Podstawy języka JavaScript

JavaScript jakby nie patrzeć jest podobnym językiem do HTML (hipertekstowy język znaczników) ale ma inne zastosowania niż wspomniany język.

Jest to pełnoprawny, skryptowy język programowania, w którym możemy zastosować cały repertuar klasycznych konstrukcji językowych, takich jak: instrukcje warunkowe, pętle, zmienne, tablice, instrukcje wyboru, własne funkcje, klasy, obiekty itd.

Podstawowe pojęcia do JavaScript

- Hoisting zmiennej - mechanizm tzw. windowania zasięgu zmiennych w języku JavaScript; zmienna stworzona z użyciem klauzuli var, np. wewnątrz funkcji, będzie widoczna nie od momentu jej deklaracji, lecz od samego początku istnienia funkcji, co prowadzi do niepożądanej sytuacji, w której zmiennej można użyć jeszcze przed jej deklaracją.

- Front-end - ogół technologii webowych (HTML, CSS oraz w wielu zastosowaniach JavaScript), w których kody źródłowe wykonywane są przez przeglądarkę internetową klienta witryny oraz pozostają jawne, czyli może do nich zajrzeć każdy użytkownik serwisu, niezależnie od posiadanych uprawnień

  Parsowanie - proces konwersji tekstowych danych wejściowych pobranych z pliku źródłowego, na odpowiadającą mu wyjściową strukturę obiektową; w procesie tym siłą rzeczy sprawdzana jest poprawność składniowa kodu, czyli zgodność z gramatyką danego języka programowania

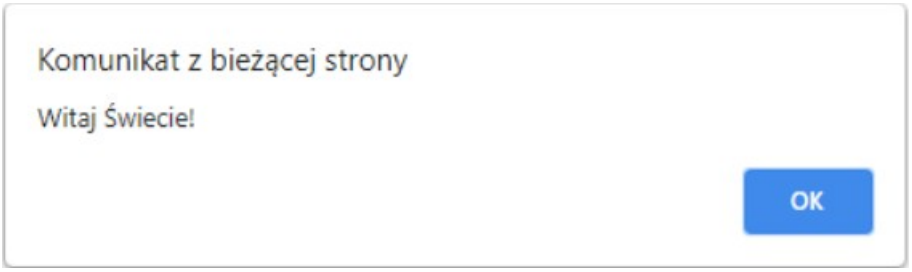
Umieszczanie skryptu bezpośrednio do dokumentu

Możliwe jest umieszczenie kodu źródłowego skryptu wprost w dokumencie HTML - wpisujemy go wtedy pomiędzy dodatkowe znaczniki :

```
<script>
   alert("Witaj Świecie!");
   //kod JS znajduje się pomiędzy znacznikami script
</script>
```

To właśnie dzięki nim przeglądarka potrafi odróżnić fragment kodu napisanego w JavaScript od zapisów, które należy interpretować jako język HTML.

W samym skrypcie użyto funkcji alert(), która wyświetla przesłany jej pomiędzy nawiasami tekst w małym oknie dialogowym generowanym przez przeglądarkę:



Skrypt osadzony z zewnętrznego pliku

Możemy również umieścić kod źródłowy skryptu w zewnętrznym pliku z rozszerzeniem .js oraz zaimplementować go w dokumencie HTML:

```
1 <script src="plik.js">
```

Zaletą tego rozwiązania jest wyraźne oddzielenie znaczników HTML od kodu JavaScript, do czego powinien zawsze dążyć programista odpowiedzialny za front-end.

- Atrybut async

W wypadku osadzenia skryptu w dokumencie zewnętrznym możemy zastosować atrybut async wprowadzony w standardzie HTML5.

```
1 <script src="plik.js" async>
```

Zachowaniem domyślnym przeglądarki, gdy napotka zewnętrzny skrypt na stronie, jest parsowanie i wykonanie skryptu natychmiast w linii dołączenia, zanim kontynuowane będzie czytanie dalszych linii źródła.

Obecność atrybutu async zmienia tę sytuację – parsowanie HTML nie zostaje zatrzymane, a pobieranie skryptu odbywa się w tym samym czasie co parsowanie kolejnych linii HTML.

Dopiero kiedy skrypt zostanie już w całości pobrany i przygotowany do wykonania, przeglądarka zatrzyma przetwarzanie kodu HTML i wykona skrypt.

- Atrybut defer

Natomiast w sytuacji, w której nie ustawimy atrybutu async, a wprowadzimy atrybut o nazwie defer:

```
1 <script src="plik.js" defer>
```

Przeglądarka wykona zewnętrzny skrypt dopiero wtedy, gdy kod źródłowy witryny zostanie w całości wczytany.

Podstawowe operatory w JavaScript

Operator	Działanie operatora
+	dodawanie liczb oraz łączenie łańcuchów (konkatenacja)
-	odejmowanie liczb
*	mnożenie liczb
/	dzielenie liczb
++	inkrementacja (zwiększenie wartości liczby dokładnie o 1)
--	dekrementacja (zmniejszenie wartości liczby dokładnie o 1)
%	reszta z dzielenia (tzw. modulo)
=	przypisanie wartości do zmiennej
==	porównanie wartości
<	mniejsze od
<=	mniejsze od lub równe
>	większe od
>=	większe od lub równe
!	zaprzeczenie (negacja)

Operator	Działanie operatora
&&	logiczne „i” (AND)
||	logiczne „lub” (OR)
+=	skrócony zapis, np. a += b odpowiada: a = a + b
-=	skrócony zapis, np. a -= b odpowiada: a = a - b
*=	skrócony zapis, np. a *= b odpowiada: a = a * b
/=	skrócony zapis, np. a /= b odpowiada: a = a / b
%=	skrócony zapis, np. a %= b odpowiada: a = a % b

Deklarowanie zmiennych

Zmienna to (najprościej mówiąc) pojemnik na dane – możemy w niej przechować np. liczbę, napis, wartość logiczną. Stosujemy ją, gdy potrzebujemy zapamiętać w skrypcie jakąś wartość. Informacja ta przechowywana jest w pamięci RAM komputera.

Zmienne w języku JavaScript deklarujemy z użyciem klauzuli var (ang. variable – zmienna), po której wpisujemy nazwę zmiennej:

```
1 <script>
2   var ile_jablek = 12;
3 </script>
```

Tworząc nazwę zmiennej, nie możemy:

- rozpocząć jej od cyfry (przez wzgląd na szesnastkowy zapis liczb),
- użyć spacji; jeżeli chcemy, aby nazwa składała się z kilku słów, stosujemy znak podkreślenia (np. ile_jablek, a nie: ile - jablek),
- wykorzystywać słów kluczowych języka JavaScript.

Powinniśmy także unikać polskich znaków diakrytycznych (choć nie jest to zakazane). Warto natomiast stosować angielskie nazewnictwo zmiennych, co usprawnia pracę w międzynarodowych zespołach programistów.

W języku JavaScript możemy zadeklarować zmienną, używając klauzuli let (ang. zezwalać):

```
<script>
  let liczba = 20;
</script>
```

Funkcje i parametry funkcji

Funkcja to wydzielony, autonomiczny fragment kodu spełniający określone zadanie. Możemy używać gotowych, wbudowanych w JavaScript funkcji lub tworzyć własne.

Ten zdefiniowany fragment kodu wykona się, jeśli zostanie wywołany do wykonania w konkretnym, zaplanowanym przez programistę miejscu w kodzie, które nazywamy wywołaniem funkcji.

Funkcję wywołujemy, zamykając jej nazwę z obu stron okrągłymi nawiasami - dzięki nim wiadomo, że chodzi o funkcję, a nie o np. zmienną:

```
<script>
  // Definicja funkcji
  function iloczyn(a, b)
  {
    return a * b;
    // Zwróć iloczyn wartości parametrów
  }

  var x = 3;
  var y = 4;

  // Wywołanie w wybranym przez nas miejscu w kodzie
  var wynik = iloczyn(x, y);

  // Pokaż wynik wywołania w wyskakującym oknie
  alert(wynik);
</script>
```

Wynikiem działania funkcji dla powyższego wywołania będzie wartość 12. Zmienne umieszczone wewnątrz nawiasów okrągłych funkcji to jej parametry. Analogicznie jak w matematycznym zapisie funkcji f(x) parametrem (lub inaczej argumentem) funkcji f jest wartość x.

Co ważne – wartości: 3 i 4 umieszczone w zmiennych: x, y występujących w wywołaniu, funkcja na własne potrzeby nazywa: a, b. Oznacza to, że funkcja działa na kopiach zmiennych x, y.

Rozwiązanie, w którym domyślnie parametr funkcji jest jedynie kopią przekazanej do funkcji oryginalnej wartości, ma wiele zalet. Dzięki temu oryginalne wartości nie ulegną przypadkowej podmianie.

W ten sposób zwiększamy bezpieczeństwo danych i oszczędzamy czas, który trzeba by było poświęcić na szukanie ewentualnych skutków zamiany wartości w zmiennych wysyłanych do funkcji.

Możemy też łatwiej przenieść własną funkcję do innych źródeł, gdyż jest całkowicie niezależna od nazw zmiennych spoza jej zasięgu.