Dom Manipulation and Events

## Simple DOM manipulation and Event handling

Add three divs to an HTML page, each with a unique id

**a)** When the page initially is loaded all divs should be given a colour of your choice

Hints: use document.getElementsByTagName() (will return an array of Dom Nodes) and element.style.backgroundColor

**b)** Add a button, and assign a click handler to the button. When the button is clicked each div should be given a unique colour.

Hints: use document.getElementById(..) to get the individual div’s

## Event Bubbling and event arguments

In a new HTML page add two div’s, each with a unique id

Style both id’s with this class:

.mydiv {width: 25px; height: 25px;background-color:yellow; margin:1px;}

**a)**

Add a click handler to each id and write code, so when clicked, each div will write to the console "Hi from idOfTheDiv".

**b)** Now, using cut and paste, add 10 more divs (still each with a unique id).

*Adding an event handler to each div would be cumbersome and non-efficient (what if you had been asked to add 100 or thousands of new id’s).*

We will use JavaScript's event bubbling mechanism to solve this with only a single event handler.

Add a new div (without the myDiv class) with the id="outer" around all our div’s and assign a single event handler (with the event argument) to this div.

Implement code to write to the console:

* The value of id pointed to by ***this***
* The value of id pointed to by the target property (which you get from the event argument)
* Make SURE you understand what happened here

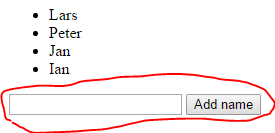
**c)**

Add an empty paragraph tag, with an id, to your HTML and change the code for both exercises above, so that output is not written to the console, but into this paragraph.

Hint: Use the innerText property of an Element

## Using map to create list’s (ul’s, tables etc.)

**a)** Yesterday you used the array types map and join methods to create a UL with a number of names (ex- 4).

Use this function+DOM manipulation to insert the UL somewhere in an HTML page.

**b)** Create a *FORM* with an *input* field + a *submit* button as sketched here.

Write the necessary code to add the new name to the array of names, and regenerated the <ul> with the updated list.

**Hint:** You will probably see, very shortly that the name is added, the screen flickers and it's gone again.

The problem is, that it submit submits to the server, so the full page is reloaded.

To prevent this you can call .preventDefault() on the event argument which will prevent the default behaviour (submit) and it should work.

You can, also use preventDefault() to prevent a link from actually forwarding to the link-address.

**c)** Add two more buttons to the form with the text: “remove first” and “remove last”. Implement the behaviour inspired by how you solved part-b.

## Using map and filter to create dynamic table-rows

Yesterday you created a number of filter functions using the array given below

var cars = [

{ id: 1, year: 1997, make: 'Ford', model: 'E350', price: 3000 },

{ id: 2, year: 1999, make: 'Chevy', model: 'Venture', price: 4900 },

{ id: 3, year: 2000, make: 'Chevy', model: 'Venture', price: 5000 },

{ id: 4, year: 1996, make: 'Jeep', model: 'Grand Cherokee', price: 4799 },

{ id: 5, year: 2005, make: 'Volvo', model: 'V70', price: 44799 }

];

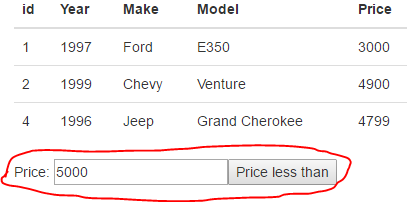
Use this array, with map and filter as outlined in the following

#### Use the map method to create a table with **all** data

Write a function which, given this array, will return an HTML string with the array formatted as a Table as sketched in this figure (styled with the Bootstrap class table):

#### Use the filter method, to filter out items from the list

Add an input field and a button (provide each with an id), as sketched in this figure.



Write the necessary code so when the button is clicked, and the input field contains a number:

* A new filtered array is created having only prices < than the value provided
* This array is passed to the function implemented in the previous step, and the DOM is updated with this new table.

## Implement a simple calculator

Implement a calculator with the functionality given below.

a. When a button is clicked the value should be appended to the display div (middle figure)

b. When the equal sign is clicked, the result of the calculation must be presented. (last figure)

|  |  |  |
| --- | --- | --- |
|  |  |  |

c. Use the HTML and Style given below to get started

|  |  |  |
| --- | --- | --- |
| <div id="container">  <div id="display" class="t4"></div>  <div id="buttons">  <div class="t1">7</div>  <div class="t1">8</div>  <div class="t1">9</div>  <div class="t1">/</div>  <div class="t1">4</div>  <div class="t1">5</div>  <div class="t1">6</div>  <div class="t1">\*</div>  <div class="t1">1</div>  <div class="t1">2</div>  <div class="t1">3</div>  <div class="t1">-</div>  <div class="t1">0</div>  <div class="t1">.</div>  <div class="t1">+</div>  <div id="calculate"class="t1">=</div>  </div>  </div> |  | <style>  #container{  width: 226px;  height: 274px;  border: 3px solid darkblue;  margin: auto  }  .t1, .t4{  border: 1px solid black;  height: 48px;  margin-left: 3px;  margin-top: 3px;  display: inline-block;  text-align: center;  line-height: 48px;  }  .t1{  width: 48px;  }  .t4{  width: 219px;  }  </style> |

Hints (and these are just hints, there are many ways to solve this problem)

* Assign a *single* click event handler to the div with the *buttons* id to handle anything but the “=” button.
* Use your knowledge about event bubbling and the events target property to get the text in the div that was clicked (via the innerText property).
* Assign a new event handler to the div with the “*calculate*” id. Use the event arguments .stopPropagation() method, to prevent this event from bubbling up to your “outer” event handler.
* Use the indexOff(..) method to test whether a string contains \*, /, + or -.
* Use split(..) (with one of the four operators) to get the two numeric values and use Number(..) to convert a numeric string into a number before you do the calculation.

### Dynamic UI manipulation using data obtained via fetch

This is marked as () since it has been a hard week. But you all need to have completed this before you start with CA-1.

It’s placed in a [separate exercise](https://docs.google.com/document/d/1eR7qJhV_fZH2VbqmyA6YWhzDCInvAHwpEeU-R3UvPAw/edit?usp=sharing) in order to add it to our list of guidelines.