Licence CRRW – IUT de Marne-la-Vallée
2019-02-14

AJAX with jQuery

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Sources used

- Lectures by Jean-Loup Guillaume
http://jlguillaume.free.fr/www/documents/teaching/ntw1213/LI385_C5_Jquery.pdf

- Advanced web programming lectures by Thierry Hamon
https://perso.limsi.fr/hamon/PWA-20122013/Cours/JQuery.pdf

- *jQuery : écrivez moins pour faire plus !*, by tit_toinou
http://openclassrooms.com/courses/jquery-ecrivez-moins-pour-faire-plus

- *jQuery, Le guide complet*, by Guillaume Allain and Timothy Stubbs

- *Javascript & Ajax pour les nuls*, by Andy Harris

- jQuery documentation : http://api.jquery.com/

Introduction à AJAX

= software architecture to update a webpage on the client side (in the browser) using information from the server side without reloading the page

asynchronous = we are not waiting for the result of previous queries to launch the next ones and the next Javascript instructions.

Introduction à AJAX

= software architecture to update a webpage on the client side (in the browser) using information from the server side without reloading the page

asynchronous = we are not waiting for the result of previous queries to launch the next ones and the next Javascript instructions.

Advantages of AJAX:

- deal with information streams in real time
- allow collaborative web tools
- optimize the loading time of a webpage and the bandwidth

Loading content

To insert some content into an object:

• Simpler than with AJAX functions

• Possible to load only part of the file (even if all the file is retrieved, then parsed to extract only the required part)

```
// version without parameters:
// retrieves file.html, puts its content into a div

$("div").load("file.html");

// version with parameters: calls file.php
// providing name=philippe (POST)

$("div#content").load("file.php",
{"name":"philippe"});

// version retrieving only the object with id #myId

$("div").load('file.php #myId');
```

http://api.jquery.com/load/

With GET and POST

```
// retrieve the file file.php
// then execute a function

// GET: parameters in the URL:

$.get(
  "http://website.com/file.php",
  {"name":"philippe"},
  function(data){
    console.log(data);
});

// POST : parameters in the message header
// (useful for special characters, big size, etc.)

$.post(
  "http://website.com/file.php",
  {"name":"philippe"},
  function(data){
    console.log(data);
});
```

Writing objects in Javascript: the JSON format

JSON = JavaScript Object Notation

An object has several **properties**. Each property is a string.
Each property has a *value*, which may be a string, a number, a boolean (*true* or *false*), the empty value (*null*), an object or a table.
An object is written between **braces**.

```
var myObject = {"lastName":"Gambette",
"firstName":"Philippe", "birth":1984, "height":1.81};
```

Same as the objects in the .css(...) function of jQuery!

```
$("#identifier").css({"color":"red","margin":"10px"})
```

To get or set the value of one property, use the **dot**:

```
MyObject.firstName="Phil";
console.log("First name: "+myObject.firstName);
```

Or use **square brackets**:

```
MyObject["firstName"]="Phil";
console.log("First name: "+myObject["firstName"]);
```

Writing objects in Javascript: the JSON format

A table contains several *values*, which may be strings, numbers, booleans, empty values, objects or tables.
A table is written between **square brackets**.

```
var names = ["Berthet", "Gambette", "Jottreau"];
```

To get or set the value of one property, use the **square brackets**, starting at number 0 for the first cell of the table.

```
names[0]="Philippe Gambette";
console.log("PHP+jQuery: "+names[0]+", "+names[1]);
```

A more complex example

```
var data = {"film":[{"attributs":
{"id":"Q21855700"},"titre":"Demain","realisation": {"personne":
[{"attributs":{"id":"Q21654137"},"prenom":"Cyril","nom":"Dion"},
{"attributs":{"id":"Q234581"},"prenom":"Mélanie","nom":"Laurent"}]}},
{"attributs":{"id":"Q25395940"},"titre":"Frantz","realisation":
{"personne": [{"attributs":
{"id":"Q266535"},"prenom":"François","nom":"Ozon"}]}}]};

data.film[0].realisation.personne[0].prenom + " " +
data.film[0].realisation.personne[0].nom;
//displays "Cyril Dion"

data["fil"+"m"][0]["realisation"]["personne"][1].prenom + " " +
data.film[0]["realisation"]["personne"][1]["nom"];
//affiche "Mélanie Laurent"
```

How to use Javascript objects retrieved by the AJAX query

If the page `http://website.com/file.php?id=9` contains the following JSON code:

```
{"lastName":"Gambette", "firstName":"Philippe",
"birth":1984, "height":1.81}
```

You can get these values in your jQuery code:

```
$.get(
  "http://website.com/file.php?id=9", function(data){
    console.log("Last name: "+data["lastName"]+
    ", first name:"+data.firstName);
});
```

AJAX functions

```
// Function to call when a request completes
.ajaxComplete()

// Function to call to handle errors in the end
.ajaxError()

// Fonction to call before any request is sent
.ajaxStart()

// Function to call before one request is sent
.ajaxSend()

// Function to call when every request is finished
// Ajax sont terminées
.ajaxStop()

// Function to call when a request finishes
// successfully
.ajaxSuccess()
```

Constraints of asynchronous execution

AJAX events are not related with one special request:

→ it is necessary to remember the requests to find out where a given request came from

```
$('.log').ajaxComplete(function(e, xhr, settings) {
  if (settings.url == 'ajax/test.html') {
    $(this).text('ok.');
  }
});
```

Testing AJAX functions

Jquery Code :

```
$('.log').ajaxStart(function(){$(this).append('Début.');});

$('.log').ajaxSend(function(){$(this).append('Envoi.');});

$('.log').ajaxComplete(function(){$(this).append('Fini.');});

$('.log').ajaxStop(function(){$(this).append('Stop.');});

$('.log').ajaxSuccess(function(){$(this).append('Succès.');});

$('.trigger').click(function(){
  $('.result').load('fichier.json');
});
```

HTML code:

```
<div class="trigger">Trigger</div>

<div class="result"></div>

<div class="log"></div>
```

Testing AJAX functions: alternatives

`[request].done(function(data, textStatus, jqXHR){ })`:
function executed when the AJAX request is successful

`[request].fail(function(jqXHR, textStatus, errorThrown){ })`:
function executed when the AJAX request is not successful

`[request].always(function(jqXHR, textStatus){ })`:
function executed in any case after `done` or `fail`.

`jqXHR`: Javascript object used for the AJAX request

`textStatus`: string describing the staus of the request (success, timeout, error, notmodified, parsererror)

`errorThrown` : the error which was sent if there is any

`data` : le data sent by the server

Retrieving JSON files or Javascript code

```
// Loading and extracting information from a
// JSON file

$.getJSON(
  "file.json",
  {id:1},
  function(users) {
    alert(users[0].name);
});

// Loading and executing a Javascript code

$.getScript(
  "script.js",
  function() {
    ...;
});
```

The generic AJAX function

Functions `get`, `post`, `getJavascript`, `getJSON`, etc. are specific cases of the `ajax` function:

```
$.ajax({
  async: false,
  type: "POST",
  url: "test.html",
  data: "nom=JL",
  success: function(msg){
    alert( "Data Saved: " + msg );}
});
```

Generic AJAX function

```
$.ajax({
  async: false,
  type: "POST",
  url: "test.html",
  data: "nom=JL",
  success: function(msg){
    alert( "Data Saved: " + msg );}
});
```

`success` equivalent to `done`

`error` equivalent to `fail`

`complete` equivalent to `always`