

## Chapter 8 - Events & other DOM properties

Console dir function

Console log shows the element DOM tree

Console dir shows the element as an object with its

properties

tag Name / node Name

Used to read tag name of an element tagName -> only exists for Element nodes nodeName -> defined for any node (text, comment etc.)

InnerHTML and outer HTML

The innerHTML property allows to get the HTML inside the element as a string.

Line only

The outer HTML property contains the full HTML-inner HTML + the element itself.

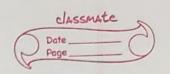
node types we can use node Value or data.

text Content Provides access to the text inside the element: only text, minus all tags.

The hidden property
The "hidden" attribute and the DOM property specifies whether the element is visible or not.



 div hidden > I am hidden < div id = "element" > I can be hidden </div> </script> Attribute methods 17 elem has Attribute (name) -> Method to sheck for existence of an attribute 2. elem get Attribule (name) -> Method used to get the Value of an attribule 3, etcm. Set Attribute (name, value) -> Method used to set the value of an attribute. 47 elem. vemove Attribute (name) - Method to remove the attribute from elem. 5> elem. attributes -> Method to get the collection of all attributes data - \* attributes We can always create custom attributes but the ones starting with "data-" are reserved for programmers use They are available in a property named dataset.

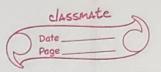


If an element has an attribute named "data-one", its available as element dataset one Insertion methods
We looked at some ways to insert elements in the DOM. Here is another way: let div = document create Element ('div') 11 create div class Name = "alert" 11 set class air inner HTML = " < span > hello < / span > document body append (die) Here were some more insertion methods: 17 node append (e) - append at the end of node 2. node prepend (e) -> Insert at the beginning of node 3> node before (e) - Insert before node 4. node. after (e) -> Insert after node 5, node replace with (e) -> replaces node with the given node. Quick duiz: Try out all these methods with your own webpage.



insert Adjacent HTML / Text / Element

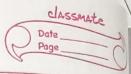
This method is used to insert HTML. The
first parameter is a code word, specifying
where to insert. Must be one of the following: 1- "before begin" - Insert HTML immediately before element 27 "afterbegin" - Insert HTML into element at the beginning 3- "before end" - Insert HTML into element at the end. 4. "afterend" - Insert HTML immediately after element. The second parameter is an HTML String Example: ∠ div id = "dw" > </div> div. insert Adjacent HTML ('before begin', ' Hello );
div. insert Adjacent HTML ('afterend', ' Bye );
</script> The output would be : <pr



Node removal To remove a node, there's a method node remove() let 1d1 = clocument get Element By Id ("id1") id1. removel) ClassName and classList

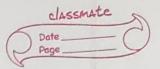
If we assign something to elem className, it replaces
the whole string of classes. Often we want to add/remove/ toggle a single class. 17 elem classlist add/remove ("class") - Adds/removes a class 27 elem classlist toggle ("class") - Adds the class if it doesn't exist, otherwise removes it 3> elem classist contains ("class") - Checks for the given class, returns true / false Set Timeout and set Interval
Set Timeout allows us to run a function once after
the interval of time Syntax of set Timeout is as follows: let timerId = set Timeout (fuction, < delay>, < arg 1>, < arg 2>)

returns a timer Id



clear Timeout is used to cancel the execution (in case we schange our mind). For example: let timer Id = set Timeout (() => a fert ("nercr"), 1000). Clear Timeout (timer Id) is concel the execution set Interval method has a similar syntax as set Timeout: let timerId = set Interval (function, <delay>, <any 17, any) All corguments have the same meaning. But valle set Time out it runs the function not only once, but regularly after the given interval of time. To Stop further calls, we can use clear Interval (timer) Browser Events
An event is a signal that something has happened.
All the DOM nodes generate such signals: Some important DOM events are: Mouse events: click, content menu (right click), mouseover/
mouseout, mousedown/mouseup, mousemove

Keyboard events: keydown and keyup



form element events: submit, focus etc. Document events: DOM Content Loaded

Handling Events

Events can be handled through HTML attributes <input Value = "Hey" onclick = "alert ('hey')" +ype = "buttoon' > Lan be another Is function Events can also be handled through the ordick property elem: onclick = function() {
 alert (" yes")
 }; Note: Adding a handler with JavaScript overwrites the existing handler add Event Listener and remove Event Listener
add Event Listener is used to assign multiple handlers to
an event. element. add Event Listener (event, handler) element remove Eventlistener (event, handler) handler must be the same function object for this to work

The Event Object when an event happens, the browser creaty an event object, puts details into it and passes it as an argument to the handler

elem: onclick = function (event) {

event: type: Event type
event: current Target: Element that handled the event
event: client X / event. client Y: Coordinates of the cursor