

Workshop 4: Guided Tour on DOM Scripting

The basic idea in DOM scripting is to first find a reference to an element which you want to manipulate. After that you set the desired attributes or change the content of that element and voila, you can see the page change on the fly.

Use the lecture-material as a reference guide side by side these exercises. It might also be a good idea to save the lines of code you write during the exercises for later use.

Intro


Open `home.html` in a browser and in an editor. Study the structure of the page.

It is important to understand what kind of classes and id's are being used in the web page we want to script. You can use Developer Toolbars Elements-tab to do this, or press `Ctrl+U` in order to view the source for entire page. Note that with `Ctrl+F` you can also search the source code. By typing "id=" to the search field, it is easy to see which id's are being used in the code.

Remember you can utilize Browser Developer Tools (F12) to study the code. Using the inspector (magnifier symbol, top left) you can select any part of the page and see the HTML code defining it. You can then browse through the code in the developer tool window.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head></head>
  <body>
    <div id="wrapper"></div>
    <div id="FooterContainer"></div>
  </body>
</html>
```

JavaScript also lets you explore the structure of the page. Typing 'document' in the console and hitting enter will return you the root node of the page. You can browse onward from this node using



The screenshot shows the Chrome DevTools Console with the 'Elements' tab selected. The document structure is displayed as follows:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>...</head>
  <body>
    <div id="wrapper">
      <div id="branding">...</div>
      <div id="content">...</div>
      <div class="midBox1">...</div>
      <div class="midBox2">...</div>
      <div id="footer">...</div>
    </div>
    <div id="footerContainer">...</div>
  </body>
</html>

```

```
document.getElementsByTagName('h3');
```

```
> document.getElementsByTagName('h3');  
< [ <h3>What We Speak.</h3>, <h3>Blogroll</h3>, ▶<h3>...</h3>, ▶<h3>...</h3>, <h3>Lorem Ipsum passage  
> |
```

```
// Returns the element from index 2
document.getElementsByTagName('h3')[2];
```

```
<h3>
  "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore m
</h3>
```

```
document.getElementsByTagName('h3')[2].innerHTML = "New heading";
```

2. Utilizing variables

It makes scripting easier, if you save a reference to a new variable:

```
// Returns the element from index 0 and saves the reference to a variable targetHeading
var targetHeading = document.getElementsByTagName('h3')[0];

// We don't need to use document.getElementsByTagName again.
targetHeading.innerHTML;
```

3. Finding items based on classname

Finding items based on HTML-tags but usually they return a lot of results to choose from. If we want to narrow down the resultset we can search using classnames.

Try finding items base on classname. Note that these are also arrays with multiple items.

```
document.getElementsByClassName('cat003');
> document.getElementsByClassName('cat003');
< [ ▶<li class="cat003">...</li>, ▶<li class="cat003">...</li>, ▶<li class="cat003">...</li>, ▶<li class="cat003">...</li>,
  ▶<li class="cat003">...</li>, ▶<li class="cat003">...</li>, ▶<li class="cat003">...</li>]
```

You can get a reference to entire sideBar with the following:

```
// Note this is an array
document.getElementsByClassName('sideBarItem');

// Try this
document.getElementsByClassName('sideBarItem')[0];

var myVar = document.getElementsByClassName('sideBarItem');

myVar[0];

myVar[0].innerHTML = "Write something here";
```

4. Finding items based on id

Finding items based on their id, makes the targeting specific items most accurate. Unlike other functions, document.getElementById - function returns only one result. Try these:

```
document.getElementById('contant');
document.getElementById('footer');
```

5. Basic DOM Manipulations: Content

Like mentioned in the lecture, there are many ways to manipulate the content of an HTML-element. Let's look at a few.

Previously we used the following line of code to get the reference to it. After that, changing the content of the tag is easy.

```
// Returns the element from index 0 and saves the reference to a variable targetHeading
var targetHeading = document.getElementsByTagName('h3')[0];

// Change the content of the tag
targetHeading.innerHTML = "New heading";

// Same thing but without using the variable could be achieved like this
document.getElementsByTagName('h3')[0].innerHTML = "New heading";

// Setting the innerHTML -field always overwrites the previous content. However, we can add text as well using
the "+=" notation:

targetHeading.innerHTML += "Add this to the end of Heading";

// You can also use HTML tags
document.getElementsByTagName('h3')[0].innerHTML = "<em>New heading</em>";
```

6. Adding style

Typical DOM scripting tasks include adding or removing style attributes to page elements. Try out the following:

```
//Changing the paragraphs styles of the page
var p = document.getElementsByTagName('p');
p[0].style.color = "green";

// Hiding items
p[2].style.display = "none";
```

7. Creating content with functions

// First create a new element and set any attributes to it

```
var uusi = document.createElement('p')
uusi.innerHTML = "New content!";
uusi.setAttribute('id', 'newData');
```

// This results to a following tag

```
<p id="newData">New content!</p>
```

// Finally attach the new element to the bottom of the page

```
document.body.appendChild(uusi);
```

// OR to a desired place on page

```
var newPlace = document.getElementsByTagName('p');  
newPlace[1].appendChild(uusi);
```

8. Creating table-content with functions

```
var table = document.createElement('table');  
table.setAttribute('border', '1');  
for (var i = 1; i < 4; i++){  
    var tr = document.createElement('tr');  
  
    var td1 = document.createElement('td');  
    var td2 = document.createElement('td');  
  
    var text1 = document.createTextNode('Text1');  
    var text2 = document.createTextNode('Text2');  
  
    td1.appendChild(text1);  
    td2.appendChild(text2);  
    tr.appendChild(td1);  
    tr.appendChild(td2);  
  
    table.appendChild(tr);  
}  
document.body.appendChild(table);
```