Lab 2

Tworzymy katalog:

L2_Nazwisko_Id_KalkulatorExtra

W tym katalogu:

- tworzymy naszą aplikację zdarzeniową,
- gromadzimy wszystkie pliki tworzone w czasie zajęć,
- na zakończenie zajęć pakujemy ten katalog z całą zawartością i przesyłamy na Moodle.

Tutaj i w dalszych oznaczeniach:

Nazwisko	– twoje nazwisko
Imię	– twoje imię
Id	– twój numer studenta
XY	– twoje inicjały

W witrynie cały czas ma być widoczne twoje imię i nazwisko
Wszystkie katalogi w projekcie mają kończyć się twoimi inicjałami!
Wszystkie tworzone pliki mają kończyć się twoimi inicjałami!
Wszystkie zmienne, id, nazwy funkcji – mają kończyć się twoimi inicjałami!
Jeśli twoje inicjały to XY, to w przykładzie powinno być:

- katalog *skryptyXY* zamiast katalogu *skrypty*
- *witaj_01XY.html* zamiast *witaj_01.html*
- *demoXY* zamiast *demo*
- *xXY* zamiast *x*

Podobnie we wszystkich kolejnych przykładach.
Zamiast słowa Nazwisko, w odpowiednich miejscach wpisz swoje Nazwisko.

Tworzymy aplikację KalkulatorExtra
Jak wszystkie aplikacje webowe, ta aplikacja też będzie aplikacją sterowaną zdarzeniami.
Nasza aplikacja na różnych przeglądarkach może zachowywać się nieco inaczej.
Różne przeglądarki mogą mieć nieco różniące się implementacje biblioteki standardowej JS, modelu DOM.

Aplikacja ma składać się z:

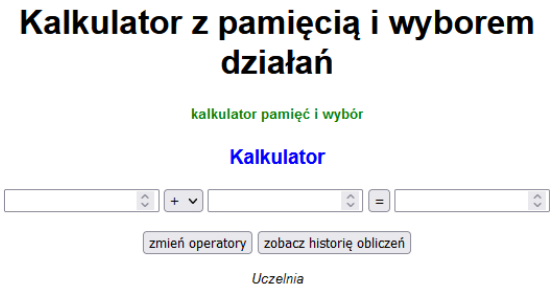
- dokładnie jednego pliku html, ten plik ma być w głównym katalogu projektu
  - w tym pliku nie ma być żadnych skryptów ani arkuszy stylów
- dokładnie jednego pliku ze skryptami
  - ten plik ma być w osobnym katalogu skryptów
- dokładnie jednego pliku ze stylami
  - ten plik ma być w osobnym katalogu stylów

Aplikacja ma trzy „okna”:

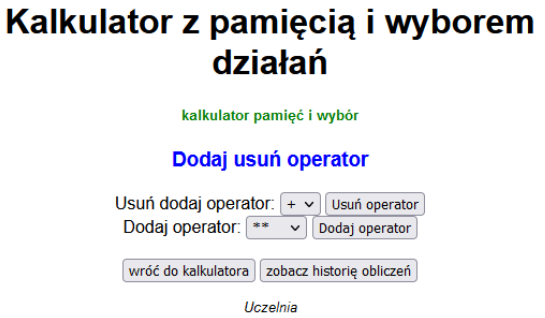
- kalkulator
  - tu są obliczenia
- operatory
  - tu możemy zmieniać dostępne operatory
- memory
  - tu możemy przeglądać historię obliczeń i usuwać elementy historii

Zawsze widoczne jest tylko jedno okno.
Zdarzenie generowane przez użytkownika – kliknięcie (naciśnięcie przycisku) – będzie decydowało co jest widoczne na ekranie – w oknie przeglądarki.

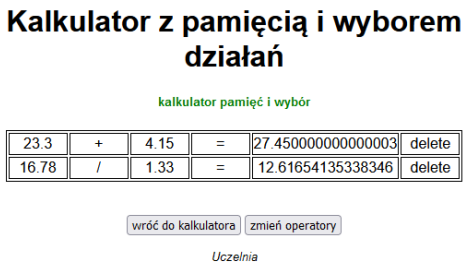
„Okno” – Kalkulator



„Okno” – Dodaj, usuń operator



„Okno” – pamięć wykonanych operacji



Krok 0
Tworzymy szkielet aplikacji zdarzeniowej (aplikacji webowej).

```
<!DOCTYPE html>
<html>
   <head>
   </head>
   <body>
   </body>
</html>
```

Tworzymy dwa pliki:

- w katalogu JS plik skrypty.js ze skryptami JavaScript
- w katalogu CSS plik style.css ze stylami CSS

Wypełniamy szkielet aplikacji podstawowymi informacjami w części <head>
Wpisujemy informację:

- jaki używany zestaw znaków
- tytuł w belce przeglądarki
- gdzie są nasze opisy formatowania CSS
- gdzie jest nasz kod JavaScript

```
<head>
    <meta charset=„utf-8”>
    <title>Kalkulator Nazwisko</title>
    <link rel=„stylesheet” href=„css\style.css”>
    <script src=„js\skrypty.js”></script>
</head>
```

Krok 1
Uzupełniamy podstawowy szkielet aplikacji.
Tworzymy szkielet **naszej** aplikacji webowej.
Dodajemy kod do każdej z sekcji.
Po uzupełnieniu widzimy na ekranie napisy informacyjne z każdej sekcji.

Plik
Kalkulator_Nazwisko.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=„utf-8”>
    <title>Kalkulator Nazwisko</title>
    <link rel=„stylesheet” href=„css\style.css”>
    <script src=„js\skrypty.js”></script>
  </head>
  <body>
    <header id=„page_header”>
      …
    </header>
    <section id=„kalkulator”>
      <h3>Kalkulator</h3>
      …
    </section>
    <section id=„operatory”>
      <h3>Dodaj usuń operator</h3>
      …
    </section>
    <section id=„memory”>
      <h3>Pamięć obliczeń</h3>
      …
    </section>
    <footer id=„page_footer”>
      …
    </footer>
  </body>
</html>
```

W pliku arkusza stylów modyfikujemy styl

- znacznika **body**
- elementu mającego id **page_footer**

```
body{
    text-align: center;
}
#page_footer{
    font-style: italic;
    font-size: small;
}
```

Krok 2
Do każdej sekcji dodajemy dwa przyciski zmieniające widoczność sekcji.

Są trzy sekcje: kalkulator, operatory, memory.
Początkowo wszystkie trzy są widoczne.
Musimy to zmienić. Zawsze tylko jedna ma być widoczna.
Dodajemy do każdej sekcji po dwa przyciski (buttony) pozwalającej „przejść" do wybranej sekcji.

```
……
  <section id="kalkulator">
    <h3>Kalkulator</h3>
    <button onclick="doOperatorow()">zmień operatory</button>
    <button onclick="doHistorii()">zobacz historię obliczeń</button>
  </section>
……
```

Musimy w pliku ze skryptami „szkielety" wywoływanych funkcji:

```
function startKalkulator(){

}
function doOperatorow(){

}
function doHistorii(){

}
function doKalkulatora(){

}
```

Przejście między sekcjami będziemy realizować „pokazywanie" i „chowanie" sekcji.
U nas przejść oznacza zmianę stylu sekcji:

- .style.display = „initial" – dla sekcji do której „przechodzimy", pokazujemy ją – stan początkowy,
- .style.display = „none" – dla pozostałych sekcji, chowamy ją.

Dodajemy funkcjonalność do kolejnych sekcji.
Najpierw sekcja **kalkulator**:

Krok 3
Listę rozwijaną *<select id=„selectOperator" ></select>* włożyliśmy do kontenera inline ** po to by:

- select był inline
- można go była zabrać z tego kontenera i włożyć do innego kontenera
- można go była zabrać z innego kontenera i włożyć do tego kontenera

```
<section id=„kalkulator">
  <h3>Kalkulator</h3>
  <input type=„number" id=„liczba1">
  <span id=„kalkulatorSelect">
    <select id=„selectOperator" ></select>
  </span>
  <input type=„number" id=„liczba2">
  <button onclick = „oblicz()">=</button>
  <input type=„number" id=„wynik" readonly>
  <br><label id=„error"></label>
  <br><button onclick=„doOperatorow()">zmień operatory</button>
  <button onclick=„zobaczHistorie()">zobacz historię obliczeń</button>
</section>
```

Musimy dodać zdefiniować funkcję **oblicz()**.
Początkowo wystarczy „szkielet" funkcji.
Musimy dodać operatory do znacznika **select**.
Nie możemy zapisać operatorów w kodzie HTML znacznika select. Musimy mieć możliwość manipulowania zawartością **select**'a. To co ma być dodane „na starcie" zapiszemy w tablicy.

Operatory startowe umieszczamy w tablicy operatory:

- let operatory = ['+', '-', '*', '/'];

Operatory dodatkowe umieszczamy w tablicy operatoryAll:

- let operatoryAll = ['**', '%', '&&', '||', '&', '|', '^', '==', '===', '>', '<'];

Gdzie umieszczamy te tablice?
W pliku ze skryptami.

Plik skrypty.js

```
let operatory = ['+','-','*','/'];
let operatoryAll = ['**','%','&&','||','&', '|','^','==','===','>','<'];

function startKalkulator(){
   document.getElementById(„kalkulator").style.display = „initial";
   document.getElementById(„operatory").style.display = „none";
   document.getElementById(„memory").style.display = „none";
}
function doOperatorow(){
   document.getElementById(„kalkulator").style.display = „none";
   document.getElementById(„operatory").style.display = „initial";
   document.getElementById(„memory").style.display = „none";
}
function doKalkulatora(){
   document.getElementById(„kalkulator").style.display = „initial";
   document.getElementById(„operatory").style.display = „none";
   document.getElementById(„memory").style.display = „none";
}
function doHistorii(){
   document.getElementById(„kalkulator").style.display = „none";
   document.getElementById(„operatory").style.display = „none";
   document.getElementById(„memory").style.display = „initial";
}
function oblicz(){
}
```

Chcemy by funkcja startKalkulator była wykonywana przy załadowywaniu (*onload*) do przeglądarki.
Modyfikujemy znacznik <body>

```
<body onload="startKalkulator()">
```

Krok 3 cd.
Do funkcji **startKalkulator** musimy jeszcze dodać „wczytanie pracujących (poczatkowych) operatorów",
Robimy to tak jak w aplikacji „**Przepisywanie Napisów**" ćwiczenia 1.
Dla czytelności kodu dodajemy funkcję **workingOperators()** i wywołujemy ją w funkcji **startKalkulator()**.

```
function workingOperators(){
   let selectWorkingOperators = document.getElementById(„selectOperator");
   for (let operator of operatory){
      let option = document.createElement('option');
      option.value = operator;
      option.innerHTML = operator;
      selectWorkingOperators.appendChild(option);
   }
```

```
}

function startKalkulator(){
   document.getElementById(„kalkulator”).style.display = „initial”;
   document.getElementById(„operatory”).style.display = „none”;
   document.getElementById(„memory”).style.display = „none”;
   workingOperators();
}
```

Krok 4
Mamy już wszystko co jest potrzebne do prostych obliczeń.
Musimy teraz uzupełnić funkcję **oblicz()**.
Uzupełniamy ją w oczywisty sposób:

```
function oblicz(){
   let liczba1 = document.getElementById(„liczba1”).value;
   let liczba2 = document.getElementById(„liczba2”).value;
   let operator = document.getElementById(„selectOperator”).value;
   let wynik = liczba1+operator+liczba2;
   wynik = eval(wynik);
   document.getElementById(„wynik”).value = wynik;
   document.getElementById(„error”).innerHTML = „”;
}
```

Wrócimy jeszcze do sekcji **Kalkulator** przy dopisywaniu obliczeń do historii.

Krok 5
Sekcja operatory.
Mamy ustawione robocze operatory, ale chcemy zestaw tych operatorów zmienić.

```
<section id=„operatory”>
   <h3>Dodaj usuń operator</h3>
   <label>Usuń dodaj operator:</label>
   <span id=„usunSelect”></span>
   <button onclick=„usunOperator()”>Usuń operator</button>
   <br>
   <label>Dodaj operator:</label>
   <select id=„selectAddOperator” ></select>
   <button onclick=„dodajOperator()”>Dodaj operator</button>
   <br><br><button onclick=„doKalkulatora()”>wróć do kalkulatora</button>
   <button onclick=„zobaczHistorie()”>zobacz historię obliczeń</button>
</section>
```

W tej sekcji mamy <*select*> do którego podłączone są „wszystkie” operatory.
Przygotowaliśmy też miejsce <*span*> do którego będziemy przełączać <*select*> z sekcji **Kalkulator**.

Krok 6
Musimy do <*select*> wszystkich operatorów podłączyć „wszystkie” operatory.
Możemy to zrobić np.: *onload* przy załadowywaniu <*body*>
Dodajemy funkcję **allOperators()** (analogiczną do funkcji **workingOperators()**):

```
function allOperators(){
   let selectAddOperator = document.getElementById(„selectAddOperator”);
   for (let operator of operatoryAll){
      let option = document.createElement('option');
```

```
        option.value = operator;
        option.innerHTML = operator;
        selectAddOperator.appendChild(option);
    }
}
```

i wywołujemy ją w funkcji **startKalkulator()** (podobnie jak wywołujemy funkcję **workingOperators()**):

```
function startKalkulator(){
    document.getElementById(„kalkulator”).style.display = „initial”;
    document.getElementById(„operatory”).style.display = „none”;
    document.getElementById(„memory”).style.display = „none”;
    workingOperators();
    allOperators()
}
```

Krok 7
Jeżeli chcemy wykonywać operacje na *selectach* musimy przenieść <*select*> z „pracującymi” operatorami z niewidocznej sekcji **Kalkulator** do widocznej (w tej chwili) sekcji **Operatory**.
Przeniesienie:

```
let selekt = document.getElementById(„selectOperator”);
let spanSelektZmiana = document.getElementById(„usunSelect”);
spanSelektZmiana.appendChild(selekt);
```

Przeniesienie dodajemy do instrukcji w funkcji **doOperatorow()**.

```
function doOperatorow(){
    document.getElementById(„kalkulator”).style.display = „none”;
    document.getElementById(„operatory”).style.display = „initial”;
    document.getElementById(„memory”).style.display = „none”;
    let selekt = document.getElementById(„selectOperator”);
    let spanSelektZmiana = document.getElementById(„usunSelect”);
    spanSelektZmiana.appendChild(selekt);
}
```

Krok 8
W pliku skryptów musimy dodać dwie funkcje:

- usunOperator()
- dodajOperator()

To są „bliźniacze” funkcje

```
function dodajOperator(){
    let selectWorkingOperators = document.getElementById(„selectOperator”);
    let selectAddOperator = document.getElementById(„selectAddOperator”);
    let option = selectAddOperator.options[selectAddOperator.selectedIndex];
    selectWorkingOperators.appendChild(option);
}
function usunOperator(){
    let selectWorkingOperators = document.getElementById(„selectOperator”);
    let selectAddOperator = document.getElementById(„selectAddOperator”);
    let option = selectWorkingOperators.options[selectWorkingOperators.selectedIndex];
    selectAddOperator.appendChild(option);
}
```

Krok 9
Możemy już dodawać i usuwać operatory.
Musimy jednak wracając do sekcji **Kalkualator** zabrać z powrotem „pracujące” operatory:

```
let selekt = document.getElementById(„selectOperator”);
let spanSelektZmiana = document.getElementById(„usunSelect”);
spanSelektZmiana.appendChild(selekt);
```

Przeniesienie dodajemy do instrukcji w funkcji **doKalkualtora()**.

```
function doKalkulatora(){
   document.getElementById(„kalkulator”).style.display = „initial”;
   document.getElementById(„operatory”).style.display = „none”;
   document.getElementById(„memory”).style.display = „none”;
   let selekt = document.getElementById(„selectOperator”);
   let spanSelektKalkulator = document.getElementById(„kalkulatorSelect”);
   spanSelektKalkulator.appendChild(selekt);
}
```

Krok 10
Chcemy jeszcze zapisywać do historii wykonane obliczenia i mieć możliwość usuwania wybranych obliczeń.
W sekcji **memory** przygotowaliśmy pustą tabelę.
Do tej tabeli będziemy dodawać wiersze z wykonanymi obliczeniami.

```
<section id=„memory”>
   <table id=„memoryObliczenia”></table>
   <br><br><button onclick=„doKalkulatora()”>wróć do kalkulatora</button>
   <button onclick=„doOperatorow()”>zmień operatory</button>
</section>
```

Dla czytelności zrobimy to w dwóch krokach

Tworzymy funkcję wstawiającą wiersz z podanymi parametrami do tabeli <table id=„memoryObliczenia”>
W tym wierszu do ostatniej komórki **Delete** dołączymy listenera *onclick()* nasłuchującego zdarzenia klikniecie w komórkę.

```
function wstawWiersz(n1,op,n2,result){
   let N1 = document.createElement('td');
   N1.innerHTML = n1;
   let OP = document.createElement('td');
   OP.innerHTML = op;
   let N2 = document.createElement('td');
   N2.innerHTML = n2;
   let equal = document.createElement('td');
   equal.innerHTML = „=„;
   let RES = document.createElement('td');
   RES.innerHTML = result;
   let DELETE = document.createElement('td');
   DELETE.innerHTML = „delete”;
   let wiersz = document.createElement('tr');
   wiersz.appendChild(N1);
   wiersz.appendChild(OP);
   wiersz.appendChild(N2);
   wiersz.appendChild(equal);
   wiersz.appendChild(RES);
   wiersz.appendChild(DELETE);
   let tablica = document.getElementById(„memoryObliczenia”);
```

```
    tablica.appendChild(wiersz);
    DELETE.onclick = function(){this.parentElement.remove()};
}
```

Krok 11
Funkcję **wstawWiersz()** wywołujemy w funkcji oblicz():

```
function oblicz(){
    let liczba1 = document.getElementById(„liczba1”).value;
    let liczba2 = document.getElementById(„liczba2”).value;
    let operator = document.getElementById(„selectOperator”).value;
    let wynik = liczba1+operator+liczba2;
    wynik = eval(wynik);
    if (wynik){
        document.getElementById(„wynik”).value = wynik;
        document.getElementById(„error”).innerHTML = „„;
        wstawWiersz(liczba1, operator, liczba2, wynik);
    }else{
        document.getElementById(„error”).innerHTML = „złe dane lub zły operator”;
        document.getElementById(„wynik”).value = „„;
    }
}
```

Krok 12
W pliku arkusza stylów modyfikujemy styl tabeli

```
table{
    border: 1px solid;
    margin-left: auto;
    margin-right: auto;
}
tr, th, td{
    border: 1px solid;
    width: 60px;
    height: 20px;
}
```