Telerik Academy

Consuming Remote Data with JavaScript

Making Request Through HTTP using XmlHttpRequest object and jQuery AJAX

Telerik Software Academy

Learning & Development Team

http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

Table of Contents (2)

- Creating HTTP request through xhr
  - Callback-oriented
- jQuery Ajax
  - jQuery.ajax()
  - jQuery.getJSON() and jQuery.postJSON()
  - jQuery.load()
  - jQuery.get();

Telerik Academy

XMLHttpRequest

Telerik Academy

XMLHttpRequest

- XMLHttpRequest is a JavaScript object, that provides a way to retrieve a resource by URL
  - Designed by Microsoft, adopted by Mozilla, Apple and Google
  - Nowadays standardized in the W3C
- XHR can retrieve resources both synchronously and asynchronously
- The data can be of any format, not strictly XML
  - JSON, HTML or just plain text

Telerik Academy

Using XMLHttpRequest

Telerik Academy

Using XMLHttpRequest

- To create an HTTP request:

Telerik Academy

Using XMLHttpRequest

- To create an HTTP request:
  1. Create an instance of `XMLHttpRequest` (xhr)

```
var httpRequest = new XMLHttpRequest();
```

Telerik Academy

Using XMLHttpRequest

- To create an HTTP request:
  1. Create an instance of `XMLHttpRequest` (xhr)

```
var httpRequest = new XMLHttpRequest();
```

  2. Make the request

```
httpRequest.open("GET", endpointUrl, true);
```

Telerik Academy

Using XMLHttpRequest

- To create an HTTP request:
  1. Create an instance of `XMLHttpRequest` (xhr)

```
var httpRequest = new XMLHttpRequest();
```

  2. Make the request

```
httpRequest.open("GET", endpointUrl, true);
```

  3. Send it to the server

```
httpRequest.send(null);
```

Telerik Academy

Using XMLHttpRequest

Live Demo

Telerik Academy

Cross-browser HTTP Request

- Microsoft first introduced an object to create HTTP requests with JavaScript
  - The XMLHTTP object as an ActiveX object
- To create cross-browser instance of the HTTP object, you should do feature detection

```
if (window.XMLHttpRequest) {
 httpRequest = new XMLHttpRequest();
}
else if (window.ActiveXObject) {
 try{
  httpRequest = new ActiveXObject("Msxml2.XMLHTTP");
 }
 catch (e) {
  httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
 }
}
```

Telerik Academy

Cross-browser HTTP Request

Live Demo

Telerik Academy

HTTP Client – Server Comunication

Telerik Academy

HTTP Client – Server Comunication

- What about the response data?
  - When a client sends an HTTP request to a server, the server always returns a response to the request
  - The response should be handled on the client
- XHR instances have an event that fires when the server sends a response
  - onreadystatechanged
  - Handlers are executed for each the states of this request

Telerik Academy

XMLHttpRequest: Ready States

- Each HTTP request goes through five states during its execution
  - Unitialized – readyState is `0`
  - Loading – readyState is 1
  - Loaded – readyState is 2
  - Interactive – readyState is 3
  - Complete – readyState is 4

Telerik Academy

XMLHttpRequest: Ready States (2)

- `readyState` 4 (complete) means the client has received a HTTP response from the server
  - Yet the response can be success or error
  - Check the `statusCode` property to see the type of the response

```
//httpRequest is created above
httpRequest.onreadystatechange = function () {
 if (httpRequest.readyState === 4) {
   console.log('Response received');
 }
}
//httpRequest is send to the server
```

Telerik Academy

XMLHttpRequest: Ready States

Live Demo

Telerik Academy

HTTP Status Code

- Ready state tell us that the response has arrived
  - But what is the actual response type?
  - Remember HTTP status codes?
    - 2XX means the request is successful (the request done its job and we have the requested data in the response)
    - 4XX and 5XX mean something is wrong

```
if (httpRequest.readyState === 4) {
 statusType = Math.floor(httpRequest.status/100);
 if (statusType === 2) { … }
 else { … }
}
```

Telerik Academy

HTTP Status Code

- Ready state tell us that the response has arrived
  - But what is the actual response type?
  - Remember HTTP status codes?
    - 2XX means the request is successful (the request done its job and we have the requested data in the response)
    - 4XX and 5XX mean something is wrong

```
if (httpRequest.readyState === 4) {
 statusType = Math.floor(httpRequest.status/100);
 if (statusType === 2) { … }
 else { … }
}
```

Handles status codes 200, 201, etc…

Telerik Academy

HTTP Status Codes

Live Demo

Telerik Academy

Consuming the Response

Telerik Academy

Consuming the Response

- The body of the response can be accessed vie the `responseText` property
  - When the readyState is 4
- `responseText` contains a string with the data
  - Based on the value of the Accept HTTP header it is in format HTML, XML, JSON or plain text
  - Must be parsed to the correct data format

Telerik Academy

Consuming the Response

Live Demo

Telerik Academy

Building an HTTP Request

Telerik Academy

Building an HTTP Request

- XMLHttpRequest creates a HTTP request with default characteristics
  - The HTTP method is "GET"
  - Content-Type and Accept have different values based on the client
  - HTTP body is empty (NULL)
  - Etc…
- All of these can be customized

Telerik Academy

Building an HTTP Request (2)

- To set an HTTP header on the request use httpRequest.setRequestHeader(header, value)
  - After the request is open
  - And before the request is sent

```
httpRequest.open('GET', endpointUrl, true);
httpRequest.setRequestHeader('Content-type',
                             'application/json');
httpRequest.setRequestHeader('Accept',
                             'application/json');
httpRequest.send(null);
```

Telerik Academy

Building an HTTP Request (2)

- To set an HTTP header on the request use httpRequest.setRequestHeader(header, value)
  - After the request is open
  - And before the request is sent

Can be any HTTP method

```
httpRequest.open('GET', endpointUrl, true);
httpRequest.setRequestHeader('Content-type',
                             'application/json');
httpRequest.setRequestHeader('Accept',
                             'application/json');
httpRequest.send(null);
```

Telerik Academy

Building an HTTP Request (2)

- To set an HTTP header on the request use httpRequest.setRequestHeader(header, value)
  - After the request is open
  - And before the request is sent

Can be any HTTP method

```
httpRequest.open('GET', endpointUrl, true);
httpRequest.setRequestHeader('Content-type',
                             'application/json');
httpRequest.setRequestHeader('Accept',
                             'application/json');
httpRequest.send(null);
```

Can pass an JSON object or string as body of the request

Telerik Academy

Building an XHR Object

- Still it will be nice is we can make something like

```
httpRequest.make({
 url:"scripts/data.js",
 type: "GET",
 contentType: "application/json",
 success: successCallback,
 error: errorCallback
});
```

Telerik Academy

Building an XHR Object

- Still it will be nice is we can make something like

```
httpRequest.make({
 url:"scripts/data.js",
 type: "GET",
 contentType: "application/json",
 success: successCallback,
 error: errorCallback
});
```

Executes success, if the response is successful

Telerik Academy

Building an XHR Object

- Still it will be nice is we can make something like

```
httpRequest.make({
 url:"scripts/data.js",
 type: "GET",
 contentType: "application/json",
 success: successCallback,
 error: errorCallback
});
```

Executes success, if the response is successful

Executes error, if the response is unsuccessful

Telerik Academy

Building an XHR Object (2)

- Or even better – working with promises

```
httpRequest.make({
 url:"scripts/data.js",
 type: "GET",
 contentType: "application/json",
})
.then(successCallback, errorCallback);
```

Telerik Academy

Building an HTTP Request

Live Demo

Telerik Academy

jQuery AJAX

The easier way of doing AJAX

jQuery AJAX

- jQuery has a functionality for creating HTTP requests
  - A full support only for GET, POST, PUT and DELETE methods
- jQuery AJAX methods:
  - jQuery.ajax(options)
  - jQuery.getJSON(url, success)
  - jQuery.postJSON(url, success)
  - jQuery(selector).load(url);

jQuery.ajax()

- `jQuery.ajax()` is the primary method for creating HTTP requests
  - The options parameter contains all the data about building a complete HTTP request

```
$.ajax({
 url: endpointUrl,
 type: 'GET',
 timeout: 5000,
 success: function(data){ //handle success }
 error: function(err) { //handle error }
});
```

Telerik Academy

jQuery.ajax()

Live Demo

Telerik Academy

jQuery.getJSON() and jQuery.postJSON()

- `jQuery.getJSON()` and `jQuery.post()` are shortcut methods to make an HTTP requests with GET and POST HTTP method
  - Takes as parameters URL of the resource and a success callback
    - An error handler should be set as a promise

```
$.getJSON(endpointUrl,
          successCallback)
  .error(errorCallback);

$.post(endpointUrl,
       data,
       successCallback,
       'json')
  .error(errorCallback);
```

Telerik Academy

jQuery.getJSON() and jQuery.postJSON()

- `jQuery.getJSON()` and `jQuery.post()` are shortcut methods to make an HTTP requests with GET and POST HTTP method
  - Takes as parameters URL of the resource and a success callback
    - An error handler should be set as a promise

```
$.getJSON(endpointUrl,
          successCallback)
  .error(errorCallback);

$.post(endpointUrl,
       data,
       successCallback,
       'json')
  .error(errorCallback);
```

Need to provide data type

Telerik Academy

jQuery.getJSON() and jQuery.post ()

Live Demo

jQuery.load()

- jQuery.load() is the only ajax method that is applied on a DOM element
  - Performs a GET HTTP request
  - Sets the innerHTML of the DOM element to the value of the response

```
$("#http-response").load("partials/details.html");
```

Telerik Academy

jQuery.load()

Live Demo

Consuming Remote Data

Questions?

Telerik Academy

Homework

1. Create a module that exposes methods for performing HTTP requests by given URL and headers
   - getJSON and postJSON
     - Both methods should work with promises
2. Read the developer documentation of Twitter
   - Create a simple application that visualizes all public tweets for a given user (maybe from a textbox)

Homework (2)

3. Using the REST API at 'localhost:3000/students' create a web application for managing students
   - The REST API provides methods as follows:
     - POST creates a new student
     - GET returns all students
     - DELETE deletes a student by Id
   - You may extend the demo for jQuery.ajax()