

# Advanced Web Programming

## Lab3 - JS

### event programming + DOM manipulation

**Goals:** The aim of the exercises is to meet with the methods of event handling in the JS environment. We will get acquainted with modern methods of event management and we will learn how to handle typical events. In practice, we will learn what a bubble model is and we will get acquainted with the details of the event itself.

In the second part, we will practice methods to manipulate the DOM model. We learn about some modification strategies: beginning from modifying the content of an existing element through hiding the content to changing the model itself. We will dynamically add or remove DOM elements.

## Section 1. Event Handling

### Understanding Events and Event Handlers

Events are actions that happen when the user or browser manipulates a page. They play an important role as they can cause elements of a web page to change dynamically. For example, when the browser finishes loading a document, then a load event occurred. If a user clicks a button on a page, then a click event has happened. Many events can happen once, multiple times, or never. You also may not know when an event will happen, especially if it is user generated.

In these scenarios, you need an **event handler** to detect when an event happens. This way, you can set up code to react to events as they happen on the fly.

JavaScript provides an event handler in the form of the **addEventListener()** method. This handler can be attached to a specific HTML element you wish to monitor events for, and the element can have more than one handler attached.

***target.addEventListener(event, function, useCapture)***

- **target:** the HTML element you wish to add your event handler to. This element exists as part of the Document Object Model (DOM).
- **event:** a string that specifies the name of the event.
- **function:** specifies the function to run when the event is detected. This is the magic that can allow your web pages to change dynamically.
- **useCapture:** an optional Boolean value (true or false) that specifies whether the event should be executed in the capturing or bubbling phase. In the case of

nested HTML elements (such as an img within a div) with attached event handlers, this value determines which event gets executed first. By default, it's set to false which means that the innermost HTML event handler is executed first (bubbling phase).

**Exercise 1.** In Catalog Example1 you find html, css and js files. HTML and CSS are ready to use, js file now is empty. Write js code to show or hide (depends on current state) message block context when you click button.

[solution with explanation](#)

Let's see how to fix the problem. That example is to show you addEventListener() in action. When a user clicks the button, a message is displayed. Another button click hides the message. Here's the relevant JavaScript:

```
let button = document.querySelector('#button');
let msg = document.querySelector('#message');
button.addEventListener('click', ()=>{
    msg.classList.toggle('reveal');
})
```

We can access an HTML element's corresponding DOM node in JavaScript via the querySelector function: `document.querySelector('css selector');`

Function returns the first element that matches the given CSS selector.

In the first line we assign element with id = "button" to JS Object which name – button.

The same in the second line.

Notice. via the querySelectorAll function: `document.querySelectorAll('css selector');` returns all elements that match the given CSS selector.

Line nr 3 is the most important. I use addEventListener() to bind event click from button object to event handling represent by function.

Going by the syntax shown previously for addEventListener():

- **target:** HTML element with id='button'
- **function:** anonymous (arrow) function that sets up code necessary to reveal/hide the message
- **useCapture:** left to default value of false

My function is able to reveal/hide the message by adding/removing a CSS class called "reveal" which changes the message element's visibility. I use properties classList for object message.

The Element.classList is a read-only property that returns a live DOMTokenList collection of the class attributes of the element. This can then be used to manipulate the class list. Using classList is a convenient alternative to accessing an element's list of classes as a space-delimited string via element.className. More about classList -> [https://www.w3schools.com/jsref/prop\\_element\\_classlist.asp](https://www.w3schools.com/jsref/prop_element_classlist.asp)

---

Exercise 2. In Catalog Example1 in file example1.html you can find attribute defer in <script> line. Delete this attribute and check what happens. Is app working properly? Use developer tools and section Console to verify it.

Exercise 3. In Catalog Example2 you can find some files. Write the code that will allow you to implement the scenario described below.

Switching between two photos. When you start the page displays a photo of the mountains with a red border. When you press the button "click me", the photo will turn into a sea photo with a blue border. Another press is to return to the version with mountains, etc.

Note how js is linked to html. It is OK? If not, correct it.

### Removing Event Handlers

If for some reason you no longer want an event handler to activate, here's how to remove it:

```
target.removeEventListener(event, function, useCapture);
```

The parameters are the same as addEventListener().

Sometimes we would like to change for example text in existing html element. We can use *innerText* or *innerHTML* to do it.

Exercise 4. In Catalog Example3 you can find some files. We have three button. Initially button test is doing nothing. Write a script that adds or removes for test button click event handling. When you click Add button you bind test button with click events and Increment Function. Pressing the test button increments the counter. The counter value will be displayed in the paragraph section. When you click Delete button you unbind test button. You delete event handling from test button. Unpinning the event resets the counter

Exercise 4A. In Example3 add check box button. When checkbox is selected bind new function to click event for Add button. Let the second function display alert message for example “Greeting from second function”. When check box is unchecked second function have to be unbind.

For most event types, handlers registered on nodes with children will also receive events that happen in the children. If a button inside a paragraph is clicked, event handlers on the paragraph will also see the click event. But if both the paragraph and the button have a handler, the more specific handler—the one on the button—gets to go first. The event is said to propagate outward, from the node where it happened to that node’s parent node and on to the root of the document. Finally, after all handlers registered on a specific node have had their turn, handlers registered on the whole window get a chance to respond to the event.

At any point, an event handler can call the stopPropagation method on the event object to prevent handlers further up from receiving the event. This can be useful when, for example, you have a button inside another clickable element and you don’t want clicks on the button to activate the outer element’s click behavior.

Exercise 5. In Catalog Example4 you can find some files. In html file you find 3 html elements: section, div and p. Let for each of them clicking in its area results in displaying a message (using the alert function) with information about event source. For example when I click in div section I see alert with message – “div was clicked”.

Check what happens when you click for example in p area. How many messages do you see?

Exercise 6. In Example4 change order of displayed messages.

Exercise 7. In Example4 for div tag you don’t want to display messages from child. How to stop display this messages?

## Default actions

Many events have a default action associated with them. If you click a link, you will be taken to the link's target. If you press the down arrow, the browser will scroll the page down. If you right-click, you'll get a context menu. And so on.

For most types of events, the JavaScript event handlers are called before the default behavior takes place. If the handler doesn't want this normal behavior to happen, typically because it has already taken care of handling the event, it can call the `preventDefault` method on the event object.

**Exercise 8.** In [Catalog Example5](#) you can find some files. In [html](#) file you find [input](#) element and [button](#) switch. Write something to [input](#) and [click](#) button. Let [alert](#) messages display text from [input](#). As you see when you [click](#) button [input](#) are reset. This is because [submit](#) send forces the page to reload. You can switch off reset. Write code to do it.

Sometimes we may want to know more information about the event, such as what element was clicked. In this situation, we need to pass in an event parameter to our function. This example shows how you may obtain the element's id:

```
button.addEventListener('click', (e)=>{  
  console.log(e.target.id)  
})
```

Here the event parameter is a variable named `e` but it can be easily called anything else such as "event". This parameter is an object which contains various information about the event such as the target id. This object holds additional information about the event. For example, if we want to know which mouse button was pressed, we can look at the event object's `button` property.

```
<button>Click me any way you want</button>
```

```
<script>
```

```
let button = document.querySelector("button");
```

```
button.addEventListener("mousedown", event => {
```

```
  if (event.button == 0) {
```

```
    console.log("Left button");
```

```
  } else if (event.button == 1) {
```

```

    console.log("Middle button");
  } else if (event.button == 2) {
    console.log("Right button");
  }
});
</script>

```

The information stored in an event object differs per type of event. The object's type property always holds a string identifying the event (such as "click" or "mousedown").

More about Event object -> [https://www.w3schools.com/jsref/obj\\_events.asp](https://www.w3schools.com/jsref/obj_events.asp)

Exercise 9. In Catalog Example6 you can find some files. In html file you find input elements and some div elements. Write code which give you information which div are you select by click on it. Display this information by alert message. For example when you click the second div let alert display nrb 2. Using console.log display in console all information about event object.

In input forms display current position for mouse. When you move mouse new position will displayed in input ( x position in one, y position in second one).

## Adding and Removing HTML content

You can add and remove HTML content in the DOM. For that, we are going to look at three methods and one property.

Let's start with the innerHTML property because it is the easiest way of adding and removing HTML content. innerHTML can either be used to get or set HTML content. But be careful when using innerHTML to set HTML content, because it removes the HTML content that is inside the element and adds the new one.

```
document.getElementById('division').innerHTML = `<ul> <li>Angular</li> <li>Vue</li> <li>React</li></ul>`;
```

If you later run the code, you will notice that everything else in the div with the id of division will disappear, and the list will be added.

We can use the methods: createElement(), createTextNode(), and appendChild() to solve this problem.

- createElement is used to create a new HTML element.
- createTextNode used to create a text node,
- appendChild is used to append a new element into a parent element.

//first we create a new p element using the createElement

```
newElement = document.createElement('p');
```

/\* then we create a new text node and append the text node to the element created\*/

```
let text = document.createTextNode('Text Added!');
```

```
newElement.appendChild(text);
```

/\* then we append the new element with the text node into the div with the id division.\*/

```
document.getElementById('division').appendChild(newElement);
```

There is also a method called removeChild used to remove HTML elements.

// first we get the element we want to remove

```
let toBeRemoved = document.getElementById('head');
```

// then we get the parent node, using the parentNode property

```
let parent = toBeRemoved.parentNode;
```

/\* then we use the removeChild method, with the element to be removed as a parameter.\*/

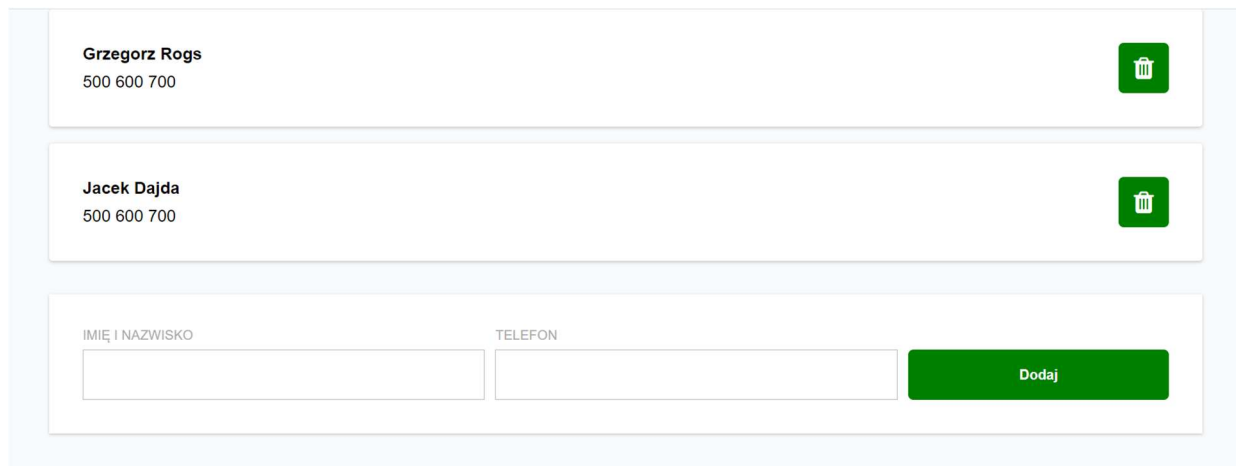
```
parent.removeChild(toBeRemoved);
```

So first we get the element that we want to remove, and then we get its parent node. Then we called the method removeChild to remove the element.



Exercise 10. In Catalog Example7 you can find some files. In html file you find three button and input element. Give any number in input click add button and now value from input will be added to list. Button del delete last element from list. Button reset clear all list. Write the appropriate code.

**Exercise 11.** In Catalog Example8 you can find some files. In html file you find two select options. Write a script which set color of header or selection based on color selected from list.

**Exercise 12.** Write a script in which, using the Add button, will add a new entry to the phone book (name, telephone number). We provide the data using the form. The added item also includes a button (in the form of a trash icon) that allows you to delete items from the address book. An example below.



The image shows a web interface for a phone book. It contains two existing entries, each with a name, a phone number, and a delete button (trash icon). Below these is a form to add a new entry, consisting of two input fields for 'IMIĘ I NAZWISKO' and 'TELEFON', and a green 'Dodaj' button.

<b>Grzegorz Rogs</b> 500 600 700	
<b>Jacek Dajda</b> 500 600 700	

IMIĘ I NAZWISKO	TELEFON	
<input type="text"/>	<input type="text"/>	<input type="button" value="Dodaj"/>