Funkcje w JavaScript

Funkcja jest oddzielnym blokiem kodu, który może być wielokrotnie wykonywany w danym programie, poprzez jej wielokrotne wywoływanie. Do funkcji przekazujemy przeważnie jakieś argumenty, a funkcja może nam zwracać jakąś wartość. Dobrze jest tworzyć funkcję tak, aby wykonywała jedno określone zadanie - czyli większe operacje w programie rozdzielamy na kilka wywoływanych kolejno funkcji. Dzięki temu tworzymy cegiełki, z których budujemy potem cały skrypt, a które możemy wykorzystać w innych skryptach

Najczęściej funkcję definiujemy na początku kodu strony - czyli w sekcji HEAD, a wywołujemy ją w dowolnym miejscu poniżej, jeżeli zajdzie taka potrzeba. Dzięki temu możemy być pewni, że funkcja jest załadowana, zanim następuje jej wywołanie.

Funkcja to zamknięty fragment kodu, który można wywołać z "każdego miejsca". Ten fragment kodu może zwracać jakiś wynik, który bardzo łatwo można wykorzystywać w innych instrukcjach.

Definicja funkcji

```
function nazwa_funkcji(){
..instrukcje funkcji
}
//lub
function NazwaFunkcji( parametry ){
    ..instrukcje funkcji
}
```

- Definicja funkcja rozpoczyna się od słowa kluczowego function, po którym znajduje się zapisany w nawiasach okrągłych zbiór jej argumentów (rozdzielanych przecinkami) i ciało funkcji.
- Funkcje w JavaScript nie sprawdzają, ile argumentów dostają na wejściu.
- Argumenty wywołania funkcji są przechowywane w zmiennej `arguments`.
- Kolejne argumenty można uzyskać korzystając z operatora indeksowania.

`NazwaFunkcji` jest to dowolna nazwa, która powinna spełniać takie same wymogi jak nazwy zmiennych (czyli pierwszym znakiem może być litera lub znak podkreślenia; kolejne znaki nazwy mogą być literą, cyfrą lub znakiem podkreślenia; nazwa nie może też być zarezerwowanym słowem kluczowym). `parametry` jest to lista nazw parametrów, rozdzielona przecinkami (nazwy parametrów również muszą spełniać wspomniane wcześniej wymagania). Lista parametrów może być pusta (pomiędzy nawiasami okrągłymi wtedy nic nie ma).

W ciele funkcji może być umieszczona dowolna liczba instrukcji. Nawiasy klamrowe są obowiązkowe i nie można ich pominąć, nawet jeżeli funkcja zawiera tylko jedną instrukcję (lub nawet nie zawiera żadnej instrukcji - w pewnych przypadkach takie funkcje które nic nie robią też mogą być przydatne).

Przykłady funkcji

Zadeklarowana funkcja może być wywoływana w dowolnym momencie skryptu. Możemy ją wywoływać np. poprzez obsługę zdarzenia (np.**onclick**), poprzez sprawdzenie jakiegoś warunku, z wnętrza innej funkcji itp.:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function pisz() {
    alert("To jest tekst zawarty w funkcji.")
    }
pisz(); //wywołujemy funkcję o nazwie pisz
/*funkcja pisz nie pobiera argumentów i nie zwraca wartości
*/
</script>
</head>
<body>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function pisz() {
    alert("To jest tekst zawarty w funkcji.")
    }
</script>
</head>
<body>
<h1>wywołanie funkcji "pisz" nastąpi po kliknięciu na button</h1>
<button onclick="pisz()">kliknij, aby wywołac funkcję</button>
</body>
</html>
```

wywołanie funkcji "pisz" nastąpi po kliknięciu na button

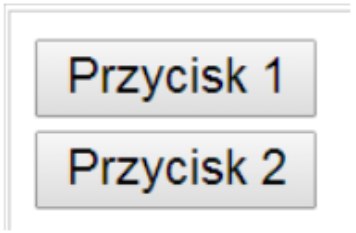
kliknij, aby wywołac funkcję

Przyciski ogólnego zastosowania- przypomnienie

Można je utworzyć na dwa sposoby: za pomocą znacznika

<input type="button"> lub za pomocą <button>:

```
<input type="button" value="Przycisk 1">
 <button>Przycisk 2</button>
```



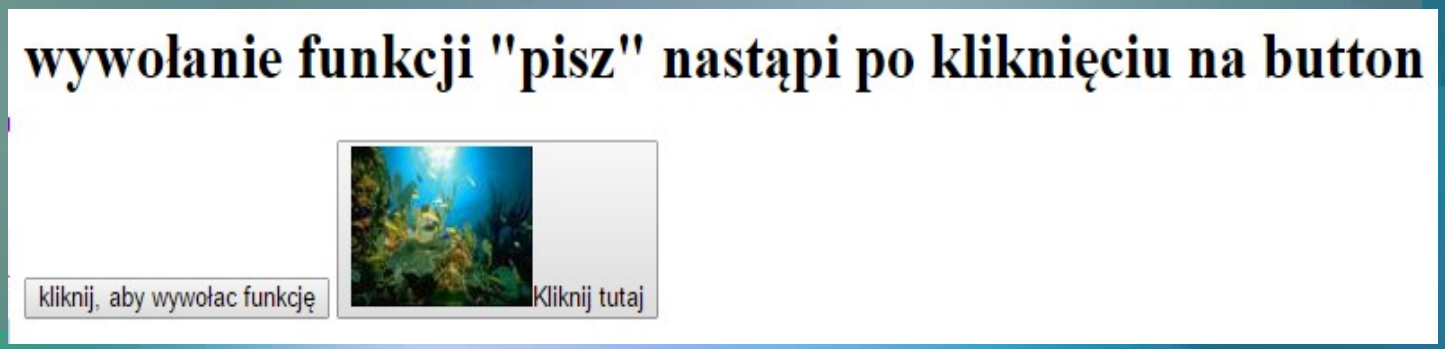
Domyślnie naciśnięcie żadnego z nich nie spowoduje wykonania się jakiejś akcji (np. wysłania formularza) - aby coś się stało, należy to oprogramować korzystając ze skryptu JavaScript.

Warto też wspomnieć, że przycisk stworzony za pomocą znacznika <button> </button>pozwala umieścić na nim nie tylko tekst, ale też np. obrazek:

```
<button><img src="images.jpg" width="101" height="77"></button>
```

cd. przykłady funkcji

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function pisz() {
   alert("To jest tekst zawarty w funkcji.")
   }
</script>
</head>
<body>
<h1>wywołanie funkcji "pisz" nastąpi po kliknięciu na button</h1>
<button onclick="pisz()">kliknij, aby wywołac funkcję</button>
<button onclick="pisz()"  ><img src="images.jpg" width="101" height="77" >Kliknij tutaj</button>
</body>
</html>
```



Parametry (argumenty) funkcji

Często konieczne będzie przekazanie do funkcji określonych danych, które następnie zostaną przez nią przetworzone. Parametry przekazuje się wypisując je między nawiasami występującymi po nazwie funkcji:

```
function nazwa_funkcji(parametr_1, parametr_2, parametr_3...) {
operacje na parametrach
                }
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function wypiszSume(liczbaA, liczbaB) {
        alert(liczbaA + liczbaB);
    }
    wypiszSume(3, 4); //wypisze 7
    wypiszSume(7, 7); //wypisze 14
    wypiszSume(100, 100); //wypisze 200

</script>
</head>
<body>
</body>
</html>
```

Funkcje zwracające wartość- instrukcja return

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function iloczyn(liczba1, liczba2) {
var wynik = liczba1 * liczba2;
return wynik;
    }
alert( iloczyn(10,4) );
document.write(iloczyn(3,4));
</script>
</head>
<body>
</body>
</html>
```

Ważną cechą instrukcji **return** jest to iż powoduje ona natychmiastowe przerwanie wykonywania funkcji i powrót **do** miejsca w skrypcie z którego funkcja była wywołana. Z tego też powodu możliwe jest także użycie komendy **return** bez podawania wartości która ma zostać zwrócona z funkcji, co jest użyteczne gdy piszemy funkcję która ma tylko coś zrobić (np. wypisać jakiś napis), ale nie ma zwracać żadnej wartości.

Funkcje zwracające wartość- instrukcja return

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script>
function DrukujLiczby(a, b)
{
    for (n = a; n <= b; ++n)
    {
        if (n >= 100)
            return;
        document.write(n+"<br>");
    }
}
</script>
</head>
<body>
<h1>Poniższa funkcja drukuje wszystkie wartości z przedziału określonego parametrami, ale kończy drukować jeżeli przekroczy liczbę 100.  </h1>
<h1>kliknięcie na button spowoduje wywołanie funkcji DrukujLiczby(90,120) z parametrami 90 i 120  </h1>
<button onclick="DrukujLiczby(90,120)">kliknij, aby wywołać funkcję</button>
</body>
</html>
```

Poniższa funkcja drukuje wszystkie wartości z przedziału określonego parametrami, ale kończy drukować jeżeli przekroczy liczbę 100.

kliknięcie na button spowoduje wywołanie funkcji DrukujLiczby(90,120) z parametrami 90 i 120

kliknij, aby wywołać funkcję

90
91
92
93
94
95
96
97
98
99

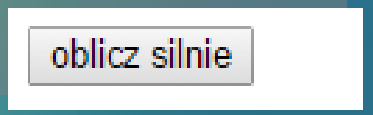
Przykłady funkcji

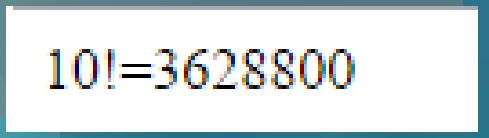
Napisz funkcję obliczającą silnię liczby naturalnej n. Liczba n jest przekazana do funkcji jako parametr. Zastosuj funkcję w programie.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<script>
function silnia(n)
{var s=1;
for(var i=1;i<=n;i++)
   {s*=i;}
   return s;
}
function wypisz_wynik()
          {
          var n1=prompt("podaj liczbę naturalną",1);
          document.write(n1+"!="+silnia(n1));
          }

</script>
<button type="button" onclick="wypisz_wynik()">oblicz silnie</button>

</body>
</html>
```





Zasięg zmiennych

Deklarowane zmienne możemy podzielić na globalne i lokalne.

Zmienne globalne deklarowane są poza funkcjami i są dostępne dla całego skryptu. Zmienne lokalne są deklarowane wewnątrz funkcji i dostęp do nich ma tylko funkcja, w której dana zmienna została zadeklarowana. Do zmiennych globalnych dostęp z wnętrza funkcji jest możliwy, do zmiennych lokalnych dostęp spoza funkcji nie jest możliwy. Dzięki temu deklarując zmienną wewnątrz funkcji, nie musimy się martwić czy przypadkiem nie będzie ona kolidowała z którąś zmienną globalną (zarówno nazwą jak i zwracaną wartością).

Zasięg zmiennych

```
<script>
var x = 10;
//zmienna globalna
//tutaj zmienna x = 10

function zmiana() {
var x = 20;  //x-zmienna lokalna, przesłania zmienną globalną x
// zmienne z wiersza nr 8. i nr 13. to różne zmienne
//jest widoczna tylko we wnętrzu funkcji
//...tutaj zmienna x = 20;
}
zmiana(); //wywołanie funkcji
//funkcja zmiana zmienia lokalną x, wiec nie wpływa na zmiennaą globalną x z poczatku skryptu
alert(x); //wypisze 10
var y = 10;
function zmiana2() {
    y = 20; //pomineliśmy operator var, zmienna jest traktowana jak zmienna globalna-
            //zmienna z wiersza 20. i 22 to ta sama zmienna
}
zmiana2();//po wywołaniu funkcji zmiana wartosc zmiennej y zadeklarowanej na poczatku skryptu zmieni
się na 20
alert(y);// wypisze 20
</script>
```

Przy deklaracji zmiennej wewnątrz funkcji użyliśmy przedrostka var. Co się stanie, jeżeli taki operator pominiemy? Jeżeli poza funkcją znajduje się zmienna o takiej samej nazwie, wówczas zamiast deklaracji nowej zmiennej, zmienimy tylko wartość zmiennej spoza funkcji. Dlatego też deklarując zmienną wewnątrz funkcji zawsze używajmy operatora var.

Zmienna lokalna, zmienna globalna

Aby zmienna była zmienną lokalną, musi spełniać dwa warunki:

1. Jej definicja musi znajdować się wewnątrz funkcji.
2. W definicji musi zostać użyte słowo var.

To oznacza, że jeżeli w funkcji w pierwszym odwołaniu do jakiejś zmiennej nie użyjemy słowa var, to zostanie to potraktowane jak odwołanie do zmiennej globalnej. Co więcej, jeżeli taka zmienna jeszcze nie istnieje, to zostanie utworzona.

Zasięg zmiennej jest to inaczej czas jej życia - czyli w jakich miejscach kodu JavaScript ją widzi i można z niej korzystać. Pod tym względem dzielimy je na lokalne i globalne. Zmienne lokalne to takie, które są deklarowane wewnątrz funkcji i które giną wraz z zakończeniem działania tej funkcji.

Zmienne globalne natomiast, deklarowane poza funkcją - są dostępne od momentu ich pierwszego użycia aż do końca kodu HTML.

Wyświetlenie kodu HTML w dowolnym elemencie document.getElementById

Każdy element na stronie może posiadac właściwośc ID jednoznacznie go identyfikującą. Wykorzystujemy ją nie tylko przy definiowaniu stylów css, ale również przy odwoływaniu się do elementu w języku JavaScript. Przy pomocy konstrukcji:

document.getElementById(idElementu)

możemy jednoznacznie odwołac się do elementu o id=idElementu. Następnie możemy odwoływać się do właściwości takiego elementu. Najpopularniejsze właściwości to:

- innerHTML – umożliwia określenie kodu HTML który zostanie umieszczony w elemencie
- innerText – umożliwia określenie ciągu znaków który zostanie umieszczony w elemencie
- className – umożliwia określenie klasy CSS użytej do prezentacji elementu.

InnerHTML ,className

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style>
.nowaKlasa{
color:red;
font-size:2em;
}
</style>
<script style="text/javascript">
function zmienTekst(){
document.getElementById("demo").innerHTML="nowy tekst w akapicie";
document.getElementById("demo").className="nowaKlasa";
}
</script>
</head>
<body>
<p>Wykorzystanie "innerHtml" do zmiany zawartości akapitu. Kliknij na poniższy tekst</p>
<p onclick="zmienTekst()"id="demo" >tekst przed zmianą</p>
</body>
</html>
```

Wykorzystanie "innerHtml" do zmiany zawartości akapitu. Kliknij na poniższy tekst

tekst przed zmianą

Wykorzystanie "innerHtml" do zmiany zawartości akapitu. Kliknij na poniższy tekst

nowy tekst w akapicie

Wynik działania funkcji zapisany w znaczniku

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<style>
#demo{
color:red;
font-size:2em;
}
p{font-size:3ex;}
</style>
<script style="text/javascript">
function silnia(n)
{var s=1;
for(var i=1;i<=n;i++)
   {s*=i;}
   return s;
}
function wypiszWynik()
          {
          var n1=Number(prompt("podaj liczbę naturalną",1));
    if(!isNaN(n1) && n1>=0)  {
     document.getElementById("demo").innerHTML=(n1+"!="+silnia(n1));
          }
          else { document.getElementById("demo").innerHTML="niepoprawne dane";}
}
</script>
</head>
<body>
<p>Program oblicza warość funkcji silnia liczby naturalnej. Wynik działania jest zapisany w akapicie.
Ponieważ w skrypcie odwołania do id "demo" występują tylko w funkcjach, skrypt może być umieszczony w
nagłówku dokumentu </head></p>
<p id="demo" ></p>
<button  onclick="wypiszWynik()">kliknij</button>
</body>
</html>
```