Języki programowania i GUI

Lista 2 - 2020

1. Utwórz dokument `html` zawierający w sekcji `<body>`:

```
<script>
document.write("ola<br>");
document.write(1/2);
var tests=["1","1/2","'1'+1","1+1","(2+4)*5"];
for(var i=0;i<tests.length;i++)
   document.write(tests[i],"=",eval(tests[i]),"<br>");
</script>
```

   Przeanalizuj jego działanie. Co robi funkcja `eval`? Co to jest `length`?

2. Wykonaj formularz zawierający:

   `<form><input name="a">+<input name="b">=<input name="c"></form>`

   Zdefiniuj `onchange` dla pól `a` i `b` tak, by po każdej ich zmianie w polu `c` pojawiała się suma liczb z pól `a` i `b`. Zobacz co stanie się, gdy zamiast do `onchange` tę samą funkcję przypiszesz do zdarzenia `onkeyup`.

3. Wykonaj dokument zawierający pole `input` i pewną liczbę przycisków `button`, oznaczonych cyframi i symbolami działań. Zdefiniuj funkcje `onclick` dla przycisków tak, by twój dokument działał jak prosty kalkulator.

4. (a) Napisz funkcję, która policzy, ile jest w dokumencie znaczników `a`.
   (b) Zmodyfikuj ją tak, by zwracała spis pól `href` oraz `innerHTML` tych tagów, postaci: `[["http://wp.pl","Wirtualna Polska"],["http://onet.pl","Onet"], ... ]`.
   (c) Napisz funkcję która zwróci listę atrybutów `src` dla szystkich obrazków, które są w dokumencie.
   Na końcu dokumentu dodaj skrypt sprawdzający działanie tych funkcji.

5. Napisz skrypt, który po załadowaniu dokumentu, wstawi do elementu `<div id="toc"></div>` spis treści, czyli wszystkich elementów `h1 h2 h3`. Spis powinien być wykonany jako zagnieżdżona lista w postaci:

```
<ol>
  <li><a href="#1">Rozdział 1</a>
        <ol>
        <li><a href="#1-2">Podrozdział 1.1 </a></li>
        <li><a href="#1-3">Podrozdział 1.2 </a></li>
        </ol>
  </li>
  <li><a href="#2">Rozdział 2</a>
        <ol>
        <li><a href="#2-1">Podrozdział 2.1 </a></li>
        <li><a href="#2-2">Podrozdział 2.2 </a></li>
        </ol>
  </li>
</ol>
```

   Zadbaj o to, by twój skrypt przypisał automatycznie identyfikatory do tych `h1 h2 h3`, które ich nie mają tak, by spis treści być "klikalny". W przypadku gdy element ma

już przypisany identyfikator użyj istniejącego. Identyfikatory przypisywane przez twój skrypt mogę mieć inną formę niż ta pokazana na przykładzie, w szczególności mogą być kolejnymi liczbami naturalnymi.

6. Napisz funkcję, która na podstawie zmiennej zawierającej tablicę figur w postaci:

```
verb fig=[
   {name:"Punkt",x:1,y:3},
   {name:"Kolo",x:2,y:4,r:4,color:"green"},
   {name:"Prostokat",x:5,y:6,a:2,b:4,color:"red"},
   {name:"Wielokat",x:[5,3,5,6,7],y:[5,6,7,8,10],color:"pink"},
];
```

   Narysuje te figury na elemencie typu `canvas` podanym jako drugi argument funkcji.

7. Dodaj do Twojego dokumentu formularz pozwalający na dodawanie nowych figur do zmiennej `fig` oraz do rysunku. formularz powinien zawierać pole typu `select` określające typ (nazwę) figury, pola `input` dla własności `x,y,r,a,b` oraz przyciski `<input type=submit>` z napisami **Dodaj** i **Anuluj**. Zdefiniuj obsługę pola `select` tak, by w zależności od wybranego typu figury ukrywały się (`a.style.display='none'`) niepotrzebne pola formularza, a pokazywały (`a.style.display='inline'`) potrzebne.

8. Dla każdej z figur zdefiniuj metodę `contains(x,y)` która sprawdza czy punkt o współrzędnych `x,y` zawiera się w figurze.

9. Zdefiniuj dla obiektu `canvas` zdarzenie `onclick`, które sprawdza, która figura z tablicy `fig` została kliknięta.

10. Napisz funkcję, która wypełnia formularz danymi figury podanej jako argument funkcji.

    Wykorzystaj ją w połączeniu z funkcją z poprzedniego zadania do obsługi zdarzenia `onclick` elementu `canvas` zawierającego narysowane figury tak, by formularz był wypełniany danymi kliniętej figury.

    Formularz powinien pozwalać na zmianę własności figury, lub jej usunięcie. Dla formularza zdefiniuj zdarzenie `onsubmit`, które zwróci wartość `false` jednak wykona porządane zmiany w odpowiedniej figurze z tablicy `fig` i spowoduje narysowanie rysunku od nowa. W formularzu zastosuj pole `hidden` do przechowamia numeru edytowanej figury.

11. Napisz `funkcje*` zwracające generatory podobne do:

```
function* idMaker(n) {
   var index = 0;
   while (index < n)
      yield index++;
}

for(let x of idMaker(5))
    console.log(x);
```

    (a) `function* dzielniki(n)` - zwraca generator dzielników liczby `n`,
    (b) `function* pierwsze(n)` - zwraca generator liczb pierwszych nie większych od `n`.
    (c) `function* elements(selector)` - zwraca generator elementów dokumentu zadanych przez `selector`.