Lekcja 6

Temat: Pętle

Pętle

Pętle to konstrukcje służące do wykonywania powtarzających się czynności. Przykładowo, gdyby trzeba było 100 razy wypisać na stronie pewien tekst, to można by w tym celu użyć 100 instrukcji document.write. Byłoby to jednak niewygodne rozwiązanie. Często spotkamy się z sytuacją, gdy liczba powtórzeń danej instrukcji nie jest z góry znana i wynika z obliczeń wykonywanych przez skrypt bądź też zależy od decyzji użytkownika strony WWW. Pętle pozwalają na automatyzację takich czynności.

Pętla for

Pętla **for** wykorzystywana jest najczęściej wtedy, gdy przed rozpoczęciem pętli znana jest liczba powtórzeń jaką należy wykonać

Pętla typu for ma następującą składnię:

```
for (wyrażenie początkowe; wyrażenie warunkowe; wyrażenie
modyfikujące){
    blok instrukcji
}
```

I tak *wyrażenie początkowe* jest stosowane do zainicjalizowania zmiennej używanej jako licznik liczby wykonań pętli; *wyrażenie warunkowe* określa warunek, jaki musi być spełniony, aby dokonać kolejnego przejścia w pętli, natomiast *wyrażenie modyfikujące* jest zwykle wykorzystywane do modyfikacji zmiennej będącej licznikiem.

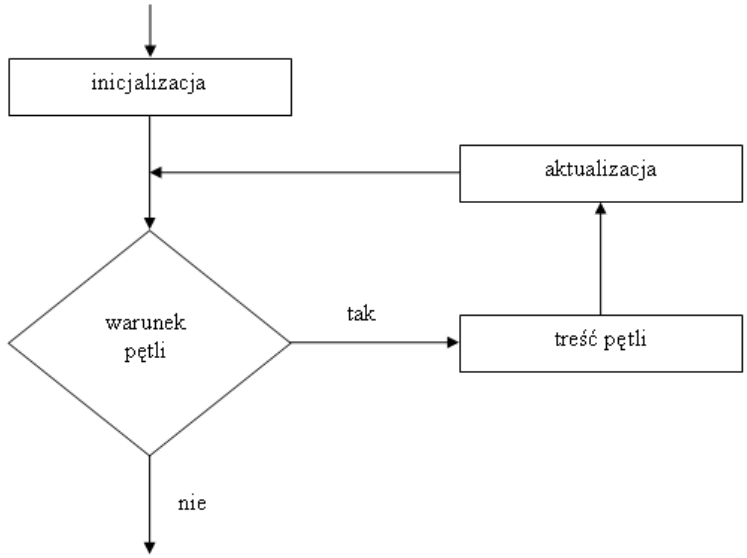
Przykład 1. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    for (var i = 0; i < 10; i++){
      console.log(`Pętla typu for ${i}`);
    }
</script>
```

Konstrukcję tę należy rozumieć następująco: utwórz zmienną i i przypisz jej wartość zero (i = 0), następnie dopóki wartość i jest mniejsza od 10 (i < 10), wykonuj instrukcje znajdujące się wewnątrz pętli oraz zwiększaj i o jeden (i++). Tym samym na ekranie pojawi się 10 razy napis Pętla typu for zawierający też kolejny numer. Zmienna i jest nazywana **zmienną iteracyjną**, czyli kontrolującą kolejne przebiegi (iteracje) pętli.

Licznik najczęściej jest zwiększany o 1 (**inkrementacja**) lub zmniejszany o 1 (**dekrementacja**) (przykład: i=i+1; co można zapisać w skrócie i++; lub i=i-1; w skrócie i--;)

Schemat blokowy pętli for:



Pętle typu for można zmodyfikować tak, aby pozbyć się wyrażenia modyfikującego. Dokładniej — przenieść je do wnętrza pętli w następujący sposób:

```
for (wyrażenie początkowe; wyrażenie warunkowe;){
  instrukcje do wykonania
  wyrażenie modyfikujące
}
```

Przykład 2. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    for (var i = 0; i < 10;){
      console.log(`Pętla typu for ${i}`);
      i++;
    }
</script>
```

W podobny sposób jak przedstawiony wyżej można też "pozbyć się" wyrażenia początkowego, przenosząc je jednak nie do wnętrza pętli, a przed nią:

```
wyrażenie początkowe;
for ( ; wyrażenie warunkowe ;){
  instrukcje do wykonania
  wyrażenie modyfikujące
}
```

Całe wyrażenie początkowe zostało przeniesione przed pętlę. Uwagę **należy zwrócić na umiejscowienie średników pętli** `for`. **Oba są niezbędne do prawidłowego działania kodu.**

Przykład 3. Przygotuj stronę html wykorzystującą poniższy kod.

```
<script>
    var i = 0;
    for (; i < 10;){
      console.log(`Pętla typu for ${i}`);
      i++;
    }//zwróć uwagę na użycie średnika w warunku
</script>
```

Przykład 4. Przygotuj stronę html wykorzystującą poniższe pętle wyświetlające w różny sposób ciągi liczb.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Pętla for - l7p4</title>
    <style>
        ul{
            list-style-type: circle;
        }
        table, th, td {
            border: 1px solid black;
        }
        table {
            border-collapse: collapse;
        }
    </style>
</head>
```

```
<body>
    <p>1 seria liczb</p>
    <script>
        var i = 0;
        for (i=1; i < 7;i++){
          document.write( i + "<br>");
        }
    </script>

    <p>2 seria liczb</p>
    <ul>
    <script>
        var i = 0;
        for (i=1; i < 7;i++){
          document.write( "<li>"+i+"</li>");
        }
    </script>
    </ul>
```

```
    <p>3 seria liczb</p>
    <table>
    <tr>
    <script>
        var i = 0;
        for (i=1; i < 7;i++){
          document.write( "<td>"+i+"</td>");
        }
    </script>
    </tr>
    </table>
</body>
</html>
```

Przykład 5. Przygotuj stronę html wykorzystującą poniższe zagnieżdżone pętle, wyświetlające tabliczkę dodawania .

```
<body>
    <table> <!-- początek tabeli -->
        <tr>
            <th>+</th>
            <script>
            var liczba_wierszy = 3;
            var liczba_kolumn = 6;
            for (i = 1; i <= liczba_kolumn; i++) {
               document.write("<th>"+i+"</th>");
            }
            </script>
        </tr>

        <script>
        for (i = 1; i <= liczba_wierszy; i++) {
            document.write("<tr><th>"+i+"</th>");
            for (j = 1; j <= liczba_kolumn; j++) {
            var s=j+i;
            document.write("<td>"+s+"</td>");
            }
            document.write("</tr>");
        }
        </script>
    </table>
</body>
</html>
```

Przykład 6. Przygotuj strony html wykorzystujące poniższe zagnieżdżone pętle. Przeanalizuj dokładnie kod.

a)

```
<body>
    <script>
        let height = parseInt(prompt("Podaj ilość poziomów"));

        for (i = 0; i <= height; i++) {

            for (j = 1; j <= i ; j++) {
                document.write ('*');
            }
            document.write ('<br>');
        }
    </script>
</body>
```

b)

```
<body>
    <p id="output1"></p>
    <script>
        let height = parseInt(prompt("Podaj ilość poziomów"));
        let output = document.getElementById("output1");
        let contents='';
        for (i = 0; i <= height; i++) {

            for (j = 1; j <= i ; j++) {
                contents+='*';
            }
            contents+='<br>';
        }
        output.innerHTML = contents;
    </script>
</body>
</html>
```

Pętla while

Pętla typu while służy, podobnie jak for, do wykonywania powtarzających się czynności. Pętlę **for** wykorzystuje się najczęściej, **gdy liczba powtarzanych operacji jest znana** (np. jest zapisana w pewnej zmiennej), natomiast pętlę **while**, **gdy liczby powtórzeń z góry nie znamy, a zakończenie pętli jest uzależnione od spełnienia pewnego warunk**u (np. zwrócenia konkretnej wartości przez pewną funkcję). Taki podział jest jednak dosyć umowny, gdyż oba typy pętli można zapisać w taki sposób, aby były swoimi funkcjonalnymi odpowiednikami.

Ogólna konstrukcja pętli typu while jest następująca:

```
while (wyrażenie warunkowe)
{
  instrukcje
}
```

Należy ją rozumieć następująco: **dopóki warunek jest prawdziwy, wykonuj instrukcje**.

Warunkiem może być dowolne wyrażenie, którego można użyć w instrukcji warunkowej, na przykład x > 10 lub answer == 'tak'. Interpreter wykonuje cały kod zapisany pomiędzy otwierającym i zamykającym nawiasem klamrowym, *jeśli* warunek jest spełniony.

Przykład 7. Przygotuj stronę html wykorzystującą poniższy kod. Użyj pętli while, aby 10 razy wyświetlić na ekranie dowolny napis.

```
<body>
    <script>
        let i = 0;
        while(i++ < 10){
          document.write("Pętla typu while [" + i + "]<br>");
        }
    </script>
</body>
```

Warto zwrócić uwagę, że kod ten nie jest w pełni funkcjonalnym odpowiednikiem przykładu 1 z pętlą for. Napisy zostaną wyświetlone 10 razy, ale wartość zmiennej wewnątrz pętli zmieniać się będzie od 1 do 10, a nie od 0 do 9. (Sprawdź oba przykłady). Można to jednak w prosty sposób zmienić, albo modyfikując pierwotną wartość zmiennej i, albo też wprowadzając instrukcję modyfikującą wartość tej zmiennej do wnętrza pętli.

Zmienna sterująca w pętli while

Zmień kod z przykładu 1 w taki sposób, aby skrypt był funkcjonalnym odpowiednikiem przykładów z pętlą for.

Przykład 8. Przygotuj stronę html wykorzystującą poniższy kod.

```
<body>
    <h2>1 sposób</h2>
    <script>
        var i = -1; //modyfikacja  wartość zmiennej i
        while(i++ < 9){
          document.write("Pętla typu while [" + i + "]<br>");
        }
    </script>
    <h2>2 sposób</h2>
    <script>
        var i = 0;
        while(i < 10){
          document.write("Pętla typu while [" + i + "]<br>");
          i++; //wprowadzenie instrukcji i++ do wnętrza pętli
        }
    </script>
</body>
```

Pętla do…while

Pętla do…while jest odmianą pętli while. Ma ona następującą postać:

```
do{
  instrukcje;
}
while(warunek);
```

Konstrukcję tę należy rozumieć: wykonuj *instrukcje*, dopóki *warunek* jest prawdziwy. A zatem jest to swoista odwrotność pętli while.

Przykład 9. Przygotuj stronę html wykorzystującą poniższy kod.

```
<body>
    <script>
        var i = 0;
        do{
          document.write("Pętla typu do...while [" + i + "]<br>");
        }
        while (i++ < 9);
    </script>
</body>
```

Różnice w działaniu pętli:

W przypadku pętli while najpierw sprawdzany jest warunek, a potem są wykonywane instrukcje, co oznacza, że może ona nie zostać wykonana w ogóle, gdy warunek jest od razu fałszywy. W przypadku pętli do…while jest dokładnie odwrotnie — najpierw wykonywane są instrukcje, a dopiero potem badany jest warunek. Oznacza to, że **w przypadku do…while instrukcje z wnętrza są wykonywane co najmniej jeden raz, nawet jeśli warunek jest ewidentnie fałszywy**.

Zwróćmy również uwagę, że w związku z odmienną konstrukcją pętli nieco inaczej wygląda wyrażenie warunkowe (i++ < 9). Tym razem jest sprawdzane, czy i jest mniejsze od 9. Gdyby pozostawić taki sam warunek, jak w przypadku pętli while (i++ < 10), instrukcje document.write zostałyby wykonane 11 razy.

Przykład 10. Przygotuj stronę html wykorzystującą poniższe pętle.

```
<body>
    <script>
        var a = 2;
        while (a==5){
            console.log("Warunek nie spełniony, petla while nie działa");
        }

        do{
          console.log("Pętla typu do...while została wykonana");
        }
        while (a==5);
    </script>
</body>
```

Przykład 11. Przygotuj stronę html wykorzystującą poniższe zagnieżdżone pętle, wyświetlające tabliczkę dodawania . Przeanalizuj dokładnie kod.

```
<table> <!-- początek tabeli -->
    <tr>
        <th>+</th>
        <script>
        var liczba_wierszy = 3;
        var liczba_kolumn = 6;
        var i=1;
        while ( i <= liczba_kolumn) {
           document.write("<th>"+i+"</th>");
           i++;
        }
        </script>
    </tr>
    <script>
    i = 1;
    while ( i <= liczba_wierszy) {
        document.write("<tr><th>"+i+"</th>");
        j = 1;
        while ( j <= liczba_kolumn) {
        var s=j+i;
        document.write("<td>"+s+"</td>");
        j++;
        }
        document.write("</tr>");
        i++;
    }
    </script>
</table>
```

Przykład 6. Przygotuj stronę html wykorzystującą poniższe zagnieżdżone pętle. Przeanalizuj dokładnie kod.

```
<script>
    var wys = prompt("Podaj ilość poziomów");
    var i = 1;
        do {
            j = 1;
            do  {
                document.write ('*');
                j++;
            }
            while (j<= i );
            document.write ('<br>');
            i++;
        }
        while(i<= wys);

</script>
```