Nazwa implementacji: JavaScript

**Autor:** michal.czyzewski

Opis implementacji:

Stworzymy aplikację w języku JavaScript.

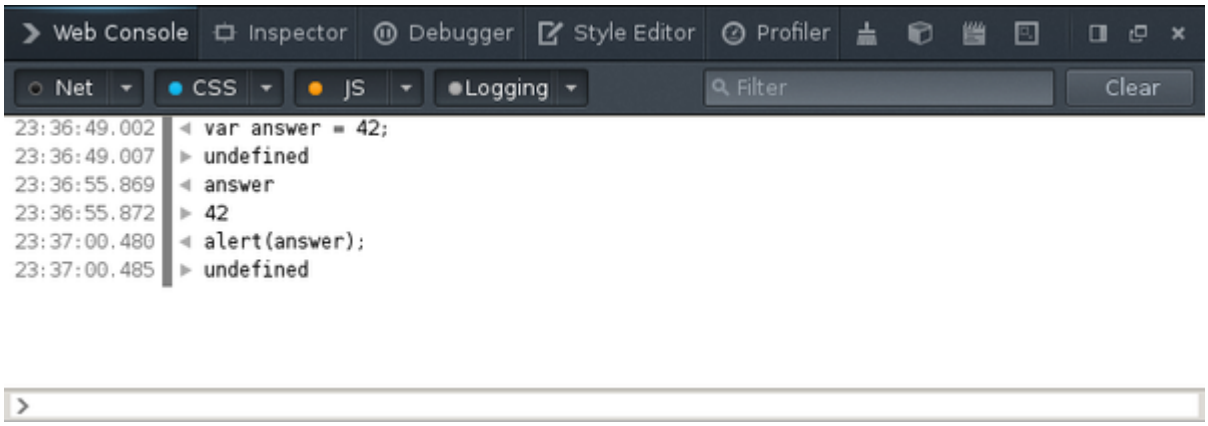
JavaScript jest jedynym językiem programowania, w którym możemy stworzyć aplikację uruchamianą w przeglądarce internetowej. Kod takiego programu najczęściej umieszczamy w oddzielnym pliku z rozszerzeniem .js, a następnie dodajemy do dokumentu HTML:

<script src="sciezka_do_pliku.js" type="text/javascript"></script>

Taki tag możemy umieścić zarówno w nagłówku dokumentu (sekcja <head>), jak i w jego treści (<body>).

Uwaga: Dodanie tagu <script> powoduje wstrzymanie ładowania strony aż zostanie pobrany plik z kodem JavaScript. Jest to argument przemawiający za umieszczaniem skryptów na samym dole dokumentu tuż przed tagiem zamykającym </body>. Z drugiej strony, raz ściągnięty z serwera plik jest zachowywany w pamięci cache przeglądarki, która korzysta z niego, aż na serwerze nie pojawi się jego nowsza wersja. Bardziej eleganckim miejscem dla zadeklarowania skryptów jest nagłówek dokumentu (<head>), ponieważ nie są one treścią strony. Skrypt zostanie wykonany tak samo – decyzja należy do programisty.

Oprócz tworzenia całego dokumentu możemy uruchomić dowolny kod JavaScript w konsoli wbudowanej w przeglądarkę (w Firefoksie konsolę można znaleźć w menu Narzędzia lub uruchomić za pomocą skrótu Ctrl+Shift+K).




Zmienne

W JavaScript zmienne definiujemy, używając słowa var. Robimy to zawsze na początku funkcji, w której chcemy ich używać (więcej o nich w następnej lekcji). Pominięcie słowa kluczowego var powoduje, że zmienna jest zdefiniowana w przestrzeni globalnej, co powoduje, że jest dostępna z każdego innego pliku JavaScript dołączonego do strony. Jest to bardzo niepożądane zachowanie i może spowodować wiele trudnych do poprawienia błędów, gdy kod napisany w innym pliku będzie modyfikował zmienną używaną w naszym.

W JavaScript mamy następujące podstawowych typy zmiennych:

```
var liczba = 5; // number
var ulamek = 4.3; // number
var tekst = 'Przykładowy tekst'; // string
var prawda = true; // boolean
var pustaZmienna; // undefined
```

- number – zawierają dowolne wartości liczbowe (całkowite lub zmiennoprzecinkowe);
- string – ciąg znaków;



KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

- boolean – jedynie wartości true albo false;
- undefined – jest specjalnym typem zmiennej, która nie posiada żadnej wartości.

Na zmiennych liczbowych możemy dokonywać tradycyjnych operacji matematycznych: + - * /. Natomiast łańcuch znaków możemy łączyć poprzez operację dodawania (+).

Zadanie: Zdefiniuj (w konsoli) kilka zmiennych o różnych typach i sprawdź jak, działają poszczególne operacje. Zwróć szczególną uwagę na operację dodawania pomiędzy różnymi typami zmiennych.

Do przechowywania bardziej rozbudowanych struktur danych możemy wykorzystać tablicę (array) lub obiekt (object).

Tablice

Tablicę tworzymy, korzystając z nawiasów kwadratowych. Wewnątrz nich możemy podać, oddzielone przecinkami, początkowe wartości mające się znaleźć w tablicy:

```
var tablica = [1, 2, 'ostatni'];
```

Do wartości zapisanych w tablicy możemy odwoływać się w tradycyjny sposób, podając numer elementu w nawisach prostokątnych:

```
tablica[0]; // 1
tablica[2]; // 'ostatni'
tablica[10]; // undefined
```

lub używając parametrów i metod:

tablica.length; – zawiera informację o liczbie elementów w tablicy; tablica.shift(); – podaje wartość pierwszego elementu w tablicy i usuwa go; tablica.unshift(a); – dodaje nowy element na początku tablicy; tablica.pop(); – podaje wartość ostatniego elementu w tablicy i usuwa go; tablica.push(a); – dodaje nowy element na końcu tablicy; tablica.sort(); – sortuje elementy w tablicy; tablica.reverse(); – odwraca kolejność elementów w tablicy; tablica.slice(a, b); – zwraca elementy tablicy umieszczone na miejscach pomiędzy podanymi argumentami. Uwaga: Bez podania żadnego argumentu tworzona jest nowa kopia tablicy z takimi samymi elementami. tablica.join('łącz'); – łączy wszystkie elementy w tablicy przy pomocy podanych znaków w jeden długi łańcuch znaków.

Uwaga: Więcej szczegółowych informacji można znaleźć na stronach Mozilla Developer Network:

https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/Array

Obiekty

Podobnie jak tablica, obiekt może przechowywać dowolną ilość elementów, ale zamiast umieszczać je pod wartościami liczbowymi, korzystamy z łańcucha znaków, co pozwala na lepsze opisanie, co przedstawia dana wartość. Obiekt tworzymy, używając nawiasów klamrowych:

```
var gracz = {
        nick: 'Imie gracza',
        level: 5,
        alive: true
};
```

Do wartości przechowywanych w obiekcie możemy odwoływać się tak samo jak do tablicy, tylko podając nazwę parametru jako ciąg znaków, albo podając nazwę parametru po kropce.

```
gracz.typ = 'pojazd';
gracz['nick']; // 'Imie gracza';
gracz.typ; // 'pojazd';
gracz.nieMaElementu; // undefined
```



KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI ORE FWIOO UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Zadanie: Stwórz (w konsoli) obiekt gracza podobny do powyższego, a następnie wszystkie elementy z obiektu skopiuj do nowo utworzonej tablicy.

Funkcje

Funkcje pozwalają wykonywać operacja na danych – przyjmują dane wejściowe (parametry) i zwykle zwracają jakieś dane wyjściowe (wynik, wartość zwracana). Są uniwersalnym pojęciem dla wielu języków programowania.

W JavaScript funkcje deklarujemy, używając słowa kluczowego function. W nawiasach okrągłych po przecinku określamy nazwy parametrów przyjmowanych przez funkcję, a wartość końcową zwracamy, używając słowa kluczowego return:

```
function dodaj(a, b) {
		return a + b;
	};
	dodaj(4, 5);  /* zwraca 9 */
```

Jak widać, komentarze w JavaScripcie zamykamy pomiędzy /* a */.

Funkcję można również zdefiniować nie nazywając jej. Taki konstrukt jest nazywany wyrażeniem funkcyjnym. Przypisanie go do zmiennej ma taki skutek jak zdefiniowanie funkcji o nazwie takiej, jak ta zmienna:

```
var dodaj = function(a, b) {
		return a + b;
	};
	dodaj(4, 5);  /* zwraca 9 */
```

Wyrażenie funkcyjne możemy uruchomić od razu po jego zdefiniowaniu, bez przypisywania do zmiennej:

```
(function(a, b) {
		return a + b;
	})(4,5);  /* zwraca 9 */
```

**Zadanie:** Utwórz plik JavaScript, zdefiniuj w nim kolejne funkcje dokonujące podstawowych operacji matematycznych na podanych parametrach: odejmuj, mnoz, dziel.

Interakcja z użytkownikiem

Większość aplikacji napisanych w JavaScript pobiera dane wprowadzane przez użytkownika poprzez formularze HTML i wyświetla informacje o wynikach działania, modyfikując resztę dokumentu HTML. Jednak na potrzeby tego ćwiczenia skorzystamy z prostych, ale już gotowych narzędzi w przeglądarce:

- alert('Komunikat'); – wyświetla dla użytkownika małe okno z podaną informacją.
- var zgoda = confirm('Komunikat?'); – wyświetla komunikat, na który użytkownik może odpowiedzieć poprzez Ok / Anuluj. Wartość wybranej odpowiedzi możemy zapisać do dowolnej zmiennej.
- var odpowiedz = prompt('Pytanie?'); – wyświetla komunikat z pytaniem, na które użytkownik może odpowiedzieć w dowolny sposób, używając pola tekstowego.
- console.log(obiekt); – jest to specjalna funkcja umożliwiająca wyświetlenie w konsoli wartości dowolnej zmiennej lub obiektu w trakcie działania programu.

Zadanie: Stwórz prosty dokument HTML zawierający jedynie powiązanie do pliku script.js.

Do pliku script.js przepisz poniższy kod, uzupełniając zaznaczone elementy, tworząc grę, w której użytkownik zgaduje wylosowaną liczbę, a aplikacja podpowiada czy podana liczba jest większa lub mniejsza od docelowej.



KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
FWIOO
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt "Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych" współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
// Losujemy liczbę od 1 do 100 zaokrąglając ją w dół.
// Następnie definiujemy wszystkie potrzebne zmienne.
var zgadnij = Math.floor(Math.random()*100+1),
   strzal,
   iloscProb = 0,
   zrezygnowal = false;

do {
 iloscProb = ?; // Zwiększamy wartość zmiennej o 1;
 strzal = prompt('Zgadnij wylosowaną liczbę.');
   // Użytkownik kliknął anuluj zamiast podać liczbę.
 if (strzal === null) {
   alert('Poddałeś się po ' + iloscProb + ' próbach');
   zrezygnowal = true;
   break;
 }

 // Zamieniamy łańcuch znaków na liczbę w systemie dziesiętnym.
 strzal = parseInt(strzal, 10);

 // Używając operacji porównania if() sprawdź
 // wartość zmiennych strzal i zgadnij, a następnie
 // wyświetl odpowiedni komunikat poprzez alert();

} while (strzal !== zgadnij);

if (! zrezygnowal) {
 alert('Zgadłeś po ' + iloscProb + ' próbach');
}
```

Zadanie dodatkowe 1: Sprawdź czy użytkownik wprowadził liczbę. Jeśli nie wyświetl komunikat o błędzie i poproś o ponowną próbę.

Wskazówka: **Funkcja parseInt()** przy błędzie przekształcania na liczbę zwraca specjalną wartość NaN, którą można wykryć używając funkcji **isNaN();.**

Zadanie dodatkowe 2: Zmienna przechowująca wylosowaną wartość nie znajduje się we wnętrzu żadnej funkcji, co oznacza, że można ją podejrzeć z dowolnego innego miejsca w kodzie. Żeby zabezpieczyć ją i nie pozwolić, aby żaden użytkownik nie oszukiwał w trakcie gry, stwórz anonimową funkcję, która automatycznie sama się uruchomi. Skorzystaj z ostatniego przykładu opisanego w punkcie 4.



KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI ORE FWIOO UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.