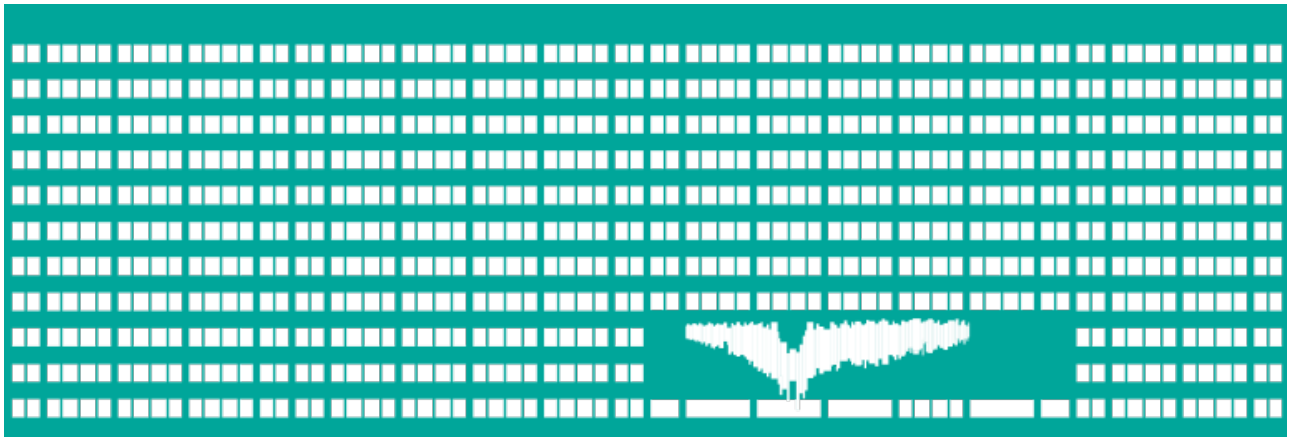
VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Fetch API/jQuery Basic HTTP communication



See: Fetch API tutorial
Codecamp examples
MDN Fetch API
http://api.jquery.com/jQuery.get/
http://api.jquery.com/jQuery.post/

TAMZ 1
Lab 5

Application deployment

Application has to be uploaded to a server

- Using of FTP/SCP/SFTP
  - e.g. scp -r -4 local_dir user@server:~/public_html/app_dir
- Using NetBeans – deployment options in project settings
  - Works only for PHP projects (HTML5 in future releases)
  - FTP/SFTP connection only
  - Defined in “Run Configuration” by selecting remote Web site and filling in the URL and connection details
- Using external deployment tool
- Creating repository (e.g. git) and using corresponding commit/update, push/pull commands
- Using cloud deployment services
- Building native application
  - e.g. Apache Cordova

HTTP protocol

- HTTP Request has following syntax:

  <METHOD> <URL/PATH> HTTP/1.<version>

  <Client header$_1$>

  …

  <Client header$_n$>

  <Empty line>

  <Client data, if any>

- HTTP Response has following syntax:

  HTTP/1.<version> <3-digit code> <Message>

  <Server header$_1$>

  …

  <Server header$_n$>

  <Empty line>

  <Data from server, if any>

Basic HTTP methods and codes

HTTP methods:

- GET
  - Asks for the resource specified by URL
- OPTIONS
  - Asks for the support of some features (methods)
- POST
  - Sends data (web form) to the server
- PUT
  - Stores the document to a file on the server
- DELETE
  - Deletes the resource

Error codes:

- 1xx – Informational
- 2xx – Successful
  - 200 – OK
- 3xx – Redirection
  - 302 Found
  - 304 Not Modified
- 4xx – Client Error
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - …
- 5xx – Server Error
  - 500 Internal Server Error
  - 502 Bad Gateway, …

Problem with JavaScript AJAX

Same-origin approach is applied when accessing remote resource via HTTP from JavaScript

- Either the same web page, or the access for different origin must be explicitly permitted in headers via CORS
  - Access-Control-Allow-Origin header for external access
  - Apache Cordova: <access origin="*/URL" /> in settings
  - If the origin does not match, the browser drops the result
  - JSONP may be a solution, we will discuss it during lectures
- Cookies may be set/retrieved within the permitted domain
  - Access-Control-Allow-Credentials to permit cookies
- Custom headers may be blocked by the server
  - Access-Control-Allow-Headers may be needed to list our custom headers which we want to send to the server
- Images, pages, … from external site may be loaded through standard HTML href and src, but without AJAX

HTTP protocol – GET method

GET /~mor03/TAMZ/TAMZ.php?login=mor03 **HTTP/1.1**
Host: linedu.vsb.cz
Connection: keep-alive
Accept: */*
Origin: http://localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36 SUSE/31.0.1650.63
Referer: http://localhost/~mor03/TAMZ1_ex/public_html/comm.html
Accept-Encoding: gzip,deflate,sdch
Accept-Language: cs,en;q=0.8,sk;q=0.6

HTTP/1.1 200 OK
Date: Sun, 09 Mar 2014 20:12:35 GMT
Server: Apache
Access-Control-Allow-Origin: *
Content-Length: 32
Connection: close
Content-Type: text/plain

8842e9d8172c3875a8b5ed8721949632

HTTP protocol – POST method

POST /~mor03/TAMZ/TAMZ.php **HTTP/1.1**
Host: linedu.vsb.cz
Content-Length: 12
Accept: */*
Origin: http://localhost
API-Token: eHh4MDAxOnh4eA==
User-Agent: Mozilla/5.0 (X11; Linux x86_64) SUSE/31.0.1650.63
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost/~mor03/TAMZ1_ex/public_html/http_test.html
Accept-Language: cs,en;q=0.8,sk;q=0.6

login=xxx001

HTTP/1.1 401 Unauthorized
Date: Mon, 10 Mar 2014 09:10:24 GMT
Server: Apache
Access-Control-Allow-Headers: API-Token
Access-Control-Allow-Origin: *
Content-Length: 43
Connection: close
Content-Type: text/plain

Sorry, the supplied secret code is invalid

Shorthand .fetch request (GET)

```
var f_r;
fetch(MY_URL_including_query_string)
      .then(result => { f_r = result; return result.text()}) //.json(), ...
      .then(data => {
        if (f_r.status >= 400) { // Server or client error from HTTP
         alert("request error (" + f_r.statusText + "): " + data);
        } else {
         console.log("C-T: "  + f_r.headers.get("Content-Type"));
         alert("success: " + data);
        }
      })
      .catch(function(error) {
              alert("error:" + error);
      })
      .finally(function() {
              alert("Request finished");
      })
```

Full form of .fetch request

We need this type of request when extended features (e.g. HTTP headers) are required (post data is in post_object):

```
const formBody = Object.keys(post_object).map(key =>
      encodeURIComponent(key) + '=' +
      encodeURIComponent(post_object[key])).join('&');
fetch(MY_URL, {
      method: "post", // Default: "get", other methods available …
      body: formBody,
      headers: { 'Header1': 'Header value 1',
         'Content-Type': 'application/x-www-form-urlencoded'
      },


})
  .then(result => result.json())
  .then(data =>  … ).catch(error => …)
```

ES8 await/async .fetch example

Make the code cleaner and without multiple .then working with promises:

```
async function fetchText(URL) {
    let response = await fetch(URL);
    console.log(`${response.status} ${response.statusText}`);

    // if (response.status === 200) { // Will not work for e.g. 204
    if (response.ok) {
        let data = await response.text();
        alert("Received data: " + data);
    } else {
      // ...
    }
}
```

Shorthand jQuery.get,.post request

```
var jqxhr = $.get(MY_URL, {attribute: 'text1'}, function(dta) {
// We can also use $.post
    alert( "success: " + dta);
})
.done(function() {
   alert( "done" );
})
.fail(function() {
   alert( "error" );
})
.always(function() {
  alert( "finished" );
});
```

Basic form of jQuery.ajax request

We need this type of request when extended features (e.g. HTTP headers) are required:

```
var jqxhr = $.ajax({
            url: MY_URL,
            type: "POST", // Also: "GET", …
            data: {attribute: 'text1'},
            headers: { 'Header1': 'Header value 1' },
            success: function(data) {
              alert('I was successful and received data: ' + data);
            }
})
.fail(function(err) {
        alert( "Something went wrong: " + err.responseText );
});
```

Task (2 points + 0.5 if uploaded)

Create application which will take:

- Your username and script URL (pre-defined value is possible). Your login will be passed as the *user* argument together with a second argument called *timestamp* indicating milliseconds since 1970-01-01 to script URL which is: https://homel.vsb.cz/~mor03/TAMZ/TAMZ22.php

You will receive a secret, base64-encoded token (as text/plain) in the reply body, which you will show decoded by atob function in a new, read-only, input.

- Post a new request to the same URL without *login* argument, but with *Authorization* header set to string "Bearer token" (i.e. "Bearer " + token_var)
- Optionally, you may upload your pages to homel.vsb.cz