


React pre začiatočníkov V. AJAX, Async

Ing. Slavomír Kožár

12. 4. 2022



Obsah

- HTTP
- XHR
- Developer Tools – Network activity
- Fetch
- Promise
- Async
- Axios

- useEffect

ToDo Projekt

<https://github.com/slavokozaar/react-pre-zaciatocnikov-todo-project/>

```
$ git clone https://github.com/slavokozaar/react-pre-zaciatocnikov-todo-project.git
```

```
$ cd react-pre-zaciatocnikov-todo-project
```

```
$ npm install
```

```
$ npm start
```

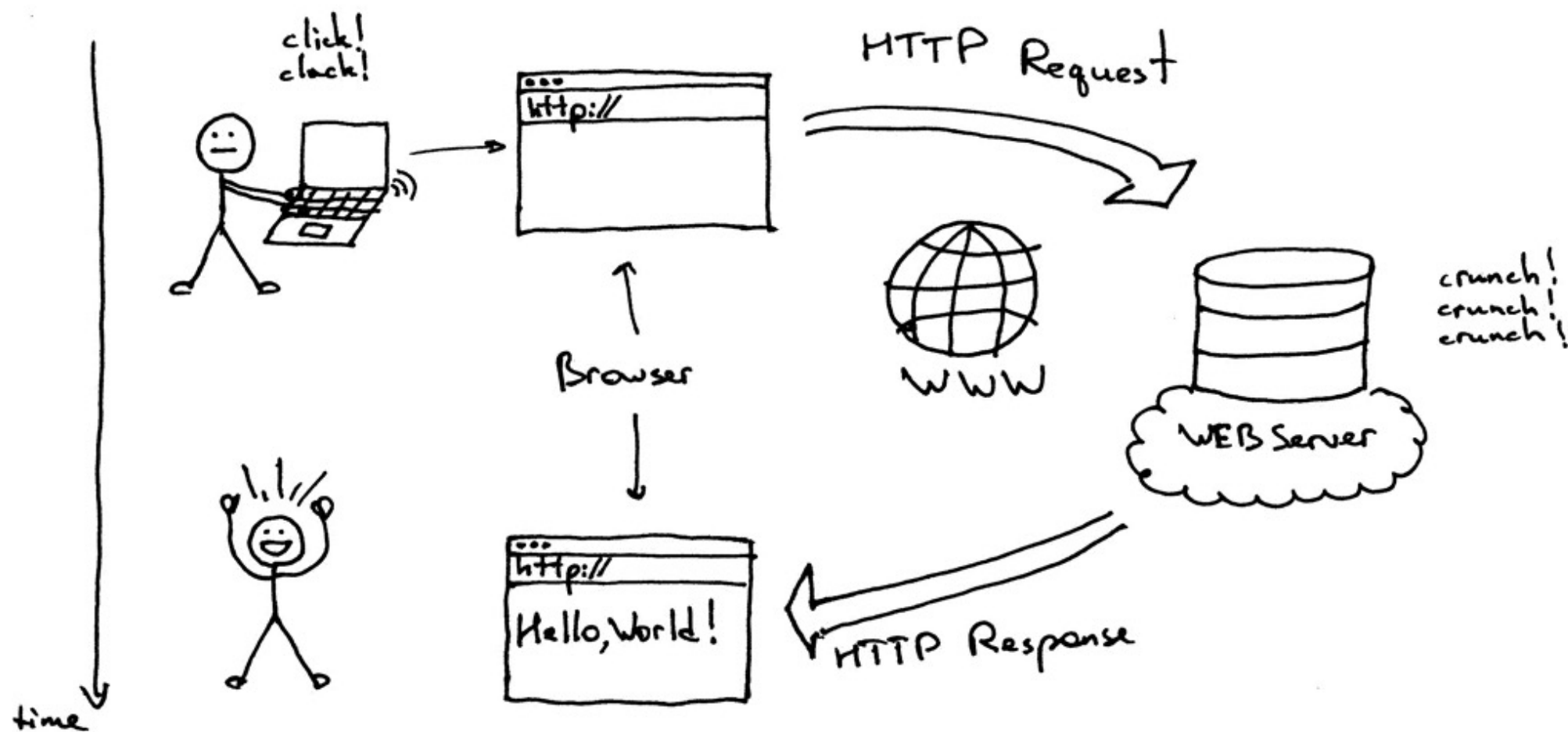
HTTP

Hyper Text Transfer Protocol

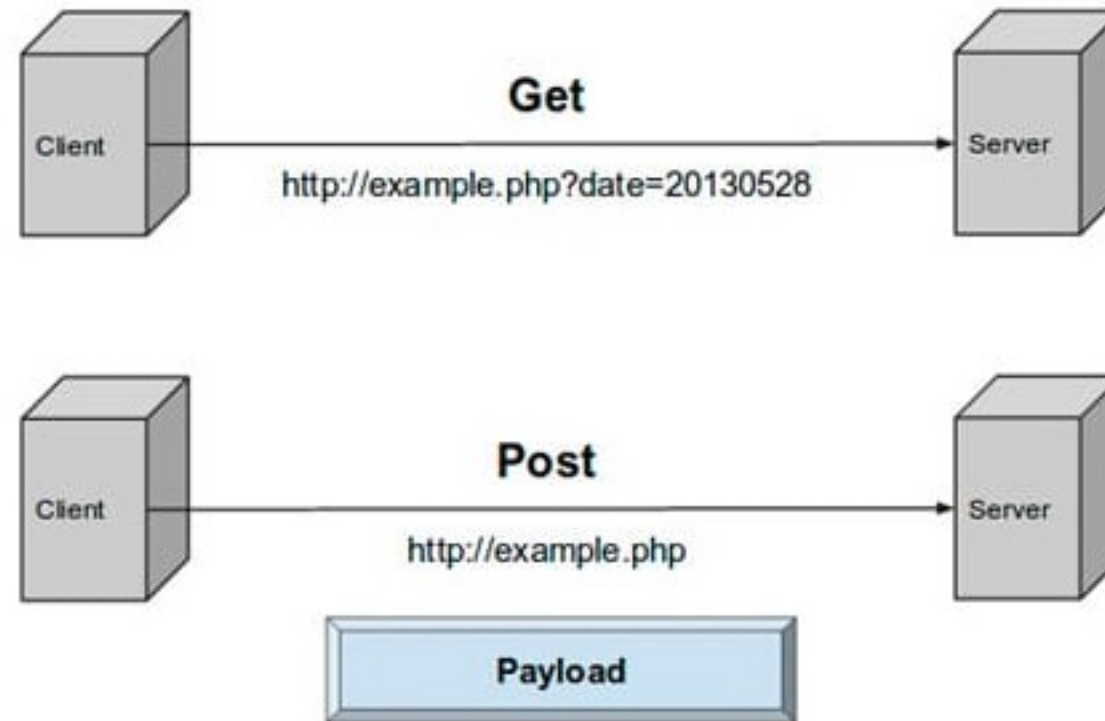
- HTTP 0.9 - Tim Berners Lee (CERN) v roku 1989
- HTTP 1.0 – v roku 1996



HTTP



GET, POST



Form Data, JSON Strings, Query Parameters, View States, etc

AJAX

Asynchronous JavaScript And XML

- 18. 2. 2005 Jesse James Garrett v článku:
Ajax: A New Approach to Web Applications

<https://thehistoryoftheweb.com/what-does-ajax-even-stand-for/>

XHR

- **XMLHttpRequest** je API vo forme objektu, ktorého metódy prenášajú dáta medzi webovým prehliadačom a webovým serverom.
- Údaje môžete získať z adresy URL bez obnovenia stránky.
- Objekt poskytuje JavaScriptové prostredie prehliadača.
- Napriek názvu je XHR možné použiť aj s inými dátami ako **XML**, ale aj **JSON**, **HTML** alebo plain text

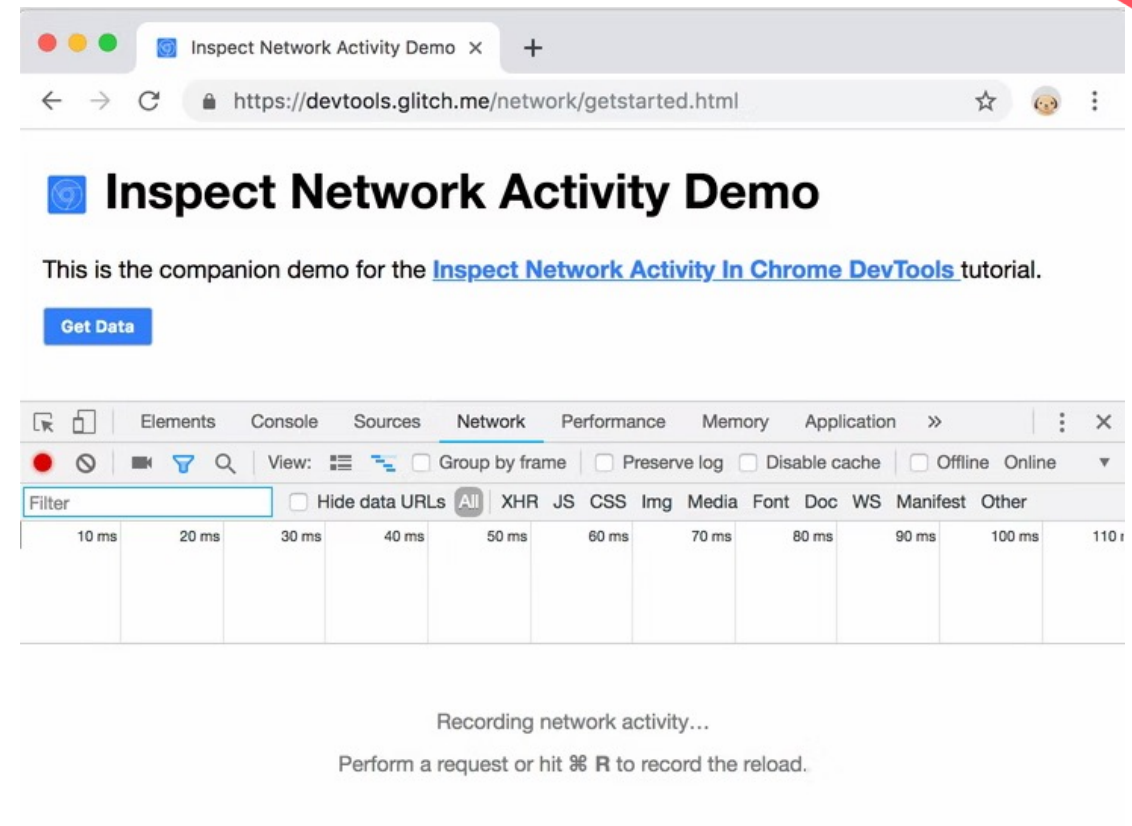
Developer Tools

Dev Tools Demo:

<https://devtools.glitch.me/network/getstarted.html>

Dev Tools Reference:

<https://developer.chrome.com/docs/devtools/network/reference/>



XHR

```
const xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {
    // kod vykonaný v momente zmeny stavu requestu...
};

xhttp.open("GET", "http://mojasuperdomena.sk", true);

xhttp.send();
```

Fetch

```
fetch('https://todo.eragon.digital/api/tasks  
?api_token=react-pro-zacatecniky')  
  .then(response => response.json())  
  .then(data => console.log(data));
```

Promise

Objekt Promise predstavuje prípadné dokončenie (alebo zlyhanie) asynchrónnej operácie a jej výslednú hodnotu.

```
const myPromise = new Promise(  
  (resolve, reject) => {  
    resolve('value')  
    reject('error')  
  }  
);  
  
myPromise  
  .then((value) => console.log(value) )  
  .catch((error) => console.log(error) );
```

Async Function

- funkcia deklarovaná s kľúčovým slovom **async**
- je v nej povolené kľúčové slovo **await**
- async a await umožňujú asynchrónne správanie založené na promises napísať **čistejším štýlom**
- vyhnutie sa **promise hell**

Promise Hell

```
connectToDatabase ()
  .then ( (database) => {
    return findAllBooks (database)
      .then ( (books) => {
        return getCurrentUser (database)
          .then ( (user) => {
            return getRecommendations (books, user);
              .then ( (recommendations) => {
                return reccomendations;
              });
          });
        });
    });
  });
```

Async Heaven

```
async function getReccomendations() {  
  
  const database    = await connectToDatabase();  
  
  const books       = await findAllBooks(database);  
  
  const user        = await getCurrentUser(database);  
  
  const recc        = await getRecommendations(books, user);  
  
  return reccommendations;  
  
}
```

axios

```
$ npm install axios
```

Axios is a *promise-based* HTTP Client for node.js and the browser.

It is *isomorphic* (= it can run in the browser and nodejs with the same codebase).

On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

axios

```
import axios from 'axios';
```

```
axios.get('url')
```

```
.then(response => // spracovanie response )
```

```
axios.post('url', {data})
```

```
axios.put('url', {data})
```

```
axios.delete('url')
```

async axios

```
import axios from 'axios';
```

```
async function getData() {  
    const response = await axios.get('url')  
  
    // spracovanie response  
}
```

useEffect()

- useEffect hook umožňuje vykonávať vedľajšie efekty v komponentoch.
- typicky načítanie dát, priama aktualizácia DOM, časovače, komplexnejšia logika...

`useEffect(() => {}, [zoznam_závislostí])`

(zoznam závislostí je nepovinný argument)

useEffect() – bez závislostí

```
import {useEffect} from 'react';  
  
useEffect( () => {  
    // kód spustený pri každej zmene stavu  
})
```

useEffect() – so závislosťou na state

```
import {useState, useEffect} from 'react';  
  
const [a, setA] = useState(0);  
  
useEffect( () => {  
    // kód spustený pri zmene stavu `a`  
}, [ a ])
```

useEffect() – so závislosťou na state

```
import {useState, useEffect} from 'react';

const [a, setA] = useState(0);
const [b, setB] = useState(0);

useEffect( () => {
    // kód spustený pri zmene stavu `a`, alebo `b`
}, [ a, b ])
```

useEffect() – s prázdnu závislosťou

```
import {useEffect} from 'react';

useEffect( () => {
    // kód spustený pri inicializácii komponenty
    // v praxi len raz...
}, [ ])
```

ToDo – načítanie taskov


1. vytvorte async funkciu, ktorá načíta data z API:

https://todo.eragon.digital/api/tasks?api_token=react-pro-zacatecniky

2. zavolajte túto funkciu v momente inicializácie komponenty
3. uložte načítané tasky do state

Congratulations!



 homm.fun

meme-arsenal.ru

ĎAKUJEME !

Ing. Slavomír Kožár

slavomir.kozar@innovis.sk

www.innovis.sk

