Assignment 5

Recommended readings:

- Lecture slides as starting literature
- MDN Links within slides for details
- Event Propagation

Note: All exercises must be solved with plain JavaScript and CSS not using additional libraries or frameworks.

Exercise 1 – Events

What is event bubbling and event capturing? How can you specify what method to use? How can you stop the propagation of events? What is the difference between `this`, `event.target` and `event.currentTarget`? What happens if parent elements have event handlers attached? What happens if parent and child elements have event handlers attached?

a) Extend the example in `ex1/events.html` for demonstrating bubbling, capturing and `stopPropagation` with event handlers defined on the outer and additionally on the inner div. You may create multiple files for demonstrating the various options.
b) Now the event handler should only do the console output if it is called with a second parameter `clicked(e,true)`. How can you pass parameters to event handlers in JS?
c) The script in the file `ex1/events2.html` should display the first name and last name of the person as content of the corresponding div, when the div is clicked. Unfortunately, it does not work. Explain why this happens and provide two solutions:
    a. Using the `bind()` method [MDN].
    b. Using an arrow function for calling the actual event handler [MDN].

Exercise 2 – Custom Events

Completing the templates provided in the `ex2` folder, implement a simple stopwatch that can be started, stopped and reset:

- Elapsed time (minutes, seconds, milliseconds) should be in the format MM:SS:MIS
- The timer can be controlled via three buttons (‘`click`’ events):
    - `Start`: starts time measurement, beginning from 0 or continuing the timer from last stop
    - `Stop`: stops time measurement, halting the timer at currently elapsed time

  - `Reset`: resets timer to 0 and aborts running time measurement
- Additionally, the timer can be controlled via keyboard input: '`s`' for `Start`, '`t`' for `Stop` and '`r`' for `Reset`. Dispatch custom events for all keypress, while listening for them on the timer output element (`timertext`).
- Create two additional custom events triggered by the '`+`' and '`-`' keys: these events should increase/decrease the timer by 5000 milliseconds, which should be passed via event payload data to the corresponding functions in the `Stopwatch` class. Be sure to allow for changing the timer even if it is not running ('`Stop`' mode) and also handle timer overflows that could occur when decreasing it beyond 0.

Feel free to add variables and methods as you see fit to the Stopwatch class (`Stopwatch.js`) and `site.js` file. Recommended readings:

- Custom Events [MDN]
- `setTimeout()` [MDN] / `setInterval()` [MDN]
- Document: keydown event [MDN]

Exercise 3 – Modules

Using the Image and Video classes created exercise 4.5, alter the module *multimedia* that exports both of these classes (see `ex3/multimedia.js`). The class Image should be the default export.

Write a test script (`ex3/site.js`) that imports the default export, imports the Video export as Movie and creates some Video and Image objects.

Exercise 4 – JSON Image Gallery

Extend the picture gallery from Assignment 3 by implementing the functionality of retrieving picture URLs from a server-side JSON file. An altered version of the gallery can be found in the *ex4* folder. Your task is to complete the object `ImageLoader`:

- Fully implement the `load` function of `ImageLoader`.
- Retrieve `gallery.json` via `XMLHttpRequest`.
- Parse the retrieved results and create appropriate Image elements for every contained image.
- Dynamically add the elements to `<div class="gallery">`.
- Initialize the gallery afterwards using the function `initGallery`.

Hint: Keep in mind that you need to host your files via an HTTP server, e.g. XAMPP on `localhost`, else HTTP requests will not work.

Exercise 5 – JSON Gallery Library

Extend the picture gallery from Assignment 3 even further by creating a Gallery library class `JsonGallery` (see *ex5* folder), which constructs a gallery within a `div` element passed to its constructor. Complete `JsonGallery.js` by adapting your previous code to fit the JS class paradigm and additionally take the following into account:

- Only edit the JavaScript files for this exercise – DOM elements should be created dynamically, e.g. use `document.createElement`/`element.innerHTML`.
- Again, image URLs must be retrieved from the server but this time use the `fetch` API for client-server communication.
- Also, be sure to use *Promises* when requesting resources.
- The gallery is now partitioned into two pages (`gallery_pg1.json`, `gallery_pg2.json`), which should be requestable one at a time in order to replace the images currently in the gallery. For this, you should create additional gallery page navigation buttons (e.g.: 'previous page'/'next page') on the page displaying the small images. Pressing one of these buttons should start a request for a new JSON file and setup the gallery according to the retrieved results.

Hint: Keep in mind that you need to host your files via an HTTP server, e.g. XAMPP on `localhost`, else HTTP requests will not work.