**JS DOM (Document Object Model)**

-DOM is a **structured representation** of your **HTML** document (can be thought as a tree of nodes or elements created by the browser)

-a browser API (Aplication Programming Interface) that is accessible to the JavaScript



-DOM can be used to **manipulate** (read, write, add, remove, etc..) those elements inside the HTML

-Is **Object Oriented**, meaning that each node has its own properties and methods

**How it Works?**

-the whole document by itself is a **literral object** (JS object)

-everything from the URL, domain, elements (body, header, etc..) is a **key/property-value** pair

-you can **read and manipulate** the **property or value**

**Readings and Manipulating**

console.dir(document);

-allows you to **read the key-value** pairs inside the **document file**

console.log(document.domain);

-shows you the **value of the property** (127.0.0.1)

document.title = 123;

-allows you to **change the HTML elements** (even though the title is set to <title>Item Lister</title>

console.log(document.body/head);

-shows the body and head tags from HTML

console.log(document.all);

-shows an **array of all the elements** (as in index-value pair) inside the document

-the „arrays“ are not actually arrays, they are **HTML Collections** (you **cant use some array methods**, but you can select them as in arrays)

console.log(document.all[10]);

-comes out as the **index :10**, which is the **<h1 id =“header-title“>Item Lister</h1>**

document.all[10].textContent = „Hello“; **(this is not the correct way!)**

-turns the **h1 tag** from **Item Lister** to **Hello**

console.log(document.forms)

-shows all the forms inside the document

console.log(document.links)

-shows all the external links in the document

console.log(document.images)

-shows all the images

**Styling**

var headerTitle = document.getElementById(‚header-title‘);  
headerTitle.style.borderBottom = ‚solid 3px #000‘; (CSS => border-bottom;

**Selectors**

**-selects elements** for you which you can later **read or manipulate**

**getElementById**

var headerTitle = document.getElementById(‚header-title‘); (the header title)

console.log(‚headerTitle‘); (shows the elements with the class header-title)

headerTitle.textContent = „Hello“; (**ignores the styling** of the elements =>the element has a styling of **display:none;**)

headerTitle.innerText = „Goodbye“; (**pays attention** to the **styling**)

**innerHTML**

-you can use **any HTML tags** which then passes **inside** that **selector element**

var headerTitle = document.getElementById(‚header-title‘);

headerTitle.innerHTML = ‚<h3>Hello</h3>‘; (passes this **h3 tag** element inside the **h1 tag** => looks like the **h1 turned into h3** but that isnt the case)

**getElementsByClassName**

var items = document.getElementsByClassName(‚list-group-item‘);

console.log(items); (shows a HTML collection of all the elements with the class of **list-group-item**)

console.log(items[1]); (shows the **second element** with **that class**)

items[1].textContent = „Hello 2“; (changes the text to Hello 2)

items[1].style.fontWeight = „bold“; (turns the text to bold)

items[1].style.backgroundColor = „yellow“; (bg turns yellow)

for(var i = 0; i < items.length; i++) { (you need to use a **for loop** for styling **all of the classes** as it works as an array)

items[i].style.backgroundColor = #f4f4f4;

}

**getElementsByTagName**

-same thing as in getElementsByClass but with tags

**querySelectors**

-you can select a element with any **CSS selector**, but it will only choose the **first one** (if there are **more of the same selector value**)

-if you want a **specific element** then you have to **specify the selector**

-if you want **all of the** **elements** then you have to use the **querySelectorAll**

var header = document.querySelector (‚#main-header‘);

header.style.borderBottom = ‚solid 4px #ccc‘;

var input = document.querySelector(‚input‘); (selected the tag element)

input.value = ‚Hello World‘; (changes the input text to Hello World)

var sumbit = document.querySelector(‚input[type=“sumbit“]‘); (example => there are **2 inputs** and the **second is a button** => we want to change the text to **SEND** => here we used the **CSS atribute selector** to select the **second over the first one**)

sumbit.value = „SEND“;

var item = document.querySelector(‚.list-group-item‘); (we selected all of the 4 classes but one the first element will change)

item.style.color = ‚red‘;

var lastItem = document.querySelector(‚.list-group-item:last-child‘);

lastItem.style.color = ‚blue‘; (the last element turned blue)

var secondItem = document.querySelector(‚.list-group-item:nth-child(2)‘);

seconItem.style.color = ‚blue‘;

**querySelectorAll**

-for selecting all of the elements

var titles = document.querySelectorAll(‚.title‘); (selects **all the classes** with this value)

console.log(titles); (shows you all the titles)

titles[0].textContent = ‚Hello‘; (**the first element** with the **class .title** changes to Hello)

let odd = document.querySelectorAll(‚li.nth-child(odd)‘);

for (let i = 0; i < odd.length; i++) {

odd[i].style.backgroundColor = ‚#f4f4f4‘; (every other element has a light gray background)

}

**Traversing the DOM**

-looking at parent nodes and child nodes

**parentNode and parentElement**

-both are almost the same thing

var itemList = document.querySelector(‚#items‘);

console.log(itemList.parentNode); (itemList = <ul>, itemList.parentNode = <div id=main>)

itemList.parentNode.style.backgroundColor = ‚f4f4f4‘; (styles the parent element of the ul)

console.log(itemList.parentNode.parentNode.parentNode); (shows the third parent of the itemList)

**childNodes and children**

-you shouldnt use **childNodes** because it also shows the **breaklines** as a **HTML collection object** which can be a pain in the ass

-children dont show it

var itemList = document.querySelector(‚#items‘);

console.log(itemList.childNodes); (shows an HTML collection of the **children element** of the **<ul>**) (but it also **shows any breakline as part of the collection**)

var itemList = document.querySelector(‚#items‘);

console.log(itemList.children); (shows it **normally**)

console.log(itemList.children[1]); (shows the 2nd element)

itemList.children[1].style.backgroundColor = ‚yellow‘;

**firstChild and firstElementChild + lastChild and lastElementChild**

**­**-firtChild gives you a **breakline** if there is one (you dont want to use this)

-firstElementChild gives you the first element normally

-lastChild and lastElementChild works the same way

var itemList = document.querySelector(‚#items‘);

console.log(itemList.firstElementChild);

itemList.firstElementChild.textContent(‚Hello‘);

**nextSibling and nextElementSibling + perviousSibling and perviousElementSibling**

-access the next or pervious sibling

console.log(itemList.nextElementSibling); (shows the next sibling)

itemList.perviousElementSibling.style.color = (‚green‘);

**Creating elements**

**Creating an element**

var newDiv = document.createElement(‚div‘);

**Adding class**

newDiv.className = ‚hello‘;

**Adding id**

newDiv.id = ‚hello‘

**Adding attribute**

newDiv.setAttribute(‚title‘, ‚Hello Div‘); (title = attribute, Hello Div = value)

**Creating text and adding into the tags**

-you have to create a **seperate variable** for this one

var newDivText = document.createTextNode(‚Hello World‘); (**created a text** inside a **new variable**)

newDiv.appendChild(newDivText); (adding the **text** into the **div element**)

**Inserting it inside the DOM**

-you need to create **2 additional variables**

-**first variable** is for which **parent element**

-**second variable** is for **before or after** which **sibling element**

var container = document.querySelector(‚header . container‘); (we selected **in which element** we want to **insert it**)

var h1 = document.querySelector(‚header h1‘); (we selected **before which element** we want to **insert it**)

container.insertBefore(‚newDiv, h1‘); (**newDiv** = in which element, **h1** = before which element)

newDiv.style.fontSize = ‚30px‘; (we changed the **styling of the div**)

**Events**

-mouse onclick, keyboard events, etc…

-adding interraction

**The old way (in HTML)**

<button **onclick=“buttonClick(1)“** class=“btn btn-dark btn-block“ id=“button“>Click Here</button> (**onclick=“buttonClick(1)“** => is **function call** connected to the **JavaScript code**

**The normal way**

var button = document.getElementById(‚button‘).addEventListener(‚click, function() {

console.log(‚123‘);

}‘); (**after clicking** the **button** it will **call an action**) (**click** = event type, **function()…** = input value => you can pass in calls like **buttonClick**)

function buttonClick() {

document.getElementById(‚header-title‘).textContent = ‚Changed‘;

document.querySelector(‚#main‘).style.backgroundColor = ‚#f4f4f4‘;

} (after clicking it will change the header to Changed and turn the background into gray)

**Event parameter**

-reading and manipulating with properties of the event

**Target parameters**

function buttonClick(e) {

console.log(e); (shows you **all the information regarding the event** => class, id, attributes of th element, position of the mouse, etc..)

}

function buttonClick(e) {

console.log(e.target); (shows you the **event element**)

}

function buttonClick(e) {

console.log(e.target.id); (shows you the **id of event element**)

}

function buttonClick(e) {

console.log(e.target.className); (shows you the **classes of event element**)

}

function buttonClick(e) {

console.log(e.target.classList); (shows you the classes of event element in a **DOM list**)

}

**Type parameters**

function buttonClick(e) {

console.log(e.type); (shows you the **type of the event** => in this case **click type**)

}

var output = document.getElementById(‚output‘);

output.innerHTML = ‚<h3>‘+e.target.id+‘</h3>‘; (**add a h3 element** with the inner text of button)

**Mouse position**

console.log(e.clientX); (shows the **location of your mouse** in the **X line** => from the browser window)

console.log(e.clientY);

console.log(e.offsetX/Y) (shows the location of your mouse in the X/Y line =>from the **element of the event** => in this case the **buttton**)

**Pressed keys**

console.log(e.altKey) (shows **true if pressed down** and then clicked, if **not pressed down** it will come as **false**)

console.log(e.ctrlKey)

console.log(e.shiftKey)

**Mouse event actions**

-all the mouse events

-after a certain action with the mouse something will change

var button = document.getElementById(‚button‘); (what element we want to change)

function runEvent(e) { (what change do we want)

console.log(‚EVENT TYPE: +e.type‘) (whenever we trigger the event it will display the type of our event)

}

button.addEventListener(‚click‘ , runEvent);

button.addEventListener(‚dblclick‘ , runEvent); (activates after a double click)

button.addEventListener(‚mousedown‘ , runEvent); (activates after clicking and moving the mouse down)

button.addEventListener(‚mouseup‘ , runEvent); (activates after holding and then releasing the click)

button.addEventListener(‚mouseenter‘ , runEvent); (activates after entering an element)

button.addEventListener(‚mouseleave‘ , runEvent); (activates after leaving an element)

button.addEventListener(‚mouseover‘ , runEvent); (activates whenever entering an element from the main element)

button.addEventListener(‚mouseout‘ , runEvent); (activates whenever leaving an element from the main element)

button.addEventListener(‚mousemove‘ , runEvent); (activates whenever moving your mouse inside the element)

var box = document.getElementById(‚box‘); (a grey div box with 200px width and length)

function runEvent(e) {

output.innerHTML = ‚<h3>MouseX: ‚+e.offsetX+‘</h3>

<h3>MouseY: ‚+e.offsetY+‘</h3>‘ (**hovering inside document** will show you the **mouse position**)

box.style.backgroundColor = „rbg(„+e.offsetX“,“+e.offsetY+“, 40)“; (hovering inside the box will **gradually change the color of the box**)

}

button.addEventListener(‚scroll‘ , runEvent); (whenever scrolled)

**Keyboard and input events**

-even that gets triggered by a keyboard action

var itemInput = document.querySelector(‚input[type=‘text‘]‘); (selectsonly the **text input**)

var form = document.querySelector(‚form‘); (for the **sumbit event**)

var select = document.querySelector(‚select‘) (for the **change event**) (<select class= „form-control mr-2“> (creates an **select form**)

<option>1</option>

<option>2</option>

<option>3</option>

</select>

function runEvent(e) { (**what change do we want**)

console.log(‚EVENT TYPE: +e.type‘); (whenever we trigger the event it will display the type of our event)

console.log(e.target.value); (shows the text/value of the target element in the console)

document.getElementById(‚output‘).innerHTML = ‚<h3>+e.target.value+</h3>‘; (whenever you type something into the input, the text will show itself in the docment)

}

itemInput.addEventListener(‚keydown‘, runEvent); (whenevera **key is pressed** **down** on the keyboard the event will trigger) (in this case **inside the form**)

itemInput.addEventListener(‚keyup‘, runEvent); (whenevera **key is let go** on the keyboard the event will trigger)

itemInput.addEventListener(‚keypress‘, runEvent); (same as **keydown?**)

itemInput.addEventListener(‚focus‘, runEvent); (whenever a form is **clicked on**)

itemInput.addEventListener(‚blur‘, runEvent); (whenever a form is **clicked out from**)

itemInput.addEventListener(‚cut‘, runEvent); (whenever you **cut out** something from the **form**)

itemInput.addEventListener(‚paste‘, runEvent); (whenever you **paste** something inside the **form**)

itemInput.addEventListener(‚input‘, runEvent); (whenever you do **anything** with the **form**)

select.addEventListener(‚change‘, runEvent); (is connected to the **select variable** => **querySelector(‚select‘)**) (whenever you **choose an new option** from the **select form** the event gets triggered)

form.addEventListener(‚submit‘, runEvent); (whenever you click the **submit button**) (you have to use inside the **runEvent => e.preventDefault();**)