the data entered in text input & textarea
document.getElementById("new-post").addEventListener("submit", event=>{
    // prevent page reloading when form is submitted
    event.preventDefault()
    const postTitle = document.getElementById("post-title").value
    const postBody = document.getElementById("post-body").value
    console.log(`${postTitle} ${postBody}`)
})
```

EVENT BUBBLING

- Event bubbling is defined as propagation of event starting to trigger from the deepest target element to its ancestors/parents



Example

- In the following example when the red coloured document part is clicked then the document clicked message is triggered
- However when the blue coloured outer part is clicked the document clicked as well as outer clicked message is also triggered

```
document.getElementById('document').addEventListener('click', function(){
    alert('Document clicked')
})


document.getElementById('outer').addEventListener('click', function(){
    alert('outer clicked')
})
```

stopPropagation()

- We can use the following method to stop event bubbling

```
// now when the outer area is clicked only it's function will be triggerd
document.getElementById('outer').addEventListener('click', function(event){
    alert('outer div clicked')
    event.stopPropagation();
})
```