

✓ Wykonano

Przeglądarki WWW są wyposażone w debugger JavaScript.

Debugging JavaScript in Chrome DevTools | STOP using console log



Run and Debug JavaScript with Firefox developer tools



1. Skrypty wewnętrzne oraz zewnętrzne

1.1. Wypisywanie danych

1. Utwórz dokument HTML o nazwie 'zadanie1A.html', zawierający następujący kod:

```
1. <!DOCTYPE html>
2. <html lang="pl">
3.
4. <head>
5.   <title>Plik 'zadanie1A.html'</title>
6. </head>
7.
8. <body>
9.   <div>Treść dokumentu HTML przed skryptem</div>
10.  <script>
11.    console.log('Tekst 1');
12.    window.alert('Tekst 2');
13.    document.write('Tekst 3');
14.  </script>
15.  <div>Treść dokumentu HTML po skrypcie</div>
16. </body>
17.
18. </html>
```

W tej, podstawowej, wersji dokumentu HTML skrypt jest wykonywany w trakcie renderowania dokumentu HTML.

2. Uruchom przeglądarkę WWW, a następnie [otwórz jej konsolę](#).
3. Załaduj powyższy dokument w bieżącej zakładce przeglądarki WWW.

4. Spróbuj zlokalizować miejsce pojawiania się tekstów: Tekst 1, Tekst 2 oraz Tekst 3. Zaobserwuj, czy okno *alert* jest modalne, czy niemodalne.
5. Utwórz plik 'zadanie1B.html' zawierający poniższy kod, a następnie załaduj go do przeglądarki WWW.

W tej, alternatywnej, wersji dokumentu HTML skrypt jest wykonywany po zakończeniu renderowania dokumentu HTML (po wygenerowaniu zdarzenia '**load**'), a nie w trakcie (renderowania).

```
1. <!DOCTYPE html>
2. <html lang="pl">
3.
4. <head>
5.   <title>Plik 'zadanie1B.html'</title>
6. </head>
7.
8. <body onLoad="funkcja_zwrotna()">
9.   <div>Treść dokumentu HTML przed skrytem</div>
10.  <script>
11.    function funkcja_zwrotna() {
12.      console.log('Tekst 1');
13.      window.alert('Tekst 2');
14.      document.write('Tekst 3');
15.    }
16.  </script>
17.  <div>Treść dokumentu HTML po skrypcie</div>
18. </body>
19.
20. </html>
```

6. Jak myślisz, dlaczego w tej wersji, treść dokumentu HTML (napisy "Treść dokumentu HTML przed skrytem" oraz "Treść dokumentu HTML po skrypcie") nie jest widoczna w przeglądarce — zakomentuj linię, która jest tego przyczyną.

1.2. Typy danych, metoda `window.prompt()`

1. Przeczytaj:
 - [Typy danych](#).
 - Operator [typeof](#).

Typy w JavaScript - [Szybki Kurs JavaScript] #02



2. Utwórz plik 'zadanie1C.html' o poniższej zawartości:

```
1. <!DOCTYPE html>
2. <html lang="pl">
3.   <head>
4.     <title>Plik 'zadanie1C.html'</title>
5.   </head>
6.   <body>
7.     <script>
8.       window.prompt("Tekst1", "Tekst2");
9.     </script>
10.  </body>
11. </html>
```

3. Zbadaj, jakie znaczenie mają poszczególne argumenty metody `window.prompt()` i czy są one obowiązkowe.

4. Napisz funkcję, która czterokrotnie wykonuje następujący kod:

- Za pomocą `window.prompt()` wczytuje wartość.
- Za pomocą `console.log()` wypisuje informację postaci: `wczytanaWartość:typWczytanejWartości`.

Pętla for() - wstawiać let, const, var a może nic?



5. Sprawdź, co jest wypisywane dla następujących czterech przypadków:

1. Użytkownik wprowadził wartość będącą liczbą i nacisnął klawisz 'Enter' lub przycisk 'OK'.
2. Użytkownik wprowadził wartość będącą napisem i nacisnął klawisz 'Enter' lub przycisk 'OK'.
3. Użytkownik nie wprowadził wartości, a następnie nacisnął powyższy klawisz / przycisk.
4. Użytkownik wprowadził wartość, a następnie nacisnął przycisk 'Anuluj'.

1.3. Elementy "input" oraz "output", DOM 0

1. Sprawdź, do czego służą elementy "input" oraz "output" — dopisz w obrębie elementu "body" poniższą zawartość, otwórz dokument w przeglądarce WWW, a następnie wprowadź dane.

```
1. <form onInput="wynik.value = pole_tekstowe.value + pole_liczbowe.value">
2.   <input id="pole_tekstowe" type="text" placeholder="Wprowadź tekst">
3.   +
4.   <input id="pole_liczbowe" type="number" placeholder="Wprowadź liczbę">
5.   =
6.   <output name="wynik" for="pole_tekstowe pole_liczbowe">Tu pojawi się wynik
   obliczeń</output>
7.   <br>
8.   <input type="button" value="Wypisz">
9. </form>
```

Na ten moment, nie zwracaj uwagi na poprawność wyniku — po wykonaniu zadania 2 będziesz w stanie poprawić powyższy fragment kodu tak, aby wynik obliczeń był prawidłowy.

2. Zmodyfikuj plik 'zadanie1C.html':

1. Dodaj obsługę kliknięcia przycisku "Wypisz" korzystając z [modelu obsługi zdarzeń DOM 0 \(rejestracja inline\)](#) — dopisz `onClick='funkcja_zwrotna()'`.
2. W części nagłówkowej dokumentu HTML utwórz funkcję `funkcja_zwrotna()`, która:
 1. Za pomocą kolekcji [DOM 0](#) — `document.forms[0].elements[]` — odczytuje wartość z obydwu pól formularza (elementy o id "pole_tekstowe" oraz "pole_liczbowe").

2. Wypisuje (`console.log()`) informację postaci:

```
wczytanaWartośćZPolaTekstowego:typWczytanejWartości
wczytanaWartośćZPolaNumerycznego:typWczytanejWartości
```

3. Zbadaj, co wypisuje `funkcja_zwrotna()` w przypadku:

1. Wprowadzenia wartości będącej liczbą i naciśnięcia powyższego przycisku.
2. Wprowadzenia wartości będącej napisem i naciśnięcia ww. przycisku.
3. Niewprowadzenia wartości i naciśnięcia przycisku "Wypisz".

4. Przenieś **definicje** wszystkich utworzonych funkcji do osobnego pliku — plik 'zadanie1.js' — utwórz zewnętrzny, zwykły, skrypt JS; **wywołania** funkcji mają pozostać tam, gdzie są, tzn. w pliku 'zadanie1C.html'.

5. Załaduj ten skrypt (plik) z poziomu dokumentu HTML 'zadanie1C.html', a następnie sprawdź, czy wszystko działa tak jak wcześniej.

Kwestie bezpieczeństwa

Jeżeli będziesz kiedyś tworzył / tworzyła aplikację WWW, to pamiętaj: „Nie należy ufać użytkownikowi w kwestii poprawności wprowadzanych danych” — dane pochodzące z formularza lub okna *prompt* należy traktować jako „brudne” i w związku z tym nie należy ich umieszczać, od razu, na stronie WWW, w bazie danych itd. — ktoś może wpisać, przykładowo, zamiast imienia i nazwiska, czy adresu e-mail, kod HTML zawierający skrypt JS: `<script>skrypt JS</script>`.

W celu oczyszczenia danych można np. dokonać ich walidacji (za pomocą wyrażeń regularnych), albo zamienić znaki specjalne '`<`' oraz '`>`' na encje.



2. Skrypty typu *module* (moduły) oraz testy

W tym zadaniu do testowania skryptów JS użyjemy dwóch bibliotek: **Mocha** oraz **Chai**. Przy okazji zaznajomimy się, praktycznie, z typami prymitywnymi — głównie z typem napisowym oraz z innymi (niż 'for ; ;') rodzajami pętli.

2.1. Moduł z testami jednostkowymi

1. Utwórz plik 'zadanie2.js' o następującej zawartości:

```
1. function sum(x,y) {
2.     return x+y;
3. }
```

2. Utwórz plik 'zadanie2.test.js' zawierający:

```
1. function test_sum() {
2.     describe('The sum() function', function () {
3.         it('should return 4 for 2+2', function () {
4.             sum(2, 2).should.equal(4);
5.         });
6.         it('should return 0 for -2+2', function () {
7.             sum(-2, 2).should.equal(0);
8.         });
9.     });
10. }
```

3. Utwórz dokument HTML o nazwie 'index.html' i poniższej zawartości:

```

1. <!DOCTYPE html>
2. <html lang="en">
3.
4. <head>
5.   <meta charset="utf-8">
6.   <title>
7.     Mocha tests
8.   </title>
9.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
10.  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/mocha/mocha.css">
11. </head>
12.
13. <body>
14.   <div id="mocha"></div>
15.   <script src="zadanie2.js"></script>
16.   <script src="https://cdn.jsdelivr.net/npm/mocha/mocha.js"></script>
17.   <script type="module">
18.     import { should } from 'https://cdn.jsdelivr.net/npm/chai/chai.js';
19.
20.     should();
21.     mocha.setup('bdd');
22.     mocha.checkLeaks();
23.     // Początek bloku wywołań funkcji testujących
24.
25.     // Koniec bloku wywołań funkcji testujących
26.     mocha.run();
27.   </script>
28. </body>
29.
30. </html>

```

Kod widoczny w liniach 17-27 to skrypt typu *module*.

4. Korzystając z [przykładu pokazanego na wykładzie](#):

1. Wyeksportuj funkcję `test_sum()` — lokalizacja — plik 'zadanie2.test.js'.
2. Zaimportuj tę funkcję — lokalizacja — plik 'index.html'.
3. Wywołaj ww. funkcję — lokalizacja — blok wywołań funkcji testujących (plik 'index.html').
4. Uruchom lokalny serwer WWW na porcie 8000.
5. Otwórz, w przeglądarce WWW, stronę <http://localhost:8000/> i sprawdź, czy testy kończą się powodzeniem.
6. Przeczytaj opis [trybu ścisłego](#), a następnie zbadaj, czy skrypty typu *module*, domyślnie, mają włączony ten tryb.

2.2. Typy prymitywne oraz pętle

1. Zaznajom się z:

- Typami prymitywnymi: [numerycznym](#), [napisowym](#) oraz [tablicowym](#).
- Pętlami: [for...in](#) oraz [for...of](#).

for...in VS for...of - różnice pomiędzy pętlami JS



2. Umieść, w 'zadanie2.test.js', definicję funkcji `test_string_operations()`:


```

1.  export function test_string_operations() {
2.      describe("String operations", function () {
3.          context("When the array contains strings", function () {
4.              it("the sum_strings() function should return the sum of those strings that
are numbers or begin with a sequence of digits", function () {
5.                  sum_strings(["123", "146a2B", "", "b3345a", "\t"]).should.equal(269);
6.              });
7.          });
8.
9.          context("When the array is empty", function () {
10.             it("the sum_strings() function should return 0", function () {
11.                 sum_strings([]).should.equal(0);
12.             });
13.          });
14.
15.          context("When the string contains only digits", function () {
16.              it("the 'digits()' function should return an array with the sum of odd and
even digits", function () {
17.                  digits("123").should.deep.equal([4, 2]);
18.              });
19.              it("the 'letters()' function should return [0, 0]", function () {
20.                  letters("123").should.deep.equal([0, 0]);
21.              });
22.          });
23.
24.          context("When the string contains only letters", function () {
25.              it("the 'digits()' function should return [0, 0]", function () {
26.                  digits("aBc").should.deep.equal([0, 0]);
27.              });
28.              it("the 'letters()' function should return an array with the number of
lowercase and uppercase letters", function () {
29.                  letters("aBc").should.deep.equal([2, 1]);
30.              });
31.          });
32.
33.          context("When the string contains letters followed by digits", function () {
34.              it("the 'digits()' function should return an array with the sum of the odd
and even digits", function () {
35.                  digits("aB123").should.deep.equal([4, 2]);
36.              });
37.              it("the 'letters()' function should return an array with the number of
lowercase and uppercase letters", function () {
38.                  letters("aB123").should.deep.equal([1, 1]);
39.              });
40.          });
41.
42.          context("When the string contains digits followed by letters", function () {
43.              it("the 'digits()' function should return an array with the sum of the odd
and even digits", function () {
44.                  digits("123aB").should.deep.equal([4, 2]);
45.              });
46.              it("the 'letters()' function should return an array with the number of
lowercase and uppercase letters", function () {

```

```

47.         letters("123aB").should.deep.equal([1, 1]);
48.     });
49. });
50.
51.     describe("When the string is empty", function () {
52.         it("the 'digits()' function should return [0, 0]", function () {
53.             digits("").should.deep.equal([0, 0]);
54.         });
55.         it("the 'letters()' function should return [0, 0]", function () {
56.             letters("").should.deep.equal([0, 0]);
57.         });
58.     });
59. });
60. }

```

3. Zaimportuj funkcję `test_string_operations()`, a następnie wywołaj ją — lokalizacja — plik 'index.html'.
4. Ponownie otwórz stronę <http://localhost:8000/> i spowoduj, aby testy kończyły się powodzeniem — zdefiniuj, w pliku 'zadanie2.js', następujące funkcje:

sum_strings(a)

Oblicza **sumę wartości "liczb"** zawartych w tablicy napisów *a*, gdzie "liczba" to napis, który wygląda jak liczba, tzn. rozpoczyna się od ciągu cyfr lub zawiera same cyfry.

digits(s)

Dla napisu *s*, oblicza **sumę** zawartych w nim **cyfr nieparzystych** oraz **parzystych**, a następnie zwraca wynik w postaci tablicy [suma_liczb_nieparzystych, suma_liczb_parzystych].

letters(s)

Oblicza **ilość** zawartych w napisie *s* **liter małych** oraz **dużych**, a następnie zwraca tablicę z wynikami — [ilość_małych_liter, ilość_dużych_liter].

Przykład działania

Dla ["123", "146a2B", "", "b3345a", "\t"] funkcja `sum_strings()` powinna zwrócić $269 \leftarrow 123 + 146 + 0 + 0 + 0$.

Pozostałe funkcje

| Dane wejściowe | Wynik działania funkcji <code>digits()</code> | Wynik działania funkcji <code>letters()</code> |
|----------------|---|--|
| 123 | [4, 2] | [0, 0] |
| 146a2B | [1, 12] | [1, 1] |
| b3345a | [11, 4] | [2, 0] |

Dla ambitnych

Spróbuj, jeśli potrafisz, zaimplementować ww. funkcje używając **podejścia funkcyjnego**. Ponieważ wbudowane funkcje wyższego rzędu są metodami obiektu prototypowego **Array**, będziesz musiał / musiała **skonwertować napis na tablicę**.

5. W oparciu o zdobytą wiedzę, zmodyfikuj wyrażenie `pole_tekstowe.value + pole_liczbowe.value` — patrz sekcja 1.3 — wynikiem ewaluacji wyrażenia ma być suma liczb, a nie konkatencja napisów.

3. Tworzenie dynamicznych grafik

Na naszych zajęciach będziemy poznawać najważniejsze obszary zastosowań języka JavaScript (JS). W tym zadaniu użyjemy JS do stworzenia dynamicznej grafiki. Do roku 2017 przeglądarki WWW miały możliwość obsługi appletów Java, czyli małych, graficznych aplikacji napisanych w Javie. Za pomocą pary znaczników '<applet>', '</applet>' można było zdefiniować, na powierzchni strony WWW, prostokątny obszar, po którym można było rysować.

Jedną z nowości HTML 5 jest płótno — przy użyciu pary znaczników '<canvas>', '</canvas>' można zdefiniować prostokątny obszar, po którym można rysować za pomocą instrukcji języka JS.

Zabawy z Canvas API  hello roman #99



1. Utwórz dokument HTML o nazwie 'zadanie3.html' i poniższej zawartości:

```





1. <!DOCTYPE html>
2. <html lang="pl">
3.   <head>
4.     <meta charset="UTF-8">
5.     <title>Płótno</title>
6.     <script>
7.       "use strict"; // Nie wyłączaj trybu
       ścisłego
8.       var canvas = document.getElementById('canvas'); // Tutaj jest użyty standard
       W3C DOM – będzie on tematem następnych ćwiczeń
9.       ctx = canvas.getContext('2d'); // Utworzenie obiektu
       'CanvasRenderingContext2D'
10.      ctx.fillText("Hello World", 10, canvas.height / 2); // Wykreślenie podanego
       tekstu na płótnie
11.    </script>
12.  </head>
13.  <body>
14.    <main>
15.      <h1>Płótno</h1>
16.      <canvas id="canvas" style="border:1px solid #000000;">
17.        Wygląda na to, że twoja przeglądarka nie obsługuje elementu "canvas"
18.      </canvas>
19.    </main>
20.  </body>
21. </html>

```

2. Dlaczego na powierzchni płótna nie pojawił się napis "Hello World" — zobacz jaki komunikat wyświetla się w konsoli przeglądarki WWW (`Ctrl+Shift+I`); spróbuj wprowadzić takie modyfikacje, aby powyższe instrukcje zadziałały.

Informacja

Skrypt zawiera dwa błędy — spróbuj je odnaleźć i poprawić — pomocny może być [przykład z wykładu](#).

- Przeczytaj kurs poświęcony podstawom [Canvas API](#).
- Opracuj, a następnie narysuj (w elemencie 'canvas') logo dla aplikacji, którą zajmujemy się na ćwiczeniach — logo powinno się składać, z co najmniej, trzech, **różnych**, figur geometrycznych.
- Zastąp, w pasku nawigacyjnym strony WWW z poprzednich ćwiczeń, ikonę marki ( /  /  / ) stworzoną grafiką; być może 🤖 będziesz musiał / musiała [przeskalować rysunek](#).



4. Zadanie

Rozbuduj dokument HTML z poprzednich ćwiczeń o skrypt JavaScript oraz dodatkowe elementy HTML (formularz, obraz itp.) — szczegóły zostaną podane na **początku ćwiczeń**. Założenia dla skryptu:

- Wczytuje dane zawarte w (jedno z poniższych):
 - [Polach wprowadzania danych](#) formularza HTML, korzystając z DOM 0 — obiektów / kolekcji 'document.forms' oraz 'elements'.
 - Oknie *prompt*
- Przechowuje dane w:
 - Kolekcjach: [indeksowanej](#) lub [kluczowanej](#).
 - [Pamięci przeglądarki \(Web Storage\)](#) lub [IndexedDB](#).
- Realizuje funkcjonalność podaną na **początku ćwiczeń**.
- Wypisuje:
 - Wyniki — w konsoli (metoda `console.log()`) lub w elemencie 'canvas'.
 - Ostrzeżenia i błędy — w konsoli (metody `console.warn()` oraz `console.error()`) lub w oknie *alert*.