

Lekcja 7

Temat: Model DOM- wyszukiwanie elementów na stronie, modyfikacja stylów

Każda strona HTML składa się z elementów. Na samej górze jest środowisko - czyli okno przeglądarki – **window**, które zawiera w sobie wszystkie obiekty, funkcje i właściwości. W tym środowisku znajduje się **document** (czyli nasza strona). Na stronie znajduje się duża ilość różnych obiektów i elementów.

Do odzwierciedlenia ułożenia elementów JS korzysta z DOM czyli Document Object Model. Jest to model, interfejs, który za pomocą metod i właściwości umożliwia działanie na dokumencie (czyli elementach strony).

Model obiektowy dokumentu (ang. Document Object Model, DOM) stanowi API dla dokumentów HTML i XML. Odpowiada za dwie rzeczy: zapewnia reprezentację struktury dokumentu oraz określa, w jaki sposób odnosić się do tej struktury z poziomu skryptu. DOM przedstawia stronę jako ustrukturyzowaną grupę węzłów.

Podczas ładowania strony internetowej przeglądarka tworzy obiektowy model dokumentu w postaci hierarchii (drzewa) w którym każdy znacznik HTML, atrybut znacznika, tekst znajdujący się pomiędzy znacznikami jest reprezentowany jako obiekt w hierarchii DOM. Obiekty nazywane są **węzłami** (ang. **nodes**) drzewa dokumentu.

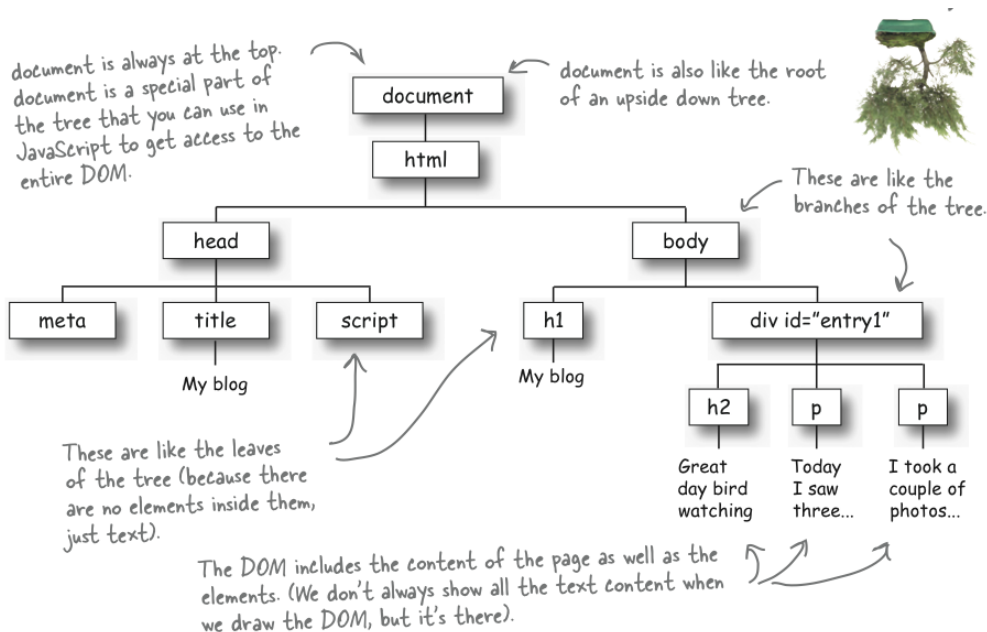
Dzięki obiektowemu modelowi dokumentu HTML za pomocą JavaScript możemy tworzyć dynamiczne strony internetowe, a w szczególności za pomocą JavaScript możemy:

- zmieniać znaczniki HTML i ich atrybuty
- zmieniać style CSS
- tworzyć oraz usuwać znaczniki HTML i ich atrybuty
- reagować na wszystkie zdarzenia

Przykład 1 hierarchii DOM HTML:

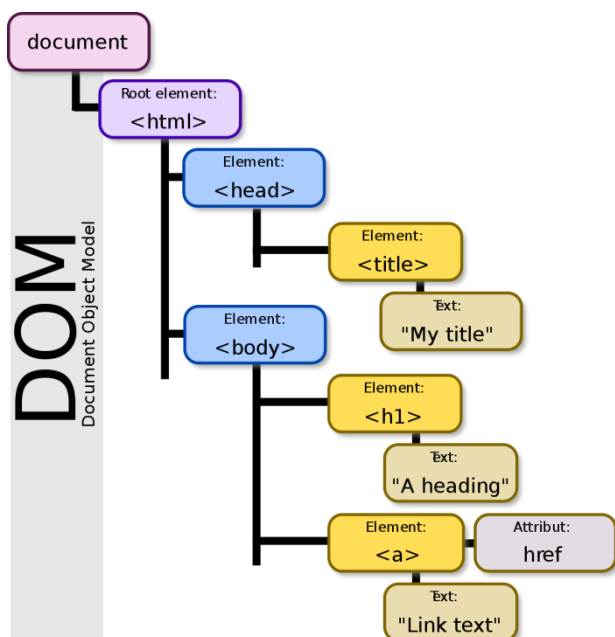
Poniżej znajduje się przykład tego, jak prosty plik HTML jest reprezentowany przez jego DOM.

```
<!doctype html>
<html lang="en">
<head>
  <title>My blog</title>
  <meta charset="utf-8">
  <script src="blog.js"></script>
</head>
<body>
  <h1>My blog</h1>
  <div id="entry1">
    <h2>Great day bird watching</h2>
    <p>
      Today I saw three ducks!
      I named them
      Huey, Louie, and Dewey.
    </p>
    <p>
      I took a couple of photos ...
    </p>
  </div>
</body>
</html>
```



Źródło: <http://cs.wellesley.edu/~cs110/reading/DOM-JQ.html>

Przykład 2 hierarchii DOM HTML:



Źródło: https://en.wikipedia.org/wiki/Document_Object_Model

Zależności między węzłami:

- Wszystkie węzły są ze sobą powiązane
- Każdy węzeł (oprócz document) ma rodzica
- Większość węzłów ma co najmniej jedno dziecko
- Węzły są rodzeństwem, jeśli mają tego samego rodzica
- Potomkami węzła są jego dzieci, dzieci tych dzieci, itd.
- Poprzednikami węzła są jego rodzice, rodzice tych rodziców, itd.

W celu modyfikacji obiektu znajdującego się na stronie internetowej musimy go najpierw wyszukać (zwykle go zapamiętujemy w zmiennej w celu późniejszego wykorzystania).

Metody wyszukiwania elementów strony internetowej:

1. Starsze rozwiązania:

- **document.getElementById("identyfikator")** – wyszukanie elementu o podanym w nawiasach identyfikatorze (id="identyfikator"); Uwaga: nie może być na stronie dwóch elementów o tym samym identyfikatorze.
- **document.getElementsByTagName("znacznik")** – wyszukanie wszystkich obiektów odpowiadających znacznikowi podanemu w nawiasach; ponieważ może być ich wiele zwracana jest tablica obiektów
- **document.getElementsByClassName("klasa")** – wyszukanie wszystkich obiektów posiadających atrybut class podany w nawiasach (class="klasa") w postaci tablicy obiektów

2. Nowsze rozwiązania:

- **querySelector(selector)** - przyjmuje argument, którym jest selektor CSS elementu, jaki chcemy znaleźć. Zwraca **pierwszy znaleziony element** (nawet jeśli istnieją inne elementy), który mógł zostać namierzony przez selektor
- **querySelectorAll(selector)** - wszystkie dopasowania

Ad1.

Metoda **getElementById** zwraca pojedynczy element.

Metody **getElementsByClassName** i **getElementsByTagName** zwracają kolekcję `HTMLCollection`, która zachowuje się jak tablica. Ta kolekcja jest "żywa", co oznacza, że jest automatycznie aktualizowana, jeśli dodatkowe elementy z tym znacznikiem lub klasą zostaną dodane do dokumentu.

Aby uzyskać dostęp do elementów otrzymanych tablic posługujemy się pętlą lub odwołujemy do konkretnego elementu otrzymanej tablicy.

Ad2.

Dają potężne możliwości dotarcia bezpośrednio do potrzebnego elementu. Mają jednak także swoje ograniczenia, co powoduje, że wszystkie wcześniejsze sposoby dotarcia do elementów strony nadal są wykorzystywane. Mówiliśmy, że funkcje **getElementsByTagName** i **getElementsByClassName** tworzą żywe kolekcje, co oznacza, że np. dodanie kolejnego elementu z klasą wykorzystaną przez funkcję powoduje aktualizację wyników jej działania. **querySelectorAll** nie tworzy takiej żywej kolekcji, więc każde dodanie nowego elementu z klasą zastosowaną w tej funkcji wymaga pewnych korekt.

Pobieranie elementu za pomocą `querySelector()` i `querySelectorAll()`

Funkcja `querySelector` działa w następujący sposób:

```
var element = document.querySelector("selektor_CSS");
```

Funkcja `querySelector` przyjmuje argument, którym jest selektor CSS elementu, jaki chcemy znaleźć. Zwraca ona **pierwszy znaleziony element** (nawet jeśli istnieją inne elementy), który mógł zostać namierzony przez selektor. Argumentami mogą być bardzo złożone selektory: np.

- `#id` - identyfikator
- `p.class1` – akapit z klasą
- `img.class2` – obrazek z klasą
- `div > p` – akapit- dziecko, którego rodzicem jest div
- `img[src='zdjecie1.png']` - obraz, którego atrybut src jest ustawiony na zdjecie1.png
- `ul li:first-of-type .btn` - pierwszy element klasy .btn w pierwszym li listy ul

Ponieważ metoda `querySelector()` zwraca tylko pierwszy element pasujący do podanych selektorów, aby otrzymać wszystkie dopasowania, należy użyć metody `querySelectorAll()`.

Funkcje `querySelector` i `querySelectorAll` są niezwykle przydatne w złożonych dokumentach, w których kierowanie na dany element często nie jest proste. Opierając się na dobrze przyjętej składni selektora CSS, możemy rzucać dowolnie małą lub szeroką sieć na elementy, które chcemy odnaleźć. Jeśli chcemy otrzymać dostęp do wszystkich obrazków, możemy po prostu powiedzieć `querySelectorAll("img")`. Jeśli chcemy otrzymać dostęp tylko do elementu `img` zawartego w `div`, możemy powiedzieć `querySelector("div + img")`.

Zmiana stylu elementu za pomocą JavaScript

Mamy dwa sposoby na zmianę stylu elementu za pomocą JavaScript. Jednym ze sposobów jest **ustawienie właściwości CSS bezpośrednio na elemencie**. Innym sposobem jest **dodanie lub usunięcie wartości klasy z elementu**, co może spowodować, że pewne reguły stylu zostaną zastosowane lub zignorowane.

Bezpośrednie ustawianie stylu

Przy pracy ze stylami CSS korzystamy m. in. z obiektu `style` zawartego w każdym węźle drzewa DOM odzwierciedlającym znacznik HTML. Obiekt ten umożliwia określenie właściwości CSS i ustawienie jej wartości.

Jeżeli wskazanie do danego elementu witryny znajduje się w zmiennej `obj`, dostęp do obiektu `style` otrzymamy, pisząc:

```
obj.style;
```

Obiekt ten jest kolekcją atrybutów CSS przypisanych danemu elementowi strony.

W sposobie tym odwołujemy się więc do właściwości CSS. Jeżeli właściwość składa się z jednego słowa, to zapisujemy ją tak jak w css. Jeżeli właściwość składa się z kilku słów oddzielonych myślnikiem, wtedy dla takiej właściwości musimy zastosować zapis **CamelCase**.

A zatem atrybut w formacie CSS, definiowany w kodzie HTML jako:

```
nazwa-atrybutu
```

zmieni się w następującą postać:

```
nazwaAtrybutu
```

Przykładowo, atrybut:

```
font-weight
```

jako właściwość obiektu `style` będzie miał postać:

```
fontWeight
```

```
list1.style.backgroundColor="#49b433";
```

Możemy tutaj zastosować też zapis:

```
list1.style['background-color']="#49b433";
```

UWAGA: niektóre słowa w JavaScript są zastrzeżone i nie można ich używać bezpośrednio. Przykładem właściwości CSS, która należy do tej kategorii specjalnej, jest `float`. W CSS jest to właściwość layoutu. W języku JavaScript oznacza coś innego. Aby użyć właściwości, której nazwa jest całkowicie zarezerwowana, należy poprzedzić właściwość znakami: **css**, w której zmienna **float** zmienia się na **cssFloat**.

Dodawanie i usuwanie klas za pomocą JavaScript

Drugie podejście polega na dodawaniu i usuwaniu wartości klas, które z kolei zmieniają stosowane reguły stylu.

To podejście bazuje na tym, że zasadniczo ustawianie wyglądu powinno się odbywać za pomocą CSS - a dokładnie klas w css, a JS powinien tylko odpowiednio manipulować tymi klasami - dodawać je, odejmować itp. Dzięki temu jeżeli w przyszłości zajdzie konieczność np. zmiany danego przycisku, zmienisz tylko odpowiednie klasy w CSS, a JS zostawisz w spokoju.

Aby zarządzać klasami css danego elementu używamy właściwości **classList**, która udostępnia kilka metod:

- **add("nazwa-klasy")** - dodawanie klasy. Jeśli te klasy już istnieją w atrybucie elementu, to są one ignorowane.
- **remove("nazwa-klasy")** - usuwanie klasy. **Uwaga:** Usunięcie klasy, która nie istnieje, NIE powoduje błędu.
- **toggle("nazwa-klasy")** - przełączanie (jak nie ma to dodaje, jak jest to usuwa) klasy
- **contains("nazwa-klasy")** - sprawdza czy element ma taką klasę

Przykład 1. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p1.html](#)
Przetestuj działanie skryptu klikając kilkakrotnie w utworzony kwadrat.

```
<body>
  <div id="kwadrat" style="background-color:red;width:400px;height:400px;"
  onclick="zmien_kwadrat();" >
  </div>
  <script>
    function zmien_kwadrat()
    {
      var k=document.getElementById("kwadrat");
      if (k.style.backgroundColor=="red")
      {
        k.style.backgroundColor ="yellow";
        k.style.width="200px";
        k.style.height="200px";
      }
      else
      {
        k.style.backgroundColor ="red";
        k.style.width="400px";
        k.style.height="400px";
      }
    }
  </script>
</body>
```

Przykład 2. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p2.html](#)

```
14 <body>
15   <p id="akapit_1">
16     Akapit 1
17   </p>
18   <p>
19     Akapit 2
20   </p>
21   <p>
22     Akapit 3
23   </p>
24   <p>
25     Akapit 4
26   </p>
27   <button onclick="formatuj_akapit();">Formatuj akapit 1</button>
28   <button onclick="formatuj_akapity();">Formatuj akapity 2, 3 oraz 4</button>
29   <script>
30     function formatuj_akapit(){
31       var akapit1 = document.getElementById('akapit_1');// znajdujemy jeden element
32       //formatujemy pierwszy akapit
33       akapit1.style.fontSize='25px';
34       akapit1.style.color='red';
35       akapit1.style.textAlign='right';
36     }
37     function formatuj_akapity(){
38       // znajdujemy wiele elementów, powstaje tablica akapitów
39       var akapity = document.getElementsByTagName('p');
40       for (var i=1;i<akapity.length;i++){
41         //zaczynamy od indeksu 1 ponieważ pierwszy (zerowy) akapit został sformatowany wcześniej
42         var ak=akapity[i].style;
43         ak.fontSize='35px';
44         ak.color='blue';
45         ak.textAlign='center';
46       }
47     }
48   </script>
49 </body>
```

Przykład 3. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p3.html](#)

```
6 <style>
7   body{
8     background-color: #002255;
9     color: #fff;
10    font-size: 20px;
11  }
12  .klasa1{
13    background-color: #ccc;
14    width:100px;
15    height:100px;
16    margin:5px;
17  }
18 </style>
19 </head>
20 <body>
21   <div id="flex1">
22     <div class="klasa1">
23       kwadrat1
24     </div>
25     <div class="klasa1">
26       kwadrat2
27     </div>
28     <div class="klasa1">
29       kwadrat3
30     </div>
31     <div class="klasa1">
32       kwadrat4
33     </div>
34   </div>
35   <button onclick="zmien_uklad()">Zmień układ</button>
36   <button onclick="zmien_kwadraty()">Zmień kwadraty</button>
```

```
37 <script>
38   function zmien_uklad(){
39     var uklad = document.getElementById('flex1');
40     uklad.style.backgroundColor='#1F6553';
41     uklad.style.display='flex';
42   }
43   function zmien_kwadraty(){
44     var kw = document.getElementsByClassName('klasa1');
45     kw[0].style.backgroundColor='red';
46     for (var i=1;i<kw.length;i++){
47       var kwa=kw[i].style;
48       kwa.backgroundColor='#48D4B0';
49       kwa.width="200px";
50       kwa.height="200px";
51     }
52   }
53 </script>
```

Przykład 4. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako l7p4.html

```
6 <style>
7 body{
8 background-color: #002255;
9 color: #fff;
10 font-size: 18px;
11 }
12 </style>
13 </head>
14 <body>
15 <h2 class="class1">Nagłówek z klasa ="class1".</h2>
16 <h3 class="class1">Kolejny nagłówek z klasa ="class1".</h3>
17 <p class="class1">Akapit z klasa ="class1".</p>
18 <p>Akapit bez klasy</p>
19 <button onclick="myFunction()">Zmień kolor tła pierwszego elementu z klasa="class1"</button><br><br>
20 <button onclick="myFunction2()">Zmień kolor tła wszystkich elementów z klasa ="class1"</button><br><br>
21 <button onclick="myFunction3()">Zmień kolor tła akapitów</button>
22
23 <script>
24 // querySelector wybiera pierwszy element z danym selektorem, w tym wypadku z klasa="class1"
25 function myFunction() {
26     document.querySelector(".class1").style.backgroundColor = "red";
27 }
28 // querySelectorAll tworzy kolekcję wszystkich elementów z klasa="class1"
29 function myFunction2() {
30     var all=document.querySelectorAll(".class1")
31     for(var i = 0; i< all.length; ++i){
32         all[i].style.backgroundColor = "green";
33     }
34 }
35 // querySelectorAll tworzy kolekcję wszystkich akapitów
36 function myFunction3() {
37     var p=document.querySelectorAll("p")
38     for(var i = 0; i< p.length; ++i){
39         p[i].style.backgroundColor = "gray";
40     }
41 }
42 </script>
```


Przykład 5. Przygotuj stronę html wykorzystującą poniższy kod oraz zewnętrzny plik JS zawierający funkcje. Zapisz przykład jako l7p5.html oraz plik z funkcjami jako skryptl7p5.js.

```
1 <!DOCTYPE html>
2 <html lang="pl">
3 <head>
4   <meta charset="utf-8">
5   <title>DOM, dynamiczna zmiana html - l19p2</title>
6   <style>
7     body{
8       background-color: #002255;
9       color: #fff;
10      font-size: 18px;
11    }
12  </style>
13  <script src="skryptl19p2.js"></script>
14 </head>
15 <body>
16   <h2 class="class1">Nagłówek z klasą = "class1".</h2>
17   <h3 class="class2">Nagłówek z klasą = "class2".</h3>
18   <p class="class1">Akapit z klasą = "class1".</p>
19   <p class="class1">Drugi akapit z klasą = "class1".</p>
20   <p id="id1">Akapit bez klasy z identyfikatorem</p>
21   <ul>
22     <li>punkt pierwszy</li>
23     <li>punkt drugi</li>
24     <li class="class2">punkt trzeci</li>
25     <li class="class2">punkt czwarty</li>
26   </ul>
27   <ol>
28     <li>punkt pierwszy</li>
29     <li>punkt drugi</li>
30     <li class="class2">punkt trzeci</li>
31     <li class="class2">punkt czwarty</li>
32   </ol>
33   <button onclick="myFunction()">Zmień kolor tła pierwszego akapitu z klasą="class1"</button><br><br>
34   <button onclick="myFunction2()">Zmień kolor tła i tekst akapitu z identyfikatorem</button><br><br>
35   <button onclick="myFunction3()">Zmień formatowanie pierwszego punktu listy wypunktowanej</button><br><br>
36   <button onclick="myFunction4()">Zmień formatowanie punktów listy numerowanej w klasie class2</button><br><br>
37   <button onclick="myFunction5()">Zmień formatowanie punktów w klasie class2</button>
38 </body>
39 </html>
```

```
1 function myFunction() {
2   document.querySelector("p.class1").style.backgroundColor = "red";
3 }
4 function myFunction2() {
5   document.querySelector("#id1").style.backgroundColor = "green";
6   document.querySelector("#id1").innerHTML="<b>Świetnie!!!</b>";
7 }
8 function myFunction3() {
9   document.querySelector("ul li:first-of-type").style.backgroundColor = "#2d7589";
10 }
11 function myFunction4() {
12   var list1=document.querySelectorAll("ol li.class2 ")
13   for(var i = 0; i< list1.length; ++i){
14     list1[i].style.backgroundColor = "#49b433";
15   }
16 }
17 function myFunction5() {
18   var list1=document.querySelectorAll("li.class2 ")
19   for(var i = 0; i< list1.length; ++i){
20     list1[i].style.backgroundColor = "#932ABD";
21   }
22 }
```

Przykład 6. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p6.html](#)

```
15 <body>
16 <h2 class="class1">Nagłówek z klasą ="class1".</h2>
17 <h3 class="class2">Nagłówek z klasą ="class2".</h3>
18 <p class="class1">Akapit z klasą ="class1".</p>
19 <p class="class1">Drugi akapit z klasą ="class1".</p>
20 <p id="id1">Akapit bez klasy z identyfikatorem</p>
21 <ul>
22 <li>punkt pierwszy</li>
23 <li>punkt drugi</li>
24 <li class="class2">punkt trzeci</li>
25 <li class="class2">punkt czwarty</li>
26 </ul>
27 <button onclick="myFunction()">Zmień kolor tła pierwszego akapitu z klasą="class1"</button><br><br>
28 <button onclick="myFunction2()">Zmień kolor tła i tekst akapitu z identyfikatorem</button><br><br>
29 <button onclick="myFunction3()">Zmień kolor tła punktów listy</button><br><br>
30 <script>
31 function myFunction() {
32     document.querySelector("p.class1").style['background-color'] = "rgba(255,255,255,0.5)";
33 }
34 function myFunction2() {
35     document.querySelector("#id1").style['background'] = "linear-gradient(#fff, #aaa)";
36     document.querySelector("#id1").style['color'] = "#000";
37     document.querySelector("#id1").innerHTML="<b>Świetnie!!!</b>";
38 }
39 function myFunction3() {
40     var list=document.querySelectorAll("li")
41     for(var i = 0; i< list.length; ++i){
42         list[i].style['background'] = "linear-gradient(#E9CDF4, #661387)";
43     }
44 }
45 </script>
46 </body>
```

Przykład 7. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p7.html](#).

```
<style>
body{
background-color: #002255;
color: #fff;
font-size: 25px;
}
.disableMenu {
display: none;
}
</style>
</head>
<body>
<nav>
<ul id="dropDown">
<li><a href="1.html">Jeden</a></li>
<li><a href="2.html">Dwa</a></li>
<li><a href="3.html">Trzy</a></li>
<li><a href="4.html">Cztery</a></li>
<li><a href="5.html">Pięć</a></li>
</ul>
</nav>
<button onclick="myFunction()">Ukryj menu</button>
<button onclick="myFunction2()">Pokaż menu</button>
<script>
function myFunction() {
var theDropDown = document.querySelector("#dropDown");
theDropDown.classList.add("disableMenu");
}
function myFunction2() {
var theDropDown = document.querySelector("#dropDown");
theDropDown.classList.remove("disableMenu");
}
</script>
```

Przykład 8. Przygotuj stronę html wykorzystującą poniższy kod. Zapisz przykład jako [l7p8.html](#)

```
7 <style>
8   .box
9   {
10    width: 50px;
11    height: 50px;
12    border: 1px #000 solid;
13    background: #fff;
14    display: inline-block;
15  }
16  .box--marked
17  {
18    background: #f00;
19  }
20 </style>
21 </head>
22 <body>
23   <div class="box"></div>
24   <div class="box"></div>
25   <div class="box"></div>
26   <div class="box"></div>
27   <div class="box"></div>
28   <script>
29     var boxes = document.querySelectorAll('.box');
30     for(var i = 0; i< boxes.length; ++i) //iterujemy po wszystkich boksach
31     {
32       boxes[i].onclick = function(e)
33       {
34         this.classList.toggle('box--marked');
35       };
36     }
37     //Każdy element ma klasę .box i po kliku dodajemy albo usuwamy klasę .box--marked
38   </script>
```

Ćwiczenia wykonaj na dwa sposoby. Wykorzystując:

- A. Metody `getElementById`, `getElementsByClassName` i `getElementsByTagName` oraz bezpośrednią zmianę stylów
- B. `querySelector()` i `querySelectorAll()` i zamianę klas

Ćwiczenie 1 Przygotuj stronę html. Umieść w niej 6 akapitów w 2 różnych klasach: parzyste i nieparzyste akapity (wprowadź do każdego akapitu dwa zdania z lorem ipsum). Utwórz skrypty zmieniające formatowanie tych akapitów (osobno parzystych, nieparzystych, trzech pierwszych), uruchamiane przyciskami. Zmianie powinny podlegać: wyrównanie (środek / do prawej), kolor tła, kolor i krój czcionki.

W ostatnim z akapitów po naciśnięciu kolejnego przycisku zwiększ odległość między literami oraz zamień czcionkę na drukowaną.

Zapisz skrypty pod nazwami [T7cw1A.html](#) [T7cw1B.html](#)

Ćwiczenie 2

- a) Przygotuj stronę html. Umieść w niej 3 linki w postaci zblokowanej z ustalonymi wymiarami, tłem, obramowaniem innego koloru niż tło i brakiem podkreślenia tekstu ustawione jeden pod drugim z lewej strony witryny. Utwórz skrypt zmieniający formatowanie linków tak, aby:
- po kliknięciu na przycisk: „Zmień położenie” zmodyfikowane zostało ich położenie na poziome z prawej strony witryny.
 - po kliknięciu na przycisk: „Zmień kolor” zmodyfikowany został kolor tła linków.

Zapisz skrypt pod nazwą T7cw2aA.html T7cw2aB.html

- b) Zmodyfikuj zadanie 2a w taki sposób, aby możliwy był powrót do wyjściowego położenia linków. Każde kliknięcie w 1 przycisk powinno powodować zmianę położenia poziome/pionowe. Kliknięcie w przycisk drugi zmianę kolorów.

Zapisz skrypty pod nazwami T7cw2bA.html T7cw2bB.html