

# JavaScript - komunikacja po HTTP




# Hello

Adrian Kukowski

Senior Frontend Developer at LIBRUS

# Agenda

1. Promise
2. XMLHttpRequest vs Fetch
3. Pobieramy dane z serwera metoda GET
4. Pobieranie danych sekwencyjnie oraz równolegle
5. Promise.all
-  6. Async i await

# Promise

Jak sama nazwa mówi jest to obietnica, czyli umawiamy się z funkcją promise, że jak skończy to poinformuje nas o tym fakcie

```
const promise1 = new Promise((resolve, reject) => {  
  setTimeout(() => {  
    resolve('foo');  
  }, 300);  
});  
  
promise1.then((value) => {  
  console.log(value);  
  // expected output: "foo"  
});  
  
console.log(promise1);  
// expected output: [object Promise]
```

# Promise

Jeżeli promise zwróci **resolve** to dane uzyskamy w metodzie **then** a jeżeli zwróci **reject** to w metodzie **catch** z założenia:

**Resolve** - wszystko poszło poprawnie

**Reject** - coś nie wyszło

# XMLHttpRequest vs Fetch

## XMLHttpRequest

```
const xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    // Typical action to be performed when the
    document is ready:
  }
};
xhttp.open("GET", "url", true);
xhttp.send();
```

## Fetch

```
fetch('http://example.com/movies.json')
  .then(response => response.json())
  .then(data => console.log(data));
```

# Pobieramy dane z serwera metoda GET

Adres url z którego będziemy pobierali dane

```
https://jsonplaceholder.typicode.com/todos/
```

Celem zadania jest pobranie danych z api oraz wyświetlanie ich w dwóch listach na stronie jako zakończone zadania i niezakończone zadania przykładowo

Zakończone zadania

☒ Jakiś tytuł

Niezakończone zadania

☐ Jakiś tytuł

# Pobieranie danych sekwencyjnie oraz równoległe

## Pobieranie sekwencyjne

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => {  
    fetch('http://example.com/movies.json')  
      .then(response => response.json())  
      .then(data => console.log(data));  
  });
```

Polega na pobraniu danych w konkretnej kolejności dopiero po skończeniu pierwszego pobierania zaczyna się kolejne



# Pobieranie danych sekwencyjnie oraz równoległe

## Pobieranie równoległe

```
fetch('http://example.com/movies.json')  
  .then(response => response.json())  
  .then(data => console.log(data));  
  
fetch('http://example.com/cars.json')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

Polega na pobraniu danych w tym samym momencie niezależnie od siebie

# Pobieranie danych sekwencyjnie oraz równoległe

## Pobieranie równoległe

```
Promise.all([  
  fetch('http://example.com/movies.json'),  
  fetch('http://example.com/car.json')  
)  
  .then(responses => /*obsługa responses*/)
```

Polega na pobraniu danych w tym samym momencie niezależnie od siebie ale daje nam możliwość wykonania jakiejś akcji po pobraniu wszystkich danych

# Async i await 🌶️

```
const getDate = async () => {  
  try {  
    const response = await fetch('http://example.com/movies.json');  
    const movies = await response.json();  
    console.log(movies);  
  } catch(error) {  
    console.log(error)  
  }  
};
```



# Dzięki



adrian-kukowski

@ kontakt@kukowskiadrian.pl

kukowskiadrian.pl