

Funkcje strzałkowe, funkcje wyższego rzędu, fetch, promisy



Cześć!


Łukasz Krawczyk

Web Developer / Graphic & Web Designer

Funkcje strzałkowe (ang. *"fat arrow" functions*)

Funkcje strzałkowe

(ang. *"fat arrow" functions*)

- Zostały wprowadzone w ES6
- Mają krótszą składnię niż zwykła funkcja
- Nie posiadają swojego „tablicopodobnego” (ang. *array-like*) obiektu arguments
- Są anonimowe
- Najlepiej sprawdzają się w przypadku funkcji, które nie są metodami
- Nie działają w przeglądarce Internet Explorer 
- Zgodnie z badaniami, to najbardziej ulubiona ze wszystkich cech ES6

Can I use... Support tables for HTML5, CSS3, etc

caniuse
.com

[Home](#) [News](#) [July 18, 2018 - New feature: Shared Array Buffer](#) [Compare browsers](#) [About](#)


Can I use

arrow functions

? [Settings](#)

2 results found

#

Arrow functions  - OTHER

Usage % of all users

Global 87.4%

Poland 91.2%

Function shorthand using => syntax and lexical this binding.

Current aligned

Usage relative

Date relative

Show all

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			66		10.3				4
		60	67		11.2				6.2
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

Notes

Known issues (0)

Resources (6)

Feedback

No notes

See support for [arrow functions](#) (ECMAScript 6)

Funkcje strzałkowe

(ang. *"fat arrow" functions*)

```
const powitanie = () => { return 'Hello, world!'; };  
// instrukcje - ciało blokowe (ang. block body)  
powitanie(); // zwraca 'Hello, world!'
```

albo

```
const powitanie = () => 'Hello, world!';  
// tylko 1 wyrażenie - ciało „zwięzłe” (ang. concise body)  
powitanie(); // zwraca 'Hello, world!'
```

Funkcje strzałkowe

(ang. *"fat arrow" functions*)

```
const powitanie = (a) => 'Hello, ' + a;  
// tylko 1 wyrażenie - ciało „zwięzłe” (ang. concise body)  
powitanie('world!'); // zwraca 'Hello, world!'
```

albo

```
const powitanie = a => 'Hello, ' + a;  
// tylko 1 wyrażenie - ciało „zwięzłe” (ang. concise body)  
powitanie('world!'); // zwraca 'Hello, world!'
```

Funkcje strzałkowe

(ang. *"fat arrow" functions*)

```
const dodawanie = (a, b, c) => { return a + b + c; };  
// instrukcje - ciało blokowe (ang. block body)  
dodawanie(1, 2, 2); // zwraca 5
```

albo

```
const dodawanie = (a, b, c) => a + b + c;  
// tylko 1 wyrażenie - ciało „zwięzłe” (ang. concise body)  
dodawanie(1, 2, 2); // zwraca 5
```


Funkcje strzałkowe (ang. *"fat arrow" functions*)

Parametry domyślne (ang. *default parameters*)

```
const dodawanie = (a, b = 0, c = 2) => a + b + c;  
// tylko 1 wyrażenie – ciało „zwięzłe” (ang. concise body)  
dodawanie(1, 2); // zwraca 5
```

```
const dodawanie = (a, b = 0) => a + b + c;  
// tylko 1 wyrażenie – ciało „zwięzłe” (ang. concise body)  
dodawanie(1, 2); // zwraca undefined
```

Zadanie 1.

Funkcje strzałkowe

Zadanie 1.

Funkcje strzałkowe

Napisz bezparametrową **funkcję strzałkową**, która wyprowadzi na ekran – za pomocą metody `console.log()` – napis `Hello, world!`.

Zadanie 2.

Funkcje strzałkowe

Zadanie 2.

Funkcje strzałkowe

- Napisz **funkcję strzałkową**, która z przekazanego do niej napisu stworzy tablicę składającą się z jego poszczególnych wyrazów.
- Przykładowo, przekazując do funkcji strzałkowej napis `Lorem ipsum dolor sit amet`, otrzymamy tablicę składającą się z 5 elementów, czyli `Array(5) ["Lorem", "ipsum", "dolor", "sit", "amet"]`.
- Wynik funkcji wyprowadź na ekran za pomocą metody `console.log()`.
- By podzielić napis na poszczególne elementy tablicy, skorzystaj z — wywoływanej na rzecz napisu — metody **`.split()`**, której parametr określa znak, w którym nastąpić ma podział, np. **`.split(' ')`**.

Zadanie 3.

Funkcje strzałkowe

Zadanie 3.

Funkcje strzałkowe

- Napisz **funkcję strzałkową**, która ustawi wartości 2 właściwości obiektu — o kluczach `id` oraz `name` — na podstawie przekazanych do tej funkcji strzałkowej 2 parametrów.
- Przykładowo, przekazując do funkcji strzałkowej 2 parametry — 1 oraz `'Jan Kowalski'` — otrzymamy obiekt składający się z 2 właściwości, czyli `Object { id: 1, name: "Jan Kowalski" }`.
- Stosując **ciało „zwięzłe”** (ang. *concise body*) funkcji strzałkowej, pamiętaj, by **opakować znajdujący się tam obiekt w nawias**, by odróżnić obiekt od ciała blokowego (ang. *block body*) funkcji strzałkowej.

Zadanie 4.

Funkcje strzałkowe

Zadanie 4.

Funkcje strzałkowe

- Napisz **funkcję strzałkową** przyjmującą 3 parametry, która zwraca ich sumę wymnożoną przez 9.
- Przykładowo, przekazując do funkcji strzałkowej 3 parametry — 2, 4, 5 — otrzymamy liczbę 99.
- Wynik funkcji wyprowadź na ekran za pomocą metody `console.log()`.

Programowanie funkcjonalne (funkcje wyższego rzędu)

Funkcje wyższego rzędu (ang. *higher- -order functions*)

`.forEach()`
`.map()`
`.filter()`
`.reduce()`
`.find()`



Programowanie funkcjonalne

Metoda `.forEach()`

Metoda `.forEach()` (ES3, ~~IE~~←9) wywołuje przekazaną do niej funkcję dla każdego elementu tablicy z osobna. Pierwszy parametr **funkcji wyższego rzędu** to aktualnie przetwarzany element tablicy, opcjonalny drugi – indeks tego elementu, opcjonalny trzeci – tablica, do której ten element należy. Np.

```
var kontynenty = ['Europa', 'Afryka', 'Azja'];  
kontynenty.forEach(function(element, index, array) {  
    console.log(index + 1 + ' . ' + element);  
});
```

```
// pierwsze wywołanie funkcji zwraca '1. Europa'  
// drugie wywołanie funkcji zwraca '2. Afryka'  
// trzecie wywołanie funkcji zwraca '3. Azja'
```

Programowanie funkcjonalne

Metoda `.map()`

Metoda `.map()` (ES5, ~~IE~~ ← 9) tworzy nową tablicę, która zawiera nowe elementy stworzone w oparciu o elementy z oryginalnej tablicy. Oryginalna tablica pozostaje niezmienną. Pierwszy parametr **funkcji wyższego rzędu** to aktualnie przetwarzany element tablicy, opcjonalny drugi – indeks tego elementu, opcjonalny trzeci – tablica, do której ten element należy. Np.

```
var kont = ['Europa', 'Afryka', 'Azja'];  
var kontNowe = kont.map(function(element, index, array) {  
    return 'Kontynent ' + element;  
});
```

```
console.log(kontNowe); // zwraca Array(3) [ "Kontynent  
Europa", "Kontynent Afryka", "Kontynent Azja" ]
```

Zadanie 5.

Metoda `.map()`

Zadanie 5.

Metoda `.map()`

- Dana jest tablica z liczbami [5, 6, 13, 0, 1, 18, 23].
- Na podstawie tej tablicy, **stwórz nową tablicę**, której elementami będą wymnożone przez wybrany mnożnik poszczególne elementy oryginalnej tablicy.
- Mnożnik przekaz jako **4. parametr metody `.map()`**, np. `mnozник = 2`.
- Przykładowo, przekazując do metody `.map()` mnożnik o wartości 2, otrzymamy następującą tablicę:
`Array(7) [10, 12, 26, 0, 2, 36, 46]`.
- **Wynik metody `.map()` wyprowadź na ekran** za pomocą metody `console.log()`.

Zadanie 6.

Metoda .map()

Zadanie 6.

Metoda `.map()`

- Dana jest **tablica obiektów ze smartfonami**.
- Każdy element tablicy jest obiektem z 2 składowymi — `name` i `price`.
- Na podstawie tej tablicy, **stwórz nową tablicę**, której elementami będą połączone informacje (w postaci ciągu znaków) z obu składowych poszczególnych obiektów.
- Przykładowo, dla pierwszego smartfona, mamy otrzymać pierwszy element nowej tablicy w postaci: `1. Apple iPhone X (4699 zł)`.
- **Wynik metody `.map()` wyprowadź na ekran** za pomocą metody `console.log()`.

Zadanie 7.

Metoda `.map()`

Zadanie 7.

Metoda `.map()`

- Dana jest **tablica z nazwiskami** `['Kochanowski', 'Bednarek', 'Twardowski']`.
- **Zmodyfikuj tę tablicę** w taki sposób, by do poszczególnych jej elementów zostało dodane imię `'Jan '`.
- Przykładowo, pierwszy element tej tablicy ma mieć postać: `'Jan Kochanowski'`.
- **Wynik metody `.map()`** wyprowadź na ekran za pomocą metody `console.log()`.

Programowanie funkcjonalne

Metoda `.filter()`

Metoda `.filter()` (ES3, ~~IE<9~~) zwraca te elementy tablicy, które spełniają kryterium zadane przez przekazaną do tej metody funkcję. Oryginalna tablica pozostaje niezmienną. Pierwszy parametr **funkcji wyższego rzędu** to aktualnie przetwarzany element tablicy, opcjonalny drugi – indeks tego elementu, opcjonalny trzeci – tablica, do której ten element należy. Np.

```
var nr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  
var nrFiltr = nr.filter(function(element, index, array) {  
    return element < 5 || element >= 9;  
});
```

```
console.log(nrFiltr); // zwraca Array(6) [ 1, 2, 3, 4, 9, 10 ]
```

Zadanie 8.

Metoda `.filter()`

Zadanie 8.

Metoda `.filter()`

- Dana jest **tablica z wiekiem (latami)** `[37, 26, 18, 16, 3]`.
- Na podstawie tej tablicy, **stwórz nową tablicę**, której elementami będą tylko te elementy oryginalnej tablicy, które spełniają **kryterium dorosłości**.
- Następnie, korzystając z metody `.sort()`, **nową tablicę posortuj** od wartości najmniejszej do największej.
- Nowa tablica ma wyglądać następująco: `Array(3) [18, 26, 37]`.
- **Wynik metody `.filter()` wyprowadź na ekran** za pomocą metody `console.log()`.

Zadanie 9.

Metoda `.filter()`

Zadanie 9.

Metoda `.filter()`

- Dana jest **tablica** z liczbami od 0 do 15.
- Na podstawie tej tablicy, **stwórz nową tablicę**, której elementami będą tylko te elementy oryginalnej tablicy, które są **podzielne przez 3**.
- Następnie, korzystając z metody `.reverse()`, **nową tablicę posortuj** od wartości największej do najmniejszej.
- Nowa tablica ma wyglądać następująco:
`Array(6) [15, 12, 9, 6, 3, 0]`.
- **Wynik metody `.filter()` wyprowadź na ekran** za pomocą metody `console.log()`.

Zadanie 10.

Metoda `.filter()`

Zadanie 10.

Metoda `.filter()`

- Dana jest **tablica z wiekiem (latami)** [37, 26, 18, 16, 3].
- W dokumencie znajduje się pole `input` (w które użytkownik ma wpisać wartość minimalną) oraz przycisk, po kliknięciu którego, poniżej w akapicie mają zostać **wyświetlone te elementy z wiekiem (latami), które są większe** od podanej przez użytkownika wartości minimalnej.
- Za pomocą metody `.filter()`, **stwórz nową tablicę**, której elementami będą **tylko te elementy oryginalnej tablicy, które są większe** od wartości minimalnej podanej przez użytkownika.
- Przykładowo, dla wartości minimalnej (wpisanej w pole `input`) równej 21, nowa tablica ma wyglądać następująco: `Array(2) [37, 26]`.

Programowanie funkcjonalne

Metoda `.reduce()`

Metoda `.reduce()` (ES3, ~~IE<9~~) wywołuje przekazaną do niej **funkcję wyższego rzędu**, której pierwszy parametr to zwrócony wynik poprzedniej iteracji, drugi – aktualnie przetwarzany element tablicy, opcjonalny trzeci – indeks tego elementu, opcjonalny czwarty – tablica, do której ten element należy. Opcjonalny drugi parametr **metody** `.reduce()` to wartość początkowa (ang. `initial value`). Np.

```
var nr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  
var nrSum = nr.reduce(function(reduced, element, index, array) {  
    return reduced += element;  
}, 100);
```

```
console.log(nrSum); // zwraca 155
```

Zadanie 11.

Metoda `.reduce()`

Zadanie 11.

Metoda `.reduce()`

- Dana jest **tablica z liczbami** `[1, 3, 4, 5, 10]`.
- W dokumencie znajduje się przycisk, po kliknięciu którego, poniżej w akapicie ma zostać **wyświetlona średnia arytmetyczna** liczb znajdujących się w tablicy.
- Za pomocą metody `.reduce()`, **zsumuj wszystkie elementy tablicy**, a następnie **wyciągnij z tej sumy średnią arytmetyczną**.
- Średnia arytmetyczna, w przypadku naszej tablicy, ma wynosić `4.6`.

Zadanie 12.

Metoda `.reduce()`

Zadanie 12.

Metoda `.reduce()`

- Dana jest **tablica obiektów z krajami**.
- Każdy element tablicy jest obiektem z 3 składowymi — `country`, `population` oraz `year`.
- W dokumencie znajduje się przycisk, po kliknięciu którego, poniżej w akapicie — za pomocą metody `.reduce()` — ma zostać **zwrócona suma ludności wszystkich krajów zawartych w tablicy, ale bez Kanady**.
- Pisząc warunek w ciele strzałkowej funkcji wyższego rzędu, skorzystaj z **3-częściowego operatora warunkowego** (ang. *ternary operator*).
- Oczekiwany wynik, w przypadku naszej tablicy obiektów, to 254450475.

Zadanie 13.

Metoda `.reduce()`

Zadanie 13.

Metoda `.reduce()`

- Dana jest **tablica**, której **elementami** są 2 inne tablice.
- W każdej z tych 2 tablic, elementami są **obiekty** z 3 składowymi (`id`, `title` oraz `artist`), które **reprezentują poszczególne piosenki**.
- Mamy przycisk, po kliknięciu którego, w konsoli — za pomocą metody `.reduce()` — ma zostać **zwrócona 1 połączona tablica z piosenkami**, powstała z tych 2 tablic wchodzących w skład głównej tablicy.
- W celu **połączenia tablic w jedną**, skorzystaj z metody `.concat()`.
- W przypadku naszej **tablicy tablic z obiektami**, mamy otrzymać:
`Array(5) [{...}, {...}, {...}, {...}, {...}]`.

Zadanie 14.

Metody `.map()`, `.filter()`, `.reduce()`

Zadanie 14.

Metody `.map()`, `.filter()`, `.reduce()` (1/2)

- W dokumencie znajduje się przycisk, po kliknięciu którego, w alercie mają zostać **wyświetlone tytuły piosenek dłuższych niż 3 minuty**.
- Dane są 2 tablice, których elementami są obiekty z piosenkami.
- **Utwórz nową tablicę** składającą się z tych 2 tablic.
- Co ważne, poszczególne metody wywołuj kaskadowo (łańcuchowo).
- Korzystając z metod `.reduce()` oraz `.concat()`, złącz te 2 tablice zawarte w nowej tablicy.

Zadanie 14.

Metody `.map()`, `.filter()`, `.reduce()` (2/2)

- Korzystając z metody `.map()`, stwórz nową tablicę z obiektami, zamieniając liczbę sekund na zaokrągloną liczbę minut.
- Korzystając z metody `.filter()`, zwróć tylko te piosenki, które są dłuższe niż 3 minuty.
- Korzystając z metody `.map()`, stwórz nową tablicę, umieszczając w niej tylko tytuły piosenek.
- Korzystając z metody `.join()`, złącz elementy tablicy w 1 ciąg znaków z przekazanym przecinkiem jako separatorem.
- W przypadku naszych tablic z obiektami reprezentującymi poszczególne piosenki, mamy otrzymać: Następna stacja, Ayo.

Programowanie funkcjonalne

Metoda `.find()`

Metoda `.find()` (ES6, ~~IE~~) zwraca pierwszy napotkany element tablicy, który spełnia warunek zadany przez **funkcję wyższego rzędu**, której pierwszy parametr to aktualnie przetwarzany element tablicy, opcjonalny drugi – indeks tego elementu, opcjonalny trzeci – tablica, do której ten element należy. Oryginalna tablica pozostaje niezmieniona. Np.

```
var nr = [4, 5, 8, 9, 10, 12, 13, 14, 15];  
var nrWarunek = nr.find(function(element, index, array) {  
    return element >= 10;  
});  
  
console.log(nrWarunek); // zwraca 10
```

Fetch API

(ang. *fetch* — przynieść, sprowadzić)

Can I use... Support tables for HTML5, CSS3, etc

caniuse
.com

[Home](#) [News](#) [July 18, 2018 - New feature: Shared Array Buffer](#) [Compare browsers](#) [About](#)

Can I use ? [Settings](#)
2 results found

Fetch - LS

Usage % of all users
Global 87.76% + 0.04% = 87.81%
Poland 92.2%

A modern replacement for XMLHttpRequest.

[Current aligned](#) [Usage relative](#) [Date relative](#) [Show all](#)

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			66		10.3				4
		60	67		11.2				6.2
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

[Notes](#) [Sub-features \(1\)](#) [Known issues \(0\)](#) [Resources \(7\)](#) [Feedback](#)

AbortController & AbortSignal - LS

Usage % of all users
Global 64.04%
Poland 85.78%

Controller object that allows you to abort one or more DOM requests made with the Fetch API.

[Current aligned](#) [Usage relative](#) [Date relative](#) [Show all](#)

RANDOM USER GENERATOR

A free, [open-source](#) API for generating random user data. Like Lorem Ipsum, but for people.

[Follow us @randomapi](#)



Hi, My name is

Pedro Palmer



SPONSORED

RandomAPI

Want to create your own **customized** data generator for your application?
Check out our other service RandomAPI!

[Learn More](#)


```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "mr",
        "first": "gio",
        "last": "aarsen"
      },
      "location": {
        "street": "5790 willem van noortplein",
        "city": "krimpenerwaard",
        "state": "overijssel",
        "postcode": "73083",
        "coordinates": {
          "latitude": "39.6973",
          "longitude": "-48.0029"
        },
        "timezone": {
          "offset": "-8:00",
          "description": "Pacific Time (US & Canada)"
        }
      },
      "email": "gio.aarsen@example.com",
      "login": {
        "uuid": "5029f7ab-96eb-4a5a-9b25-534a7d900eb7",
        "username": "blackfrog408",
        "password": "ravens",
        "salt": "Jz5RNvOL",
        "md5": "afa6ba11e668e12f6792d20d274ac82f",
        "sha1": "945781ae11f2917ab90d69fd5d36f1e8598c1ced",
        "sha256": "7a0b7188638f3dd3559f1c877966b8b0eaeb7f854617f457f6b1edc5fb4c7ca1"
      },
      "dob": {
        "date": "1956-05-07T18:13:01Z",
        "age": 62
      },
      "registered": {
        "date": "2007-03-23T07:16:30Z",
        "age": 11
      },
      "phone": "(485)-008-7458",
      "cell": "(893)-488-7250",
      "id": {
        "name": "BSN",
        "value": "20314737"
      },
      "picture": {
        "large": "https://randomuser.me/api/portraits/men/64.jpg",
        "medium": "https://randomuser.me/api/portraits/med/men/64.jpg",
        "thumbnail": "https://randomuser.me/api/portraits/thumb/men/64.jpg"
      },
      "nat": "NL"
    }
  ],
  "info": {
    "seed": "ccc7504421b02c0f",
    "results": 1,
    "page": 1,
    "version": "1.2"
  }
}
```

randomuser
.me/
api

```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "mr",
        "first": "nooa",
        "last": "jokela"
      },
      "location": {
        "street": "4710 esplanadi",
        "city": "veteli",
        "state": "central ostrobothnia",
        "postcode": "88733",
        "coordinates": {
          "latitude": "-10.3840",
          "longitude": "-96.2540"
        },
        "timezone": {
          "offset": "+5:00",
          "description": "Ekaterinburg, Islamabad, Karachi, Tashkent"
        }
      },
      "email": "nooa.jokela@example.com",
      "login": {
        "uuid": "a3a71f1d-ec35-41c7-bb76-855e6104c26d",
        "username": "heavygorilla595",
        "password": "qiao",
        "salt": "4VmvgYES",
        "md5": "6dd1ccc18978ad1a3aab94d3fb2683ee",
        "sha1": "8088339545e0b25ce65dfaadb4b128d758d2f4d8",
        "sha256": "6da5113752ef79fd15b433afb144285ede5e85ddb41fe59b702703419611833a"
      },
      "dob": {
        "date": "1962-11-29T10:56:35Z",
        "age": 55
      },
      "registered": {
        "date": "2007-05-31T18:20:12Z",
        "age": 11
      },
      "phone": "05-103-713",
      "cell": "048-522-54-93",
      "id": {
        "name": "HETU",
        "value": "NaNNA397undefined",
        "picture": {
          "large": "https://randomuser.me/api/portraits/men/93.jpg",
          "medium": "https://randomuser.me/api/portraits/med/men/93.jpg",
          "thumbnail": "https://randomuser.me/api/portraits/thumb/men/93.jpg"
        },
        "nat": "FI"
      }
    }
  ],
  "info": {
    "seed": "3889fd1c6fb4f3f9",
    "results": 1,
    "page": 1,
    "version": "1.2"
  }
}
```

randomuser
.me/
api

```
{
  "results": [
    {
      "gender": "male",
      "name": {
        "title": "mr",
        "first": "kenneth",
        "last": "jenkins"
      },
      "location": {
        "street": "9989 sunset st",
        "city": "port macquarie",
        "state": "victoria",
        "postcode": "2653",
        "coordinates": {
          "latitude": "-83.0925",
          "longitude": "-55.6331"
        },
        "timezone": {
          "offset": "+3:30",
          "description": "Tehran"
        }
      },
      "email": "kenneth.jenkins@example.com",
      "login": {
        "uuid": "9798cbd2-2147-4fa4-b0db-2cd4c47a34ea",
        "username": "smallpanda233",
        "password": "2525",
        "salt": "lfrC7Ptn",
        "md5": "120e8e1dc6c3c71aabecfb3d38f45d77",
        "sha1": "4ba2a3e84ce038ee1fd7539e6fb643bdbbba85b0",
        "sha256": "ec32c552d3e8fb1800238c1567ed364e5d1438694c4e5361713b1470a02d2385"
      },
      "dob": {
        "date": "1986-01-23T16:51:15Z",
        "age": 32
      },
      "registered": {
        "date": "2016-01-28T07:07:57Z",
        "age": 2
      },
      "phone": "08-2462-3965",
      "cell": "0401-531-255",
      "id": {
        "name": "TFN",
        "value": "334142741"
      },
      "picture": {
        "large": "https://randomuser.me/api/portraits/men/30.jpg",
        "medium": "https://randomuser.me/api/portraits/med/men/30.jpg",
        "thumbnail": "https://randomuser.me/api/portraits/thumb/men/30.jpg"
      },
      "nat": "AU"
    }
  ],
  "info": {
    "seed": "36cec0982804255a",
    "results": 1,
    "page": 1,
    "version": "1.2"
  }
}
```

randomuser
.me/
api

JSONPlaceholder

- Fake online
REST API
for developers

jsonplaceholder
.typicode.com

JSONPlaceholder

Fake Online REST API for Testing and Prototyping

Powered by [JSON Server](#) and [LowDB](#)

Sponsors



Traversy Media

Intro

JSONPlaceholder is a free online REST service that you can use whenever you need some fake data.

It's great for tutorials, faking a server, sharing code examples, ...

Example

Run this code in a console or from anywhere.

HTTP and HTTPS are both supported.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
```

/posts
100 items

jsonplaceholder
.typicode.com/
posts

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
    "body": "nullam et esse reiciendis voluptatibus adipisci\nsit amet autem assumenda"
```

/comments
500 items

jsonplaceholder
.typicode.com/
comments

```
[
  {
    "postId": 1,
    "id": 1,
    "name": "id labore ex et quam laborum",
    "email": "Eliseo@gardner.biz",
    "body": "laudantium enim quasi est quidem magnam voluptate ipsam eos\ntempora quo necessitatibus\ndolor quam autem quasi\nreiciendis et nam sapiente accusantium"
  },
  {
    "postId": 1,
    "id": 2,
    "name": "quo vero reiciendis velit similique earum",
    "email": "Jayne_Kuhic@sydney.com",
    "body": "est natus enim nihil est dolore omnis voluptatem numquam\net omnis occaecati quod ullam at\nvoluptatem error expedita pariatur\nnnihil sint nostrum voluptatem reiciendis et"
  },
  {
    "postId": 1,
    "id": 3,
    "name": "odio adipisci rerum aut animi",
    "email": "Nikita@garfield.biz",
    "body": "quia molestiae reprehenderit quasi aspernatur\naut expedita occaecati aliquam eveniet laudantium\nomnis quibusdam delectus saepe quia accusamus maiores nam est\ncum et ducimus et vero voluptates excepturi deleniti ratione"
  },
  {
    "postId": 1,
    "id": 4,
    "name": "alias odio sit"
```

/albums
100 items

jsonplaceholder
.typicode.com/
albums

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "quidem molestiae enim"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "sunt qui excepturi placeat culpa"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "omnis laborum odio"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "non esse culpa molestiae omnis sed optio"
  },
  {
    "userId": 1,
    "id": 5,
    "title": "eaque aut omnis a"
  },
  {
    "userId": 1,
    "id": 6,
    "title": "natus impedit quibusdam illo est"
  }
]
```

/photos
5000 items

jsonplaceholder
.typicode.com/
photos

```
[
  {
    "albumId": 1,
    "id": 1,
    "title": "accusamus beatae ad facilis cum similique qui sunt",
    "url": "http://placeholder.it/600/92c952",
    "thumbnailUrl": "http://placeholder.it/150/92c952"
  },
  {
    "albumId": 1,
    "id": 2,
    "title": "reprehenderit est deserunt velit ipsam",
    "url": "http://placeholder.it/600/771796",
    "thumbnailUrl": "http://placeholder.it/150/771796"
  },
  {
    "albumId": 1,
    "id": 3,
    "title": "officia porro iure quia iusto qui ipsa ut modi",
    "url": "http://placeholder.it/600/24f355",
    "thumbnailUrl": "http://placeholder.it/150/24f355"
  },
  {
    "albumId": 1,
    "id": 4,
    "title": "culpa odio esse rerum omnis laboriosam voluptate repudiandae",
    "url": "http://placeholder.it/600/d32776",
    "thumbnailUrl": "http://placeholder.it/150/d32776"
  },
  {
    "albumId": 1,
```


/todos
200 items

jsonplaceholder
.typicode.com/
todos

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "delectus aut autem",
    "completed": false
  },
  {
    "userId": 1,
    "id": 2,
    "title": "quis ut nam facilis et officia qui",
    "completed": false
  },
  {
    "userId": 1,
    "id": 3,
    "title": "fugiat veniam minus",
    "completed": false
  },
  {
    "userId": 1,
    "id": 4,
    "title": "et porro tempora",
    "completed": true
  },
  {
    "userId": 1,
    "id": 5,
    "title": "laboriosam mollitia et enim quasi adipisci quia provident illum",
    "completed": false
  }
]
```

/users
10 items

jsonplaceholder
.typicode.com/
users

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains"
```

Placeholder.com

- Quick & Simple Placeholder Images, Text & More

placeholder
.com

Welcome to **PLACEHOLDER.COM**

Quick & simple image placeholders.

Just put your image size after our URL and you'll get a placeholder image.

Like this:
`http://via.placeholder.com/350x150`

You can also use it in your code, like this:

```
<a href="https://placeholder.com">  
</a>
```

Have fun!

Custom Text

?text=Hello+World

Custom text can be entered using a query string at the very end of the url. This is optional, default is the image dimensions (300x250). A-z (upper and lowercase), numbers, and most symbols will work just fine.

Note: Spaces become +

**`http://via.placeholder.com/300?
text=Placeholder.com+rocks!`**

Other cool stuff you can do:

Format

Text

Color

Size

See also:

Lorem Ipsum

Li European Lingues

Filler Text

As used by:

Blogging.com

WholsHostingThis

WebsiteBuilders.com

Zadanie 15.

Fetch API — pokazanie zasobu

Zadanie 15.

Fetch API – pokazanie zasobu

- Korzystając z metody **.fetch()**, **pobierz** w formacie JSON wpis o **id=1**.
- `https://jsonplaceholder.typicode.com/posts/1`
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera.

Zadanie 16.

Fetch API — listing zasobów

Zadanie 16.

Fetch API – listing zasobów

- Korzystając z metody `.fetch()`, **pobierz** w formacie JSON **wszystkie wpisy**.
- `https://jsonplaceholder.typicode.com/posts`
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona — oczywiście już wcześniej „zjsonowana” — odpowiedź serwera.

Zadanie 17.

Fetch API – tworzenie zasobu (POST)

Zadanie 17.

Fetch API – tworzenie zasobu (POST)

- Korzystając z metody `.fetch()` oraz metody komunikacji POST, dodaj wpis o parametrach: `userId: 1, title: 'Kot w butach', body: 'Lorem ipsum...'`.
- `https://jsonplaceholder.typicode.com/posts`
- Pamiętaj, że **dane wysyłane na serwer** muszą być ciągiem znaków.
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera.

Zadanie 18.

Fetch API – uaktualnianie zasobu (PUT)

Zadanie 18.

Fetch API – uaktualnianie zasobu (PUT)

- Korzystając z metody `.fetch()` oraz metody komunikacji PUT, uaktualnij cały wpis o `id=1`, definiując jego nowe parametry: `userId: 1, id: 1, title: 'Kot w butach', body: 'Lorem ipsum...'`.
- `https://jsonplaceholder.typicode.com/posts/1`
- Pamiętaj, że **dane wysyłane na serwer** muszą być ciągiem znaków.
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera.

Zadanie 19.

Fetch API – uaktualnianie zasobu (PATCH)

Zadanie 19.

Fetch API – uaktualnianie zasobu (PATCH)

- Korzystając z metody `.fetch()` oraz metody komunikacji PATCH, uaktualnij wpis o `id=1`, definiując nową wartość tylko jednego parametru: `title: 'Kot w butach'`.
- `https://jsonplaceholder.typicode.com/posts/1`
- Pamiętaj, że **dane wysyłane na serwer** muszą być ciągiem znaków.
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera.

Zadanie 20.

Fetch API – usuwanie zasobu (DELETE)

Zadanie 20.

Fetch API – usuwanie zasobu (DELETE)

- Korzystając z metody `.fetch()` oraz metody komunikacji DELETE, usuń wpis o `id=1`.
- `https://jsonplaceholder.typicode.com/posts/1`
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera.

Zadanie 21.

Fetch API — odczyt pliku tekstowego

Zadanie 21.

Fetch API – odczyt pliku tekstowego

- Korzystając z metody `.fetch()`, odczytaj zawartość pliku tekstowego.
- `./lukasz_krawczyk_zad21_fetch_api_lorem_ipsum.txt`
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w konsoli ma zostać wyświetlona – oczywiście już wcześniej „stekstowana” – odpowiedź serwera.

Zadanie 22.

Fetch API — wyświetlenie obrazków

Zadanie 22.

Fetch API – wyświetlenie obrazków

- Korzystając z metody **.fetch()**, **pobierz** w formacie JSON **kolekcję zdjęć** (tablicę obiektów ze zdjęciami).
- `https://jsonplaceholder.typicode.com/photos`
- W przypadku **błędu**, w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd:\n' + error`.
- W przypadku **powodzenia**, w elemencie body ma zostać wyświetlona – oczywiście już wcześniej „zjsonowana” – odpowiedź serwera pod postacią **pierwszych 10 miniatur**, wraz z linkami do większych zdjęć.
- Skorzystaj z metod **.map()**, **.filter()** oraz **.reduce()**.

Obiekt Promise

(ang. *promise* — obietnica)

Can I use... Support tables for HTML5, CSS3, etc

caniuse
.com

[Home](#) [News](#) [July 18, 2018 - New feature: Shared Array Buffer](#) [Compare browsers](#) [About](#)

Can I use ? [Settings](#)

2 results found

Promises OTHER

Usage % of all users

Global 89.52% + 0.02% = 89.55%

Poland 93.14%

A promise represents the eventual result of an asynchronous operation.

[Current aligned](#) [Usage relative](#) [Date relative](#) [Show all](#)

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			66		10.3				4
		60	67		11.2				6.2
11	17	61	68	11.1	11.4	all	67	11.8	7.2
	18	62	69	12	12				
		63	70	TP					
			71						

[Notes](#) [Known issues \(0\)](#) [Resources \(8\)](#) [Feedback](#)

No notes

Async functions OTHER

Usage % of all users

Global 85.06%

Poland 90.17%

Async function make it possible to treat functions returning Promise objects as if they were synchronous.

Zadanie 23.

Obiekt Promise

Zadanie 23.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz obietnicę otrzymania prezentu.
- W przypadku powodzenia (metoda **.then()**), w konsoli ma zostać wyświetlony **obiecany napis**: Oto prezent!
- W przypadku błędu (metoda **.catch()**), w konsoli ma zostać wyświetlony komunikat w postaci: 'Błąd: \n' + error.

Zadanie 24.

Obiekt Promise

Zadanie 24.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz obietnicę **powitania** po upływie 3 sekund.
- W przypadku **powodzenia** (metoda **.then()**), w konsoli ma zostać wyświetlony **obiecany napis**: Hello, world!
- W przypadku **błędu** (metoda **.catch()**), w konsoli ma zostać wyświetlony komunikat w postaci: 'Błąd:\n' + error.

Zadanie 25.

Obiekt Promise

Zadanie 25.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz obietnicę powitania.
- W przypadku powodzenia (metoda **.then()**), w konsoli ma zostać wyświetlony napis: `Hello, user!`, wówczas w konsoli ma zostać wyświetlony **obiecany napis**: `Hello, world!`, wówczas, ale już po 3 sekundach, w konsoli ma zostać wyświetlony napis: `Bye, everybody!`
- W przypadku błędu (metoda **.catch()**), w konsoli ma zostać wyświetlony komunikat w postaci: `'Błąd: \n' + error`.

Zadanie 26.

Obiekt Promise

Zadanie 26.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz 3 różne obietnice – pierwszego powitania, drugiego powitania oraz pożegnania.
- W przypadku powodzenia (metoda **.then()**) każdej z tych obietnic, w konsoli mają zostać wyświetlone odpowiednio – **obiecany napis**: Hello, world!, po 3 sekundach **obiecany napis**: Hello, user!, a po 5 sekundach **obiecany napis**: Bye, everybody!
- W przypadku błędu (metoda **.catch()**), w konsoli mają zostać wyświetlone komunikaty w postaci: 'Błąd:\n' + error (z opóźnieniem takim, jak w przypadku powodzenia).

Zadanie 27.

Obiekt Promise

Zadanie 27.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz 3 różne obietnice – pierwszego powitania, drugiego powitania oraz pożegnania.
- Za pomocą metody **Promise.all()**, w przypadku powodzenia każdej z tych obietnic (metoda **.then()**), w konsoli mają zostać wyświetlone odpowiednio – **obiecany napis**: Hello, world!, po 3 sekundach **obiecany napis**: Hello, user!, a po 5 sekundach **obiecany napis**: Bye, everybody!
- Za pomocą tej samej metody **Promise.all()**, w przypadku błędu (metoda **.catch()**), w konsoli mają zostać wyświetlone komunikaty w postaci: 'Błąd: \n' + error (z opóźnieniem takim, jak w przypadku powodzenia).

Zadanie 28.

Obiekt Promise

Zadanie 28.

Obiekt Promise

- Korzystając z obiektu **Promise**, stwórz obietnicę **otrzymania prezentu**.
- W przypadku **powodzenia** (metoda **.then()**), w konsoli ma zostać wyświetlony **obiecany napis**: Oto prezent!.
- W przypadku **błędu** (ta sama metoda **.then()**), w konsoli ma zostać wyświetlony **obiecany napis**: Prezentu nie ma.



Dzięki!

Masz jakieś pytania?

- lukasz.krawczyk.lublin@gmail.com
- goldenline.pl/lukasz-krawczyk81/
- pl.linkedin.com/in/lukasz-krawczyk