**13 April 2020**

**ACCESS TO ELEMENTS OF THE WEBPAGE**

<https://developer.mozilla.org/en-US/docs/Web/api>

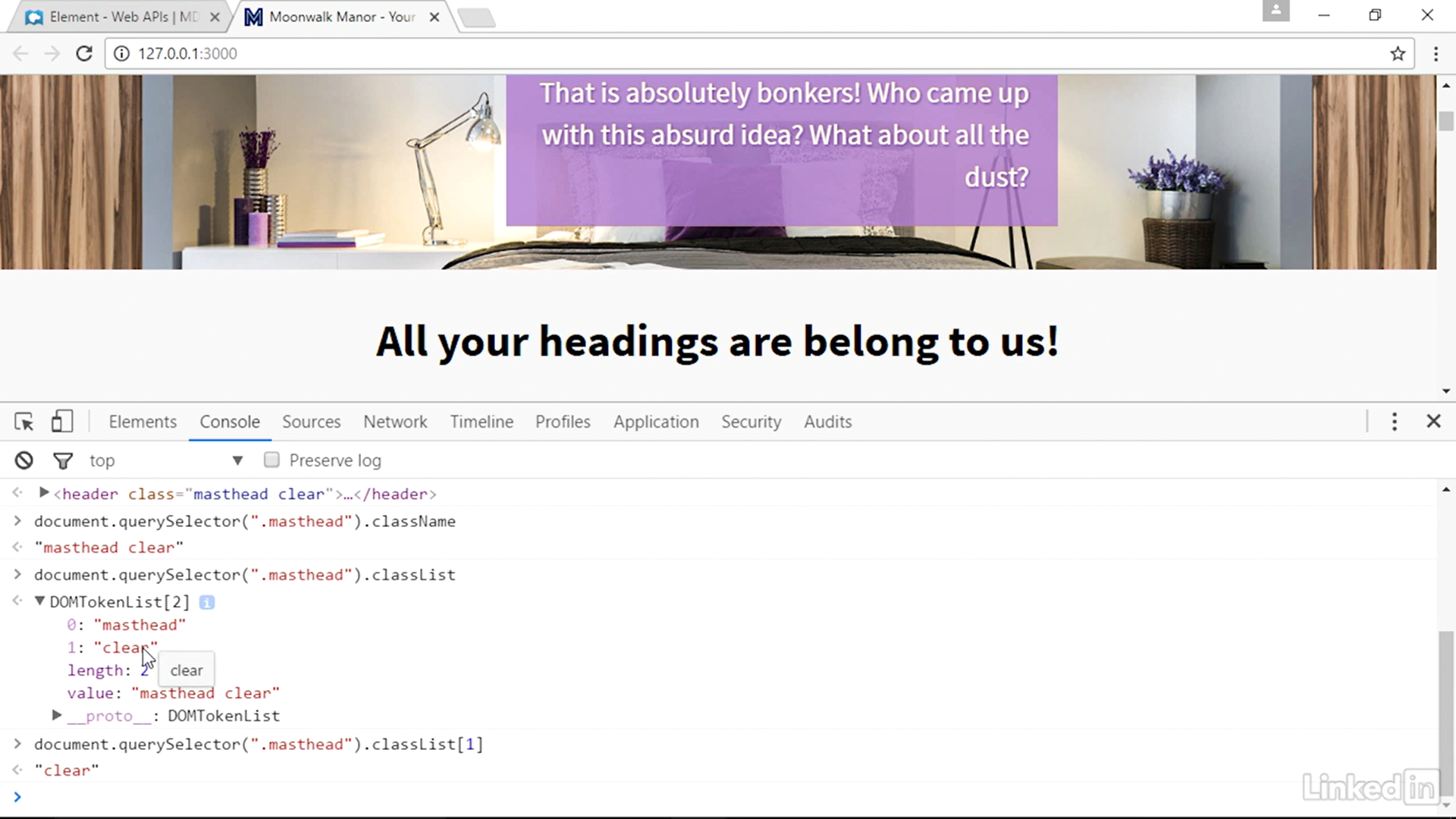
<https://developer.mozilla.org/en-US/docs/Web/API/Element>

by the way – access to element using protractor:

<a class=”classname” manifest=”randomTag”>

|  |
| --- |
| element(by.css('html')).getAttribute('manifest').then((data) => {  elementHtmlValue = data;  }); |

To access document by classname from array of elements we can use such operations as:



**14 April 2020**

Working with element on the web page:

1. Select the element by:

* Const currentElement = document.querySelector(“.classname”)

1. Then we can operate with property currentElement, for example – to see what attributes its have:

* currentElement.attributes – it returns a NamedNodeMap – array with all the attributes of the element.

**ADDING NEW ELEMENTS TO THE DOM :**

Let we have some element what already exists on the webpage with class name “elementBlock”.

*Let ourElement = document.querySelector(“.elementBlock”);*

Example that we have some text: *const someText = “Greetings Text”;*

1. Create the element

For that we using command: *var newElement = document.createElement(“elementName”);*

1. Create the text node that goes inside the element

We can create it by the command: *var newTextNode = document.createTextNode(someText);*

1. Add text node to the element

For that we need to append text node to our newly created element. We can do it by this way:

*newElement.appendChild(newTextNode);*

1. Add the element to the DOM tree

For that we need to append our newElement to already existed element on the webpage:

*ourElement.appendChild(newElement);*

**ADDING CUSTOM CSS TO ELEMENT**

To see what style element have we can with command:

*Document.querySelector(“.classname”.style);*

If css properties was described outside the current page, it will not be shown.

To change a css property in document we can by command:

*Document.querySelector(“.classname”).style.color = “red”;*

*Document.querySelector(“.classname”).style.backgroundColor = “yellow”;*

Notice that **css attributes ARE DIFFERENT then css style operators from document parent commands**, so please check sources for correct usage of that.

To apply a multiple css parameters we need to use command – **cssText**

*Document.querySelector(“.classname”).style.cssText = “color:green; background-color:black;”;*

Another way to add some new css attribute to the element is:

*Document.querySelector(“.classname”).setAttribute(“style”,”padding-left:30px”);*

It allows to add complex string of css attributes in the element as well.

Inline CSS overrides whatever CSS is applied to an element.

**21 April 2020**

Full list of web events: <https://developer.mozilla.org/en-US/docs/Web/Events>

|  |
| --- |
| Function MouseCoordinates(e){  Var h = e.clientX;  Var V = e.clientY;  } |

**23 April 2020**

**Iteration through the arrays of elements**

Selects all elements a what contains attribute href which equals to http and https:

Let linksArray = document.**querySelectorAll**(‘a[href^=”http”]’);

So,

Getting array of elements and checks if the elements in the array have an specified attribute. If some element don’t have it, we assign this attribute to this element:

|  |
| --- |
| For (var i=0;i<array.length;i++) {  If (!array[i].hasAttribute(“target”)) {  Array[i].setAttribute(“target”,”blank”);  }  } |

**11 june 2020**

Array:

The best way to create array is:

Let myArray = [‘one’,’two’,’three’,’four’];

Object:

Const myObject = {

Value1 : ’valueText’,

Value2 : ‘186565’,

….

}

Function with unlimited number of the arguments:

Function summarize (… **all**) {

// iteration:

(for let num of **all**) {

}

}

All arguments will be considered as array.