

Dom Manipulation and Events

1) 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 color of your choice

Hints: use `document.getElementsByTagName()` 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 color.

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

2) 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 eventhandler 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

3)

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

4)

a) Yesterday you used the arrays `map` and `join` methods to create an UL with a number of names (ex- 4).

Use this function+DOM man. 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 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 behavior (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 of how solved part-b.

- Lars
- Peter
- Jan
- Ian



5)

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 }  
];
```

a)

Write a function that, 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`):

id	Year	Make	Model	Price
1	1997	Ford	E350	3000
2	1999	Chevy	Venture	4900
3	2000	Chevy	Venture	5000
4	1996	Jeep	Grand Cherokee	4799
5	2005	Volvo	V70	44799

b)

Add an input field and a button, each with an id as sketched here.

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

- A new filtered array is created having only prices < than the value provided
- This array is passed to the function implemented in a), and the DOM is updated with this new array.

id	Year	Make	Model	Price
1	1997	Ford	E350	3000
2	1999	Chevy	Venture	4900
4	1996	Jeep	Grand Cherokee	4799

Price:

6) Implement a calculator with functionality as given below.

- When a button is clicked the value should be appended to the display div (middle figure)
- When the equal sign is clicked the result of the calculation should be displayed. (last figure)
- Use the HTML and Style given below to get started

7	8	9	/
4	5	6	*
1	2	3	-
0	.	+	=

25.5+4.5			
7	8	9	/
4	5	6	*
1	2	3	-
0	.	+	=

30			
7	8	9	/
4	5	6	*
1	2	3	-
0	.	+	=

```
<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 this is just hints, there are many ways to solve this problem)

- Assign a *single* click eventhandler 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 `indexOf(..)` 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.