

# Aplikacje Internetowe 1

Sprawozdanie – Lab 3 – JS i DOM na przykładzie Todo List

Data laboratorium: 07.11.2021

Autor: Lewicki Maciej

Grupa: 42

Wersja: 2021/2022-lato

## Wprowadzenie

W poniższym sprawozdaniu opisane są laboratoria które polegały na stworzeniu aplikacji internetowej z rodzaju TODO-LIST.

Stworzona przeze mnie strona zawiera dodatkowy CSS umożliwiający zmianę funkcji wyszukiwania przy wyświetlaniu mobilnym.

Dla ekranów poniżej 800px szerokości strona wyświetla jeden pasek wyszukiwania, który filtruje wszystkie kolumny.

Dodatkowo przy niepoprawnym wprowadzeniu danych pojawia się element informujący nas o błędzie walidacji.

## Strona HTML ze wszystkimi elementami

Należy przedstawić HTML ze wszystkimi wymaganymi elementami, tj. pole wyszukiwarki, lista, pole dodawania, przycisk usuwania. 1 punkt.

Code...	Task...	dd . mm . rrrr	Q	Clear
A24	Finish the task list, duh...	4.11.2021	Edit	×
A24	Implement search bar	4.11.2021	Edit	×
B01	Fix site for mobile displays	5.11.2021	Edit	×
T01	This is a long task. Looooong task. Very very long task. Uncomfortably long task. Is this how u spell uncomfortably? I'm not sure. Anyway long task!	10.11.2021	Edit	×
T02	This is a task without a date		Edit	×

Code...	Task...	dd . mm . rrrr	Add	Clear
---------	---------	----------------	-----	-------

NAGŁÓWEK:



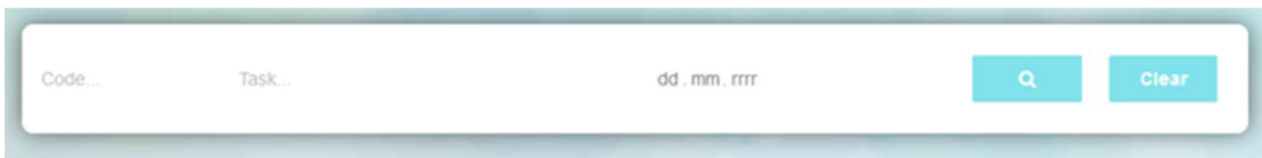
CSS:

```
h1 {  
  mix-blend-mode: screen;  
  background-color: white;  
  font-family: 'Lato', sans-serif;  
  font-size: 100px;  
  font-weight: bold;  
  letter-spacing: -1px;  
  line-height: 1.2;  
  text-align: center;  
}
```

HTML:

```
<h1>TODO List</h1>
```

PASEK WYSZUKIWANIA:



CSS:

```
.TODO table {  
  border-radius: 10px;  
  margin: auto;  
  margin-bottom: 50px;  
  margin-top: 50px;  
  border-collapse: collapse;  
  table-layout: fixed;  
  font-size: 0.9em;  
  width: 100%;  
  height: 100%;  
  background-color: white;  
  min-width: 200px;  
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.55);  
}
```

HTML:

```
<div class="TODO">  
  <h1>TODO List</h1>  
  <table>  
    <tr class="searchRow">  
      <td class="col1"><input id="codeSearch" class="listInput" type="text" placeholder="Code..."></td>  
      <td class="col2"><input id="taskSearch" class="listInput" type="text" placeholder="Task..."></td>  
      <td class="col3"><input id="dateSearch" class="listInput" type="date" placeholder="dd-mm-yyyy"></td>  
      <td class="col4">  
        <span id="searchBtn" onclick="filterElements()" class="mainButton">  
          <i class="fa fa-search"></i>  
        </span>  
        <span id="clearSearchBtn" onclick="clearSearch()" class="mainButton">Clear</span>  
      </td>  
    </tr>  
  </table>  
</div>
```

PRZYCISKI SEARCH ORAZ CLEAR SEARCH:



CSS:

```
.mainButton {
  color: white;
  padding: 2px;
  display: flex;
  background: #82e2eb;
  border-radius: 2px;
  float: left;
  margin: 5%;
  width: 40%;
  height: 40px;
  box-sizing: border-box;
  align-items: center;
  justify-content: center;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  transition: 0.2s;
}
```

HTML:

```
<span id="searchBtn" onclick="filterElements()" class="mainButton">
  <i class="fa fa-search"></i>
</span>
<span id="clearSearchBtn" onclick="clearSearch()" class="mainButton">Clear</span>
```

TABELA ZADAŃ:

A24	Finish the task list, duh...	4.11.2021	Edit	✕
A24	Implement search bar	4.11.2021	Edit	✕
B01	Fix site for mobile displays	5.11.2021	Edit	✕
T01	This is a long task. Looooong task. Very very long task. Uncomfortably long task. Is this how u spell uncomfortably? I'm not sure. Anyway long task!	10.11.2021	Edit	✕
T02	This is a task without a date		Edit	✕

CSS:

```
.TODO table {
  border-radius: 10px;
  margin: auto;
  margin-bottom: 50px;
  margin-top: 50px;
  border-collapse: collapse;
  table-layout: fixed;
  font-size: 0.9em;
  width: 100%;
  height: 100%;
  background-color: white;
  min-width: 200px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.55);
}
```

HTML:

```
<table id="mainTable">
</table>
```

PRZYCISKI EDIT ORAZ REMOVE:

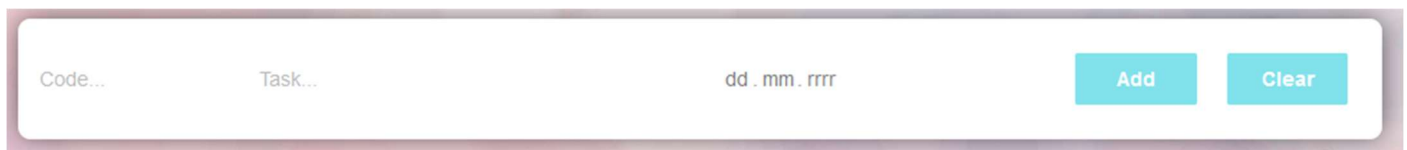


CSS:

```
.listButton {
  padding: 2px;
  display: flex;
  background: #e990a3;
  color: white;
  border-radius: 2px;
  float: left;
  margin: 5%;
  width: 40%;
  height: 30px;
  box-sizing: border-box;
  align-items: center;
  justify-content: center;
  font-size: 14px;
  cursor: pointer;
  transition: 0.2s;
}
```

HTML: (brak – tworzony dynamicznie w JS)

PASEK DODAWANIA:



CSS:

```
.TODO table {
  border-radius: 10px;
  margin: auto;
  margin-bottom: 50px;
  margin-top: 50px;
  border-collapse: collapse;
  table-layout: fixed;
  font-size: 0.9em;
  width: 100%;
  height: 100%;
  background-color: white;
  min-width: 200px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.55);
}
```

HTML:

```
<table>
<tr id="addNewRow">
  <td class="col1"><input id="inputCode" class="listInput" type="text" placeholder="Code..."></td>
  <td class="col2"><input id="inputTask" class="listInput" type="text" placeholder="Task..."></td>
  <td class="col3"><input id="inputDate" class="listInput" type="date" placeholder="dd-mm-yyyy"></td>
  <td class="col4">
    <span id="addButton" onclick="addElement()" class="mainButton">Add</span>
    <span onclick="clearInput()" class="mainButton">Clear</span>
  </td>
</tr>
</table>
```

PRZYCISKI ADD ORAZ CLEAR ADD/EDIT:



CSS:

```
.mainButton {
  color: white;
  padding: 2px;
  display: flex;
  background: #82e2eb;
  border-radius: 2px;
  float: left;
  margin: 5%;
  width: 40%;
  height: 40px;
  box-sizing: border-box;
  align-items: center;
  justify-content: center;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  transition: 0.2s;
}
```

HTML:

```
<span id="addButton" onclick="addElement()" class="mainButton">Add</span>
<span onclick="clearInput()" class="mainButton">Clear</span>
```

ALERT:

Error: Invalid code input, code has to comprise of one letter and two digits.  
Error: Invalid task input, task has to be between 3 and 255 characters long (inclusive).



```
.alert {
  padding: 20px;
  position: fixed;
  border-radius: 10px;
  left: 50%;
  transform: translate(-50%, -50%);
  margin: 0 auto;
  bottom: 0px;
  background-color: #e990a3;
  color: rgb(172, 52, 52);
  animation: fadeOut 0.3s ease-out;
  opacity: 0;
  overflow: hidden;
}

.alert.active {
  opacity: 1;
  animation: fadeIn 0.3s ease-out;
}

@keyframes fadeIn {
  0% {
    opacity: 0;
  }
  100% {
    opacity: 1;
  }
}

@keyframes fadeOut {
  0% {
    opacity: 1;
  }
  100% {
    opacity: 0;
  }
}

<div class="alert" id="alert" style="display: none">
  <span class="closebtn" onclick="alertClose()"><i class="fa fa-times"></i></span>
  <span id="alertText"></span>
</div>
```

## Wczytywanie i zapisywanie listy z/do Local Storage

Funkcja „updateLocalStorage()” zapisuje wszystkie zadania do Local Storage.

```
function updateLocalStorage()
{
    // We do this everytime we make any change, this is incredibly inefficient
    // but no user will ever create enough tasks for it to matter

    var tasks = []
    for (let i = 0; i<mainTable.children.length; i++)
    {
        var task = new Task()
        task.code = mainTable.children[i].children[0].textContent
        task.task = mainTable.children[i].children[1].textContent
        task.date = mainTable.children[i].children[2].id // json date is stored in id
        tasks.push(task)
    }
    localStorage.setItem("tasks", JSON.stringify(tasks));
}
```

Wykorzystuje do tego klasę Task:

Zadania zapisane w klasie Task są następnie dodawane do listy.

Lista jest serializowana za pomocą JSON i dodawana do Local Storage jako jeden element.

Funkcja „restoreRows()” przywraca zadania z Local Storage

```
function restoreRows()
{
    // Go through all the tasks in storage and create rows
    var tasks = JSON.parse(localStorage.getItem("tasks"));
    for(let i = 0; i<tasks.length; i++)
    {
        mainTable.appendChild(createRow(tasks[i].code, tasks[i].task, tasks[i].date));
    }
}
```

Wykorzystuje do tego funkcję „createRow()” która jest opisana w następnym punkcie



## Dodawanie pozycji listy

Funkcja „createRow()” tworzy elementy DOM wymagane do dodania nowego zadania i wypełnia je danymi podanymi w argumentach.

```
function createRow(code, task, date)
{
    var row = document.createElement("tr");
    row.className = "elementRow";

    // Code cell
    var col1 = document.createElement("td");
    col1.className = "col1";
    col1.appendChild(document.createTextNode(code.toUpperCase()));
    row.appendChild(col1);

    // Task cell
    var col2 = document.createElement("td");
    col2.className = "col2";
    col2.appendChild(document.createTextNode(task));
    row.appendChild(col2);

    // Date cell
    var col3 = document.createElement("td");
    col3.className = "col3";
    col3.id = date === "" ? "" : (new Date(date)).toJSON() // The id is used for locale independant date storage
    var date = date === "" ? "" : new Date(date).toLocaleDateString();
    col3.appendChild(document.createTextNode(date));
    row.appendChild(col3);

    // Buttons
    var col4 = document.createElement("td");
    col4.className = "col4";
    var span1 = document.createElement("span");
    var span2 = document.createElement("span");
    var icon = document.createElement("i");
    span1.className = "listButton edit";
    span1.onclick = editElement;
    span2.className = "listButton close";
    span2.onclick = removeElement;
    icon.className = "fa fa-times";

    span1.appendChild(document.createTextNode("Edit"));
    span2.appendChild(icon);

    col4.appendChild(span1);
    col4.appendChild(span2);

    row.appendChild(col4);
    return row
}
```

Funkcja „validateInput()” przeprowadza walidację danych przed dodaniem ich do DOM

```
function validateInput(code, task, date)
{
    let valid = true
    var errorText = ""

    // Regex below means one letter uppercase or lowercase and two digits
    const regex = /^[a-zA-Z]([0-9]{2})$/;
    if(!regex.test(code))
    {
        valid = false;
        errorText += "Error: Invalid code input, code has to comprise of one letter and two digits.<br/>"
    }

    // Task has to be between 3 and 255 chars long, inclusive.
    if(!(task.length <= 255 && task.length >= 3))
    {
        valid = false;
        errorText += "Error: Invalid task input, task has to be between 3 and 255 characters long (inclusive).<br/>"
    }

    // We can have no date or valid date
    if(date !== "")
    {
        date = new Date(date)
        if(!date.isValid())
        {
            valid = false;
            errorText += "Error: Invalid date input.<br/>"
        }
    }
    if(!valid)
        showAlert(errorText)
    return valid;
}
```

Wszelkie błędy wynikające z podania niepoprawnych wartości są raportowane do użytkownika za pomocą funkcji „showAlert()”

Do walidacji kodu wykorzystuje RegExp

Funkcje odpowiedzialne za pokazywanie, chowanie i animacje okienka błędów:

```
// Hide alert when animation finishes
alertBox.addEventListener("animationend", (ev) => {
    if (ev.type === "animationend" && ev.animationName === "fadeOut")
        alertBox.style.display = "none";
}, false);

function showAlert(text)
{
    alertText.innerHTML = text;
    alertBox.style.display = "block"; // show element
    alertBox.classList.toggle('active'); // start animation
}

function alertClose()
{
    alertBox.classList.toggle('active'); // start animation
    // at the end of animation eventListener fires and hides the alert
}
```



Funkcja „addElement()”, wykorzystując powyższe funkcje, dodaje element do listy zadań

```
function addElement()
{
    var code = codeInput.value;
    var task = taskInput.value;
    var date = dateInput.value;
    if(!validateInput(code, task, date))
        return;

    // If the user is editing the row we replace it, otherwise add new row
    if(editedRow)
    {
        editedRow.parentNode.replaceChild(createRow(code, task, date), editedRow);
        clearInput();
    }
    else
        mainTable.appendChild(createRow(code, task, date));

    updateLocalStorage();
}
```

W wypadku gdy „validateInput()” zwróci fałsz, funkcja jest przerywana.

„addElement()” obsługuje również edycję elementu. W przypadku gdy element jest edytowany następuje podmiana na nowy element. Z kolei gdy użytkownik nie jest w trakcie edycji funkcja dodaje nowy element na koniec listy. Na koniec Local Storage zostaje uaktualniony najnowszymi danymi z listy.

## Usuwanie pozycji listy

Funkcja „removeElement()” usuwa pozycję z DOM oraz uaktualnia Local Storage

```
function removeElement()
{
    var div = this.parentElement.parentElement;
    div.parentNode.removeChild(div);
    updateLocalStorage();
}
```

## Edycja pozycji listy

Funkcja „editElement()” pozwala użytkownikowi na edycję elementu listy zadań

```
function editElement()  
{  
    editedRow = this.parentElement.parentElement;  
    editedRow.style.backgroundColor = "#A4E5EB";  
    btn = document.getElementById("addButton");  
    btn.textContent = "Apply"  
    codeInput.value = editedRow.children[0].textContent;  
    taskInput.value = editedRow.children[1].textContent;  
    date = new Date(editedRow.children[2].id);  
  
    // This below is the date conversion that is required by the <input type="date">  
    // This does not bring joy...  
  
    var month = ('0' + (date.getMonth() + 1)).slice(-2)  
    var day = ('0' + date.getDate()).slice(-2)  
    var godHelpMe = date.getFullYear()+"-"+month+"-"+day;  
  
    dateInput.value = godHelpMe;  
}
```

Funkcja wyróżnia edytowany element za pomocą zmiany koloru, następnie kopiuje dane do „paska dodawania” i zmienia tekst przycisku dodaj z „Add” na „Apply”. Pole „input” typu „date” wymaga konwersji na stringa formatu „yyyy-MM-dd”.

Potwierdzenie edycji obsługiwane jest przez funkcję „addElement()”

## Wyszukiwanie

Za wyszukiwanie odpowiedzialna jest funkcja „filterElements()” która ukrywa elementy listy nie spełniające kryteriów. Funkcje wyszukiwania nie zwracają uwagi na wielkość liter i wyszukują fragmenty tekstu a nie tylko całe wyrazy.

```
function filterElements()
{
    // if the clearSearchBtn is not displayed then
    // we have only the taskSearch input, run filterElementsMobile
    var b = document.getElementById("clearSearchBtn")
    if(window.getComputedStyle(b).display === "none")
    {
        filterElementsMobile();
        return;
    }

    var code = codeSearch.value;
    var task = taskSearch.value;
    var date = dateSearch.value;

    // restore the rows
    for (let i = 0; i<mainTable.children.length; i++)
        mainTable.children[i].style.display = "";

    // Hide elements that match code
    if(code !== "")
        for (let i = 0; i<mainTable.children.length; i++)
        {
            if(!mainTable.children[i].children[0].textContent.toUpperCase().includes(code.toUpperCase()))
                mainTable.children[i].style.display = "none";
        }

    // Hide elements that match task
    if(task !== "")
        for (let i = 0; i<mainTable.children.length; i++)
        {
            if(!mainTable.children[i].children[1].textContent.toUpperCase().includes(task.toUpperCase()))
                mainTable.children[i].style.display = "none";
        }

    // Hide elements that match date
    date = new Date(date)
    if(date.isValid())
    {
        date = date.toLocaleDateString()
        for (let i = 0; i<mainTable.children.length; i++)
        {
            var rowDate = new Date(mainTable.children[i].children[2].id).toLocaleDateString();
            if(rowDate !== date)
                mainTable.children[i].style.display = "none";
        }
    }
}
```

Na początku sprawdzamy czy element „clearSeachBtn” jest ukryty, jeżeli tak to jesteśmy w trybie wyświetlania mobilnego więc odwołujemy się do funkcji „filterElementsMobile()” która obsługuje pojedyncze pole wyszukiwania.

Następnie przywracamy wszystkie elementy które mogły być poprzednio schowane i ponownie chowamy elementy które nie spełniają kryteriów oddzielnie dla każdego pola wyszukiwania.

Funkcja jest wywoływana za każdą zmianą w polach wyszukiwania oraz po naciśnięciu przycisku szukaj.

Funkcja odpowiedzialna za wyszukiwanie w trybie wyświetlania mobilnego „filterElementsMobile()”

```
// For mobile displays we have only single search input
// we search one phrase in all the rows
function filterElementsMobile()
{
    var search = taskSearch.value;

    // restore the rows
    for (let i = 0; i<mainTable.children.length; i++)
        mainTable.children[i].style.display = "";

    // hide the rows not matching our search
    if(search !== "")
    {
        for (let i = 0; i<mainTable.children.length; i++)
        {
            var matches = false;
            if(mainTable.children[i].children[0].textContent.toUpperCase().includes(search.toUpperCase()))
                matches = true;
            if(mainTable.children[i].children[1].textContent.toUpperCase().includes(search.toUpperCase()))
                matches = true;
            if(mainTable.children[i].children[2].textContent.toUpperCase().includes(search.toUpperCase()))
                matches = true;
            if(!matches)
                mainTable.children[i].style.display = "none";
        }
    }
}
```

Funkcja analogiczna do „filterElements()” różni się tym, że obsługuje pojedyncze pole wyszukiwania.

## Oznaczanie wyszukiwanej frazy na liście

Wyszukiwana fraza jest oznaczona poprzez schowanie wszystkich fraz nie spełniających kryteriów.

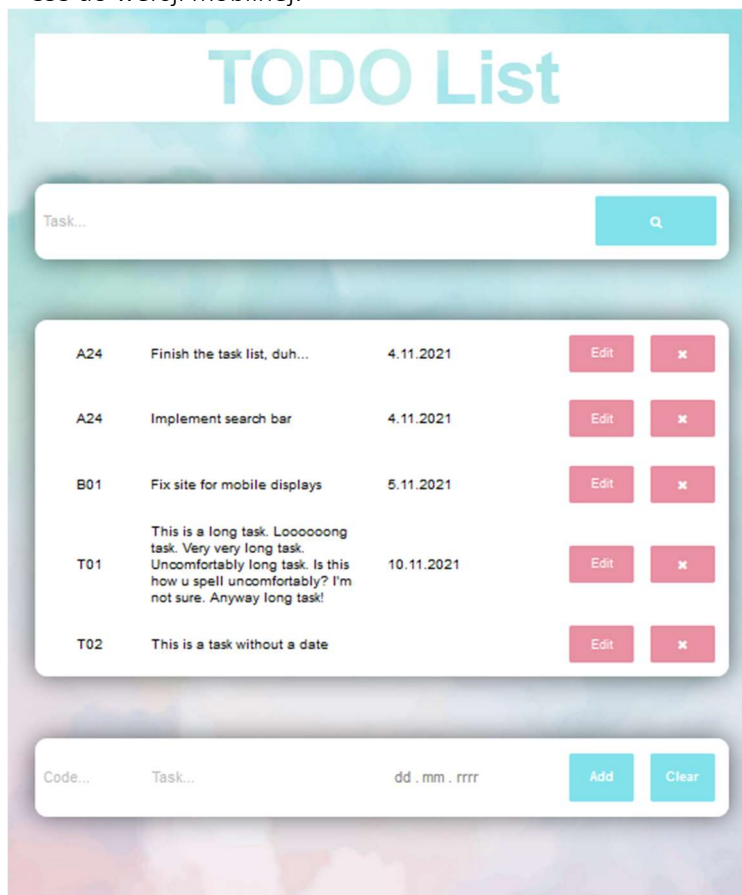
Code...	Task...	Code...	ta
A24	Finish the task list, duh...	A24	Finish the task list, duh...
A24	Implement search bar	T01	This is a long task. Loooooong task. Very very long task. Uncomfortably long task. Is this how u spell uncomfortably? I'm not sure. Anyway long task!
B01	Fix site for mobile displays	T02	This is a task without a date
T01	This is a long task. Loooooong task. Very very long task. Uncomfortably long task. Is this how u spell uncomfortably? I'm not sure. Anyway long task!		
T02	This is a task without a date		

## Linki i uwagi

Link do strony: [http://www.ai-labo1.ml/todo\\_list.html](http://www.ai-labo1.ml/todo_list.html)

Uwagi:

CSS do wersji mobilnej:



```
@media only screen and (max-width: 800px) {  
  input[type="text"]  
  {  
    font-size: 12px;  
  }  
  input[type="date"]  
  {  
    font-size: 12px;  
  }  
  
  .searchRow .col1 {  
    display: none;  
  }  
  .searchRow .col3 {  
    display: none;  
  }  
  .searchRow .mainButton[id="clearSearchBtn"] {  
    display: none;  
  }  
  .searchRow .mainButton[id="searchBtn"] {  
    width: 90%;  
  }  
  .searchRow .col2 {  
    width: 80%;  
  }  
  .searchRow .col4 {  
    width: 20%;  
  }  
  
  .TODO {  
    font-size: 12px;  
    width: 90%;  
    min-width: 0;  
  }  
  h1 {  
    font-size: 60px;  
  }  
  .listButton {  
    font-size: 10px;  
  }  
  .mainButton {  
    font-size: 10px;  
  }  
  .col1 {  
    text-align: center;  
    padding: 5px;  
  }  
  
  .col2 {  
    padding: 5px;  
  }  
  
  .col3 {  
    padding: 5px;  
  }  
  
  .col4 {  
    padding: 5px;  
  }  
  .listButton:hover {  
    background: #e990a3;  
    font-size: 10px;  
  }  
  .mainButton:hover {  
    background: #82e2eb;  
    font-size: 10px;  
  }  
}
```