AJAX and Data Access

Marco Torchiano

Version 2.0.0 - May 2020

License

AJAX

AJAX

Asynchronous Javascript And XML

A technique for creating fast and dynamic web pages.

Allows javascript code to:

- retrieve data from a server (after the page has loaded),
- send data to a server (in background),
- update a web page contents without reloading it.

AJAX steps

- Javascript generate a request to a server
    - an `XMLHttpRequest` object is used to build and send the request
- When the response from the server arrives
    - a (call-back) operation is performed on the data

Note 1: the content has *NOT* necessarily to be XML.

Note 2: the request can contain data and the reply be just a confirmation.

`XMLHttpRequest` Object

Issuing requests:

- Create the object: `new XMLHttpRequest()`
- Prepare the request: `open( method, url, async)`
    - *method* is either `"GET"` or `"POST"`
    - *url* is the url of the resource
    - *async* is a boolean
- Send the request: `send( data )`
    - *data* is optional, used for `POST` request

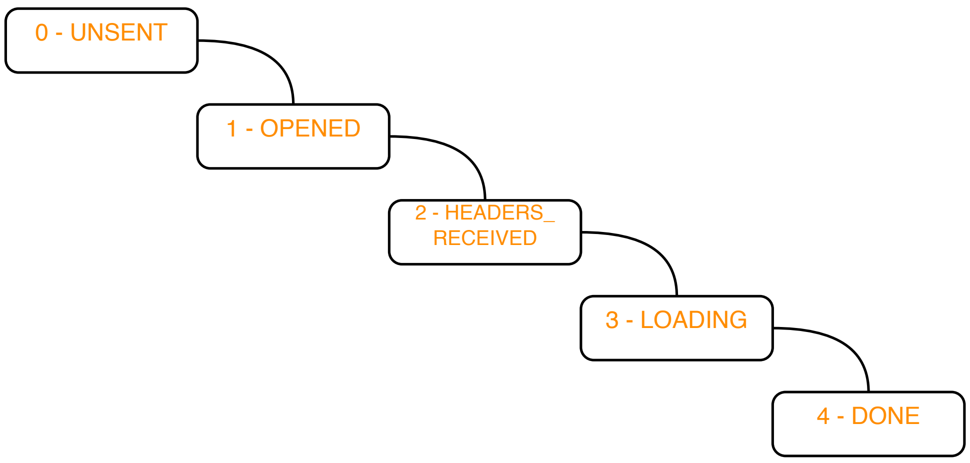
`XMLHttpRequest` Object

Retrieving responses:

- `onreadystatechange` callback function invoked on any state change event
- `readyState` current state of the request
- `status` final HTTP response code (e.g. `200` for OK)
- `responseText` response body (when *done*)
- `responseXml` response as XML document
  - (traversable with a DOM)

State diagram for `XMLHttpRequest`

The current state is available through `readyState`:



The values are defined as constants in `XMLHttpRequest`

Response `status`

- It is the HTTP response status code
  - Defined after `readyState===XMLHttpRequest.DONE`
- Success outcomes of the request are:
  - `200`: resource retrieved
  - `304`: resource not modified
    - response to a conditional request (e.g. get if not changed)

```
okStatus = s => [200,304].indexOf(s) >= 0;
```

Synchronous request

The `send()` method blocks until a response is received.

```
  function loadSync(url,success){
    var xhr = new XMLHttpRequest();
    xhr.open("GET",url,false); // synch
    try{
      xhr.send();
      if(okStatus(xhr.status)) {
        success(xhr.responseText); // call back
      }
    }catch(err){
      console.error("Request failed: " + url);
    }
  }
```

Synchronous request usage

```
var DS= // prefix for data files
"http://softeng.polito.it/courses/VIQ/datasets/";
loadSync(DS+"GoT.txt",function(txt){
            console.log("Got it!:\n" + txt)
        })
console.log("--- Completed! ---")
```

↪
```
Got it!:
Jon Snow
Tyrion Lannister
Daenerys Targaryen
Arya Stark
--- Completed! ---
```

Asynchronous request (AJAX)

```
function loadASync(url,success){
  var xhr = new XMLHttpRequest();
  xhr.open("GET",url,true);
  xhr.onreadystatechange = function(){
    if(this.readyState == XMLHttpRequest.DONE)
      if(okStatus(this.status)){
        success(this.responseText);
      }else{
        console.error("Request failed : " + url);
      }
  };
  xhr.send();
}
```

AJAX sample

```
loadASync(DS+"GoT.txt",function(txt){
  var res = document.createElement("pre");
  res.appendChild(document.createTextNode(txt));
  document.getElementById("ajax-sample").
           appendChild(res)
});
```

```
Jon Snow
Tyrion Lannister
Daenerys Targaryen
Arya Stark
```

Load Dynamic Content

Dynamic content loading requires:

- a data file with the content
  - e.g. `GoT.csv`
- a placeholder where to show content
  - e.g. element inside a page
- a function to generate content
  - e.g. generate a `<table>` from a table object

Generating content (innerHTML)

```
function tableToHtmlElement(data){
  var res = document.createElement("table")
  var html ="<tr>";
  for(h in data[0])
    if(data[0].hasOwnProperty(h)) html+="<th>"+h;
  html+="</tr>"
  for(var i=0; i<data.length; ++i){
    html+="<tr>";
    for(f in data[i]) html+="<td>"+data[i][f];
    html+="</tr>";
  }
  res.innerHTML = html;
  return res;
}
```

Generating content (Full DOM)

```
function tableToHtmlElement1(data){
  let res = document.createElement("table")
  let row = document.createElement("tr");
  for(h in data[0]){
    let c = document.createElement("th");
    c.appendChild(document.createTextNode(h))
    row.appendChild(c); }
  res.appendChild(row);
  for(var i=0; i<data.length; ++i){
    row = document.createElement("tr");
    for(f in data[i]){
      let c = document.createElement("td");
      c.appendChild(document.createTextNode(data[i][f
      row.appendChild(c); }
    res.appendChild(row); }
  return res;
```

CSV load sample

```
loadASync(DS+"GoT.csv",function(data){
  var tab = tableToHtmlElement(csvParse(data));
  document.getElementById("csv-load-sample").
            appendChild(tab)
});
```

ID	Surname	Name
4321	Snow	Jon
5765	Lannister	Tyrion
4663	Targaryen	Daenerys
9896	Stark	Arya

JSON load sample

```
loadASync(DS+"GoT.json",function(txt){
  let data = JSON.parse(txt);
  let tab = tableToHtmlElement(data);
  document.getElementById("json-load-sample").
           appendChild( tab );
});
```

ID	Surname	Name
4321	Snow	Jon
5765	Lannister	Tyrion
4663	Targaryen	Daenerys
9896	Stark	Arya

Same-Origin Policy

Same-Origin Policy

Browsers restrict a page in using resources from the *same origin* only , i.e. same: protocol, host, and port.

Cross-origin resources are typically:

- allowed for *write* operations (e.g. link open)
- allowed for *embedding* operations (e.g. images, scripts)
- NOT allowed for *read* operations, i.e. when a script attempts to read properties of the resource

SOP limitation is adopted to avoid vulnerabilities, e.g. Cross-Site Request Forgery

Relaxing Same-Origin Policy

- Setting `document.domain` to a common domain
  - works for different frames, not for XHR requests
- Using the Cross-Origin Resource Sharing standard
- Using the JSONP standard

Cross-Origin Resource Sharing

CORS allows a foreign CO server to enable sharing

- Client sends a `Origin` header specifying the requester origin
- CO Server returns a `Access-Control-Allow-Origin` header that specifies which origins are allowed
  - e.g. `*` for unlimited sharing
  - In *Apache* a the `.htaccess` file should contain a line like: `Header set Access-Control-Allow-Origin "*"`

JSONP

- Client
  - includes a `<script>` element
  - whose `src` attribute points to the JSON content
  - adds a `?callback=process` at the end of the URL
- Server wraps the JSON content with `process(` and `)`
- Client evaluates the returned code as Javascript, so
  - invokes the `process` function
  - the function receive the JSON as argument

This works because scripts are not restricted by Same-Origin Policy

Load and process JSONP

Inject a `<script>` element and an handler function that are later removed.

```
function loadJSONP(url,process){
  var s = document.createElement("script");
  var cbname = "_cbf"+Math.round(Math.random()*1e9);
  window[cbname]=function(obj){
    process(obj);
    document.body.removeChild(s);
    delete window[cbname] ;
  };
  s.src = url + "?callback=" + cbname;
  document.body.appendChild(s);
}
```

JSONP load sample

```
loadJSONP(DS+"GoT.php",function(data){
  var tab = tableToHtmlElement(data)
  document.getElementById("jsonp-load-sample").
           appendChild( tab );
});
```

ID	Surname	Name
4321	Snow	Jon
5765	Lannister	Tyrion
4663	Targaryen	Daenerys
9896	Stark	Arya

JSON server-side script (PHP)

```
<?php header('content-type: application/json;' .
             'charset=utf-8');

$myfile = fopen("GoT.json", "r") or
                die("Unable to open file!");
$data =  fread($myfile,filesize("GoT.json"));

echo $_GET['callback'] . '('. $data . ')';
```

References

- W3C. XMLHttpRequest Level 1. http://www.w3.org/TR/XMLHttpRequest/
- W3Schools. AJAX Tutorial. http://www.w3schools.com/ajax/default.asp
- Jesse Ruderman (MDN). Same-origin Policy. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- MDN, HTTP access control (CORS). https://developer.mozilla.org/en-US/docs/HTTP/Access_control_CORS
- Wikipedia. JSONP. https://en.wikipedia.org/wiki/JSONP
- B.Ippolito. Remote JSON - JSONP. https://bob.ippoli.to/archives/2005/12/05/remote-json-jsonp/
- OWASP. Cross-Site Request Forgery. https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29