Web Development and Web Design

by R.Hasko.

Draft 01.2019

Lab 5. JavaScript, AJAX, jQuery, JSON and online API.

Create new directory **Project5** inside **web_dev** at your laptop. All tasks must be synhronized with **web_dev/Project5** at your **GitHub Pages**.

Use **your original own Landing Page from Project 4 (Lab 4)** as base web page and add all new tasks (parts) to the bottom side.

Project 5 must have additionally 7 (seven) separated parts (bloks):

1. **JavaScript (Input, Output, Form Validation).**
   Create JavaScript HTML form with validation and next attributes:
   1. **max and min**. Specifies requires a value for this field.
   2. **pattern**. Specifies the value pattern for input.
   3. **valid**. Specifies the input with valid date.

   If any of form field is empty, the functions alerts a different messages, and returns false, to prevent the form from being submitted. Show alerts using functions **alert()** and **document.getElementById("...").innerHTML**. Use **<button onclick="myFunction()">** instead of **<form action="..." >** with **<... Submit>** button. Provide minimum 10 different fields/alerts. read more https://www.w3schools.com/js/js_validation.asp

2. **Drag and Drop.**
   Drag (and drop) an image back and forth between two ** ** elements. Show current position of image using **.getElementById("...").innerHTML**

3. **HTML Web Storage**.
   With web storage, web applications can store data locally within the user's browser. Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
   Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

HTML web storage provides two objects for storing data on the client:

`window.localStorage` - stores data with no expiration date

`window.sessionStorage` - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage with alert and forms for manipulate in localStorage.

Create **CRUD** functionality for **localStorage** using **web forms** and `localStorage.setItem('key', 'value'), localStorage.getItem('key'), localStorage.removeItem('key'), localStorage.clear().` Read more https://developer.mozilla.org/uk/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

Retrieve the values and show using `console.log(...)`

4. Web Workers

When executing scripts in an HTML page, the page becomes unresponsive until the script is finished. A **web worker** is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the **web worker runs in the background**. Since web workers are in external files, they do not have access to the following JavaScript objects:

- The window object
- The document object
- The parent object

Create counter using **web worker** with **set/reset** count from **web form**.

read more https://www.w3schools.com/html/html5_webworkers.asp and https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Functions_and_classes_available_to_workers

5. AJAX = Asynchronous JavaScript And XML.

AJAX is a developer's dream, because you can: read data from a web server - after the page has loaded, update a web page without reloading the page. Send data to a web server - in the background. AJAX is not a programming language. AJAX just uses a combination of:

A browser built-in `XMLHttpRequest` object (to request data from a web server)

JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

5.1 Try online https://resttesttest.com/ for understanding of basic concepts.

5.2 Create own simple test of AJAX with request to https://httpbin.org/get

Sample web page:

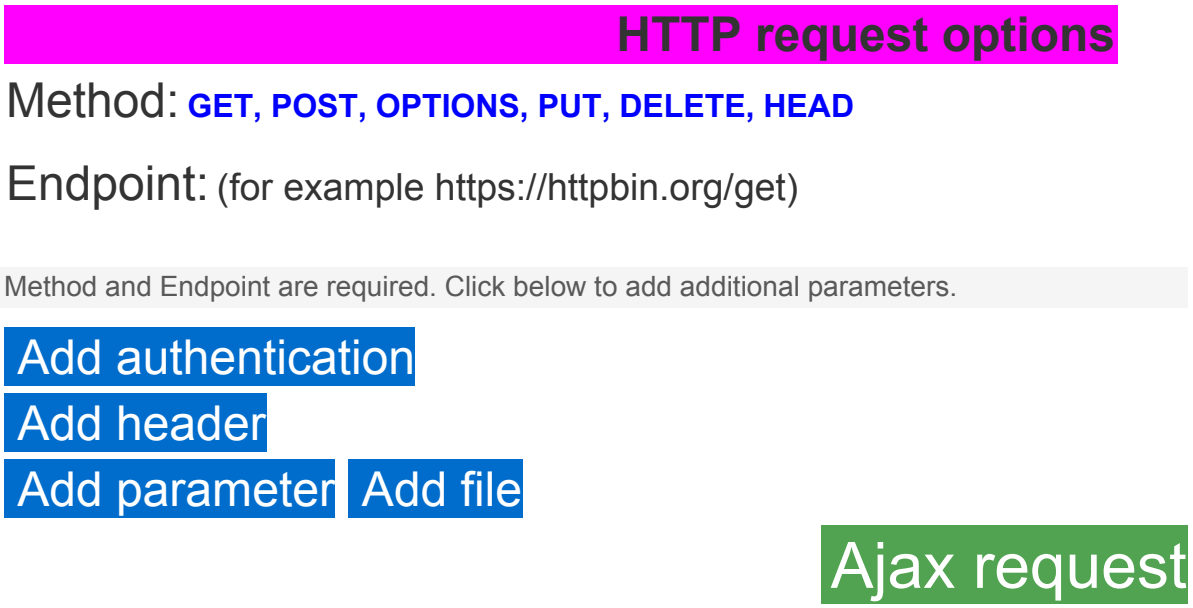
```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "https://httpbin.org/get", true);
  xhttp.send();
}
</script>
</body>
</html>
```

Sample answer:

{ "args": {}, "headers": { "Accept": "*/*", "Accept-Encoding": "gzip, deflate, br", "Accept-Language": "uk-UA,uk;q=0.9,ru;q=0.8,en-US;q=0.7,en;q=0.6", "Host": "httpbin.org", "Origin": "https://**<yourlogin>.github.io**", "Referer": "https://**<yourlogin>.github.io/web_dev/Project5**", "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,

like Gecko) Chrome/73.0.3683.103 Safari/537.36" }, "origin": "178.212.111.226, 178.212.111.226", "url": "https://httpbin.org/get" }
5.3 Create **UI (user interface)** at the **your Project 5 web page** like two boxes design of **resttesttest.com** with working functionality of **AJAX.**
For example:



6. jQuery

The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation. The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Tip: In addition, jQuery has plugins for almost any task out there.

Why jQuery?

There are lots of other JavaScript frameworks out there, but **jQuery** seems to be the most popular, and also the most extendable.

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
</head>
```

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){
  // jQuery methods go here...
});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s). jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
All selectors in jQuery start with the dollar sign and parentheses: $().

The jQuery element selector selects elements based on the element name.
Example:

```
$(document).ready(function(){
  $("button").click(function(){
      $("p").hide();
  });
});
```

jQuery is tailor-made to respond to events in an HTML page.

What are Events?
All the different visitors' actions that a web page can respond to are called events.
An event represents the precise moment when something happens.
Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

Task: Create a **set of jQuery Effects** at your Project 5 web page:

1. **jQuery Hide/Show**
2. **jQuery Fade**
3. **jQuery Slide**
4. **jQuery Animate**
5. **jQuery stop()**
6. **jQuery Callback**
7. **jQuery Chaining**

7. jQuery + AJAX + JSON

JSON: Java**S**cript **O**bject **N**otation.

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

Exchanging Data

When exchanging data between a browser and a server, the data can only be text. JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server. We can also convert any JSON received from the server into JavaScript objects. This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

If you receive data in JSON format, you can convert it into a JavaScript object:

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

JSON can very easily be translated into **JavaScript**. JavaScript can be used to make HTML in your web pages. Make an HTML table with data received as JSON:

```
obj = { table: "customers", limit: 20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
      myObj = JSON.parse(this.responseText);
    txt += "<table border='1'>"
    for (x in myObj) {
      txt += "<tr><td>" + myObj[x].name + "</td></tr>"; }
     txt += "</table>"
      document.getElementById("demo").innerHTML = txt;
  }
}
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

Read more https://www.w3schools.com/js/js_json_html.asp

For testing purpose we use https://www.jsontest.com/

JSONTest.com is a testing platform for services utilizing JavaScript Object Notation (JSON). To use, make a request to servicename.jsontest.com, where servicename is the name of a service listed below. We also support a number of parameters, such as callback, allowing you to test Javascript and other web applications. For example, try this: http://ip.jsontest.com/?callback=showMyIP

Services

1. IP Address
2. HTTP Headers
3. Date & Time
4. Echo JSON
5. Validate
6. Arbitrary JS Code
7. Cookie
8. MD5

Parameters

1. Callback
2. Allow Origin
3. MIME

Task: Create own UI for set of JSON Services at your Project 5:

1. IP Address with HTTP Headers, Date & Time
2. Echo JSON
3. Validate
4. Arbitrary JS Code
5. MD5

Summary.

Resulting updated landing page with additional 7 separated parts (blocks) must work as a static (HTML+CSS+JavaScript only) web page from **<yourlogin>.github.io/web_dev/Project5**, provide **UI** for interacting with online services resttesttest.com and jsontest.com according description above (**all 7 parts**).