MODUŁ 3

Język JavaScript

JavaScript należy do grupy tzw. języków skryptowych. Wspólną cechą charakterystyczną dla tego typu języków jest to, iż programy (funkcje) w nich napisane są wykonywane z poziomu innej aplikacji. Funkcje napisane w JavaScript są uruchamiane przez przeglądarki internetowe, które w tym przypadku są tymi „innymi aplikacjami", stąd JavaScript jest językiem skryptowym.

JavaScript jest także językiem obiektowym. Jest wykorzystywany głównie przez programistów aplikacji internetowych do tworzenia stron interaktywnych. Dzięki funkcjom napisanym w JavaScript strona internetowa może być bardziej dynamiczna, czego nie można osiągnąć przy pomocy samego HTML-a. JavaScript umożliwia tworzenie stron, w których następuje określona reakcja na różnorodne zdarzenia (np. kliknięcie w danym miejscu), pozwala na tworzenie elementów nawigacyjnych, sprawdzanie poprawności danych wpisanych do formularzy, tworzenie tzw. ciasteczek, czy manipulowanie oknami przeglądarki.

W lekcji omówiono wybrane elementy języka JavaScript:

- operatory, instrukcje warunkowe, pętle,
- zmienne tablicowe,
- operacje na datach,
- wyrażenia regularne,
- metody sprawdzania poprawności danych,
- zdarzenia.

Język JavaScript – wprowadzenie

JavaScript jako język skryptowy

JavaScript należy do grupy tzw. języków skryptowych. Wspólną cechą charakterystyczną tego typu języków jest to, iż programy (funkcje) w nich napisane są wykonywane z poziomu innej aplikacji. Funkcje napisane w JavaScript są uruchamiane przez przeglądarki internetowe, które w tym przypadku są tymi „innymi aplikacjami", stąd JavaScript jest językiem skryptowym.
JavaScript jest także językiem obiektowym. Jest wykorzystywany głównie przez programistów aplikacji internetowych do tworzenia stron interaktywnych. Dzięki funkcjom napisanym w JavaScript strona internetowa może być bardziej dynamiczna, czego nie można osiągnąć przy pomocy samego HTML-a. JavaScript umożliwia tworzenie stron, w których następuje określona reakcja na różnorodne zdarzenia (np. kliknięcie w danym miejscu), pozwala na tworzenie elementów nawigacyjnych, sprawdzanie poprawności danych wpisanych do formularzy, tworzenie tzw. ciasteczek czy manipulowanie oknami przeglądarki.

JavaScript - wybrane zasady

Tak, jak w każdym języku programowania, również w JavaScript należy się trzymać pewnych reguł, aby tworzone programy miały poprawną składnię. Do tych reguł należy zaliczyć m.in.:

- instrukcje mogą ale nie muszą kończyć się średnikiem,
- bloki instrukcji są umieszczane w nawiasach klamrowych,
- deklarowanie zmiennych nie jest wymagane,
- wielkość liter w nazwach identyfikatorów jest istotna,
- łańcuchy znakowe są łączone przy pomocy znaku plus,
- komentarze składające się z wielu wierszy są zawarte między znakami /* i */, komentarze umieszczone w jednym wierszu lub jego końcowym fragmencie rozpoczynają się od dwóch ukośników (ang. slash) //

Kod JavaScript w pliku HTML lub pliku zewnętrznym

Choć nie jest to wymóg formalny, dla zachowania porządku na stronie internetowej zaleca się, aby wszystkie funkcje w JavaScript były zdefiniowane w pliku HTML w części nagłówkowej `<head>`, zawsze jednak muszą być umieszczone wewnątrz znacznika `<script>`:

```
<script type="text/javascript"> lub <script language="javascript">
definicje funkcji (skrypty)
</script>
```

Funkcje JavaScript mogą być także umieszczone w pliku zewnętrznym. Aby go dołączyć do dokumentu HTML należy, w sekcji `<head>`, wpisać wiersz:

```
<script language="javascript" src="katalog/nazwa_pliku.js"/>
```

Składnia definicji funkcji w JavaScript

Funkcje definiowane są przy pomocy słowa kluczowego *function*, po którym występuje nazwa funkcji oraz, w nawiasie, lista parametrów formalnych oddzielonych przecinkami. Jeśli funkcja zwraca wyrażenie, należy je wpisać po słowie kluczowym *return*. W jednej funkcji może wystąpić wiele instrukcji *return*, funkcja jest kończona po napotkaniu pierwszej z nich. Składnia funkcji w JavaScript:

```
function nazwa_funkcji (lista parametrów formalnych)
{
instrukcje;
return zwracana_wartość;
}
```

Funkcja jest wywoływana poprzez podanie jej nazwy oraz, w nawiasie, listy parametrów aktualnych (jeśli takowe są wymagane, czyli jeśli w definicji wskazano parametry formalne), z którymi ma ona być wykonana:

```
nazwa_funkcji (lista parametrów aktualnych)
```

Wskaż zdania prawdziwe:

- [x] **W JavaScript deklarowanie zmiennych nie jest wymagane**
- [ ] JavaScript jest językiem skryptowym, gdyż jego kod jest osadzony w kodzie innego języka
- [x] **Jeśli funkcja w JavaScript zwraca wyrażenie, to należy je wpisać w instrukcji return**
- [x] **Instrukcje w JavaScript mogą, ale nie muszą kończyć się średnikiem**
- [x] **W JavaScript wielkość liter w nazwach identyfikatorów jest istotna**
- [ ] Jeśli kod JavaScript jest umieszczony w pliku HTML, to znajduje się wewnątrz znacznika <javascript>
- [ ] Łańcuchy znakowe w JavaScript są łączone przy pomocy znaku &

Operatory, pętle, tablice, operacje na datach

Operatory w JavaScript

W tabeli zestawiono operatory relacyjne, logiczne oraz operator przypisania w JavaScript. Zwróć uwagę szczególne na to, jak tworzy się operatory logiczne:

Nazwa	Składnia	
Operatory relacyjne	==	równy
	!=	różny
	> >=	większy, większy lub równy
	< <=	mniejszy, mniejszy lub równy
Operatory logiczne	&&	iloczyn logiczny (and)
	||	suma logiczna (or)
	!	negacja (not)
Operator przypisania	=	

Operator logiczny w JavaScript to:

- not
- !
- &&
- !!

Pętle w JavaScript

Podobnie, jak w każdym języku programowanie, także w JavaScript istnieje funkcja warunkowa oraz pętle. W poniższej tabeli podano i omówiono instrukcję warunkową oraz wybrane pętle:

Nazwa	Składnia

Składnia instrukcji warunkowej	`if (warunek)`
	`{ instrukcje }`
	`[ else { instrukcje } ]`
	człon umieszczony w nawiasie kwadratowym (`else`) jest opcjonalny

Pętla `WHILE`	`while (warunek)`
	`{`
	`instrukcje`
	`}`
	pętla `while` jest kończona gdy *`warunek`* przyjmie wartość fałszu

Pętla `FOR`	`for (wartość początkowa; warunek zakończenia;`
	`inkrementacja licznika)`
	`{`
	`instrukcje`
	`}`
	Przykład:
	`for (i=1; i<=5; i++)`
	pętla zostanie wykonana pięć razy, kolejno dla `i=1,2,3,4,5`

Pętla `FOR IN`	`for (zmienna in obiekt)`
	`{`
	`instrukcje`
	`}`
	`Obiekt` reprezentuje obiekt lub tablicę. Pętla jest wykonywana tyle razy, ile właściwości ma obiekt lub elementów ma tablica. Podczas kolejnych iteracji pod *`zmienną`* podstawiana jest następna właściwość obiektu lub kolejny element tablicy

Tworzenie obiektów

JavaScript jest językiem obiektowym. Nowy obiekt tworzy się przy pomocy operatora new:

var s = new typ(wartość);

Przykład:
var s = new String('abc');
tworzony jest obiekt o nazwie s klasy (typu) String z wartością początkową 'abc'.

Tablice

Zmienne tablicowe przechowują wiele wartości. Elementy tablicy są numerowane (indeksowane) liczbami całkowitymi począwszy od zera. Tablicom można przypisywać wartość w dowolnym momencie, także w chwili ich tworzenia. Liczba elementów tablicy może się zmieniać w trakcie wykonywania programu. Tablice są obiektami, deklaruje się je w następujący sposób:
`t=new Array();`
Można także, choć nie jest to konieczne, sprecyzować liczbę elementów:
`t=new Array(x);`
w miejsce *x* wpisując liczbę naturalną oznaczającą, ile elementów ma mieć tablica.

Istnieje również możliwość definicji tablicy z wartościami początkowymi:
`var t = new Array(x1,x2,...,xn);`
gdzie:
x1, ..., xn to wartości przypisywane kolejnym indeksom tablicy.
Aby otrzymać element tablicy *t* o indeksie *n*, należy użyć konstrukcji:
`t[n]`

Tablice - przykład:

Utwórz plik HTML i wpisz w nim kod:

```
1 <script language="javascript">
2 var t = new Array(6,2,13,9);
3 x=t[3];
4 document.write(x);
5 </script>
```

Następnie zapisz plik i wyświetl jego zawartość w przeglądarce.
W przykładzie utworzono tablicę o nazwie *t* i przypisano jej cztery wartości, tablica jest więc czteroelementowa (wiersz 2). Następnie, zmiennej *x* przypisano liczbę *9* (wiersz 3): element tablicy *t* o indeksie *3* to element czwarty. Jego wartość w tym przypadku jest równa 9.
W efekcie w przeglądarce pojawi się liczba 9 (ostatnia instrukcja *document.write* wyświetla na ekranie wartość podanego w nawiasie parametru)

Po wykonaniu poniższych instrukcji zmienna x będzie równa:
var t=new Array(2,2,3,4,5)
x=t[3]

- W instrukcjach tych jest błąd
- 2
- **4**
- 3
- 5

Obiekt date - do wykonywania operacji na datach

Operacje na datach

Do wykonywania operacji na datach wykorzystywany jest obiekt
`Date(`*`data`*`)`
gdzie *data* jest równa dacie w postaci łańcucha znakowego albo liczb oddzielonych przecinkami (kolejno: rok, miesiąc, dzień), albo w ogóle nie jest podawana. Jeśli parametr *data* nie zostanie podany, tworzony obiekt będzie miał wartość równą dacie bieżącej. Należy pamiętać, że miesiące są numerowane od zera. Do wykonywania operacji na datach można wykorzystywać metody:
`getDate()` - numer dnia w miesiącu
`getMonth()` - numer miesiąca (miesiące są numerowane od 0–11)
`getFullYear()` - rok czterocyfrowy
`getDay()` - numer dnia w tygodniu (0-niedziela, 1-poniedziałek itd.)
`getHours()` - godzina
`getMinutes()` - minuta
`getSeconds()` – sekunda

Operacje na datach - przykład

Utwórz plik HTML o następującej zawartości:

```
1 <script language="javascript">
2 var d = new Date();
3 godzina = d.getHours();
4 minuta = d.getMinutes();
5 document.write (godzina+":"+minuta)
6 </script>
```

Najpierw utworzono obiekt o nazwie *d*, któremu przypisano bieżącą datę (wiersz 2). Następnie zmiennej *godzina* została przypisana wartość równa aktualnej godzinie (wiersz 3) a zmiennej minuta aktualnej minucie (wiersz 4). Następnie instrukcja *document.write* wyświetla na ekranie aktualną godzinę i minutę oddzielone dwukropkiem (wiersz 5).

Wyrażenia regularne

Wyrażenia regularne - wprowadzenie

Wyrażenia regularne są wykorzystywane m.in. do sprawdzania, czy wartość tekstowa (na przykład wprowadzona przez użytkownika w formularzu) ma poprawny format. W celu wykonania takiej operacji należy najpierw utworzyć wzorzec, któremu odpowiadać ma sprawdzana wartość. Wzorzec zaczyna się i kończy ukośnikiem (/). Po ukośniku otwierającym dobrze jest wstawić znak daszka (^), a przed zamykającym znak dolara ($), wtedy dokonane zostanie tzw. zakotwiczenie, dzięki czemu nie będzie możliwe wpisanie innych znaków przed lub po sprawdzanym ciągu (jeśli takie znaki się pojawią, łańcuch znakowy zostanie potraktowany jako nie pasujący do wzorca).

Zasady tworzenia wyrażeń regularnych

W tabeli zestawiono wybrane zasady tworzenia wyrażeń regularnych:

Znaki	Opis	Objaśnienie lub przykład
`.` (kropka)	oznacza dowolny znak	
`\` (*backslash*)	poprzedza znaki specjalne (m.in. `^$\/?*.+[]{}()`), które w wyrażeniach regularnych pełnią określone funkcje, aby więc wskazać, że chodzi dokładnie o te znaki, należy je poprzedzić *backslashem*	
`[]` (nawiasy kwadratowe)	wewnątrz nawiasów kwadratowych umieszcza się znaki dopuszczalne	`a[0-9]` odpowiada ciągowi składającemu się z litery `a` oraz dowolnej cyfry
`^` (daszek)	oznacza „za wyjątkiem", musi być umieszczony jako pierwszy znak w nawiasie kwadratowym	`[^0-9a-z]` oznacza dowolny znak oprócz cyfry i małej litery alfabetu łacińskiego

sekwencje ze znakiem `\`:
`\d`
`\D`
`\w`
`\W`
	
	dowolna cyfra (to samo co `[0-9]`)
	dowolny znak oprócz cyfry: `[^0-9]`
	cyfra, litera lub podkreślenie `[0-9a-zA-Z_]`
	dowolny znak, byle nie cyfra, litera lub znak podkreślenia `[^0-9a-zA-Z_]`	

Krotności występowania znaków:
`?`
 	
`*`
`+`
`{i,j}`
`{i,}`
`{i}`
	
	brak lub pojedyncze wystąpienie
	
	dowolna liczba znaków, także zero
	dowolna liczba znaków, co najmniej jeden
	minimalna `i` oraz maksymalna `j` liczba znaków
	minimalna liczba `i`, maksymalna dowolna
	dokładnie `i` znaków
		Krotności występowania znaków podaje się po informacji, jakie znaki mogą być użyte, np.:
		`[0-9]{1,4}`
		odpowiada ciągowi 1–4 cyfr
		`\w+`
		ciąg znaków odpowiadający temu wzorcowi składa się z minimum jednego (znak +) znaku cyfry, litery lub podkreślenia (`\w`)

`|` (*pipeline*)	alternatywa	aby można było wpisać albo znak małpy albo `(at)` należy użyć alternatywy:
		`@|\(at\)`
		aby możliwe było wpisanie cyfry `0` lub `9` należy użyć wyrażenia:
		`0|9`

Wyrażenia regularne - przykład

Oto przykład wyrażenia regularnego wraz z objaśnieniem:

```
wzor=/^[0-9a-zA-Z\-_]+\@[0-9a-zA-Z]{2,3}$/;
```

Objaśnienie:

`wzor`	`/^`	`[0-9a-zA-Z\-_]+`	`\@`	`[0-9a-zA-Z.]{2,3}`	`$/`
Nazwa zmiennej, której przypisane jest wyrażenie regularne	Początek wzorca	Ciąg składający się co najmniej z jednego znaku (+), którym może być cyfra, litera, myślnik i znak podkreślenia	Znak `@`	Dwa lub trzy znaki, którymi mogą być cyfry, litery albo kropki	Koniec wzorca

[redacted]

W wyrażeniach regularnych dopuszczalne znaki umieszcza się:

[redacted]

- w nawiasach kwadratowych
- w nawiasach klamrowych
- między ukośnikami
- w nawiasach okrągłych

Zastosowanie wyrażeń regularnych

Wyrażenia regularne są wykorzystywane m.in. do sprawdzenia poprawności danych wpisanych w formularzach. Sprawdzenie takie następuje przed wysłaniem danych na serwer. Załóżmy, że w polu tekstowym formularza użytkownik ma wpisać adres e-mailowy. Jak wiadomo adres taki, aby miał poprawną składnię, musi posiadać dokładnie jeden znak @, który nie może znajdować się na początku ani na końcu, przynajmniej jedną kropkę po nim, ale nie bezpośrednio i nie na końcu, itd. Jeśli, świadomie lub nie, użytkownik popełni błąd, taki e-mail powinien on zostać odrzucony. Jednym z rozwiązań jest sprawdzenie składni bezpośrednio w przeglądarce internetowej, a więc po stronie klienta i, jeśli wpisany tekst ma poprawną formę, przesłanie go na serwer, a jeśli nie, poinformowanie użytkownika o błędzie i żądanie jego skorygowania.

Sprawdzanie danych w formularzach

Aby sprawdzić, czy wartość wpisana w formularzu pasuje do zdefiniowanego wzorca należy użyć instrukcji:
`wynik = document.`*`nazwa_formularza`*`.`*`nazwa_pola`*`.value.match(`*`wzor`*`);`
gdzie:
`wzor` - zmienna zawierająca zdefiniowany wzór w postaci wyrażenia regularnego,
`nazwa_formularza` - wartość parametru *name* formularza, w którym zdefiniowano pole o nazwie *nazwa_pola*,
`value` - właściwość wskazująca, że z pola, którego nazwa została podana jako *nazwa_pola* należy pobrać jego wartość (*value*),
`match` - właściwość sprawdzająca dopasowanie do wzorca podanego w nawiasie (przypisanego zmiennej *wzor*),
`wynik` - zmienna, której przypisywany jest wynik wyrażenia i która otrzyma wartość TRUE lub FALSE: TRUE, jeśli wartość wpisana do pola *nazwa_pola* zdefiniowanego w formularzu *nazwa_formularza* pasuje do wzorca przypisanego zmiennej *wzor*, FALSE w przeciwnym przypadku.

Sprawdzanie danych w formularzach, c.d.

Niejednokrotnie wyrażenie ścieżkowe wskazujące dostęp do obiektów i ich właściwości można zapisać w prostszy sposób. Jeśli występuje w nim odwołanie do pola, z którego uruchamiana jest funkcja, można użyć skróconego wyrażenia ścieżkowego i zamiast
`document.`*`nazwa_formularza`*`.`*`nazwa_pola`*`.value`
wystarczy wtedy
`nazwa_pola``.value`

Sprawdzanie danych w formularzach - przykład

```
wyr_reg = /^[a-zA-Z]{5,20}\.\D{4,}$/;
poprawnie = document.symbole.wiertarka.value.match(wyr_reg);
```

W pierwszej instrukcji zmiennej *wyr_reg* przypisano wyrażenie regularne:
`/^` - początek wyrażenia regularnego,
`[a-zA-Z]` - użytkownik może wpisać tylko małe lub duże litery alfabetu łacińskiego (bez „polskich” znaków),

`{5,20}` - tych liter musi być minimum 5, maksimum 20,
`\.` - następnie użytkownik musi wpisać kropkę. W wyrażeniu regularnym kropka musi być poprzedzona znakiem \, gdyż jest ona specjalnym znakiem,
`\D{4,}` - po kropce muszą być minimum cztery dowolne znaki oprócz cyfry (\D),
`$/` - znaki końca wyrażenia regularnego.
Przy pomocy drugiej instrukcji zostanie sprawdzone, czy w polu o nazwie *wiertarka* formularza, który ma nazwę *symbole* wpisano wyrażenie, które pasuje do wzorca przypisanego zmiennej *wyr_reg*. Jeśli tak, zmienna *poprawnie* otrzyma wartość prawdy (TRUE), w przeciwnym przypadku zostanie jej przypisany fałsz (FALSE).

Instrukcja
wynik=document.zamowienie.data_zamowienia.value.match(wzor)
przypisze zmiennej wynik:

- pole data_zamowienia formularza zamowienie
- **wartość logiczną**
- wartość pola data_zamowienia formularza zamowienie, jeśli wartość ta będzie pasowała do wzoru
- wartość pola data_zamowienia formularza zamowienie

Okna dialogowe. Sprawdzanie poprawności danych

Okna dialogowe

W języku JavaScript do komunikacji z użytkownikami wykorzystywane są m.in. okna informacyjne i dialogowe. Pojawiają się one na ekranie w przypadku zaistnienia jakiegoś zdarzenia:

- okno informacyjne z przyciskiem OK, wyświetlające tekst podany w nawiasie:
  `alert("łańcuch znakowy")`
- okno dialogowe z przyciskami OK i Cancel, wyświetlające tekst podany w nawiasie:
  `confirm("łańcuch znakowy")`

W przypadku okna dialogowego (confirm) zwracana jest wartość TRUE, gdy kliknięty zostanie przycisk OK lub FALSE, gdy kliknięty będzie przycisk Cancel.

Przykład wykorzystania okien dialogowych

W pliku HTML wpisz następujący kod, a następnie otwórz ten plik w przeglądarce:

```
1 <script language="javascript">
2 alert("Okno Alert");
3 x=confirm("Okno dialogowe Confirm");
4 document.write(x);
5 </script>
```

Po wyświetleniu zawartości strony w przeglądarce, najpierw pojawi się okno informacyjne z tekstem *Okno Alert* (wiersz 2). Po kliknięciu OK, na ekranie pojawi się okno dialogowe, w którym użytkownik może podjąć decyzję klikając OK lub Cancel. Efekt tego kliknięcia jest przypisywany zmiennej *x* (wiersz 3). Jeśli użytkownik kliknie OK, zmiennej *x* przypisana zostanie wartość *true*, jeśli kliknie Cancel, zmienna *x*

otrzyma wartość *false*. Następnie zawartość zmiennej *x* jest wyświetlana na ekranie (wiersz 4). Na ekranie pojawi się więc napis false lub true, zależnie od tego, co zostanie kliknięte w oknie dialogowym.

W wyniku wykonania skryptu z ostatniego przykładu zmiennej x została przypisana wartość:

- alert
- confirm
- **logiczna**
- Okno dialogowe confirm

Sprawdzanie poprawności danych

Oprócz wyrażeń regularnych, poprawność danych wpisanych lub wybranych w polach formularza można sprawdzić stosując inne metody. Poznajmy niektóre z nich.
Wszystkie przykłady zamieszczone w lekcji wpisuj w pliku tekstowym z rozszerzeniem HTML i wyświetlaj w przeglądarce, aby sprawdzić ich działanie.

Sprawdzenie, czy wyrażenie (zwykle tekst) jest liczbą

```
isNaN(wyrażenie)
```
Jeśli wyrażenie podane w nawiasie nie jest liczbą (Not a Number), funkcja zwraca TRUE, w przeciwnym przypadku FALSE.

Przykład:
```
1 <script language="javascript">
2 document.write(isNaN('23.4')+'<br>');
3 document.write(isNaN('23a4'));
4 </script>
```
Ponieważ wyrażenie 23.4 będące parametrem funkcji *isNaN* w wierszu 2 jest liczbą, wynikiem działania funkcji będzie false i taki napis pojawi się na ekranie. W wierszu 3 napis 23a4 nie jest liczbą, więc wynikiem działania *isNaN* jest true. Użyty znacznik
 na końcu wiersza 2 spowoduje, że napisy false i true pojawią się w przeglądarce w osobnych liniach.

Zamiana wyrażenia na liczbę całkowitą

```
parseInt(wyrażenie)
```
Funkcja *parseInt* zamienia wyrażenie (zwykle łańcuch znakowy) na liczbę całkowitą. Jeśli wykonanie operacji jest niemożliwe, zwraca NaN (Not a Number). Jeśli na początku wyrażenia jest liczba całkowita, funkcja ją zwraca.

Przykład:
```
1 <script language="javascript">
2 s="a123.45";
3 if (!parseInt(s))
4 alert("To nie jest liczba całkowita");
5 else alert(parseInt(s));
6 </script>
```

Na ekranie pojawi się okienko informacyjne z napisem *To nie jest liczba całkowita*, gdyż początek tekstu przypisanego zmiennej *s* (wiersz 2) nie może zostać zamieniony na liczbę całkowitą, ponieważ zaczyna się od litery.

Przykład:
```
1 <script language="javascript">
2 s="123.45";
3 if (!parseInt(s))
4 alert("To nie jest liczba całkowita");
5 else alert(parseInt(s));
6 </script>
```
Funkcja *parseInt(s)* pobiera tyle początkowych znaków zmiennej *s*, ile tworzy poprawną liczbę całkowitą (wiersz 3). Zwróci więc liczbę 123. W związku z tym warunek instrukcji if jest fałszywy (wiersz 3), wykona się więc instrukcja występująca po else (wiersz 5). Ostatecznie na ekranie pojawi się okienko informacyjne, a w nim liczba 123.

parseInt(wyrażenie)

- **zamienia wyrażenie na liczbę całkowitą**
- zawsze zwraca liczbę całkowitą
- sprawdza czy wyrażenie jest liczbą całkowitą
- zwraca wartość logiczną

Wybieranie fragmentu łańcucha znakowego

`s.substring(m,n)`
Wyrażenie zwraca łańcuch znaków pobrany ze zmiennej *s* (typu String) od znaku m do n-1 (znaki są numerowane od zera).

Przykład:
```
1 <script language="javascript">
2 s='abcdefgh'
3 document.write(s.substring(2,5))
4 </script>
```
Na ekranie pojawi się napis **cde**: znak o indeksie 2 to trzeci znak tekstu przypisanego zmiennej s. Znak o indeksie 4 (5−1) to e.

Wskaż, co pojawi się na ekranie w wyniku wykonania poniższych instrukcji:
s='12345678'
document.write(s.substring(3,6))

- 3456
- 4567

- **456**
- 567

Zmiana wielkości liter

`toUpperCase('łańcuch_znakowy')`
W wyrażeniu *łańcuch_znakowy* funkcja zamienia wszystkie litery na duże.

`toLowerCase('łańcuch_znakowy')`
W wyrażeniu *łańcuch_znakowy* funkcja zamienia wszystkie litery na małe.

Wyszukiwanie podciągu w łańcuchu znakowym

Metoda `indexOf` użyta dla obiektu typu String zwraca liczbę wskazującą, w którym miejscu łańcucha znakowego znajduje się podany znak (lub ciąg znaków). Jeśli znak (ciąg) nie występuje w łańcuchu, zwracana jest liczba –1.

Przykład:
```
1 <script language="javascript">
2 var s = new String("abcd#ef");
3 document.write(s.indexOf("#"));
4 </script>
```
W przeglądarce wyświetlona zostanie liczba 4 (znak # jest piątym elementem łańcucha *s*, więc ma numer 4).

Przykład:
```
1 <script language="javascript">
2 s='123abc'
3 document.write(s.indexOf('ab',0))
4 </script>
```
Drugi parametr metody *indexOf* (wiersz 3) równy 0 oznacza, że przeszukiwanie zmiennej *s* ma się rozpocząć od początku znajdującego się w niej tekstu (0 to pierwszy znak).
W efekcie na ekranie pojawi się liczba 3 (ciąg 'ab' zaczyna się od czwartego znaku ciągu '123abc' znajdującego się w zmiennej *s*, a czwarty znak ma numer 3).

Wskaż, co pojawi się na ekranie w wyniku wykonania poniższych instrukcji:
```
s='abcdeedcba'
document.write(s.indexOf('d',4))
```

- 3
- 4
- **6**
- eedc

Zdarzenia

Zdarzenia

Funkcje języka JavaScript są często wykorzystywane do podjęcia określonej akcji w odpowiedzi na wystąpienie zdarzeń lub wykonanie działań przez użytkowników. W lekcji omówione zostały wybrane zdarzenia.

Zdarzenia w JavaScript

Wybrane zdarzenia obsługiwane przez JavaScript:

- **onFocus**
  Zdarzenie to oznacza uaktywnienie obiektu (uzyskanie przez niego tzw. fokusu), co ma miejsce, gdy obiekt zostanie kliknięty lub w inny sposób (na przykład przez przyciśnięcie klawisza TAB) uaktywniony
- **onBlur**
  dezaktywacja obiektu. Zdarzenie takie zachodzi, gdy obiekt utraci fokus (jest dezaktywowany)
- **onChange**
  dokonanie zmiany w obiekcie lub dezaktywacja obiektu
- **onClick** (**onDblClick**)
  kliknięcie (podwójne kliknięcie) obiektu
- **onSubmit**
  zdarzenie oznaczające próbę wysłania formularza (kliknięcie przycisku Submit) do przetwarzania
- **onMouseOver**
  umieszczenie kursora myszy nad obiektem
- **onMouseOut**
  przesunięcie kursora myszy poza wskazywany obiekt

Zdarzenia są bardzo często związane z formularzami lub ich elementami, choć nie zawsze. Zobacz przykłady...

Zdarzenie onChange - przykład

```
1 <script language="javascript">
2 function fkolor(k)
3 {
4 alert("Został wybrany kolor "+k.value)
5 }
6 </script>
7 <select name="kolory" onChange="fkolor(this)">
8 <option value="czerwony" selected>czerwony</option>
9 <option value="zielony">zielony</option>
10 <option value="niebieski">niebieski</option>
11 </select>
```

W sekcji `<script>` (wiersze 1-6) zdefiniowano funkcję *fkolor* (wiersze 2-5), która wyświetla okno dialogowe a w nim tekst oraz wartość (*value*) parametru *k*.
Poniżej zdefiniowano listę (wiersze 7-11), z której użytkownik może wybrać jedną z trzech opcji. Jeśli dokona zmiany na liście (wybierze inny kolor), zajdzie zdarzenie *onChange* i uruchomiona zostanie funkcja *fkolor* z parametrem *this* (wiersz 7).
Ostatecznie, jeśli użytkownik dokona zmiany na wyświetlonej liście, uruchomiona zostanie funkcja *fkolor* i na ekranie pojawi się okienko informacyjne. Jeśli użytkownik kliknie listę lecz nie zmieni koloru, nie zajdzie zdarzenie *onChange*, funkcja nie zostanie uruchomiona i okienko informacyjne nie zostanie wyświetlone.

Użyty w wierszu 7 parametr *this* oznacza nazwę obiektu (w tym przypadku kontrolki formularza), w którego definicji jest używany (w tym przypadku jest to pole formularza, czyli lista o nazwie *kolory*). Identyczny efekt dałoby więc wpisanie w miejsce *this* nazwy elementu, czyli *kolory*. Jednak zgodnie z tzw. zasadą *dry* (*don't repeat yourself*), jeśli nie jest to niezbędne, nie powinno się powtarzać nazw, a w ich miejsce należy używać właśnie *this.*

Zdarzenie onMouseOver - przykład

Zdarzenia są najczęściej wykorzystywane do podejmowania jakiejś akcji w wyniku wykonania określonego działania przez użytkownika w formularzu lub umieszczonych w nim kontrolkach. Ale nie jest to jedyna możliwość. Przeanalizuj przykład:

```
1 <script language="javascript">
2 function fon()
3 {
4 alert("Kursor został umieszczony nad tekstem")
5 }
6 </script>
7 <p onMouseOver ="fon()">Test</p>
```

W sekcji <script> (wiersze 1-6) zdefiniowano bezparametrową funkcję o nazwie *fon*. Poniżej (wiersz 7) utworzono akapit (znacznik <p>) i umieszczono w nim przykładowy tekst. Parametrem znacznika <p> jest *onMouseOver* z wartością równą nazwie funkcji *fon*. Jeśli kursor myszy zostanie umieszczony nad tekstem (bez klikania), zajdzie zdarzenie *onMouseOver* i uruchomiona zostanie funkcja *fon*, która wyświetli na ekranie okienko informacyjne.
Zmieniaj nazwę zdarzenia w wierszu 7 (np. na *onMouseOut, onClick, onDblClick*) i obserwuj efekt.

Zdarzenia i właściwości pól formularza - przykład

```
1 <script language="javascript">
2 function sprawdz(d)
3 {
4 info=""
5 if (d.pesel.value.length != 11)
6 info="Błędna długość numeru PESEL\n"
7 wzor=/^\d{11}$/
8 if (! d.pesel.value.match(wzor))
9 info=info+"PESEL musi się składać z 11 cyfr\n"
10 if (! d.zgoda.checked)
11 info=info+"Nie zaznaczono pola wyboru\n"
12 if (info)
13 alert(info)
14 else
15 alert("Wprowadzone dane są poprawne")
16 }
17 </script>
18 <form name=dane onSubmit="sprawdz(this)">
19 <input type="text" name="pesel"><br>
20 <input type="checkbox"name="zgoda"><br>
21 <input type="submit" value="Wykonaj">
22 </form>
```

W przykładzie utworzono formularz (wiersze 18-22) i umieszczono w nim trzy kontrolki: pole tekstowe o nazwie *pesel* (wiersz 19), pole wyboru o nazwie *zgoda* (wiersz 20) oraz przycisk polecenia *submit* (wiersz 21). W nagłówku formularza (wiersz 18) wskazano, że jeśli zajdzie zdarzenie *onSubmit*, co oznacza, że kliknięty zostanie przycisk *submit*, uruchomiona ma zostać funkcja *sprawdz(this)*. Parametr *this* oznacza w

tym przypadku nazwę formularza.
Funkcja została zdefiniowana w sekcji `<script>` (w wierszach 2-16). Najpierw inicjowana jest zmienna *info* (wiersz 4), w której umieszczane będą kolejne teksty, jeśli w danych wpisanych lub wybranych w formularzu pojawią się błędy. Następnie instrukcja *if* (wiersze 5-6) sprawdza, czy długość (*length*) wartości (*value*) pola formularza *d* o nazwie *pesel* nie jest równa 11. Jeśli nie jest, do zmiennej *info* dodawany jest tekst "Błędna długość..". Tekst ten jest zakończony znakami *\n*, co oznacza, że następny tekst będzie wyświetlany od nowej linii.
W wierszu 7 zmiennej *wzor* przypisywane jest wyrażenie regularne oznaczające tekst o długości 11 cyfr, a kolejna instrukcja *if* (wiersze 8-9) dołącza do zmiennej *info* kolejny tekst, jeśli wartość (*value*) pola *pesel* nie pasuje do wzorca.
Trzecia instrukcja *if* (wiersze 10-11) dodaje do zmiennej *info* kolejny tekst, ale tylko w przypadku, gdy nie zaznaczone zostanie (właściwość *checked*) pole *zgoda* formularza *d*.
Ostatnia instrukcja *if* (wiersze 12-15) wyświetla na ekranie zawartość zmiennej *info*, gdy jest ona niepusta lub tekst "Wprowadzone dane są poprawne", gdy zmienna *info* jest pusta, a więc żaden tekst nie został jej przypisany, co w praktyce oznacza, że dane wprowadzone w formularzu są poprawne.

W poniższym tekście w miejsca oznaczone od (1) do (7) wstaw odpowiednie określenia spośród następujących:
a) onBlur
b) onChange
c) onClick
d) onFocus
e) onMouseClick
f) onMouseOut
g) onMouseOver
h) onMouseOn
i) onSend
j) onSubmit
Jako odpowiedź na pytanie wpisz ciąg znaków (bez spacji, wielkość liter jest nieistotna) oznaczających kolejno wstawiane wyrazy. Przykładowa odpowiedź: cbeda
oznacza:1c, 2b, 3e, 4d, itd.

Uaktywnienie obiektu to zdarzenie (1). Dezaktywacja obiektu to zdarzenie (2). Dokonanie zmiany w obiekcie lub jego dezaktywacja to zdarzenie (3). Kliknięcie przycisku wysyłającego dane z formularza na serwer to zdarzenie (4). Umieszczenie kursora myszy nad obiektem bez klikania to zdarzenie (5). Przesunięcie kursora myszy poza wskazywany obiekt to zdarzenie (6). Kliknięcie na obiekcie to zdarzenie (7).

Odp:dabjgfc

Sprawdzanie poprawności danych w formularzach - przykład

Obsługa zdarzeń - przykład

W przykładzie utworzono pole formularza służące do wpisania dwucyfrowej liczby całkowitej oraz napisano funkcję, która sprawdza, czy wpisany tekst jest poprawną liczbą (jedna lub dwie cyfry) oraz czy liczba ta należy do przedziału <5−20>.

```
1 <script type="text/javascript">
2 function fliczba(x) {
3 wzor = /^[0-9]{1,2}$/;
4 if (! x.value.match(wzor)) {
5 alert("Wpisany ciąg nie jest liczbą całkowitą");
6 return false; }
```

```
7 if (x.value < 5 || x.value > 20) {
8 alert("Liczba musi być z przedziału <5-20>");
9 return false; }
10 return true; }
11 </script>
12 <body>
13 <form>
14 Wpisz liczbę całkowitą z przedziału 5-20:
15 <input type="text" name="liczba" maxlength="2"
onBlur="fliczba(this)"/>
16 </form>
17 </body>
```

W sekcji `<script>` zdefiniowano funkcję *fliczba* (wiersze 2−10). Najpierw zmiennej *wzor* (wiersz 3) przypisano wyrażenie regularne – odpowiada mu ciąg znaków składających się z jednej lub dwóch cyfr. Warunek w instrukcji *if* (wiersz 4) będzie prawdziwy, gdy wartość (*value*) obiektu *x* nie będzie zgodna ze wzorcem znajdującym się w zmiennej *wzor* − wtedy na ekranie pojawi się komunikat (wiersz 5) i funkcja skończy działanie zwracając FALSE (wiersz 6). Jeśli warunek pierwszej instrukcji *if* będzie fałszywy, wiadomo, że wartość *x* jest liczbą jedno lub dwucyfrową (w przeciwnym przypadku pierwsza instrukcja *if* doprowadzi do zakończenia działania funkcji). Warunek logiczny w drugiej instrukcji *if* (wiersz 7) będzie prawdziwy, gdy wartość (*value*) wpisana do zmiennej *x* będzie spoza przedziału <5−20>. Wtedy na ekranie pojawi się stosowny komunikat (wiersz 8) i funkcja skończy działanie zwracając FALSE (wiersz 9). Jeśli warunki w obydwu instrukcjach *if* będą fałszywe, oznacza to, że w polu formularza wpisano liczbę i że jest ona z przedziału <5−20>. W takiej sytuacji funkcja zwraca TRUE (wiersz 10). Funkcja *fliczba* jest wywoływana z parametrem *this* w wyniku dezaktywacji pola tekstowego o nazwie *liczba* (wiersz 15). Aby sprawdzić działanie funkcji, po wpisaniu wartości do pola tekstowego należy przycisnąć klawisz Tab (nie Enter).