FUTURE DEV DAY

2015

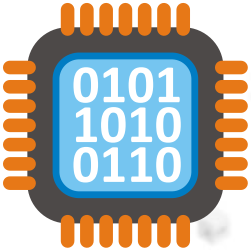
Mateusz Łyczek

ReactiveX - okiełznać asynchroniczność

Future Processing

Google
facebook

Google
facebook


0101
1010
0110

Problemy z asynchronicznością

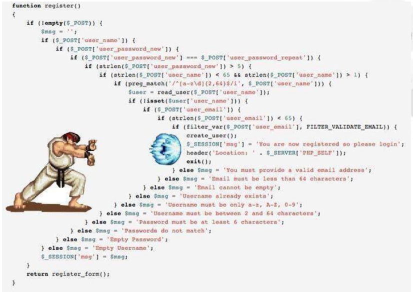
- Różne API w każdym języku
  - `Thread` w Javie
  - `async/await` w .NET
  - callbacks, `Promise` w JavaScript
  - ...

Problemy z asynchronicznością

- Różne API w każdym języku
  - `Thread` w Javie
  - `async/await` w .NET
  - callbacks, `Promise` w JavaScript
  - ...
- Ciężka obsługa błędów (`try/catch`?)

Problemy z asynchronicznością

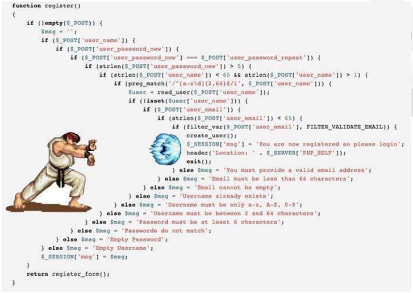
- Różne API w każdym języku
    - Thread w Javie
    - `async/await` w .NET
    - callbacks, `Promise` w JavaScript
    - ...
- Ciężka obsługa błędów (`try/catch`?)
- Callback hell



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

Problemy z asynchronicznością

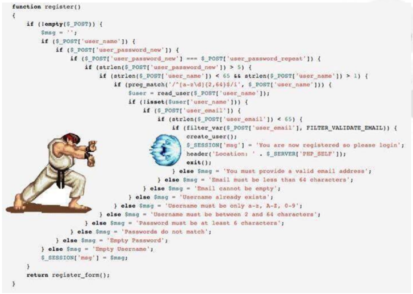
- Różne API w każdym języku
  - `Thread` w Javie
  - `async/await` w .NET
  - callbacks, `Promise` w JavaScript
  - ...
- Ciężka obsługa błędów (`try/catch`?)
- Callback hell
- Zdarzenia?



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

Problemy z asynchronicznością

- Różne API w każdym języku
  - Thread w Javie
  - async/await w .NET
  - callbacks, Promise w JavaScript
  - ...
- Ciężka obsługa błędów (try/catch?)
- Callback hell
- Zdarzenia?
- Kompozycja?



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

ReactiveX

An API for asynchronous programming with observable streams

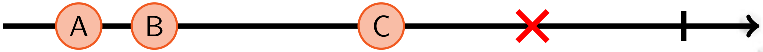
ReactiveX is a combination of the best ideas from the Observer pattern, the Iterator pattern and functional programming

ReactiveX

An API for asynchronous programming
with observable streams

ReactiveX is a combination of the best ideas from the Observer pattern, the Iterator pattern and functional programming

Observable streams



●	element strumienia
✗	błąd
|	koniec strumienia

Co może być elementem strumienia?

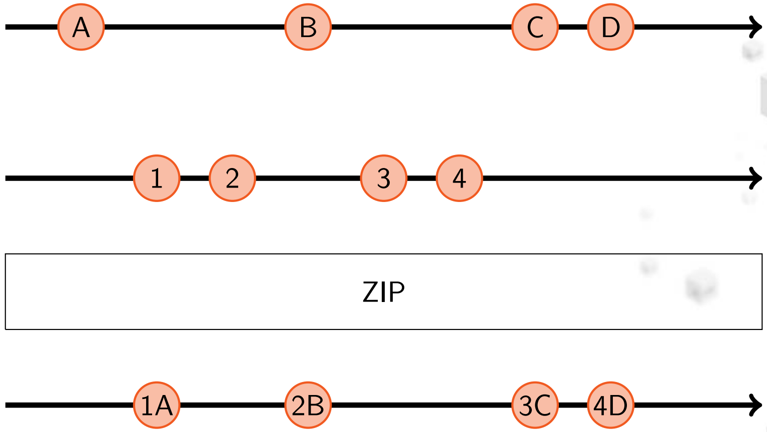
Co może być elementem strumienia?

wszystko

Co może być elementem strumienia?

- zdarzenia użytkownika (np. klikanie myszki)
- `String`
- odpowiedź serwera na zapytanie HTTP
- tweet
- pojedynczy wiersz wyniku zapytania SQL
- ...

Łączenie strumieni



Operatory

`From`	`Join`	`Range`	`Interval`
`Just`	`FlatMap`	`GroupBy`	`Map`
`Scan`	`Window`	`Debounce`	`Distinct`
`ElementAt`	`Filter`	`First`	`Last`
`Sample`	`Skip`	`SkipLast`	`Take`
`TakeLast`	`And`	`CombineLatest`	`Join`
`Merge`	`Switch`	`Zip`	`Retry`

...

Tylko Java i JavaScript?

- Java
- JavaScript
- C#
- Scala
- Clojure
- C++
- Ruby
- Python
- Groovy
- JRuby
- Kotlin
- Swift

Co dalej?

http://reactivex.io

Główne źródło wiedzy na temat ReactiveX wraz z linkami do dokumentacji dla wszystkich implementacji w różnych językach

http://rxmarbles.com/

Interaktywne diagramy operatorów

https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

“The introduction to Reactive Programming you’ve been missing”

https://github.com/mlyczek/fdd2015-reactivex

Slajdy z tej prezentacji oraz ich kod (LaTeX)

Co dalej?

?

Future Processing

{FDD FUTURE DEV DAY

WWW.FUTUREDEVDAY.PL

deadline24

QUALITY EXCITES

wud silesia
coś więcej niż konferencja

DPTO
Dobre Praktyki Tworzenia Oprogramowania

FPAcademy

Adaptive Vision