Warsztaty front-endowe
Zacznij w IT

JavaScript
Część 2

JavaScript - wprowadzenie

JavaScript

I przechodzimy do najważniejszej części, czyli do JavaScriptu :) Najpierw zapoznamy się z podstawami, potem przejdziemy przez proste ćwiczenia, a potem zabierzemy się za zadania z warsztatów.

- JavaScript to język skryptowy, który używany jest na większości stron internetowych
- JavaScript najczęściej służy do interakcji z użytkownikiem i dodania do strony dynamicznych elementów
- Plik ze skryptem dodajemy do dokumentu HTML używając znacznika <script> i podając ścieżkę do pliku w atrybucie **src**

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

Sprawdź się!

W katalogu, w którym tworzyłaś/łeś testowe pliki HTML i CSS dodaj nowy folder o nazwie *js*. W tym folderze utwórz plik JavaScript *app.js* i wykonaj poniższe zadania.

1. Dołącz swój plik ze skryptem do dokumentu HTML używając znacznika <script>.
2. Sprawdź, czy plik dołączony jest odpowiednio – w pliku JS wpisz **console.log("działa!")**.
3. Zapisz plik i otwórz w przeglądarce podgląd swojego pliku index.html. Włącz konsolę deweloperską (na Windowsie – wciśnij F12, na OS X – Cmd + Opt + I). Interesuje Cię zakładka *Console*. To tam powinien pojawić się Twój napis *działa*. Widzisz go?
4. Jeśli coś jest nie tak, w konsoli zobaczysz błędy, które występują na stronie.

MAŁE WYJAŚNIENIE

Dlaczego na razie widzisz tekst tylko przez konsolę deweloperską? Otóż jeszcze w żaden sposób nie odwołaliśmy się do elementów HTML, więc nic nie zobaczymy bezpośrednio na stronie. W dalszej części prezentacji dowiesz się, w jaki sposób wpływać na elementy znajdujące się na stronie :)

Zmienne

Zmienne pozwalają nam przechowywać fragmenty pamięci. To dane, które opisane są odpowiednią etykietką. Zmienne w JS definiujemy przez użycie słowa kluczowego **var** (od angielskiego *variable*).

- Nazwa zmiennej może składać się z dowolnej liczby cyfr, liter oraz niektórych znaków specjalnych
- Nazwy zmiennych nie mogą zawierać spacji
- Nazwy zmiennych zaczynamy od małej litery
- Jeśli zmienna ma więcej niż jedno słowo stosujemy zapis camelCase
- Tekst, czyli zmienne typu string, zawsze zapisujemy w apostrofach

CIEKAWOSTKA
Obecnie częściowo implementowana jest też wersja JS ES6/2015, gdzie zmienne możemy tworzyć używając słów kluczowych **let** i **const**, w zależności, czy chcemy przypisywać wartość do zmiennej raz czy wielokrotnie

Przykładowe zmienne

```
var myName = 'Bella';
var myAge = 18;
var myFavouriteSnack = 'nothing in particular';
```

Tablice

Tablice pozwalają na przechowywanie wielu danych w jednym miejscu. Gdy na przykład chcemy zapisać imiona wszystkich osób na warsztacie, nie musimy tworzyć zmiennej dla każdej osoby. Wystarczy, że stworzymy sobie tablicę, która będzie przechowywała wszystkie imiona.

- Elementy tablicy umieszczamy w nawiasach kwadratowych i oddzielamy przecinkami
- Tablice możemy przypisać do zmiennej
- JS umożliwia nam wykonywanie różnych działań na tablicach
- Żeby dostać się do konkretnego elementu tablicy, musimy podać jej nazwę oraz indeks danego elementu w nawiasie kwadratowym
- Długość tablicy (czyli liczbę elementów) pokazuje jej atrybut length
- **Pierwszy element w tablicy ma zawsze indeks 0**

CIEKAWOSTKA
Tablice mogą w sobie zawierać kolejne tablice.
Wtedy nazywamy je **tablicami wielowymiarowymi**

-

Przykładowe tablice

```
var numbers = [ 1, 2, 3 ] ;
var fruits =  [ 'apple', 'plum', 'orange'  ] ;
var mixedThings = [ 1, 4, 6, 'apple', 'banana' ] ;

console.log(numbers[0]) // wypisze w konsoli 1
console.log(fruits[1]) // wypisze w konsoli plum
console.log(mixedThings[5]) // wypisze w konsoli
undefined
console.log(numbers.length) // wypisze w konsoli 3
```

Funkcje

Funkcje to zbiory czynności, które mają zostać wykonane. Inaczej mówiąc – to instrukcje dla skryptu. Funkcja może coś przeliczać, zwracać, sprawdzać, itd.

- Funkcję tworzymy przez użycie słowa *function*
- Funkcje mogą przyjmować argumenty
- Funkcje zazwyczaj coś zwracają (wskazujemy to słowem *return),* ale mogą też na przykład coś wypisywać w konsoli (przez *console.log*)
- **Żeby funkcje zadziałały, muszą zostać wywołane, samo zapisanie ich w skrypcie ich nie wywoła**
- **Funkcję wywołujemy zapisując jej nazwę razem z nawiasem okrągłym (spójrz na przykład - dwie ostatnie linijki to wywołanie funkcji); jeśli funkcja przyjmuje argumenty, podajemy je w tym nawiasie**
-

Przykładowe funkcje

```
function myFunction() {
  console.log('abc');
}

function mySecondFunction(number) {
  var x = 1;
  return number + x;
}

myFunction(); // wypisze w konsoli abc
mySecondFunction(1) // zwróci 2
```

Instrukcje warunkowe

Instrukcje warunkowe wykonują podany kod, gdy zostaje spełniony określony warunek.

- Instrukcje warunkowe wprowadzamy do kodu używając słowa *if*
- Po **if** w nawiasie okągłym wpisujemy warunek, który chcemy sprawdzić
- Następnie w nawiasie klamrowym podajemy kod, który ma się wykonać, gdy warunek zostanie spełniony
- Do zapisu warunków używamy różnych operatorów porównania np. **<**, **>**, **<=**, **>=**, **==**
- Aby sprawdzić, czy elementy są równe używamy operatora **==**
- **1 == 1 oznacza 1 jest równe 1**
- Aby sprawdzić, czy elementy nie są równe (czyli są różne) używamy **!=**
- **1 != 2 oznacza 1 nie równa się 2 albo 1 jest różne od 2**

Przykładowe instrukcje warunkowe

```
var a = 1;
var b = 100;

if (b > a) {
  // ten kod się wykona, bo jest spełniony warunek
}

if (a > b) {
  // ten kod się teraz nie wykona, bo a nie jest większe od b
} else {
  // ten kod się teraz wykona
}
```

Pętle

Pętle pozwalają nam wykonać kod pewną ilość razy.

- ➢ Pętlę zaczynamy od zdefiniowana jej momentu początkowego, w pętli **for** podajemy początek pętli jako pierwszy; w naszym przykładzie pętlę rozpoczynamy od 0 przypisując je do zmiennej i
- ➢ Następnie podajemy, gdzie pętla ma zakończyć działanie; w naszym przykładzie pętla kończy się, gdy i będzie równe albo mniejsze od długości tablicy minus 1, czyli gdy będzie równe lub mniejsze od 2
- ➢ Na sam koniec wskazujemy jak zmienia się zmienna i przy każdej iteracji; w przykładzie i zwiększa sie o jeden z każdą iteracją, służy do tego zapis **i++**, który jets skrótem od **i + 1**
- ➢ W nawiasia klamrowym zapisujemy kod; kod ten wykona się tyle razy, ile mamy iteracji; w przykładzie kod wykona się 3 razy, za każdym razem wyświetlając w konsoli element tablicy fruits o indeksie równym zmiennej i w trakcie danej iteracji

My podczas warsztatu najczęściej będziemy wykonywać pętlę, by przejść przez wszystkie elementy danej tablicy

Przykładowa pętla for

```
var fruits = [ 'apple', 'banana', 'orange' ];

for (var i = 0; i <= fruits.length-1; i++) {
  console.log(fruits[i]);
}

// ta pętla wypisze w konsoli elementy tablicy fruits
```

Sprawdź się!

Spróbuj wykonać poniższe zadania w swoim pliku JS.

1. Stwórz zmienną ze swoim imieniem, a następnie wypisz ją w konsoli. Pamiętaj, że zmienne typu string zawsze zapisujemy w apostrofach.
2. Stwórz tablicę z trzema imionami: swoim i dwóch osób siedzących najbliżej Ciebie.
3. Wypisz w konsoli imię uczestniczkia, który jest zapisana w Twojej tablicy jako drugi. Skorzystaj z wcześniej zadeklarowanej tablicy (nie wpisuj imienia ręcznie, tylko wybierz element z tablicy). Następnie dopisz do tablicy z imionami (na samym początku) jeszcze jedno imię. Jak wpłynęło to na wyświetlone wcześniej imię? Dlaczego?
4. Stwórz dwie zmienne i przypisz do nich dowolne liczby.
5. Napisz warunek – w konsoli ma zostać wypisane słowo “hurra!”, jeśli Twoja pierwsza zmienna jest większa od drugiej.
6. Na podstawie podanego wcześniej przykładu, wypisz w konsoli imiona wszystkich uczestników z Twojej tablicy wykorzystując do tego pętlę for.

WAŻNE!
Nie martw się, jeśli nie wszystko do końca rozumiesz. To naprawdę dużo materiału! Zaraz przejdziemy do samego warsztatu, gdzie znajdziesz instrukcje krok po kroku, jak wykonać poszczególne zadania. Najlepiej uczymy się przez praktykę, więc nie zrażaj się i próbuj dalej :)

Jak się dostać do elementu strony?

Jak się dostać do elementu strony?

Wypisywanie różnych rzeczy w konsoli jest fajne, ale nie do końca przydatne na stronie internetowej. Dlatego teraz nauczymy się, w jaki sposób odwoływać się do faktycznych elementów w kodzie, żeby móc wchodzić z nimi w interakcje.

Do wyszukiwania elementu na stronie będą nam służyć następujące metody:

document.querySelector('selector') - ta metoda pobierze element na podstawie jego selektora CSS;
document.querySelectorAll('selector') - ta metoda pobiera wszystkie elementy o danym selektorze i zwróci je jako tablicę
document.getElementById('id') - ta metoda pobiera element na podstawie jego id
document.getElementsByTagName('tag') - pobiera wszystkie elementy o danym tagu i zwraca je jako tablicę
document.getElementsByClassName('class') - pobiera wszystkie elementy o danej klasie i zwraca je jako tablicę

Zaraz dowiesz się, jak to wszystko wygląda w praktyce :)

Co dalej?

Teraz wykonaj zadania z repozytorium :)

W repozytorium znajdziesz listę zadań do wykonania. Będziesz dodawać do warsztatowej strony różne funkcjonalności.

Pamiętaj, że dobrze jest robić commit po każdym wykonanym zadaniu.

Rozwiązanie każdego zadania możesz zobaczyć wchodząc na odpowiedni branch. Wystarczy, że wpiszesz w terminalu **git checkout nazwa-brancha**

Dokładne nazwy i instrukcje znajdziesz w każdym zadaniu. Rozwiązania podane są w osobnych plikach JS, ale Ty stwórz jeden plik JS, w którym zapisuj cały kod. Tak żeby na końcu mieć działającą stronę :)

Na swój branch powrócisz wpisując **git checkout master**

UWAGA! Przed wejściem na branch z rozwiązaniem skomituj swoje zmiany, bo inaczej je utracisz.

Jeśli uda Ci się zrobić wszystko szybciej, w repozytorium znajdziesz folder Zadania dodatkowe. Są w nim propozycje dodatkowych funkcjonalności, które możesz wprowadzić na stronie.

POWODZENIA!